A New Version of kt3d with Test Cases

Jared L. Deutsch and Clayton V. Deutsch

Kriging encompasses a number of computationally intensive methods for estimation with spatial data. The kt3d *program is a widely used kriging program capable of performing many kriging variants. Although the theory of kriging has remained unchanged since long before this initial version, numerous kriging variants and measures have been introduced since then. Many of these kriging variants have been introduced in specialized versions of* kt3d *designed to address a specific problem. An updated, backwards compatible version of* kt3d *that includes some of the more common recent kriging variations and measures has been developed. Kriging programs are complex and have many interactions with user interfaces, databases, matrix solvers and other unit operations in the software. Good practice would be to test these programs carefully against established results. Such recursive testing could not include all possible combinations of parameters, but some comfort would be gained if the block estimates and estimation variances stayed the same when aspects of the program were updated or recompiled. This paper presents a new, modern version of* kt3d*,* kt3dn*, and a series of recursive test cases which are implemented to compare the results of* kt3d *and* kt3dn*. The recursive test case framework introduced in this paper could be used to test future kriging programs.*

Introduction and Criteria for a Modern Kriging Program

The original GSLIB version of kt3d has been updated numerous times since its initial release in 1989 (Deutsch and Journel, 1998). The most recent version of kt3d, version 3.0 (Deutsch, 2005), is a general purpose kriging program capable of performing point or block kriging with simple kriging, ordinary kriging, kriging with a polynomial trend. Over the years, numerous additions have been made to kt3d and released as specialized versions for solving specific kriging problems. Some of the many specialized versions of kt3d include kt3d_com which incorporated collocated cokriging without the Gaussian assumption and an alternate implementation of a locally varying mean (Neufeld, 2005), kt3d_lva which performs kriging with a locally varying anisotropy field (Boisvert, 2007) and kt3dd which can perform kriging with up to 4-D data (additional dimension to account for data error and scale) and added inverse distance estimation as an option (Deutsch and Deutsch, 2010). Integrating all of the available variants of kt3d would be time consuming and challenging. In addition to the different kriging variants, there are also a number of kriging measures, such as kriging efficiency and the slope of regression, which can be useful in determining the efficacy of a kriged estimate. The authors believe that a general purpose, modern variant of kt3d which implements some of the more widely used features and measures while avoiding needless complexity would be far more useful. The following criteria were used in the development of modern version of kt3d: kt3dn.

A modern version of kt3d should be backward compatible with parameter files from kt3d version 3.0. This backwards compatibility accomplishes two tasks: (1) it encourages comparison of results from both kriging programs since the user does not have to reconfigure a new parameter file and (2) it encourages the practitioner to use kt3dn and look at some of the reported measures which may aid in making better decisions.

Although backwards compatibility with older versions of kt3d is a prerequisite for the modern version, the goal of kt3dn is to incorporate the more widely used kriging variants and measures. Some of these measures, such as reporting the number of data used in constructing the kriged estimate are useful in picking out poorly informed areas of the estimation grid. Other measures, such as the theoretical slope of regression and the kriging efficiency should also be included (Krige, 1997).

The theoretical slope of regression and kriging efficiency are defined for simple and ordinary kriging. The kriging efficiency, *KE*, of a block estimate with block variance *BV* and kriging variance *KV* is originally defined by Krige (1997) as:

$$KE(\%) = \frac{BV - KV}{BV} \tag{1}$$

The block variance is the variance of the true block values, σ^2. The kriging variance is the estimation variance from kriging, σ_K^2. Calculating a kriging efficiency for a block estimate is a method to report the kriging variance, normalized by the true block variance as a percentage. An efficiency of 100% indicates that the variance of the block estimates is exactly equal to the true block variance. A low kriging efficiency implies a high kriging variance. Given this definition, a high kriging efficiency is desirable.

The slope of regression is an approximation of the conditional expectation of the estimates given the true values. A slope of regression, *b*, equal to 1 implies that the estimates are conditionally unbiased.

$$b = \frac{Cov\left\{Z_V, Z_V^*\right\}}{\sigma^2_{Z_V^*}} = \frac{\sum_{i=1}^{n} \lambda_i \bar{C}(v_i, V)}{\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j C(v_i, v_j)} \tag{2}$$

Simple kriging, which is theoretically conditionally unbiased will have a slope of regression of 1. Generally, ordinary kriging estimates will have a slope of regression less than 1 which means that high grade estimates are higher than the true values and low grade estimates are lower than the true block values. A review of the slope of regression and the relationship to conditional bias has been completed by Deutsch (2007). A kriging efficiency close to 1 is desirable to minimize the conditional bias of estimates. Krige (1997) proposes that a slope of regression greater than 0.95 is a requisite for conditional unbiasedness. Given the use of kriging efficiency and slope of regression by some practitioners in evaluating kriged estimates, a modern version of `kt3d` should report these measures. For a thorough discussion of kriging efficiency, refer to Paper 306 in this report (Deutsch and Deutsch, 2012).

A number of other features are also useful. In kriging search strategies, it is common to specify a maximum number of close data to be used. This is done to better estimate the local data mean, decrease reliance on global stationarity and increase computation speed. When evaluating the use of ordinary kriging with a limited search strategy, it is useful to understand the increase in estimation variance from restricting the search. This increase in estimation variance should be compared against the lowest theoretically possible estimation variance from a linear estimator: the global simple kriging variance. Performing global simple kriging for every estimate location with large amounts of data is not practical; instead the use of a very large number of data, such as 100 is suggested. This arbitrary number of data could be adjusted. The (nearly) global simple kriging variance should be reported to assist the practitioner in understanding the tradeoff between excessive reliance on stationarity and increased estimation variance.

The selection of a maximum number of close data to use can be time consuming depending on the type of estimate desired, see (Deutsch and Deutsch, 2012). To assist in this process, auto search optimization (ASO) would be useful. Kriging could be performed with different numbers of data automatically in the kriging program and the results post processed to find the one that best matches the smoothness predicted by volume variance relations. Ideally, this post processing could be done using existing simulation processing programs.

There are issues that accompany a restricted search radius. Figure 1 shows a common situation where, the data have been blocked to the same resolution as the block model and the data are available along a drillhole oriented perpendicular to one of the model coordinates.

Consider estimation at the block location labeled 0. Most search strategies specify that a maximum number of close data be used. Choosing an odd number of data from the drillhole would give a unique result from the search (and the kriging), but choosing an even number would lead to some ambiguity since the distance from 0 to composite 2 is the same as the distance from 0 to composite 4 and so on, that is, d02=d04 and d01=d05. If two data were to be kept, should sample 2 or 4 be chosen together with 3? The authors are not aware of a reproducible “despiking” procedure for choosing between data that have the same distance to the location being estimated.

There are a number of obvious suggestions: (1) design the test cases to avoid tied distances, (2) presort the data according to some arbitrary vector orientation to try and force the same values to be chosen, and (3) have the kriging program flag those estimates where the *n*+1 data is at the same distance as the last *n*th data used in kriging. Also, in practice, the kriging search should perhaps be set large

enough that increasing from *n* to *n*+1 does not significantly change the estimate. Nevertheless, the problem is annoying in those specific cases where completely reproducible results are required for testing. All three obvious suggestions should be implemented in a modern kriging program. In addition to preprocessing the data by sorting along an arbitrary vector, the data should also be checked for duplicates. Duplicates will result in a singular kriging matrix and unestimated grid. Searching the data for duplicates should be done before kriging.

Of the specialized kriging variants discussed, collocated cokriging and inverse distance are the most commonly used. Collocated cokriging is a useful variant of kriging for using secondary data sources. Inverse distance estimation is very useful in quickly checking a kriged model. Parameters required for inverse distance estimation are very limited; often a slight tweaking of the power applied to the distance functions is all that is required to have a reasonable inverse distance estimate. The kriging model can be visually compared with the inverse distance estimate to look for anomalous kriged estimates. Both of these estimation methods have been included in `kt3dn`.

New Features in `kt3dn`

Every effort has been made to ensure that `kt3dn` is fully backwards compatible with parameter files used for `kt3d` (version 3.0). To ensure backwards compatibility, the new features for `kt3dn` assume a default value or are not activated if a parameter is missing in the parameter file.

There are a number of features which have been added which are non-optional. We believe that checking the data for duplicate values and pre-sorting the data along an arbitrary vector should always be done to increase the reproducibility of kriging results. Should the user wish not to use these features, the relevant source code could be removed and the program recompiled. These areas are clearly indicated in the code.

Duplicate Removal

Duplicates in data files can arise from a number of sources including the updating of data at old locations or the concatenation of different data files. Duplicate data must be removed before kriging to prevent a singular kriging matrix. Duplicate removal is trivial if all data points at the same location have the same value; however this problem has numerous solutions if data values are different. In a previous CCG paper, (Deutsch and Deutsch, 2010), three methods were implemented for duplicate removal. These three options were 1) random selection of a duplicate value to keep and removal of all other duplicated points, 2) using the average of all data values at a location as the value or 3) choosing the duplicate value closest to the mean of immediately surrounding points. The selection of which method to use will depend on the reason for duplicates in the data file. However, given no information on the reason for duplicate values in the data file it is reasonable to use the average of the data values. This approach is also repeatable. If the data values were significantly different and a random duplicate value was chosen, nearby estimates would not be preserved between runs of the kriging program.

In `kt3dn`, a search of all data values is run before kriging to look for duplicates. The minimum distance between data values is hardcoded as:

$$d_{\min} = \frac{\big(\max(x) - \min(x)\big) + \big(\max(y) - \min(y)\big) + \big(\max(z) - \min(z)\big)}{100,000} \tag{3}$$

If multiple data locations are within d_{min} of each other, the average of the data values is used as the data value at the duplicate location. Should duplicate data values be found, the location and average value used in estimation are written to the screen and debug file.

Data Presorting

Figure 1 showed a common case which occurs when estimating a grid near evenly spaced drillhole data. If a limited number of data are used for the kriging estimate, then the data used in the estimate (if they are equidistant) will depend on the original order in the file. The `kt3d` kriging program in GSLIB takes no special steps for this problem. The data within the valid superblocks are identified, the distances are calculated to all those data, and then the data are accepted by closest anisotropic distance subject to octant constraints until the maximum data are found. The data are sorted by distance according to a

Fortran translation of Algorithm 271, quickersort, (Scowen, 1965). The position of data at tied distances depends on the position of the data in the original data file and the sequence of steps required for sorting. Following Figure 1, the program would find the closest to be 3,2,4,1,5,6,7 if the data are ordered from the top in the original data file and would find 3,4,2,5,1,6,7 if the data are ordered from the bottom. Clearly, the kriged estimate would be different if only two or four data were retained. The kriging variance would not change in this case, but it could change if there are data from multiple drillholes. This makes the results of kt3d, and programs with similar search algorithms, not reproducible in all circumstances. This makes recursive automatic testing difficult to fully implement and test.

One solution is to presort the data along an arbitrary vector (Figure 2). This presort prevents data configuration in the file from changing the selection of data values in kriging since data at tied distances will always be chosen in the order of the presort vector. Presorting must be done after duplicate removal, since the presence of any duplicates will mean that there is no presort vector will result in a unique sorting of the data file.

This is implemented in `kt3dn` by first sorting the data along the rotation vector [12.5°,12.5°,12.5°] (see GSLIB for angle specifications). If the distance of any data along the presort vector are tied, then the presort vector angles are incremented as:

$$\text{Presort Angles [ang1,ang2,ang3]} = \begin{cases} \text{Iteration } 1 \rightarrow \text{Presort Angles} = [12.5°,12.5°,12.5°] \\ \text{Iteration 2-10} \rightarrow \text{Presort Angles} = \text{Presort Angles} + 3°_{\text{ang1}} \\ \text{Iteration 11-20} \rightarrow \text{Presort Angles} = \text{Presort Angles} + 3°_{\text{ang2}} \\ \text{Iteration 21-30} \rightarrow \text{Presort Angles} = \text{Presort Angles} + 3°_{\text{ang3}} \\ \text{Iteration } 31 \rightarrow \text{Presort failed, write warning and proceed with estimation} \end{cases}$$

If no unique presort vector is found, then a warning is written to the screen and kriging proceeds as before. The ultimate presort vector used is written to the screen.

New Debugging Framework

The number of new measures implemented has necessitated a new debugging framework. In `kt3d`, the amount of debugging information written out was specified by `idbg` in the parameter file, nominally 0, 1, 2 or 3. The information written out was:

0. No debugging information written out
1. Write out the block covariance
2. Write out the block covariance and kriging matrix weights for each estimate location
3. All of the above plus the original covariance matrices for kriging at each estimate location

This debugging framework has worked well for the past years. In only rare circumstances would the practitioner want to check the kriging weights and matrices so set the debugging level higher than 1.

A different set of debugging options are used in `kt3dn` to specify whether the additional kriging measures are calculated and written out with the kriging estimates and estimation variance. The new `idbg` parameters are:

0. No debugging information written out
1. Write out the block covariance and summary information such as the average estimate value and average estimation variance
3. All of the above plus the number of data used in making the estimate, a flag if there is an equidistant next nearest neighbor not used in the estimate, the theoretical slope of regression and kriging efficiency and more summary information
5. All of the above plus the global simple kriging variance for that estimate location
10. All of the above plus all of the kriging matrices (old debugging level 3)

These additional output options are explained in more detail in the following sections.

Kriging Measures

A number of kriging measures are written out alongside the estimate and estimation variance in the output file if the debugging level is set to 3 or greater. For block estimation, the new gridded output file will now look like Table 1.

Table 1: Sample gridded output file with kt3dn and a debugging level of 5.

Line	Output
1	KT3DN ESTIMATES WITH: Normal Score Transform:Amoco2D
2	8 65 65 1
3	Estimate
4	EstimationVariance
5	Number of data used
6	Equidistant NN?
7	Slope of regression
8	Kriging efficiency (DK)
9	Global SK Variance
10	Kriging efficiency (JD/CD)
11	-1.2655458 0.39493874 16 0 0.923142 0.590505 0.384616 0.973862
12	-1.3138933 0.29320785 16 0 0.951801 0.695985 0.288239 0.983055

The first kriging measure output is the number of data used in the estimate. Depending on the kriging search parameters and data configuration, this number will vary throughout the estimation area. The second measure output is a flag if there is an equidistant next nearest neighbor that could have been used in the estimate. If another data was equidistant to the block being estimated but not included in the estimate due to restrictions on the number of data, this value is set to 1. The third measure output is the theoretical slope of regression. This is calculated according to equation 2. The slope of regression will always be 1 for simple kriging. Danie Krige's kriging efficiency is output according to equation 1. For a debugging level of 5 or greater, the approximated global simple kriging variance and kriging efficiency (proposed in (Deutsch and Deutsch, 2012)) are output.

Global Simple Kriging Variance

For a debugging level of 5 or greater, the global simple kriging variance is approximately calculated and written out alongside the other kriging measures. The global simple kriging variance is the theoretical minimum estimation variance for a linear, unbiased estimator. The difference between the kriging variance from simple or ordinary kriging and the global simple kriging variance represents the tradeoff between a decreased reliance on stationarity and increased estimation variance.

The global simple kriging variance is approximated by considering the closest 100 data (this parameter can be increased, but will increase run times) within the specific search radius and calculating the simple kriging variance. The large number of data used in this calculation means that this simple kriging variance is a good approximation of the global simple kriging variance.

Max Per Drillhole

In kt3d, the user can specify a maximum number of data to be used per octant, but is not given the option to specify a maximum number of data per drillhole. This option has been added. Note that for cross validation, if drillhole identities were provided then data originating from the same drillhole as the point being estimated are excluded as per the original implementation in kt3d.

Collocated Cokriging

The implementation of collocated cokriging by Chad Neufeld (2011) in kt3d_com2 was preserved in this kt3dn. This implementation allows the user to specify a correlation between the collocated variables and estimates locations that do not have secondary data. This implementation can perform cokriging with non-Gaussian data.

Inverse Distance Estimation

The implementation of inverse distance estimation in kt3dd by Jared and Clayton Deutsch (2010) was included. Inverse distance estimation is a very useful method to check kriged results since it relies on very few parameters. If the kriged result is massively different than the inverse distance result, then it would

be wise to check the kriging parameters used in estimation. Inverse distance weights are assigned using the formula:

$$\lambda_i^* = \frac{1}{\left(h+\varepsilon\right)^{\alpha}} = \frac{1}{\left(h+c0(1)\right)^{cc(1)}} \tag{4}$$

These weights are then rescaled to sum to 1. When specifying the small additive factor ε, and power α, these are the nugget effect and contribution of the first variogram structure, respectively, in the parameter file. Also note that the anisotropic distance *h* is used. More continuous directions can have a longer range assigned to them to decrease the anisotropic distance. For isotropic distance weighting, the ranges can all be set to the same value. The search parameters, grid specification and other relevant parameters all remain the same as for simple or ordinary kriging.

Auto Search Optimization

Auto search optimization (ASO) was implemented in `kt3dn` to facilitate the optimization of the number of search data so that the kriged estimates matches the smoothness predicted by volume variance relations. For ASO, a lower bound and upper bound on the maximum number of data for kriging is supplied as well as an increment. For example, a lower bound of 8 and an upper bound of 20 with an increment of 4 would result in kriging with a maximum of 8, 12, 16 and 20 data used at each location. The results are output in the same manner as GSLIB realizations, such as from `SGSIM`, so that the results can be post processed using any of the realization post processing software in the GSLIB or CCG software catalogues.

Although the most common use of auto search optimization would be estimates with ordinary kriging, the ability to perform simple kriging or inverse distance kriging has also been preserved in this implementation. All necessary parameters are described in the parameter file.

Compiler Optimization and Program Speed

The Intel Fortran compiler has a number of optimization options to improve the execution speed of code. This can significantly decrease the run time of computationally expensive kriging programs. The main optimization options: /Od, /O1, /O2 and /O3 will disable program optimization (for debugging), minimize program size, maximize speed (normal release level, can be forced by /O2 in the command line section) and use high level optimization for maximum speed, respectively. In Visual Studio, these options are selected from the program property page (Figure 3).

Other options are available including the option to enable automatic parallelization or loop unrolling. In loop unrolling, more computations are added to the inside of the loop to minimize the number of loop iterations. Increasing the optimization level to 3 can cause the program to crash, particularly for large code files such as kt3d or kt3dn. After increasing the optimization level and recompiling, the new executable should always be tested against the results of previous versions with a lower level of optimization.

The program execution time (speed) for a number of test cases of varying size plotted for different levels of optimization (Figure 4). An optimization level of 3 was not used nor is recommended for kt3dn as this can cause program instability. Note that when interacting with files through a large amount of input/output, program speed will significantly decrease due to time spent writing to the disk. Therefore, a debug level of 10 will significantly decrease program execution speed. A debug level of 3 was used in all of these tests. The data files used are detailed in the following sections discussing each test case.

Increasing the optimization level approximately halved the execution time for each of these test cases. An optimization level of /O2 with I/O Buffering (loads a buffer of memory full of output or input data before inputting or outputting to decrease time spent waiting for the system to read or write to the disk was optimal and resulted in the exact same results as lower optimization levels. This optimization level is recommended.

***handkriging.dat* Test Case**

A very small kriging exercise, designed to demonstrate how to solve a kriging matrix by hand, is given in handkriging.dat. A script is included which kriges a very small grid around the two points given and calculates the same grid using inverse distance estimation to demonstrate how the results compare and how to set up the parameter files.

***red.dat* Test Case**

The test data, *red.dat*, is a 2D data set containing 67 drillholes with information on thickness, gold, silver, copper and zinc concentration. For the purposes of this test case, the thickness in original units was modeled. The goal of this recursive test case is to compare the output of `kt3d` with `kt3dn` when kriging is done using with an old `kt3d` parameter file or new `kt3dn` parameter file. This comparison is done by extracting the four decimal place estimates and estimation variances produced by both programs with both parameter files and comparing the outputs with the Unix utility `diff` which is included in `Cygwin`, a Linux-style command prompt useful when running GSLIB-style programs. A description of the files used in this test case is included a readme file supplied with the test case.

A summary file is generated which lists the differences between the estimates and estimation variances. An example summary file is shown in Table 2. It can be seen that the kriged output using either an old `kt3d` or `kt3dn` parameter file is identical. There are slight (4^{th} decimal place) differences in the kriged estimates between the old `kt3d` and new `kt3dn` estimates. For example, `kt3d` computes an estimate of 2.7948 compared to a `kt3dn` estimate of 2.7947 at line 124 in the kriged output file. This difference is due to the differences in rounding, precision and compiler optimization. Using a different computer with the exact same programs and scripts, the output was slightly different. All differences are minor and indicative of minor computational differences and not a systematic error. Rounding and precision differences may change between computers which handle calculations differently.

Table 2: Sample difference summary file comparing estimation differences for *red.dat* test case.

Line	Output
1	Summary of kriging program differences
2	------------------------------
3	kt3d_kt3dnpar_vs_kt3dpar
4	
5	Files out.kt3d_with_kt3dnpar and out.kt3d_with_kt3dpar are identical
6	------------------------------
7	kt3dn_kt3dnpar_vs_kt3dpar
8	
9	Files out.kt3dn_with_kt3dnpar and out.kt3dn_with_kt3dpar are identical
10	------------------------------
11	kt3dnpar_kt3d_vs_kt3dn
12	
13	124c124
14	< 2.7948 0.2493
15	---
16	> 2.7947 0.2493
17	313c313
18	< 1.3667 0.4061
19	---
20	> 1.3666 0.4061
21	------------------------------
22	kt3dpar_kt3d_vs_kt3dn
23	
24	124c124
25	< 2.7948 0.2493
26	---
27	> 2.7947 0.2493
28	313c313
29	< 1.3667 0.4061
30	---
31	> 1.3666 0.4061

Buggy Test Case

Recall the situation of evenly spaced drillholes illustrated in Figure 1. Assigning arbitrary data values to these locations and orienting in two different directions, the data files shown in Table 3 can be generated (note that header information has been put over columns to make the file easier to read).

Table 3: Data files used for the buggy test case.

Line	Data File 1 – eventop.dat				Data File 2 – evenbottom.dat			
1	Well	X	Y	Val	Well	X	Y	Val
2	2	1	5	0.	2	1	5	0.
3	1	4	7	1.	1	4	1	-1.
4	1	4	6	-1.	1	4	2	0.5
5	1	4	5	0.5	1	4	3	-0.5
6	1	4	4	1.	1	4	4	1.
7	1	4	3	-0.5	1	4	5	0.5
8	1	4	2	0.5	1	4	6	-1.
9	1	4	1	-1.	1	4	7	1.

Kriging was done using both `kt3d` and `kt3dn` parameter files with `kt3d` and `kt3dn` using 4 search data. The results were compared with the diff utility (Table 4). It is clear that the orientation changed the estimate from `kt3d` as different values were used in kriging. Note that although the estimate value changes, the kriging variance is not changed as the distance from the locations are the same.

Table 4: Difference summary file generated when kriging the buggy test case.

Line	Output
1	Summary of kriging program differences
2	------------------------------
3	kt3d_top_vs_bottom
4	
5	5c5
6	< -0.1110 0.4247
7	---
8	> 0.4653 0.4247
9	------------------------------
10	kt3dn_top_vs_bottom
11	
12	Files out.kt3dn_with_top and out.kt3dn_with_bottom are identical
13	------------------------------
14	top_kt3d_vs_kt3dn
15	
16	Files out.kt3d_with_top and out.kt3dn_with_top are identical
17	------------------------------
18	bottom_kt3d_vs_kt3dn
19	
20	5c5
21	< 0.4653 0.4247
22	---
23	> -0.1110 0.4247

***lgp.dat* Test Case**

A low grade 3D porphyry data set, lgp.dat, consisting of copper and molybdenum data was used to further demonstrate the importance of presorting a data set which contains evenly spaced drillhole data for repeatable kriging results. Here, differences between the kriging results from `kt3d` and `kt3dn` are observed at locations with equidistant next nearest neighbours which could have been included in the kriging neighbourhood. An example of this difference, flagged by diff and the corresponding lines is shown below in Table 5. Note that at this location, there is an equidistant next nearest neighbour which results in a different estimate.

Table 5: Section of difference summary file indicating difference between kt3d and kt3dn estimates when there are equidistant next nearest neighbours.

Line	Output – difference summary file		
1	879c879		
2	<	1.1016	0.2847
3	---		
4	>	1.1499	0.2847
Line	Corresponding section of kt3d output file		
	Est	Est Variance	
884	1.1015503	0.28470308	
Line	Corresponding section of kt3dn output file		
	Est	Est Variance	Equidistant NN?
888	1.1499461	0.28471848	1

***2DWellData.dat* Test Case**

In addition to the other test cases presented here, a test case using the GSLIB training data set, 2DWellData.dat, has been implemented. The results are similar to other test cases. A script has been configured to execute the test and output a difference summary file. All differences are only minor rounding differences, similar to the red.dat test case.

Conclusion

Although a large number of additions have been made to `kt3d` in the creation of `kt3dn`, backwards compatibility has been maintained to facilitate the use of the new features. Scripts and methods for recursive testing complete this latest implementation so that the user may check the results. The new parameters are detailed in the `kt3dn` parameter file. In summary, the new features which have been implemented are:

- Checking and removal of duplicates in a data set with a warning to the user
- Pre-sorting of the data along an arbitrary vector to preserve the uniqueness of kriging solutions with limited search data
- A new debugging framework capable of writing out a summary file and specifying how many kriging measures to calculate
- The calculation of kriging measures such as the number of data used per estimate, kriging efficiency, slope of regression and approximate global simple kriging variance
- Auto search optimization which performs kriging sequentially with a different maximum number of search data
- Ability to specify a maximum number of data per drillhole
- Collocated cokriging from Chad Neufeld's implementation in `kt3d_com2`
- Inverse distance estimation with the same search anisotropy and parameters available for kriging

References

Deutsch, C.V., 2007. The slope of regression for kriging estimators. Centre for Computational Geostatistics, 9:311.

Deutsch, C.V. and Journel, A.G., 1998. GSLIB: Geostatistical Software Library and User's Guide. Applied Geostatistics Series. Oxford University Press, New York, New York, 369 pp.

Deutsch, J.L. and Deutsch, C.V., 2010. Some Geostatistical Software Implementation Details. Centre for Computational Geostatistics, 12:412.

Deutsch, J.L. and Deutsch, C.V., 2012. Kriging, Stationarity and Optimal Estimation: Measures and Suggestions. Centre for Computational Geostatistics, 14:306.

Krige, D.G., 1997. A practical analysis of the effects of spatial structure and of data available and accessed, on conditional biases in ordinary kriging. Geostatistics Wollongong '96, Vols 1 and 2, 8(1-2): 799-810.

Scowen, R.S., 1965. Algorithm 271: quickersort. Communications of the ACM, 8(11): 669-670.

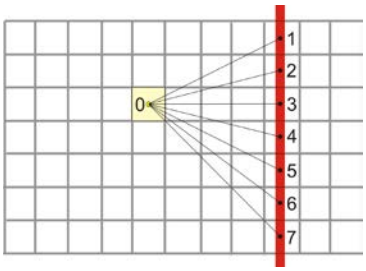

Figure 1: A sketch showing a common situation with equally spaced data along a drillhole oriented perpendicular to one of the model coordinates.

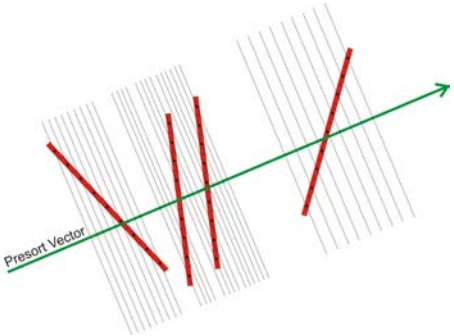

Figure 2: A sketch showing an example of presorting the data along an arbitrary vector.

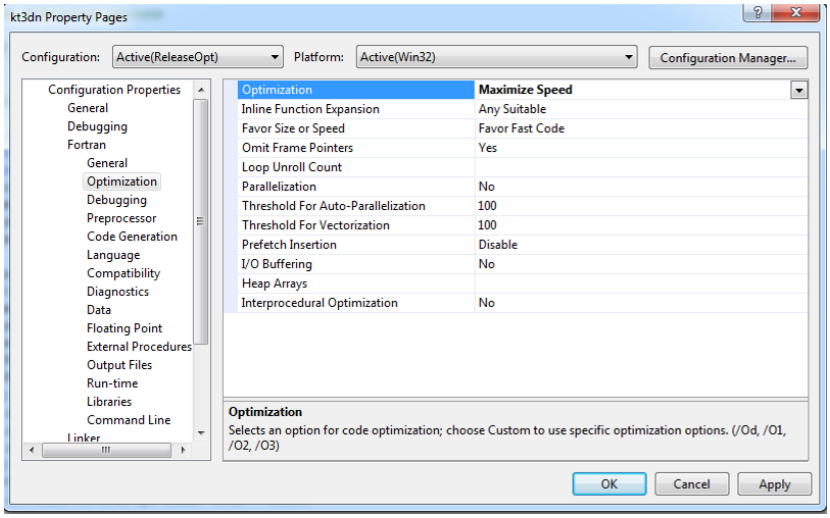

Figure 3: Screen capture showing Intel Fortran compiler optimization page.

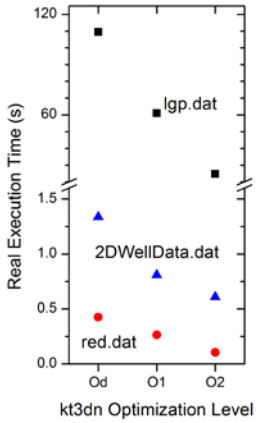

Figure 4: Speed of test cases for a variety of compiler optimization levels.