Gemini3D ionospheric modeling tutorial

Michael Hirsch

Boston University Center for Space Physics

2022 June 19

Gemini3D model by Matt Zettergren

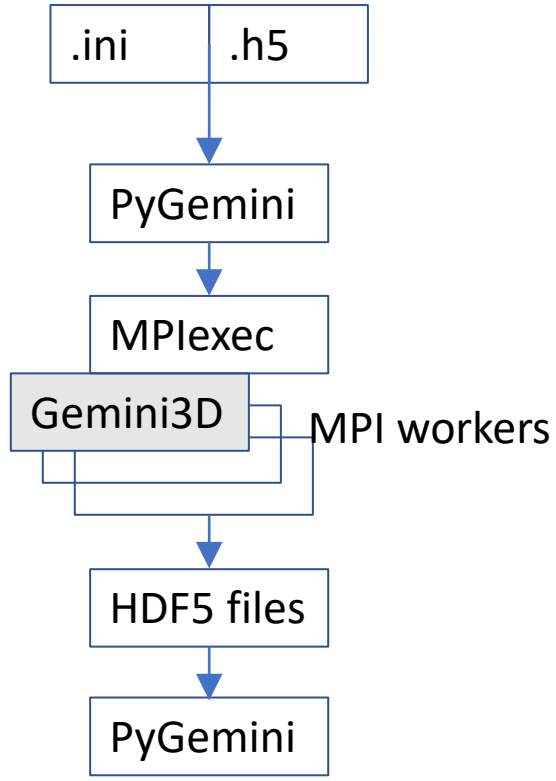
Gemini3D/PyGemini funded in part by NSF CAREER, NASA HDEE ROSES, and DARPA Cooperative Agreement HR00112120003 Boston University subcontract for Embry-Riddle Aeronautical University. This work is approved for public release; distribution is unlimited. The information does not necessarily reflect the position or the policy of the Government.

Background

- Ionospheric dynamics: well-established models cover a range of scales
- Gemini3D is designed to cover ionospheric dynamics from local to regional and global scale, with a key limiting factor being computer system memory and computation time
- Gemini3D grid and model physics fidelity can be user-configured
- Gemini3D can run on a student's laptop or workstation all the way up to the largest HPC
  - Also cloud computing services like Google Colab, AWS, Azure, etc.
- C/C++/Fortran interfaces to most Gemini3D functions allow assembling a custom model ensemble

Simulation setup

- Simulation parameters are specified in a .ini-like text file
- Data files can be loaded as specified in the text file for a particular initialization
  - For example, patches, irregularities, …
- Models such as MSIS00/2.x, HWM, GLOW, etc. can optionally be used to set/drive ionospheric drivers
- Multiple grid types available (cartesian, curvilinear, …) as appropriate for geographic size vs. simplicity/efficiency
  - Near future: adaptive grid mesh refinement



Software Design

- While interfaces such as mpi4py exist, Gemini3D uses a file-based API to interface with scripted languages
  - scale up to large simulations on same or different computing platform
- We use scripted input/output because:
  - Input/output is the code users most frequently change
  - Runtime performance less important for I/O (vast majority of computation time is in simulation itself)

Scripted Interfaces

- Gemini3D C++ function interfaces may allow individual Gemini3D function use from scripted languages
- Using the whole Gemini3D model can typically use 1 GB of memory up to 100s of GB or even terabytes of memory
- A text-file and optional HDF5 data files initialize a simulation
- Milestones allow restarting a simulation
  - e.g. ran over HPC time quota

Scripted Interfaces

- Python (PyGemini) and Matlab (MatGemini) interfaces work essentially equally well with Gemini3D
- Some Gemini3D user groups prefer Matlab, while others prefer Python
- The data files can be seamlessly interchanged between scripting languages
- It is also relatively simple to “port” scripts between Python and Matlab or simply invoke the scripts from the other language using the general Matlab=>Python and Python=>Matlab binary interfaces

Notebooks

- PyGemini, MatGemini, and Gemini3D itself work with the typical “notebooks”
  - Jupyter Notebook for Python, and Matlab’s built-in notebook capability
- Download ABI-compatible Gemini3D binary executable files to cloud instances avoids the need to recompile each time

Computer Requirements

- Gemini3D scales from a 2x2 cell overall grid up to arbitrarily large grid sizes
- For the smallest simulations, less than 1 GB of RAM/disk is needed
- Many 2D simulations and smaller 3D simulations can be run on a laptop or desktop
- Limitation is usually how long user is willing to wait for results—drives users to HPC or workstations with large CPU count

Software Requirements

- standard-conforming C++ and Fortran throughout Gemini3D
- Works on any modern operating system and several modern compilers
- All external libraries are built as part of Gemini3D install
- optional scripts to build Python and/or GCC for restricted environments

Known working:

- GCC >= 7.5, Clang >= 6.0, or Intel oneAPI
- Python >= 3.7 or Matlab >= R2020b

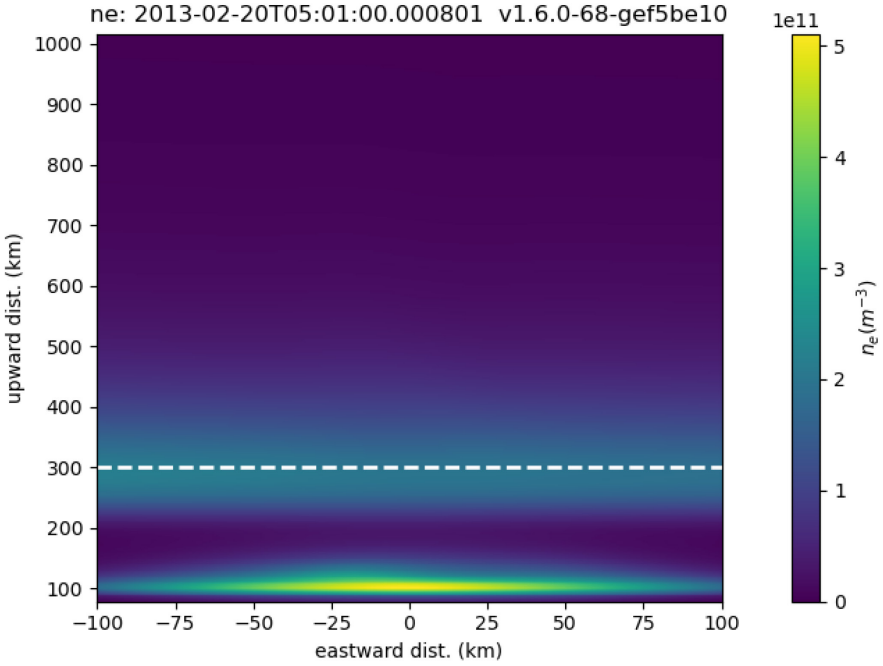
Example: 2D simulation: Python PyGemini

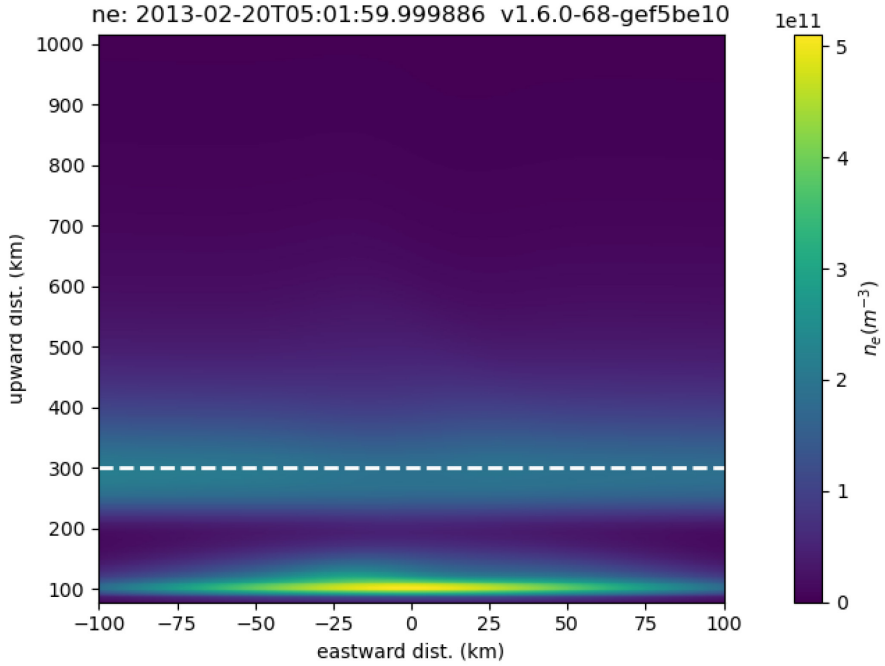
- Almost identical procedure to use Matlab MatGemini
- Let’s use an example from the CI test suite for Gemini3D: https://github.com/gemini3d/gemci/tree/main/cfg/hourly
- We choose mini2dew_fang as it’s one of the fastest and simplest

Simulation: Python PyGemini

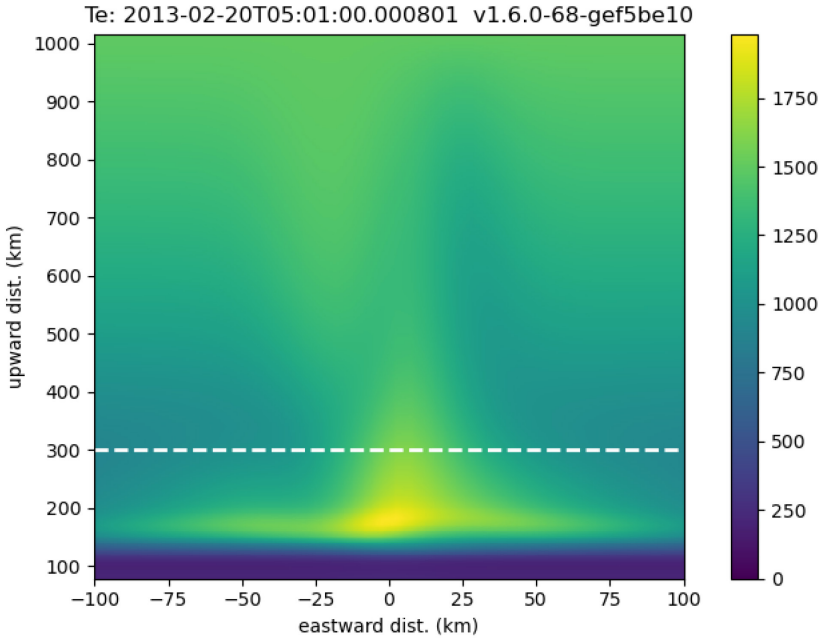
- Set environment variable GEMINI_ROOT to Gemini3D install location
  - PyGemini and MatGemini use this to facilitate easy Gemini3D run on a laptop
- Notebook build example: https://colab.research.google.com/drive/1z3wGzoXH7xSp8gtkUrSBLt09V5FP91NJ?usp=sharing
- PyGemini uses “xarray”, which is popular among geospace Python packages
- HDF5 is used throughout the Gemini3D suite for standard data interchange across code languages and computing platforms
- Plotting is directly to PNG files to significantly speed up and save human effort

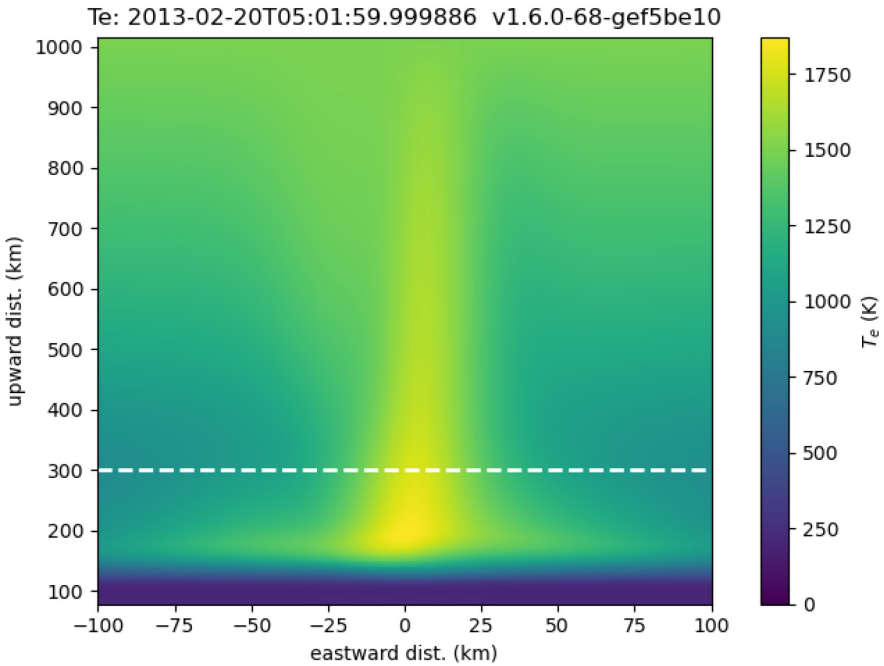
N_e





T_e





Two time steps:
The actual simulation time step is much smaller and dynamic

Community model

- Over 20 code repositories are associated with Gemini3D
  - Many under https://github.com/gemini3d
- Comprehensive unit tests and integrations tests help mitigate bugs introduced by changes
  - OS: MacOS, Linux, Windows
  - Compilers: GCC, Clang, Intel oneAPI
  - Also sweeps across several compiler versions—we've found/reported several compiler bugs across vendors