Parallel computation of spherical harmonics at single point
単独点における球面調和関数の並列計算

Toshio Fukushima, National Astronomical Observatory
福島登志夫（国立天文台）

Abstract

We present a formulation of the parallel computation of spherical harmonics and their first order partial derivatives at a single point. The key techniques used are the vector computation of Helmholtz polynomials, the folding of a trapezoidal do loop into box-shaped one, and the interleaved computation of Chebyshev polynomials and simple powers. These are easily implemented by a slight modification of existing programs and embedding a few OpenMP directives. The formulation significantly speeds up all kinds of computation using spherical harmonics unless the maximum degree and/or order is small. For example, the resulting parallel computation of the acceleration vector of a low-altitude satellite, GRACE, caused by the EGM2008 geopotential model of 2190 degrees and 2159 orders and conducted at a PC with a quad-core 8-thread processor runs 3.3 times faster than the serial computation.

単独点における球面調和関数およびその 1 階偏微分係数の並列計算法を述べる。主な手法はヘルムホルツ多項式のベクトル計算と台形形状の do-loop の折りたたみ、ならびにチェビシェフ多項式および単純ベキ単項式の間引き計算である。これらの手法は、従来のプログラムに対して OpenMP 命令文の活用により、微小な修正だけで実現可能である。本方法により、最大次数が極端に少なくない限り、球面調和関数の計算は確実に加速される。例として、2190×2159 次まで展開されている EGM2008 地球重力場の下で低高度衛星 GRACE の軌道計算を行う場合、4 コア 8 スレッドの商用 PC で並列計算を実施した場合、3.3 倍のスピードアップが実現された。

1 Introduction

The spherical harmonic expansion is one of basic mathematical tools in various fields of science and engineering (Olver et al., 2010). Especially it is commonly used to express the Newtonian gravitational potential of a nearly spherical body such as the Earth, the Moon, planets, dwarf planets, and large natural satellites (Kaula, 2000). Once the physical parameters such as the Stokes coefficients are given, it is straightforward to evaluate the acceleration vector through the partial derivatives of the potential.

However, their numerical evaluation requires quite a few computational labor. Indeed, the computational amount of the partial derivatives occupies more than 99% of the total computational amount of the satellite orbit integrations even if the maximum degree and/or order of the spherical harmonics are as small as a few tens. The situation significantly intensifies if these numbers become larger. An extreme case occurs in dealing with a latest geopotential model EGM2008 (Pavlis et al., 2008) where they exceed 2000. As a result, it is heavily time-consuming to conduct precise and/or long-term orbit integrations of low-altitude Earth satellites such as GRACE. The same thing can be said for the evaluation of the potential itself as well as its second and higher order partial derivative tensors (Casotto and Fantino, 2007; Fantino and Casotto, 2009).

Unfortunately the recent increase in the clock rate of CPUs of commercial computers have been hindered by two kind of difficulties: the theoretical barrier of signal propagation time in electric circuits and the practical problem of cooling the computer chips (Domeika, 2008).

These have driven the computer industry to shift from a single to multi-processor computers. In other words, the recent computers are becoming not faster but *wider* (Nyland et al., 2007). For example, latest cutting-edge supercomputers are composed of huge number of processors. Even inexpensive consumer PCs are with a multi-core and/or multi-thread CPU. This tendency will be enhanced in the near future under the name of many-core processors (Domeika, 2008). Thus it is sincerely required to develop efficient parallel computing schemes to deal with large-scale number-crunching problems like the evaluation of geopotential model of ultra-high degree and order.

If we permit degrades in computing precision and loss of mathematical rigorousness, there do exist some fast computation techniques like fast multipole expansions (Greengard and Rokhlin, 1996). However, they are not suitable for precise computations as required in geodesy and celestial mechanics. Of course, there are a few works on the precise parallel computation of spherical harmonic synthesis/analysis (Xiao and Lu, 2007). They are using the technique to execute the same serial computation program at large number of points concurrently. An example is the simultaneous evaluation of geopotential at two-dimensional grid points in latitude and longitude on a surface of constant radius. Namely a gain by parallelism in these methods is in the multiplicity of evaluating points. Obviously this kind of parallelization does not work in case of orbital integrations where we need to compute the acceleration vector using the spherical harmonics and their partial derivatives at a single point.

In order to overcome this situation, we present here a parallel scheme to compute the given spherical harmonics at a single point. It consists of three techniques. The first is a vector computation of a variant of the associated Legendre functions named the quasi-normalized Helmholtz polynomials. By using their recurrence formula with the order fixed, we concurrently execute massive recursions to prepare the polynomial values. The second is folding a do loop of trapezoidal shape into box-shaped (Ito and Fukushima, 1997; Fukushima, 2011). It is theoretically expected to almost double the parallel execution speed. And the last is interleaving the one-dimensional recurrence formulas to compute Chebyshev polynomials and simple powers of a given argument. However, its contribution to the speed-up of the whole process to compute spherical harmonics is relatively small. Thanks to the simplicity of Open Multi-Processing (OpenMP) architecture (OpenMP ARB, 2009), we may implement this formulation in existing Fortran or C programs to evaluate spherical harmonic expansions after their slight rewriting and embedding a few OpenMP directives.

2 Method

2.1 Helmholtz Polynomials

One of the main computational tasks in spherical harmonic expansions is the simultaneous evaluation of associated Legendre functions (ALF) of various degrees and orders for the given argument. This is because it is a two-dimensional problem. Therefore, its computational labor increases quadratically with respect to the maximum degree and/or order considered. Our scheme of parallelization, which will be explained below, is irrelevant with (1) the difference in normalization of ALFs (Fantino and Casotto, 2009), (2) the modification of the functional forms of ALFs by factoring their irrational part out (Holmes and Featherstone, 2002), (3) the application of Clenshaw summation method (Tscherning and Poder, 1982), (4) the usage of global scaling before computing ALFs (Wenzel, 1998), or (5) the extension of the floating point exponents (Wittwer et al., 2008). Thus, we fix one formulation in the following description in order to make the explanation more understandable.

As the base functions of such a formulation, we choose quasi-normalized Helmholtz polynomials:

$$\tilde{H}_{nm}(X) \equiv \sqrt{\frac{(n-m)!}{(n+m)!}} \left(\frac{d^m P_n}{dX^m} \right), \tag{1}$$

where $P_n(X)$ is the Legendre polynomial of degree n. Hereafter we drop the dependence on the argument X such as $\tilde{H}_{nm} = \tilde{H}_{nm}(X)$ for simplicity. Once $\tilde{H}_{nm}$ is given, the fully normalized ALF of the first kind, $\overline{P}_n^m$, is simply obtained as

$$\overline{P}_n^m = \sqrt{(2-\delta_{0m})\,(2n+1)\,(1-X^2)^m}\,\tilde{H}_{nm}, \tag{2}$$

where δ_{nm} is Kronecker's delta. It is easy to see that $\tilde{H}_{nm}$ is a polynomial of the degree $n-m$ and reduces to 0 when $n < m$ since P_n is a polynomial of the degree n and $\tilde{H}_{nm}$ is

its m-th order derivative except a numerical constant. As a result, its certain high-order derivatives become zero. This analytic characteristic is one of the reasons why we prefer the Helmholtz polynomials to ALFs, which are irrational functions when $m \neq 0$.

Also the derivatives of the Helmholtz polynomials are reproduced by themselves as

$$\frac{d\tilde{H}_{nm}}{dX} = F_{nm}\tilde{H}_{n,m+1}, \tag{3}$$

where

$$F_{nm} \equiv \sqrt{(n+m+1)(n-m)}. \tag{4}$$

is a numerical constant. If F_{nm} is prepared beforehand as usual, only one multiplication is needed to compute the first-order derivatives once $\tilde{H}_{nm}$ are all known. Namely there is no need to evaluate the derivatives by different algorithms as in the case of ALFs (Bosch, 2000). This is another reason why we use the Helmholtz polynomials as base functions.

2.2 Vector Computation of Helmholtz Polynomials

In general, the ALFs or their variants such as the Helmholtz polynomials are effectively computed by recurrence relations (Olver et al., 2010). Typically used are consecutive three-term recurrence formulas with the degree or order fixed (Tscherning and Poder, 1982; Holmes and Featherstone, 2002; Casotto and Fantino, 2007; Fantino and Casotto, 2009). This is because the formulas mixing different degrees and orders are generally known to be unstable (Holmes and Featherstone, 2002). As the main algorithm for the simultaneous evaluation of Helmholtz polynomials, i.e. that except the preparation of numerical constants, we select a double do loop computing the fixed-order, increasing-degree, consecutive three-term recurrence formula:

$$\tilde{H}_{nm} = A_{nm}X\tilde{H}_{n-1,m} - B_{nm}\tilde{H}_{n-2,m}, \tag{5}$$

where the starting values are prepared as

$$\tilde{H}_{mm} = D_m, \;\; \tilde{H}_{m+1,m} = E_m X, \tag{6}$$

while A_{nm}, B_{nm}, D_m, and E_m are numerical constants defined as

$$A_{nm} \equiv \frac{2n-1}{\sqrt{(n+m)(n-m)}}, \tag{7}$$

$$B_{nm} \equiv \sqrt{\frac{(n+m-1)(n-m-1)}{(n+m)(n-m)}}, \tag{8}$$

$$D_m \equiv \sqrt{\frac{(2m-1)!!}{(2m)!!}}, \tag{9}$$

$$E_m \equiv \sqrt{\frac{(2m+1)(2m-1)!!}{(2m)!!}}. \tag{10}$$

The vectorial nature of the main algorithm assures that, once the starting values are prepared, the recurrence formulas can be conducted independently with each other. Thus, they are simply parallelized without significant modification. However, its naive parallelization turns out to be inefficient. In fact, the number of degrees to be increased depends on the given order. Meanwhile the computational amount of each recurrence formula is the same. These facts mean that the computational amount of the total recursion for a specific order is a linear function of the order. Namely the double do loop is of trapezoidal shape (Fukushima, 2011). This is the source of the inefficiency of the naive parallelization.

2.3 Folding Trapezoidal Do Loop

Let us abstract the main algorithm described in the previous subsection in the form of a single do loop as

$$\texttt{do m} = \texttt{m1,m2} \, \{ \, \texttt{task(m)} \, \}$$

where we rewrote the contents of the outer do loop as `task(m)` in short. As we noted earlier, the do loop is of trapezoidal shape, namely the computational amount of `task(m)` is not constant but a linear function of the loop index, `m`. A simple parallel execution of such a do loop is inefficient because the load balance among different computing threads is poor. However, as we experienced in the vectorization of the extrapolation method to integrate an ordinary differential equation numerically (Ito and Fukushima, 1997) and in the parallel computation of all-pairs N-body acceleration (Fukushima, 2011), we can equalize the computational amount inside the do loop by folding it into box-shaped as

$$\texttt{do m} = \texttt{m1}, (\texttt{m1} + \texttt{m2})/2 \, \{ \, \texttt{task(m)};$$

$$\texttt{if}(\texttt{m1} + \texttt{m2} - \texttt{m} \neq \texttt{m}) \, \{ \, \texttt{task}(\texttt{m1} + \texttt{m2} - \texttt{m}) \, \} \, \}$$

The conditional statement added assures the rewritten program to work correctly whether the length of the original do loop, $\mathtt{m2}-\mathtt{m1}+1$, is even or odd. Of course, if the length is known to be even *a priori*, we may skip the `if` statement as

$$\mathtt{do\ m} = \mathtt{m1}, (\mathtt{m1}+\mathtt{m2})/2\ \{\ \mathtt{task(m)};\ \mathtt{task}(\mathtt{m1}+\mathtt{m2}-\mathtt{m})\ \}$$

The parallel execution is simply conducted by embedding an OpenMP 3.0 parallel do directive (OpenMP ARB, 2009). It enables parallel execution of the subsequent do loop, namely the statements until the next comment line. The program is understood by modern parallel Fortran compilers, such as the Intel Visual Fortran Composer XE 2011, to produce parallelized executable codes. Meanwhile the program does not lose efficiency in the serial computation mode since an ordinary (serial) Fortran compiler interprets the embedded directive as a comment and simply ignores it.

2.4 Parallel Computation of Spherical Harmonic Synthesis

The other main computational task is the double summation appearing in the evaluation process of the potential and/or the acceleration vector since it is again a two-dimensional problem. From the viewpoint of computational amount, its parallelization is more important than that of the evaluation of Helmholtz polynomials. This is because the computational task of its innermost do loop is much more than that of ALFs and their variant.

If all the components appearing in the summation process are computed beforehand, it is trivial to parallelize the summation. However, the double do loop is of trapezoidal shape, too. Thus we conduct the do loop folding again. Note that we may incorporate the evaluation of the potential itself, too. This is because it is achieved with negligible amount of the additional computational labor.

3 Numerical Experiments

3.1 Environment of Experiments

In order to measure the performance of the method described in the previous section, we obtained the wall clock times to conduct serial and parallel computations needed in evaluating the acceleration vector at a single point. The points of evaluation is assumed to be on a nominal orbit of a low-altitude satellite resembling GRACE. We used EGM2008 (Pavlis et al., 2008) as a reference geopotential although we set the maximum degree and order the same for simplicity.

In order to avoid inappropriate underflows and/or overflows during the recursion computations, we premultiplied the Stokes coefficients by a scaling constant σ as $2^{768} \approx 1.56 \times 10^{231}$ and the Helmholtz polynomials and their derivatives by $1/\sigma$, respectively. We experimentally learn that this is enough in the IEEE754 double and quadruple precision computations[1]. In case of the single precision environment, however, we must reduce its magnitude such as $2^{60} \approx 1.15 \times 10^{18}$. This limits the maximum order/degree of the polynomials to be less than a few hundreds.

All the computation codes were written in Fortran 90, enhanced with OpenMP 3.0 directives if necessary, compiled by the Intel Visual Fortran Composer XE 2011 with the level 3 optimization, and executed under the Windows XP OS. The used PCs are those with Intel CPUs (the main memory) as Atom N270 (1 GB), Core2 Duo E8500 (8 GB), Core i7-620M (4 GB), Core2 Quad Q9500 (2 GB), and Core i7-930 (3 GB), respectively. Using the number of cores (C) and threads (T) on the CPU, we denote them by 1C2T, 2C2T, 2C4T, 4C4T, and 4C8T, respectively.

3.2 Effect of Folding

First of all, we examine the effect of folding. Figure 1 shows the speed-up factors of the parallel computation of Helmholtz polynomials. Here we define the speed-up factor as the ratio of wall clock times of the serial and parallel computations. The actual wall clock times are measured by means of a system call to access the machine clock in the unit of microsecond and averaged for 2^{10} to 2^{26} latitudes evenly distributed in the range $(0, \pi/2)$. Here the number of measured points are adjusted such that the total wall clock time exceeds a certain threshold value, say a few tens minutes, so as to be reliable with 3 significant digits. The measurements are conducted at a PC with the most parallelized chip among the tested CPUs, a quad-

[1] A somewhat different value 10^{-280} was chosen in Holmes and Featherstone (2002).

core 8-thread (4C8T) processor. The results obtained with and without folding in the double and quadruple precision environments are plotted as functions of L, the maximum order of the geopotential. Here we assume that L is equal to K, the maximum degree, for simplicity. Obviously, the folding is always effective when conducting the parallel computation since the folded computation runs equally or faster than that without folding.

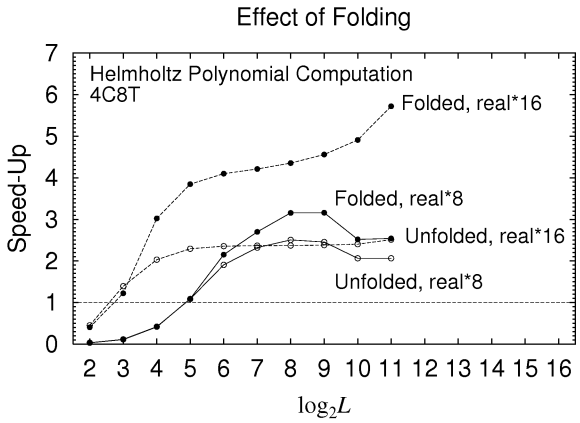

Figure 1: Effect of Folding. Shown are the wall clock time ratio of the parallel and serial computations of Helmholtz polynomials.

3.3 Acceleration Computation

Finally, we show the results of the whole computation of acceleration vector in the double precision environment. Based on the experience described in the previous subsections, we incorporated the folding both in the computation of Helmholtz polynomials and the double summation to evaluate the acceleration but not in the preparation of Chebyshev polynomials and powers of $\cos\phi$ and a_e/r in case of double precision computations.

Figure 2 illustrates the speed up factors of thus-described parallel computation conducted at various kind of consumer PCs. The tested PCs are of processors with different number of cores (C) and threads (T) as Intel Atom N270 (1C2T), Intel Core2 Duo E8500 (2C2T), Intel Core i7-620M (2C4T), Intel Core2 Quad Q9500 (4C4T), and Intel Core i7-930 (4C8T).

It is interesting that not only the number of cores but also that of threads affect the speed-up factor. For example, see the similarity of the speed-up factors of 2C4T and 4C4T and of 1C2T and 2C2T when the maximum degree/order is as high as around 2000. As a result, even a single core processor PC realizes a significant speed-up by the folded parallelization if it is multi-threaded. See the curve 1C2T. This indicates that Intel Hyper-Threading Technology (Intel, 2003), which tries to employ all available arithmetic units, is significantly effective to the parallel computation of spherical harmonics. The observed situation is entirely different from that of all-pair N-body accelerations mainly consisting of the inverse square root computation. In that case, the acceleration factors do depend on the number of cores (Fukushima, 2011).

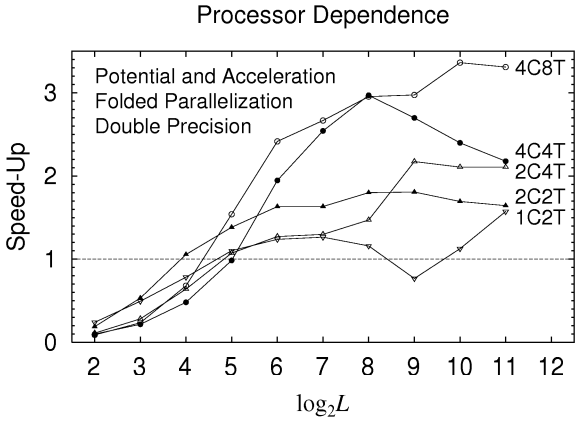

Figure 2: Processor Dependence of Folded Parallel Computation of Potential and Acceleration: Double Precision.

4 Conclusion

We present a method to parallelize the general computation of spherical harmonics at a single point. The main technique is the folding of time-consuming double do loops of trapezoidal shape. Its implementation is simple: adding a dual of the inner loop as well as its pre- and post-processes and embedding an Open MP parallel do directive with suitable 'private' clause just before the outer loop begins. This device works in the double summation

required in computing arbitrary partial sum of spherical harmonic expansions as well as their derivatives and/or integrals. Also it is applicable to any kind of fixed-order or fixed-degree recurrence formulas to evaluate associate Legendre functions (ALF) and their variants.

As an example of the new method, by using a PC with a quad-core 8-thread CPU, we confirmed that the speed-up factor of the orbit integration of a satellite under the non spherical gravitational field becomes 3.3 in the double precision environment when $L \sim 2000$. These features will significantly accelerate almost all types of computation using large scale spherical harmonics.

References

Bosch W (2000) On the computation of Derivatives of Legendre Functions. Phys Chem Earth 25: 655–659

Casotto S, Fantino E (2007) Evaluation of methods for spherical harmonic synthesis of the gravitational potential and its gradients. Adv Space Res 40: 69–75

Domeika M (2008) Software development for embedded multi-core systems: a practical guide using embedded Intel architecture. Newnes/Elsevier Inc, Burlington, MA

Fantino E, Casotto S (2009) Methods of harmonic synthesis for global geopotential models and their first-, second- and third-order gradients. J Geod 83: 595–619

Fukushima T (2011) Efficient parallel computation of all-pairs N-body acceleration by folding trapezoidal do loop. Astron J submitted

Greengard L, Rokhlin V (1996) A New Version of the Fast Multipole Method for the Laplace Equation in Three Dimensions. Res Rep Yale U Dept Comp Sci: RR-1115

Holmes SA, Featherstone WE (2002) A unified approach to the Clenshaw summation and the recursive computation of very high degree and order normalized associated Legendre functions. J Geod 76: 279–299

Intel (2003) Intel Hyper-Threading Technology Technical User's Guide. Intel Corp.

Ito T, Fukushima T (1997) Parallelized extrapolation method and its application to the orbital dynamics. Astron J 114: 1260–1267

Kaula WM (2000) Theory of satellite geodesy: applications of satellites to geodesy. Dover Publ Inc, Mineora

Mason JC, Handscomb DC (2003) Chebyshev Polynomials. Chapman and Hall/CRC, Boca Raton

Nyland L, Harris M, Prins J (2007) Fast N-body simulation with CUDA. in GPU Gems 3, Nguyen H (ed): Chapt. 31 Addison-Wesley, Upper Saddle River NJ

Olver FWJ, Lozier DW, Boisvert RF, Clark, CW (eds) (2010) NIST Handbook of Mathematical Functions. Cambridge Univ Press, Cambridge. http://dlmf.nist.gov/

OpenMP Architecture Review Board (2009) Summary of OpenMP 3.0 Fortran Syntax. http://openmp.org/

Pavlis NK, Holmes SA, Kenyon SC, Factor JK (2008) An Earth gravitational model to degree 2160: EGM2008. presented at the 2008 General Assembly of the European Geosciences Union, Vienna, Austria, April 13-18, 2008, see http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm2008/

Tscherning CC, Poder K (1982) Some geodetic applications of Clenshaw summation. Boll Geofis Sci Aff 4: 351–364

Wittwer T, Klees R, Seitz K, Heck B (2008) Ultra-high degree spherical harmonic analysis and synthesis using extended-range arithmetic. J Geod 82: 223–229

Wenzel G (1998) Ultra-high degree geopotential models GPM98A, B, and C to degree 1800. Paper presented to the joint meeting of the International Gravity Commission and International Geoid Commission, 7–12 September, Trieste

Xiao HD, Lu Y (2007) Parallel computation for spherical harmonic synthesis and analysis. Comp Geosci 33: 311–317