UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Implementation and evaluation of a particle mover using mixed precision arithmetic

DEGREE FINAL WORK

Degree in Computer Engineering

Author: Francisco José Palacios Márquez
Tutor: Stefano Markidis // José Miguel Alonso

Course 2018-2019

Abstract

Computer simulations are broadly used nowadays in order to obtain information that would be impossible to gain otherwise. These computational workloads have grown in size to the point where even a small improvement in its implementation can lead to a substantial speed-up. For this reason, researchers have studied the impact linked to the usage of less precise numbers, since it would accelerate calculations. Likewise, this study aims to determine the feasibility of using mixed-precision arithmetic in iPIC3D, a 3D implicit Particle-in-cell (PIC) implementation used in plasma simulations. Specifically, the use of mixed-precision numbers will be limited to the particle mover, the section that solves the equations of motion for each particle of the plasma, resulting in the most time-consuming part of the code.

The results show a maximum divergence or error of about 2% between the original implementation and the new one when comparing the output values. All of this, performing a relatively short simulation of 2250 cycles, therefore, with longer tasks we the error could increase. Thus, we come to the conclusion that in most cases, the loss in precision is too high to justify the use of this new implementation.

Acknowledgments

I would like to acknowledge everyone who played a role in the fulfillment of this thesis, from my supervisor and examiner to everyone who provided feedback to improve upon it.

But, I wanted to specially acknowledge Pablo, without whom I would have never been able to finish this project.

Contents

Introduction

Computer simulations are extremely powerful tools that have been used by the scientific community for a long time. The main reason behind their popularity is that they provide knowledge that would be impossible to gain otherwise.

These simulations have increased in size and complexity, from simple interactions between atoms [1] to complex systems like liquids [2] or, like in this study, plasma [3]. Thus, the power required to recreate these conditions is higher than ever before, resulting in the mandatory use of parallelism in order to be able to complete these workloads in a reasonable amount of time. The computations are typically conducted using double precision floating point numbers in most cases [4] due to their rigorous nature. We aim to study if we can use a combination of both single and double precision in one of these critical simulation workloads without loosing accuracy.

iPIC3D is a PIC implementation used in particle simulations [5], that will provide us valuable information since it is properly tested and therefore, a reliable source of data.

0.1 Research Question

In this project we aim to answer:

How is iPIC3D's particle mover affected by the precision of the numbers used?

0.2 Initial hypothesis

Even before the study, we can hypothesize that the reduction in precision is going to have an impact in the final results, both in performance as well as reliability. The aim of this thesis is to evaluate if this impact is negligible enough, enabling us to improve the performance of iPIC3D's simulations.

0.3 Scope

This project has a limited scope, focusing only in iPIC3D and specifically, its particle mover. Due to the dependency between our results and iPIC3D's implementation we will not be able to generalize our results to any other simulation software.

On the other hand, due to the deterministic nature of iPIC3D, the results obtained in this project could, in theory, be generalised to some extent when performing similar simulations.

Background

In this section we will provide some context and explain some specific terms that may be required for any reader that is not familiarized in the field of study.

0.4 High Performance Computing (HPC)

High-performance computing refers to the use of multiple computers, usually clusters, which take advantage of parallelization techniques in order to perform calculations that would be impossible otherwise.

The term is often used interchangeably with super-computing and broadly popular in computer science since its use is required in certain fields like molecular simulations[6], cryptography [7], etc.

0.5 Computer Simulation

Computer simulations aim to mimic a given physic system making use of computers following a certain mathematical model. This model describes the behaviour of the physical system in a way that makes possible estimate with meticulous precision its progress.

Simulations have been used in multiple fields, as we mentioned in the introduction. Because of its complexity, is one of the fields that would benefit the most if we could make use of mixed precision numbers in its calculations, accelerating a great amount of heavy workloads that require a long time to be completed.

0.6 Floating-point format

In order to be capable of using real numbers in computer calculations, a new way of representing them with binary digits had to be established. At first, each manufacturer used their own representation, shortly after however, it became apparent that a standard was needed in order to improve portability and save resources. This standard [8] was created back in 1985 by the Institute of Electrical and Electronics Engineers (IEEE) in the document known as "IIIE754".

IEEE754 defined floating point numbers as a composition of three fields. The first one is 1 bit that indicates the sign, the second is a biased exponent and the last one is the significand of the number. The length of both the exponent and the significand, also known as mantissa, depends on the precision used to represent those given number, the more bits used the more precise our representation of that number will be.

These are the most common formats that were defined in IEEE754:

- Single precision:

  In single precision 32 bits are used. The first one is always the sign, the next 8 bits are the exponent which bias is 127 and, finally, the mantissa is 23 bits long. This method of representation provides a precision of roughly 7 decimal digits.

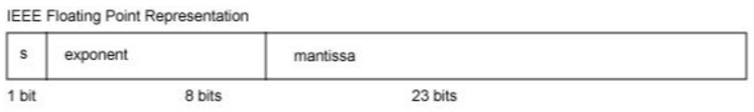


Figure 1: Single precision representation format

- Double precision:

  Double precision numbers are represented using 64 bits. Just like in single precision, the first bit is the sign. The exponent however, is different (1023) and is represented with 11 bits. Finally, The mantissa is also longer, specifically 52 bits, which provides roughly 16 decimal digits of precision.

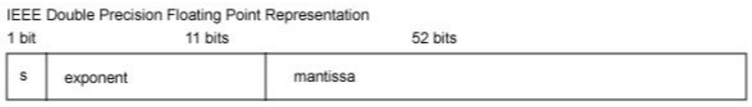


Figure 2: Double precision representation format

0.7 PIC: Particle-in-cell

PIC has been extensively explored since the 50s when it was designed [9]. One of the best explanations about this method can be found in a paper written by D. Tskhakaya, K. Matyash, R. Schneider and F. Taccogna [10]. In that article it is explained that PIC is a widely used method in simulation software that allows the program to recreate the progress of certain particle systems using discretization of time and space and interpolation . It is mostly used in plasma simulations[11] but it also can be applied in some other fields like quantum physics[12] and pure electromagnetism [13]. The goal of PIC is to keep track of each individual particle and use their position and velocity to calculate the macro-quantities of the whole system at each step of the simulation. In order to achieve this, the computer solves the equation of motion for every particle and time step.

$$\frac{\partial \vec{X}_i}{\partial t} = \vec{V}_i \tag{1}$$

$$\frac{\partial \vec{V}_i}{\partial t} = \vec{F}_i(t, \vec{x}_i, \vec{v}_i, A) \tag{2}$$

Figure 3: Motion equations for a given particle i

In the previous equations L_1 and L_2 are some operators, A = L_1(B) is a macro-field acting on particles, B $L_2(\vec{X}_1, \vec{V}_1, ..., \vec{X}_N, \vec{V}_N)$ for i = 1, ..., N is a macro-quantity associated with particles and $\vec{F}_i$ is the force acting on a particle i.

Further details about PIC implementations applied to plasma and electromagnetic fields (using MaxWell's equations) can be found at the previously mentioned article[10], delving into the equations is out of the scope of this project.

0.8 Particle mover

A particle mover is the part of the code that follows the trajectory of the particles during a particle-in-cell simulation. Due to the huge amount of particles found in a real system, it is not possible to simulate every one of them. Instead, every simulated particle is assumed to be a group of physical particles called super particle.

Superparticles keep the same properties than the real particles they are supposed to represent, the ratio charge/mass is the same and thus, the trajectory it follows is also identical. This reduction in the number of particles allows much more complex systems to be simulated. However, there are cases where even the number of super particles is extremely high, thus, optimizing this part of the code is crucial in any simulation software. There have been some improvements in the algorithm over the years [14] and any increase in performance is welcomed, so much so that this need for optimization motivates this whole project.

0.9 iPIC3D

iPIC3D is a parallelized, implicit PIC implementation used in computer simulation of plasma [5] in 3 dimensions. From the software point of view, it is written in C++, with a complex structure and around 20000 lines of code. From the scientific point of view, is also pretty complex, it makes use of the Vlasov [15] and Maxwell equations in order to describe the evolution in time of a distribution function of plasma particles. That equations system is then solved using the PIC method previously described (figure 3).

Let E and B be the electric and magnetic fields, respectively, v the particle velocity, r its position, q and m the charge and mass, respectively:

$$\frac{\partial f}{\partial t} + v \cdot \frac{\partial f}{\partial r} + \frac{q}{m}(E + \frac{v \times B}{c}) \cdot \frac{\partial f}{\partial v} = 0 \tag{3}$$

$$\rho = \sum q \int f dv \tag{4}$$

$$J = \sum q \int v f dv \tag{5}$$

Figure 4: Vlasov, charge density and current density equations, respectively

The way iPIC3D simulations workflow is always the same, the particles are initialized following a distribution function, after this, the parameters r, v, B, E are set as established in the input file and finally, it solves Maxwell and motion equations for each particle as many cycles as it is supposed to. Alternatively, the simulation can be finished after a certain amount of time.

0.10 Related Work

There have been several studies regarding the use of mixed precision arithmetic in sensitive workloads, therefore, we will only cover the most relevant and closest to our own project.

Performance and accuracy of mixed-precision solvers

In this study [16], they compared double precision solvers with emulated (two single float numbers) and mixed-precision solvers. The performance advantages were described as speedups over native double precision code and also as reductions in memory use.

The conclusion reached is that mixed precision works nicely when using parallelization, beating double precision by a factor of 4-5 in time performance and 3-4 in memory use, all of this without losing any significant amount of accuracy compared to a double precision execution.

Hybrid CPU-GPU PIC implementation

The paper[17] describes an efficient, mixed-precision hybrid CPU-GPU implementation of an implicit 1D PIC method that was proposed in 2011 [18]. The method uses a JFNK [19] solver that is kept on the CPU in double precision while the particle mover is implemented on the GPU using single precision arithmetic.

The particle mover using a Nvidia GeForce GTX580 is about 100 times faster than using a single-core implementation in an Intel Xeon X5460. Likewise, they claim that with the test case they chose, the mixed-precision hybrid implementation performs about 100 times faster than the old implementation using only CPU and double precision arithmetic.

Mixed precision algorithms

This paper [20] describes how to improve some previously used algorithms using a combination of double and single precision arithmetic. It argues that in many cases, a single precision solution can be improved until it reaches double precision accuracy. Likewise, since 32-bit operators usually perform almost twice as fast than the ones using 64-bit arithmetic, some modifications can be made in several algorithms in order to get an improvement in performance while not losing any accuracy.

They provide some algorithms that have been modified with this idea in mind, for example, the LU factorization of the coefficient matrix using Gaussian elimination that is used in the solution of linear systems. Their results show a significant speed-up, from 1.5 to almost 11, varying greatly with the hardware and matrix used.

Finally, they provide some guidelines in how to apply their method to other algorithms in order to encourage more scientists to apply this knowledge.

Method

0.11 Objectives

Our approach in this study consisted in:

- Determine the inputfile used to run our tests (testGEM2Dsmall)
- Obtain the data and plots from the original implementation of iPIC3D, the one using double precision numbers.
- Locate the particle mover from the source code, finding the function that implements it.
- Change the variables used to represent position and velocity from double to float.
- Run the simulation again using the new implementation
- Compare the results

0.12 Data set

The inputfile used in both simulations is "testGEM2Dsmall.inp", a modified version from the Geospace Environmental Modeling (GEM) magnetic reconnection challenge [21] which is included in iPIC3D's source code and also can be found in the git repository.The only modifications done in the file are the output path and the variables that MPI uses to distribute the simulation across all cores. The file can be found in GitHub [1].

0.13 Test machine

The computer used in the study is an "HP Pavilion - 15-bc450ns" laptop. It integrates an i5-8300H CPU, 8 GB of RAM, and as the GPU, the 4GB version of the NVIDIA GTX 1050. The operating system is Windows 10, fully-updated as May 2018. From the native Windows OS, Virtualbox v.6.0.8 r130520 was used to launch an Ubuntu 16.04 virtual machine that has 4 GB of ram available.

0.14 Implementation

The particle mover is located inside the "particles" folder of iPIC3D, in a file called "Particles3D.cpp". Specifically, the method "mover_PC_Aos".The only required change in the

[1] https://gits-15.sys.kth.se/fjpm2/thesis/.

code is the transformation of the variables used in both position and velocity from "double" to "float". These changes are located from the lines 866 to 871:

```
double xavg = xorig;
double yavg = yorig;
double zavg = zorig;
double uavg_old, uavg = uorig;
double vavg_old, vavg = vorig;
double wavg_old, wavg = worig;
```

Figure 5: Original particle mover code

These lines of code are modified changing the variables type to float:

```
float xavg = xorig;
float yavg = yorig;
float zavg = zorig;
float uavg_old, uavg = uorig;
float vavg_old, vavg = vorig;
float wavg_old, wavg = worig;
```

Figure 6: Modified particle mover code

Finally, in order to obtain the plots used to compare both implementations, we will visualize the data that iPIC3D outputs making use of Paraview, a visualization software as well as Microsoft Excel for some additional plots.

0.15 Evaluation

We want to provide both qualitative and quantitative data. First, we will show the plots from both simulations at each step of the simulation, one next to the other, so the different quantities can be perceived in a graphical way. After that, we will show the plot obtained by subtracting both data sets, this will provide an intuitive and easy way to spot any difference between them. As for the quantitative part, we will plot the error in percentage of the macro quantities iPIC3D outputs. These quantities are dependant of the whole system, like the total energy of the system or the momentum and therefore, we come to the conclusion that they would be a good way to quantify the overall error.

```python
import vtk
input1 = self.GetInputDataObject(0, 0)
input2 = self.GetInputDataObject(0, 1)
scalars1 = input1.GetPointData().GetScalars('#VariableName')
scalars2 = input2.GetPointData().GetScalars('#VariableName')
auxArray1 = vtk.vtkDoubleArray()
auxArray1.SetName('Set Signed Err')
auxArray2 = vtk.vtkDoubleArray()
auxArray2.SetName('Set Rel Err')
auxArray3 = vtk.vtkDoubleArray()
auxArray3.SetName('Set Rel Err')
for i in xrange(input1.GetNumberOfPoints()):
    v1 = scalars1.GetValue(i)
    v2 = scalars2.GetValue(i)
    error = v1 - v2
    auxArray1.InsertNextValue(error)
    auxArray2.InsertNextValue(abs(error))
    auxArray3.InsertNextValue(abs(error)/abs(v1))
# Initialize the output and add the labels array
output = self.GetOutput()
output.ShallowCopy(input2)
output.GetPointData().AddArray(auxArray1)
output.GetPointData().AddArray(auxArray2)
output.GetPointData().AddArray(auxArray3)
```

Figure 7: Custom python filter used in Paraview in order to show the difference between our obtained datasets

Results and Analysis

We have created different types of plots from the data provided by iPIC3D. First, we plotted the state of both simulations side-by-side from cycle 0 to cycle 2250 with a step of 250 cycles, this will provide a general picture of both simulations. The parameters shown are E (Electric field), B (Magnetic field) and both components J of the charge density (Je and Ji). This decision was made because we thought they would be the most representative of all, since there were more parameters like several pressures that were difficult to understand. It should be mentioned that not every one of the plots will be shown in this section, only the ones from both electric and magnetic fields. This is due to certain problems we encountered with Paraview and space limitations as explained in the appendix, the rest of the plots are also provided in the appendix A. Lastly, all 4 plots follow the same pattern so we think that showing half of the plots is enough for the understanding of the results.

0.16 Electric field

Here are the plots that provide us with information from the electric field, the plots from the top represent the intensity of the electric field at each time step. The greater the intensity at a given point, the more red it will show in the picture. On the other hand, we can see at the bottom part the plot that shows the difference from the other 2. The greater the difference is in this case, the more white it will show.

From the cycle 0 to the 1500 we see no noticeable difference in any of the plots. However, from that point forward we can see that there are a moderate amount of points that show how the simulations are, in fact, not the same.

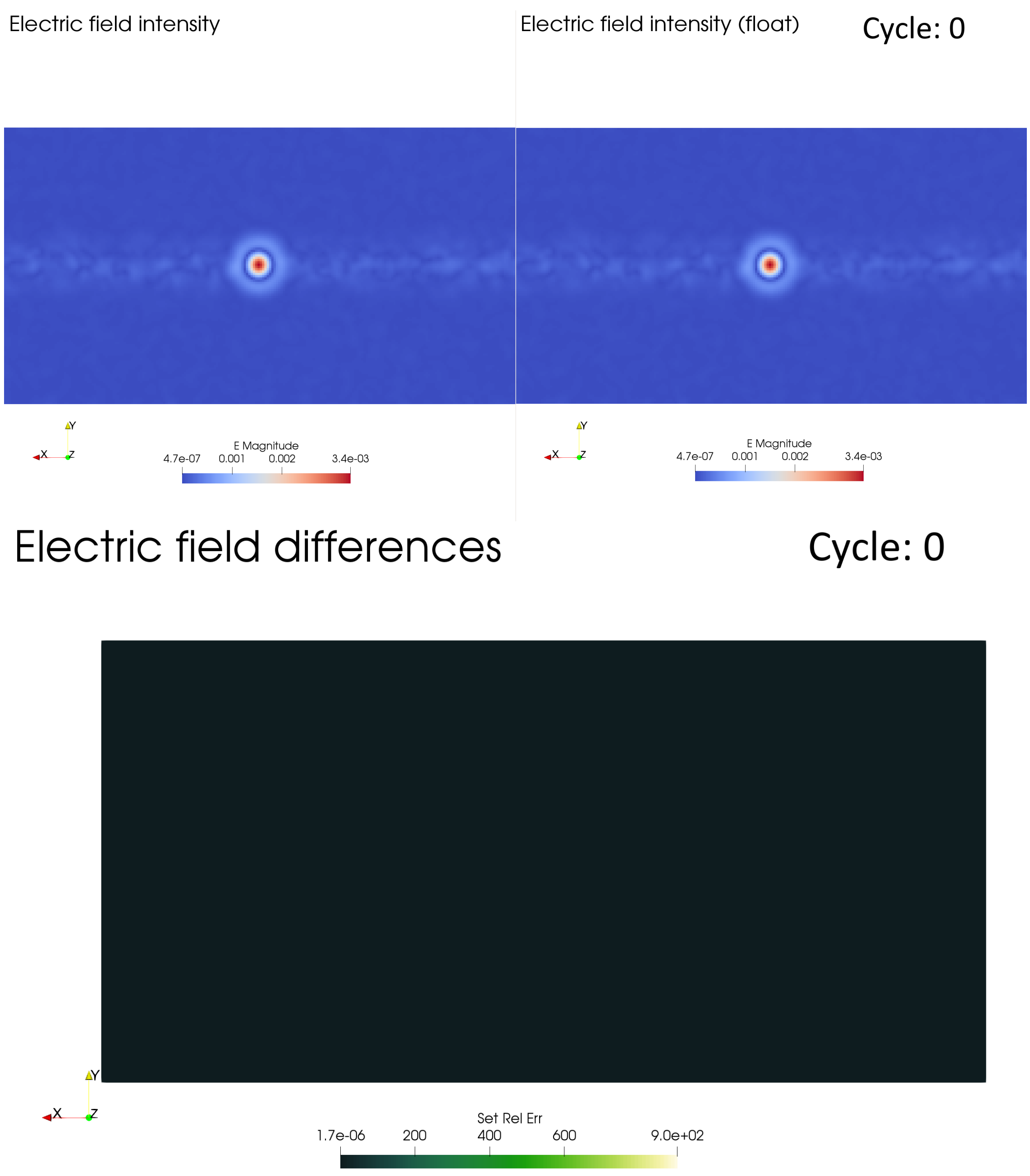
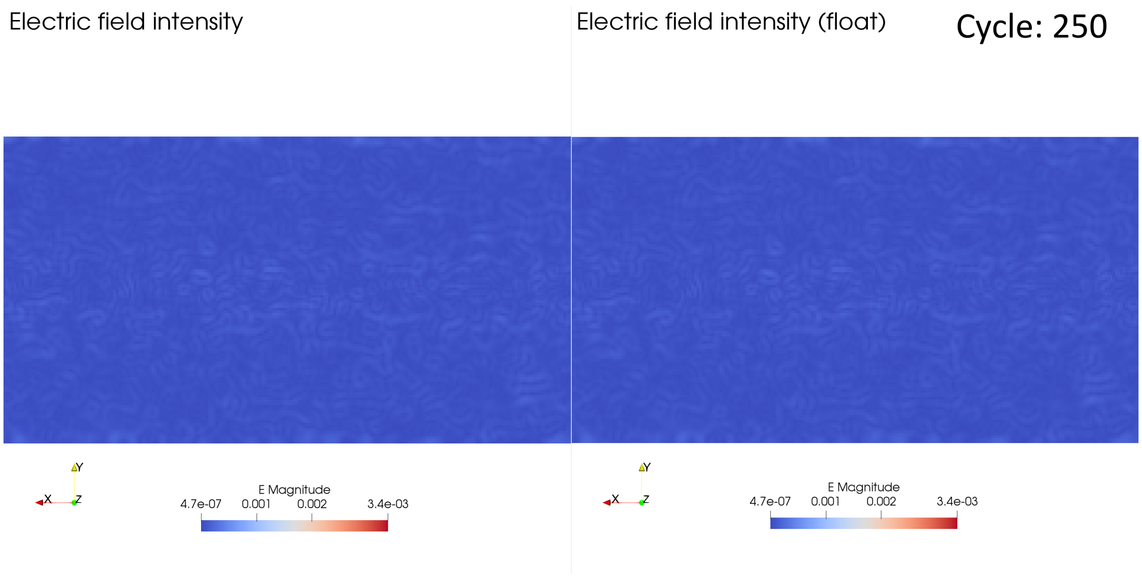


Figure 8: We see no difference between implementations at the initialization stage, since both simulations are performed with the same parameters. The second graph shows a perfectly black graph, showing that there is no error or difference.



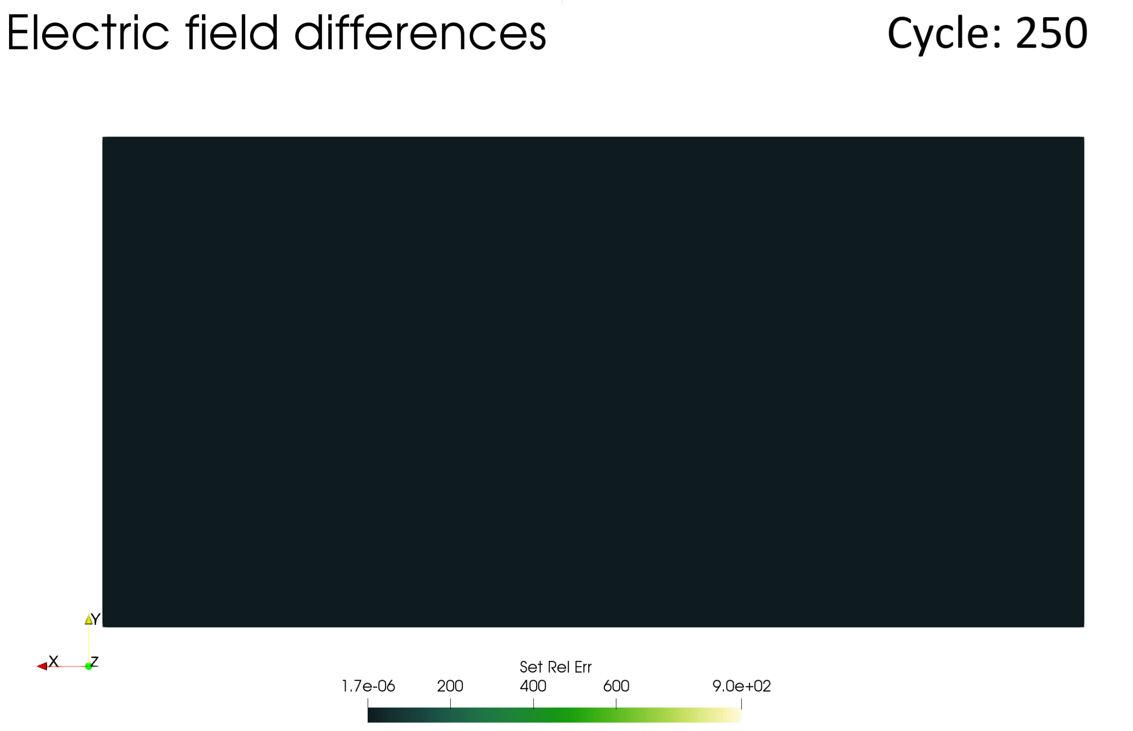
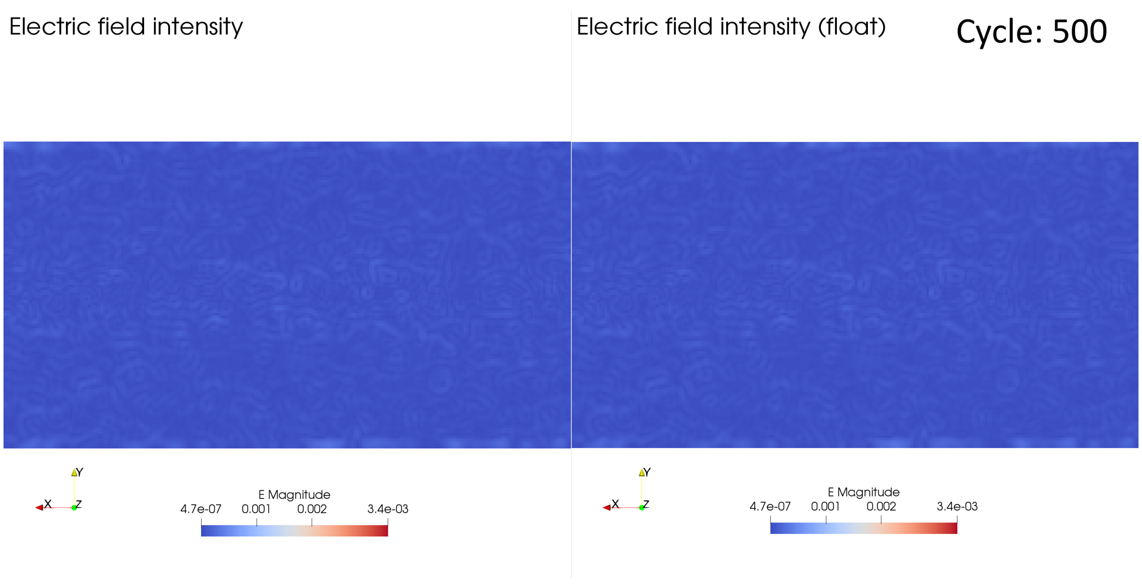


Figure 9: After 250 cycles, there is no difference shown in the graphs. This is more obvious when looking at the difference plot, where we can observe that the image is still perfectly black.



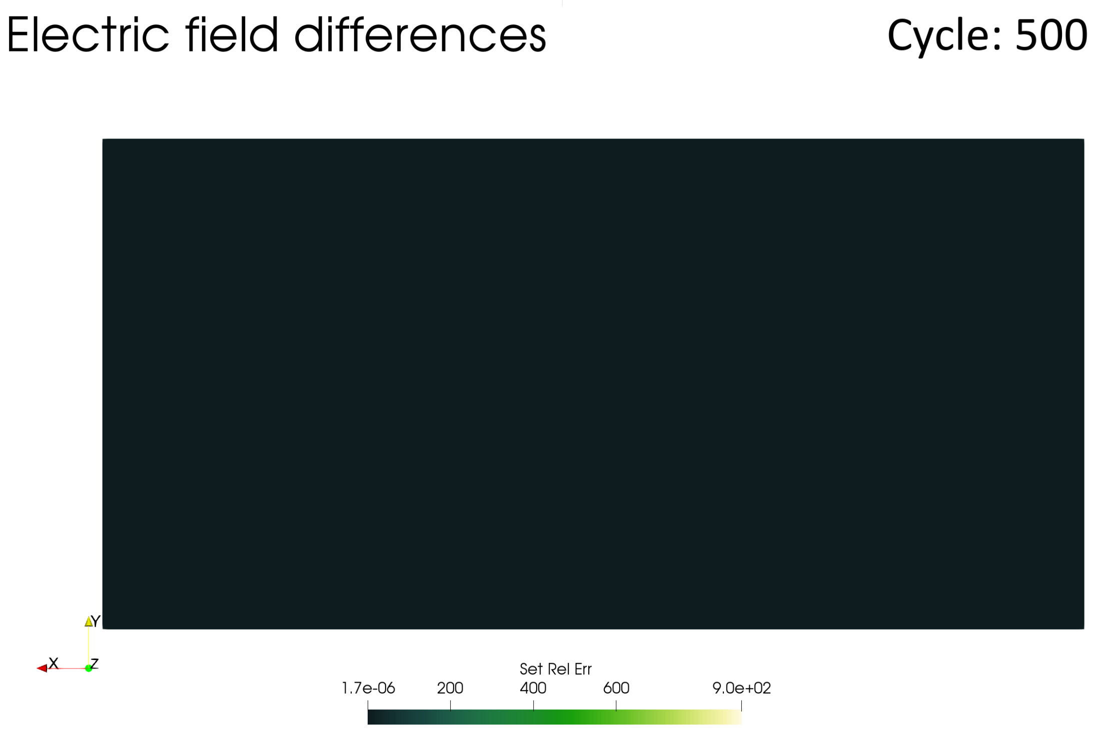
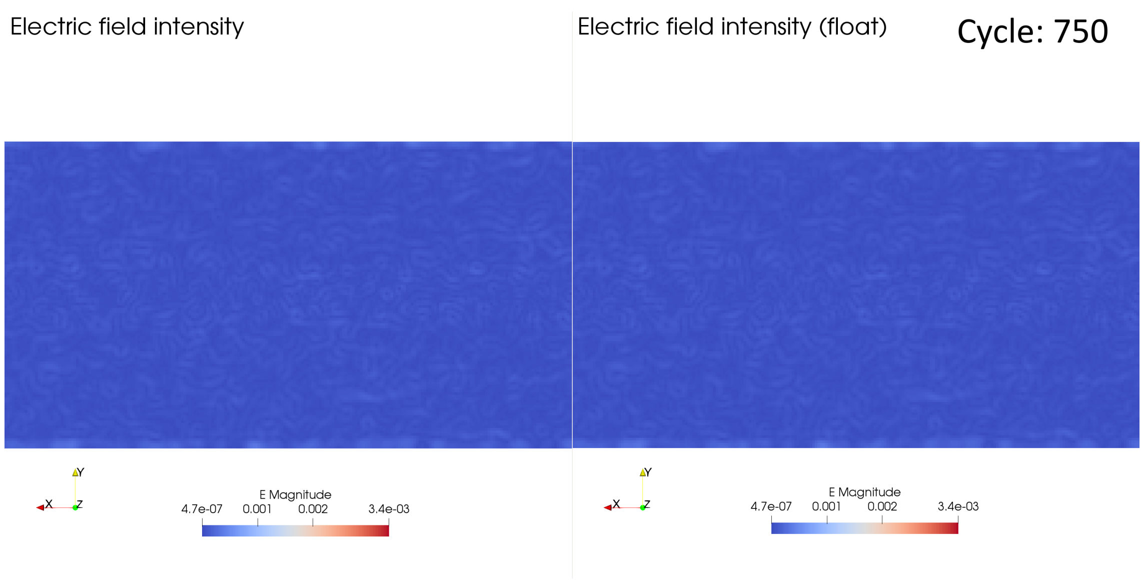


Figure 10: After 500 cycles, both simulations still are, from the electric field point of view, equal.



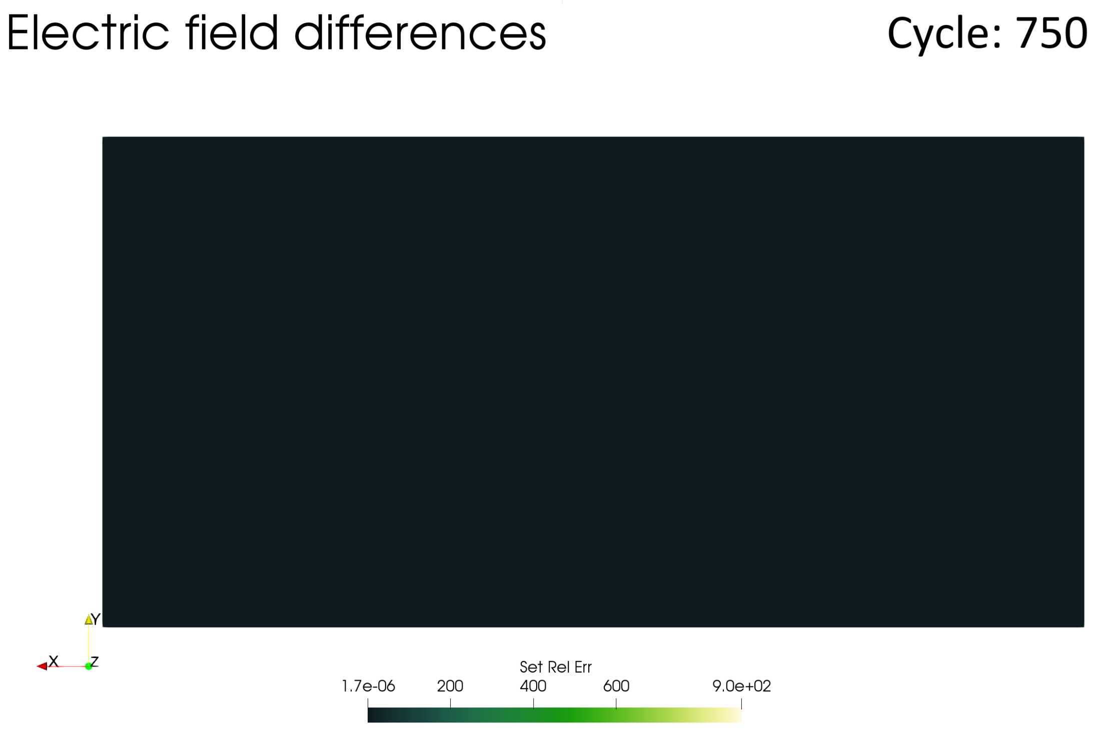


Figure 11: After 750 cycles, the same criteria still applies, showing no difference whatsoever.

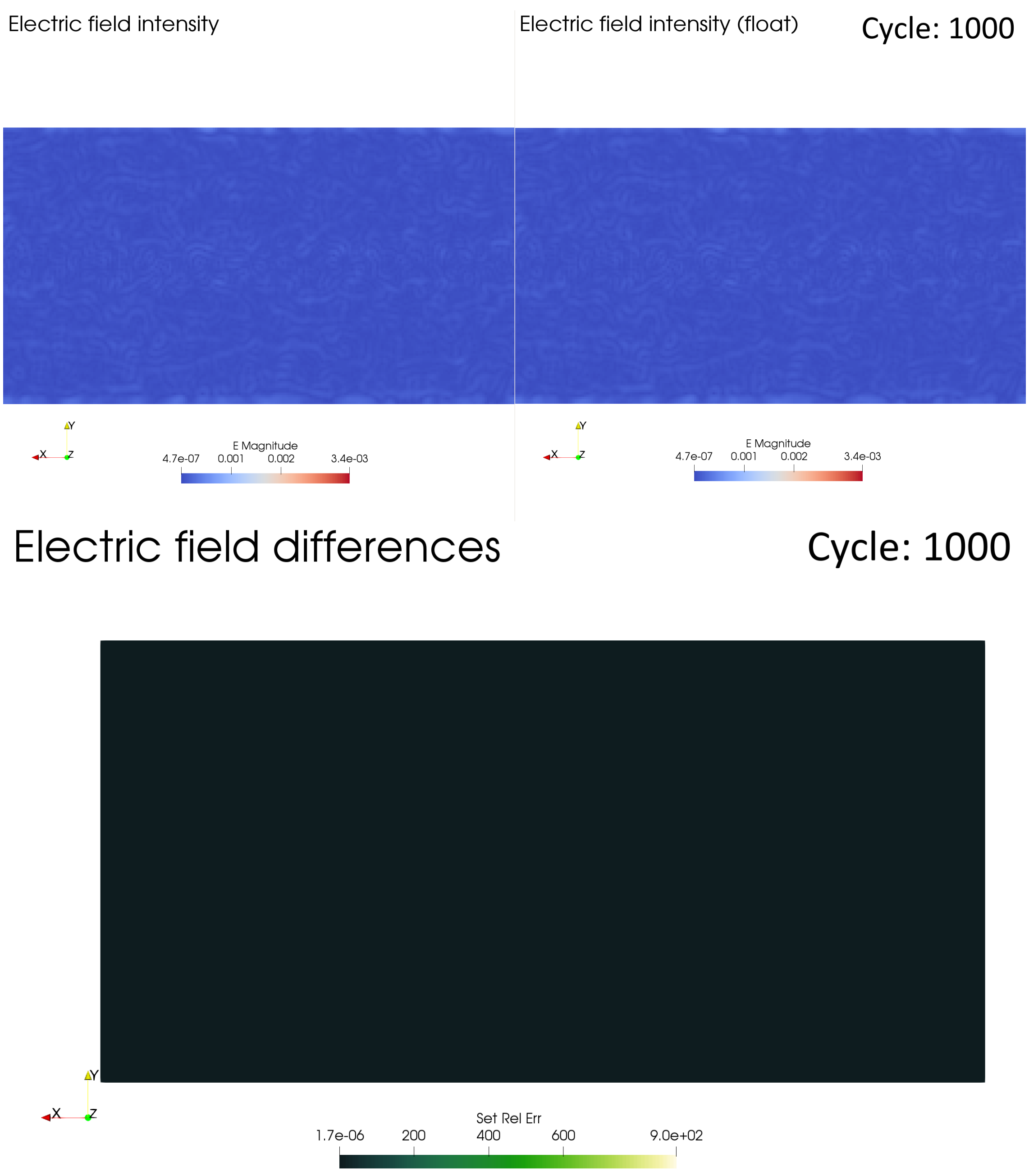
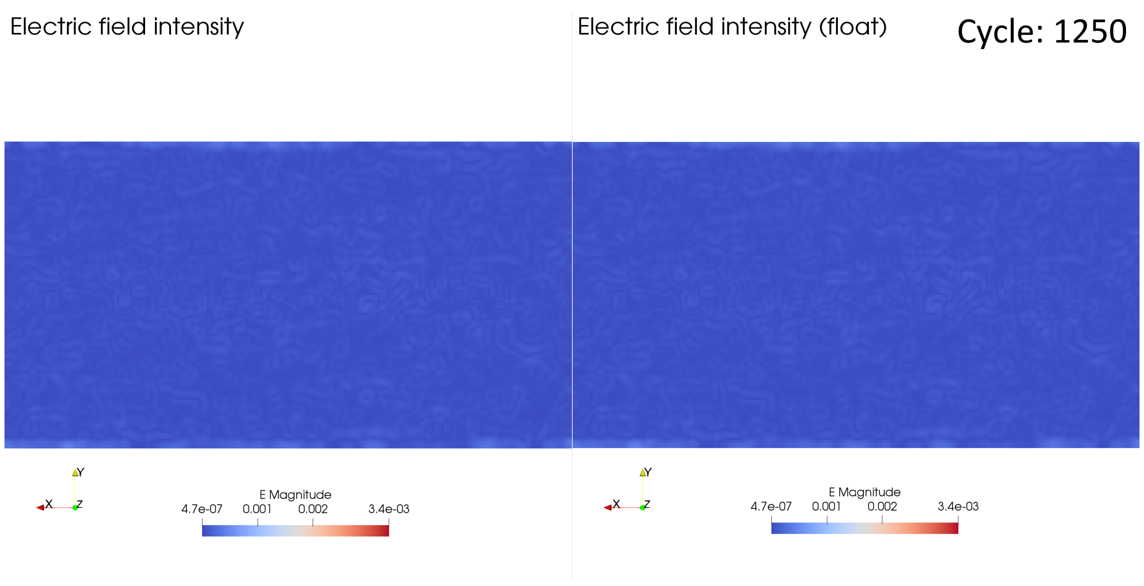


Figure 12: Even after 1000 cycles performed by the software, there is no difference between implementations.



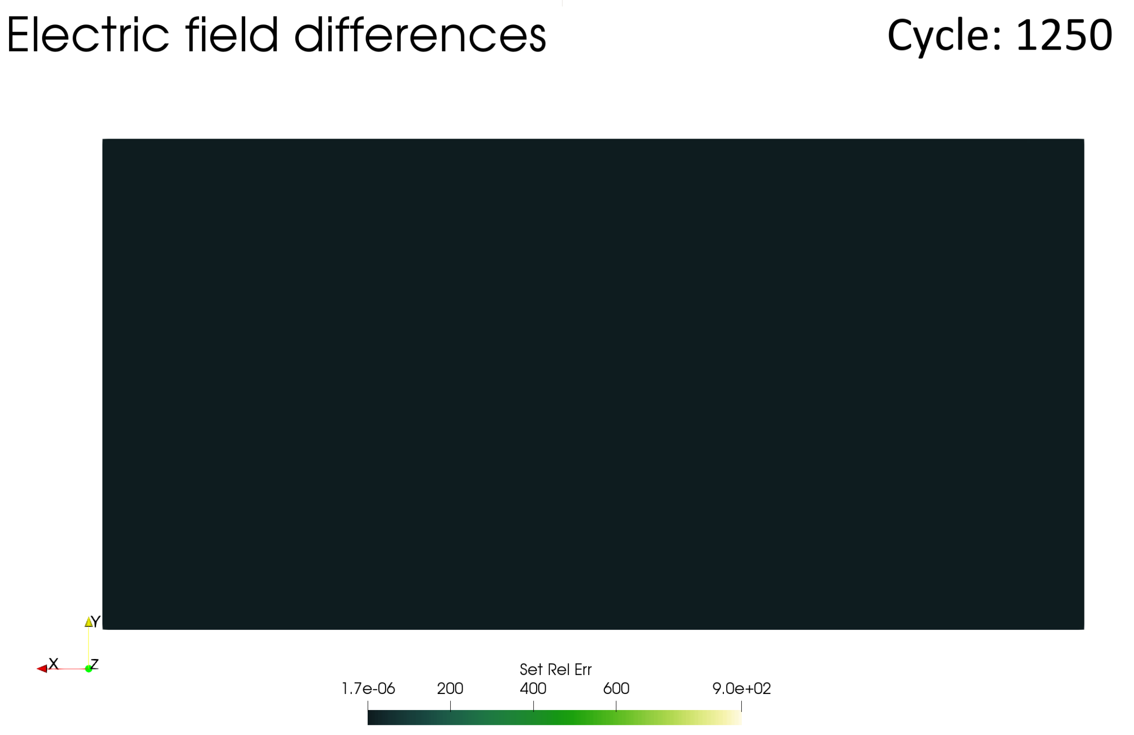
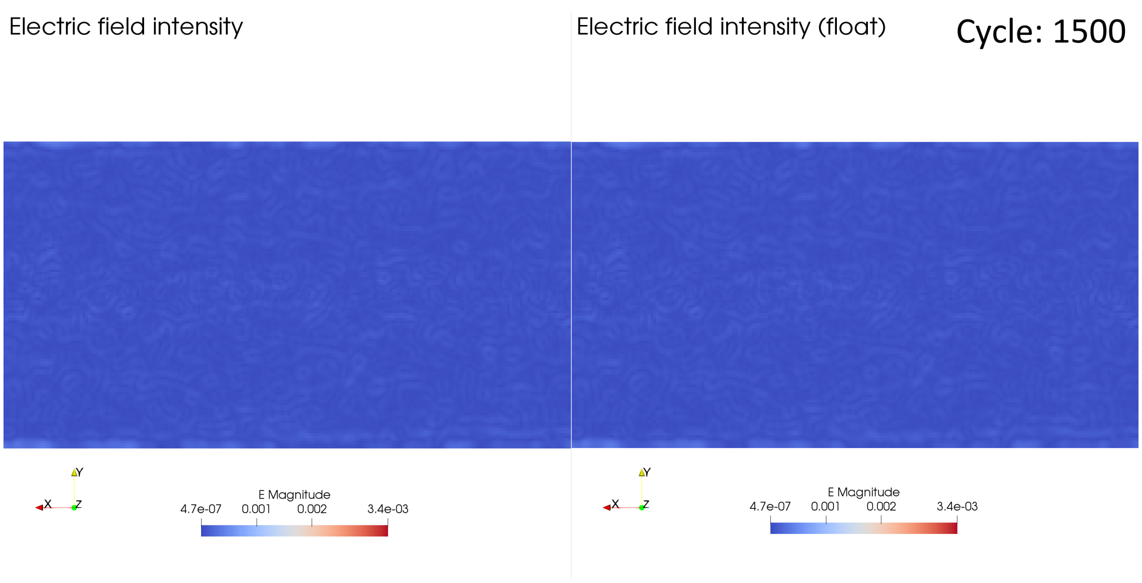


Figure 13: At cycle 1250, we still see no difference.



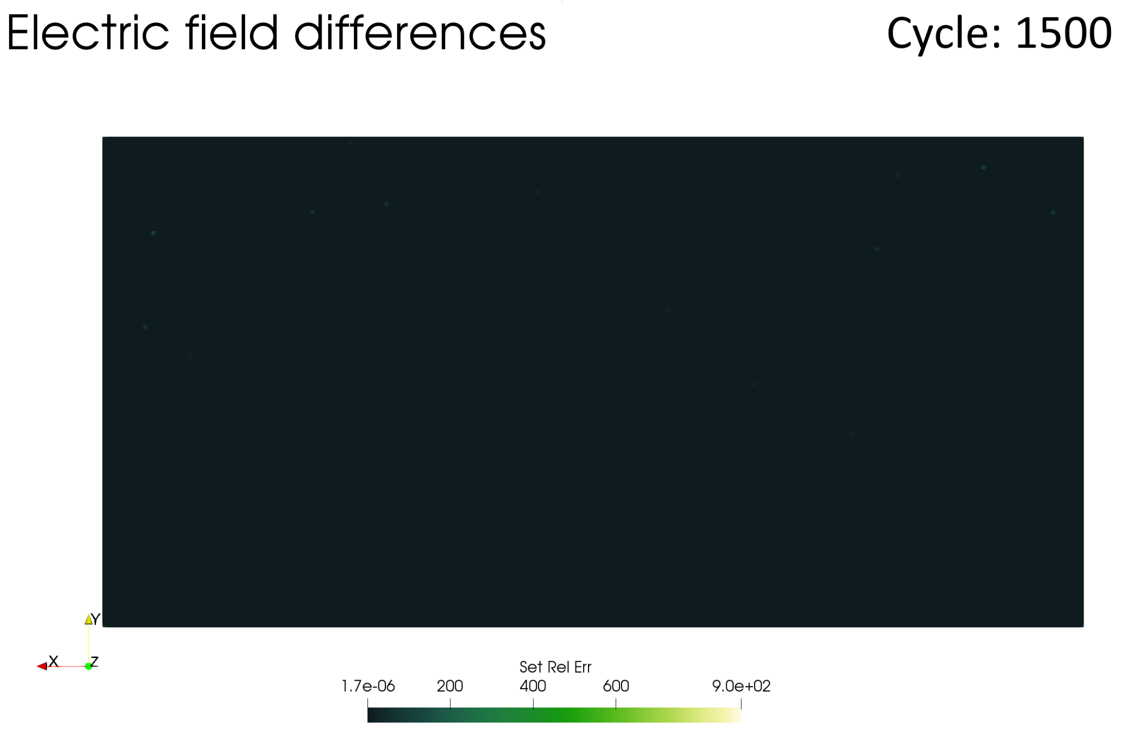
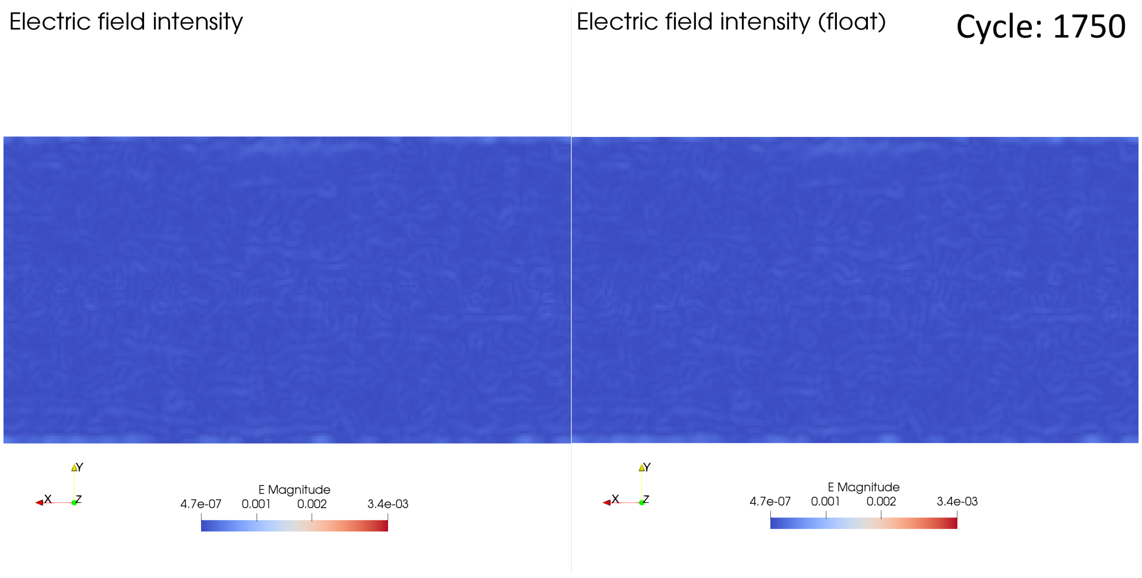


Figure 14: After 1500 cycles, finally, there is a difference, we can observe green dots at the top-left and top-right parts of the error chart. This means that, in those spots, the intensities of the electric field differ. Since the dots are isolated and dark, the error is relatively small.



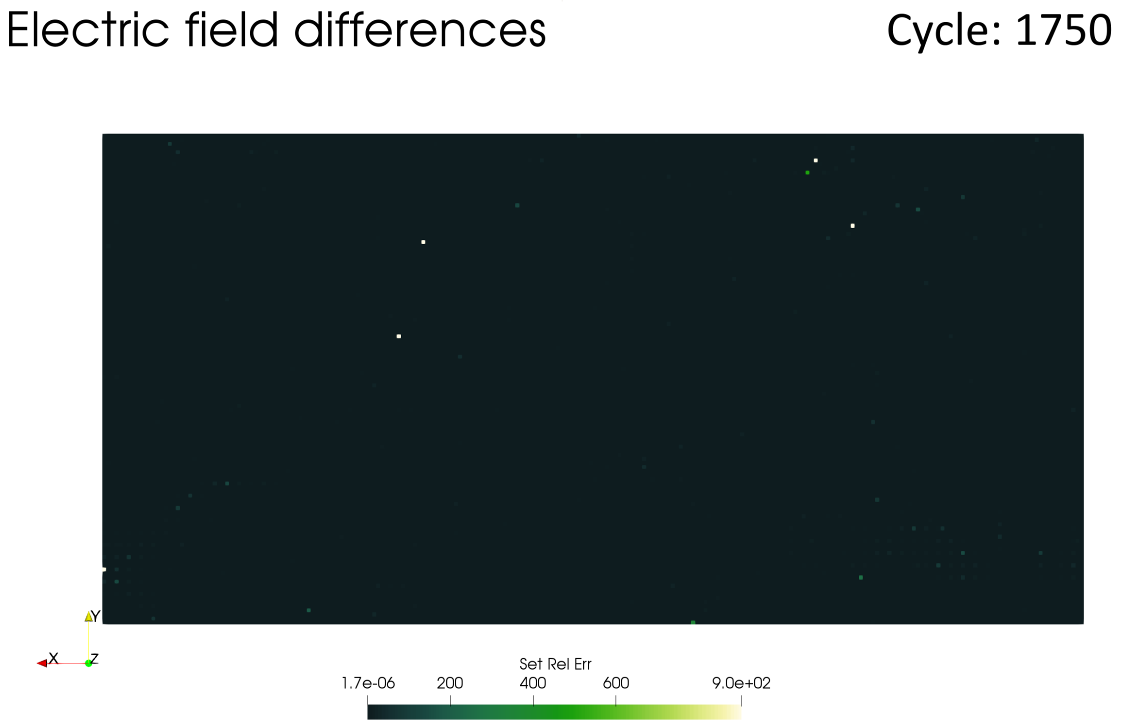


Figure 15: After 1750 cycles, the error increases significantly to the point where we see some spots that show a white color, this means that the difference at those points reaches the maximum error of 9.0e+2.

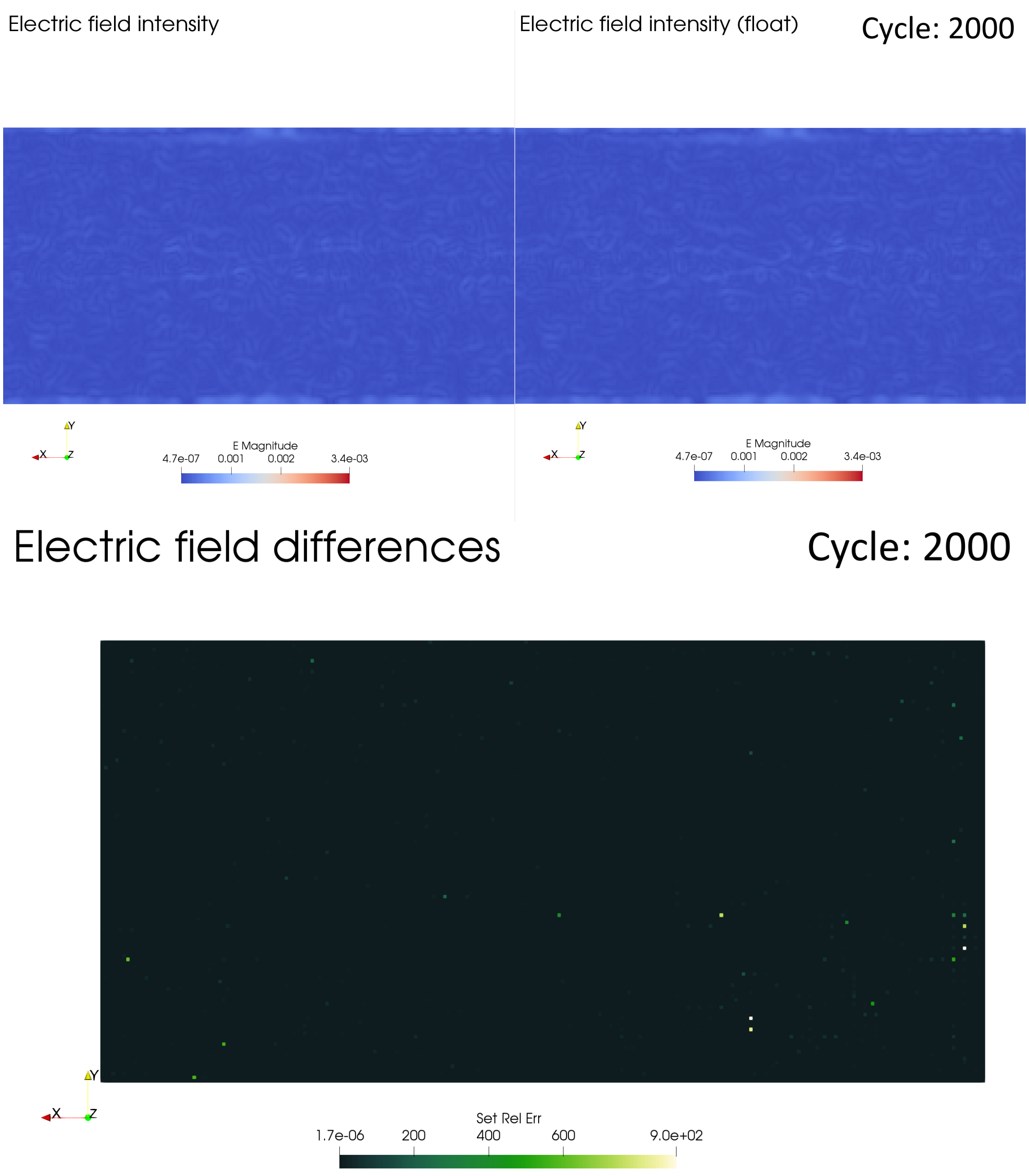


Figure 16: After 2000 cycles, there is an increase in the number of points showing a different intensity, and therefore, the error. The points are not the same as before, but the number of white and whiter points has increased overall, mostly in the bottom part of the graph.

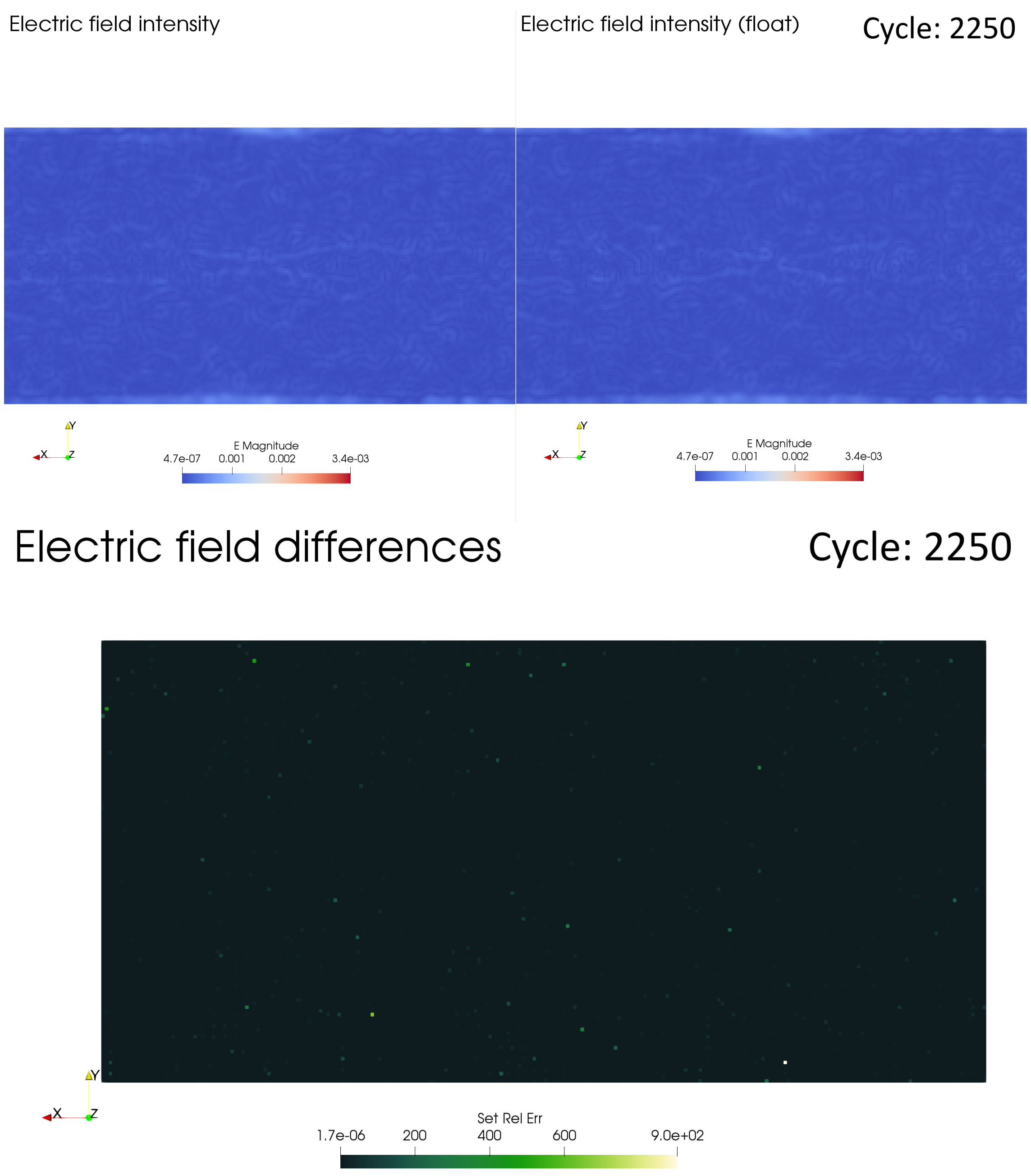


Figure 17: After 2250 cycles, the error seems to have distributed across the whole graph, showing multitude of points wit noticeable error.

0.17 Magnetic field

As for the magnetic field, we found a fairly similar behaviour, with no difference until around cycle 1750, from that point forward the differences begin to appear, although these differences seem smaller than in the electric field case. Is worth mentioning that in this case the difference is shown with blue and red colors in its respective plot and we also find some yellow lines that, from what we understand, are due to a problem with the filter used in Paraview and they should not have any impact in the overall results.

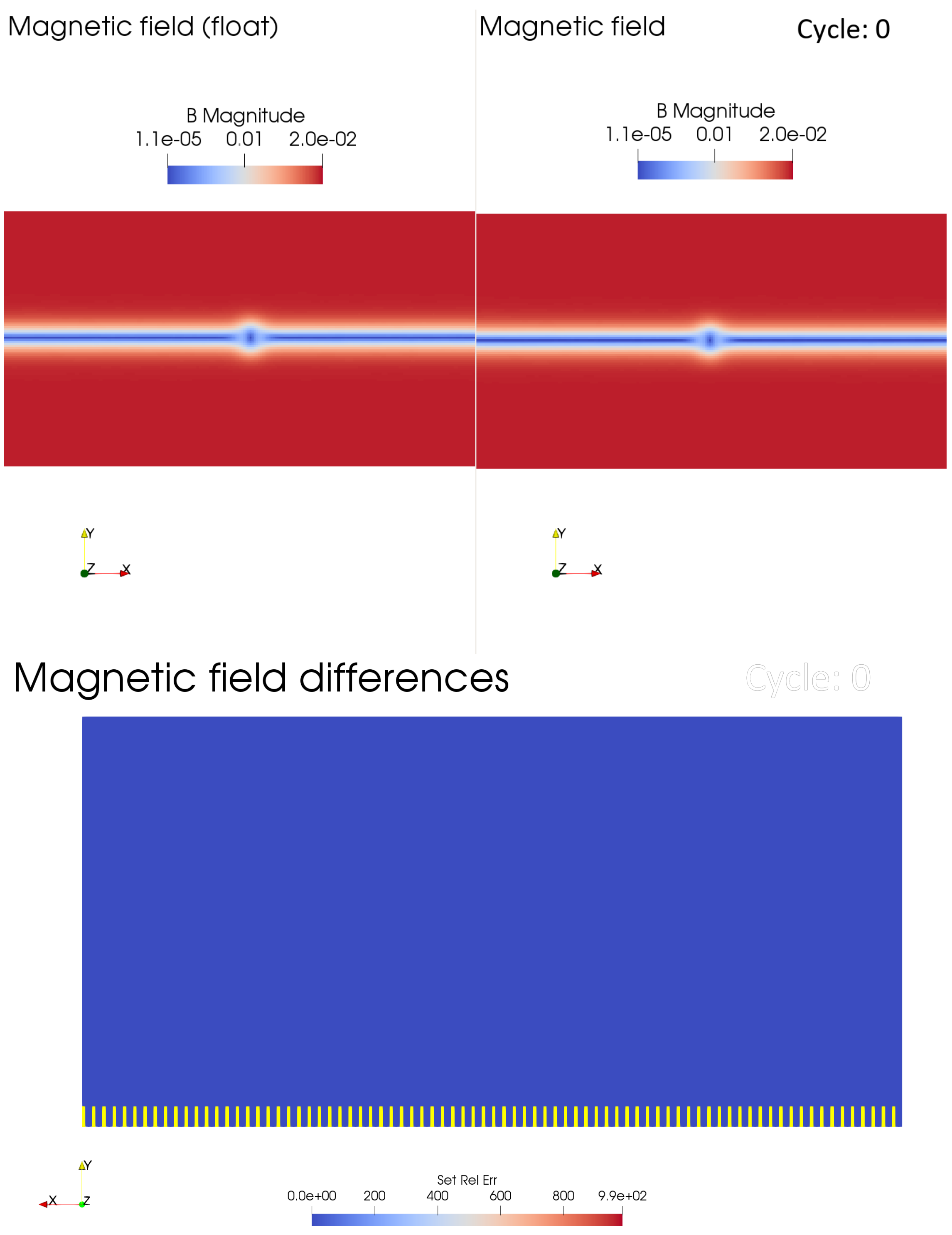
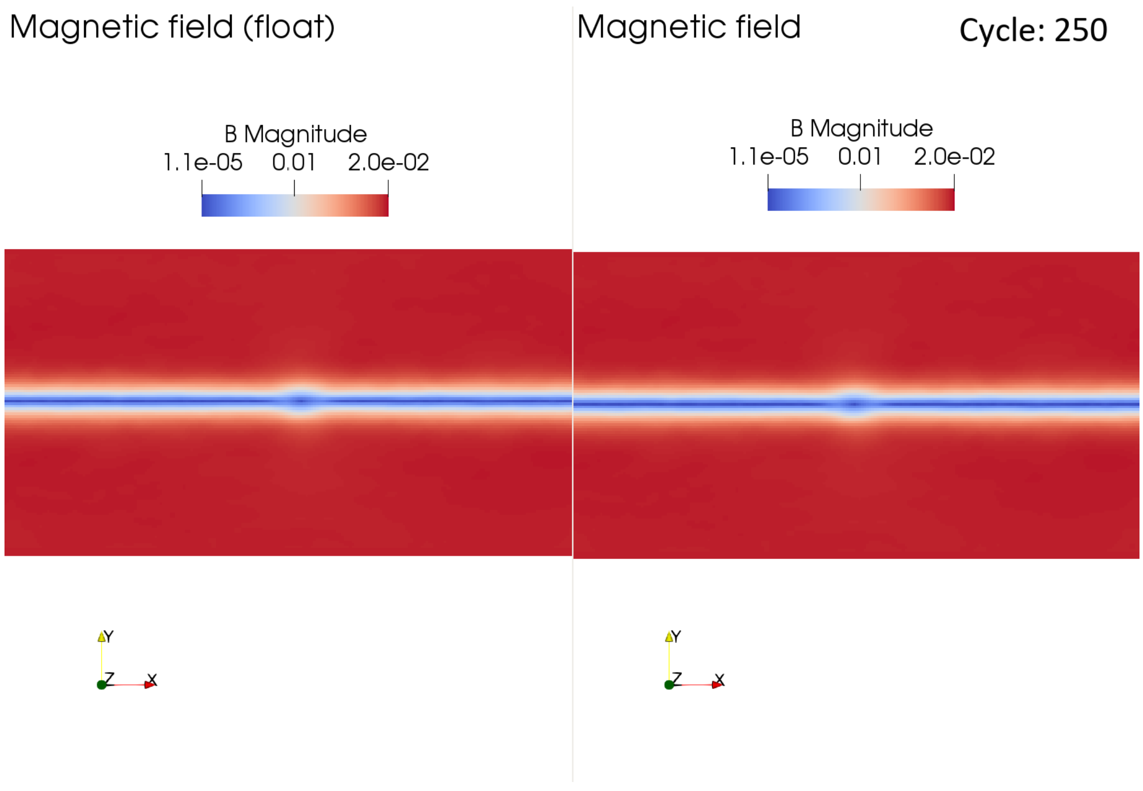


Figure 18: Just as in the electric field case, at the initialization there is no difference between simulations. The graph showing the error has changed colors, if it is blue like in this case, there is no error. Meanwhile, if it is red, the error is maximum. The yellow lines are not part of the data, they are due to an error with Paraview and they will not be considered.



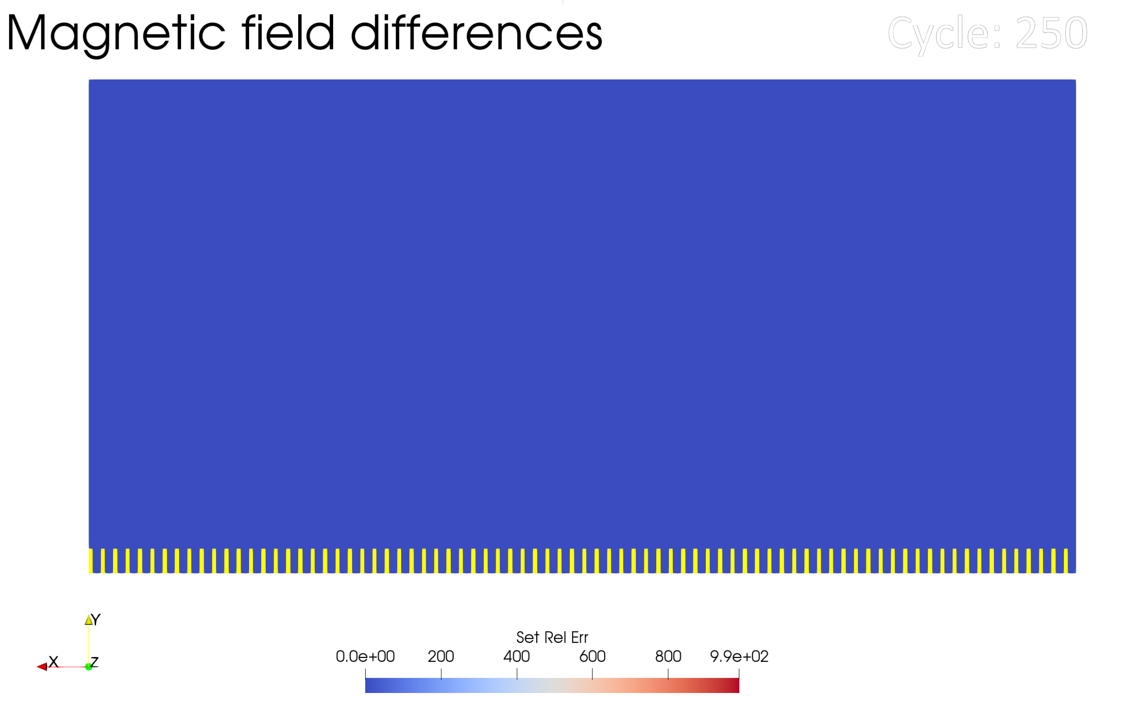


Figure 19: After 250 cycles, there is no difference whatsoever, the only thing that catches the eye are the yellow lines, but as stated before, they are not due to the data and should not be considered

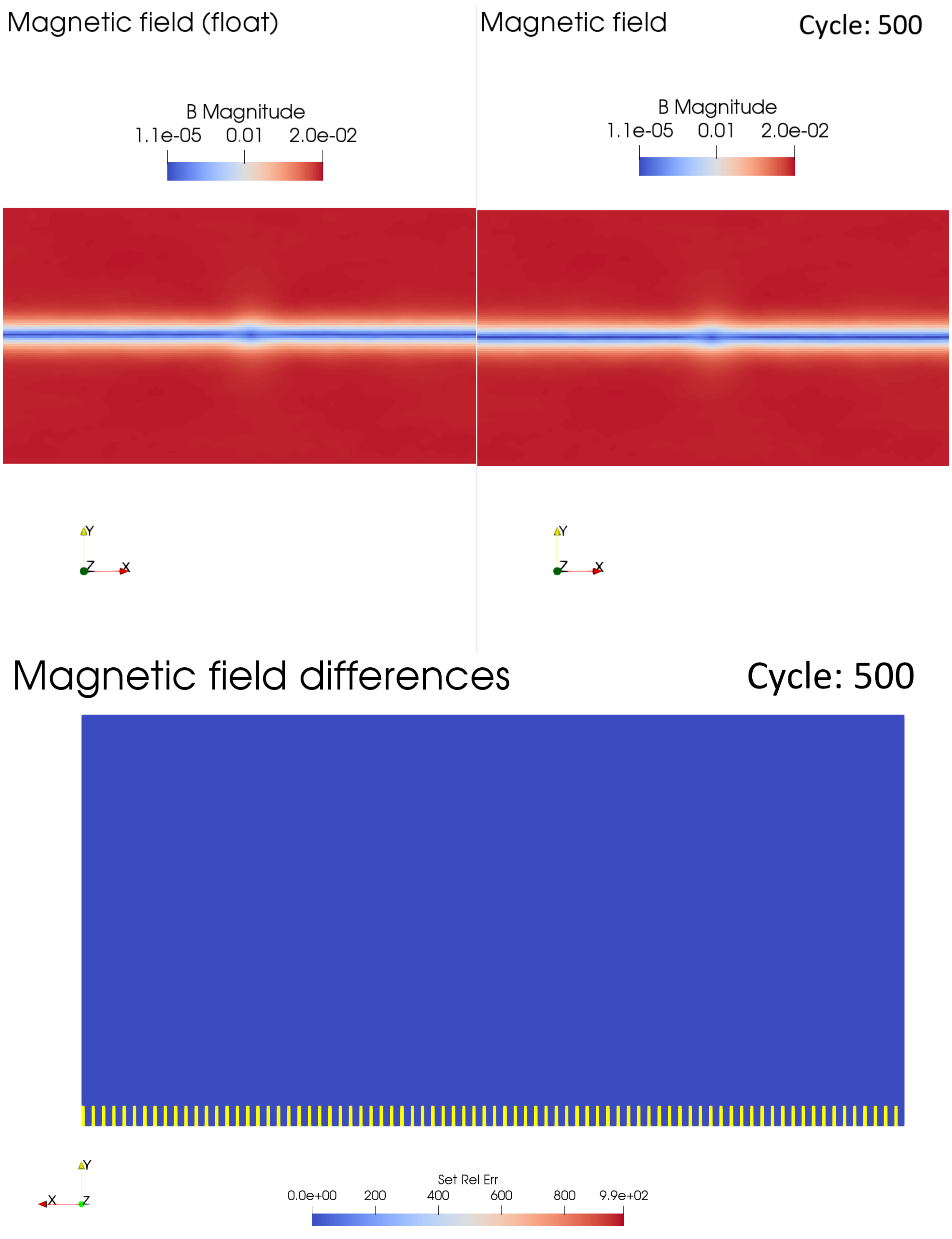
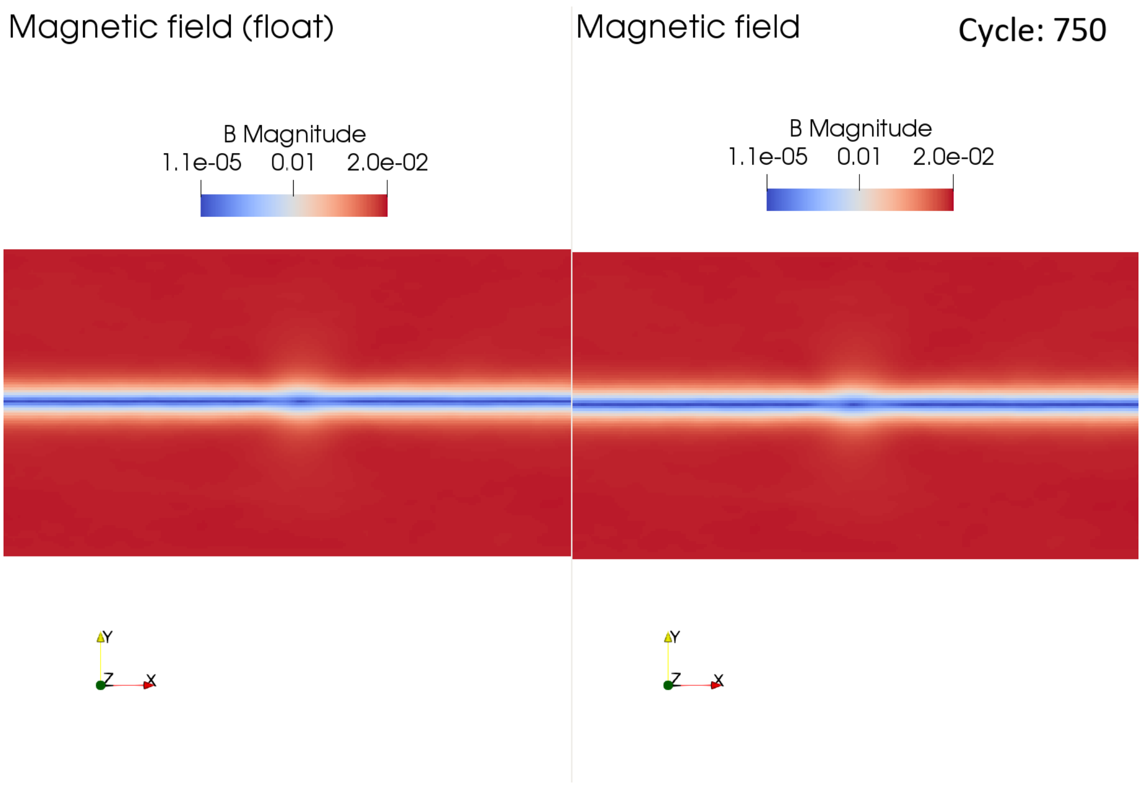


Figure 20: After 500 cycles, the plots are the same



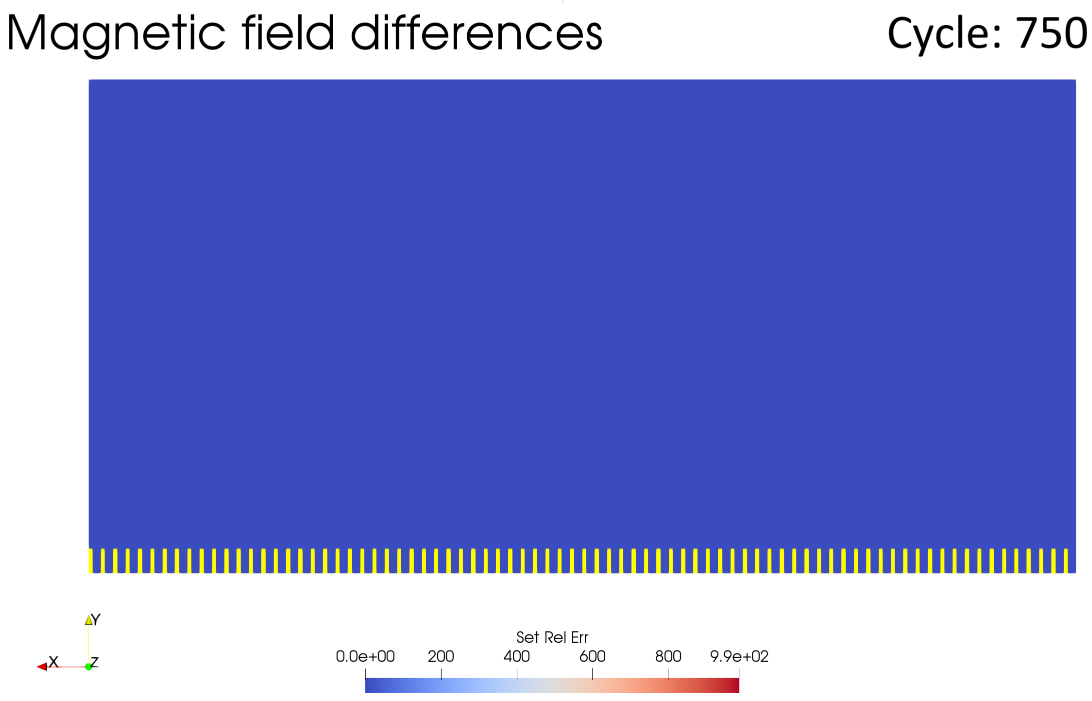


Figure 21: After 750 cycles, we still do not see any difference

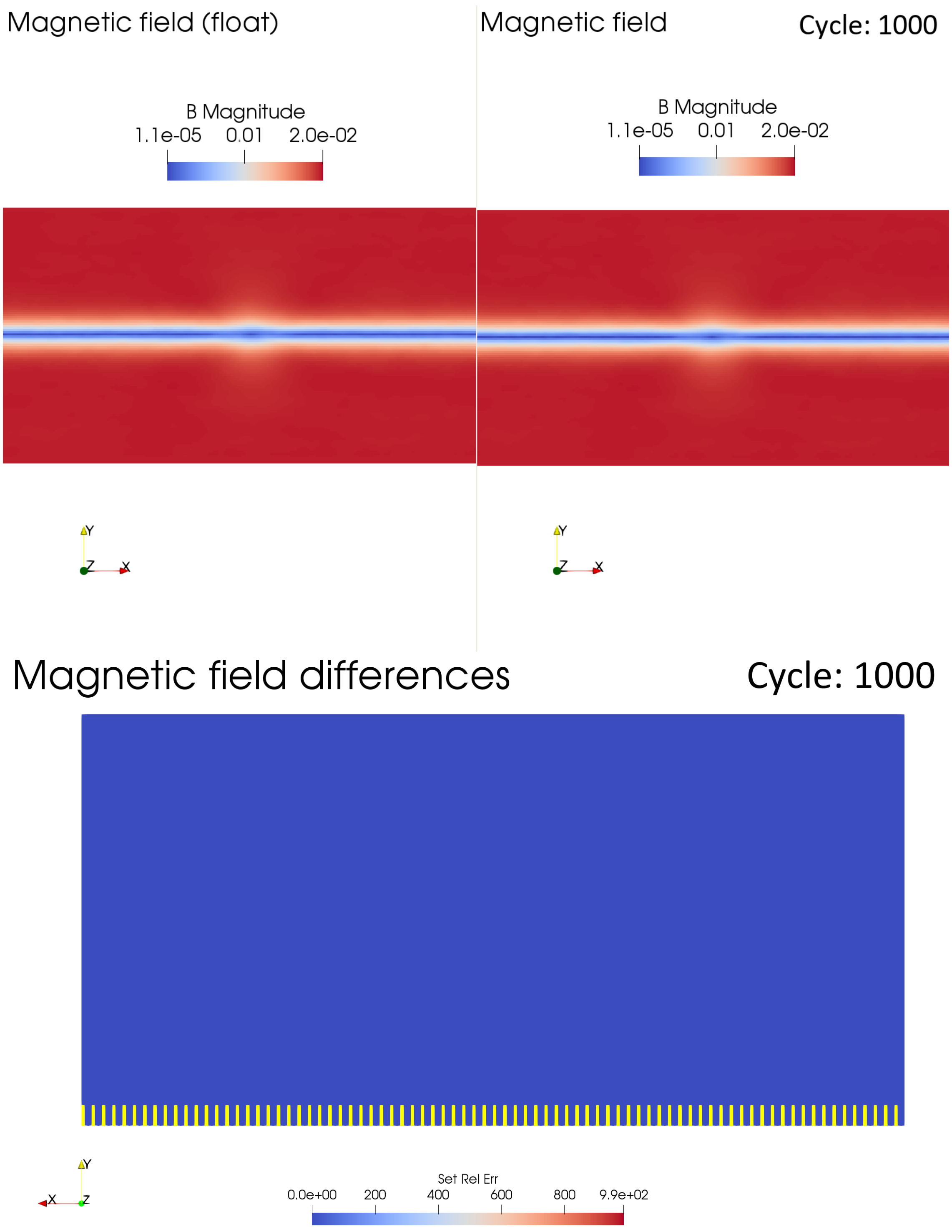
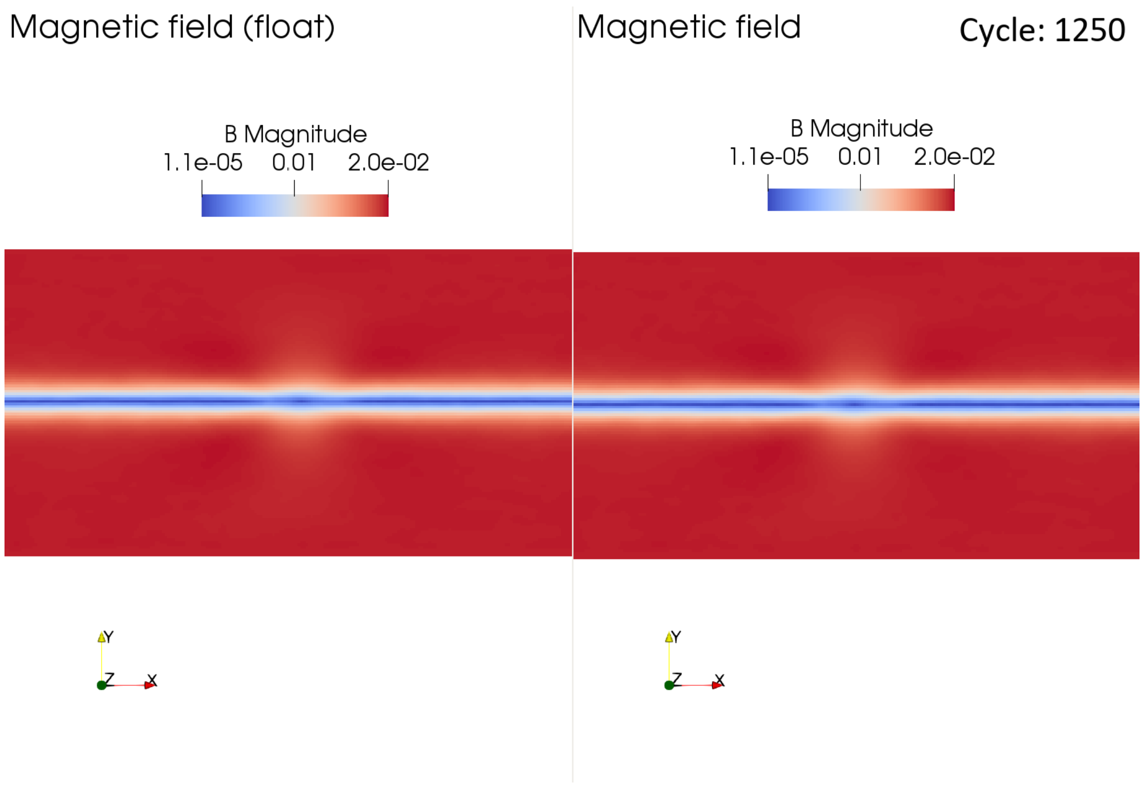


Figure 22: After 1000 cycles, as well as in the electric field, there is no discrepancy between both cases



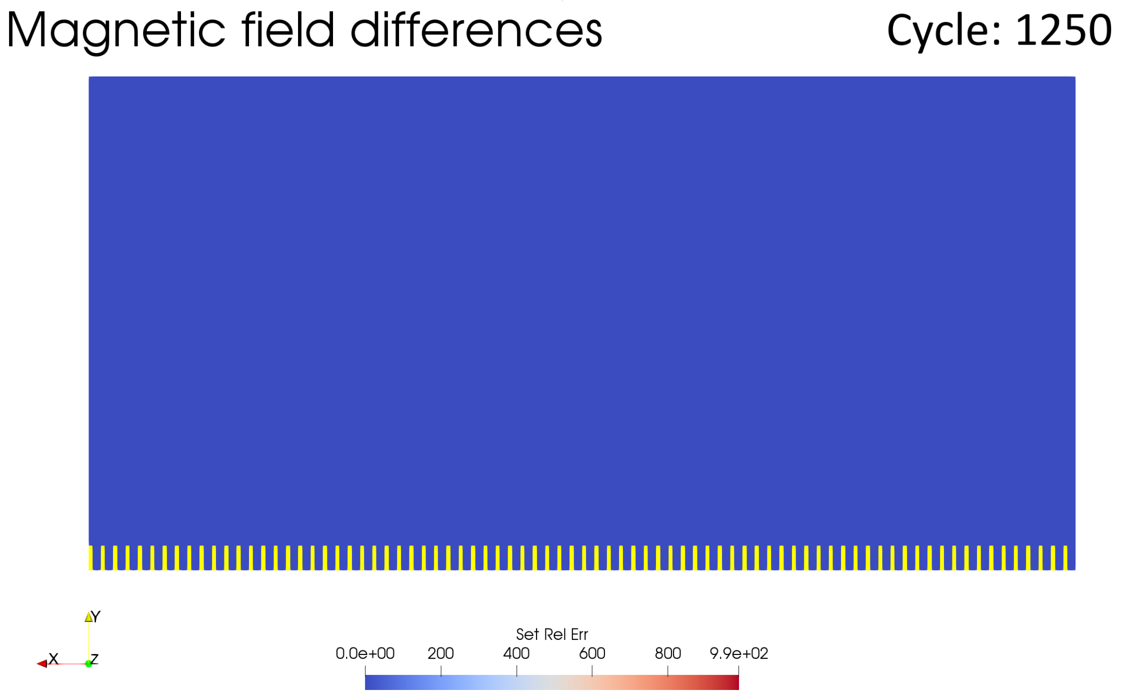
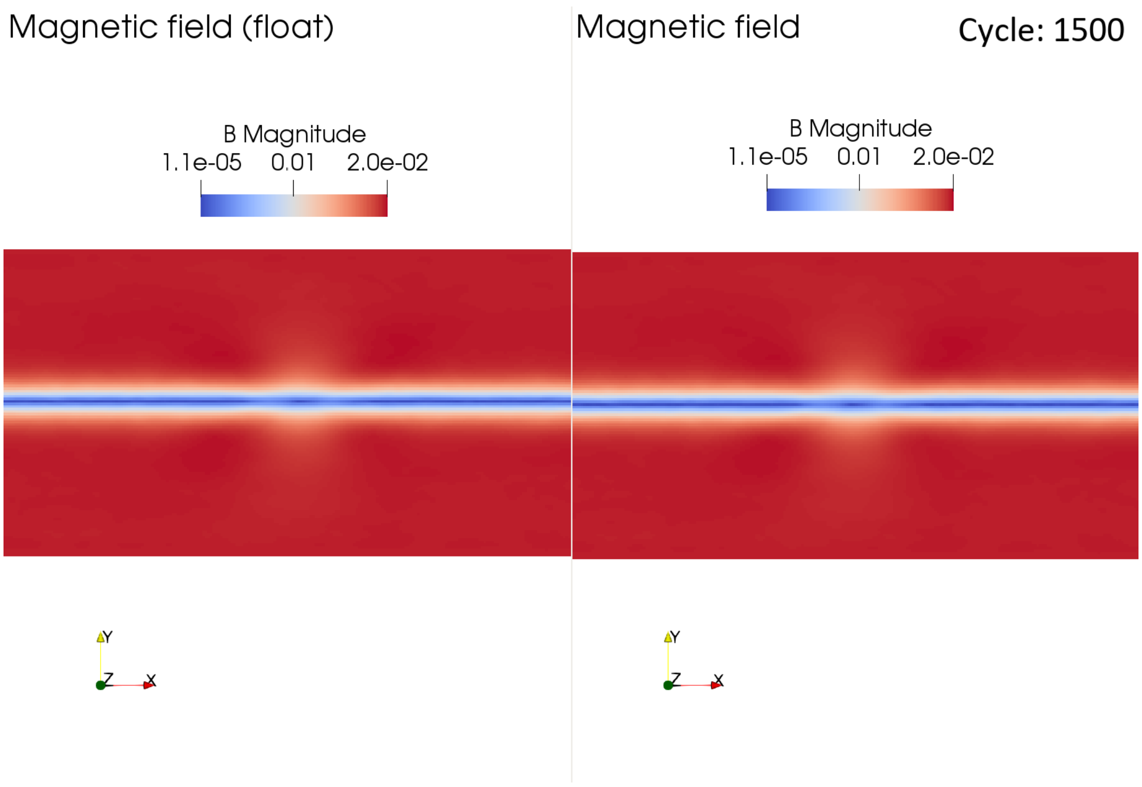


Figure 23: After 1250 cycles, the variation between simulations is non-existent



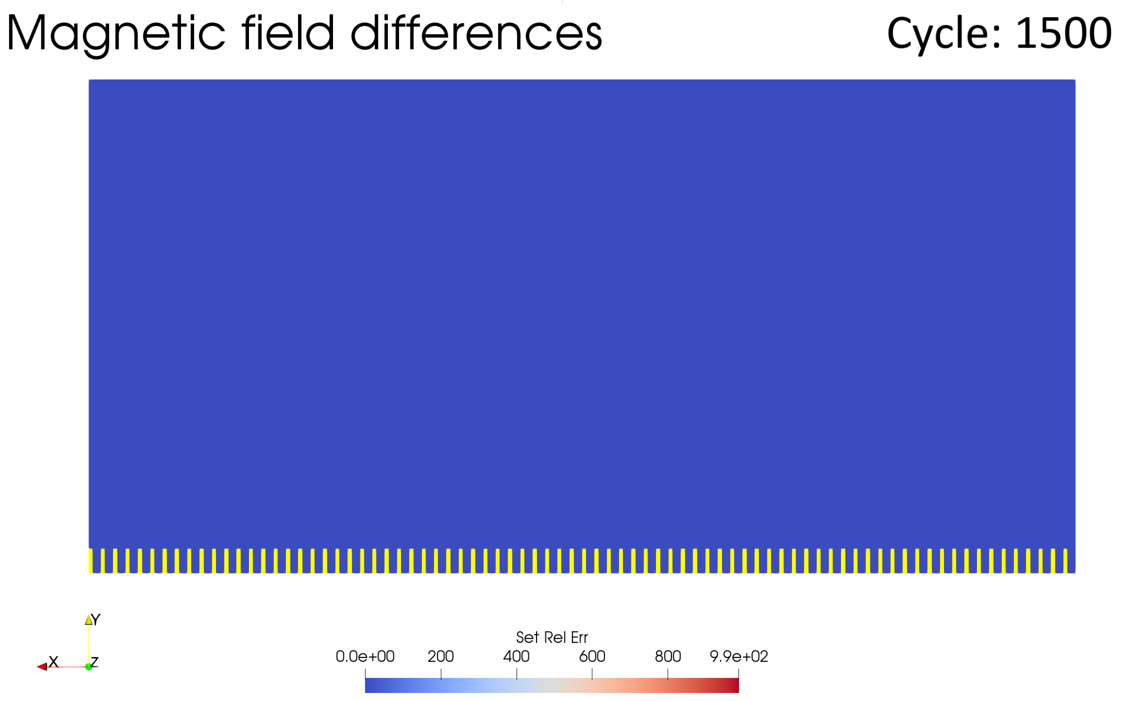
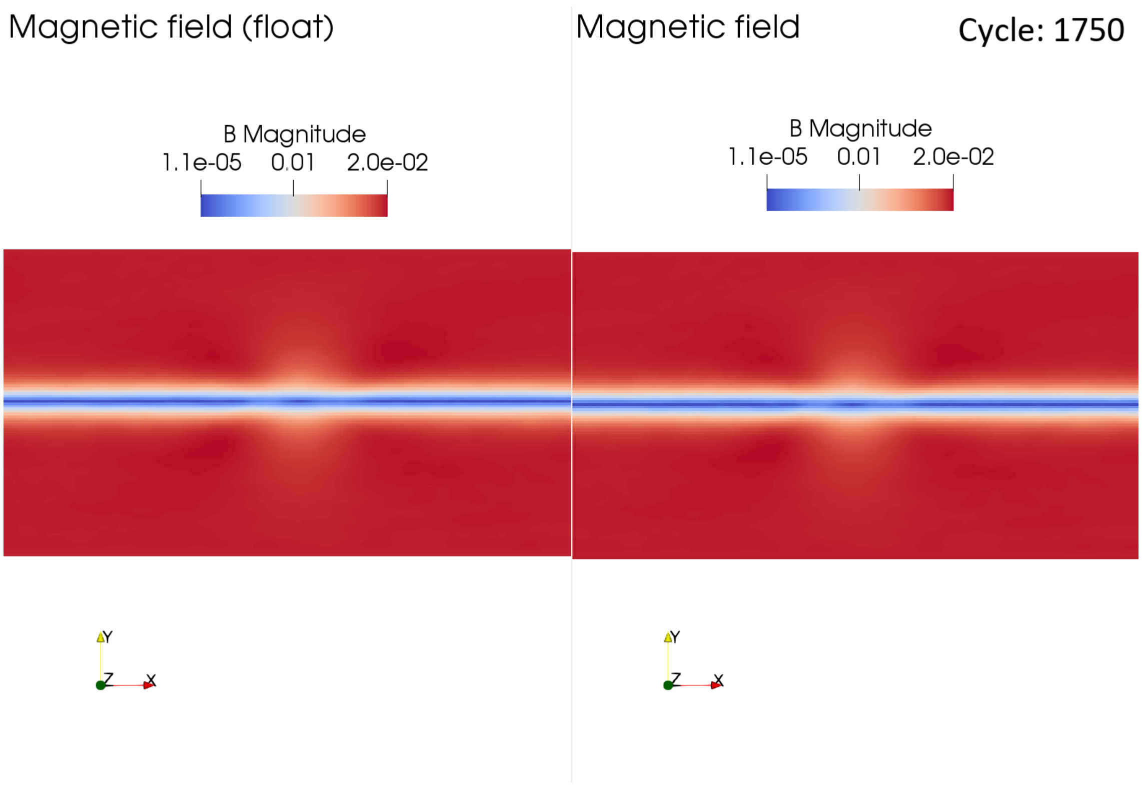


Figure 24: After 1500 cycles, unlike the electric field, the magnetic field does not show any error.



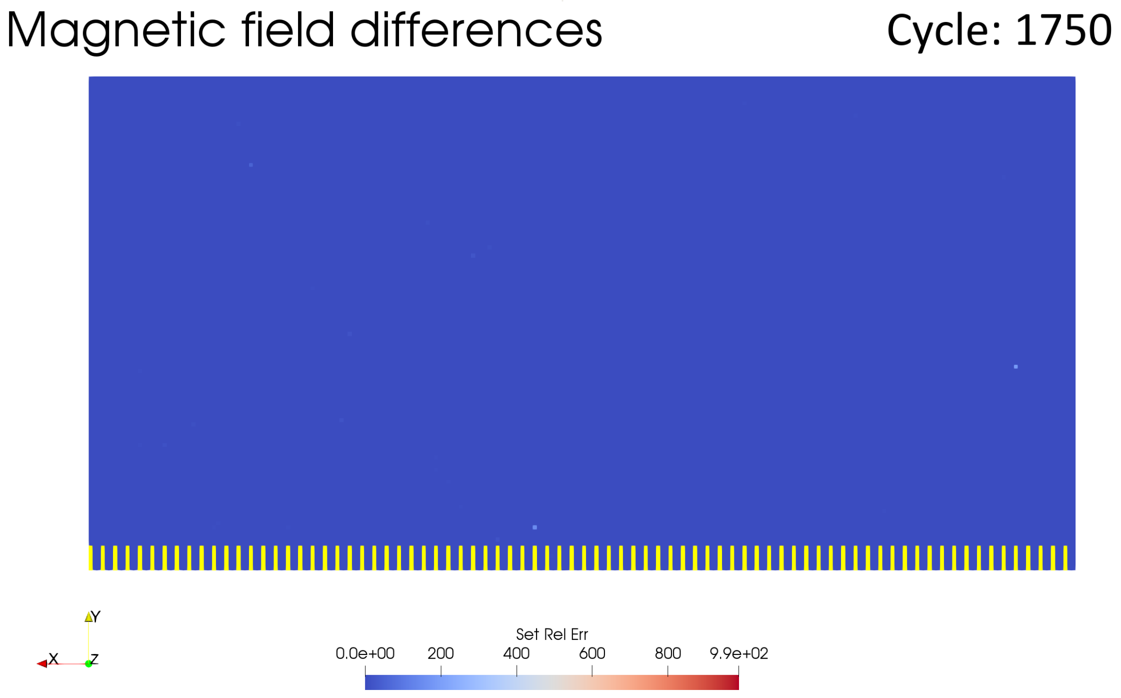
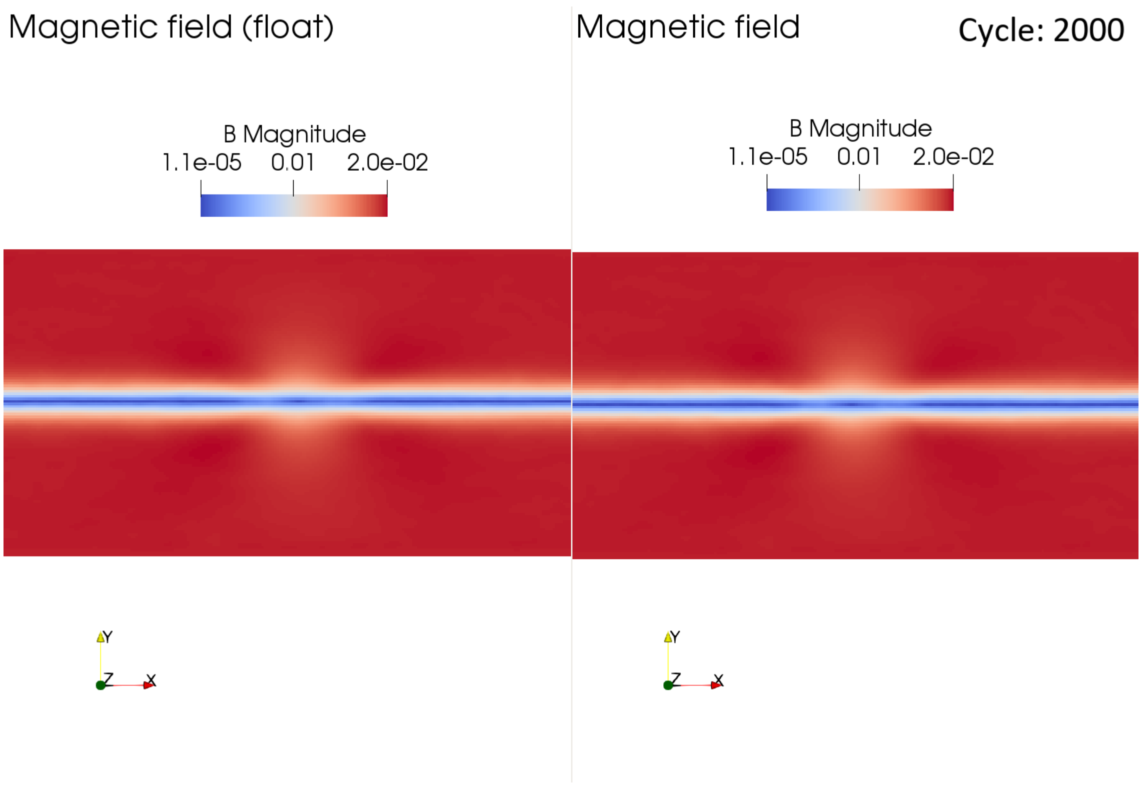


Figure 25: After 1750 cycles, this is the point where we can begin to see a contrast. There are around 3 points that show difference between implementations, the error is not high since the points are bluish, but it exists.



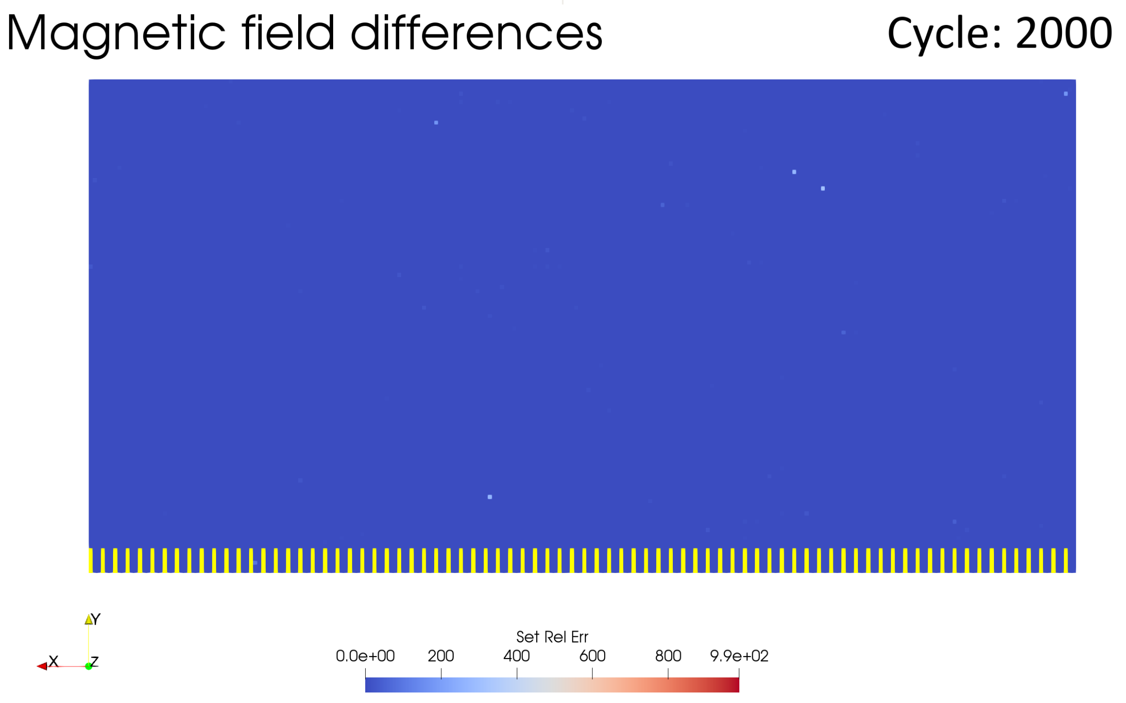
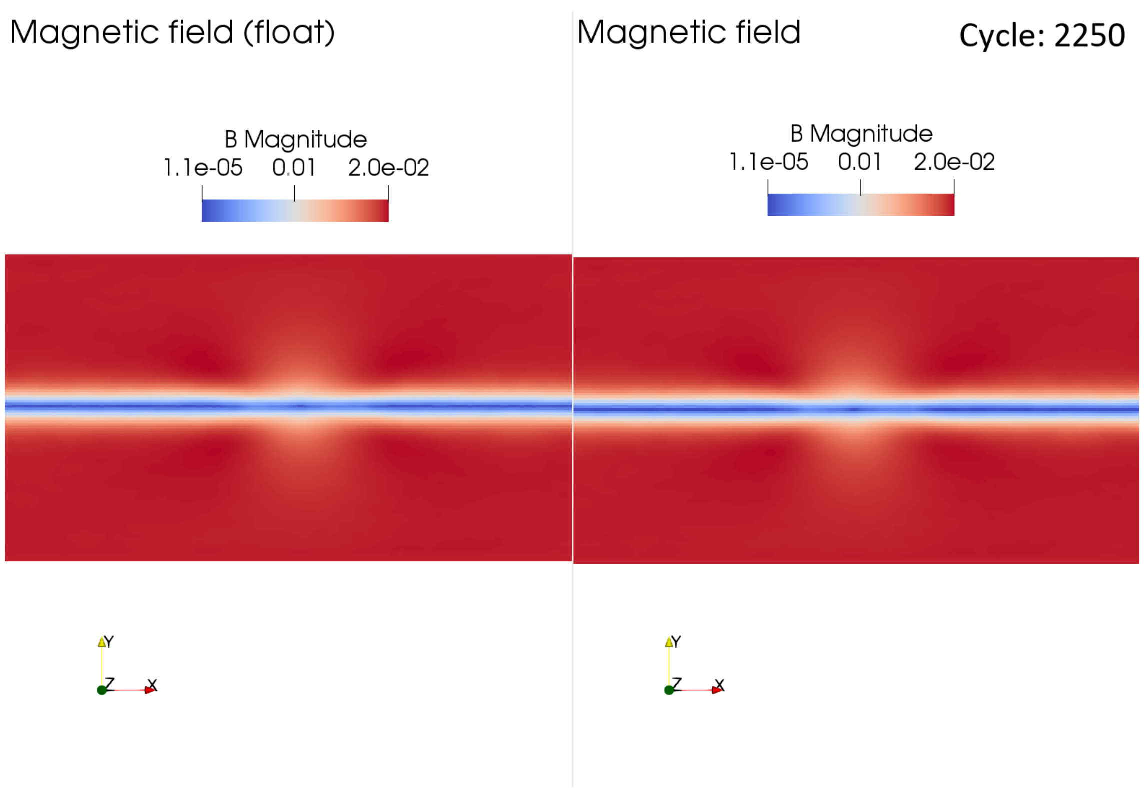


Figure 26: At the 2000 cycle mark, there has been an increase in the number of points showing a discrepancy between both cases. Is worth mentioning that the points have moved and they are not located in the same spots than before.



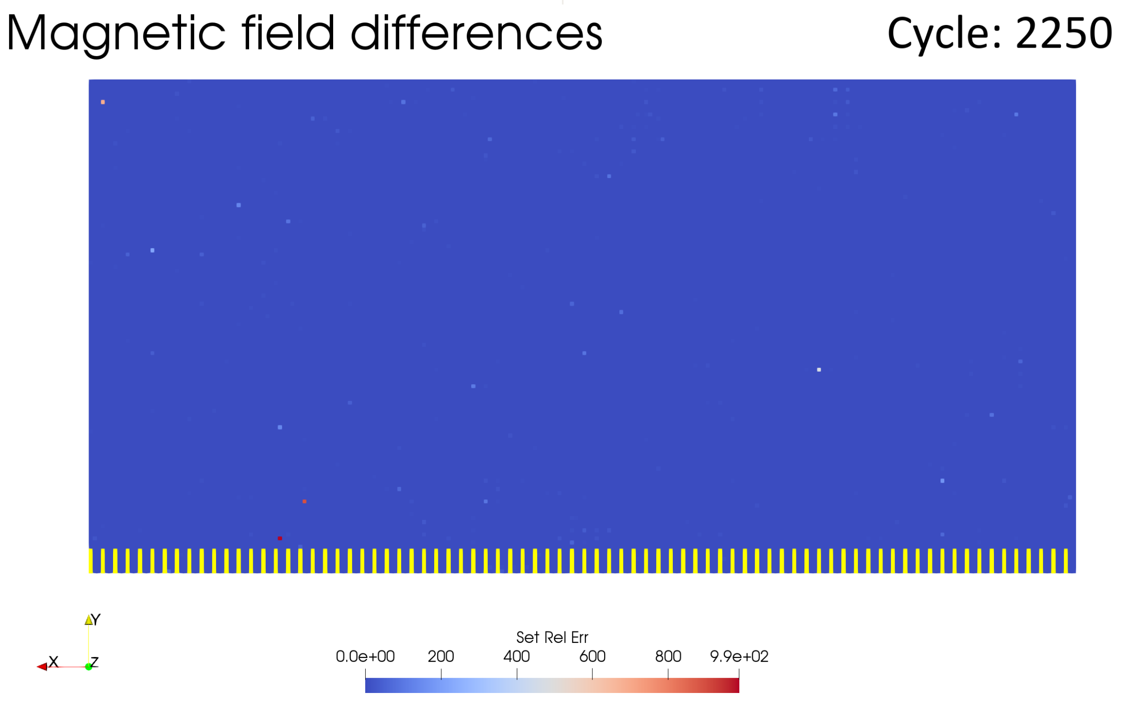


Figure 27: After 2250 cycles, we see a broad variety of colors in the difference graph, showing that the error has spread over the whole system. The maximum error is located around the bottom-left part of the graph where we can find some red spots.

0.18 Cumulative difference

Showing the evolution of the simulation is not enough, we also need to show the actual impact that the different implementation has in the accuracy of the simulation. In order to obtain this data, some more plots are provided.

As for the first one (Figure 4.21), this plot shows the cumulative difference between each of the parameters (32 in total) along the simulation, this is done subtracting each one of the parameters from both simulations and adding up the errors at each cycle. The results correlate with the previous graphs since we only see a major difference from the cycle 1750 forward and the tendency the plot follows suggests that it will keep growing after.

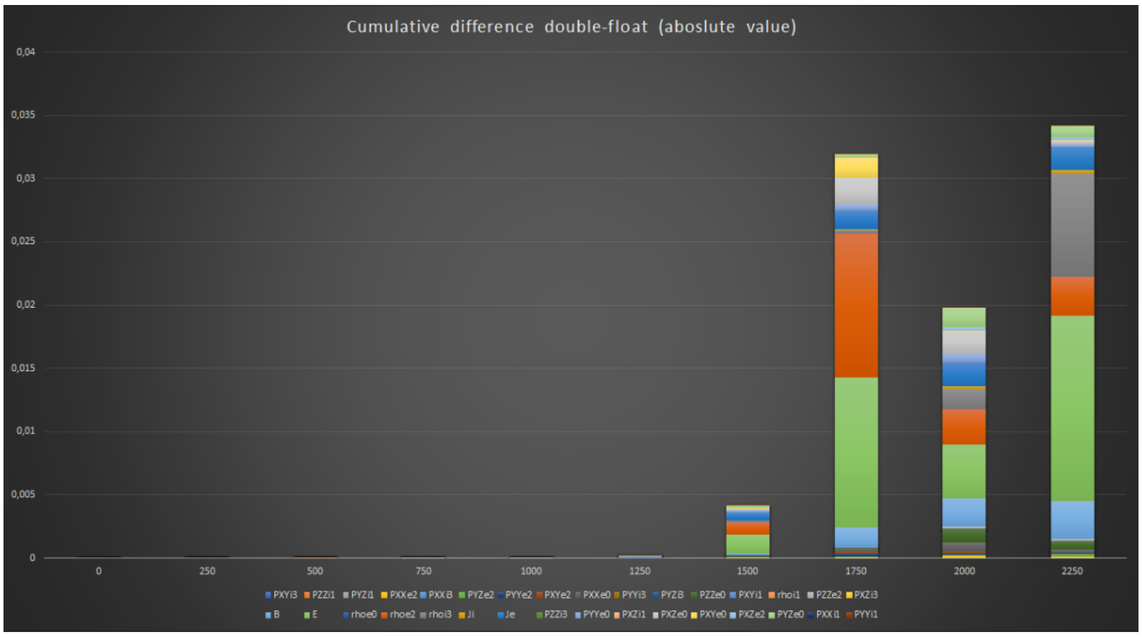


Figure 28: Cumulative difference at each cycle

0.19 Macro quantities

As for the last graph we have (Figure 4.22), it was obtained from the macro quantities iPIC3D calculates along the simulation. These quantities are some different types of energy and the overall momentum of the system. They were used in order to quantify the actual difference between simulations due to the fact that these magnitudes are dependent of the whole system and therefore, sensitive to any changes.

This graph presents 2 major differences from the others, finding a significant error at the cycle 500 that is later reverted and also a major decrease in the error when it reaches the cycle 1250, suggesting that the trend is not the same as before, not growing as the number of cycles. The maximum error encountered along the simulation is around 2% at the cycle 2000.

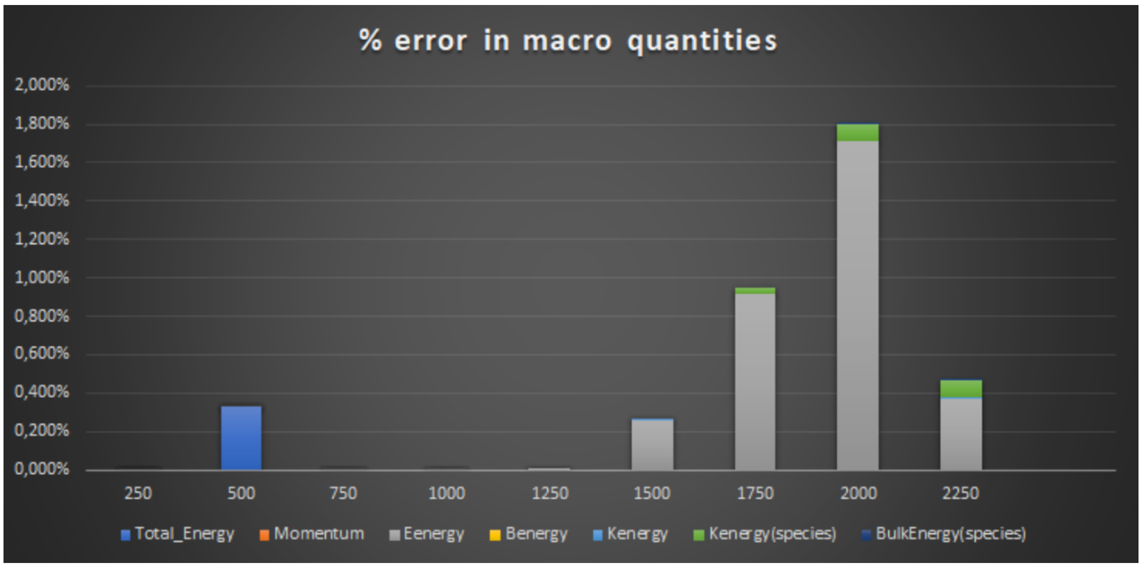


Figure 29: Error in macro quantities in %

Discussion

0.20 Precision impact

From the results we can deduce a pattern. The trend seems to be that, until around cycle 1750, there is no major difference between both implementations and the values are fairly similar. From that point forward, we see a growth in the error that would seem to increase with time, following our initial hypothesis. However, that is not the case of the macro-quantities, since they show the maximum error at the cycle 2000, from where it decreases very significantly. This could mean that, somehow, either by chance or due to other cause, the total value of the energies converge. Further study in this regard should be performed since we have not found a logical explanation for the divergence in the data. We will suppose, however, that the overall conclusions of the study are correct and the maximum error encountered is the 1.8% found at the cycle 2000.

0.21 Performance impact

Due to the limitations in our study (virtualized machine) there is no possibility to properly measure a difference in performance and therefore we can not reach any scientific conclusion in this regard. However, as an informal measure we observed a 15-20% speed-up in the time of execution of the float implementation. The degree of relation this speed-up has with the implementation is uncertain, since it could also be related with the management done by the OS during the execution of the code.

Conclusion

At this point, we are able to answer the question we set at the beginning of the study. We wanted to answer how is iPIC3D's particle mover affected by the precision of the numbers used and therefore determine whether we could improve the speed of the software without any significant impact in its precision. In our particular study we found no significative difference between implementations until around cycle 1750, where the error begins to grow and reach a maximum of around 2%. This means that, if anyone that uses iPIC3D in order to perform their simulations knows that they can allow an error grater than than this 2%, the speed-up will be "free". However, we think that plasma and electromagnetic simulations are from a field of study so concrete and scientific by nature that this will not be the case most of the time. Is also worth mentioning that we found a divergence that we are not able to explain between the error shown by the macro quantities and any other result obtained in the study, and thus, should be investigated.

Finally, we think that future research could be further developed regarding mixed precision impact in iPIC3D, since our limitations do not allow us to cover every possibility that the program provides. We think that larger simulations could be useful in order to observe whether the impact we found is smaller or greater on a larger computation.

Bibliography

[1] Frank H Stillinger and Thomas A Weber. Computer simulation of local order in condensed phases of silicon. *Physical review B*, 31(8):5262, 1985.

[2] Michael P Allen and Dominic J Tildesley. *Computer simulation of liquids*. Oxford university press, 2017.

[3] Charles K Birdsall and A Bruce Langdon. *Plasma physics via computer simulation*. CRC press, 2004.

[4] Dominik Göddeke, Robert Strzodka, and Stefan Turek. *Accelerating double precision FEM simulations with GPUs*. Univ. Dortmund, Fachbereich Mathematik, 2005.

[5] Stefano Markidis, Giovanni Lapenta, et al. Multi-scale simulations of plasma with ipic3d. *Mathematics and Computers in Simulation*, 80(7):1509–1519, 2010.

[6] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C Smith, Berk Hess, and Erik Lindahl. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 1:19–25, 2015.

[7] Guerric Meurice de Dormale and Jean-Jacques Quisquater. High-speed hardware implementations of elliptic curve cryptography: A survey. *Journal of systems architecture*, 53(2-3):72–84, 2007.

[8] IEEE Computer Society. Standards Committee and American National Standards Institute. *IEEE standard for binary floating-point arithmetic*, volume 754. IEEE, 1985.

[9] Martha W Evans, Francis H Harlow, and Eleazer Bromberg. The particle-in-cell method for hydrodynamic calculations. Technical report, LOS ALAMOS NATIONAL LAB NM, 1957.

[10] D Tskhakaya, K Matyash, R Schneider, and F Taccogna. The particle-in-cell method. *Contributions to Plasma Physics*, 47(8-9):563–594, 2007.

[11] Vladimir V Serikov, Shinji Kawamoto, and Kenichi Nanbu. Particle-in-cell plus direct simulation monte carlo (pic-dsmc) approach for self-consistent plasma-gas simulations. *IEEE transactions on plasma science*, 27(5):1389–1398, 1999.

[12] M Radmilović-Radjenović, Jae Koo Lee, Felipe Iza, and GY Park. Particle-in-cell simulation of gas breakdown in microgaps. *Journal of physics D: applied physics*, 38(6):950, 2005.

[13] John P Verboncoeur, A Bruce Langdon, and NT Gladd. An object-oriented electromagnetic pic code. *Computer Physics Communications*, 87(1-2):199–211, 1995.

[14] Guangye Chen and Luis Chacón. An analytical particle mover for the charge-and energy-conserving, nonlinearly implicit, electrostatic particle-in-cell algorithm. *Journal of Computational Physics*, 247:79–87, 2013.

[15] Anatoli Aleksandrovich Vlasov. Reviews of topical problems: the vibrational properties of an electron gas. *Soviet Physics Uspekhi*, 10:721–733, 1968.

[16] Dominik Göddeke, Robert Strzodka, and Stefan Turek. Performance and accuracy of hardware-oriented native-, emulated-and mixed-precision solvers in fem simulations. *International Journal of Parallel, Emergent and Distributed Systems*, 22(4):221–256, 2007.

[17] Guangye Chen, Luis Chacón, and Daniel C Barnes. An efficient mixed-precision, hybrid cpu–gpu implementation of a nonlinearly implicit one-dimensional particle-in-cell algorithm. *Journal of Computational Physics*, 231(16):5374–5388, 2012.

[18] Guangye Chen, Luis Chacón, and Daniel C Barnes. An energy-and charge-conserving, implicit, electrostatic particle-in-cell algorithm. *Journal of Computational Physics*, 230(18):7018–7036, 2011.

[19] Dana A Knoll and David E Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[20] Marc Baboulin, Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Julie Langou, Julien Langou, Piotr Luszczek, and Stanimire Tomov. Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications*, 180(12):2526–2533, 2009.

[21] J Birn, JF Drake, MA Shay, BN Rogers, RE Denton, M Hesse, M Kuznetsova, ZW Ma, A Bhattacharjee, A Otto, et al. Geospace environmental modeling (gem) magnetic reconnection challenge. *Journal of Geophysical Research: Space Physics*, 106(A3):3715–3719, 2001.

Additional plots

These are not all the plots obtained, the rest of them are in GitHub [2] since it was impossible to obtain them from Paraview with the proper format (probably due to some bug). Thus, our only possibility was to obtain the images with a transparent background but that made them impossible to properly show in the thesis. They follow the same structure as shown in the results section so they will be understandable by anyone that has read the whole paper.

Current density (Je)

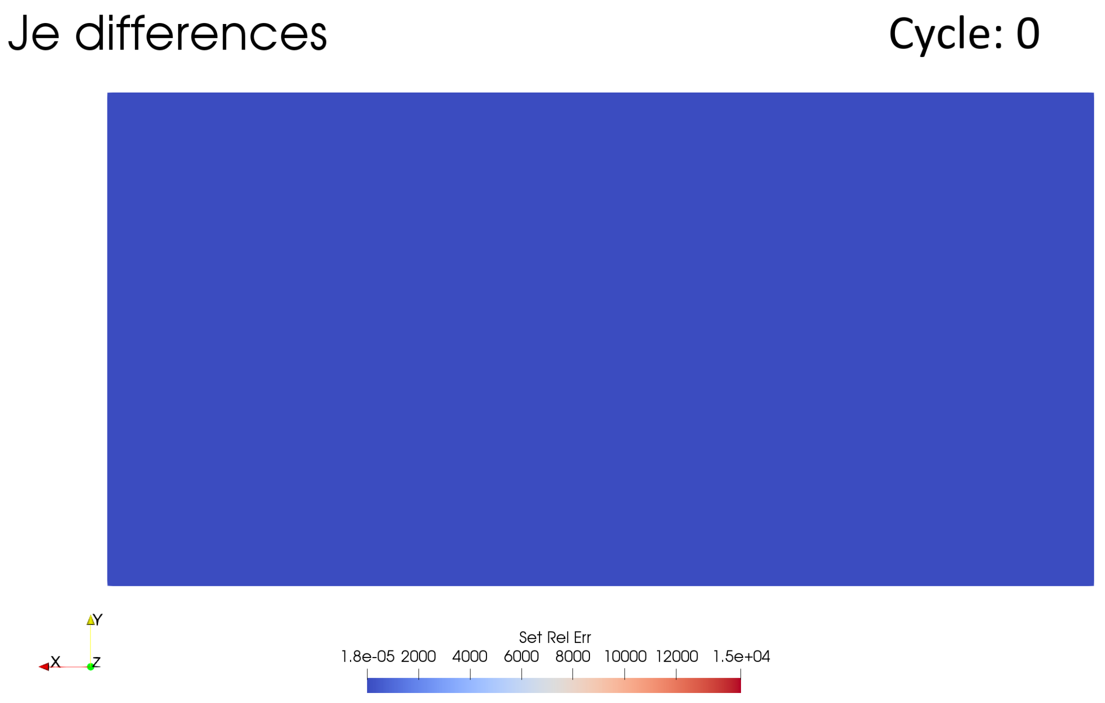


Figure 30: As well as in the magnetic field case, this difference graph shows the error to discrepancies between data sets corresponding to both simulations in a blue-red color. It is completely blue and therefore, the error is non-existent.

[2]https://gits-15.sys.kth.se/fjpm2/thesis/.

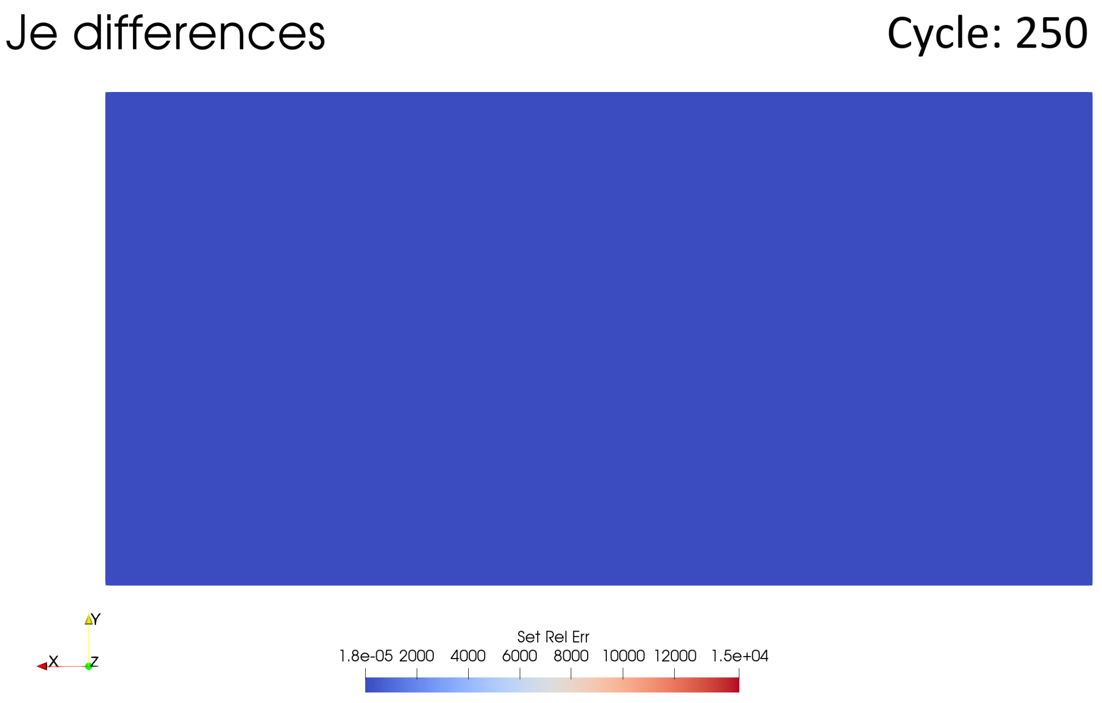


Figure 31: After 250 cycles, the plot it is still completely blue.

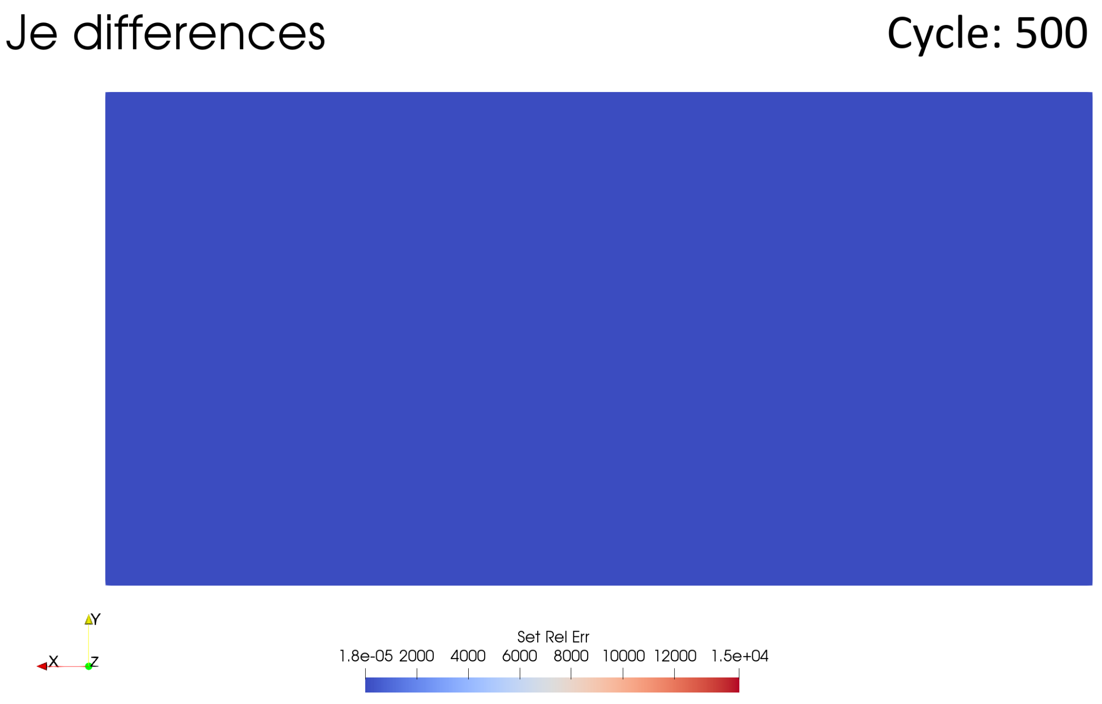


Figure 32: After 500 cycles, the plots are the same

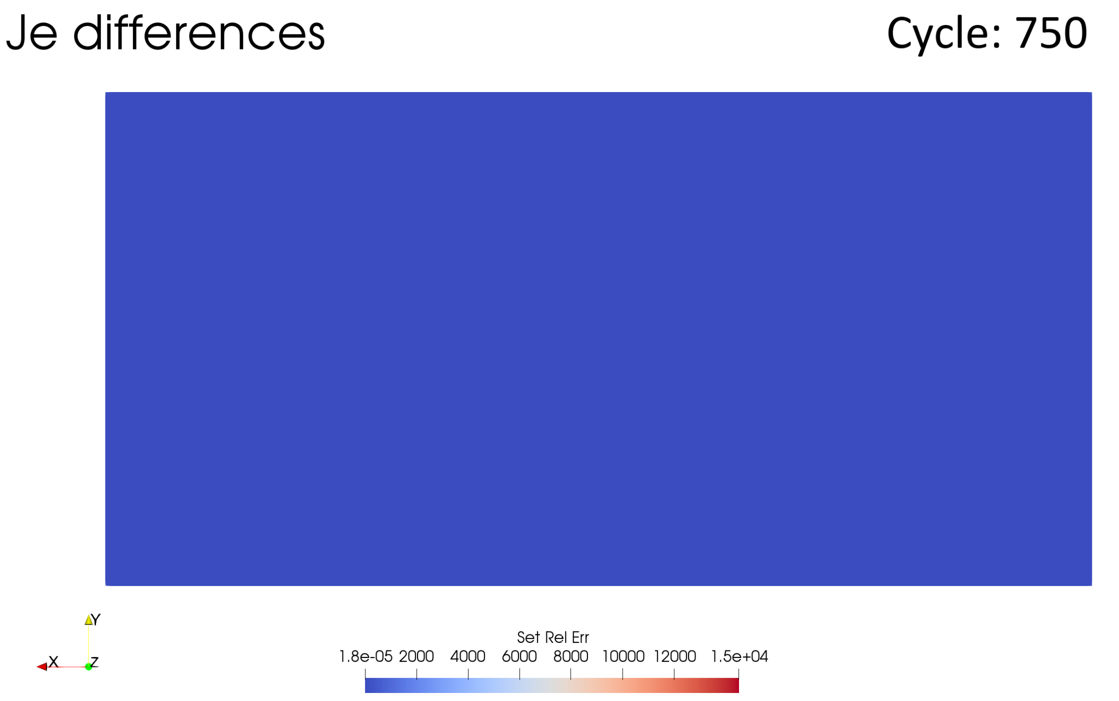


Figure 33: After 750 cycles, we still do not see any difference

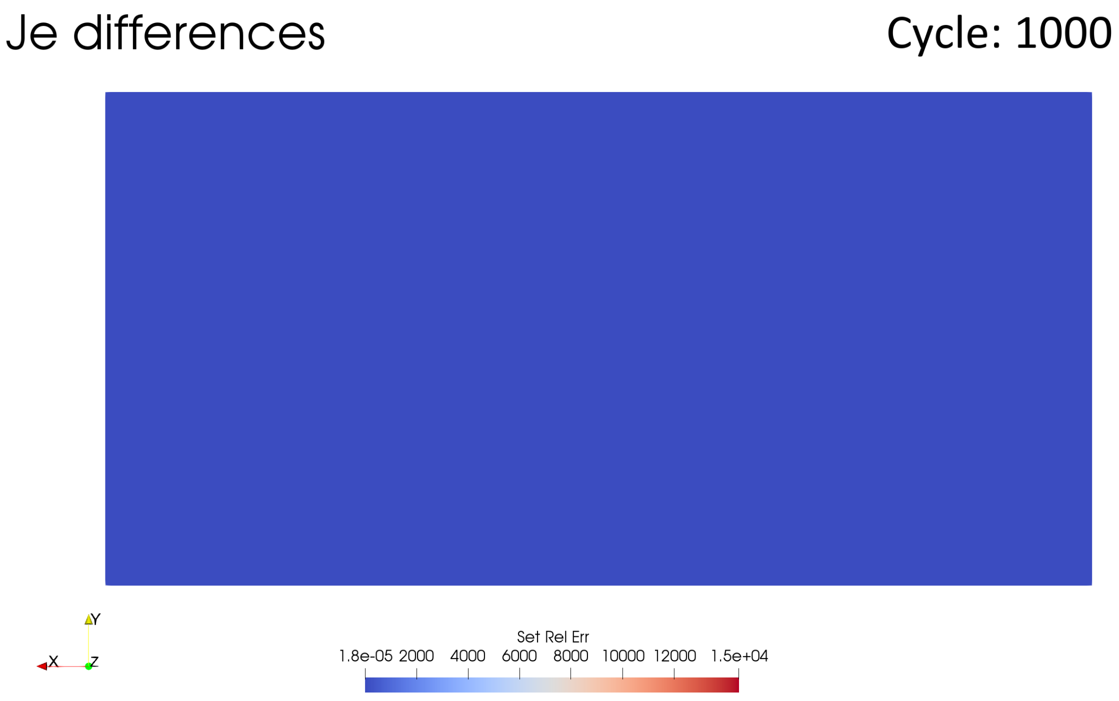


Figure 34: After 1000 cycles there is no discrepancy between both cases

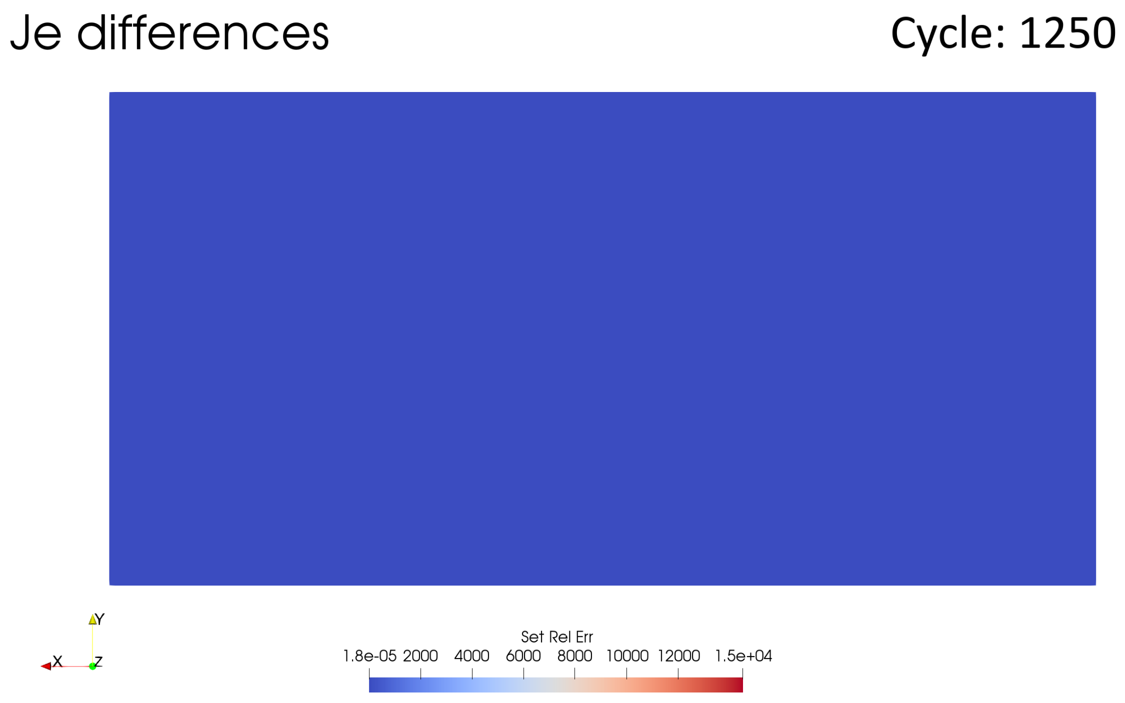


Figure 35: After 1250 cycles, the variation between simulations is non-existent

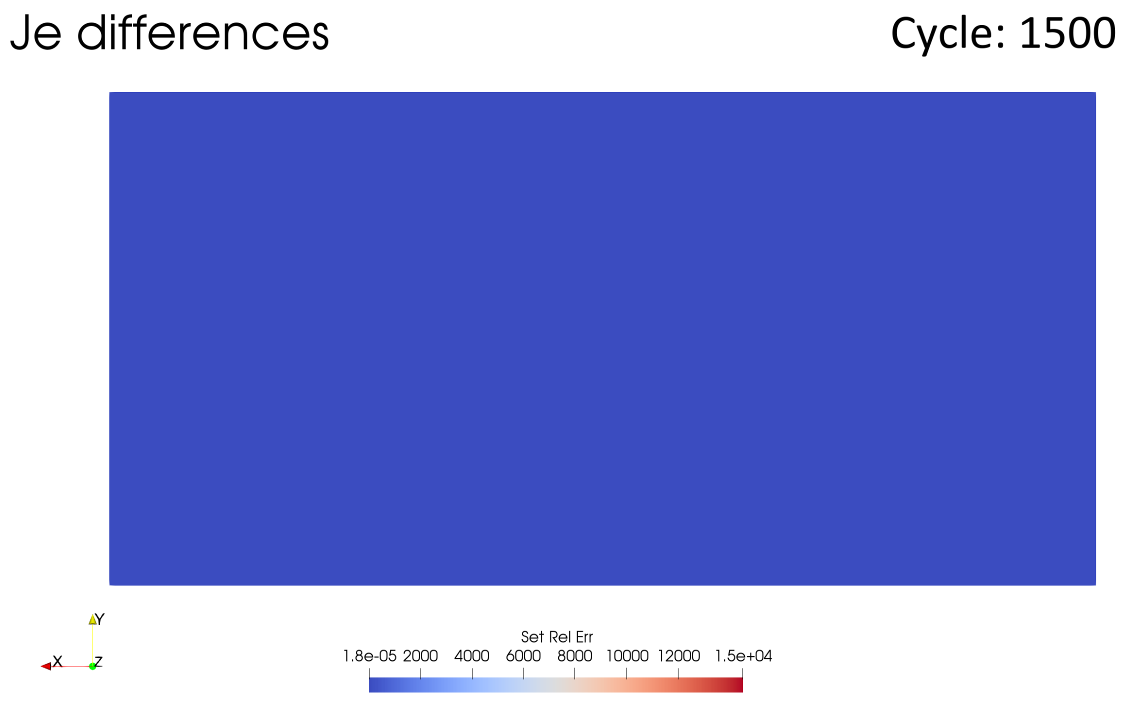


Figure 36: After 1500 cycles the current density does not show any error.

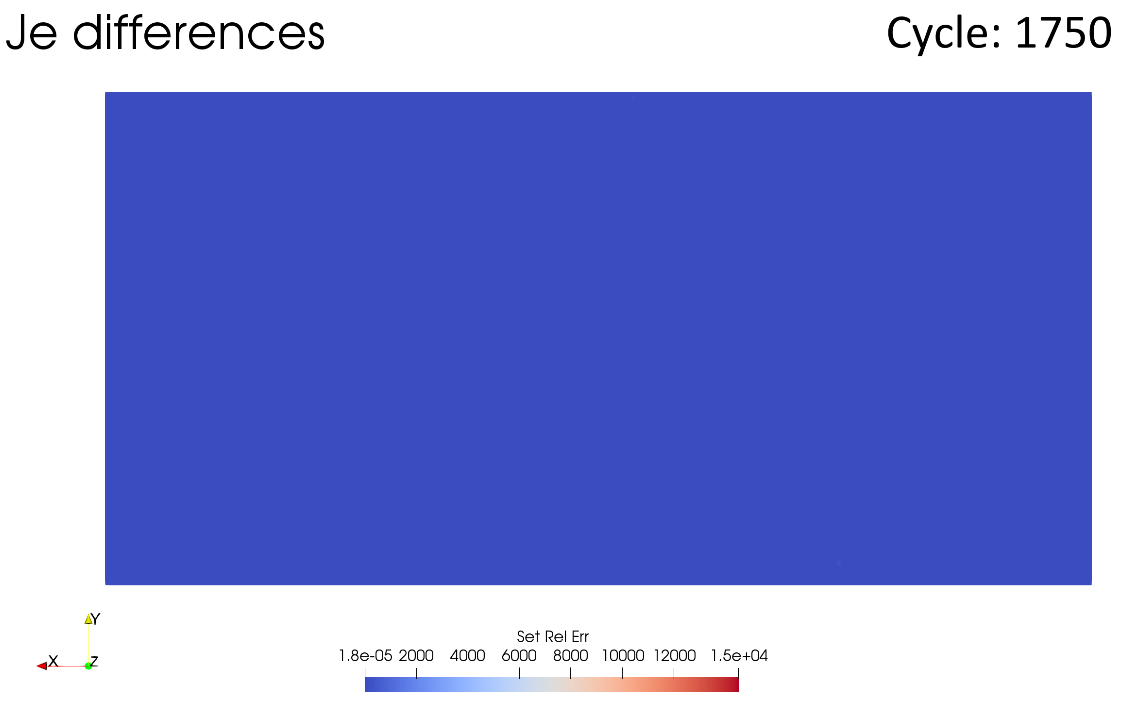


Figure 37: After 1750 cycles, we still see no difference, showing that the current density exhibits a lower error than the other magnitudes

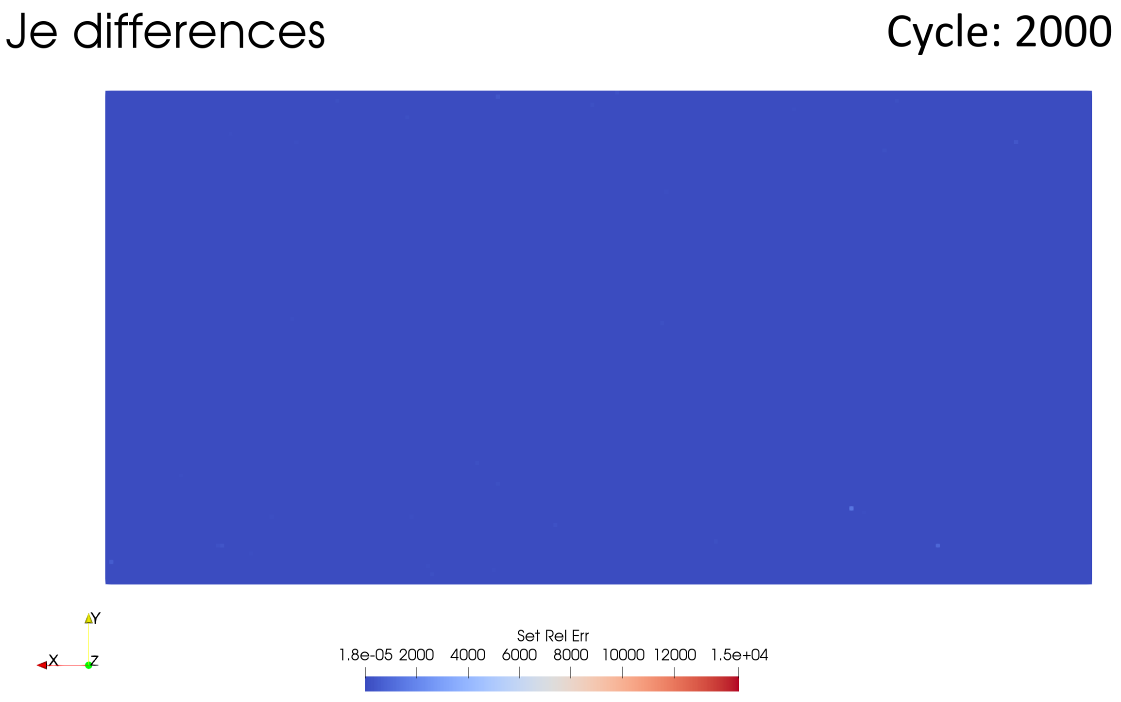


Figure 38: After 2000 cycles we begin to see some discrepancies, they are located at the bottom-rigth part of the graph. The error is still low.

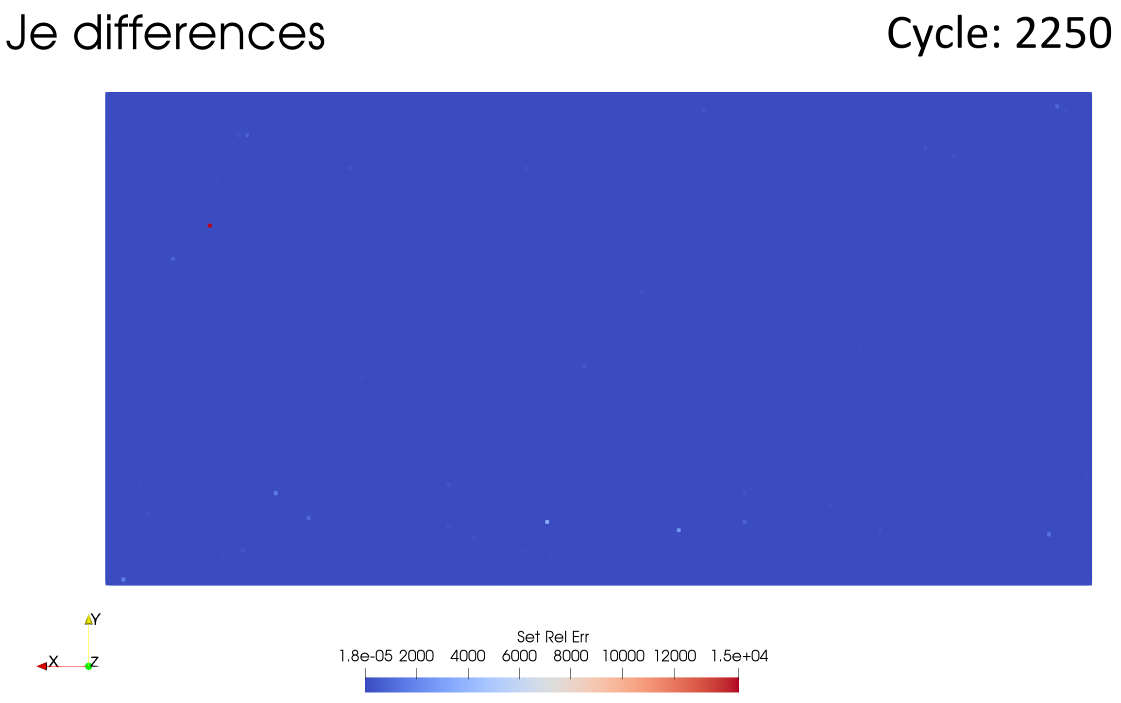


Figure 39: After 2250 cycles, we finally see some more noticeable points at the bottom part of the grpah that show that the error has increased at this stage.