GEO2CGM

Documentation for MATLAB code to convert GEO units to CGM units

Version 1.0

Aaron Hendry

University of Otago
Dunedin, New Zealand

February 2016

Te Whare Wānanga o Otāgo

NEW ZEALAND

Support

Although care has been taken to make this code bug-free, errors and unforeseen corner cases are always a possibility. Please do not hesitate to contact the author (Aaron Hendry - aaron.hendry@otago.ac.nz) with bug reports, comments, queries, etc.

1 Outline

This code is intended to facilitate the conversion from spherical geographic coordinates (i.e. altitude, latitude, longitude) to spherical corrected geomagnetic coordinates in MATLAB. All magnetic field calculations for years between 1900–2015 are done using the IGRF-12 internal field model (Thébault et al., 2015). For years 1590–1900, the *gufm*1 model is used (Jackson et al., 2000); currently no magnetic field calculations are possible prior to 1590. For the years 2015-2115, Gauss coefficients based on the predicted evolution of the geodynamo are used (Aubert, 2015). For years after 2115, secular variation values from 2115 (also from the Aubert (2015) model) are used to extrapolate forward in time, though obviously with increasing uncertainty.

1.1 History

This code is based off a set of FORTRAN function called GEO-CGM derived from GEOPACK-96 - a collection of subroutines which perform various geomagnetic field calculations, originally written by N. A. Tsyganenko.

GEOPACK has undergone many iterations, with the most recent version being GEOPACK-08. GEOPACK is written in FORTRAN which, while allowing for very fast computation, is somewhat unwieldy in modern computing. Compiling FORTRAN on modern computers is not always an easy task, and the number of people fluent enough in FORTRAN to update and debug it is decreasing over time. Converting this code to run in modern environments (for instance MATLAB, IDL, SciPy) makes it much easier to keep this code relevant and updated.

The MATLAB code described within has been written to have no MATLAB toolbox dependencies, and should work on most modern MATLAB installations.

1.2 Algorithm

The algorithm used in this code is the the same as was used in the GEO-CGM FORTRAN code, with some modifications to improve efficiency and readability. An outline of the algorithm is as follows:

For a given latitude, longitude, and starting height:

1. Calculate the magnetic field vector $< B_x, B_y, B_z >$ at the current point (using IGRF, or the appropriate magnetic field model for the given epoch).

2. Step a distance ds up the magnetic field line passing through the current point.

3. Calculate the (dipole) MAG coordinates at the current point, and check if the magnetic latitude is within some tolerance of 0 (i.e. the dipole magnetic equator has been reached).

   (a) If the dipole magnetic equator has not been reached, repeat steps 1-3 until the dipole magnetic equator has been reached.

   (b) If the dipole magnetic equator has been reached, calculate the (dipole) MAG coordinates of the footpoint of the magnetic field line passing through the current point, at the starting altitude. These coordinates are the CGM coordinates of the starting point.

The "step" procedure in step 2 is a numerical integration along the magnetic field line, using some numerical integration routine. In both this code, and the original GEOPACK code, a Runge-Kutta-Merson algorithm is used to calculate the steps up the field line. This step could be sped up slightly by using an Adams predictor-corrector method, which reduces the number of magnetic field calculations that need to be made at the expense of slightly reduced accuracy and an increased memory profile. For simplicity, I have stuck with the original Runge-Kutta method.

This process fails for low-latitudes, where CGM coordinates are no longer defined. The paper by Gustafsson et al. (1992) gives two methods of calculating pseudo-CGM coordinates in the regions where CGM is forbidden:

1. Calculate the closest points in latitude (both North and South) along the same GEO longitude for which CGM coordinates are defined, and interpolate between these values to obtain a CGM latitude/longitude pair, OR

2. Calculate the 'dip equator' for the longitude in question, and interpolate between this equator and the nearest point for which CGM coordinates are defined.

For simplicity, I have implemented the first of these methods - the GEO-CGM code used the second method. This results in some significant differences between the produced CGM coordinates in these regions, however as CGM is technically not defined in these regions, neither approach is any more or less accurate.

In the Gustafsson et al. (1992) paper, they mention that above $\sim 10\,R_E$ the IGRF field becomes essentially dipolar, as as such "...the tracing of the field line from the high-latitude area could be stopped on the 10 Re surface and the dipole field line might be restored from this surface to determine the corrected geomagnetic coordinates for that area". This STOP criterion was implemented in GEOPACK (although due to how they have implemented it, it actually stops at $> 11\,R_E$ rather than $> 10\,R_E$). Note that we have NOT implemented this STOP case, so the tracing will continue to the top of the field line. However, to prevent a runaway case in the MATLAB code where the tracing continues forever, we have included a hard-limit of 1000 steps for any given point. In reality we believe this limit will likely never be reached.

1.2.1 IGRF calculation

The IGRF coordinates for a point are calculated from the scalar field V:

$$V(r,\theta,\phi,t) = R_E \sum_{n=1}^{N}\sum_{m=0}^{n}\left(\frac{R_e}{r}\right)^{n+1}\left[g_n^m(t)\cos m\phi + h_n^m(t)\sin m\phi\right]P_n^m(\cos\theta)$$

where R_E is the radius of the Earth (defined as 6371.2 km) r is the altitude of the point (km), θ is the geographic co-latitude, ϕ is the geographic longitude, $g_n^m(t)$ and $h_n^m(t)$ are the Gauss coefficients for time t, and $P_n^m(x)$ are the Schmidt quasi-normalized associated Legendre functions of degree n and order m.

$P_n^m(x)$ is somewhat slow to calculate, and must be recalculated every time IGRF is called with a different latitude. To speed this process up, we pre-calculate $P_n^m(x)$ to create a look-up table, and interpolate between these pre-calculated values to get a close approximation to the actual values of $P_n^m(x)$. The error introduced by this process is negligible (i.e. typically $< 10^{-8}$ degrees), while the speed up is significant (i.e. often cutting processing time in half).

When the magnetic field calculation process is called, $g_n^m(t)$ and $h_n^m(t)$ must be calculated. For the IGRF model, $g_n^m(t)$ and $h_n^m(t)$ are provided at 5 yearly intervals from 1900 through to 2015 (as of the time of writing), with new coefficients produced every 5 years. Linear interpolation is used to produce coefficients for years in between the 5-year milestones. For the other models, similar interpolation is done.

The GEO-CGM code (and the original GEOPACK code) interpolated $g_n^m(t)$ and $h_n^m(t)$ using only the year of the date in question - this results in a discontinuity at the year boundaries. For instance, the same $g_n^m(t)$ and $h_n^m(t)$ would be used for any date in 2010, with a sudden change in the coefficients at the start of 2011. To produce a more realistic variation, we use the entire date to interpolate the $g_n^m(t)$ and $h_n^m(t)$ values, resulting in a smooth change from day to day and year to year. The original GEO-CGM behaviour can be reproduced by using the 1st of January of whatever year as the epoch date.

The GEO-CGM code (and the original GEOPACK code) also only use a subset of the IGRF coefficients, using fewer coefficients at higher altitudes, I presume to speed up the IGRF calculations. I do not do this, instead using as many coefficients as possible (13 for the year 2000 onwards). The increase in computational time is fairly insignificant, and could potentially offer greater accuracy.

Finally, the GEO-CGM code that I was given used single-precision floating point values in all of its calculations. MATLAB by default uses double-precision floating points, which offer a far greater degree of accuracy. Though the difference is very small on a case-by-case basis, these small differences add up over the numerical integration process, resulting in measurably different results from otherwise identical code.

There are several differences between the GEO-CGM code and this MATLAB code, which result in slightly different calculated CGM latitude and longitude values. The differences are most pronounced near the geomagnetic poles (where the geomagnetic equator is not well defined) and near the equatorial region (where CGM coordinates are 'forbidden'). For $2 < L < 6$, the variance between the GEOPACK

and MATLAB values is typically $< 0.02^\circ$ CGM latitude and $< 0.05^\circ$ CGM longitude. For $6 < L < 11$, the variance is typically $< 0.04^\circ$ CGM latitude and $< 0.1^\circ$ CGM longitude. At higher L-shells, the difference between the GEO-CGM and MATLAB code becomes more pronounced, possibly due to the difficulty associated with finding the geomagnetic equator in these regions. For $L > 11$, the variance is typically $< 0.1^\circ$ CGM latitude and $< 0.5^\circ$ CGM longitude. Near the equatorial region, due to the different interpolation methods used the variance is large, often with several degrees difference between the two.

There are slight differences in the L-shells calculated by the two methods, as well. As with the latitudes, the differences in the equatorial regions are complicated and often significant due to the different interpolation methods used. For $2 < L < 10$, the average difference in L-shell is $\sim 0.005L$. For $10 < L < 16$, the average difference in L-shell is $\sim 0.03L$. For $L > 16$, the GEO-CGM code returns NaN values, so no further comparison can take place.

Finally, there are slight differences between the models from 1990-2010, as the original GEO-CGM code was using IGRF coefficients, whereas the new code uses DGRF values for these years (where the DGRF values are the "definitive" values).

2 Using the code

This code is designed to be simple to use - call the `GEO2CGM` function with a set of latitudes, longitudes, a start height, and a date, and receive CGM latitude and longitude values as a result. Note that all latitudes and longitudes are in degrees.

The following code snippets describe a set of use cases:
Calculating a single CGM coordinate:

```
% Calculate the CGM coordinates of a single point
lat_geo = -78.5; % Geocentric latitude (in degrees)
lon_geo = 156.4; % Geocentric latitude (in degrees)
height = 110; % start height in km
epoch = datenum(2013,1,5); % Date to calculate for
[lat_cgm,long_cgm,l_shell] = geo2cgm(lat_geo,lon_geo,height,epoch);
```

Calculating multiple CGM coordinates:

```
% Calculate the CGM coordinates of multiple points
% If you want to calculate CGM coordinates for a lot of latitudes
% and only a single longitude (or vice versa), you only have to
% specify the longitude once
lat_geo = [45,50,55,60,65,70]; % Geocentric latitude (in degrees)
lon_geo = 90; % Geocentric latitude (in degrees)
height = 110; % start height in km
epoch = datenum(2013,1,5); % Date to calculate for
[lat_cgm,long_cgm,l_shell] = geo2cgm(lat_geo,lon_geo,height,epoch);
```

Calculating a grid of CGM coordinates:

```
% Calculate the CGM coordinates of a grid of latitudes and longitudes
lat_range = -88.75:0.5:88.75; % Geocentric latitude (in degrees)
lon_range = 0:0.5:360; % Geocentric latitude (in degrees)
```

```
[lat_geo,lon_geo] = meshgrid(lat_range,lon_range); % Generate the grid
height = 0; % start height in km
epoch = datenum(2013,1,5); % Date to calculate for
[lat_cgm,long_cgm,l_shell] = geo2cgm(lat_geo,lon_geo,height,epoch);
```

Due to the efficiency with which MATLAB deals with, it is much faster to calculate a vector/matrix of latitudes and longitudes than it is to calculate them all individually.

Note that to use this code, the following files must all be located on your MATLAB path:

- `cart2sph.m`
- `dipole.m`
- `geo2cgm.m`
- `geo2mag.m`
- `igrf.m`
- `igrf_coeff.mat`
- `igrf_gh.m`
- `igrf_pq.m`
- `mag2geo.m`
- `sph2cart.m`
- `step.m`
- `trace_fieldline.m`

References

Aubert, J. (2015), Geomagnetic forecasts driven by thermal wind dynamics in the Earth's core, in: *Geophysical Journal International 203*(3), 1738–1751, doi:10.1093/gji/ggv394, eprint: http://gji.oxfordjournals.org/content/203/3/1738.full.pdf+html.

Gustafsson, G., N. Papitashvili, and V. Papitashvili (1992), A revised corrected geomagnetic coordinate system for Epochs 1985 and 1990, in: *Journal of Atmospheric and Terrestrial Physics 54*(1112), 1609–1631, doi:http://dx.doi.org/10.1016/0021-9169(92)90167-J.

Jackson, A., A. R. T. Jonkers, and M. R. Walker (2000), Four centuries of geomagnetic secular variation from historical records, in: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 358*(1768), 957–990, doi:10.1098/rsta.2000.0569, eprint: http://rsta.royalsocietypublishing.org/content/358/1768/957.full.pdf.

Thébault, E. et al. (2015), International Geomagnetic Reference Field: the 12th generation, in: *Earth, Planets and Space 67*(1), 1–19, doi:10.1186/s40623-015-0228-9.