# DEVELOPMENT OF THE FAST 3D PARTICLE TRACKER: PTRAX

**Dudley J. Benton**
**Environmental Consulting Engineers**
**Knoxville, Tennessee**

***presented at the:***
**Tennessee Water Resources Symposium**
**Nashville, Tennessee, February, 1997**

## ABSTRACT

Particle tracking has been successfully used to model diverse phenomena. Although the present application is for the transport of contaminants within a groundwater system, the same algorithm could be used for a variety of applications. In addition to deterministic particle tracking, in order to simulate variability in the transport media, a random walk is included in the algorithm.

Deterministic 3D particle tracking is a formidable task. The addition of a random walk adds considerably to this complexity. In order to accurately model statistically variable systems, it is necessary to have a very large sample, that is, to track a large number of particles. Practical simulations using this technique must, therefore, rely on a very fast algorithm. PTRAX is a computer code developed to implement these concepts.

## NOMENCLATURE

n number of steps and/or particles
R a normalized random number
ΔS random step length
ΔT time step
U velocity component in the X direction
V vector velocity
V velocity component in the Y direction
W velocity component in the Z direction

**Greek**

α dispersion length

**Subscripts**

i particle step
L in the longitudinal direction
M mean (i.e., non-random)
T in the horizontal-transverse direction
V in the vertical-transverse direction
X in the X direction
Y in the Y direction
Z in the Z direction

## INTRODUCTION

The present application is contaminant transport within a groundwater system. In this case a velocity field is computed using an Eulerian framework code. The particles are then tracked within this field. Conventional particle tracking is based on a Lagrangian approach where time is the primary independent variable and velocity is the secondary. The location of a particle is determined by integrating the velocity with respect to time; thus, in an analytical sense, conventional particle tracking involves the solution of temporal Volterra integral equations. For non-trivial problems, the velocity varies with location, making the problem implicit. The most common integration scheme used is the Runge-Kutta. Various predictor/corrector schemes are also employed. Because the problem is implicit, step-length control is preferred to uniform small steps. As will be detailed subsequently, a variable time step increases the complexity of the random walk.

Computation of stream lines within an Eulerian framework might be thought of as analogous to Lagrangian particle tracking. The difference between two adjacent stream lines is the mass flowing between them. Contaminant transport could be computed in this framework by solving along the stream lines. Within an Eulerian framework, location becomes the primary independent variable; and partial differential equations are solved.

Locating the stream lines within an Eulerian framework and solving equations along them is quite computationally intensive, and essentially impractical. This is why most codes resort to Lagrangian particle tracking. Flow fields are rarely solved within a Lagrangian framework because the boundary and initial conditions are problematic, making the method impractical. Thus, obtaining a flow field and particle tracks in a practical implementation necessitates combining differing frameworks.

The Lagrangian and Eulerian frameworks are different enough to make their coupling in this way inherently inefficient. A different framework is proposed which is somewhere between these two. In this alternative framework, location and velocity are the primary and secondary independent variables and time is the dependent. This combination of variables may be thought of as analogous to a Hamiltonian framework. Within this framework, in an analytical sense, spatial Fredholm integral equations are solved.

Whatever framework is employed, there is considerable transformation and computation between a grid, velocity field, and particle seeds and the tracking of the particles. The present particle tracking algorithm is shaped by these transformation and computational steps; therefore, it is necessary to first describe these.

## GRID TRANSFORMATION

The process of transforming a grid into a cell-linked space within which particles can be tracked is the first step in implementing the present scheme. The following steps accomplish this transformation:

### Defining Nodes

Nodes are defined by unique locations in space which are read from a data file. The number of dimensions is inferred by the number of coordinates. Two coordinates are interpreted as X,Y and imply a 2D space. Three coordinates are interpreted as X,Y,Z and imply a 3D space. Non-unique or coincident nodes will lead to singular basis equations and must be rejected. A check is performed for this after all the nodes have been read.

### Defining Elements

Elements are defined by groups of nodes and are read from a data file. In a 2D space, three nodes imply triangular elements and

four nodes imply quadrangular elements. In a 3D space, four nodes imply tetrahedral elements, six nodes imply prismatic elements, and eight nodes imply brick elements. The nodes comprising an element must be unique. A check is performed for this as the elements are read. Each node must appear in at least one element. Every node must be connected to every other node through the elements in order to have a unified (as opposed to a disjointed) domain. A check is performed for each of these after all of the elements have been read. No assumptions are made as to the angles formed by the sides of the elements (e.g., sides of 3D bricks are not assumed to form right angles).

**Element Orientation**

Elements must be numbered according to some convention in order to determine their connectivity. The convention adopted by PTRAX is counter-clockwise orientation. In 2D, this means that the area of each element is computed as a positive number. Any element having an area less than some small value (e.g., $10^{-9}$ times the total area) is rejected as degenerate. In 3D, this means that the volume of each element is computed as a positive number. Any element having a volume less than some small value (e.g., $10^{-9}$ times the total volume) is rejected as degenerate.

3D elements have the added complexity of orientation of the faces. These too must be positive in the vector sense (i.e., the dot product of the outward normal area vector with the vector beginning at the volume centroid and passing through the area centroid of each face must be positive). As numbering schemes differ, and so as to provide the greatest convenience, PTRAX analyzes each element and sets bit flags to indicate the orientation of each element and face.

In the case of 2D elements, it is always possible to renumber the elements so as to conform their orientation. However, in the case of 3D elements, ambiguities may arise which force the element to be rejected. If the elements can be renumbered without ambiguity, the process continues and the total number of miss-oriented elements is listed for information. If a minor ambiguity occurs (i.e., involving symmetrically opposed faces), a warning is issued and the process continues. If a major ambiguity occurs (i.e., involving adjacent faces), the element is rejected. The necessity of this distinction between fatal and non-fatal ambiguities will become apparent as the element splitting is described. As the orientation of each element is analyzed, various numbering conventions can be mixed within a single grid and the end result will be the same.

**Node: Element Links**

The first step in connecting the elements is to build a list of node:element links. When complete, this list will contain each element in which each node appears. As PTRAX uses dynamic memory allocation and pointers, this list will resemble a simple database structure and require a minimum amount of storage. Each node will be assigned an index where its list of elements begins as well as a count. The maximum count for all nodes is a measure of the bandwidth. This is listed for information.

**Element: Element Links**

The second step in connecting the elements is to build the list of element:element links. This is structured like the node:element list and represents a recursional search of the node:element list for nodes common to each face of each element. Any element face which is not connected is external (i.e., represents a boundary); while any connected face is interior.

A recursional search of the node:element list for nodes common to each face represents a potentially immense calculation. If an exhaustive search were performed using nested loops, the number of comparisons would be proportional to the number of elements cubed. Clearly, this would be impractical for any sizable grid. PTRAX uses a *hashing*, followed by a *Q-sort* on the element faces, followed by a *bubble-up* on the facial nodes. This combined algorithm selects, in order, the members common to a variable number of lists, each of variable length. The time required for this procedure is roughly twice that required to determine the connections to a single face using nested loops. For a grid containing thousands of elements, this procedure reduces the computational time by orders of magnitude. The time required for this procedure is of the same order of magnitude as reading the node and element files. The maximum number of connections between elements is a measure of the bandwidth and is listed for information.

**Element Face Orientation**

As the elements are analyzed, bit flags are stored indicating the element orientation as well as the orientation of each face for 3D elements. The elements are not actually renumbered, only the bit flags are set to indicate their orientation. When the elements are connected at the faces, it is necessary to mate their orientation. The orientation flags must be adjusted so that each corresponding face where two elements are connected satisfy orientational reciprocity. The dot product of the outward normal area vector for each pair of element faces must be negative (i.e., opposite in direction).

**Element Splitting**

In numerical analysis, the variation of parameters within an element is approximated by basis functions. These basis functions vary over the element and are typically assumed to combine linearly (i.e., superimpose). The only basis functions which assure continuity of a varying parameter at every point on a face between two elements are the linear combinations of the spatial directions (i.e., $C_1+C_2X+C_3Y+C_4Z$). An added benefit of this selection of basis functions is that the area of triangular elements and the volume of tetrahedral elements is equal to the determinant of the basis matrix and must be computed anyway.

Selection of these basis functions fixes the element type in 2D to be triangular and in 3D to be tetrahedral. In order to analyze the grid, PTRAX splits quadrilaterals into two triangles, prisms into three tetrahedra, and bricks into five tetrahedra. Splitting quadrilaterals into triangles is a simple matter; however, splitting 3D elements requires that adjacent sides have a certain orientation.

Uniformly oriented prisms when split into three tetrahedra do not have a line of symmetry, and thus, do not match-up. Proper splitting of prisms requires that every other prism be split as a mirror image. As there is an odd number of sides to the triangles forming the ends of the prisms, this does not inherently lead to orientational conflict and is basically a matter of bookkeeping, which is handled by the element orientation flags.

3D bricks do have a line symmetry, but must be alternately rotated 90°. Grids where 3D brick elements are inserted in odd numbers around an internal boundary cannot be properly split into tetrahedra, as there is no combination of rotations which will result in the tetrahedra properly connecting. A check is performed for this as the elements are split and any such conflicts result in the grid being rejected.

The basic 2D building block is the triangle. The basic 3D building block is the tetrahedra. As more complex elements are split into these basic building blocks, the term *cell* is used.

### Node: Cell Links

As PTRAX uses dynamic memory allocation and pointers, grids whose elements do not require splitting into cells, only require that the pointers be equivalenced. Grids whose elements are split into cells require additional storage. Grids whose elements require splitting must be reanalyzed for node:cell links. During this process, several auxiliary parameters are calculated, such as the cell centroids. Grids whose elements are not split, already have the node:cell links established, as node:element links. Some runtime is reported even in these cases, because of the auxiliary parameter calculations. Once again, each node must appear in at least one cell; and every node must be connected through the cells to every other node. Grids failing any one of these tests are rejected.

### Cell: Cell Links

As PTRAX uses dynamic memory allocation and pointers, grids whose elements do not require splitting into cells, only require that the pointers to the list of cell:cell links be equivalenced to the list of element:element links. Grids whose elements are split into cells require additional storage and must be reanalyzed. During this process external faces are flagged by -1; whereas internal faces have some index greater than or equal to zero. Some runtime is reported even in these cases, because of the face flag calculations. The same fast algorithm is used to establish the cell:cell links.

Once the list of cell:cell links has been established, given a starting point within any cell, all of the paths leading from that cell are given in the list. A path leaving a cell may end at a boundary or enter an adjacent cell. This cell:cell link list is the *road map* for particle tracking.

### Additional Checks and Information

While this cell:cell list is created, a number of checks and non-essential calculations are performed. As these have been carefully optimized, the cost in runtime is minimal while valuable information is gained about the grid structure and important checks performed which are not done in many codes. Varying amounts of this information can be listed through command options. In addition to these, an entire nearest neighbor node list, associated bandwidth, and pivot matrix, which are the core of finite element modeling, can optionally be generated and listed through command options. While this information is not used by PTRAX, it can be useful in grid analysis and can serve as a further check. These steps are included as PTRAX had its origin in a FEM code, and may eventually use this information for enhanced modeling.

## PARTICLE TRACKING

Particles can be seeded automatically, scattered throughout the grid, or specifically in a data file. The seed locations can be specified by element or by X,Y,Z location and initial mass. As finding the cell containing the seed is a time-consuming process, the element in which the particle is seeded can optionally be specified along with the location. This directly specifies the cell if the elements are not split, or confines the range of cells to be searched for the nearest centroid if the elements are split. This can save considerable runtime.

A particle must lie unambiguously within a cell on the first step. Incorrectly specifying the starting cell will result in the particle being artificially trapped. As there is no prior step or history at the start, a particle must not be seeded on a boundary between two cells (On subsequent steps it may frequently lie on cell boundaries.). As indicated previously, PTRAX can handle some ambiguously numbered 3D elements. Ambiguously numbered elements can result in particle reflection. Reflection will trap a particle if it is not seeded unambiguously within the interior of the cell. Whether or not a particle is unambiguously within the interior of a cell depends on factors such as cell aspect and round-off and is not easily quantified. In order to avoid these problems, PTRAX by default repositions all seeds at their start to the centroid of the nearest cell. This feature can be defeated by a command parameter.

### Velocity Field and Properties

The velocity field can be specified as a default for all elements or separately for each element in a data file. The velocity is initially divided by the porosity and retardation factor. Porosity and retardation factor can be specified as a default for all elements or separately for each element in a data file. The default values are set in the code or are read from the optional configuration file, which can be modified as needed. The concentration (mass/volume) is multiplied by the porosity upon completion of the simulation. Cells which are split inherit the properties of the parent element.

### Solving for Particle Direction

Within a cell, a particle moves from its current position in the direction of the local velocity vector until it intersects a face. [The random walk is a modification to this procedure and will be detailed subsequently.] The intersected face is determined by the following procedure: The equation for the line in 2D or plane in 3D defined by each face of the current cell is determined. This arises directly from the assumed linear basis functions and results in an unambiguous calculation. The necessity of splitting prisms and bricks into tetrahedra can be seen from this: four unique points do not necessarily lie in the same plane in 3D. Four points may form a saddle. There is no ambiguity in the lines and planes defined by the faces of triangles and tetrahedra. The length of the side in 2D or area of the face in 3D is the determinant of the basis matrix. As the cells have already been screened and the lengths and areas computed, this assures non-singular results and reduces the number of calculations within a deeply-nested loop.

The distance along the local velocity vector, from the current particle location to the intersection with a face, is the time. If the computed time for a given face is zero, the particle does not

move. [This may seem to be a trivial case, but will have applicability in the random walk as detailed subsequently.] If the time is negative, this represents a backward step and is rejected. The face which produces the minimum time, greater than zero, is the first intersected, and thus, the point of exit from the cell. If the cell:cell link corresponding to the exiting face is greater than or equal to zero, then the particle continues on into that cell. If the link is equal to -1, the particle exits at the boundary.

### Particle Track Termination

Several causes may result in the termination of a particle track. These include: capture by a well, complete decay of mass, escape through a boundary, stagnation, and the end of the tracking period.

Wells are defined by capture zones. Wells are specified in a data file by nodes, elements, or location, screen opening, and capture radius. If a particle enters the capture zone of a well, its track ends and its mass and time of capture are transferred to the corresponding well.

The decay of particle mass may be specified as a half-life for each element or a constant value for all elements. If the half-life for a cell is greater than zero, then the particle mass decays based on how long the particle stays in the cell. The decayed mass is transferred permanently to the cell in which the decay occurred. If the half-life within a cell is zero, then all of the particle mass is transferred to the current cell at the time it enters the cell and the track is terminated.

If a particle escapes at a boundary, its track is terminated and its mass and time of escape is transferred to the global counters for escaped particles.

If a particle enters a cell having zero velocity, the time to reach any face would be infinite, so its track is terminated and its mass is transferred to the current cell along with its time of entry.

If a particle enters a cell where the velocity is not zero, but no exit times are computed for the faces which are greater than some small value (e.g., $10^{-9}$ times the previously defined small length divided by the r.m.s. average field velocity), then it is considered to be *trapped*. Any occurrence of this trapping is considered anomalous (i.e., should not occur under normal circumstances). A count of such *trapped particles* is listed as an additional check.

A final cause for particle track termination is the maximum steps along a track. Before tracking a particle, it is necessary to allocate storage for its history. This is used for bookkeeping and calculation of *snapshots* or field samples at specific times. The maximum steps along a track is defined in the code or specified in the optional configuration file, which can be modified as needed.

The number of particle tracks terminating for each of these causes is listed after all of the particle tracks are computed. In addition, if particle tracks are to be saved for plotting, the cause for termination of each particle track is filed in both numerical form (i.e., an index) and text form (i.e., a string such as *boundary* or *decay*).

Two separate lists are kept for particle track termination: one associated with the particle and one associated with the receptor of the particle (e.g., a well or boundary). Any disagreement between these two lists is considered anomalous (i.e., should not occur under normal circumstances). A count of the difference between these lists, or the *missing particles*, is listed as an additional check.

## SNAPSHOTS AND WELL LOGS

The ensemble of particles is sampled at specific times as defined in the code or specified in the optional configuration file, which can be modified as needed. The information associated with these specific times, or *snapshots*, is saved in sequentially-named files after all of the particles have been tracked. The contribution of each particle is added to each snapshot at the end of its track. Although the particles are tracked on a cell basis, the snapshots are accumulated on an element basis. The storage for these snapshots must be allocated before any particles are tracked. If the particle tracks are to be saved for plotting, each track is filed after the snapshots are updated. The storage for a particle track is used over again so that the requirement does not grow with the number of particles.

Every time a particle is captured by a well, the mass and time of capture is recorded. Two separate lists are maintained for this capture. One is based on the snapshot interval. This requires minimal storage, which is allocated before any of the particles are tracked. A second optional list is kept which contains every particle captured by every well, its mass when captured, and when it was captured. This list grows with the number of particles and may become very large. After all of the particles are tracked, this optional list is sorted and filed by well.

Each snapshot file contains a summary by particle and mass. This summary includes the particle count and mass for each track termination cause as well as the double-checking for *missing* and *trapped* particles (Which should always be zero if the grid, velocity and property fields, seeds, and wells are properly defined.). The centroid, concentration (volume/mass), mass, accumulation, and element number are filed for each element. By default, only those elements containing some mass are filed. Optionally, a command parameter can be used to force all elements to be filed.

If there are any wells, the total mass captured by each well is listed at the bottom of each snapshot file. As these entries contain fewer numbers than the element-by-element concentrations, data analysis and presentation programs should be able to directly distinguish these results. Alternately, these results might be stripped off and filed separately.

## RANDOM WALK

A random walk is used to model variability in the transport media. The random walk is a means of quantifying the results of dispersion, which is thought to arise from this variability.

### Dispersion

Dispersion in groundwater transport is characterized by a dispersion length. A separate dispersion length can be specified for the two or three dimensions of the grid space. Default dispersion lengths can be assigned to all elements or separate values can be specified for each element in a data file. The default dispersion lengths are defined in the code or specified in the optional configuration file, which can be modified as needed.

Cells which are split inherit the properties of the parent element, including the dispersion lengths. The dispersion lengths can be applied along the grid axes (i.e., X,Y,Z) or along the local axes (i.e., longitudinal, horizontal-transverse, and vertical-transverse).

The random step length associated with a dispersion length is defined by the following equation:

$$\Delta S = R\sqrt{2\,\alpha\,|\vee_m|\,\Delta T} \tag{1}$$

where $\Delta S$ is the random step length, $R$ is a normalized random number (i.e., having a mean of 0 and a standard deviation of 1), $\alpha$ is the dispersion length, $|V_M|$ is the magnitude of the mean (i.e., non-random) velocity, and $\Delta T$ is the time step.

For dispersion in several directions, multiple random numbers (i.e., $R$ s) and directionally associated dispersion lengths (i.e., $\alpha_X$, $\alpha_Y$, $\alpha_Z$ or $\alpha_L$, $\alpha_T$, $\alpha_V$) are combined to form the random steps (i.e., $\Delta S_X$, $\Delta S_Y$, $\Delta S_Z$ or $\Delta S_L$, $\Delta S_T$, $\Delta S_V$). For a particle traversing a cell, there is an effective random velocity associated with the random step length and implied time step.

$$\left|\vee_R\right| = \frac{\Delta S}{\Delta T} \tag{2}$$

For dispersion in several directions, the effective velocity components can be represented by a mean and random part:

$$U_T = U_M + U_R = U_M + \frac{\Delta S_X}{\Delta T} \tag{3}$$

$$V_T = V_M + V_R = V_M + \frac{\Delta S_Y}{\Delta T} \tag{4}$$

$$W_T = W_M + W_R = W_M + \frac{\Delta S_Z}{\Delta T} \tag{5}$$

where $U$, $V$, and $W$ are the velocity components in the $X$, $Y$, and $Z$ directions, respectively. If dispersion lengths are specified along the longitudinal, horizontal-transverse, and vertical-transverse directions, the corresponding steps along the principle axses are computed using standard trigonometric relationships.

### Statistical Requirements

For a statistically large sample (i.e., many particles), the net influence of the random walk on the ensemble of particles must exhibit several properties:

1) The spreading (over that without dispersion) in the direction associated with each $\alpha$ is proportional to the square-root of $\alpha$ and $\Delta T$.

2) The net displacement of the particles (compared to that without dispersion) is zero.

3) The net movement of the mass-weighted centroid of the particles is the same with or without dispersion.

Given these properties and the relationships between the random step length, mean and random velocity components, and time steps, the following requirements can be deduced:

$$\sum_{i=1}^{n} \Delta S_i \approx 0 \tag{6}$$

$$\sum_{i=1}^{n} \frac{\Delta S_i}{\Delta T_i} \approx 0 \tag{7}$$

These summations must hold for a single particle as well as for the ensemble, and they must hold in each dimension. In order to simultaneously satisfy these pairs of relationships, the time steps must be equal (i.e., if they are equal, then $\Delta T$ can be brought outside the summation).

Because these statistical relationships require equal time steps, an immediate problem arises, regardless of whether conventional Lagrangian particle tracking or the present method is used. Efficient implementation of a Lagrangian method requires a dynamically adjusted step length. Implementation of the present scheme results in time steps varying over orders of magnitude, as a particle may pass through a cell near a vertex and cover a small distance in a correspondingly small time. If a constant time step is required, then the smallest required time step becomes a limiting factor and results in impractical runtimes. It is for this reason that the random walk is not frequently used in large particle tracking applications or such applications are run on super computers.

### Synchronous Time Steps

If variable time steps are adjusted such that they fall-out on a constant time step with sufficient frequency to represent a statistically significant sample, and the statistical calculations (i.e., the random walk steps) are performed on these synchronous time steps, this limitation can be overcome. This requires keeping two lists of particle history: one based on the constant time step and one based on the variable steps (which periodically add-up to, and thus, synchronize with the constant time step). This can be accomplished with a Lagrangian tracking scheme by requiring the refined time steps to be integer divisions of the coarse time step. It is accomplished in the present scheme by summing and/or truncating sequential particle steps (i.e., *lurching*) through the cells so as to synchronize with a constant time step. The present scheme uses an innovative bookkeeping algorithm to combine these two lists.

Computational experiments comparing the present scheme and an analytical solution indicate that the synchronous time step need not be equal to the smallest cell traverse time. The synchronous time step need only be small enough such that there are sufficient steps along a single particle track to represent a significant statistical sample (25 has proven to be sufficient for these experiments). If the statistical results are satisfied for each particle, then they will necessarily be satisfied for the ensemble. On a practical level, even if the number of synchronous steps along some of the particle tracks is smaller than this, the combined result for the ensemble will be satisfied if there are many particles (100 has proven to be sufficient for these experiments). The present scheme selects a synchronous time step such that the particle tracks will endure for an average of approximately 25 intervals. The average particle track may endure for over 100 asynchronous intervals. A typical Lagrangian scheme may require 1000 steps for the average particle track; thus, the present scheme represents a significant improvement in runtime and storage.

## RESULTS

Figure 1 shows the track of 11 particles seeded at slightly different locations in the absence of dispersion. Figure 2 shows the track of the same particles with typical dispersion factors (viz. $\alpha_X$=12, $\alpha_Y$=6, $\alpha_Z$=0). In both cases the velocity field is uniform as are the properties. Figure 3 shows the track of 100 particles seeded at the origin with the same dispersion factors. [Note that it is not practical to plot the tracks of more than 10,000 particles.]

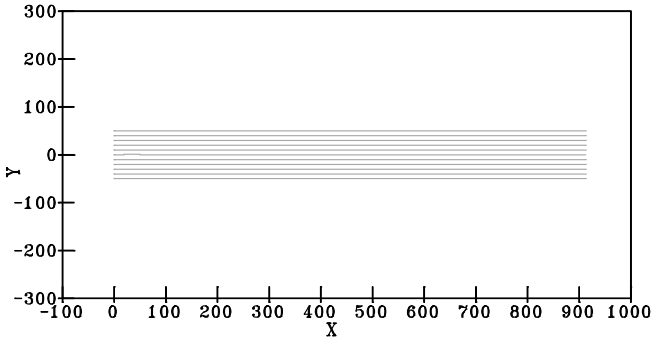


**Figure 1**

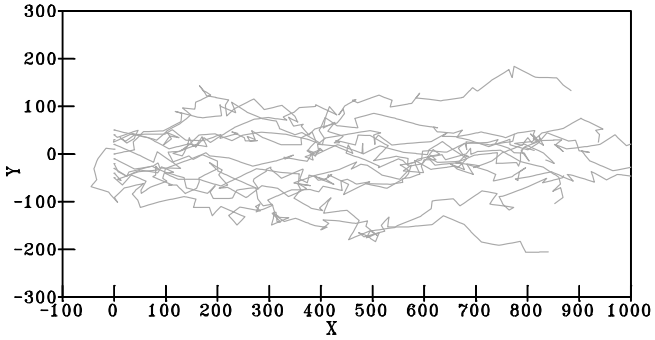


**Figure 2. Track of 11 Particles with Dispersion**

### Comparison with Analytical Solution

PTRAX has been compared to the analytical model, AT123D (Yeh, 1981). Dispersion and migration of a point source was selected as the test case. The flow was assumed to be uniform in the X direction at 0.02 meters/day. The properties were assumed to be constant with a porosity of 0.2 and a retardation factor of 1 (i.e., no retardation). The domain was selected having dimensions of -95 to 605 meters in the X direction, -255 to 255 meters in the Y direction, and -34 to 34 meters in the Z direction. The grid was constructed of 10x10x4 meter bricks (in the X, Y, and Z dimensions, respectively). This results in 60,690 elements and 66,456 nodes. The source was represented by 8,000 particles of equal mass (0.01 grams), seeded at X=Y=Z=0. This combination of particle mass, element size, and porosity produces an initial concentration of 1 grams/meter$^3$. A variety of dispersion factors and two time durations were selected. The results of these numerical experiments are given in Table 1. The agreement between PTRAX and the analytical solution of Yeh was found to be excellent. Figures 4 and 5 show contours of computed concentration for PTRAX and the analytical solution, respectively.

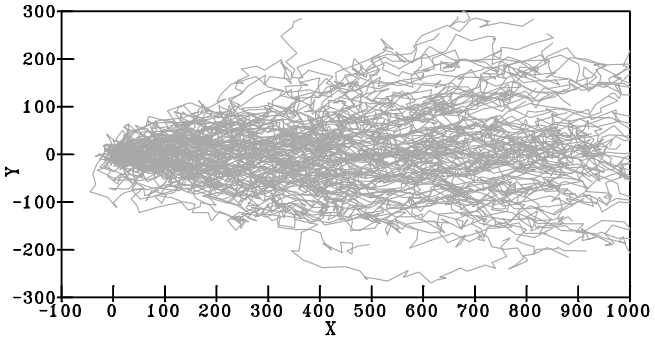


**Figure 3. Track of 1000 Particles with Dispersion**

### Performance

PTRAX was developed on and for the PC platform. Runtimes for the test cases (8,000 particles for 30 years) were under 2 minutes on a Pentium 90. Tracking 500,000 particles for 50 years requires 80 minutes and 18 MB RAM on the same machine. PTRAX has been run with as many as 1,000,000 particles, with up to 250,000 elements, and for simulations up to 10,000 years. All of these runs were made on a Pentium; however, an 80386 machine might have been used, as no special features of the Pentium are required. PTRAX does, however, require extended memory and runs in protected mode (i.e., requires an 80386 or better processor).

## SUMMARY

An alternative approach to Lagrangian particle tracking has been presented. This approach is more easily mated with Eulerian flow fields. This approach is also considerably faster than conventional particle tracking. This improved performance is due in part to its integration methodology, which is based on velocity and spatial steps rather than temporal steps. A computer program, PTRAX, which is coded in C, has been developed for the PC platform to implement this approach. Transformation of the grid into a cell-linked space is accomplished using an innovative algorithm. The random walk is added to model dispersion. Time step synchronization is achieved by a *lurching* procedure which eliminates the necessity of uniformly small steps. Dual particle histories (i.e., asynchronous and synchronous) are kept in the same list to reduce storage, and are used for creating *snapshots* of the particle locations at selected times. PTRAX uses dynamic memory allocation and optimized data structures so that very large systems can be modeled with modest hardware within a reasonable time frame (e.g., 20 MB RAM for 1,000,000 particles in under 3 hours on a Pentium 90).

## REFERENCES


Davis, J. C., 1986, *Statistics and Data Analysis in Geology*, John Wiley & Sons, New York.

Frind, E. O. and G. B. Matanga, 1985, "The Dual Formulation of Flow for Contaminant Transport Modeling," Water Resources Research, 21:2, pp. 159-169

Matanga, G. B., 1993, "Stream Functions in Three-Dimensional Groundwater Flow," Water Resources Research, 29:9, pp. 3125-3133.

Prickett, T. A., T. G. Naymick, and C. G. Lonnquist, 1981, "A

*Random Walk* Solute Transport Model for Selected Groundwater Quality Evaluations," Illinois State Water Survey Bulletin 65.

Yeh, G. T., 1981, "AT123D: Analytical Transient One-, Two-, and Three-Dimensional Simulation of Waste Transport in the Aquifer System," Oak Ridge National Laboratory Report 5602, Oak Ridge, Tennessee.

USGS, 1989, "Documentation of Computer Programs to Compute and Display Pathlines Using Results from the U. S. Geological Survey Modular Three-Dimensional Finite-Difference Ground-Water Flow Model," USGS Report 89-381

Young, S. C., 1995, "Verification of PTRAX," Report prepared for Martin Marietta Energy Systems, Oak Ridge, Tennessee, by Environmental Consulting Engineers, Knoxville, Tennessee.

Zheng, C., 1990, "PATH3D Version 2.0 User's Manual," S. S. Papadopulos and Associates, Rockville, Maryland.

**Table 1. Comparison of Numerical Model and Analytical Solution**

| Test Case | Dispersivity | | | Model | 15 Years | | | | | | 30 Years | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\alpha_X$ | $\alpha_Y$ | $\alpha_Z$ | | X | Y | Z | $\sigma_X$ | $\sigma_Y$ | $\sigma_Z$ | X | Y | Z | $\sigma_X$ | $\sigma_Y$ | $\sigma_Z$ |
| 1 | 3 | 3 | 0.3 | Numerical | 109.4 | 0.4 | 0 | 25.6 | 25.8 | 8.0 | 218.7 | 0.8 | 0 | 37.1 | 36.9 | 11.3 |
| | | | | Analytical | 109.5 | 0 | 0 | 25.6 | 25.6 | 8.1 | 219.0 | 0 | 0 | 36.2 | 36.2 | 11.4 |
| 2 | 3 | 0.3 | 0.3 | Numerical | 109.3 | 0.1 | 0 | 25.9 | 8.7 | 8.1 | 218.7 | 0.3 | 0.1 | 37.1 | 12.0 | 11.4 |
| | | | | Analytical | 109.5 | 0 | 0 | 25.6 | 8.1 | 8.1 | 219.0 | 0 | 0 | 36.2 | 11.4 | 11.4 |
| 3 | 12 | 12 | 0.3 | Numerical | 109.1 | 0.5 | 0 | 51.1 | 51.1 | 8.1 | 217.7 | 1.4 | 0.1 | 73.5 | 72.8 | 11.4 |
| | | | | Analytical | 109.5 | 0 | 0 | 51.3 | 51.3 | 8.1 | 219.0 | 0 | 0 | 72.4 | 72.4 | 11.4 |
| 4 | 12 | 1.2 | 0.3 | Numerical | 109.5 | 0.3 | 0.2 | 50.6 | 16.4 | 8.2 | 218.2 | 0.5 | 0.1 | 72.7 | 23.3 | 11.4 |
| | | | | Analytical | 109.5 | 0 | 0 | 51.3 | 16.2 | 8.1 | 219.0 | 0 | 0 | 72.4 | 22.9 | 11.4 |

**Notes:** **Numerical** indicates the results of PTRAX
**Analytical** indicates the results of the AT123D analytical model
**X** is the mean distance traveled by the plume in the X direction (i.e., the first moment of the mass about X)
**Y** is the mean distance traveled by the plume in the Y direction (i.e., the first moment of the mass about Y)
**Z** is the mean distance traveled by the plume in the Z direction (i.e., the first moment of the mass about Z)
$\boldsymbol{\sigma_X}$ is the X dispersion (i.e., the second moment with respect to X of the mass about X)
$\boldsymbol{\sigma_Y}$ is the Y dispersion (i.e., the second moment with respect to Y of the mass about Y)
$\boldsymbol{\sigma_Z}$ is the Z dispersion (i.e., the second moment with respect to Z of the mass about Z)

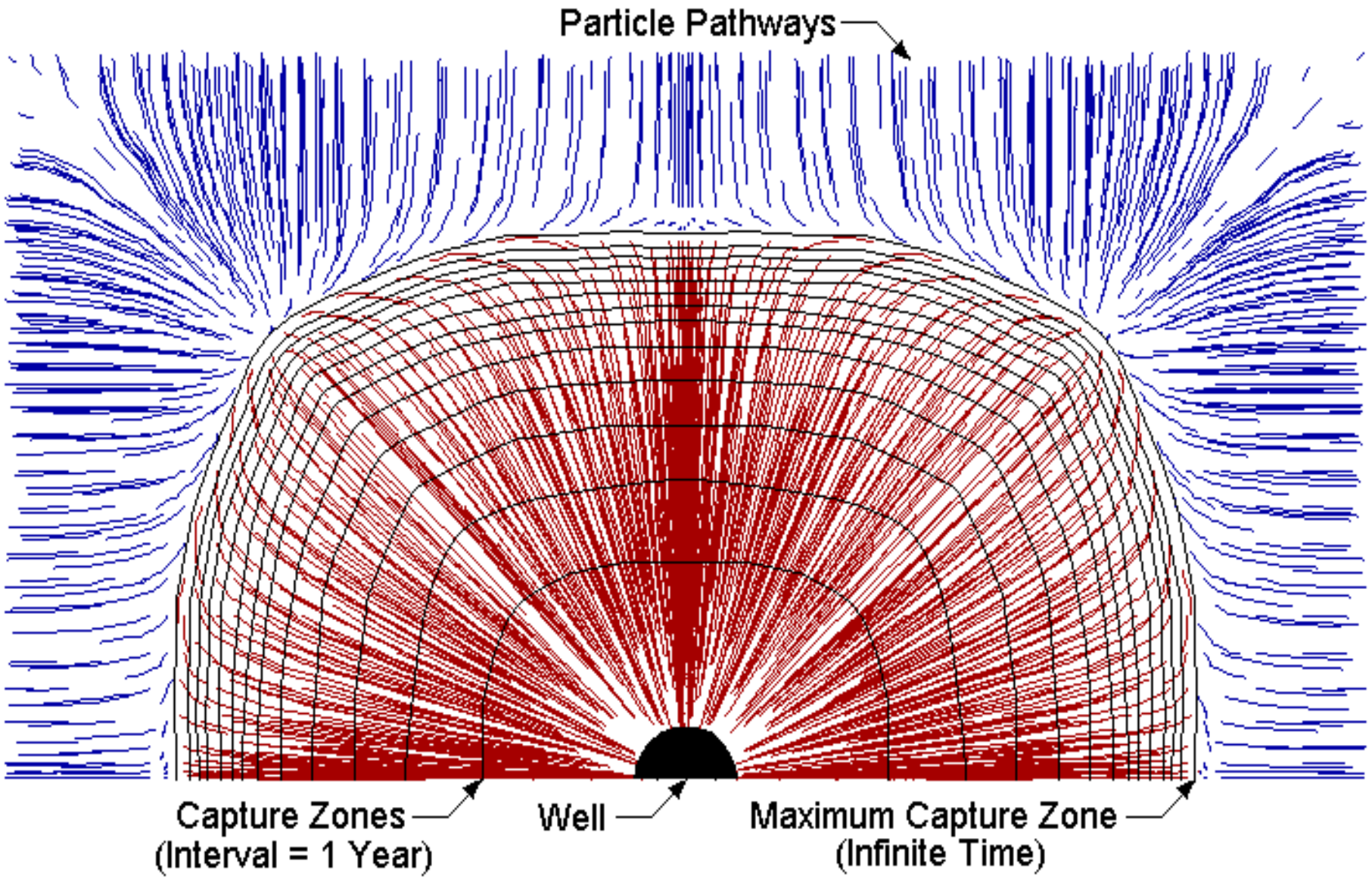


Fig 4. Particle track showing well capture

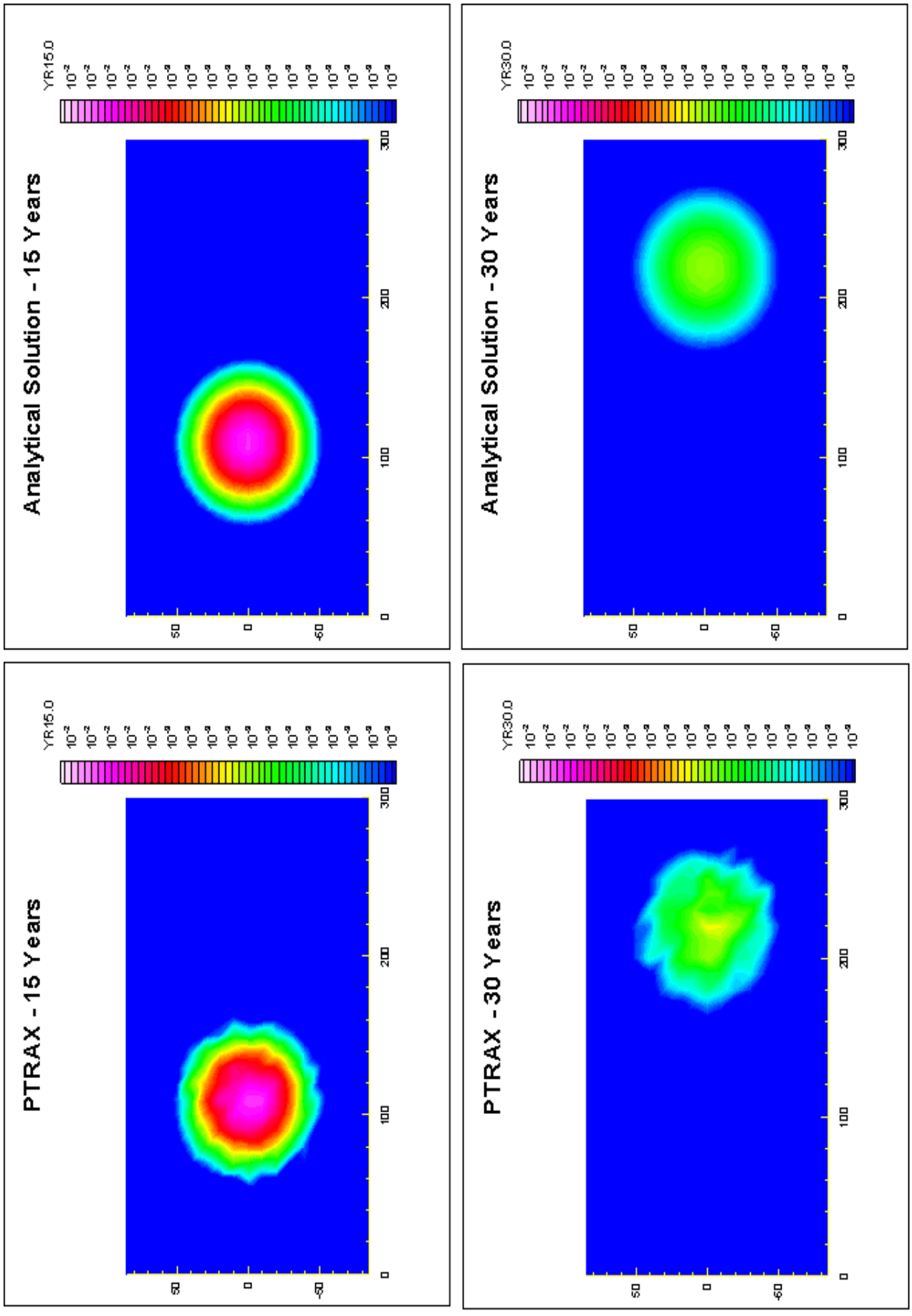


**Fig. 5. Dispersion Resulting from ax=ay=3m.**

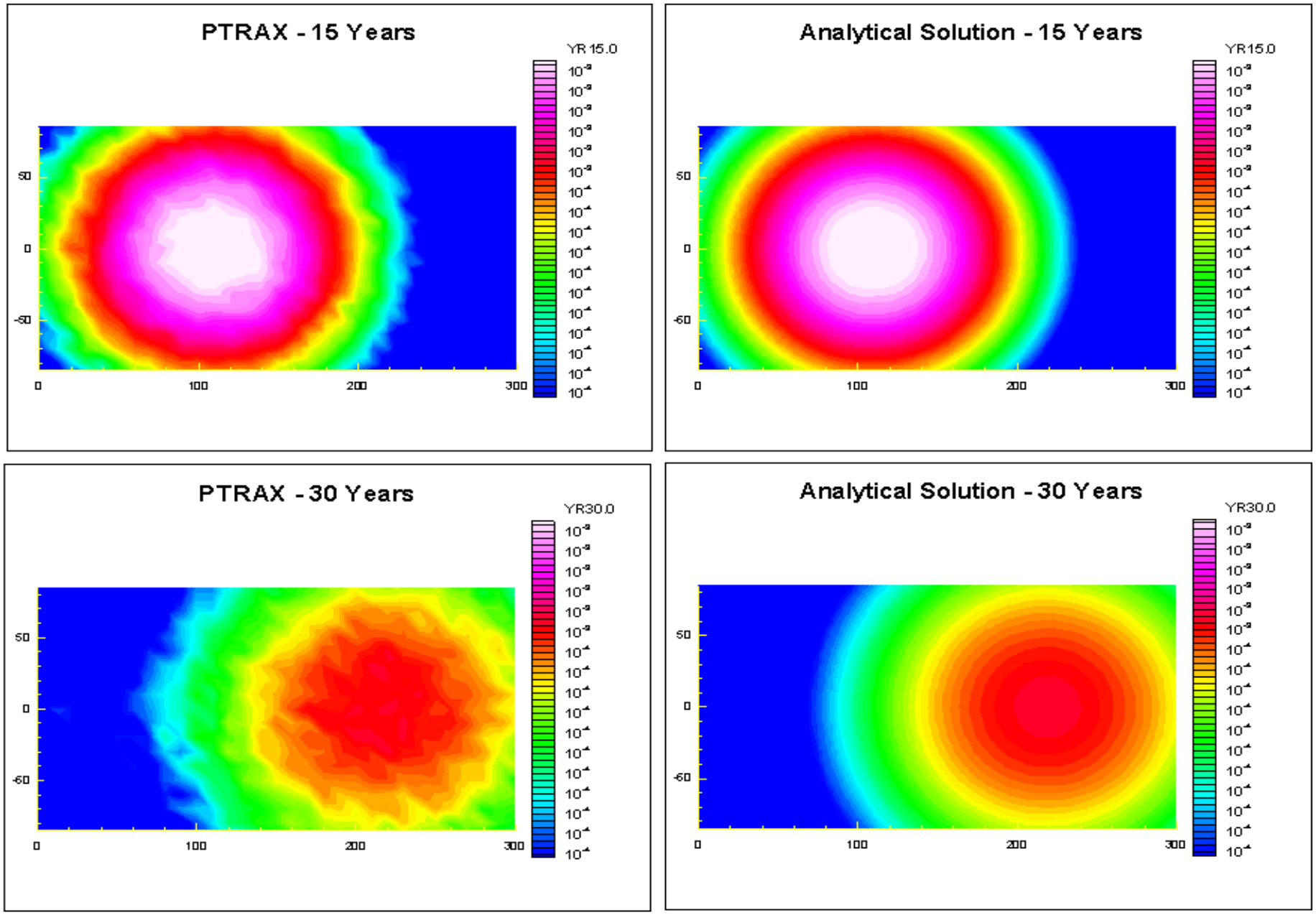


**Fig. 6. Dispersion Resulting from ax=ay=12m.**

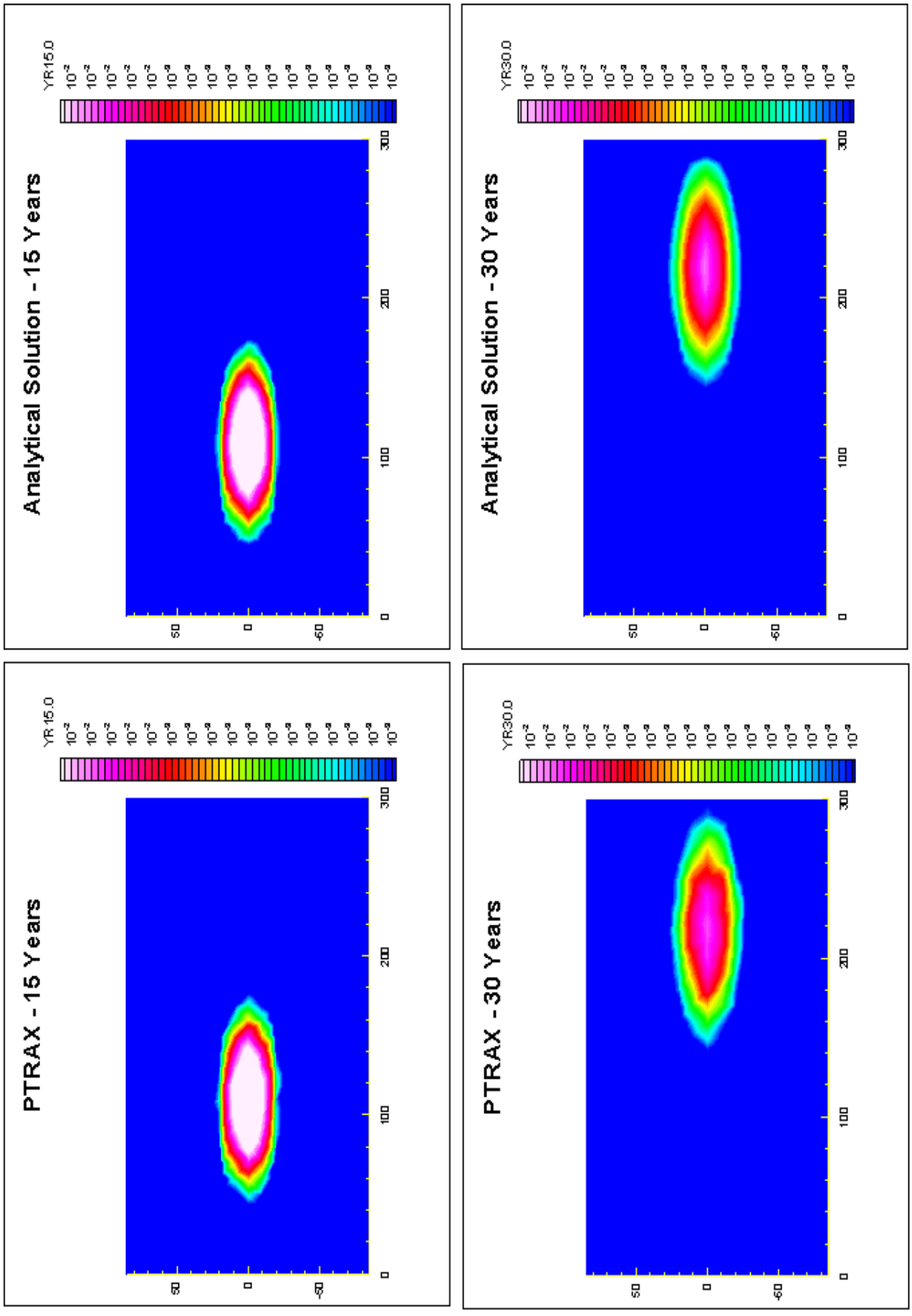


**Fig. 7. Dispersion Resulting from ax=3m, ay=0.3m.**

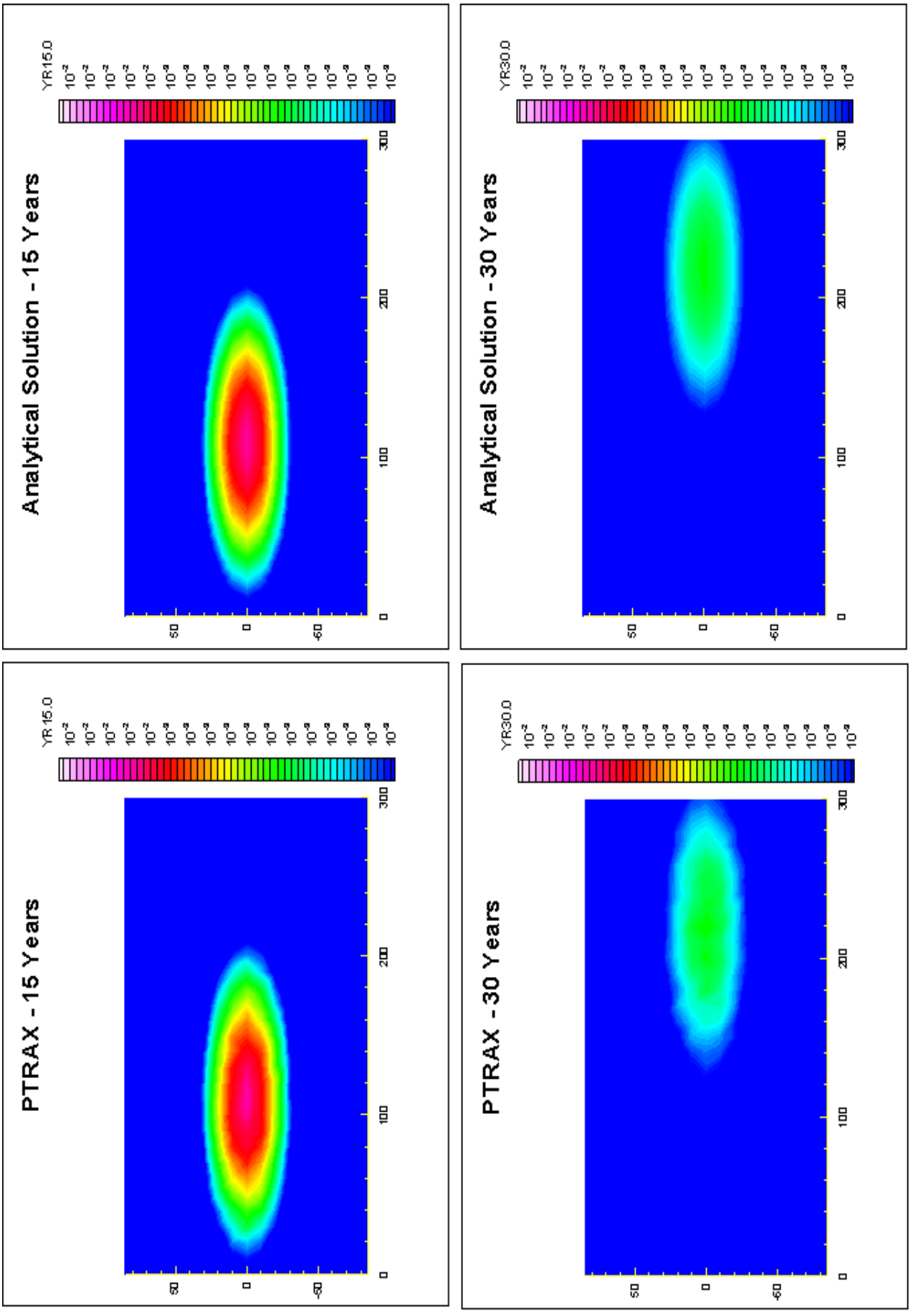


**Fig. 8. Dispersion Resulting from ax=12m. ay=0.3m.**