British Geological Survey

NATURAL ENVIRONMENT RESEARCH COUNCIL

User's manual for the particle tracking model ZOOPT

Groundwater Systems & Water Quality Programme

Internal Report IR/04/141

BRITISH GEOLOGICAL SURVEY

GROUNDWATER SYSTEMS & WATER QUALITY PROGRAMME

INTERNAL REPORT IR/04/141

User's manual for the particle tracking model ZOOPT

C.R. Jackson

Keywords

Particle tracking; ZOOPT; ZOOMQ3D.

Bibliographical reference

JACKSON, C.R. 2004. User's manual for the particle tracking model ZOOPT. *British Geological Survey Internal Report*, IR/04/141. 46pp.

Keyworth, Nottingham British Geological Survey 2004

BRITISH GEOLOGICAL SURVEY

The full range of Survey publications is available from the BGS Sales Desks at Nottingham, Edinburgh and London; see contact details below or shop online at www.geologyshop.com

The London Information Office also maintains a reference collection of BGS publications including maps for consultation.

The Survey publishes an annual catalogue of its maps and other publications; this catalogue is available from any of the BGS Sales Desks.

The British Geological Survey carries out the geological survey of Great Britain and Northern Ireland (the latter as an agency service for the government of Northern Ireland), and of the surrounding continental shelf, as well as its basic research projects. It also undertakes programmes of British technical aid in geology in developing countries as arranged by the Department for International Development and other agencies.

The British Geological Survey is a component body of the Natural Environment Research Council.

British Geological Survey offices

Keyworth, Nottingham NG12 5GG

☎ 0115-936 3241 Fax 0115-936 3488
e-mail: sales@bgs.ac.uk
www.bgs.ac.uk
Shop online at: www.geologyshop.com

Murchison House, West Mains Road, Edinburgh EH9 3LA

☎ 0131-667 1000 Fax 0131-668 2683
e-mail: scotsales@bgs.ac.uk

London Information Office at the Natural History Museum (Earth Galleries), Exhibition Road, South Kensington, London SW7 2DE

☎ 020-7589 4090 Fax 020-7584 8270
☎ 020-7942 5344/45 email: bgslondon@bgs.ac.uk

Forde House, Park Five Business Centre, Harrier Way, Sowton, Exeter, Devon EX2 7HU

☎ 01392-445271 Fax 01392-445371

Geological Survey of Northern Ireland, 20 College Gardens, Belfast BT9 6BS

☎ 028-9066 6595 Fax 028-9066 2835

Maclean Building, Crowmarsh Gifford, Wallingford, Oxfordshire OX10 8BB

☎ 01491-838800 Fax 01491-692345

Sophia House, 28 Cathedral Road, Cardiff, CF11 9LJ

☎ 029–2066 0147 Fax 029–2066 0159

Parent Body

Natural Environment Research Council, Polaris House, North Star Avenue, Swindon, Wiltshire SN2 1EU

☎ 01793-411500 Fax 01793-411501
www.nerc.ac.uk

Foreword

The development of the modelling software within the ZOOM family, of which ZOOPT is a part, has been undertaken through a continuing tripartite collaboration between the University of Birmingham, the Environment Agency and the British Geological Survey. The development of the flow model ZOOMQ3D was initially undertaken at the University of Birmingham between 1998 and 2001 but continued after this time as a collaborative project between the three partner organisations. Since the inception of the collaborative project, the development of the software has been directed by the ZOOM steering committee, the members of which are:

University of Birmingham

Dr Andrew Spink

Environment Agency

Steve Fletcher

Paul Hulme

British Geological Survey

Dr Denis Peach

Dr Andrew Hughes

Dr Chris Jackson

Acknowledgements

The author would like to acknowledge the assistance of A.G. Hughes and M.M. Mansour of the British Geological Survey and P.J. Hulme of the Environment Agency for their help in reviewing the ZOOPT software. Additionally, the author would like to acknowledge the assistance of A.T. Williams of the British Geological Survey for reviewing this document.

Preface to the second edition

The production of the second edition of the ZOOPT manual coincides with the release of version 1.03 of the code. This version of the code incorporates one only change to version 1.02.

DIFFERENCES BETWEEN VERSION 1.03 AND 1.02 OF ZOOPT

- All executables should now be placed in a suitable directory e.g. ‘c:\Program Files\ZOOM’ and this folder should be added to the Windows system PATH variable. ZOOPT can then be run from any working directory by typing the name of the executable followed by the path of the working directory e.g. ‘ZOOPT c:\myDirectory’. Alternatively, this string could be placed in a batch file and the batch file run from the command line.

Contents

FIGURES

TABLES

Summary

This report describes the development of a steady-state particle tracking code for use in conjunction with the object-oriented groundwater flow model, ZOOMQ3D (Jackson and Spink, 2004). Like the flow model, the particle tracking software, ZOOPT, is written using an object-oriented approach to promote its extensibility and flexibility.

ZOOPT enables the definition of steady-state and time-variant path lines in three dimensions. Particles can be tracked in both the forward and reverse directions in steady-state flow fields enabling the rapid definition of borehole catchments, recharge and discharge areas and the visualisation of groundwater flow fields, for example. The program also enables the visualisation of steady-state particle tracks that are based on the node-by-node flows at a specific instant of a time-variant simulation. For example, this capability allows the examination of the changing shape of an approximate borehole catchment over an annual recharge or abstraction cycle. Particles can currently only be tracked in the forward direction in dynamic, or time-variant, flow fields.

Path lines are defined using the semi-analytical method (Pollock, 1988), however, around particular model features the Runge-Kutta technique is implemented in order to solve some specific problems associated with particle tracking. The problem of particle termination at 'weak' sink nodes is solved by the application of the special velocity interpolation scheme presented by Zheng (1994). This approach enables the definition of borehole catchments around wells that induce weak sinks which is not possible with many other widely used particle tracking codes.

ZOOMQ3D incorporates the representation of the vertical variation of hydraulic conductivity with depth (VKD) within finite difference nodes. This has been implemented in the flow model to enable the more accurate description of the variation of hydraulic conductivity in limestone, and particularly Chalk aquifers. ZOOPT is fully compatible with VKD models.

ZOOMQ3D also enables the local refinement of the finite difference grid, for example, around pumping wells. Again, ZOOPT is fully compatible with this model feature and can be used to track particles through such refined meshes.

ZOOPT has been rigorously tested through its comparison with an analytical solution and another particle tracking code and through the inspection of path lines generated using numerous test models (Jackson, 2002b).

1 Introduction

ZOOPT is the particle tracking code associated with the groundwater flow model ZOOMQ3D. The program can track the advective movement of particles through steady-state flow fields in both the forward and backward directions. It can also be used to forward track particles through non-steady groundwater flow fields.

The particle tracking model is straightforward to run and only requires a few input files in addition to those required by the flow model, ZOOMQ3D. All input to ZOOPT is in the form of ASCII text files. ZOOPT produces ASCII text and dxf files as output. To run ZOOPT a subset of the ZOOMQ3D input files is required (Table 1), in addition to two of the output files produced by the flow model (Table 2). A further small number of input files are required which are specific to ZOOPT (Table 3). Detailed descriptions of the files listed in Table 1, which form input to both ZOOMQ3D and ZOOPT, are presented in the ZOOMQ3D manual (Jackson and Spink, 2004). Consequently, they are not described here.

This document describes the development of the particle tracking code, ZOOPT, which is based on the object-oriented groundwater flow model, ZOOMQ3D (Jackson and Spink, 2004). Particle tracking methods are described prior to the description of how to run the program.

1.1 TERMINOLOGY

ZOOPT is written using an object-oriented programming language. Whilst the users do not need to concern themselves with what this means, the term *object-oriented* is used within this manual and consequently, a brief explanation is required.

The object-oriented method is an approach to structuring and developing software applications. Instead of an application being based on a step-wise process, a set of objects are defined that exchange messages. The user of ZOOPT can think of an object in abstract terms as any distinct entity that stores data and performs tasks. In ZOOMQ3D and ZOOPT objects are defined to represent real world features. For example, a pumped well is represented by an object. Pumped wells are described by data such as a depth and radius, and have the capability to pump water out of an aquifer. References are made in this manual to *particles* and these are also represented by objects. Each particle is characterised by a position and can move by advection through the aquifer.

1.2 UNIT CONVENTION

All lengths in ZOOPT are specified in metres. The unit of time is specified as days.

Table 1 List of the ZOOMQ3D input files required by ZOOPT

1	anisotropy##.map & anisotropy##.cod	OR	anisotropy##.dat	per layer	☒
2	aquifer.map				☒
3	boundary.dat				☒
4	clock.dat				☒
5	entry_method.dat				☒
6	fixedheads.dat				
7	grids.dat				☒
8	hydcond##.map & hydcond##.cod	OR	hydcond##.dat	per layer	☒
9	leakage.dat				☒
10	noflow##.map			per layer	☒
11	pumping.dat				
12	recharge.dat				
13	recharge.cod & recharge.map	& / OR	recharge_rates.dat		
14	rivers.dat				☒
15	springs.dat				☒
16	vcond##.map & vcond##.cod	OR	vcond##.dat	per layer	☒
17	vkd.cod & vkd.map				☒
18	vkd.dat				☒
19	vkdkx##.map & vkdkx##.cod	OR	vkdkx##.dat	per vkd scheme	
20	vkdky##.map & vkdky##.cod	OR	vkdky##.dat	per vkd scheme	
21	vkdzp##.map & vkdzp##.cod	OR	vkdzp##.dat	per vkd scheme	
22	vkdgrad##.map & vkdgrad##.cod	OR	vkdgrad##.dat	per vkd scheme	
23	zbase##.map & zbase##.cod	OR	zbase##.dat	per layer	☒
24	zoomq3d.dat				☒
25	ztop##.map & ztop##.cod	OR	ztop##.dat	per layer	☒

☒ Note the names of these files are fixed. The names of the remaining files can be specified by the user in the input file 'zoomq3d.dat'

Table 2 List of the ZOOMQ3D output files required by ZOOPT

1	heads.txt	☒
2	flowbal.txt	☒

☒ Note the names of these files are fixed.

Table 3 List of the additional input files required by ZOOPT

1	zoopt.dat				☒
2	particles.dat				☒
3	porosity##.map & porosity##.cod	OR	porosity##.dat	per layer	☒
4	porosity##a.map & porosity##a.cod	OR	porosity##a.dat	per layer	☒

☒ Note the names of these files are fixed.

2 Particle tracking theory

Particle tracking is commonly used to define the path lines of solute particles under purely advective transport. The technique is often applied for the definition of borehole catchments and associated source protection zones, the identification of recharge and discharge areas and the visualisation of groundwater flow patterns. However, the method also forms the basis of a number of solute transport models, which simulate the effects of hydrodynamic dispersion. Random walk methods (Prickett et al., 1981; Farahmand-Razavi, 1995) and the method of characteristics (Konikow and Bredehoeft, 1978; Zheng, 1990) use particle tracking to describe the advective component of solute transport.

2.1 MATHEMATICAL BACKGROUND

Assuming that fluid density is uniform, the path lines of contaminants under advection alone are governed by the equation

$$\frac{dp}{dt} = v(p,t) \tag{2.1}$$

where

$p = x\,\mathbf{i} + y\,\mathbf{j} + z\,\mathbf{k}$ is the position vector and,

$v = v_x\,\mathbf{i} + v_y\,\mathbf{j} + v_z\,\mathbf{k}$ is the seepage velocity vector,

$v_x = \frac{K_x}{\theta}\frac{\partial h}{\partial x}$,

K_x is the hydraulic conductivity in the x-direction,

θ is the porosity and,

$h(x,y,z,t)$ is the groundwater head, which is a function of space and time.

The solution of Equation 2.1 for the position of a particle at time, t, is

$$p(t) = p(t_0) + \int_{t_0}^{t} v(p,t)dt \tag{2.2}$$

where

$p(t_0)$ is the initial position of the particle at time t_0.

The solution of Equation 2.2 requires the evaluation of the velocity field at any given time and position in the model domain. If an exact solution for the velocity field exists then Equation 2.2 can be solved analytically for p(t). However, this is generally not the case and Equation 2.2 must then be solved numerically. In a finite difference model velocity components are only known at specific locations in the aquifer, that is, at the position of the cell wall between two adjacent nodes. Consequently, an interpolation scheme must be used to evaluate the velocity field at arbitrary positions and times. This means that an analytical solution to Equation 2.2 cannot be calculated. Furthermore the selection of the interpolation method determines which numerical integration techniques can be used to define the path line. These considerations are discussed next.

2.2 VELOCITY INTERPOLATION

Different velocity interpolation methods have been used in particle tracking codes, of which a number are listed in Table 4. Each velocity interpolation scheme has its advantages and disadvantages, though the selection of a method is often based on the comparison between linear and multi-linear interpolation techniques. The benefit of using simple linear velocity interpolation in each co-ordinate direction is that the technique satisfies finite difference cell-by-cell mass balances (Goode, 1990) and preserves velocity discontinuities at cell boundaries in heterogeneous systems. A disadvantage of the method is that it can produce less realistic path lines in homogeneous aquifers when compared to higher order interpolation methods, such as bi-linear interpolation. However, a significant benefit of the use of linear velocity interpolation is that it allows Equation 2.2 to be solved using a *semi-analytical* method, which is computationally efficient. The efficiency of the method is discussed in Section 2.3.1.

Table 4 Velocity interpolation and solution methods used in particle tracking codes

Particle Tracking Code	Author	Interpolation scheme	Particle movement technique
MOC	Konikow & Bredehoeft (1978)	Bi-linear	Euler integration
RANDOM WALK	Prickett et al. (1981)	Bi-linear	Euler integration
GWPATH	Shafer (1987)	Bi-cubic	Runge-Kutta
MODPATH	Pollock (1989)	Linear	Semi-analytical
PATH3D	Zheng (1989)	Linear	Fourth order Runge-Kutta
FLOWPATH	Franz and Guiger (1990)	Reverse distance	Euler integration
WHPA	Blandford & Huyakorn (1991)	Linear	Semi-analytical or Euler

Linear velocity interpolation is implemented in ZOOPT. As stated above, this enables the analytical solution of the integral in Equation 2.2. The approach also maintains consistency with the finite difference mass balance equations, is generally more accurate than other methods in heterogeneous media and is computationally efficient. These issues are discussed by Zheng and Bennett (1995) who state that simple linear interpolations schemes are generally preferable to multi-linear interpolation schemes. Furthermore, the approach can easily form the basis of a semi-analytical *time-variant* particle tracking technique presented by Lu (1994).

The calculation of the velocity at the cell wall between two nodes is based on the inter-nodal volumetric flow rate calculated by the flow model, ZOOMQ3D. Considering the component in the x-direction, as illustrated in Figure 1, the velocity at the cell wall at position (i-$\frac{1}{2}$, j), denoted by $V_{x_{i-\frac{1}{2},j}}$ is

$$V_{x_{i-\frac{1}{2},j}} = \frac{Q_{x_{i-\frac{1}{2},j}}}{\theta \Delta y \Delta z} \text{ (m day}^{-1}\text{)} \qquad (2.3)$$

where

$Q_{x_{i-\frac{1}{2},j}}$ is the inter-nodal flow rate calculated by the flow model ($m^3 day^{-1}$),

θ is the aquifer porosity of cell (i, j),

$\Delta y = \left(y_{j+1} - y_{j-1}\right)/2$ (m) and Δz is the aquifer thickness of cell (i, j) (m).

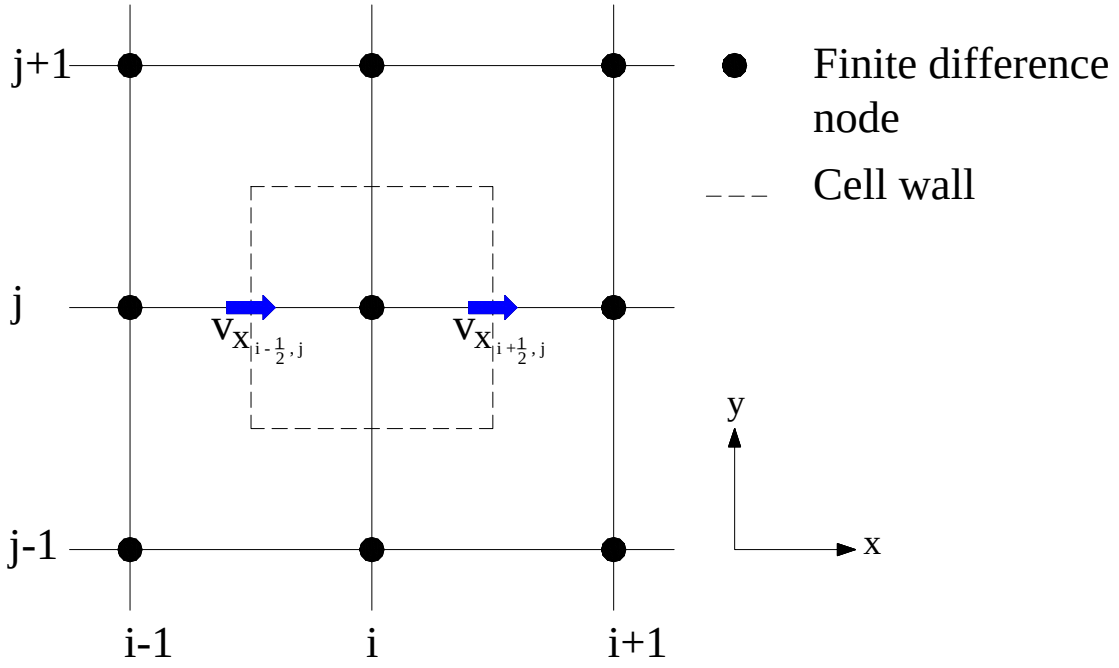


Figure 1 Inter-nodal velocity components in the x-direction

Given the inter-nodal velocities obtained from Equation 2.3, the x-component of the velocity at any arbitrary location within the cell (i, j) can be calculated using linear interpolation between the two opposite cell walls. The component of the velocity in the x-direction is calculated at an arbitrary x co-ordinate between $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$ using the following linear interpolation equation

$$V_x(x) = A_x\left(x - x_{i-\frac{1}{2}}\right) + V_{x_{i-\frac{1}{2}}} \qquad (2.4)$$

where

$V_x(x)$ is the component of the velocity in the x-direction at x and

$$A_x = \left(V_{x_{i+\frac{1}{2}}} - V_{x_{i-\frac{1}{2}}}\right) \Big/ \left(x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}\right) \qquad (2.5)$$

Similar equations to Equations 2.3-2.5 are defined in the y and z-directions by replacing the terms, x and i, by the terms y and j, or z and k, respectively. These are used to calculate the remaining two directional components of the velocity vector at any arbitrary position in the model domain.

2.3 PATHLINE DEFINITION

2.3.1 Semi-analytical technique

If linear interpolation is used to calculate the x, y and z-directional components of the velocity field at a particular position in the finite difference model domain, as described above, then the integral in Equation 2.2 can be solved analytically. Pollock (1988) called this method of particle tracking the 'semi-analytical' technique because of the combination of a numerical velocity interpolation routine and an analytical path line definition procedure. Considering the x-component of the velocity field only, then the equation of the particle track, Equation 2.1, is written

$$\frac{dx}{dt} = V_x \qquad \text{or} \qquad \frac{1}{V_x}dx = dt \tag{2.6}$$

Substituting Equation 2.4 into 2.6 and integrating between two arbitrary times, t_1 and t_2, gives

$$\int_{x(t_1)}^{x(t_2)} \frac{1}{A_x\left(x - x_{i-\frac{1}{2}}\right) + V_{x_{i-\frac{1}{2}}}}dx = \int_{t_1}^{t_2} dt \tag{2.7}$$

where

$x(t_1)$ and $x(t_2)$ are the particle co-ordinates at arbitrary times t_1 and t_2.

Equation 2.7 is integrated to give

$$\ln\frac{A_x\left(x(t_2) - x_{i-\frac{1}{2}}\right) + v_{x_{i-\frac{1}{2}}}}{A_x\left(x(t_1) - x_{i-\frac{1}{2}}\right) + v_{x_{i-\frac{1}{2}}}} = A_x\Delta t \tag{2.8}$$

where

$x(t_1)$ and $x(t_2)$ are the particle x co-ordinates at time t_1 and t_2 and,

$\Delta t = (t_2 - t_1)$.

Noting that from Equation 2.4

$$V_x(t_1) = A_x\left(x(t_1) - x_{i-\frac{1}{2}}\right) + V_{x_{i-\frac{1}{2}}} \tag{2.9}$$

then Equation 2.8 can be re-arranged to give

$$x(t_2) = x_{i-\frac{1}{2}} + \frac{1}{A_x}\left[v_x(t_1)\cdot\exp(A_x\Delta t) - v_{x_{i-\frac{1}{2}}}\right] \tag{2.10a}$$

Equivalent equations to Equation 2.10a can be derived in the y and z-directions. These are

$$y(t_2) = y_{j-\frac{1}{2}} + \frac{1}{A_y}\left[v_y(t_1)\cdot\exp(A_y\Delta t) - v_{y_{j-\frac{1}{2}}}\right] \tag{2.10b}$$

$$z(t_2) = z_{k-\frac{1}{2}} + \frac{1}{A_z}\left[v_z(t_1)\cdot\exp(A_z\Delta t) - v_{z_{k-\frac{1}{2}}}\right] \tag{2.10c}$$

Equations 2.10a to 2.10c are used to delineate the path line of the particle as it moves through the model domain. However, these equations are only applicable when the linear interpolation coefficients, A_x, A_y and A_z are constant. Consequently, a particle cannot be allowed to cross a cell wall between the time t_1 and time t_2. Pollock (1988) presents an efficient algorithm, which eliminates the possibility of this event by calculating the length of the tracking step that is required for a particle to travel from its current location to the cell wall through which it exits the node. For example, consider Figure 2, which shows the path of a particle in a two-dimensional model grid from its initial position (x_p, y_p) at time, t_p, to the point at which it exits the cell, (x_e, y_e), at time t_e. In this example the cell wall velocities are denoted by V_{x1} and V_{x2} in the x-direction and V_{y1} and V_{y2} in the y-direction for simplicity.

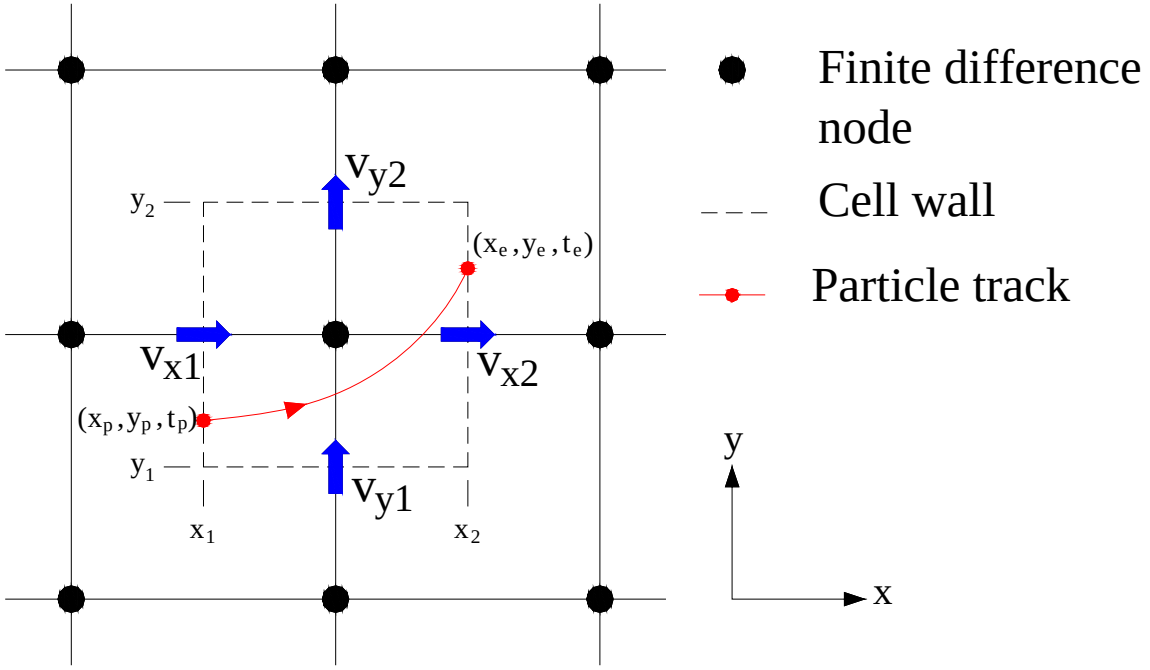


Figure 2 Particle track through a two-dimensional cell

To illustrate Pollock's algorithm, the assumption is made that all the cell wall velocities are greater than zero. Then if we also assume, in a first instance, that the particle leaves the cell through the wall at x_2, that is, in the positive x-direction, then Equation 2.10a can be used to calculate the length of time, Δt_x, that the particle takes to travel from x_p to x_2. Equation 2.10a gives

$$x_2 = x_1 + \frac{1}{A_x}\left[v_{xp} \cdot \exp\left(A_x \Delta t_x\right) - v_{x1}\right] \tag{2.11}$$

since

$x(t_2) = x_2$ and,

v_{xp} is the x-component of the velocity at the point (x_p, y_p)

Rearranging Equation 2.11 gives

$$A_x\left(x_2 - x_1\right) + v_{x1} = v_{xp} \exp\left(A_x \Delta t_x\right) \tag{2.12}$$

From Equation 2.4

$$A_x\left(x_2 - x_1\right) + v_{x1} = v_{x2} \tag{2.13}$$

and therefore by substituting this in Equation 2.12 we obtain

$$\Delta t_x = \frac{1}{A_x} \ln\left(\frac{v_{x2}}{v_{xp}}\right) \tag{2.14}$$

If it is assumed that the particle leaves the cell through the wall at y_2, then by a similar process the length of time, Δt_y, that the particle takes to travel from y_p to y_2 can be derived

$$\Delta t_y = \frac{1}{A_y} \ln\left(\frac{v_{y2}}{v_{yp}}\right) \tag{2.15}$$

The comparison of Δt_x and Δt_y defines through which cell wall the particle exits. The time for the particle to exit the cell, Δt_e, is taken as the smaller of Δt_x and Δt_y. If Δt_x is smaller than Δt_y the particle will exit the cell through the interface at $x = x_2$ and vice versa. If $\Delta t_x = \Delta t_y$ then the particle will exit at the corner of the cell through the point (x_2, y_2).

This method of moving the particle from cell wall to cell wall is easy to implement and computationally efficient. If the particle track needs to be defined in greater detail, Equations 2.10a to 2.10c can be used to define intermediate points along the path line within the cell. This is achieved by dividing Δt_e by the number of intermediate steps within the cell and then using multiples of this smaller time-step to calculate the intermediate points using Equations 2.10a to 2.10c.

In the above discussion it has been assumed that the interfacial velocities are all greater than zero for simplicity. However, there are three other possible flow conditions that must be identified before the semi-analytical solution algorithm can be applied. These are shown in Figure 3.

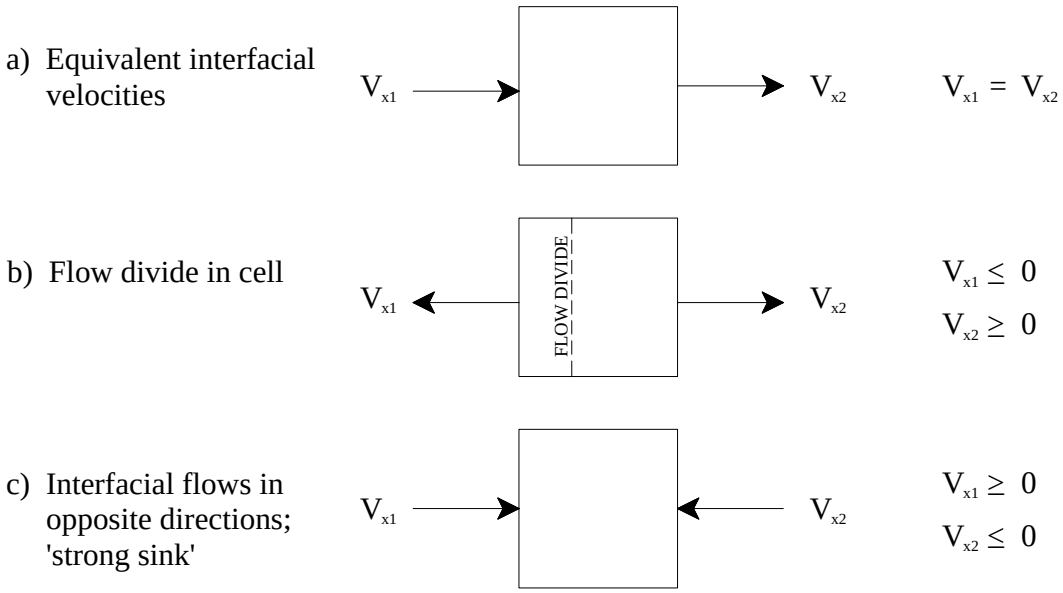


Figure 3 Cell wall velocity conditions needing consideration during particle tracking

In Figure 3a, the interfacial velocities are the same. In this case Equation 2.14 is undefined and the time of travel to exit the cell must be estimated by

$$\Delta t_x = \left(x_2 - x_p\right)/v_{x1} \quad \left(\text{if } v_{x1} > 0\right) \tag{2.16a}$$

$$\Delta t_x = \left(x_1 - x_p\right)/v_{x1} \quad \left(\text{if } v_{x1} < 0\right) \tag{2.16b}$$

In Figure 3b a local groundwater divide exists. In this case, if the particle is to the left of the divide it exits to the left otherwise it exits the cell to the right. Finally, Figure 3c shows the situation in which flow is toward the cell centre from both x-directions. In this case the particle cannot leave the cell in the positive or negative x-direction. If this is simultaneously true of the flows in the y and z-directions then the node is termed a 'strong sink' and the particle is terminated at the cell.

2.3.2 Numerical integration techniques

Equation 2.2 can also be solved numerically. In fact, if a higher order interpolation scheme is used then it can only be solved using numerical methods. Numerical integration methods involve the movement of the particles in discrete *tracking steps* along the path line. Of the numerical integration techniques, Euler's method is the simplest. In this method the velocity at the current point is extrapolated over the tracking step. The particle is moved along the path line using the equations

$$x' = x_p + v_{xp}\Delta t \tag{2.17a}$$

$$y' = y_p + v_{yp}\Delta t \tag{2.17b}$$

$$z' = z_p + v_{zp}\Delta t \tag{2.17c}$$

where

x', y' and z' are the co-ordinates of the particle's new location,

x_p, y_p and z_p are the co-ordinates of the particle's current location,

v_{xp}, v_{yp} and v_{zp} are the components of the particles velocity at its current location and,

Δt is the length of the tracking step.

Whilst Euler's method is straightforward to implement, the length of the tracking step, Δt, must generally be small to maintain accuracy. This is because the velocity is extrapolated over the tracking interval. A numerical method with a higher order of accuracy is that of the Runge-Kutta technique. This is implemented in ZOOPT in addition to the semi-analytical technique. The Runge-Kutta method moves a particle over a tracking step, Δt, by combining information from a number of Euler-type steps. It is generally more accurate than Euler's method but computationally less efficient, however, the Runge-Kutta method need only be implemented in ZOOPT in a few specific situations. These are discussed in detail in Section 3 of this report.

In the Runge-Kutta method the velocity is calculated four times for each tracking step; once at the current particle location, twice at two trial midpoints and once at a trial end point. With reference to Figure 4, a two-dimensional case, the process is defined using the following equations

$$x_{n+1} = x_n + \frac{\left(v_{xp1} + 2v_{xp2} + 2v_{xp3} + v_{xp4}\right)}{6}\Delta t \tag{2.18a}$$

$$y_{n+1} = y_n + \frac{\left(v_{yp1} + 2v_{yp2} + 2v_{yp3} + v_{yp4}\right)}{6}\Delta t \tag{2.18b}$$

where

$\left(v_{xpi}, v_{ypi}\right)$ are the velocity components at the points $\left(x_{pi}, y_{pi}\right)$ for i = 1 to 4,

Δt is the length of the tracking step and,

$\left(x_{n+1}, y_{n+1}\right)$ is the final position of the particle at the end of the tracking step.

Equations 2.18a and 2.18b cannot be applied directly because the velocities, v_{xpi+1} and v_{ypi+1} depend on the co-ordinates of x_{pi} and y_{pi}. Hence, the co-ordinates of the particle at the end of the tracking step are calculated iteratively.

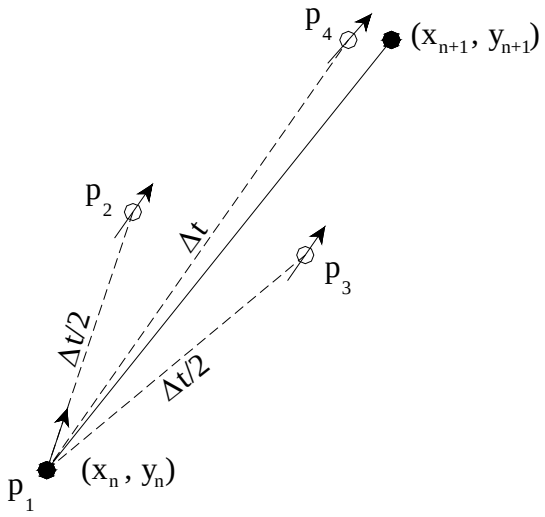


Figure 4 Intermediate steps taken during fourth-order Runge-Kutta method

The velocities at each of the points p_1, p_2, p_3 and p_4 are calculated using linear interpolation of the cell wall velocities after the co-ordinate of each previous trial point has been calculated. The co-ordinates of these points are calculated by performing the following steps. The co-ordinates of p_2 are calculated using

$$x_{p2} = x_{p1} + v_{xp1}\,\Delta t/2 \tag{2.19a}$$

$$y_{p2} = y_{p1} + v_{yp1}\,\Delta t/2 \tag{2.19b}$$

from which the co-ordinates of the point p_3 are subsequently calculated using

$$x_{p3} = x_{p1} + v_{xp2}\,\Delta t/2 \tag{2.20a}$$

$$y_{p3} = y_{p1} + v_{yp2}\,\Delta t/2 \tag{2.20b}$$

and then finally the co-ordinates of the point p_4 are determined using

$$x_{p4} = x_{p1} + v_{xp3}\Delta t \tag{2.21a}$$

$$y_{p4} = y_{p1} + v_{yp3}\Delta t \tag{2.21b}$$

By repeating this procedure the particle is moved through the model domain until it reaches a discharge point. The above equations are easily extended for application of the technique to three-dimensional problems as in ZOOPT, the particle tracking code developed here.

Whilst the Runge-Kutta technique incorporates a higher order of accuracy than the simpler Euler's method, it is computationally less efficient. Consequently, it is important to optimise the length of the tracking step during the procedure to both maintain accuracy and minimise computational effort. This is performed in ZOOPT using the 'step doubling' procedure presented by Zheng and Bennett (1995). In this procedure the tracking step is performed twice. First the tracking step is made using a time interval of Δt and then it is repeated by taking two steps of half the length i.e. $\Delta t / 2$. The distance, Δs, between the two points calculated using a full step and two half steps is used to adjust the full length of the tracking step. As Zheng and

Bennett (1995) state, if the fourth-order Runge-Kutta technique is used, the tracking solution is accurate to the fourth-order, hence, Δt can be scaled as $(\Delta s)^{\frac{1}{5}}$. Equation 2.22 is used to calculate the required tracking step size, Δt_0, that will yield an error less than Δs_0, given an initial calculation of Δs using an initial tracking step of Δt. The term f_s is a safety factor and is given a value slightly smaller than one e.g. 0.9.

$$\Delta t_0 = f_s \, \Delta t \left(\frac{\Delta s_0}{\Delta s} \right)^{\frac{1}{5}} \tag{2.22}$$

3 Capabilities of the particle tracking code, ZOOPT

3.1 INTRODUCTION

The particle tracking code ZOOPT has been developed for use in conjunction with the object-oriented groundwater model ZOOMQ3D. ZOOMQ3D is described in detail in Jackson (2001, 2002a and 2002b) and Jackson and Spink (2004). In addition to representing hydrogeological features, such as rivers and pumped wells, that are commonly simulated using groundwater flow models, ZOOMQ3D incorporates the vertical variation of hydraulic conductivity with depth and local grid refinement. Particle tracking within ZOOPT is compatible with all of these mechanisms.

At this stage of development, ZOOPT enables steady-state particle tracking under advective transport in both the forward and reverse directions. The code can be applied to the definition of borehole catchments and associated source protection zones, the identification of recharge and discharge areas and the visualisation of groundwater flow patterns, for example. Back tracking is easy to implement within a particle tracking code. Steady-state particle tracking can be applied easily at any time step of a time-variant simulation. The flows calculated at the end of a time step are used to define an approximate borehole catchment by implementing steady-state particle tracking routine. This approach can be useful to analyse, for example, the approximate change in shape of a borehole catchment during a seasonal recharge or abstraction cycle. In addition to tracking particles under steady-state conditions, the code can be used to forward track particles in unsteady flow fields.

Particles are tracked using the semi-analytical technique described above. However, around particular model features, for example 'weak' sink nodes or nodes which exhibit a vertical variation of hydraulic conductivity, particles have to be tracked using the Runge-Kutta method. The switch between the use of the semi-analytical and Runge-Kutta methods is made automatically within the model code. Though the Runge-Kutta technique is only implemented occasionally, the user can enforce its continuous use. This option provides an alternative to the semi-analytical technique, though, the semi-analytical method is computationally more efficient and should be used in preference to Runge-Kutta tracking.

ZOOPT is designed to track particles in models where the horizontal hydraulic conductivity varies with depth (VKD) and this is one feature that requires the application of the Runge-Kutta method. In this case the horizontal velocity depends on the hydraulic conductivity at the elevation of the particle within the node. Consequently, the integral in Equation 2.2 cannot be determined analytically and a fully numerical tracking method must be employed.

In addition to the application of the particle tracking code to VKD nodes, the code is compatible with the local grid refinement technique incorporated within ZOOMQ3D. Local grid refinement enables the zooming of the mesh within discrete areas of a model grid to increase accuracy or model detail. ZOOPT tracks particles through these locally refined grids.

The occurrence of 'weak' sinks, which is a problem associated with particle tracking is circumvented by ZOOPT. Weak sinks are commonly generated when, for example, an abstraction well, which distributes its effect over the volume of the cell, is not sufficiently strong to cause groundwater to flow into the associated finite difference node through all of its faces. In this case, it is not possible to determine whether a particle should leave the cell through one of its walls or whether the particle should terminate at the well. An approach is adopted in ZOOPT that eliminates the problems associated with weak sinks. This is discussed in Section 3.2.

3.2 VELOCITY CALCULATION

3.2.1 Weak sinks

A problem that can be associated with particle tracking codes is that of 'weak' sinks. In finite difference models groundwater can flow out of a node through a sink that is distributed throughout the whole volume of the cell. For example, abstraction wells, rivers, leakage nodes and springs are all simulated in ZOOMQ3D using such sinks. If the discharge rate of the sink is sufficient, water will flow into the node across all of its six faces. This is termed a 'strong' sink. However, if the discharge rate of the sink is insufficient to cause inflow across all sides of the node, that is, water flows out of the cell across one or more of its six faces, then a weak sink is generated. Weak sinks present a problem because it is not possible to determine whether a particle leaves the node through a cell wall or through the sink. This is because the calculation of the velocity within the cell is based on the discharge rates across the cell interfaces only and does not take account of the effect of a distributed sink on the velocity field.

Weak sinks caused by abstraction wells are dealt with separately in the next section. With regard to the other model features listed above that can cause weak sinks, the problem can be circumvented by assuming that the discharge to the sink actually occurs through one of the cell walls. In effect, therefore, the sink is removed from the node and one of the cell wall velocities is re-calculated. In ZOOPT the flows between the aquifer and rivers, head dependent leakage nodes and springs, which are represented as distributed sinks, are all assigned to the upper face of the corresponding finite difference node. The velocity is then recalculated across the node's upper face.

Recharge is dealt with in a similar manner. Recharge is assumed to fall vertically onto the aquifer. Consequently, the velocity across the upper face of the node is adjusted to account for this inflow. Abstraction wells are the other features of ZOOMQ3D that can generate weak sinks. These are less straightforward to deal with and are thus discussed separately in the next section.

3.2.2 Weak sinks caused by abstraction wells

As stated in the last section, the creation of weak sinks by mechanisms other than wells is dealt with by assigning the discharge to one of the walls of the finite difference node. The selection of the appropriate wall to which the flow is assigned is based on physically justifiable assumptions, for example, groundwater recharge arrives at the water table from above. However, with regard to abstraction wells, such an assumption is not justifiable because groundwater is drawn towards wells from all directions. Consequently, abstraction wells present a more significant problem. To identify if a particle terminates at a weak sink well Zheng (1994) uses a special velocity interpolation scheme within the corresponding finite difference node. This is based on the superposition of an analytical solution for radial flow to a well and a solution for unidirectional regional groundwater flow. The interpolation scheme alters the velocity components in the x and y-directions but continues to use linear interpolation in the z-direction. With reference to Figure 5, the horizontal components of the particles velocity are given by

$$V_{xp} = \frac{Q'_w \sqrt{a}}{2\pi(z_2 - z_1)\theta}\left[\frac{x_p}{x_p^2/a + y_p^2}\right] + \frac{(V_{x1} + V_{x2})}{2} \tag{3.1}$$

$$V_{yp} = \frac{Q'_w \sqrt{a}}{2\pi(z_2 - z_1)\theta}\left[\frac{y_p}{x_p^2/a + y_p^2}\right] + \frac{(V_{y1} + V_{y2})}{2} \tag{3.2}$$

where

x_p and y_p are the particle's x and y co-ordinates with respect to the centre of the node,

V_{xp} and V_{yp} are the x and y components of the particle's velocity,

V_{x1}, V_{x2}, V_{y1}, V_{y2}, V_{z1}, and V_{z2} are the x, y and z components of the cell wall velocities,

$Q'_w = Q_{WELL} - [(V_{z1} - V_{z2}) \cdot (x_2 - x_1) \cdot (y_2 - y_1)]$,

Q_{WELL} is the abstraction rate of the well,

x_1, x_2, y_1, y_2, z_1 and z_2, are the x, y and z co-ordinates of the cell walls,

$a = (K_y / K_x)$ is the ratio of hydraulic conductivities in the x and y-directions and,

θ is the porosity of the node.

This interpolation scheme forces particles either to converge towards the well or to leave the finite difference node through one of its interfaces. Particles are terminated if they enter within the radius of the well. The scheme is implemented in ZOOPT and is an elegant solution to the problem of weak sink wells. At wells that generate strong sinks this interpolation scheme is not applied and all particles are terminated as they enter the node.

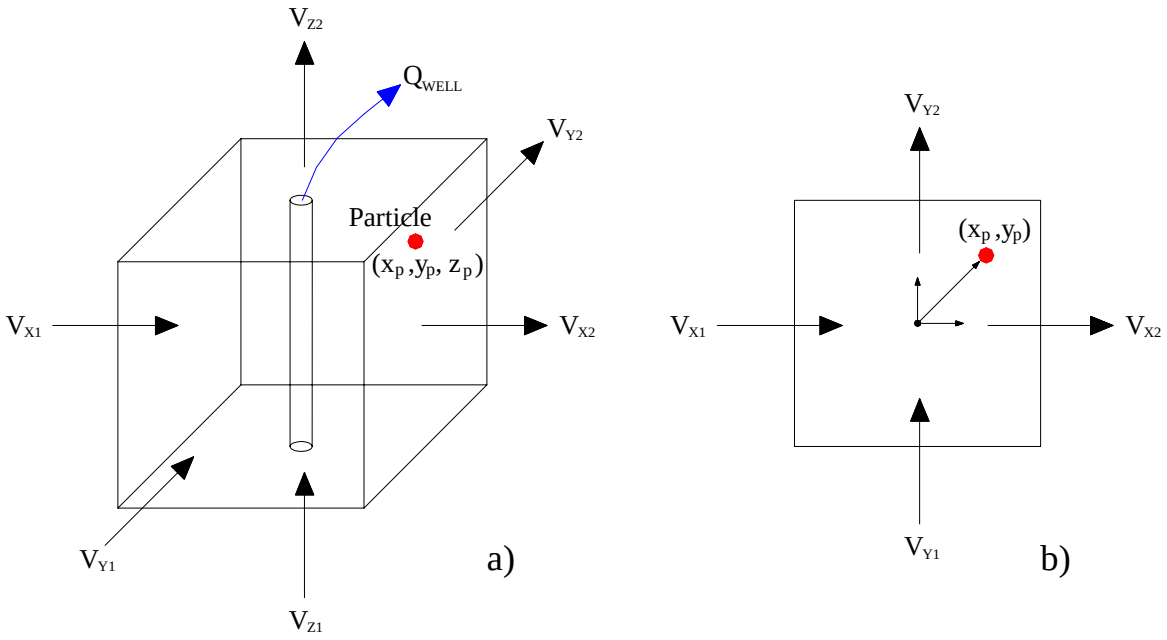


Figure 5 Illustration of a well as a distributed sink at the centre of a finite difference node showing surrounding cell walls in a) three dimensions and b) plan view

3.3 GRID CONSIDERATIONS

3.3.1 Distorted vertical discretisation

Because of efficiency considerations finite difference models are often constructed in which the elevation of the top and bottom of model layers varies over the model domain. This variation generally results from the need to approximate the changing shape of hydrogeological units. However, vertical distortion of the mesh causes problems because the particle tracking solution is based on a fixed orthogonal grid. For example, consider that it is calculated that a particle leaves a model cell in the horizontal and positive x-direction. If the particle leaves towards the top of the cell and the grid is distorted then it is possible for the particle to enter a node contained

in an upper layer. This situation is shown in Figure 6a. If this occurs the particle position must be corrected before the next particle move is made.

When the semi-analytical solution is adopted the particle's elevation is adjusted on the interface between nodes, that is, when its passes from one node to the next. At this point the local z co-ordinate of the particle prior to the correction, with respect to the top and bottom elevations of the first node, must be equal to its local z co-ordinate in the node it is entering after the correction. This calculation is shown in Figure 6b. Because of this correction the particle path can appear unsmooth when plotted. This problem is inherent in the representation of three-dimensional models as vertically varying layers.

When the Runge-Kutta method is used the particles position must be modified after the tracking step within the node the particle has entered. The required vertical correction is calculated by considering that if the vertical velocity component was zero, the particle's local z co-ordinate within each cell would have remained the same. The correction factor, Δz_c, is calculated using this assumption and is given by

$$\Delta z_c = \frac{\Delta z_2}{\Delta z_1}\left(z_n - z_1\right) + z_2 - z_n \tag{3.3}$$

The correction procedure is illustrated in Figure 6c. A full derivation of this correction term is presented by Zheng (1994).

3.3.2 Unconfined aquifer layers

In phreatic aquifer layers the elevation of the water table determines the vertical thickness of the finite difference node. The elevation of the water table is defined as the simulated groundwater head. Consequently, in unconfined model layers the nodes are distorted vertically again. The position of particles that are tracked through unconfined nodes are corrected in the same way as described for fixed but vertically distorted grid nodes.

3.4 QUASI THREE-DIMENSIONAL LAYERING

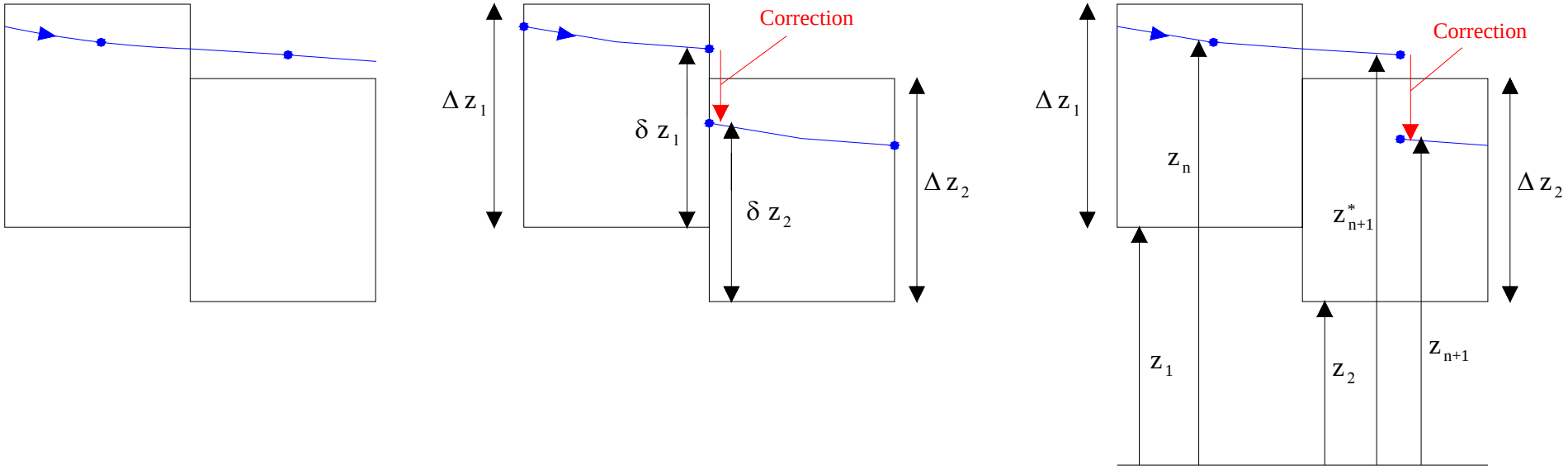
Not all hydrogeological layers are always included explicitly in a groundwater model. Consider that a groundwater system is composed of a sequence of high permeability horizontal layers that are separated by a series of low permeability aquitards. Often these aquitards are not represented by a series of finite difference nodes in a groundwater model. This is because the assumption can be made that the flow in the low permeability layers will be essentially vertical. Consequently, low permeability layers are commonly modelled by adjusting the vertical conductance between the two adjacent aquifer layers. When this approach is adopted the model is stated to be *quasi three-dimensional*.

Whilst this method of representing aquitards in groundwater models is computationally efficient, the particle tracking routine has to be modified because there are no grid nodes associated with these low permeability layers. ZOOPT recognises the presence of quasi three-dimensional layers and moves the particles vertically through them. The time of travel through the low permeability layer is calculated from the leakage between the two adjacent simulated aquifers, however, the user must specify the porosity of each aquitard.

a) Uncorrected particle path

b) Semi-analytical solution
Correction at nodal interface

c) Runge-Kutta solution
Correction after tracking step



$$\frac{\delta z_1}{\Delta z_1} = \frac{\delta z_2}{\Delta z_2}$$

$$z_{n+1} = z^*_{n+1} + \frac{\Delta z_2}{\Delta z_1}\left(z_n - z_1\right) + z_2 - z_n$$

Figure 6 Correction of particle elevation in vertical distorted model layers

3.5 VERTICAL VARIATION OF HYDRAULIC CONDUCTIVITY WITH DEPTH

The flow model ZOOMQ3D incorporates the variation of hydraulic conductivity with depth within layers of the finite difference grid. This representation of the vertical variation of hydraulic conductivity provides an alternative to the development of multi-layer models, in which individual layers are characterised by uniform horizontal hydraulic conductivity in the vertical direction.

The approach has been developed to enable the more accurate description of the variation of hydraulic conductivity in limestone, and particularly Chalk, aquifers, in which higher hydraulic conductivity values are often associated with the zone of fluctuation of the water table. The method circumvents numerical difficulties that are related to the de-watering of layers in multi-layer models. The variation of the horizontal hydraulic conductivity with depth is defined by profiles such as that shown in Figure 7.

A VKD profile describes the change in hydraulic conductivity with depth at a particular point in the aquifer. Profiles are defined by two sections. In the lower section, between Z_{BOTTOM} and Z_P in Figure 7, hydraulic conductivity is constant. In the upper section, between Z_P and Z_{TOP}, hydraulic conductivity increases linearly with elevation. Because different values of hydraulic conductivity can be specified in the two orthogonal horizontal directions (x and y), six values are used to parameterise the profile:

1. The elevation of the base of the profile, Z_{BOTTOM}.
2. The elevation of the top of the profile, Z_{TOP}.
3. The elevation of the point of inflection, Z_P.
4. The hydraulic conductivity in the x direction, K_x^*, below Z_P.
5. The hydraulic conductivity in the y direction, K_y^*, below Z_P.
6. The gradient of the profile above Z_P, VKDGrad. This is equal to the increase in hydraulic conductivity per metre rise in elevation.

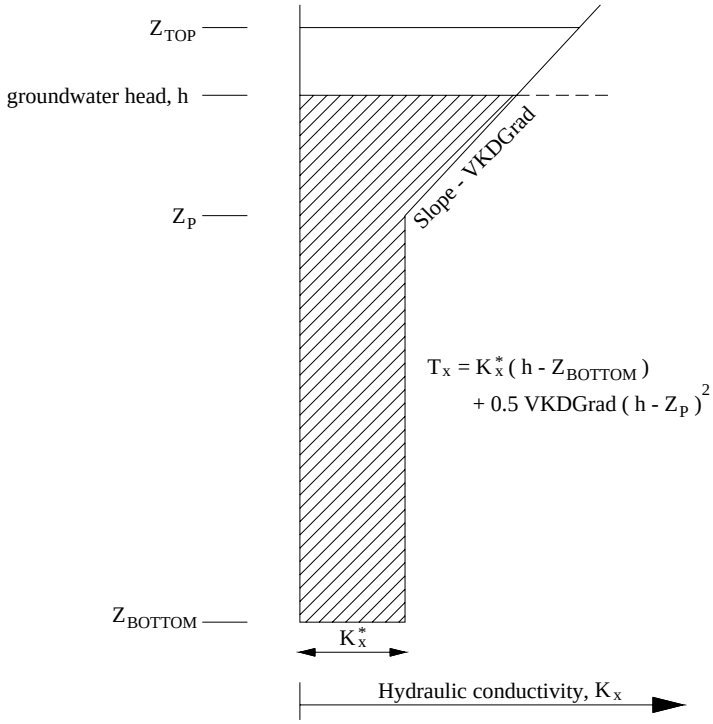


Figure 7 Parameters used to define VKD profiles in ZOOMQ3D

The value of the VKDGrad parameter may be either negative, zero or positive. Consequently, in addition to an increase in hydraulic conductivity with depth above Z_P, hydraulic conductivity can be specified to decrease or remain constant. To calculate transmissivity the following equations are used

$$T_x = K_x^*(h - Z_{BOTTOM}) + 0.5\,VKDGrad \cdot (h - Z_P)^2$$

$$T_y = K_y^*(h - Z_{BOTTOM}) + 0.5\,VKDGrad \cdot (h - Z_P)^2$$

for $h > Z_P$, and

$$T_x = K_x^*(h - Z_{BOTTOM})$$

$$T_y = K_y^*(h - Z_{BOTTOM})$$

for $h \le Z_P$, where h is the water table elevation.

At those nodes of the finite difference grid where hydraulic conductivity varies with elevation the Runge-Kutta particle tracking technique must be used to define path lines. The integral in Equation 2.2 cannot be evaluated analytically because the horizontal velocity varies in the z-direction, that is, towards the top of the VKD profile the horizontal velocity is greater than towards its base. Consequently the semi-analytical method cannot be used. Within ZOOPT the assumption is made that the horizontal velocity of the particle is proportional to the horizontal hydraulic conductivity at its location. With reference to Figure 8, the component of the velocity in the x-direction at the cell walls at x_1 and x_2 are given by

$$V_{x1}(z) = \frac{K_x(z)}{\overline{K}_x} \cdot \frac{Q_{x1}}{\theta \Delta y (z_2 - z_1)} \tag{3.4}$$

$$V_{x2}(z) = \frac{K_x(z)}{\overline{K}_x} \cdot \frac{Q_{x2}}{\theta \Delta y (z_2 - z_1)} \tag{3.5}$$

where

$V_{x1}(z)$ and $V_{x2}(z)$ are the x-components of velocity (m day^{-1}) on the cell walls at elevation z,

Q_{x1} and Q_{x2} are the flow rates entering and exiting the cell in the x-direction (m^{3}day^{-1}),

$K_x(z)$ is the hydraulic conductivity (m day^{-1}) in the x-direction at elevation z,

$\overline{K}_x = \frac{1}{(z_2 - z_1)} \int_{z_1}^{z_2} K_x(z)\, dZ$ is the mean hydraulic conductivity (m day^{-1}) in the x-direction,

θ is the porosity of the node,

Δy is the width of the node in the y-direction (m) and,

z_1 and z_2 are the elevations of the bottom and top of the node (m).

Similar equations are written for the component of velocity in the y-direction. However, because the hydraulic conductivity in the z-direction is considered uniform throughout the node, no modification is made to way in which the z-component of velocity is calculated. ZOOPT recognises when a particle enters a node in which hydraulic conductivity varies with depth and then invokes the use of the Runge-Kutta technique and the application of Equations 3.4 and 3.5.

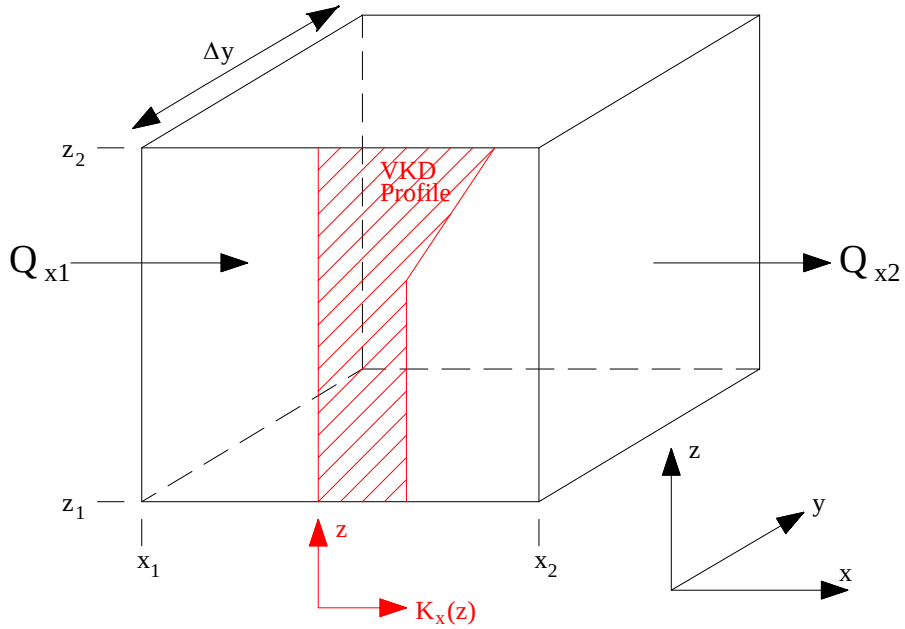


Figure 8 Illustration of the interpolation of velocity in VKD nodes

4 Running ZOOPT

To install ZOOPT on a Windows PC copy the executable 'zoopt.exe' into suitable directory such as 'c:\Program Files\ZOOM'. Then add this directory to the Windows system PATH variable (Control Panel➔System➔Advanced Tab➔Environment Variables). No installation procedure is run in which ZOOPT program files are added to the system registry. All the input files required by ZOOPT must be located in a single directory. All the output files produced by ZOOPT will be created in the same directory

ZOOPT should be run from the command line in a MS-DOS console box and not started from Windows Explorer. To start a command window select 'Run' from the Windows start menu and type 'cmd' in the drop down list box (Figure 9). The user should then change directory to that of the working directory where the input files are located (Figure 10). For help on the commands used to change directory type 'help cd' within the console box (Figure 10). To run the model type 'zoopt' followed by the path to the working directory on the command line e.g. '**zoopt c:\myDirectory**'. Alternatively, this string can be placed in a batch file (a text file with a .bat extension e.g. 'runzoopt.bat') and the name of this batch file can be typed on the command line (omit the extension when doing this e.g. type 'runzoopt').

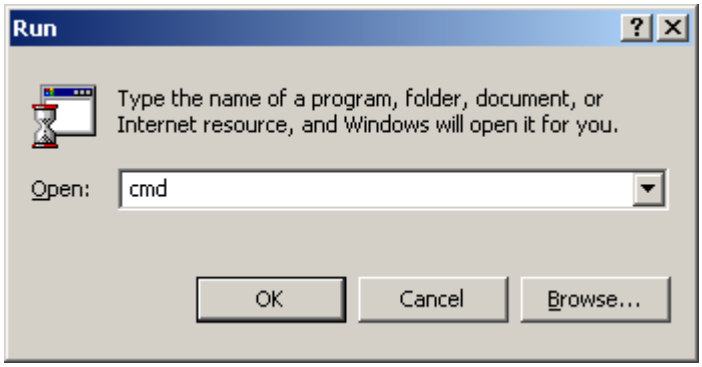


Figure 9 Starting a command line window from the Windows start menu

In the event that an error occurs, messages are written to the screen. If ZOOPT is run from Explorer it may terminate before the user is able to read the error messages. The program reads data from ASCII text input files, which must be located in a single working directory. The formats of all the input data files, which are specific to ZOOPT and not also required by ZOOMQ3D, are described in detail in the relevant section of this manual.

Running the particle tracking ZOOPT model is straightforward. The following procedure is undertaken.

- Run the flow model ZOOMQ3D.
- Copy the ZOOMQ3D input files listed in Table 1 and the ZOOMQ3D output files 'heads.txt' and 'flowbal.txt' into the ZOOPT directory.
- Create and edit the ZOOPT input files listed in Table 3.
- Run ZOOPT.

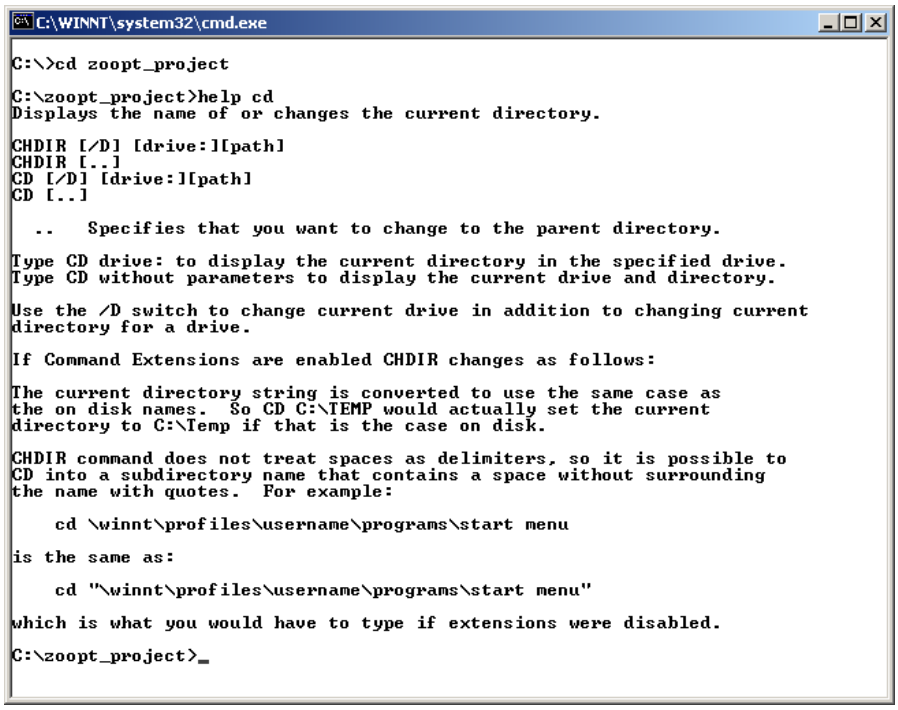


Figure 10 Example of changing the working directory within a console window

The size of the console box can be adjusted by clicking on the icon in the top left hand corner of its window and selecting 'Properties' from the menu list. Suitable values for the width and height of the window and its associated screen buffer are shown in Figure 11.

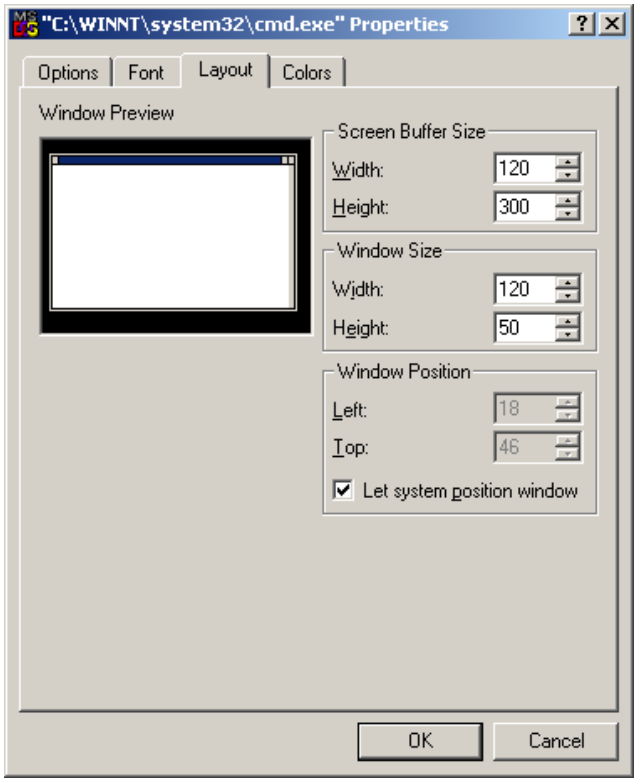


Figure 11 Changing the properties of the console window

5 ZOOPT input files

Most of the input files for ZOOPT are also required to run the flow model ZOOMQ3D. Only a small number of additional input files need to be created to run the particle tracking model. The ZOOMQ3D input files that also form input to ZOOPT are listed in Table 1. These relate to the structure of the numerical model, to the aquifer's hydraulic parameters and to the stresses applied to the system. The format of these files is discussed in detail in the ZOOMQ3D user's manual (Jackson and Spink, 2004) and consequently, they are not described within this document.

In addition to the ZOOMQ3D input files listed in Table 1, two output files produced by the flow model are required as input for ZOOPT: 'heads.txt' and 'flowbal.txt'. These contain the groundwater heads and components of flow simulated at each model node for each time-step of the simulation. The formats of these two files are discussed in Section 5.5.

Finally, the files listed in Table 3 are also required as input to ZOOPT. These are specific to the particle tracking model. The formats of these input files are discussed in Sections 5.1 to 5.4

5.1 INPUT FILE 'ZOOPT.DAT'

This input file is the main control file for the particle tracking simulation. It is used to specify the direction of tracking and whether the flow field is steady or dynamic, in addition to parameters relating to the particle tracking technique. An example file and its format is shown in Figure 12. It is composed of seven pairs of lines, the first of each being a comment line and the second a data line. A description of each of the data lines is presented next.

Line 2. Two character flags are specified on this line separated by a space. The first specifies whether the tracking direction is forward (f) or backward (b). The second specifies whether the run is a steady-state (s) time-instant (i) or time-variant (t) simulation. If it is a steady-state run the particles are tracked using the flow field simulated during the first time-step of the flow model simulation. Steady-state particles tracks can also be produced for any other time-step of the flow model simulation. This is referred to here as time-instant tracking. The steady-state flow field can be based on the nodal flows and heads for any time-step of the flow model simulation (see line 16). Finally, particles can be tracked through unsteady flow fields across a number of model time-steps. Time-variant backward tracking has not been implemented yet.

Line 4. A single integer is entered on this line. This specifies the number of points at which the particles position will be calculated between the cells walls. If a zero is entered, the particles path lines will only be defined at those points on the interface between two finite difference nodes. In this case the path lines may appear non-smooth when visualised.

Line 6. Two decimal numbers are entered on this line. They are the Runge-Kutta safety factor and error criterion, which are discussed in Section 2.3.2 of this manual. The safety factor should be a assigned a value slightly smaller than one e.g. 0.9. The error criterion is a distance in metres.

Line 8. A single character flag is entered on this line, which can be used to enforce the use of the Runge-Kutta technique to track the particles. If this is not enforced the semi-analytical method is used to track the particles except within weak sink nodes, where Runge-Kutta is implemented automatically.

Line 10. A single decimal value is entered on this line. Three-dimensional plots of the particles path lines are plotted as dxf files. The plots can be stretched in the z-direction by entering a number greater than 1.0. This number is used to multiply the z values of all the particle locations when writing the dxf file.

Line 12. A single character flag is entered on this line. This determines if the model grid is written to the dxf files for visualisation in addition to the particle path lines.

Line 14. On this line the length of time is specified (days) for which to track the particles. The distance that particles move during this period is dependent on the porosity of the model layers.

Line 16. On this, the last line of the input file, three integers are defined. These are the time-step, stress period and block used to define time-instant particle tracks (see description of line 2). The block number is a counter and not the actual value of the block i.e. if the blocks represented years, which had values starting from 1970, a value of five would be entered to define the fifth year, not the number 1974.

5.2 INPUT FILE 'PARTICLES.DAT'

The number of particles to be tracked and their starting positions are defined in the input file 'particles.dat'. This file has the following format:

```
n
x   y   z   layer   t_r        (one line of data for each of n particles)
```

where

`n` is the number particles to be tracked,

`x` is the x co-ordinate (m) of the particle on release,

`y` is the y co-ordinate (m) of the particle on release,

`z` is the z co-ordinate (m) of the particle on release (local or global depending on the layer number),

`Layer` is the number of the layer in which the particle is released and,

t_r is the time of release of the particle (days).

The z co-ordinate can be specified as either a local co-ordinate within a layer or as a global co-ordinate. If the layer number specified in the file is 0, the z co-ordinate is *global* and is related to the elevation of the datum of the model as defined by the user. If the layer integer number is greater than zero, the z co-ordinate is *local* and must be between 0.0 and 1.0 i.e. it is defined as a fraction of the layer thickness. If it is 0.0 then the particle is placed on the bottom of the layer. If it is 1.0 then it is placed at the top of the layer.

If the program is tracking particles backwards the time of release t_r is automatically re-set to zero days.

	‘zoopt.dat’	File format
1	// Forward (f) or back tracking (b), steady-state (s), time-instant (i) or time-variant (t)	Comment line
2	b s	Two character flags
3	// Number of intermediate tracking points within cell (>=0)	Comment line
4	10	Integer
5	// Runge-Kutta safety factor (<1.0) and error criterion (m)	Comment line
6	0.9 0.000001	Two decimal numbers
7	// Enforce the use of Runge-Kutta (y or n)	Comment line
8	n	Character
9	// DXF drawing z scale factor (>=1.0)	Comment line
10	10.0	Decimal
11	// Draw grid in DXF file (y or n)	Comment line
12	y	Character
13	// Length of time for which to track particles (days)	Comment line
14	100000.0	Decimal
15	// Time-step, stress period & block for time-instant tracking	Comment line
16	3 12 4	Three integers

Figure 12 Example ‘zoopt.dat’ input file and file format

5.3 FILE FORMATS FOR INPUTTING SPATIAL DATA (POROSITY)

Spatial information can be entered into the model using two different methods, or sets of input files. These methods employ either:

1. *map* and *code* input files or,
2. *numeric* input data files

In method 1 a pair of ASCII text files is required for each model layer with the file extensions *map* and *cod*. A map file contains a single array, or multiple arrays if grid refinement is incorporated, of characters in the range [a, b, ... y, z, A, B, ...Y, Z] to define zones of different parameter values. That is, the range is composed of the lower case alphabet followed by the upper case alphabet. Each array represents one of the grids contained in the model. These characters define zones across the model mesh at which a particular value of a parameter is specified. The values of the parameters are specified in the corresponding file with the *cod* extension. A maximum of fifty-two zones can be used to define the spatial distribution of model parameters when using this method. The name of each of these pairs of files is suffixed with a two-digit number that represents the model layer that the data applies to. For example, the pair of files 'porosity01.map' and 'porosity01.cod' would be used to specify the spatial distribution of porosity in layer 01, the upper model layer.

In method 2 a single ASCII text file is required for each model layer with the *dat* file extension. Each of these files contains a single array, or multiple arrays, of numeric data with each array representing one of the grids contained in the model. These numbers specify the value of a specific parameter at each finite difference node within the model directly. The name of each of these files is suffixed with a two-digit number that represents the model layer that the data applies to. For example, the file 'porosity03.dat' would be used to specify the spatial distribution of porosity in layer 03, i.e. two layers below the top layer of finite difference nodes.

5.3.1 Selection of spatial data entry method

Either of the methods described above can be used to assign values for each of the parameters that can vary spatially across the model domain. To specify which data entry method is to be used, the user must adjust the input file 'entry_method.dat'. The format of this file is shown in Figure 13. It consists of a series of integers, each of which is preceded by a comment line. There is also an additional comment line at the top of the file. Each comment line, except for the first, notifies the user of the related parameter, e.g. specific yield. The entry mode of the specified parameter is entered on the next line as an integer value. If this integer is equal to 1, a pair of map and code files is required for each model layer. If the integer is equal to 2, one numeric data file, with the *dat* extension, is required for each layer.

```
// Entry method 1: Map and codes files  2: Raw data
// Base elevations
2
// Top elevations
2
// Specific storage
1
// Specific yield
2
// Hydraulic conductivity
2
// Anisotropy
1
// VKD
1
// Vertical conductance
2
// Wetting thresholds
1
// Post wetting heads
1
// Porosity
1
```

Figure 13 'entry_method.dat' file format

5.3.2 Specifying spatial data using map and code files

In this method the spatial distribution of parameter values is defined using 'maps'. These maps are contained in files with the *map* extension. For example 'porosity02.map' would be used to define porosity in layer 02 of the model, i.e. one layer down from the top layer. The map files contain one character array for each grid in the model. For example, in Figure 14 the model mesh is composed of four grids: the coarsest base grid, two child grids (on grid level 2) and one grandchild grid (on grid level 3). Consequently, four character arrays are required in each of the map files for this example model. These are listed in grid level order and are separated by a comment line.

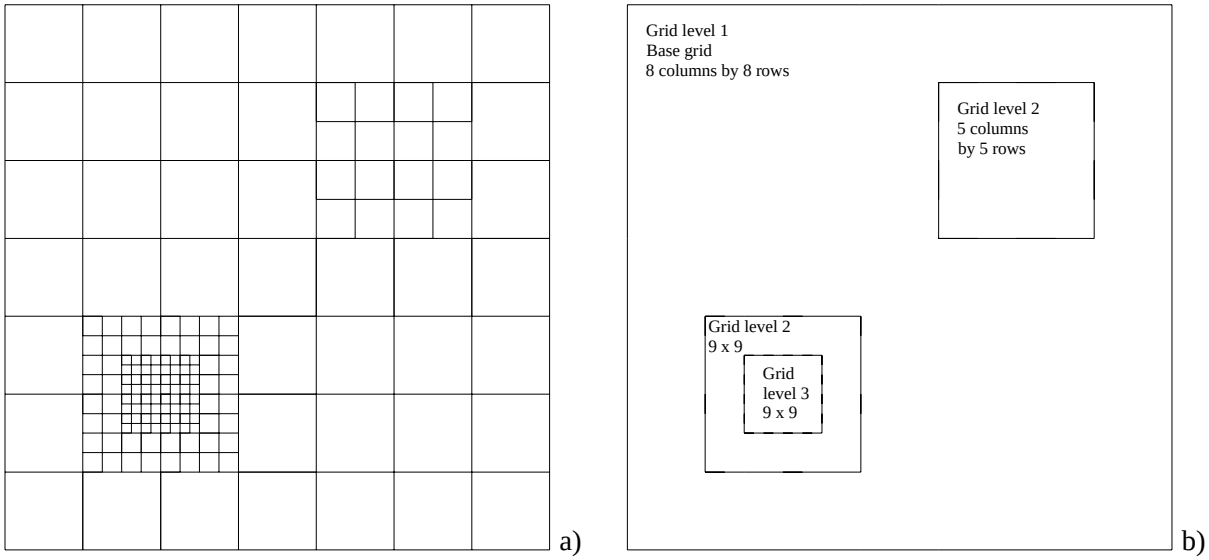


Figure 14 a) Example mesh composed of four grids and b) representation of the grid hierarchy

The structure of the map file for this example model is shown in Figure 15. As described above, this contains four character arrays separated by comment lines. The comment lines contain information required to identify to which grid each array relates. ZETUP, the set up program for ZOOMQ3D, produces this file and writes the co-ordinates of the bottom left and top right hand corners of the mesh within the comment lines. Consequently the user need only adjust the letters in the arrays in order to modify the model's parameter values.

```
---- Map for grid on level: 1  Bottom left: 0,0  Top right: 700,700 ----
aaaaaaaa
bbbbbbbb
cccccccc
dddddddd
eeeeeeee
ffffffff
gggggggg
hhhhhhhh
---- Map for grid on level: 2  Bottom left: 100,100  Top right: 300,300 ----
bbbbbbbbb
bbbbbbbbb
bbbbbbbbb
bbbbbbbbb
bbbbbbbbb
aaaaaaaaa
aaaaaaaaa
aaaaaaaaa
aaaaaaaaa
---- Map for grid on level: 2  Bottom left: 400,400  Top right: 600,600 ----
abcde
abcde
abcde
abcde
abcde
---- Map for grid on level: 3  Bottom left: 150,150  Top right: 250,250 ----
aaaaaaaaa
aaaaaaaaa
aaaaaaaaa
aaaaaaaaa
aaaaaaaaa
aaaaaaaaa
aaaaaaaaa
aaaaaaaaa
aaaaaaaaa
```

Figure 15 Example map file for the entry of spatial data within a layer

Each letter in the array represents a value of a parameter that is specified in the corresponding code file i.e. the file with the same name but with the *cod* extension. Code files have the same structure as map files, except each of the character arrays is replaced by a set of data defining the values of the letters contained within the array.

The code file which corresponds to this example model and map file is shown in Figure 16. Similarly to the map file, the code file contains four sets of data separated by comment lines. The comment lines contain the information required to identify to which grid each data set relates. Again, ZETUP, the set up program for ZOOMQ3D, produces this file and writes the co-ordinates of the bottom left and top right hand corners of the mesh within the comment lines. Consequently, the user need only adjust the parameter values for each grid.

In Figure 15, eight letters are used in the base grid, so eight parameter values must be defined in the code file for this grid. On the second grid listed in the map file, only two zones are defined (by the letters 'a' and 'b') and consequently only two parameter values must be assigned in the code file for this grid. This relationship is the same for the final two grids in

the map file. Re-iterating, because eight letters are defined on the base grid in the map file, eight corresponding parameter values must be defined in the code file for this grid. The first integer number in each block of data between the comment lines in the code file is the number of zones, or letters, specified in the character array. After this integer value, an equivalent number of parameter values must be specified. Therefore, for example if twenty-six letters were used in the map file for a specific grid, i.e. the letters a to z, twenty-six parameters values must defined within the code file for the corresponding grid.

In the above description it is stated that there must be an equivalent number of parameter values specified in the code file to the number of letters used in the corresponding character array in the map file. However, this is not strictly the case. In fact, there must be *at least* the same numbers of parameters specified in the code file as are used in the map file for the corresponding grid. Therefore it is allowable for example, to specify, parameter values for all fifty-two characters in the range [a, b, ... y, z, A, B, ...Y, Z] but only use a subset of these letters in the map file.

The final rule to follow when editing the map and code files is that, the minimum number of parameters to enter in the code file is determined by the 'highest' letter in the range [a, b, ... y, z, A, B, ...Y, Z]. For example, if C is the 'highest' letter used then twenty-nine parameters values must be defined in the code file. Similarly, if m is the 'highest' letter used then thirteen parameters values are required. This means that if only the two letters 'a' and 'z' are used to define a grid character array in a map file, at least twenty-six parameters values must still be defined in the corresponding code file for the corresponding grid. This is because the parameter values defined in code files must be listed in the order 'a' to 'z' then 'A' to 'Z' i.e. in alphabetical order with the lower case alphabet preceding the upper case alphabet in the fifty-two character scheme.

```
---- Codes for grid on level: 1  Bottom left: 0,0  Top right: 700,700 ----
8
0.10
0.12
0.14
0.20
0.01
0.02
0.03
0.07
---- Codes for grid on level: 2  Bottom left: 100,100  Top right: 300,300 ----
2
0.10
0.20
---- Codes for grid on level: 2  Bottom left: 400,400  Top right: 600,600 ----
5
0.13
0.15
0.09
0.08
0.07
---- Codes for grid on level: 3  Bottom left: 150,150  Top right: 250,250 ----
1
0.05
```

Figure 16 Example code file for the entry of spatial data within a layer

5.3.3 Specifying spatial data using numeric data files

Instead of using a pair of map and code files to specify the spatial variation of a parameter within a model layer, a single data file can be used containing raw numeric data. This has the same name as the map and code files but has the *dat* extension. For example, the file 'porosity04.dat' could be used to specify porosity values in layer 04 of the model (i.e. three layers down from the top layer) instead of the files 'porosity04.map' and 'porosity04.cod'. The numeric data file has the same structure as the map file except that each character array is replaced by an array of numbers representing the parameter values that will be applied at the corresponding nodes of the finite difference mesh within the model layer that the file relates to.

An example numeric data file is shown in Figure 17. This specifies exactly the same model data as that specified in the example above, in which a pair of map and code files are used for defining a hydraulic conductivity distribution.

```
---- Map for grid on level: 1  Bottom left: 0,0  Top right: 700,700 ----
0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10
0.12 0.12 0.12 0.12 0.12 0.12 0.12 0.12
0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14
0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20
0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03
0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
---- Map for grid on level: 2  Bottom left: 100,100  Top right: 300,300 ----
0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20
0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20
0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20
0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20
0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20
0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10
0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10
0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10
0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10
---- Map for grid on level: 2  Bottom left: 400,400  Top right: 600,600 ----
0.13 0.15 0.09 0.08 0.07
0.13 0.15 0.09 0.08 0.07
0.13 0.15 0.09 0.08 0.07
0.13 0.15 0.09 0.08 0.07
0.13 0.15 0.09 0.08 0.07
---- Map for grid on level: 3  Bottom left: 150,150  Top right: 250,250 ----
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
```

Figure 17 Example numeric data file for the entry of spatial data within a layer

5.3.4 Definition of data at coincident points on multiple grids

It is apparent from the example model and input files described above that parameters values at particular finite difference nodes are specified more than once in an input file when the model mesh is locally refined. This occurs at those nodes on a refined grid that are coincident with the nodes on a coarser grid. Where this occurs the data defined on the refined grid will always override that defined on the coarse mesh.

5.4 SPATIAL DEFINITION OF POROSITY

Layers of finite difference nodes in ZOOMQ3D and ZOOPT may abut or may be separated by a *quasi-layer*. This quasi-layer is not represented by a layer of finite difference nodes. Rather, flow in this layer, which for example could represent a low permeability aquitard between two aquifers, is assumed to be vertical and depends on the vertical conductance that is assigned between the numerical layers. In ZOOPT values of porosity must be assigned to all of the finite difference layers and to the quasi-layers. This is achieved though the use of two sets of data files; one set for the finite difference layers and one for the quasi-layers.

Input file names and format

As described above, the porosity of nodes in the finite difference layers must be entered into the model using either,

1. pairs of map and code files named ‘porosity##.map’ and ‘porosity##.cod’ or,
2. numeric data files named ‘porosity##.dat’.

The porosity of nodes in the quasi-layers must be entered into the model using either,

1. pairs of map and code files named ‘porosity##a.map’ and ‘porosity##a.cod’ or,
2. numeric data files named ‘porosity##a.dat’.

The ## symbols must be replaced by a two digit (01 to 99) number representing the layer to which the data files refer. The upper layer is layer 01 and layer numbers are incremented from the top to the bottom of the model. Either a pair of map and code files is required for each of the model layers or a single numeric data file is required for each of the layers. The format of these files is described in Section 5.3. The user specifies whether map and code files or numeric data files are used in the input file ‘entry_method.dat’.

Note that file names relating to the input of porosity in the quasi-layers are appended with the letter ‘a’ after the layer number. The quasi-layer 01a is *below* layer 01 of finite difference nodes. Consequently, there are never porosity input files for a quasi-layer beneath the bottom finite difference layer. The file numbering scheme is illustrated in Figure 18.

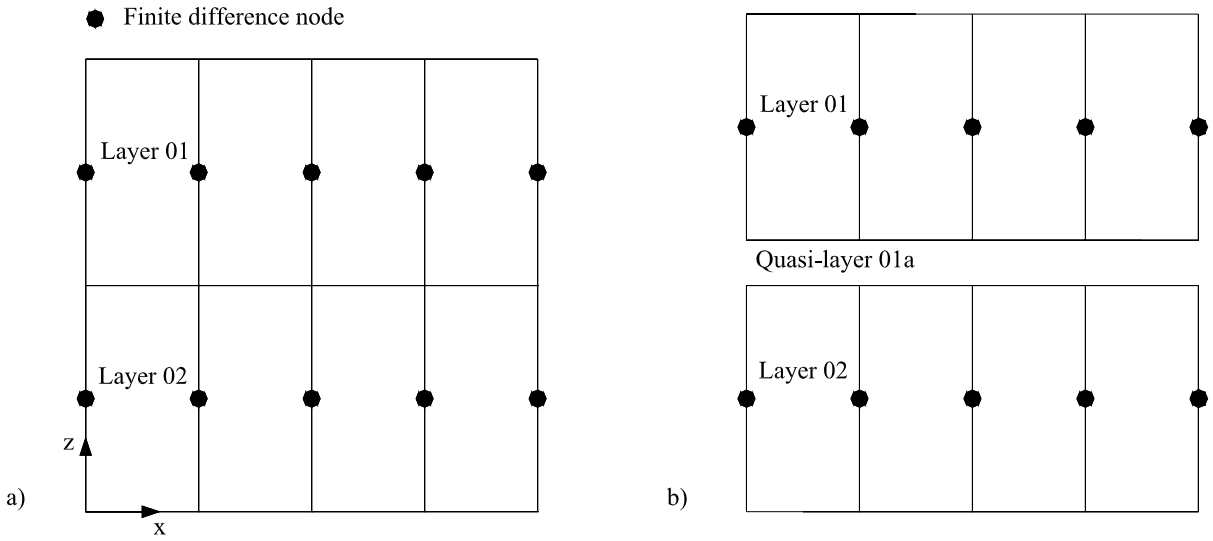


Figure 18 Layer numbering scheme used for specification of porosity in a) a model without quasi-layers and b) a model with quasi-layers

5.5 NODE-BY-NODE HEADS AND FLOWS

Two space delimited ASCII text output files are produced by ZOOMQ3D if the user specifies that particle tracking simulations are to be performed after it has been run. These are named 'heads.txt' and 'flowbal.txt'. The first contains simulated groundwater head values for each model node for each time step of the simulation. The second contains the flow rates (m^3day^{-1}) into and out of each node for each time-step during the simulation. These ZOOMQ3D output files are required by ZOOPT as input. The formats of each of the files are described next.

5.5.1 Output file 'heads.txt'

Heads are contained in this file for each model node at the end of each time-step of the ZOOMQ3D simulation. The heads are written in the following order during the flow simulation:

1. For each time-step of the simulation
2. For each layer in the model. Starting with the top layer (layer 01).
3. For each model grid in the layer. Listed in the same order as they are in the 'grids.dat' input file. The base grid is the first grid listed. Grids are then listed in *grid level* order. Heads are output as an array representing each grid.

This format of this file is illustrated in Figure 19. Some points on a grid may be located outside the model boundary, however, a groundwater head value is written for all elements of the rectangular array, or matrix, of grid points. For the nodes outside the model boundary a dummy head value of –999.0 is output. This head value is also output at those nodes that have de-watered during the time-step.

5.5.2 Output file 'flowbal.txt'

All the components of flow into and out of a node are contained in this file for each model node at the end of each time-step of the simulation. The flows are written in the following order during the ZOOMQ3D simulation:

1. For each time-step of the simulation
2. For each layer in the model. Starting with the top layer (layer 01).
3. For each model grid in the layer. Listed in the same order as they are in 'grids.dat'. The base grid is the first grid listed. Grids are then listed in *grid level* order.
4. Data for each node within the model are written on separate lines. The grids are scanned sequentially by row. Data are written for the nodes in the top row (furthest in the positive y-direction) first and bottom row (furthest in the negative y-direction) last. Within each row nodes are scanned from left to right (in the positive x-direction). Data for a node within a subgrid is only output to the file if it does not also exist on a grid at a coarser grid level i.e. if it is not located on a parent grid line.

Because the data is not structured in grid arrays, dummy flow values relating to nodes outside the model boundary are not written to the file. Table 5 shows the flow components that are listed within each line of nodal data. The first parameter output is an integer flag, which specifies if the node was active during the time-step i.e. if it was wet (1) or dry (0). If the node was dry a zero is written on the line for the node but its flow components are not. Flow data is only output if the node was active (wet) during the time-step.

Time-step 1	Layer 1	Base grid array
		Subgrid 1 array
		Subgrid 2 array
	Layer 2	Base grid array
		Subgrid 1 array
		Subgrid 2 array
	Layer 3	Base grid array
		Subgrid 1 array
		Subgrid 2 array
Time-step 1	Layer 1	Base grid array
		Subgrid 1 array
		Subgrid 2 array
	Layer 2	Base grid array
		Subgrid 1 array
		Subgrid 2 array
	Layer 3	Base grid array
		Subgrid 1 array
		Subgrid 2 array
Time-step 3	Layer 1	Base grid array
		Subgrid 1 array
		Subgrid 2 array
	Layer 2	Base grid array
		Subgrid 1 array
		Subgrid 2 array
	Layer 3	Base grid array
		Subgrid 1 array
		Subgrid 2 array

NB. Example file structure for three time-steps only for a model containing three layers and three grids.

Figure 19 Order of data written in 'heads.txt'

Table 5 Order of variables listed on line of 'flowbal.txt'

	Flow component	
1	Active / inactive integer flag	0 = inactive/dry, 1 = active/wet,
2	Inflow from left (x-direction)	Positive into node
3	Outflow to right (x-direction)	Positive out of node
4	Inflow from below (y-direction)	Positive into node
5	Outflow above (y-direction)	Positive out of node
6	Inflow from base (z-direction)	Positive into node
7	Outflow through top (z-direction)	Positive out of node
8	Specified inflow to node if boundary	Positive into node
9	Increase in storage	Positive values are equivalent to an outflow in terms of the flow balance calculation
10	Recharge	Positive into node
11	River leakage	Positive out of node
12	Leakage	Positive out of node
13	Pumping	Positive into node
14	Spring flow	Positive out of node

Considering that the flow components 2 to 14 in Table 5, are represented by the terms $Q_2, Q_2, ... Q_{13}, Q_{14}$, then the following equation represents the flow balance for a node

$$Q_2 - Q_3 + Q_4 - Q_5 + Q_6 - Q_7 + Q_8 - Q_9 + Q_{10} - Q_{11} - Q_{12} + Q_{13} - Q_{14} = 0$$

5.6 CONSIDERATIONS WHEN UNDERTAKING STEADY-STATE PARTICLE TRACKING RUNS

When the user specifies that the program is to be used to perform a steady-state particle tracking simulation by entering 's' on line 2 of the input file 'zoopt.dat', it reads head and flow data from the files 'heads.txt' and 'flowbal.txt' for a single time-step only. However, if ZOOMQ3D was used to model the steady-state flow field by running time-variantly to steady conditions using multiple time-steps, the data relating to the steady-state conditions, i.e. to the last time-step of the time-variant flow model simulation, will be located at the end of the 'heads.txt' and 'flowbal.txt' files and not at their beginning. In this case *time-instant* tracking must implemented in order to track the particles through the steady-state flow field. This is achieved by entering an 'i' on line 2 of 'zoopt.dat' and by specifying the use of the last time-step's heads and flow data on line 16 of the input file.

6 ZOOPT output files

ZOOPT produces six output files, two giving information on particle tracks, three DXF files used to plot the path lines for visual examination and, an error message file. These are listed in Table 6 and described below.

Table 6 ZOOPT output files

1	tracks.out
2	ptend.out
3	tracks_xy.dxf
4	tracks_xz.dxf
5	tracks_yz.dxf
6	zoopt.err

6.1 OUTPUT FILE 'TRACKS.OUT'

This output file contains information describing the movement of the particles during the tracking process. The file contains six space-delimited columns that contain the data listed below. Each line of the file relates to a single position of a single particle.

1. Particle number
2. x co-ordinate of the particle
3. y co-ordinate of the particle
4. z co-ordinate of the particle
5. Total travel time (days)
6. Total distance moved (m)

6.2 OUTPUT FILE 'PTEND.OUT'

This output file contains information on the starting and finishing positions of each particle and the total distance and time of travel. The file contains nine space-delimited columns that contain the following data listed below. Each line of the file relates to a single particle.

1. Particle number
2. x co-ordinate of the end position of the particle (m)
3. y co-ordinate of the end position of the particle (m)
4. z co-ordinate of the end position of the particle (m)
5. Total travel time (days)
6. Total distance moved (m)
7. x co-ordinate of the start position of the particle (m)
8. y co-ordinate of the start position of the particle (m)
9. z co-ordinate of the start position of the particle (m)

6.3 VISUALISATION OF PARTICLE PATHS

ZOOPT produces dxf files for visualisation of the particle path lines. This is a standard file format used by CAD software which can be viewed using for example, AutoCAD, ArcView / ArcMap, or Surfer. If these commercial products are not available to the user, dxf viewers can be downloaded from the internet.

Three dxf files are produced by ZOOPT which are the same except for the orientation of the Cartesian axes: 'tracks_xy.dxf', 'tracks_xz.dxf' and 'tracks_yz.dxf'. Only one of these files is required if the software package used to open them enables visualisation in three dimensions, such as AutoCAD. If such software is not available, and the files can only be viewed in two dimensions, 'tracks_xy.dxf' shows the path lines in the x-y plane, 'tracks_xz.dxf' shows the path lines in the x-z plane and 'tracks_yz.dxf' shows the path lines in the y-z plane. An example of the three different views in two dimensions is shown in Figure 20. This shows the advective movement of particles towards two abstraction boreholes.

When time-variant particle tracking simulations are performed each particle's path line is drawn using four colours. Each coloured section of the path line represents the distance the particle moved during one time-step of the simulation.

As described in Section 5.1 the user can specify if the model mesh is drawn within the dxf files by adjusting one of the parameters in the 'zoopt.dat' input file. In this case the model mesh is placed in a different *dxf layer* to the particle paths. The dxf layers can be turned on and off when using certain software packages to view the file, for example AutoCAD.

6.4 ERROR FILE 'ZOOPT.ERR'

The file 'zoopt.err' contains a log of errors that are encountered during the particle tracking. This might for example include messages informing the user that the co-ordinates have been incorrectly defined for certain particles and that consequently these are not created.

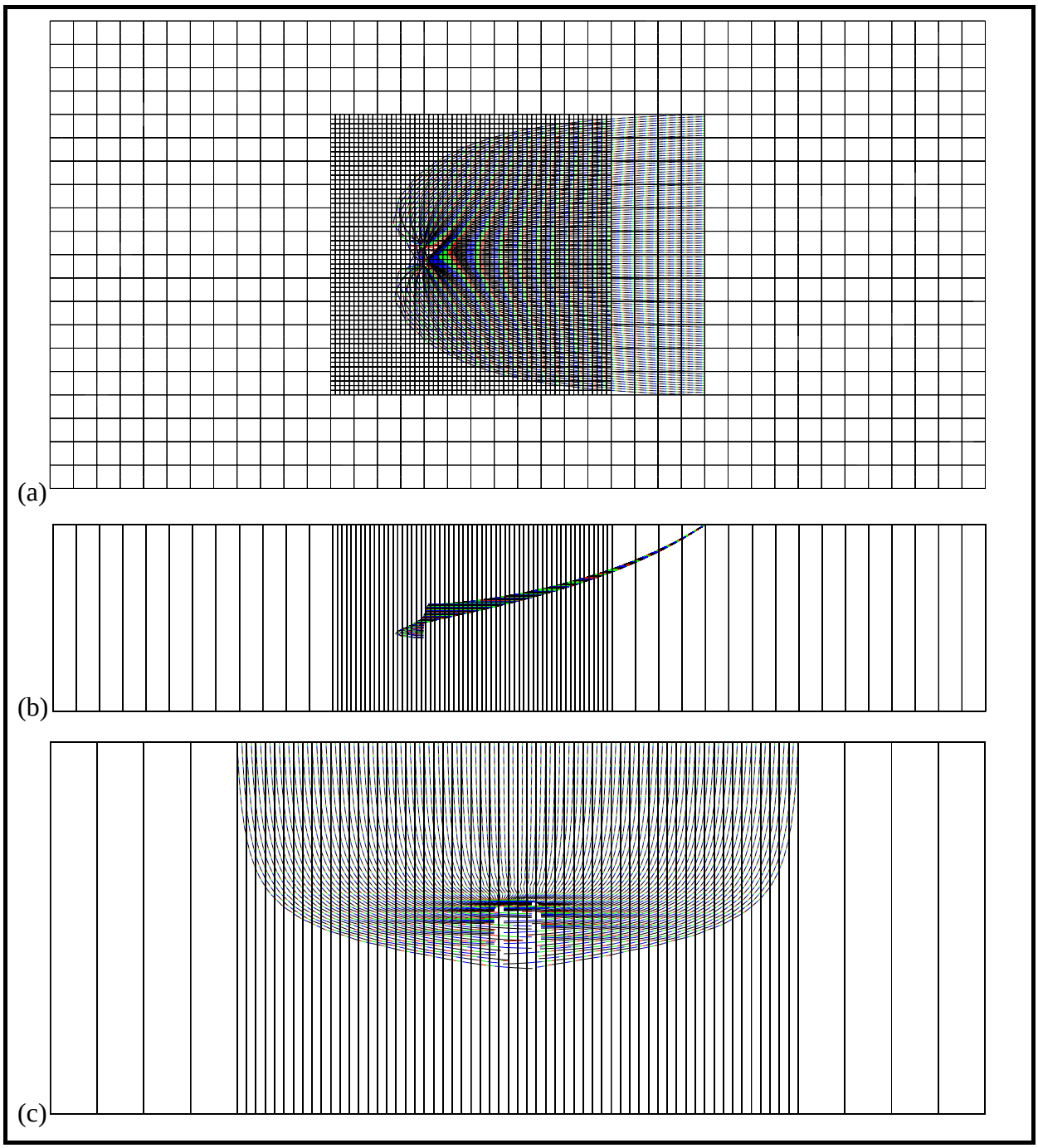


Figure 20 Example dxf files for visualisation of particle path lines in a) x-y plane, b) x-z plane and c) y-z plane

References

BLANDFORD, T.N. AND HUYAKORN, P.S. (1991). WHPA: A modular semi-analytical model for the delineation of wellhead protection areas, version 2.0. Report to the U.S. Environmental Protection Agency, Washington, DC.

FARAHMAND-RAZAVI, A. (1995). Theory and practice of contaminant transport modelling using the random walk. Unpublished Ph.D. Thesis, University of Birmingham.

FRANZ, T. AND GUIGER, N. (1990). FLOWPATH, two-dimensional horizontal aquifer simulation model. Waterloo Hydrogeologic Software, Waterloo, Ontario.

GOODE, D.J. (1990). Particle velocity interpolation in block-centred finite difference groundwater flow models. Water Resources Research, Vol. 26, No. 5, pp 925-940.

JACKSON, C.R. (2001). The development and validation the object-oriented quasi three-dimensional regional groundwater model ZOOMQ3D. British Geological Survey Internal Report IR/01/144.

JACKSON, C.R. (2002a). The representation of the variation of hydraulic conductivity with depth in the object-oriented regional groundwater model ZOOMQ3D. British Geological Survey Commissioned Report CR/02/152N. Environment Agency Technical Report NC/01/38/1.

JACKSON, C.R. (2002b). Steady-state particle tracking in the object-oriented regional groundwater model ZOOMQ3D. British Geological Survey Commissioned Report CR/02/210N. Environment Agency Technical Report NC/01/38/2.

JACKSON, C.R. AND SPINK A.E.F. (2004). User's manual for the groundwater flow model ZOOMQ3D. British Geological Survey Internal Report IR/04/140.

KONIKOW, L.F. AND BREDEHOEFT, J.D. (1978). Computer model of two-dimensional solute transport and dispersion in ground water. USGS Water Resources Investigations, Book 7, Chapter C2.

LU, N. (1994). A semi-analytical method of path line computation for transient finite-difference groundwater flow models. Water Resources Research. Vol. 30, No. 8, 2449-2459.

POLLOCK, D.W. (1988). Semianalytical computation of path lines for finite-difference models. Ground Water, Vol. 26, No. 6.

POLLOCK, D.W. (1989). Documentation of computer programs to compute and display path lines using results from the U.S. Geological Survey modular three-dimensional finite-difference ground-water model. U.S. Geological Survey Open-File report 89-381.

PRICKETT, T.A., NAYMIK, T.G. AND LONNQUIST, C.G. (1981). A random walk solute transport model for selected groundwater quality evaluations. Bulletin 65. Illinois State Water Survey.

SHAFER, J.M. (1987). GWPATH: interactive ground-water flow path analysis. Bulletin 69, Illinois State Water Survey, Champaign, IL.

ZHENG, C. (1989). PATH3D, a groundwater path and travel-time simulator, version 3.0 user's manual. S.S. Papadopolous & Associates, Inc. Bethesda, MD.

ZHENG, C. (1990). MT3D: A modular three-dimensional transport model for simulation of advection, dispersion and chemical reactions of contaminants in groundwater systems. Report to the U.S. E.P.A., Ada, OK.

ZHENG, C. (1994). Analysis of particle tracking errors associated with spatial discretisation. Ground Water, Vol. 32, No. 5, pp 821-828.

ZHENG, C. AND BENNETT, G.D. (1995). Applied contaminant transport modelling. Van Nostrand Reinhold, New York.