Sherical Harmonic Representation of Earth Elevation Data

Charles Bouillaguet

v1.0 — September 13th 2022

1 Preliminaries

Let S denote a sphere (of any positive radius, it does not matter) in $\mathbb{R}^3$. We consider functions $f : S \to \mathbb{R}$ that associate a number to each point of the sphere. We use *spherical coordinates* (see fig. 1). A point on the sphere (ϕ, λ) is given by its latitude ϕ and its longitude λ. Angles are expressed in radians.

We are particularly interested in the function f that gives the altitude of each point on planet Earth. For instance, the "place Jussieu" is located at $\phi = 0.85252611$ and $\lambda = 0.04110117$, and the elevation of the ground is $f(\phi, \lambda) = 37.3$ meters. While it is possible to perform measurements and obtain the value of f on many points, there is no simple mathematical formula that gives the value of f. However, earth elevation data has been acquired by satellite measurements with great accuracy, and some of this data is publicly available.

Any function on the sphere that is sufficiently regular (e.g. square-integrable) can be written:

$$f(\phi, \lambda) = \sum_{\ell=0}^{+\infty} \sum_{m=0}^{\ell} P_{\ell m}(\sin \phi) \left[C_{\ell m} \cos m\lambda + S_{\ell m} \sin m\lambda \right] \tag{1}$$

Where $C_{\ell m}$, $S_{\ell m}$ are (fully normalized) *spherical harmonic* coefficients of degree ℓ and order m; $P_{\ell m}$ are the (fully normalized) Associated Legendre Functions of degree ℓ and order m.

These coefficients are the analogs of Fourier coefficients. Obtaining this representation can be

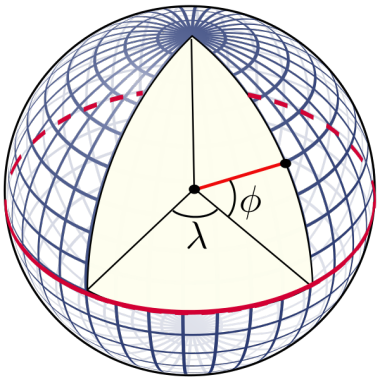

Figure 1: Spherical coordinate system.

seen as some kind of Fourier transform on the sphere. The (infinite) sequence of $C_{\ell m}$ and $S_{\ell m}$ coefficients thus describe the function.

Truncating the series in (1) to a finite degree $\ell_{\max}$ yields only a finite numbers of coefficients. Storing them using (double-precision) floating point numbers, we hope to construct a compact representation of f, and obtain a way to evaluate f everywhere, with a reasonable accuracy. Of course, the higher the degree $\ell_{\max}$, the better the approximation.

The Associated Legendre Functions $P_{\ell m}(\sin\phi)$ can be computed in many ways; the simplest (which is reasonably stable numerically) consists in using a recurrence relation. Let $x = \sin\phi, y = \cos\phi$ and $P_{\ell m}$ denotes the result of the function *evaluated on* x.

$$\begin{aligned} P_{00} &= 1 \\ P_{\ell\ell} &= -y\sqrt{1 + 1/2\ell} \cdot P_{\ell\ell} \\ P_{(\ell+1)\ell} &= x\sqrt{2\ell + 3} \cdot P_{\ell\ell} \\ P_{\ell m} &= a_{\ell m}\left(xP_{(\ell-1)m} - b_{\ell m}P_{(\ell-2)m}\right) \end{aligned}$$

Where $a_{\ell m}, b_{\ell m}$ are constants given by

$$\begin{aligned} a_{\ell m} &= \sqrt{\frac{4\ell^2 - 1}{\ell^2 - m^2}} \\ b_{\ell m} &= \sqrt{\frac{(\ell-1)^2 - m^2}{4(\ell-1)^2 - 1}} \end{aligned}$$

2 Recovering the Spherical Harmonic coefficients

We provide you with four `.csv` files containing earth elevation data (from NASA's ETOPO1 dataset) at various spatial resolution.

File name	Spatial resolution	# points	size
`ETOPO1_small.csv`	116Km	64,800	3.2MB
`ETOPO1_medium.csv`	38Km	583,200	28MB
`ETOPO1_high.csv`	11Km	6,480,000	311MB
`ETOPO1_ultra.csv`	2Km	233,280,000	11GB

These files contain lines with $(\lambda, \phi, f(\phi, \lambda))$. The numbers are separated by tabs.

Using any of these datasets, you have to build the most precise spherical harmonic representation of the underlying mathematical function. In other terms, you must compute the spherical harmonic coefficients for the highest possible degree $\ell_{\max}$. You can assess the quality of the approximation you have obtained by trying to recompute the elevation at given points, and comparing that with the actual value.

For each data point, we have an equation:

$$f(\phi_i, \lambda_i) = \sum_{\ell=0}^{\ell_{\max}} \sum_{m=0}^{\ell} P_{\ell m}(\sin\phi_i)\left[C_{\ell m}\cos m\lambda_i + S_{\ell m}\sin m\lambda_i\right] \tag{2}$$

Here, the only unknowns are the $C_{\ell m}, S_{\ell m}$, and they appear linearly, so this is in fact a system of *linear equations*. It has $(\ell_{max} + 1)$ unknowns (observe that the value of $S_{\ell 0}$ does not matter).

Using all the available data points yields an *overdetermined* linear system (more equations than unknowns). This system is likely to *not* have any solution at all, because of the limited precision of the data points, and because a reduced-degree model only approximates the actual function. Therefore, we will compute a *best possible* solution, namely a solution that minimizes the norm of the error $\|Ax - b\|$. The more data points are used, the best the resulting approximation will be.

Note that if you choose to use a subset of data points from one of our files, you have to be careful to pick equally spaced points (if you take the first 1000 points that are concentrated in a specific location of the sphere, then the model you will compute will be horrible everywhere else).

There are two parameters that can be tuned: the order of the approximation ($\ell_{\max}$) and the number of data points used (let's call this N). Adjusting these two values yields a trade-off between speed and accuracy.

The matrix A is completely dense. It requires $8N(\ell_{max} + 1)^2$ bytes of memory. Computing the least-square solution of $Ax \approx b$ is done through the computation of a QR factorization of the matrix A. This requires about $2N(\ell_{max} + 1)^4$ FLOPS. Both time and space can be limiting.

3 What you Have to Do

We provide you with

- The datasets mentioned above.
- A `model.c` sequential C program that reads N points from a dataset, computes a degree-$\ell_{\max}$ model and stores it in a file.
- A `validate.c` sequential C program that reads a model from a file, reads a full dataset and computes an "accuracy score".

Your goal is to compute a model with the best possible accuracy (on the "ultra" dataset).

If you don't parallelize these programs, you will be fairly limited in the accuracy that can be obtained. Obtaining the best possible accuracy requires dealing with larger and larger problems, and writing a parallel version of the program that computes the model.

You can use MPI, OpenMP, or whatever you want. You may use the external libraries that are usually available in computing centers.

Whatever you do, you **must** :

1. *Describe* what you have done.
2. *Measure* the performance of your implementation and its scalability (compared to the sequential program we gave you).
3. *Comment* these results: are they good or not? If not, why? Can you design an experiment that would confirm your opinion? Would it have been possible to tell in advance what has happened?

Your work must be submitted by Friday, November 11th before 23:59 (using Moodle). You must submit the best model you have computed, the source code used to obtain and validate it, a

`Makefile` that compiles your code (without errors nor warnings, even with `-Wall`), and a ≈ 5 pages report in `.PDF` format that explains what you have done.

Please follow our guidelines about writing reports (on Moodle).

4 Some Last Precisions

- You can work in pairs (submit one report with both names)
- You **MUST** respect the grid5000 usage policy. Do big computations at night.
- In you believe that you have found an error in our programs (it does happen), or if you and your classmates are facing a common technical problem, don't hesitate to contact us.