Convergence of ART in Few Projections

Tanuja Srivastava[1]
Department of Mathematics
Indian Institute of Technology
Roorkee, Uttarakhand

Nirvikar[2]
Department of CS & IT
Shobhit University
Meerut, UP

Raghuvir Singh[3]
Shobhit University
Meerut, UP

Abstract— Algebraic Reconstruction Technique (ART) is an iterative algorithm to obtain reconstruction from projections in a finite number of iterations. The present paper discuses the convergence achieved in small number of iteration even when projection data is available in only four directions.

Keywords-ART, Image Reconstruction, Convergence, Projections, Computed Tomography

I. INTRODUCTION

Computed Tomography (CT) is a diagnostic procedure that uses special x-ray equipment to obtain cross-sectional pictures of the body. The CT computer displays these pictures as detailed images of organs, bones, and other tissues. This procedure is also called CT scanning, computerized tomography, or computerized axial tomography (CAT) [1]. CT is a two step process of collecting the projection data, then calculating the attenuation values that could have generated these projection values (reconstruction). Two modalities that limit the radiation from Computed Tomography are then presentation: 3-D cone beam reconstruction, and limited view Computed Tomography.

Computed Tomography has a vital role in medical diagnostics as an imaging method that yields detailed information. As an X-ray technology, however, it exposes the patient to ionizing radiation that is known to be harmful. A challenge for this technology is to obtain the high quality images that have come to be expected from it, while limiting this harmful radiation. CT uses multiple X-ray views of the target for image reconstruction [2]. Each view is associated with a dose of radiation, hence limiting the number of views will reduce the radiation. Using a true 3-D reconstruction from 2-D views, rather than assembling from 2-D reconstructions of 1-D views, should theoretically reduce the number of views required. A second approach is to limit the number of views outright. Modifications of the commonly used convolution algorithms allow for some limited reconstruction in the third dimension, but these algorithms are not ideal for a full 3-D reconstruction [3]. Limiting the number of views outright causes the standard reconstruction algorithms to fail. The *Algebraic Reconstruction Technique* (ART) and similar *iterative* algorithms yield better quality reconstructions using limited views or a true 3-D reconstruction, but these algorithms are much more costly in execution time and memory. We can ameliorate this cost in execution time and memory by running the algorithm in parallel. Significant speed benefits are obtained compared to the sequential version of the ART algorithm. The iterative algorithms may have a role to play in limited view CT reconstruction or in 3-D CT reconstruction, and should not be rejected out of hand because of speed or memory limitations. These limitations are overcome significantly by implementing the algorithms in parallel, if communication is limited and an appropriate partitioning scheme is used.

The goal of medical imaging is to determine the internal structure of an organism with sufficient detail to yield diagnostic information. Imaging strives to achieve this in the least invasive manner possible, minimizing discomfort and harm to the patient. Two dimensional plain film X-ray pictures have been the standard medical imaging technique for a century, and remain a common technique today. An X-ray exposure of sufficient intensity and duration is used to project shadows of body tissues onto the detecting surface. The X-ray 'beam' is attenuated by scattering and absorption of the intervening tissue proportional to the distance the beam must traverse the tissue, the density of the tissue, and the atomic numbers of the contained elements. X-ray projections have at least three limitations: limited projection angles from which the X-ray view can be taken; inability to localize the 3-D position of a structure; and, most importantly, a lack of detail due to lack of contrast. The limitation of viewing angles is imposed by the target object and the imaging equipment.

There are two major families of reconstruction methods: filtered or Fourier backprojection (FBP), or convolution backprojection (CBP) methods, and iterative, or algebraic techniques. An image can be obtained by adding the detection value to every contributing voxel in the projection. If the target object has sharply defined contrasting regions, this *summation* method will cause these to be *blurred* much like a photograph out of focus. This summation is termed *backprojection* because it involves placing the projections back into the image. The terms *straight* backprojection or *unfiltered* backprojection is refered to the process when no other operations are performed on the backprojection image. We use the terms *Filtered backprojection (FBP)* or *convolution backprojection (CBP)* to refer to the whole group of filtered backprojection methods [4]. Although in principle the backprojected image could be deconvoluted using a 2-D filter, an equivalent transformation can be obtained by passing a 1D filter over the projection data *before* the backprojection in the case of parallel projections [5].

II. ALGEBRAIC RECONSTRUCTION TECHNIQUE (ART)

The Algebraic Reconstruction Technique (ART) was proposed by Gordon, Bender, and Herman as a method for the reconstruction of three-dimensional objects from electron-microscopic scans and X-ray photography [6]. There are number of variants which are known by the acronyms ART [7], SIRT (simultaneous iterative reconstruction technique) and SART (simultaneous algebraic reconstruction technique). In algebraic methods, the reconstruction is done by solving a system of linear equations. More precisely, ART can be written as a linear algebra problem, Af = P, where f is the unknown ($N^2 \times 1$) vector storing the values ($f_1, ..., f_N$) of all $N = n^2$ *surface elements* or pixels in 2D or $N = n^3$ *volume elements* or voxels in 3D respectively, in the reconstruction grid. So, the image is represented as a single point in a N-dimensional space. P is the ($LK\times 1$) vector composed of the p_i values that represent the ray-sum measured with the ith ray, where L is number of views covering whole image suitable dispersed (equispaced on angular view) and K is the number of equispaced lines along each view, M is the total number of rays in all acquired projections. Finally, A is the ($M\times N$) weight (coefficient) matrix in which an emlement w_{ij} represents the contribution of the jth cell to the ith ray integral. The factor w_{ij} is equal to the fractional area of the jth image cell intercepted by the ith ray for one of the cells. The most of the w_{ij}'s are zero since only a small number of cells contribute to any given ray-sum. Algebraic Reconstruction Techniques (ART) was first published in the biomedical imaging literature in 1970 [7]. ART is a form of Gauss-Seidel iteration, and can be viewed as a generalization of the method of Kaczmarz in 1937 [8]. Algebraic Reconstruction Technique (ART) is a widely-used iterative method for solving sparse systems of linear equations. The main advantages of ART are its robustness, its cyclic convergence on inconsistent systems, and its relatively good initial convergence. ART is widely used as an iterative solution to the problem of image reconstruction from projections in computerized tomography (CT), where its implementation with a small relaxation parameter produces excellent results. It is shown that for this particular problem, ART can be implemented in parallel on a linear processor array [9].

The problem of CT reconstruction can be viewed as a system of linear equations. In this model, each pixel (voxel) j is assumed to have a homogenous attenuation fj, an unknown value to be solved. The measured projection data is a set of attenuation sums Pi. Each Pi is the weighted sum of the attenuations of pixels along a given ray, also known as a ray integral or ray sum. Different variations of the model can be used to determine the weight *wij* that each pixel j contributes to the ith weighted attenuation sum Pi. Let us use a model where each weight *wij* is the product of the pixel's attenuation fj and the length of the ray's intersection with the pixel (expressed in pixel widths). The weights can then be determined geometrically from the angle and position of the ray (these are determined from the geometry of the scanner) and the chosen pixel dimensions. As an example, we have an image of N = 4 pixels. There are 2 detectors in the detector array, and the array is rotated through 2 views (horizontal and vertical) to produce M = 4 ray sums [10]. We therefore have $M\,N = 16$ weights. The weights for raysum P_1 are calculated easily in this case. The ray traverses the width of pixel 1, so the weight of contribution of pixel 1 to the raysum is $w_{11} = 1$. Likewise, $w_{12} = 1$. Pixels 3 and 4 do not intersect ray 1, so $w_{13} = w_{14} = 0$. Similarly for the other rays in this example, all weights are 0 or 1, and the ray sum equations are as follows:

$$\begin{aligned} P_1 &= f_1w_{11} + f_2w_{12} + f_3w_{13} + f_4w_{14} = f_1 + f_2 \\ P_2 &= f_1w_{21} + f_2w_{22} + f_3w_{23} + f_4w_{24} = f_3 + f_4 \\ P_3 &= f_1w_{31} + f_2w_{32} + f_3w_{33} + f_4w_{34} = f_2 + f_4 \\ P_4 &= f_1w_{41} + f_2w_{42} + f_3w_{43} + f_4w_{44} = f_1 + f_3 \end{aligned} \qquad (1)$$

In general, each ray Pi can be represented as:

$$P_1 = \sum_{j=0}^{N} w_{ij} f_j, i = 1,2,\ldots,M \qquad (2)$$

where M is the total number of rays(in all the projections) and w_{ij} is the weighting factor that represents the contribution of the jth image cell to the ith ray sum. The subscript i represents the projection index from a total of $\boldsymbol{M}$ projections. The subscript j represents the image index among $\boldsymbol{N}$ image cells. Half of the NM weights w_{ij} are zero. For the case of the 9 pixel Fig. 1 approximately two thirds of the weights are zero.

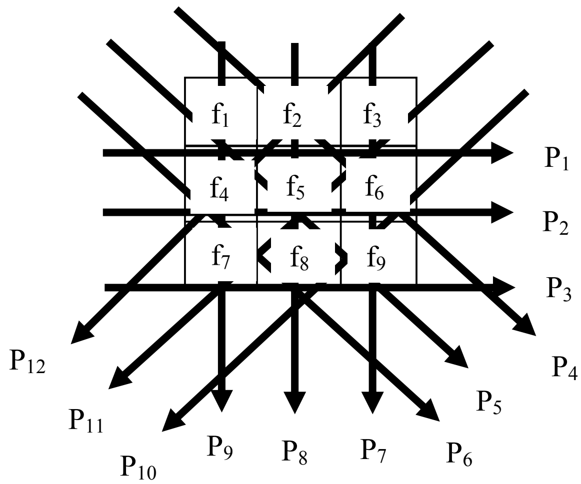

Fig. 1 The reconstruction problem as a system of linear equation

For the computer implementation of this method, we first make an initial guess at the solution. This guess, denoted by $f_1^{(0)}, f_2^{(0)}, \ldots, f_N^{(0)}$, is represented vectorially by $\vec{f}^{(0)}$ in the N-dimensional space. In most cases, we simply assign a value of zero to all the f_i's. This initial guess is projected on the hyperplane represented by the equation in (2). When we project the (i - 1)th solution onto the ith hyperplane [ith equation in (2)] the gray level of the jth element, whose current value is $f_j^{(i-1)}$, is obtained by correcting its current value by $\Delta f_j^{(i)}$, where

$$\Delta f_j^{(i)} = f_j^{(i)} - f_j^{(i-1)} = \frac{p_i - q_i}{\sum_{k=1}^{N} w_{ik}^2} w_{ij}. \quad (3)$$

Note that while p_i is the measured ray-sum along the ith ray, q_i may be considered to be the computed ray-sum for the same ray based on the (i -1)th solution for the image gray levels. The correction Δf_j, to the jth cell is obtained by first calculating the difference between the measured ray-sum and the computed ray-sum, normalizing this difference by $\sum_{k=1}^{N} w_{ik}^2$, and then assigning this value to all the image cells in the ith ray, each assignment being weighted by the corresponding w_{ij} In general, for large images, a substantial portion of the weights are zero, because many of the pixels make no contribution to a particular raysum. One approach to solving large systems of equations, iterative approximations, forms the basis of the *iterative* or *algebraic* methods. Successive adjustments are made to the attenuation values until a solution is reached that is consistent with the projection values by some criterion. Iterative methods compare the computed ray sums of an estimated image with the original projection measurements and use the error obtained from this comparison to correct the estimated image. Though there is unlikely to be an exact solution because of inconsistencies, this method yields an approximate solution to the attenuation values.

III. ART EXAMPLE

ART consists of three steps:

1. Make an initial guess at the solution
2. Compute projections based on the guess
3. Refine the guess on the weighted difference between the actual projections and desired projections: $p^{i+1} = p^i + a(desired - actual)$

We have an image of 8 X 8 e.g. N = 64 pixels. There are 3 detectors in the detector array, and the array is rotated through 4 views (horizontal, vertical, diagonal and antidiagonal) to produce M = 46 raysums.

Starting with initial guess $f^{(0)}$ and projections p.

Table 1. Given Projection Value (p)

18	16	19	10	08	06
12	09	15	12	07	23
18	14	17	05	23	09
14	17	22	24	28	13
19	12	10	27	26	24
12	29	26	15	16	12
21	24	19	09	12	13
11	10	23	14	0	0

Table 2. Initial Image Data ($f^{(0)}$)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Table 3. Reconstructed Image after 55 iterations

9.52	5.12	2.91	3.06	5.06	7.56	7.44	2.18
8.26	2.38	2.45	2.48	2.57	9.36	6.89	2.89
4.77	5.37	1.97	7.80	5.38	2.43	4.81	4.17
4.12	4.63	4.03	1.95	3.71	1.87	1.86	4.11
2.48	3.71	4.09	6.74	1.79	1.78	1.79	2.25
2.56	5.01	3.04	4.7	2.01	1.81	1.95	2.32
1.74	2.38	2.32	7.52	7.09	2.28	2.11	5.34
4.66	1.58	2.57	3.73	5.19	2.76	8.19	2.57

Table 4. Error calculated in Image pixel values in each iteration

Iterations	$\lvert f_{i+1} - f_i \rvert$	$(f_{i+1} - f_i)^2$
iteration 1	260.593750	1196.627930
iteration 2	23.196289	13.700123
iteration 3	11.863609	3.781413
iteration 4	7.293658	1.328680
iteration 5	4.558233	0.521878
iteration 6	3.167742	0.264142
iteration 7	2.423008	0.162381
iteration 8	2.041324	0.120152
iteration 9	1.793079	0.097458
iteration 10	1.660157	0.085381
iteration 11	1.581798	0.077365
iteration 12	1.523060	0.071226
iteration 13	1.475091	0.066224
iteration 14	1.392426	0.058907
iteration 15	1.280310	0.049823
iteration 16	1.194516	0.044186
iteration 17	1.149647	0.040100
iteration 18	1.113286	0.036847
iteration 19	1.080504	0.034161
iteration 20	1.031179	0.031026
iteration 21	0.921514	0.026042
iteration 22	0.870406	0.023270
iteration 23	0.828576	0.021274
iteration 24	0.793581	0.019673
iteration 25	0.763604	0.018317
iteration 26	0.730673	0.016935
iteration 27	0.667299	0.014663
iteration 28	0.625723	0.013222
iteration 29	0.595970	0.012142
iteration 30	0.570434	0.011251
iteration 31	0.547994	0.010501
iteration 32	0.526815	0.009855
iteration 33	0.507945	0.009292
iteration 34	0.490216	0.008796
iteration 35	0.473685	0.008356
iteration 36	0.458431	0.007963
iteration 37	0.445886	0.007611
iteration 38	0.405795	0.006125
iteration 39	0.383962	0.005421
iteration 40	0.364060	0.005004
iteration 41	0.349298	0.004716
iteration 42	0.337096	0.004498
iteration 43	0.326862	0.004320
iteration 44	0.312715	0.004116
iteration 45	0.303181	0.003968
iteration 46	0.282642	0.003183
iteration 47	0.260532	0.002731
iteration 48	0.247859	0.002447
iteration 49	0.239172	0.002243
iteration 50	0.231610	0.002081

Iterations	$\lvert f_{i+1} - f_i \rvert$	$(f_{i+1} - f_i)^2$
iteration 51	0.224570	0.001947
iteration 52	0.218310	0.001832
iteration 53	0.212647	0.001732
iteration 54	0.207277	0.001643
iteration 55	0.202114	0.001562
iteration 56	0.197140	0.001489
iteration 57	0.189942	0.001412
iteration 58	0.184316	0.001340
iteration 59	0.179717	0.001276
iteration 60	0.175271	0.001217
iteration 61	0.171076	0.001162
iteration 62	0.167109	0.001112
iteration 63	0.163492	0.001064
iteration 64	0.160142	0.001020
iteration 65	0.156920	0.000978
iteration 66	0.153788	0.000939
iteration 67	0.150822	0.000901
iteration 68	0.147948	0.000866
iteration 69	0.145214	0.000832
iteration 70	0.142620	0.000800
iteration 71	0.140079	0.000769
iteration 72	0.137586	0.000740
iteration 73	0.135143	0.000712
iteration 74	0.132742	0.000685
iteration 75	0.130393	0.000660
iteration 76	0.128084	0.000635
iteration 77	0.125825	0.000612
iteration 78	0.123611	0.000589
iteration 79	0.121439	0.000568
iteration 80	0.119310	0.000547
iteration 81	0.117219	0.000527
iteration 82	0.115158	0.000508
iteration 83	0.113141	0.000490
iteration 84	0.111161	0.000472
iteration 85	0.109214	0.000455
iteration 86	0.107293	0.000439
iteration 87	0.105417	0.000423
iteration 88	0.103568	0.000408
iteration 89	0.101752	0.000393
iteration 90	0.099965	0.000379
iteration 91	0.098218	0.000366
iteration 92	0.096495	0.000353
iteration 93	0.094801	0.000340
iteration 94	0.093147	0.000328
iteration 95	0.091513	0.000317
iteration 96	0.089912	0.000306
iteration 97	0.088335	0.000295
iteration 98	0.086793	0.000284
iteration 99	0.085267	0.000274
iteration 100	0.083781	0.000265

Table 5. Error calculated in projection values in each iteration

Iteration	$\lvert p_{i+1} - p_i \rvert$	$(p_{i+1} - p_i)^2$
iteration 1	743.000000	552049.000000
iteration 2	299.375000	89625.390625
iteration 3	34.851563	1214.631444
iteration 4	7.800537	60.848377
iteration 5	2.252258	5.072666
iteration 6	1.376648	1.895160
iteration 7	1.503113	2.259349
iteration 8	0.413391	0.170892
iteration 9	0.408203	0.166630
iteration 10	0.084229	0.007095
iteration 11	0.162964	0.026557
iteration 12	0.281250	0.079102
iteration 13	0.287781	0.082818
iteration 14	0.296143	0.087701
iteration 15	0.178223	0.031763
iteration 16	0.026123	0.000682
iteration 17	0.151184	0.022857
iteration 18	0.176514	0.031157
iteration 19	0.169983	0.028894
iteration 20	0.170715	0.029144
iteration 21	0.112366	0.012626
iteration 22	0.139038	0.019332
iteration 23	0.039978	0.001598
iteration 24	0.069458	0.004824
iteration 25	0.083069	0.006900
iteration 26	0.085999	0.007396
iteration 27	0.069275	0.004799
iteration 28	0.086426	0.007469
iteration 29	0.016052	0.000258
iteration 30	0.037842	0.001432
iteration 31	0.045349	0.002057
iteration 32	0.049072	0.002408
iteration 33	0.051208	0.002622
iteration 34	0.053040	0.002813
iteration 35	0.054199	0.002938
iteration 36	0.055054	0.003031
iteration 37	0.055420	0.003071
iteration 38	0.055847	0.003119
iteration 39	0.165771	0.027480
iteration 40	0.128113	0.016413
iteration 41	0.078308	0.006132
iteration 42	0.049988	0.002499
iteration 43	0.032349	0.001046
iteration 44	0.020630	0.000426
iteration 45	0.041748	0.001743
iteration 46	0.018555	0.000344
iteration 47	0.109314	0.011950
iteration 48	0.098145	0.009632
iteration 49	0.060730	0.003688
iteration 50	0.039795	0.001584

Iteration	$\lvert p_{i+1} - p_i \rvert$	$(p_{i+1} - p_i)^2$
iteration 51	0.025269	0.000639
iteration 52	0.015503	0.000240
iteration 53	0.008484	0.000072
iteration 54	0.003235	0.000010
iteration 55	0.000122	0.000000
iteration 56	0.003113	0.000010
iteration 57	0.005066	0.000026
iteration 58	0.016296	0.000266
iteration 59	0.008850	0.000078
iteration 60	0.012573	0.000158
iteration 61	0.014099	0.000199
iteration 62	0.013611	0.000185
iteration 63	0.013855	0.000192
iteration 64	0.013977	0.000195
iteration 65	0.014221	0.000202
iteration 66	0.013855	0.000192
iteration 67	0.014099	0.000199
iteration 68	0.013977	0.000195
iteration 69	0.014343	0.000206
iteration 70	0.013916	0.000194
iteration 71	0.013916	0.000194
iteration 72	0.013672	0.000187
iteration 73	0.013672	0.000187
iteration 74	0.013611	0.000185
iteration 75	0.013306	0.000177
iteration 76	0.013000	0.000169
iteration 77	0.013123	0.000172
iteration 78	0.012695	0.000161
iteration 79	0.012695	0.000161
iteration 80	0.012451	0.000155
iteration 81	0.012207	0.000149
iteration 82	0.011780	0.000139
iteration 83	0.011902	0.000142
iteration 84	0.011719	0.000137
iteration 85	0.011475	0.000132
iteration 86	0.011230	0.000126
iteration 87	0.010986	0.000121
iteration 88	0.010803	0.000117
iteration 89	0.010803	0.000117
iteration 90	0.010376	0.000108
iteration 91	0.010315	0.000106
iteration 92	0.010132	0.000103
iteration 93	0.010010	0.000100
iteration 94	0.009583	0.000092
iteration 95	0.009827	0.000097
iteration 96	0.009338	0.000087
iteration 97	0.009338	0.000087
iteration 98	0.009155	0.000084
iteration 99	0.008789	0.000077
iteration 100	0.008850	0.000078

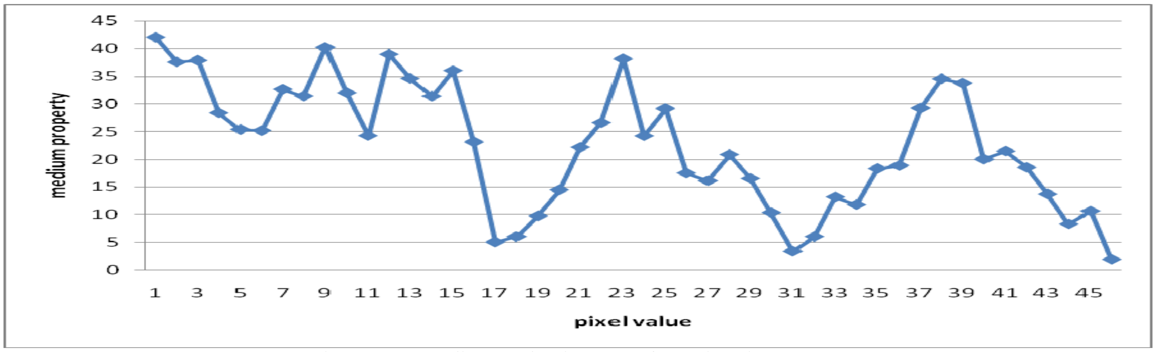

Fig 1. Image as a line Graph using ART after 01 iteration

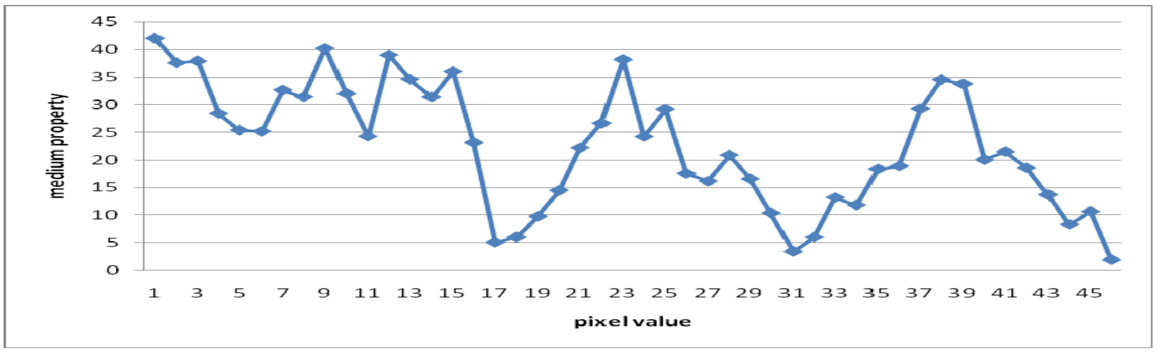

Fig 2. Image as a line Graph using ART after 15 iterations

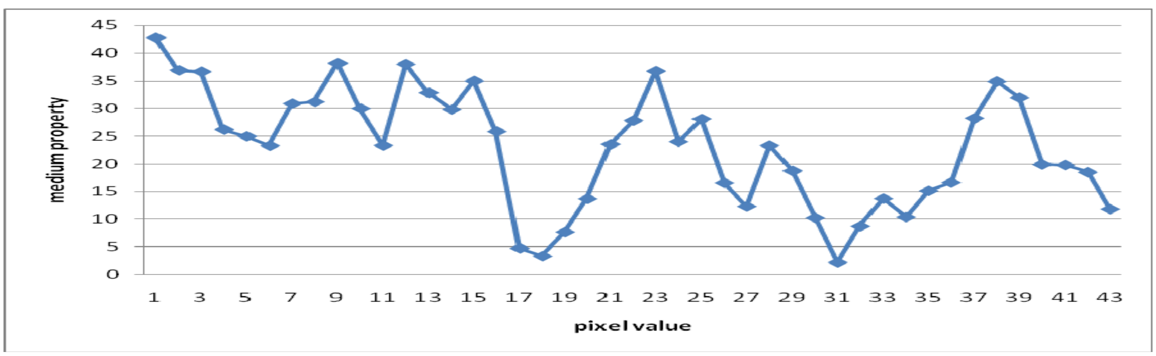

Fig 3. Image as a line Graph using ART after 25 iterations

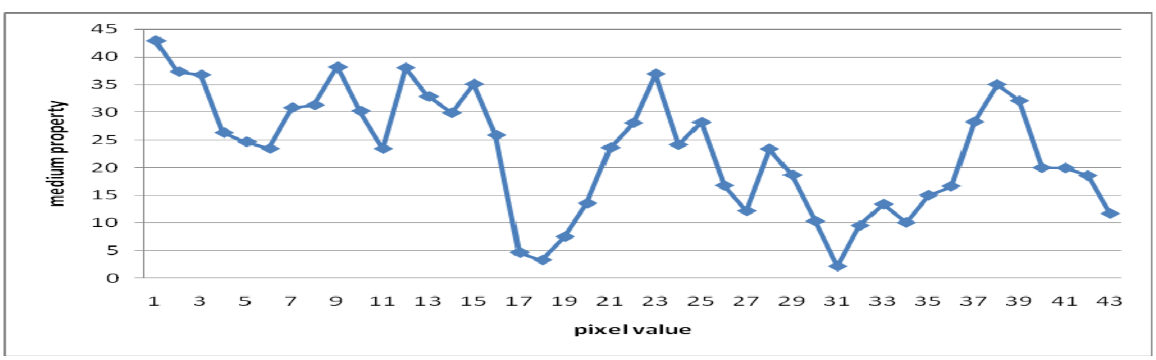

Fig 4. Image as a line Graph using ART after 55 iterations

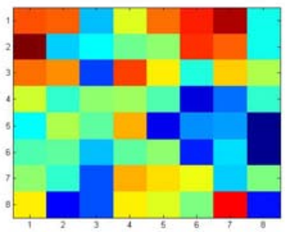

Image (a) after 1 iteration

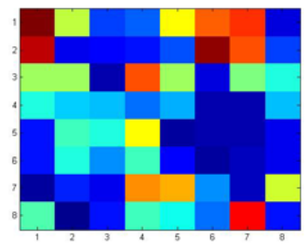

Image (b) after 10 iterations

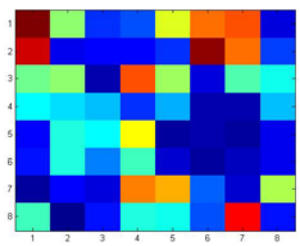

Image (c) after 20 iterations

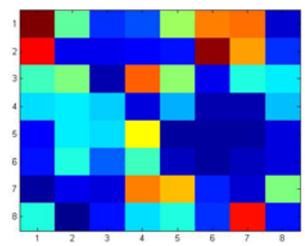

Image (d) after 30 iterations

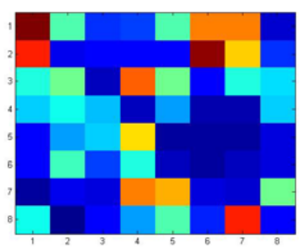

Image (e) after 50 iterations

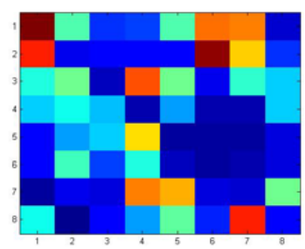

Image (f) after 55 iterations

Fig.5 Reconstruction at different iterations

CONCLUSION

The convergence is tested by difference in projection data at each iteration which is taken as $\sum_{i=1}^{M} |p_i - p_i^{(k)}|$ and $\sum_{i=1}^{M} (p_i - p_i^{(k)})^2$ at kth iteration (k = 1, 2, 3, ………). The stopping criterion taken as $\sum_{i=1}^{M} (p_i - p_i^{(k)})^2$ is small enough or stabilizes. The accuracy is also tested by another measure, image difference at successive iterations i.e. $\sum_{j=1}^{N} |f_j^{(k+1)} - f_j^{(k)}|$ and $\sum_{j=1}^{N} (f_j^{(k+1)} - f_j^{(k)})^2$. These results for our example are given in table 5 and table 4 respectively. We observe that after the difference in projection reaches at its minimum it again starts increasing, which says our stopping criterion should be guided by projection difference rather than a large number of iterations. In present example we reach at this minima in 55th iteration.

REFERENCES

[1] Seeram E. Computed tomography, physical principles – clinical applications, and quality control. 2nd edition, WB Saunders Co. 2000; 1-8.

[2] Shepp, L.A. and Kruskal, J.B., "Computerized tomography: The new medical x-ray technology", Am. Math. Monthly, 85, pp. 420-439, 1978.

[3] J. Friedhoff, Aufbereitung von 3D-Digitalisierdaten für den Werkzeug-, Formen-und Modellbau, Vulkan Verlag, Essen 1997.

[4] C. Kak, M. Slaney, *Principles of Computerized Tomography*, Society for Industrial and Applied Mathematics, 2001, pp49-60, 275-285

[5] Robert M. Lewitt (1983): Reconstruction Algorithms: Transform Methods, IEEE proceeding vol. 71, pp 390-408.

[6] R. Gordon *et al.*, "Three-Dimensional Reconstruction from Projections: A Review of Algorithms", International Review of Cytology, Vol. 38, p. 111 (1974).

[7] R. Gordon, R. Bender, and G. T. Herman, “Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography,” *J. Theoret. Biol.,* vol. 29, pp. 471-482, 1970.
[8] S. Kaczmarz, “Angentihrte Auflosung von Systemen linearer Gleichungen,” *Bull. Int. Acad. Pol. Sei. Lett., A,* vol. 35, pp. 355-357, 1937.
[9] Dan Gordan, “Parallel ART for image reconstruction in CT using processor arrays”, The International Journal of Parallel, Emergent and Distributed Systems, Vol. 21, No. 5, October 2006, 365–380.
[10] Tanuja Srivastava, Raghuveer Singh and Nirvikar, “ART for Image Reconstruction Using Parallel Beam Projection Data”, International Journal of Information Sciences and Applications, ISSN 0974-2255 Volume 2, Number 4 (2010), pp. 627-630.

AUTHORS PROFILE

NIRVIKAR
Research Scholar
School of CS & IT
Shobhit University, Meerut (UP), India

- Pursuing Ph.D. from School of Computer Engg. & IT, SHOBHIT UNIVERSITY, MEERUT on the topic of “Algebraic Reconstruction Technique for Computerized Tomography using Parallel Beam Projection Data”.