Deflated CG Method for Modelling Groundwater Flow Near Faults

Interim Report

L.A. Ros

Delft, Utrecht

January 2008

Delft University of Technology | Deltares

Supervisors: Prof.dr.ir. C. Vuik (TU Delft)
Dr. M. Genseberger (Deltares)
Ir. J. Verkaik (Deltares)

Preface

This report is written as a result of the literature study of my Masters Thesis project for the study of Applied Mathematics at the Delft University of Technology. The Masters project is being carried out at Deltares.

Deltares is a new Dutch institute for Delta Technology, starting on 1 January 2008. Deltares is formed from parts of Rijkswaterstaat /DWW, RIKZ and RIZA, WL | Delft Hydraulics, GeoDelft, and a part of TNO Built Environment and Geosciences. Together, they work on problems in the field of water, soil and the subsurface. Deltares works for and cooperates with Dutch government, provinces and water boards, international governments, knowledge institutes and market parties. The institute is located in two cities: Delft and Utrecht.

Since September 2007 I am working on a groundwatermodel for Limburg, IBRAHYM, developed by the former TNO. My goal is to improve the convergence behavior of the solver near faults. This can probably be done by using a deflation based preconditioner.

Lennart Ros,

Delft, Utrecht, December 2007.

Contents

Chapter 1

Introduction

In densely populated areas, land use and planning are closely related to demands on water management of, for example, natural, agricultural and recreational areas. It is therefore important to base management of both groundwater and surfacewater systems on these demands.

Modelling groundwater flow can be done by the finite-volume code named MODFLOW [5], designed by the USGS [15], that is commonly used in the world. The equations used for solving groundwater flow are based on Darcy's Law. This code solves the equation with the conjugate gradient solver using an incomplete Cholesky preconditioning. However, this preconditioner seems to fail for applications with large contrasts in medium parameters, resulting in poor convergence behavior. This happens, for example, near faults such as the Peelrandbreuk in Limburg. A possible technique to improve convergence is to use a deflation-based preconditioner.

In this chapter a short overview is given about the geology and hydrology basics for modelling groundwater flow. The last paragraph gives a short overview for the rest of the report.

1.1 Subsurface, Groundwater and Faults

1.1.1 Geology

The subsurface consist of different layers of alternating sand and clay. Directly under the surface we have an unconfined aquifer. An aquifer is a sediment that is sufficiently porous and permeable to store and transmit groundwater. In an aquifer water can move in all directions, but usually the flow is horizontal. After an aquifer we find an aquitard. An aquitard is a sediment that is not permeable enough to let water flow easily, but it can supply water to the underlying or overlaying aquifer. In an aquitard we assume water to flow vertically only. An example of an aquitard is a package of clay. After some of those aquifers and aquitards we find the aquiclude or the so called geo-hydrological base. This is an impermeable body of rock that may absorb water slowly, but does not transmit it. A simple overview is given in Figure 1.1.

1.1.2 Hydrology

Hydrology describes the movement of water in all it forms. The cycle is shown in Figure 1.2. It can be seen that a lot of processes deal with groundwater.

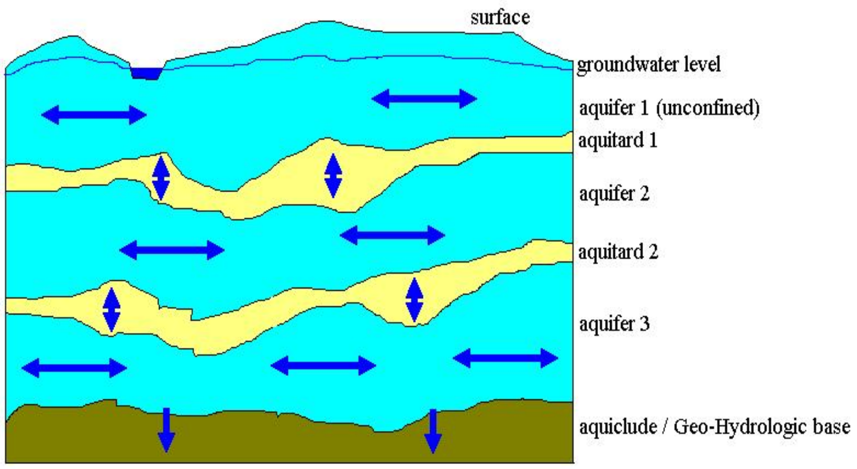

Figure 1.1: Schematization of the subsurface and the possible directions of the groundwater flow.

1.1.3 Geohydrology

Geohydrology describes the groundwater flow. Almost everywhere in the Netherlands groundwater can be found upto a couple of meters beneath the surface. In the lower parts of the Netherlands it can even be found within a meter. Groundwater moves through the pores of the aquifers. Not every type of ground has the same porosity. Porosity is defined as the volume of the pores divided by the volume of the material. See also Figure 1.3. Conductivity K is a measure for the possibility of movement of a liquid through the pores of a sediment.

To separate the effect of the medium and the liquid we define the intrinsic permeability, k, as:

$$k = \frac{K\mu}{\gamma},$$

where:

μ	dynamic viscosity $[ML^{-1}T^{-1}]$,
γ	specific weight $(\rho\ g)$ $[ML^{-2}T^{-2}]$,
ρ	specific density $[ML^{-3}]$
g	force of attraction $[LT^{-2}]$

The driving force for groundwater flow are the differences in height and pressure. To represent these differences we introduce the concept of hydraulic heads, h $[L]$. This groundwater potential equals the sum of the kinetic energy, the pressure and the potential energy heads:

$$h = \frac{v^2}{2g} + \frac{p}{\rho g} + z, \tag{1.1}$$

where ρg is the specific weight of the liquid. The term h is sometimes called the *groundwater head* or the *piezometric head*. The first term, the kinetic energy, is in practice negligible. So Equation (1.1) reduces to:

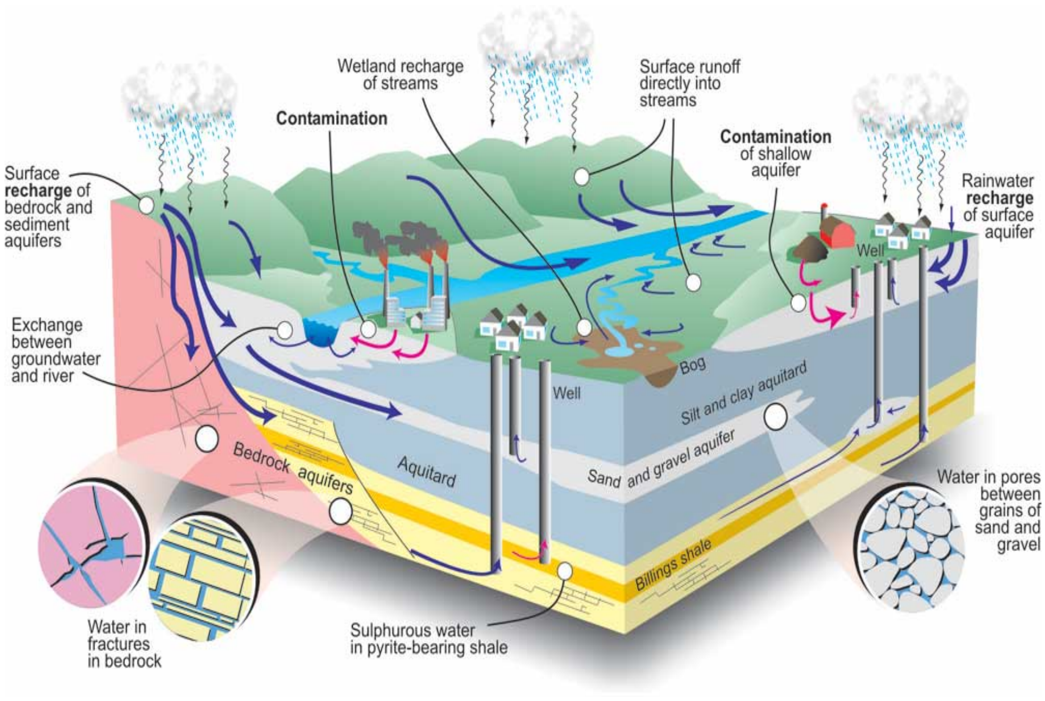
Surface recharge of bedrock and sediment aquifers
Wetland recharge of streams
Contamination
Surface runoff directly into streams
Contamination of shallow aquifer
Rainwater recharge of surface aquifer
Exchange between groundwater and river
Well
Bog
Silt and clay aquitard
Sand and gravel aquifer
Aquitard
Bedrock aquifers
Billings shale
Water in pores between grains of sand and gravel
Water in fractures in bedrock
Sulphurous water in pyrite-bearing shale

Figure 1.2: Overview of the hydrology cycle.

$$h = \frac{p}{\rho g} + z.$$

We can use a piezometer to determine the hydraulic head (see Figure 1.8).

1.1.4 Faults

A fault can have big consequences for the groundwater flow. A fault is a planar rock fracture, which shows evidence of relative movement. The largest examples are at tectonic plate boundaries but many faults occur far from active plate boundaries. Faults do not usually consist of a single, clean fracture, so sometimes the term *fault zone* is used. There are different types of faults, as shown in Figure (1.5). The main property about faults is that they have a low permeability and we have to deal with large contrasts in medium parameters.

1.2 MODFLOW

MODFLOW is a software package which calculates hydraulic heads, developed by the U.S. Geological Survey. Since it is an open-source code everyone can use and improve this program. A complete manual can be found on the internet, [5], but we will discuss only some important properties.

MODFLOW works with a rectangular grid and uses a cell-centered variables. In Section 3.1.1 is described how we set up this grid. The cells we use have three dimensions and they are numbered using an (i, j, k) grid. The i stands for the number of the column, the j is for the number of the row and k is the number of the layer. Since

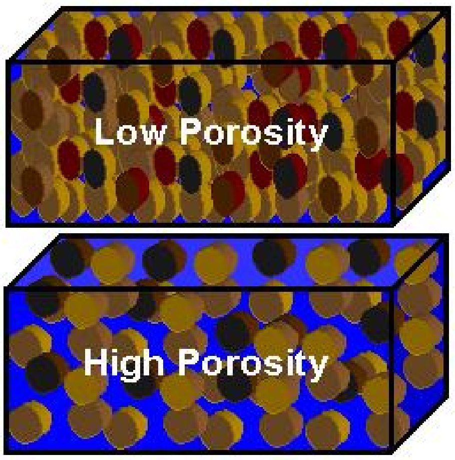

Figure 1.3: Low and high porosity.

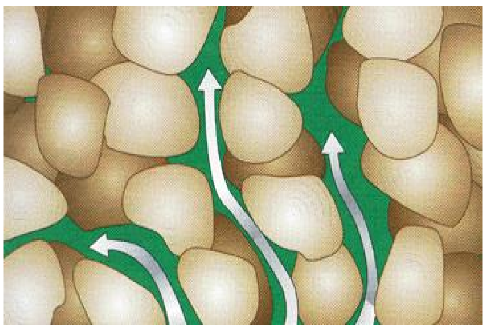

Figure 1.4: Connected pores give a rock permeability.

in real applications the groundstructure is not rectangular, you have to specify in every cell the transmissivity. This is some measure for permeability, corrected for the real dimension represented by the cell. This value is also called kD-value, since you multiply the conductivity with the thickness (Dutch: dikte) of the layer. The conductance for a cell is now calculated as the transmissivity (TR) times the width of the cell divided by its length. The volumetric flow, Q, now is defined by

$$Q = C(h_1 - h_2). \tag{1.2}$$

Conductance is defined for a particular prism of material and for a particular direction of flow. If a prism of porous material consists of two or more subprisms in serie and for all the subprisms the conductance is known, then the conductance representing the entire prism can be calculated. We know that we have:

$$C = \frac{Q}{h_A - h_b}.$$

If we assume that the heads across each subprism is continuous, we get the identity:

$$\sum_{i=1}^{n} \Delta h_i = h_A - h_b.$$

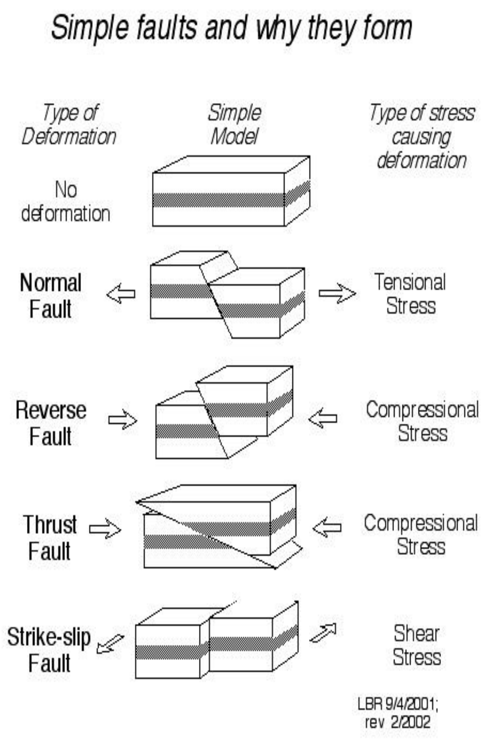

Figure 1.5: Overview different types of faults [16].

Substituting Equation (1.2) gives:

$$\sum_{i=1}^{n} \frac{q_i}{c_i} = h_A - h_B.$$

where:

q_i	is the flow across subprism i, and
c_i	is the conductance of subprism i.

Since we consider flow in one direction and we assume no accumulation or depletion in storage, each q_i is equal to the total flow Q and therefore:

$$Q \sum_{i=1}^{n} \frac{1}{c_i} = h_A - h_B \text{ and } \frac{h_A - h_B}{Q} = \sum_{i=1}^{n} \frac{1}{c_i}. \tag{1.3}$$

Comparing Equation (1.3) and Equation (1.2), one can conclude that:

$$\frac{1}{C} = \sum_{i=1}^{n} \frac{1}{c_i},$$

which is called the harmonic mean. So if we only have two subprisms the equivalent conductance reduces to:

$$C = \frac{c_1 c_2}{c_1 + c_2}. \tag{1.4}$$

If we look into the row-direction (see also Figure 3.1 in Section 3.1.1), we get for the conductance between cell (i, j, k) and cell $(i, j+1, k)$ (see also Figure 1.6):

$$CR_{i,j+1/2,k} = \frac{\frac{TR_{i,j,k}DELC_i}{(1/2)DELR_j} \times \frac{TR_{i,j+1,k}DELC_i}{(1/2)DELR_{j+1}}}{\frac{TR_{i,j,k}DELC_i}{(1/2)DELR_j} + \frac{TR_{i,j+1,k}DELC_i}{(1/2)DELR_{j+1}}}, \tag{1.5}$$

where the R stands for row-direction. Simplification of Equation (1.5) gives the final equation:

$$CR_{i,j+1/2,k} = 2DELC_i \frac{TR_{i,j,k}TR_{i,j+1,k}}{TR_{i,j,k}DELR_{j+1} + TR_{i,j+1,k}DELR_j}. \tag{1.6}$$

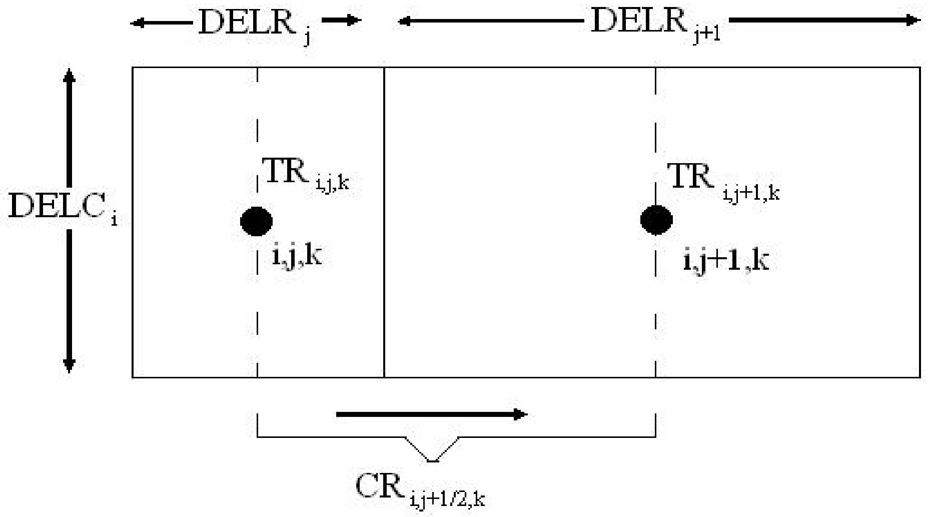

Figure 1.6: Calculation of conductance between two cells using transmissivities.

The same process can be applied to the calculation of $CC_{i+1/2,j,k}$ (C for column) giving:

$$CC_{i+1/2,j,k} = 2DELR_j \frac{TC_{i,j,k}TC_{i+1,j,k}}{TC_{i,j,k}DELC_{i+1} + TC_{i+1,j,k}DELR_i}. \tag{1.7}$$

Vertical conductance can also be calculated in the same way as the horizontal conductances. We only give the result:

$$CV_{i,j,k+1/2} = \frac{DELR_j DELC_i}{\frac{(1/2)DELV_k}{VK_{i,j,k}} + \frac{(1/2)DELV_{k+1}}{VK_{i,j,k+1}}}. \tag{1.8}$$

where VK is the vertical (hydraulic) conductivity between layers.

MODFLOW is a quasi-3D model since it models the aquitards as a vertical flux between layers. So it does not calculate anything inside an aquitard. Because of this we have to adapt Equation (1.8). An exact derivation can be found in [5]. We only give the result:

$$CV_{i,j,k+1/2} = \frac{DELR_j DELC_i}{\frac{(1/2)DELV_k}{VK_{i,j,k}} + \frac{\Delta v_{CB}}{VKCB_{i,j,k}} + \frac{(1/2)DELV_{k+1}}{VK_{i,j,k+1}}}, \tag{1.9}$$

where:

$VKCB_{i,j,k}$	is the hydraulic conductivity of the aquitard between (i,j,k) and (i,j,k+1), and
Δv_{CB}	is the thickness of the aquitard.

The term $\frac{\Delta v_{CB}}{VKCB_{i,j,k}}$ compensates for the conductance in the aquitard between layer k and $k+1$.

MODFLOW uses an iterative method to solve a system of equations obtained from the finite-element discretization as carried out in Chapter 3. The method which is used in MODFLOW is a preconditioned conjugate gradient method to improve convergence. As a preconditioner, the incomplete Cholesky Factorization is used.

1.3 IBRAHYM

IBRAHYM is a groundwater model that is developed for several waterboards in Limburg. Limburg is a province in the Netherlands that has a large variety of faults in the subsoil. Such faults cause the model to suffer from bad convergence behavior of the solver. IBRAHYM uses at most 19 layers to model the groundwater flow in that area. Furthermore it uses grid cells of 25 times 25 meter to get detailed information.

The most famous fault in Limburg is "de Peelrandbreuk".

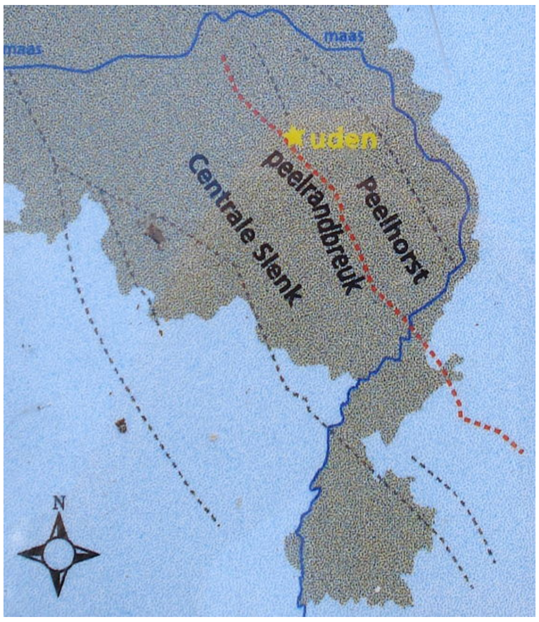

Figure 1.7: The Peelrandbreuk in Limburg [17].

1.4 Problem Description

Deltares uses MODFLOW to simulate groundwater flow in the Netherlands. In Limburg the process converges very slowly by several impermeable faults. In this report the model is decribed in more detail and possible strategies are given to solve the problem in the next months.

In Chapter 2 the governing equation for porous media flow is derived. We start by explaining Darcy's Law (Section 2.1) and derive the steady-state flow equation (Section 2.2). After that we derive the equation for time variant problems (Section 2.3).

Chapter 3 starts with the derivation for the finite volume equation. In Secion 3.2 an overview is given of how MODFLOW deals with boundary conditions. The last section is about how MODFLOW deals with faults in the subsoil.

In Chapter 4 we look at iterative solvers for linear systems. Section 4.1 describes the basic iterative methods. Section 4.2 deals with projection methods, where Section 4.3 introduces the Krylov subspace. After that the Preconditioned Conjugate Gradient Method is introduced in Section 4.4. The last section, Section 4.5, is about convergence, starting vectors and stopping criteria.

In Chapter 5 the deflation technique is explained. Section 5.1 explains the basics of deflation. Section 5.2 uses the eigenvalues of the coefficientmatrix to deflate and Section 5.3 gives alternative techniques.

The last chapter, Chapter 6, gives an overview of the first testcases and the first conclusions. In Section 6.4 the research questions are stated for the upcoming six months of my Master's Thesis.

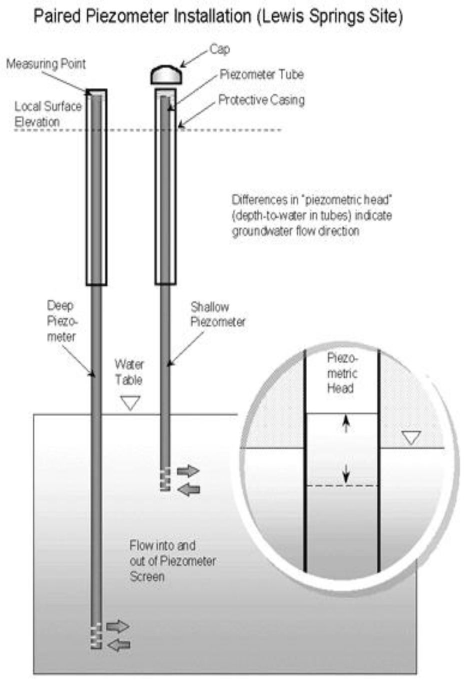
Paired Piezometer Installation (Lewis Springs Site)
Cap
Measuring Point
Piezometer Tube
Protective Casing
Local Surface Elevation
Differences in "piezometric head" (depth-to-water in tubes) indicate groundwater flow direction
Deep Piezo-meter
Shallow Piezometer
Water Table
Piezo-metric Head
Flow into and out of Piezometer Screen

Figure 1.8: A piezometer [18].

Chapter 2

The Governing Equation for Porous Media Flow

Groundwater flow is three dimensional in space and it is also a function of time. The velocity of the flow is a function of x, y, z and t.

The flow can be evaluated quantitively when the velocity, pressure, density, temperature, and viscosity of the water flowing through the ground are known. If these variables are only functions of space, then the flow is defined as being *steady*. If the variables are also functions of time, then the flow is defined as being unsteady or time-dependent.

In this section we will derive the fundamental equation based on Darcy's Law. The derivation can also be found in chapter 2 of [9].

2.1 Darcy's Law

The flow through the porous media is a slow frictional flow governed by Darcy's law which we can write as:

$$v = K \cdot i. \tag{2.1}$$

Here:

v	=	Darcy's velocity $[LT^{-1}]$,
K	=	matrix with hydraulic conductivities of medium $[LT^{-1}]$,
i	=	hydraulic gradient.

The tensor K is given by:

$$K = \begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{bmatrix}. \tag{2.2}$$

Each element of K is defined as: $K = \frac{k\gamma}{\mu}$, where:

k	=	intrinsic permeability $[L^2]$,
γ	=	specific weight (ρg) $[ML^{-2}T^{-2}]$,
μ	=	dynamic viscosity $[ML^{-1}T^{-1}]$.

From a macroscopic point of view it can be shown that K is symmetric and positive definite. Symmetric means that we have $k_{xy} = k_{yx}, k_{zx} = k_{xz}, k_{yz} = k_{zy}$ and positive definite means we have $k_{xx}, k_{yy}, k_{zz} > 0$. This assures that energy is always dissipated during flow. MODFLOW assumes a quasi 3D-flow, i.e. the head gradient does not vary vertically within aquifers and does not vary horizontally within aquitards. These kind of flows are also know as *Dupuit-Forchheimer flows*. As a consequence of this, the components $k_{xz} = k_{zx} = k_{yz} = k_{zy} = 0$ and so will the corresponding K's [13]. In MODFLOW we also assume that $k_{xy} = k_{yx} = 0$, so we only model anisotropy along the main axes. If the material is isotropic we have $K_{xx} = K_{yy} = K_{zz}$, but if the material is anisotropic these values can differ.

The hydraulic gradient i results from a difference in groundpotential across an element of the medium. If Δh represents the total potential loss of the fluid over a distance Δs, then in the limit we have:

$$i = -\frac{dh}{ds}. \tag{2.3}$$

If we now substitute Equations (2.2) and (2.3) into Equation (2.1), then we end up with the following equations:

$$\begin{aligned} v_x &= -K_{xx}\frac{\partial h}{\partial x}, \\ v_y &= -K_{yy}\frac{\partial h}{\partial y}, \\ v_z &= -K_{zz}\frac{\partial h}{\partial z}, \end{aligned} \tag{2.4}$$

where K_{xx}, K_{yy}, K_{zz} are the values of hydraulic conductivity along x, y and z coordinate axes. We assume axes to be parallel to the major axes of hydraulic conductivity. The lowercase x, y and z at the v's mean that it is the velocity in that direction.

Note that these equations only hold when the x, y, and z axes coincide with principal permeability axes. Darcy's Law is not sufficient to solve the problem of groundwater flow. We now have only three equations, but we have four unknowns. These unknowns are the three components of the velocity and the groundwater potential. So we need a fourth equation, which we obtain by realizing that the flow has to satisfy the principle of the conservation of mass. No matter what kind of flow we are looking at, we cannot gain or loose mass. This will lead to a continuity equation.

2.2 Steady-state Flow

Consider a small elementary volume. Assume that the fluid filters in such a manner that the porous ground is incompressible. We can use the conservation of mass law. So the sum of mass flow entering the three faces is equal to the sum of the mass flow leaving by the opposite faces. Since we are working with water, with a constant density, the conservation principle can be fullfilled by ensuring that the volume entering equals the volume that leaves the volume. We also assume the water to be incompressible.

Now let P be the center point of such an element and let for the moment $x = y = z = 0$. At this point, the velocities are given by v_x, v_y and v_z. So on $x = -\frac{dx}{2}$, the flow in

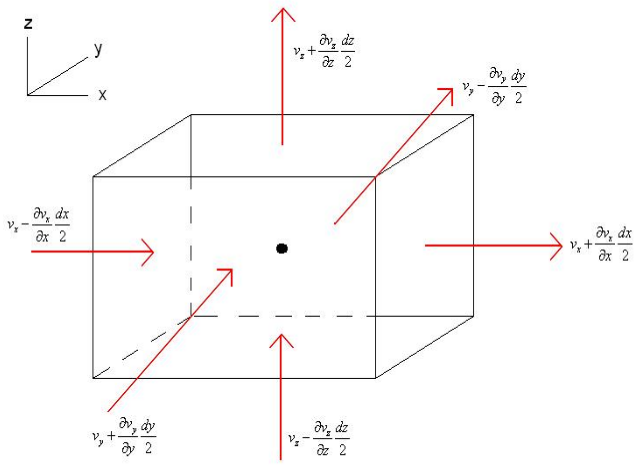
z
y
x
$v_z + \frac{\partial v_z}{\partial z}\frac{dz}{2}$
$v_y - \frac{\partial v_y}{\partial y}\frac{dy}{2}$
$v_x - \frac{\partial v_x}{\partial x}\frac{dx}{2}$
$v_x + \frac{\partial v_x}{\partial x}\frac{dx}{2}$
$v_y + \frac{\partial v_y}{\partial y}\frac{dy}{2}$
$v_z - \frac{\partial v_z}{\partial z}\frac{dz}{2}$

Figure 2.1: Flow for an elementary volume of fluid.

the x-direction will be $v_x\,(x - dx/2) \approx v_x - \frac{\partial v_x}{\partial x}\frac{dx}{2}$. We use here a first order Taylor approximation. If we multiply this by the area, which is $dy\,dz$, we end up with the volume entering the plane through this area. So in the x-direction the following volume will enter the element:

$$\left(v_x - \frac{\partial v_x}{\partial x}\frac{dx}{2}\right) dy\,dz.$$

For the other area in the x-direction fluid will leave the element. The volume leaving the element will be:

$$\left(v_x + \frac{\partial v_x}{\partial x}\frac{dx}{2}\right) dy\,dz.$$

So the net flow, which is outflow minus inflow, in the x-direction will be:

$$\left(\frac{\partial v_x}{\partial x}\right) dx\,dy\,dz.$$

Similarly we have a net flow in the y-direction of $\left(\frac{\partial v_y}{\partial y}\right) dx\,dy\,dz$ and in the z-direction of $\left(\frac{\partial v_z}{\partial z}\right) dx\,dy\,dz$. So the volume of the flow through the elementary volume will then be:

$$\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}\right) dx\,dy\,dz.$$

According to the conservation principle this must be equal to 0, so we must have:

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} = 0. \tag{2.5}$$

We call this the *steady-state continuity equation*.

If we now substitute equation (2.4) into (2.5) we end up with:

$$\frac{\partial}{\partial x}\left(K_{xx}\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_{yy}\frac{\partial h}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_{zz}\frac{\partial h}{\partial z}\right)=0, \tag{2.6}$$

which describes basic steady-state flow without source and sink terms.

2.2.1 Boundary Conditions

Boundary conditions are needed to derive a unique solution for a partial differential equation. Various types of boundary conditions are possible in a steady state flow:

1. Known values of the groundwater potential are given on the boundary (Dirichlet condition).
2. If the magnitude of the flow crossing through the boundary is known we can set: $\frac{\partial h}{\partial n}=\phi$, where ϕ is the velocity normal to the boundary divided by the permeability normal to the boundary (inhomogenous Neumann condition).
3. At an impermeable boundary we do not have a flow through this boundary. So we set: $\frac{\partial h}{\partial n}=0$, where n is the direction normal to the boundary (homogenous Neumann condition).)
4. Head-dependent boundary conditions. These are described in more detail in Section 3.2.1 (Cauchy conditions).

2.3 Time-variant Problems

Most groundwater problems are also time dependent. If we look also at the dependence of time, we must determine if we have a confined or an unconfined aquifer. We call groundwater flow confined when all the boundaries or bounding surfaces of the space through which the flow goes are fixed in space for different states of the flow. See the left picture of Figure 2.2. If the groundwater potential goes a little bit up or down, our aquifer is still completely filled with water. Groundwater flow is unconfined when it possesses a free surface, so if the position of the boundary varies with the state of the flow. See also the right picture in Figure 2.2. Since the groundwater potential lies within our aquifer we are facing a moving boundary here.

For this project we restric ourself to a confined aquifer. A drop in the groundwater potential of Δh results in a reduction in pressure. The volume of water released *per unit volume* of aquifer due to a unit decrease in head will be named the *specific storage coefficient* S_s. If we look at the mass balance for a saturated element in a confined aquifer, we see that both the compressibility of the water and the change in pore volume due to the vertical compression of the aquifer contribute to the specific storage. So we conclude that the specific storage is a function of the density of water, the porosity, the pore volume compressibility and the compressibility of water. Although we can set up an expression for the specific storage, usually it is determined from field measurements. In general S_s is within the range of 10^{-5} to 10^{-7}. The dimension of the specific storage is $[L^{-1}]$.

We can extend the steady-state continuity equation (2.5) from section 2.2 to obtain a differential equation including the effect of the specific storage. The effects of the compressibility and density are included in the specific storage, so we are allowed to work in terms of the conservation of volume.

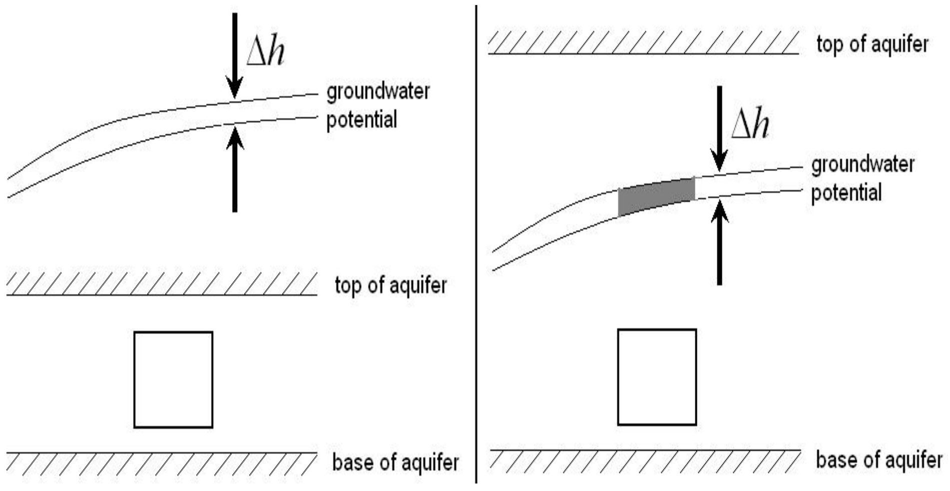

Figure 2.2: A confined (left) and an unconfined (right) aquifer.

If we do the same derivation as in section 2.2 we see that the net volume leaving the element during a time δt due to changing velocities equals:

$$dx\,dy\,dz\left(\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}+\frac{\partial v_z}{\partial z}\right)\delta t.$$

During the time δt, the groundwater potential at point P increases by δh. So the volume of water taken into storage in groundwater potential due to this increase is

$$dx\,dy\,dz\,S_s\,\delta h.$$

Since we have the continuity principle these two equations must sum to zero, so we end up with:

$$\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}+\frac{\partial v_z}{\partial z}=-S_s\frac{\partial h}{\partial t}.$$

If we now substitute equation (2.4) again, we end up with the following differential equation, which holds for any element within the saturated aquifer:

$$\frac{\partial}{\partial x}\left(K_{xx}\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_{yy}\frac{\partial h}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_{zz}\frac{\partial h}{\partial z}\right)=S_s\frac{\partial h}{\partial t}. \tag{2.7}$$

In the next chapter we are going to discretize this equation by using a finite volume method.

Chapter 3

Finite Volume Method in MODFLOW

3.1 Finite Volume Equation for our Governing Equations in MODFLOW

In chapter 2 we derived that the three-dimensional movement of groundwater of constant density through porous media can be described by partial-differential equation (2.7). Since we want to study effects of sources and sinks of water, we add the term W to the equation. We then have the following equation:

$$\frac{\partial}{\partial x}\left(K_{xx}\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_{yy}\frac{\partial h}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_{zz}\frac{\partial h}{\partial z}\right)+W=S_s\frac{\partial h}{\partial t}, \tag{3.1}$$

where:

K_{xx}, K_{yy}, K_{zz}	the values of hydraulic conductivities along x,y, and z coordinate axes (we assume them to be parallel to the major axes of hydraulic conductivity) $[LT^{-1}]$,
h	potentiometric head $[L]$,
W	volumetric flux per unit volume representing sources and sinks of water $[T^{-1}]$,
S_s	specific storage of porous material $[L^{-1}]$,
t	time $[T]$.

For W we define a flow out of the groundwater system as $W < 0$ and a flow into the system as $W > 0$.

In general S_s, K_{xx}, K_{yy}, and K_{zz} are functions of space where W may be a function of space and time. If we specify the flow and/or head conditions at the boundaries of an aquifer system and we specify the initial-head conditions, we have a mathematical representations of a groundwater flow system. Except for very simple systems it is rarely possible to determine analytic solutions for such a problem. So we must use numerical methods to obtain an approximate solution. For this purpose we use the finite volume method.

We want to simplify the mathematical treatment and explain the computational procedure in terms of familiar physical concepts regarding the flow system.

3.1.1 Discretization Convention in MODFLOW

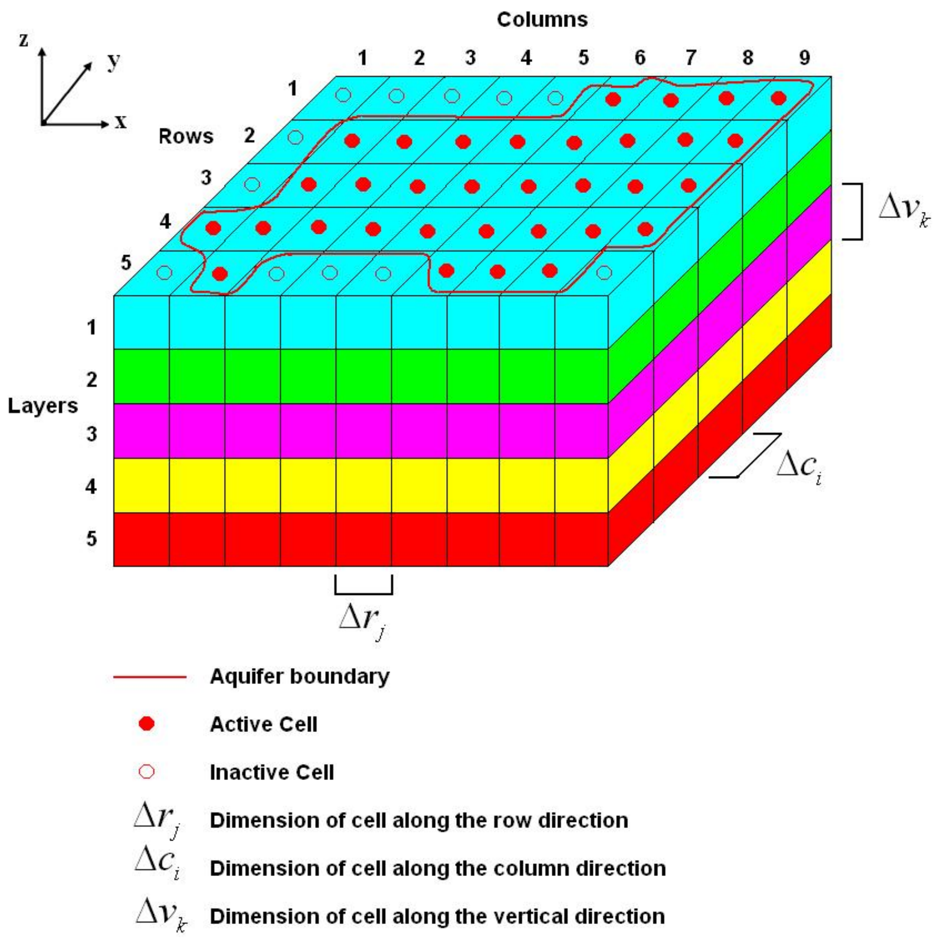
Columns
1 2 3 4 5 6 7 8 9
z
y
x
Rows
1 2 3 4 5
Layers
1 2 3 4 5
Δv_k
Δc_i
Δr_j
Aquifer boundary
Active Cell
Inactive Cell
Δr_j Dimension of cell along the row direction
Δc_i Dimension of cell along the column direction
Δv_k Dimension of cell along the vertical direction

Figure 3.1: Three-dimensional grid.

In Figure 3.1 we see a three-dimensional spatial discretization of an aquifer system as used in MODFLOW (see section 1.2). We have a grid of blocks, which we will call cells from now on. The location of such a cell is described in terms of rows, colums, and layers. We use an i, j, k indexing system, where $i = 1, 2, \ldots, NROW$ is the row index, $j = 1, 2, \ldots, NCOL$ is the column index and $k = 1, 2, \ldots, NLAY$ is the layer index. The convention followed in this model is to number the layers from top to bottom. So, an increase in k means a decrease in z. Rows are considered parallel to the x-axis, such that an increase in i corresponds with a decrease in y. Columns are considered to be parallel to the y-axis, such that an increase in j corresponds to an increase in x.

Within each cell there is a point in which head must be calculated. This point is called *node*. We will use a cell-centered formulation in which the nodes are in the centre of the cells.

If we use the conventions used in Figure 3.1 we see that the width in the row direction at a given column j is denoted by Δr_j. The width of a given row i in the column direction is denoted Δc_i. The thickness of cells in an given layer k is denoted Δv_k. So the cell with coordinates (i, j, k) has a volume of $\Delta r_j \, \Delta c_i \, \Delta v_k$. We assume that the grid is rectangular both in horizontal and vertical direction.

3.1.2 Finite-Volume Equation

We use the continuity equation to develop the finite-volume form of the groundwater flow equation. The sum of all flow into and out of the cell must equal the rate of storage within the cell. We multiply Equation (3.1) by the volume of a cell (ΔV). If we assume the density to be constant we have for a cell:

$$\sum Q_i = SS\frac{\Delta h}{\Delta t}\Delta V, \tag{3.2}$$

where:

Q_i	flow rate into the cell $[L^3T^{-1}]$,
SS	specific storage, the volume of water that can be injected per unit of volume of aquifer material per unit of change in head $[L^{-1}]$,
ΔV	volume of the cell $[L^3]$,
Δh	change in head over a time interval of length Δt $[L]$.

The right hand term is equivalent to the volume of the water taken into storage over a time interval Δt given a change in head of Δh. Outflows and loss are represented in Equation (3.2) by defining outflow as negative inflow and loss as negative gain.

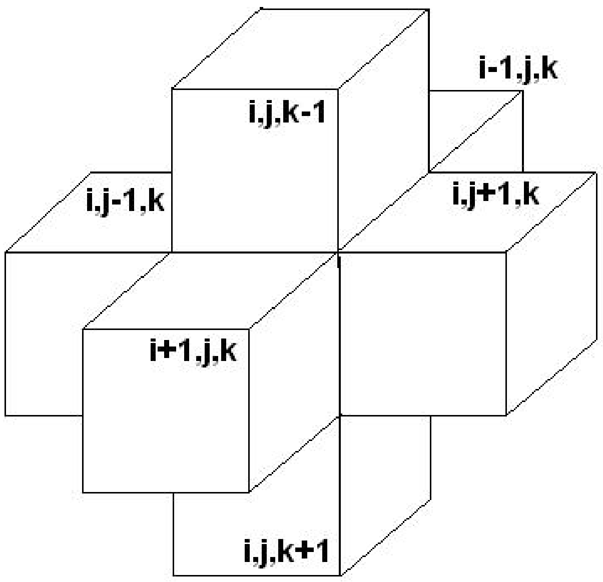

Figure 3.2: Indices for the six cells surrounding cell (i, j, k).

In Figure 3.2 we see the six aquifer cells adjacent to cell (i, j, k). For simplification we consider flow positive if they enter a cell. The negative sign usually incorporated in Darcy's Law has been dropped from all terms. If we now follow these conventions, the flow into cell (i, j, k) in the row direction from cell $(i, j-1, k)$ is given by Darcy's Law as (see also Figure 3.3):

$$q_{i,j-1/2,k} = KR_{i,j-1/2,k}\Delta c_i \Delta v_k \left(\frac{h_{i,j-1,k} - h_{i,j,k}}{\Delta r_{j-1/2}}\right), \tag{3.3}$$

where:

$h_{i,j,k}$	head at node i, j, k,
$q_{i,j-1/2,k}$	volumetric flow rate through the face between cells (i, j, k) and $(i, j-1, k)$ $[L^3T^{-1}]$ (blue area in Figure 3.3),
$KR_{i,j-1/2,k}$	hydraulic conductivity along the row between nodes i, j, k and $i, j-1, k$ $[LT^{-1}]$,
$\Delta c_i \Delta v_k$	area of the cell faces normal to the row direction,
$\Delta r_{j-1/2}$	distance between nodes i, j, k and $i, j-1, k$ $[L]$.

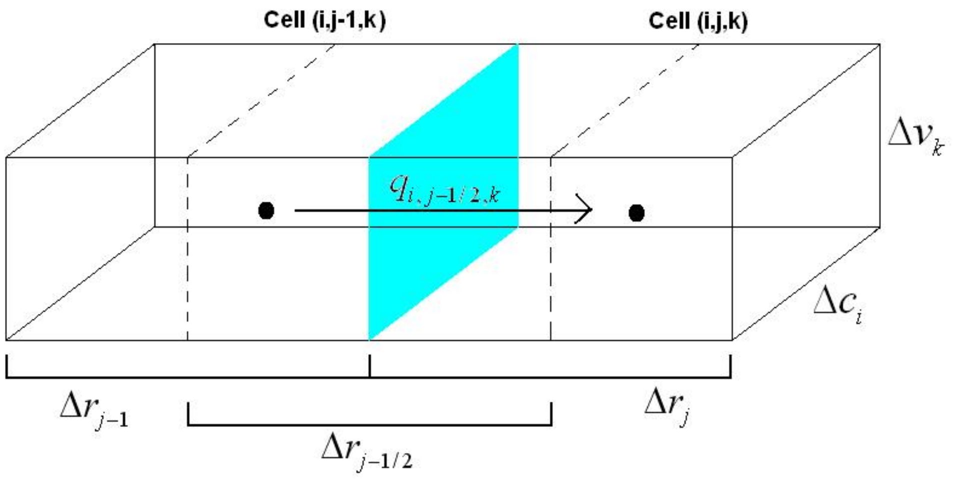

Figure 3.3: Flow into cell (i, j, k) from cell $(i, j-1, k)$.

Now Equation 3.3 gives the exact flow in one dimension for the steady-state case. In this equation it is the flow through a block of aquifer from node $i, j-1, k$ to node i, j, k, having a cross sectional area $\Delta c_i \Delta v_k$. The conductivity $KR_{i,j-1/2,k}$ of the material between nodes i, j, k and $i, j-1, k$ is the effective hydraulic conductivity for the entire region between the nodes. It is normally calculated as a harmonic mean. The term $1/2$ is used to indicate the region *between* nodes.

In the same way we can derive equations for the flow in the five other faces of the cell (i, j, k). So we have for the flow in the row direction through the face between cells (i, j, k) and $(i, j+1, k)$:

$$q_{i,j+1/2,k} = KR_{i,j+1/2,k}\Delta c_i \Delta v_k \frac{(h_{i,j+1,k} - h_{i,j,k})}{\Delta r_{j+1/2}}. \tag{3.4}$$

In the column direction we have for the flows through the front and rear face:

$$q_{i+1/2,j,k} = KC_{i+1/2,j,k}\Delta r_j \Delta v_k \frac{(h_{i+1,j,k} - h_{i,j,k})}{\Delta c_{i+1/2}}, \tag{3.5}$$

$$q_{i-1/2,j,k} = KC_{i-1/2,j,k}\Delta r_j \Delta v_k \frac{(h_{i-1,j,k} - h_{i,j,k})}{\Delta c_{i+1/2}}. \tag{3.6}$$

For the vertical direction we have for the inflow through the bottom an upper face:

$$q_{i,j,k+1/2} = KV_{i,j,k+1/2}\Delta r_j \Delta c_i \frac{(h_{i,j,k+1} - h_{i,j,k})}{\Delta v_{k+1/2}}, \tag{3.7}$$

$$q_{i,j,k-1/2} = KV_{i,j,k-1/2}\Delta r_j \Delta c_i \frac{(h_{i,j,k-1} - h_{i,j,k})}{\Delta v_{k-1/2}}. \tag{3.8}$$

The six equations, (3.3) - (3.8), express the flow through a boundary in terms of heads, grid dimensions, and hydraulic conductivity. We simplify the notation by combining hydraulic conductivity and grid dimensions into one single constant *hydraulic conductance*:

$$CR_{i,j-1/2,k} = \frac{KR_{i,j-1/2,k}\Delta c_i \Delta v_k}{\Delta r_{j-1/2}}$$

So, we define conductance as the product of hydraulic conductivity and cross-sectional area of flow divided by the length of the flowpath, which is here the distance between two nodes. For the other equations we can do the same and we end up with the following six equations:

$$q_{i,j-1/2,k} = CR_{i,j-1/2,k}\left(h_{i,j-1,k} - h_{i,j,k}\right), \tag{3.9}$$
$$q_{i,j+1/2,k} = CR_{i,j+1/2,k}\left(h_{i,j+1,k} - h_{i,j,k}\right), \tag{3.10}$$
$$q_{i-1/2,j,k} = CC_{i-1/2,j,k}\left(h_{i-1,j,k} - h_{i,j,k}\right), \tag{3.11}$$
$$q_{i+1/2,j,k} = CC_{i+1/2,j,k}\left(h_{i+1,j,k} - h_{i,j,k}\right), \tag{3.12}$$
$$q_{i,j,k-1/2} = CV_{i,j,k-1/2}\left(h_{i,j,k-1} - h_{i,j,k}\right), \tag{3.13}$$
$$q_{i,j,k+1/2} = CV_{i,j,k+1/2}\left(h_{i,j,k+1} - h_{i,j,k}\right). \tag{3.14}$$

The values CR, CC and CV are the conductances along rows and colums and between layers, respectively. They are calculated the same way as in Equation (1.5)

3.1.3 External Sources

The flow into cell (i, j, k) from the six adjacent cells are governed by the equations (3.9)-(3.14). To account for boundary conditions extra terms need to be added. These flows can depend on the head in the cell, but are independent of the head in the adjacent cells. It is also possible that they are completely independent of the head in the cell. We may represent the flow from outside the aquifer by the expession:

$$a_{i,j,k,n} = p_{i,j,k,n}h_{i,j,k} + q_{i,j,k,n},$$

where:

$a_{i,j,k,n}$	flow from the n^{th} external source into cell (i, j, k) $[L^3T^{-1}]$,
$p_{i,j,k,n}$ and $q_{i,j,k,n}$	constants ($[L^2T^{-1}]$ and $[L^3T^{-1}]$ respectively).

If a source is independ of the head, we can set $p_{i,j,k.n}$ equal to zero. If we have N external sources or stresses, then we can express the general external flow for cell (i, j, k) as:

$$\sum_{n=1}^{N} a_{i,j,k,n} = P_{i,j,k}h_{i,j,k} + Q_{i,j,k},$$

with:

$$P_{i,j,k} = \sum_{n=1}^{N} p_{i,j,k,n},$$

$$Q_{i,j,k} = \sum_{n=1}^{N} q_{i,j,k,n}.$$

If we now apply the continuity equation (3.2) to cell (i, j, k), we get:

$$\begin{aligned} & q_{i,j-1/2,k} + q_{i,j+1/2,k} + q_{i-1/2,j,k} + q_{i+1/2,j,k} \\ & + q_{i,j,k-1/2} + q_{i,j,k+1/2} + P_{i,j,k} h_{i,j,k} + Q_{i,j,k} \\ & = SS_{i,j,k} \left(\Delta r_j \Delta c_i \Delta v_k\right) \tfrac{\Delta h_{i,j,k}}{\Delta t}, \end{aligned} \tag{3.15}$$

where:

$\Delta h_{i,j,k}/\Delta t$	finite difference approximation for the derivative of head with respect to time $[LT^{-1}]$,
$SS_{i,j,k}$	the specific storage of cell (i, j, k) $[L^{-1}]$,
$\Delta r_j \Delta c_i \Delta v_k$	volume of cell (i, j, k) $[L^3]$.

Now we can substitute Equations (3.9) - (3.14) into Equation (3.15) which gives the *finite volume approximation* for cell (i, j, k) as:

$$\begin{aligned} & CR_{i,j-1/2,k} \left(h_{i,j-1,k} - h_{i,j,k}\right) + CR_{i,j+1/2,k} \left(h_{i,j+1,k} - h_{i,j,k}\right) \\ & + CC_{i-1/2,j,k} \left(h_{i-1,j,k} - h_{i,j,k}\right) + CC_{i+1/2,j,k} \left(h_{i+1,j,k} - h_{i,j,k}\right) \\ & + CV_{i,j,k-1/2} \left(h_{i,j,k-1} - h_{i,j,k}\right) + CV_{i,j,k+1/2} \left(h_{i,j,k+1} - h_{i,j,k}\right) \\ & + P_{i,j,k} h_{i,j,k} + Q_{i,j,k} = SS_{i,j,k} \left(\Delta r_j \Delta c_i \Delta v_k\right) \tfrac{\Delta h_{i,j,k}}{\Delta t}. \end{aligned} \tag{3.16}$$

More on the external sources can be found in Paragraph 3.2

3.1.4 Discretization in Time

We still need to express the finite-difference approximation for the time derivative of head in terms of heads and times. Let t^m and t^{m-1} denote the current and past time, respectively. An approximation to the time derivative of head at time t^m is given by:

$$\frac{\Delta h_{i,j,k}}{\Delta t} \simeq \frac{h^m_{i,j,k} - h^{m-1}_{i,j,k}}{t^m - t^{m-1}}.$$

This method is called the *Euler backwards method*. The term $h^m_{i,j,k}$ denotes the head value at cell (i, j, k) at time t^m. We can approximate the time derivative of head in other ways, but the Euler backwards method is unconditionally numerically stable. Stability means that a small pertubation in the initial conditions does not effect the solution much. So we prefer the backward-difference approach even though this leads to large systems of equations that must be solved simultaneously for each time step. The advantage of a stable method is that we can take bigger time steps without caring about stability.

We can rewrite Equation (3.16) in a backwards-difference form. We must specify flow terms at t^m, the end of a time interval, and approximate the derivative of the head over the time interval from t^{m-1} to t^m. So we get:

$$\begin{aligned}
&CR_{i,j-1/2,k}\left(h^m_{i,j-1,k}-h^m_{i,j,k}\right)+CR_{i,j+1/2,k}\left(h^m_{i,j+1,k}-h^m_{i,j,k}\right)\\
&+CC_{i-1/2,j,k}\left(h^m_{i-1,j,k}-h^m_{i,j,k}\right)+CC_{i+1/2,j,k}\left(h^m_{i+1,j,k}-h^m_{i,j,k}\right)\\
&+CV_{i,j,k-1/2}\left(h^m_{i,j,k-1}-h^m_{i,j,k}\right)+CV_{i,j,k+1/2}\left(h^m_{i,j,k+1}-h^m_{i,j,k}\right)\\
&+P_{i,j,k}h^m_{i,j,k}+Q_{i,j,k}=SS_{i,j,k}\left(\Delta r_j\Delta c_i\Delta v_k\right)\frac{h^m_{i,j,k}-h^{m-1}_{i,j,k}}{t^m-t^{m-1}}.
\end{aligned} \tag{3.17}$$

Since we can write down for each active cell an equation of this type, we are left with a system of "n" equations in "n" unknowns. Such a system can be solved simultaneously.

We start by taking $m = 1$ and calculate the values of $h^1_{i,j,k}$ for each active cell. Of course we need to specify the initial head distribution $h^0_{i,j,k}$, the boundary conditions, the hydraulic parameters and the external sources. If we found all the values for $h^1_{i,j,k}$ we can go on to calculate the values of $h^2_{i,j,k}$ by using $h^1_{i,j,k}$ as an "initial" condition. This process is continued for as many time steps as necessary to cover the time range of interest.

3.1.5 System of Equations

By reordering the terms of Equation (3.17) we can make a system of equations of the form $Ah = q$. All terms that are independent of head at the end of the current timestep we put on the right-hand side (RHS). All terms of $h^m_{i,j,k}$ that has nothing to do with the conductance between nodes we put into a single term ($HCOF$). We then write Equation (3.17) as:

$$\begin{aligned}
&CR_{i,j-1/2,k}h^m_{i,j-1,k}+CR_{i,j+1/2,k}h^m_{i,j+1,k}+CC_{i-1/2,j,k}h^m_{i-1,j,k}\\
&+CC_{i+1/2,j,k}h^m_{i+1,j,k}+CV_{i,j,k-1/2}h^m_{i,j,k-1}+CV_{i,j,k+1/2}h^m_{i,j,k+1}\\
&+\Big(-CR_{i,j-1/2,k}-CR_{i,j+1/2,k}-CC_{i-1/2,j,k}h^m_{i-1,j,k}-CC_{i+1/2,j,k}-\\
&CV_{i,j,k-1/2}-CV_{i,j,k+1/2}+HCOF_{i,j,k}\Big)h^m_{i,j,k}=RHS_{i,j,k},
\end{aligned} \tag{3.18}$$

where:

$$\begin{aligned}
HCOF_{i,j,k} &= P_{i,j,k}-\frac{SS_{i,j,k}\Delta r_j\Delta c_i\Delta v_k}{t^m-t^{m-1}},\\
RHS_{i,j,k} &= -Q_{i,j,k}-SS_{i,j,k}\Delta r_j\Delta c_i\Delta v_k\frac{h^{m-1}_{i,j,k}}{t^m-t^{m-1}}.
\end{aligned}$$

Now this entire system of equations, which includes one equation for each variable head cell in the grid, may be written in matrix form as

$$Ah = q,$$

where:

A A matrix of the coefficients of head, for all the active nodes in the grid,

h vector of head values at the end of time step m for all active nodes in the grid,

q vector of the constant terms, RHS, of all active nodes in the cell.

The matrix A now has the form as shown in Figure 3.4. The diagonal are the elements corresponding to $h^m_{i,j,k}$ in Equation (3.18). The first off-diagonal are exactly the CR-values, the second off diagonal are the CC-values and the last off-diagonal are the CV-values. When you only have one layer (2D-problem) then the third off-diagonal will not appear.

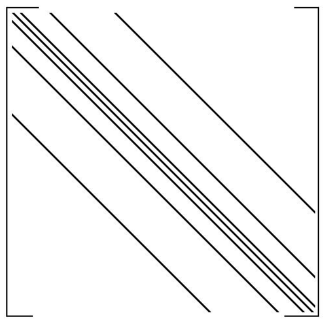

Figure 3.4: Form of system matrix A.

3.2 Simulation of Boundaries in MODFLOW

In MODFLOW we do not need to formulate an equation of the form of Equation (3.17) for every cell in a model grid. The status of certain cells is specified in advance to simulate the boundary conditions of the problem. We group the cells that simulate the boundary conditions into two categories:

- contstant-head cells,
- no-flow cells.

A constant-head cell is used if for that cell the head is specified for each time. The head values do not change as a result of solving the flow equations. The no-flow cells are those for which no flow into or out of the cell is permitted. We will call the remaining cells *variable head* cells. These cells are characterized by heads that are unspecified and free to vary in time. For each variable-head cell in the grid we must formulate an equation of the form of Equation (3.17). The resulting system must be solved simultaneously for each time step in the simulation.

We can use constant-head cells and no-flow cells to represent conditions along various hydraulic boundaries inside the grid. To explain this, we can for example look at Figure 3.5. It shows a configuration of an aquifer boundary superimposed onto a grid of cells generated for the model.

Although the aquifer is of irregular shape, our model grid is always rectangular in outline. If the boundary of the aquifer coincide with the outside edge of the grid, we do not need special designations since MODFLOW does not compute inter-cell flow

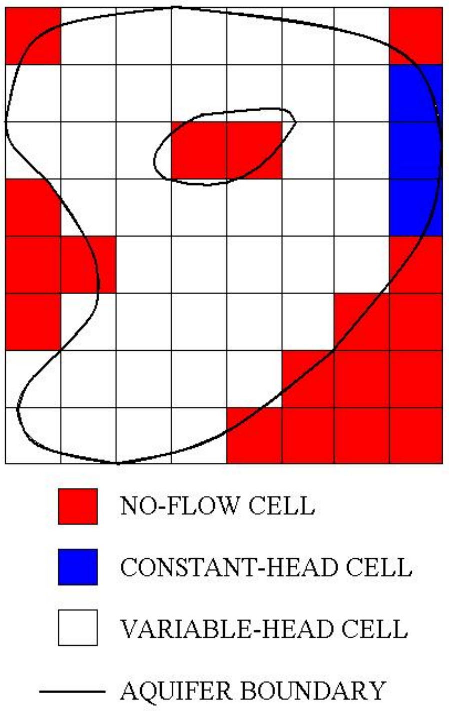

Figure 3.5: Discretized aquifer showing boundaries and cell designations.

through the outside edges of the grid, including top and bottom. Within the grid we use no-flow cells to delete the part of the grid beyond the aquifer boundary. The constant-head cells can be used, for example, if at that place the aquifer is in direct contact with major surface water features.

Other boundary conditions can be simulated by using external source terms possible in combination with no-flow cells. In MODFLOW this is done by stress packages. These packages are called stress packages because in MODFLOW we define stress periods. These stress periods are time intervals in which the input data for all external sources are constant. These stress periods are subdevided into time steps. See also Figure 3.6

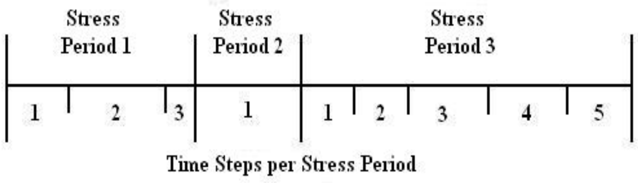

Figure 3.6: Division of simulation time into stress periods and time steps.

3.2.1 Stress Packages

The stress in MODFLOW adds terms to the flow equation representing inflow and outflow. Mathematically, these are boundary conditions. There are two types of important boundary conditions. First we have the constant flux conditions, or the so called Neumann conditions. The WEL and the RCH packages simulate such conditions. Furthermore we have the so called Cauchy boundary conditions. These are head-dependent boundary conditions. The GHB, RIV and DRN packages simulate such conditions.

The finite-volume flow equation for MODFLOW is Equation (3.18). The equation is formulated such that the inflows are added to the left side, with the outflows represented as negative inflows. Stresses are added to the equation by adding terms to $HCOF$ and RHS. A stress term that is a coefficient of head, $h_{i,j,k}$, is added to $HCOF$. A constant stress term is subtracted from RHS.

Well Package (WEL)

With the Well Package we can simulate features such as wells that withdraw water from or add water to the aquifer at a constant rate during a stress period. The rate is independent on both the cell area and the head in the cell. Mostly the package is used to simulate wells, either discharge or recharge. However, the package can also be used to simulate any features for which the recharge or discharge can be specified directly. The dimension of recharge or discharge is $[L^3T^{-1}]$.

The user has to specifiy the flow rate, Q, for a well and a fluid volume per unit time at which water is added to an aquifer. Negative values are used to indicate well discharge (pumping). The user can specify the location of the well by specifying the row, column and layer number of the well. The value of Q for each well is subtracted from the RHS of Equation (3.18) for the cell containing that well. Figure 3.9 shows the initial condition near a well in one of the layers.

Recharge Package (RCH)

The Recharge Package is designed to simulate areally distributed recharge to the ground-water system. Most commonly, recharges occurs as a result of precipitation that enters to the ground-water system and evaporation leaving the system. These processes occur normally at the surface, but also processes in other layers can be simulated. The recharge applied to the model is defined as:

$$QR_{i,j} = I_{i,j}\, \Delta r_j\, \Delta c_i,$$

where:

$QR_{i,j}$	the recharge flow rate applied to the model at horizontal cell location (i, j) expressed as a fluid volume per time $[L^3T^{-1}]$,
$I_{i,j}$	the recharge flux applicable to the map area of the cell $[LT^{-1}]$,
Δr_j	dimension of cell allong the row direction,
Δc_i	dimension of cell allong the collumn direction.

The recharge flow rate, $QR_{i,j}$, associated with a given horizontal cell location (i, j) and a vertical location, k, is subtracted from the value of $RHS_{i,j,k}$. Since recharge is independent of the aquifer head, nothing is added to the coefficient of head, $HCOF_{i,j,k}$.

General-Head Boundary Package (GHB)

The General-Head Boundary Package simulates flow into or out of a cell $(i, j, k,)$, from an external source in proportion to the difference between the head in the cell and head assigned to the external source. The constant of proportionality is called the *boundary* *conductance*. We have the following linear relation between flow into the cell and head in the cell:

$$QB_n = CB_n \left(HB_n - h_{i,j,k} \right), \tag{3.19}$$

where:

n	boundary number
QB_n	flow into cell (i, j, k) from the boundary $[L^3T^{-1}]$,
CB_n	the boundary conductance $[L^2T^{-1}]$,
HB_n	the head assigned to the external source $[L]$,
$h_{i,j,k}$	the head in cell (i, j, k) $[L]$.

A graph of the inflow from a general-head boundary and head in the cell containing the boundary as given by Equation (3.19) is shown in Figure 3.10.

River Package (RIV)

Rivers and streams contribute water to or drain water from the ground-water system. This depends on the head gradient between the river and the ground-water regime. The River Package simulates the effects of flow between surface-water features and ground-water systems. The River Package does not simulate surface-water flow in the river, but only the river/aquifer seepage.

By assumption, measurable head losses between the river and the aquifer are limited to those across the riverbed layer itself. Another assumption is that the water level does not drop below the bottom of the riverbed layer. Under these assumptions, flow between the river and the ground-water system for each n is given by:

$$\begin{cases} QRIV_n = CRIV_n \left(HRIV_n - h_{i,j,k} \right) & h_{i,j,k} > RBOT_n, \\ QRIV_n = CRIV_n \left(HRIV_n - RBOT_n \right) & h_{i,j,k} \leq RBOT_n, \end{cases} \tag{3.20}$$

where:

$QRIV_n$	flow between the river and the aquifer, taken positive if it is directed into the aquifer $[L^3T^{-1}]$,
$HRIV_n$	the water level (stage) in the river $[L]$,
$CRIV_n$	the hydraulic conductance of the river-aquifer interconnection $[L^2T^{-1}]$,
$RBOT_n$	bottom of riverbed layer $[L]$,
$h_{i,j,k}$	the head at the node in the cell underlying the river reach $[L]$.

The graph of the flow from a river as a function of the head in the corresponding cell as calculated in Equation (3.20) is shown in Figure 3.11. Figure 3.12 shows how the river "De Maas" is implemented in the IBRAHYM model.

Drain Package (DRN)

The effects of features as agricultural drains are simulates with the Drain Package. Drains remove water from the aquifer at a rate proportional to the difference between the head in the aquifer and some fixed head or elevation, called the drain elevation. The drain works as long as the head in the aquifer is above the elevation. If the aquifer head falls below the drain elevation, then the drain has no effect on the aquifer. The constant of proportionality is called the drain conductance. A mathematical statement of this situation is:

$$\begin{cases} QD_n = CD_n\,(HD_n - h_{i,j,k}) & h_{i,j,k} > HD_n, \\ QD_n = 0 & h_{i,j,k} \leq HD_n, \end{cases} \tag{3.21}$$

where:

QD_n	flow from aquifer into the drain $[L^3T^{-1}]$,
CD_n	drain conductance $[L]$,
HD_n	Drain elevation $[L]$,
$h_{i,j,k}$	the head in the cell containing the drain $[L]$,
n	number of the drain.

Figure 3.13 shows a graph of flow from a drain and head in the cell containing the drain as defined by Equation (3.21). Figure 3.14 shows where drains are constructed in and around Eindhoven which are implemented in the IBRAHYM model.

3.3 Faults in MODFLOW

A fault in the subsoil effects the system matrix as defined in section 3.1.5. In MODFLOW a fault is always translated to a fault on the cell boundaries as shown in Figure 3.7. Faults are modelled in MODFLOW as a change in the CR and the CC values between two grid-points. So a fault causes a change in the system matrix.

In MODFLOW when a fault occurs it is modelled between two cells. So we have to adapt the conductance between two cells. The formula for the conductance between two cells is given by Equation (1.5). In case of a fault we use the following formula:

$$CR_{i,j+1/2,k} = \frac{CR^{original}_{i,j+1/2,k} \times C_{barrier}}{CR^{original}_{i,j+1/2,k} + C_{barrier}}.$$

Here:

$$C_{barrier} = \frac{K_{barrier}\Delta v_{barrier} DELC_i}{L_{barrier}}$$

where:

$K_{barrier}$	is hydraulic conductivity of the barrier,
$\Delta v_{barrier}$	is the vertical thickness of the barrier,
$L_{barrier}$	is the distance across the barrier in the flow direction.

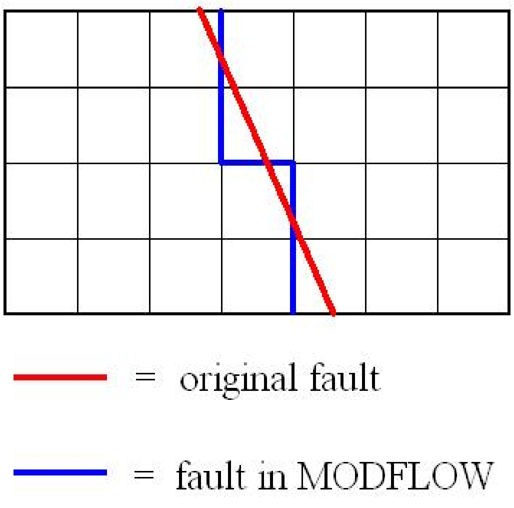

Figure 3.7: Fault as modelled by MODFLOW.

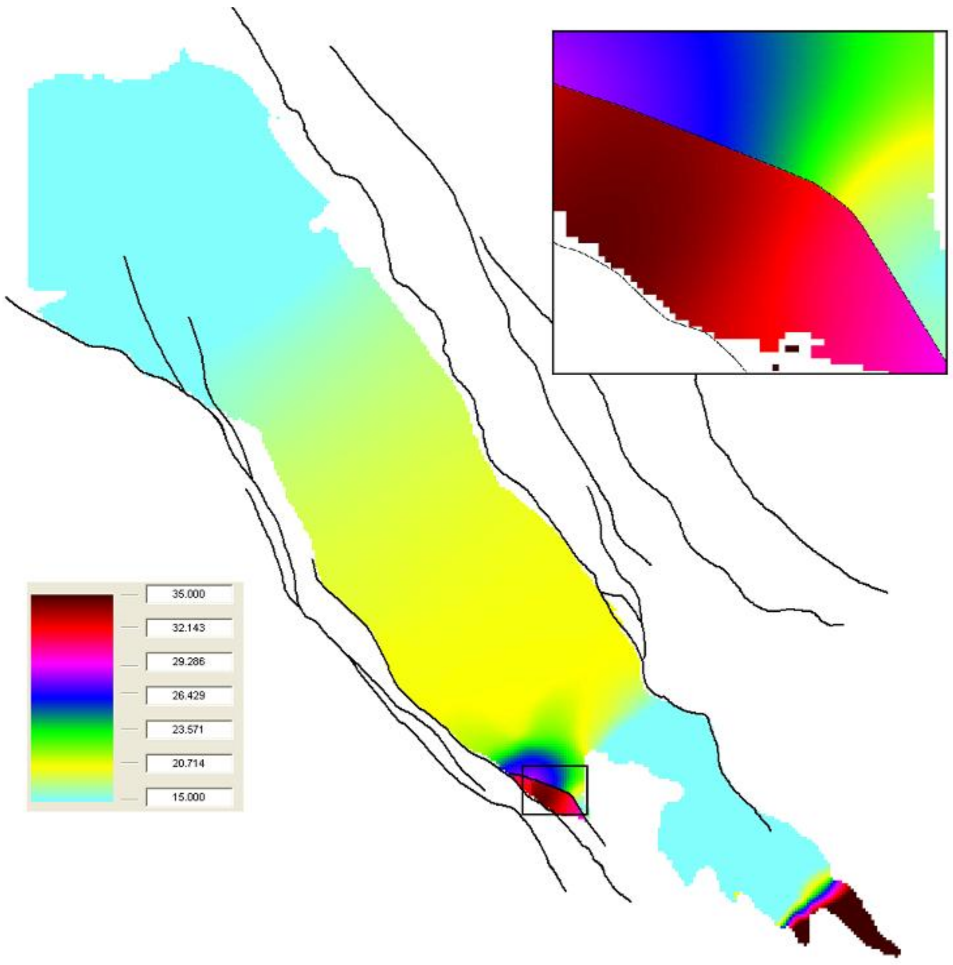

Figure 3.8: Heads in the 19th layer in the IBRAHYM model.

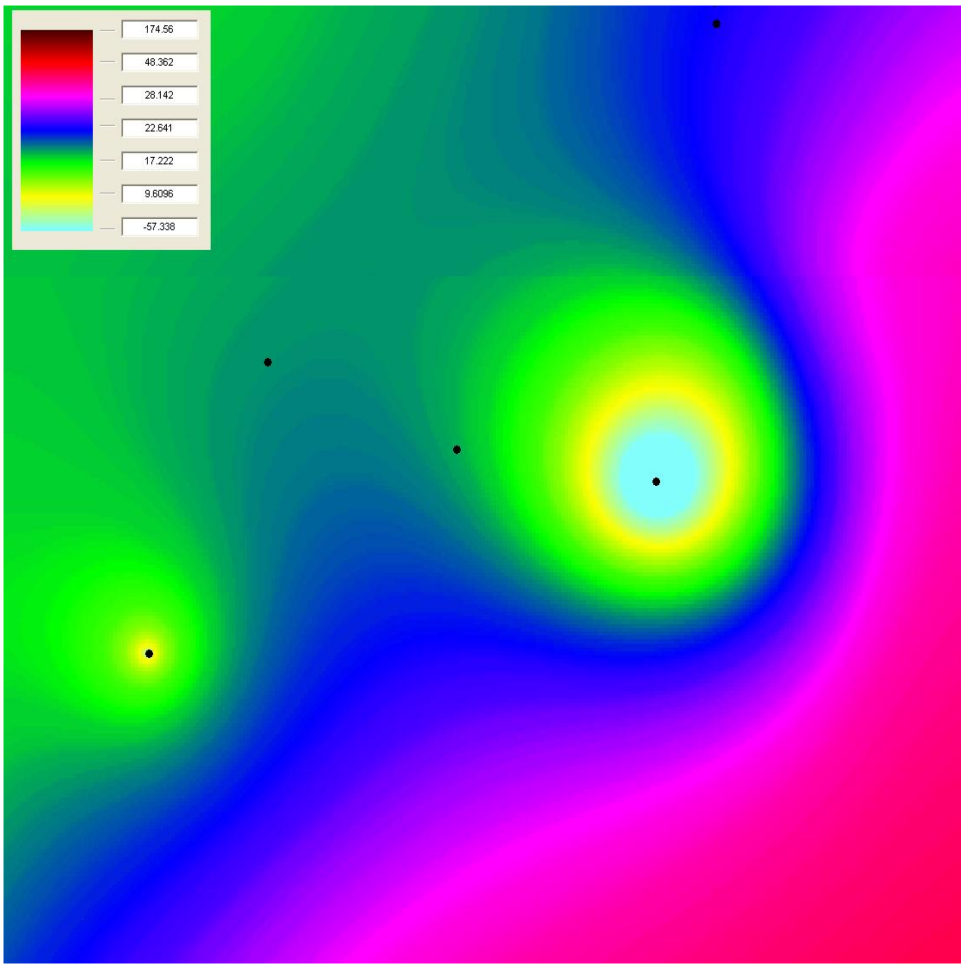

Figure 3.9: Example of the heads near a well.

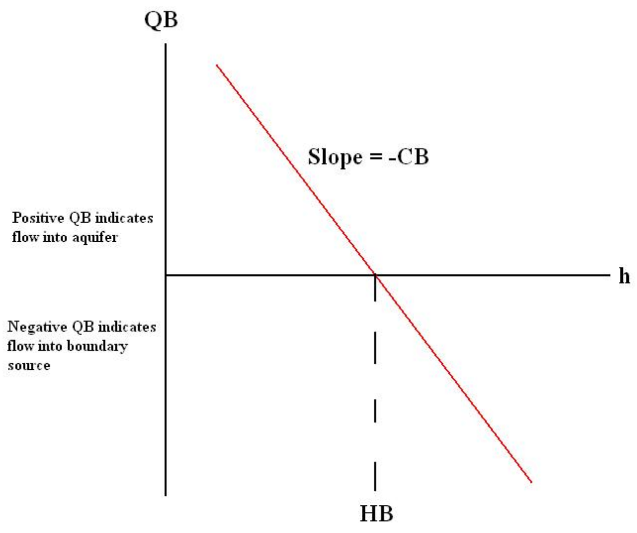

Figure 3.10: Plot of flow, QB, from a general-head boundary source into a cell as function of head.

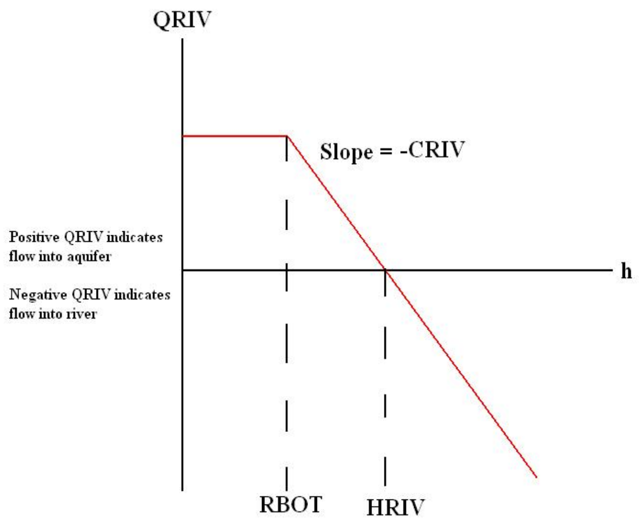

Figure 3.11: Plot of flow, QRIV, from a river into a cell as function of head.

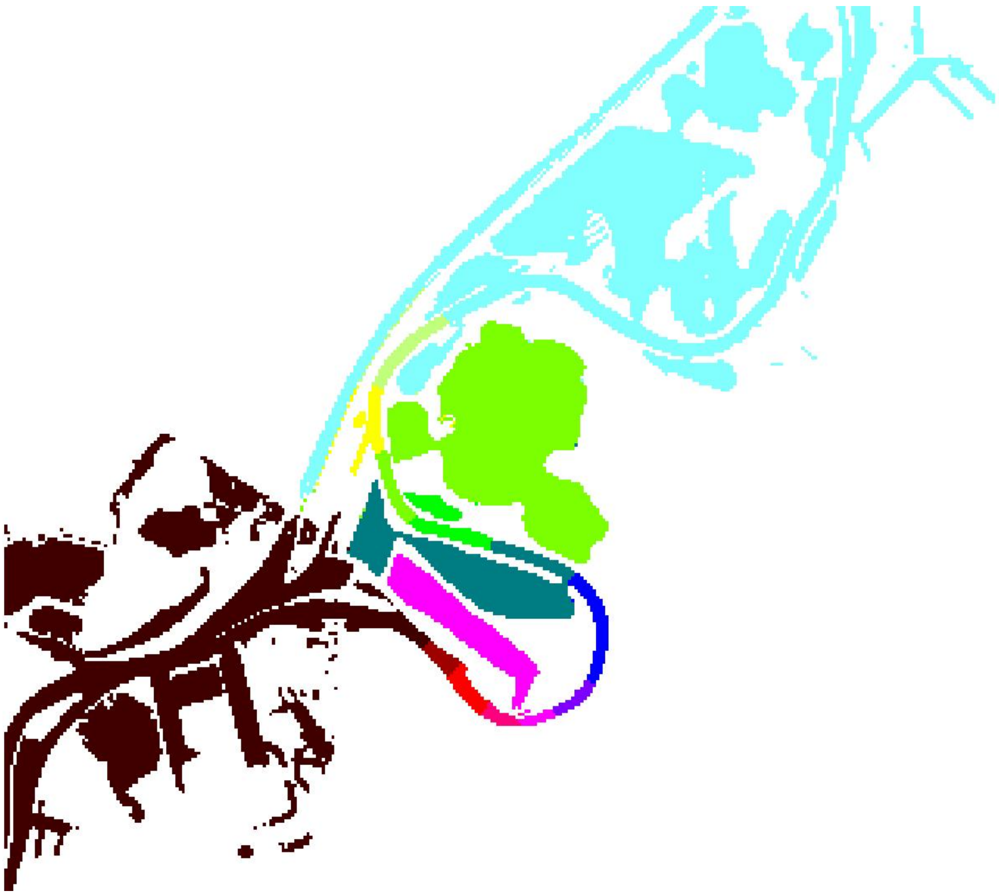

Figure 3.12: Example of a river, de Maas, in MODFLOW.

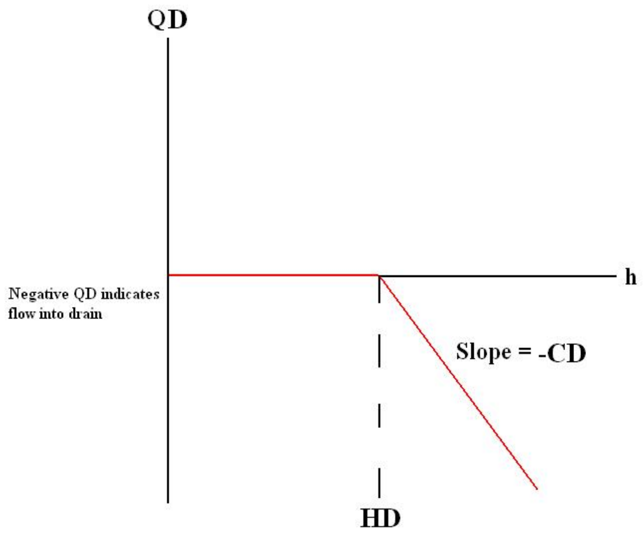

Figure 3.13: Plot of flow, QD, into a drain as a function of head.

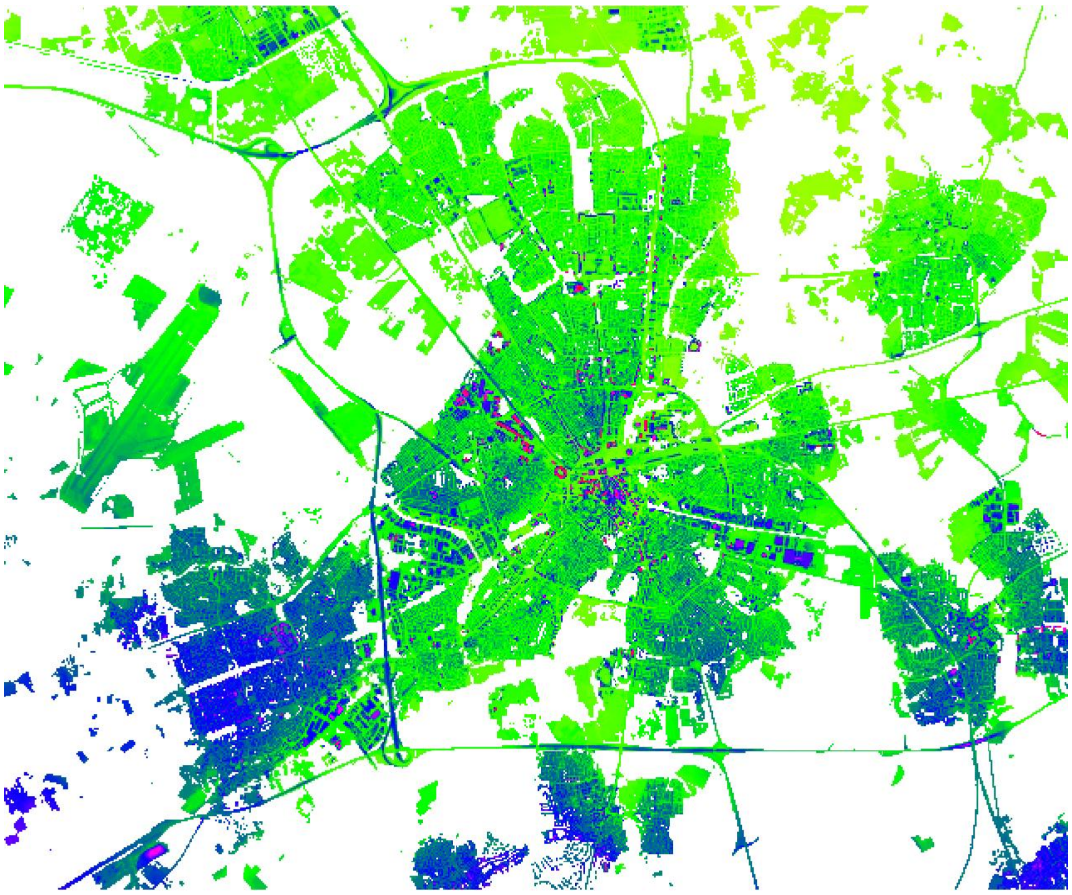

Figure 3.14: Overview of a drain system (Eindhoven) in MODFLOW.

Chapter 4

Iterative Solvers for Linear Systems

The iterative solvers we discuss here are used to solve linear algebraic systems of equations [4, 8, 10]. Given an $n \times n$ real matrix and a real vector $b \in \mathbb{R}^n$, the problem considered is: find a vector $y \in \mathbb{R}^n$ such that:

$$Ay = b. \tag{4.1}$$

This equation is a *linear system*. We call the matrix A the *coefficient matrix*, b the *right-hand side vector* and y the *vector of unknowns*. We assume A to be invertible, i.e. $A^{-1}b$ can be computed.

Iterative methods for the solution of linear system of equations are useful in the following cases:

- if the number of iterations necessary is not too big,
- if A is large and sparse,
- if A has a special structure,
- if a good initial guess for y is available, as in time-stepping methods,
- if A is, for example, not known explicitly.

In other cases it is more advantageous to use direct methods.

4.1 Basic Iterative Methods

From equation (4.1), we construct a splitting $A = M - N$, such that M is easy invertible. Note that we have that $N = M - A$. We can now write:

$$My - Ny = b, \tag{4.2}$$

which is still exact. We now use the following iteration:

$$My^{(k+1)} - Ny^{(k)} = b, \tag{4.3}$$

or if we rewrite it:

$$\begin{aligned} y^{(k+1)} &= M^{-1}Ny^{(k)} + M^{-1}b \\ &= M^{-1}\left(M - A\right)y^{(k)} + M^{-1}b \\ &= y^{(k)} - M^{-1}Ay^{(k)} + M^{-1}b \\ &= y^{(k)} + M^{-1}\left(b - Ay^{(k)}\right). \end{aligned} \tag{4.4}$$

In general we use the following iteration:

$$y^{(k+1)} = Gy^{(k)} + f, \ (k = 0, 1, 2, \ldots), \tag{4.5}$$

so that the system $y = Gy + f$ is equivalent to the original problem (4.1). G is called the *iteration matrix*.

If we look at equation (4.5) then we see that for a basic iterative method we have:

$$G = M^{-1}N = M^{-1}\left(M - A\right) = I - M^{-1}A, \ \ f = M^{-1}b.$$

This recurrence is also called a *linear fixed-point iteration*. If we take $M = D$, where D contains the diagonal elements of A on its diagonal and all other elements are zero, we have the Jacobi-method. If we take $M = D - E$, where $-E$ is the strict lower part of A, we have the forward Gauss-Seidel-method.

We can view the iteration $y^{(k+1)} = Gy^{(k)} + f$ as a technique to solve the system

$$(I - G)y = f$$

Since G has the form $G = I - M^{-1}A$, we can rewrite this system as

$$M^{-1}Ay = M^{-1}b$$

This system has the same solution as the original system and is called *left preconditioned system*, where M is the *preconditioning matrix* or *preconditioner*. There also exist right preconditioned systems. Preconditioning is used to speed up the convergence. The idea is that M is more or less similar to A, but easy to invert.

4.2 Projection Methods

The idea of *projection techniques* is to extract an approximate solution to the problem in Equation (4.1) from a subspace of $\mathbb{R}^n$. Let $\mathcal{K}_m \subset \mathbb{R}^n$ be this subspace of candidate approximants, or search subspace. Let m be the dimension of $\mathcal{K}_m$, then, in general, m constraints must be imposed to be able to extract such an approximation from $\mathcal{K}_m$. Sometimes one imposes m (independent) orthogonality conditions. Specifically, the residual vector $b - Ay$ is constrained to be orthogonal to m linearly independent vectors. These constraints define another subspace of dimension m. This *subspace of constraints* we denote with $\mathcal{L}_m$. In many different mathematical methods this simple framework is known as the Petrov-Galerkin condition.

Let A be an $n \times n$ real matrix and $\mathcal{K}_m$ and $\mathcal{L}_m$ two m-dimensional subspaces of $\mathbb{R}^n$. We want to have a projection technique onto the subspace $\mathcal{K}_m$ and orthogonal to $\mathcal{L}_m$. It should be a process which finds an approximate solution $\tilde{y}$ to (4.1) where we impose that $\tilde{y}$ belongs to $\mathcal{K}_m$ and that the new residual vector $b - A\tilde{y}$ is orthogonal to $\mathcal{L}_m$:

$$\text{find } \tilde{y} \in \mathcal{K}_m, \ \text{ such that } \ b - A\tilde{y} \perp \mathcal{L}_m.$$

If we have the knowledge of an initial guess y_0 and we want to exploit that, then we look for our approximation in the affine space $y_0 + \mathcal{K}_m$. So our problem then becomes:

$$\text{find } \tilde{y} \in y_0 + \mathcal{K}_m, \text{ such that } b - A\tilde{y} \perp \mathcal{L}_m. \tag{4.6}$$

If we write $\tilde{y}$ in the form $\tilde{y} = y_0 + \delta$ and we define the initial residual vector as:

$$r_0 := b - Ay_0,$$

then Equation (4.6) becomes $b - A\tilde{y} \perp \mathcal{L}_m$, or:

$$r_0 - A\delta \perp \mathcal{L}_m.$$

Now we can find the approximate solution $\tilde{y}$ by using that:

$$\begin{cases} \tilde{y} = y_0 + \delta & ,\delta \in \mathcal{K}_m \\ (r_{new}, \mathbf{w}) = 0 & ,\forall \mathbf{w} \in \mathcal{L}_m \end{cases} \tag{4.7}$$

where r_{new} is defined as $r_0 - A\delta$ (see also figure 4.1).

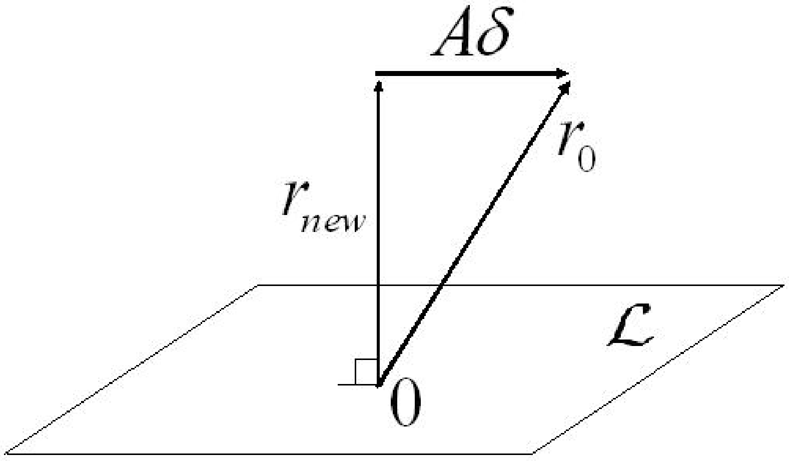

Figure 4.1: Interpretation of the orthogonality condition.

This is a projection method in its most general form. Typically, a new projection step uses a new pair of subspaces $\mathcal{K}_m$ and $\mathcal{L}_m$ and a new initial guess y_0 equal to the most recent approximation obtained from the previous step.

4.2.1 Matrix Representation

We define $V = [v_1 \cdots v_m]$ to be an $n \times m$ matrix whose column-vectors form a basis of $\mathcal{K}_m$ and, similarly, $W = [w_1 \cdots w_m]$ an $n \times m$ matrix whose column-vectors form a basis of $\mathcal{L}_m$. Now write the approximate solution as:

$$y = y_0 + Vz.$$

If z would be known, then we would have:

$$\begin{aligned} Ay &= Ay_0 + AVz, \\ Ay - Ay_0 &= AVz, \\ b - Ay_0 &= AVz, \\ r_0 &= AVz. \end{aligned}$$

Since we still have to determine z we use the orthogonality relation, which states:

$$\text{span}\{W\} \perp r_0 - AVz.$$

Therefore we have:

$$W^T r_0 = W^T AVz.$$

If we assume the matrix $W^T AV$ to be non-singular we can conclude that

$$z = \left(W^T AV\right)^{-1} W^T r_0,$$

and we end up with:

$$\tilde{y} = y_0 + V\left(W^T AV\right)^{-1} W^T r_0.$$

4.3 The Krylov Subspace

In Equation (4.4) we compute the iterates by the recursion:

$$y^{(k+1)} = y^{(k)} + M^{-1}\left(b - Ay^{(k)}\right) = y^{(k)} + M^{-1} r^{(k)}$$

If we write out the first terms of this process we obtain:

$$\begin{aligned} & y^{(0)}, \\ y^{(1)} &= y^{(0)} + M^{-1} r^{(0)}, \\ y^{(2)} &= y^{(1)} + M^{-1} r^{(1)} \\ &= y^{(1)} + M^{-1}\left(b - Ay^{(1)}\right), \\ &= y^{(1)} + M^{-1}\left(b - Ay^{(0)} - AM^{-1} r^{(0)}\right), \\ &= y^{(1)} + M^{-1}\left(r^{(0)} - AM^{-1} r^{(0)}\right), \\ &= y^{(1)} + M^{-1} r^{(0)} - M^{-1}AM^{-1} r^{(0)}, \\ &= y^{(0)} + M^{-1} r^{(0)} + M^{-1} r^{(0)} - M^{-1}AM^{-1} r^{(0)}, \\ &= y^{(0)} + 2M^{-1} r^{(0)} - M^{-1}AM^{-1} r^{(0)}, \\ y^{(3)} &= y^{(2)} + M^{-1} r^{(2)}, \\ &= y^{(2)} + M^{-1}\left(b - Ay^{(2)}\right), \\ &= y^{(2)} + M^{-1}\left(b - A\left(y^{(0)} + 2M^{-1} r^{(0)} - M^{-1}AM^{-1} r^{(0)}\right)\right), \\ &= y^{(2)} + M^{-1} r^{(0)} - 2\left(M^{-1}A\right) M^{-1} r^{(0)} + \left(M^{-1}A\right)^2 M^{-1} r^{(0)}, \\ &= y^{(0)} + 3M^{-1} r^{(0)} - 3\left(M^{-1}A\right) M^{-1} r^{(0)} + \left(M^{-1}A\right)^2 M^{-1} r^{(0)}, \\ & \vdots \end{aligned}$$

which implies that:

$$y^{(k)} = y^{(0)} + \text{span}\left\{M^{-1} r^{(0)}, \left(M^{-1}A\right) M^{-1} r^{(0)}, \ldots, \left(M^{-1}A\right)^{k-1} M^{-1} r^{(0)}\right\}.$$

The subspace $\mathcal{K}_k\left(A; r^{(0)}\right) := \text{span}\left\{r^{(0)}, Ar^{(0)}, \ldots, A^{k-1} r^{(0)}\right\}$ is called the *Krylov subspace* of dimension k corresponding to the matrix A and the initial residual vector $r^{(0)}$. Now a $y^{(k)}$ calculated by the basic iterative method is an element of $y^{(0)} + \mathcal{K}_k\left(M^{-1}A; M^{-1} r^{(0)}\right)$

4.4 Preconditioned Conjugate Gradient Methods

4.4.1 The basic algorithm

To explain the basics of the Conjugate Gradient Method we first make some assumptions. We assume:

- $M = I$,
- $y^{(0)} = 0$,
- $r^{(0)} = b$,
- A is symmetric, i.e. $A^T = A$,
- A is positive definite, i.e. $x^T Ax > 0$, $\forall x \neq 0$.

The first three assumptions are only needed to facilitate the formula's, but are not necessary for the CG method itself. The last two assumptions are necessary for the CG method to work.

Our first idea would be to construct

$$y^{(i)} \in \mathcal{K}_i(A; r^{(0)}) \text{ such that } \|y - y^{(i)}\|_2 \text{ is minimal.}$$

If we look at the first iterate, $y^{(1)}$, we see that it can be written as $y^{(1)} = \alpha_0\, r^{(0)}$, where α_0 is a constant which has to be choosen such that $\|y - y^{(i)}\|_2$ is minimal. Now:

$$\|y - y^{(i)}\|_2^2 = \left(y - \alpha_0 r^{(0)}\right)^T \left(y - \alpha_0 r^{(0)}\right) = y^T y - 2\alpha_0 (r^{(0)})^T y + \alpha_0^2 (r^{(0)})^T r^{(0)}.$$

This is minimal if:

$$\alpha_0 = \frac{(r^{(0)})^T y}{(r^0)^T r^{(0)}},$$

but since we do not know y this choice cannot be determined and we have to look for another solution. We can define the *A-inner product* and the *A-norm* to be:

$$(y, z)_A = y^T Az,$$

$$\|y\|_A = \sqrt{(y, y)_A} = \sqrt{y^T Ay},$$

respectively. If A is a Symmetric Positive Definite (SPD) matrix, then $(y, z)_A$ and $\|y\|_A$ statisfy the rules for inner product and norm respectively. We now want to construct:

$$y^{(i)} \in \mathcal{K}_i(A; r^{(0)}) \text{ such that } \|y - y^{(i)}\|_A \text{ is minimal.}$$

Now:

$$\|y - y^{(i)}\|_A^2 = y^T Ay - 2\alpha_0 (r^{(0)})^T Ay + \alpha_0^2 (r^{(0)})^T Ar^{(0)},$$

which is minimal if:

$$\alpha_0 = \frac{(r^{(0)})^T Ay}{(r^{(0)})^T Ar^{(0)}} = \frac{(r^{(0)})^T b}{(r^{(0)})^T Ar^{(0)}}.$$

We now have a new inner-product which leads to a minimization problem, which is easily solved. In the iterations we compute $y^{(i)}$ such that:

$$\|y - y^{(i)}\|_A = \min_{x \in \mathcal{K}_i(A; r^{(0)})} \|y - x\|_A$$

The solution of this minimization problem leads to the Conjugate Gradient Method. This algorithm can be found, for example, in [4] or in [8].

If we now also use a preconditioner, then we end up with the Preconditioned Conjugate Gradient Method, or the PCG method. The idea behind it is that we solve a transformed system $\tilde{A}\tilde{y} = \tilde{b}$, where:

$$\begin{aligned} &\tilde{A} = P^{-1}AP^{-T}, \\ &\tilde{y} = P^T y, \\ &\tilde{b} = P^{-1}b. \end{aligned}$$

We assume P to be a nonsingular matrix. We now define M to be $M := PP^T$ to be the preconditioner. We now write down the algorithm such that the tildes do not appear. A complete derivation can be found, for example, in [4].

Algorithm 4.1 (PCG)

```
k = 0;                  $y^{(0)}$                                  initialization
                        $r^{(0)} = b - Ay^{(0)}$

while $(r^{(k)} \neq 0)$ do                                           terminial criterion
                        $s^{(k)} = M^{-1}r^{(k)}$                   preconditioning
                        $k := k + 1$

                        if $(k = 1)$ do
                            $p^{(1)} = s^{(0)}$
                        else
                            $\beta_k = \frac{(r^{(k-1)})^T s^{(k-1)}}{(r^{(k-2)})^T s^{(k-2)}}$        update of $p^{(k)}$
                            $p^{(k)} = s^{(k-1)} + \beta_k p^{(k-1)}$
                        end if

                        $v^{(k)} = Ap^{(k)}$
                        $\alpha_k = \frac{(r^{(k-1)})^T s^{(k-1)}}{(p^k)^T v^{(k)}}$
                        $y^{(k)} = y^{(k-1)} + \alpha_k p^{(k)}$          update iterate
                        $r^{(k)} = r^{(k-1)} - \alpha_k v^{(k)}$          update residual
end while
```

For this algorithm we have that the denominator $(r^{(k-2)})^T s^{(k-2)} = (s^{(k-2)})^T M s^{(k-2)}$ never becomes zero while $r^{(k-2)} \neq 0$, because M is a symmetric positive definite matrix.

The PCG method solves the following minimization problem:

$$\|y - y^{(i)}\|_A = \min_{x \in \mathcal{K}_i(M^{-1}A; M^{-1}r^{(0)})} \|y - x\|_A.$$

4.4.2 Incomplete Cholesky Decomposition

To explain these kinds of preconditioners we assume $A \in \mathbb{R}^{n\times n}$ to be a matrix with at most 5 non-zero elements per row. Furthermore we assume A to be symmetric and positive definite. This matrix can, for example, represent a discretization of a partial differential equation on a 2 dimensional grid. If we assume that we have a 5-point stencil and $m = n^2$, then A is of the form:

$$\mathbf{A} = \begin{bmatrix} a_1 & b_1 & & c_1 & & & & \\ b_1 & a_2 & b_2 & & c_2 & & & \emptyset \\ \vdots & \ddots & \ddots & & \emptyset & \ddots & & \\ c_1 & \emptyset & b_m & a_{m+1} & b_{m+1} & & c_{m+1} & \\ \emptyset & \ddots & & \ddots & \ddots & \ddots & & \ddots \end{bmatrix}. \tag{4.8}$$

In order to be optimal with respect to the convergence we take a lower triangular matrix L such that $A = L^T L$ and we take $P = L$. L is called the Cholesky factor. It is known that we have to deal with so called fill-in. If we observe the fill-in, we see a decrease in size if the 'distance' to the non zero element of A increases. This decrease motivates to discard the fill-in elements entirely. This will lead to an *incomplete Cholesky decomposition* of A.

Now we have to construct L first, before we can use it in the PCG algorithm (Algorithm 4.1). Inside the algorithm we need multipications with L^{-1} and L^{-T}. Since it is expensive to calculate these matrices we do not calculate them. If we for example have to calculate $s = L^{-1}r$, we compute s by solving the linear system $Ls = r$. Since L is a lower triangular matrix this is can be done with low effort using Gaussian eliminations.

We now want to compute the incomplete Cholesky decompostion: $A = LD^{-1}L^T - N$. The elements of L and D have to satisfy the following rules:

- $l_{ij} = 0$ for all (i, j) where $a_{ij} = 0 \;\; i > j$,
- $l_{ii} = d_{ii}$,
- $(LD^{-1}L^T)_{ij}$ for all (i, j) where $a_{ij} \neq 0 \;\; i \geq j$.

If the elements of L are given as follows:

$$\mathbf{L} = \begin{bmatrix} \tilde{d}_1 & & & & \\ \tilde{b}_1 & \tilde{d}_2 & & & \\ & \ddots & \ddots & \emptyset & \\ \tilde{c}_1 & & \tilde{b}_m & \tilde{d}_{m+1} & \\ & \ddots & \emptyset & \ddots & \ddots \\ \emptyset & & & & \end{bmatrix}, \tag{4.9}$$

then we have:

$$\left.\begin{array}{rcl} \tilde{d}_i & = & a_i - \frac{b_{i-1}^2}{\tilde{d}_{i-1}} - \frac{c_{i-m}^2}{\tilde{d}_{i-m}} \\ \tilde{b}_i & = & b_i \\ \tilde{c}_i & = & c_i \end{array}\right\} i = 1, \ldots, N. \tag{4.10}$$

If we now define the preconditioner M to be $LD^{-1}L^T$ and we use this M in Algorithm 4.1, we have the *Incomplete Cholesky Conjungate Gradient Method* or ICCG. More information can be found in, for example, [8].

4.4.3 Modified ICCG

The modified incomplete Cholesky preconditioner is constructed by slightly adapted rules. We again split $A = LD^{-1}L^T - N$, but L and D must now satisfy:

- $l_{ij} = 0$ for all (i, j) where $a_{ij} = 0 \;\; i > j$,
- $l_{ii} = d_{ii}$,
- $rowsum(LD^{-1}L^T)$ = $rowsum(A)$ for all rows and $(LD^{-1}L^T)_{ij}$ for all (i, j) where $a_{ij} \neq 0 \;\; i \geq j$.

The consequence of the third point is that we have:

$$LD^{-1}L^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = A \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \Rightarrow (LD^{-1}L^T)^{-1}A \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}. \tag{4.11}$$

We can conclude that if $Ay = b$ and x and/or b are slowly varying vectors, then this modified incomplete Cholesky decomposition is a very good approximation for the inverse of A with respect to x and/or b. We now end up with:

$$\left.\begin{array}{rcl} \tilde{d}_i & = & a_i - (b_{i-1} + c_{i-1})\frac{b_{i-1}}{\tilde{d}_{i-1}} - (b_{i-1} + c_{i-1})\frac{c_{i-m}}{\tilde{d}_{i-m}} \\ \tilde{b}_i & = & b_i \\ \tilde{c}_i & = & c_i \end{array}\right\} i = 1, \ldots, N. \tag{4.12}$$

MODFLOW uses a different MICCG method, with relaxation. In appendix B you can read how this is implemented.

4.5 Convergence, Starting Vectors and Stopping Criteria

Convergence

The rate of convergence of the conjugate gradient method is bounded as a function of the condition number of the system matrix to which it is applied. We let $A \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix and we assume the vector $b \in \mathbb{R}^n$ to represent a discrete function on a grid Ω. We are searching for the vector $y \in \mathbb{R}^n$ on Ω which solves the linear system

$$Ay = b.$$

If we use a finite volume method to discretize a partial differential equation, then we end up with such a system which we have to solve.

We denote the spectrum of A by $\sigma(A)$ and the ith eigenvalue in nondecreasing order by $\lambda_i(A)$, or simply λ_i if it is clear to which matrix we refer. It is known that after k iterations of the conjugate gradient method, the error is bounded by:

$$\|y - y^{(i)}\|_A \leq 2\|y - y^{(0)}\|_A \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k. \tag{4.13}$$

The proof can be found, for example, in [3] or [8]. Here, $\kappa = \kappa(A) = \lambda_n/\lambda_1$ is the spectral condition number of A and the A-norm of y is given by $\|y\|_A = \left(y^T Ay\right)^{1/2}$. In [7] one can find the following bound for the preconditioned conjugate gradient method,

$$\|y - y^{(i)}\|_A \leq 2\|y - y^{(0)}\|_A \left(\frac{\sqrt{\kappa(MA)} - 1}{\sqrt{\kappa(MA)} + 1}\right)^k . \tag{4.14}$$

Starting Vectors

All iterative solution methods to solve $Ay = b$ start with a given vector $y^{(0)}$. Choosing a good starting vector can decrease the number of iterates that are needed. The choice you make is dependent on the problem and the information you have. If you do not have any information then one typically starts with $y^{(0)} = \mathbf{0}$. In case of a nonlinear problem, the solution is approximated by the solution of a number of linear systems. One can use the solution of a given outer iteration as a starting vector for the next iterative method used to solve the next linear system.

Sometimes starting vectors can also be obtained by the solution of a related problem. One can use the analytic solution of a simplified problem or a solution which is obtained by using a coarser grid.

The better the starting vector approximates the solution, the lesser iterations are needed.

Stop Criterion

It is also very important to define a good criteria to stop. If the criterion is too weak, then the approximate solution is useless. But if the criterion is too severe, then the iterative solution method might never stop or simply costs too much work.

An iterative method is linearly convergent if it satisfies:

$$\|y^{(k)} - y^{(k-1)}\|_2 \approx \bar{\rho}\|y^{(k-2)} - y^{(k-1)}\|_2, \;\; \bar{\rho} < 1,$$

and $y^{(k)} \to A^{-1}b$ for $k \to \infty$. This criterion is easily checked during the iteration. Initially, this relation will not hold, but after some iterations the quantity $\frac{\|y^{(k)} - y^{(k-1)}\|_2}{\|y^{(k-2)} - y^{(k-1)}\|_2}$ will converge to $\bar{\rho}$. Theorem 4.1 from [8] states that we have the following inequality:

$$\|y - y^{(k)}\|_2 \leq \frac{\bar{\rho}}{1-\bar{\rho}}\|y^{(k)} - y^{(k-1)}\|_2.$$

This result can be used to define the following stopping criterion for linear convergent methods:

$$\text{Stop if: } \frac{\bar{\rho}}{1-\bar{\rho}} \frac{\|y^{(k)} - y^{(k-1)}\|_2}{\|y^{(k)}\|_2} \leq \epsilon.$$

If this condition holds, then you can show that the relative error is smaller than ϵ:

$$\frac{\|y - y^{(k)}\|_2}{\|y\|_2} \cong \frac{\|y - y^{(k)}\|_2}{\|y^{(k)}\|_2} \leq \frac{\bar{\rho}}{1-\bar{\rho}} \frac{\|y^{(k)} - y^{(k-1)}\|_2}{\|y^{(k)}\|_2} \leq \epsilon.$$

For iterative method that have a different convergence behavior most stopping criteria are based on the norm of the residual. Now we list some possible criteria with some comments.

Criterion 1	$\|b - Ay^{(k)}\|_2 \leq \epsilon$
Criterion 2	$\frac{\|b-Ay^{(k)}\|_2}{\|b-Ay^{(0)}\|_2} \leq \epsilon$
Criterion 3	$\frac{\|b-Ay^{(k)}\|_2}{\|b\|_2} \leq \epsilon$
Criterion 4	$\frac{\|b-Ay^{(k)}\|_2}{\|y^{(k)}\|_2} \leq \epsilon/\|A^{-1}\|_2$

Criterion 1 is not scaling invariant. If $\|b - Ay^{(k)}\|_2 \leq \epsilon$, then it does not hold for $\|100(b-Ay^{(k)})\|_2$, while the accuracy of $y^{(k)}$ remains the same. In criterion 2 the number of iterates is now dependent on the initial estimate $y^{(0)}$. So a better initial estimate will not lead to a decrease in the number of iterations. If the initial estimate is very good, then it is possible that the method never stops due to round-off errors. The third criterion is a good one. The norm of the residual should be small with respect to the norm of the right-hand side. If we replace ϵ by $\epsilon/\kappa_2(A)$, then we can show that the error in y is less than ϵ, where $\kappa_2(A)$ is the condition number: $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2$. In general $\|A\|_2$ and $\|A^{-1}\|_2$ are not known, but some iterative methods give approximations of these quantities. Criterion 4 is related to criterion 3 and in many cases this criterion implies that the relative error is smaller that ϵ.

It is also possible that a stopping criterion come from physical relations. This can happen when the residuals have a physical meaning.

Chapter 5

Deflation Techniques

It is known that the rate of convergence is dependent on the condition number of A. The condition number is defined as λ_n/λ_1. In order to improve convergence we can try to increase the smallest eigenvalue or to decrease the biggest. In [14] it is proven that if we have large contrasts in our coefficients, for example due to a fault in the subsoil, that we have some extreme small eigenvalues. The idea of deflation now is to define a deflation subspace, which approximates the eigenspace belonging to those small eigenvalues. Then we can project these eigenvalues out of the residual in order to improve convergence.

We start by explaining the basic idea of deflation in section 5.1. In section 5.2 and 5.3 we list some known possible deflation techniques. This chapter is mostly based on [3, 11, 12, 14].

5.1 Basic Idea of Deflation

Again we look at a general linear system of the form

$$Ay = b, \tag{5.1}$$

where we assume A to be symmetric positive definite. Let $Z \in \mathbb{R}^{n\times m}$ be a matrix where $m \leq n$. Furthermore we assume that the rank of Z is m, i.e. Z has linear independent columns. We call the columns of Z the *deflation vectors* and they span the deflation subspace. The deflation subspace is the space that approximates the eigenspace belonging to the smallest eigenvalues and which is to be projected out of the residual. For this purpose we define the projector

$$P \;=\; I - AZE^{-1}Z^T, \tag{5.2}$$

where we define the $m \times m$ matrix $E = Z^TAZ$. Note that we have the following expressions:

$$E^{-T} = (Z^TAZ)^{-T} = (Z^TA^TZ^{TT})^{-1} = (Z^TAZ)^{-1} = E^{-1},$$
$$P^T = (I - AZE^{-1}Z^T)^T = I^T - Z^{TT}E^{-T}Z^TA^T = I - ZE^{-1}Z^TA.$$

If we now want to solve system (5.1) using deflation, we can write $y = (I - P^T)y + P^Ty$. Since we have:

$$(I - P^T)y = Z(Z^TAZ)^{-1}Z^TAy = ZE^{-1}Z^Tb, \tag{5.3}$$

which can be computed immediately, we only need to compute P^Ty. Because we have that $AP^T = PA$ we can solve the system

$$PA\tilde{y} = Pb, \tag{5.4}$$

for $\tilde{y}$, and simply add $P^T\tilde{y}$ to Equation (5.3). The system (5.4) is called the *deflated system*.

Note that we have

$$\begin{aligned} PAZ &= (I - AZE^{-1}Z^T)AZ, \\ &= AZ - AZE^{-1}Z^TAZ, \\ &= AZ - AZE^{-1}E, \\ &= AZ - AZ, \\ &= 0, \end{aligned}$$

and therefore the system is singular, so there is no unique solution. However it can be shown that P^Ty is unique. A proof for the case that A is symmetric positive definite can be found in [14].

Since we want to use an iterative solution method we need an initial guess $\tilde{y}^{(0)}$. When $y^{(0)}$ is given we solve:

$$Au = f,$$

where $u \equiv y - y^{(0)}$ and $f \equiv b - Ay^{(0)}$. To do this we solve the deflated system

$$PA\tilde{u} = Pf,$$

for $\tilde{u}$ and we construct:

$$y = \left(I - P^T\right)u + P^T\tilde{u} + y^{(0)}$$

5.1.1 Preconditioning, Starting Vectors and Deflated CG method

It is also possible to combine deflation with preconditioning. If M is a suitable preconditioner of A, then Equation (5.4) can be replaced by:

$$M^{-1}PA\tilde{y} = M^{-1}Pb,$$

and we have to solve $\tilde{y}$ from it. After that we have to form $Q\tilde{y}$ again. This is called left preconditioning. We can also use right preconditioning, then we must solve $\tilde{y}$ from

$$PAM^{-1}\tilde{y} = Pb$$

and form $QM^{-1}\tilde{y}$.

Both systems can be solved by a Krylov subspace method. Since A is a symmetric positive definite matrix we can use for example a CG method. If we use deflation we have to determine expressions as $f = E^{-1}g$. To do this efficiently we solve the system $Ef = g$ instead of calculating E^{-1} directly. To solve such a system we use an LL^T factorization which involves the following steps:

- compute $E = LL^T$,
- solve $Lq = g$,
- solve $L^T f = q$.

The algoritms for evaluating these steps are given in appendix A, by respectively Algorithms A.1, A.2 and A.3.

Furthermore, for deflation to work correctly we must solve the system involving E with high accuracy. We can use preconditioning, for example, an modified incomplete Cholesky decomposition as given in Algorithm 5.1. The algorithm is written such that it follows closely the implementation of [6]. The method is called the *Deflated Incomplete Cholesky Conjugate Gradient Method*, or simply DICCG. The succes of the DICCG method is related to the choices of Z.

Algorithm 5.1 (DICCG) *Given an $n \times n$ symmetric positive definite matrix A, a vector b, the modified Cholesky preconditioner $LD^{-1}L^T$ where L is lower diagonal and D^{-1} the inverse diagonal of L, the projector P as in Equation(5.2), and an initial guess $y^{(0)}$. This algorithm solves the linear system $Ay = b$.*

```
$r^{(0)} = b - Ay^{(0)}$;
if deflation do                                   /* deflation pre-processing phase
    $\tilde{y}^{(0)} = y^{(0)}$;
    $y^{(0)} = 0$;
    Decompose $Z^T AZ = \tilde{L}\tilde{U}$
    Solve $\tilde{U}\tilde{q}_1 = Z^T r^{(0)}$; $\tilde{L}q_1 = \tilde{q}_1$
    $r^{(0)} := r^{(0)} - AZq_1$
end if
for $k = 0, 1, \ldots,$ convergence do
    Solve $L\tilde{s} = r^{(k)}$; $D^{-1}L^T s^{(k)} = \tilde{s}$;
    if $k = 0$ do
        $p^{(0)} = s^{(0)}$;
    else
        $\beta = \frac{(s^{(k)})^T r^{(k)}}{(r^{(k-1)})^T s^{(k-1)}}$;
        $p^{(k)} = s^{(k)} + \beta p^{(k-1)}$;
    end if
    $v^{(k)} = Ap^{(k)}$;
    if deflation do                               /* deflation run-time phase
        Solve $\tilde{U}\tilde{q}_2 = Z^T v^{(k)}$; $\tilde{L}q_2 = \tilde{q}_2$;
        $v^{(k)} := v^{(k)} - AZq_2$;
    end if
    $\alpha = \frac{(r^{(k)})^T s^{(k)}}{(p^{(k)})^T v^{(k)}}$;
    $y^{(k+1)} = y^{(k)} + \alpha p^{(k)}$;
    $r^{(k+1)} = r^{(k)} - \alpha v^{(k)}$;
end for
if deflation do                                   /* deflation post-processing phase
    Solve $\tilde{U}\tilde{q}_2 = Z^T Ay^{(k+1)}$; $\tilde{L}q_2 = \tilde{q}_2$;
    $y^{(k+1)} := y^{(k)} + \tilde{y}^{(0)} + Z(q_1 - q_2)$;
end if
```

5.2 Eigenvalue Deflation

Eigenvalue deflation means that we take exact or perturbated eigenvectors, corresponding to the small eigenvalues, to build our deflation matrix Z. First we look at deflation with exact eigenvectors and after that we look at perturbated eigenvectors.

5.2.1 Deflation with Exact Eigenvectors

We order the eigenvalues of A such that $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and we choose their corresponding eigenvectors $\mathbf{v}_j$ such that $\mathbf{v}_j^T \mathbf{v}_i = \delta_{ij}$. Furthermore we define $Z := [\mathbf{v}_1 \mathbf{v}_2 \ldots \mathbf{v}_m]$. We will show that the spectrum of PA satisfies:

$$\sigma(PA) = \{0, \ldots, 0, \lambda_{m+1}, \ldots, \lambda_n\}$$

Note that we can denote AZ as:

$$AZ = [A\mathbf{v}_1 \; A\mathbf{v}_2 \; \cdots \; A\mathbf{v}_n] = [\lambda_1\mathbf{v}_1 \; \lambda_2\mathbf{v}_2 \; \cdots \; \lambda_n\mathbf{v}_n],$$

so that we have:

$$\begin{aligned} E = Z^TAZ &= \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_m^T \end{bmatrix} \begin{bmatrix} \lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \cdots & \lambda_m\mathbf{v}_m \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1\mathbf{v}_1^T\mathbf{v}_1 & \lambda_2\mathbf{v}_1^T\mathbf{v}_2 & \cdots & \lambda_m\mathbf{v}_1^T\mathbf{v}_m \\ \lambda_1\mathbf{v}_2^T\mathbf{v}_1 & \lambda_2\mathbf{v}_2^T\mathbf{v}_2 & \cdots & \lambda_m\mathbf{v}_2^T\mathbf{v}_m \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1\mathbf{v}_m^T\mathbf{v}_1 & \lambda_2\mathbf{v}_m^T\mathbf{v}_2 & \cdots & \lambda_m\mathbf{v}_m^T\mathbf{v}_m \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 & & & \emptyset \\ & \lambda_2 & & \\ & & \ddots & \\ \emptyset & & & \lambda_m \end{bmatrix} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m). \end{aligned}$$

Now look at the following:

$$E^{-1}Z^T = \begin{bmatrix} \frac{1}{\lambda_1} & & \emptyset \\ & \ddots & \\ \emptyset & & \frac{1}{\lambda_m} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_m^T \end{bmatrix} = \begin{bmatrix} \frac{1}{\lambda_1}\mathbf{v}_1^T \\ \vdots \\ \frac{1}{\lambda_m}\mathbf{v}_m^T \end{bmatrix},$$

$$AZE^{-1}Z^T = \begin{bmatrix} \lambda_1\mathbf{v}_1 & \cdots & \lambda_m\mathbf{v}_m \end{bmatrix} \begin{bmatrix} \frac{1}{\lambda_1}\mathbf{v}_1^T \\ \vdots \\ \frac{1}{\lambda_m}\mathbf{v}_m^T \end{bmatrix} = \mathbf{v}_1\mathbf{v}_1^T + \ldots + \mathbf{v}_m\mathbf{v}_m^T, \tag{5.5}$$

$$ZZ^T = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_m \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_m^T \end{bmatrix} = \mathbf{v}_1\mathbf{v}_1^T + \ldots + \mathbf{v}_m\mathbf{v}_m^T. \tag{5.6}$$

From Equation (5.5) and Equation (5.6) we can conclude that $AZE^{-1}Z^T = ZZ^T$ and therefore it holds that:

$$P = I - AZE^{-1}Z^T = I - ZZ^T, \tag{5.7}$$

and we can write:

$$PA\mathbf{v}_j = \left(I - ZZ^T\right)\lambda_j \mathbf{v}_j. \tag{5.8}$$

If we now denote Z as:

$$Z = \begin{bmatrix} v_1^1 & v_2^1 & \cdots & v_m^1 \\ v_1^2 & v_2^2 & \cdots & v_m^2 \\ \vdots & \vdots & \ddots & \vdots \\ v_1^n & v_2^n & \cdots & v_m^n \end{bmatrix},$$

then:

$$ZZ^T = \begin{bmatrix} (v_1^1 v_1^1 + \ldots + v_m^1 v_m^1) & (v_1^1 v_1^2 + \ldots + v_m^1 v_m^2) & \cdots & (v_1^1 v_1^n + \ldots + v_m^1 v_m^n) \\ (v_1^2 v_1^1 + \ldots + v_m^2 v_m^1) & (v_1^2 v_1^2 + \ldots + v_m^2 v_m^2) & \cdots & (v_1^2 v_1^n + \ldots + v_m^2 v_m^n) \\ \vdots & \vdots & \ddots & \vdots \\ (v_1^n v_1^1 + \ldots + v_m^n v_m^1) & (v_1^n v_1^2 + \ldots + v_m^n v_m^2) & \cdots & (v_1^n v_1^n + \ldots + v_m^n v_m^n) \end{bmatrix}.$$

Now we are going to calculate $ZZ^T\mathbf{v}_j$ for $j = 1, \ldots, m, m+1, \ldots, n$. Herefore we look at the k^{th} column. That column is given by:

$$\left\{v_1^k v_1^1 v_j^1 + \ldots + v_m^k v_m^1 v_j^1\right\} + \left\{v_1^k v_1^2 v_j^2 + \ldots + v_m^k v_m^2 v_j^2\right\} + \ldots + \left\{v_1^k v_1^n v_j^n + \ldots + v_m^k v_m^n v_j^n\right\},$$

which can be rewritten as:

$$\begin{aligned} & \left\{v_1^k v_1^1 v_j^1 + v_1^k v_1^2 v_j^2 + \ldots + v_1^k v_1^n v_j^n\right\} + \left\{v_2^k v_2^1 v_j^1 + v_2^k v_2^2 v_j^2 + \ldots + v_2^k v_2^n v_j^n\right\} + \\ & \quad \cdots + \left\{v_m^k v_m^1 v_j^1 + v_m^k v_m^2 v_j^2 + \ldots + v_m^k v_m^n v_j^n\right\} \\ = \; & v_1^k \left(\mathbf{v}_1^T \mathbf{v}_j\right) + v_2^k \left(\mathbf{v}_2^T \mathbf{v}_j\right) + \cdots + v_m^k \left(\mathbf{v}_m^T \mathbf{v}_j\right). \\ = \; & \begin{cases} v_j^k & \text{if} \quad j = 1, 2, \ldots, m \\ 0 & \text{if} \quad j = m+1, \ldots, n \end{cases} \end{aligned}$$

where the last step is a consequence of $\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$. So we can draw the following conclusion:

$$ZZ^T\mathbf{v}_j = \begin{cases} \mathbf{v}_j & \text{if} \quad j = 1, 2, \ldots, m \\ 0 & \text{if} \quad j = m+1, \ldots, n \end{cases} \tag{5.9}$$

If we now substitute expression (5.9) into Equation (5.8) we can conclude that:

$$PA\mathbf{v}_j = \begin{cases} \mathbf{0} & \text{if} \quad j = 1, 2, \ldots, m \\ \lambda_j \mathbf{v}_j & \text{if} \quad j = m+1, \ldots, n \end{cases}$$

This shows that eigenvector deflation cancels the smallest m eigenvalues of the spectrum of A, leaving the rest of the spectrum untouched. Although deflation with exact eigenvectors leads to fast convergence, such a choice will be inefficient in practice. This is because it is relatively expensive to compute and use them.

5.2.2 Deflation with Inexact Eigenvectors

It is also possible to use perturbated or approximated eigenvectors, such as Ritz vectors (see, for example, [4] or [8]). This means that we define $\bar{Z} := [\bar{\mathbf{v}}_1\bar{\mathbf{v}}_2\ldots\bar{\mathbf{v}}_m]$, where each $\bar{\mathbf{v}}_i$ is an approximation of the exact eigenvector $\bar{\mathbf{v}}_i$. So:

$$\bar{\mathbf{v}}_i := \mathbf{v}_i + \delta_i, \quad \delta_i \in \mathbb{R}^n$$

It would be convenient that deflation with $\bar{V}$ has the same favorable features as deflation with V. In [11] the author states that, as far as he knows, in the literature no results are given on how to choose the $\bar{\delta}_i$ such that $\bar{V}$ has such features.

5.3 A Priori Deflation Vectors

Since it is expensive to calculate eigenvalues and eigenvectors of a (large) matrix we have to find other types of a priori deflation vectors. We list some possibilities.

5.3.1 Subdomain Deflation

In section 5.2 it is mentioned that deflation of an eigenspace cancels the corresponding eigenvalues without affecting the rest of the spectrum. Calculating the eigenvectors and eigenvalues of a (large) system is expensive and therefore not feasible. This motivates to deflate with "nearly invariant" subspaces obtained during the iteration. We can make a specific choice for the subspace Z, based on the decomposition of the domain Ω into m disjunct sets Ω_j such that $\bigcup_{j=1}^{m}\Omega_j = \Omega$. The division in subdomains can be motivated, for example, by jumps in the coefficients.

Since MODFLOW uses a cell-centered grid we look at possible deflation techniques for this discretization. Since in a cell-centered grid the unknowns are located in the interior of a finite volume, domain decomposition is straightforward. The algebraic deflation vectors z_j are uniquely defined as:

$$z_j(\mathbf{x}_i) = \begin{cases} 1, & \text{for} \quad \mathbf{x}_i \in \Omega_j \\ 0, & \text{for} \quad \mathbf{x}_i \in \Omega\backslash\Omega_j. \end{cases}$$

So now the question remains how we are going to choose the domains. If we do not have any information we can just cut the domain into squares (2D) or cubes (3D) and make the corresponding deflation vectors, see also Figure 5.1. This method is straight forward to implement on a computer.

Note that is is also possible to use linear vectors instead of constant vectors. See also Figure 5.2. The red and the blue one are constant vectors, where the green and the magenta ones are linear.

5.3.2 Deflation based on Physics

In our case it is possible to define the subdomain by using the geographical places of the faults. Since MODFLOW defines the fault to be exactly on the border of a cell, theoretically this is possible. For this kind of deflation we have two options.

The first option is to divide the domain into subdomains as in subsection 5.3.1, but now using the faults as boundaries. A second option could be to define the vectors such that an element next to the fault has value 1 and the rest value 0. We can make for each fault a vector. But we have to think about what to do with the elements that are next to two or more faults.

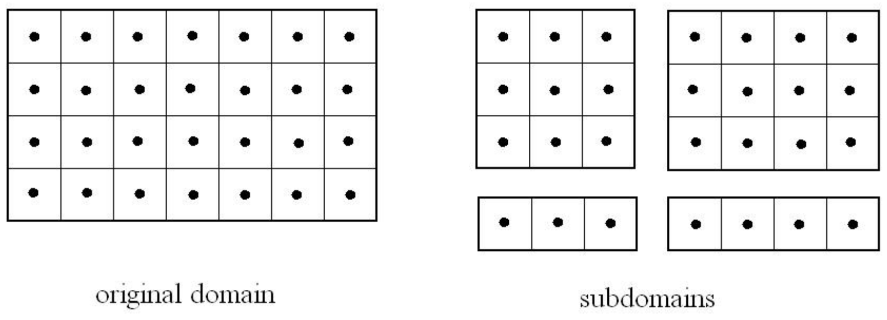

Figure 5.1: Random decomposition for a cell centered discretization.

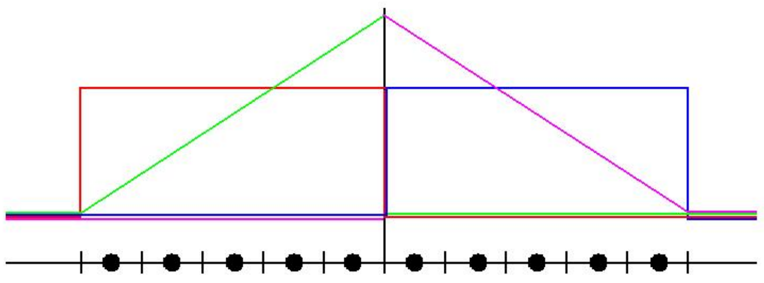

Figure 5.2: 4 possible deflation vectors in a 1-D case.

Chapter 6

Testproblems, First Results and Research Questions

In this chapter results are given for several test problems in MATLAB. In Section 6.1 we explain the most simple test case. This one is mainly to give us insight in the algorithms that are being used. In Section 6.2 the test problem is described that is exported from MODFLOW to MATLAB, Section 6.3 gives the results. In Section 6.4 a research proposal is given for the next six months.

All the figures belonging to this chapter are in Appendix C to keep this chapter conveniently arranged.

6.1 First Test Problem

The first test case consists of a rectangular area containing of 15 rows, 15 colums and only one layer. On the east side of the domain we defined constant head cells with $h = 0$. On the other boundaries we defined the so called no-flow cells. There also is a recharge on the whole area of $7 \cdot 10^{-4}$ meters per day. To make it a realistic problem we also added drains to the whole domain. The last thing we added was a horizontal flow barrier to simulate the faults. This barrier is located vertically on one thirth of the domain. So in this case it was located between the fifth and sixth column. The conductance of the barrier was set at 10^{-9}.

We build this testcase in MODFLOW to see what the outcome was. We defined the cells to be 25 meters times 25 meters. This is the same scaling as used in the IBRAHYM model. MODFLOW gives for this testcase the solution as showed in Figure C.1. It is clear that the barrier causes a big jump in the calculated heads. This is due to the big contrast in the conductance. On the left part of the domain we see heads around the 44 meters and at the right side of the fault we see heads between 0.1 and 0 meters.

6.2 MATLAB

The first goal was to make sure we could use exactly the system of equations as used in MODFLOW in Matlab. During the run in MODFLOW we made sure that MODFLOW gave all the coefficients needed to rebuild our system of equations as given in Equation (3.18). With the coefficients CC, CR, CV, HCOF and RHS we were able to rebuild the system.

Now that we have the right system implemented in Matlab we have to rebuild the PCG solver in Matlab. This algorithm can be found in Algorithm 4.1. In the MATLAB code you are able to choose between no preconditioning or the ICCG preconditioning as given in Section 4.4.2. After we have a good PCG solver we want to build in deflation also. We use the most common deflation vectors, namely the vectors which divide the area into two parts. The first domain is left of the fault and the second is the domain right of the fault.

Our stop criterion is now based on the residual. The 2-norm of the residual divided by the 2-norm of the righthand side should be smaller than a given tolerance. MODFLOW uses a different stop criterion. Actually it uses two criteria. The maximum of the difference between two calculated heads should be small enough and the difference between two residuals should also be small enough. This still has to be implemented in our matlab program to match matlab with MODFLOW.

6.3 First Results

6.3.1 Results for our First Test Problem

For our simple test case as described in section 6.1 we found results that are expected. The values of the heads using matlab correspond to those calculated by MODFLOW. So this shows that the MATLAB code works correctly. If we now look at the number of iterates used by MATLAB for the two runs (using deflation and not using deflation) we already see an improvement. The results are given in Figure C.2.

We can also look at the spectrum of the matrices $M^{-1}A$ and $M^{-1}PA$, see Figure C.3. We see that the eigenvalues are almost the same, but if we zoom in to the smallest eigenvalue (Figure C.4), we see that by using deflation this one is zero. So the smallest eigenvalue has been cancelled and therefore the condition of the deflated matrix is better.

6.3.2 More Results

Enlarging the number of cells

After we had the first test results we decided to enlarge the problem. We now used the same setup but we used a bigger area. Instead of a 15 x 15 grid we used a 30 x 30 grid and a 45 x 45 grid. Also for this cases we found that the number of iterations drops when using deflation. The results are showed in Figure C.5 and Figure C.6. We see that the number of iterates grows as the number of cells grows.

Variation of the conductance of the fault

If we keep the dimensions at 45 x 45 and we only vary the conductance of the fault we see the following results. The solution changes, but the number of iterates does not. The solutions using a conductance of 10^{-3}, 10^{-6}, 10^{-9} and 0 are given in Figures C.7, C.9, C.11 and C.13 respectively. The number of iterates are given in Figures C.12, C.10, C.8 and C.14 respectively.

This is not in agreement with the problems they find at Deltares. If they make the conductance lower, than they see that they need more iterates. It could be that our test case is simply to small to see this problem.

6.4 Research Questions

This chapter shows that the deflation method has good potential to improve convergence behavior in MODFLOW, considering a simple flow model. During the following six months of the Masters project this method is being implemented in MODFLOW and suitable deflation vectors are tried to be chosen such that:

- the deflation vectors are easy to construct,
- a priori information is used to construct these vectors,
- the choice of vectors is generic.

Furthermore we do not want to use deflation if we do not have to. So we have to think of some kind of condition for the computer to decide if deflation is needed or not.

The goal of the research is to reduce the number of iterations used for the PCG solver and to gain wall-clock times. The resulting new algorithm will be tested for a selected part of the IBRAHYM model.

Appendix A

LU Factorization Algorithms

Algorithm A.1 (LU Decomposition) *Given an $n \times n$ matrix A, and the $n \times n$ matrices L and U, containing only zero coefficients. Then this algorithm computes an LU factorization of A.*

```
for $m = 1, \ldots, n-1$ do
    for $i = k+1, \ldots, n$ do
        $L(i,k) = \frac{A(i,k)}{A(k,k)}$;
    end for
    for $j = k+1, \ldots, n$ do
        for $i = k+1, \ldots, n$ do
            $U(i,j) = A(i,j) - L(i,j)A(k,j)$;
        end for
    end for
    $L(k,k) = 1$;
end for
$L(n,n) = 1$;
```

Algorithm A.2 (Forward Substitution) *Given an $n \times n$ matrix L which is unit lower triangular, and a vector $\boldsymbol{b}$. The this algorithm solves the system $L\boldsymbol{y} = \boldsymbol{b}$.*

```
for $i = 1, \ldots, n$ do
    for $j = 1, \ldots, i-1$ do
        $y(i) = b(i) - L(i,j)y(j)$;
    end for
end for
```

Algorithm A.3 (Backward Substitution) *Given an $n \times n$ matrix U which is upper triangular, and a vector $\boldsymbol{b}$. The this algorithm solves the system $U\boldsymbol{y} = \boldsymbol{b}$.*

```
for $i = 1, \ldots, n$ do
    for $j = i+1, \ldots, n$ do
        $y(i) = \frac{b(i) - U(i,j)y(j)}{U(i,i)}$;
    end for
end for
```

Appendix B

MICCG in MODFLOW

In MODFLOW a slightly different version of the MICCG method is used. They use a preconditioner $M = U^T DU$, where U is an upper triangular matrix. U has nonzero elements along the main diagonal and on the off-diagonal positions where A itself has nonzero values. D is a diagonal matrix with $d_{ii} = 1/u_{ii}$. For a problem with 2 rows, 3 colums and 2 layers, U should be of the following form:

$$\mathbf{U} = \begin{bmatrix} u_1 & -r_1 & 0 & -c_1 & 0 & 0 & -v_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & u_2 & -r_2 & 0 & -c_2 & 0 & 0 & -v_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & u_3 & -r_3 & 0 & -c_3 & 0 & 0 & -v_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_4 & -r_4 & 0 & -c_4 & 0 & 0 & -v_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & u_5 & -r_5 & 0 & -c_5 & 0 & 0 & -v_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & u_6 & -r_6 & 0 & -c_6 & 0 & 0 & -v_6 \\ 0 & 0 & 0 & 0 & 0 & 0 & u_7 & -r_7 & 0 & -c_7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_8 & -r_8 & 0 & -c_8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_9 & -r_9 & 0 & -c_9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{10} & -r_{10} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{11} & -r_{11} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{12} \end{bmatrix}, \tag{B.1}$$

Here, r_n, c_n and v_n are the conductances along rows and colums and between layers, respectively. They are the same as CR, CC and CV from section 3.1.2. They are exactly the same as in the matrix A.

The u_n are defined such that the sum of the elements along a row of M equals the sum of the elements along the row of A. If we do the calculations we find that:

$$\begin{aligned} u_1 &= a_{11}, \\ u_2 &= a_{22}\frac{r_1^2}{u_1} - \frac{r_1 c_1}{u_1} - \frac{r_1 v_1}{u_1}, \\ u_3 &= a_{33}\frac{r_2^2}{u_2} - \frac{r_2 c_2}{u_2} - \frac{r_2 v_2}{u_2}, \\ & \text{etc.} \end{aligned}$$

The general algorithm can now be expressed as:

$$u_{ii} = (1+\delta)\,a_{ii} - \sum_{k=1}^{i-1} \frac{u_{ki}^2}{u_{kk}} - \alpha \sum_{j=1}^{i-1} f_{ji} + \sum_{j=i+1}^{N} f_{ij} \tag{B.2}$$

where:

$$f_{ij} = \begin{cases} \sum_{k=1}^{i-1} u_{ki} u_{kj} u_{kk} & a_{ij} = 0 \\ 0 & a_{ij} \neq 0 \end{cases},$$

and

$$u_{ij} = 0 \text{ for } j < i.$$

The parameter α is a relaxation parameter, defined by the user. It is used to deminish the value of f_{ij} in equation B.2. Using a relaxation parameter value of 0.97, 0.98 or 0.99 instead of 1.00 sometimes improves convergence.

The parameter δ is a overcompensation parameter. It usually equals zero, but when a value of u_{ii} gets zero or negative, δ is increased.

More details can be found in [6]

Appendix C

Figures for Chapter 6: Testproblems, First Results and Research Questions

C.1 Figures for Section 6.1: First Testproblem

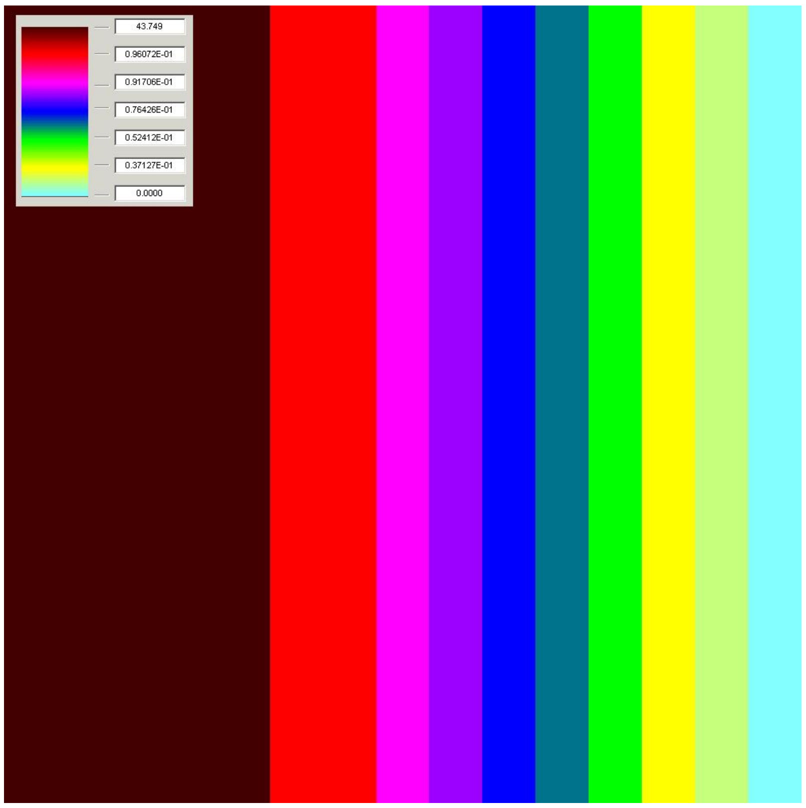

Figure C.1: Heads for first testproblem as generated by MODFLOW.

C.2 Figures for Section 6.2: Matlab

C.3 Figures for Section 6.3: First Results

C.3.1 Results for First Testproblem

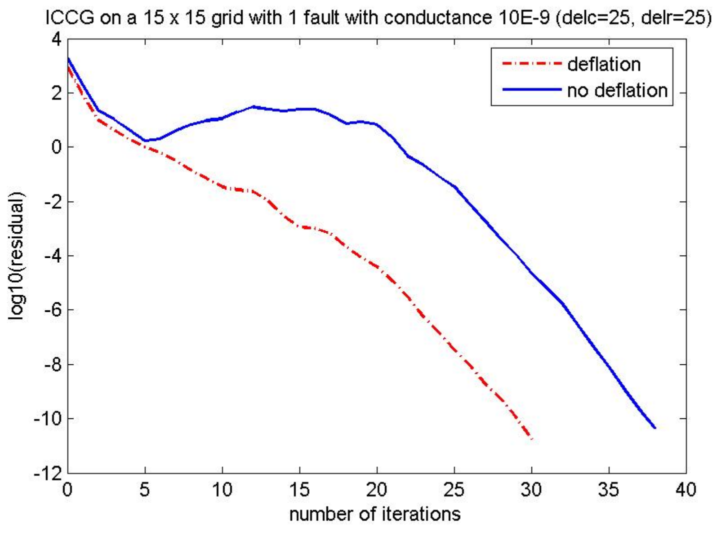

Figure C.2: The residuals of the simple testcase.

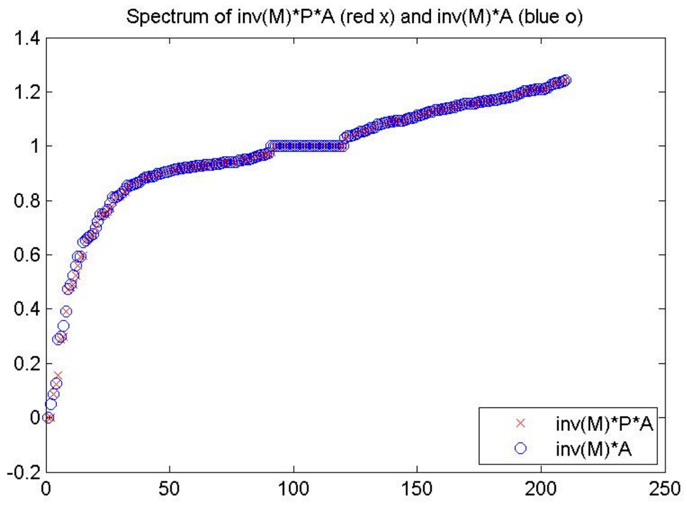

Figure C.3: The eigenvalues of the simple testcase.

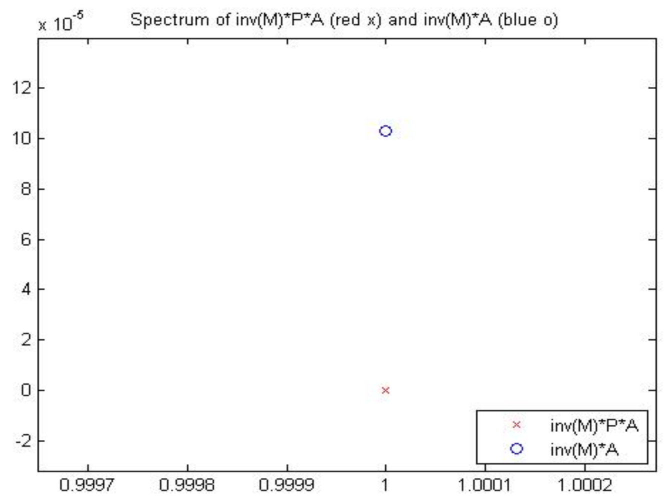

Figure C.4: The smallest eigenvalue is set to zero by using deflation.

C.3.2 More Results

Enlarging the number of cells

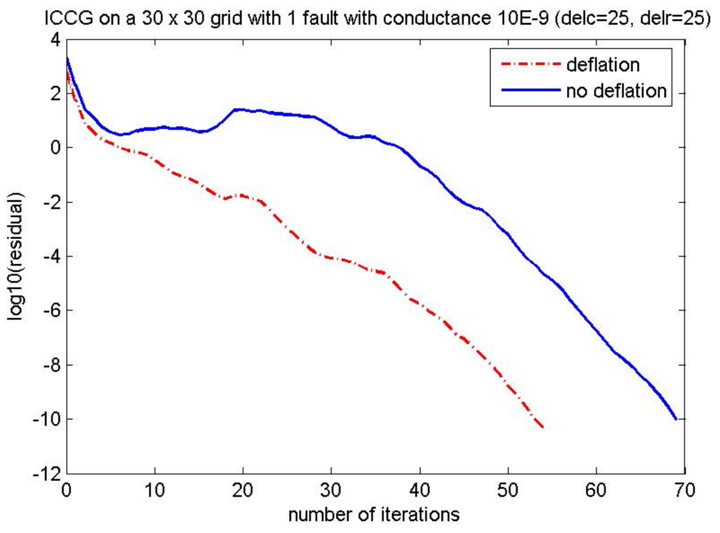

Figure C.5: Number of iterates on a 30x30 grid.

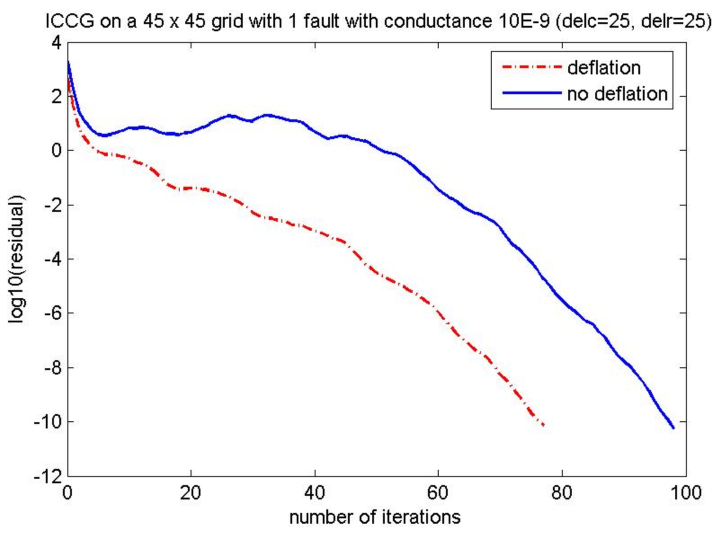

Figure C.6: Number of iterates on a 45x45 grid.

Variation of the conductance of the fault

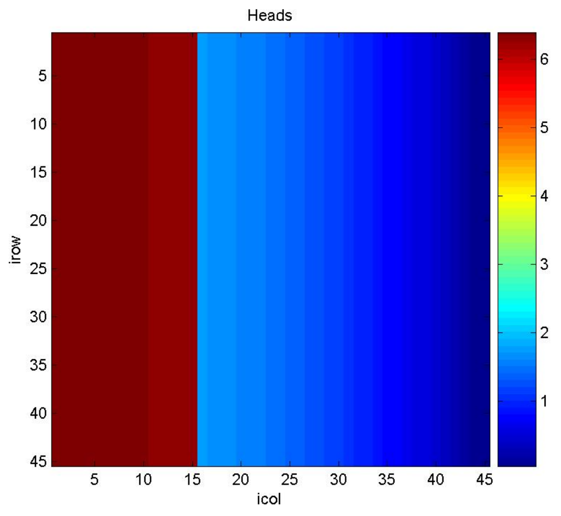

Figure C.7: Solution with conductance of 10E-3.

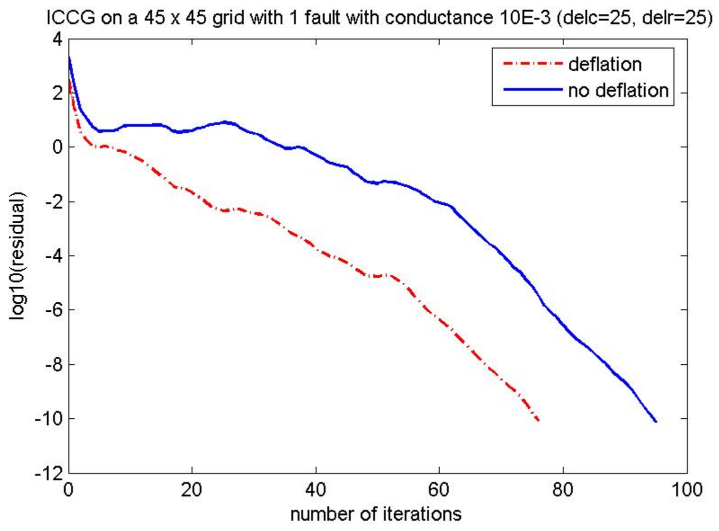

Figure C.8: Residuals with conductance of 10E-3.

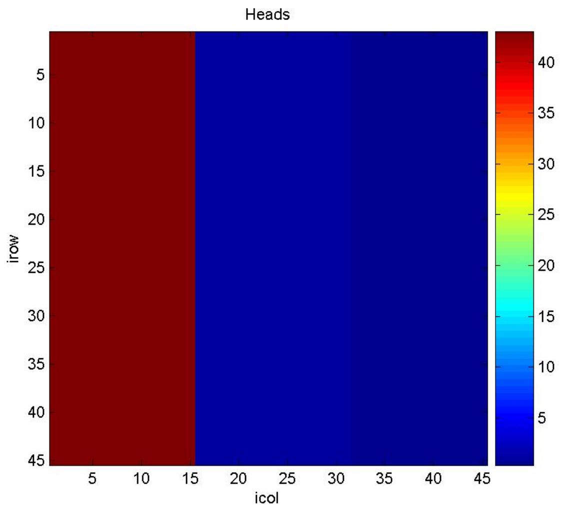

Figure C.9: Solution with conductance of 10E-6.

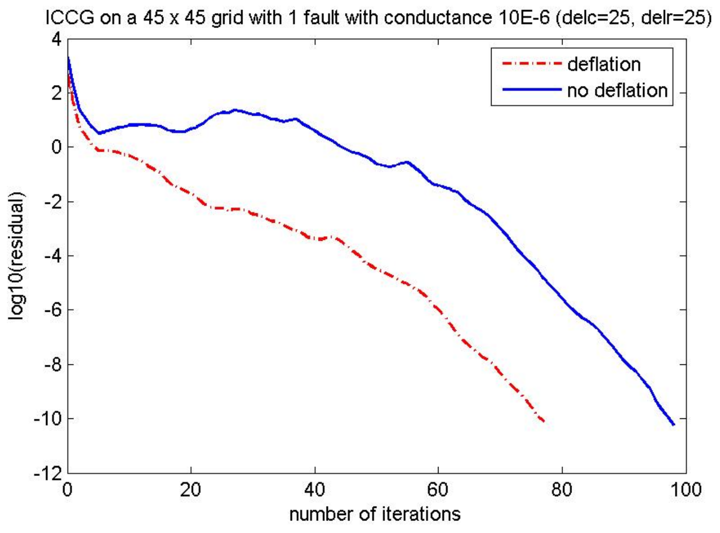

Figure C.10: Residuals with conductance of 10E-6.

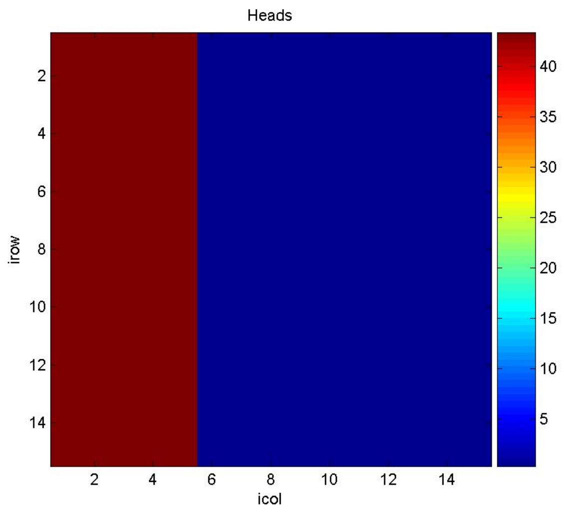

Figure C.11: Solution with conductance of 10E-9.

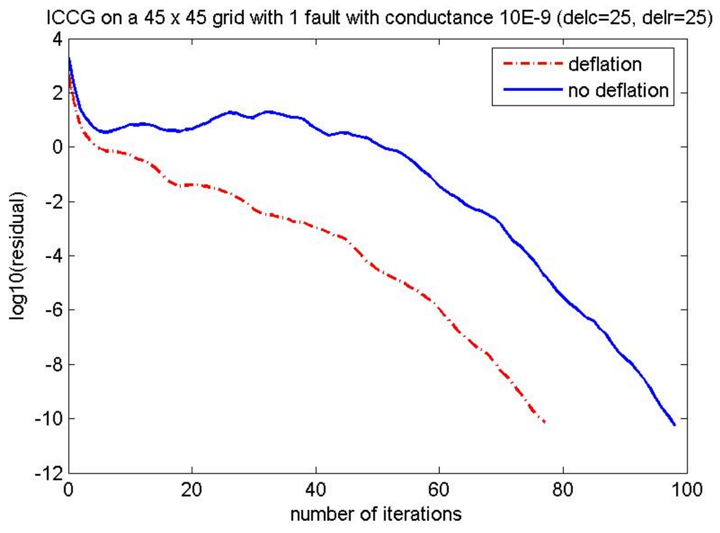

Figure C.12: Residuals with conductance of 10E-9.

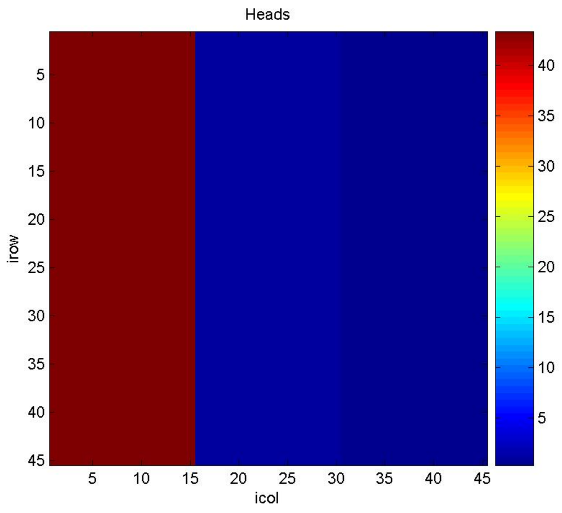

Figure C.13: Solution with conductance of 0.

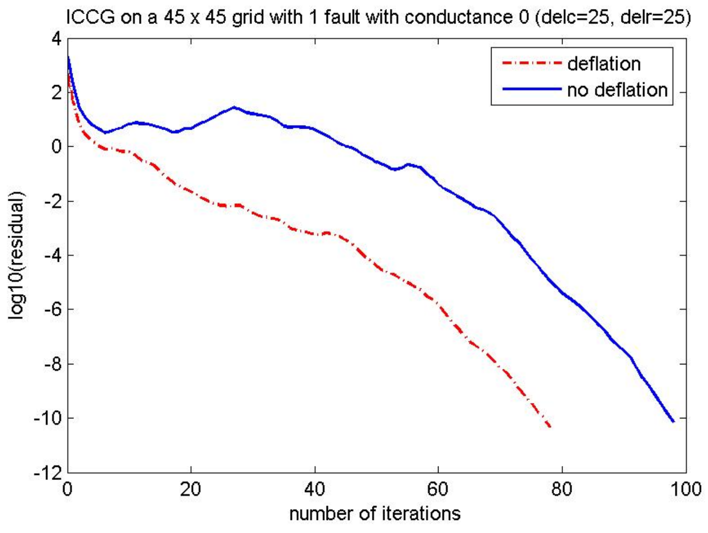

Figure C.14: Residuals with conductance of 0.

Bibliography

[1] Akker, C. van der, en Savenije, H. (2006), *'Hydrologie'*, dictaat Civiele techniek TU Delft (CT1310).

[2] Dufour, F.C. (1998), *'Grondwater in Nederland'*, ISBN: 90 6743 536 8, Van de Rhee, Rotterdam.

[3] J. Frank and C. Vuik (2001), *On the Construction of Deflation-Based Preconditioners*. SIAM J. Sci. Comput., Vol. 23, No. 2, pp. 442-462.

[4] Golub, Gene H. & Van Loan, Charles F. (1996), *'Matrix Computations'*, ISBN: 0-8018-5414-8, Baltimore : Johns Hopkins University Press.

[5] Harbaugh, A.W., *'MODFLOW 2005, the U.S. Geological Survey Modular Ground-Water Model - The Ground Water Flow Process'*. Online document: `http://pubs.usgs.gov/tm/2005/tm6A16/PDF/TM6A16.pdf`.

[6] Hill, M.C., *Preconditioned Conjugate-Gradient 2 (PCG2), 'A Computer Program for Solving Ground-Water Flow Equations'*. United States Geological Survey, Report 90-4048. Online document: `http://water.usgs.gov/nrp/gwsoftware/modflow2000/PCG-usgs-wrir_90-4048-\second_printing.pdf`.

[7] Nabben, R. and Vuik, C. (2004), *'A comparison of abstract versions of deflation, balancing and additive coarse grid correction preconditioners'*, SIAM J. Numer. Anal., 42, pp. 1631-1647.

[8] Oosterlee, C.W. and Vuik, C. (2005), *'Scientific Computing'*, lecture notes Applied Mathematics TU Delft (wi4201).

[9] Rushton, K.R. (1979), *'Seepage and Groundwater Flow'*, ISBN: 0 471 99754 4, Wiley, Chichester.

[10] Saad, Y. (2000), *'Iterative Methods for Sparse linear Systems' (First edition)*. Online: `http://www-users.cs.umn.edu/~saad/books.html`.

[11] Tang, J.M. and Vuik, C. (2007), *New Variants of Deflation Techniques for Pressure Correction in Bubbly Flow Problems*, Journal of Numerical Analysis, Industrial and Applied Mathematics.

[12] Verkaik, J. (2003), *De flated Krylov-Schwarz Domain Decomposition for the Incompressible Navier-Stokes Equations on a Colocated Grid*, Master's Thesis.

[13] Vermeulen, Peter (2006), *'Model-Reduced Inverse Modeling'*, Ph.D. thesis Delft University of Technology, Printpartners IPSKAMP B.V. Enschede.

[14] Vermolen, F.J. and C. Vuik (2001), *The influence of deflation vectors at interfaces on the deflated conjugated gradient method*. Report of the Department of Applied Mathematical Analysis, ISSN 1389-6520, Delft University of Technology.

[15] `http://water.usgs.gov/`.

[16] `http://www.gly.uga.edu/railsback/1121SimpleFaults.jpeg`.

[17] `http://upload.wikimedia.org/wikipedia/commons/thumb/6/68/Le_Peelhorst,_le_Peelrandbreuk_(faille)_et_le_Slenk_Central.jpg/250px-Le_Peelhorst,_le_Peelrandbreuk_(faille)_et_le_Slenk_Central.jpg`.

[18] `http://www.tucson.ars.ag.gov/salsa/research/research_1997/AMS_Posters/gw-sw_interactions/gw-sw_f1.html`.

List of Figures