Components of the c-APK methodology

Pavan Mehta

Table of Contents

1. Introduction

This report introduces the main components of the c-APK methodology. The structure of the report is as follows:

- Section 2: A brief introduction of c-ANOVA, Krigging and POD is outlined.
- Section 3: User guide – The strategies involved using the supplied Python script along with this report.
- Section 4: Numerical investigations – Test case: Lid driven cavity.

2. Literature Review

2.1 Anchored ANOVA Decomposition

The ANOVA decomposition technique (Yang et al., 2012; Margheri & Sagaut, 2016) or High Dimensional Model Representation (Rabitz and Alış, 1999) is a functional decomposition of a high dimensional function in to series of subsequent lower dimensional functional evaluation (1) within a domain Ω. The first term on the right hand side or the zeroth order term, f_0 , is a constant throughout the domain. The second or first order term $f_i(x_i)$ describes the independent action of x_i in the entire domain; any non-linearities of the functions are addressed too (Rabitz and Alış, 1999).

$$f(x) = f_0 + \sum_{i \le N} f_i(x_i) + \sum_{i<j \le N} f_{ij}(x_i, x_j) + \ldots + \sum_{i<j<\ldots N} f_{ij\ldots}(x_i, x_j, \ldots, x_N) \quad (1)$$

The second order term or the third term in (1) describes the joint effect of x_i and x_j, this argument follows for all higher order terms. A special point to note about the last term $f_{ij\ldots n}(x_{ij..n})$, it contributes towards any residue arising from all terms considered together. For the integral function *f(x)*, can be evaluated either by Dirac or Lebesgue measure. The advantage of the former over the later: functional values can be evaluated at discrete points as per any sampling strategy or quadrature technique. In this work, the Dirac measure methodology is introduced. Popularly known as Anchored ANOVA decomposition or cut-HDMR in the literatures. For notational convenience we closely, follow Yang et al. (2012) and Margheri and Sagaut (2016).

The zeroth order term is evaluated at an “Anchor” point. An anchor point is a fixed point in space usually denoted as $c_1, c_2, c_3, \ldots, c_N$ in an N – dimensional space. These anchor points are generally taken at the center of computational domain. However, this choice is arbitrary and accuracy varies as per the choice of these anchor points (Zhang et al., 2011, 2012). The first and second order terms are computed as per equations (2) and (3) respectively. Here, q^i_1 and q^j_2 are the quadrature points in first and second dimensions respectively. The higher order terms can be computed in a similar fashion. From (3), it can be clearly seen; for computation of the second order terms, it requires first order term. Hence, computation of higher order terms can be expensive. Moreover, it is found that the majority of the variance is captured by the first and second order terms. Thereby, truncation of (1) after second order terms does not produce a larger error. In (4) ϵ_T is the truncation error.

$$\begin{aligned} f_1(q_1^i) &= f(q_1^i, c_2, c_3, \ldots, c_N) - f_0 \\ f_2(q_2^j) &= f(c_1, q_2^j, c_3, \ldots, c_N) - f_0 \\ &\vdots \end{aligned} \tag{2}$$

$$f_{1,2}(q_1^i, q_2^j) = f(q_1^i, q_2^j, c_3, \ldots, c_N) - f_1(q_1^i) - f_2(q_2^j) - f_0 \tag{3}$$
$$\vdots$$

$$f(x) = f_0 + \sum_{i \le N} f_i(x_i) + \sum_{i<j \le N} f_{ij}(x_i, x_j) + \epsilon_T \tag{4}$$

2.2 Proper Orthogonal Decomposition

Proper Orthogonal Decomposition has two other sister methods, “Karhuen-Loeve Expansion” and the “Principal Component Analysis” . The fundamental question is to find the pairs of orthogonal basis function where the data could be projected onto this new sub-space. This sub-space may not necessarily have the same dimensions as the original data itself. In Principal Component Analysis, this new set of orthogonal basis (also refereed to as principal components) are such that, the first principal component has the maximum variance, followed the second component which not only is orthogonal to the first but has next maximum variance of the data and so on. In order to numerically, evaluate the components we introduce the “method of snapshots”.

The Snapshot matrix, S is given by (5); where $f(x)$ is an N – dimensional function and are refereed to as snapshots for total number of p functional evaluation. Here, I_p is the identity matrix of shape, $p \; x \; p$ and 1_p is the matrix of ones of the same shape. Next, we compute the Singular value decomposition of this snapshot matrix (6). Each column of the matrix U is a principal component; there is no advantage in keeping all the p components. Hence, we truncate it to a finite order r and the orthogonal basis vectors Ψ are given by (7)

$$S = [f\ (x_i)\ ,\ \dots\ ,\ f\ (x_p)]\ [I_p - (1/p)\ 1_p] \tag{5}$$

$$S = U\ \Sigma\ V^T \tag{6}$$

$$\begin{aligned} \psi &= U_r\ \Sigma_r\ , \\ \Sigma_p &= diag\ ([\sigma_1^2 > \sigma_2^2 > \dots > \sigma_p^2]) \end{aligned} \tag{7}$$

In Karhuen – Loeve expansion the solution is projected over an orthogonal basis and expanded as per (8). The main issue was in addressing the computation of these orthogonal vectors Ψ which is as per the earlier discussion.

$$f^s\ (x) = \sum_{m=1}^{r} \nu_m^s\ \psi_m \tag{8}$$

Furthermore, the coefficients β in (9) are computed using krigging of the coefficients v^s for a solution at any non-sampled location. However, for the purposes of this study and c-APK algorithm, equation (9) is not used. POD is only used for smoothing noisy Quantity of Interest using (8). For computing the solution for any non-sampled location, Krigging is directly applied, detailed in the next section.

$$f\ (x) \simeq \sum_{m=1}^{r} \beta_m\ \psi_m + \epsilon_r \tag{9}$$

Finally, to find the coefficients v^s_m in (8), the correlation matrix, C is computed from the snapshot matrix, S as per (10). The coefficient v^s_m is the s-th component of the m-th right eigenvector.

$$C = \frac{1}{p}\ S\ S^T \tag{10}$$

2.3 Krigging

Krigging or Gaussian Process Regression evolves from multivariate Gaussian theory. The term "krigging" is coined after the South African mining engineer, Dane Krige. The definition for a Gaussian process as stated in Soto et al, (2011); "*A Gaussian Process is a collection of random variables, any finite number of which have (consistent) joint Gaussian distributions*.". Formerly written in (11), where the process is fully defined by its mean (μ) and the covariance matrix (Σ). It is to be noted that a Gaussian distribution is over vectors while, while a Gaussian process is over functions.

$$f \sim N\ (\mu\ ,\ \Sigma) \tag{11}$$

The covariance matrix is used to generate functional values at any non-sampled location. In (12), *k*, is a covariance function also refereed to as "*Kernel*" in Machine Learning terminology. It is solely depended on the corresponding distances between two sample points, implying that the function's output value is heavily depended on its neighboring samples. It is obvious now, that relative accuracy in interpolation is better than extrapolation. Also, the results may vary with the choice of Kernel. Hence, a good Kernel approximation is required. Loosely speaking, the choice of the Kernel can be based on the type of fitting required. For example, if its in our prior knowledge that the system behaves in a quadratic sense, a choice of Kernel resembling the quadratic nature would be wiser to use. We introduce two Kernel functions here, the "Square Exponential" as given in Soto et al., (2011) and "Polynomial Cubic Spline" kernel as per Margheri and Sagaut, (2016) in (13) and (14) respectively. In (14) θ is known as hyper-parameter and its values is taken to be 0.1.

$$\Sigma_{ij} = k\ (x_i\ ,\ x_j) \tag{12}$$

$$k\ (x_i\ ,\ x_j) = \exp\ (-\frac{1}{2}\ (x_i - x_j)^2) \tag{13}$$

$$\begin{aligned} k\ (x_i\ ,\ x_j) &= 1 - (6\ |x_i - x_j|\ \theta)^2 + (6\ |x_i - x_j|\ \theta)^3\ ;\quad \forall\ |x_i - x_j| < \frac{1}{(2\ \theta)} \\ k\ (x_i\ ,\ x_j) &= 2\ (1 - |x_i - x_j|\ \theta)^3\ ;\quad \forall\ \frac{1}{(2\ \theta)} \leq |x_i - x_j| < \frac{1}{(\theta)} \\ k\ (x_i\ ,\ x_j) &= 0\ ;\qquad \forall\ |x_i - x_j| \geq \frac{1}{(\theta)} \end{aligned} \tag{14}$$

3. User Guide

This section introduces the strategies involved in using the Python scripts. The scripts of ANOVA decomposition, Krigging and POD are discussed before getting into the Software Architecture . Primarily because, these scripts form the basis of the c-APK method and can be used individually too. At first we introduce index mapping for tracking the data.

3.1 Index Mapping

For tracking data corresponding to a particular solution or a value we introduce a technique of “index mapping”. From, Figure 1, two sets in a form of array is used, one for the real data and the other for the indice’s. The need for an index array arises from its usage in calling the a particular solution automatically, given that python indexing system uses integers only.

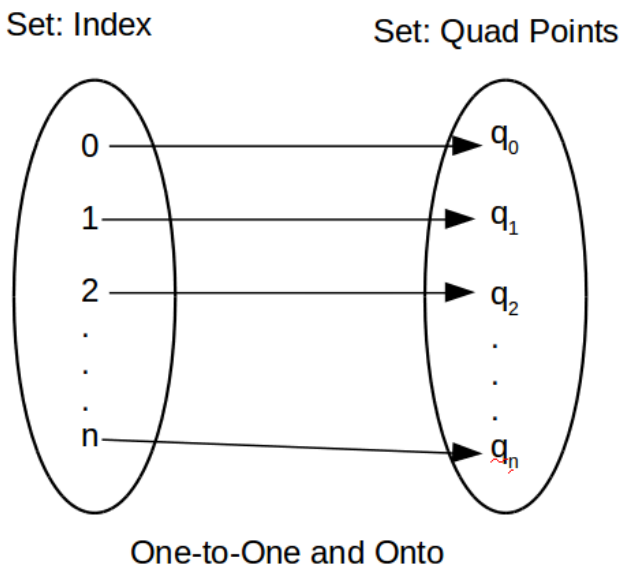


Figure 1: Index Mapping

The shape of both the array is (*data sets x N*), where *N* is the total number of dimensions. For example, consider the dimension of the problem is 4 and the number of quadrature points = 2 per dimension. Moreover, let the number of data sets be all possible combinations of these quadrature points. Hence, the total number of data sets = 4^2 = 16. Specifically for anova decomposition or otherwise to maintain generality, we start incrementing the last index first, followed by the second last index and on on. This discussion can be summarized visually in Figure 2. Here, the quadrature points in dimensions 1, 2, 3, 4 are [0.25, 0.75], [-0.25, -0.75], [0.25, 0.75], [-0.25, -0.75]

respectively. The row numbers are an interesting quantity, by calling the particular row number the corresponding solution to the quadrature points can be called.

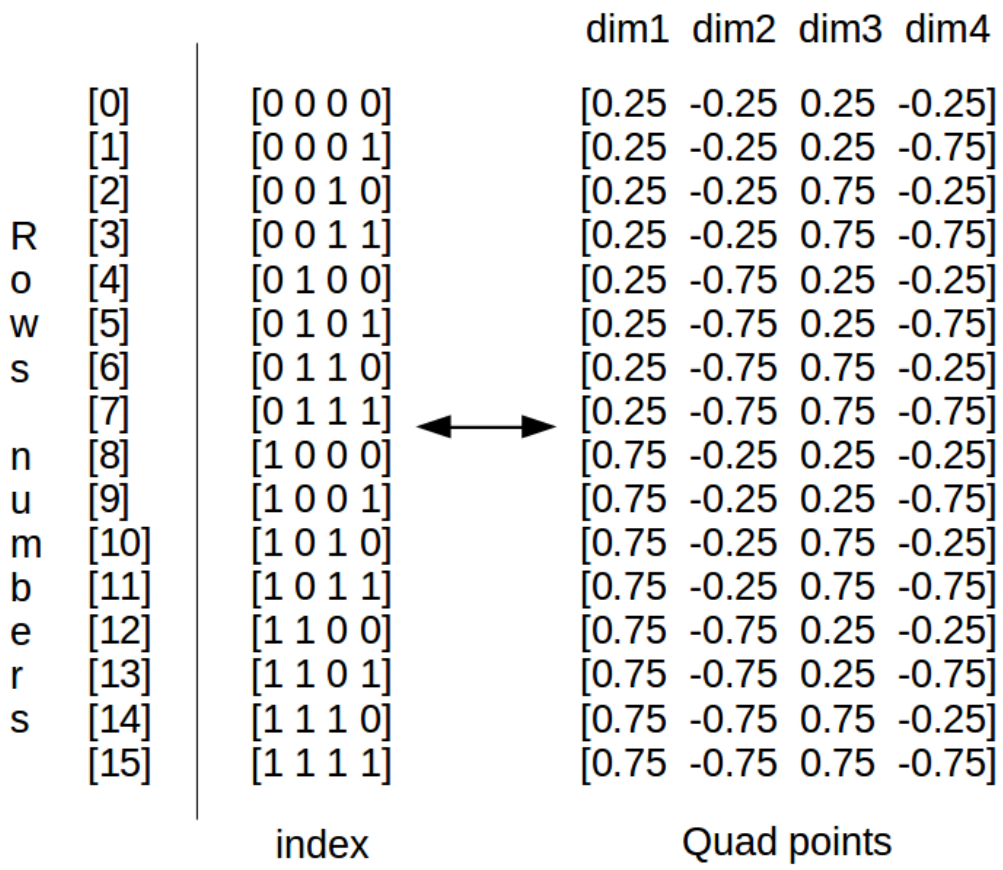


Figure 2: Index mapping example

3.2 ANOVA Decomposition

The python script “anova.py” performs the anchored ANOVA decomposition in 4 dimension with truncation dimension = 2. The important variables along with usage is commented in the file itself. Here, the usage of the script, data input and output for this file is illustrated. As an input, the mean, 1st order and 2nd order quadrature terms are provided, while the output consists of results from anova decomposition and indices's corresponding to each result (refer Index Mapping).

For a given experiment the anchored point is per-selected and data is generated on the hyper-lines and hyper-planes with respect to the anchored point. For this reason, supplying an anchored point along with the data would not make much sense. As data may not be available for corresponding to this random selection of this anchor point. Hence, the mean, 1st order and 2nd order terms are directly supplied. The strategy is as follows with reference to Figure 3 in 4 dimensions:

- For the mean or anchored point term: Supply the data as the 1 dimensional array.
- 1st order experimental data: In form of three dimensional array. Along dimension 1 of this array supply the data obtained form a single simulation or experiment for the corresponding term and quadrature point. Dimension 2 of this array: increment the quadrature points for the corresponding term. Dimension 3 of this array: number of first order terms. As an example: at location [10, 1, 2] has the 11th CFD grid point data is stored for $f_3(c, q_1, c, c)$. Notice that, in python indices's begin with zero. Shape (*net grid pts x net quad pts x net 1st order terms*)
- 2nd order experimental data: Similar strategy as per 1st order terms, with only difference in 3rd dimension instead of 1st order terms, we have 2nd order terms in a chronological fashion.

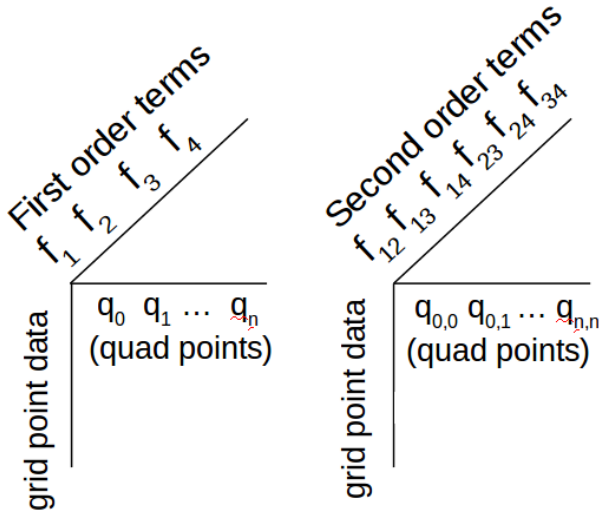


Figure 3: Array for 1st and 2nd order terms (Input)

The output c-ANOVA decomposition is a two dimensional array. With reference to Figure 4, each column is an individual solution where along the 1st dimension of this output array we have grid points data. By calling a particular row number the corresponding solution can be called, refer

Figure 2. It is to be noted that this output strategy is being used for all the input and outputs in subsequent sections, unless specifically mentioned.

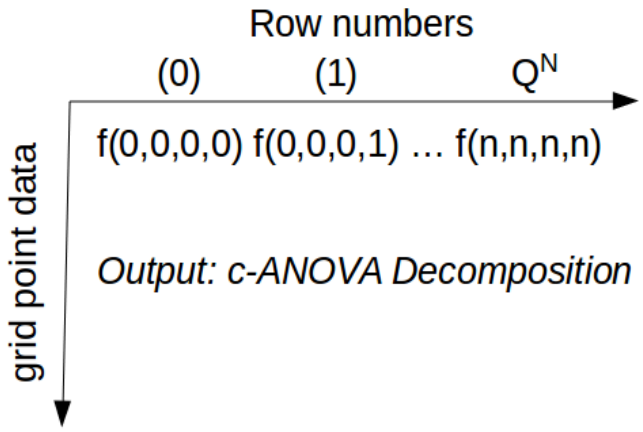


Figure 4: c-ANOVA decomposition output array

3.3 POD and Krigging

Proper Orthogonal Decomposition is performed by the script, “pod.py”, while krigging is done using “krigging.py”. As mentioned earlier the input and output for both of these techniques follow the discussion of ANOVA output (refer Figure 4). Additional flexibility is provided; in POD, the technique is robust enough to find the orthogonal basis (refer (7)) for any random data, however it is recommended for the data to be in the form described in Figure 4. It is to be noted that the output is as per the input. For example, if we have a different indexing strategy, the output would now correspond to this new indexing system. Nonetheless, it still needs to be an array in 2 dimensions only.

While, for krigging; the flexibility provided algorithmically than arising mathematically. Here, as long as the indice’s array is supplied along with the data, the python script will identify and perform krigging accordingly. However, the indexing system along with the data needs to be in the form described in the earlier sections. We now introduce, how krigging is being performed from 1 dimension to 2 dimensions, and this discussion can be extend to N – dimensions.

Consider an example; there are two quadrature points in two dimensions each, let (0.25, 0.75) and (-0.25, -0.75) be those quadrature points in dimension 1 and 2 respectively. Now, we require the krigging interpolated data at the midpoint of these quadrature points in each dimension. Hence, our *discretization samples* = 3. In general, (*discretization samples = number of required point data + 2)* for each dimension. Referring Figure 5, we start krigging in the first dimension. The interpolation

is carried out for the point (0.5, -0.25), here we need the data at (0.25, -0.25) and (0.75, -0.25); then for (0.5, -0.75), here the required data is at (0.25, -0.75) and (0.75, -0.75) and so on.

In general, we fix all the quadrature points in all dimensions except the dimension where the interpolation is happening. For example, consider an *N*-dimensional case, where the interpolation is happening at dimension 1 then, (discretization, c1,c2, .. ,cN) would require data from (all quad points, c1, c2, ..,cN), here (c1, … cN) are points fixed in space.

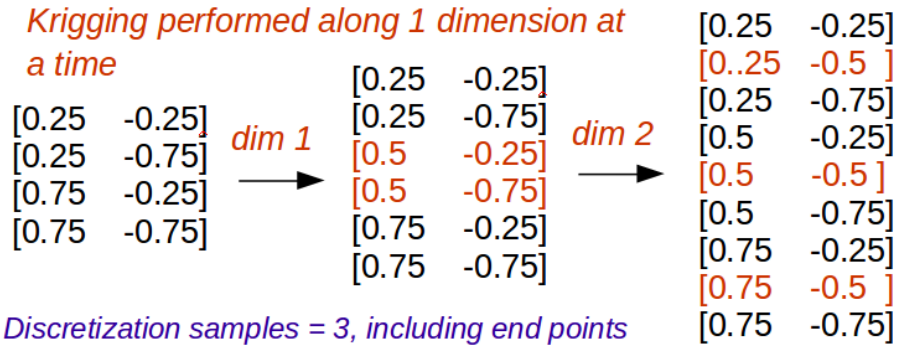


Figure 5: Krigging: Interpolating data in each dimension at a time

It is to be noted that Kernel is only a feature of Krigging. Also, by default it takes uniform samples, unless samples are provided (refer Sampling)

3.4 Software Architecture

In the previous sections; the main files input – output functionality was explained. In this section, the software architecture is layed-out. It is to be noted, due to the objected oriented nature of all these scripts; any script or method can be called, in any order. From Figure 6, the “cAPK – main” python script is the control file. The experimental data for performing an UQ study is built in the “case.py” script as per the above discussions. This script is then called by the main script and supplied to specific UQ – technique as required in the study. Data from one methodology can be supplied to any other methodology as long as the indices's are provided as well (refer Index Mapping).

It is to be noted that for Krigging, by default it takes uniform sampling, generated by “samples.py”, unless the samples are provided from any other sampling strategy. In order to generate “sobol

sequence", the "samples.py" is needed to be called. The samples now generated using this sampling strategy can be fed into the anyother UQ methodology, where-ever applicable.

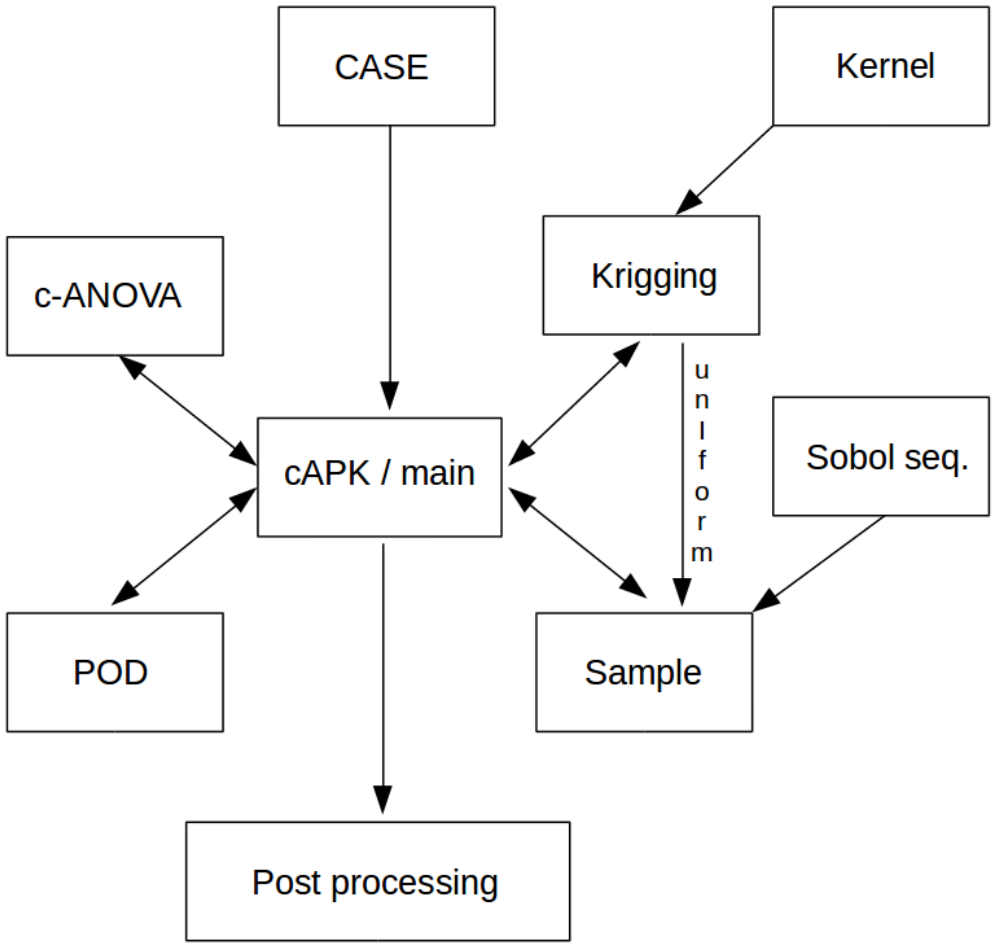


Figure 6: Software Architecture

3.5 Miscellaneous

3.5.1 Sampling and Sobol sequence generator

The "sample.py" generates samples as per uniform or sobol sequence. In the uniform strategy, the end points are also returned. The "sobol.py" is used to generate samples using sobol sequence. It is recommended to add additional sampling strategy in this file.

3.5.2 Kernel

In this UQ study kernel is only a feature of krigging. However, due to object orientated nature of the script, it can be called any other script. In he "kernel.py" python script, currently only two kernels are implemented. Namely, the "square expectational" and "polynomial cubic splines" as per equation (13) and (14) respectively. Again, it is recommended to add additional kernels in this script. Moreover, python's "sckit-learn" library has additional kernels to use.

4. Numerical Investigations

For our numerical investigation a case of a Lid Driven cavity with 4 uncertain dimensions as outlined in Figure 7. For carrying out this test we have u1 = [0.25, 0.75], u2 = [-0.25, -0.75], v1 = [0.25, 0.75] and v2 = [-0.25, -0.75].

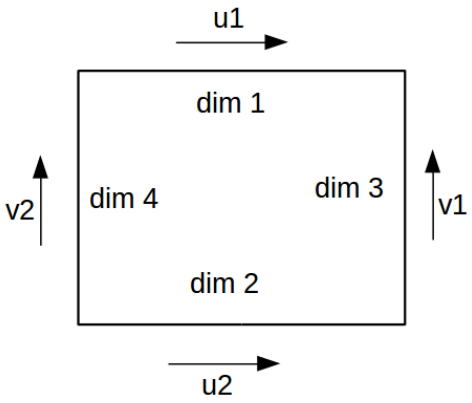


Figure 7: Lid Driven cavity with 4 uncertain dimensions.

4.1 ANOVA Decomposition

The anchored ANOVA decomposition is carried out in 4 dimensions with the truncation dimension (v) = 2. From Figure A.1, qualitatively speaking the results are well converged for both ANOVA decomposition with truncation dimension v = 1, 2; with $v = 2$ performing better compared to $v = 1$. From literature it is well known fact the v = 1 captures the maximum variance in the data, hence, our convergence of $v = 1$ result is well justified. However, the interesting part was to look at the quantitative data. In order to determine that the error from ANOVA decomposition when compared to CFD results, the absolute difference between the two was computed as per (15).

$$(ANOVA - CFD \;\; Error) = |f\;(anova) - f\;(cfd)| \tag{15}$$

It was found that certain low velocity regions did not converge well, while the regions having higher values did. Initial investigation was about Python's capability for handling low or near zero value; as the default data type in Python is double and after algebraic operations, near zero values are represented as (+/-) 1e-8. In this effect a Zero handler method (not shown here) was created; where detection of such values it would force it take reasonably low value. However, the relative error did not change much, as both the ANOVA and CFD result would scale itself to produce the same difference. On the contrary, (generality speaking) this behavior of Python provides a fail safe

algorithm, where division by zero would not crash the computation and having 1e-8 value would supply a very negligible error to the over all solution. Hence, the we eliminate the algorithmic possibility of Python contributing towards the non-convergent regions. This leaves us with two other possibilities:

- Bad choice of anchor point
- High variance in the data – implying the need for higher order terms.

Either way, if high variance is the reason, it should be well reflected in our present data. First, the mean deviation of CFD results was determined as per (16).

$$Variance = (f\ (cfd) - f\ (anchor\ \ points))^2 \tag{16}$$

The assumption of high variance was true in the regions of highest error. Contrastingly, the variance around the boundary was higher too, which is a well converged region. This convergence along the boundary region can be attributed to well converged c-ANOVA results with 1st order terms as seen in Figure A.1. Implying an higher variance in second order terms. Hence, an estimator for computation has been formulated inspired by Sobol's work given in (17)

$$(Variance\ order\ 2) = \sum_{i<j}^{N} f_{i,\ j}^2 \tag{17}$$

The results of (17) shows the highest variance in the regions of highest error (Figure A.1). The assumption of variance based error can be now be concluded in the second order terms. Finally, to determine the relative error in ANOVA decomposition the use of (15) is not feasible in all cases, owing to the fact of unavailability of high dimensional CFD simulation data. An additional estimator was formulated based on c-ANOVA decomposition only, given in (18), where f' is the ANOVA decomposition results with truncation dimension = 1, while f'' is the results with truncation dimension = 2.

$$(ANOVA\ Error\ Estimator) = |f' - f''| \tag{18}$$

From Figure A.1, there is a close resemblance of this new estimator with (15), owing to the fact the ANOVA decomposition reasonably converged with truncation dimension = 1. Furthermore, this new estimator is not limited to this case only, as literature shows us that maximum variance in the data is captured by 1st order terms, which is our case too. In conclusion, interpretation from variance

order 2 (17) and ANOVA Error estimator (18) can help us identify regions of high error without the need for additional CFD computations.

4.2 Krigging

At first the effect of two different kernel ((13) and (14)) is investigated for two different equations. From Figure 8, the square exponential kernel is not able to capture the quadratic equation ($y = x\hat{}2$), while the "Polynomial Cubic Spline" kernel does. Furthermore, we look at the function ($y = x*cos(x)$) with extrapolation. As expected, in the extrapolated region the variance or the error is too high. At the interpolated regions the square exponential kernel behaves poorly too with a high variance.

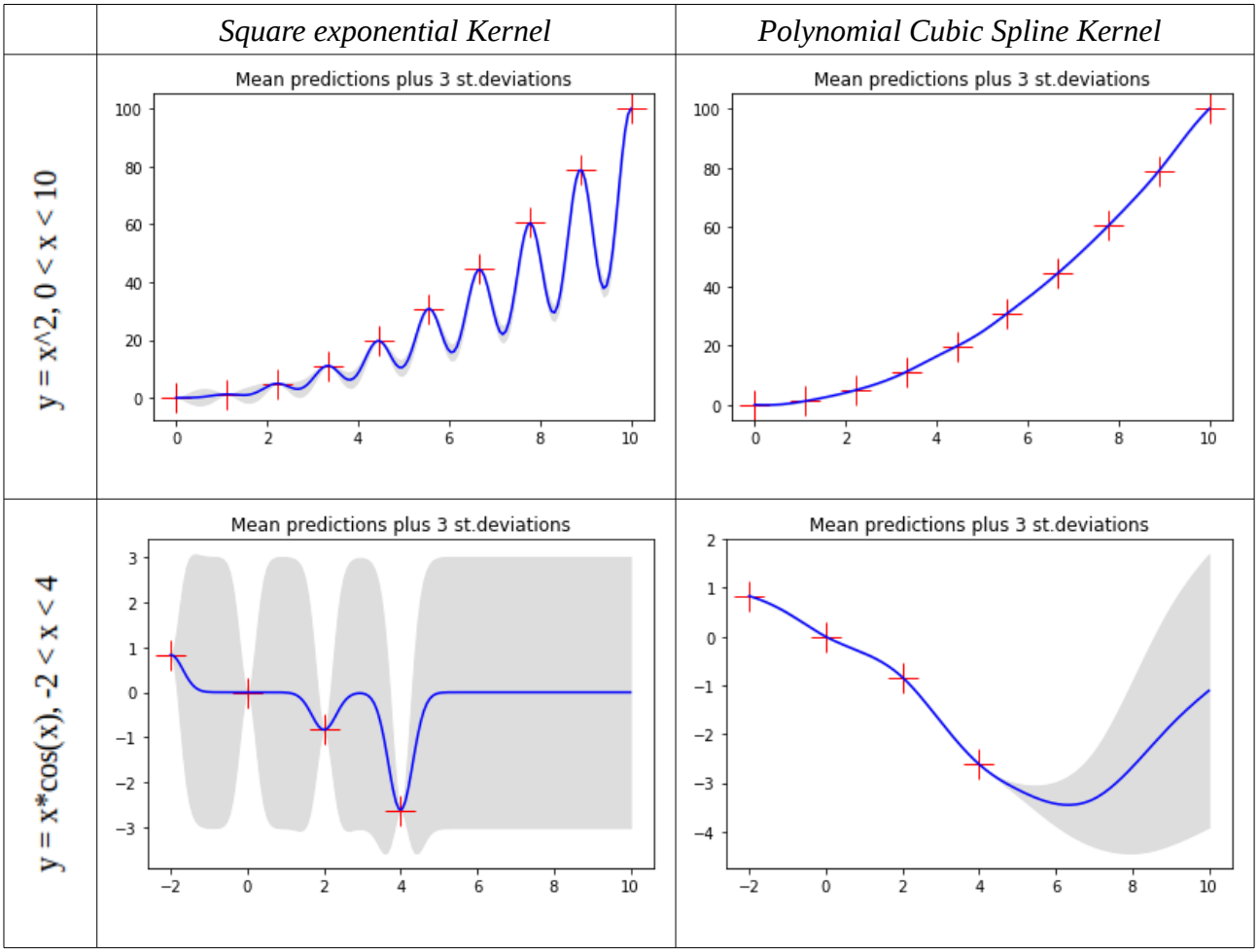
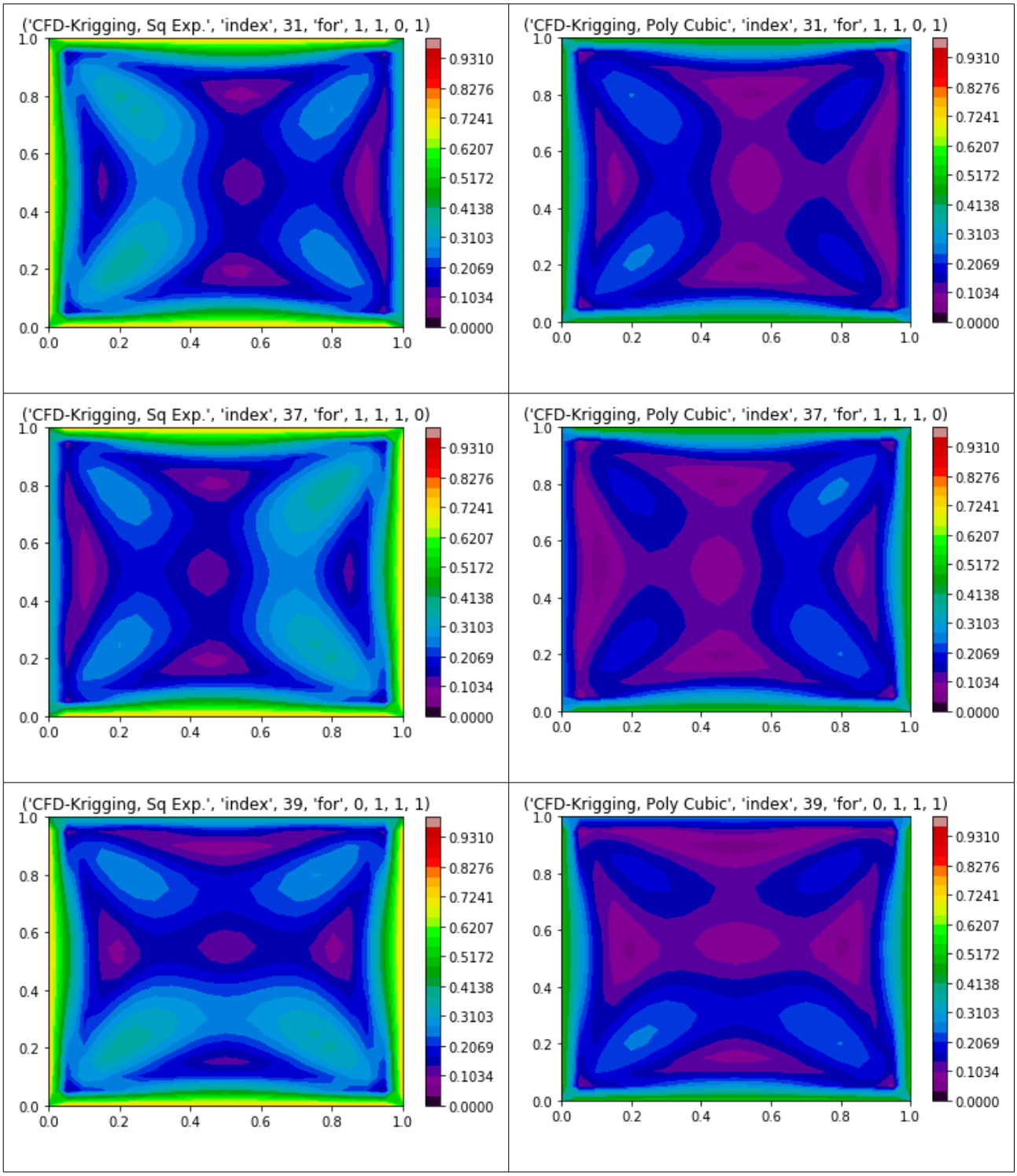


Figure 8: Results from two different Kernels ((13) and (14))

Furthermore, investigation of krigging applied to CFD simulations are carried out in determining the relative error of both of these kernel. As expected the error in any given individual solution is uniform and found to be in the range of [0%, 40%]. This high error can be attributed to poor

performance of the “square exponential” kernel. Selected results of krigging applied to CFD are given in Figure 9, where the discretization sample = 3 (refer Figure 5 for index map). Qualitatively speaking, the results from Krigging using “polynomial cubic spline” kernel are very close to actual case, while the “square exponential” kernel are overestimated.



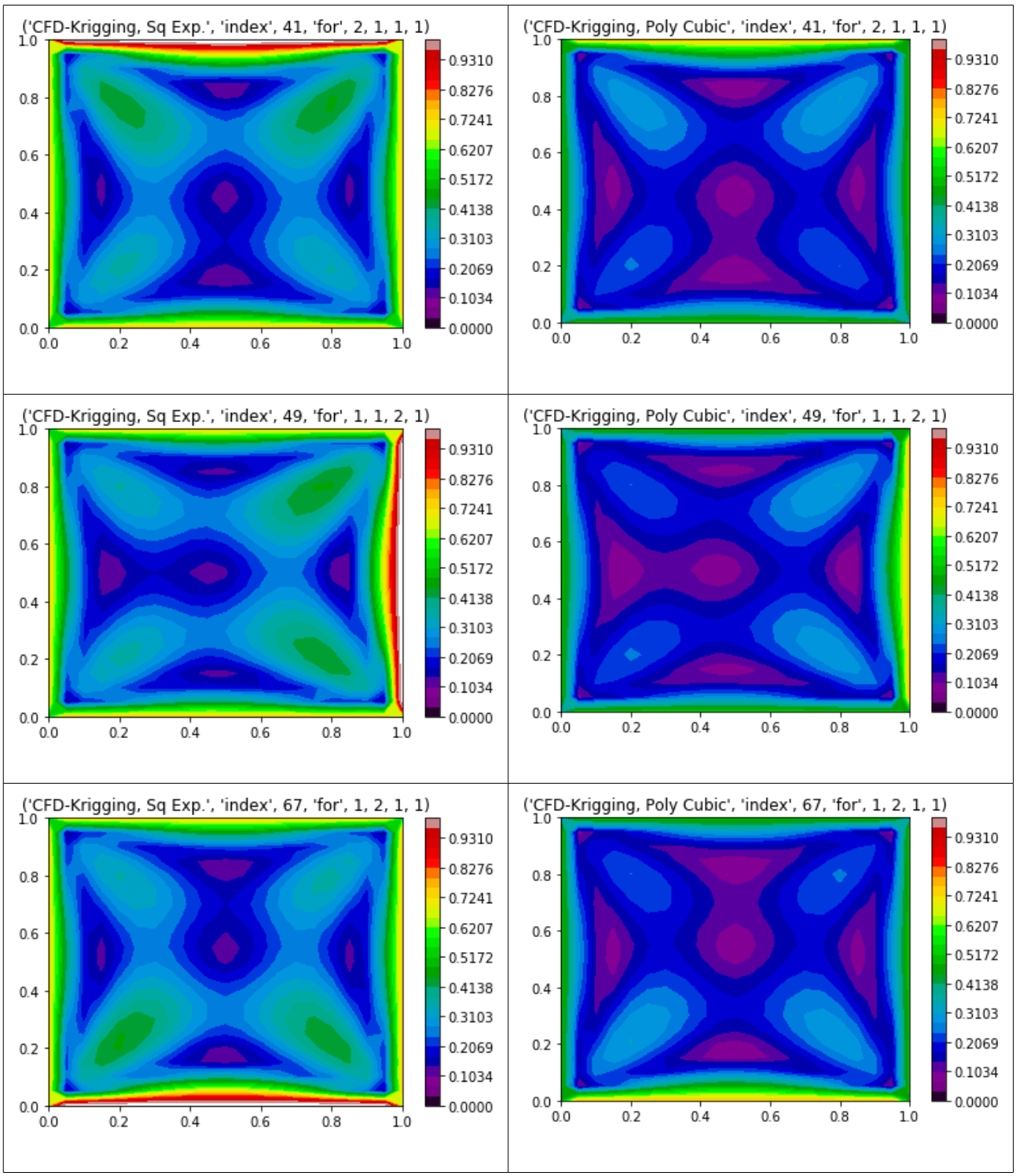


Figure 9: Selected results from Krigging applied to CFD results with discretization sample = 3 (total number of results = 3^4 = 81), obtained using two kernels (refer (13) and (14))

For the reasons outlined above (with reference to Figure 8 and Figure 9). The “polynomial cubic spline” kernel would was used for our further study. In Figure B.1, krigging was applied to CFD and ANOVA ($v = 2$), qualitatively shows great similarity. An interesting pattern was observed when computing their absolute difference. This error arises from ANOVA decomposition, where a similar error paterns are observed (refer Figure A.1).

4.3 Proper Orthogonal Decomposition

The results of POD were highly erroneous. The failure of POD is due to un-converged correlation matrix. In this effect upon increasing the samples for POD from Krigging was investigated. However, there has been an error reduction upon increase in samples but very slowly and required very high number of samples, as seen Figure 10. Many of the POD results had much higher error than aforementioned figure (not shown here).

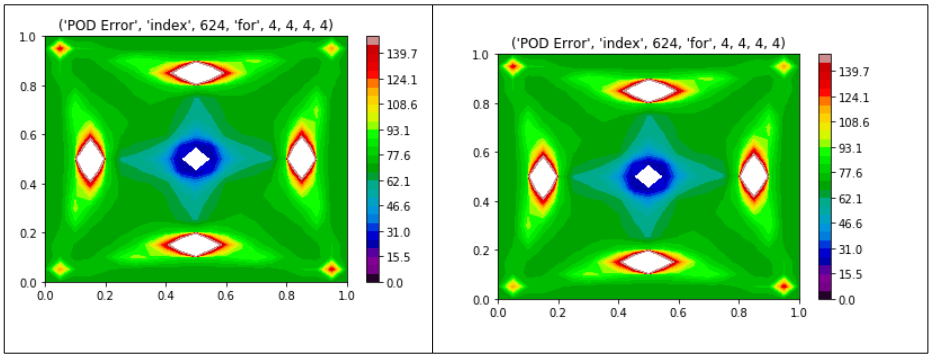
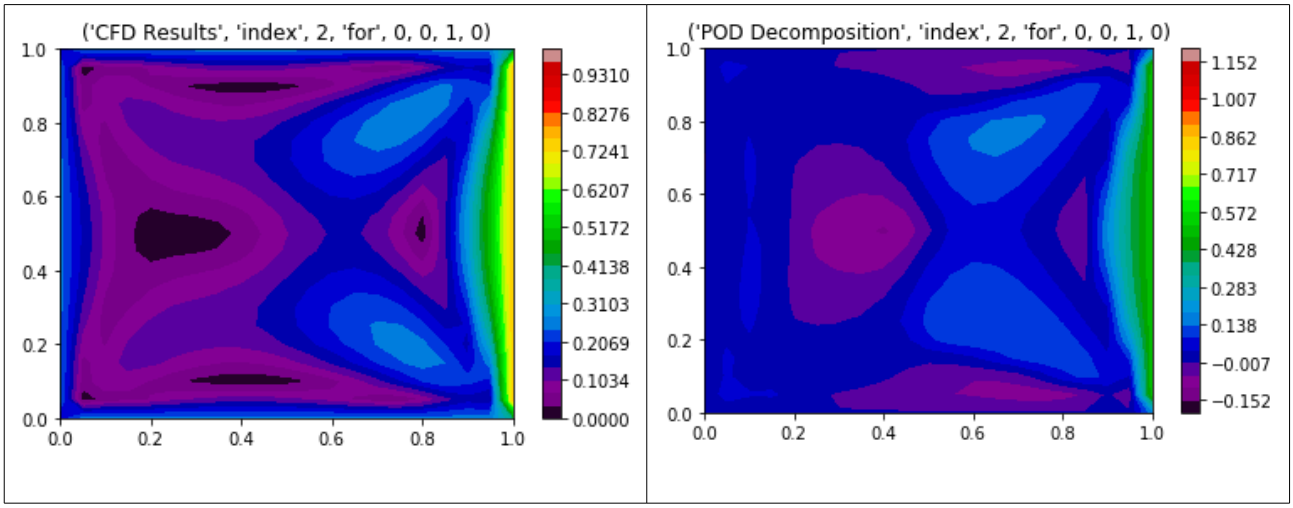
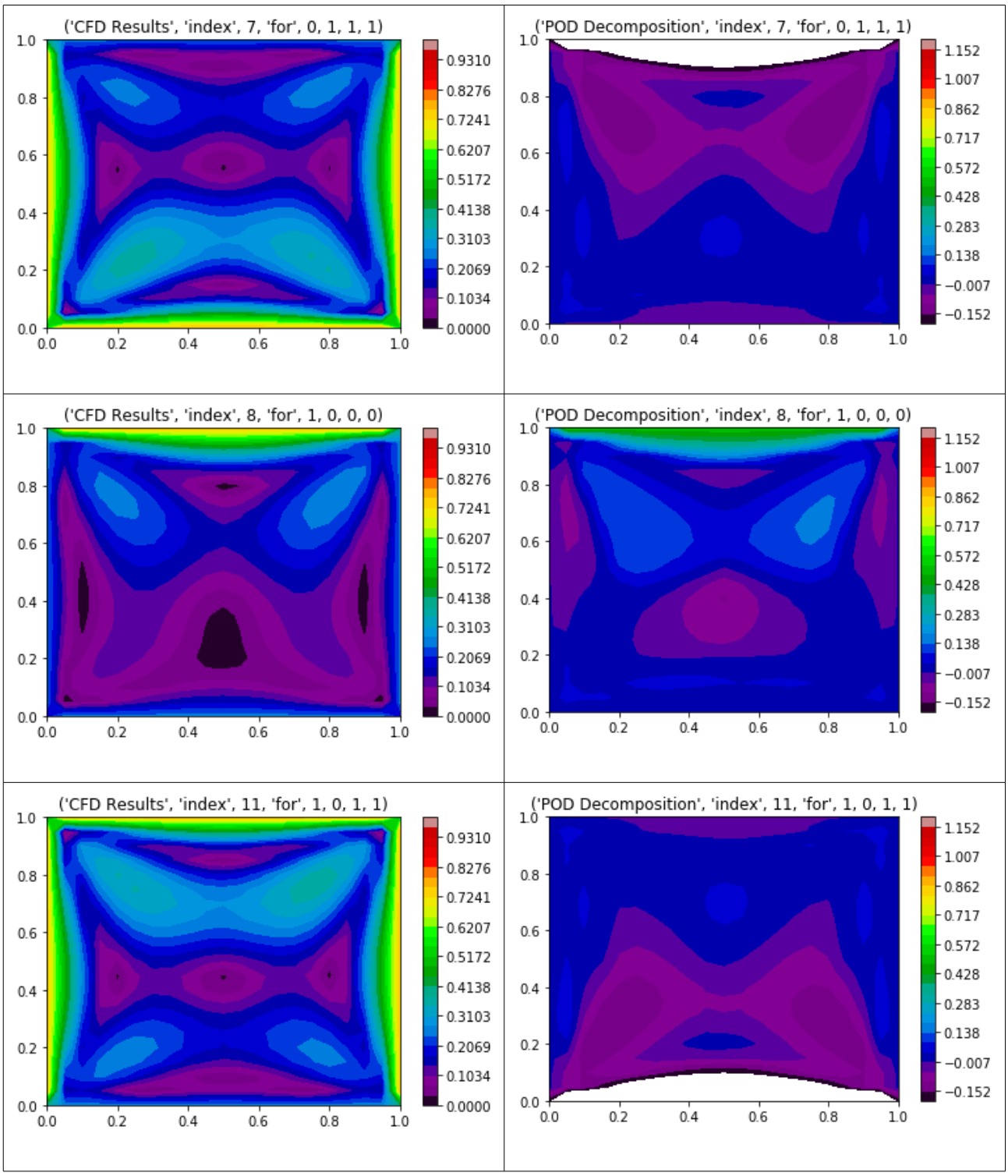


Figure 10: Relative error of POD, normalized using krigging data,

(from left to right) total number of samples = 625 and 6561.

Upon a closer look at the results, it was observed, POD averaged out the solution and identified the regions of high velocity (Figure 11). This observation is consistent with many literatures where POD is primarily used to identify structures or regions of high kinetic energy in a given flow. The last result in Figure 11 is the at the same sampling point as Figure 10, which is [0.75, -0.75, 0.75, -0.75].




('CFD Results', 'index', 7, 'for', 0, 1, 1, 1)
('POD Decomposition', 'index', 7, 'for', 0, 1, 1, 1)
('CFD Results', 'index', 8, 'for', 1, 0, 0, 0)
('POD Decomposition', 'index', 8, 'for', 1, 0, 0, 0)
('CFD Results', 'index', 11, 'for', 1, 0, 1, 1)
('POD Decomposition', 'index', 11, 'for', 1, 0, 1, 1)
0.0 0.2 0.4 0.6 0.8 1.0
0.9310 0.8276 0.7241 0.6207 0.5172 0.4138 0.3103 0.2069 0.1034 0.0000
1.152 1.007 0.862 0.717 0.572 0.428 0.283 0.138 −0.007 −0.152

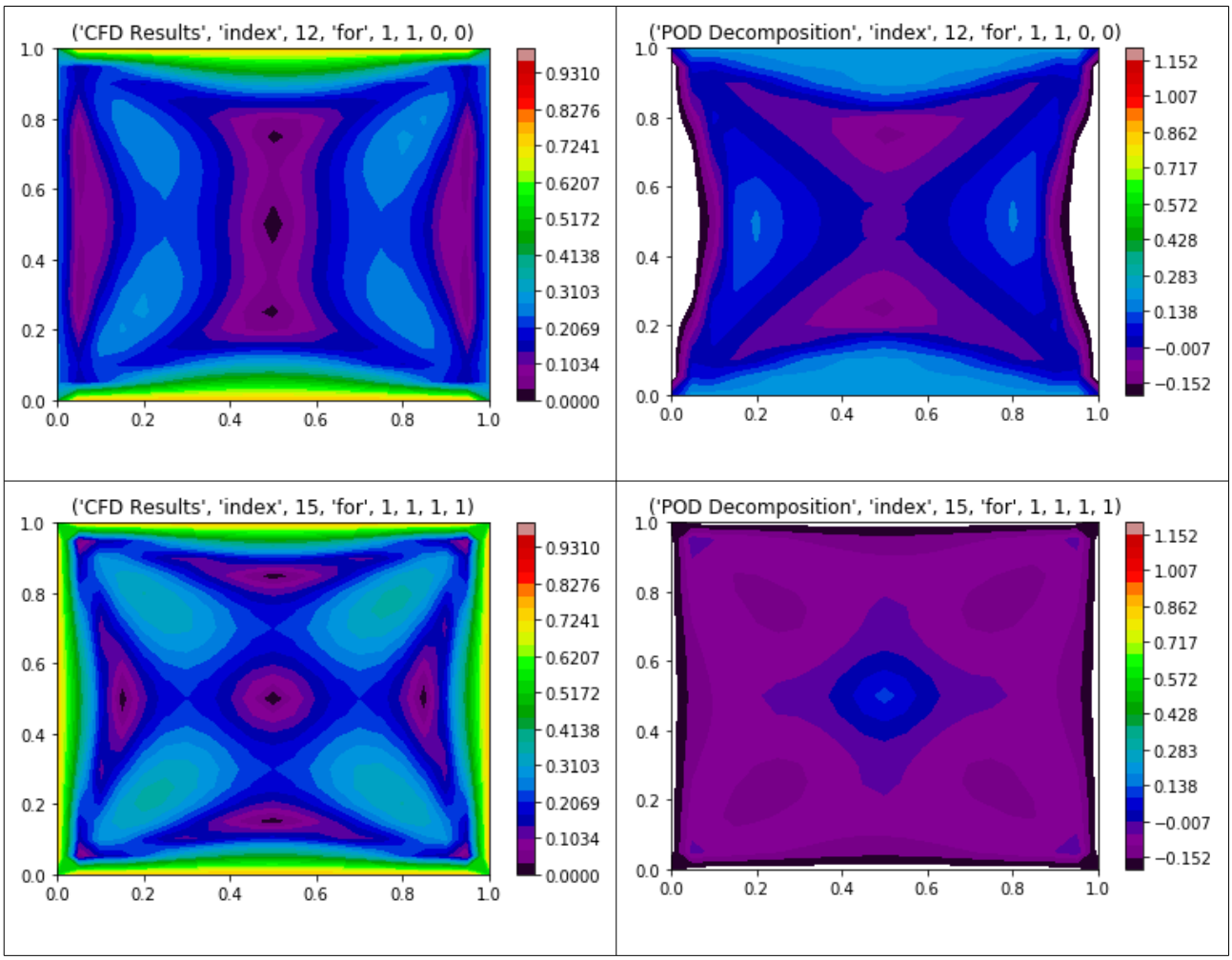


Figure 11: CFD results and their POD on right

Bibliography

Margheri, L., & Sagaut, P. (2016). A hybrid anchored-ANOVA – POD/Kriging method for uncertainty quantification in unsteady high-fidelity CFD simulations. *Journal of Computational Physics*, *324*, 137–173. https://doi.org/10.1016/j.jcp.2016.07.036

Rabitz, H., Alis, Ö., & Alış, Ö. F. (1999). General foundations of high-dimensional model representations. *J. Math. Chem.*, *25*(2–3), 197–233. https://doi.org/10.1023/A:1019188517934

Soto, A. J., Strickert, M., Vazquez, G. E., & Milios, E. (2011). *Lecture Notes in Artificial Intelligence* (Vol. 6657).

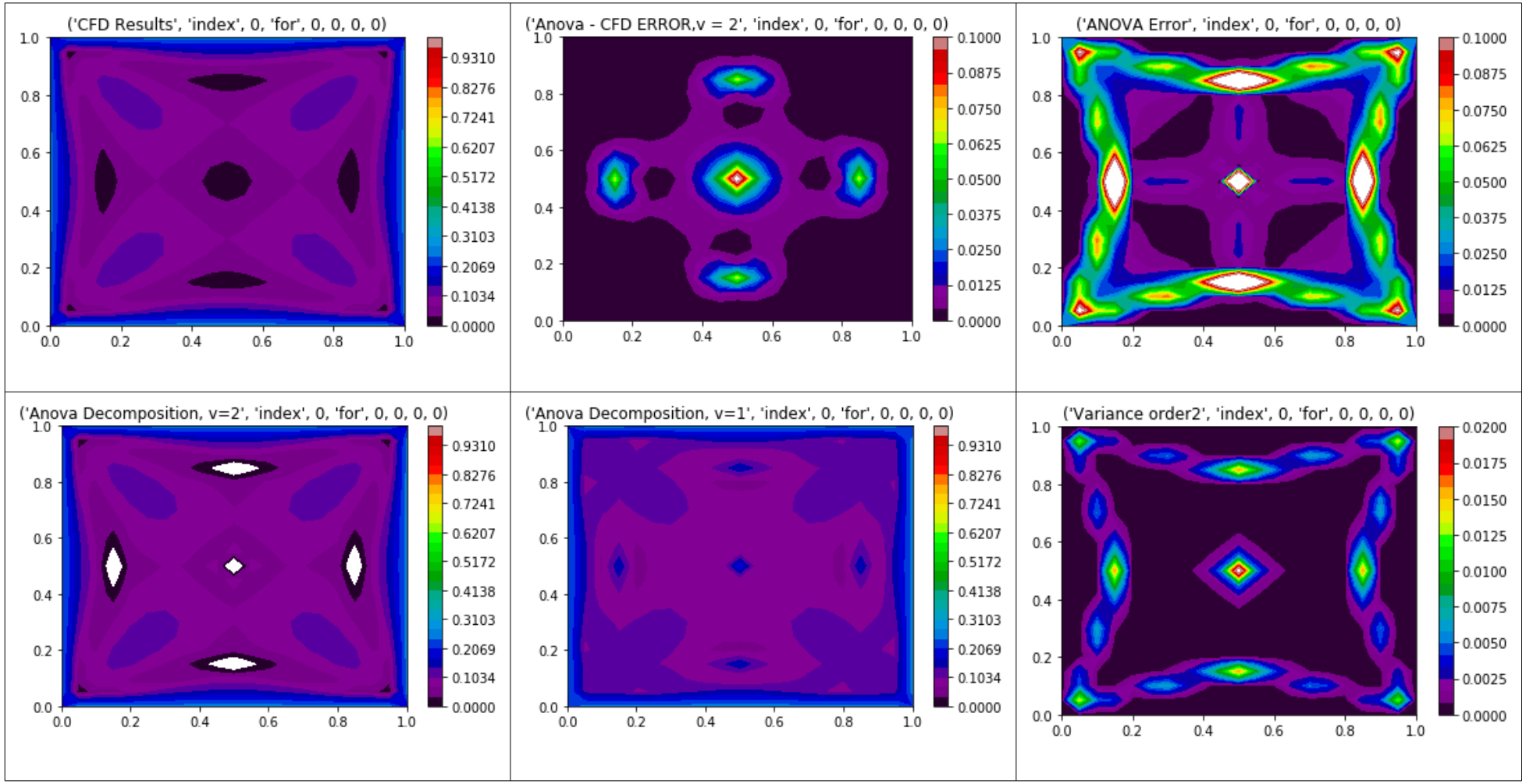
Yang, X., Choi, M., Lin, G., & Karniadakis, G. E. (2012). Adaptive ANOVA decomposition of stochastic incompressible and compressible flows. *Journal of Computational Physics*, *231*(4), 1587–1614. https://doi.org/10.1016/j.jcp.2011.10.028

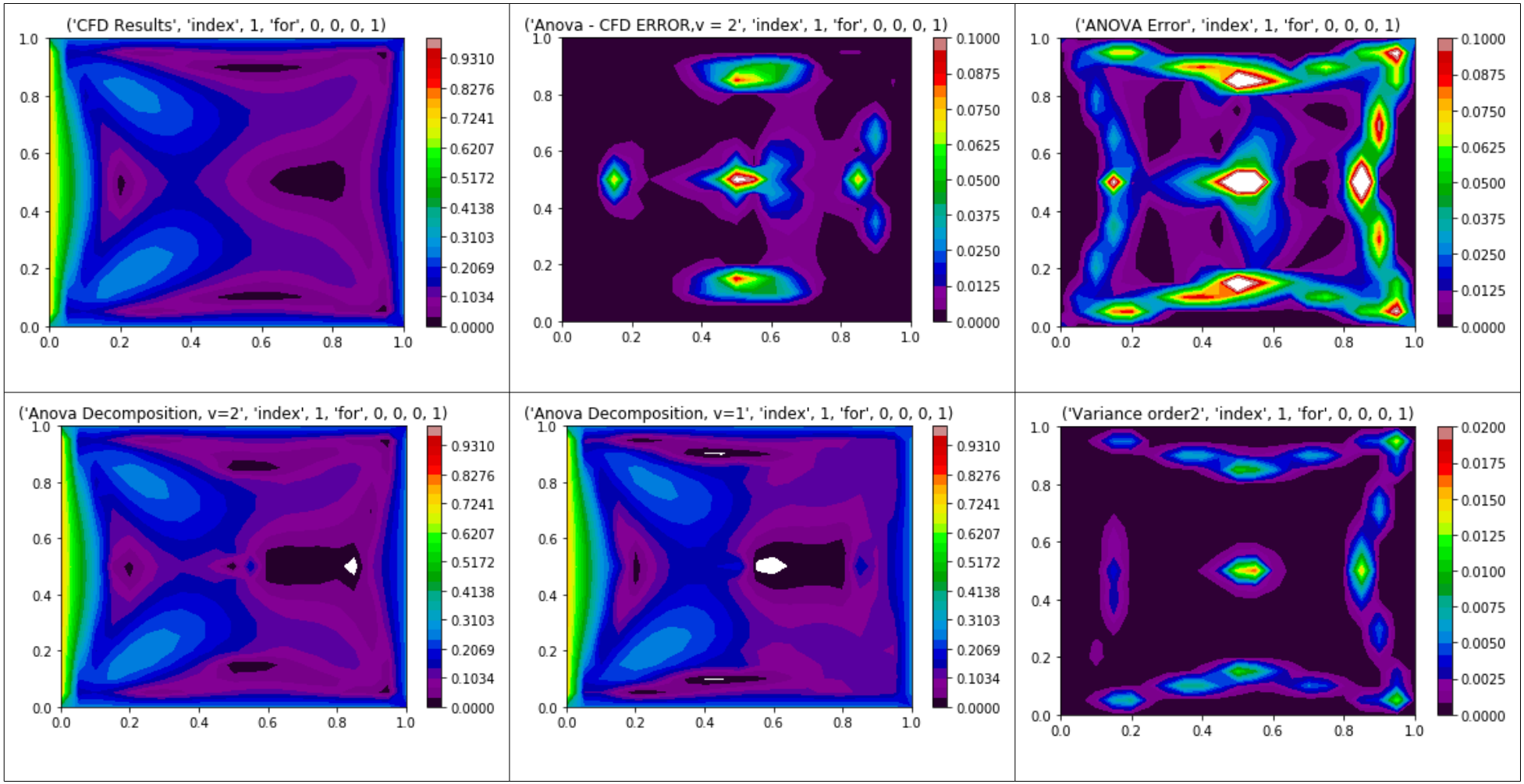
Zhang, Z., Choi, M., & Karniadakis, G. E. (2011). Anchor points matter in ANOVA decomposition. In *Lecture Notes in Computational Science and Engineering*. https://doi.org/10.1007/978-3-642-15337-2_32

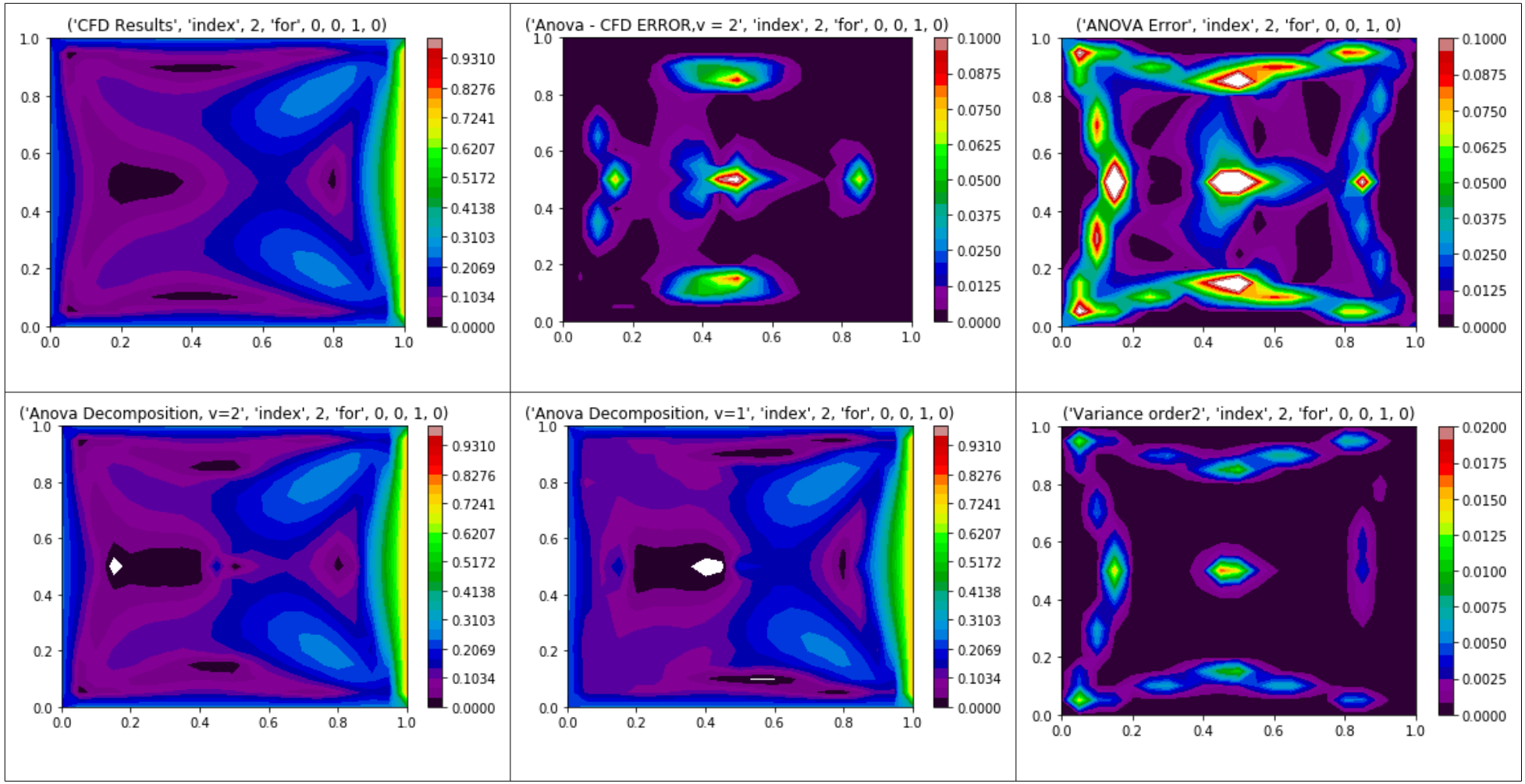
Zhang, Z., Choi, M., & Karniadakis, G. E. (2012). Error Estimates for the ANOVA Method with Polynomial Chaos Interpolation: Tensor Product Functions. *SIAM Journal on Scientific Computing*. https://doi.org/10.1137/100788859

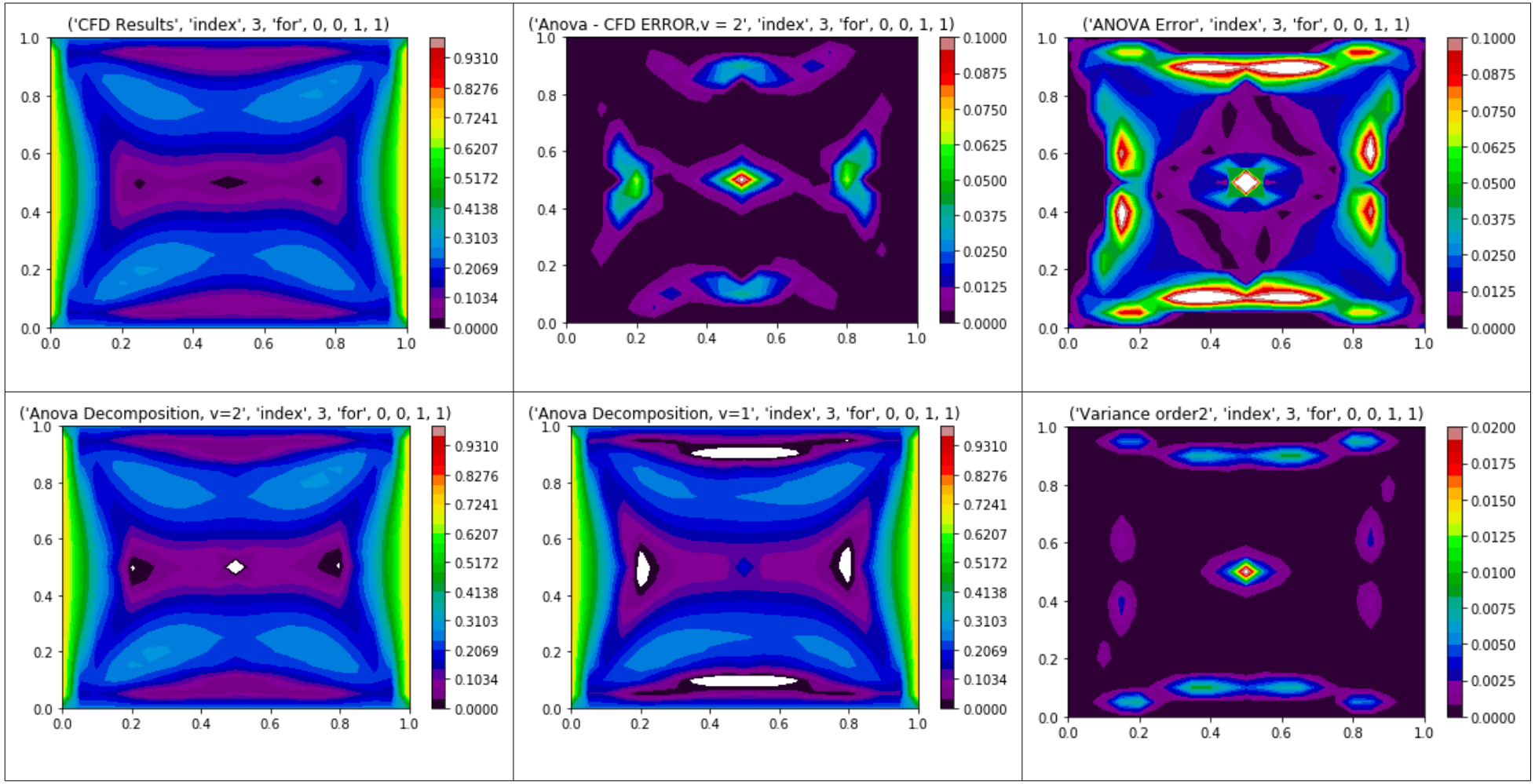
Appendix A: ANOVA Results

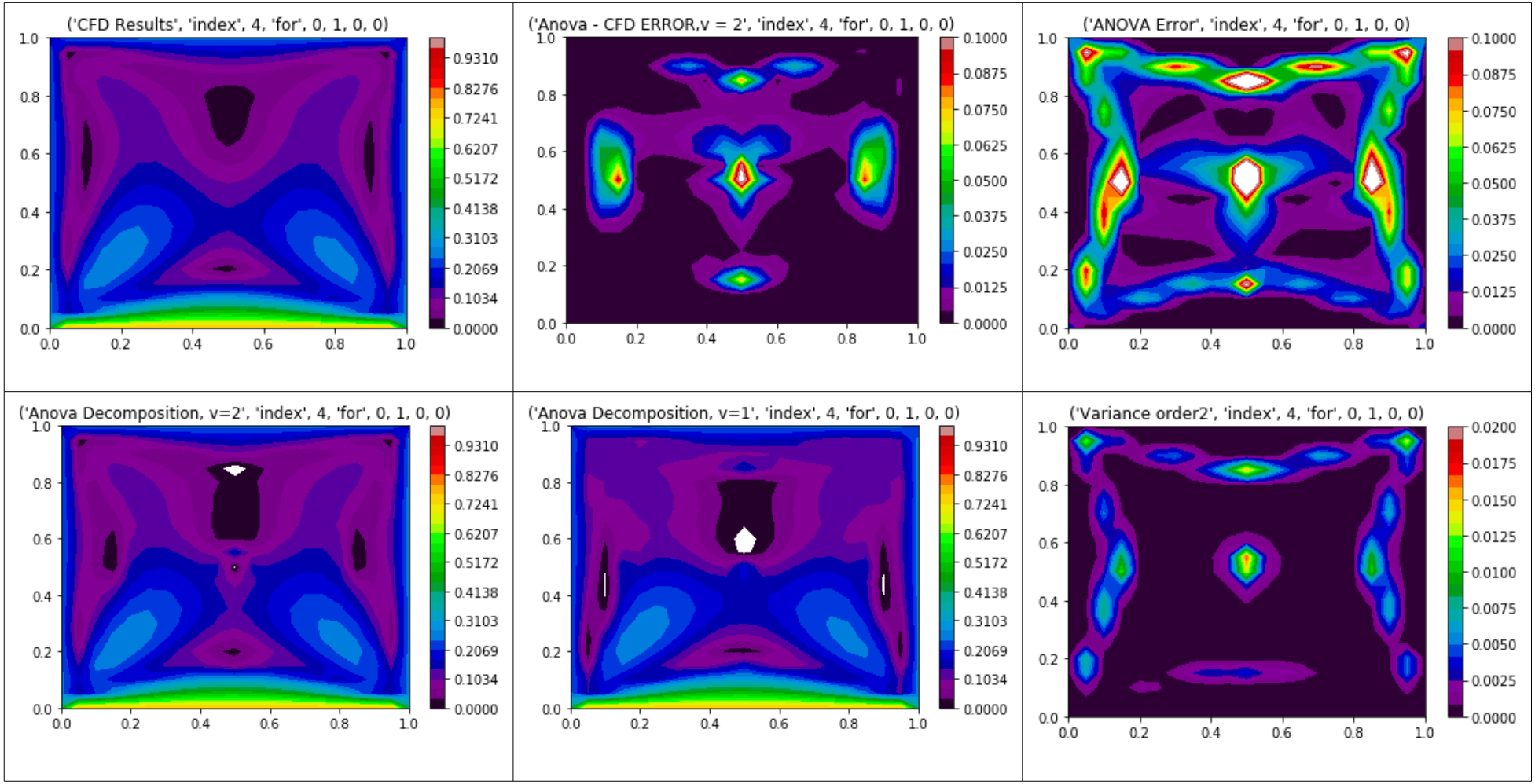
Figure A.1: Results from CFD, Error of ANOVA decomposition, $v = 2$ (refer (15)) , ANOVA Error Estimator (18), Variance of 2nd Order terms (refer (17)), ANOVA decomposition ($v = 1$) and ANOVA decomposition ($v = 2$), in a clockwise sense. Indices's as per Figure 2

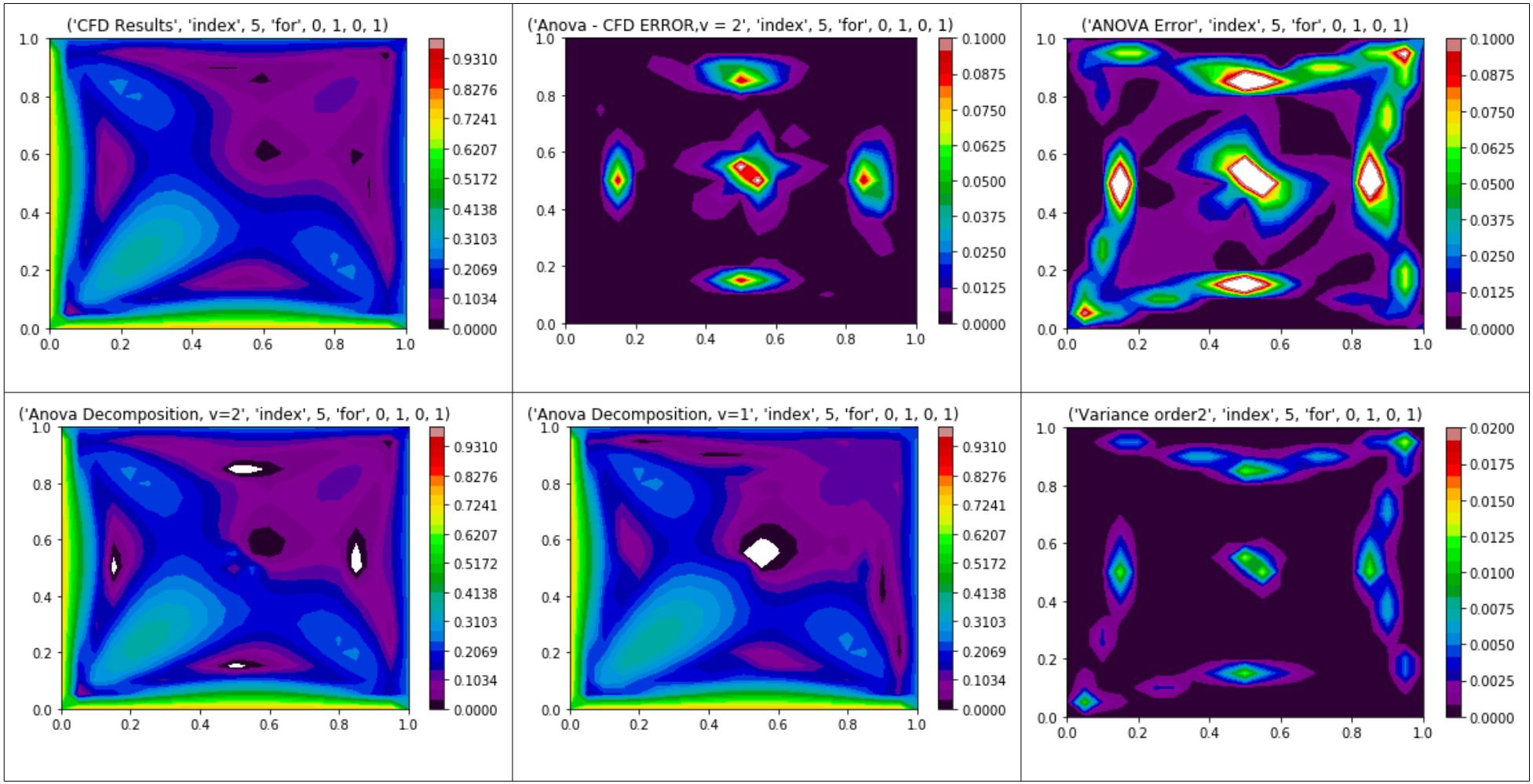


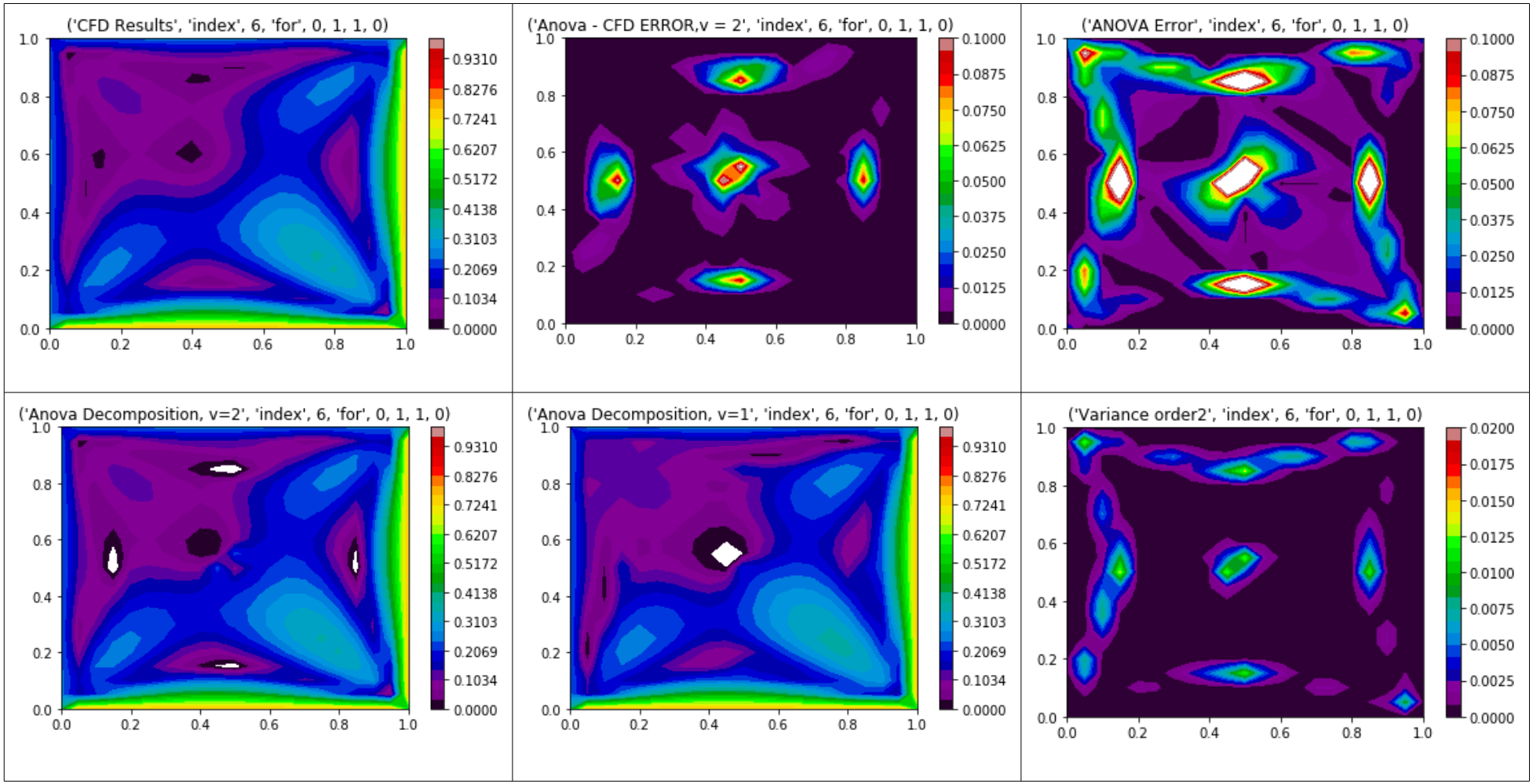

('CFD Results', 'index', 1, 'for', 0, 0, 0, 1)
('Anova - CFD ERROR,v = 2', 'index', 1, 'for', 0, 0, 0, 1)
('ANOVA Error', 'index', 1, 'for', 0, 0, 0, 1)
('Anova Decomposition, v=2', 'index', 1, 'for', 0, 0, 0, 1)
('Anova Decomposition, v=1', 'index', 1, 'for', 0, 0, 0, 1)
('Variance order2', 'index', 1, 'for', 0, 0, 0, 1)

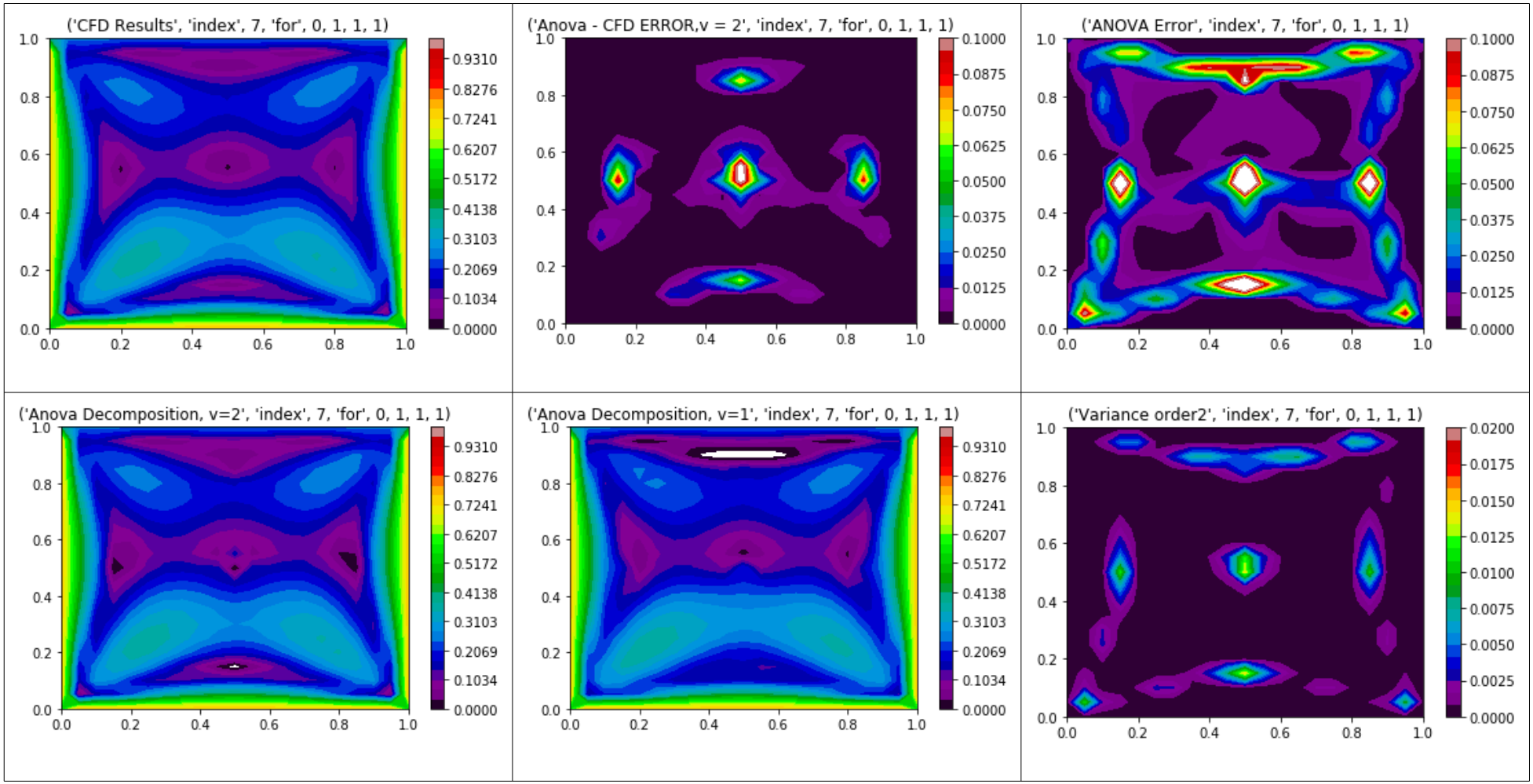

('CFD Results', 'index', 2, 'for', 0, 0, 1, 0)
('Anova - CFD ERROR,v = 2', 'index', 2, 'for', 0, 0, 1, 0)
('ANOVA Error', 'index', 2, 'for', 0, 0, 1, 0)
('Anova Decomposition, v=2', 'index', 2, 'for', 0, 0, 1, 0)
('Anova Decomposition, v=1', 'index', 2, 'for', 0, 0, 1, 0)
('Variance order2', 'index', 2, 'for', 0, 0, 1, 0)

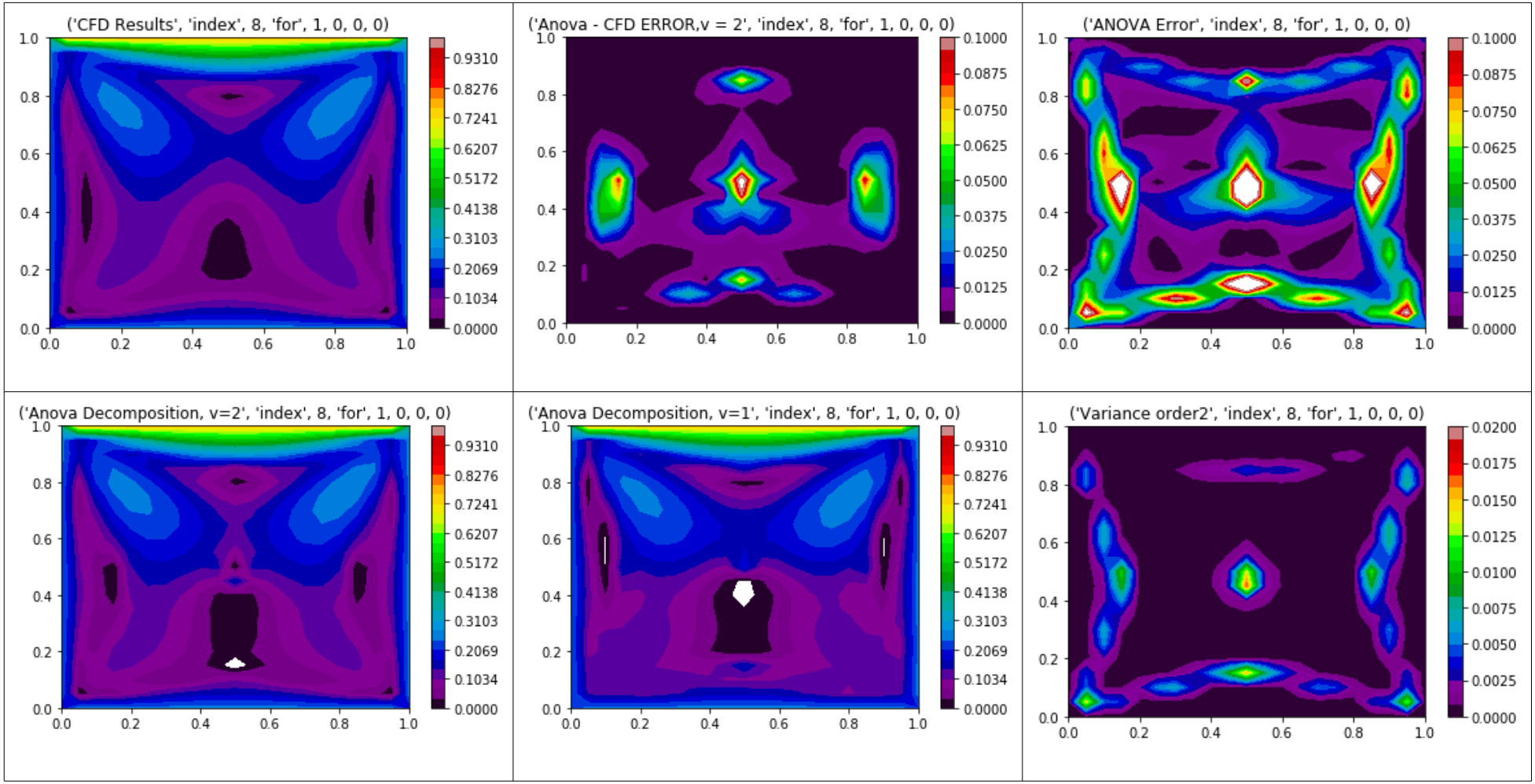

('CFD Results', 'index', 3, 'for', 0, 0, 1, 1)
('Anova - CFD ERROR,v = 2', 'index', 3, 'for', 0, 0, 1, 1)
('ANOVA Error', 'index', 3, 'for', 0, 0, 1, 1)
('Anova Decomposition, v=2', 'index', 3, 'for', 0, 0, 1, 1)
('Anova Decomposition, v=1', 'index', 3, 'for', 0, 0, 1, 1)
('Variance order2', 'index', 3, 'for', 0, 0, 1, 1)

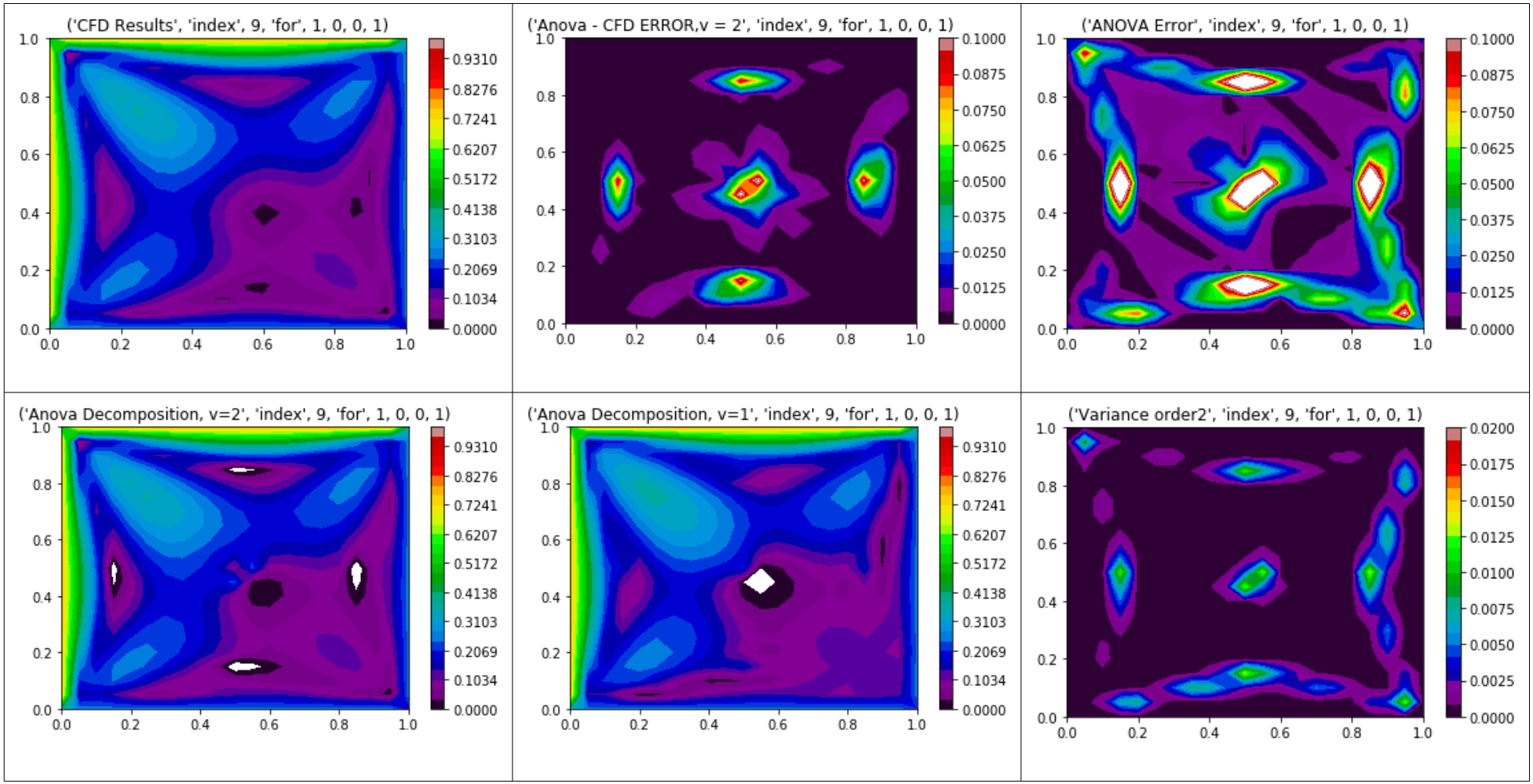

('CFD Results', 'index', 4, 'for', 0, 1, 0, 0)
('Anova - CFD ERROR,v = 2', 'index', 4, 'for', 0, 1, 0, 0)
('ANOVA Error', 'index', 4, 'for', 0, 1, 0, 0)
('Anova Decomposition, v=2', 'index', 4, 'for', 0, 1, 0, 0)
('Anova Decomposition, v=1', 'index', 4, 'for', 0, 1, 0, 0)
('Variance order2', 'index', 4, 'for', 0, 1, 0, 0)

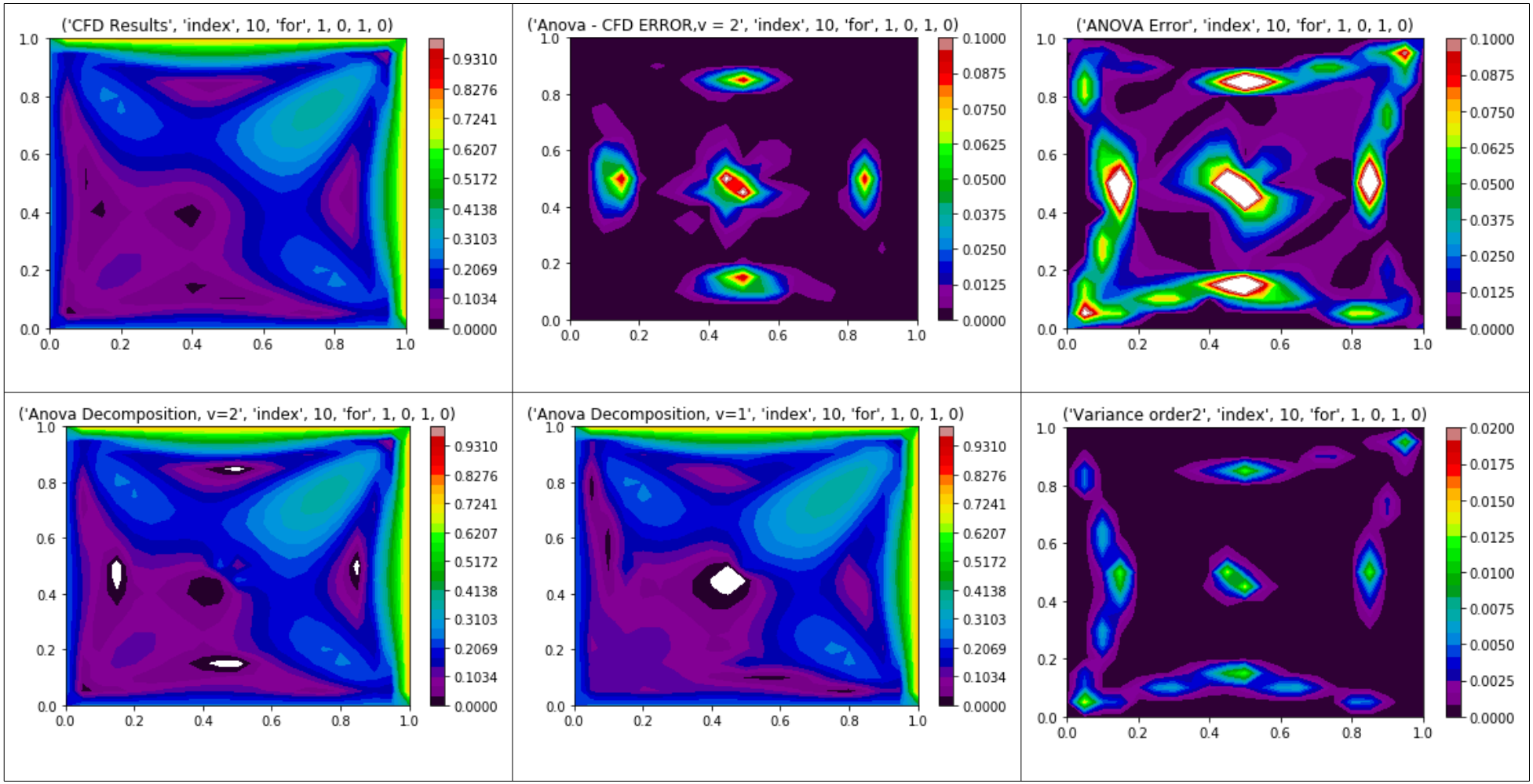

('CFD Results', 'index', 5, 'for', 0, 1, 0, 1)
('Anova - CFD ERROR,v = 2', 'index', 5, 'for', 0, 1, 0, 1)
('ANOVA Error', 'index', 5, 'for', 0, 1, 0, 1)
('Anova Decomposition, v=2', 'index', 5, 'for', 0, 1, 0, 1)
('Anova Decomposition, v=1', 'index', 5, 'for', 0, 1, 0, 1)
('Variance order2', 'index', 5, 'for', 0, 1, 0, 1)

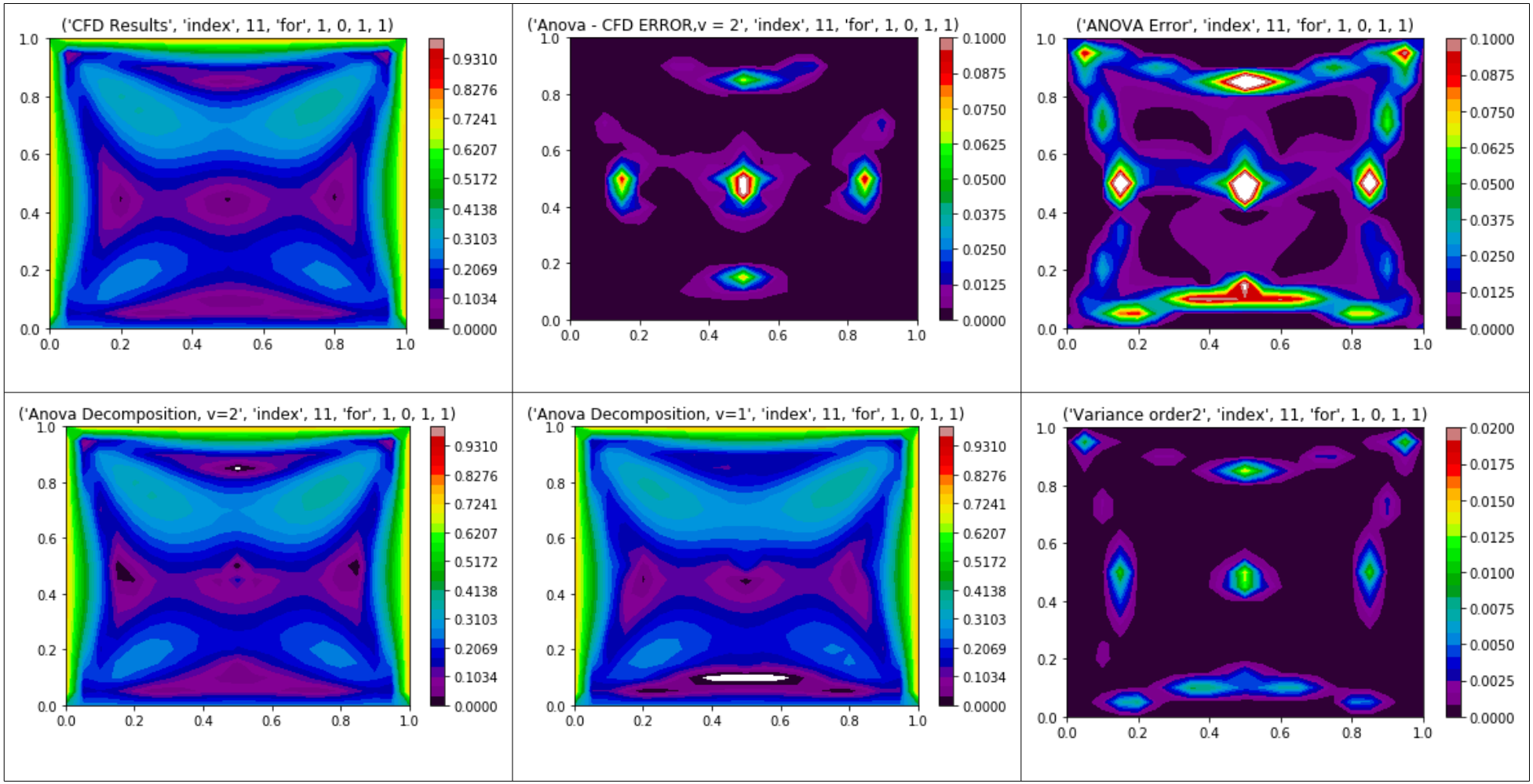

('CFD Results', 'index', 6, 'for', 0, 1, 1, 0)
('Anova - CFD ERROR,v = 2', 'index', 6, 'for', 0, 1, 1, 0)
('ANOVA Error', 'index', 6, 'for', 0, 1, 1, 0)
('Anova Decomposition, v=2', 'index', 6, 'for', 0, 1, 1, 0)
('Anova Decomposition, v=1', 'index', 6, 'for', 0, 1, 1, 0)
('Variance order2', 'index', 6, 'for', 0, 1, 1, 0)

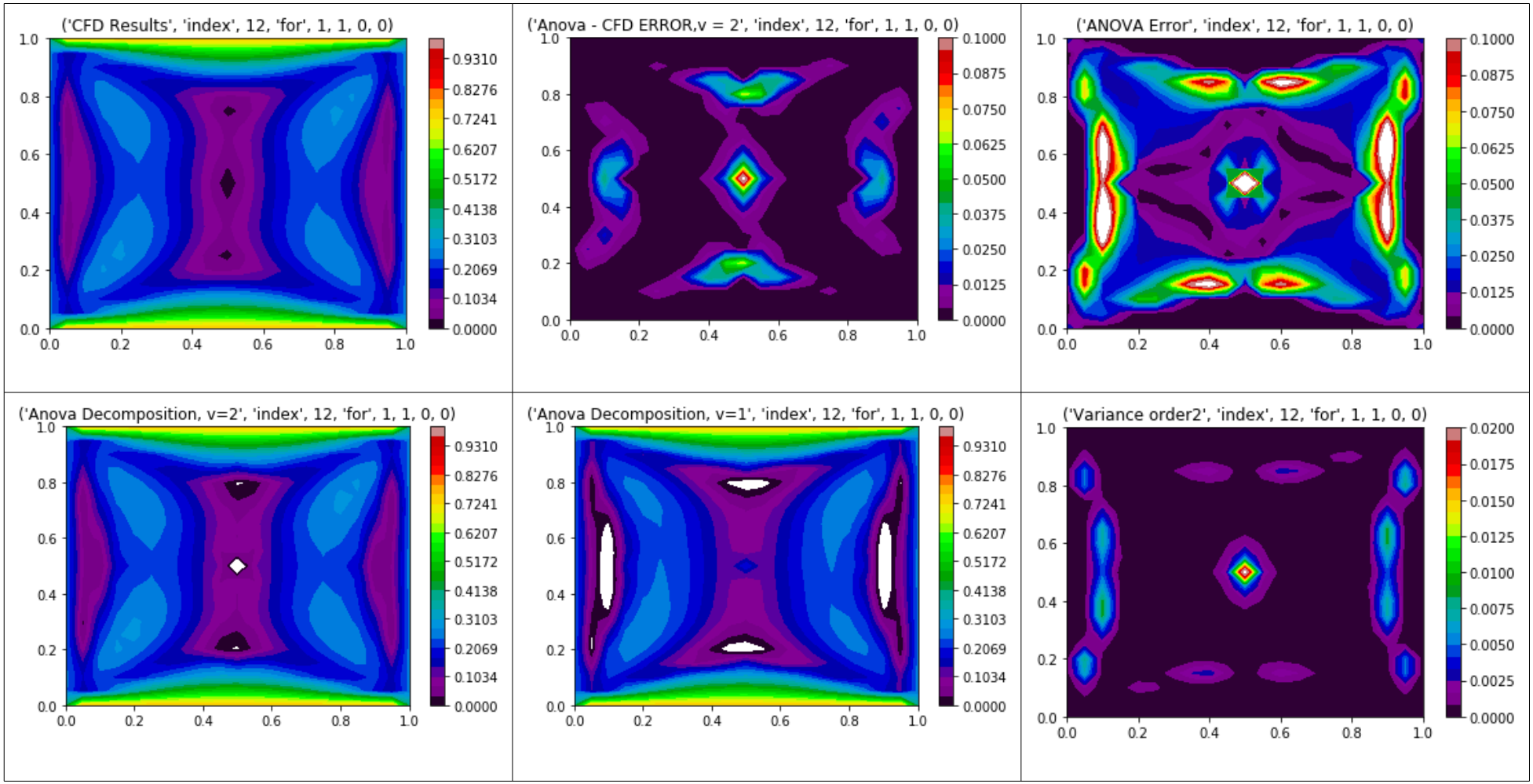

('CFD Results', 'index', 7, 'for', 0, 1, 1, 1)
('Anova - CFD ERROR,v = 2', 'index', 7, 'for', 0, 1, 1, 1)
('ANOVA Error', 'index', 7, 'for', 0, 1, 1, 1)
('Anova Decomposition, v=2', 'index', 7, 'for', 0, 1, 1, 1)
('Anova Decomposition, v=1', 'index', 7, 'for', 0, 1, 1, 1)
('Variance order2', 'index', 7, 'for', 0, 1, 1, 1)

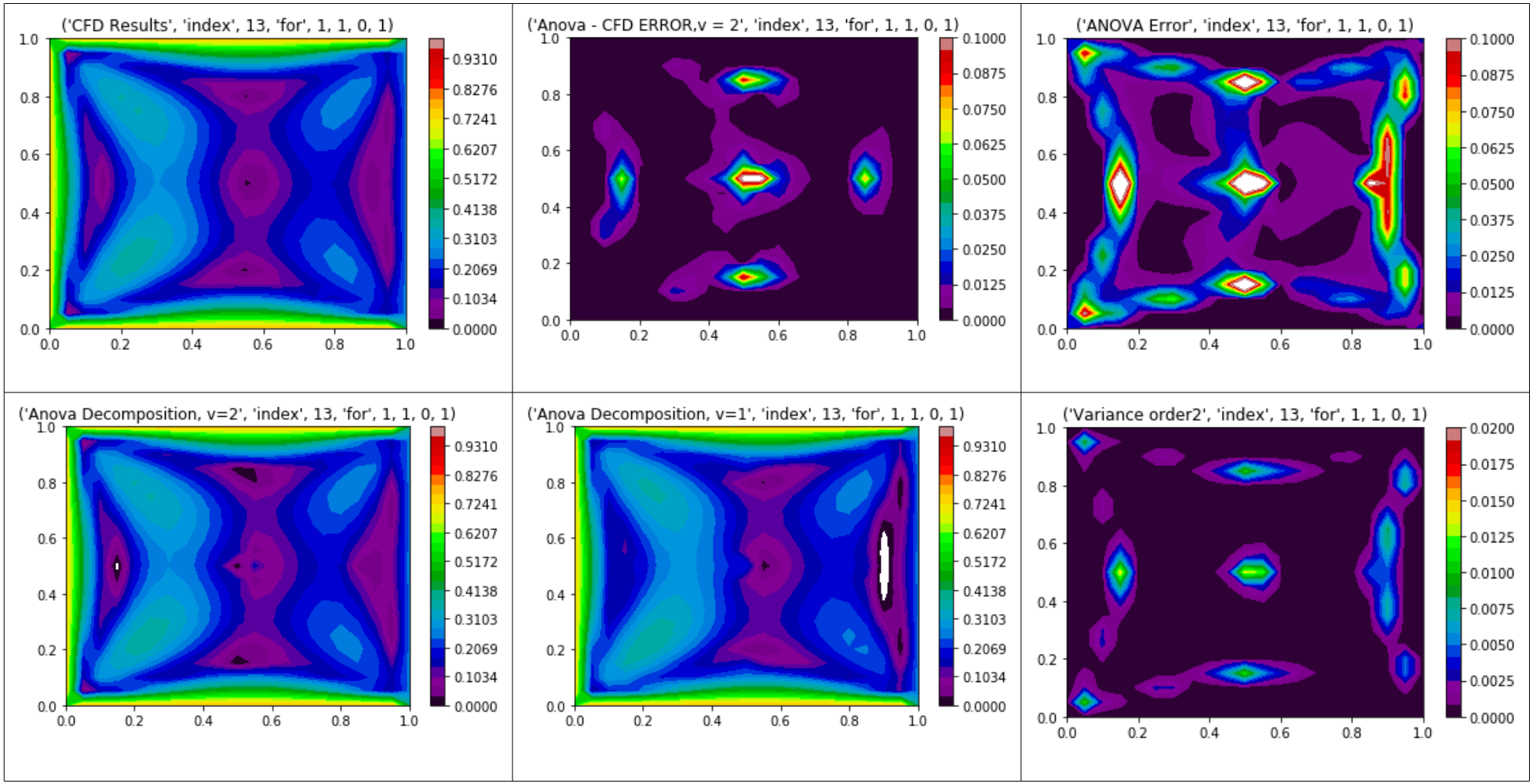

('CFD Results', 'index', 8, 'for', 1, 0, 0, 0)
('Anova - CFD ERROR,v = 2', 'index', 8, 'for', 1, 0, 0, 0)
('ANOVA Error', 'index', 8, 'for', 1, 0, 0, 0)
('Anova Decomposition, v=2', 'index', 8, 'for', 1, 0, 0, 0)
('Anova Decomposition, v=1', 'index', 8, 'for', 1, 0, 0, 0)
('Variance order2', 'index', 8, 'for', 1, 0, 0, 0)

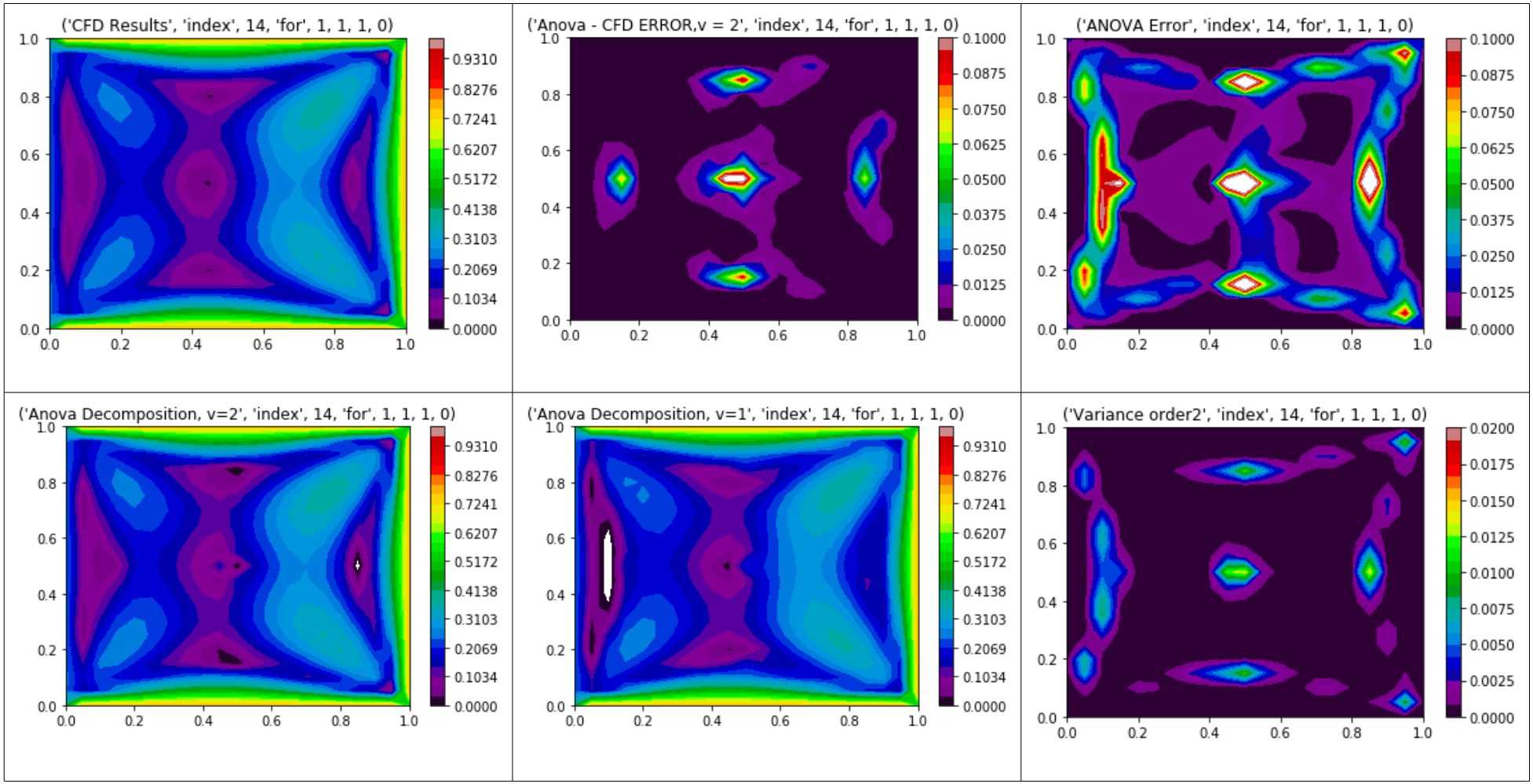

('CFD Results', 'index', 9, 'for', 1, 0, 0, 1)
('Anova - CFD ERROR,v = 2', 'index', 9, 'for', 1, 0, 0, 1)
('ANOVA Error', 'index', 9, 'for', 1, 0, 0, 1)
('Anova Decomposition, v=2', 'index', 9, 'for', 1, 0, 0, 1)
('Anova Decomposition, v=1', 'index', 9, 'for', 1, 0, 0, 1)
('Variance order2', 'index', 9, 'for', 1, 0, 0, 1)

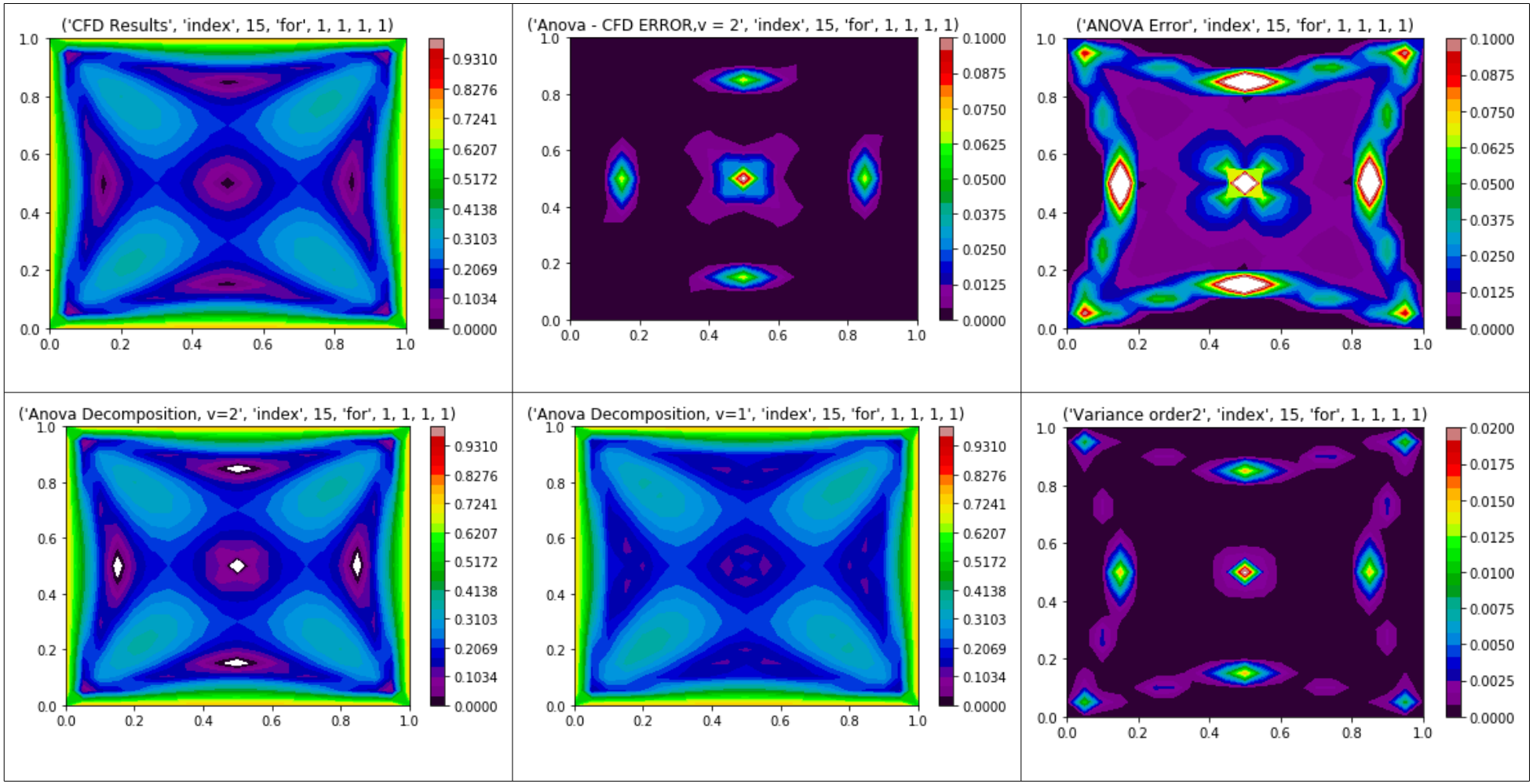

('CFD Results', 'index', 10, 'for', 1, 0, 1, 0)
('Anova - CFD ERROR,v = 2', 'index', 10, 'for', 1, 0, 1, 0)
('ANOVA Error', 'index', 10, 'for', 1, 0, 1, 0)
('Anova Decomposition, v=2', 'index', 10, 'for', 1, 0, 1, 0)
('Anova Decomposition, v=1', 'index', 10, 'for', 1, 0, 1, 0)
('Variance order2', 'index', 10, 'for', 1, 0, 1, 0)


('CFD Results', 'index', 11, 'for', 1, 0, 1, 1)
('Anova - CFD ERROR,v = 2', 'index', 11, 'for', 1, 0, 1, 1)
('ANOVA Error', 'index', 11, 'for', 1, 0, 1, 1)
('Anova Decomposition, v=2', 'index', 11, 'for', 1, 0, 1, 1)
('Anova Decomposition, v=1', 'index', 11, 'for', 1, 0, 1, 1)
('Variance order2', 'index', 11, 'for', 1, 0, 1, 1)


('CFD Results', 'index', 12, 'for', 1, 1, 0, 0)
('Anova - CFD ERROR,v = 2', 'index', 12, 'for', 1, 1, 0, 0)
('ANOVA Error', 'index', 12, 'for', 1, 1, 0, 0)
('Anova Decomposition, v=2', 'index', 12, 'for', 1, 1, 0, 0)
('Anova Decomposition, v=1', 'index', 12, 'for', 1, 1, 0, 0)
('Variance order2', 'index', 12, 'for', 1, 1, 0, 0)

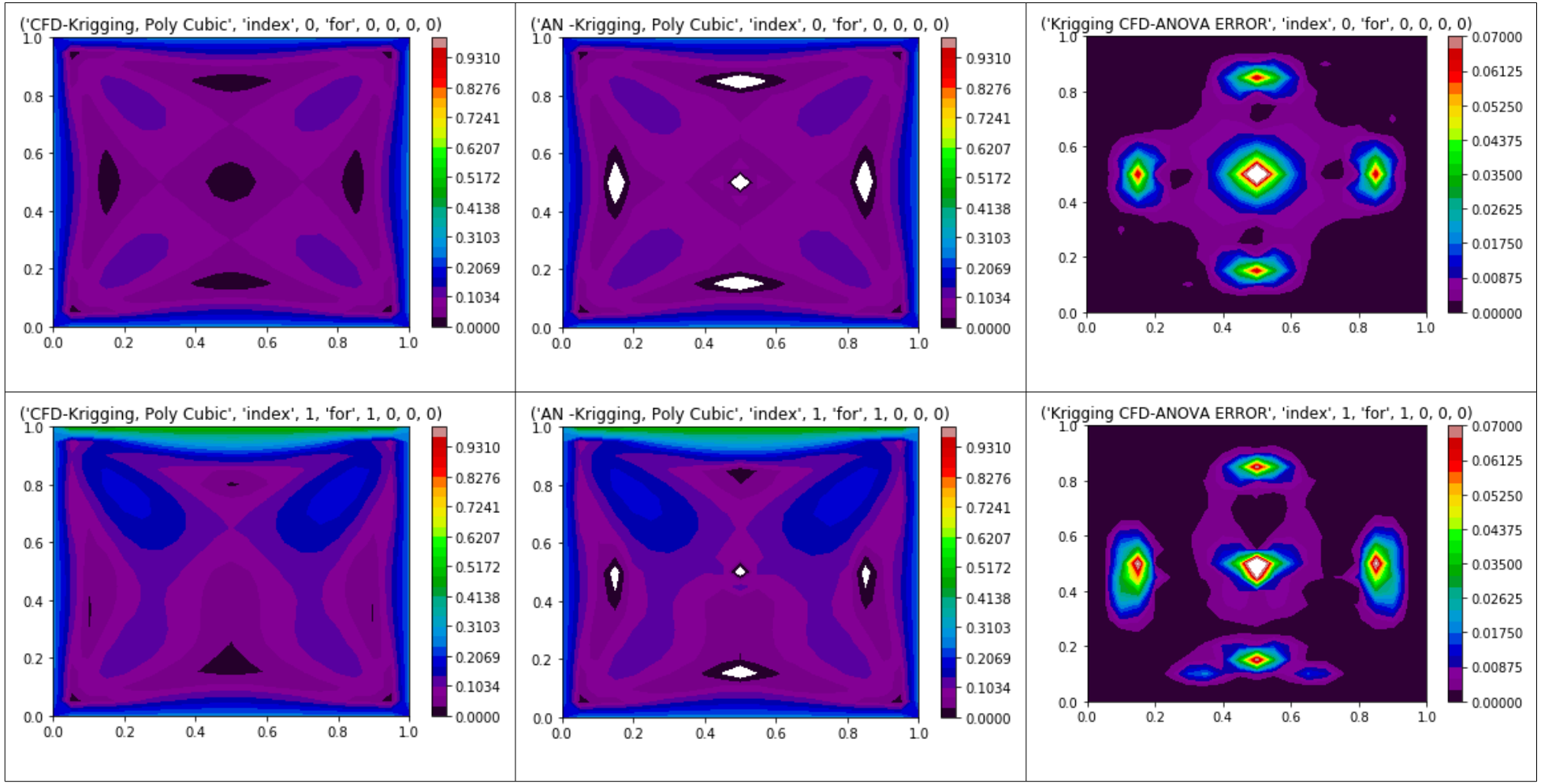

('CFD Results', 'index', 13, 'for', 1, 1, 0, 1)
('Anova - CFD ERROR,v = 2', 'index', 13, 'for', 1, 1, 0, 1)
('ANOVA Error', 'index', 13, 'for', 1, 1, 0, 1)
('Anova Decomposition, v=2', 'index', 13, 'for', 1, 1, 0, 1)
('Anova Decomposition, v=1', 'index', 13, 'for', 1, 1, 0, 1)
('Variance order2', 'index', 13, 'for', 1, 1, 0, 1)

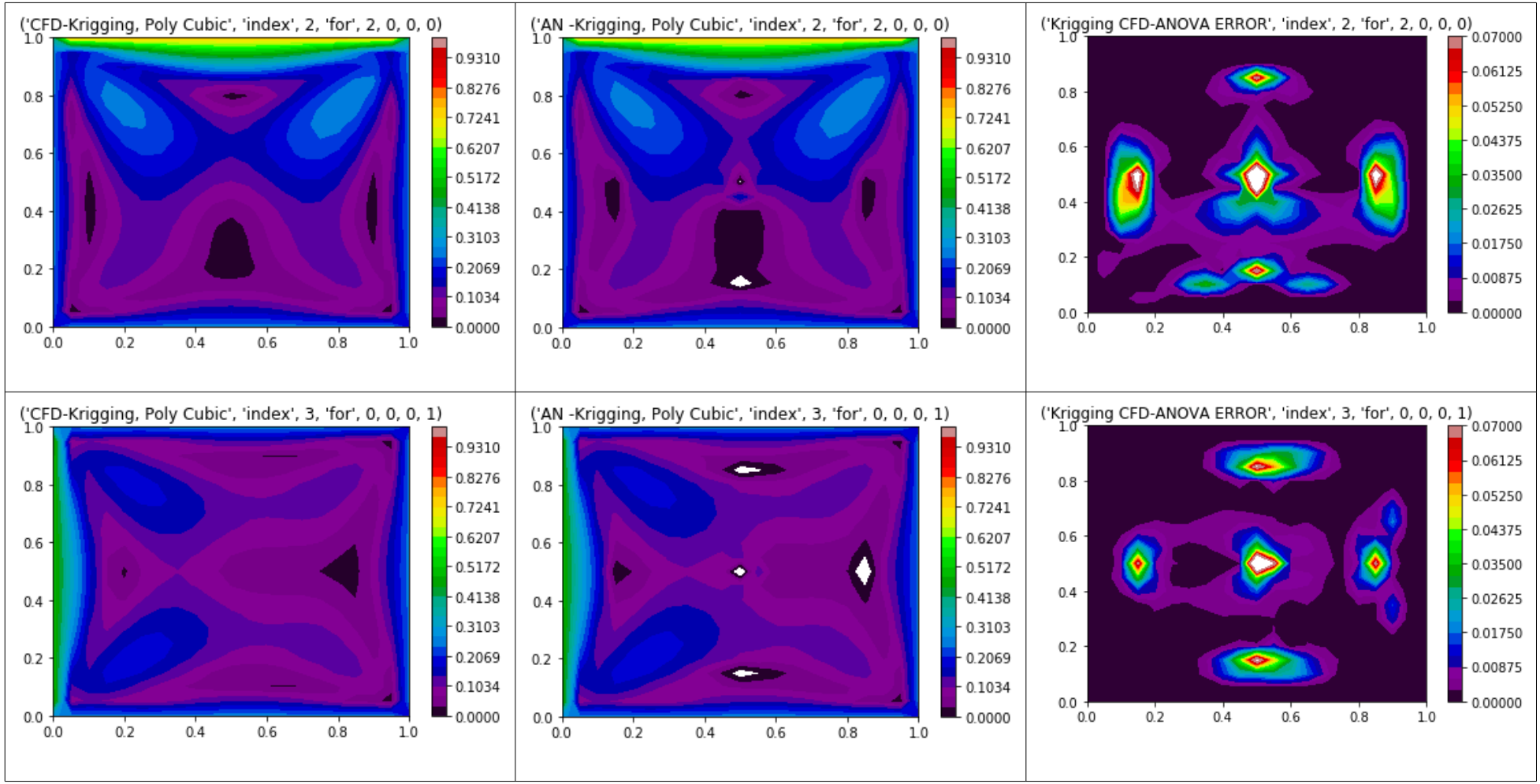

('CFD Results', 'index', 14, 'for', 1, 1, 1, 0)
('Anova - CFD ERROR,v = 2', 'index', 14, 'for', 1, 1, 1, 0)
('ANOVA Error', 'index', 14, 'for', 1, 1, 1, 0)
('Anova Decomposition, v=2', 'index', 14, 'for', 1, 1, 1, 0)
('Anova Decomposition, v=1', 'index', 14, 'for', 1, 1, 1, 0)
('Variance order2', 'index', 14, 'for', 1, 1, 1, 0)

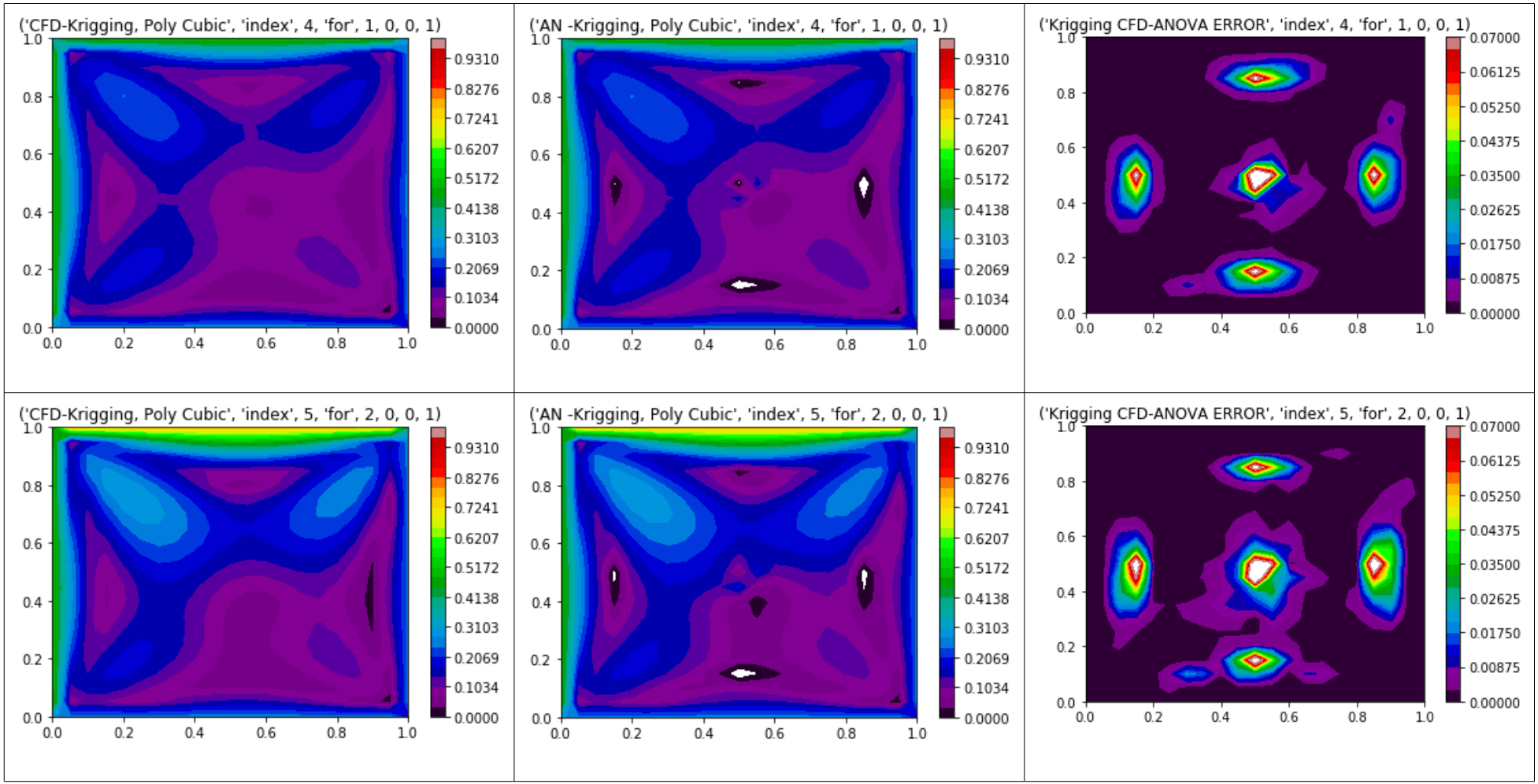

('CFD Results', 'index', 15, 'for', 1, 1, 1, 1)
('Anova - CFD ERROR,v = 2', 'index', 15, 'for', 1, 1, 1, 1)
('ANOVA Error', 'index', 15, 'for', 1, 1, 1, 1)
('Anova Decomposition, v=2', 'index', 15, 'for', 1, 1, 1, 1)
('Anova Decomposition, v=1', 'index', 15, 'for', 1, 1, 1, 1)
('Variance order2', 'index', 15, 'for', 1, 1, 1, 1)

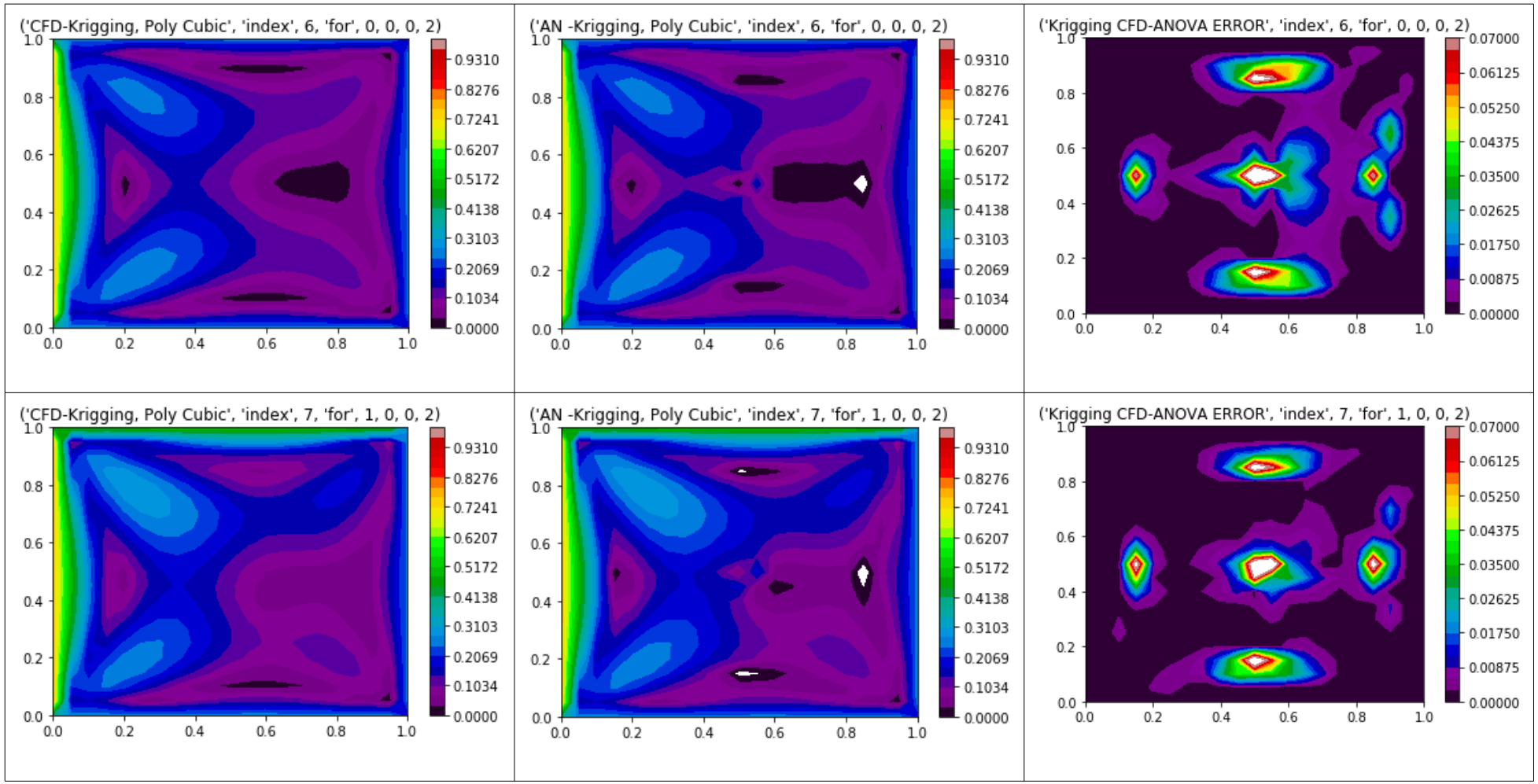
Appendix B: Krigging Results

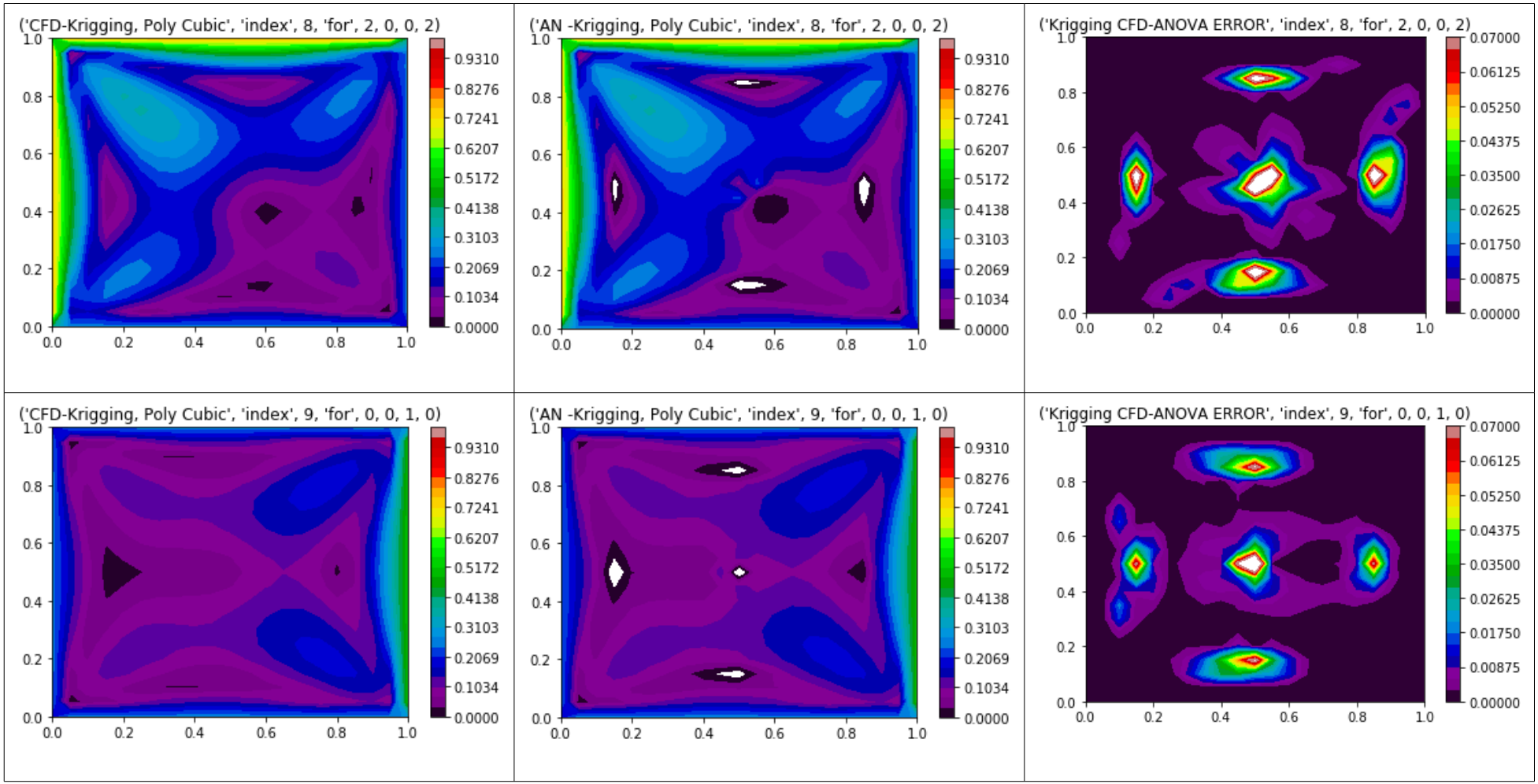
Figure B.1: From left to right, Krigging using “Polynomial Cubic Spline” kernel with discretization sample = 3; applied to CFD & ANOVA; and their absolute difference (*only the 1st 20 results are shown here out of 81*).

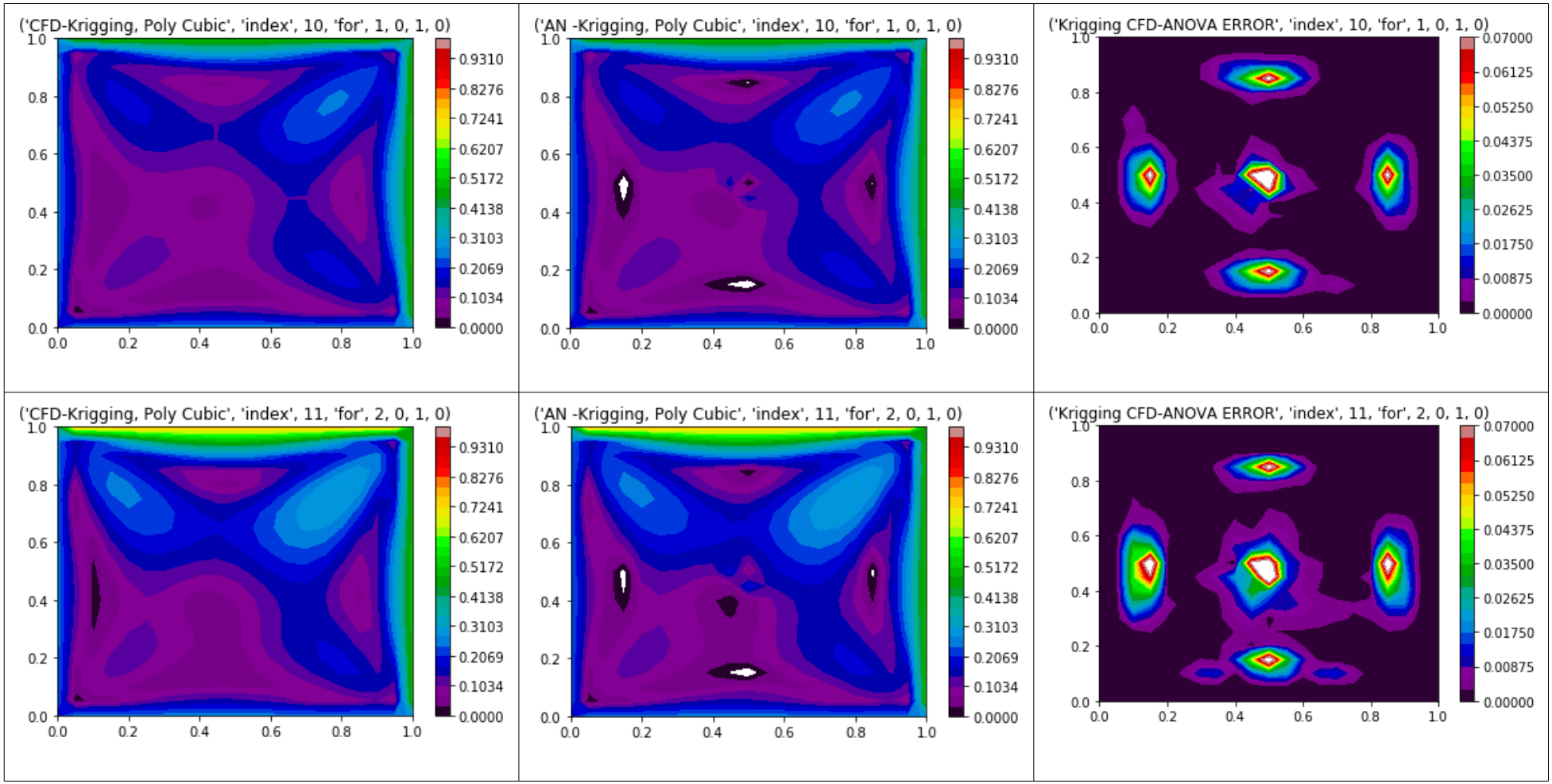


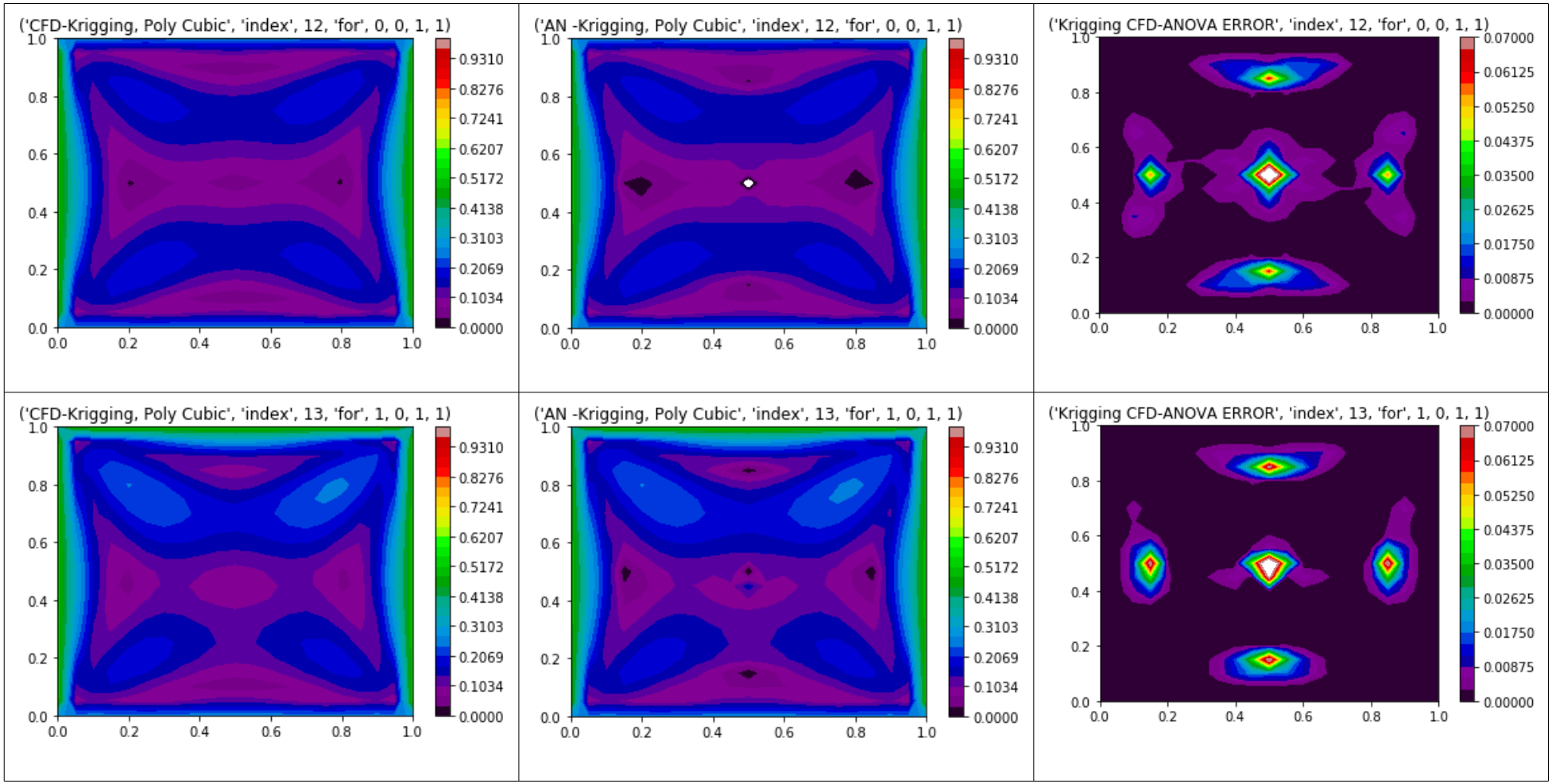

('CFD-Krigging, Poly Cubic', 'index', 2, 'for', 2, 0, 0, 0)
('AN -Krigging, Poly Cubic', 'index', 2, 'for', 2, 0, 0, 0)
('Krigging CFD-ANOVA ERROR', 'index', 2, 'for', 2, 0, 0, 0)
('CFD-Krigging, Poly Cubic', 'index', 3, 'for', 0, 0, 0, 1)
('AN -Krigging, Poly Cubic', 'index', 3, 'for', 0, 0, 0, 1)
('Krigging CFD-ANOVA ERROR', 'index', 3, 'for', 0, 0, 0, 1)
0.0 0.2 0.4 0.6 0.8 1.0
0.9310 0.8276 0.7241 0.6207 0.5172 0.4138 0.3103 0.2069 0.1034 0.0000
0.07000 0.06125 0.05250 0.04375 0.03500 0.02625 0.01750 0.00875 0.00000

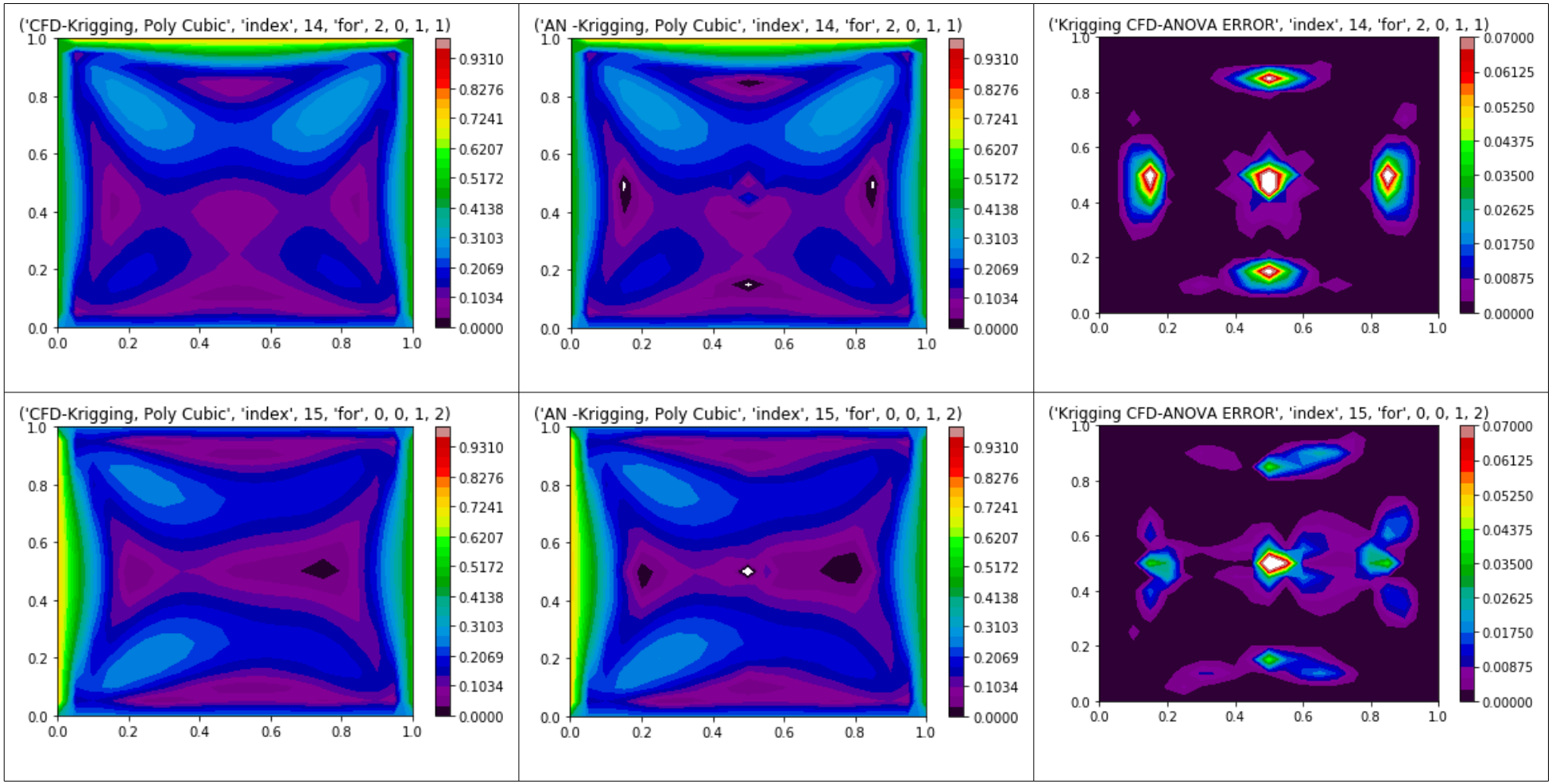

('CFD-Krigging, Poly Cubic', 'index', 4, 'for', 1, 0, 0, 1)
('AN -Krigging, Poly Cubic', 'index', 4, 'for', 1, 0, 0, 1)
('Krigging CFD-ANOVA ERROR', 'index', 4, 'for', 1, 0, 0, 1)
('CFD-Krigging, Poly Cubic', 'index', 5, 'for', 2, 0, 0, 1)
('AN -Krigging, Poly Cubic', 'index', 5, 'for', 2, 0, 0, 1)
('Krigging CFD-ANOVA ERROR', 'index', 5, 'for', 2, 0, 0, 1)
0.0 0.2 0.4 0.6 0.8 1.0
0.9310 0.8276 0.7241 0.6207 0.5172 0.4138 0.3103 0.2069 0.1034 0.0000
0.07000 0.06125 0.05250 0.04375 0.03500 0.02625 0.01750 0.00875 0.00000

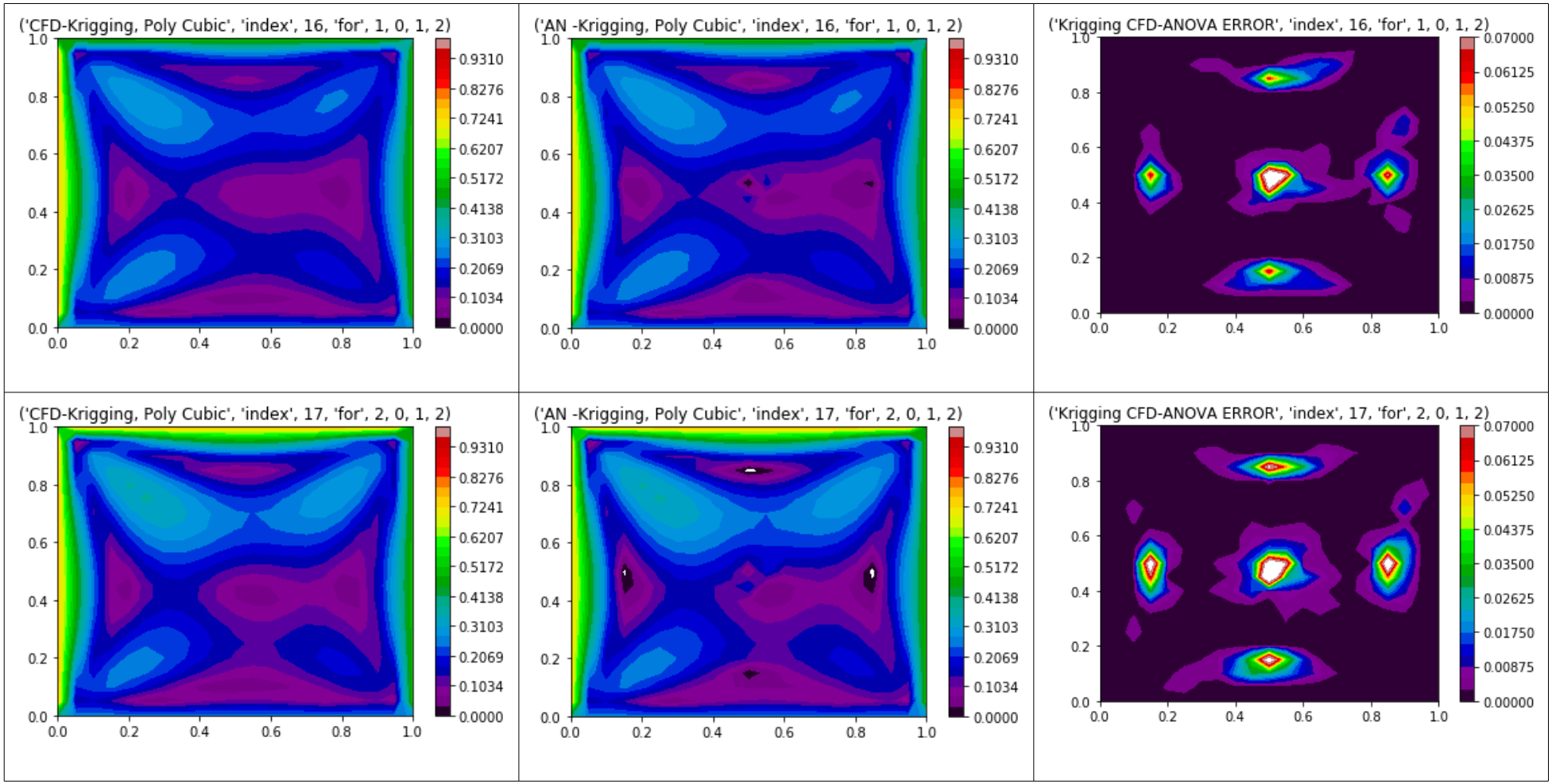

('CFD-Krigging, Poly Cubic', 'index', 6, 'for', 0, 0, 0, 2)
('AN -Krigging, Poly Cubic', 'index', 6, 'for', 0, 0, 0, 2)
('Krigging CFD-ANOVA ERROR', 'index', 6, 'for', 0, 0, 0, 2)
('CFD-Krigging, Poly Cubic', 'index', 7, 'for', 1, 0, 0, 2)
('AN -Krigging, Poly Cubic', 'index', 7, 'for', 1, 0, 0, 2)
('Krigging CFD-ANOVA ERROR', 'index', 7, 'for', 1, 0, 0, 2)
0.0 0.2 0.4 0.6 0.8 1.0
0.9310 0.8276 0.7241 0.6207 0.5172 0.4138 0.3103 0.2069 0.1034 0.0000
0.07000 0.06125 0.05250 0.04375 0.03500 0.02625 0.01750 0.00875 0.00000

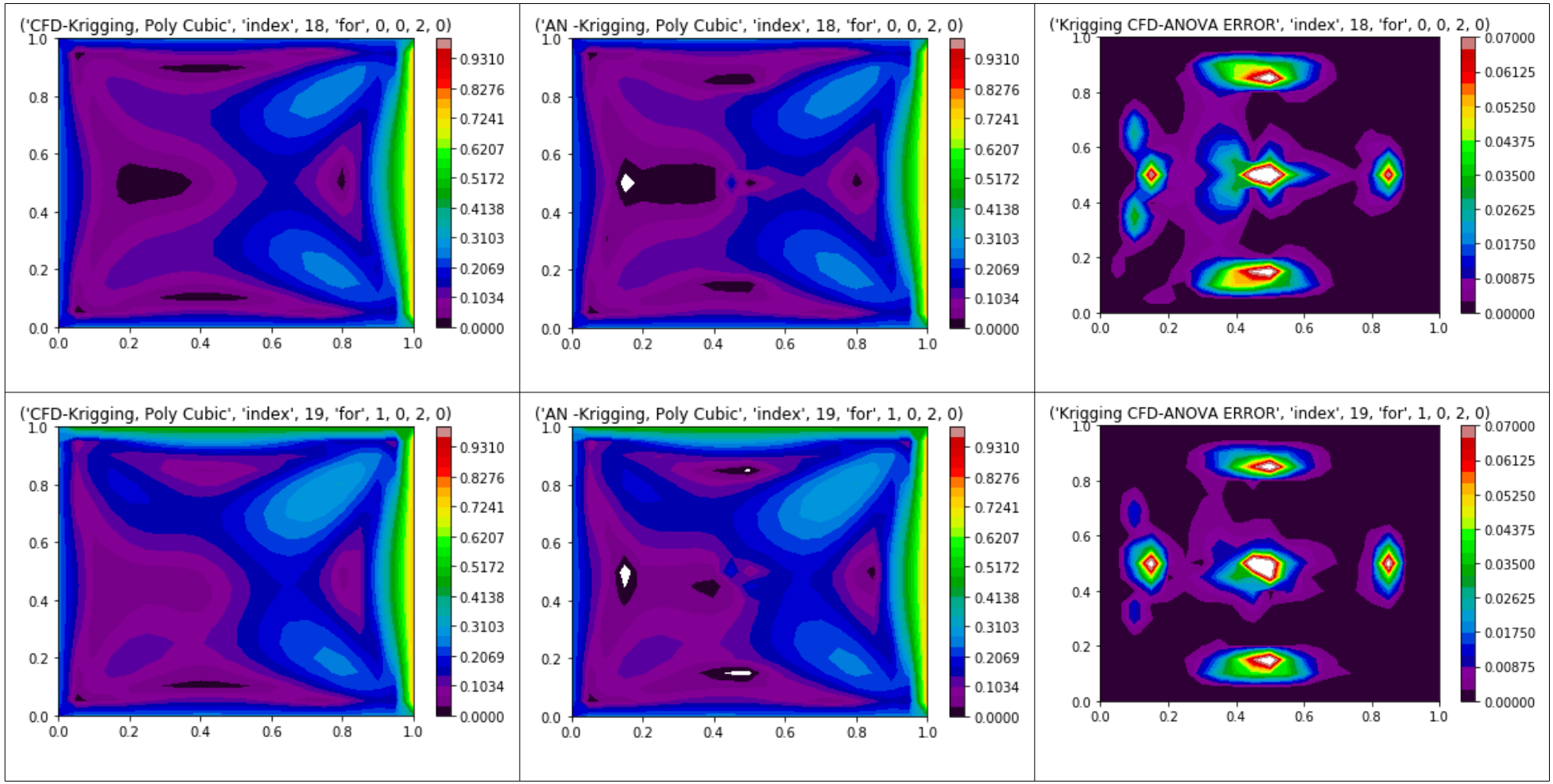

('CFD-Krigging, Poly Cubic', 'index', 8, 'for', 2, 0, 0, 2)
('AN -Krigging, Poly Cubic', 'index', 8, 'for', 2, 0, 0, 2)
('Krigging CFD-ANOVA ERROR', 'index', 8, 'for', 2, 0, 0, 2)
('CFD-Krigging, Poly Cubic', 'index', 9, 'for', 0, 0, 1, 0)
('AN -Krigging, Poly Cubic', 'index', 9, 'for', 0, 0, 1, 0)
('Krigging CFD-ANOVA ERROR', 'index', 9, 'for', 0, 0, 1, 0)
0.0 0.2 0.4 0.6 0.8 1.0
0.9310 0.8276 0.7241 0.6207 0.5172 0.4138 0.3103 0.2069 0.1034 0.0000
0.07000 0.06125 0.05250 0.04375 0.03500 0.02625 0.01750 0.00875 0.00000


('CFD-Krigging, Poly Cubic', 'index', 10, 'for', 1, 0, 1, 0)
('AN -Krigging, Poly Cubic', 'index', 10, 'for', 1, 0, 1, 0)
('Krigging CFD-ANOVA ERROR', 'index', 10, 'for', 1, 0, 1, 0)
('CFD-Krigging, Poly Cubic', 'index', 11, 'for', 2, 0, 1, 0)
('AN -Krigging, Poly Cubic', 'index', 11, 'for', 2, 0, 1, 0)
('Krigging CFD-ANOVA ERROR', 'index', 11, 'for', 2, 0, 1, 0)
0.0 0.2 0.4 0.6 0.8 1.0
0.9310 0.8276 0.7241 0.6207 0.5172 0.4138 0.3103 0.2069 0.1034 0.0000
0.07000 0.06125 0.05250 0.04375 0.03500 0.02625 0.01750 0.00875 0.00000


('CFD-Krigging, Poly Cubic', 'index', 12, 'for', 0, 0, 1, 1)
('AN -Krigging, Poly Cubic', 'index', 12, 'for', 0, 0, 1, 1)
('Krigging CFD-ANOVA ERROR', 'index', 12, 'for', 0, 0, 1, 1)
('CFD-Krigging, Poly Cubic', 'index', 13, 'for', 1, 0, 1, 1)
('AN -Krigging, Poly Cubic', 'index', 13, 'for', 1, 0, 1, 1)
('Krigging CFD-ANOVA ERROR', 'index', 13, 'for', 1, 0, 1, 1)
0.0 0.2 0.4 0.6 0.8 1.0
0.9310 0.8276 0.7241 0.6207 0.5172 0.4138 0.3103 0.2069 0.1034 0.0000
0.07000 0.06125 0.05250 0.04375 0.03500 0.02625 0.01750 0.00875 0.00000


('CFD-Krigging, Poly Cubic', 'index', 14, 'for', 2, 0, 1, 1)
('AN -Krigging, Poly Cubic', 'index', 14, 'for', 2, 0, 1, 1)
('Krigging CFD-ANOVA ERROR', 'index', 14, 'for', 2, 0, 1, 1)
('CFD-Krigging, Poly Cubic', 'index', 15, 'for', 0, 0, 1, 2)
('AN -Krigging, Poly Cubic', 'index', 15, 'for', 0, 0, 1, 2)
('Krigging CFD-ANOVA ERROR', 'index', 15, 'for', 0, 0, 1, 2)
0.0 0.2 0.4 0.6 0.8 1.0
0.9310 0.8276 0.7241 0.6207 0.5172 0.4138 0.3103 0.2069 0.1034 0.0000
0.07000 0.06125 0.05250 0.04375 0.03500 0.02625 0.01750 0.00875 0.00000


('CFD-Krigging, Poly Cubic', 'index', 16, 'for', 1, 0, 1, 2)
('AN -Krigging, Poly Cubic', 'index', 16, 'for', 1, 0, 1, 2)
('Krigging CFD-ANOVA ERROR', 'index', 16, 'for', 1, 0, 1, 2)
('CFD-Krigging, Poly Cubic', 'index', 17, 'for', 2, 0, 1, 2)
('AN -Krigging, Poly Cubic', 'index', 17, 'for', 2, 0, 1, 2)
('Krigging CFD-ANOVA ERROR', 'index', 17, 'for', 2, 0, 1, 2)
0.0 0.2 0.4 0.6 0.8 1.0
0.9310 0.8276 0.7241 0.6207 0.5172 0.4138 0.3103 0.2069 0.1034 0.0000
0.07000 0.06125 0.05250 0.04375 0.03500 0.02625 0.01750 0.00875 0.00000


('CFD-Krigging, Poly Cubic', 'index', 18, 'for', 0, 0, 2, 0)
('AN -Krigging, Poly Cubic', 'index', 18, 'for', 0, 0, 2, 0)
('Krigging CFD-ANOVA ERROR', 'index', 18, 'for', 0, 0, 2, 0)
('CFD-Krigging, Poly Cubic', 'index', 19, 'for', 1, 0, 2, 0)
('AN -Krigging, Poly Cubic', 'index', 19, 'for', 1, 0, 2, 0)
('Krigging CFD-ANOVA ERROR', 'index', 19, 'for', 1, 0, 2, 0)
0.0 0.2 0.4 0.6 0.8 1.0
0.9310 0.8276 0.7241 0.6207 0.5172 0.4138 0.3103 0.2069 0.1034 0.0000
0.07000 0.06125 0.05250 0.04375 0.03500 0.02625 0.01750 0.00875 0.00000