Silent error analysis in linear pipelines

Hadrien Croubois [*]

ENS de Lyon
Argonne National Laboratory

1 Introduction

With the increase in size and complexity of modern HPC plateforms, resiliency and error detections is concidered as a key component of new dataflow managers.

In this paper, we evaluate the impact of silent errors produced by memory corruption in unstructured data in linear pipeline. Linear pipeline, as they do not have loop, do not have a temporal dimension and are therefore not subject to time differetiation and machine learning based detectors. We also focused on unstructured data, such as particles clouds, as corruption in thoses is not likely to cause hard errors. Futhermore, hardware errors are likely to corrupt such datasets.

mber

We focus our study on pipelines used in real HPC applications, building a memory corruption injection model and metrics for results quality evaluation.

2 Pipeline description

In the following, we are focusing our study on non iterative pipelines. Thoses pipeline are DAGs where errors are contained by the topology of the dependency graph. Therefore, and unlike cyclic physical simulation, there is no temporal dimension throughout which errors can spread.

The pipeline we will consider in this paper is a density estimation pipeline 1 computing a density field from a particles sampling. In real applications of this pipeline, this sampling might be some data comming from the acquisition of a real experiment or from simulation program such as the Hacc cosmology simultation code.

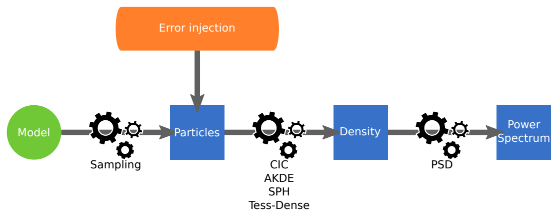

Figure 1: Full pipeline

2.1 Syntetic particles generation

In order for us to have a good idea of the expected results, we will use syntetic particles generated by the sampling of an analytical distribution with caracteristics close to thoses expected from the real application values. That way we have analytical values helping us compared all our results to a known ground truth.

This analytical distribution is a density profil produced by a sum of Navarro-Frenk-White functions [Navarro et al., 1996]. This density profile is further caracterized in .

ref - Self-Adapti Densit Estima tion of Particl Data

$$\mathcal{D}(p) = \sum_{\{k_i,c_i\}} \frac{k_i}{\|p-c_i\|(\|p-c_i\|+1)^2} \qquad (1)$$

We will also use values produced by real application in order to confirm the results comming from the synthetic particles.

2.2 Density estimation

Density estimation aims at reconstructing the density function from a set of samples. This is achieved by adding the contribution of all sample, each sample representing a local density profile. The distribution of this local density profile is caracteristed by a weighting function and by a domain, which caracteristics have been extensivelly studied .

ref

However, more accurate methods also tends to be more computationnaly expensive and not so qualitative methods tends to be prefered in cases where intermediate quality results are satisfactory and computation cost needs to be limited.

We therefore will consider different methods representing differents compromise between results quality and computation cost.

2.3 Density analysis

In order to compare different density fields produced by the density estimation methods, we need a metric caracterising the representative elements of those density fields. This metric should differentiate values according to the relevant elements in regard to their futher analysis in real application.

For this step, the use a power spectrum analysis which makes visible variation responses for different caracteristic length, hence detecting bias such as high frequency noise or over smoothness. In order to compare distances between those spectral responces we use the following integral based metric :

[*]`hadrien.croubois@ens-lyon.fr`

$$d(ps_1, ps_2) = \int_{\Delta f} \frac{\left\| \log\left(\frac{ps_1(f)}{ps_2(f)}\right) \right\|}{f^2} \, \mathrm{d}f \qquad (2)$$

3 Density estimation methods

Several methods exist to perform density estimation. Those methodes give result with different level of quality for different computational cost. This section will focus on describing some of those methods, which represent different compromise.

3.1 Methods description

3.1.1 CIC

CIC [Birdsall and Fuss, 1969] (Cloud in cell) is the simpler and cheapest of all methods presented in this paper.

This method simply distribute each sample weight among the closest grid points. This method has a very low computational cost ($\mathcal{O}(n)$) but suffer from a very low quality in parse areas where each sample should be distributed over a larger area, resulting in a lot of noise.

3.1.2 AKDE

AKDE [Rosenblatt, 1956] [Parzen, 1962] (Adaptive Kernel Density Estimator) methods are the natural variant of CIC.

To try and solve the high quantity of noise, Kernel Density Estimator uses large windows over which they distribute the weight of each sample. Those weight follow a kernel function, typically gaussian function.

In order to avoid noise in parse areas and oversmoothing in dense areas, the windows sizes are follow the local density. Windows adaptation criterions have been extensivelly studied [Heidenreich et al., 2013] but can, in some cases, dramatically increase the computational cost.

Our implementation uses a kd-tree construction for selecting the window sizes. Overall the complexity is $\mathcal{O}(n \log n + g^3)$ with far better results then CIC (cf 3.2).

3.1.3 SPH

write

...

3.1.4 Tess-Dense

Unlike all previous algorithm, Tess-Dense doesn't use fixed shape windows. Indeed, Tess-Dense uses a voronoï tesselation to determine each sample influence domaine. This precomputation is independant of the result grid's resolution therefore reducing the memory footprint on high resolution reconstruction. This cost is however paid even for small resolutions.

-
lf-
aptive
nsity
tima-
n of
rticle
ta

The cost of this method is $\mathcal{O}(n^2 + g^3)$. While produced results are good, they suffer from some noise caused by voronoï cells' instability in sparse areas.

3.2 Methods results

To compare thoses different methods results we plot the frequencial reponse of their result and compare them to the analytical result. One should not forget that the input set of those methods are sampling of a density function. As those sampling are inherently random, we cannot just observe on sampling and the associated result.

In order to take this randomness into account we computed several samplings of our density function model and then display both the median of the frequencial results overs thoses samplings as well as extreme values. That way we can not only describe the expected quality of those methods but also the natural variability we can expect. Figure 2 present thoses results for all prevously described methods.

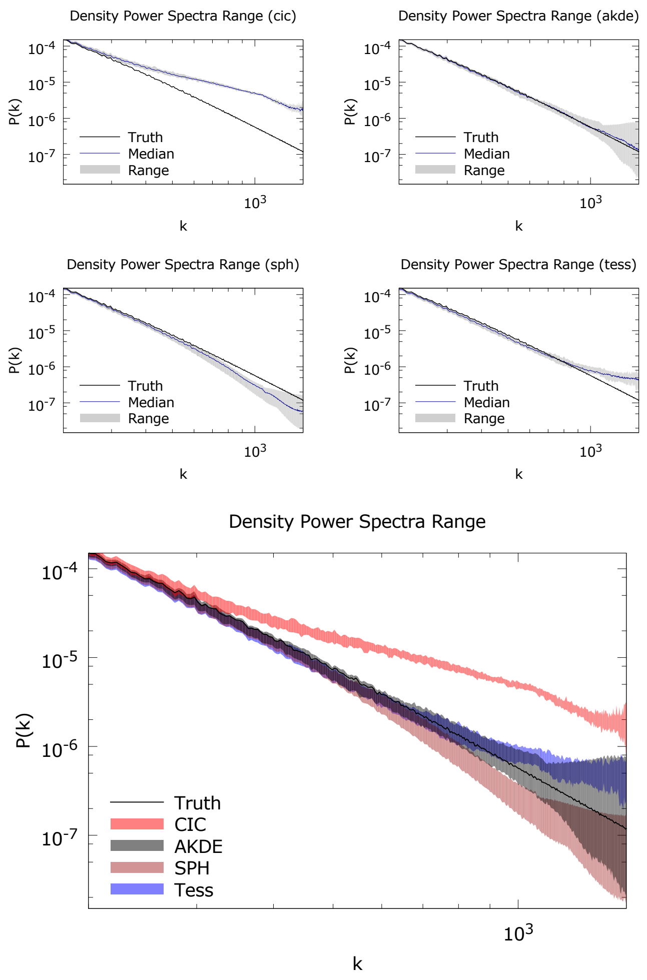

Figure 2: Spectral analysis of density estimation methods

Thoses results clearly shows the noisiness of CIC. As for the other methods, they give different shape of results with different level of variability while not showing one as clearly better than the others.

Tess-Dense's slight noisiness might be related to the use of flat weight distribution among each cell's inner grid point. Secondary methods used for weight distribution in dense area are also subject to some aliasing and therefore noise.

On the AKDE/SPH side, we mostly notice a high intrinsic variability. That may be caused by the sampling's random variation affecting the window size computation in such a way that we could locally encounter noisyness or oversmoothness.

4 Memory corruption study

Memory corruption related error can happen in different ways. Cosmic rays and radiation have, for a long time, been suspected of creating random data . More rescently we realised that thoses error can also happen because of hardware issues.

Beyound the question of those corruptions causes, we are here focussing on the impact such event have on our pipeline.

4.1 Error classification

Depending on a large number of factor, memory corruption can have very different concequences. Such factor include many things from hardware components (ECC memory one of the most well know mecanism against memory correption) to software certification and computation redondancy.

As for the consequences, they can be divided into two different categories :

Hard errors: Memory corruption affect the system of the program flow will most likely cause dramatic errors such as the program stopping abruptly or the whole system failling. In such cases we do not get any results back, redering irrelevant the question of the result's validity. Some memory corruptions affecting critical data such as table indices also falls into this category.

Silent errors: Memory corruption in some application's data may not cause any crash of the application while affecting the results if not corrected by the hardware. This is particularly the case of large array's contents like particles positions in our pipeline.

In this section we will try to caracterise silent errors' impact on our pipeline's results.

4.2 Corruption injection

Simulating random memory corruption can be done by voluntarly modifying our applications data by randomly flipping bits. While this isn't hard to do, studying silent errors means we have to enshure that those random bit flips will not cause hard errors.

Differents software quality implies different level of tolerence. For exemple some code can handle particles positions being outside of our considered domain while other might crash. Form here on we will mostly focus on our AKDE implementation as it very resistant to such errors and therefore more suitable to create and analyse silent errors.

rderror
io for
s ?

In our model, simulating memory corruption is achieved by randomly modifying our input data set. In real application this dataset would be provided by another program and would have therefor been send through the network of saved on disk, increasing the probability of memory corruption. This injection model will be controlled by two parameters, on the one hand the number of bit flips and one the second hand the weight of the potentially affected bits.

While the first parameter is used to simulate different degre of corruption, the second one is used to study the impact of differents bit flip position. The construction of IEEE floating point arithmetics [Kahan, 1996] (IEEE 754) is such different bit modification produce different arithmetic modification.

Studies have shown that in some HPC pipelines, some bits' position are critical, the modification of those bits resulting in hard errors while some other bits' modifications are un noticeable as the resulting modification are bellow the accuracy of floating computation.

ref -
Leonar
tech
report

4.2.1 Single error injection

As a first step in our analysis of memory corruption's impact on density estimator we will study the impact of single bitflip positions.

We are therefore going to modify, for various samplings of our density function, the value of one randomly selected floating value by flipping it's n-th bit. For single precision floating values this n value varies between 0 and 31 as simple precision floating value are 32 bits long. Once this mofication has been done, the corrupted sampling is processed by the pipeline and compared to uncorrupted results.

This single bit flip injection experiment gives expected yet interresting results.

Injecting a single bit flip moves on particle by modifiying one of its floating coordinates. While some bit's position only have a small impact, other can have an impact on the pipeline. Modification of the exponent bits can make the affected particule exit the donsidered domain, which some code cannot handle. As a consequensed, some bit flips cause Tess-Dense to crash. Other code, like our AKDE implementation can handle particle exiting the domain and, for such corruption in fact produce silent errors.

The first conclusion that some specific memory corruption produce hard errors due to bad coding practices. Those same memory corruption can in fact be detected inside the process by checking that the data verify some specific criterion.

Once the corrupted sampling processed using AKDE, the resulting density could not be distinguish for expected results ass they where whitin the range of expected results. This was to be expected has a single memory corruption could be seen as a verry slight modification of one of many particles in the intrinsically random sampling and therefore be statically indiscernible.

4.2.2 Multiple error injection

As the first approch showed us that single errors are indiscernible, we will now study the impact of the memory corruption rate on result's quality. For that we will inject large number of error, randomly distributed throughout

	Analytical	CIC	AKDE	SPH	Tess-Dense
Analytical	–	0.00314	0.00049	0.00214	0.00194
CIC	0.00314	–	0.00338	0.00513	0.00435
AKDE	0.00049	0.00338	–	0.00179	0.00151
SPH	0.00214	0.00513	0.00179	–	0.00099
Tess-Dense	0.00194	0.00435	0.00151	0.00099	–

Table 1: Distance between methods mean results

out input data. Our input data are sampling containing : We are here going to inject various number of bit flips

$$\begin{aligned} 2 \times 10^5 \quad \text{particles} \quad &= \quad 6 \times 10^5 \quad \text{floats} \\ &= \quad 1,92 \times 10^7 \quad \text{bits} \end{aligned}$$

(up to 10^6 independant bit flips) and then evaluate the difference to the expected range.

Figure 3 shows the density field computed by AKDE after injection of random bit flips.

Beyound the small noise on the sides, made visible by the density log scale, an cross is visible at the center of the domain. This cross is caused by bit flips in exponent part of floating point values, making them converge toward 0. We therefore have a high density around plane $\mathcal{P}_{x=0}$, $\mathcal{P}_{y=0}$ and $\mathcal{P}_{z=0}$.

Figure 4 shows the power spectrum of those density fields.

Those results both show a discrepancy between corrupted pipeline and expected results for error numbers between 10^4 and 10^5.

Using the power spectrum comparaison metrics (see equation 2) we can see the variation of the distance between analytical and corrupted results as a function of the corruption rate. Those results are visible in figure 5. This figure also shows, for information, an horizontal line showing the distance between Tess-Dense and AKDE mean results. This line therefore represent the threshold of variability between methods. Curve points bellow this threshold represent input set for which the effect of data corruption is smaller than the difference we can expect between different methods.

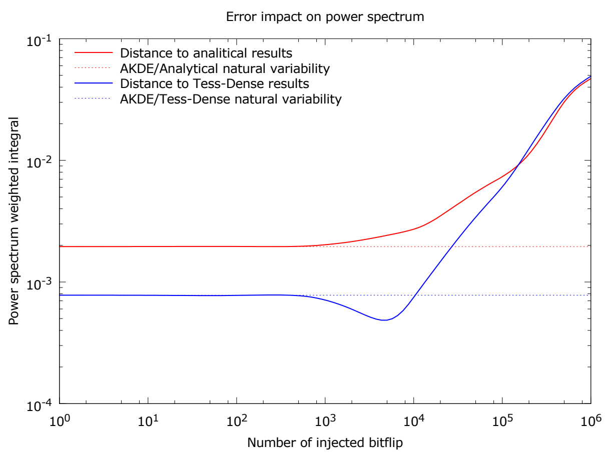

Figure 5: Power spectrum displacement

That shows use that corruption could only be detected using redondent computation with a different method if about 2.5×10^4 particles (or more) are affected, as we this distance has to be twice the inter-method threshold to be clearly noticeable. Compared to our 1.92×10^7 bits input data size, this gives us a detectable corruption rate thresold of about 0.13%

5 Conclusion

...

to writ

References

[Birdsall and Fuss, 1969] Birdsall, C. K. and Fuss, D. (1969). *Cloud-in-Cell Computer Experiments in Two and Three Dimensions.*

[Heidenreich et al., 2013] Heidenreich, N. B., Schindler, A., and Sperlich, S. (2013). Bandwidth selection for kernel density estimation: A review of fully automatic selectors. *AStA Advances in Statistical Analysis*, 97:403–433.

[Kahan, 1996] Kahan, W. (1996). IEEE standard 754 for binary floating-point arithmetic. *Lecture Notes on the Status of IEEE*, (May 1995):1–23.

[Navarro et al., 1996] Navarro, J. F., Frenk, C. S., and White, S. D. M. (1996). The Structure of Cold Dark Matter Halos. *Astrophysical Journal*, 462:563.

[Parzen, 1962] Parzen, E. (1962). On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33:1065–1076.

[Rosenblatt, 1956] Rosenblatt, M. (1956). Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27.

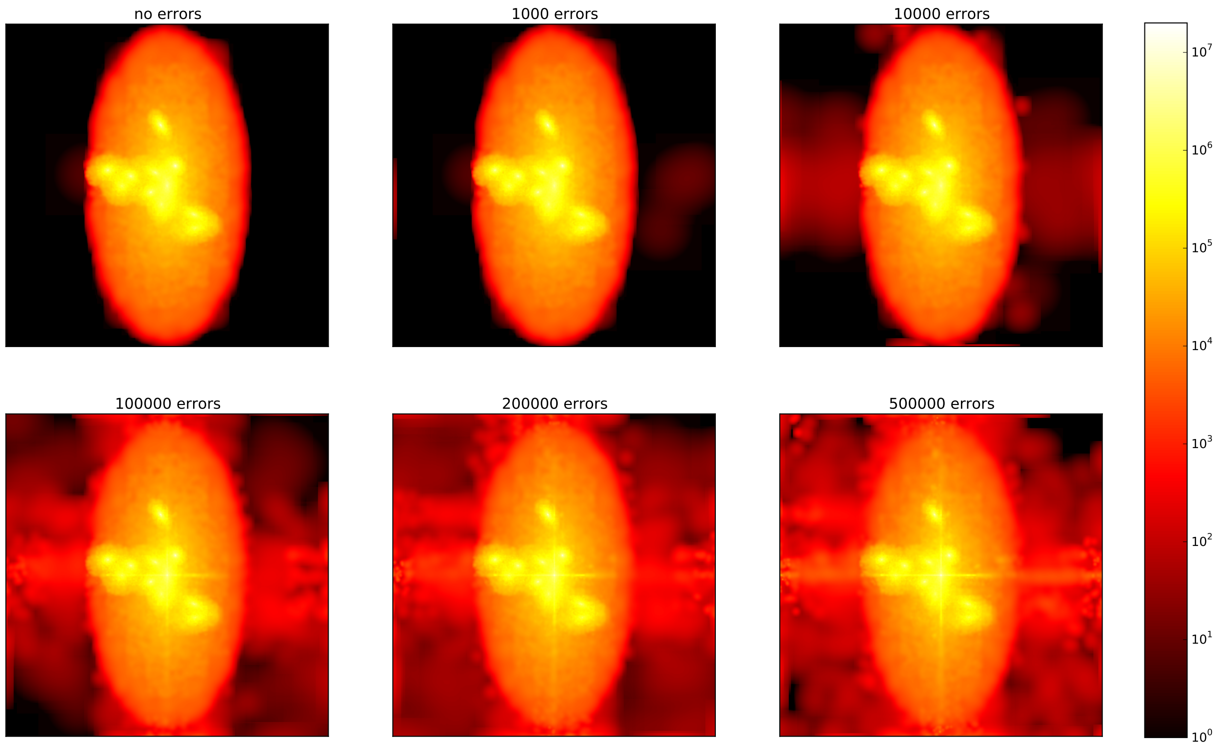

Figure 3: AKDE density fields after error injection

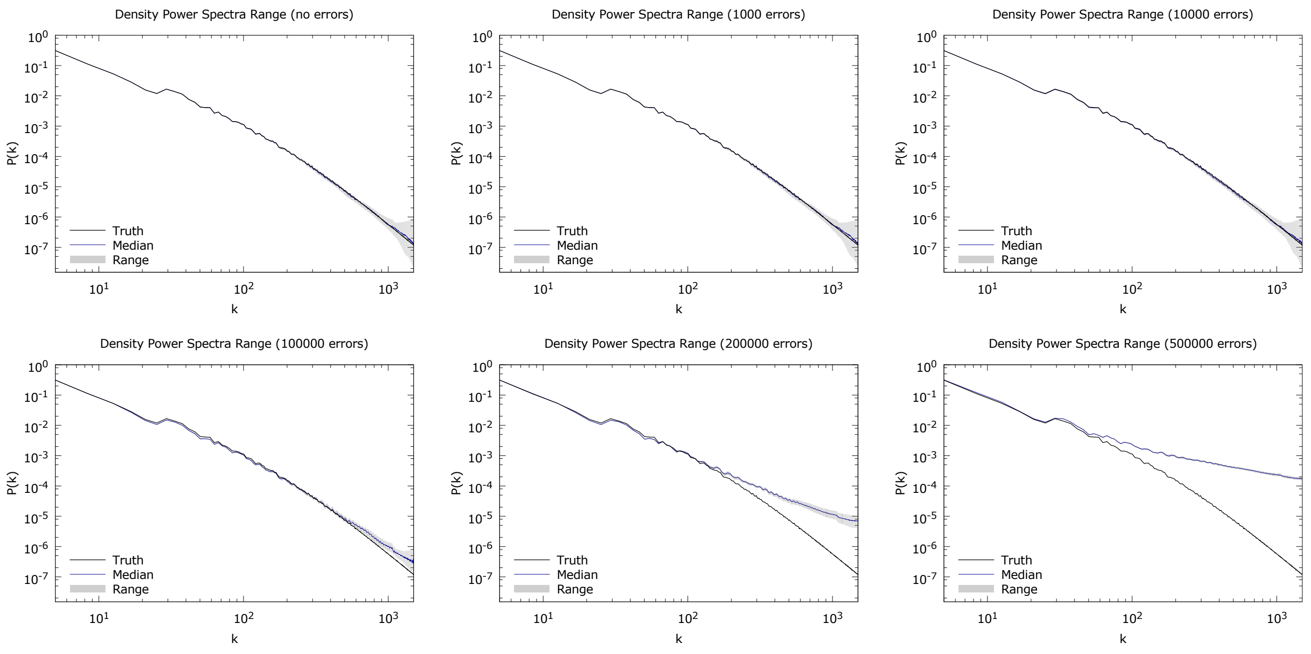

Figure 4: Bitflip influence on AKDE power spectrum range