ASSIGNMENT PORTFOLIO

High Performance Simulation - Computational Mathematics 2021-2022

Xabier Oyanguren Asua 1456628

1 Foreword

The present document is a guideline of the work done during the course of "High Performance Simulation" for the fourth year of the Bachelor's in Computational Mathematics (UAB).

In this subject, we were instructed in the usage of professional simulators, with special emphasis in the computational workload-result quality trade-off. Every two weeks, a new open source simulator feature was introduced in a short lecture, following which, in groups, preliminary exercises were completed, to culminate the sessions with an application of the acquired knowledge in a particular study case. Such study cases were intended to leave the students more analytic freedom. The resulting analysis was then typically exposed to the rest of the class.

The simulators employed in the subject were in order, the following ones:

- WindNinja [1]: A simulator developed by the Missoula Fire Sciences Laboratory in the U.S., for high resolution spatially varying wind field prediction in complex terrain.
- FARSITE [2]: A simulator also developed by the Missoula Fire Sciences Laboratory, now for wildfire growth and spread prediction under heterogeneous terrain, fuel and weather conditions.
- WRF and WRF-Chem [3, 4]: The Weather Research and Forecasting (WRF) model developed by NCAR's Mesoscale and Microscale Meteorology Laboratory, is a simulator designed for numerical weather and meteorological parameter prediction. The WRF atmospheric chemistry model (WRF-Chem) conveys an additional module to predict chemical parameters related to the atmosphere.

In principle, all these simulators can be coupled together, since each feature they intend to predict is dependent on each other. This way a more adequate prediction about each of them is indeed possible. During the course we were taught to couple for instance WindNinja with FARSITE, as will be explained in the following.

Along with the simulators, visualization tools like QGis [5] (an open source geographic information displaying tool) or GrADS (Grid Analysis and Display System) [6] (an open source gridded data visualizer for Earth sciences) were also heavily used to display the results and proceed with the competent anlayses.

2 Project Compendium

Let us present here the guide to the reports of each project module, with the relevant analysis briefly explicated as a summary of the relevant discussion and results.

2.a WindNinja

The complete report for the work done with this simulator can be found at:

https://github.com/Oiangu9/_Miscellaneous/blob/main/HPS/WindNinja_High_Performace_Computing_Group6.pdf

In this work, after studying different file formats to input orographic information about the terrain (where the `.lcp` format was found to be the completest one, for its versatility and ammount of information) we began analyzing the times taken by the different parts of the WindNinja simulator, as a function of the number of parallel execution threads employed, while varying the size of the map over which the wind fields were predicted. Most notably we found there was a clear bottleneck, such that the speed-up (measured as the ratio between the sequential simulation time and the parallel one) got stuck at around 1.5 from 4 threads on, for all sizes of the maps. Timing each step of the simulator, we actually found the problem was in that the iterative solver, core of the resolution, contained a hardly paralellizable matrix multiplication operation.

We then proceeded to analyze the effect of different parameters on the execution time. As could be expected, the mesh increment was the most influential parameter we could freely vary. Too big and the simulations took almost no time, but the quality of the predictions would be awful (the numerical error could be clearly seen to explode). Too small and the simulation time was found to increase exponentially, even if the simulation also got more stable.

In this line, the overall simulation time-quality trade-off was needed to be further examined, since we were challenged to get the best quality simulation with the least time possible. Changing the vegetation type or the initial condition wind angle and speed had no significant effect on the times, yet we found them to be really good indicators of the simulation quality, which was an important point since we had no ground-truth to compare with. The quality was easily assessed varying the wind direction and speed, because these were found to be parameters to which the simulation algorithm was very sensitive. To notice the numerical disagreements, plotting the resulting wind field differences was helpful, yet, we found even more helpful to plot the histogram distribution of the predicted wind speeds all over the map. This was because it allowed us to check which was the main order of magnitude of the predicted winds, letting us assert the significance of the maximum found differences in the difference maps, for each parameter changes. With all, we could achieve an adequate computational load (core number) to simulation quality, by adjusting mainly the mesh cell side.

2.b FARSITE

The report for the work done with this simulator can be found at:

https://github.com/Oiangu9/_Miscellaneous/blob/main/HPS/Farsite_High_Performace_Computing_Group6.pdf

We began the analysis of the FARSITE simulator by evaluating the computation time and visual changes produced by the variation of the fundamental parameters of the model in a trivial fire simulation. We found that increasing the nominal time-step does lighten the simulation, and has a more significant effect in the result quality than for instance activating the tree crown fire model or the random spot fire generation (both of which only supposed a small workload increase). This time-step parameter however, did not appear to play a role as significant as the two parameters controlling the fire perimeter propagation (which is simulated as a Huygens-Fresnel wavefront composed of individual spherical wavelets, which are discretized in a set of point-like propagator nodes). The first of such parameters, controls the maximum allowed orthogonal step taken away from the front by the perimeter nodes: if the step to be taken is bigger than this maximum distance parameter, an additional sub-step is made from there. The second one of the wavefront parameters, controls when new perimeter nodes are generated (called "re-discretization"), by fixing the linear distance between adjacent perimeter nodes to be surpassed in order for an additional node to be generated in the midpoint. The former influences the computation time especially when the simulations get more complicated as we will see, since it will only be applied when and where the steps taken by the fire are too big. The latter on the other hand, heavily influences the execution time in all cases, since it is ultimately the parameter controlling the density of nodes in the perimeter, and thus the amount of demanding computations to be performed in each propagation.

After paying attention to the times, we conducted a quality analysis on a less trivial case, by comparing the predictions with a ground-truth wildfire case (known as the "La Junquera case"). The main influencing factors (as with the time) were the two parameters ruling the perimeter node evolution. In fact an interesting interplay between the explicitly set time-step and the effective one emerged. The quality of the results did not decrease even if the nominal time-step parameter was enlarged, this was because what really controlled the amount of perimeter reconstructions, and thus the effective steps taken, is the maximum orthogonal distance for sub-steps. In fact, this interplay between the apparent time-steps and the effective ones turns out to also produce a different *de facto* perimeter resolution than the one expected naively. This is because after each sub-step a re-discretization is computed (if necessary), meaning the perimeter will get denser not only when the nominal time step is taken, but much more times in between.

Once this interplay between the nominal time step and the wavefront parameters was clear, we performed an optimization of a particular study case for which we also had a ground-truth, trying to obtain the most accurate prediction using the least resources. Interestingly, we found that the predictions were mainly affected when the perimeter steps were in the order of the grid step of the underlying orographic information employed by the simulator. Very importantly however, we found no good match between the predictions and the observed reality. We found this could be mainly because of the simple wind conditions we employed (when it turns out the wind is one of the main factors affecting a fire). Thus we concluded that a more accurate wind-field input would help the prediction significantly.

All the simulations of the section were automated employing a Python script that managed the generation of parameter files, calling of the simulator and result gathering, organization and plot.

2.c WindNinja + FARSITE

The report for the work done with this simulator can be found at:

https://github.com/Oiangu9/_Miscellaneous/blob/main/HPS/Farsite_WindNinja_High_Performace_Computing_Group6.pdf

In this work, the wind field input to FARSITE was studied. The input wind field can be a static global one that only globally varies at some certain times. This is the standard input we employed in the previous work. However, as we concluded, the predictive capacity of the model is poor in such a case. Instead, one can generate a fully detailed wind filed simulation adapted to the terrain orography with WindNinja and use its outputs as the input wind conditions for FARSITE. In this module, such a coupling was studied and proved to provide a more detailed and realistic wildfire propagation.

Among others, the most relevant issue evaluated was the automatized coupling of both simulators conducted employing a Python script. In addition, particularly relevant for the computational workload analysis was the study of the execution time effect when several parallel threads were allowed in FARSITE. As what happened with WindNinja, a bottleneck was found that made the speed-up factor stabilize from 4 threads on.

2.d WRF

The report for the work done with this simulator can be found at:

https://github.com/Oiangu9/_Miscellaneous/blob/main/HPS/WRF_High_Performace_Computing_Group6.pdf

In this work the pipeline to forecast dynamical atmospheric variables over a region of land, employing the WRF simulator, was studied. In general, the starting point for WRF, is a global meteorological model over a very coarse-grained grid, which has to be adequately interpolated to be used as inital and boundary conditions of the local denser 3D grid of the simulation support. The relevant modules (*geogrid*, setting up the 3D grid, *ungrib*, taking the data for the local region and *metgrid*, horizontally interpolating it to the two planar dimensions of the 3D grid) were carefully analyzed first. The simulator offers the option for building nested grids, for which as theoretically expected, a smooth transition of the grid spacings was found to generate numerically more stable results. In particular, a relation of 1:3 in the nesting grid spacing was found to be not only helpful for its care with the stability, but also because the interpolation from coarse-grain grids to very fine grids generates spurious sub-domain effects that can lead to anomalous results, unlike when one uses smooth transitions.

For the rest of the pipeline, concerning the core of the simulation (the *WRF-real* module, interpolating the initial and boundary conditions on the vertical axis of the 3D grid and the *WRF-exe* module, running the proper simulation on the grid), the resulting quality of the simulations was studied as a function of the simulation time-step and the finnes of the nested grid. This culminated in a study case for the city of Cadiz in a given day. Particularly simple to interpret were the temperature over the surface and the wind speed and angle maps (as wind barb plots). Adding the aerial contaminant chemical prediction to all the pipeline was then a straightforward step in the instructions.

All the result analysis was automated using GrADS scripts.

2.e WRF-Chem

The report for the work done with this simulator can be found at:

https://github.com/Oiangu9/_Miscellaneous/blob/main/HPS/WRF_Chem_High_Performace_Computing_Group6.pdf

For this last work, the meteorological condition and chemical contaminant predictions made by the WRF-Chem simulator were compared with the measurements taken by three meteorological stations in different spots of Catalunya (Montacada i Reixac, Vic and Pardines). This time, instead of using the GrADS tool for the prediction plot generation, the *netCDF4* library for Python was employed together with *matplotlib*. The former library is capable to open and parse the results of the WRF-Chem simulator in numpy array format. Using simple geometric triangulation, we could find the closest grid points to the three studied locations, where the variables at surface-level (where the meteorological station detectors are typically placed) were extracted.

We found that even if the predicted numbers did not match with the measured ones, the predicted trends for the time evolution of the different variables did match, for which we actually found a potential rationale. For example, the NO_2 concentration had a steep peak in the morning and in the afternoon, while during the day and night its presence was diminished, while the O_3 concentration started to rise right after the NO_2 peak of the morning, only to get down back at night. This was expectable, since NO_2 is a contaminant tightly related with the vehicle pollution, and the greatest vehicle density in the highways occurs in the early morning and the afternoon, when humans go to- or get out of- work. When it comes to the O_3 it could seem like the nitrogen has opposed effects on it, yet, we acknowledged that the O_3 concentration curve actually followed the temperature one. This implied both a higher temperature and NO_2 presence seemed to be correlated with a high O_3 concentration. Indeed after a little research we found that O_3 production is given by chemical reactions between oxides of nitrogen (NO_x) and volatile organic compounds (VOC) in the presence of higher temperatures and sunlight.

3 Conclusions and Outlook

In a very Aristotelian manner, by fighting with particular simulator implementations, we have been able to abstract several important lessons about high performance simulators. For their generalizability, these conclusions are thus the most valuable lessons we have harvested in this subject.

On the one hand, we have seen that simulation parameters can satisfyingly be adjusted to fit the simulation quality, execution time and resource demands within certain bounds. This however, is only possible if the deeper meaning of the parameters and their implications are adequately understood. It turns out, such an understanding requires not only theoretical knowledge but also practical trial-and-error, for which we found of invaluable importance the usage of authomatizing scripts (either Python scipts, shell scripts or ad hoc software scripts). The ammount of simulations and result post-processing that are required for a proper resource and time comparison are clear explainers of such a need. Yet, dealing with demanding simulators, not only critically entails authomatization, but also requires thorough experiment design before running a bunch of simulations that can take even days to be run, and can be useless if were redundant or not correctly chosen. Thus: trade-off balances can be used in ones favour, but for this both theoretical and trial-and-error knowledge of the parameters is necessary, the latter of which requires automatization scripts and thorough experiment design to maximize the conclusion yield.

On the other hand, we found that unless the simulators are fed with adequate initial conditions and boundary data, the resulting predictions might have nothing to do with reality. Thus, it is of capital importance for the goodness of predictions, to take as much care as possible in the grid and initial condition design. For this, coupling several simulators was found invaluable, where each generated detailed conditions to feed with quality data the execution of the others. Perhaps even more importantly, having access to updated and detailed registers from where to begin the simulations, was found essential for quantitative forecasting.

Finally, we have learned that not only a simulator is to be used as a prediction toolbox, but most importantly for scientific research, it can be helpful to detect and understand the mechanisms and interplay between different predicted emergent and fundamental variables. That is, it can help, not only to predict bare phenomenological results, but also to unveil unnoticed theoretical phenomena and mechanisms in all of their glory.

After all, it is clear that for a researcher, a good enough simulator is equivalent to a virtual Universe sandbox with which to experiment and explore the underlying orders that rule the emergent phenomena we observe, without needing to wait for- or needing to be the cause of- such events in the real world, nor needing to wait for someone to find analytical solutions to the underlying equations. Undeniably, a powerful enough simulator is the mighty power of a God right in our hands.

References

[1] WindNinja.

The responsible developer team, a brief description and highlighted related publications can be found at:

https://www.firelab.org/project/windninja

The official project web-page is:

https://weather.firelab.org/windninja/

[2] FARSITE.

The original project description and relevant publications can be found at:

https://www.firelab.org/project/farsite

It was latter included in the FlamMap package, so the updated download and information page is actually:

https://www.firelab.org/project/flammap

[3] WRF.

The main user page with all the manual, release and download information is:

https://www2.mmm.ucar.edu/wrf/users/

[4] WRF-Chem.

The main user support page where the Github release page is linked can be found at:

https://ruc.noaa.gov/wrf/wrf-chem/

[5] QGis.

The main project page, with the releases and documentation is:

https://www.qgis.org/en/site/

[6] GrADS.

The main project and release page is:

http://cola.gmu.edu/grads/