Parallel Programming Project 1

VOYLES Evan

November 20, 2022

1 Preliminaries

Any sufficiently regular function defined on a sphere can be written

$$f(\theta,\phi)=\sum_{l=0}^{+\infty}\sum_{m=0}^{l}\bar{P}_l^m(\cos\theta)[C_l^m\cos m\phi+S_l^m\sin m\phi]. \tag{1}$$

Where C_l^m, S_l^m are *spherical harmonic* coefficients (also referred to as Laplace series coefficients) of degree l and order m; $\bar{P}_l^m$ are the fully normalized Associated Legendre Functions of degree l and order m; θ is the polar angle and ϕ is the azimuthal angle, as per ISO convention[1].

The project proposal suggests computing the coefficients C_l^m, S_l^m by setting up an overdetermined linear system of equations and finding the least squares solution. Unfortunately, this problem runs in $O(Nl_{max}^4)$ time and requires about $8N(l_{max}+1)^2$ bytes of memory. That is absurdly prohibitive. Furthermore, there are no *self-evident* parallelization techniques to speed up computation time. In this report we propose an alternative approach to computing the Laplace series coefficients C_l^m and S_l^m that runs in $O(Nl_{max}^2)$ time complexity with trivial parallelization models.

2 Alternative Approach

The coefficients C_l^m and S_l^m are defined as the orthogonal projection of $f_(\theta,\phi)$ onto the basis functions (spherical harmonics Y_l^m). Written in terms of the real spherical harmonics:

$$C_l^m=\int_0^{2\pi}\int_0^{\pi}f(\theta,\phi)P_l^m(\cos\theta)\cos(m\phi)\sin\theta d\theta d\phi$$
$$S_l^m=\int_0^{2\pi}\int_0^{\pi}f(\theta,\phi)P_l^m(\cos\theta)\sin(m\phi)\sin\theta d\theta d\phi$$

2.1 Numeric Integraion

We first remark that the NASA data sets are provided as discretizations of $[-\frac{\pi}{2},\frac{\pi}{2}]\times[-\pi,\pi]$, represented as a $n_\theta\times n_\phi$ equidistant grid. For example, for `ETOP01_small.csv`, $n_\theta=180$, $n_\phi=360$, and $N:=n_\theta n_\phi=64000$; We can easily approximate these 2 dimensional integrals (using the midpoint rule?) via numeric quadrature rules

$$\begin{aligned}C_l^m&\approx\sum_{i=1}^{n_\phi}\sum_{j=1}^{n_\theta}f(\theta_j,\phi_i)P_l^m(\cos\theta_j)\cos(m\phi_i)\sin\theta_j\Delta\theta\Delta\phi\\&\approx\Delta\theta\Delta\phi\sum_{i=1}^{n_\phi}\cos(m\phi_i)\sum_{j=1}^{n_\theta}f(\theta_j,\phi_i)P_l^m(\cos\theta_j)\sin\theta_j\end{aligned}$$

[1] We have chosen the ISO standard coordinate system to ensure the integrity of future computations. For a discussion on transforming the (ϕ,λ) coordinates provided by NASA, see Appendix A

$$\begin{aligned}S_l^m &\approx \sum_{i=1}^{n_\phi}\sum_{j=1}^{n_\theta} f(\theta_j,\phi_i)P_l^m(\cos\theta_j)\sin(m\phi_i)\sin\theta_j\Delta\theta\Delta\phi \\ &\approx \Delta\theta\Delta\phi\sum_{i=1}^{n_\phi}\sin(m\phi_i)\sum_{j=1}^{n_\theta} f(\theta_j,\phi_i)P_l^m(\cos\theta_j)\sin\theta_j\end{aligned}$$

This method computes *individual* Laplace series coefficients in $O(N)$ time! Thus, computing the coefficients for a Discrete Laplace Series (DLS) of order $l_m ax$ runs in $O(Nl_{max}^2)$. More interestingly, each C_l^m or S_l^m can be computed **independently** of any other coefficient. Whereas solving the linear least squares problem requires computing every C_l^m and C_l^m in a single operation, this method allows us the ability to compute any arbitrary C_l^m directly. Why do we care? If I alraedy have the first 500 C_l^m coefficients computed and stored in a text file, I can *directly* compute the 501st in $O(N)$ time. Furthermore, this method gives rise to obvious and easy-to-implement parallelization schemes.

2.2 Comparison with Linear Squares

Tested on the `ETOPO1_small.csv` data set with $l_{max} = 20$, the model computed via numerical integration had an average error of about 2 times that of the least squares approach, but ran over 100 times faster.

TODO

- Provide more detailed numeric comparison
- Provide timing analysis

3 Improvements to the model

The least squares method provides a more accurate model, but runs in $O(Nl_{max}^4)$ time. In this section we propose improvements to our model computed via quadrature.

3.1 Gradient Descent

We can apply methods of optimization to minizmize the mean squared error (MSE) of our model. To begin we explicitze the MSE then compute its gradient

TODO I have already derived the gradient of the MSE with respect to all C_l^m and S_l^m on paper

- Write up derivation in latex

Computing the gradient vector runs in $O(Nl_{max}^2)$ time

3.1.1 Stochastic Gradient Descent

Computing the gradient with respect to a single C_l^m requires summing up certain values for all N points of the data set. For larger data sets (e.g. $N_{high} = 1800 * 3600 = 6480000$), this computation can become intractable. We can speed up our gradient computation by computing as estimate of the gradient, $\hat{\nabla}$MSE, by randomly sampling n points from our entire data set. This reduces the computation of a gradient vector from $O(Nl_{max}^2)$ to $O(nl_{max}^2)$. This is an **insane** improvement to the run time of our algorithm. Instead of summing all 6480000 points of the large data set, we could fix n at say $n = 1000$ instead, sacrificing accuracy for a speedeup of 6,480.

3.2 MCMC Algorithms

In initial tests my gradient descent learning is *sometimes* reducing the MSE but other times it's increasing it... If I don't figure out exactly what's going wrong then I plan on implementing MCMC algorithms like Gibb's sampling to explore the sample space and only accept proposal vectors that decrease the MSE.

4 Parallelization

TODO (trivial)

5 Conclusion

Instead of calculating the model with a direct method that runs in $O(Nl_{max}^4)$ time, We propose a hybrid model that first directly computes the coefficients in $O(Nl_{max}^2)$ time then iteratively improves the MSE via stochastic methods running in $O(n_{training}n_{sample}l_{max}^2)$ time.

Results are to follow :)

A Spherical Coordinates Transformation

The data sets `ETOPO1_*.csv` provided by NASA represent spherical coordinates in an unconventional format:

$$(\phi, \lambda) \in [-\frac{\pi}{2}, \frac{\pi}{2}] \times [-\pi, \pi]$$

where ϕ refers to the latitude and λ refers to the longitude. In this section we outline the transformation from NASA's coordinate system to the ISO standard representation and prove equivalence between (1) and the Project presentation's $f(\phi, \lambda)$.

To avoid confusion, we denote the spherical coordinate pair in ISO coordinates as $(\theta_{iso}, \phi_{iso})$ and NASA's coordinate scheme as $(\phi_{nasa}, \lambda_{nasa})$. We remark that the polar angle θ_{iso} is simply equal to $\frac{\pi}{2} - \phi_{nasa}$. We corroborate this fact by remarking that a latitude of $\phi_{nasa} = 0$ corresponds to a polar angle of $\phi_{iso} = \frac{\pi}{2}$. Conveniently, the longitude λ_{nasa} is equivalent to the azimuthal angle ϕ_{iso}.

Proposition 1. Let the spherial coordinate pairs $(\theta_{iso}, \phi_{iso})$, $(\phi_{nasa}, \lambda_{nasa})$ be defined as above.

$$f(\theta_{iso}, \phi_{iso}) \equiv f(\phi_{nasa}, \lambda_{nasa})$$

Proof. $f(\phi_{nasa}, \lambda_{nasa})$ is defined in the Project proposal as

$$\begin{aligned}
f(\phi_{nasa}, \lambda_{nasa}) &= \sum_{l=0}^{+\infty} \sum_{m=0}^{l} \bar{P}_l^m(\sin\phi_{nasa})[C_l^m \cos m\lambda_{nasa} + S_l^m \sin m\lambda_{nasa}] && (2) \\
&= \sum_{l=0}^{+\infty} \sum_{m=0}^{l} \bar{P}_l^m(\sin(\pi/2 - \theta_{iso}))[C_l^m \cos m\phi_{iso} + S_l^m \sin m\phi_{iso}] && (3) \\
&= \sum_{l=0}^{+\infty} \sum_{m=0}^{l} \bar{P}_l^m(\cos\theta_{iso}))[C_l^m \cos m\phi_{iso} + S_l^m \sin m\phi_{iso}] && (4) \\
&\equiv f(\theta_{iso}, \phi_{iso}) && (5)
\end{aligned}$$

□