LMDZ Zoom Project
July-August, 2015
Nalanda Sharadjaya

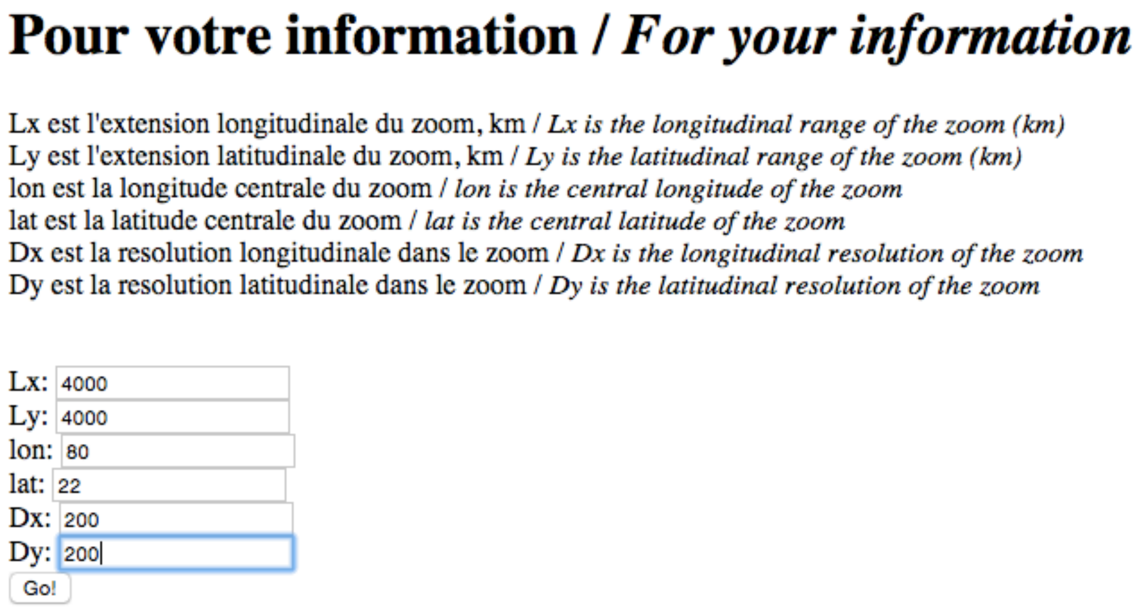
Abstract

The purpose of this project was to create an interface through which users could experiment with the LMDZ model (http://lmdz.lmd.jussieu.fr/?set_language=en) without having to run it through a shell, which would require a certain degree of familiarity with the organization of the model. Users first toggle a set of six parameters which modify the zoom (a method by which a certain location on the map can display data in greater detail without increasing the total number of squares on the grid), and can then run the model to overlay actual data (in this case, temperature) with their preferred zoom.

Methods and Results

The project was coded almost exclusively with Python. The following Python modules were used: `cgi`, `cgitb`, `subprocess`, and `os.path`. In addition to three Python files (`zoom.py`, `make.py`, and `model.py`), one small shell program (`gcm.sh`) was created as an intermediate step to call the model, `gcm.e`. These programs are original and were created exclusively for this project, although they use and interact with programs that were written by others (such as `choix_zoom.sh` and `gcm.e`), which help create the desired images. Ferret is the language used to configure the images themselves, based on instructions from other files (such as `choix_zoom.sh`).

The first page, `zoom.py`, is a simple HTML form which accepts six values: the horizontal and vertical ranges of the zoomed area (in kilometers), the central latitude and longitude of the zoom, and the resolution of the zoom (in meters). This is what `zoom.py` looks like:

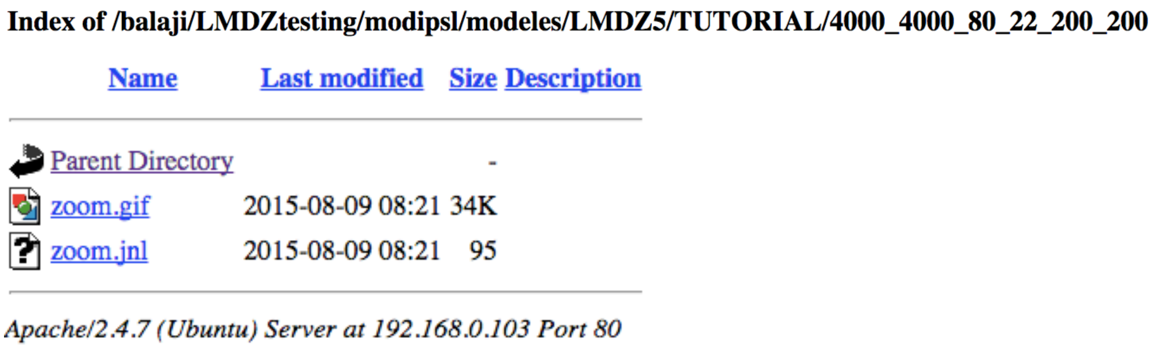

Pour votre information / *For your information*

Lx est l'extension longitudinale du zoom, km / *Lx is the longitudinal range of the zoom (km)*
Ly est l'extension latitudinale du zoom, km / *Ly is the latitudinal range of the zoom (km)*
lon est la longitude centrale du zoom / *lon is the central longitude of the zoom*
lat est la latitude centrale du zoom / *lat is the central latitude of the zoom*
Dx est la resolution longitudinale dans le zoom / *Dx is the longitudinal resolution of the zoom*
Dy est la resolution latitudinale dans le zoom / *Dy is the latitudinal resolution of the zoom*

Lx: 4000
Ly: 4000
lon: 80
lat: 22
Dx: 200
Dy: 200

Using `cgi` and `cgitb`, the second page, `make.py`, takes these values and creates a directory based on them (for example, `4000_4000_80_22_200_200/`) where any and all files created for the purposes of this zoom configuration will be stored.

Using a Python module called `subprocess`, `make.py` then sends these values to `choix_zoom.sh`, a preexisting file which constructs the zoom and saves it as a `.gif` file in the directory created by `make.py`, which then displays the image. `choix_zoom.sh` creates two files: one, called `zoom.jnl`, which displays the contents of the Ferret file used to write the image; and two, the image itself, `zoom.gif`. The directory currently looks like this:



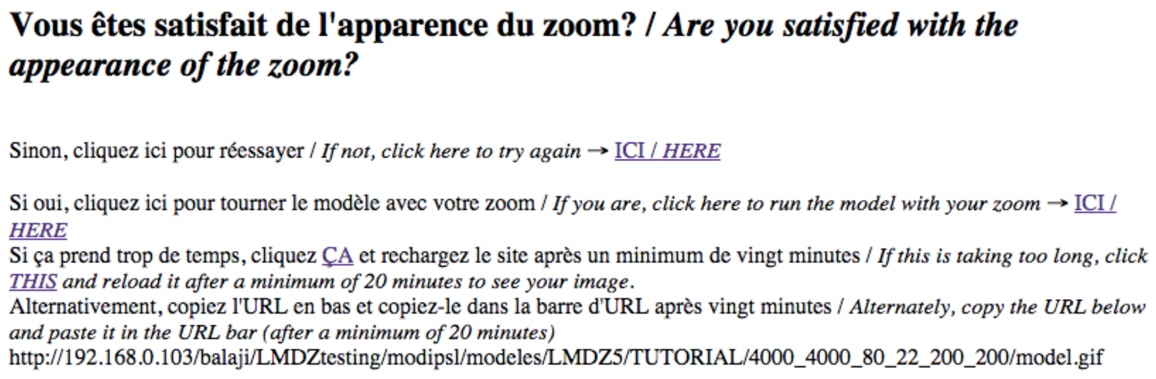
Index of /balaji/LMDZtesting/modipsl/modeles/LMDZ5/TUTORIAL/4000_4000_80_22_200_200

Name	Last modified	Size	Description
Parent Directory		-	
zoom.gif	2015-08-09 08:21	34K	
zoom.jnl	2015-08-09 08:21	95	

Apache/2.4.7 (Ubuntu) Server at 192.168.0.103 Port 80

Before creating `zoom.gif`, make.py uses `os.path`, a Python module, to check whether there is already a `zoom.gif` inside the specified directory—if so, `make.py` won't bother creating a new one; it will simply display the existing one in the browser.

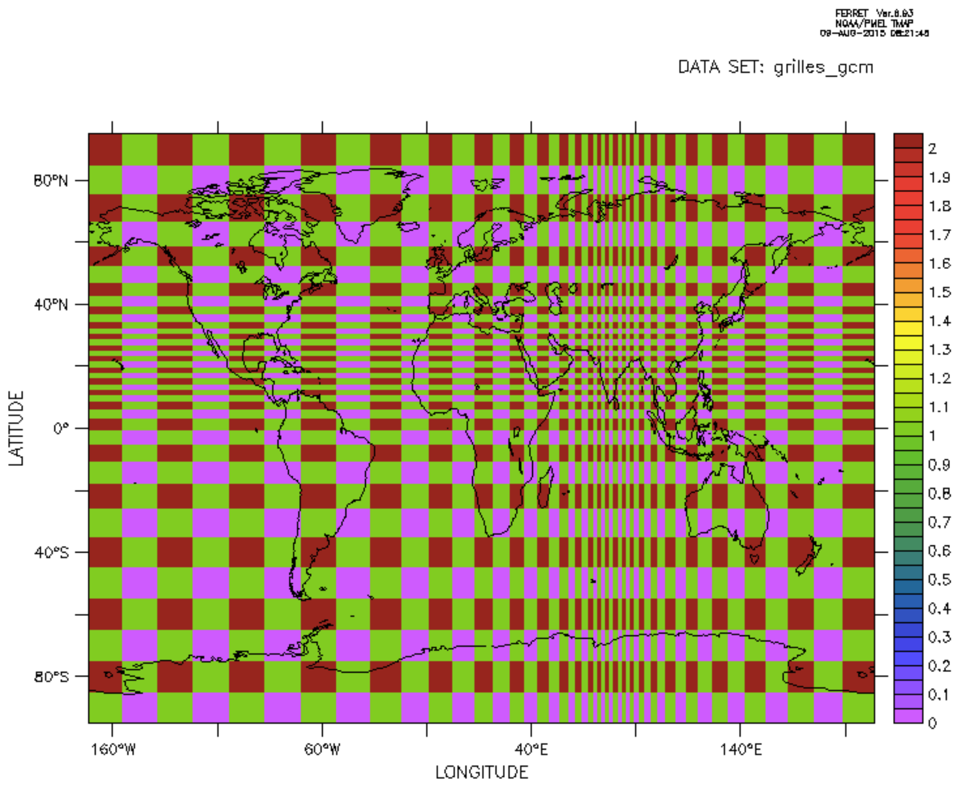
The user is then prompted to decide whether they are satisfied with their zoom or not; if they are not, they can click on a link to return to `zoom.py` and try again. Otherwise, they can continue on to `model.py` to overlay the model with their zoom. (The zoom itself is displayed on the next page. As you will see, it is centered over India.)



Vous êtes satisfait de l'apparence du zoom? / *Are you satisfied with the appearance of the zoom?*

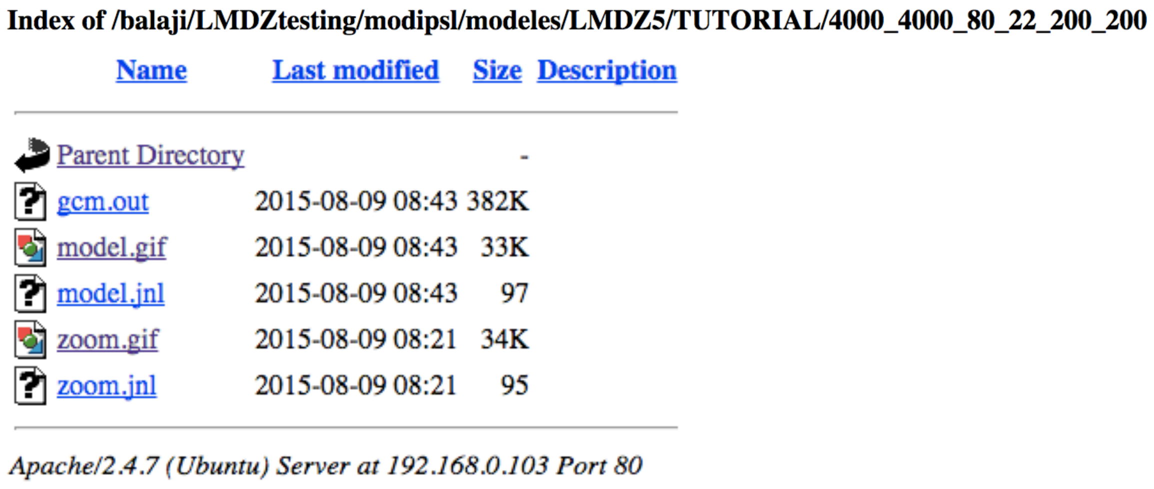
Sinon, cliquez ici pour réessayer / *If not, click here to try again* → ICI / *HERE*

Si oui, cliquez ici pour tourner le modèle avec votre zoom / *If you are, click here to run the model with your zoom* → ICI / *HERE*
Si ça prend trop de temps, cliquez ÇA et rechargez le site après un minimum de vingt minutes / *If this is taking too long, click THIS and reload it after a minimum of 20 minutes to see your image.*
Alternativement, copiez l'URL en bas et copiez-le dans la barre d'URL après vingt minutes / *Alternately, copy the URL below and paste it in the URL bar (after a minimum of 20 minutes)*
http://192.168.0.103/balaji/LMDZtesting/modipsl/modeles/LMDZ5/TUTORIAL/4000_4000_80_22_200_200/model.gif



Grille aux point (Kelvin)

When the user clicks on the link to run the model, `model.py` uses `subprocess` to call `gcm.sh`, a small shell file which calls the model itself (`gcm.e`) and also gives instructions for the Ferret journal (`.jnl`) file which will eventually create the image itself. The model will take some time to run, but two new files will eventually be created: `model.jnl`, which will be created as soon as the model has finished, and `model.gif`, which is created by `model.jnl`. The directory will also contain a file called `gcm.out`, which contains the output of the model throughout its full run. The directory now looks like this:



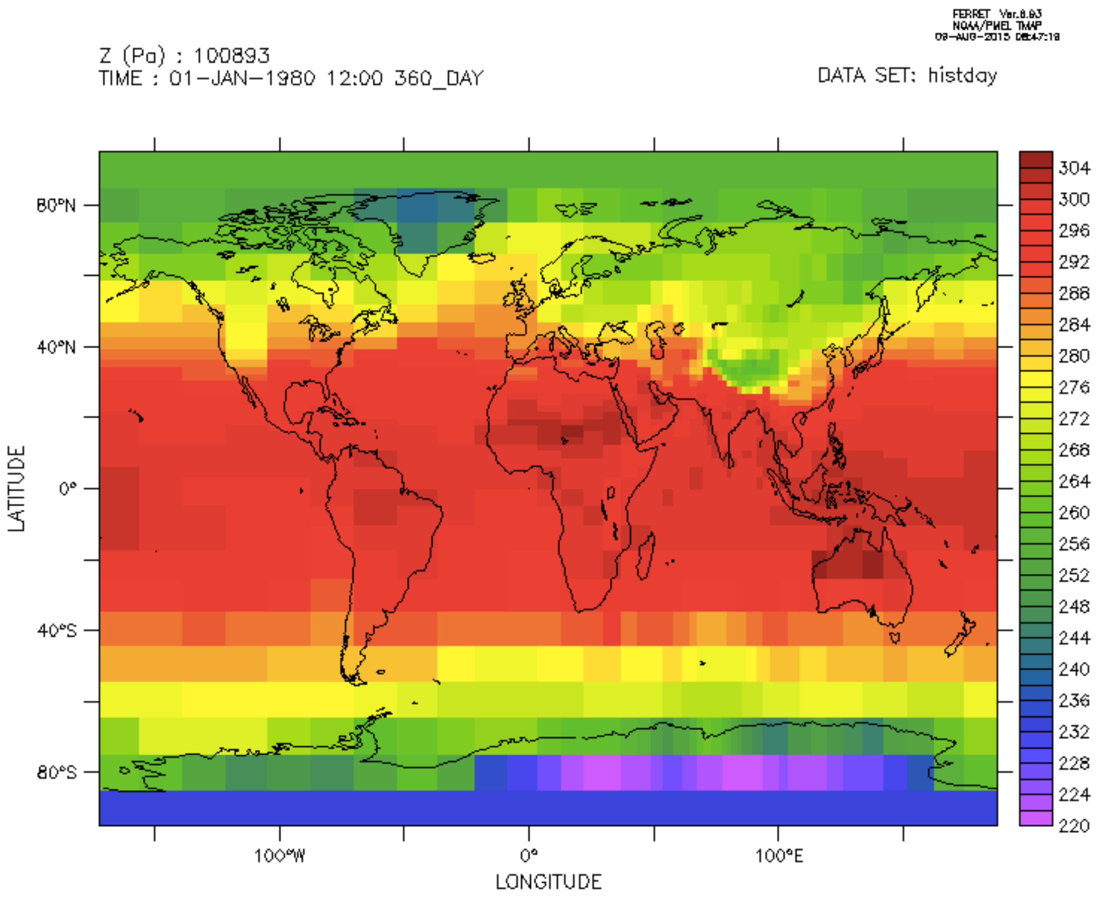
Index of /balaji/LMDZtesting/modipsl/modeles/LMDZ5/TUTORIAL/4000_4000_80_22_200_200

Name	Last modified	Size	Description
Parent Directory		-	
gcm.out	2015-08-09 08:43	382K	
model.gif	2015-08-09 08:43	33K	
model.jnl	2015-08-09 08:43	97	
zoom.gif	2015-08-09 08:21	34K	
zoom.jnl	2015-08-09 08:21	95	

Apache/2.4.7 (Ubuntu) Server at 192.168.0.103 Port 80

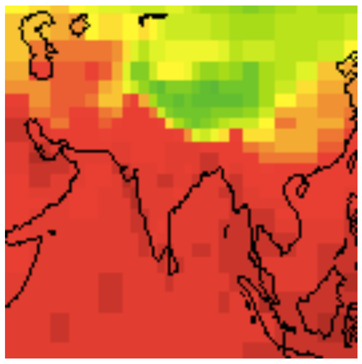
When the model finishes running, `model.py` will display this page:

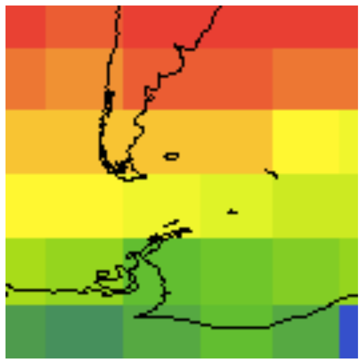
Here is your image:



Air temperature (K)

Note the difference in resolution between the zoomed region (hint: the zoom is centered around India) and a non-zoomed region like Antarctica or South America:





Both images are 132x132 pixels.

Summary

This program offers users the capability to run the LMDZ GCM model through the web even to those not intimately familiar with the model itself, or shell programming.

The project offered me the opportunity to manipulate a climate model in order to understand how it works; being able to modify certain parameters and discern the effects of those modifications gave me great satisfaction. In addition, I became much more familiar with Shell programming (beyond `cd` and `ls`) and learned some new Python syntax (particularly with regards to the `subprocess` module, which I used to call all non-Python programs).

Beyond syntax, however, this project also taught me some general approaches to programming. I learned some techniques for testing code usefully—knowing which cases to cover to make sure the code does what it's supposed to, and knowing what (or where) to check when it does not. I also developed an approach to avoid the unnecessary creation of duplicate files by having the code check for the existence of a preexisting image with the given zoom conditions before running the code to create it (see page 2). This allowed me to preset some sample zoom configurations in order to present my project without long gaps between each section.

Realizing that running the model would take several minutes, I decided to find a way for users to be able to access their image without having to wait for their browser tab to load. Initially I thought I could have them input their email through a form and then send them their image when it was finally created, but this proved much more complicated than I initially thought, so I settled on offering the link to the final image (it would, unless the zoom configurations had already been used, be a broken link when the user first clicked on it) and instructing them to reload it after a minimum time limit. This was not a particularly elegant solution, but under the circumstances it was the most reasonable one I could think of.

If I had more time, I would have liked to allow for more configuration—specifically, for users to be able to select `im` and `jm` values (the number of horizontal and vertical grid points, respectively) as well as `taux` and `tauy` (values that determine how quickly the resolution of the grid changes from zoomed to non-zoomed). At this point, there would be ten data points to configure, and using the system of directory nomenclature that the code uses right now would have been inconvenient. I may have instead assigned each directory an `md5` hash (which would be unique) and instructed the user to keep it safely.

I also would have liked to come up with a nicer way of offering the final `model.gif` to the user at his or her own convenience, as opposed to that of the model. Perhaps I would have tried harder to get the email feature working, although it seemed a little out of my reach at the time.

Acknowledgements

I would like to thank:

Frederic Hourdin, for coming up with the idea for this project and giving me some of the preliminary instructions.
Marie-Pierre Lefebvre, for providing me with instructions to download the model and get it properly running.
Marie-Alice Foujols, for giving me advice and assistance throughout the course of this project.
Venkatramani Balaji, for the generous (if temporary) donation of his laptop.

Notes from Marie-Alice Foujols, internship supervisor

Nalanda spent 5 weeks with us. First, she helped us during an international scientific conference: Our Common Future Under Climate Change, 7-10 July 2015, Paris, France. For 3 days, she helped with the organization of more than 5 posters sessions. Second, she wrote a useful program regarding LMDZ, IPSL's atmospheric general circulation model.

A poster session during the conference requires someone to help take posters from scientists, use identification numbers to find the location of the poster, to set it up and to help scientists to take their posters back once the session was finished. Nalanda demonstrated a lot of skills: fast understanding, efficiency, calm, and curiosity. She was even able to join a couple of scientific sessions, and she followed talks and could summarize them perfectly.

During the second phase of the internship, Nalanda showed us how she likes programming. She understood and studied the main difficulties associated with her project, proposed pragmatic solutions and wrote efficient and useful programs.

Finally, I want to insist on Nalanda's qualities. She demonstrates smart and serious skills, qualities not so common for such a young girl. I really appreciate Nalanda's discussion. She quickly understands difficult scientific and even ethical concepts.