Accelerated Direct Demodulation Imaging Method *

SHEN Zong-Jun[1)] ZHOU Jian-Feng

(Department of Engineering Physics and Center for Astrophysics, Tsinghua University, Beijing 100084, China)

Abstract The most popular iteration method used in Direct Demodulation Method (DD) is Richardson-Lucy (RL) Iteration. The formula of RL iteration can be rewritten in matrix form. There are two matrix multiplications which contain the main computation cost. They can be transformed into convolution if the system is shift-invariant. As is well known, convolution can be computed by Fast Fourier Transform much faster than computed directly. This paper introduces the details of the above procedure, which is called accelerated direct demodulation method (ADD), and applies the procedure to image restorations of Hard X-ray Modulation Telescope data. This paper also compares the computation cost between the original DD and ADD.

Key words direct demodulate method, Richardson-Lucy iteration, HXMT

1 Introduction

The observation of an object in space x with intensity distribution $f(x)$ can be regarded as a modulation process which transforms $f(x)$ into observed data $d(\omega)$, where ω denotes the parameters of observation. The modulation process can be described by the following equation

$$\int p(\omega,x)f(x)\mathrm{d}x=d(\omega), \tag{1}$$

where the integral kernel $p(\omega,x)$ is the response coefficient of the instrument to a point x during an observation ω. Image restoration is to extract enough information from Eq. (1) and then restore $f(x)$. To extract more information from Eq. (1), many restoration methods have been developed, such as cross-correlation, maximum likelihood ratio, maximum entropy and so on. But these conventional methods cause loss of information contained in the modulation Eq. (1) seriously, and get low signal-noise ratio (SNR) and poor resolution images.

DD technique[1, 2] solves directly the modulation equations by iterations with physical constraints. The iteration method used by DD could be Gauss-Seidel iteration, Jacobi iteration or Richardson-Lucy iteration[3, 4] and so on. The DD method can gain high signal-noise ratio images, and mightily decrease the affection of background and noise. And it can greatly improve the locating precision and angular resolution, even much higher than the intrinsic angular resolution of the instruments.

DD method is a general inversion method, which can be used to deal with the observational data obtained by different kinds of instruments and has been used to analyze both synthetic data and the data from various types of space telescope, such as rotating modulation telescope[5], coded aperture mask telescope[6], imaging telescope, e.g. COS-B[7], ROSAT/PSPC[8, 9], XMM-Newton[10], Compton scattering telescope[11, 12], all-sky monitor on RXTE[13], slat collimator telescope EXOSAT/ME[14], HEAO1-A4[15], etc. All results show that the DD technique can significantly improve spatial resolutions and sensitivities.

Received 5 December 2006, Revised 15 January 2007

* Supported by National Natural Science Foundation of China (10603004)

1) E-mail: shenzj98@mails.tsinghua.edu.cn

Although DD can get images with much higher resolution and sensitivity than other image restoration methods, it converges slow and spends large computation time and memory. This restricts its application and causes difficulties in analyzing the data from large area detector with high sensitivity, e.g. Hard X-ray Modulate Telescope (HXMT).

To solve this problem, we rewrite the formula of RL iteration into matrix form. Then we prove that two multiplications of matrix, which contain the main computation cost, in RL iteration can be transformed into convolutions. It's well known that convolution can be computed much faster by Fast Fourier Transform[16] (FFT) than computed directly. Thus DD can be accelerated by many times and needs much less memory. In this paper, we will firstly introduce the accelerated DD (ADD) based on FFT in detail. Then, we will apply the ADD to the image restoration of HXMT.

2 Accelerated DD method

The discrete form of Eq. (1) is

$$\sum_j p(i,j)f(j)=d(i). \tag{2}$$

It can be written in matrix form as

$$\boldsymbol{PF}=\boldsymbol{D}\ , \tag{3}$$

where $\boldsymbol{P}$ is called response matrix. Image restoration is to solve Eq. (3) to estimate $\boldsymbol{F}$ when $\boldsymbol{P}$ and $\boldsymbol{D}$ are known. The DD method solves Eq. (3) by iterations with some physical constraints, where the background constraint is most widely used. In most cases RL iteration is a good choice. The DD method with RL iteration can be represented as

1. Let $r=0$, initialize $f^{(0)}(j)$ for every j.

2.

$$f^{(r+1)}(j)=f^{(r)}(j)\cdot\sum_k p(k,j)\frac{d(k)}{d^{(r)}(k)}\cdot\frac{1}{\sum_h p(h,j)}, \tag{4}$$

where

$$d^{(r)}(k)=\sum_j p(k,j)f^{(r)}(j). \tag{5}$$

3. For every j, if $f^{(r+1)}(j)<b(j)$, $f^{(r+1)}(j)=b(j)$.

4. If the criteria to stop the iteration is not satisfied, $r=r+1$, go to 2.

When practicing, the following matrix form is used

1. For every j,

$$s(j)=\sum_h p(h,j). \tag{6}$$

2. Let $r=0$, initialize $f^{(0)}(j)$ for every j.

3.

$$\boldsymbol{D}^{(r)}=\boldsymbol{PF}^{(r)}. \tag{7}$$

4. For every k

$$\boldsymbol{G}^{(r)}: g^{(r)}(k)=\frac{d(k)}{d^{(r)}(k)}. \tag{8}$$

5.

$$\boldsymbol{P}_0^{(r)}=\boldsymbol{P}^{\mathrm{T}}\boldsymbol{G}^{(r)}. \tag{9}$$

6. For every j

$$f^{(r+1)}(j)=\frac{f^{(r)}(j)p_0^{(r)}(j)}{s(j)}. \tag{10}$$

7. For every j, if $f^{(r+1)}(j)<b(j)$, $f^{(r+1)}(j)=b(j)$.

8. If the criterion to stop the iteration is not satisfied, $r=r+1$, go to 2.

In the above 8 steps, the main computation costs come from Eqs. (7) and (9), which contain two matrix multiplications. If there are M elements in $\boldsymbol{D}$ and N elements in $\boldsymbol{F}$, each multiplication contains $M\times N$ float multiplications. The store of the response matrix $\boldsymbol{P}$ holds the majority of memory. If stored in double precision, $\boldsymbol{P}$ needs $8\times M\times N$ bytes memory.

For a shift-invariant system, its response matrix $\boldsymbol{P}$ has the following property:

$$p(i,j)=p(i-k,j-k), \tag{11}$$

thus Eq. (5) can be transformed as

$$\begin{aligned} d^{(r)}(k)\ =\ & \sum_j p(k,j)f^{(r)}(j)= \\ & \sum_j p(k-j,0)f^{(r)}(j)= \\ & \sum_j psf(k-j)f^{(r)}(j), \end{aligned} \tag{12}$$

where $psf(j)=p(j,0)$, and then Eq. (7) can be transformed as

$$\boldsymbol{D}^{(r)}=\boldsymbol{PSF}*\boldsymbol{F}^{(r)}, \tag{13}$$

where $*$ means convolution, as is well known, which can be computed much faster than computed directly.

Similarly

$$p_0^{(r)}(j) = \sum_k p(k,j)g^{(r)}(k) = \sum_k p(k-j,0)g^{(r)}(k) = \sum_k psf(k-j)g^{(r)}(k) = \sum_k fsp(j-k)g^{(r)}(k), \tag{14}$$

and then Eq. (9) can be transformed as

$$\boldsymbol{P}_0^{(r)} = \boldsymbol{FSP} * \boldsymbol{G}^{(r)}, \tag{15}$$

where $\boldsymbol{FSP}$ is gotten by rotating $\boldsymbol{PSF}$ by 180°.

At last we replace Eqs. (7) and (9) by Eqs. (13) and (15) and then get the accelerated DD method (ADD). The method introduced above can be proved applicable to two dimensional cases in the same way.

3 Computation cost of ADD

Note that according to Eqs. (6—10), each original DD iteration contains $2MN$ float multiplications in Eqs. (7) and (9). Then we analyze the computation cost of ADD.

The main computation cost of ADD comes from Eqs. (13) and (15). As is well known, the number of complex multiplications needed by L-points FFT is $\frac{L}{2}\log_2 L$. When we compute Eq. (13), we need first to add zeros to the end of $\boldsymbol{PSF}$ and $\boldsymbol{F}^{(r)}$ until their lengths are $M+N-1$. It needs two FFTs and one inverse FFT to compute Eq. (13). But $\boldsymbol{PSF}$ and $\boldsymbol{FSP}$ won't change during the iteration, so their FFT can be computed only once before the iteration starts and the expense can be ignored. Thus the main computation cost of Eq. (13) is $(M+N-1)\log_2(M+N-1) \approx (M+N)\log_2(M+N)$ complex multiplication. Similarly, the computation cost of Eq. (15) is $(2M-1)\log_2(2M-1) \approx 2M\log_2(2M)$. Considering that one complex multiplication contains 4 float multiplications, we get the acceleration factor of ADD to the original DD

$$a = \frac{2MN}{4(M+N)\log_2(M+N) + 8M\log_2(2M)}. \tag{16}$$

In many cases we get $M = N$, thus $a = \frac{N}{8(1+\log_2 N)}$.

In two dimensional cases, we have to add more zeros to $\boldsymbol{PSF}$ and $\boldsymbol{F}^{(r)}$ before computing Eqs. (13) and (15). So we get a smaller factor of acceleration. If $\boldsymbol{D}$ and $\boldsymbol{F}$ have N elements respectively, $a_2 \approx \frac{N}{16(2+\log_2 N)}$. when $N = 121\times 121$, $a_2 = 58$; when $N = 1024\times 1024$, $a_2 = 2980\cdots$ Larger N, larger acceleration.

In DD method, $8MN$ bytes memory is needed to store response matrix $\boldsymbol{P}$, but in ADD method, $\boldsymbol{P}$ is replaced by $\boldsymbol{PSF}$ which is just one column of $\boldsymbol{P}$ and expends only $8M$ bytes memory.

The FFTs in Eqs. (13) and (15) transform real sequence to complex sequence, and IFFTs in Eqs. (13) and (15) transform complex sequence to real sequence. Both of them don't transform complex to complex. So an FFT function optimized for this feature gives ADD a further acceleration. The libraries of FFT are optimized in different ways for various applications. Thus it is difficult to precisely calculate the acceleration factor a in real implementation. However, Eq. (16) is still a good approximate estimation.

4 The application of ADD to HXMT

Based on the DD technique, a high energy astrophysics mission, Hard X-ray Modulation Telescope (HXMT), has been proposed. Also an imaging test setup has been built[17] to verify the detector performance and the properties of HXMT. HXMT uses simple non-position-sensitive collimated detectors to realize high sensitivity and high resolution hard X-ray imaging survey. The detector of HXMT consists of eighteen identical NaI(Tl)/CsI(Na) phoswiches. The area of single module is 283cm^2, and the total detecting area is about 5000cm^2. the FOV of HXMT is $5.7°\times 5.7°$ (FWHM), composed by eighteen non-symmetric FOV of $5.7°\times 1.1°$ which is related to one of eighteen collimators. The eighteen collimators are placed with a cross angle of 10° with respect to one another. Fig. 1 is the diagrammatic sketch of the installation of eighteen detector modules (collimators).

In theory, the data from any individual detector of HXMT can be used for imaging by ADD introduced in Section 2. However, due to the small effective area of one detector, the obtained images will have low sensitivity and SNR, and poor angular resolutions along the direction of the long axis of FOV. The data of all 18 detectors must be used simultaneously to fully exert the imaging ability of HXMT. But we can't use the data of more than one detector in ADD introduced in Section 2 directly. ADD needs to be modified.

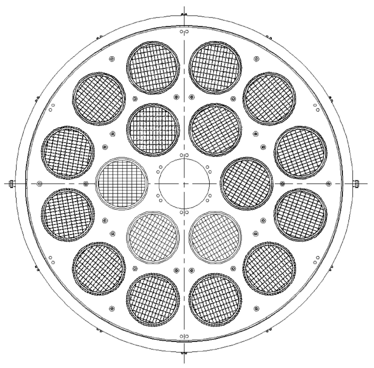

Fig. 1. The installation of the 18 detector modules (collimators) of HXMT.

The response matrix $\boldsymbol{P}$ of the whole system (all the 18 detectors) consists of 18 response matrixes of 18 subsystems (each corresponds to one detector), and the data $\boldsymbol{D}$ of the whole system consists of the data of 18 subsystems. Eq. (7) can be transformed to

$$\boldsymbol{D}^{(r)}=\begin{bmatrix}\boldsymbol{D}_1^{(r)}\\ \boldsymbol{D}_2^{(r)}\\ \cdots\\ \boldsymbol{D}_{18}^{(r)}\end{bmatrix}=\boldsymbol{P}\boldsymbol{F}^{(r)}=\begin{bmatrix}\boldsymbol{P}_1\boldsymbol{F}^{(r)}\\ \boldsymbol{P}_2\boldsymbol{F}^{(r)}\\ \cdots\\ \boldsymbol{P}_{18}\boldsymbol{F}^{(r)}\end{bmatrix}, \tag{17}$$

namely

$$\boldsymbol{D}_i^{(r)}=\boldsymbol{P}_i\boldsymbol{F}^{(r)}=\boldsymbol{PSF}_i*\boldsymbol{F}^{(r)},\quad i=1,2,\cdots,18. \tag{18}$$

Eq. (9) can also be transformed as

$$\boldsymbol{P}_0^{(r)} = \boldsymbol{P}^{\mathrm{T}}\boldsymbol{G}^{(r)}=\left[\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{P}_2^{\mathrm{T}}\cdots\boldsymbol{P}_{18}^{\mathrm{T}}\right]\begin{bmatrix}\boldsymbol{G}_1^{(r)}\\ \boldsymbol{G}_2^{(r)}\\ \cdots\\ \boldsymbol{G}_{18}^{(r)}\end{bmatrix}=$$

$$\sum_{i=1}^{18}\boldsymbol{P}_i^{\mathrm{T}}\boldsymbol{G}_i^{(r)}=\sum_{i=1}^{18}\boldsymbol{FSP}_i*\boldsymbol{G}_i^{(r)}\ . \tag{19}$$

So, the 8 steps in Section 2 can be rewritten as

1. For every j,

$$s(j)=\sum_{i=1}^{18}\sum_{h}p_i(h,j). \tag{20}$$

2. Let $r=0$, initialize $f^{(0)}(j)$ for every j

3.

$$\boldsymbol{D}_i^{(r)}=\boldsymbol{PSF}_i*\boldsymbol{F}^{(r)},\quad i=1,2,\cdots,18\ . \tag{21}$$

4. For every k,

$$\boldsymbol{G}_i^{(r)}:g_i^{(r)}(k)=\frac{d_i(k)}{d_i^{(r)}(k)},\quad i=1,2,\cdots,18\ . \tag{22}$$

5.

$$\boldsymbol{P}_0^{(r)}=\sum_{i=1}^{18}\boldsymbol{FSP}_i*\boldsymbol{G}_i^{(r)}. \tag{23}$$

6. For every j,

$$f^{(r+1)}(j)=\frac{f^{(r)}(j)p_0^{(r)}(j)}{s(j)}. \tag{24}$$

7. For every j, if $f^{(r+1)}(j)<b(j)$, $f^{(r+1)}(j)=b(j)$.

8. If the criteria to stop the iteration is not satisfied, $r=r+1$, go to 3

There is an FFT of F in every convolution in Eq. (21), so the FFT of F can be computed only once. Thus we can reduce 17 FFTs.

A test which aims to restore an image of size $12^\circ\times 12^\circ$ is applied to compare the performance between DD and ADD. There are two sources, one located at $(-0.5^\circ,-0.5^\circ)$, another located at $(0.5^\circ,0.5^\circ)$. Their intensities are both $1.24\times10^{-3}/(\mathrm{cm}^2\cdot\mathrm{s})$. The intensity of background is $0.027/(\mathrm{cm}^2\cdot\mathrm{s})$. The image is split into 121×121 pixels. Each pixel is $0.1^\circ\times0.1^\circ$. When the collimators point to one pixel of the image, 18 observed data from 18 detectors are obtained. Thus $\boldsymbol{F}$ and $\boldsymbol{PSF}_i$, $\boldsymbol{D}_i$ are two dimensional matrixes which have 121×121 elements. And $\boldsymbol{P}_i$ is four dimensional matrix which has $121\times121\times121\times121$ elements, and $\boldsymbol{P}$ consists of $\boldsymbol{P}_1$, $\boldsymbol{P}_2\cdots\boldsymbol{P}_{18}$.

Table 1 is the comparison between the computation cost of original DD iteration and ADD iteration (100 iterations). According to Table 1, the above application of ADD to HXMT is about 36 times faster than the original DD method. And the memory requirement is decreased from unacceptable 28.75GB to 60MB. This enables the DD imaging of HXMT be run on a PC easily.

Table 1. The comparison between the computation cost of original RL iteration and ADD iteration (100 iterations).

	time cost	memory cost
original DD iteration	1440s	28.75GB
ADD iteration	40s	60MB

ADD is equivalent to the original DD in mathematical sense. They ought to obtain identical result with the same input. The largest relative difference between DD image and ADD image (the maximum of |DDimage−ADDimage|/DDimage) in this simulation is 5×10^{-9}. This tinny difference is due to the errors in numerical computation and is ignorable. The left part of Fig. 2 is the cross-correlation image, in which the two sources are merged into one wide peak. But in the DD (ADD) image (the right part of Fig. 2), they are separated clearly.

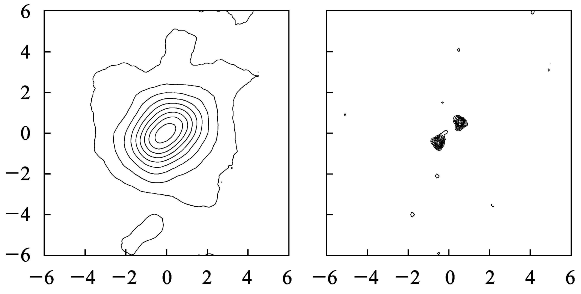

Fig. 2. Left: Cross-correlation image; Right: DD(ADD) image.

5 Discussion

To optimize the matrix operation like that in RL iteration, the first and natural idea which comes to our mind is to use sparse matrix, which removes the operations of zero-element in the matrixes. In the above application, 12.4% elements of the response matrix P are zeros. Considering the memory cost of the indexes of nonzero elements, the total memory cost is about $28.75\text{GB}\times1.5\times12.4\%=5.33\text{GB}$. It is also unacceptable for a PC. Even if there is no extra cost of the indexing of sparse matrix, the speed can be only increased by about 8 times. The ultimate angular resolution of HXMT is $5'$. To separate two sources $5'$ away from each other, the pixels of the image must be smaller than $2.5'\times2.5'$. The size of such an image would be larger than 288×288. It is impossible for DD to deal with such a big image even optimized by sparse matrix. But ADD can easily process 1024×1024 images whose pixel resolution is about $0.7'$ by a PC only. Thus the imaging ability of HXMT can be fully exerted. And in such a case ADD will be thousands of times faster than the original DD (if there were computers powerful enough to compute such DD iterations).

In the above application, the memory cost of the original DD is 28.75G. But it's very hard to find a computer which has so much memory. So the “28.75GB” comes from theoretical calculation, and the time cost of the original DD in Table 1 comes from the direct computation of the convolutions in Eqs. (13) and (15), because they contains the same amount of multiplications and additions with Eqs. (7) and (9), but Eqs. (13) and (15) demand an acceptable amount of memory.

In some practical cases computing these two convolutions directly is better than computing them by FFT. When the system's $\boldsymbol{PSF}$ is a small matrix, out-of-focus blur and motion blur for example, computing the convolutions in Eqs. (13) and (15) by FFT expends more time and memory than computing them directly which is the best way in such a case. The orbits of high resolution scout satellites are usually very low and the ground speed of these satellites is very high. This makes the photographs taken by these satellites blur. Otherwise, the impulse responses of the optical systems and CCD arrays are not δ functions. This brings blur too. It's feasible to remove these kinds of blur and improve the quality of these images by ADD.

The algorithm introduced in this paper is to make the computing of each DD iteration faster, not to make the DD method converge faster. The ordered subset method[18] is a good choice to make the DD iteration converge faster. In the application in Section 4, the data of each detector can be grouped into one subset. Thus the ordered subset would make the DD iteration converge 18 times faster than the original. And if the computer have enough memory for sparse-matrix-optimized DD, the data of each detector could be grouped into more subsets and the speed of convergence would be higher. The subsets ought to be carefully selected, otherwise it will cause.

In these years, there are more and more multi-CPU computers and multi-kernel CPUs. This makes the parallel computation more and more important and widely used. And ADD can be programmed in multi-thread mode. Some FFT libraries, such as FFTW[19], which support multi-thread computation, can be used to accelerate the generic ADD introduced in Section 2. In the case of HXMT, each of the 18 convolutions in Eq. (21) or Eq. (23) can be computed by a single thread simultaneously. This is better than using multi-thread FFT functions because there is less dependency between these threads than between the threads in multi-thread FFT functions. Moreover, modern GPUs (graphic processing units) have many pipelines for stream computation, which make a GPU much more powerful than the best CPUs in parallel stream computations[20]. The main part of DD, as well as ADD, is stream computations. And there are libraries for GPU computation. ADD can be accelerated considerably by these libraries.

This research has made use of the Astrophysical Integrated Research Environment (AIRE) which is operated by the Center for Astrophysics, Tsinghua University.

References

1 LI T P, WU M. Ap & SS, 1993, **206**: 91

2 LI T P, WU M. Ap & SS, 1994, **215**: 213

3 Richardson B M. J. Opt. Soc. Am., 1972, **62**: 55

4 Lucy L B. AJ, 1974, **79**: 745

5 CHEN Y, LI T P, WU M. A & AS, 1998, **128**: 363

6 LI T P. Exper. Astron., 1995, **6**: 63

7 LI T P, WU M, LU Z G et al. Ap & SS, 1993, **205**: 381

8 CHEN Y, LI T P, WU M. In: Proc. ADASSVI. ASP Conf. Ser., 1997, **125**: 178

9 LU F J, Aschenbach B, SONG L M. A & A, 2001, **370**: 570

10 FENG H, CHEN Y, ZHANG S N et al. A & A, 2003, **402**: 1151

11 ZHANG S, LI T P, WU M et al. In: AIP Conf. Proc. 1997, **410**: 578

12 ZHANG S, LI T P, WU M. A & A, 1998, **340**: 62

13 SONG L M, LI T P, CUI W. Acta Astrophys. Sinica, 1999, **19**(1): 27—32 (in Chinese)
(宋黎明, 李惕碚, 崔伟. 天体物理学报, 1999, **19**(1): 27—32)

14 LU F J, LI T P, SUN X J et al. A & AS, 1996, **115**: 395

15 LU F J, LI T P, SUN X J et al. In: Shellard & Nguyen (eds) Proceedings of CHEP'95. 1995. 848

16 Cooley J W, Tukey J W. Mathematics of Computation, 1965, **19**(Apr): 297—301

17 DONG Yong-Wei, WU Bo-Bing, LI Yan-Guo et al. HEP & NP, 2006, **30**(5): 392—397 (in Chinese)
(董永伟, 吴伯冰, 李延国等. 高能物理与核物理, 2006, **30**(5): 392—397)

18 Hudson H M, Larkin R S. IEEE Trans. Med. Im., 1994, **13**(4): 601

19 http://www.fftw.org

20 Buck I, Foley T, Horn D et al. ACM Transactions on Graphics, 2004, Special Issue, **23**(3): 777—786

直接解调成像的快速算法*

沈宗俊1) 周建锋

(清华大学工程物理系, 清华大学天体物理中心 北京 100084)

摘要 直接解调方法(Direct Demodulate Method, DD)中运用的最多的迭代方法是Richardson-Lucy (RL)迭代. RL迭代的公式可以改写成矩阵形式, 其中包含了主要计算量的两次矩阵乘法在点扩展函数满足平移不变性的情况下可以写成卷积的形式, 而卷积可以使用快速傅里叶变换(Fast Fourier Transform, FFT)做快速计算. 详细介绍了上述过程, 并将该方法应用到了硬X射线调制望远镜(Hard X-ray Modulate Telescope, HXMT)的DD成像中, 通过理论的和模拟的计算对比了优化前后DD方法对计算资源的开销.

关键词 直接解调方法 Richardson-Lucy迭代 硬X射线调制望远镜

2006 – 12 – 05 收稿, 2007 – 01 – 15 收修改稿
∗ 国家自然科学基金(10603004)资助
1) E-mail: shenzj98@mails.tsinghua.edu.cn