Proc. Estonian Acad. Sci. Phys. Math., 2004, **53**, 2, 107–115
https://doi.org/10.3176/phys.math.2004.2.06

Stability analysis of the fast Legendre transform algorithm based on the fast multipole method

Reiji Suda

Department of Computer Science, the University of Tokyo/CREST of JST, Hongo, Bunkyo-ku, Tokyo 113-0033, Japan; reiji@is.s.u-tokyo.ac.jp

Received 20 November 2003, in revised form 7 January 2004

Abstract. The fast Legendre transform algorithm based on the fast multipole method proposed by Suda and Takami (*Math. Comput.*, 2002, **71**, 703–715) is discussed. The alpha-beta product is introduced as an indicator of instability, and the effects of the interpolations, splits, and shifts on the alpha-beta products are evaluated. The interpolations are proved to be stable. The instability of the splits and the shifts are evaluated numerically, and the stability is shown to be sufficient for practical use. The fast transform scheme must be applicable to other functions that are stable in the recurrence formula and the Clenshaw summation formula.

Key words: fast Legendre transform, fast multipole method, polynomial interpolation, matrix computation, matrix product, stability analysis, alpha-beta product.

1. INTRODUCTION

We have proposed a fast Legendre transform algorithm based on the fast multipole method (FMM) in [1]. Our algorithm has several similarities with the Driscoll–Healy algorithm [2,3]. Both have the divide-and-conquer strategy and use fast polynomial interpolation. However, the stability characteristics of the two algorithms show a vivid contrast. Our algorithm runs quite stably, while the Driscoll–Healy algorithm is severely unstable for large order m. The source of the instability of the Driscoll–Healy algorithm is quite clear: the use of the FFT for fast polynomial interpolations. The FFT requires equispaced points, and polynomial interpolations with equispaced points tend to be unstable (known as Runge's phenomena). Our algorithm uses the FMM for polynomial interpolations, which accepts any distribution of points. However, that is not enough to guarantee the stability of the algorithm. This paper analyses the stability of our algorithm theoretically and experimentally.

First, our fast transform algorithm is explained. There the fast polynomial interpolation, the split of the Legendre functions, and the shift of the split points affect the stability. The alpha-beta product is introduced as the indicator of instability. A small alpha-beta product is a necessary condition of the stability of a linear computation. Then the stability of our algorithm is analysed in terms of the alpha-beta product. We prove that there is a set of the sampling points that limits the increase in the alpha-beta products, and thus numerically stable interpolations are always possible. The splits and the shifts sometimes increase the alpha-beta products, and the increasing factor depends on the scaling of the functions. Those increasing factors are evaluated numerically, and it is shown that the stability of our algorithm is sufficient for practice.

2. THE FAST TRANSFORM ALGORITHM

This section briefly reviews our fast transform algorithm. A fuller description is in the original paper [1].

Let us consider the inverse associated Legendre function transform

$$g^m(\mu_k) = \sum_{n=m}^{N} g_n^m P_n^m(\mu_k). \tag{1}$$

The forward transform, given as

$$g_n^m = \sum_k \omega_k g^m(\mu_k) P_n^m(\mu_k),$$

where μ_k are Gaussian nodes and ω_k are the corresponding weights, is a transposition of the inverse transform. Thus it is enough to consider the inverse transform (1). We refer to $\{\mu_k\}$ as the *evaluation points*.

2.1. Fast polynomial interpolation

The associated Legendre function can be factorized as

$$P_n^m(x) = p_{n-m}^m(x) P_m^m(x),$$

where $p_{n-m}^m(x)$ is a polynomial of degree $n-m$. Thus $g^m(\mu_k)/P_m^m(\mu_k)$ is a polynomial of degree $N-m$, and the transform $g^m(\mu)$ at any point μ can be computed from the values at a set of $N-m+1$ points $\{\mu_i\}$ by the polynomial interpolation (the Lagrange formula)

$$g^m(\mu) = \sum_i \frac{P_m^m(\mu)\omega_i(\mu)}{P_m^m(\mu_i)\omega_i(\mu_i)} g^m(\mu_i), \tag{2}$$

where $\omega_i(\mu)$ is a polynomial of degree $N-m$:

$$\omega_i(\mu) = \prod_{j\neq i}(\mu - \mu_j).$$

Based on the above observation, our algorithm chooses a set of sampling points $\{\mu_i\}$ and evaluates the transform (1) on the other evaluation points (called *interpolation points*) by the interpolation (2). The interpolation can be approximately computed in time $O(N)$ by the FMM [4], where N is the number of the evaluation points.

2.2. Divide-and-conquer

In order to apply the fast polynomial interpolation by the FMM repeatedly, our algorithm divides the summation of (1) as

$$\sum_{n=m}^{N} g_n^m P_n^m(\mu_k) = \sum_{n=m}^{\nu-1} g_n^m P_n^m(\mu_k) + \sum_{n=\nu}^{N} g_n^m P_n^m(\mu_k), \tag{3}$$

where $\nu \approx (m+N)/2$.

The first term of the right-hand side is a Legendre transform of a half size. Thus it accepts polynomial interpolation again.

2.3. Split Legendre functions

The second term of the right-hand side of (3) does not accept polynomial interpolation as it is. To solve that difficulty, our algorithm splits the associated Legendre functions as

$$P_n^m(x) = P_{n,\nu}^{m,0}(x) + P_{n,\nu}^{m,1}(x),$$

where each *split Legendre function* $P_{n,\nu}^{m,l}(x)$ ($l = 0,1$) is the product of a polynomial and an associated Legendre function

$$P_{n,\nu}^{m,l}(x) = q_{n,\nu}^{m,l}(x) P_{\nu+l}^m(x).$$

The polynomial $q_{n,\nu}^{m,l}(x)$ vanishes (i.e. $\equiv 0$) for $n-\nu+l-1=0$, otherwise $\deg(q_{n,\nu}^{m,l}(x)) = |n-\nu+l-1|-1$. That split can be easily derived from the recurrence formula

$$(n-m)P_n^m(x) = (2n-1)xP_{n-1}^m(x) - (n+m-1)P_{n-2}^m(x),$$

which already gives the split for $\nu = n - 2$. The parameter ν is called the *split point*. The split point can be different from the lower bound of the summation, but that choice is the best for numerical stability. This is because a split with $n < \nu$ corresponds to the backward recurrence, which is unstable for the associated Legendre functions.

The second term of the right-hand side of (3) is split as

$$\sum_{n=\nu}^{N} g_n^m P_n^m(\mu_k) = \sum_{n=\nu}^{N} g_n^m P_{n,\nu}^{m,0}(\mu_k) + \sum_{n=\nu}^{N} g_n^m P_{n\nu}^{m,1}(\mu_k).$$

Then each term of the split sum can be interpolated as a polynomial.

2.4. Shift of the split point

Our algorithm divides the partial sums recursively. After dividing a split sum

$$\sum_{n=\nu}^{N} g_n^m P_{n,\nu}^{m,l}(\mu_k) = \sum_{n=\nu}^{\nu'-1} g_n^m P_{n,\nu}^{m,l}(\mu_k) + \sum_{n=\nu'}^{N} g_n^m P_{n,\nu}^{m,l}(\mu_k),$$

the second term of the right-hand side does not accept polynomial interpolation again. The split point needs to be shifted to the lower bound of the sum ν' as

$$\begin{aligned}\sum_{n=\nu'}^{N} g_n^m P_{n,\nu}^{m,l}(\mu_k) &= T_{\nu,\nu'}^{m,0l} \sum_{n=\nu'}^{N} g_n^m P_{n,\nu'}^{m,0}(\mu_k) + T_{\nu,\nu'}^{m,1l} \sum_{n=\nu'}^{N} g_n^m P_{n,\nu'}^{m,1}(\mu_k) \\ T_{\nu,\nu'}^{m,\lambda l}(x) &= P_{\nu+\lambda,\nu'}^{m,l}(x)/P_{\nu+\lambda}^{m}(x).\end{aligned}$$

It is easy to show that the shift operation is equivalent to an application of the Clenshaw summation formula (with zero coefficients).

2.5. Computational complexity

Using splits and shifts, the transform (1) can be divided and interpolated arbitrarily many times. The recursion of the divide-and-conquer scheme stops when the evaluation–interpolation scheme needs more computational costs than the direct computation.

Every operation (interpolation via FMM, summation, and shift) in each level can be computed in linear time, and the recursion stops in $O(\log N)$ levels. Thus, assuming that the number of evaluation points $\{\mu_k\}$ is $O(N)$, the total computational complexity for the transform (1) is $O(N \log N)$. The spherical harmonic transform consists of $O(N)$ Legendre transforms ($m = 0$ to N), thus it can be done in time $O(N^2 \log N)$. Here, the precision required for the FMM is assumed to be constant.

3. STABILITY ANALYSIS

The stability of numerical algorithms has been investigated in various ways. However, our algorithm is too complex to analyse with the conventional schemes of numerical error analysis. I propose a simpler analysis based on the *alpha-beta products*, which make a necessary condition of numerical stability.

3.1. The alpha-beta product

The Legendre transform is a linear transform, and any part of that must also be a linear transform. Such an algorithm can be expressed as a series of linear transforms

$$y = F_0 F_1 \cdots F_Q x,$$

where each F_q (for $q = 0, \ldots, Q$) is a matrix that represents the numerical computation of the qth step of the algorithm. Let us define

$$G^{(q)} = F_0 F_2 \cdots F_{q-1}, \quad H^{(q)} = F_q F_{q+1} \cdots F_Q,$$

so that we have

$$y = G^{(q)} H^{(q)} x \qquad \text{for } q = 1, \cdots, Q.$$

Let $x^{(q)} = H^{(q)} x$ be the precise value of the intermediate vector. Assume that the computed value $\tilde{x}^{(q)}$ is given by rounding $x^{(q)}$. Then we have

$$|\tilde{x}_i^{(q)} - x_i^{(q)}| \le \epsilon |x_i^{(q)}| = \epsilon |h_i^{(q)} x| \le \epsilon \beta_i^{(q)} \|x\|,$$

where ϵ is the machine epsilon and $\beta_i^{(q)} = \|h_i^{(q)}\|$.

The final result vector y is affected by that rounding as

$$\begin{aligned} \|\tilde{y} - y\| = \left\| \sum_i g_i^{(q)} (\tilde{x}_i^{(q)} - x_i^{(q)}) \right\| &\le \sum_i \alpha_i^{(q)} |\tilde{x}_i^{(q)} - x_i^{(q)}| \\ &\le \epsilon \|x\| \sum_i \alpha_i^{(q)} \beta_i^{(q)}, \end{aligned}$$

where $\alpha_i^{(q)} = \|g_i^{(q)}\|$. The above arguments show that a large value of $\sum_i \alpha_i^{(q)} \beta_i^{(q)}$ allows a large error on the result vector y. Thus the value of $\sum_i \alpha_i^{(q)} \beta_i^{(q)}$ works as an indicator of instability. Let us call that the *alpha-beta product*.

3.2. The alpha-beta products after interpolation

Let C be an interpolation in our algorithm, and $y = ACBx$ be the whole transform. Define the alpha-beta product *before* the interpolation $\bar{\alpha}_i \bar{\beta}_i$ as

$$\bar{\alpha}_i = \|A_i\|, \quad \bar{\beta}_i = \|c_i B\|,$$

where A_i is the ith column of A and c_i is the ith row of C. The alpha-beta product *after* the interpolation $\alpha_i\beta_i$ should be defined as

$$\alpha_i = \|AC_i\|, \quad \beta_i = \|b_i\|,$$

where C_i is the ith column of C and b_i is the ith row of B. Then we can prove the following theorem, which guarantees that numerically stable interpolation is always possible.

Theorem. *For any interpolation C in the form of* (2), *one can choose the sampling points so that*

$$\alpha_i\beta_i \le (1+J)\bar{\alpha}_i\bar{\beta}_i,$$

where J is the number of the interpolation points.

Proof. Let I be the set of indices of the sampling points and J be the set of indices of the interpolation points. First note that $\beta_i = \bar{\beta}_i$ for $i \in I$ and $c_{ik} = \delta_{ik}$ (Kronecker delta) for $i, k \in I$. Also we have

$$\alpha_i \quad \le \quad \bar{\alpha}_i + \sum_{j\in J} \bar{\alpha}_j |c_{ji}| \, . \tag{4}$$

The computation ACB can be rewritten as

$$\begin{aligned} \begin{pmatrix} A_I & A_J \end{pmatrix} \begin{pmatrix} B_I \\ B_J \end{pmatrix} &= \begin{pmatrix} A_I & A_J \end{pmatrix} \begin{pmatrix} I \\ \Theta \end{pmatrix} B_I \\ &= \begin{pmatrix} A_I S_I^{-1} & A_J S_J^{-1} \end{pmatrix} \begin{pmatrix} I \\ S_J \Theta S_I^{-1} \end{pmatrix} S_I B_I, \end{aligned}$$

where $S_I = \mathrm{diag}(\bar{\alpha}_i)$ and $S_J = \mathrm{diag}(\bar{\alpha}_j)$ are scaling matrices. Here we have $S_J\Theta S_I^{-1} = S_J B_J (S_I B_I)^{-1}$ and from the Cramer rule

$$(S_J\Theta S_I^{-1})_{ji} = \frac{\det(S_J B_J)^{(ji)}}{\det(S_I B_I)},$$

where $(S_J B_J)^{(ji)}$ is defined from $S_J B_J$ by replacing its jth row by the ith row of $S_I B_I$. Now choose the sampling points I so that $|\det(S_I B_I)|$ is maximized. Then we have

$$|(S_J\Theta S_I^{-1})_{ji}| = \bar{\alpha}_j |c_{ji}| \bar{\alpha}_i^{-1} \le 1.$$

From (4) we obtain the desired result

$$\alpha_i\beta_i \le (1+|J|)\bar{\alpha}_i\bar{\beta}_i.$$

3.3. The alpha-beta products after split and shift

Our algorithm consists of interpolation, division, split, and shift. The effects of the interpolation on the alpha-beta products were discussed above. The division is easy: α does not change after division, and β is computed for a smaller range of n's. Thus the alpha-beta products become smaller after division. The remaining part of this section discusses the effects of the splits and shifts on the alpha-beta products.

3.3.1. *The alpha-beta products after split*

After a split the α values remain the same. The β value before the split is defined as

$$\beta_i = \sqrt{\sum_n |P_n^m(\mu_i)|^2}$$

in the case of the 2-norm, and after the split as

$$\beta_i^l = \sqrt{\sum_n |P_{n,\nu}^{m,l}(\mu_i)|^2}.$$

From the equation $P_n^m(\mu) = P_{n,\nu}^{m,0}(\mu) + P_{n,\nu}^{m,1}(\mu)$, we have

$$\beta_i \le \beta_i^0 + \beta_i^1.$$

If the right-hand side is much larger than the left-hand side, then the alpha-beta product increases after the split, and thus the stability is lost. So let us define the *instability factor for the split* γ_i as

$$\gamma_i = \frac{\beta_i^0 + \beta_i^1}{\beta_i}.$$

3.3.2. *The alpha-beta products after shift*

Next consider the $\alpha\beta$ values after a shift. Let the ith alpha-beta product before the shift be $\bar{\alpha}_i^l\bar{\beta}_i^l$ and that after the shift be $\alpha_i^l\beta_i^l$. We have

$$\alpha_i^l \le \bar{\alpha}_i^0|T_{n,\nu}^{m,l0}(\mu_i)| + \bar{\alpha}_i^1|T_{n,\nu}^{m,l1}(\mu_i)|,$$

and thus

$$\sum_{l=0}^{1} \alpha_i^l\beta_i^l \le \sum_{l=0}^{1} \gamma_i^l\bar{\alpha}_i^l\bar{\beta}_i^l,$$

where γ_i^l is the *instability factor for the shift* defined as

$$\gamma_i^l = \frac{|T_{n,\nu}^{m,0l}(\mu_i)|\beta_i^0 + |T_{n,\nu}^{m,1l}(\mu_i)|\beta_i^1}{\bar{\beta}_i^l}.$$

Table 1. The maximum values of the instability factors

	N					
	127	255	511	1023	2047	4095
Normal max γ_i	108.49	219.26	440.86	884.10	1770.59	3543.59
Normal max γ_i^l	72.55	147.19	297.88	601.28	1211.45	2438.90
Damped max γ_i	5.88	7.00	8.14	9.29	10.44	11.61
Damped max γ_i^l	2.29	2.49	2.69	2.92	3.14	3.38

3.3.3. *Magnitudes of the instability factors*

Currently we have no theory to bound the instability factors. Table 1 shows the numerically computed maximum values of the instability factors. For the upper half of the table, the Legendre functions are normalized as

$$\int_{-1}^{1} |P_n^m(x)| dx = 2,$$

and the 2-norm is used. Both instability factors are almost proportional to N.

For the lower half of the table, the Legendre functions are *damped* as $\tilde{P}_n^m(x) = n^{-1}P_n^m(x)$, where $P_n^m(x)$ is the normalized one. The instability factors are much smaller than the normalized case and almost proportional to $\log N$. Thus the damping greatly reduces the numerical instability, but it depends on the application problem whether it is acceptable or not. The damping factor should be chosen carefully for each application problem, to balance the approximation precision and the numerical stability.

4. SUMMARY

This paper analysed the stability of the fast Legendre transform algorithm. The alpha-beta product was introduced as an indicator of instability. The effects of interpolation, the split of the associated Legendre functions, and the shift of the split points on the alpha-beta products were investigated theoretically and experimentally. The interpolation was proved to be always stable, and the stability of the split/shift was examined through numerical experiments.

Our scheme of the fast Legendre transform is applicable to other transforms by functions of similar recurrence formulae. We can show that the instability factors for the split/shift also work as instability measures for the recurrence formula and the Clenshaw summation formula. Therefore it is implied that our scheme will provide fast function transforms if the recurrence formula (actually, either forward or backward) of the functions is numerically stable in terms of the above-defined instability factors.

ACKNOWLEDGEMENTS

This research was partly supported by Research for the Future Program (JSPS), CREST (JST), the 21st Century COE program, and Grants-in-Aid for Scientific Research (MEXT).

REFERENCES

1. Suda, R. and Takami, M. A fast spherical harmonics transform algorithm. *Math. Comput.*, 2002, **71**, 703–715.
2. Healy, D. M. Jr., Rockmore, D. and Moore, S. S. B. FFTs for 2-Sphere – Improvements and Variations. Tech. Rep. PCS-TR96-292, Dartmouth Univ., 1996.
3. Potts, D., Steidl, G. and Tasche, M. Fast and stable algorithms for discrete spherical Fourier transforms. *Linear Algebra Appl.*, 1998, **275–276**, 433–450.
4. Greengard, L. and Rokhlin, V. A fast algorithm for particle simulations. *J. Comput. Phys.*, 1987, **73**, 325–348.

FMM-meetodil põhineva kiire Legendre'i teisenduse algoritmi stabiilsuse analüüs

Reiji Suda

Artiklis vaadeldakse Suda ja Takami [1] poolt välja töötatud FMM-meetodil põhinevat kiiret Legendre'i teisenduse algoritmi. Mittestabiilsuse indikaatoriks defineeritakse alfa-beeta korrutis ning hinnatakse interpolatsioonide, tükelduste ja nihete mõju sellele. Näidatakse, et interpolatsioonid on stabiilsed. Tükelduste ja nihete ebastabiilsust hinnatakse numbriliselt ning näidatakse, et üldine stabiilsus on praktiliste rakenduste jaoks piisav. Kiire teisenduse skeem peab olema rakendatav ka teistele funktsioonidele, mis on stabiilsed rekursioonivalemi ning Clenshaw' summeerimisvalemi suhtes.