alexander.pletzer@noaa.gov,
WWW home page: http://ncvtk.sf.net/

Ncvtk: A program for visualizing planetary data

Alexander Pletzer[1,4], Remik Ziemlinski[2,4], and Jared Cohen[3,4]

[1] RS Information Systems
[2] Raytheon
[3] The College of New Jersey
[4] Geophysical Fluid Dynamics Laboratory, Princeton NJ 08542, USA

Abstract. Ncvtk is a visualization tool offering a high degree of interactivity to scientists who need to explore scalar and vector data on a longitude-latitude based grid. Ncvtk was born out of four recent trends: open source programming, the availability of a high quality visualization toolkit, the emergence of scripting, and the wide availability of fast and inexpensive hardware.

1 Why Ncvtk?

In geophysical sciences as in many other areas, the volume of data generated by simulation codes has steadily increased over time. Present climate modeling codes can generate tens of Gigabytes (Gbyte) of data for a single run. Plowing through such amounts of data requires visualization tools that go well beyond simple plotting utilities. The challenge then is to provide a simple visualization tool that can be used by non-experts.

We have developed Ncvtk [1] to offer climate scientists a means to explore data interactively and intuitively. Ncvtk was designed from the ground up to be straightforward to use. No prior programming or other experience is required. We believe users should be no more than a few mouse clicks away from seeing their data. Moreover, in contrast to a number of proprietary programs, Ncvtk is compiled from source and thus can accommodate rapid advances in software libraries, compiler, and hardware. This distinguishes Ncvtk from most commercial visualization tools.

Ncvtk focuses on planetary science data, which lie on a spherical domain and are elevation- and/or time-dependent. Typically, the maximum elevation/depth is three orders of magnitudes smaller than the horizontal length scale. For most purposes, the data are heavily stratified in elevation. Therefore, it is convenient to regard the data as lying flat on the surface of a sphere with the most interesting phenomena taking place on the longitude-latitude plane.

2 Building blocks

We have chosen to build Ncvtk by using the Visualization Toolkit (VTK) [2] as our foundation. VTK is a powerful object oriented, 3D visualization library,

which is open-source and is being actively developed. VTK is known to run on many Unix flavors (Linux x86, ia64, Mac OS X) and Windows. Contrary to other visualization packages, VTK does not provide a visual programming interface or a custom scripting language. Instead, VTK is written in C++ while offering bindings to the Java, Tcl and Python programming languages.

We settled our choice on Python, an object oriented scripting language that offers a vast collection of modules (graphical user interface, regular expression engine, calendar arithmetic, etc.). Such capability is not readily available, out-of-the-box in C/C++. Numerically demanding tasks are carried out using the Python Numeric module [3], whose performance we found to be on par with compiled code. By some fortuitous coincidence, it was realized that VTK methods expecting a raw pointer array can also take a Python Numeric array. This allows communication between Python and C++ without the need for data copying.

The climate simulation codes of our users comply to the Flexible Modeling System (FMS) [4] guidelines, which require data to be written in the self-described NetCDF file format. Attributes of fields such as name, physical units, axes dependency, etc. are stored within the file. Ncvtk assumes that the simulation codes loosely follow the CDC conventions [5] for describing valid data range, missing values, whether the data are packed or not, and whether the grid is rectilinear or warped. NetCDF data may consist of floats, doubles or shorts (when packed). Ncvtk will automatically cast the data to working precision (float or double) without the need for user intervention.

3 Example

Some of Ncvtk's capabilities are now illustrated. Figure 1 was obtained from a run performed by the non-hydrostatic, atmospheric code ZETAC, which solves the compressible fluid equations for velocity and pressure using a terrain following vertical coordinate. This makes ZETAC ideal for studying cyclogenesis in the vicinity of mountain ridges [6].

A color pipeline reading a color map from an external file was used to represent the topology and a piecewise linear opacity function was applied to indicate the presence of clouds. The last layer consists of labeled temperature contours and colored pressure contours, all at surface level. The horizontal wind velocity field at surface level is shown as white arrow glyphs. The opacity of the glyphs increases with wind magnitude. To emphasize the hour of the day, an external light simulating the sun was added.

Each visual pipeline (color plot, opacity, contour, vector, etc.) comes with its own graphical user interface containing a field selector, an elevation slider and specific widgets for user interactivity. Any number of pipelines can be instantiated. A pipeline can be removed simply by destroying the graphical user interface window.

In this onion skin approach, an arbitrary number of fields can be displayed simultaneously, layered on top of each other. Field values may also be probed by moving the mouse over regions of interest. Since only two-dimensional fields

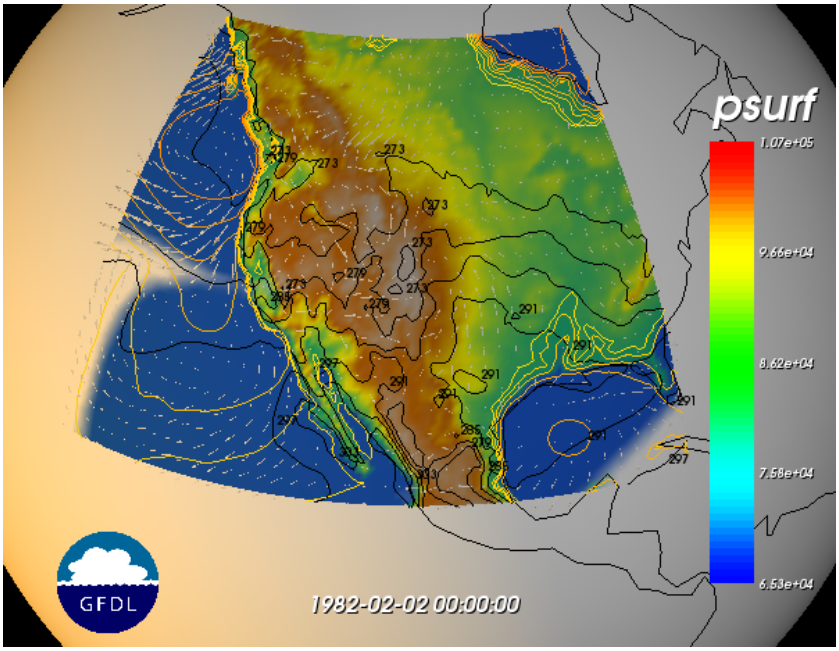

Fig. 1. Non-hydrostatic atmospheric simulation obtained using the ZETAC code.

need to be stored at any given time and elevation, the memory footprint remains modest even in the case of Gbyte-sized data sets. This feature allows Ncvtk to run on personal computers with typically less than 1 Gbyte of memory.

Note that most of the heavy number crunching involves computing structured grid coordinates and evaluating minimum/maximum data ranges across space and time. Recall that the data of a single field can amount to Gbytes. To preserve responsiveness in Ncvtk, the data range calculation is performed by subsampling in time and elevation.

4 Summary

Ncvtk was released in October 2004 and an executable version has since been made available to users at the Geophysical Fluid Dynamics Laboratory. Before Ncvtk, the tool of choice was Ncview [7], a program using color texture to represent scalar fields on a plane. Ncview is straightforward to use and provided our initial inspiration. Like Ncvtk, Ncview reads data from a NetCDF file and is targeted at planetary sciences. Ncview's main advantage over Ncvtk is that it is lightweight and fast. We have attempted to emulate Ncview's ease of use while adding many more features including support for multiple fields, multiple staggered grids, warped coordinates, opacity and contour plot, and vector fields.

A workshop on Ncvtk has recently been organized [8] attracting about 30 participants. This provided constructive feedback to Ncvtk's developers. It appears that scientists especially value accuracy, robustness and ease-of-use.

By implementing Ncvtk in Python, we were able to leverage on a vast body of existing Python modules and so cover significant ground in a few months of development time. Few application specific programming languages that ship with many commercial visualization packages can claim to offer comparable scope to that of a generic programming language. Among the many Python features that played a prominent role in Ncvtk one can cite lists and hash tables to represent object collections, module unit tests to allow classes to be tested in isolation, and on demand, event-driven programming. Thanks to Python's dynamic nature, a scripting programming interface that maps the GUI functionality on a one-to-one basis could be written with little extra effort. In retrospect, Python's biggest strength may have been, however, the error handling that comes built into the language. Python failures do not lead to segmentation violation or other forms of hard crashes, conferring to Ncvtk a high degree of robustness. We believe that many projects could benefit from the shift from compiled code to scripting.

Ncvtk's design emphasizes scalability in performance and functionality. So far we have only scratched the surface of VTK' capability; there remains ample room to add more functionality to Ncvtk. Extensions that are presently being considered include vertical slices of scalar fields, texture based visualization of vector fields, the ability to read files from several NetCDF files simultaneously, support for several scene renderers, a mechanism for adding user extensions (add-ons), and the ability to plot vertical profiles. The challenge will be to allow Ncvtk to evolve while remaining simple to use, robust and predictable.

Acknowledgments

We would like to thank Drs C. L. Kerr and S. T. Garner for providing the impetus for this work and for providing initial test data. Ncvtk is graciously hosted by SourceForge [9], which includes a CVS repository, a bug tracking tool, a feature request tool, and mailing lists. Ncvtk would not exist without the freely available VTK library [2].

References

1. http://ncvtk.sourceforge.net
2. http://public.kitware.com/VTK/
3. http://www.pfdubois.com/numpy/
4. http://www.gfdl.noaa.gov/∼fms/
5. http://www.cdc.noaa.gov/cdc/conventions/cdc_netcdf_standard.shtml
6. Orlanski, I., Gross, B. D.: Orographic modification of cyclode development. J. Atmos. Sci., **51**, 589–611.
7. http://meteora.ucsd.edu/∼pierce/ncview_home_page.html
8. http://ncvtk.sourceforge.net/ncvtk_tutorial.pdf
9. http://sourceforge.net