Legendre polynomials Triple Product Integral and lower-degree approximation of polynomials using Chebyshev polynomials

Mohit Gupta Srinivasa G. Narasimhan

CMU-RI-TR-07-22

May 2007

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

In this report, we present two mathematical results which can be useful in a variety of settings. First, we present an analysis of Legendre polynomials triple product integral. Such integrals arise whenever two functions are multiplied, with both the operands and the result represented in the Legendre polynomial basis. We derive a recurrence relation to calculate these integrals analytically. We also establish the sparsity of the triple product integral tensor, and derive the **Legendre polynomial triple product integral theorem**, giving the exact closed form expression for the sparsity structure.

Secondly, we derive a truncation scheme to approximate a polynomial with a lower degree polynomial, while keeping the approximation error low under the L_∞ norm. We use the Chebyshev polynomials to derive our truncation scheme. We present empirical results which suggest that the approximation error is quite low, even for fairly low degree approximations.

Contents

1 Legendre Polynomial Triple Product Integral

Given two functions $f(x)$ and $g(x)$, suppose we want to find the product $h(x) = f(x) * g(x)$, where all the functions are represented in Legendre Polynomials basis:

$$f(x) = \sum_i a_i L_i(x), \quad g(x) = \sum_i b_i L_i(x), \quad h(x) = \sum_i c_i L_i(x),$$

where L_i is the i^{th} Legendre Polynomial.

$$\Rightarrow \sum_k c_k L_k(x) = (\sum_i a_i L_i(x)) * (\sum_j b_j L_j(x))$$

$$\Rightarrow \sum_k c_k L_k(x) = \sum_i \sum_j a_i b_j L_i(x) L_j(x)$$

$c_k' s$ are the unknowns here. To simplify, we exploit the orthogonality of Legendre Polynomials (refer [1] for a list of properties of the Legendre polynomials). Multiplying by $L_k(x)$ and integrating from -1 to 1:

$$c_k = \frac{2k+1}{2} \sum_i \sum_j a_i b_j TI(i,j,k)$$

where $TI(i,j,k) = \int_{-1}^{1} L_i(x) L_j(x) L_k(x) dx$ is defined as the Legendre Polynomial Triple Product Integral.

1.1 Motivation and Technical Contribution

Given K terms each in the expansion of $f(x)$ and $g(x)$, $h(x)$ will have $2K$ terms. Thus, we have a total of $2K \times K \times K = 2K^3$ computations to calculate all the c_k. **However, looking at the slices** of the TI tensor (3D), one can observe that a large fraction of the entries are zero (Figure 1). Also, the sparsity of the TI tensor has a particular structure. In this report, we establish the sparsity formally, and find a closed form expression for the sparsity structure.

Once we establish sparsity of the TI tensor , we can reduce the number of computations, specially if we know the **exact** distribution of the non-zero elements apriori. **The main technical contributions** here are:

- Establish **sparsity** of the TI tensor – prove that only a small fraction of elements are non-zero.
- Find out the **distribution of non-zero** elements, i.e. find the **closed analytical form** of the function $N(i,j,k)$, which returns 1 if the element is non-zero, and 0 otherwise.

1.2 Previous Work

Ng et al [3] provide an analysis of triple product integrals for basis like Haar Wavelets, Spherical Harmonics and Fourier Series. However, to the best of our knowledge, there is no such previous work on Legendre Polynomials. Legendre Polynomials form a system of basis polynomials with a **wide support**, as compared to Haar Wavelets which provide only **compact support**. Thus, in practise, a function can be represented using a relatively small number of Legendre Polynomials. We believe that a thorough analysis of triple product integrals for Legendre Polynomials would be useful, given the generality of the problem (multiplication of two functions in Legendre Coefficients domain).

1.3 Calculation of the Triple Integral

As earlier, $TI(i,j,k) = \int_{-1}^{1} L_i(z) L_j(z) L_k(z) dz$. We will now try to calculate the triple integrals by formulating a recurrence relation for the same:

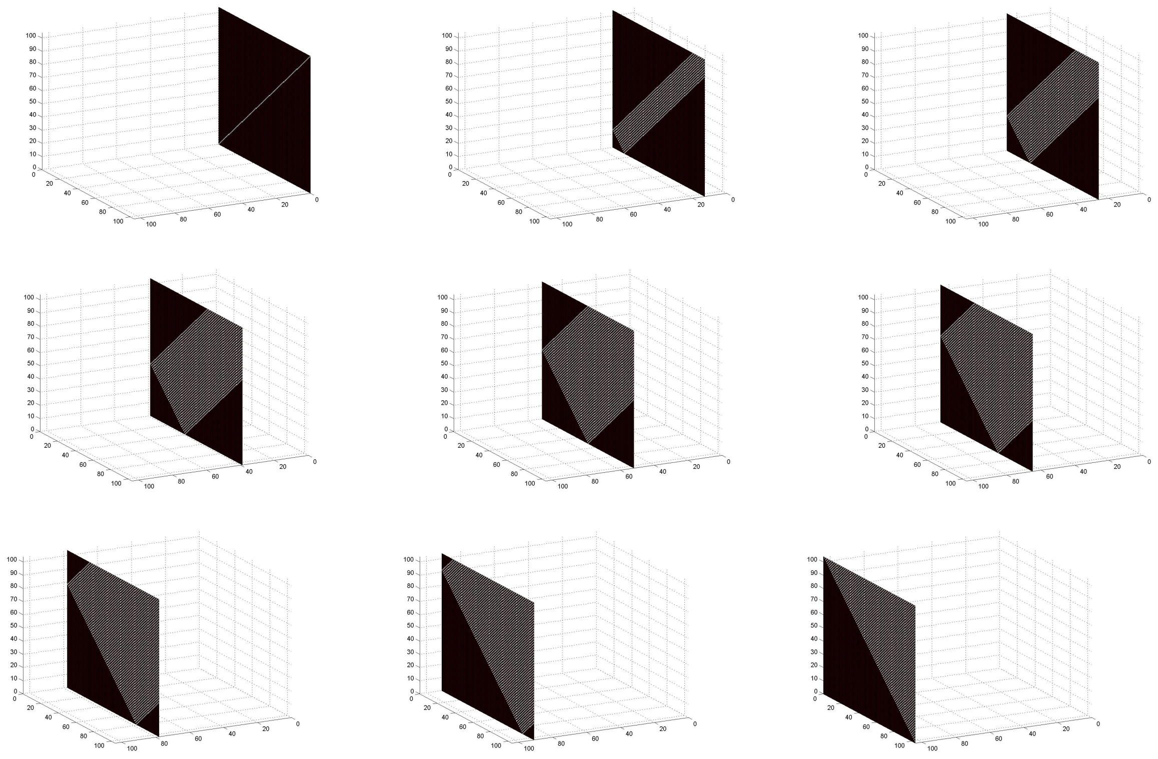

Figure 1: Various Slices of the TI tensor. Area in black is zero. We can observe that a significant fraction of the TI tensor is zero. Also, the sparsity of the TI tensor seems to have a particular structure. In this project, we want to establish the sparsity formally, and find a closed form expression for the sparsity structure.

$$\begin{aligned}
TI(i,j,k) &= \int_{-1}^{1} \underbrace{L_i(z)L_j(z)}_{Term1}\underbrace{L_k(z)}_{Term2}\,dz \qquad \ldots\ldots \qquad \text{(Integration By Parts)} \\
&= \left[L_iL_j\int L_k\right]_{-1}^{1} - \int_{-1}^{1}(L_iL_j)'\int L_k \\
&= \frac{1}{2k+1}\left[L_iL_j(L_{k+1}-L_{k-1})\right]_{-1}^{1} - \frac{1}{2k+1}\int_{-1}^{1}(L_iL_j)'(L_{k+1}-L_{k-1}) \qquad \ldots\ldots \qquad \left(L_k = \frac{L'_{k+1}-L'_{k-1}}{2k+1}\right)\ [1] \\
&= 0 - \frac{1}{2k+1}\int_{-1}^{1}(L'_iL_j + L_iL'_j)(L_{k+1}-L_{k-1}) \qquad \ldots\ldots \qquad \big(L_k(1)=1, L_k(-1)=(-1)^k\big)\ [1] \\
&= -\frac{1}{2k+1}\int_{-1}^{1}\left(L_j\sum_{r=1}^{\lceil\frac{i}{2}\rceil}(2i+3-4r)*L_{i+1-2r} + L_i\sum_{r=1}^{\lceil\frac{j}{2}\rceil}(2j+3-4r)*L_{j+1-2r}\right)*(L_{k+1}-L_{k-1}) \\
&\qquad \ldots\ldots \qquad \left(L'_k = \sum_{r=1}^{\lceil\frac{k}{2}\rceil}(2k+3-4r)*L_{k+1-2r}\right)
\end{aligned}$$

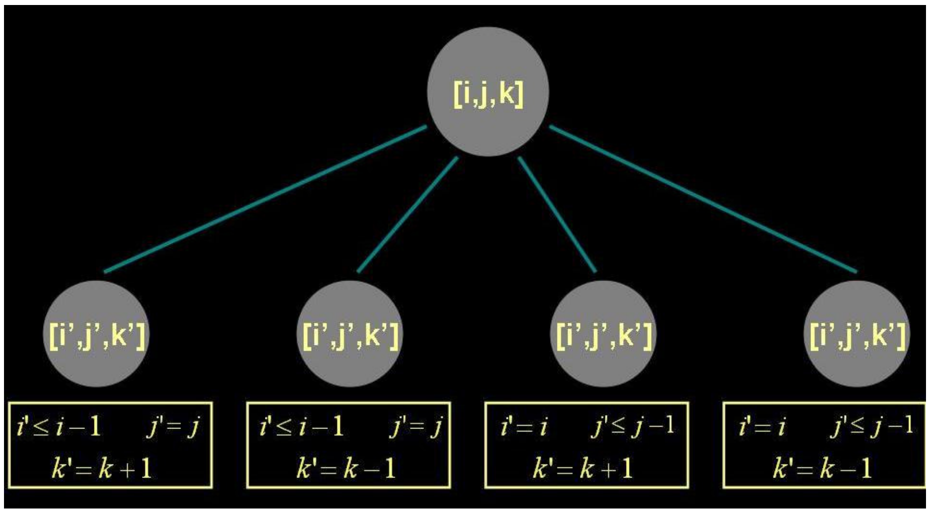
[i,j,k]
[i',j',k']
[i',j',k']
[i',j',k']
[i',j',k']
$i' \le i-1 \quad j'=j \quad k'=k+1$
$i' \le i-1 \quad j'=j \quad k'=k-1$
$i'=i \quad j' \le j-1 \quad k'=k+1$
$i'=i \quad j' \le j-1 \quad k'=k-1$

Figure 2: First Level of the recurrence tree

$$\begin{aligned} = \quad & -\frac{1}{2k+1}\sum_{r=1}^{\lceil \frac{i}{2} \rceil}(2i+3-4r)*[TI(i+1-2r,j,k+1)-TI(i+1-2r,j,k-1)] \\ & -\frac{1}{2k+1}\sum_{r=1}^{\lceil \frac{j}{2} \rceil}(2j+3-4r)*[TI(i,j+1-2r,k+1)-TI(i,j+1-2r,k-1)] \end{aligned}$$

Thus, we have expressed TI's in terms of a summation of TI's of lower order. We can calculate the $3D$ tensor of TI's using Dynamic Programming in time $O(K^4)$. We can also exploit the symmetry $TI(i,j,k) = TI(i,k,j) = TI(j,k,i) = \ldots$ to make the computations fast and numerically stable.

1.4 Sparsity of the Triple Integral

We use the recurrence relation derived above to prove the sparsity of the TI tensor, as given by the following **Legendre Polynomials Triple Product Integral** theorem:

Theorem 1. $TI(i,j,k) = 0$ if either of the following two conditions hold:

- The triplet (i,j,k) doesn't satisfy **triangle inequality**, i.e. either $i+j<k$, or $i+k<j$ or $j+k<i$
- $(i+j+k) \ mod \ 2 \neq 0$.

Proof. We will prove the two cases separately:

- **The triplet (i,j,k) doesn't satisfy triangle inequality:** Let the triplet (i,j,k) doesn't satisfy triangle inequality. Without loss of generality, lets assume that $i+j<k$. Consider the recurrence tree for $TI(i,j,k)$. The first level is shown in Figure 2. We will first prove that if the root triplet (i,j,k) doesn't satisfy triangle inequality, then neither of the triplets in the corresponding sub-tree does.

  Let a child of $TI(i,j,k)$ be $TI(i',j',k')$. We consider two cases:

  – $k' = k+1$: Since either $i' < i, j' = j$ or $i' = i, j' < j$, we have $i' + j' < k'$
  – $k' = k-1$: Since either $i' <= i-1, j' = j$ or $i' = i, j' <= j-1$, we still have $i' + j' < k$

Hence, given a root triplet (i, j, k) not satisfying triangle inequality, neither of the triplets in the corresponding sub-tree does.

Now we look at the leaves of such a sub-tree, or the **base cases**. We stop recursing when one of the indices (i,j or k) becomes 0. This is when we have reached a leaf of the tree. Let the triplet at a leaf be i_b, j_b, k_b. Without loss of generality, let $i_b = 0$. Now, $i_b + j_b < k_b$ (by above argument). Since $i_b = 0$, we have $j_b \neq k_b$. Thus, the value at the leaf is given by:

$$TI(i_b, j_b, k_b) = \int_{-1}^{1} L_0(x) L_{j_b}(x) L_{k_b}(x) dx$$
$$\Rightarrow TI(i_b, j_b, k_b) = \int_{-1}^{1} L_{j_b}(x) L_{k_b}(x) dx = 0$$

So, if $i + j < k$ for the root of the recurrence tree, $TI = 0$ for all the leaves in the sub-tree. Since the value at the root is a linear combination of all the leaves, the value at the root is zero too.

- $(i + j + k)\ mod\ 2 \neq 0$**:** Consider the recurrence tree for $TI(i, j, k)$ (Figure 2). The sum of indices of all the children of $TI(i, j, k)$ is $(i+j+k)+(2-2r)$ or $(i+j+k)-2r$, for a given positive integer r. The same will hold for all the recursion sub-trees as well. This implies that like the *Triangle Inequality-ness*, the *even-ness* of the sum of indices $(i + j + k)$ is preserved into the sub-tree, starting from the root. The sum will be even for all the nodes in the tree if it is even for the root and will be odd if it is odd for the root.

  Now we again look at the leaves of the tree, or the **base cases**. We stop recursing when one of the indices (i,j or k) becomes 0. If the sum of indices of the root of the tree is odd, sum of indices for all the leaves will be odd too (from above). Since one of the indices is zero, the other two won't be the same, and hence, as above, values at all the leaves in the tree would be zero. Thus, the value at the root, $TI(i, j, k) = 0$ if $(i + j + k)\ mod\ 2 \neq 0$.

□

Using the above theorem, the following result can be derived:

Result 1. Let $1 \leq i, j, k \leq K$. The total number of TI(i,j,k) is K^3. The number of **non-zero** TI's $\leq \lceil \frac{1}{4}k^3 + \frac{3}{8}k^2 + \frac{1}{4}k \rceil$. It is easy to derive the above number by using Theorem 1 and simple counting. **This result implies that only about $\frac{1}{4}$ of the TI entries are non-zero.**

2 Polynomial truncation scheme using Chebyshev polynomials

In many settings, we are interested in approximating a polynomial by a lower degree polynomial, while keeping the approximation error low. For example, we stated in Section 1.1 that if we use K terms for the Legendre expansion of two polynomials $f(x)$ and $g(x)$, then the product polynomial will have $2K$ terms. For a variety of computational considerations (memory, speed), it would be desirable to keep the degree of the product polynomial the same as the operands, i.e. K.

In this section of the report, we derive a truncation scheme to approximate a polynomial with a lower degree polynomial, while keeping the approximation error low under the L_∞ **norm**. In the particular case of computing a product of two polynomials in the Legendre domain, this scheme will help keep the degree of the product the same as that of the operands.

We use the Chebyshev polynomials to derive our truncation scheme. Chebyshev Polynomials are a set of orthogonal polynomials, and are denoted by $T_n(x)$. First few Chebyshev polynomials are given in Figure 3.They are defined using the following recurrence:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \qquad \text{with } \; T_0(x) = 1, \;\; T_1(x) = x$$

$T_0(x)$	1
$T_1(x)$	x
$T_2(x)$	$2x^2 - 1$
$T_3(x)$	$4x^3 - 3x$
$T_4(x)$	$8x^4 - 8x^2 + 1$
$T_5(x)$	$16x^5 - 20x^3 + 5x$
$T_6(x)$	$32x^6 - 48x^4 + 18x^2 - 1$

Figure 3: First few Chebyshev Polynomials

2.1 Approximation using Chebyshev Polynomials

Chebyshev polynomials find wide use in approximating polynomials with a lower degree one. In particular, the polynomial

$$p_{n-1}(x) = x^n - 2^{1-n}T_n(x)$$

is the best $n-1$ degree approximation for $f(x) = x^n$ on the interval $[-1,1]$ under the L_∞ norm, with the maximum deviation being 2^{1-n} [2]. We call $p_{n-1}(x)$ as the *degree $n-1$ mini-max polynomial*, as it minimizes the maximum deviation.

We extend the result by coming up with an approximation for the function $f(x) = x^{n+1}$ defined on the interval $[-1,1]$ with a polynomial of degree k, for any k, $0 \leq k \leq n$. It is a useful result since we can approximate x^{n+1} with a polynomial of any lower degree.

2.2 Truncation Scheme

Suppose we want a degree k approximation for x^n with $0 \leq k \leq n-1$. We start with the expression for $p_n(x)$, the degree $n-1$ approximation. Since, it is a degree $n-1$ polynomial, it can be written as:

$$p_n(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \ldots + c_m x^k + \ldots + c_0$$

Now, we replace x^{n-1} with degree $n-2$ mini-max polynomial in the above expression, which in turns gives us a degree $n-2$ approximation for x^n. Similarly, we cascade down, each time replacing the leading power of x in successive approximations with the one-lower degree mini-max polynomial. In each step, we reduce the degree by one; we repeat until we have a degree k approximation.

Note that although we don't have any theoretical error bounds, we present empirical results which suggest that the approximation error is quite low (Figure 4) , even for fairly low degree approximations.

2.3 Truncation as a matrix operation

In this section, we formulate the degree truncation operation as a matrix multiplication operation. We represent a polynomial $p(x)$ by a vector P of its coefficients. Let the degree k polynomial approximation of $f(x) = x^n$, according to our scheme, is given by $t_k^n(x)$, and the corresponding coefficient vector by T_k^n (size $k+1$).

Then, given a degree n polynomial $p_n(x)$, and its coefficient vector P_n, we can calculate $p_k(x)$ (P_k), its degree k approximation $(k < n)$ as a matrix multiplication:

$$\underbrace{P_k}_{k+1\times 1} = \underbrace{M_{nk}}_{k+1\times n+1} * \underbrace{P_n}_{n+1\times 1}$$

where,

$$M_{nk} = \begin{bmatrix} T_k^n & T_k^{n-1} & \dots & T_k^{k+1} & I_{k\times k} \end{bmatrix} \qquad , \quad I_{k\times k} \text{ is a } k \times k \text{ identity matrix.}$$

References

[1] M. ABRAMOWITZ AND I. A. STEGUN, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York, 1965, ch. 22.

[2] S. D. CONTE AND C. W. D. BOOR, *Elementary Numerical Analysis: An Algorithmic Approach*, McGraw-Hill Higher Education, 1980, pp. 242–244.

[3] R. NG, R. RAMAMOORTHI, AND P. HANRAHAN, *Triple product wavelet integrals for all-frequency relighting*, ACM Trans. Graph., 23 (2004), pp. 477–487.

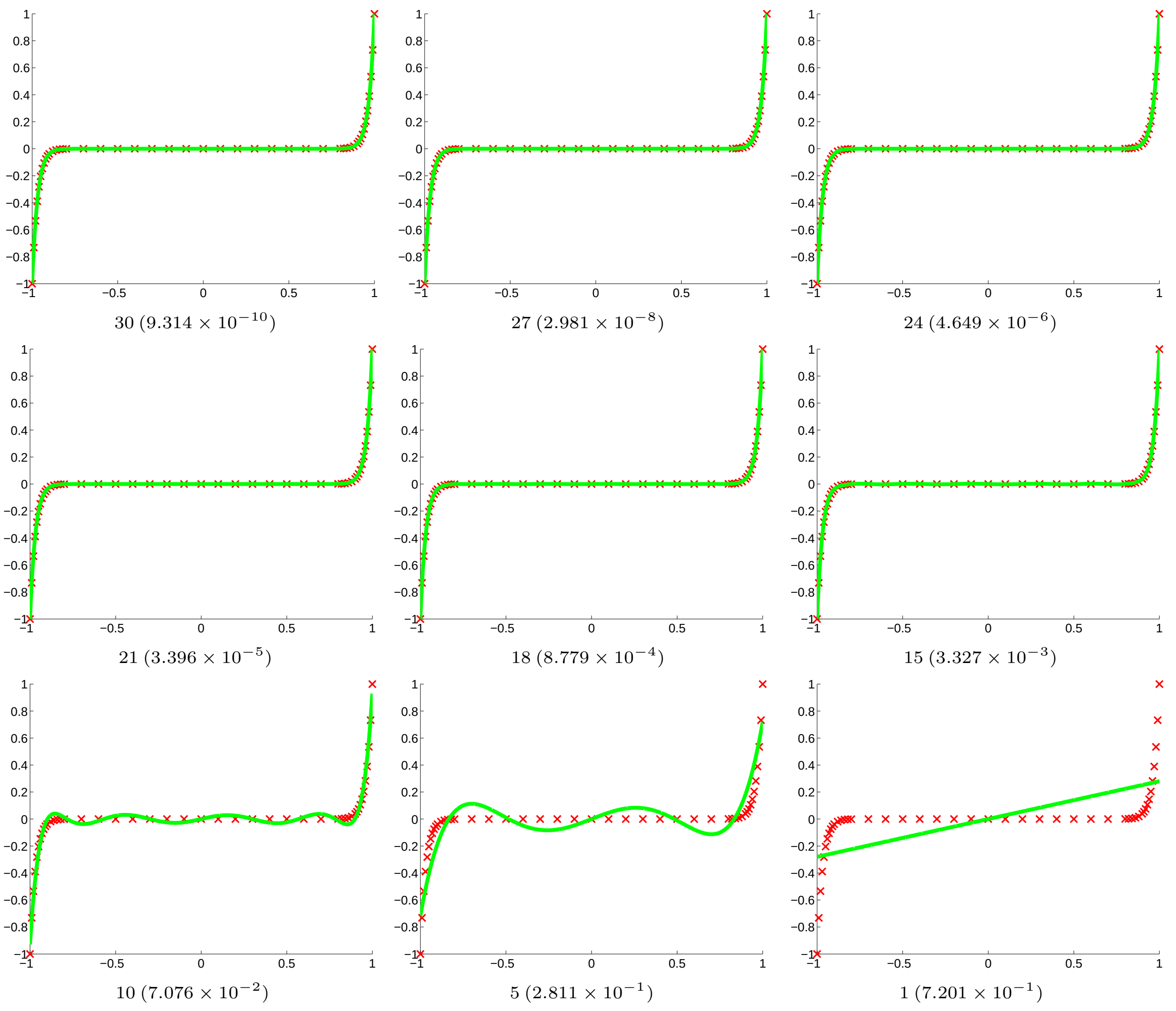

Figure 4: Lower Degree Polynomial Approximations of x^{31} with our algorithm in the interval $[-1, 1]$. Green plot is the approximation, and red crosses indicate the original function. Sub-Captions of plots are the degrees of approximation m. The numbers in parentheses are the maximum approximation error (L_∞ error norm) over the interval $[-1, 1]$. We can observe that for $m > 15$, the two plots are almost indistinguishable, which is remarkable, since we are approximating x^{31}. This establishes the accuracy of the algorithm in an empirical sense. For $m < 15$, we start noticing some error.