GeoModelGrids: A storage scheme and API for multiscale Earth models

Brad Aagaard baagaard@usgs.gov
Geologic Hazards Science Center

Poster #186

Availability during Poster Sessions

Unfortunately, I am not available during the scheduled Poster Sessions. However, I am available at earlier times during the day, so please leave a comment or send me an email (baagaard@usgs.gov) if you want to discuss my poster.

Objective

Portable, flexible, and efficient storage of raster-based Earth models

- **Storage requirements**
  - → Self-describing binary format
  - → Allow models with and without topography
  - → Support wide range of geographic coordinate systems
  - → Support depth variation of resolution
  - → Allow vertical resolution independent of horizontal resolution
- **API requirements**
  - → Serial: Only load a subset of the model into memory and use caching
  - → Parallel: Efficient loading of models on massively parallel computers

Motivation

Need standard storage scheme and API for portability, effcient serial and parallel access, and ease of use.

- **CMU Etree** used for 3D USGS San Francisco Bay region models
  - → Efficient binary format ✔
  - → Need CMU Etree library to access data ✘
  - → Cannot read Etree data written on machine with different endian type ✘
- **IRIS EMC**
  - → Transparent, portable binary format via NetCDF ✔
  - → Self-describing format (description, attribution, model information) ✔
  - → Can support efficient parallel access ✔
  - → Hardwired longitude/latitude coordinate system ✘
  - → Limited to uniform resolution grid ✘
- **SCEC UCVM**
  - → Common API for several seismic velocity models ✔
  - → Not a storage scheme ✘
  - → Relies on underlying model-specific APIs ✘
  - → Incompatible with massively parallel access ✘

Status

- **Serial interface** 95% complete with full test coverage
- **Documentation** 90% complete
- **Parallel interface** 0% complete
  - → Identified some bookkeeping differences with LLNL SW4 sfile implementation
  - → Seeking volunteers to help with implementation and/or testing
- **Application to USGS San Francisco Bay region model**
  - → Successfully regenerated v08.3.0 for the detailed domain
  - → TODO: Regenerate v08.3.0 for the regional domain
  - → TODO: Generate updated version with Evan Hirakawa's changes
- **GeoModelGrids on GitHub** https://github.com/baagaard-usgs/geomodelgrids
  - → Documentation and source code continuously updated
  - → Releases with source code and binary packages for Linux and MacOS (coming soon)

External Resources

- HDF5 https://www.hdfgroup.org/solutions/hdf5/
  Widely used, portable, self-describing binary files
- Proj https://proj.org
  Geographic coordinate systems and transformations

Storage Scheme

Key Features

Simple Stack of logical grids with uniform resolution

Fast Index into grid for any location can be computed from metadata alone

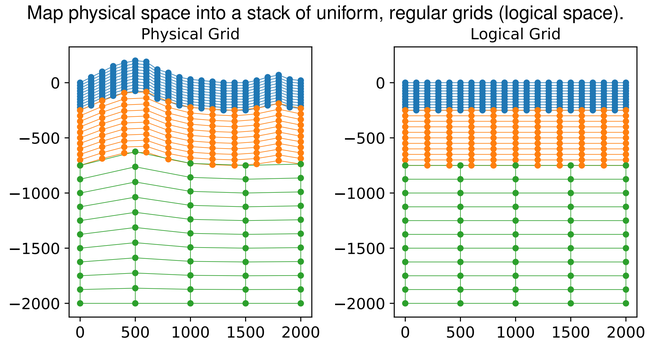
Compact No extra points in "air"

Flexible Independent of the data (values stored in model) and region (any Proj compatible CRS)

Complete Sufficient metadata to describe model and source (with attribution)

Model Representation

Map physical space into a stack of uniform, regular grids (logical space).

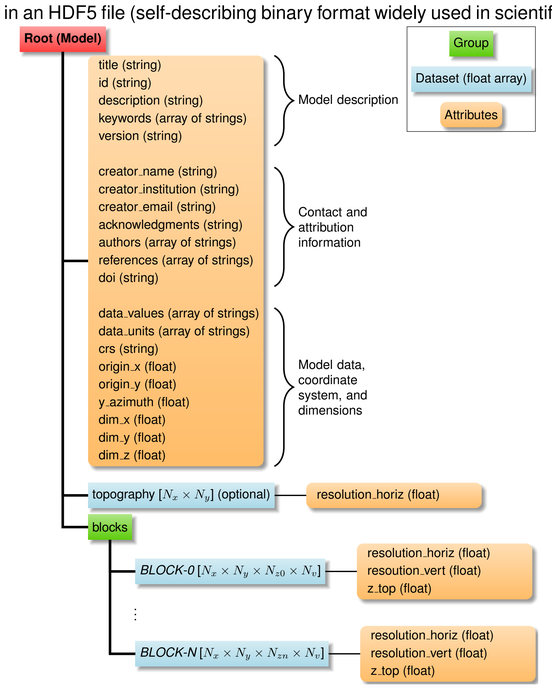


Linear mapping between positions in logical grid, $z_{logical}$, and positions in physical grid, $z_{physical}$.

$$z_{logical} = -dim_z(z_{topography} - z_{physical})/(z_{topography} + dim_z)$$

Implementation

Model is stored in an HDF5 file (self-describing binary format widely used in scientific applications).



Query Interface

C++ API

```
static const std::vector<std::string>& filenames{ "cencalvm_detailed_08.4.0.h5" }; // Set model(s) to query.
static const std::vector<std::string>& valueNames{ "vp", "vs" }; // Set query values and order.
static const char* const crs = "EPSG:4326"; // Points in query will be in lat/lon/elev in WGS84 datum.
// Not shown: Setup arrays (latitude, longitude, and elevation) for coordinates of query points.

geomodelgrids::serial::Query query; // Create query.
query.initialize(filenames, valueNames, crs); // Initialize query.

double values[2]; // Array to store query results.
for (size_t iPt = 0; iPt < numPoints; ++iPt) {
   const double groundSurface = query.queryElevation(latitude[iPt], longitude[iPt]); // Query for elevation of ground surface.
   const int err = query.query(values, latitude[iPt], longitude[iPt], elevation[iPt]); // Query for Vp and Vs.

   const double vp = values[0];
   const double vs = values[1]];
   // Do something useful with Vp, Vs, and elevation of ground surface.
}
```

A C API is also available. See the GeoModelGrids documentation for the complete C++ and C APIs and examples of their use.

Command Line Programs

- Show model metadata

```
geomodelgrids_info [--help] [--description] [--coordsys] [--values] [--blocks] [--all] --models=FILE_0,...,FILE_M
```

- Query for elevation of model ground surface

```
geomodelgrids_queryelev [--help] [--log=FILE_LOG] --values=VALUE_0,...,VALUE_N --models=FILE_0,...,FILE_M \
    --points=FILE_POINTS --output=FILE_OUTPUT [--squash-min-elev=ELEV] [--points-coordsys=PROJ|EPSG|WKT]
```

- Query for model values

```
geomodelgrids_query [--help] [--log=FILE_LOG] --values=VALUE_0,...,VALUE_N --models=FILE_0,...,FILE_M \
    --points=FILE_POINTS --output=FILE_OUTPUT [--squash-min-elev=ELEV] [--points-coordsys=PROJ|EPSG|WKT]

# Example: Query the USGS San Francisco Bay region CMV detailed domain for Vs and Vp with points given in UTM zone 10 NAD83 horizontal
# datum and elevation (m). Inputs points are in points.in and the output is stored in points.out.
geomodelgrids_query --models=cencalvm_detailed_08.4.0.h5 --points=points.in --output=points.out --values=vs,vp \
    --points-coordsys=EPSG:26910

# Input: points.in, easting (m), northing (m), and elevation (m)
570713.0 4150642.0   -200.0
579233.0 4184006.0   -800.0
526196.0 4250240.0  -2500.0

# Output: points.out, easting (m), northing (m), elevation (m), Vs (m/s), Vp (m/s)
  5.707130e+05  4.150642e+06 -2.000000e+02  1.351096e+03  2.930344e+03
  5.792330e+05  4.184006e+06 -8.000000e+02  1.300362e+03  2.872483e+03
  5.261960e+05  4.250240e+06 -2.500000e+03  3.146844e+03  5.223286e+03
```

- Query for model values in virtual borehole

```
geomodelgrids_borehole [--help] [--log=FILE_LOG] --values=VALUE_0,...,VALUE_N --models=FILE_0,...,FILE_M \
    --location=X,Y --output=FILE_OUTPUT [--max-depth=DEPTH] [--dz=RESOLUTION] [--points-coordsys=PROJ|EPSG|WKT]
```

- Query for depth of isosurface, e.g., Z1.0 and Z2.5 (coming soon)