# User’s Guide for mod-PATH3DU

A groundwater path and travel-time simulator

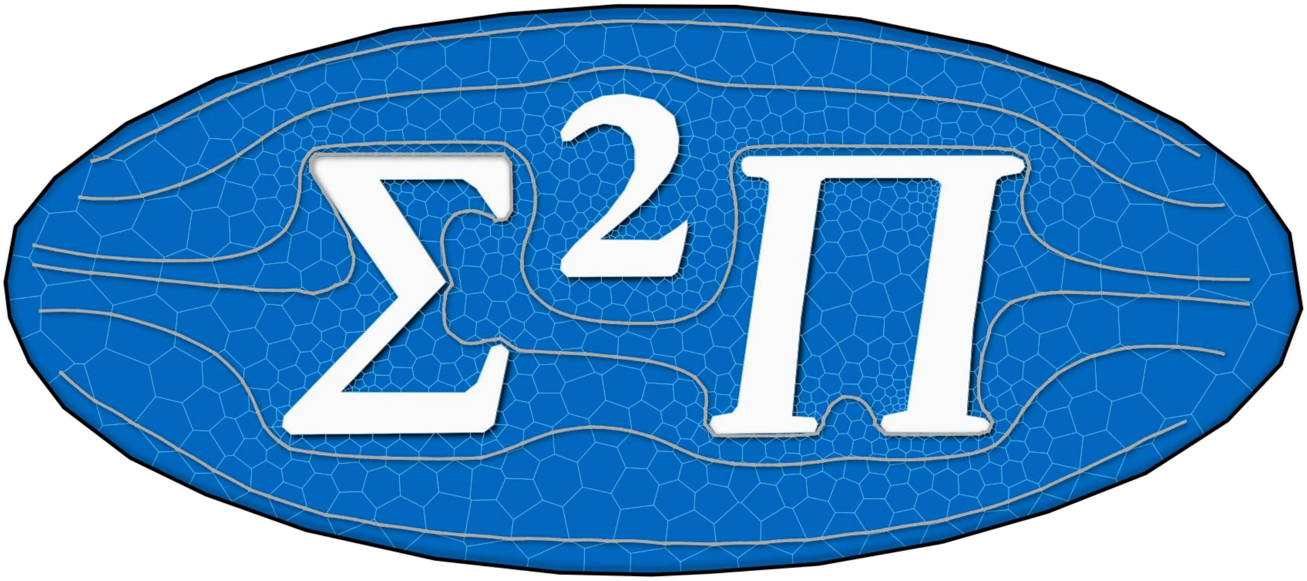
S.S. Papadopulos & Associates, Inc.

UNIVERSITY OF WATERLOO

Christopher Muffels
Xiaomin Wang

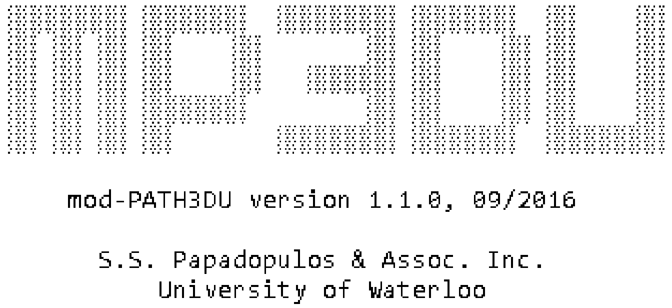
Matthew Tonkin
Christopher Neville

Muhammad Ramadhan
James R. Craig



Bethesda, Maryland
Waterloo, Ontario

2016



```
mod-PATH3DU version 1.1.0, 09/2016

  S.S. Papadopulos & Assoc. Inc.
      University of Waterloo
```

<u>Disclaimer</u>

S.S. Papadopulos & Associates Inc. (SSP&A) and the University of Waterloo (UW) make no representations or warranties with respect to the contents hereof and specifically disclaim any implied warranties of fitness for any particular purpose. Further, SSP&A and UW reserve the right to revise this publication and software, and to make changes from time to time in the content hereof without obligation of SSP&A or UW to notify any person of such revisions or changes.

# Preface to Version 1.1.0

The Waterloo method developed at the University of Waterloo is available as a tracking option in this release. The Waterloo method is a semi-analytical approach, akin to the Pollock method, that is applicable to quad-based and Voronoi grids. See the input instructions for the MPBAS file for details on how to invoke the method.

mod-PATH3DU has been available for a few years now and during this time several issues with v.1.0.0 have been brought to our attention – some big, some small, most of which we have been able to resolve. These experiences, improvements, bug-fixes, etc. constitute this latest release. We appreciate users who were able to share their models (when requested) for particularly difficult problems.

We have changed the format of the Grid-specification file (GSF) that we support (documented herein). The format of that file is general by-design – and too general for mod-PATH3DU. We provide a program, WriteP3DGSF (documented in Appendix A), to convert files in the original GSF format to the mod-PATH3DU-specific format – it will also convert a shapefile. Further, mod-PATH3DU now requires a GSF file be specified in the MPNAM file, regardless of whether a model grid is structured or unstructured. WriteP3DGSF will write a GSF file for structured model grids defined by DELR and DELC to be used by mod-PATH3DU for structured grids.

In the original release of mod-PATH3DU we spent considerable time developing logic to restrict the movement of a particle from one cell to the next. We did this for a variety of reasons, not the least of which is it provided a pragmatic means to account for the cell a particle was in and which it would move to without having to search out its location each time it moved. However, we could never make this logic general enough to work with the variety of models we encountered from users. In this latest release we do away with that cell-restriction and rely on the adaptive time-stepping Runge-Kutta algorithm to control the movement of the particle – as it was intended to do. As a consequence, we needed an efficient algorithm to determine which cell a particle was in. Fortunately, this is a common problem and we found a very fast algorithm that is general and applicable to all the model grid types supported by MODFLOW-USG.

## Preface to Version 1.0.0

This report describes a particle tracking program, called mod-PATH3DU, which supports the latest MODFLOW release by the U.S. Geological Survey, MODFLOW-USG (Panday et al. 2013). mod-PATH3DU currently implements the Pollock method (Pollock, 1994), together with a novel technique, referred to herein as the SSP&A method, for calculating groundwater path and travel-time for unstructured grids. The focus of mod-PATH3DU development to date has been on the logic to map shared faces between cells, and validating the SSP&A tracking algorithm for a variety of simple systems. Future applications of mod-PATH3DU may reveal errors that were not detected during testing. In the event a bug is found please contact the corresponding author listed below; every effort will be made to correct any identified bugs.

# Support

S.S. Papadopulos & Associates Inc. in collaboration with the University of Waterloo developed mod-PATH3DU to support specific project applications, but is distributing it free-of-charge as a service to our industry. As such, only limited technical support can be provided by S.S. Papadopulos & Associates Inc. through the corresponding author.

Corresponding author:
Christopher Muffels
7944 Wisconsin Ave.
Bethesda, MD 20895
301-718-8900
cmuffels@sspa.com
www.sspa.com

## Acknowledgements

The authors would like to thank Mike Kulbersh and the New England District Office of the U.S. Army Corps of Engineers for their support (Contract No. W912WJ-16-P-0040) in developing this latest version of mod-PATH3DU. A little goes a long way – and it is greatly appreciated.

The authors wish to acknowledge Graham Stonebridge, who, as an intern at SSP&A, helped to develop the proof-of-concept for the SSP&A tracking method and Karl Mihm, also a former intern, for his help in writing the first version of this software. Also, Robert Gordon, an intern, for his help testing the Waterloo method and for implementing and testing different location search algorithms.

We wish to thank Jim Rumbaugh, the developer of Groundwater Vistas, for his early adoption of mod-PATH3DU. Also Alan Lemon and Michael Kennard with Aquaveo, developers of GMS for their early adoption and testing. And Wayne Hesch, formerly Senior Project Manager for Visual MODFLOW Flex, for his early testing. We thank Sorab Panday for his support of this development and his tireless pursuit of MODFLOW-USG. We also acknowledge all of our colleagues at S.S. Papadopulos & Assoc. Inc., especially Chunmiao Zheng, Charlie Andrews, Steve Larson, Jim Cousins and Doug Hayes. This software is a company wide effort drawing upon years, decades even, of research and testing in various forms and applications in support of our clients.

# Quick Start Guide

1. Read this manual in its entirety.
2. Download the mod-PATH3DU and related executables from SSP&A's website at www.sspa.com/software.
3. Create the (4) required input files (MPSIM, MPBAS, MPNAM, particle starting location) following the instructions in the Input Instructions section of this manual. Create the optional MP3DU file if desired.
4. Put these 4 (or 5) files, and the mod-PATH3DU executables in the same folder as the MODFLOW-USG model files.
5. Ensure the GSF file is in correct format. If not, convert it using writep3dgsf.exe (downloaded as part of mod-PATH3DU package) – see Appendix A for more details.
6. At a command prompt enter the name of the mod-PATH3DU executable, followed by the name of the MPSIM file. For example : *mod-PATH3DU.exe NAME.MPSIM*
7. Review the log and listing files created, in particular mp3du.log and the MPLST file to ensure the program executed as expected.

To support mod-PATH3DU, SSP&A has, through the mod-PATH3DU webpage, created a simple interface that can be used to generate an initial set of the required (and optional) input files (Step 3). On the webpage users can enter some basic information about their problem (i.e. number of model layers, tracking direction, etc.) and download the corresponding input files. These files can then be modified as needed by the user following the Input Instructions section of this manual.

# Contents

## Figures

## Tables

# Abstract

mod-PATH3DU is a particle tracking code for calculating the three-dimensional flow pathlines of purely advective solute particles. This program is fully compatible with the groundwater simulation model MODFLOW, including MODFLOW-USG, an unstructured-grid version based on an underlying control volume finite difference (CVFD) formulation (Panday et al., 2013).

mod-PATH3DU contains two different algorithms for calculating the local velocity at particle locations required to advect it: the Waterloo method (Ramadhan, 2015), and the SSP&A method. The Waterloo method is based on an analytical Taylor series reconstruction of the local velocity within a cell and is applicable to cells with arbitrary geometry. The SSP&A method is grid independent. It uses universal kriging of a MODFLOW hydraulic-head solution and Darcy's law to calculate the velocity vectors necessary to advect a particle. For both methods, the pathlines are generated using a higher-order Runge-Kutta scheme or a simple Euler approach. Currently, the Pollock method (Pollock, 1988, Pollock 1989) is used to track particles in the *z*-direction. Both the Waterloo and SSP&A methods can be used to track particles on structured and unstructured grids, and users can specify which method to use on a per-cell basis – but it is recommended users default to the Waterloo method. Previous releases of mod-PATH3DU supported the Pollock method - this release does not - however, it is expected a future release will.

The program uses the input file structure of MODPATH6 with some changes detailed in the Input Instructions section. Depending on the type of simulation specified, the program writes different output files, but a summary of the input and simulation results is always generated.

The executable file of mod-PATH3DU is available for download at www.sspa.com/software.

# Introduction

Dr. Sorab Panday, in collaboration with the U.S. Geological Survey (USGS), in 2013, released a version of the groundwater flow model MODFLOW, called MODFLOW-USG, to support a wide variety of structured and unstructured grid types. The USGS particle tracking model, MODPATH6, and its previous versions, can only support groundwater flow simulations for structured grids based on MODFLOW. Therefore, a particle tracking code accommodating the newly developed flow model MODFLOW-USG is required, and mod-PATH3DU has been developed for this purpose.

The main differences between mod-PATH3DU and MODPATH v.6 (MODPATH6) are the algorithms available to calculate velocity and the pathline integration. Currently, two different tracking schemes are implemented in mod-PATH3DU, the Waterloo and SSP&A methods. The semi-analytical particle tracking scheme, the Pollock method, implemented in MODPATH6, evaluates the velocity field using linear interpolation of the groundwater velocities within each finite-difference grid (Pollock, 1989). A particle's path is then computed by moving the particle from one cell to another until it reaches a boundary or meets another termination criterion. For unstructured grids that provide greater flexibility in spatial discretization, the linear velocity interpolation used in the Pollock method is not applicable. Like the Pollock method, the Waterloo method is a semi-analytical approach, but it is applicable to grids with arbitrary cell geometries in the *x* and *y*-directions. The SSP&A method computes the velocity field based on the hydraulic head distribution generated by MODFLOW-USG using kriging. Both methods use a higher order numerical Runge-Kutta scheme (Zheng, 1992) or a simple Euler approach to track a particle within an unstructured cell.

This document describes the particle tracking program mod-PATH3DU and provides instructions for using the program. Some of the required input files are created outside of MODFLOW-USG and it is important to read the instructions to make sure that all of the required input is specified. mod-PATH3DU does not include either a graphical user interface or tools for visualizing the results, although it is expected that the program will be supported by popular modeling interfaces[1]. Users will need to devote care to preparing the input and analyzing the output. All of the files for the test cases detailed in the Examples section are available for download. These files illustrate the structure of the input and the procedures required to execute mod-PATH3DU. mod-PATH3DU is intended to be compatible with MODPATH6 and MODFLOW-USG in terms of input and output formats. The variable names and the corresponding definitions are consistent with the two programs as well. More detailed descriptions of the input parameters are presented in Harbaugh (2005), Panday et al. (2013) and Pollock (2012).

[1] SSP&A has developed 3D visualization software for MODFLOW-USG that fully supports mod-PATH3DU. It is available at www.groundwaterdesktop.com

## Compatibility with MODFLOW

mod-PATH3DU uses information in both the cell-by-cell flow (CBB)[2] and head-save (HDS) files written by MODFLOW to calculate velocity. These files list the components of flow and hydraulic head for each cell. The Waterloo method relies (almost) exclusively on the CBB file, while the SSP&A method requires both. In structured grid mode, MODFLOW-USG writes these two files in a format identical to previous releases. As such, mod-PATH3DU is compatible with previous versions of MODFLOW, specifically MODFLOW-2000 and MODFLOW-2005. It does not support earlier releases.

MODFLOW-USG is built upon the MODFLOW-2005 framework and uses modified versions of its predecessor’s subroutines to read input files. These modified routines are incorporated into mod-PATH3DU. As such, mod-PATH3DU does not support the global (GLO) package available with MODFLOW-2000. Further, the connected-linear network (CLN) package available with MODFLOW-USG supersedes the multi-node well (MNW) package in previous releases of MODFLOW. Because mod-PATH3DU relies on the CBB file for boundary and source/sink flow information, and not the input packages for these routines explicitly, it supports both the MNW and CLN packages. Currently, it does not track particles within a CLN network.

MODFLOW-USG does not require information about cell shapes or how cells are positioned in space (Panday et al. 2013). Spatial information about cells, however, is critical to a particle tracking model. Fortunately, a grid specification file (GSF) was decided upon that provides x, y and z coordinates for the vertices of a cell. mod-PATH3DU requires a modified version of this file for both structured and unstructured grid models. A pre-processing program, writeP3DGSF.exe, is included with mod-PATH3DU and is detailed in Appendix A. This program converts a GSF file in the original format into the mod-PATH3DU specific format. Additionally, it will write the required GSF file for a structured MODFLOW model defined by DELR and DELC. The format for the modified GSF file is provided in the Input Instructions section.

[2] mod-PATH3DU requires the CBB file be written using the COMPACT BUDGET option in the MODFLOW output control (OC) file.

# Background

## Structured Grids

The type of grid available to the original MODFLOW formulation is a structured grid. A structured grid is characterized by regular connectivity. In the case of a structured MODFLOW grid this connectivity is seven-point: each cell in the grid is connected to itself, and the two adjacent cells along each coordinate direction. Each cell in a structured grid can be addressed by index (*i*,*j*) in two-dimensions (2D) or (*i*,*j*,*k*) in three-dimensions (3D), where *i*, *j*, *k* refer to row, column and layer number, respectively. An example of a 2D projection of a 3D structured grid is shown in Figure 1. The grid spacing between each cell in the *x*-, *y*- or *z*- direction is *dx*, *dy* or *dz*. Structured grids may be used in different numerical approximation algorithms of differential equations, for example, the finite element method, the finite volume method, as well as, the finite difference method.

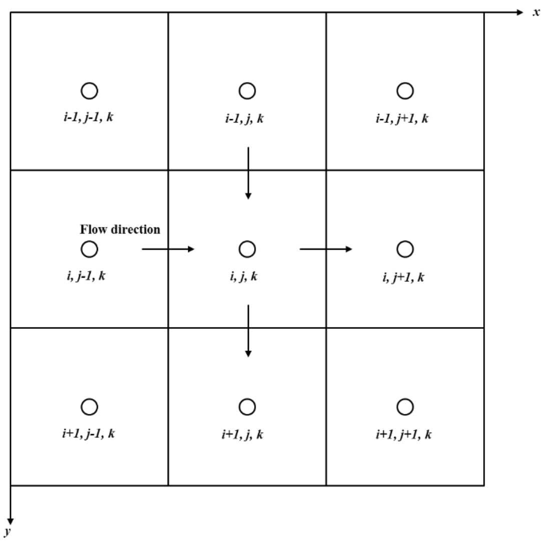


Figure 1 - Example of a 2D structured grid

## Unstructured Grids

An unstructured grid is characterized by irregular connectivity, that is, the number of adjacent cells is variable for each cell. The commonly used elements for this grid type are triangles and quadrilaterals in 2D or tetrahedra and hexahedra in 3D, in irregular patterns. Compared with structured grids, unstructured grids provide better representation of complex boundaries and can facilitate refinement of the spatial discretization of a model in areas of abrupt changes in groundwater velocities, for example in the vicinity of sources and sinks. The tradeoff is that increased computer memory is required to explicitly store neighboring connectivity. Grids of this type may be used with the finite volume or finite element methods. MODFLOW-USG supports a variety of unstructured grid types. Examples of unstructured grid types are shown in Figure 2.

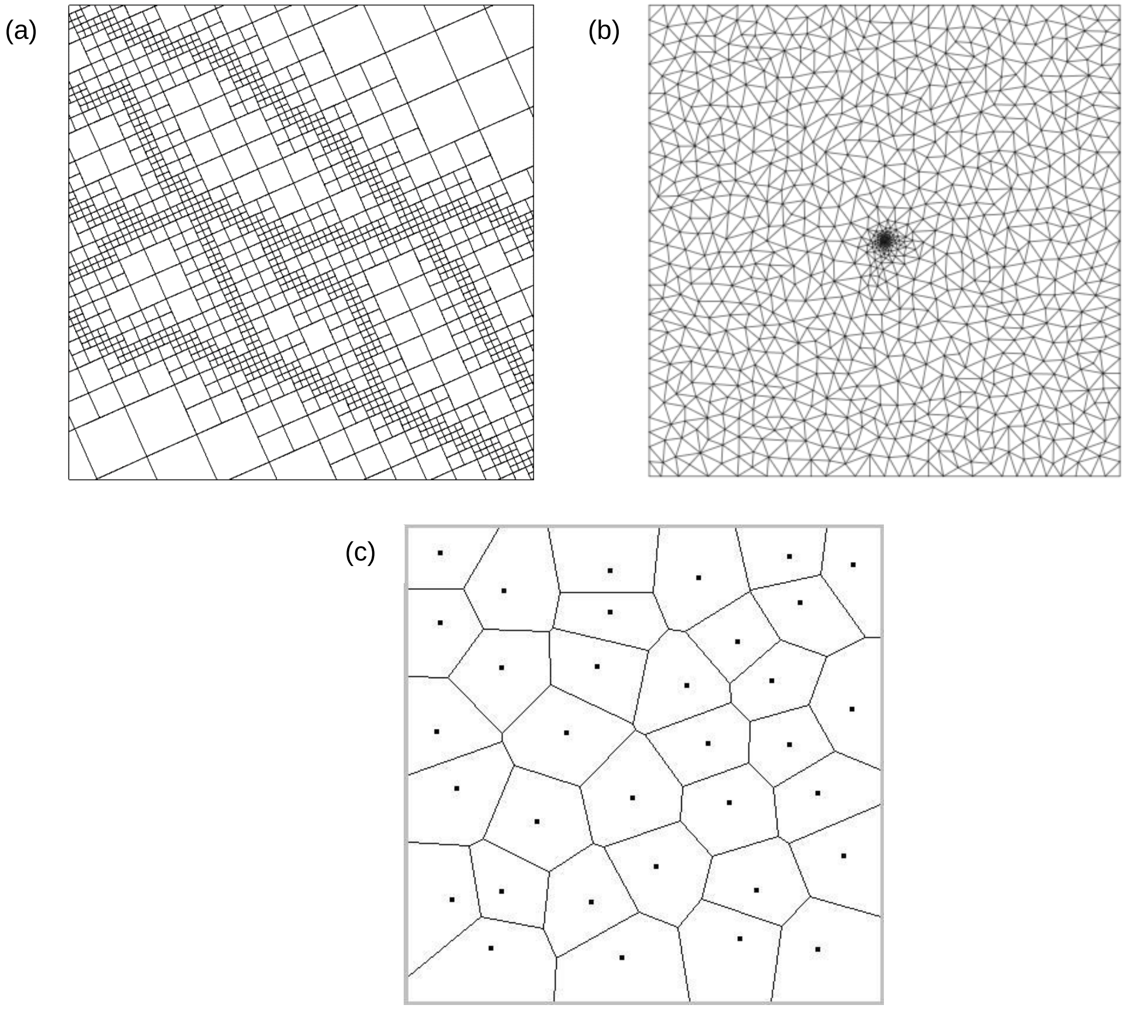


Figure 2 - Examples of unstructured grids: (a) rectangular quadtree, (b) triangular, and c) Voronoi

A stable type of unstructured grid, Voronoi cells, is used widely in the fields of science and technology. In this type of grid, each of the finite difference cells is a Voronoi tessellation of a Delaunay triangulation. An example Voronoi grid is shown in Figure 2c.

# Particle Tracking Algorithms

## The Pollock Method

The Pollock (1989) method is a semi-analytical particle tracking method implemented for structured grids in MODPATH (Pollock, 2012). The principle components of the velocity vector of a particle within an individual cell are computed using linear interpolation based on the velocity components on the cell faces.

$$v_x = A_x(x - x_1) + v_{x_1} \qquad \text{(2-1a)}$$
$$v_y = A_y(y - y_1) + v_{y_1} \qquad \text{(2-2a)}$$
$$v_z = A_z(z - z_1) + v_{z_1} \qquad \text{(2-3a)}$$

where

$$A_x = \frac{v_{x_2} - v_{x_1}}{\Delta x} \qquad \text{(2-2a)}$$
$$A_y = \frac{v_{y_2} - v_{y_1}}{\Delta y} \qquad \text{(2-2b)}$$
$$A_z = \frac{v_{z_2} - v_{z_1}}{\Delta z} \qquad \text{(2-2c)}$$

The components of the velocity of a particle, p, are defined as:

$$v_{xp} = \frac{dx}{dt}; \qquad v_{yp} = \frac{dy}{dt}; \ \ldots \qquad v_{zp} = \frac{dz}{dy}$$

Therefore:

$$(\frac{dx}{v_x})_p = dt \qquad \text{(2-3a)}$$
$$(\frac{dy}{v_y})_p = dt \qquad \text{(2-3b)}$$
$$(\frac{dz}{v_z})_p = dt \qquad \text{(2-3c)}$$

Substituting Equations (2-1) into Equations (2-3) and integrating Equations (2-3) over $[t_1, t_2]$:

$$\int_{x_p(t_1)}^{x_p(t_2)} \frac{dx_p}{A_x(x_p - x_1) + v_{x_1}} = \int_{t_1}^{t_2} dt \qquad \text{(2-4a)}$$
$$\int_{y_p(t_1)}^{y_p(t_2)} \frac{dy_p}{A_y(y_p - y_1) + v_{y_1}} = \int_{t_1}^{t_2} dt \qquad \text{(2-4b)}$$

$$\int_{z_p(t_1)}^{z_p(t_2)} \frac{dz_p}{A_z\left(z_p - z_1\right) + v_{z_1}} = \int_{t_1}^{t_2} dt \tag{2-4c}$$

Since, $A_x$, $A_y$, and $A_z$ are constant within a given finite-difference cell, Equation (2-4a) can be integrated to yield a closed form expression:

$$\ln \frac{A_x[x_p(t_2) - x_1] + v_{x_1}}{A_x[x_p(t_1) - x_1] + v_{x_1}} = A_x \Delta t \tag{2-5}$$

Rearranging Equation (2-5) by substituting $A_x\left[x_p(t_1) - x_1\right] + v_{x_1}$ for $v_{x_p}(t_1)$:

$$x_p(t_2) = x_1 + \frac{1}{A_x}[v_{x_p}(t_1)\exp(A_x \Delta t) - v_{x_1}] \tag{2-6a}$$

Similarly, Equations (2-4) in the *y*- and *z*- directions can be rewritten as:

$$y_p(t_2) = y_1 + \frac{1}{A_y}[v_{y_p}(t_1)\exp\left(A_y \Delta t\right) - v_{y_1}] \tag{2-6b}$$

$$z_p(t_2) = z_1 + \frac{1}{A_z}[v_{z_p}(t_1)\exp(A_z \Delta t) - v_z] \tag{2-6c}$$

The Pollock particle tracking algorithm can be illustrated using a two-dimensional example in the *x*-*y* plane shown in Figure 3. As shown in the figure, the particle trajectory starts at point $(x_p, y_p)$ at time $t_p$ and ends at the particle exiting point $(x_e, y_e)$ at time $t_e$. This example assumes that the velocity components in the *x*- and *y*- directions at each face, $v_{x_1}$, $v_{x_2}$, $v_{y_1}$, and $v_{y_2}$, are all greater than zero. The particle is expected to exit the cell across either face $x_2$ or face $y_2$ depending on the time required for the particle to reach the two downstream faces. The time required to reach face $x_2$ can be computed using Equation (2-5):

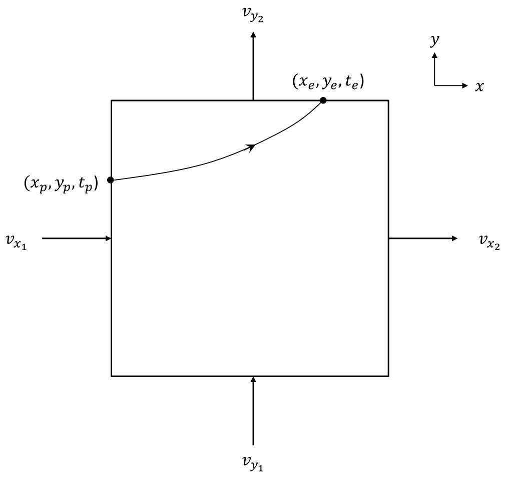


Figure 3 - Example of computing pathlines using the Pollock method

$$\ln \frac{v_{x_2}}{v_{x_p}} = A_x \Delta t_x \tag{2-7}$$

where $A_x\left[x_p(t_2) - x_1\right] + v_{x_1} = A_x[x_2 - x_1] + v_{x_1} = v_{x_2}$

and $A_x[x_p(t_1) - x_1] + v_{x_1} = v_{x_p}(t_p) = v_{x_p}$

Rearranging Equation (2-7) for $\Delta t_x$ yields:

$$\Delta t_x = \frac{1}{A_x}\ln\frac{v_{x_2}}{v_{x_p}} \tag{2-8}$$

Similarly, the time required to reach face $y_2$ is expressed as:

$$\Delta t_y = \frac{1}{A_y}\ln\frac{v_{y_2}}{v_{y_p}} \tag{2-9}$$

The time required for the particle to move from the initial position to the exit point, $\Delta t_e$, is the smaller of $\Delta t_x$ and $\Delta t_y$. If $\Delta t_x$ is smaller than $\Delta t_y$, the particle will leave the cell from face $x_2$; If $\Delta t_y$ is smaller than $\Delta t_x$, the particle will leave the cell from face $y_2$. The corresponding exit coordinates $(x_e, y_e)$ are:

$$x_e = x_1 + \frac{1}{A_x}[v_{x_p}\exp(A_x\Delta t_e) - v_{x_1}] \tag{2-10a}$$

$$y_e = y_1 + \frac{1}{A_y}[v_{y_p}\exp(A_y\Delta t_e) - v_{y_1}] \tag{2-10b}$$

The total elapsed time at which the particle leaves the cell is given by $t_e = t_p + \Delta t_e$. The same procedure is then repeated to move the particle through the next cell until it reaches a discharge point or one of the termination criteria is met (for example, the particle enters a hydraulic sink).

The Pollock method assumes that the velocity field varies linearly across a cell and a constant velocity vector at each cell face is used to represent the local velocity field within a cell. The velocity components of a particle in the *x*-, *y*- or *z*- directions depend only on the local coordinate of the particle's position in that direction. The particle path and travel time within a cell are computed by integration for the finite difference method with structured grids. This method cannot be used for finite-element or integrated finite-difference schemes that have unstructured grids with more complex cell and node connections.

### The Waterloo Method

The Waterloo method is detailed by (Ramadhan, 2015). MORE ON THE METHOD TO COME. Muhammad (Rama) and Dr. Craig are currently working on the paper for their approach – we will update this section once the paper is released.

## The SSP&A Method

The SSP&A method was originally developed by Tonkin and Larson (2002) to track particles and estimate hydraulic capture on a mapping of ground water level data under the influence of pumping. Later it was used by Karanovic et al. (2009) to calculate capture frequency maps. It is a grid independent approach and therefore, sufficiently flexible to be used with either structured or unstructured grids. Unlike the Pollock method, which uses linear velocity interpolation based on the flow rates at cell faces calculated by MODFLOW, the SSP&A method evaluates the velocity components of a particle in *x* and *y*-directions using the interpolated head based on the distribution solved for by MODFLOW. The head interpolation is implemented in the vicinity (blue points) of a particle (black point) using universal kriging, from which the velocity components in the *x*- and *y*- directions (red lines) are calculated according to Darcy’s law (Figure 4).

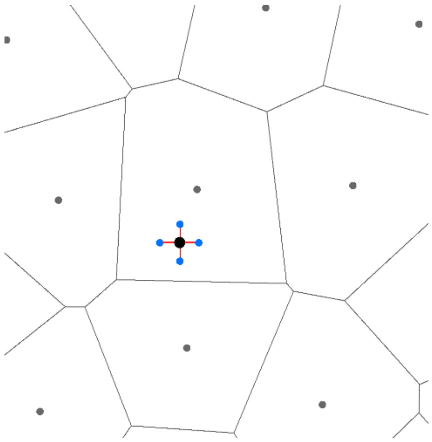


Figure 4 - Schematic of SSP&A method velocity calculation

### *Kriging*

Kriging is often used to interpolate irregularly spaced measurement data to unsampled locations (typically a grid of points suitable for contouring). The values at unknown locations are then estimated by minimizing the error variance of predicted values based on their spatial distribution. In the context of mod-PATH3DU, the heads calculated by MODFLOW that are used to determine velocity constitute the “measured” data, while the position of a particle is the “unsampled” location. Kriging is an exact interpolator in the absence of measurement error or co-located data. Two popular forms of kriging are simple and ordinary. In simple kriging, the mean of the data is assumed to be constant everywhere and its value known *a-priori*. With ordinary kriging, the mean is assumed to be unknown, but is estimable using some function of the measured data. In addition, ordinary kriging can support a spatially varying mean that is not only a function of the data, but includes some trend or “drift”. This form of ordinary kriging is called universal kriging. Universal kriging is a means of incorporating the shape associated with different hydrologic features into the estimate at an unsampled location. For example, in the vicinity of a pumping well, a point sink/source of known strength, derived from the Thiem equation, can be used to incorporate the logarithmic drawdown shape (Tonkin and Larson, 2002). The mathematical details on kriging have been well documented by Journel and Huijbregts (1992), Cressie (1993) and Deutsch and Journel (1998). The kriging algorithm used in the SSP&A method is adapted from the public domain software Geostatistical Software Library (GSLIB) (Deutsch and Journel, 1998) and Skrivan and Karlinger (1980). Use of drift terms with the SSP&A method is invoked using the IFACE variable discussed in the Boundary Packages and Internal Sinks/Sources section.

## Runge-Kutta Numerical Scheme

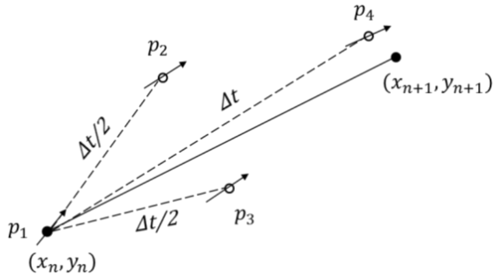


Figure 5 - Illustration of the fourth-order Runge Kutta method

The fourth-order Runge-Kutta scheme used in PATH3D (Zheng, 1989; Zheng, 1992) or a simple Euler approach can be used to advect a particle. To increase the accuracy of the particle tracking calculations relative to a first-order Euler tracking scheme, a higher order of numerical scheme is desired. mod-PATH3DU uses a fourth-order Runge-Kutta method. The basic principle of this method for solving the particle tracking equation is to advance a particle from the initial position $(x_n, y_n, z_n)$ over a time interval $\Delta t$ by combining the information from several trial steps, and then using the information to match a fourth-order Taylor series expansion. The velocity is evaluated four times for each particle tracking step: once at the initial point $(p_1)$, twice at trial midpoints of the step ($p_2$ and $p_3$) and once at a trial endpoint $(p_4)$, as shown in Figure 5. Based on the velocities evaluated at the four points, the position of the particle at the beginning of the next step $(x_{n+1}, y_{n+1}, z_{n+1})$ is calculated as:

$$x_{n+1} = x_n + \frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6} \quad \text{(2-11a)}$$

$$y_{n+1} = y_n + \frac{(l_1 + 2l_2 + 2l_3 + l_4)}{6} \quad \text{(2-11b)}$$

$$z_{n+1} = z_n + \frac{(m_1 + 2m_2 + 2m_3 + m_4)}{6} \quad \text{(2-11c)}$$

where

$$\begin{aligned} k_1 &= \Delta t v_x(x_n,\ y_n,\ z_n,\ t_n) \\ k_2 &= \Delta t v_x(x_n + \frac{k_1}{2},\ y_n + \frac{l_1}{2},\ z_n + \frac{m_1}{2},\ t_n + \frac{\Delta t}{2}) \end{aligned} \quad \text{(2-12)}$$

$$\begin{aligned}
k_3 &= \Delta t v_x(x_n + \frac{k_2}{2},\ y_n + \frac{l_2}{2},\ z_n + \frac{m_2}{2},\ t_n + \frac{\Delta t}{2}) \\
k_4 &= \Delta t v_x(x_n + k_3,\ y_n + l_3,\ z_n + m_3,\ t_n + \Delta t)
\end{aligned}$$

$$\begin{aligned}
l_1 &= \Delta t v_y(x_n,\ y_n,\ z_n,\ t_n) \\
l_2 &= \Delta t v_y(x_n + \frac{k_1}{2},\ y_n + \frac{l_1}{2},\ z_n + \frac{m_1}{2},\ t_n + \frac{\Delta t}{2}) \\
l_3 &= \Delta t v_y(x_n + \frac{k_2}{2},\ y_n + \frac{l_2}{2},\ z_n + \frac{m_2}{2},\ t_n + \frac{\Delta t}{2}) \\
l_4 &= \Delta t v_y(x_n + k_3,\ y_n + l_3,\ z_n + m_3,\ t_n + \Delta t)
\end{aligned} \qquad (2\text{-}13)$$

$$\begin{aligned}
m_1 &= \Delta t v_z(x_n,\ y_n,\ z_n,\ t_n) \\
m_2 &= \Delta t v_z(x_n + \frac{k_1}{2},\ y_n + \frac{l_1}{2},\ z_n + \frac{m_1}{2},\ t_n + \frac{\Delta t}{2}) \\
m_3 &= \Delta t v_z(x_n + \frac{k_2}{2},\ y_n + \frac{l_2}{2},\ z_n + \frac{m_2}{2},\ t_n + \frac{\Delta t}{2}) \\
m_4 &= \Delta t v_z(x_n + k_3,\ y_n + l_3,\ z_n + m_3,\ t_n + \Delta t)
\end{aligned} \qquad (2\text{-}14)$$

The velocity components at points $p_1$, $p_2$, $p_3$ and $p_4$ are calculated using either the Waterloo or SSP&A method. The series of calculations is repeated to move a particle step by step until the particle reaches a discharge point or until a termination criterion is met.

Compared with the Pollock (1994) semi-analytical method, the fourth-order Runge-Kutta method is generally more computationally intensive and may introduce numerical truncation errors. However, the Runge-Kutta method is more general, in that it is not limited to linear interpolation, but applicable to any velocity calculation scheme.

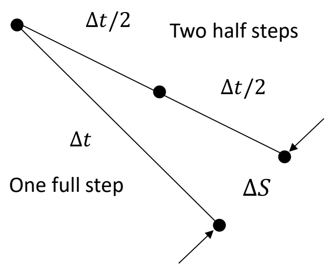


Figure 6 - Illustration of the adaptive step size control procedure

The accuracy of the Runge-Kutta method depends on the tracking step size $\Delta t$. If $\Delta t$ is too large, the calculation of the velocity components may be inaccurate and the particle tracking pathline may divert from the actual flow path. If $\Delta t$ is too small, significant computational effort may be required to move a particle over a given distance. Thus, it is important to determine an appropriate step size $\Delta t$ for the particle tracking process. The adaptive step size control procedure used in the PATH3D code (Zheng, 1989; Zheng, 1992), "step doubling", is also implemented in mod-PATH3DU. The tracking step $\Delta t$ is taken twice: once as a full step and once as two half steps, illustrated in Figure 6. If $\Delta t$ is sufficiently brief for accurate tracking, the difference between the particle endpoint locations calculated with a full step and by taking two half steps, denoted as $\Delta S$, will be small. Because the Runge-Kutta is a fourth-order accurate method, $\Delta S$ can be scaled as $(\Delta t)^5$:

$$(\frac{\Delta t_0}{\Delta t})^5 = \frac{\Delta S_0}{S} \qquad (2\text{-}15)$$

where $\Delta S_0$ and $\Delta S$ are the differences in particle locations with respect to time steps $\Delta t_0$ and $\Delta t$. To estimate the step size $\Delta t_0$, a "safety factor", $f_s$, is assigned in Equation (2-15) and rearranging the equation to yield,

$$\Delta t_0 = f_s \Delta t (\frac{\Delta S_0}{S})^{0.2} \qquad (2\text{-}16)$$

The safety factor has a value slightly smaller than unity (e.g., 0.9). Equation (2-16) is implemented in every tracking step to estimate the required step size adjustment. If the value of $\Delta S$ calculated with respect to the step size $\Delta t$ is larger than the required accuracy specified by $\Delta S_0$, the step size is then reduced to $\Delta t_0$ and the tracking calculation is repeated for that step. If $\Delta S$ is smaller than $\Delta S_0$, the tracking calculation based on the step size $\Delta t$ is acceptable, and the initial step size for the next step is taken as $\Delta t_0$. The difference in particle locations $\Delta S$ is treated as an indicator for the step size adjustment and is a vector in *x*-, *y*- and *z*- directions which can be expressed in terms of a single error criterion, $\varepsilon$, as,

$$\begin{aligned} \Delta X_0 &= \varepsilon \times XMAX \\ \Delta Y_0 &= \varepsilon \times YMAX \\ \Delta Z_0 &= \varepsilon \times ZMAX \end{aligned} \qquad (2\text{-}17)$$

where $XMAX$, $YMAX$ and $ZMAX$ are scaling factors in the three directions and are the maximum lengths of the flow domain in *x*-, *y*- and *z*- directions. Given an error criterion, $\varepsilon$, as specified by the user, the maximum allowed error in all directions can be calculated according to Equation (2-17).

## Boundary Packages and Internal Sinks/Sources

### IFACE

MODPATH computes the velocity vector based on the flow components of the cell faces. It is important that the flux across each cell face include boundary condition flows when appropriate. The auxiliary variable IFACE is the mechanism by which boundary flows are assigned to a particular cell face. For example, consider recharge, its flux is vertical and enters a cell through the top face. If the recharge flux is not assigned to the top face the velocity in the *z*-direction calculated by MODPATH would be incorrect for the upper

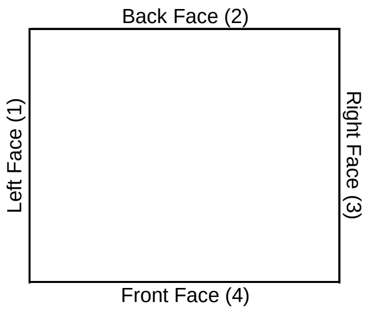


Figure 7 - MODPATH IFACE designation for side faces

layer and a particle could only be reliably tracked within layers below. In MODPATH, IFACE ranges from 0 to 6, where 1 through 4 represent the different side faces of a cell (Figure 7), 5 and 6, denote the bottom and top faces, respectively, and 0 is used to specify an internal sink/source (i.e. flow is not assigned to a face). The default IFACE for cells not explicitly specified is 0.

IFACE can be specified in two ways: 1) using the auxiliary IFACE variable available to MODFLOW, or 2) using the default IFACE specification of MODPATH. By specifying IFACE through MODFLOW, it can be assigned on a per-cell basis, whereas, the default IFACE option of MODPATH specifies IFACE for all cells belonging to a particular boundary package. For example, if the WEL package of MODFLOW is being used to represent both recharge and pumping wells, specifying IFACE through MODFLOW would allow IFACE to be set to 6 for the recharge “wells”, and 0 for the pumping wells. The default IFACE specification in MODPATH would set all cells listed in the WEL package to the desired IFACE value (0 or 6, for example).

**Internal Strong/Weak Sinks/Sources (and UK drift terms)**

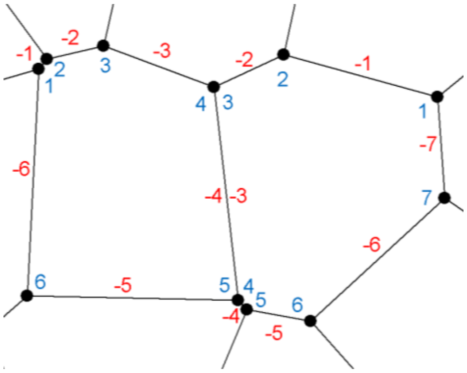


Figure 8 - mod-PATH3DU IFACE designation for side faces

In mod-PATH3DU use of IFACE is maintained to support the Waterloo method and is the mechanism by which drift terms are invoked in the SSP&A method. With mod-PATH3DU, the side face designation is different because the number of faces can vary from cell-to-cell in an unstructured grid. Figure 8 illustrates how side faces (red) are numbered. It is dependent on the order in which the vertices (blue) of a cell are listed in the grid specification file (GSF). The face between the first and second node is assigned -1, between the second and third -2, and so on. Bottom and top faces are denoted in the same way as MODPATH, with 5 and 6, respectively. Specification of an internal sink/source is different. In the vicinity of a pumping well, the Waterloo and SSP&A methods use a local analytic solution to augment its calculation of velocity. IFACE is the mechanism by which this functionality is activated. If IFACE is set to 0 for a boundary cell, it is assumed that boundary is a pumping well and is used in the local analytic approximation. IFACE of 7 is used to indicate an internal sink/source for which the local analytic solution does not apply, for example, a constant-head cell. An excellent discussion of the impact and importance of IFACE on MODPATH results is given by Abrams et al. (2013).

## Special Cases

### Quasi Three-dimensional Representation of Confining Layers

Implicit confining layers are not supported at this time. The program is setup to handle these layers, but at the time of this writing the option was not tested. If there is interest in this option please contact the corresponding author.

### Distorted Vertical Discretization and the Water Table

The manner in which distorted vertical discretization is handled in mod-PATH3DU is identical to MODPATH. In each cell, elevation, $z$, is transformed to a local coordinate system, $z_L$, that ranges from 0 at the bottom of a cell to 1 at the top or water table elevation. By scaling the velocity in the vertical direction by the saturated thickness of the cell, a particle can be tracked in this transformed space provided that $z_L$ is updated when a particle changes layers. The “Non-Rectangular Vertical Discretization” and “Water Table Layers” sections of the MODPATH manual (Version 4) provide excellent discussions of tracking in this transformed space.

### Cross Section Models

To track on a cross section model using the SSP&A method the XSECTION option must be specified in the basic package (BAS) of MODFLOW-USG. This requirement is necessary because the SSP&A method is head based and velocity in the y-direction cannot be accurately calculated for cross section models. When the XSECTION option is used velocity in the y-direction is set to 0. This requirement is not necessary for the Waterloo method.

### Weak/Strong Sinks/Sources

mod-PATH3DU does not currently support the weak sink/source options that are incorporated within MODPATH (Pollock, 1994). The Waterloo and SSP&A methods provide more meaningful results in the vicinity of pumped wells regardless of grid geometry as flow in the vicinity of a well is explicitly incorporated via an analytical correction. Particles are tracked accurately within a cell containing a weak well – see Example 2 for a demonstration.

### Connected Linear Networks (CLN)

Currently, mod-PATH3DU only supports the CLN package if it is used to simulate a well. It does not track particles within networks, however, it is hoped a future release will – please contact the corresponding author if you are interested in this feature. To track particles in the vicinity of a CLN well a default IFACE of 0 must be assigned to the CBB file property “GWF TO CLN” in the MPBAS file (see Input Instructions). Currently, it is assumed all networks in the CLN package represent a pumping well. This assumption will be rectified in a future release – if interested in supporting this development, please contact the corresponding author.

### Ghost Node Correction

If the ghost node correction (GNC) package is used for the flow model – be sure to comment out the package in the NAM file after running MODFLOW and before running mod-PATH3DU. At this time there is an issue processing the GNC package that causes mod-PATH3DU to crash when it is listed in the NAM file. The GNC package does not affect the execution or results of mod-PATH3DU. It is expected this limitation will be remedied in the near term.

# mod-PATH3DU Overview

## Time Concepts

To understand the particle tracking process in mod-PATH3DU, it is necessary to be familiar with the time concepts defined in the program, illustrated in
Figure 9. The time concepts are defined below.

- MODFLOW-USG simulation time: the time generated from the MODFLOW-USG simulation. It starts from zero and increases through to the end of the simulation.
- Reference time: the starting time for the particle-tracking with respect to the start of the MODFLOW-USG simulation.
- mod-PATH3DU simulation time (or tracking time): the mod-PATH3DU simulation time which for forward tracking is defined as the difference between the MODFLOW-USG simulation time and the reference time, and for backward tracking is defined as the difference between the reference time and the MODFLOW-USG simulation time. The tracking time is always a positive value.
- Release time: is the value of time when a particle is released with respect to the mod-PATH3DU simulation time.
- Stop time: the time at which to stop the tracking. For forward tracking, any timesteps after the timestep within which the stop time is will be erased; for backing tracking, any timesteps after the stop time will be completely erased.
- If the *StopOption* equals to 3, mod-PATH3DU will allow specifying a value at which to stop the particle tracking. This time value is not the absolute MODFLOW simulation time, but a relative value with respect to the reference time:
  *The absolute stop time = reference time + stop time (forward tracking);*
  *The absolute stop time = reference time – stop time (backward tracking).*

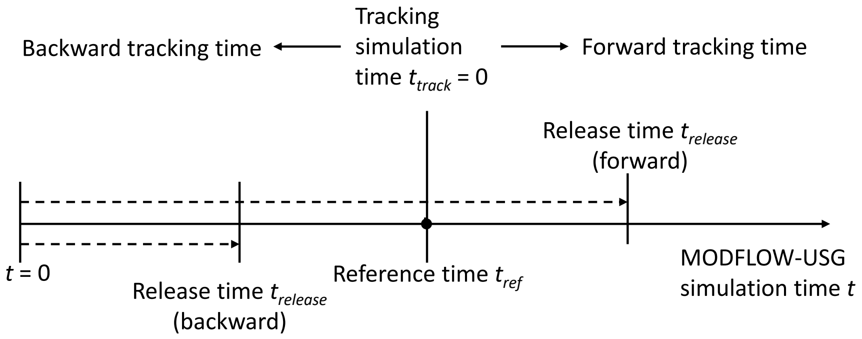

Backward tracking time
Tracking simulation time $t_{track}$ = 0
Forward tracking time
Release time $t_{release}$ (forward)
$t$ = 0
Release time $t_{release}$ (backward)
Reference time $t_{ref}$
MODFLOW-USG simulation time $t$

Figure 9 - Illustration of time concepts in mod-PATH3DU

## Velocity Algorithm Selection

Currently, mod-PATH3DU provides two algorithms to calculate velocity: the SSP&A method, and the Waterloo method. This section provides guidance on which scheme to use for a particular application. mod-PATH3DU is quite flexible in that it allows users to choose which velocity algorithm to use on a per-cell basis. As such, one method is not required for an entire application, rather the choice of method is dictated by the computational efficiency of each method and grid and flow characteristics in different parts of the groundwater model domain. In this release support for the Pollock method has been disabled because issues arose when using the GSF file to specify cell vertices. We expect to support it again, but the focus has been on the development of the Waterloo method for the last two years. Thus the choice of algorithm in this version is not so nuanced – we recommend the Waterloo method – however, the SSP&A algorithm, currently, is much faster, and so when it is applicable it is advantageous to use it. Both methods have limitations and understanding these is critical to choosing the appropriate algorithm for a problem, however the Waterloo method is much more robust – it has all the advantages of the SSP&A method with many limitations.

## Limitations

### Waterloo Method

The Waterloo method, because it is semi-analytical, has similar limitations to the Pollock Method. For a detailed description of the Waterloo Method, including limitations, see Ramadhan (2015). This section will be updated at a later time.

### SSP&A Method

The applicability of the SSP&A method to tracking problems is limited by 1) the assumptions in the underlying tracking scheme, 2) the interpolation method it employs, and 3) limitations in the groundwater flow model, including discretization and boundary effects.

The accuracy of the Runge-Kutta scheme depends on the tracking step size $\Delta t$. If $\Delta t$ is too large, the calculation of the velocity components may be inaccurate and the particle tracking pathline may divert from the actual flow path. However, mod-PATH3DU mitigates this error by implementing the adaptive step size control procedure used in PATH3D called "step doubling". For more information on the Runge-Kutta scheme please see Zheng (1989;1992;1994) and Zheng and Bennett (2002).

The accuracy of the universal kriging interpolation is dictated by (a) grid discretization, (b) severe heterogeneities, and (c) proximity to certain boundaries. The refinement obtained by adding more cells spaced closer together will provide the approach with better information to calculate velocity.

Nevertheless, the use of universal kriging can overcome many discretization and boundary effects provided an appropriate drift term is used. However, because the coefficients of these drift terms are determined through a regression, they can be made zero or their sign reversed (although this is unlikely). For example, when pumping in the presence of a strong regional gradient, the pumping well signal (its contribution to the hydraulic-head in a cell) may be overwhelmed by the signal from the regional gradient and thus made zero in the regression. Drift terms to overcome hydraulic conductivity dichotomies between two cells and to account for additional boundaries, including no-flow and streams/rivers, are possible, but are not yet available in this version of mod-PATH3DU.

Because the Pollock method is used to calculate velocity in the *z*-direction vertical sub-discretization is not supported at this time. Further, the cell structure in each layer must be identical.

# Input Instructions

## Input Data Format

mod-PATH3DU supports the input and output file formats of MODPATH6 detailed by Pollock, 2012. These files include the simulation, name, basic, and pathline files. However, with MODPATH6, some of the required information is already listed in MODFLOW-USG input files. All references to duplicate information are removed from the input files in mod-PATH3DU. Detailed input directions for each file are provided in this section.

Input data are read free format: the spacing of values for each record is not fixed and between two records a SPACE or COMMA can be used for separation. It is important to specify all input data items explicitly. Apostrophes are not required for character data items if the character data item is the only record on a line; otherwise, if multiple records on a line and a character data item contains a space, apostrophes must be used for data specification. For reading array input, mod-PATH3DU uses the same array reader subroutines (U2DREL, U2DINT, and U1DREL) used by MODFLOW (Harbaugh, 2005); detailed information can be found in Harbaugh (2005).

## Tracking Simulation File (MPSIM)

The mod-PATH3DU simulation file contains information for defining a particle tracking problem and specifying the tracking method. The name of the simulation file can be specified on the mod-PATH3DU command line.

```
(1)   Example.mpnam
(2)   Example.mplist
(3)       2       1       1       1       2       3       2       1       2       2       1       1
(4)   Example.ept
(5)   Example.ptl
(6)           1           1       1.000
(7)       365.0
(8)   Example.ptr
(9)           0
```

Figure 10 - Example MPSIM file

Table 1 - MPSIM input instructions

| Item | Variable[1] | Type[2] | Options / Values | Description |
|---|---|---|---|---|
| 0 | [HEADER] | C | #TEXT | Header. |
| 1 | MPNAMFile | C | - | Name of the MPNAM file. |
| 2 | MPLSTFile | C | - | Name of the mP3DU listing file. |
| 3 | SimulationType | I | 2 - Pathline simulation | A flag indicating the type of particle-tracking simulation. |
| | TrackingDirection | I | 1 - Forward tracking<br>2 - Backward tracking | A flag indicating the direction of the particle tracking computation. |
| | WeakSinkOption | I | 1 - Allow particles to pass through weak sinks | Flag indicating how weak sinks are to be treated. Only used with Pollock method. |
| | WeakSourceOption | I | 1 - Allow particles to pass through weak sources | Flag indicating how weak sources are to be treated. Only used with Pollock method. |
| | ReferenceTimeOption | I | 1 - Specify a value for reference time<br>2 - Specify stress period, time step and relative time position within the time step to use to compute the reference time | A flag indicating how reference time will be specified. |
| | StopOption | I | 1 - Stop at the end (forward tracking) or | A flag indicating how the particle |

| | | | | |
|---|---|---|---|---|
| | | | beginning (backward tracking) of the MODFLOW-USG simulation<br>2 - Track until all particles reach their termination points<br>3 - Specify a value of tracking time at which to stop | tracking simulation should be terminated. |
| | ParticleGenerationOption | I | 2 - Read particle locations from an external file | A flag indicating how particle starting locations are generated. |
| | TimePointOption | I | 1 - Time points are not specified | Ignored. |
| | BudgetOutputOption | I | 1 - No budget checking | Ignored. |
| | ZoneArrayOption | I | 1 - Zone data are not supplied | Ignored. |
| | RetardationOption | I | 1 - Retardation factors are not read or used in the velocity calculations | Ignored. |
| | AdvectiveObservationOption | I | 1 - Advective observations are not computed or saved | A flag indicating if advective observations are computed and saved as output. |
| 4 | EndpointFile | C | - | File name for the endpoint file (ignored). |
| 5 | PathlineFile | C | - | File name for the pathline file. |
| 6 | [ReferenceTime] | R | Reference time | If ReferenceTimeOption = 1 |
| | | I | Stress period | If ReferenceTimeOption = 2 |
| | | I | Time step | If ReferenceTimeOption = 2 |
| | | R | Relative time (between 0 and 1) | If ReferenceTimeOption = 2 |
| 7 | [StopTime] | R | Stop time | If StopOption = 3 |
| 8 | StartingLocationsFile | C | - | File name of the particle starting locations file. |
| 9 | StopZone | I | 0 - Particles should not be stopped at a user specified zone | Ignored. |

[1] Square brackets indicate optional or dependent variables

[2] C Character
I Integer
R Real

# Tracking Name File (MPNAM)

The name file contains the names of input files that are used by mod-PATH3DU. Currently, the files required by mod-PATH3DU are (1) the MODFLOW-USG name file, (2) mod-PATH3DU basic file, and (3) the grid specification file if MODFLOW-USG was executed in “unstructured” mode.

```
(1)    MFNAM                        11   Example.nam
(1)    GSF                          12   Example.gsf
(1)    MPBAS                        13   Example.mpbas
(1)    MP3DU                        14   Example.mp3du
```

Figure 11 - Example MPNAM file

Table 2 - MPNAM input instructions

| Item | Variable[1] | Type[2] | Options / Values | Description |
|---|---|---|---|---|
| 0 | [HEADER] | C | #TEXT | Header. |
| 1[3] | FileType | C | MFNAM - MODFLOW-USG NAM file<br>GSF - Grid specification file<br>MPBAS - mod-PATH3DU basic data file<br>MP3DU – mod-PATH3DU settings file | Type of the file. |
|  | FileUnit | I | Greater than 10 | Unique unit number to be assigned to the file. |
|  | FileName | C | - | Name of the file. |

[1] Square brackets indicate optional or dependent variables

[2]
C Character
I Integer
R Real

[3] Repeat for each file to be listed

## Tracking Basic Package (MPBAS)

The mod-PATH3DU basic file contains additional data that is dependent on the MODFLOW-USG grid and boundary conditions. The name of the MPBAS file is specified in the MPNAM file.

```
(1)                    1
(2)     RECHARGE
(3)     6
(4)     CONSTANT                     0.200   Porosity Layer 1
(4)     CONSTANT                     0.200   Porosity Layer 2
(5)     CONSTANT                         2   MTH Layer 1
(5)     CONSTANT                         2   MTH Layer 2
```

Figure 12 - Example MPBAS file

Table 3 - MPBAS input instructions

| Item | Variable[1] | Type[2] | Options / Values | Description |
|---|---|---|---|---|
| 0 | [HEADER] | C | #TEXT | Header. |
| 1 | DefaultIFACECount | I | Greater than or equal to 0 | The number of budget items for which a default IFACE is specified. |
| | | | Repeat items 2 and 3 DefaultIFACECount times | |
| 2 | BudgetLabel | C | CONSTANT HEAD<br>WELLS<br>RECHARGE<br>etc. | Text label used in the MODFLOW-USG budget file. The mod-PATH3DU log reports these for each stress period if in doubt. |
| 3 | DefaultIFACE | I | less than 0 - Flow is assigned to the corresponding cell face (see IFACE section)<br>0 - Flow is treated as an internal sink or source, point-logarithmic drift activated for this cell<br>5 - Flow is assigned to the bottom face of the cell<br>6 - Flow is assigned to the top face of the cell<br>7 – Flow is treated as an internal sink or source | The value of IFACE to be assigned to this BudgetLabel |
| 4[3] | Porosity | R | Between 0.0 and 1.0 | Porosity. Read using U2DREL. |
| 5[3] | MTH | I | 1 - Pollock Method<br>2 - SSP&A Method | The particle tracking algorithm to use. Read using U2DINT. |

1 Square brackets indicate optional or dependent variables

2 C Character
I Integer
R Real

3 Repeat NLAY times

## mod-PATH3DU Tracking Options Package (MP3DU)

The MP3DU package is an optional input file that can be used to control tracking options, specifically related to the SSP&A method. It uses a keyword input style as detailed below. The keywords are optional and can be listed in any order. If a keyword is not present the default value is used for that option.

```
(1)    TRACK_TYPE                        RK4
(2)    STEP_ERROR                   1.00E-06
(3)    DT_INIT                      1.00E+01
(4)    DT_MAX                       1.00E+06
(5)    WELL_CAPTURE_RADIUS          1.00E+01
```

Figure 13 - Example MP3DU file

Table 4 – MP3DU file input instructions

| Item | Variable[1] | Type[2] | Options / Values | Description |
|---|---|---|---|---|
| 1 | [TRACK_TYPE] | C | “RK4” (Default)<br>“EULER” | Tracking method. Default is “RK4”. |
| 2 | [STEP_ERROR] | R | - | Adaptive stepsize error term, $\varepsilon$, in equation 2-17. Default is 1.00E-06. Larger values will result in a faster runtime with, potentially, less accurate paths. To turn-off the adaptive step size option make STEPERROR large (1.00E+06). |
| 3 | [DT_INIT] | R | - | Initial step size. Default is 10. |
| 4 | [DT_MAX] | R | - | Maximum step size. Default is 1.00E+06. |
| 5 | [WELL_CAPTURE_RADIUS] | R | - | When FORWARD tracking using the adaptive time step option (STEP_ERROR < 1), WELL_CAPTURE_RADIUS is the radial distance from the well, within which, a particle is considered captured. |

[1] Square brackets indicate optional or dependent variables

[2]

| | |
|---|---|
| C | Character |
| I | Integer |
| R | Real |

If a simple Euler scheme is desired make STEP_ERROR large (1.00E+06) and set an appropriate maximum time step size.

## Particle Starting Locations File

The mod-PATH3DU particle starting locations file contains information on the time and locations of starting particles. The format for this file has not been finalized and may change in future releases. A particle starting location has two facets: 1) temporal options, and 2) spatial options. As mod-PATH3DU continues to develop, the plan is to evolve the format of this file (and corresponding output files) in such a way as to make typical tracking exercises more automated and therefore, user friendly. For example, a future release will have a "capture zone" starting option. The user will simply indicate wells or stream segments and mod-PATH3DU will execute and write a polygon-shapefile of the capture zone for each of the listed elements.

```
(1)   GRID2D
(2)                 1                0.0
(5)                 1                0.5
(6)            1000.0             1000.0            100.0
(7)            1000.0             1000.0            100.0
(1)   WELL2D
(2)                 1                0.0
(5)                 6                0.1
(8)             100.0              100.0               10              20.0
(1)   OTHER
(2)                 1                0.0
(9)              10.0               10.0                1               0.5
(9)              20.0               20.0                1               0.5
(9)              30.0               30.0                1               0.5
```

Figure 14 - Example particle starting locations file

Table 5 - Particle starting locations input instructions

| Item | Variable[1] | Type[2] | Options / Values | Description |
|---|---|---|---|---|
| | | Repeat as needed (unless SpatialOption = OTHER, see note 3) | | |
| 1 | SpatialOption | C | "GRID2D" - Release particles in a regular grid pattern for a specified layer<br>"WELL2D" - Release particles at some radius around a well for a specified layer<br>"OTHER"[3] - Release particles at listed global X, Y coordinates | Release option. |
| 2 | TemporalOption | I | 1 - A single ReleaseTime will be used for all particles<br>2 - Particles will be released | A flag indicating whether a single or multiple release events will be used for particles. |

| | | | | | |
|---|---|---|---|---|---|
| | | | ReleaseEventCount times every ReleasePeriodLength<br>3 - Particles will be released ReleaseEventCount times at the specified ReleaseTimes | | |
| | ReleaseTime | R | Greater than 0.0 | Release time of particles relative to mod-PATH3DU tracking time. | |
| 3 | [ReleaseEventCount] | I | Greater than 0 | If TemporalOption = 2 or 3<br>The number of release events. | |
| | [ReleasePeriodLength] | R | Greater than 0.0 | If TemporalOption = 2<br>The time interval between particle release events. | |
| 4 | [ReleaseTimes] | R | Greater than 0.0 | If TemporalOption = 3<br>ReleaseEventCount release times relative to mod-PATH3DU tracking time. | |
| 5 | [ReleaseLayer] | I | 1 - NLAY | Layer number. | If SpatialOption is not OTHER |
| | [RelZ] | R | 0.0 - 1.0 (bottom - top or WT) | Local Z coords. | |
| 6 | [XMIN] | R | - | If SpatialOption = GRID2D | |
| | [XMAX] | R | - | | |
| | [DX] | R | - | | |
| 7 | [YMIN] | R | - | If SpatialOption = GRID2D | |
| | [YMAX] | R | - | | |
| | [DY] | R | - | | |
| 8 | [X] | R | - | If SpatialOption = WELL2D | |
| | [Y] | R | - | | |
| | [NumberOfParticles] | I | - | | |
| | [Radius] | R | - | | |
| 9 | [X] | R | - | If SpatialOption = OTHER | |
| | [Y] | R | - | | |
| | [Layer] | I | - | | |
| | [Relz] | R | - | | |

1 Square brackets indicate optional or dependent variables

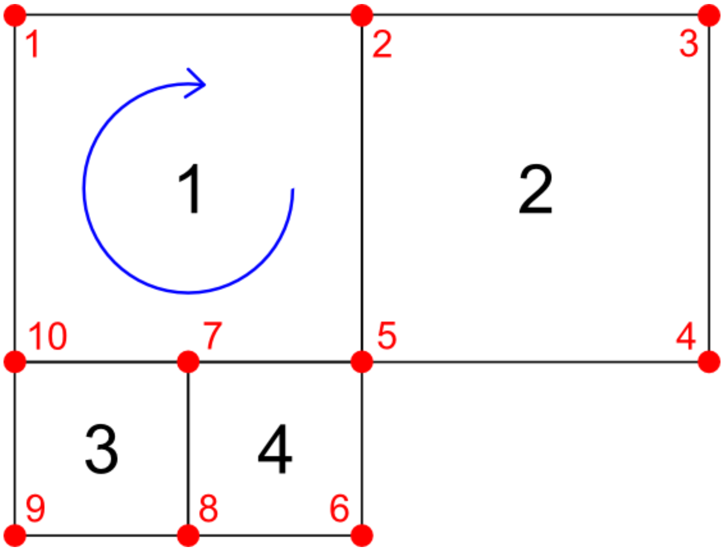
2 C Character
I Integer
R Real

3 If SpatialOption = OTHER it is assumed that, following entry of item 2, all remaining lines in the file are item 9

## Grid Specification File (GSF)

The grid specification file provides x and y coordinates for the vertices of each cell in a MODFLOW-USG grid. The format of this file for mod-PATH3DU v.1.1.0 is a modified form of the original format required for previous mod-PATH3DU releases. The mod-PATH3DU-specific GSF format is less general than the original format – the restrictions are:

1. Duplicate vertices are not listed. For example, between two identical square cells, for example 1 and 2 in the figure below, there are two shared nodes, 2 and 5, those shared nodes are listed only once – and referenced by both cells.
2. The z-coordinate is not needed when listing vertex coordinates. mod-PATH3DU gets elevations from the discretization and head-save files.
3. For each cell, the vertex nodes must be listed clockwise. It does not matter which node is listed first for a cell, but the subsequent nodes must be listed clockwise from that one. For example, for cell 2 in the figure below, its nodes are 4,5,2,3 or 2,3,4,5 etc.
4. For quad-based grids – if a cell has more than one connection on a side, all of the shared vertices must be listed for that cell. For example, for cell 1 in the figure below, its nodes are 1,2,5,7,10 – node 7 must be included.
5. For each vertex pair making a cell "face", list the connected cell that shares that face. For example, for cell 1, if the vertices are listed as 5,7,10,1,2, the cell that shares the face given by vertices 5 and 7 is 4, 7 and 10 is 3, 10 and 1 a -999 is used to indicate there is not a connected cell, the same for 1 and 2, and between 2 and 5 is 2.



At this point, most of the popular MODFLOW interfaces support the original GSF file and it is expected they will support this mod-PATH3DU specific format. Regardless, a utility is provided with mod-PATH3DU that will convert a GSF file in the original format into the mod-PATH3DU specific one. This program is described in Appendix A of this manual. This program, WriteP3DGSF, will:

1. Check for and exclude duplicate nodes – its tolerance is 1.e-4; any nodes within a distance of 1.e-04 units are considered identical.
2. Remove any reference to z-coordinates.
3. List vertices for each cell clockwise.
4. Ensure all vertices are listed for any cells with a side connected to more than one other cell. For example, in the figure above, if the GSF in the original format only lists nodes 1,2,5,10 for cell 1, WriteP3DGSF will automatically add node 7 to those listed for this cell. It does this by buffering each cell and identifying additional nodes inside this buffer.
5. Determines the connected cell that shares each face of a cell.
6. Write a new GSF file in the required format.

The mod-PATH3DU GSF file, corresponding to the figure above, is given in Figure 15.

```
(1)  mod-PATH3DUv110
(2)                10
(3)                 1       0.0       7.5
(3)                 2       5.0       7.5
(3)                 3      10.0       7.5
(3)                 4      10.0       2.5
(3)                 5       5.0       2.5
(3)                 6       5.0       0.0
(3)                 7       2.5       2.5
(3)                 8       2.5       0.0
(3)                 9       0.0       0.0
(3)                10       0.0       2.5
(4)     1   2.5   5.0     5     5     7    10     1     2     4     3  -999  -999     2
(4)     2   7.5   5.0     4     4     5     2     3          -999     1  -999  -999
(4)     3  1.25  1.25     4     8     9    10     7          -999  -999     1     4
(4)     4  3.75  1.25     4     6     8     7     5          -999     3     1  -999
```

Figure 15 - Example GSF file

Table 6 - GSF input instructions

| Item | Variable[1] | Type[2] | Options / Values | Description |
|---|---|---|---|---|
| 1 | Version | C | "mod-PATH3DUv110" | Version of the file. It is possible this file will change over time and this will allow for backward compatibility. |
| 2 | NVERT | I | - | Total number of nodes or vertices. |
| Repeat Item 3 NVERT times | | | | |
| 3 | NodeID | I | - | Node ID |
| | XVertex | R | - | X-coordinate of vertex |
| | YVertex | R | - | Y-coordinate of vertex |
| Repeat Item 4 for every model cell | | | | |
| 4 | CellID | I | - | Id of cell. |
| | XCellCenter | R | - | X-coordinate of cell center. |
| | YCellCenter | R | - | Y-coordinate of cell center. |
| | NumberOfVertices | I | - | The number of vertices that define this cell. For example, for a square cell, the number of vertices is 4. |
| Repeat VertexId on line 4 NumberofVertices times. The vertices must be listed in clockwise order. | | | | |
| | VertexID | I | - | The NodeID of the vertex. |
| Repeat ConnectedCellId on line 4 NumberofVertices times. Listed in the order corresponding to the VertexId. | | | | |
| | ConnectedCellId | I | - | The CellID that shares the cell face for each vertex pair. -999 indicates no connected cell. |

[1] Square brackets indicate optional or dependent variables

[2]
C Character
I Integer
R Real

# Output File Formats

mod-PATH3DU writes a variety of output files. Primarily these files are listing or log files that report particle tracking options, MODFLOW-USG input, and progress. Two tracking related files, an endpoint and pathline are also written similar to MODPATH. These files are detailed herein.

## Endpoint File

The endpoint file summarizes the starting and ending locations for each particle. This file has convenient column headers to facilitate its use.

Table 7 - Endpoint file format

| Item | Variable[1] | Type[2] | Options / Values | Description |
|---|---|---|---|---|
| 1 | Header | C | - | Descriptive Column Header |
| | | | Item 2 is listed for each particle | |
| 2 | Zone_Code_Final_Loc | I | - | Unused – 1 is always entered. |
| | Filler_Ignore | I | - | Unused - -999 is always entered. |
| | CellID_Final | I | - | ID of the cell containing the termination point of the particle path. |
| | K_Final | I | - | Layer of the particle termination point. |
| | Global_X_Final_Loc | R | - | Global X location of the final point. |
| | Global_Y_Final_Loc | R | - | Global Y location of the final point. |
| | Global_Z_Final_Loc | R | - | Global Z location of the final point. |
| | Total_Tracking_Time | R | - | Total tracking time of the particle. |
| | Global_X_Start_Loc | R | - | Starting global X location of the particle. |
| | Global_Y_Start_Loc | R | - | Starting global Y location of the particle. |
| | Local_Z_Start_Loc | R | - | Starting local Z location of the |

| | | | | |
|---|---|---|---|---|
| | | | | particle. |
| | Filler_Ignore | I | - | Unused - -999 is always entered. |
| | CellID_Start | I | - | ID of the cell containing the starting point of the particle path |
| | K_Start | I | - | Layer of the particle starting point. |
| | Zone_Code_Start_Loc | I | - | Unused – 1 is always entered. |
| | SPER_Final | I | - | Flow model stress period corresponding to final tracking time. |
| | TSTEP_Final | I | - | Flow model time step corresponding to final tracking time. |
| | Part_Term_Code | I | - | Unused – 1 is always entered. |
| | Part_Release_Time | R | - | Particle release time.. |

[1] Square brackets indicate optional or dependent variables

[2] C Character
I Integer
R Real

## Pathline File

The pathline file provides temporal and spatial information for each particle as it moves through the flow field calculated by MODFLOW-USG. It lists the time and *x*, *y*, and *z*-coordinates of every step a particle takes.

Table 8 - Pathline file format

| Item | Variable[1] | Type[2] | Options / Values | Description |
|---|---|---|---|---|
| 1 | Label | C | - | mod-PATH3DU does not support these header entries. It writes defaults to the file every time. |
| | Version | I | - | |
| | Revision | I | - | |
| 2 | TrackingDirection | I | - | Echo tracking direction. |
| | ReferenceTime | R | - | Echo reference time. |
| 3 | EndHeader | C | “END HEADER” | End-of-header flag. |

Item 4 is repeated for every step every particle takes

| | | | | |
|---|---|---|---|---|
| 4 | ParticleID | I | - | Particle identification number. |
| | ParticleGroupNumber | I | - | Particle group number. |
| | UnusedFlag | I | - | - |
| | Ktime | I | - | Cumulative time step, from 1 to the last time step of the MODFLOW-USG simulation. |
| | Time | R | - | mod-PATH3DU tracking time. |
| | X | R | - | Global *x*-coordinate. |
| | Y | R | - | Global y-coordinate. |
| | Z | R | - | Global z-coordinate. |
| | CellID | I | - | Cell ID particle is within. Cell ID corresponds to the MODFLOW-USG cell number. |
| | UnusedFlag | I | - | - |
| | Layer | I | - | Layer number. |
| | UnusedFlag | I | - | - |
| | UnusedFlag | R | - | - |
| | UnusedFlag | R | - | - |
| | UnusedFlag | R | - | - |
| | Zloc | R | - | Local *z*-coordinate. |

[1] Square brackets indicate optional or dependent variables

[2]

| | |
|---|---|
| C | Character |
| I | Integer |
| R | Real |

# Examples

## Example 1a

### Tracking in the Vicinity of a “Weak” Pumping Well (Structured Grid)

The purpose of this example is to highlight the strength of the Waterloo and SSP&A method in tracking near a pumping well – the analytical correction terms implicitly handles weak and strong sinks. The groundwater system is confined (50 feet thick), steady-state and two-dimensional, in plan-view. The ambient groundwater flow field is uniform throughout the whole domain and flow is from left to right. The fully penetrating well is pumped at a constant rate of 50 feet$^3$/d. Hydraulic conductivity is 10 feet/d and porosity is 0.3. The numerical domain consists of 21 rows, each 5 feet wide, and 87 columns with varying widths ranging from 5 to 20 feet, for a total model width of 500 feet. The conceptual model is shown in Figure 16. The flow balance is presented in Table 9.

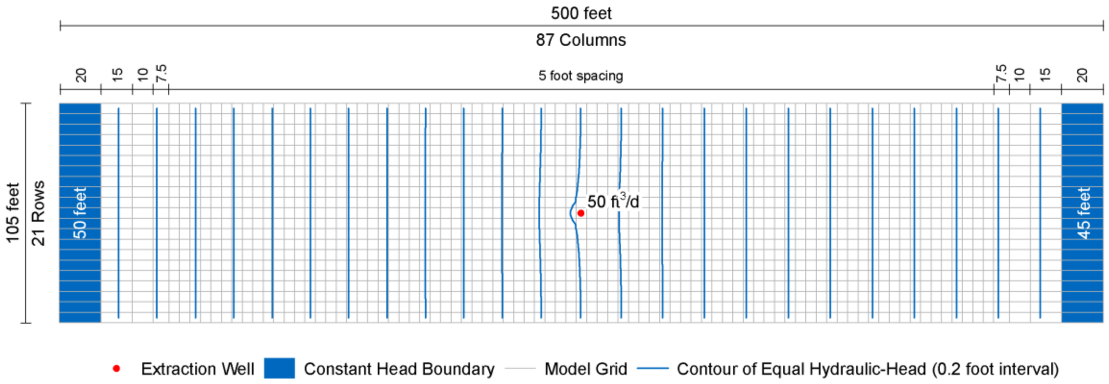


Figure 16 - Conceptual Model - Example 1a

Table 9 - Flow Balance - Example 1a

| Item | In (ft$^3$/d) | Out (ft$^3$/d) | Net |
|---|---|---|---|
| Constant Head | 571.9 | 521.9 | 50.0 |
| Well (MNW2) | 0.0 | 50.0 | -50.0 |
| Total | 571.9 | 571.9 | 0.0 |

Particle paths, in the vicinity of the pumping well, calculated by MODPATH6 and both the Waterloo and SSP&A methods of mod-PATH3DU are shown in Figure 17. The Waterloo method tracks the same as the Pollock method, but also calculates pathlines in the cell containing the weak well – unlike the Pollock method. For both the Waterloo and SSPA methods, pathlines within the analytically calculated capture zone (using Jacob, 1949; p.344.), are captured by the well, while those outside, flow through the cell and continue tracking to their proper terminus. The head contours presented are an expression of what mod-PATH3DU tracks on in the vicinity of a well. They were calculated by interpolating from the MODFLOW calculated heads to a very fine grid using the universal kriging algorithm available with mod-PATH3DU. The shape visible around the well is determined by the linear-log drift term applied in this case.

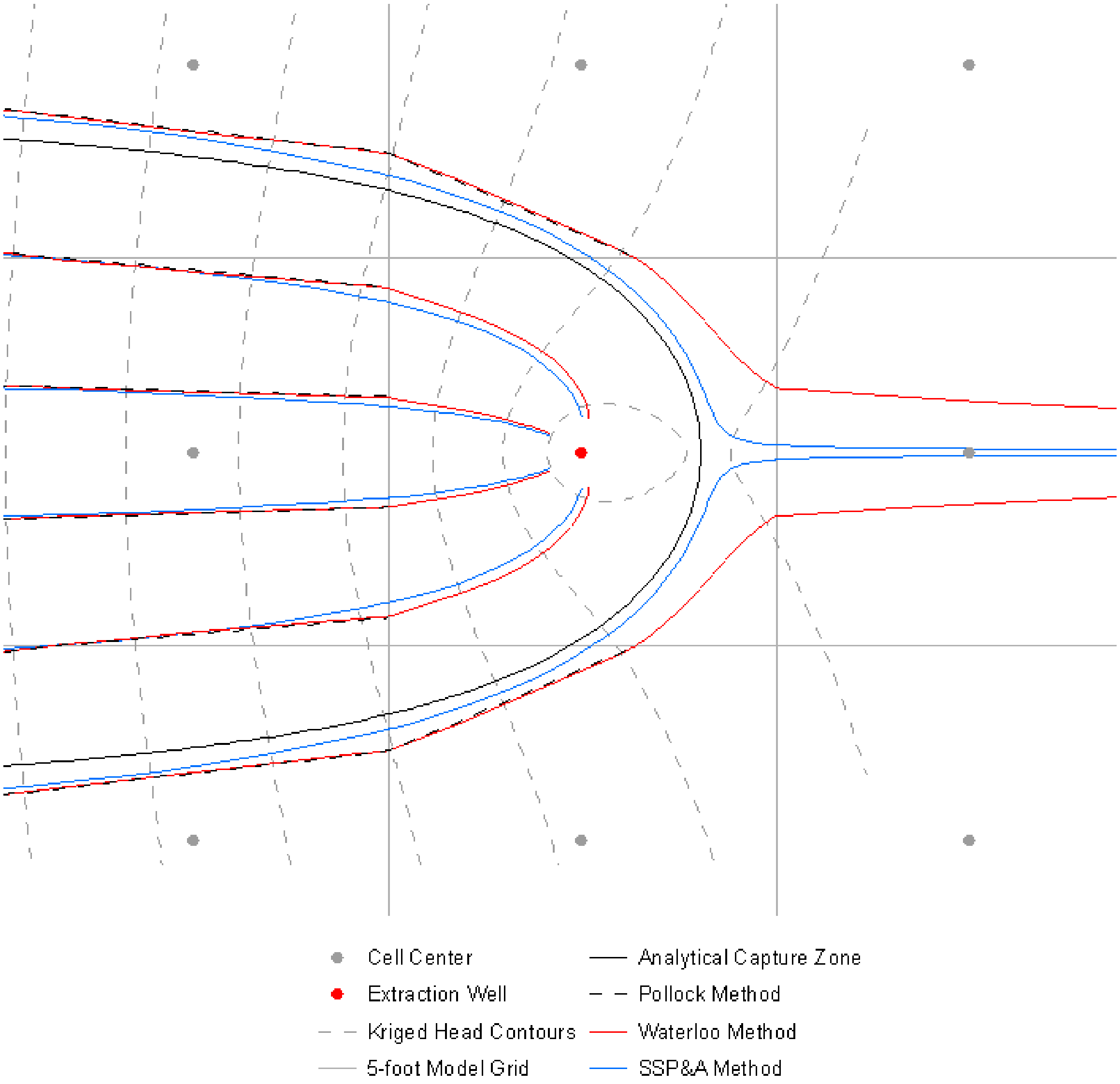


Figure 17 – Particle paths calculated using MODPATH6 and mod-PATH3DU in the vicinity of a "weak" sink.

Because both the Waterloo and SSP&A methods include an analytic solution when tracking near a well, they can mitigate some coarse grid discretization issues. Further, they both can simulate off-center pumping wells. An example of how mod-PATH3DU can be used to calculate capture zones for off-center pumping wells using the Area Based Redistribution (ABRD) approach of Pinales et al. (2003;2005) is presented in Muffels et al. 2011.

## Example 1b

### Tracking in the Vicinity of a "Weak" Pumping Well (Unstructured Grid)

The modeling exercise in Example 1a is repeated using an unstructured grid defined by Voronoi cells, Figure 18. The resulting flow balance is presented in Table 10. The difference in the flow entering the system between this model and the five-foot structured model is due to the different cell geometry along the boundaries.

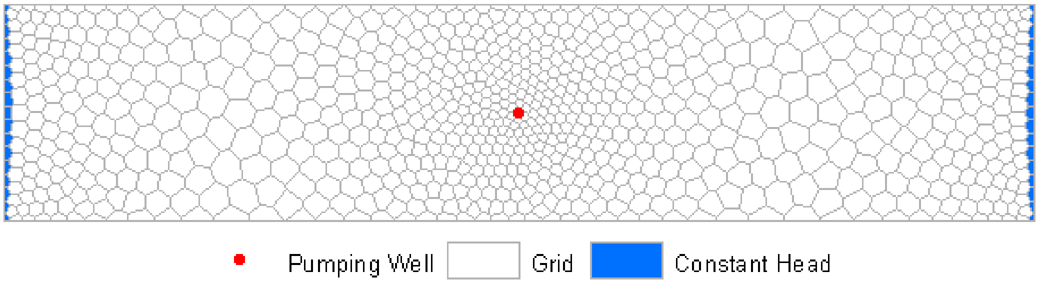


Figure 18 – MODFLOW-USG Voronoi example using AlgoMesh

Table 10 - Flow Balance - Example 1b

| Item | In ($ft^3$/d) | Out ($ft^3$/d) | Net |
|---|---|---|---|
| Constant Head | 532.4 | 482.4 | 50.0 |
| Well (MNW2) | 0.0 | 50.0 | -50.0 |
| Total | 532.4 | 532.4 | 0.0 |

The mod-PATH3DU calculated particle paths, for both the Waterloo and SSP&A methods, for this model are shown on Figure 19.

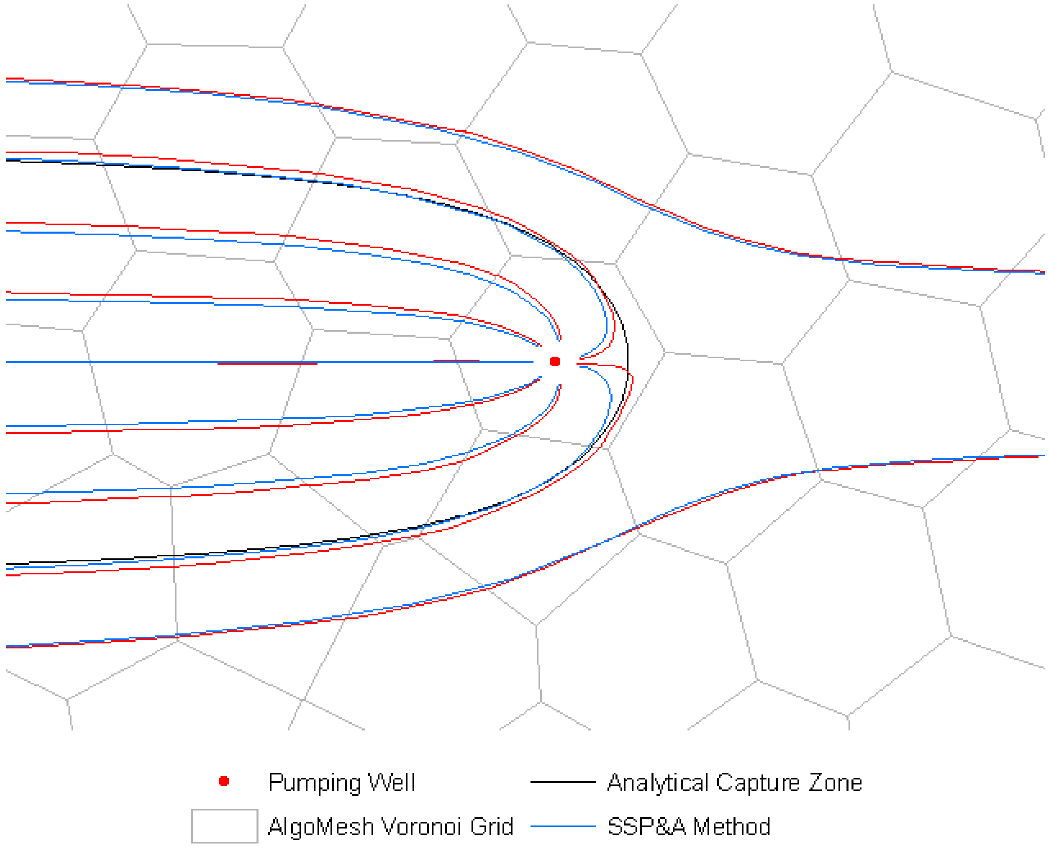


Figure 19 - Particle paths calculated by mod-PATH3DU in the vicinity of a “weak” pumping well for a MODFLOW-USG model with Voronoi cells

## Example 2

### Flow Path in Heterogeneous Cross-section

This example is adapted from the documentation of PATH3D ver. 4.6. The example was developed to test the calculation of groundwater pathlines in a heterogeneous cross-section. The groundwater model comprises three strata; the hydraulic conductivity of each stratum is shown in Figure 20. The model is discretized into 20 layers. The cross-section is bordered by no-flow boundaries along the left, right and bottom edges. The top boundary is assigned specified-heads to represent a linear water table declining from 41 feet at *x* = 100 feet to 40 feet at *x* = 0 feet. Porosity is 0.2. The resulting flow balance in presented in Table 11.

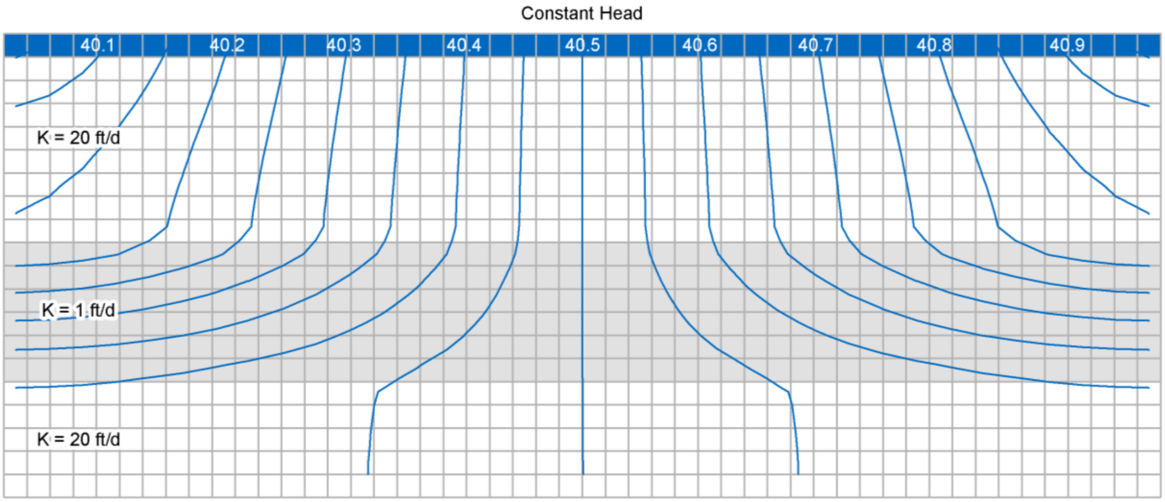


Figure 20 - Conceptual Model - Example 2

Table 11 - Flow Balance - Example 2

| Item | In ($ft^3/d$) | Out ($ft^3/d$) | Net |
|---|---|---|---|
| Constant Head | 3.7 | 3.7 | 0.0 |
| Total | 3.7 | 3.7 | 0.0 |

A particle is tracked backwards from *x* = 1 foot. The particle paths calculated using PATH3D v.4.6 and Waterloo method in mod-PATH3DU for this particle are shown in Figure 21. The total travel time calculated by each approach are nearly identical; PATH3D calculated 636 days, while mod-PATH3DU calculates 640 days. This example primarily serves to demonstrate the implementation of the Pollock method in the *z*-direction when used in conjunction with the Waterloo method is correct and that the method can be applied to cross-section models. This example, in and of itself, is not particularly challenging for the Pollock method.

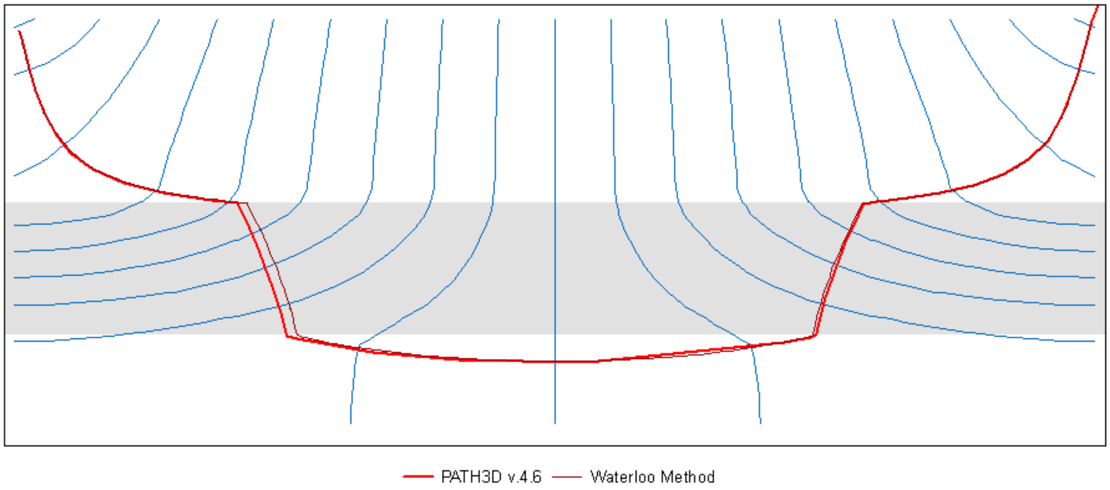


Figure 21 - Comparison of pathlines computed using mod-PATH3DU and PATH3D v4.6 - Example 2

# Example 3

### Example model from Pollock (2015)

Example 4 is taken from Pollock (2015). It is a two-dimensional steady-state model with a quad-based unstructured grid. The system is confined with a constant hydraulic conductivity (100 feet/day) and porosity (0.25). No-flow boundaries surround the model on all four sides. Flow circulation is induced with constant-head cells in the center of the model domain (between 15 feet – blue in Figure 22, and 2 feet – red). There are a total of 352 cells in the unstructured grid. mod-PATH3DU was used to calculate flow paths for the unstructured grid model and these were compared with the results in Pollock (2015) – Figure 22. The flow paths calculated by mod-PATH3DU are comparable to those calculated by the algorithm described by Pollock (2015).

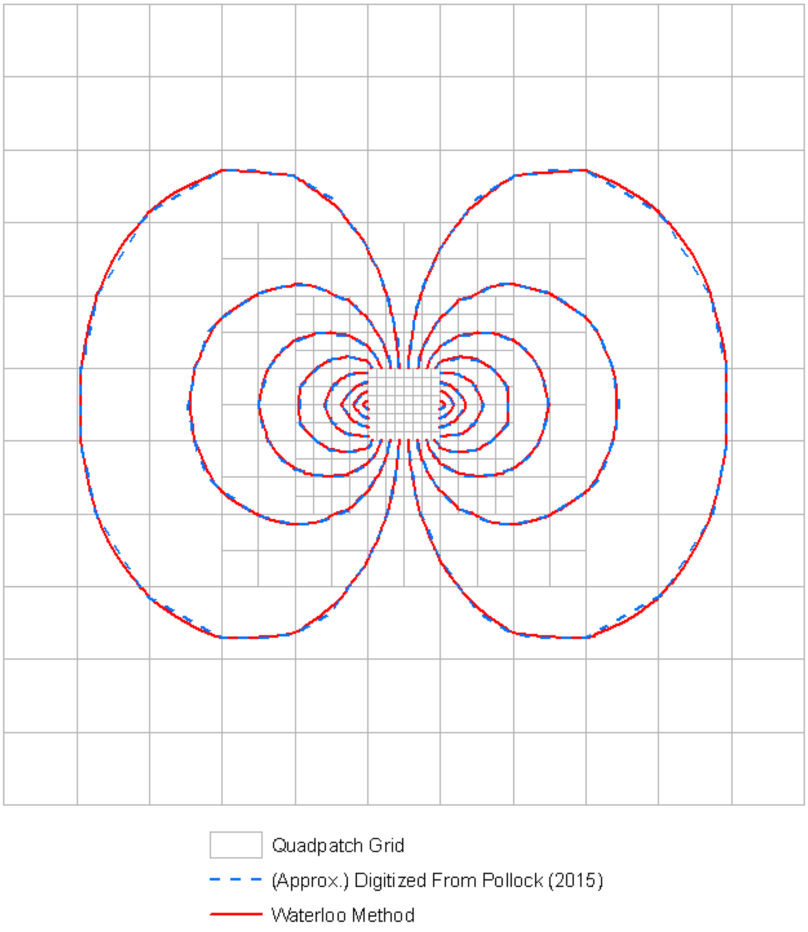
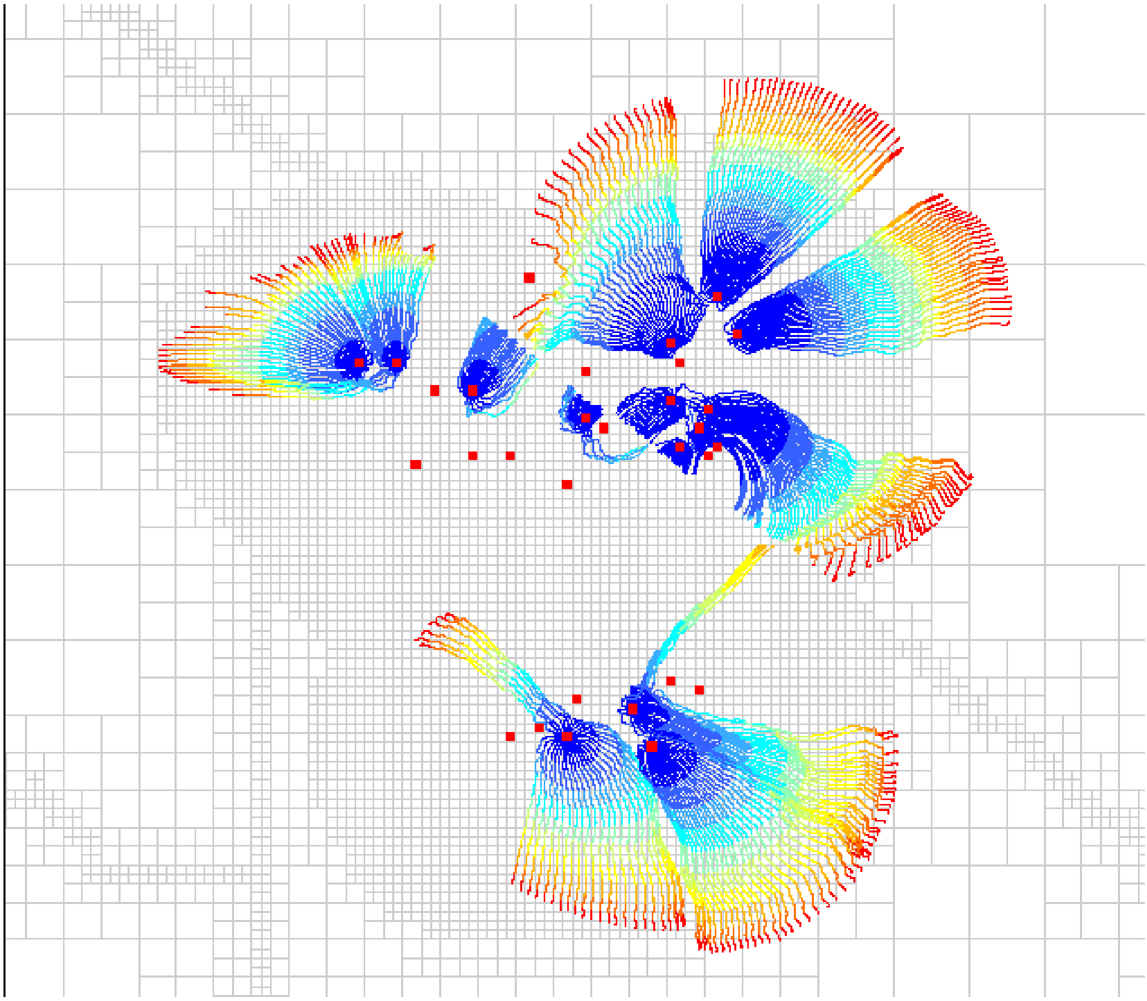


Figure 22 - Comparison of mod-PATH3DU flow paths with those calculated by Pollock (2015) for an unstructured grid

## Example 4

**Quadtree refinement example from Panday et al. (2013).**

In the following example particle paths were calculated for different wells in the Biscayne quadtree demonstration example in Panday et al. (2013). Particle paths were calculated using the Waterloo method. One-thousand daily stress periods were simulated in the flow model – the model includes spatially varying hydraulic conductivity, and spatially and temporally varying boundary conditions, including the river (RIV) and well (WEL) packages. Particles were tracked backward from different wells from day 1000. These paths are provided here for demonstration purposes only.



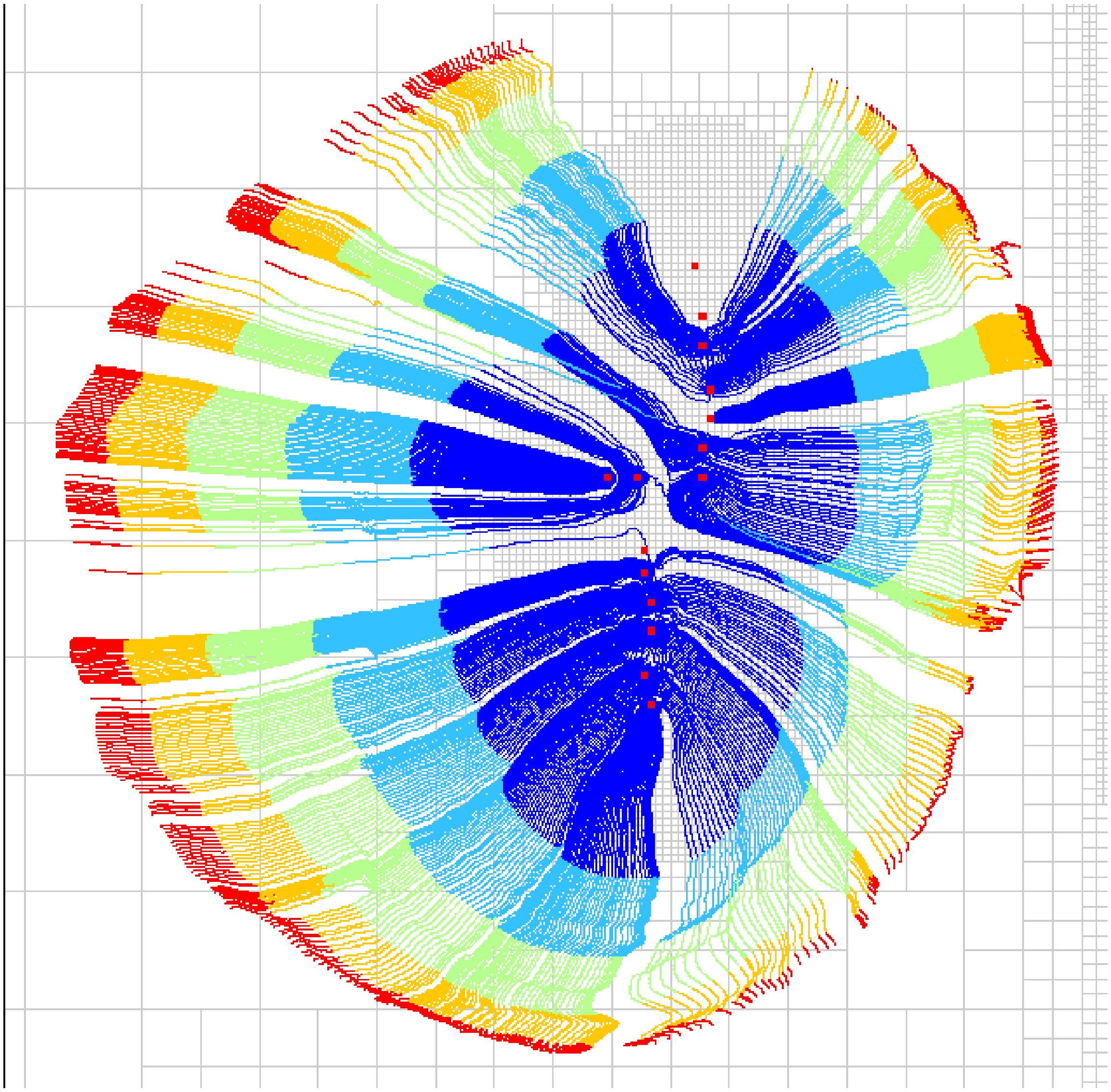


Figure 23 - Demonstration pathlines for different well fields calculated using the Waterloo method for the Biscayne quadtree example in Panday et al. (2013)

## References

Abrams, D., H. Haitjema, and L. Kauffman, 2012. On Modeling Weak Sinks in MODPATH: Ground Water 51, no. 4: 597-602, doi 10.1111/j.1745-6584.2012.00995.x.

Cressie, N., 1993, Statistics for Spatial Data, revised ed., 900pp., Wiley, New York.

Deutsch, C. V. and Journel, A. G., 1998, GSLIB: Geostatistical Software Library and User's Guide, 2nd ed. Oxford University Press, Oxford.

Harbaugh, A.W., 2005, MODFLOW-2005, the U.S. Geological Survey modular ground-water model—The Ground-Water Flow Process: U.S. Geological Survey Techniques and Methods 6–A16, variously paginated.

Jacob, C.E., 1949, Flow of ground water, in Engineering Hydraulics, Proceedings of the 4th Hydraulics Conference, Iowa Institute of Hydraulic Research, H. Rouse (ed.), June 12-15, 1949, John Wiley & Sons, Inc., New York.

Journel, A.G. and Huijbregts, C.J., 1992, Mining Geostatistics. Academic Press, New York.

Karanovic, M., M. Tonkin, and D. Wilson, 2009. KT3D_H20: A Program for Kriging Water-Level Data Using Hydrologic Drift Terms: Ground Water 45, no. 4: 580-586, doi 10.1111/j.1745-6584.2009.00565.x.

Muffels, C., Stonebridge, G., Tonkin, M.J., and Karanovic, M., 2011. An Unstructured Version of PATH3D, PATH3DU. Presentation at MODFLOW and More 2011, Integrated Hydrologic Modeling, International Ground Water Modeling Center, Colorado School of Mines, Golden, CO, June 6-8, 2011, v. 1, pp. 272-275.

Panday, S., Langevin, C.D., Niswonger, R.G., Ibaraki, M., and Hughes, J.D., 2013, MODFLOW-USG version 1: An unstructured gird version of MODFLOW for simulating groundwater flow and tightly coupled processes using a control volume finite difference formulation: U.S. Geological Survey Techniques and Methods 6 A45.

Pinales, A., A. Keer, F. Espinosa, L. Manzanares, G. Llerar, and A. Chavez. (2003), An Alternative Approach for Assigning Well Rates in a Block-Centered Finite-Difference Grid. In MODFLOW and More 2003: Understanding through Modeling – Conference Proceedings.

Pinales, A. Chvez, G. Llerar, L. Manzanares, and A. Keer. (2005), An Improved Approach for Assigning Pumping Rates to Heterogeneous Aquifer Models. Ground Water, vol. 43:274-279.

Pollock, D.W., 1988, Semi-analytical computation of pathlines for finite difference models, Ground Water vol. 26, no. 6: 743–750.

Pollock, D.W., 1989, Documentation of a computer program to compute and display pathlines using results from the U.S. Geological Survey modular three-dimensional finite-difference ground-water flow model: U.S. Geological Survey Open-File Report 89–381.

Pollock, D.W., 1994, User’s guide for MODPATH/MODPATH-PLOT, version 3: A particle-tracking post-processing package for MODFLOW, the U.S. Geological Survey finite-difference ground-water flow model: U.S. Geological Survey Open-File Report 94–464.

Pollock, D.W., 2012, User’s guide for MODPATH version 6 A particle-tracking model for MODFLOW: U.S. Geological Survey Techniques and Methods 6–A41, 58p.

Pollock, D.W., 2015, Extending the MODPATH algorithm to Rectangular Unstructured Grids, Ground Water vol. 54, no. 1: 121–125. doi: 10.1111/gwat.12328.

Rhamadhan, M., 2015, A Semi-Analytical Particle Tracking Algorithm for Arbitrary Unstructured Grids. Unpublished MASc. Thesis. University of Waterloo, Waterloo Ontario.

Skrivan, J. A., and Karlinger, M. R., 1980, Semi-variogram estimation and universal kriging program; U. S. Geological Survey Computer Contribution, 98 p. Tacoma, Washington. (Computer Program K603).

Tonkin, M.J., and Larson, S. P., 2002. Kriging Water Levels with a Regional-linear and Point-logarithmic Drift. Ground Water, vol. 40, no. 2: 185-193.

Zheng, C., 1989, PATH3D, A ground-water path and travel-time simulator, version 3.0 user’s manual, S.S. Papadopulos & Associates, Inc., Bethesda, MD.

Zheng, C., 1992, PATH3D, A ground-water path and travel-time simulator, version 3.2 user’s manual, S.S. Papadopulos & Associates, Inc., Bethesda, MD.

Zheng, C., 1994, Analysis of particle tracking errors associated with spatial discretization, Ground Water, vol. 32 no. 5: 821-828.

Zheng, C., and Bennett, G., 2002. Applied Contaminant Transport Modeling, 2nd ed., John Wiley & Sons, New York.

# Appendix A: WriteP3DGSF.exe

## Preface to Version 1.1.1

In anticipation of mod-PATH3DU v.2, and the need for version control and backward compatibility, we have changed the input format for writeP3DGSF.exe from what was supported in v.1.1.1, and added support to convert a MODPATH7 grid definition file.

## Description

Regardless of whether a model is structured or unstructured, this latest release of mod-PATH3DU requires a modified, program specific, GSF file for input of the spatial location of each model cell. If a GSF file is not listed in the MPNAM file or if the GSF file is not in this updated format, mod-PATH3DU will terminate in error. To facilitate the transition to this new format WriteP3DGSF.exe was developed to:

1. Convert a GSF file in another format into the mod-PATH3DU specific format, or
2. Convert a polygon SHAPEFILE with a cell id attribute (cell ids must begin at 0), or
3. Convert a structured MODFLOW grid defined by DELR and DELC; and
4. Write the corresponding GSF file for a structured model.

The program is executed from the command line as follows:

```
writeP3DGSF FILENAME
```

where *FILENAME* is the name of the required input file, detailed below, and must be the first input on the command line after the name of the writeP3DGSF executable.  Optionally, and recommended, users can supply *colorcode* after FILENAME, i.e.

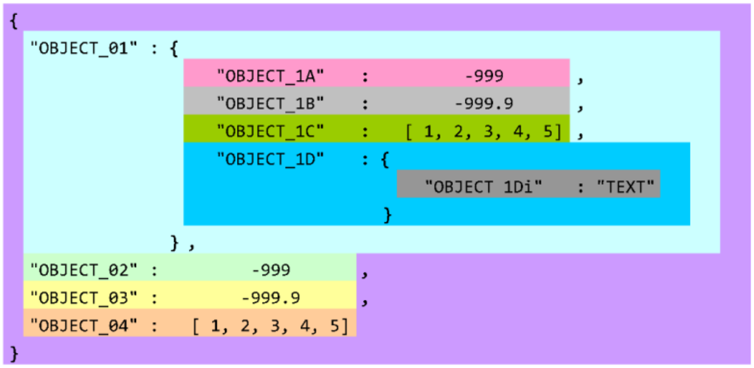
```
writeP3DGSF FILENAME colorcode
```

which will color the text of the console as the program runs. This color coding is more aesthetic than a true codification, except when there is an error – errors, and only errors, are provided in red text.

Input for WriteP3DGSF.exe is provided via a file that is human readable using *key*:*value* pairs, specifically, the JSON[3] format. JSON was developed as a lightweight data-interchange format,

[3] www.json.org

however it has few and simple rules and lends itself to being a flexible, dynamic and easily read input file format.



```
{
  "OBJECT_01" : {
                   "OBJECT_1A"   :          -999     ,
                   "OBJECT_1B"   :         -999.9    ,
                   "OBJECT_1C"   :    [ 1, 2, 3, 4, 5] ,
                   "OBJECT_1D"   : {
                                      "OBJECT 1Di"   : "TEXT"
                                 }
                } ,
  "OBJECT_02" :          -999         ,
  "OBJECT_03" :         -999.9        ,
  "OBJECT_04" :    [ 1, 2, 3, 4, 5]
}
```

Figure 24 - Example JSON file. Each color represents a unique key:value pair, or object. Purple indicates the primary object (for which a key is not required).

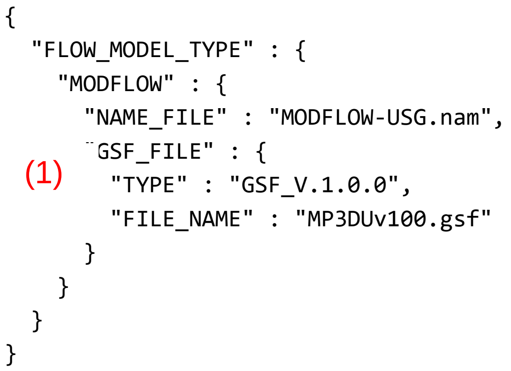
A *key* must be a string (enclosed in quotation marks), for example "OBJECT_01", and a *value* can be one of three types, 1) a single value, 2) a vector of values, or 3) an object (listing of additional *key*:*value* pairs). Vectors are denoted by square-braces, [ ], objects by curly-braces, { }, and single values by the absence of braces. Multiple objects can be listed for a *value*, each object being separated by a comma. Further, a *value*, whether it be a single entry, part of a vector list, or in a *key*:*value* pair, can be either:

- a string (denoted by quotes),
- an integer number, or
- a real number.

WriteP3DGSF requires the keyword(s): FLOW_MODEL_TYPE, GSF_FILE
WriteP3DGSF supports optional keyword(s): TRANSFORMATION
See Appendix B for details on these key words.

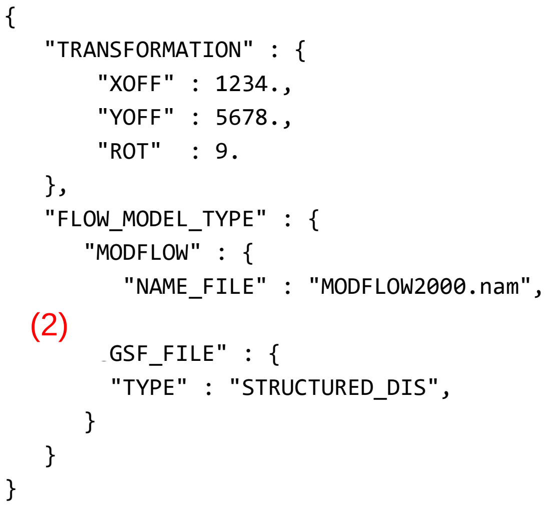
## Example(s):

(1)



```
{
  "FLOW_MODEL_TYPE" : {
    "MODFLOW" : {
      "NAME_FILE" : "MODFLOW-USG.nam",
      "GSF_FILE" : {
        "TYPE" : "GSF_V.1.0.0",
        "FILE_NAME" : "MP3DUv100.gsf"
      }
    }
  }
}
```

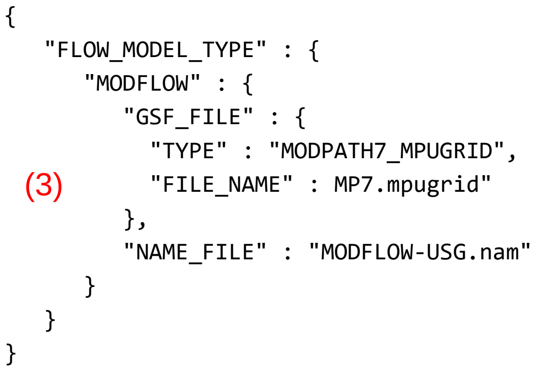
(2)



```
{
   "TRANSFORMATION" : {
       "XOFF" : 1234.,
       "YOFF" : 5678.,
       "ROT"  : 9.
   },
   "FLOW_MODEL_TYPE" : {
      "MODFLOW" : {
         "NAME_FILE" : "MODFLOW2000.nam",

        GSF_FILE" : {
        "TYPE" : "STRUCTURED_DIS",
      }
   }
}
```

(3)



```
{
   "FLOW_MODEL_TYPE" : {
      "MODFLOW" : {
         "GSF_FILE" : {
           "TYPE" : "MODPATH7_MPUGRID",
           "FILE_NAME" : MP7.mpugrid"
         },
         "NAME_FILE" : "MODFLOW-USG.nam"
      }
   }
}
```

The first example (1), will convert an existing GSF (in general format, v.1.0.0) into the updated format for a MODFLOW-USG model, while the second example (2), will write a GSF file from a structured MODFLOW-2000 model, in global coordinates. The third example (3), will convert a MODPATH7 MPUGRID file.

During execution you will be prompted for a name for the new GSF file. It is this file that must be supplied in the mod-PATH3DU MPNAM file - be sure to update this file accordingly before executing that program.

# Appendix B: Supported *Key*:*Value* Pairs

| Key | Value | Description |
|---|---|---|
| TRANSFORMATION | Optional keys:<br>*XOFF*<br>*YOFF*<br>*ROT* | Specifies the transformation to convert between global and local model coordinates. This key is used when WriteP3DGSF converts a structured grid defined by DELR and DELC into a GSF with global coordinates. |
| FLOW_MODEL_TYPE | Required keys:<br>MODFLOW | Specifies the type of flow model. Currently, only MODFLOW is supported. |
| XOFF | Real number. | The offset in the x-direction when converting between global and local model coordinates. |
| YOFF | Real number. | The offset in the y-direction when converting between global and local model coordinates. |
| ROT | Real number. | The rotation (in degrees) when converting between global and local model coordinates. |
| MODFLOW | Required keys:<br>NAME_FILE<br>GSF_FILE | Indicates MODFLOW is the flow model type. |
| NAME_FILE | String (in quotes) | The name of the MODFLOW name file. |
| GSF_FILE | Required keys:<br>TYPE<br>FILE_NAME | Specifies the type of GSF file. For writeP3DGSF.exe, it indicates the file to convert. For mod-PATH3DU, it indicates the file to read – this file has to be of type “GSF_V.1.1.0”. FILE_NAME is not required when converting a structured DIS file to GSF. |
| TYPE | String (in quotes) | GSF_FILE supported types:<br>GSF_V.1.0.0,<br>GSF_V.1.1.0,<br>SHAPEFILE,<br>STRUCTURED_DIS, and<br>MODPATH7_MPUGRID. |
| FILE_NAME | String (in quotes) | The name of a required file. |