Conditioning Simulation to Block Data with Application to Tomography and Downscaling

Yongshe Liu and Andre G. Journel

Stanford Center for Reservoir Forecasting
Petroleum Engineering Department
Stanford University

February 27, 2006

Abstract

The challenge of integrating data with very different support volumes is addressed. Based on the theory of block kriging and direct sequential simulation, a C++ program bdsim (block data conditioned simulation) was developed within SGEMS (Stanford Geostatistics Modeling Software). Several important implementation problems such as simulation path, data search and block covariance computation are discussed and solutions are proposed. The most critical issue in bdsim is the computation of block-related covariances. The traditional integration approach is not practical CPU-wise in presence of a large number of block data or blocks of large size. An analytical and a Fast Fourier Transform (FFT) approaches are proposed to improve computation efficiency. A curvilinear ray tomography test case is used to test the performance of bdsim. Additional case applications such as VSP, downscaling and conditioning to arbitrary-shaped block are also presented.

1 Introduction

In earth sciences, data with different support volumes of large and small scales may be presented and must be integrated. The large support data, such as tomographic data, VSP data and remote sensing data, are hereafter called "block" data. The small support data, such as core data and well data, are called point data. All block data are assumed linear averages of the point values inside their respective support volumes. The goal of this study is to generate stochastic images at the smallest point resolution conditioned to both point and block data. The stochasticity of the images provides an uncertainty assessment.

The methodology for block data conditioning and the implementation of the original Fortran program visim (Volume data Incorporation Simulation) were presented in previous reports (Hansen et al. 2004, Liu 2005). In this report, the theory of block simple kriging and direct sequential simulation algorithm is recalled briefly. Then, the development of the new C++ program bdsim (Block Data conditioned Simulation) is presented. As different from visim, several new techniques such as a user-friendly GUI, a better data search and a fast block covariance calculation are implemented in bdsim to its improve its efficiency.

In the original code visim, the traditional integration (discrete summation) method is used for the block covariance calculation. This approach becomes too expensive as the block discretization increases or the number of blocks increases as would be the case in 3D applications . This traditional method is retained as an option in bdsim, but it is complemented by the much faster analytical and Fast Fourier Transform (FFT) approaches. bdsim can handle straight and curvilinear ray paths, and more generally any data defined on any block shape. Beyond tomographic inversion, diverse applications of bdsim are presented, such as Vertical Seismic Profile (VSP) inversion, grid downscaling and arbitrary shape block data integration.

2 Recall of theory

In this section, we recall the theory of block data conditioning (Journel & Huijbregts 1978, Tarantola 2005, Hansen et al. 2004, Liu 2005).

2.1 Block kriging

In block kriging, both block and point data are considered simultaneously.

Simple kriging estimation

Consider the SK estimation of the Z value at any location $\mathbf{u}$ in space (Deutsch & Journel 1998):

$$Z^*_{SK}(\mathbf{u}) - m_0(\mathbf{u}) = \sum_{\alpha=1}^{n(\mathbf{u})} \lambda_\alpha(\mathbf{u}) \cdot [D(\mathbf{u}_\alpha) - m_D(\mathbf{u}_\alpha)] \tag{1}$$

where $Z^*_{SK}(\mathbf{u})$ is the SK estimator of the unknown value $Z(\mathbf{u})$ at location $\mathbf{u}$

$m_0(\mathbf{u}) = E\{Z(\mathbf{u})\}$ is the known expected value of the random variable $Z(\mathbf{u})$

$\lambda_\alpha(\mathbf{u}), \alpha = 1, \cdots, n(\mathbf{u})$ are the SK weights which vary with $\mathbf{u}$

$D(\mathbf{u}_\alpha)$, $\alpha = 1, \cdots, n(\mathbf{u})$ are the point or block data retained in the neighborhood of location $\mathbf{u}$ for estimation of $Z(\mathbf{u})$.

$m_D(\mathbf{u}_\alpha) = E\{D(\mathbf{u}_\alpha)\}$ is the known expected value of the datum random variable $D(\mathbf{u}_\alpha)$. Under stationarity: $m_0(\mathbf{u}) = m_D(\mathbf{u}_\alpha) = m$, $\forall\ \mathbf{u}, D(\mathbf{u}_\alpha)$

Note that the number of data $n(\mathbf{u})$ and their configuration vary with the location $\mathbf{u}$. The notation of Expression(1) does not allow considering a single column vector of estimators $\mathbf{Z}^* - m_0$, because the data set retained in the kriging system changes for each location $\mathbf{u}$.

Each datum $D(\mathbf{u}_\alpha)$ is the linear average:

$$D(\mathbf{u}_\alpha) = \frac{1}{|v_\alpha|} \int_{v_\alpha(\mathbf{u}_\alpha)} L_\alpha(Z(\mathbf{u}'))\mathrm{d}\mathbf{u}' + R_\alpha, \quad \forall \alpha \tag{2}$$

where $v_\alpha(\mathbf{u}_\alpha)$ denotes the volume support of datum $D(\mathbf{u}_\alpha)$, $Z(\mathbf{u}')$ is the point variable at location $\mathbf{u}'$ within the block support v_α. The error term R_α and the averaging function L_α are defined hereafter.

Each datum $D(\mathbf{u}_\alpha)$ is seen as a spatial average of a known linear function $L_\alpha(\cdot)$ of point values $Z(\mathbf{u}')$. The averaging is typically limited to a finite volume v_α centered at location $\mathbf{u}_\alpha$. The function $L_\alpha(\cdot)$, however, can vary from one datum $D(\mathbf{u}_\alpha)$ to another $D(\mathbf{u}_\beta)$. For examples: v_α could be the 1D trace of a seismic ray; or v_α could be limited to the single point support "hard datum" at location $\mathbf{u}_\alpha$, in which case, $D(\mathbf{u}_\alpha) = Z(\mathbf{u}_\alpha) + R_\alpha$. That hard datum could be either an original hard sample $Z(\mathbf{u}_\alpha)$, or it could be a previously simulated value $D(\mathbf{u}_\alpha) = Z^{(l)}(\mathbf{u}_\alpha)$ during a sequential simulation process. There are always some noise R_α in any data. The following assumptions are considered for the noise term R_α:

$$\text{independent with the signal:} \quad R_\alpha \perp Z(\mathbf{u}_\alpha), \forall \mathbf{u} \tag{3}$$

$$\text{spatial white noise:} \quad R_\alpha \perp R_\beta, \forall \alpha \neq \beta \tag{4}$$

Expression(3) means homoscedasticity, i.e. the noise is independent of the true value $Z(\mathbf{u}_\alpha)$ being measured. Expression(4) means that the errors are not correlated in space. Even if R_α, R_β relate to the same functional $L_\alpha = L_\beta = L$ and correspond to different locations $\mathbf{u}_\alpha \neq \mathbf{u}_\beta$, the errors are assumed hereafter homoscedastic and uncorrelated with:

- zero mean: $E\{R_\alpha\} = 0$, $\forall \alpha$
- known variance: $Var\{R_\alpha\} = \sigma^2_{R_\alpha}$, possibly different for each datum, which could be obtained from a prior calibration of the different types of noise, hence
- known diagonal covariance matrix $\mathbf{C}_R = [Cov\{R_\alpha, R_\beta\}]$.

In this initial implementation, we consider that all data are noise free, i.e. $R_\alpha = 0, \forall \alpha$. The noise of the block data could be taken into account in future work (Goovaerts 1997, p. 172).

Stationarity decision

We assume that the study property, say velocity, is stationary within the study area. Thus, we have $m_0(\mathbf{u}) = m_0$, $\forall \mathbf{u}$.

The expected value of $D(\mathbf{u}_\alpha)$, $\mathbf{m}_D(\mathbf{u}_\alpha)$ can then be simplified:

$$\begin{aligned} m_D(\mathbf{u}_\alpha) = E\{D(\mathbf{u}_\alpha)\} &= \frac{1}{|v_\alpha|}\int_{v_\alpha(\mathbf{u}_\alpha)} L_\alpha(E\{Z_{v_\alpha}(\mathbf{u}')\})\mathrm{d}\mathbf{u}' + E\{R_\alpha\} \\ &= \frac{1}{|v_\alpha|}\int_{v_\alpha(\mathbf{u}_\alpha)} L_\alpha(m_0(\mathbf{u}'))\mathrm{d}\mathbf{u}' \\ &= L_\alpha(m_0') \\ &= m_0 \text{ if the averaging function } L_\alpha \text{ is the identity function} \end{aligned} \quad (5)$$

The SK expression (1) then reduces to:

$$Z^*_{SK}(\mathbf{u}) = \sum_{\alpha=1}^{n(\mathbf{u})} \lambda_\alpha(\mathbf{u}) \cdot D(\mathbf{u}_\alpha) + \left(1 - \sum_{\alpha=1}^{n(\mathbf{u})} \lambda_\alpha(\mathbf{u})\right) m_0 \quad (6)$$

This is the standard stationary SK expression.

Data-to-data covariance matrix

$$\mathbf{K} = [Cov\{D_\alpha, D_\beta\}]_{n(\mathbf{u})\times n(\mathbf{u})} = \left[\bar{C}_{D_\alpha D_\beta}\right] \quad (7)$$

with:

$$\begin{aligned} \bar{C}_{D_\alpha D_\beta} &= Cov\{D(\mathbf{u}_\alpha), D(\mathbf{u}_\beta)\} \\ &= \frac{1}{|v_\alpha|\cdot|v_\beta|}\int_{v_\alpha(\mathbf{u}_\alpha)} \mathrm{d}\mathbf{u}' \int_{v_\beta(\mathbf{u}_\beta)} Cov\{Z(\mathbf{u}'), Z(\mathbf{u}'')\}\mathrm{d}\mathbf{u}'' + Cov\{R_\alpha, R_\beta\} \end{aligned} \quad (8)$$

which is a known average covariance value, $\forall \alpha, \beta$.

We could distinguish the two types of data. If the first volume in Expression(8) reduces to point support: $D_\alpha = Z(\mathbf{u}_\alpha)$. The point data $Z(\mathbf{u}_\alpha)$ is denoted as P_α to simplify the notation. The second integral D_β corresponds to a block support v_β denoted: $D_\beta = B_\beta$, where B_β is the block data over the v_β. Thus the block-to-point covariance is written as:

$$Cov\{B_\alpha, Z(\mathbf{u}_\beta)\} = \frac{1}{|v_\alpha|}\int_{v(\mathbf{u}_\alpha)} Cov\{Z(\mathbf{u}'), Z(\mathbf{u}_\beta)\}\mathrm{d}\mathbf{u}' + 0 = \bar{C}_{B_\alpha P_\beta} \quad (9)$$

since: $Cov\{R_\alpha, Z(\mathbf{u}_\beta)\} = 0$, per homoscedasticity.

Similarly, if the two volumes in Expression(8) reduce to point support, we have:

$$Cov\{Z(\mathbf{u}_\alpha), Z(\mathbf{u}_\beta)\} = C\{\mathbf{u}_\alpha, \mathbf{u}_\beta\} = C_{P_\alpha P_\beta}, \quad \text{since } Cov\{R_\alpha, R_\beta\} = 0 \quad (10)$$

The previous data type distinction calls for blocking the data-to-data covariance matrix $\mathbf{K}$ into submatrices, with: $n(\mathbf{u}) = n_1(\mathbf{u}) + n_2(\mathbf{u})$, and $n_1(\mathbf{u})$ usually different from $n_2(\mathbf{u})$:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{PP} & \mathbf{K}_{PB} \\ \mathbf{K}_{PB}^t & \mathbf{K}_{BB} \end{bmatrix} = Cov\{\mathbf{D}^t, \mathbf{D}\} \tag{11}$$

Note that the error variance $Var\{R_\alpha^2\}$ affect only the diagonal of the matrices $\mathbf{K}_{PP}$ and $\mathbf{K}_{BB}$ because all error cross covariances are equal to zero per the error model (3) and (4).

Data-to-unknown covariance matrix

$$\mathbf{k} = Cov\left\{\mathbf{D}^t, Z(\mathbf{u})\right\}_{n(\mathbf{u})\times 1} = \left[\bar{C}_{D_\alpha P_0}\right] \tag{12}$$

where P_0 denotes the point value to be estimated at location $\mathbf{u}$. We can develop this expression similarly to Equation(11). Again, we would distinguish the two types of data:

$$\mathbf{k} = \begin{bmatrix} \mathbf{k}_P \\ \mathbf{k}_B \end{bmatrix} \tag{13}$$

with:

$$\mathbf{k}_P = \left[Cov\left\{Z(\mathbf{u}_\alpha), Z(\mathbf{u})\right\}\right] = \left[C_{P_\alpha P_0}\right]$$

$$\mathbf{k}_B = \left[Cov\left\{B_\alpha, Z(\mathbf{u})\right\}\right] = \left[\bar{C}_{B_\alpha P_0}\right]$$

.

SK estimator $Z^*_{SK}(\mathbf{u})$, under stationarity

$$Z^*_{SK}(\mathbf{u}) - \mathbf{m}_0 = \Lambda^t \cdot \mathbf{D} = \sum_{\alpha=1}^{n(\mathbf{u})} \lambda_\alpha(\mathbf{u}) \cdot [\mathbf{D}(\mathbf{u}_\alpha) - m_0] \tag{14}$$

with:

$$\Lambda^t = [\lambda_\alpha(\mathbf{u}), \alpha = 1, \cdots, n(\mathbf{u})]_{1\times n(\mathbf{u})} \quad : \quad \text{SK weights}$$

$$\mathbf{D} = \begin{bmatrix} \mathbf{P} \\ \mathbf{B} \end{bmatrix} = \begin{bmatrix} P_\alpha - \mathbf{m}_0 \\ B_\beta - \mathbf{m}_0 \end{bmatrix} \quad : \quad \text{data of types } P \text{ and } B.$$

SK system

$$\mathbf{K} \cdot \Lambda = \mathbf{k} \tag{15}$$

Thus, we have:

$$\Lambda = \mathbf{K}^{-1} \cdot \mathbf{k}, \quad Z^*_{SK}(\mathbf{u}) - m_0 = \mathbf{k}^T \cdot \mathbf{K}^{-1} \cdot \mathbf{D}$$

With:

$$\mathbf{k}^t = \begin{bmatrix} \mathbf{k}_P & \mathbf{k}_B \end{bmatrix}_{1\times n(\mathbf{u})}, \quad \mathbf{D} = \begin{bmatrix} P_\alpha - m_0 \\ B_\beta - m_0 \end{bmatrix}_{n(\mathbf{u})\times 1},$$

$$\mathbf{K}^{-1} = \mathbf{W} = \begin{bmatrix} W_{PP} & W_{PB} \\ W_{PB}^t & W_{BB} \end{bmatrix}_{n(\mathbf{u})\times n(\mathbf{u})}$$

We write the SK estimator as:

$$Z^*_{SK}(\mathbf{u}) = m_0(\mathbf{u}) + \begin{bmatrix} \mathbf{k}_P & \mathbf{k}_B \end{bmatrix} \cdot \begin{bmatrix} W_{PP} & W_{PB} \\ W_{PB}^t & W_{BB} \end{bmatrix} \cdot \mathbf{D} \tag{16}$$

SK variance

The SK or estimation variance is then written as:

$$\begin{aligned} \sigma^2_{SK}(\mathbf{u}) = Var\left\{Z(\mathbf{u}) - Z^*_{SK}(\mathbf{u})\right\} &= Var\left\{Z(\mathbf{u})\right\} - \Lambda^t \cdot \mathbf{k} \\ &= C(0) - \mathbf{k}^t \cdot \mathbf{k}^{-1} \cdot \mathbf{k} \qquad (17) \\ &= C(0) - \begin{bmatrix} \mathbf{k}_P & \mathbf{k}_B \end{bmatrix} \cdot \begin{bmatrix} W_{PP} & W_{PB} \\ W_{PB}^t & W_{BB} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{k}_P \\ \mathbf{k}_B \end{bmatrix} \end{aligned}$$

with: $C(0) = Var\left\{Z(\mathbf{u})\right\}$ equal to the field stationary variance.

2.2 Direct sequential simulation

Sequential Gaussian Simulation is a robust and widely-used algorithm for simulation of continuous reservoir properties within an homogeneous facies. However, one major drawback of sgsim is that its multinormal assumption requires to transform the data into normal space before simulation and perform back transform after simulation. The normal score transform is a nonlinear transform, thus is not applicable to our case because it would undo the linear averaging characteristics of all block average data. Thus, we need to work in the original space to preserve the linearity of block average. The solution is the algorithm of direct sequential simulation (dssim), whereby simulation is performed in the original data space and does not call for any multi-Gaussian assumption. We can use any distribution type (uniform, log-normal, etc) for anyone of the local conditional distributions as long as the conditional mean and variance identify the SK mean and variance. In the present bdsim, the log-normal distribution is used since in most real seismic cases the seismic properties have distributions close to being log-normal. The theory of dssim can be found in Journel (1994).

The related problem of dssim is that the simulated results may not reproduce the target histogram. Some post-processing, such as through the rank-preserving algorithm trans (Deutsch & Journel 1998), could be used to approximate the target histogram. Another solution (Soares 2001) consists of scaling the global target histogram to each local SK mean and variance. This latter solution has been implemented in SGEMS dssim code (Remy et al. 2007).

3 Implementation

The Block data conditioned simulation (bdsim) presented hereafter is based on the public domain Stanford Geostatistics Modeling Software SGEMS (Remy 2004, Remy et al. 2007) and the Geostatistical Template Library GsTL (Remy 2001). bdsim follows the same programming styles as other geostatistical plug-in within SGEMS, such as sgsim and cosgsim.

3.1 Tomography code

Seismic tomographic inversion is an application of block data conditioning. We recall the goal of this application. In the process of data integration for reservoir modeling, we may have to integrate valuable interwell tomographic information. Tomography data are different from well data or other seismic data by their volume support. Well data are quasi point-support data, each assumed representative of one grid node of the fine scale geocellular model being built, while crosswell tomography data are travel time (first arrivals) averaged over a ray path that is typically not rectilinear and of uncertain geometry (Sherrif & Geldart 1995). This is a type of block average data. Tomography data do not provide high resolution local information, but they can detect major heterogeneities between wells. We aim at improving velocity images by incorporating crosswell tomography data, then assessing the uncertainty of these images through stochastic simulation.

For the methodology presentation, only simple ray-tracing is used since the actual geometry of the rays, straight or curvilinear, does not impact the application of bdsim. Complex rays tracing can be built-in later as a plug-in into SGEMS using state-of-the-art ray-tracing freeware.

bdsim was originally coded for tomographical data integration. For that application, the block data are the seismic ray data. Some implementations are specific for this type of block data. However, the bdsim code is more general and can be applied to any type of block data conditioned simulation.

3.2 Flowchart

Figure 1 gives a flowchart of the algorithm. Like all other SGEMS plug-ins, in bdsim, there are two major functions, **initialize()** and **execute()**. The first one is used to load input parameter values, load grids, point and block data, set up covariance model parameters and set up the search neighborhood. The execution function is the essential part of the algorithm. Two different simulation paths, random path and block-first path, are allowed. Different search schemes are considered for point(P) and block(B) data. The different left hand side covariances, $C_{P_\alpha P_\alpha}$, $\bar{C}_{B_\alpha B_\alpha}$, $C_{P_\alpha P_\beta}$, $\bar{C}_{P_\alpha B_\beta}$ and $\bar{C}_{B_\alpha B_\beta}$ and right hand side covariances $C_{P_\alpha P_0}$, $\bar{C}_{B_\alpha P_0}$, are computed. The cokriging system is built and solved. At each simulation node, we compute the kriging mean and variance, from which a log-normal conditional cumulative distribution function (ccdf) is derived. A value is then simulated by drawing from that distribution. There is no normal-score transform involved because direct sequential simulation is used. The block data reproduction is checked and an E-type map is produced if multiple realizations are generated.

3.3 Simulation path

In most geostatistical simulation algorithms, the path visiting the grid to be simulated is random determined by a random seed. The randomness of the simulation path allows to explore the space of uncertainty through multiple different realizations. Thus, the fully random path is one option of `bdsim`. An average block data give additional information at locations located within that block. We may want to simulate those within-block nodes first such that the additional block data information gets spread faster to other nodes through sequential simulation.

More computation is involved with the previous block-first (partially random) path than with the fully random path. Figure 2 gives an implementation flowchart for the block-first simulation path. A block is randomly picked from the block data set. All the nodes within that block are simulated first in a random sequence. Then we move to the next informed block, perform point simulation within it, then move again until all informed blocks are simulated. Finally all the remainder nodes are simulated along a random path.

Figure 3 shows different simulation paths in the 18-ray cases presented in SCRF meeting, 2005 (Hansen et al. 2004, Liu 2005). The smaller (bluer) value means that the node is simulated earlier. Comparing Figure 3(a) with Figure 3(b), we note that from one realization to another the rays are picked in random sequence and the remainder nodes are simulated in random sequence. Figure 3(c) shows the fully random simulation path.

3.4 Data search

In all geostatistical algorithms, one typically retain only the closest n data, either original data or previously simulated values, to reduce the size of the kriging systems. The measure of closeness to the simulation location is the variogram distance, which accounts for possible anisotropy. For point data, it is done as follows. Given a simulation location and its search ellipse (or ellipsoid), all nodes within the search ellipse are sorted according to their variogram distances to the central location. Spiraling away from that center, grid nodes which carry point data are marked. If there is a datum or a previously simulated value on that node, that information is put into the point data neighborhood. The spiraling process is continued until we get the specified maximum number of conditioning data or we pass the search range.

As for block or ray data search, it is also constrained by a maximum number of conditioning rays and a ray search neighborhood. The previous point data search algorithm is adapted to allows both point data and ray data to be searched simultaneously, see Figure 4. The search starts from the simulation node (green dot in Figure 4). Then the search path spirals away from the green dot. Ray 1 is treated first since one of its nodes is found earlier than any node of the other two rays. Ray 2 and 3 are treated in subsequent searches.

Before presenting the implementation of that ray search, it is useful to understand how ray data are stored. In SGEMS, each node of the simulation grid is assigned an identification number, denoted as *node_id*, see Figure 5. This allows to assign or retrieve a property value at any location knowning its *node_id*. When we load the ray data and create a ray object, we assign to each ray a ray identification number, denoted as *ray_id*. For example in Figure 5, *ray_id* for **ray 1** is 1. Then we can store a ray path by recording all the *node_id*'s of the nodes along that ray into a *ray_id* set [21, 17, 18, 14, 15]. We can also retrieve the ray(s) passing by any specific node by attaching the *ray_id* to the *node_id*. For example, when we check *node_id* 6, we know that there is a ray, **ray 2**, passing through it if the *ray_id* 2 has been attached to the *node_id* 6. Similarly, we can retrieve **ray 1** and **ray 2** from *node_id* 18. In the C++ Standard Template Library(STL) (Austern 1998), the container, **multimap**, can handle this situation. A **multimap** can have two parameters, *key* and *object*, where *key* is used to map or look up the *object*. The *key* can relate to one or multiple *objects*. This fits our purpose to find the *ray_id* (considered as an *object*) from a given *node_id* (considered as *key*). In the **initialize()** function of bdsim, a ray **multimap** is created, which stores the ray path(*node_id*s) and ray average value information.

Figure 6 gives the flowchart for point and ray data search. Starting form the node to be simulated, we move to the next closest node in terms of variogram distance until we move out of both point and ray search ellipses, or until we reach their input maximum number of data. At each node location *node_id*, we perform two checks. One is whether this node is informed by a hard datum or a previously simulated value. The second check consists of going to the ray **multimap** to look up for ray(s) passing through the given *node_id*. This tells us whether that node is informed by ray data, if yes, how many rays and what they are. Ray data search is performed simultaneously to point data search. For example, rays 1, 2 and 3 in Figure 4 are found and added to the ray neighborhood as we visit their tangent point to the spiral. If we only want to retain the closest two rays, only ray 1 and 2 are retained. Note that this ray search scheme not only gives us the the rays within the search neighborhood but also sort them according to their distances to the simulation node (green dot in Figure 4). The details of the work flow is given in Figure 6. In some situations, we may want to specify different anisotropic search ranges for point and block data due to the different supports between point and block data or the different configurations of the block data. The present version of bdsim allows to specify different search ranges, which requires searching point and ray data separately. However, the test results show that the data search is still fast because ray data are searched the same way as the point data.

In order to honor any given block (ray) average data, all informed points within that block must be included into the point data neighborhood (Liu 2005, Hansen et al. 2004). Figure 7 shows the results of ray reproduction check for the 18-ray case study in the previous report (Liu 2005). If we include all informed nodes along the passing ray(s), the simulated ray average values are very close to the input ray data (Figure 7(a)). Otherwise, the block average data are not well reproduced, see Figure 7(b) in which we only retain the closest 20 informed points, whether or not they are located within the passing ray(s). If there are several rays passing through any one node and these rays are very long, the corresponding kriging matrix for this node becomes very large especially when most of the nodes along the passing rays are simulated. The solution could consist in simulating first the nodes intersected by more than one ray (Hansen et al. 2004) or cutting the long rays into shorter ray segments and honor each of them separately. Additional work is needed to investigate an acceptable compromise solution.

3.5 Block covariance computation

This section relates to issues involved when computing block covariances.

There are 3 types of covariances: the point-point covariance $C_{PP'}$, the point-to-block covariance $\bar{C}_{PB}$ and the block-to-block covariance $\bar{C}_{BB'}$. In the present bdsim implementation, we have abandoned the covariance-set method (Liu 2005, Remy 2001). The basic idea of this covariance-set method is to build a 2×2 table, which contains $C_{PP'}$, $\bar{C}_{PB}$, $\bar{C}_{BP}$ and $\bar{C}_{BB'}$, then chose one of the four according to the input data pair (point or block). It turns out that this table can be easily replaced by using overloading functions in C++ (Lippman & LaJoie 1999). For example, when we call the function **covariance**(*data1*, *data2*) to compute the covariance between *data1* and *data2*, where *data1* and *data2* are either point data or block data, the overloading functions allow to automatically select the proper covariance calculation equation based on the input data pair, *data1* and *data2*.

3.5.1 Traditional integration approach

Figure 8 shows how $\bar{C}_{PB}$ and $\bar{C}_{BB'}$ are computed in the traditional integration approach, based on the arithmetic average of the point covariance $C_{PP'}$ along the ray path:

$$\begin{aligned} \bar{C}_{PB} &= \frac{1}{n}\sum_{i=1}^{n} C_{PP_i} \\ \bar{C}_{BB'} &= \frac{1}{nn'}\sum_{i=1}^{n}\sum_{j=1}^{n'} C_{P_iP'_j} \end{aligned} \tag{18}$$

where n is the number of nodes in block B and n' is the number of nodes in block B'.

Figure 9 gives the cokriging system and the number of computation elements for the different covariance types. Since the covariance matrix on the left hand side is symmetric, only the upper triangle elements need to be calculated. The basic calculation element is the point-point covariance $C_{PP'}$. The other two covariance types, C_{PB} and $C_{BB'}$, are composites of this basic covariance, see Equation(18) and Figure 8. In Figure 9, the cokriging system is divided into 5 parts, Part **I** for $C_{PP'}$, Part **II** for $\bar{C}_{PB}$, Part **III** for $\bar{C}_{BB'}$, Part **IV** for $C_{P_0P'}$, and Part **V** for $\bar{C}_{P_0B}$, where $C_{P_0P'}$ and $\bar{C}_{P_0B}$ are the unknow-to-point and unknown-to-block covariances, respectively. We assume that there are n_1 number of point data and n_2 number of ray data and the basic computation time of $C_{PP'}$

is $\triangle t$. We denote the number of point data in block i as N_i. We express the computation time for each part in terms of $\triangle t$. If all rays are composed of the same number of nodes N, the computations for the 5 parts are given in the following table.

Part	**computation time**	**if each block has N nodes**
Part **I**	$\frac{(n_1+1)n_1}{2}\triangle t$	$\frac{(n_1+1)n_1}{2}\triangle t$
Part **II**	$n_1 \sum_{i=1}^{n_2} N_i \triangle t$	$n_1 n_2 N \triangle t$
Part **III**	$(N_1 \sum_{i=1}^{n_2} N_i + N_2 \sum_{i=2}^{n_2} N_i + ... + N_{n_2} N_{n_2})\triangle t$	$\frac{(n_2+1)n_2}{2} N \triangle t$
Part **IV**	$n_1 \triangle t$	$n_1 \triangle t$
Part **V**	$\sum_{i=1}^{n_2} N_i \triangle t$	$n_2 N \triangle t$

Building a cokriging system using the traditional integration approach is thus very CPU demanding. For the typical 18-ray case test, we have an average of 20 point data and 8 ray data in each neighborhood. The computation time and their percent contribution to the total time are shown below.

Part	**computation time**	**Percentage (%)**
Part **I**	$210\triangle t$	0.14
Part **II**	$10000\triangle t$	6.75
Part **III**	$137500\triangle t$	92.76
Part **IV**	$20\triangle t$	0.01
Part **V**	$500\triangle t$	0.34

92.8% of time is spent on computing $\bar{C}_{BB'}$ and 98.5% time is spent on computing the block-related covariances; this is the dominant part of the CPU cost of the whole simulation process. Note that incorporating the 10 block data does not increase the size of cokriging matrix dramatically. The covariance matrix only increases by 10 columns and 10 rows, which does not significantly affect the time taken to solve the kriging system. This approach is not feasible in presence of a large number of blocks or large-size blocks. It follows that one should focus on reducing the burden of computing the block-related covariances.

3.5.2 Fast Fourier Transfer approach

The FFT approach is an every efficient block covariance calculation approach, especially in presence of a large number of block data or blocks constituted by a large number of discretization cells. Like the traditional integration method, this approach can handle blocks with any arbitrary shape, see the associated SCRF 2006 report *Calculation of Average Covariance Using Fast Fourier Transform (FFT)* (Liu et al. 2006) for extensive details. Based on the recent work of Kyriakidis (Kyriakidis et al. 2005), a fast C++ code for FFT block average covariance calculation has been developed.

3.5.3 Analytical approach

Another solution to improve block covariance calculation is solving analytically the $\bar{C}$ integrals as proposed very early by Journel (Journel & Huijbregts 1978). The idea is to approximate all rays (straight or curvilinear) by a set of segments. Then the $\bar{C}$ for each point-to-segment or segment-to-segment can be derived derived analytically.

Analytical derivation

Point-to-segment covariance $\bar{C}_{PB}$ There are typically two types of average $\bar{C}_{PB}$ values, see Figure 10. Denote the point as O, the segment as AB, the projection O onto the line AB is O'. Figure 10(a) shows the case of $O' \in AB$ and Figure 10(b) shows the case $O' \notin AB$.

In Figure 10(a), the average value $\bar{C}_{PB}$ is written:

$$\bar{C}_{PB} = \bar{C}(O, AB) = \frac{l_1}{l}\bar{C}(O, O'A) + \frac{l_2}{l}\bar{C}(O, O'B), \quad \text{with } l = l_1 + l_2 \tag{19}$$

where $\bar{C}(O, O'A) = \frac{1}{l_1}\int_0^{l_1} C(\sqrt{x^2 + d^2}) \cdot dx = \varphi(d, l_1)$
and $\bar{C}(O, O'B) = \varphi(d, l_2)$, assuming an isotropic covariance $C(|\mathbf{h}|)$.

If $C(|\mathbf{h}|)$ is not isotropic, $C(\sqrt{x^2 + d^2})$ is replaced by $C(x, d)$.

The average covariance $\bar{C}_{PB}$ corresponding to Figure 10(b) is written:

$$\bar{C}_{PB} = \bar{C}(O, AB) = \frac{l_1}{l_1 - l_2}\bar{C}(O, O'A) - \frac{l_2}{l_1 - l_2}\bar{C}(O, O'B) \tag{20}$$

with $O'A = O'B + AB$ and $||AB|| = l_1 - l_2 > 0$,
where $\bar{C}(O, O'A) = \varphi(d, l_1)$ and $\bar{C}(O, O'B) = \varphi(d, l_2)$.

Segment-to-segment covariance $\bar{C}_{BB}$ The average covariance between two segments A_1B_1 and A_2B_2 (Figure 11) can be calculated by averaging the corresponding point-to-segment covariances of type $\bar{C}_{PB}$:

$$\bar{C}(A_1B_1, A_2B_2) = \frac{1}{l_1} \sum_{O \in \{A_1B_1\}} \bar{C}(O, A_2B_2) \tag{21}$$

where each of the $\bar{C}(O, A_2B_2)$ is calculated as before. In a 3D case, the two segments A_1B_1 and A_2B_2 are not necessarily coplanar, but the result (21) still applies.

Basic integral to solve The basic integral to solve is that involved in Equation (19), in the isotropic case:

$$\varphi(d, l) = \frac{1}{l} \int_0^l C(\sqrt{x^2 + d^2}) \cdot dx \tag{22}$$

Consider first the isotropic linear variogram: $\gamma(\mathbf{h}) = |\mathbf{h}|$ or equivalently the pseudo-covariance: $C(\mathbf{h}) = C - |\mathbf{h}|$, where C is an arbitrary constant to be canceled out (for example through ordinary kriging):

$$\begin{aligned}
\varphi(d, l) &= \frac{1}{l} \int_0^l \sqrt{x^2 + d^2} \cdot dx \\
&= \frac{1}{2l} [x\sqrt{x^2 + d^2} + d^2 \ln(x + \sqrt{x^2 + d^2})]|_{x=0}^{x=l} \\
&= \frac{1}{2l} [l\sqrt{l^2 + d^2} + d^2 \ln(l + \sqrt{l^2 + d^2}) - d^2 \ln d]
\end{aligned} \tag{23}$$

Consider the numerical examples corresponding to Figure 10(a) and Figure 10(b). For Figure 10(a), assuming $l_1 = l_2 = d = 1$, for $\gamma(\mathbf{h}) = |\mathbf{h}|$ we have:

$$\begin{aligned}
\bar{C}(O, AB) &= \frac{1}{2}\bar{C}(O, O'A) + \frac{1}{2}\bar{C}(O, O'B) \\
&= \varphi(1, 1) = \frac{1}{2}[\sqrt{2} + \ln(1 + \sqrt{2})] \\
&= 1.15
\end{aligned}$$

For Figure 10(b), assuming $l_1 = 2$ and $l_2 = d = 1$, we have:

$$\bar{C}(O, AB) = 2 \times \bar{C}(O, O'A) - \bar{C}(O, O'B)$$

$$
\begin{aligned}
&= 2 \times \varphi(1,2) - \varphi(1,1) \\
&= 2 \times \frac{1}{4}(2 \times \sqrt{5} + \ln(2+\sqrt{5})) - \frac{1}{2}(\sqrt{2} + \ln(1+\sqrt{2})) \\
&= 1.8
\end{aligned}
$$

Implementation issues of the analytical approach

In order to evaluate the analytical $\bar{C}$ expression (23), several implementation issues must be addressed.

Distance of a point to a line The integral expression (23) calls for the projection distance d of a point onto a line, see Figure 10-11.

In Figure 12, we have a segment AB of length l, defining a line L. The coordinate vectors are, respectively, (A_x, A_y, A_z) and (B_x, B_y, B_z). The point O has coordinates (O_x, O_y, O_z), and the point-to-line distance is denoted d. In both 2D and 3D cases, we can use the cross-product to directly compute the distance between any point O and line L. Consider the parallelogram defined by the vectors $\mathbf{u} = B - A$ and $\mathbf{v} = O - A$. The area of the parallelogram is $|\mathbf{v} \times \mathbf{u}|$. Consider the segment AB as the base and d as the height of the parallelogram. It comes:

$$d = \frac{|\mathbf{u} \times \mathbf{v}|}{|\mathbf{u}|} = \frac{|(B-A) \times (O-A)|}{l}$$

For the 3D case, let $\mathbf{u} = (\mathbf{u_x}, \mathbf{u_y}, \mathbf{u_z})$, where $\mathbf{u_x} = B_x - A_x$, $\mathbf{u_y} = B_y - A_y$ and $\mathbf{u_z} = B_x - A_z$. Let $\mathbf{v} = (\mathbf{v_x}, \mathbf{v_y}, \mathbf{v_z})$, where $\mathbf{v_x} = O_x - A_x$, $\mathbf{v_y} = O_y - A_y$ and $\mathbf{v_z} = O_x - A_z$. The definition of cross-product gives:

$$
\begin{aligned}
\mathbf{u} \times \mathbf{v} &= \left(\begin{vmatrix} \mathbf{u_y} & \mathbf{u_z} \\ \mathbf{v_y} & \mathbf{v_z} \end{vmatrix}, \begin{vmatrix} \mathbf{u_z} & \mathbf{u_x} \\ \mathbf{v_z} & \mathbf{v_x} \end{vmatrix}, \begin{vmatrix} \mathbf{u_x} & \mathbf{u_y} \\ \mathbf{v_x} & \mathbf{v_y} \end{vmatrix} \right) \\
&= (\mathbf{u_y v_z} - \mathbf{u_z v_y}, \mathbf{u_z v_x} - \mathbf{u_x v_z}, \mathbf{u_x v_y} - \mathbf{u_y v_x})
\end{aligned}
$$

Thus the point-to-line distance d in the 3D case is:

$$d = \frac{|\mathbf{u} \times \mathbf{v}|}{|\mathbf{u}|} = \frac{\sqrt{(\mathbf{u_y v_z} - \mathbf{u_z v_y})^2 + (\mathbf{u_z v_x} - \mathbf{u_x v_z})^2 + (\mathbf{u_x v_y} - \mathbf{u_y v_x})^2}}{\sqrt{\mathbf{u_x}^2 + \mathbf{u_y}^2 + \mathbf{u_z}^2}} \tag{24}$$

For the 2D case, the coordinates of the 3 points are $A = (A_x, A_y, 0)$, $B = (B_x, B_y, 0)$ and $O = (O_x, O_y, 0)$. The cross-product is then written as:

$$\begin{aligned}
\mathbf{u} \times \mathbf{v} &= (\mathbf{u_x}, \mathbf{u_y}, 0) \times (\mathbf{v_x}, \mathbf{v_y}, 0) \\
&= (B_x - A_x, B_y - A_y, 0) \times (O_x - A_x, O_y - A_y, 0) \\
&= \left(0, 0, \begin{vmatrix} (B_x - A_x) & (B_y - A_y) \\ (O_x - A_x) & (O_y - A_y) \end{vmatrix}\right)
\end{aligned}$$

The point-to-line distance d in the 2D case is:

$$d = \frac{|\mathbf{u} \times \mathbf{v}|}{|\mathbf{u}|} = \frac{|(B_x - A_x)(O_y - A_y) - (B_y - A_y)(O_x - A_x)|}{\sqrt{(B_x - A_x)^2 + (B_y - A_y)^2}} \tag{25}$$

Approximation of curvilinear rays by ray segments All above analytical $\bar{C}$ calculations are based on segments. In a real situation, there may be curvilinear rays. We propose to approximate any such curvilinear ray by a series of ray segments. In Figure 13, the curvilinear ray AB is approximated by 3 ray segments: AC, CD and DB. Each of the 3 point-to-ray segment average covariance can be analytically calculated as above, then the point-to-full ray covariance can be retrieved by the proper weighted average.

A program independent of bdsim pre-processes curvilinear rays. The splitting of the rays need careful thinking. Should the ray be splitted into equal length segments or according to equal arc degree? How many segments should be considered? These decisions may be left to the user with corresponding input parameters and their default values.

Geometric anisotropy The classical correction for geometric anisotropy consists of a prior angle rotation and affinity multiplication of the coordinates (x, d) (Journel 2005).

In Figure 14, for a 2D case, we first rotate the horizontal axes X and Y into X' and Y' by an angle α to identify the azimuth angle of the anisotropy ellipse. Any point $O(O_x, O_y)$ is transformed into the point $O'(O'_x, O'_y)$ with:

$$\begin{bmatrix} O'_x \\ O'_y \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} O_x \\ O_y \end{bmatrix}$$

This transform does not change the length of any segment.

Next we stretch (or squeeze) the Y' coordinate by multiplying it by the affinity ratio $\lambda = a_1/a_1 > 1$ to transform the anisotropy ellipse into a isotropic circle; a_1 and a_2 are the long and short axes of the ellipse, respectively. The point O' is then transformed into O'' with coordinates:

$$\begin{bmatrix} O''_x \\ O''_y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \lambda \end{bmatrix} \begin{bmatrix} O'_x \\ O'_y \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\lambda\sin\alpha & \lambda\cos\alpha \end{bmatrix} \begin{bmatrix} O_x \\ O_y \end{bmatrix} \tag{26}$$

Equation (26) changes the node locations. For example O' is moved to O'' in Figure 14. Thus, lengths and distances are changed by this transform. Also right projection angles are not preserved by this transform. Therefore, we cannot obtain the new point-to-ray distance d'' by merely transforming the original distance d. The solution is actually much simpler: transform all grid nodes into isotropic field with equation (26), and then compute all distances d'' and covariances in that transformed space.

Block with arbitrary shape All expressions above relate to a curvilinear (1D) ray data. How should a block with arbitrary shape be handled is still an open issue. Two possible approaches could be considered. One is to approximate the blocks by a set of lines, see Figure 15(b). The other is to resample the block by sparse point data to reduce the CPU cost, see Figure 15(b).

This analytical approach has not been retained for coding within bdsim.

4 Tomographic inversion with curvilinear rays

In previous reports (Hansen et al. 2004, Liu 2005), we showed the tomographic inversion results conditioned to straight rays. In a real application, seismic rays are bended and curvilinear due to local heterogeneities. This curvilinear ray case is now considered.

For this study, we modify the previous reference field (Hansen et al. 2004, Liu 2005) by multiplying the peanut-shaped heterogeneity by a factor 1.5 in order to better detect it. Figure 16 shows the background velocity, the peanut with its higher values and the final combined reference model. The mean of the combined model is 3.15. From this model we extracted 3 sets of curvilinear rays, see Figure 17. Figure 17(b) gives the 10 lateral rays. Figure 17(c) gives the 8 vertical rays. Figure 17(d) gives the combination of all 18 rays. Note that the highest ray average data correspond to the two central vertical rays. Figure 18 gives the parameter settings for the lateral ray case. The settings for the other two cases are almost the same.

4.1 Different ray models

Figure 19 shows the E-type (point-wise average of multiple realizations) built from 20 realizations for these three curvilinear ray configurations. The lateral rays tend to expand the central ray high data value laterally (Figure 19(a)), while the vertical rays tend to be expanded vertically (Figure 19(c)). Since the lateral rays do not carry high value data, they do not result in a crisp detection of the peanut. The two central high value vertical rays result in a better detection. Combining both lateral and vertical rays, the peanut-shaped heterogeneity is well detected, Figure 19(e). Again, the means of the simulated realizations are higher than that of the reference model. We should not expect to obtain the peanut with an average value as high as the one in the reference model because of the averaging effect of rays. Increasing the length of the ray path make the rays less informative; the longer the ray paths, the smoother the E-type results.

4.2 Different simulation paths

In Figure 20 , we compare the results using a fully random path and a ray-first path. The differences appear negligible.

4.3 Number of data in neighborhood

Figure 21 gives the number of point data and ray data found within the neighborhood of each node location. No matter which simulation path is used, the nodes along the ray paths always have more point conditioning data because we include all closest n point data in the point data neighborhood and all previously simulated data along any ray passing through the simulation node. The red outliers (large number of found point data) in Figures 21(a) for the random path case are higher than those for the ray-first case (Figure 21(c)); see also the maximum value from their histograms in Figures 21(b) and 21(d). The reason is as follows. When we simulate a node located on ray(s), the point data are of two different types, first are the data within the point search neighborhood, second are previously simulated values along the ray(s) passing through the node. In the ray-first case, when we simulate the nodes along the rays, those nodes that are not located along any ray have not been simulated. Hence not many data of the first type are found. In the random path case, when we simulate a node along a ray, nodes not located along any ray may have been already simulated. Many point data of the second type may be found. Thus the ray-first scheme leads to smaller size kriging systems when simulating the nodes along rays. Also note that the outliers (red dots) in Figures 21(a) are located where rays cross each other because the more rays cross a simulation node, the more previously simulated nodes along these crossing rays are included as data. A solution is to simulate first the crossing nodes. The map of the number of block data found within the block data neighborhood is the same for both random and ray-first scheme (Figure 21(e)). In the central area of the simulation field, more ray data are found within the search range. If we decrease the size of search neighborhood, see the parameters in Figure 22, we observe more variation of the number of data found in the neighborhood, compare Figures 23 and 21. The mean number of data found decreases due to the smaller search neighborhood.

4.4 Run time analysis

Analyzing the CPU time needed by different parts of program will help understanding how the program works and improving the speed accordingly.

Figures 24(c) and 24(d) give the simulation time spent on each node when using all 18 rays. In the central part of the study area, simulation takes longer than in the edge. This is because there are more conditioning ray data there. Figures 24(e)and 24(f) shows that the simulation time is not significantly influenced by the different simulation paths.

In order to understand the time spent in different stages, we decompose the total run time into different parts. We always track the most expensive part, see the highlighted boxes in Figure 25. For example, **Total run time** is composed of **initialize** part and **execute** part. If we use all 18 rays as conditioning data, **initialize** part costs 47ms and **execute** part costs 99240ms. Thus we can neglect the first one. Again, we split the **execute** part into **search** and **estimation** and keep tracking the dominant time-consuming part. The flowchart of Figure 25 gives all the tracked parts and the percentage of time they take.

5 Case studies

In this section, we will present the new applications of bdsim to VSP, downscaling and irregular shape block data conditioning.

5.1 Vertical Seismic Profile (VSP)

VSP gives a high-resolution seismic image of the vicinity of the borehole. The sources are located at or very close to the surface and the receivers are located along the borehole. Because they are closely spaced through the area of interest, the seismic waves do not have to travel far and thus undergo less attenuation. There are several types of VSP, such as zero-offset VSP, offset VSP, directional VSP (Sherrif & Geldart 1995). We consider here offset VSP, corresponding to a set of sources located away from the borehole location with different offsets. Offset VSP can be used to detect heterogeneities, such a fault or reef, on one side of the borehole and the information does not apply to other remote areas. In this study, two different configurations are tested: vertical well and deviated well. Both well data (hard data) and ray average data are used for conditioning.

5.1.1 VSP with vertical well

Reference field

A 2D background field (vertical section) is created with 40×50 cells. The cell dimension is $0.025km \times 0.02km$. The velocity map is populated with a velocity histogram from the Stanford V dataset (Mao 1999) and a variogram model $\gamma(h) = 0.1 +$

$0.9Sph\left(\sqrt{\left(\frac{h_x}{0.6}\right)^2+\left(\frac{h_y}{0.2}\right)^2}\right)$. The resulting map and its histogram are given in Figures 26(a) and 26(b). Two high velocity heterogeneities (Figures 26(c) and 26(d)) are added into the previous background field. Figures 26(e) and 26(f) give the resulting reference velocity model and its histogram.

The left column data in the reference model is retained as well data, see Figure 27(a) and its histogram in 27(b). Three sources, **S1**, **S2** and **S3**, are located on the surface and four receivers, **R1**, **R2**, **R3** and **R4**, are located in the vertical well. The first arrival rays are obtained by tracing straight lines between sources and receivers (Figure 27(c)). Assume that the bottom of the reference field is a flat reflector. The reflected rays are traced between sources **S2** and **S3** and receivers **R1**, **R2** and **R3** (Figure 27(e)). The ray values are arithmetic averages of the point velocity values along the ray paths, see Figures 27(d) and 27(f). Note that the ray values are neither very high nor very low due to the averaging effect.

In practice, in order to detect water or gas intrusion, time-lapse VSP method could be used. In the early stage of water or gas injection, the sources are put closer to the well location since the fluid has not traveled far away. As the time lapses, sources are continuously put further away from the well location to track the fluid flow. Three cases are designed to mimic such time-lapse VSP application. In the first case, only the ray data from the source **S1** close to the well location is used (Figure 29(a)). Then the two sources **S1** and **S2** are used (Figure 30(a)). Last, all ray data from the three sources are used (Figure 31(a)). Figure 28 gives the basic bdsim parameter setting for these cases.

Results

For the first case, the values of the 4 ray data are low with mean 2.70 because their paths do not interest the high value heterogeneities. The middle-left high value spot appearing in the E-type map built from 20 simulated realizations (Figure 29(b)) results from the high value well data in that area. The E-type mean is lower than the reference mean (2.83) because of the low value ray data. We do see some high value heterogeneity spots at random locations in different realizations, see Figures 29(d) and 29(f), which indicates high uncertainty away from the ray data constraint.

In the second case, some rays, such as the first arrivals of **S2-R1** and **S2-R3**, carry valuable high value information. The E-type map (Figure 30(b)) reveals the actual high value heterogeneities on the top and middle-left of the section. The mean of E-type velocity increases to 2.86. This indicates that the additional information from shot 2 is valuable due to its better ray coverage of the field. The lower-right high value ball can not be detected because no ray passes through it. In different realizations (Figures 30(d) and 30(f)), the locations of the simulated high value heterogeneities reflect uncertainty as they do not have the crisp edges of the reference high value heterogeneities.

In the third case, the ray coverage increases further as the ray data of shot 3 are included. The simulated results are improved in the areas covered by shot 3 (Figure 31(b)). Note that the middle-left high value spot expands toward the right side. Again, the lower-right ball can not be seen, even though the simulated realization 2 (Figure 31(f)) reveals the potential for high value heterogeneities in the lower right corner.

Because the reflected rays travel along longer paths than the first arrivals, their average velocity values are closer to the mean value of reference model (2.83), see Figure 27. In this sense, the reflected rays are not as informative as the first arrivals.

5.1.2 VSP with deviated well

VSP data with deviated well could give better results because it increases the coverage areas of rays. In this study, we will test the deviated well case using another reference velocity field.

Reference field

A 2D vertical field is created with 50×50 cells. The cell dimension is $0.02km \times 0.04km$. Rectangular cells are used because in most practical geophysical situations, better resolution is obtained in the vertical direction. The velocity map is populated with a velocity histogram from the Stanford V dataset and the variogram model

$$\gamma(h) = 0.1 + 0.9Sph\left(\sqrt{\left(\frac{h_x}{1.0}\right)^2 + \left(\frac{h_y}{0.5}\right)^2}\right)$$

The resulting background section and its histogram are given in Figures 32(a) and 32(b). Three high velocity heterogeneities (Figure 32(c) and 32(d)) are added into that background velocity field. Figures 32(e) and 32(f) give the resulting overall reference velocity model and its histogram.

A deviated well is drilled from the top left side of the field, see the well path and the well data histogram in Figures 33(a) and 33(b). Note that this well path misses all three high value heterogeneities. Three sources, **R1**, **R2** and **S3**, are located on the surface and five receivers, **R1**, **R2**, **R3**, **R4** and **R5**, are located along the deviated well. The rays data are obtained by tracing straight lines between sources and receivers (Figure 33(c)). The ray values are listed in Figure 33(d).

Results

We consider the four cases corresponding to conditioning to shot 1 ray data (Figure 35(a)), shot 2 ray data (Figure 36(a)), shot 3 ray data (Figure 37(a)) and, last, all shots ray data (Figure 38(a))). Figure 34 gives the basic bdsim parameter setting for these tests. The results are given in Figures 35, 36, 37 and 38. In each figure, we present the ray configuration, the E-type map, one realization and the check for reproduction of ray data values.

In case 1, the high value ray **S1R4** is bounded by the lower value rays **S1R5** and **S1R3** and the lower value deviated well data. This creates a well bounded ellipse-shaped high value area in the E-type map (Figure 35(b)). In case 2, the values of the 5 rays are close with the highest one being at the far right **S2R5**. The E-type map (Figure 36(b)) shows that the red area spreads over a large area starting around ray **S2R5**. In case 3, there is one lower value ray **S3R2** among other higher value rays. This results into the red high value area being cut into two parts in the E-type map (Figure 37(b)). Conditioning to all rays in case 4, we get three reasonably separate high value areas (Figure 38(b)) centered around the actual three heterogeneities of the reference field (Figure 32(e)). This results from the high value data carried by rays **S1R4**, **S2R5**, **S3R4** and **S2R1**, and the edges created by the low value rays **S3R2**, **S1R1**, **S1R5**. The E-type mean is close to the reference global mean 3.0. Again uncertainty is assessed by the different realizations (Figures 35(d), 36(d), 37(d), 38(d)). The ray data are well reproduced in all cases, see Figure 35(f), 36(f), 37(f), and 38(f).

5.2 Downscaling conditioned to point data

In many applications in earth sciences, we have data on different supports, small or large. We want to obtain a grid model defined on the smallest (point) support conditioned to all data of any support. This is known as the process of downscaling. The algorithm and code bdsim can address this problem if we consider the large support grid data as block data.

Reference field

The reference field built for the vertical well VSP case is used again, see Figure 39(a) and 39(b). In this field, a 2×5 coarse grid of block values is obtained by arithmetically averaging the corresponding fine grid cell values (Figure 39(c)). The values of the 10 blocks are listed in Figure 39(d). The two column of fine scale data on the left and right sides of the reference field are retained as well hard data, see locations in Figure 39(e) and the histogram in Figure 39(f).

The basic bdsim parameter settings for this study are given in Figure 40.

Results

Figure 41(a) gives the E-type map built from 20 simulated realizations. Since both the well data and all 10 block data are used for conditioning, the middle-left and lower-left high value heterogeneities are detected. The overall spatial distribution of the reference field (Figure 39(a)) is reflected in the E-type map. Note the averaging effect due to the block data: the values of the heterogeneities in the E-type map are not as high as those in the reference field. The smaller the support volume of the block data, the less smooth is the E-type image.

The different simulated realizations (Figures 41(c) and 41(e)) show less fluctuations than those seen in the VSP case studies because the 10 block data provide a full coverage of the study area. The mean of the simulated velocity in each realization of Figures 41(d) and 41(f) equals the mean of reference model 2.83 which is also the mean of all block data. This is because each of the block data is approximately reproduced (Figure 41(g)), hence the global mean is also reproduced.

This particular downscaling using the traditional integration covariance calculation approach is slow because there is a larger number of blocks in this case than in the other case studies. This problem is solved by using the FFT covariance method proposed in the second paper (Liu et al. 2006).

5.3 Conditioning to arbitrary shape blocks

Using the reference velocity field with a peanut heterogeneity in its center (Figure 42(a) and 42(b)), we extracted 5 block data of very different shapes, the four **SCRF** letters and a happy face in the center. These block "paths" and average values are shown in Figure 42(c) and 42(d). Using the bdsim parameter settings given in Figure 43, we perform simulation conditioned to the two bounding well data and the 5 average block values. Figure 44 gives two simulated realizations and the E-type map obtained from 20 realizations. From the E-type map, we can clearly identify the central high value discontinuity and the general pattern of high and low values of the reference field. The two realizations show fuzzy patterns, but both with a high value central part. The realization means (3.1, 3.2) are a bit higher than the reference mean 2.98 because the block data have a higher mean of 3.2.

The results for this block case are better than those obtained from the 18-ray case (Hansen et al. 2004, Liu 2005) because the five block data provide a full coverage with resolution superior to that of a ray crossing both high and low-valued areas.

We also tried different simulation paths, fully random or partially random (ray-first), see Figure 45. Their results turn out quite similar. In the ray-first case, we get a slightly better image because the ray information is used earlier in the simulation path, that information then conditions better the simulation at subsequent nodes. Both schemes reproduce well the ray data values.

In the present version of bdsim, we can handle three types of conditioning data: point data only, block data only and simultaneously point and block data. If we only uses point data, bdsim is not different from the traditional point support direct sequential simulation algorithm and code dssim (Remy et al. 2007). Figure 46 gives the results for different data conditioning. If we condition only to point data (the 2 wells), the high value discontinuity between the wells is not revealed, Figure 46(c). If we condition only to block data, even though the central peanut is identified, the upper right and lower left high value areas informed by well data are not revealed, Figure 46(e). If we condition to both well data and the central block data (smiling face) only, we get the good result of Figure 46(g). This confirms how critical an informative block datum is, as opposed to many non-informative blocks. The best result is obtained by conditioning to all wells and 5 block data, see Figure 46(i), but the improvement is only marginal.

6 Conclusions

Different approaches to simulation conditioned to block data were implemented in the code bdsim to improve its efficiency. Fully random and ray-first simulation path schemes are provided as two options. Theoretically, the ray-first scheme is preferable, although the differences were not found to be significant in our preliminary test results. The proposed scheme to search block and point data in the same manner makes the data search faster. More than 98% of CPU time is used for block-related covariance calculation in the present bdsim due to the slow traditional block covariance calculation approach. Analytical and FFT block covariance methods were proposed and will be implemented into bdsim. In the mean time, a standalone FFT block covariance C++ code has been developed and is presented in the companion SCRF 2006 paper *Calculation of Average Covariance Using Fast Fourier Transform (FFT).* The test results show that bdsim is a general program for simulation with any type of block data conditioning, such as tomography data, VSP, downscaling and arbitrary shape block data conditioning. The results honor both point data and block data and provide both an LS estimate (E-type) and an uncertainty assessment. One drawback is that the present bdsim alone can not reproduce a target histogram; this issue can be addressed by either a rank-preserving post processing of each simulated realization through a program such as the Gslib **trans** (Deutsch & Journel 1998), or by rescaling the target histogram to be the local SK mean and variance (Soares 2001).

7 Future work

The following items will be considered and implemented in future

- Integrate the newly developed standalone FFT block covariance program into bdsim.
- Implement a 3D code and provide 3D test case studies.
- Implement the local scaling of conditional distributions to allow reproduction of a target histogram.
- Implement Ordinary kriging approach in bdsim.
- Implement the analytical block covariance computing approach, and perform comparison tests.
- Apply bdsim to case study of tomographic and VSP inversion using more difficult synthetic data or real seismic data with curvilinear ray tracing.
- Explore new applications of bdsim.
- Find a faster way to import ray data into SGEMS and improve the GUI accordingly.

8 Acknowledgments

We would like to thank Dr. Youli Quan of the Geophysics department at Stanford University for his discussions and advices on the test case design. Also we would like to express gratitudes to Dr. Thomas Mejer Hansen at University of Copenhagen, Denmark, Professor Albert Tarantola at Institut de Physique du Globe de Paris, France for their suggestions. Professor Phaedon Kyriakidis provided the original Matlab code for FFT calculation of block covariances.

References

Austern, M. H. (1998), *Generic Programming and the STL*, Addison Wesley Longman, Inc.

Deutsch, C. V. & Journel, A. G. (1998), *GSLIB: Geostatistical Software Library and User's Guide*, second edn, Oxford Press, N. Y.

Goovaerts, P. (1997), *Geostatistics for Natural Resources Evaluation*, Oxford Press, N. Y.

Hansen, T. M., Liu, Y., Journel, A. G. & Tarantola, A. (2004), Geostatistical tomography, Technical report, Stanford Center for reservoir forecasting.

Journel, A. G. (2005), Geostatistics for spatial phenomena, *in* 'Stanford University course reader'.

Journel, A. G. & Huijbregts, C. J. (1978), *Mining Geostatistics*, first edn, Academic Press, London.

Kyriakidis, P. C., Schneider, P. & Goodchild, M. F. (2005), Fast geostatistical areal interpolation.

Lippman, S. B. & LaJoie, J. (1999), *C++ Primer*, third edn, Addison Wesley Longman.

Liu, Y. (2005), Crosswell tomographic inversion with block kriging, *in* 'the 18th SCRF annual meeting report'.

Liu, Y., Jiang, Y. & Kyriakidis, P. (2006), Calculation of average covariance using Fast Fourier Transform (fft), *in* 'the 19th SCRF annual meeting report'.

Mao, S. (1999), Multiple layer surface mapping with seismic data and well data, PhD thesis, Stanford University, Stanford, California.

Remy, N. (2001), Gstl: The geostatistical template library in c++, Master's thesis, Stanford University.

Remy, N. (2004), S-GEMS, A Geostatistical Earth Modeling Library and Software, PhD thesis, Stanford University.

Remy, N., Journel, A. G., Boucher, A. & Wu, J. (2007), *Stanford Geostatistics Modeling Software*, first edn.

Sherrif, R. E. & Geldart, L. P. (1995), *Exploration Seismology*, second edn, Cambridge

University Press, Cambridge.

Soares, A. (2001), 'Direct sequential simulation and cosimulation', *Mathematical Geology* **33**, 911–926.

Tarantola, A. (2005), *Inverse Problem Theory and Model Parameter Estimation*, SIAM, Philadelphia.

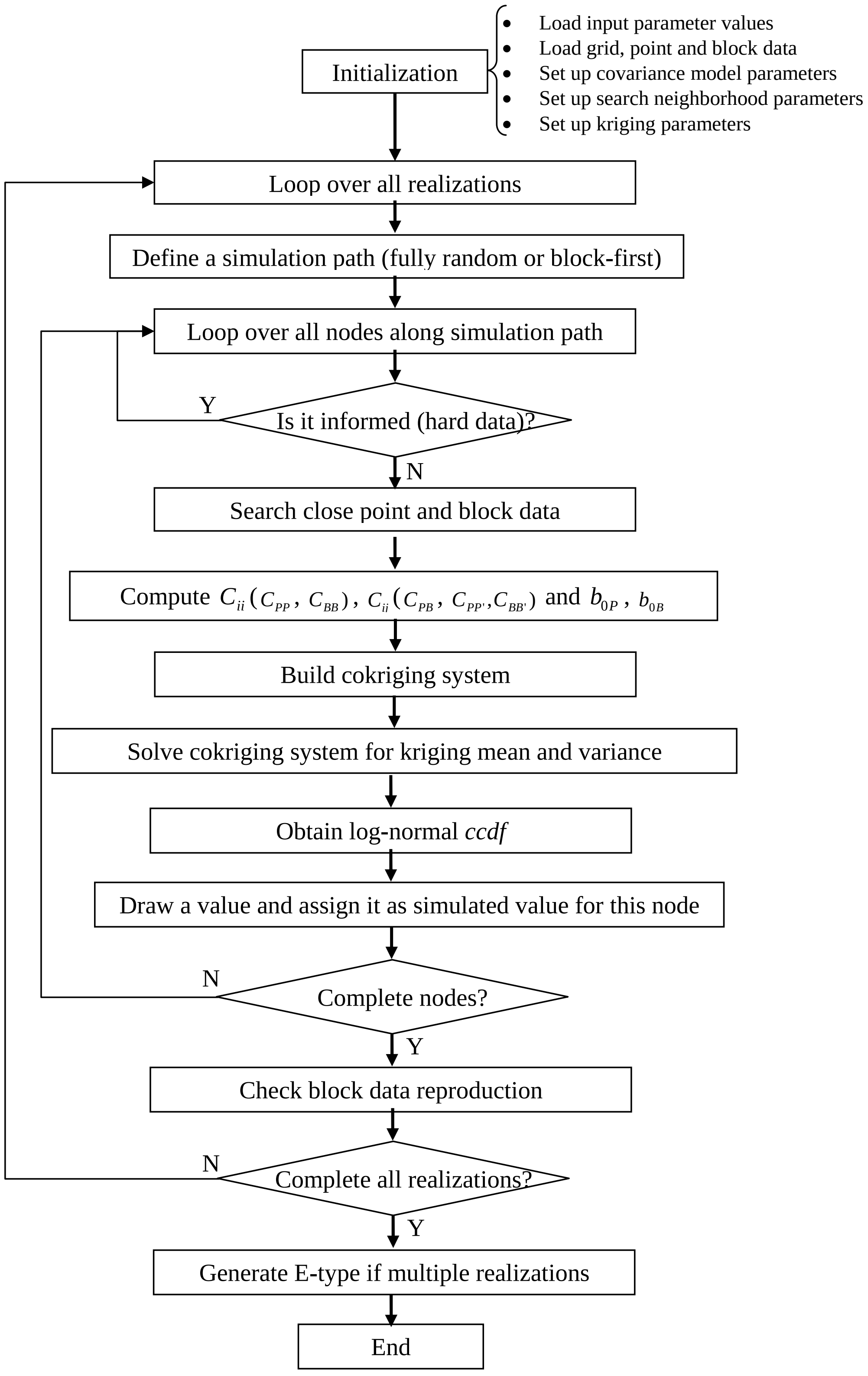

Initialization
Load input parameter values
Load grid, point and block data
Set up covariance model parameters
Set up search neighborhood parameters
Set up kriging parameters
Loop over all realizations
Define a simulation path (fully random or block-first)
Loop over all nodes along simulation path
Is it informed (hard data)?
Y
N
Search close point and block data
Compute $C_{ii}(C_{PP}, C_{BB})$, $C_{ii}(C_{PB}, C_{PP'}, C_{BB'})$ and b_{0P}, b_{0B}
Build cokriging system
Solve cokriging system for kriging mean and variance
Obtain log-normal *ccdf*
Draw a value and assign it as simulated value for this node
Complete nodes?
N
Y
Check block data reproduction
Complete all realizations?
N
Y
Generate E-type if multiple realizations
End

Figure 1: bdsim flowchart

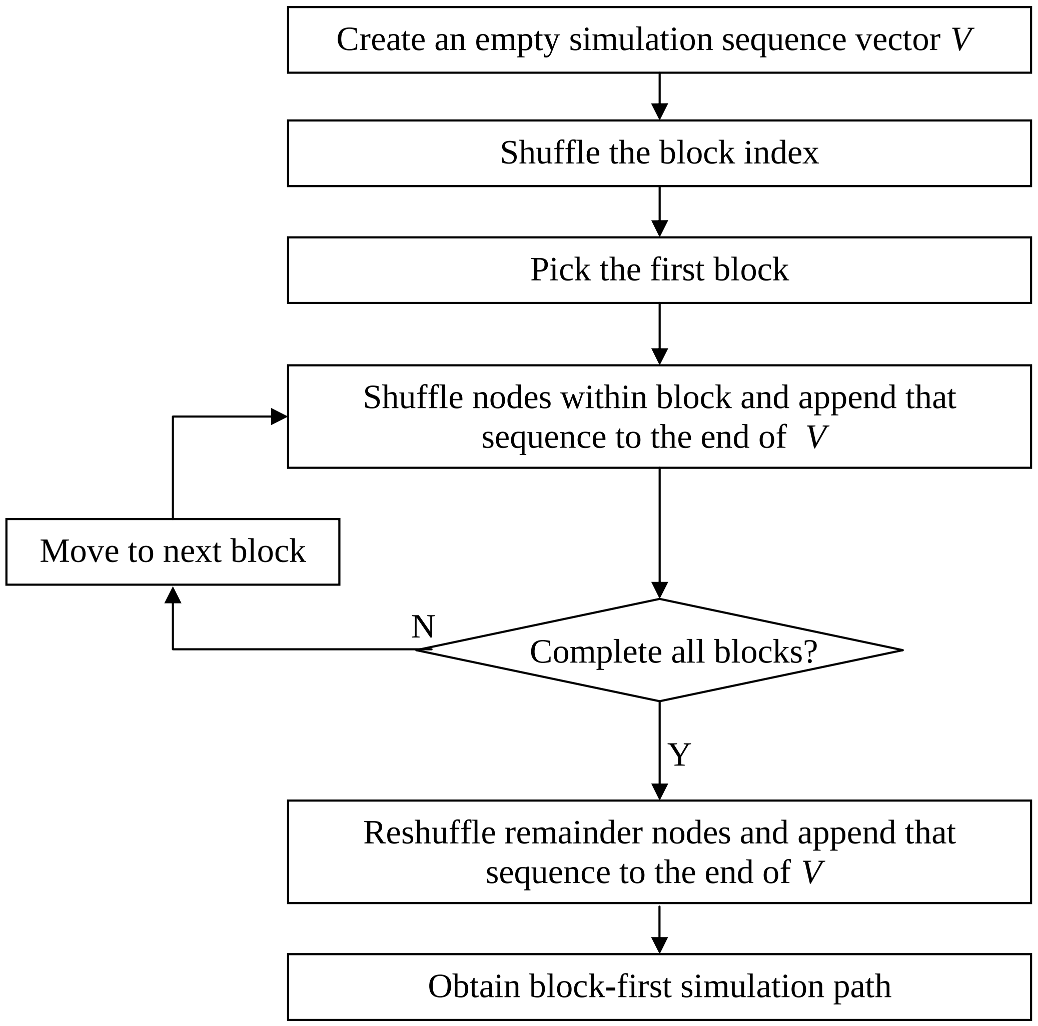
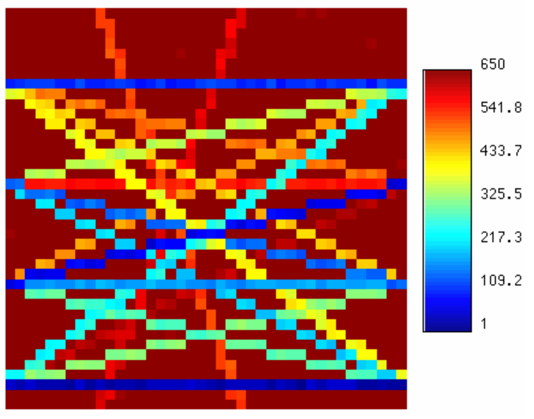
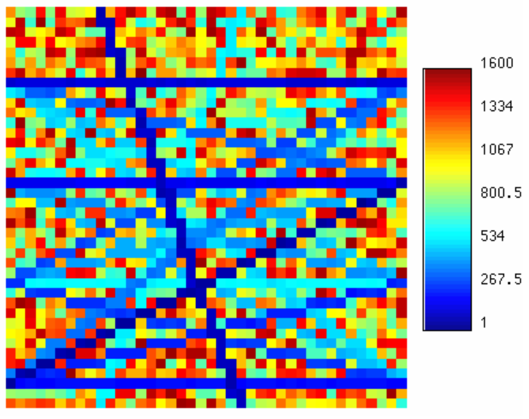

Create an empty simulation sequence vector V
Shuffle the block index
Pick the first block
Shuffle nodes within block and append that sequence to the end of V
Move to next block
N
Complete all blocks?
Y
Reshuffle remainder nodes and append that sequence to the end of V
Obtain block-first simulation path

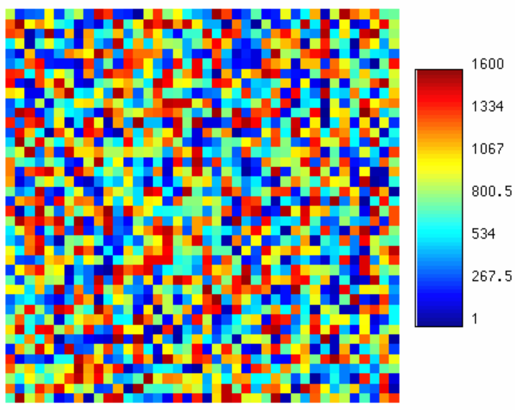
Figure 2: Flowchart of the block-first simulation path



(a) Color code of simulation sequence (ray-first case). The ray data locations are simulated first with each ray randomly selected. Within each ray, the nodes are simulated in random sequence.



(b) Color code of simulation sequence, with a larger color bar scale than Figure 3(a) (ray-first case). The nodes not located on any ray are simulated later in random sequence.



(c) Fully random simul. path

Figure 3: Color code of simulation sequence (The color indicates the order of visit)

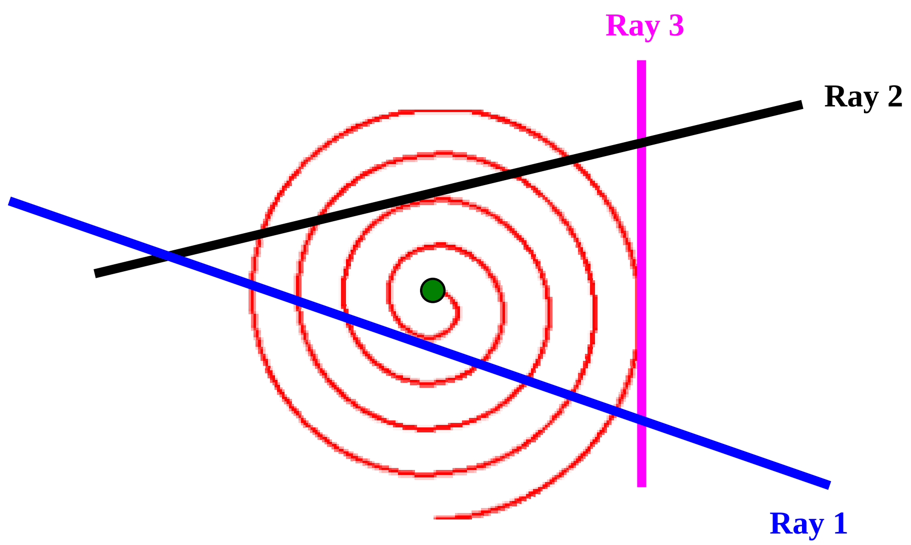


Figure 4: Spiraling search for point and ray data search

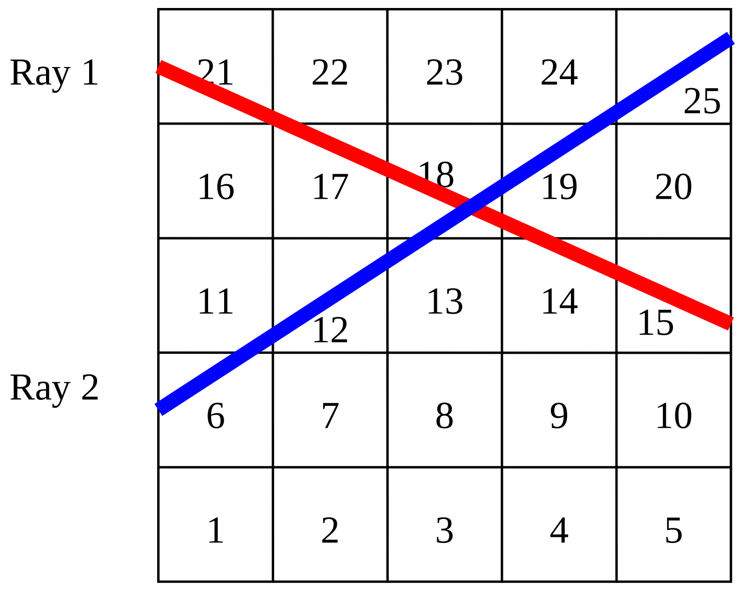
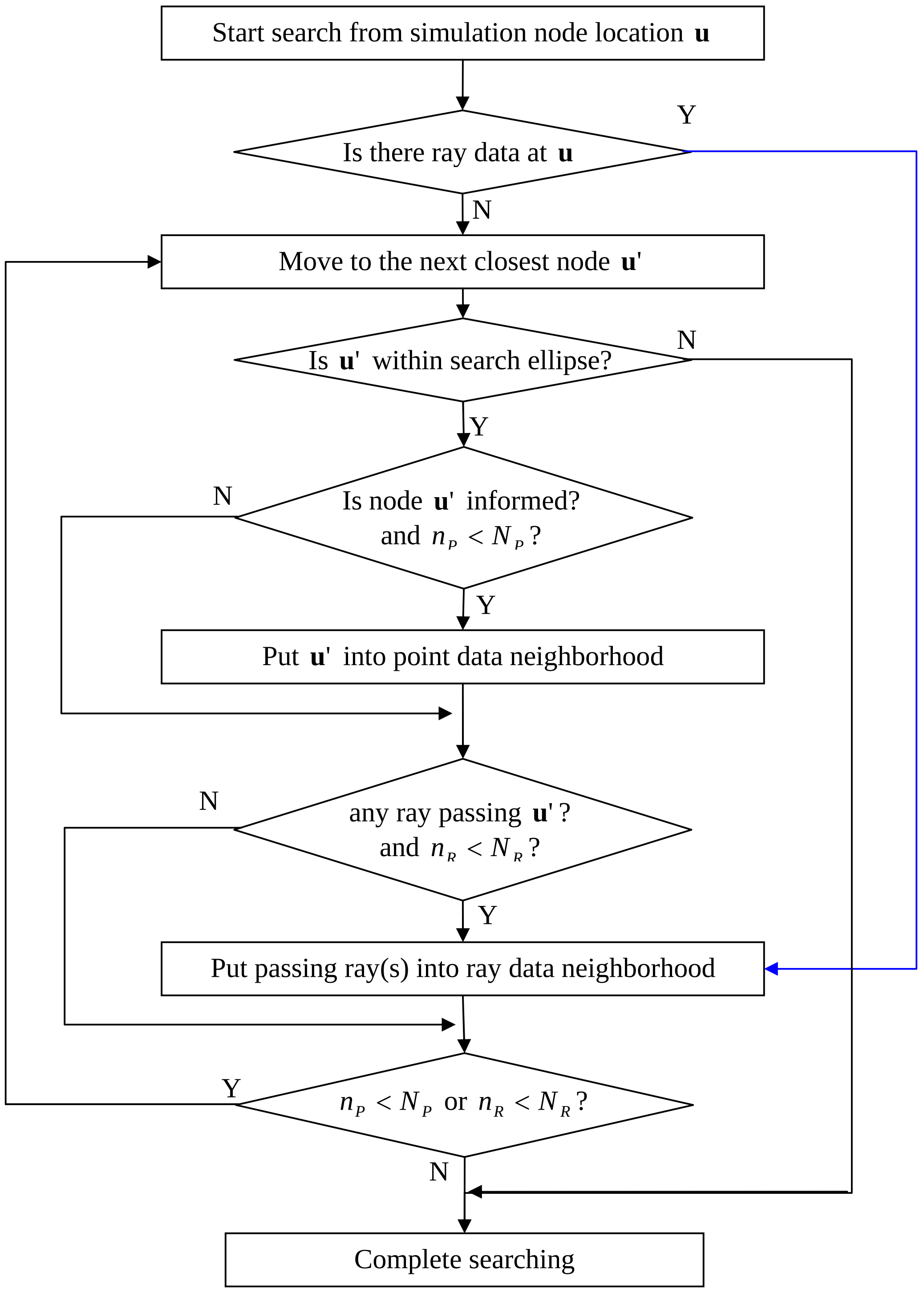
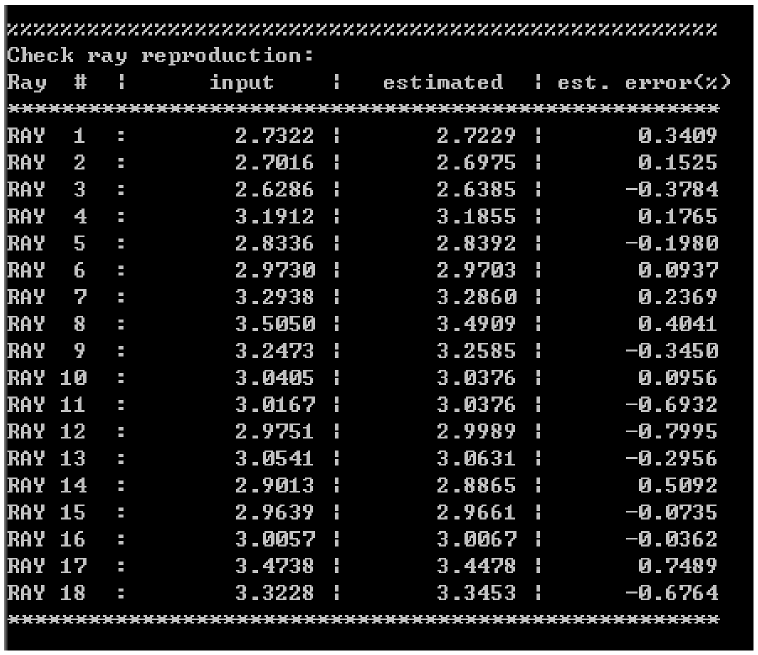


Figure 5: Ray path on the simulation grid


Start search from simulation node location **u**
Is there ray data at **u**
Y
N
Move to the next closest node **u'**
Is **u'** within search ellipse?
N
Y
Is node **u'** informed? and $n_P < N_P$?
N
Y
Put **u'** into point data neighborhood
any ray passing **u'**? and $n_R < N_R$?
N
Y
Put passing ray(s) into ray data neighborhood
$n_P < N_P$ or $n_R < N_R$?
Y
N
Complete searching

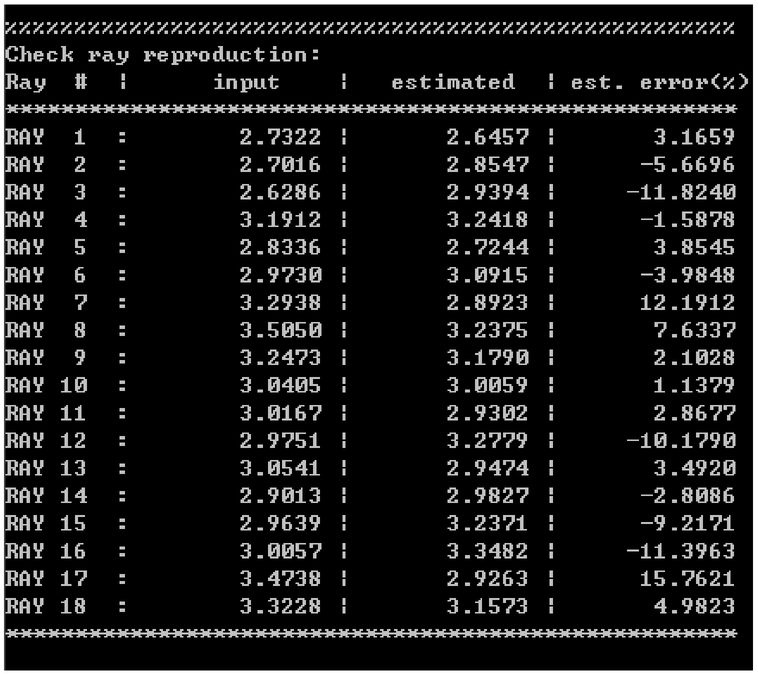
Notations: n_P : number of point neighbors found; n_R : number of ray neighbors found;
N_P : maximum number of point neighbors; N_R : maximum number of ray neighbors.

Figure 6: Point and ray data search flowchart



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Check ray reproduction:
Ray  #  |      input     |   estimated   | est. error(%)
***************************************************
RAY  1  :         2.7322 |        2.7229 |        0.3409
RAY  2  :         2.7016 |        2.6975 |        0.1525
RAY  3  :         2.6286 |        2.6385 |       -0.3784
RAY  4  :         3.1912 |        3.1855 |        0.1765
RAY  5  :         2.8336 |        2.8392 |       -0.1980
RAY  6  :         2.9730 |        2.9703 |        0.0937
RAY  7  :         3.2938 |        3.2860 |        0.2369
RAY  8  :         3.5050 |        3.4909 |        0.4041
RAY  9  :         3.2473 |        3.2585 |       -0.3450
RAY 10  :         3.0405 |        3.0376 |        0.0956
RAY 11  :         3.0167 |        3.0376 |       -0.6932
RAY 12  :         2.9751 |        2.9989 |       -0.7995
RAY 13  :         3.0541 |        3.0631 |       -0.2956
RAY 14  :         2.9013 |        2.8865 |        0.5092
RAY 15  :         2.9639 |        2.9661 |       -0.0735
RAY 16  :         3.0057 |        3.0067 |       -0.0362
RAY 17  :         3.4738 |        3.4478 |        0.7489
RAY 18  :         3.3228 |        3.3453 |       -0.6764
***************************************************
```

(a) Ray data are reproduced if we include all the informed values within the passing ray(s) into the data neighborhood



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Check ray reproduction:
Ray  #  |      input     |   estimated   | est. error(%)
***************************************************
RAY  1  :         2.7322 |        2.6457 |        3.1659
RAY  2  :         2.7016 |        2.8547 |       -5.6696
RAY  3  :         2.6286 |        2.9394 |      -11.8240
RAY  4  :         3.1912 |        3.2418 |       -1.5878
RAY  5  :         2.8336 |        2.7244 |        3.8545
RAY  6  :         2.9730 |        3.0915 |       -3.9848
RAY  7  :         3.2938 |        2.8923 |       12.1912
RAY  8  :         3.5050 |        3.2375 |        7.6337
RAY  9  :         3.2473 |        3.1790 |        2.1028
RAY 10  :         3.0405 |        3.0059 |        1.1379
RAY 11  :         3.0167 |        2.9302 |        2.8677
RAY 12  :         2.9751 |        3.2779 |      -10.1790
RAY 13  :         3.0541 |        2.9474 |        3.4920
RAY 14  :         2.9013 |        2.9827 |       -2.8086
RAY 15  :         2.9639 |        3.2371 |       -9.2171
RAY 16  :         3.0057 |        3.3482 |      -11.3963
RAY 17  :         3.4738 |        2.9263 |       15.7621
RAY 18  :         3.3228 |        3.1573 |        4.9823
***************************************************
```

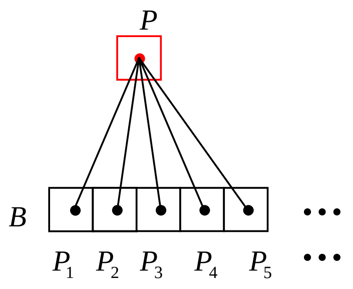
(b) Ray data are not reproduced if we only use a maximum number (say 20) of point data within the search neighborhood

Figure 7: Check for ray data reproduction for the 18-ray case

Point-Block covariance $\bar{C}_{PB}$:

$$\bar{C}_{PB} = \frac{1}{n}\sum_{i=1}^{n} C_{PP_i}$$

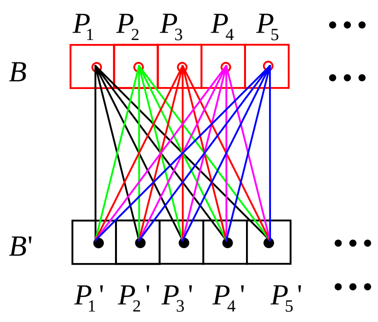
where n is the number of points P_i in block B



Block-Block covariance $\bar{C}_{BB'}$:

$$C_{BB'} = \frac{1}{nn'}\sum_{i=1}^{n}\sum_{j=1}^{n'} C_{P_iP_j'}$$

where n is the number of points P_i in block B, n' is the number of points P_j' in block B',



P_1 P_2 P_3 P_4 P_5 • • •
B
B'
P_1' P_2' P_3' P_4' P_5' • • •

Figure 8: Computation of point-block and block-block covariance

Compute $C_{PP'}$ $\frac{(n_1+1)n_1}{2}$ times — **I** $\downarrow$

Compute $\bar{C}_{PB}$ $n_1 \times n_2$ times — **II** $\downarrow$

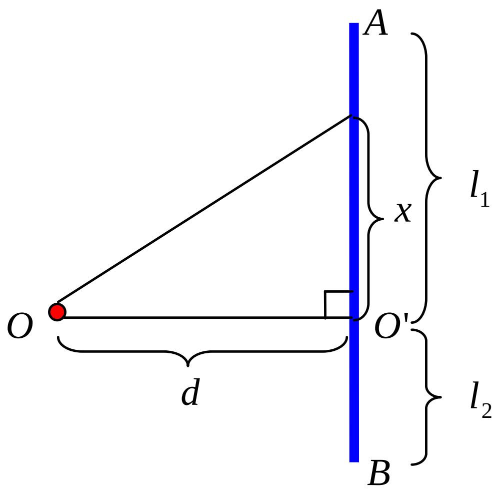
Compute C_{P_0P} n_1 times — **IV** $\downarrow$

$$\begin{bmatrix} \begin{bmatrix} C_{P_1P_1} & \cdots & C_{P_1P_{n_1}} \\ & \ddots & \vdots \\ & & \bar{C}_{P_{n_1}P_{n_1}} \end{bmatrix} & \begin{bmatrix} \bar{C}_{P_1B_1} & \cdots & \bar{C}_{P_1B_{n_2}} \\ \vdots & \ddots & \vdots \\ \bar{C}_{P_{n_1}B_1} & \cdots & \bar{C}_{P_{n_1}B_{n_2}} \end{bmatrix} \\ & \begin{bmatrix} \bar{C}_{B_1B_1} & \cdots & \bar{C}_{B_1B_{n_2}} \\ & \ddots & \vdots \\ & & \bar{C}_{B_{n_2}B_{n_2}} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \lambda_{P_1} \\ \vdots \\ \lambda_{P_{n_1}} \end{bmatrix} \\ \begin{bmatrix} \lambda_{B_1} \\ \vdots \\ \lambda_{B_{n_1}} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} C_{P_0P_1} \\ \vdots \\ C_{P_0P_{n_1}} \end{bmatrix} \\ \begin{bmatrix} \bar{C}_{P_0B_1} \\ \vdots \\ \bar{C}_{P_0B_{n_2}} \end{bmatrix} \end{bmatrix}$$

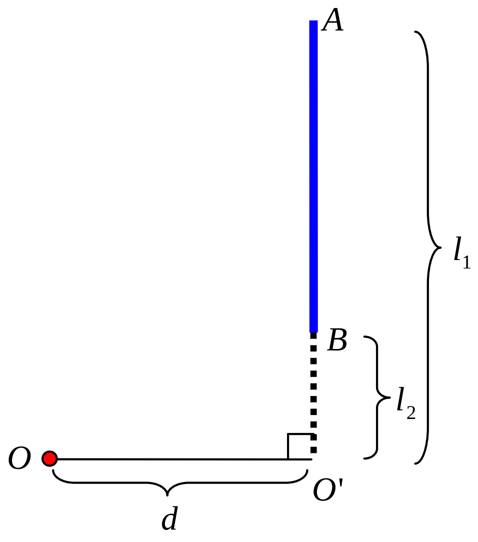
III $\uparrow$ — Compute $\bar{C}_{BB'}$ $\frac{(n_2+1)n_2}{2}$ times

V $\uparrow$ — Compute $\bar{C}_{P_0B}$ n_2 times

Figure 9: Cokriging system



(a) Case $O' \in AB$



(b) Case $O' \notin AB$

Figure 10: Point-to-segment covariance calculation $\bar{C}_{PB}$

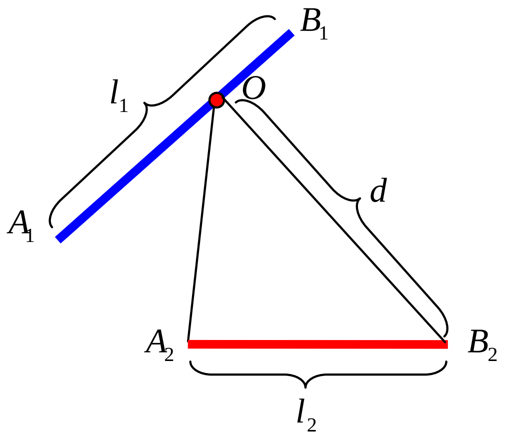


Figure 11: Segment-to-segment covariance calculation $\bar{C}_{BB}$

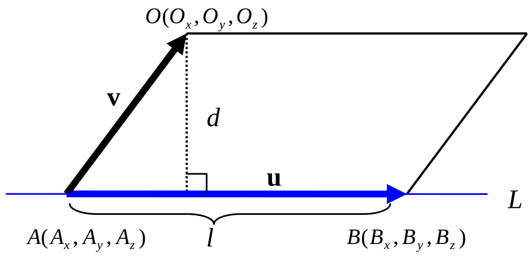


Figure 12: Point to line distance

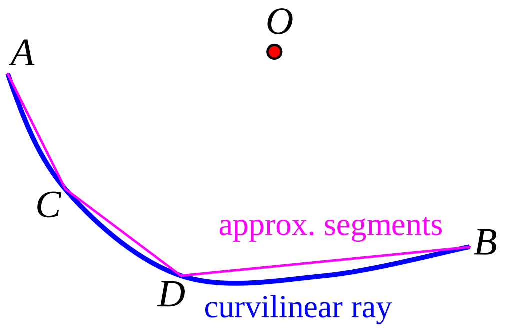


Figure 13: Approximation of a curvilinear ray by ray segments

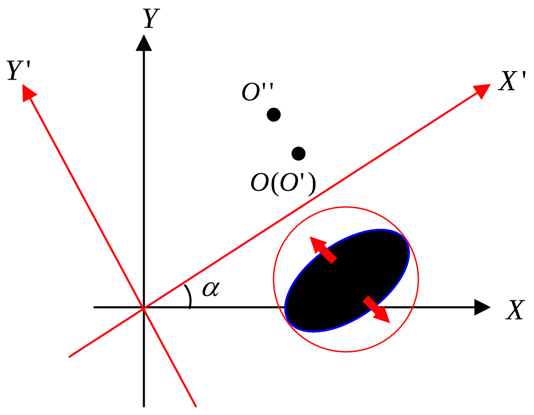
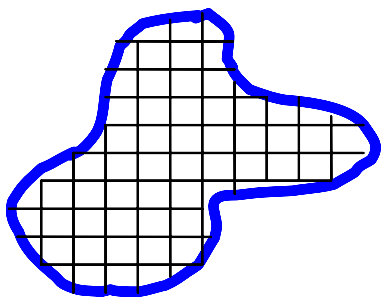
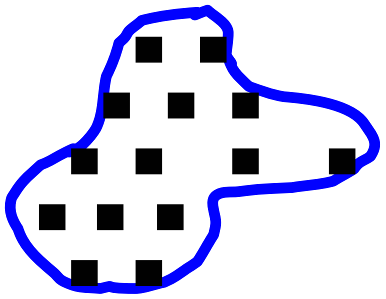


Figure 14: Coordinate transform from anisotropy to isotropy

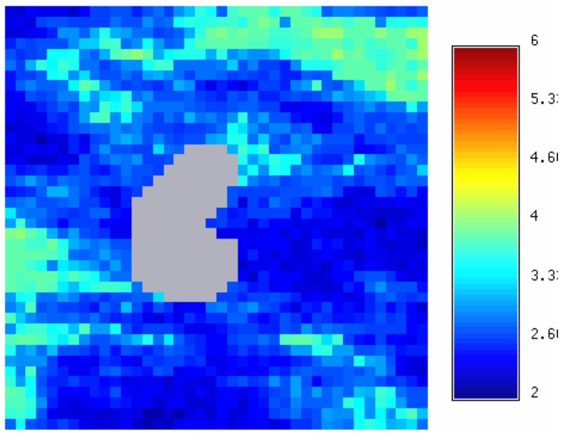


(a) Approximation by line

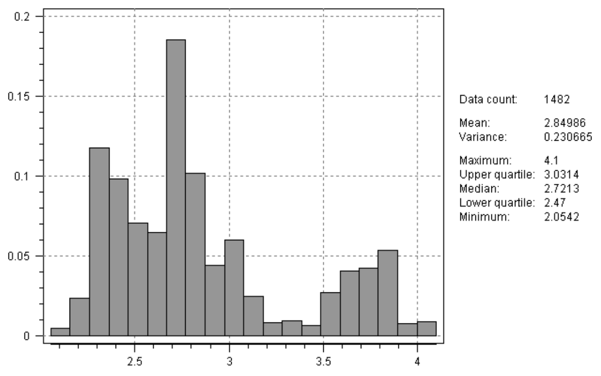


(b) Approximation by point

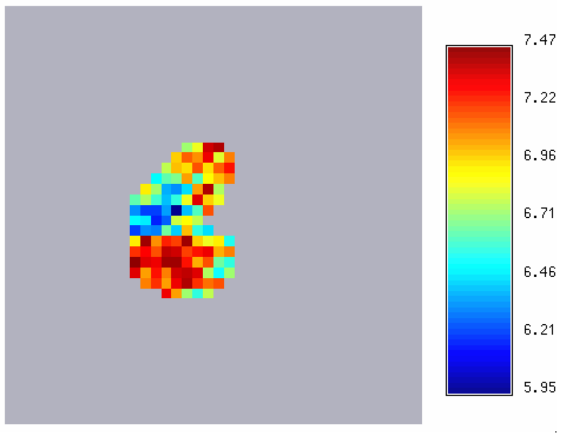
Figure 15: Approximation of a block with arbitrary shape



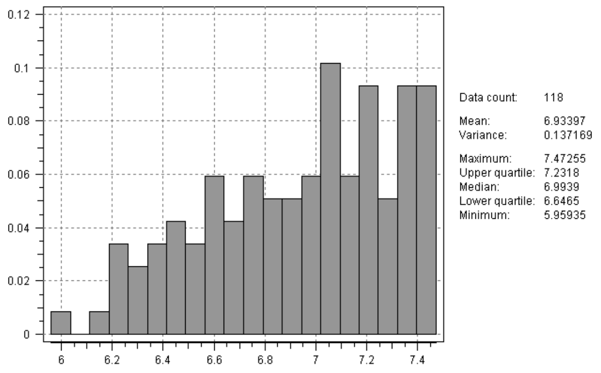
(a) Background velocity



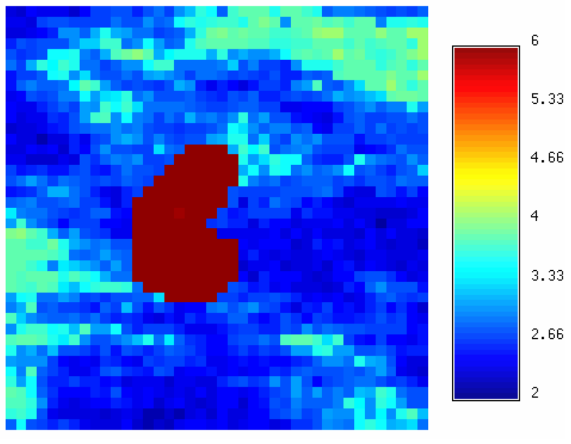
(b) Histogram of background velocity



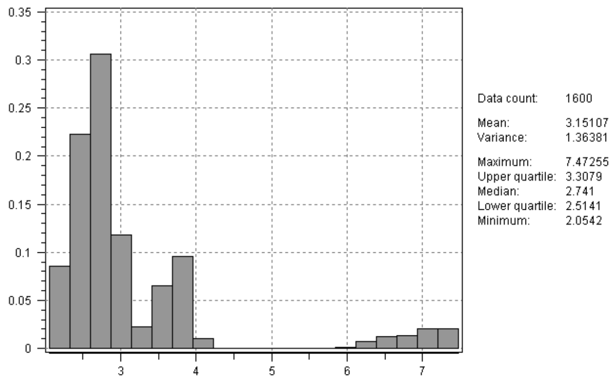
(c) Increased peanut velocity



(d) Histogram of peanut velocity

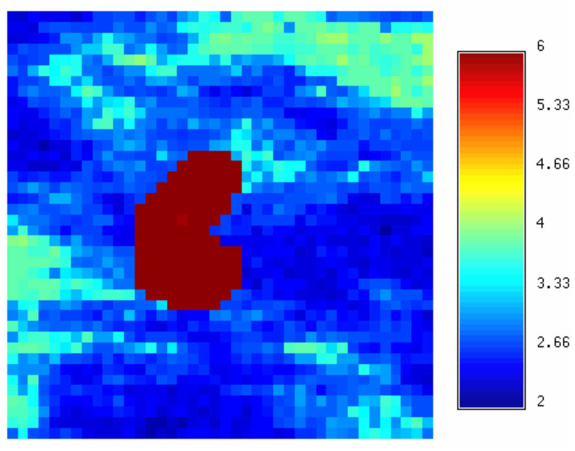


(e) Combined velocity model

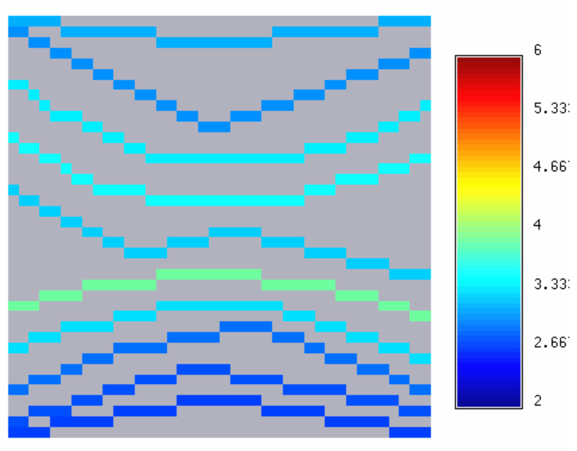


(f) Histogram of combined velocity model

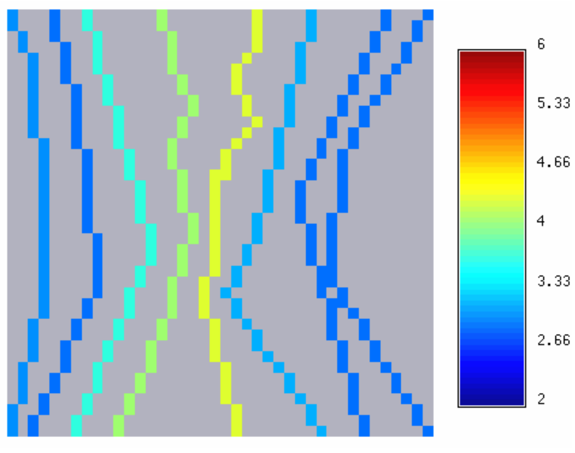
Figure 16: Model resulting from increasing the peanut values by 1.5 times



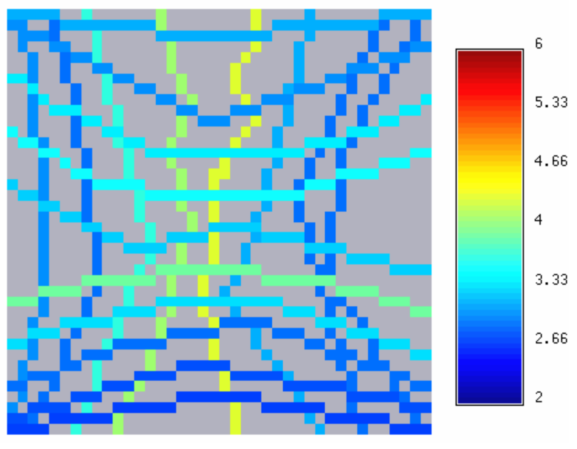
(a) Reference velocity model



(b) Extracted lateral ray average data

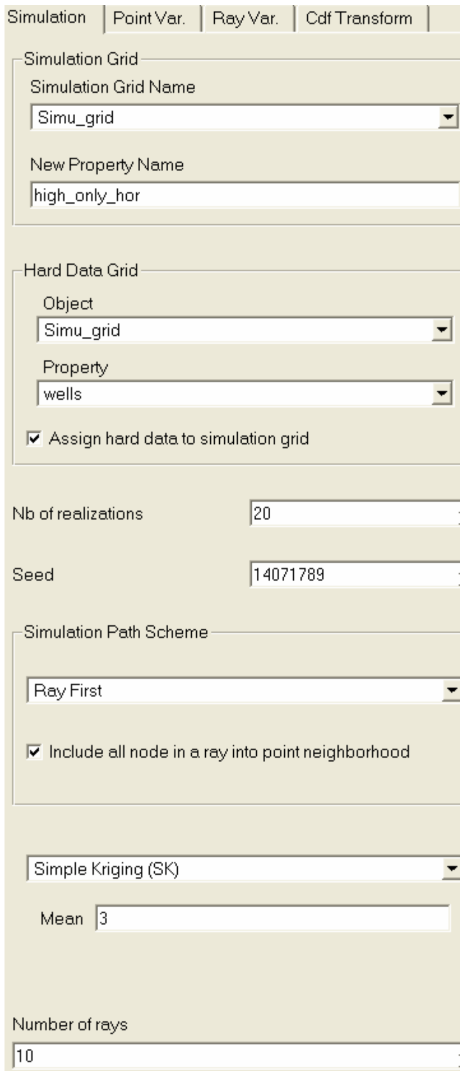


(c) Extracted vertical ray average data



(d) Both lateral and vertical ray average data

Figure 17: 3 ray data configurations (The color indicates the values of velocity)



Simulation Grid

Simulation Grid Name: Simu_grid

New Property Name: high_only_hor

Hard Data Grid

Object: Simu_grid

Property: wells

☑ Assign hard data to simulation grid

Nb of realizations: 20

Seed: 14071789

Simulation Path Scheme

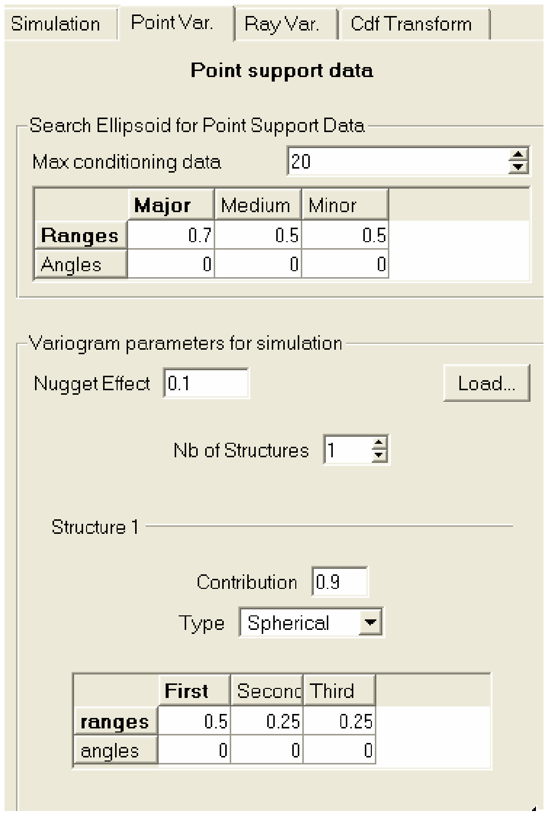
Ray First

☑ Include all node in a ray into point neighborhood

Simple Kriging (SK)

Mean: 3

Number of rays: 10



Point support data

Search Ellipsoid for Point Support Data

Max conditioning data: 20

	Major	Medium	Minor
Ranges	0.7	0.5	0.5
Angles	0	0	0

Variogram parameters for simulation

Nugget Effect: 0.1

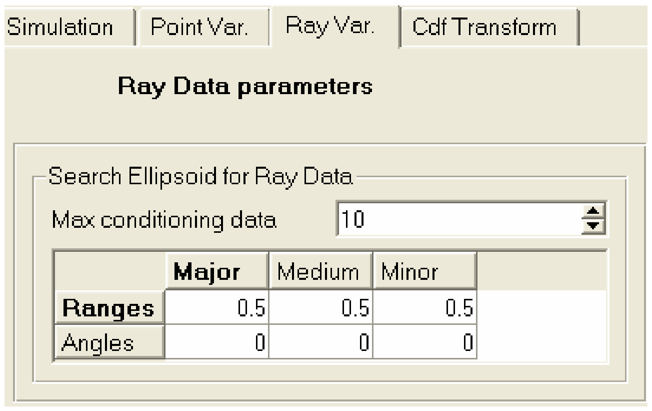
Nb of Structures: 1

Structure 1

Contribution: 0.9

Type: Spherical

	First	Second	Third
ranges	0.5	0.25	0.25
angles	0	0	0



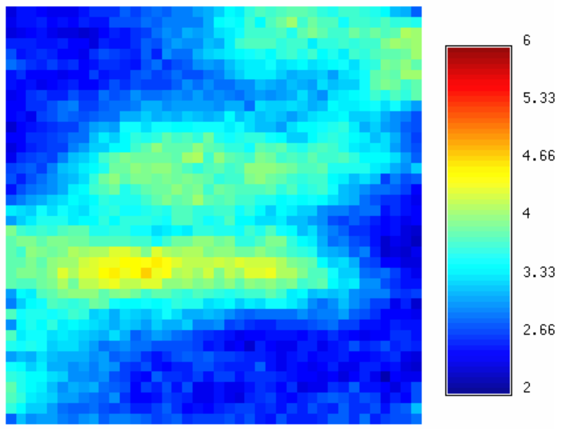
Ray Data parameters

Search Ellipsoid for Ray Data

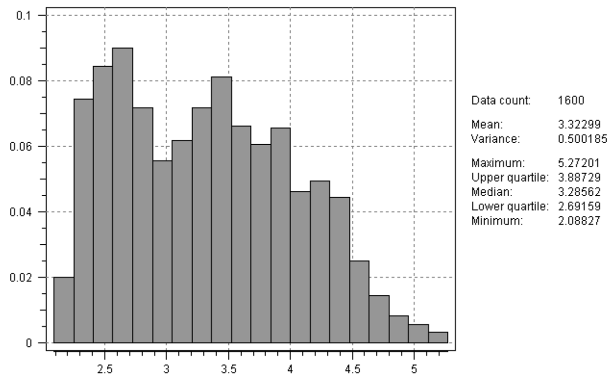
Max conditioning data: 10

	Major	Medium	Minor
Ranges	0.5	0.5	0.5
Angles	0	0	0

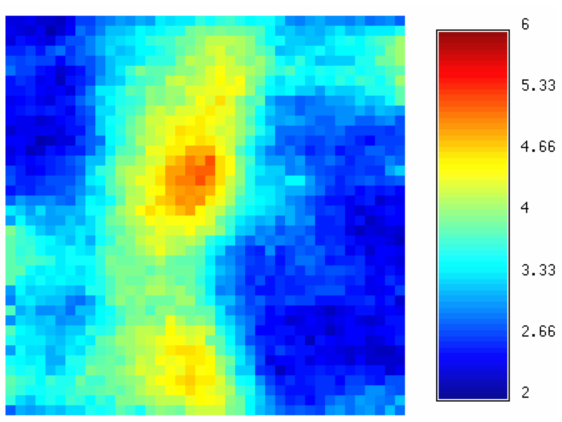
Figure 18: Parameter settings for lateral ray case



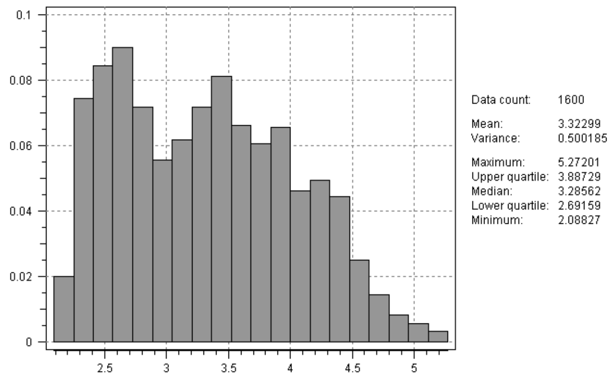
(a) E-type of the lateral ray case



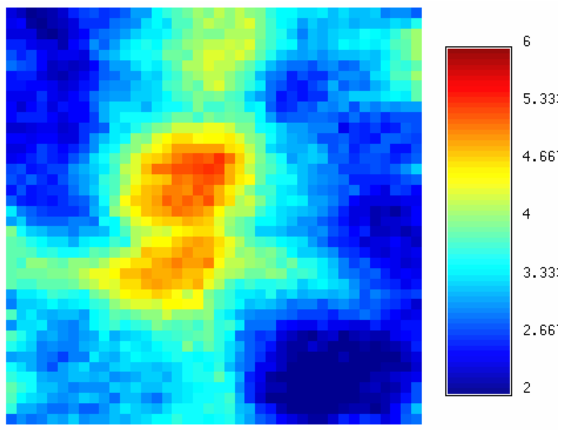
(b) Hist. of E-type of the lateral ray case



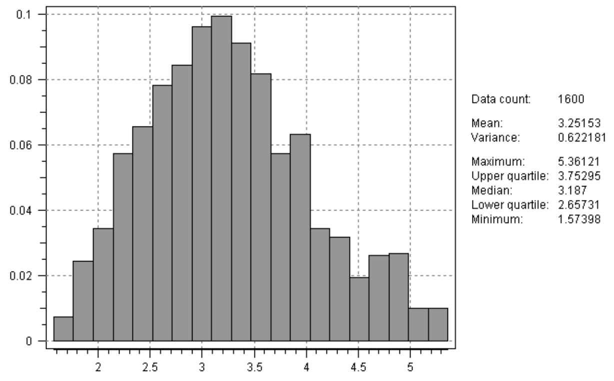
(c) E-type of the vertical ray case



(d) Hist. of E-type of the lateral ray case

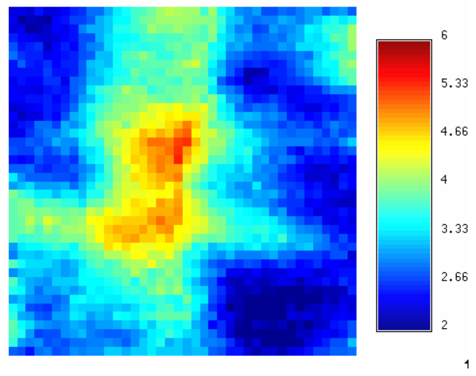


(e) E-type of the all ray case

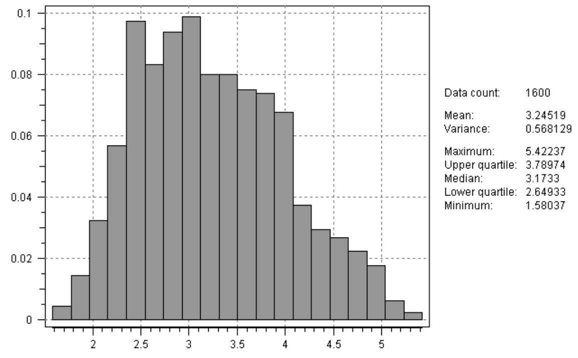


(f) Hist. of E-type of the lateral ray case

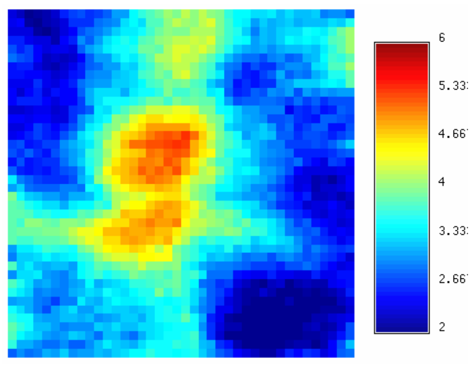
Figure 19: Results of the 3 curvilinear ray configurations averaged from 20 simulated realizations



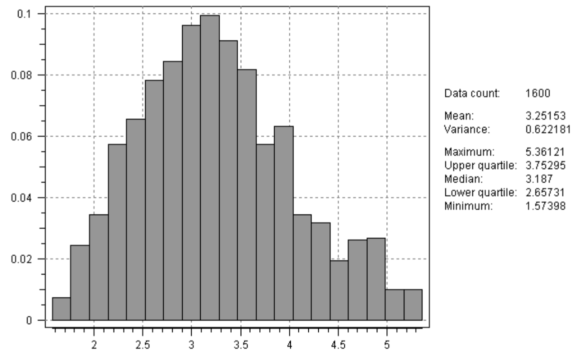
(a) E-type using all rays with random path



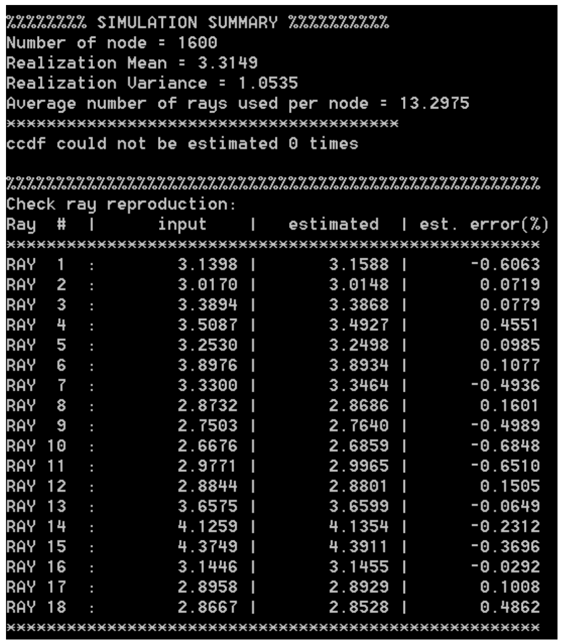
(b) Hist. of E-type values



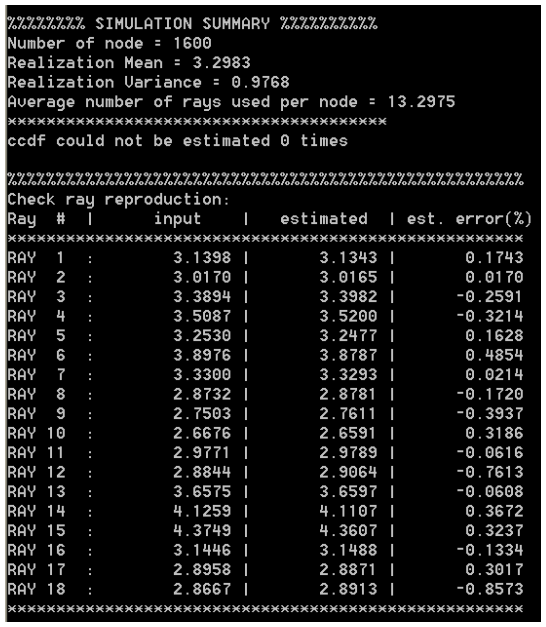
(c) E-type with ray-first



(d) Hist. of E-type values

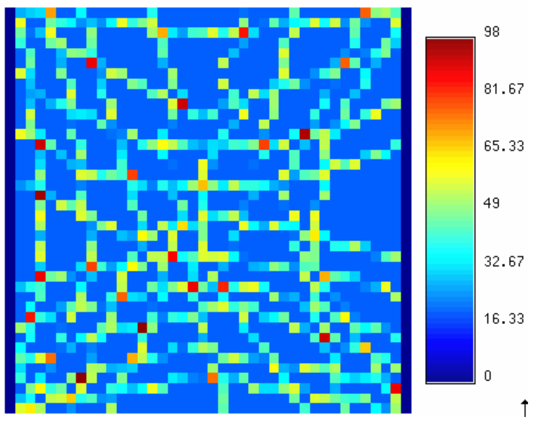

```
%%%%%%%% SIMULATION SUMMARY %%%%%%%%%%
Number of node = 1600
Realization Mean = 3.3149
Realization Variance = 1.0535
Average number of rays used per node = 13.2975
****************************************
ccdf could not be estimated 0 times

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Check ray reproduction:
Ray  #  |      input     |   estimated   | est. error(%)
*****************************************************
RAY  1  :        3.1398 |        3.1588 |       -0.6063
RAY  2  :        3.0170 |        3.0148 |        0.0719
RAY  3  :        3.3894 |        3.3868 |        0.0779
RAY  4  :        3.5087 |        3.4927 |        0.4551
RAY  5  :        3.2530 |        3.2498 |        0.0985
RAY  6  :        3.8976 |        3.8934 |        0.1077
RAY  7  :        3.3300 |        3.3464 |       -0.4936
RAY  8  :        2.8732 |        2.8686 |        0.1601
RAY  9  :        2.7503 |        2.7640 |       -0.4989
RAY 10  :        2.6676 |        2.6859 |       -0.6848
RAY 11  :        2.9771 |        2.9965 |       -0.6510
RAY 12  :        2.8844 |        2.8801 |        0.1505
RAY 13  :        3.6575 |        3.6599 |       -0.0649
RAY 14  :        4.1259 |        4.1354 |       -0.2312
RAY 15  :        4.3749 |        4.3911 |       -0.3696
RAY 16  :        3.1446 |        3.1455 |       -0.0292
RAY 17  :        2.8958 |        2.8929 |        0.1008
RAY 18  :        2.8667 |        2.8528 |        0.4862
*****************************************************
```
(e) Ray reproduction in random sequence case

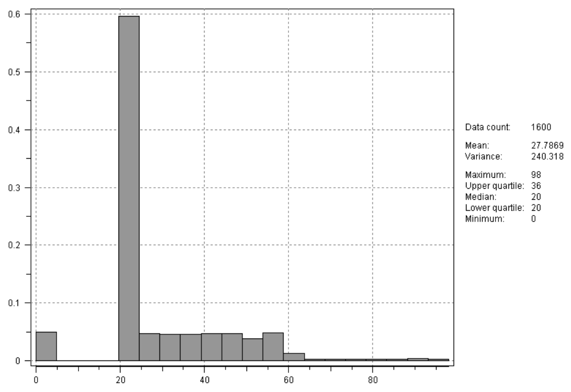

```
%%%%%%%% SIMULATION SUMMARY %%%%%%%%%%
Number of node = 1600
Realization Mean = 3.2983
Realization Variance = 0.9768
Average number of rays used per node = 13.2975
****************************************
ccdf could not be estimated 0 times

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Check ray reproduction:
Ray  #  |      input     |   estimated   | est. error(%)
*****************************************************
RAY  1  :        3.1398 |        3.1343 |        0.1743
RAY  2  :        3.0170 |        3.0165 |        0.0170
RAY  3  :        3.3894 |        3.3982 |       -0.2591
RAY  4  :        3.5087 |        3.5200 |       -0.3214
RAY  5  :        3.2530 |        3.2477 |        0.1628
RAY  6  :        3.8976 |        3.8787 |        0.4854
RAY  7  :        3.3300 |        3.3293 |        0.0214
RAY  8  :        2.8732 |        2.8781 |       -0.1720
RAY  9  :        2.7503 |        2.7611 |       -0.3937
RAY 10  :        2.6676 |        2.6591 |        0.3186
RAY 11  :        2.9771 |        2.9789 |       -0.0616
RAY 12  :        2.8844 |        2.9064 |       -0.7613
RAY 13  :        3.6575 |        3.6597 |       -0.0608
RAY 14  :        4.1259 |        4.1107 |        0.3672
RAY 15  :        4.3749 |        4.3607 |        0.3237
RAY 16  :        3.1446 |        3.1488 |       -0.1334
RAY 17  :        2.8958 |        2.8871 |        0.3017
RAY 18  :        2.8667 |        2.8913 |       -0.8573
*****************************************************
```
(f) Ray reproduction in ray-first sequence case

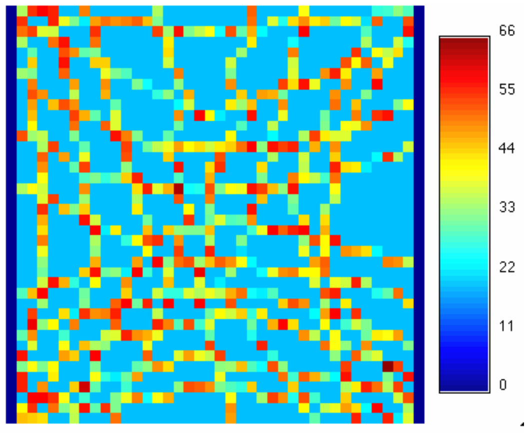
Figure 20: Comparison of the E-type from random path and ray-first path



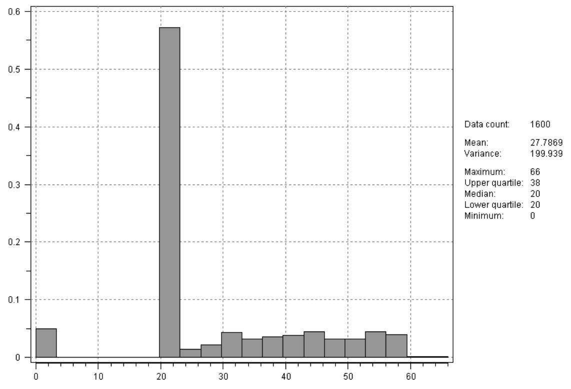
(a) No. of point data per node(random path)



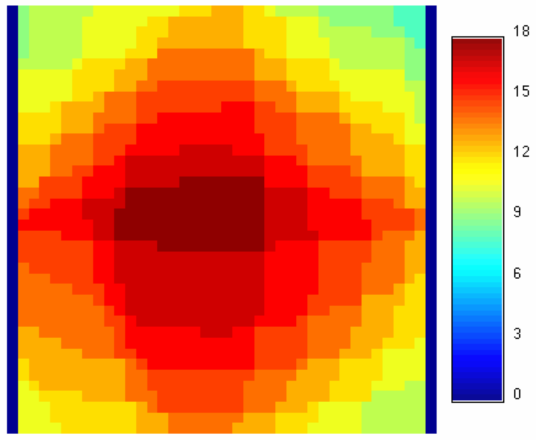
(b) Histogram of left figure



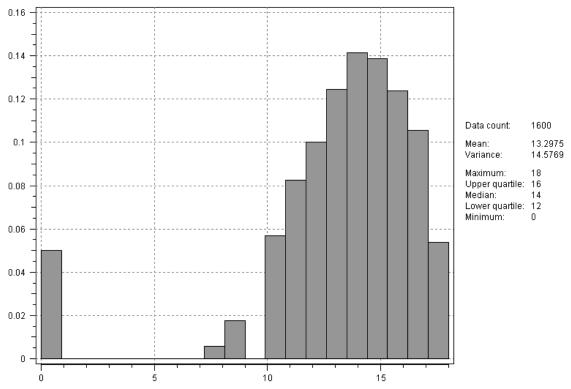
(c) No. of point data per node(ray-first path)



(d) Histogram of left figure

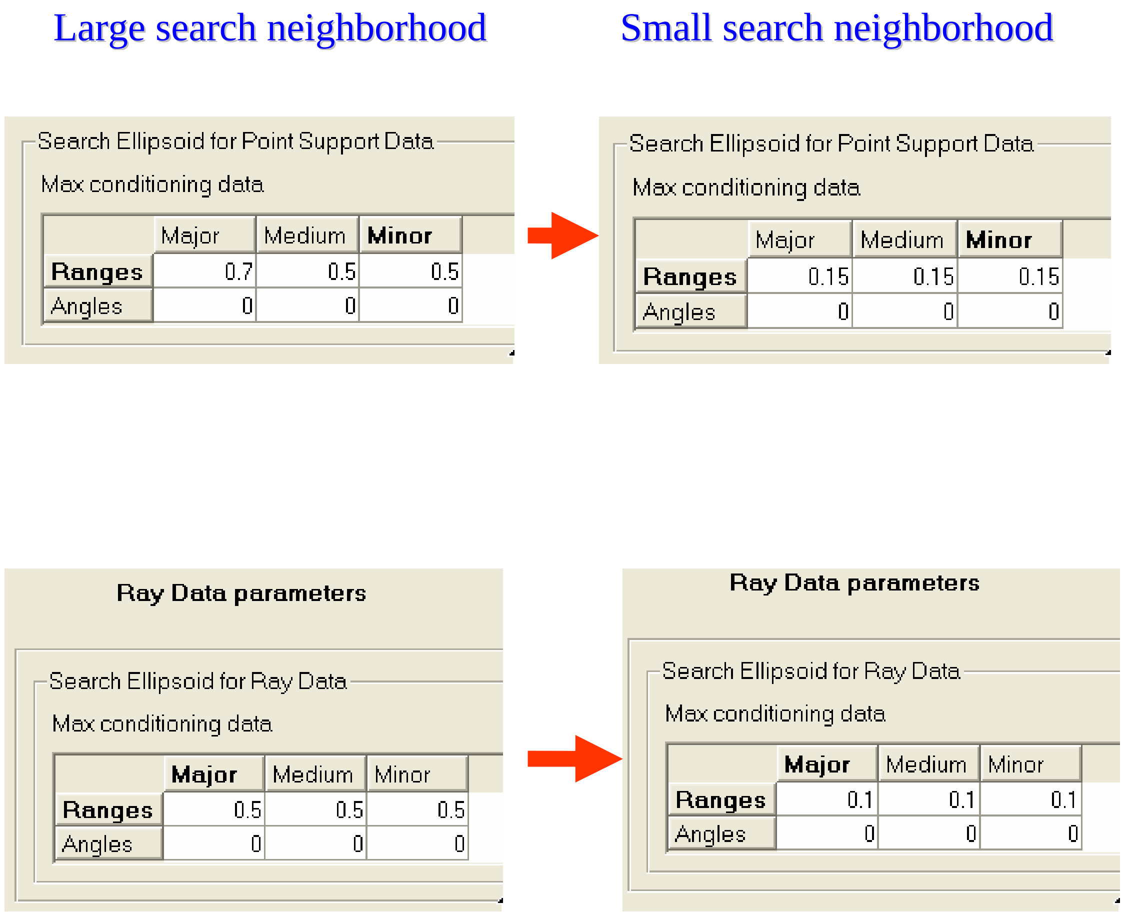


(e) No. of ray data per node



(f) Histogram of left figure

Figure 21: Point and ray data found per node (all rays case). The color indicates the number of point or ray data found within each node neighborhood.



Search Ellipsoid for Point Support Data

Max conditioning data

	Major	Medium	**Minor**
Ranges	0.7	0.5	0.5
Angles	0	0	0

Search Ellipsoid for Point Support Data

Max conditioning data

	Major	Medium	**Minor**
Ranges	0.15	0.15	0.15
Angles	0	0	0

Ray Data parameters

Search Ellipsoid for Ray Data

Max conditioning data

	Major	Medium	Minor
Ranges	0.5	0.5	0.5
Angles	0	0	0

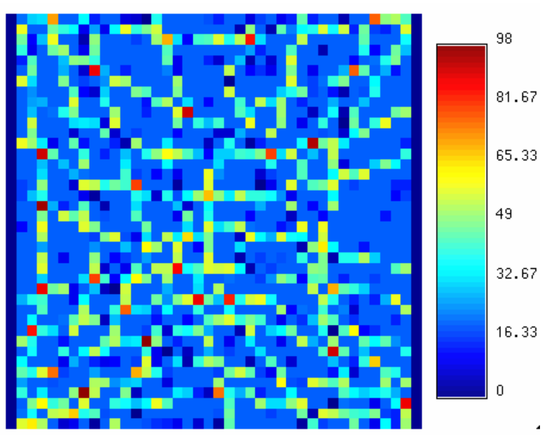
Ray Data parameters

Search Ellipsoid for Ray Data

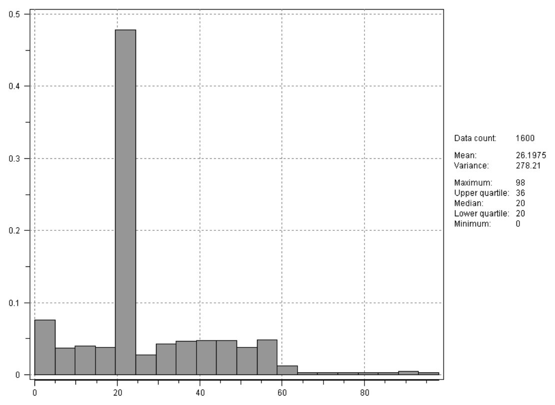
Max conditioning data

	Major	Medium	Minor
Ranges	0.1	0.1	0.1
Angles	0	0	0

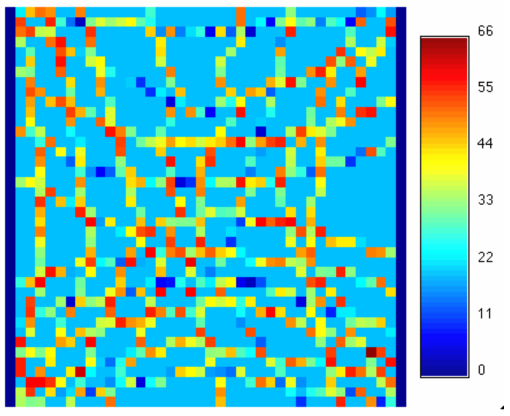
Figure 22: Parameter change from large search neighborhood to small search neighborhood



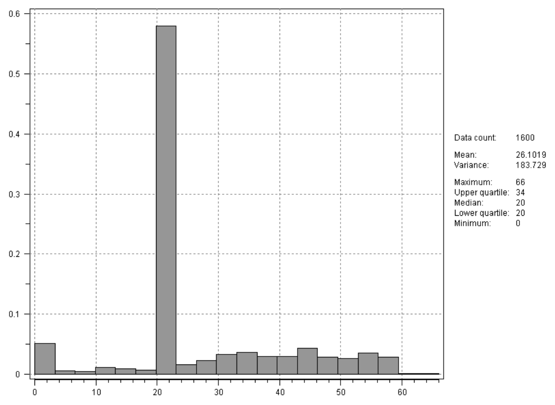
(a) No. of point data per node(random path)



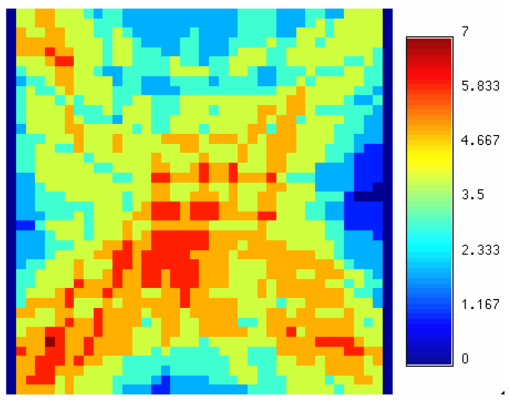
(b) Histogram of left figure



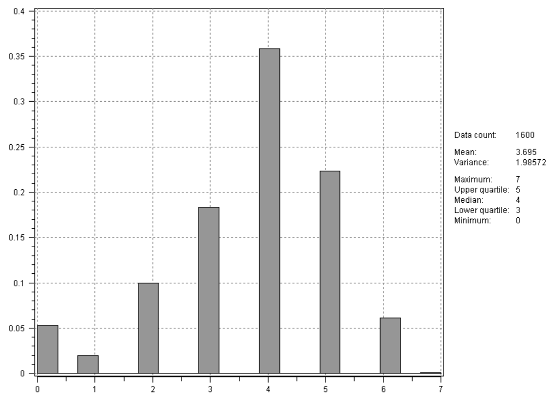
(c) No. of point data per node(ray-first path)



(d) Histogram of left figure

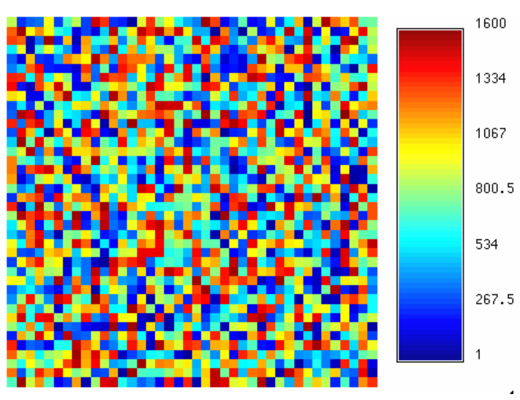


(e) No. of ray data per node

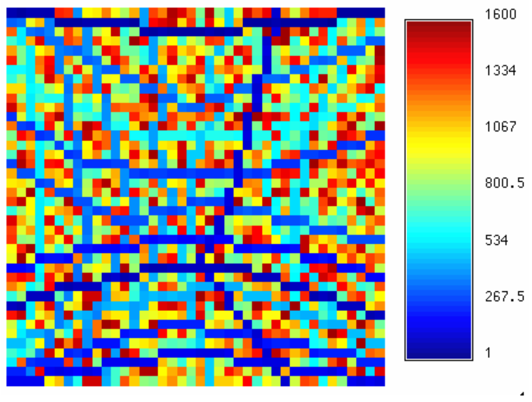


(f) Histogram of left figure

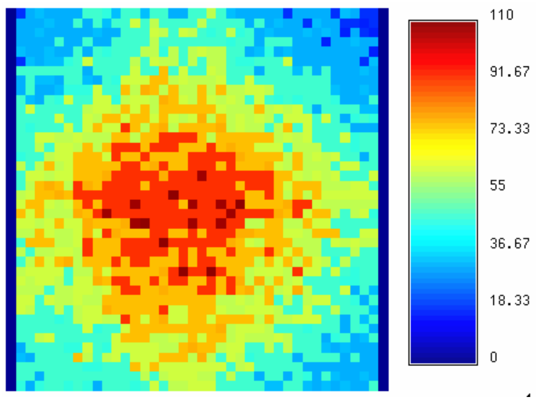
Figure 23: Point and ray data neighborhood per node with smaller search neighborhood(both lateral and vertical rays case). The color indicates the number of point or ray data found within each node neighborhood.



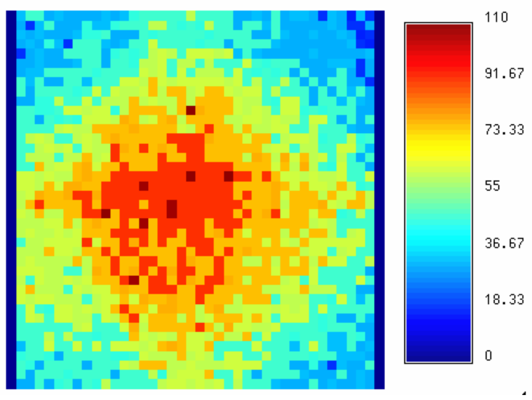
(a) Simul. path (random path)



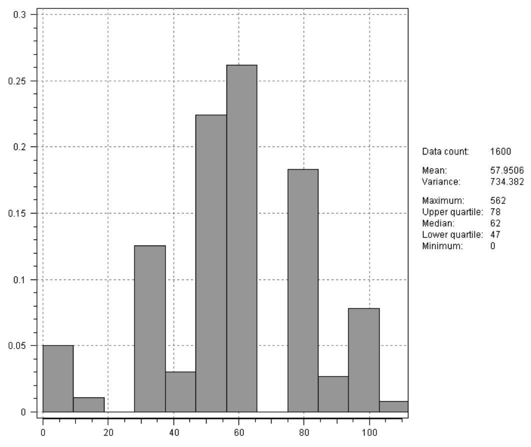
(b) Simul. path (ray-first, path)



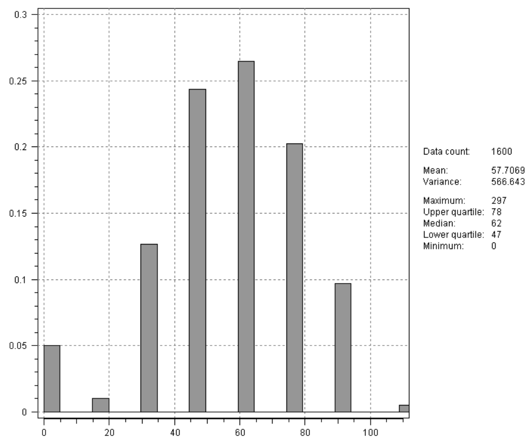
(c) Simul. time per node(random path)



(d) Simul. time per node(ray-first path)



(e) Hist. of Simul. time per node(random path)



(f) Hist. of Simul. time per node(ray-first path)

Figure 24: The simulation paths and simulation time per node (random or ray-first path, conditioning to all rays). The color indicates the order of simulation in the top two figures. The color indicates the simulation time in millisecond (ms) in the middle two figures.

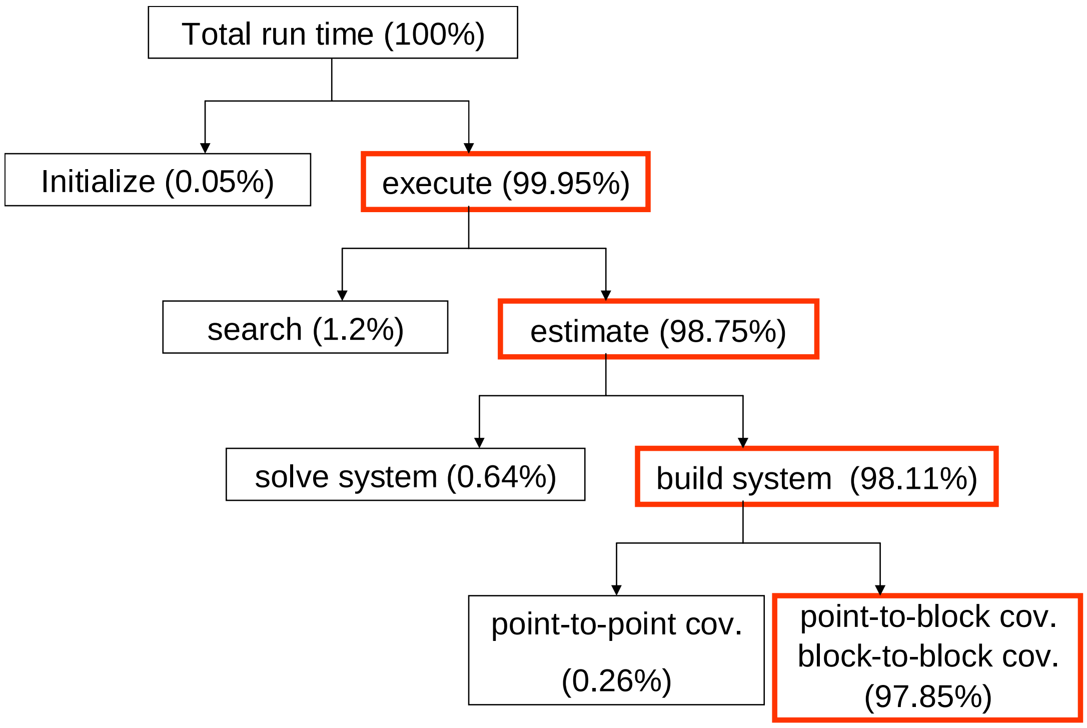
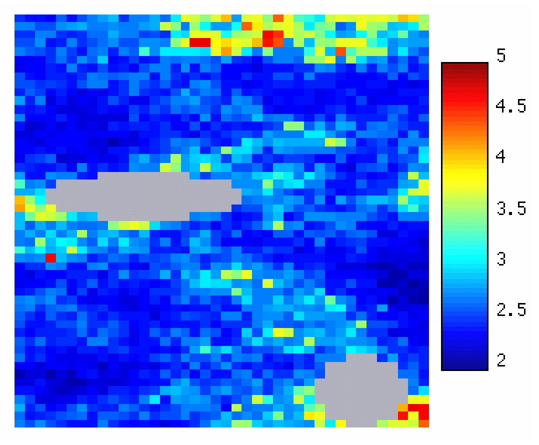
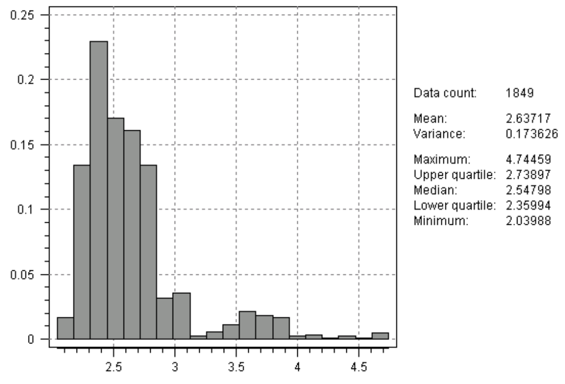


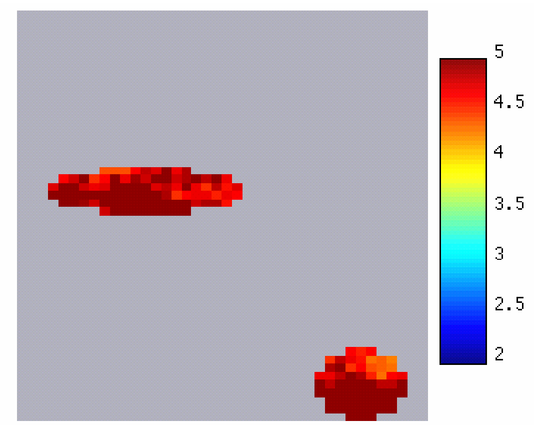
Figure 25: Flowchart for tracking simulation time (the highlighted parts are the dominant ones)



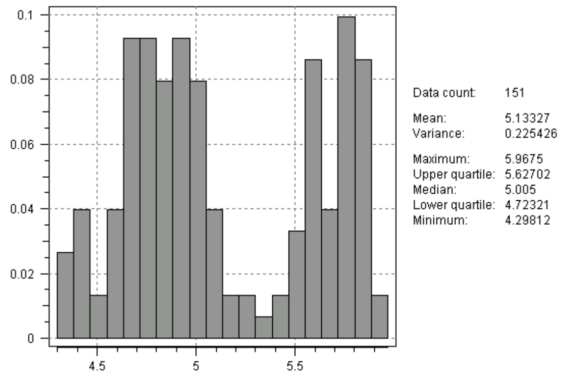
(a) Background velocity



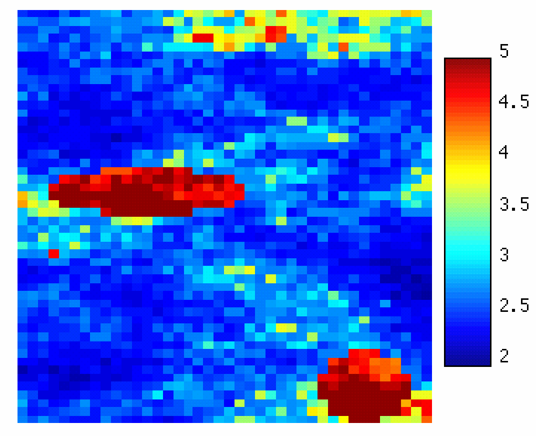
(b) Histogram of background velocity



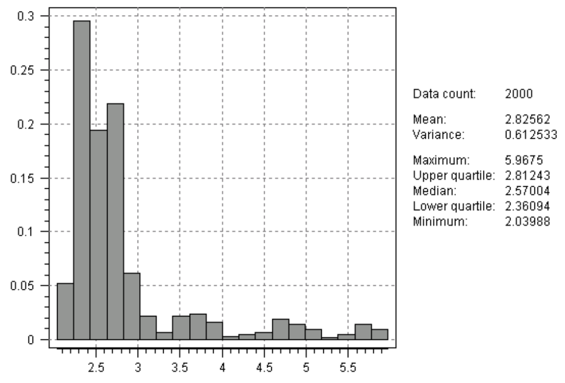
(c) Two high velocity heterogeneities



(d) Histogram of high value velocity

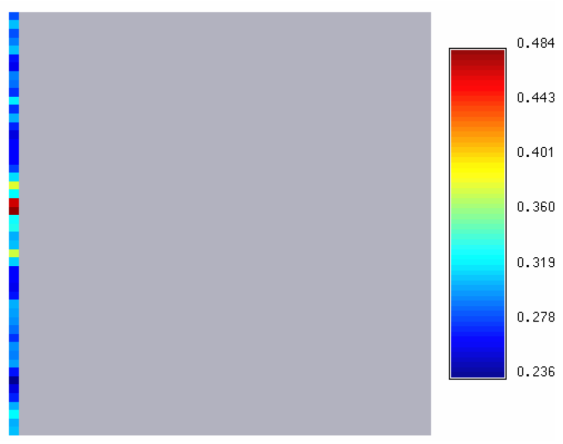


(e) Combined velocity model

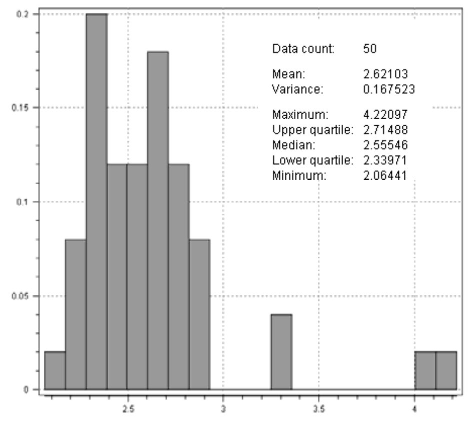


(f) Histogram of combined velocity model

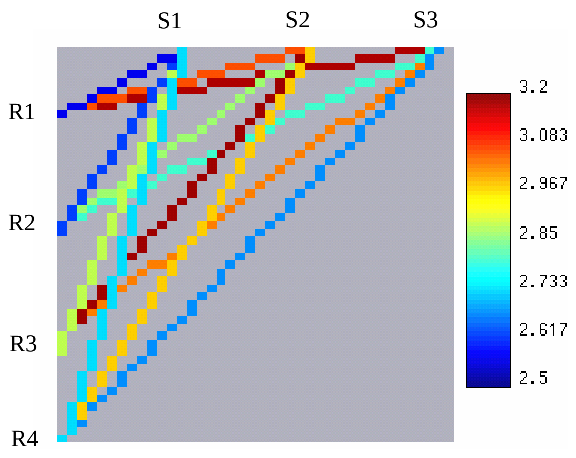
Figure 26: Reference field for vertical well VSP case study



(a) Well location



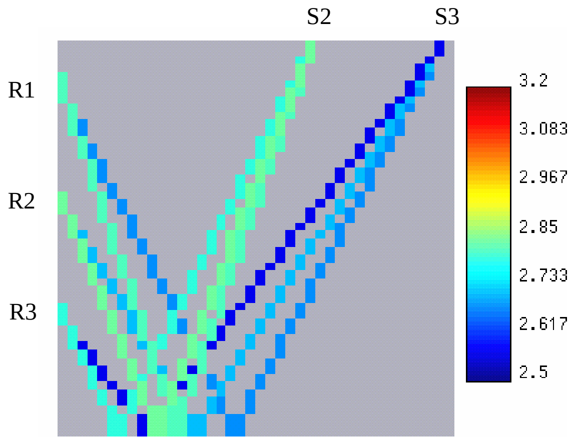
(b) Histogram of the well



(c) The first arrival ray path

Ray #	Ray value
S1-R1	2.5586
S1-R2	2.6234
S1-R3	2.8864
S1-R4	2.7317
S2-R1	3.0720
S2-R2	2.8700
S2-R3	3.1925
S2-R4	2.9829
S3-R1	3.1792
S3-R2	2.8008
S3-R3	3.0391
S3-R4	2.6802

(d) Average velocities along first arrivals

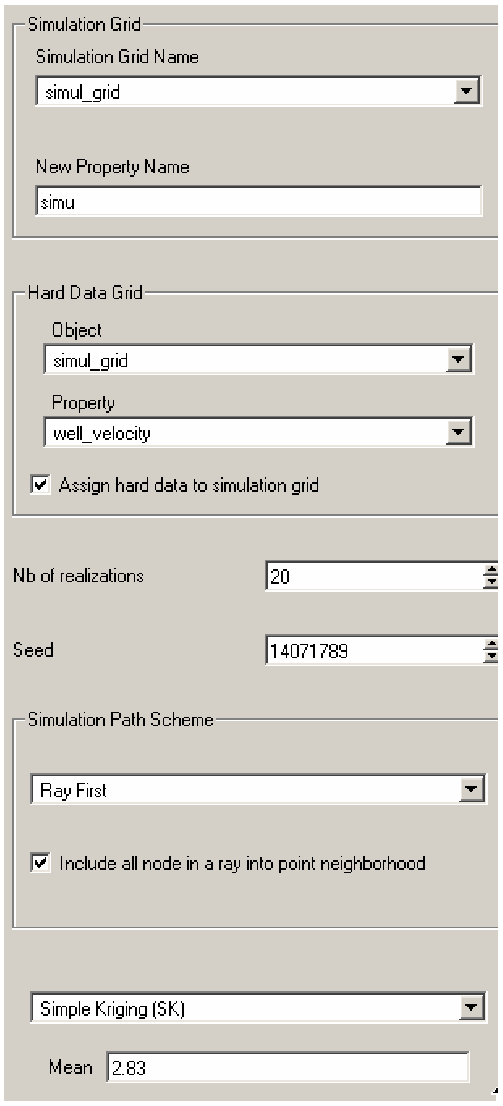


(e) Reflected ray path

Ray #	Ray value
S2-R1	2.8055
S2-R2	2.8312
S2-R3	2.7869
S3-R1	2.6796
S3-R2	2.7137
S3-R3	2.5677

(f) Average velocities along reflected rays

Figure 27: The configuration of well and ray paths in vertical well VSP



Simulation Grid

Simulation Grid Name: simul_grid

New Property Name: simu

Hard Data Grid

Object: simul_grid

Property: well_velocity

[✓] Assign hard data to simulation grid

Nb of realizations: 20

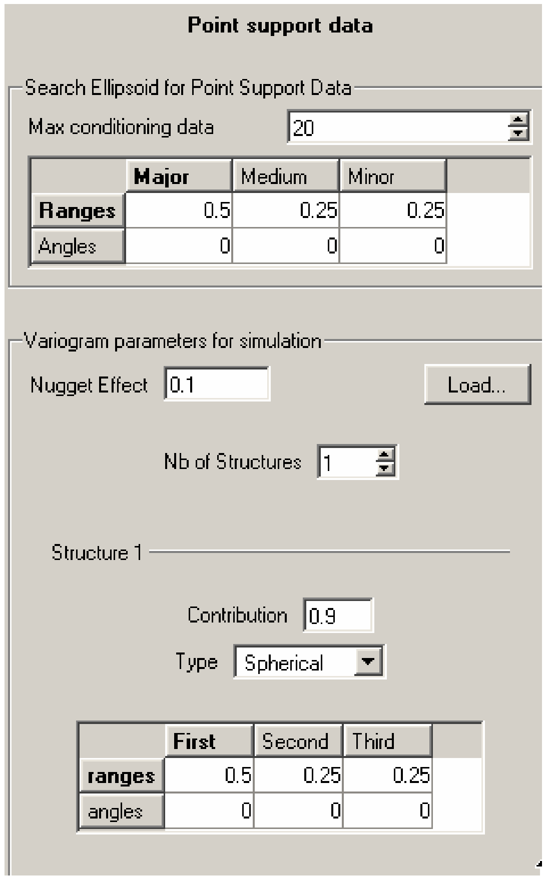
Seed: 14071789

Simulation Path Scheme: Ray First

[✓] Include all node in a ray into point neighborhood

Simple Kriging (SK)

Mean: 2.83



Point support data

Search Ellipsoid for Point Support Data

Max conditioning data: 20

	Major	Medium	Minor
Ranges	0.5	0.25	0.25
Angles	0	0	0

Variogram parameters for simulation

Nugget Effect: 0.1

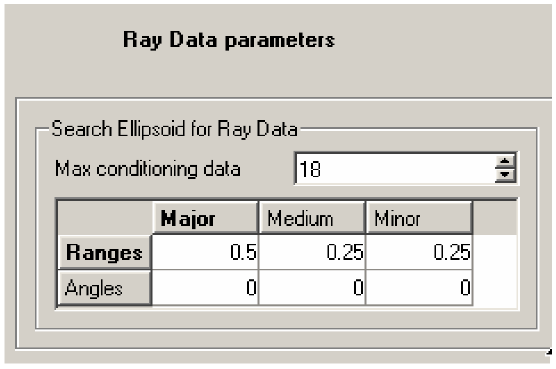
Nb of Structures: 1

Structure 1

Contribution: 0.9

Type: Spherical

	First	Second	Third
ranges	0.5	0.25	0.25
angles	0	0	0



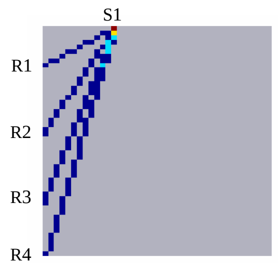
Ray Data parameters

Search Ellipsoid for Ray Data

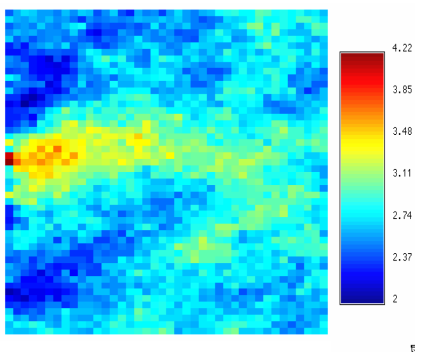
Max conditioning data: 18

	Major	Medium	Minor
Ranges	0.5	0.25	0.25
Angles	0	0	0

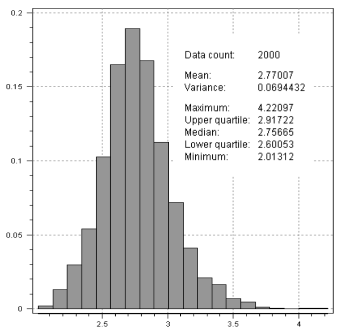
Figure 28: Parameter settings for the VSP case



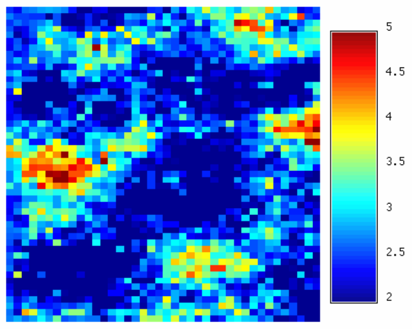
(a) Ray configuration



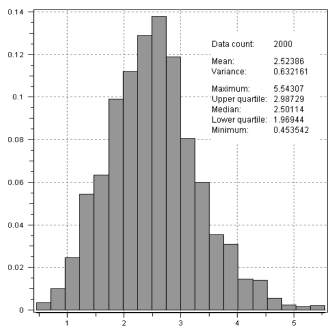
(b) E-type



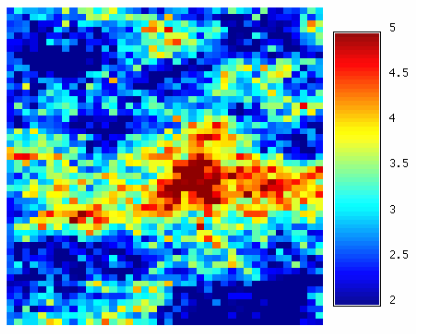
(c) Histogram of E-type



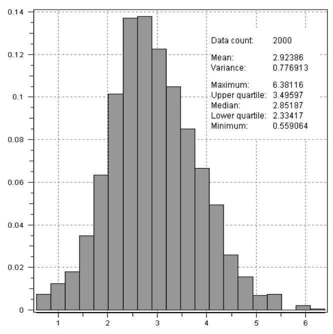
(d) Realization 1



(e) Histogram of real. 1

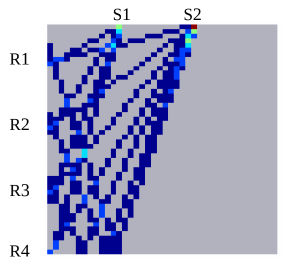


(f) Realization 2

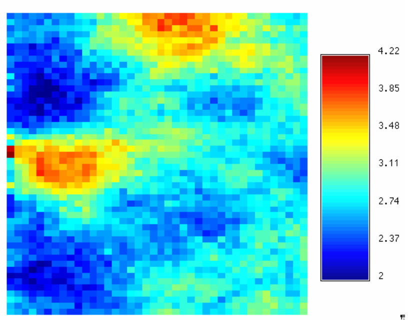


(g) Histogram of real. 2

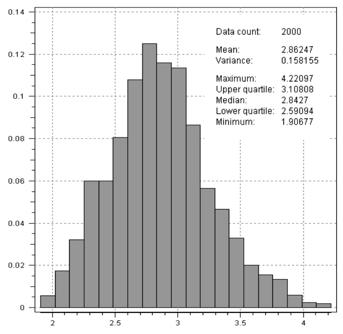
Figure 29: Results conditioned to the first shot ray data



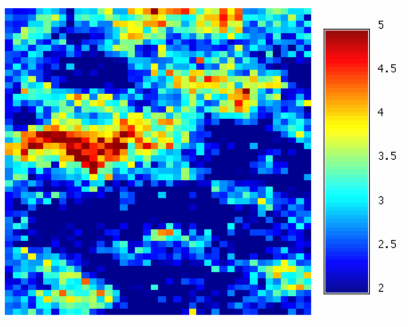
(a) Ray configuration



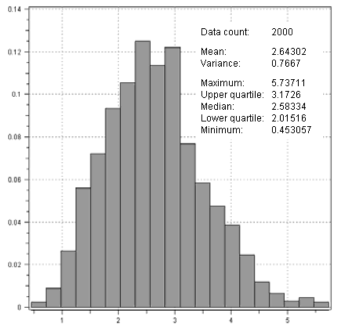
(b) E-type



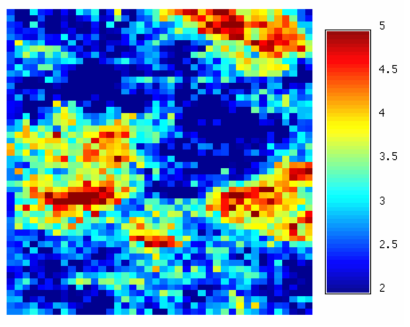
(c) Histogram of E-type



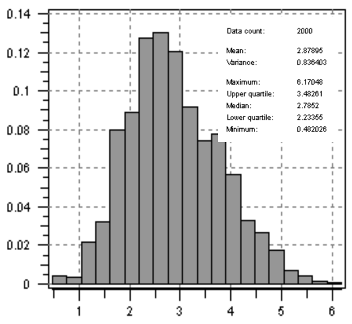
(d) Realization 1



(e) Histogram of real. 1

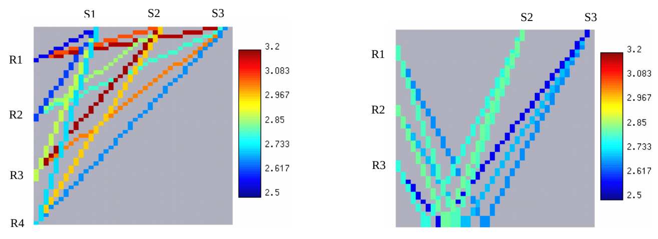


(f) Realization 2

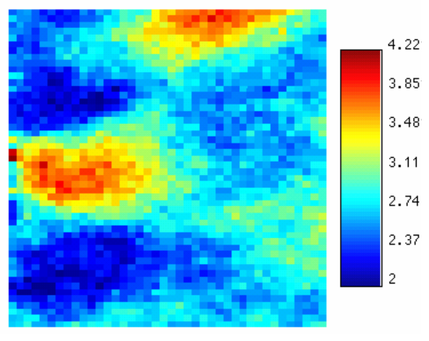


(g) Histogram of real. 2

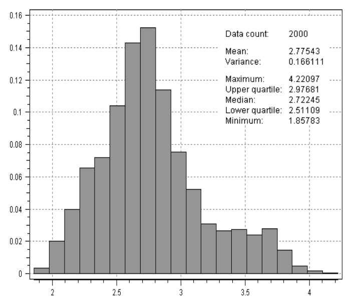
Figure 30: Results conditioned to the first two shots ray data



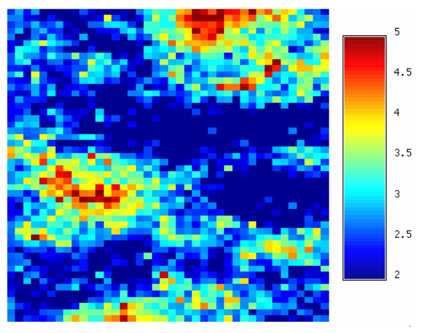
(a) Ray configuration



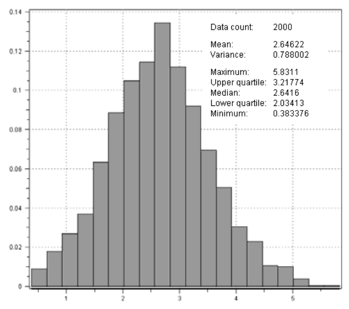
(b) E-type



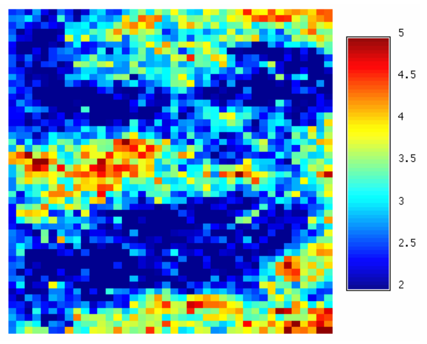
(c) Histogram of E-type



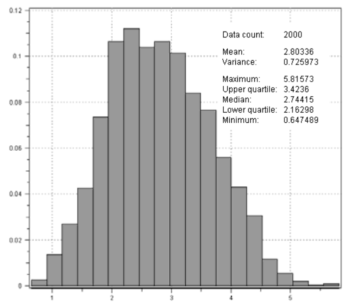
(d) Realization 1



(e) Histogram of real. 1

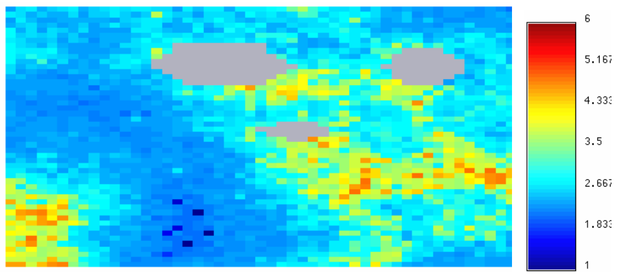


(f) Realization 2

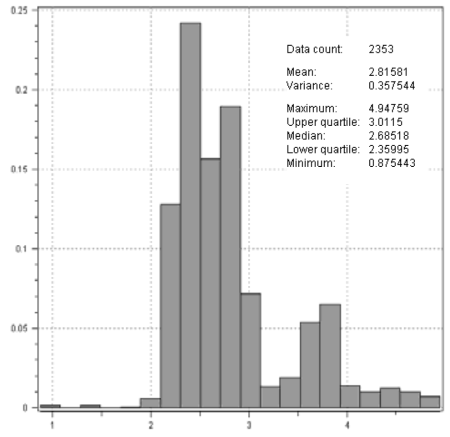


(g) Histogram of real. 2

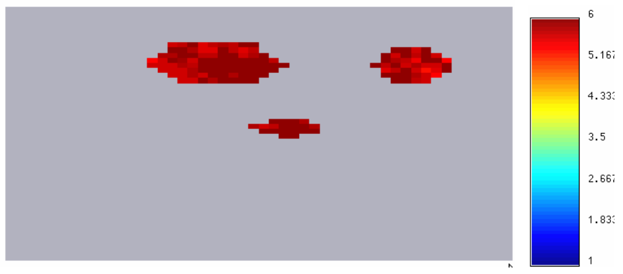
Figure 31: Results conditioned to all 3 shots ray data



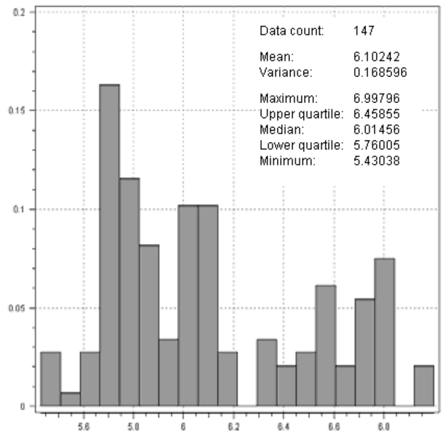
(a) Background velocity



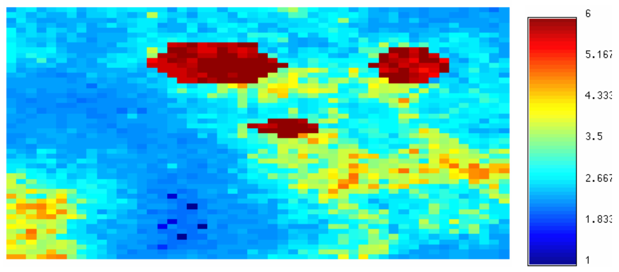
(b) Histogram of background velocity



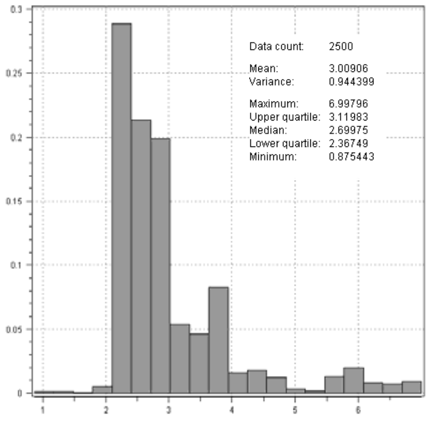
(c) Three high velocity heterogeneities



(d) Histogram of high value velocity

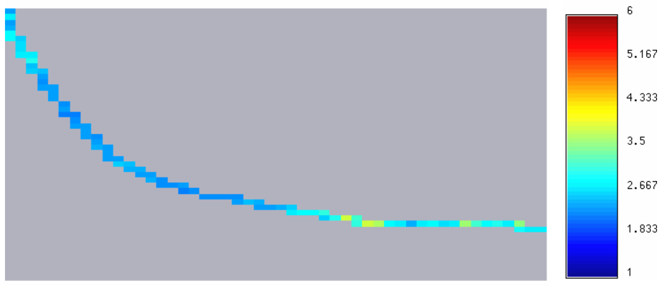


(e) Combined velocity model

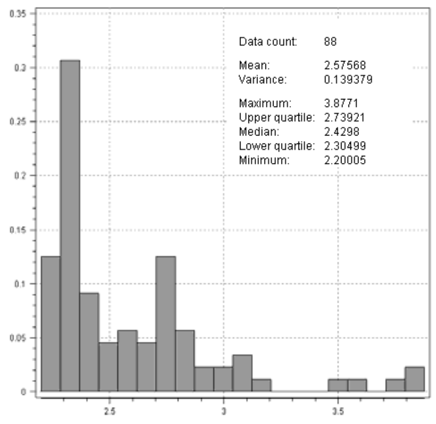


(f) Histogram of combined velocity model

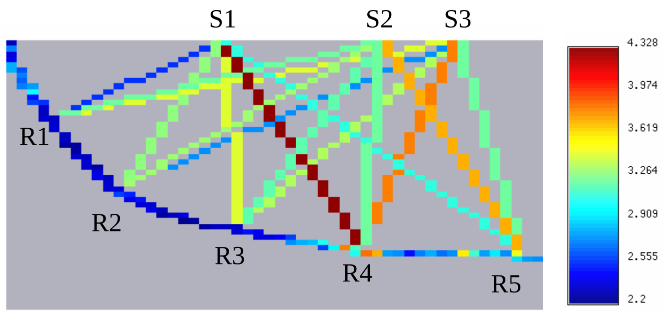
Figure 32: Reference field for deviated well VSP study



(a) Well location



(b) Histogram of the well data

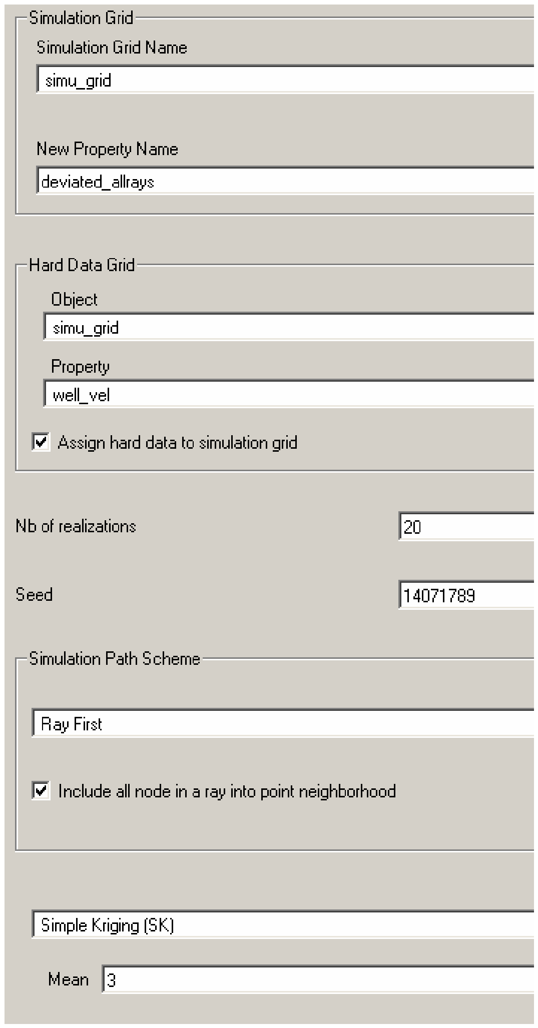


(c) All ray paths(the bottom curve is well path)

Ray #	Ray value
S1-R1	2.5289
S1-R2	3.2715
S1-R3	3.4473
S1-R4	4.3283
S1-R5	3.0627
S2-R1	3.2197
S2-R2	3.3121
S2-R3	3.1840
S2-R4	3.2247
S2-R5	3.7289
S3-R1	3.4413
S3-R2	2.7325
S3-R3	3.3631
S3-R4	3.8255
S3-R5	3.1974

(d) Average velocities along ray paths

Figure 33: The configuration of well and ray paths in deviated well VSP case



Simulation Grid

Simulation Grid Name: simu_grid

New Property Name: deviated_allrays

Hard Data Grid

Object: simu_grid

Property: well_vel

☑ Assign hard data to simulation grid

Nb of realizations: 20

Seed: 14071789

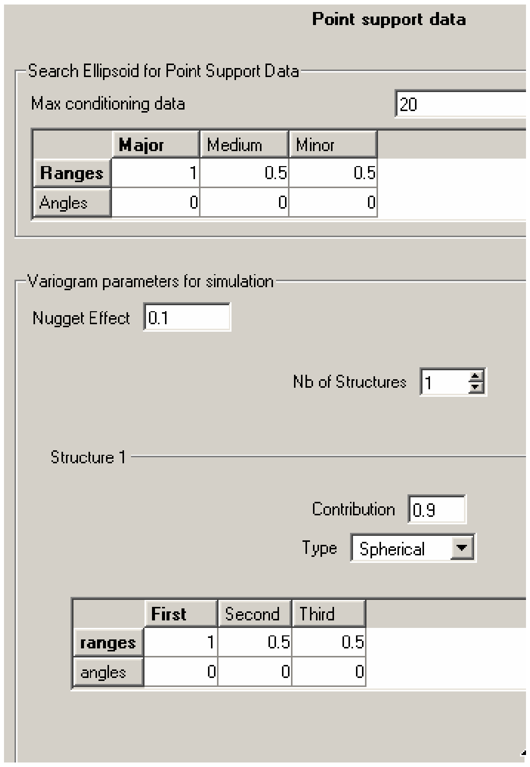
Simulation Path Scheme

Ray First

☑ Include all node in a ray into point neighborhood

Simple Kriging (SK)

Mean: 3



Point support data

Search Ellipsoid for Point Support Data

Max conditioning data: 20

	Major	Medium	Minor
Ranges	1	0.5	0.5
Angles	0	0	0

Variogram parameters for simulation

Nugget Effect: 0.1

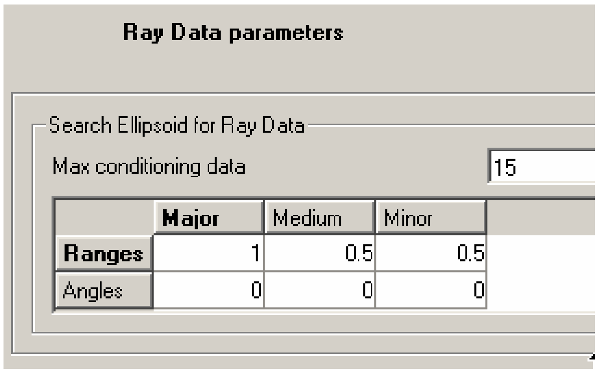
Nb of Structures: 1

Structure 1

Contribution: 0.9

Type: Spherical

	First	Second	Third
ranges	1	0.5	0.5
angles	0	0	0



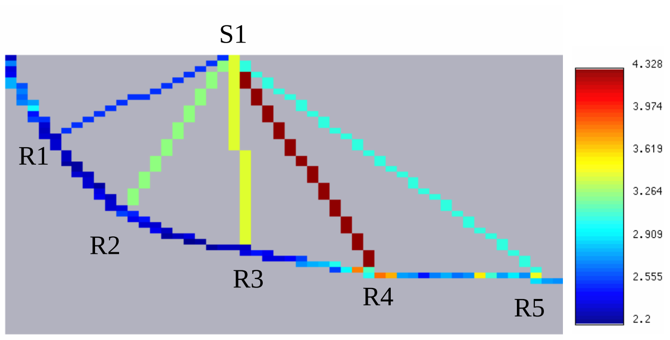
Ray Data parameters

Search Ellipsoid for Ray Data

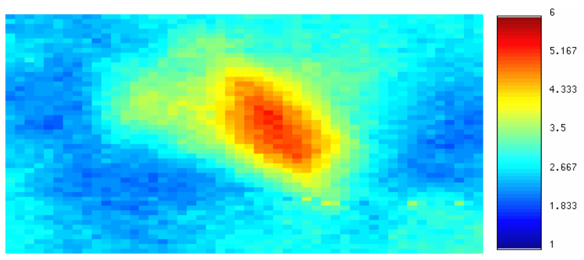
Max conditioning data: 15

	Major	Medium	Minor
Ranges	1	0.5	0.5
Angles	0	0	0

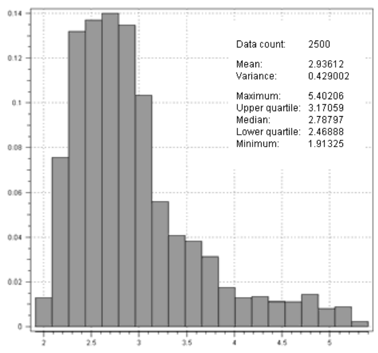
Figure 34: Parameter settings for the deviated well VSP case



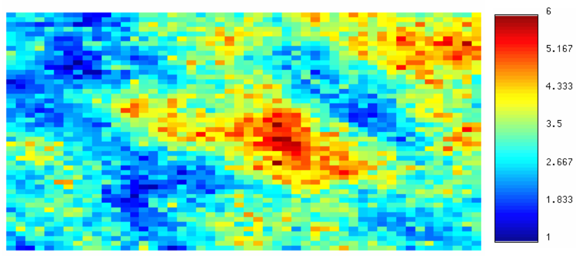
(a) Ray configuration



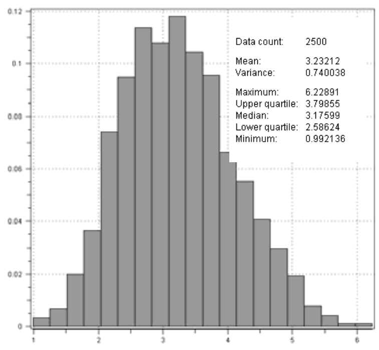
(b) E-type



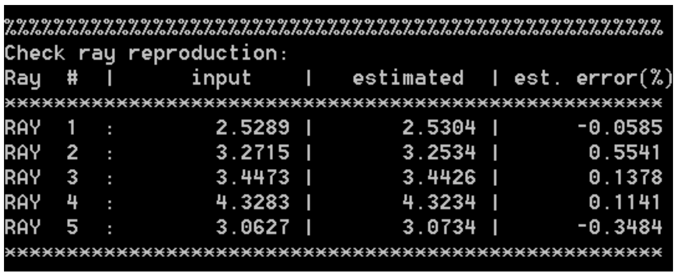
(c) Histogram of E-type



(d) One realization



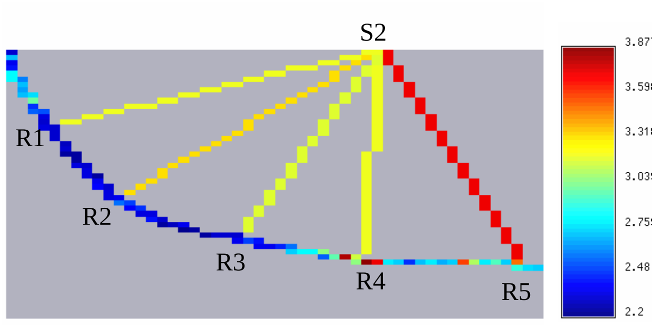
(e) Histogram of the realization



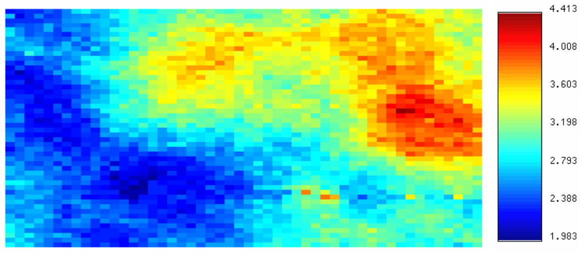
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Check ray reproduction:
Ray  #  |      input    |   estimated   | est. error(%)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
RAY  1  :        2.5289 |        2.5304 |      -0.0585
RAY  2  :        3.2715 |        3.2534 |       0.5541
RAY  3  :        3.4473 |        3.4426 |       0.1378
RAY  4  :        4.3283 |        4.3234 |       0.1141
RAY  5  :        3.0627 |        3.0734 |      -0.3484
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

(f) Ray reproduction check

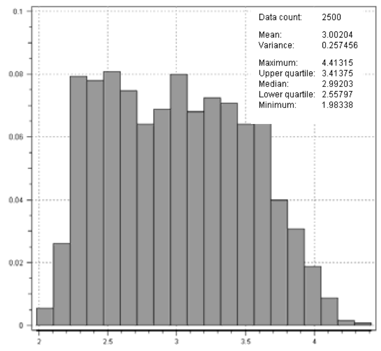
Figure 35: Results conditioned to shot 1 ray data



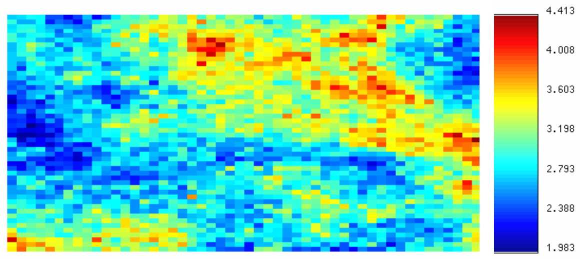
(a) Ray configuration



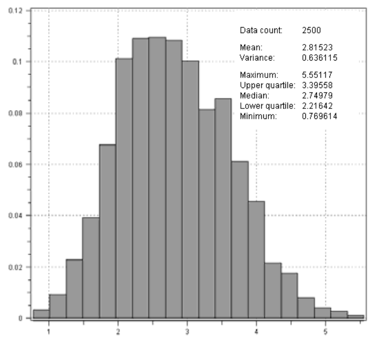
(b) E-type



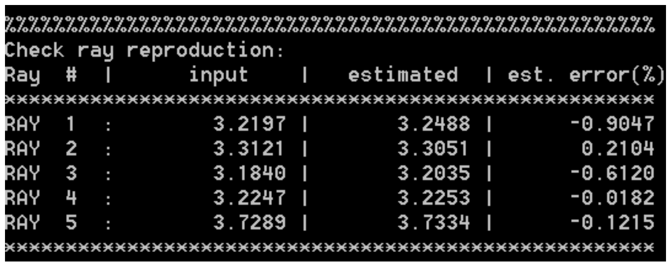
(c) Histogram of E-type



(d) One realization



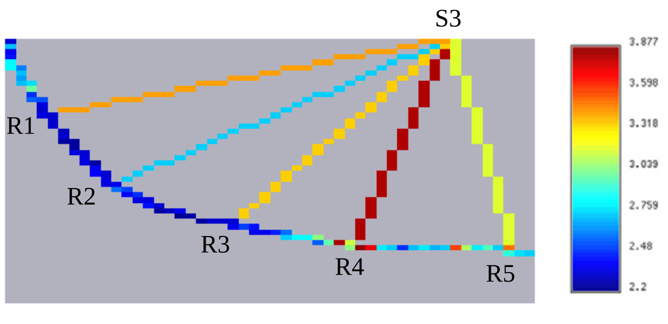
(e) Histogram of the realization



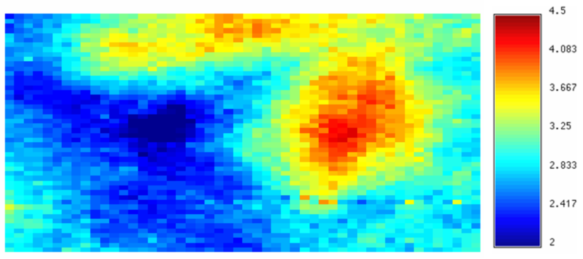
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Check ray reproduction:
Ray  #  |      input     |   estimated  | est. error(%)
*****************************************************
RAY  1  :        3.2197 |      3.2488 |      -0.9047
RAY  2  :        3.3121 |      3.3051 |       0.2104
RAY  3  :        3.1840 |      3.2035 |      -0.6120
RAY  4  :        3.2247 |      3.2253 |      -0.0182
RAY  5  :        3.7289 |      3.7334 |      -0.1215
*****************************************************
```

(f) Ray reproduction check

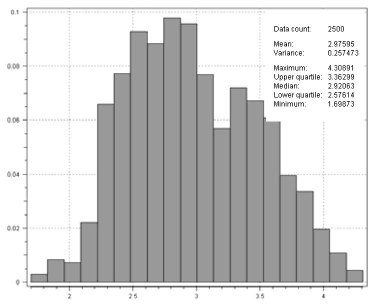
Figure 36: Results conditioned to shot 2 ray data



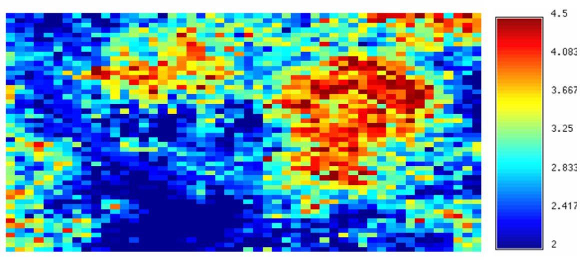
(a) Ray configuration



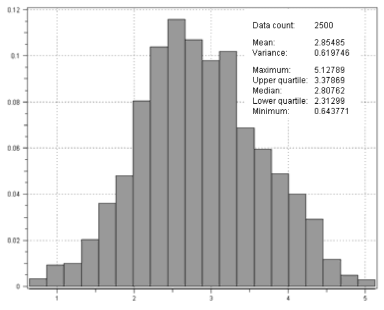
(b) E-type



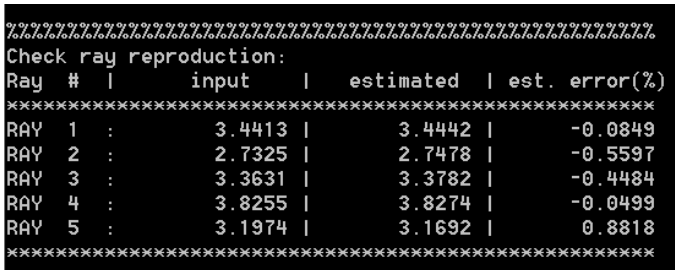
(c) Histogram of E-type



(d) One realization



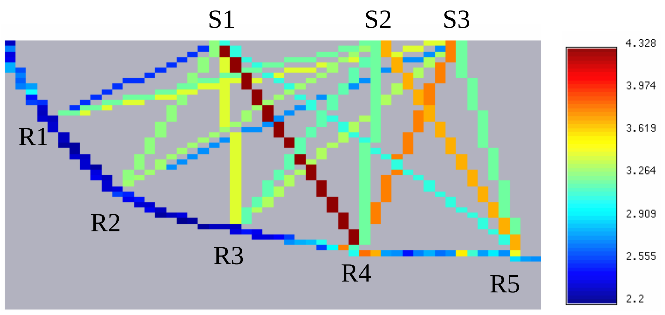
(e) Histogram of the realization



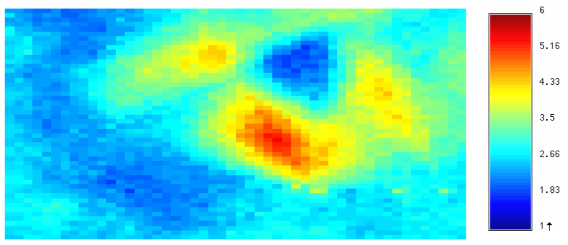
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Check ray reproduction:
Ray  #  |      input     |   estimated   | est. error(%)
*****************************************************
RAY  1  :         3.4413 |        3.4442 |      -0.0849
RAY  2  :         2.7325 |        2.7478 |      -0.5597
RAY  3  :         3.3631 |        3.3782 |      -0.4484
RAY  4  :         3.8255 |        3.8274 |      -0.0499
RAY  5  :         3.1974 |        3.1692 |       0.8818
*****************************************************
```

(f) Ray reproduction check

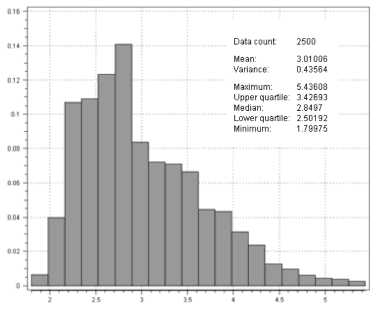
Figure 37: Results conditioned to shot 3 ray data



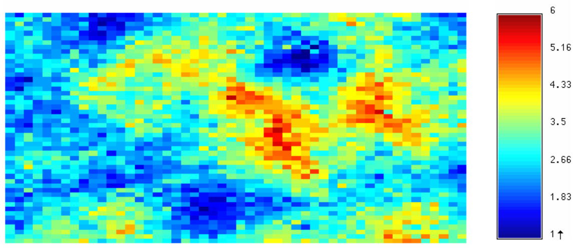
(a) Ray configuration



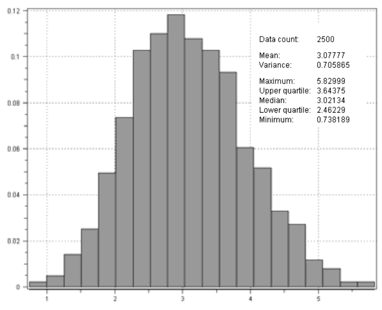
(b) E-type



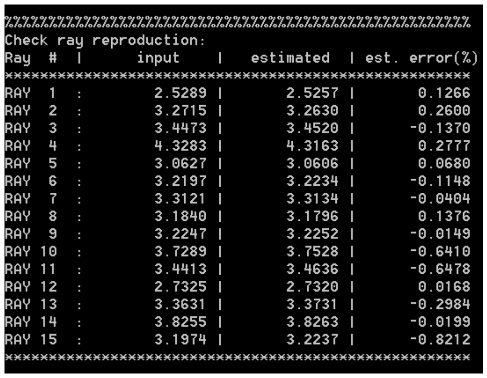
(c) Histogram of E-type



(d) One realization



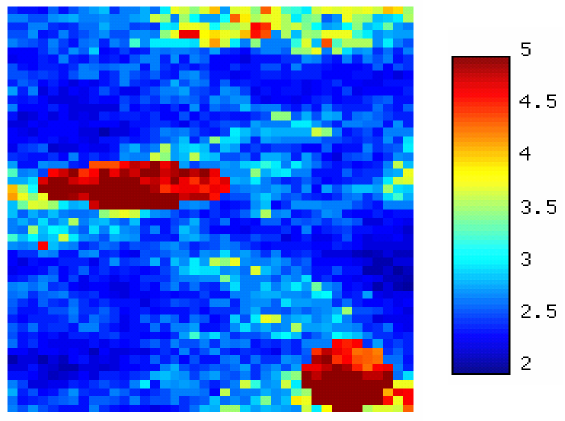
(e) Histogram of the realization



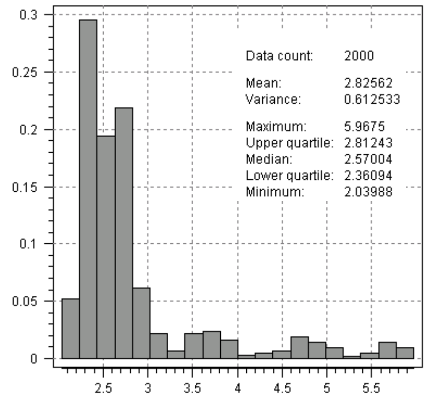
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Check ray reproduction:
Ray   #  |      input     |    estimated   | est. error(%)
***************************************************
RAY   1  :        2.5289 |        2.5257 |        0.1266
RAY   2  :        3.2715 |        3.2630 |        0.2600
RAY   3  :        3.4473 |        3.4520 |       -0.1370
RAY   4  :        4.3283 |        4.3163 |        0.2777
RAY   5  :        3.0627 |        3.0606 |        0.0680
RAY   6  :        3.2197 |        3.2234 |       -0.1148
RAY   7  :        3.3121 |        3.3134 |       -0.0404
RAY   8  :        3.1840 |        3.1796 |        0.1376
RAY   9  :        3.2247 |        3.2252 |       -0.0149
RAY  10  :        3.7289 |        3.7528 |       -0.6410
RAY  11  :        3.4413 |        3.4636 |       -0.6478
RAY  12  :        2.7325 |        2.7320 |        0.0168
RAY  13  :        3.3631 |        3.3731 |       -0.2984
RAY  14  :        3.8255 |        3.8263 |       -0.0199
RAY  15  :        3.1974 |        3.2237 |       -0.8212
***************************************************
```

(f) Ray reproduction check

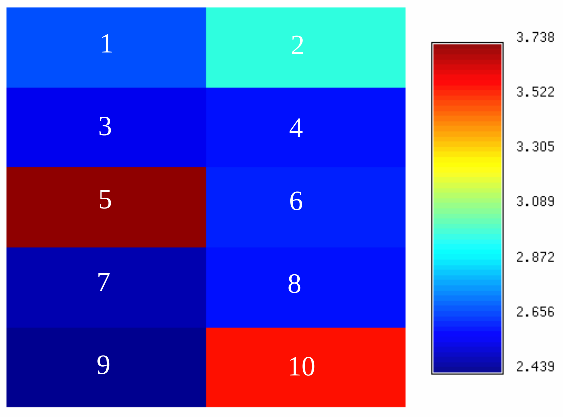
Figure 38: Results conditioned to all 3 shots ray data



(a) Reference velocity model



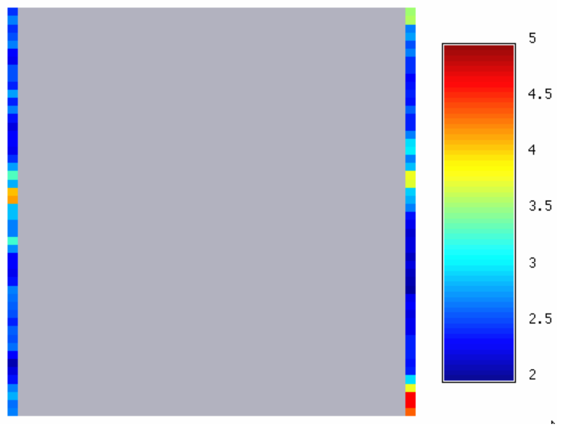
(b) Histogram of reference model



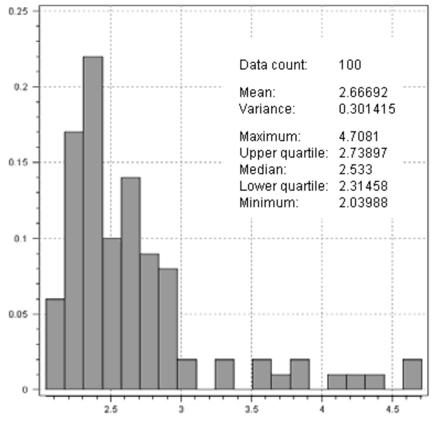
(c) The blocks

Block #	Ray value
1	2.6821
2	2.9647
3	2.5554
4	2.6035
5	3.7382
6	2.6208
7	2.4777
8	2.5926
9	2.4393
10	3.5818

(d) The values of blocks

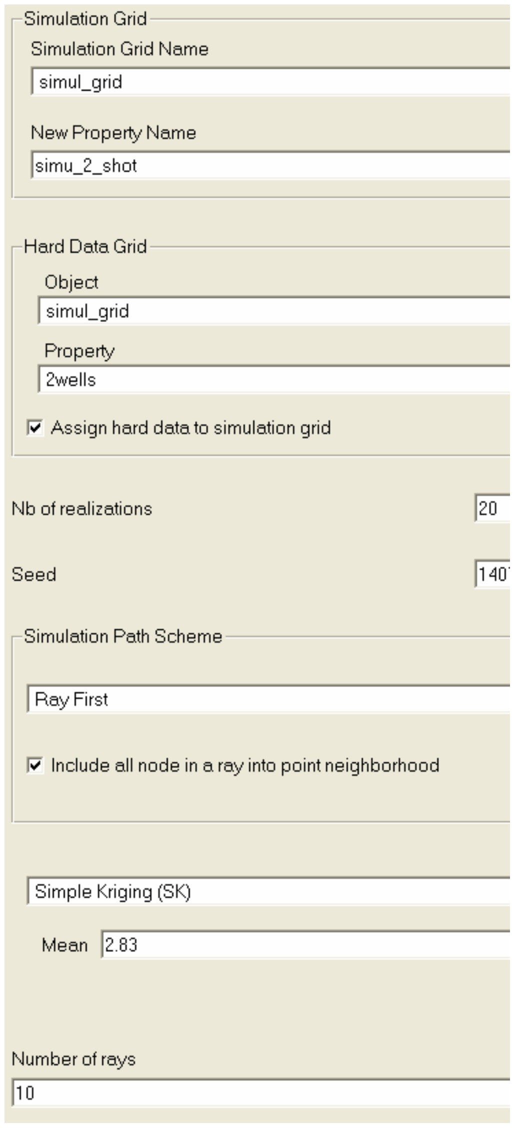


(e) Well data



(f) Histogram of well data

Figure 39: The reference model and data for the downscaling case study



Simulation Grid

Simulation Grid Name: simul_grid

New Property Name: simu_2_shot

Hard Data Grid

Object: simul_grid

Property: 2wells

☑ Assign hard data to simulation grid

Nb of realizations: 20

Seed: 140

Simulation Path Scheme

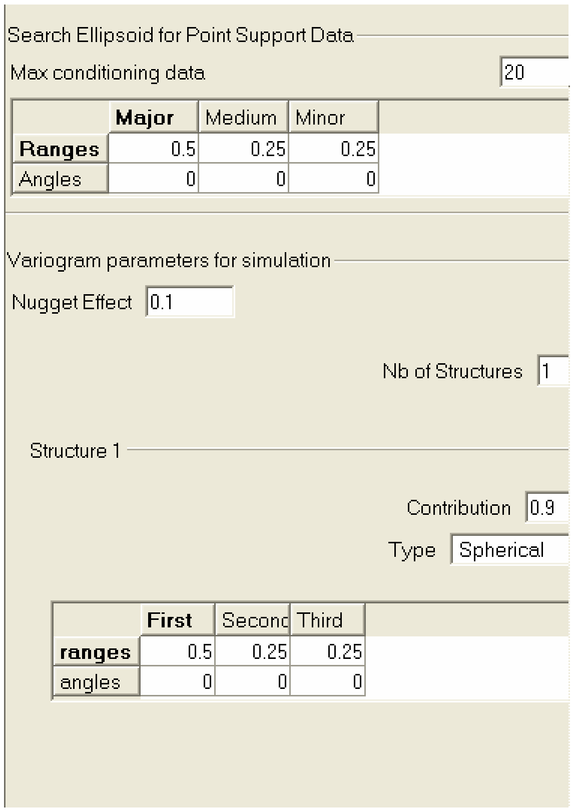
Ray First

☑ Include all node in a ray into point neighborhood

Simple Kriging (SK)

Mean: 2.83

Number of rays: 10



Search Ellipsoid for Point Support Data

Max conditioning data: 20

	Major	Medium	Minor
Ranges	0.5	0.25	0.25
Angles	0	0	0

Variogram parameters for simulation

Nugget Effect: 0.1

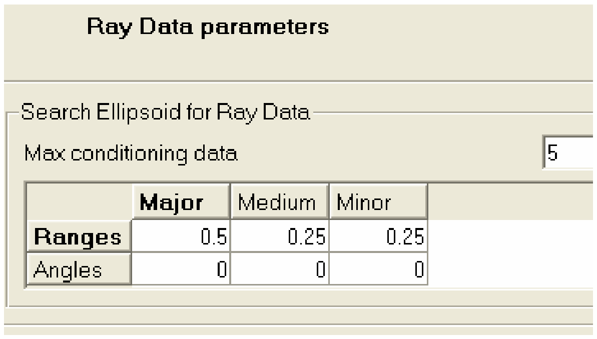
Nb of Structures: 1

Structure 1

Contribution: 0.9

Type: Spherical

	First	Second	Third
ranges	0.5	0.25	0.25
angles	0	0	0



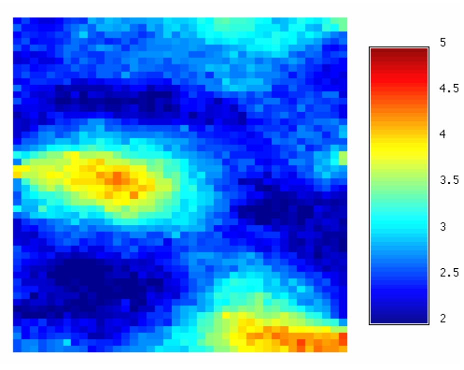
Ray Data parameters

Search Ellipsoid for Ray Data

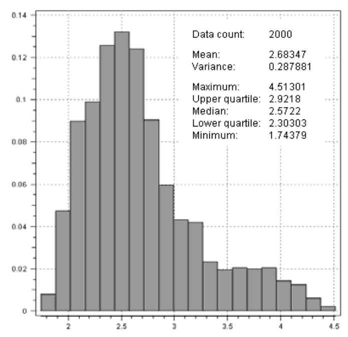
Max conditioning data: 5

	Major	Medium	Minor
Ranges	0.5	0.25	0.25
Angles	0	0	0

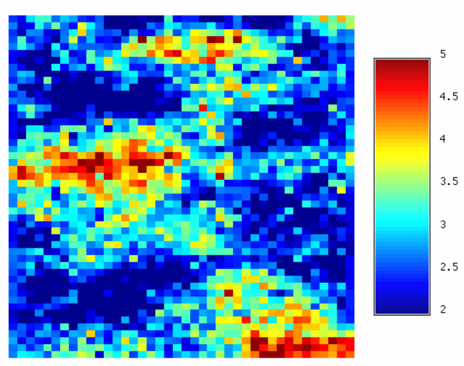
Figure 40: Parameter settings for the downcaling case



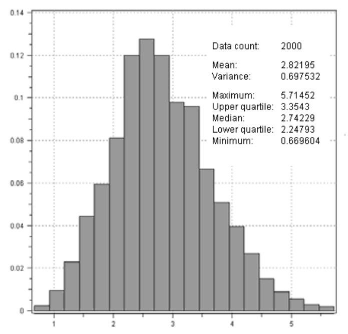
(a) E-type



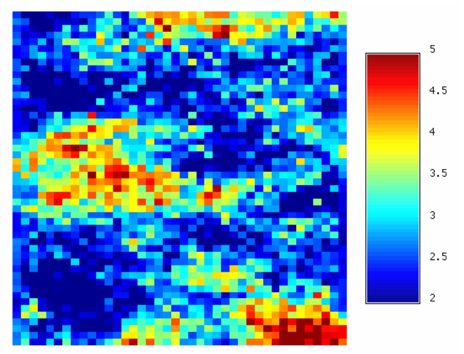
(b) Histogram of E-type



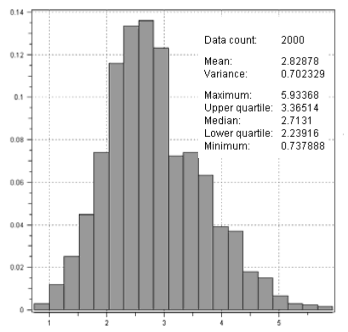
(c) Realization 1



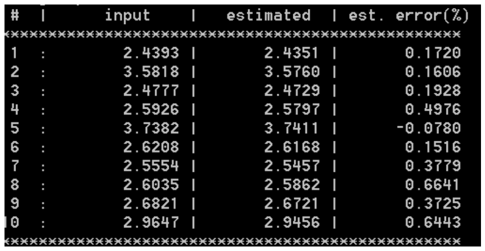
(d) Histogram of real. 1



(e) Realization 2



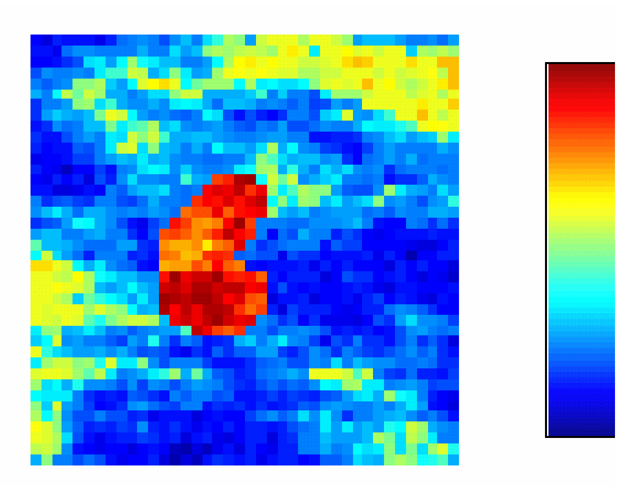
(f) Histogram of real. 2



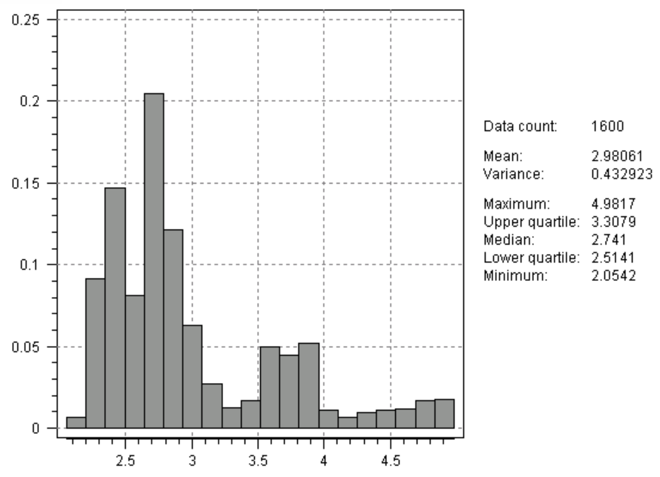
```
#  |     input    |   estimated   | est. error(%)
**********************************************
1  :       2.4393 |        2.4351 |      0.1720
2  :       3.5818 |        3.5760 |      0.1606
3  :       2.4777 |        2.4729 |      0.1928
4  :       2.5926 |        2.5797 |      0.4976
5  :       3.7382 |        3.7411 |     -0.0780
6  :       2.6208 |        2.6168 |      0.1516
7  :       2.5554 |        2.5457 |      0.3779
8  :       2.6035 |        2.5862 |      0.6641
9  :       2.6821 |        2.6721 |      0.3725
10 :       2.9647 |        2.9456 |      0.6443
**********************************************
```

(g) Block data reproduction check

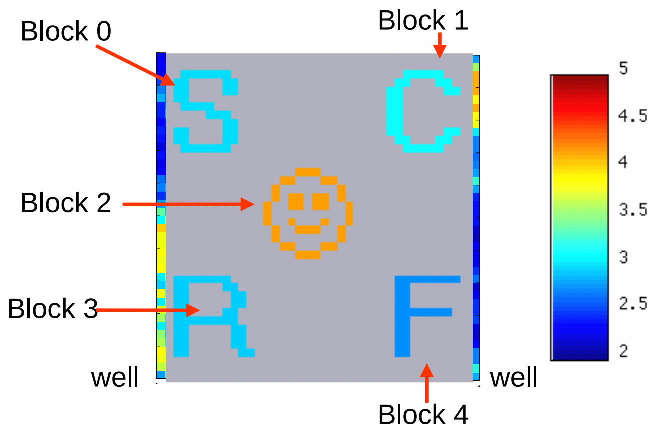
Figure 41: Downscaling results



(a) Reference velocity model



(b) Histogram of reference velocity model

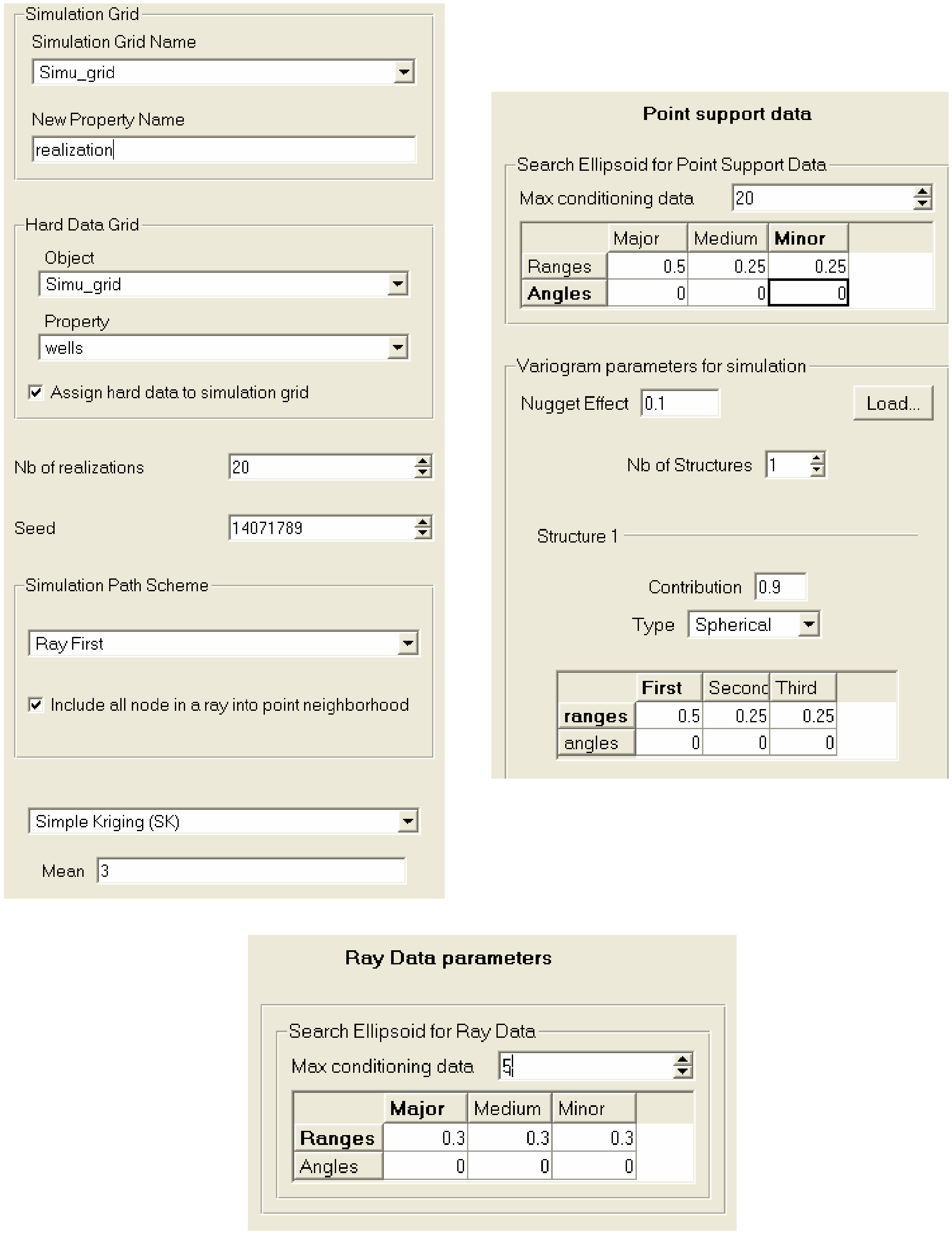


(c) Five blocks extracted from the reference model

Block#	Average value
0	3.0105
1	3.1865
2	4.2222
3	2.9620
4	2.7925

(d) The average values of the five blocks

Figure 42: Reference model and five block average data



Simulation Grid

Simulation Grid Name: Simu_grid

New Property Name: realization

Hard Data Grid

Object: Simu_grid

Property: wells

☑ Assign hard data to simulation grid

Nb of realizations: 20

Seed: 14071789

Simulation Path Scheme

Ray First

☑ Include all node in a ray into point neighborhood

Simple Kriging (SK)

Mean: 3

Point support data

Search Ellipsoid for Point Support Data

Max conditioning data: 20

	Major	Medium	**Minor**
Ranges	0.5	0.25	0.25
Angles	0	0	0

Variogram parameters for simulation

Nugget Effect: 0.1

Nb of Structures: 1

Structure 1

Contribution: 0.9

Type: Spherical

	First	Second	Third
ranges	0.5	0.25	0.25
angles	0	0	0

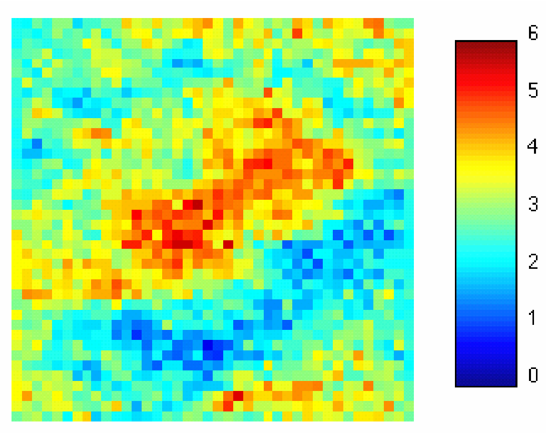
Ray Data parameters

Search Ellipsoid for Ray Data

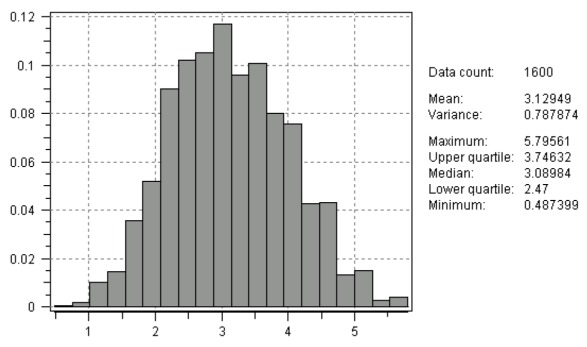
Max conditioning data: 5

	Major	Medium	Minor
Ranges	0.3	0.3	0.3
Angles	0	0	0

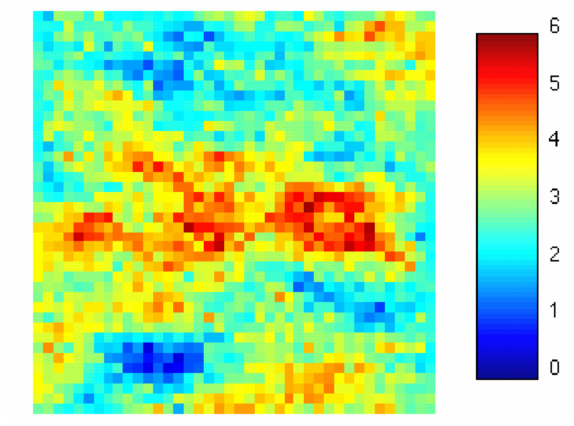
Figure 43: Parameter settings for the 5-block case



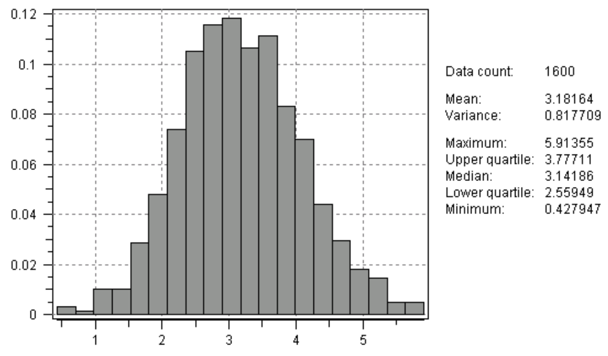
(a) Realization 1



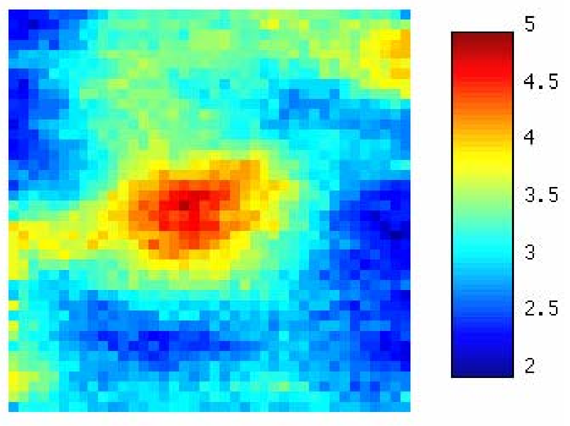
(b) Histogram of Realization 1



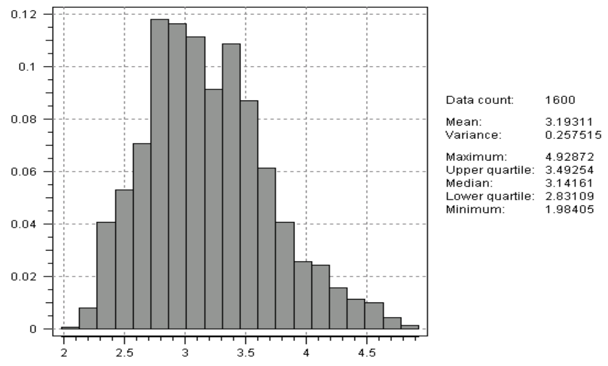
(c) Realization 2



(d) Histogram of Realization 2

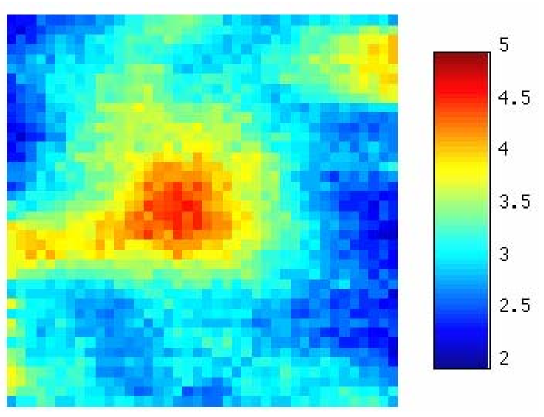


(e) E-type from 20 realizations

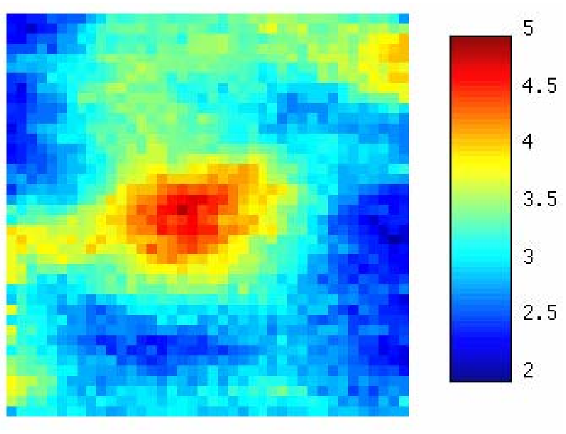


(f) Histogram of E-type

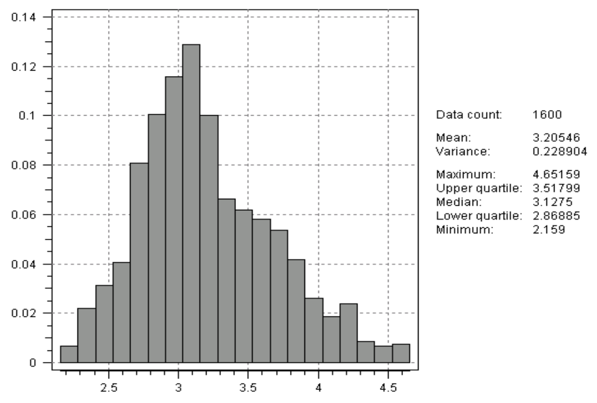
Figure 44: Realizations and E-type of the 5-block case



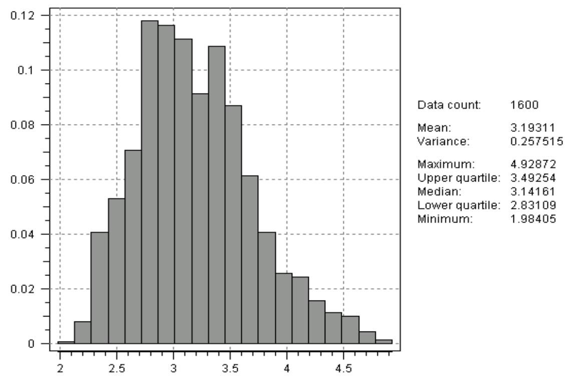
(a) E-type (fully random path)



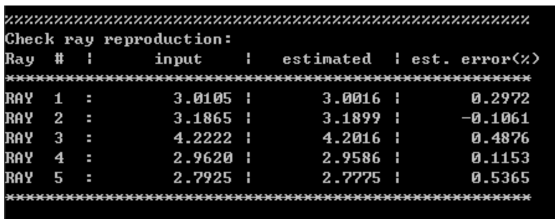
(b) E-type (ray-first path)



(c) Histogram of E-type (fully random path)

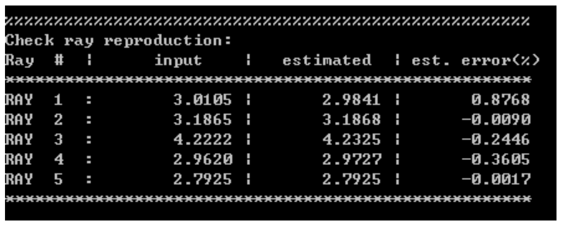


(d) Histogram of E-type (ray-first path)



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Check ray reproduction:
Ray  #  |       input     |   estimated    | est. error(%)
****************************************************************
RAY  1  :         3.0105 |         3.0016 |         0.2972
RAY  2  :         3.1865 |         3.1899 |        -0.1061
RAY  3  :         4.2222 |         4.2016 |         0.4876
RAY  4  :         2.9620 |         2.9586 |         0.1153
RAY  5  :         2.7925 |         2.7775 |         0.5365
****************************************************************
```

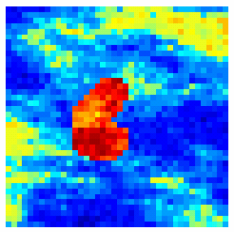
(e) Ray production check (fully random path)



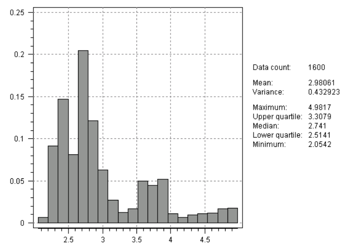
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Check ray reproduction:
Ray  #  |       input     |   estimated    | est. error(%)
****************************************************************
RAY  1  :         3.0105 |         2.9841 |         0.8768
RAY  2  :         3.1865 |         3.1868 |        -0.0090
RAY  3  :         4.2222 |         4.2325 |        -0.2446
RAY  4  :         2.9620 |         2.9727 |        -0.3605
RAY  5  :         2.7925 |         2.7925 |        -0.0017
****************************************************************
```

(f) Ray production check (ray-first path)

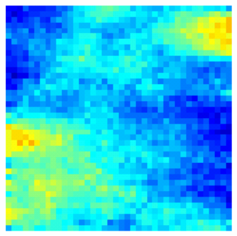
Figure 45: Results for different simulation paths for the 5-block case



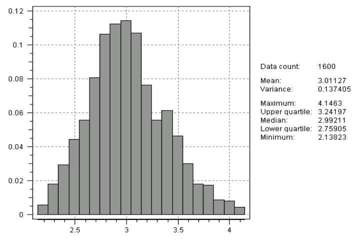
(a) Reference model



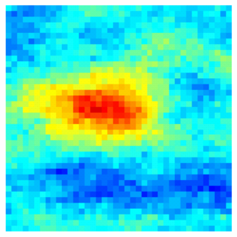
(b) Histogram of reference model



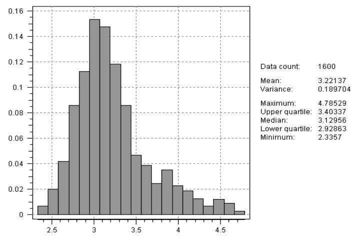
(c) E-type (only hard data)



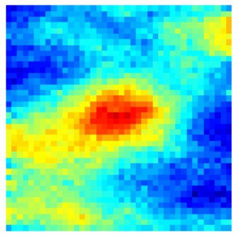
(d) Histogram of E-type (only hard data)



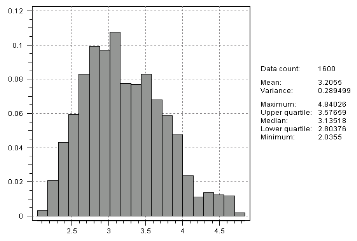
(e) E-type (only block data)



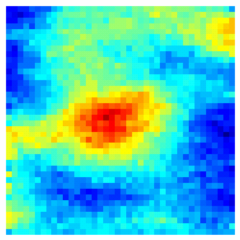
(f) Histogram of E-type (only block data)



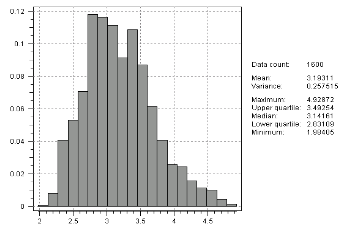
(g) E-type (well + central block)



(h) Histogram of E-type (well + central block)



(i) E-type (well + all block data)



(j) Histogram of E-type (well + all block data)

Figure 46: Comparison of E-types resulting from different sets of conditioning data