Proceedings of the Fifth Workshop on
Mathematical Modelling
of Environmental and Life Sciences Problems
Constanţa, Romania, September, 2006, pp. 67–78

Algebraic Reconstruction Technique versus Conjugate Gradient in Image Reconstruction from Projections

Tiberius Duluman[*] and **Constantin Popa**[**♮]

In this paper we consider three iterative algorithms for inconsistent linear least squares problems arising in image reconstruction from projections in Computerized Tomography: Kaczmarz Extended projection algorithm, conjugate gradient iteration and a hybrid method combining them. All these methods use in each iteration both rows and columns of the problem matrix. For this, we propose an efficient algorithm for constructing and storing it. Moreover, the implemented software allows graphical visualization and editing of the way that the scanning occurs, and making different reconstructions tests. Numerical experiments are presented for some real images from medical applications.

1. Image reconstruction from projections – scanning procedure and least squares formulation

The Image Reconstruction from Projections technique (IRP, for short) is essentially based on the research and results due to A. M. Cormack and G. Hounsfield in early 50's, which combined their efforts in the construction of the first computer tomograph (1972). And, although its first and main applications are related to the medical investigation context (see for details [3]), the IRP technique also found other important fields in which it was successfully applied (as e.g. geotomography). The first and very important step in the practical applications of the IRP technique is the "scanning" or "data acquisition" procedure. In medical Computerized Tomography (CT) investigations the most used scanning method is the "fan-beam" procedure. In

[*] Route 66, Braşov, Romania, e-mail: tiberius_d@hotmail.com
[**] "Ovidius" University of Constanţa, Romania, e-mail: cpopa@univ-ovidius.ro
[♮] the paper was supported by the Grant CEEX 05-D11-25/2005.

this case, as we can see from Figure 1 the X-rays source S moves on a circle around the scanned area ABCD, between positions $\alpha_{initial}$ and α_{final}. From each position $S_i, i = 1, \ldots, p$, of the source, a fan-beam of X-rays is emitted, their number being equal with the number of detectors $D_j, j = 1, \ldots, q$, on a screen that moves in the same time with S_i. The total number of X-rays, $M = pq$, will give the number of rows in the problem matrix A, whereas the resolution N (number of pixels) of the scanned area ABCD, gives the number of its columns. The construction of the matrix and right hand side coefficients, A_{ij} and b_i^0, respectively is sketched in Figure 2.

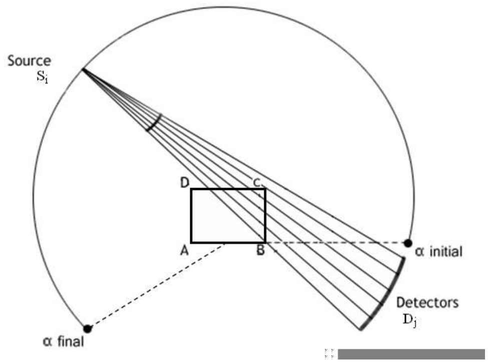


Fig. 1. Fan beam scanning.

There A_{ij} represents the length of the intersection between the i-th X-ray and the j-th pixel, whereas b_i^0 is a function of the intensities I_{imput} and I_{output} of the corresponding X-ray (see for details [3]) For a good reconstruction we need both, an accurate scanning of the region ABCD and a good resolution of it, which grows the dimensions M and N of A. The sparsity of A is determined by the fact that, e.g. in the case from Figure 2, if $N = n^2$, then we cannot have more than $2n-1$ nonzero elements on a row of A. Concerning the right hand side $b^0 = (b_1^0, \ldots, b_M^0) \in \mathbb{R}^M$, the measurement errors modify it by a perturbation term, $pert \in \mathbb{R}^M$ such that $b = b^0 + pert$ doesn't anymore belong to the range of A and instead of a consistent reconstruction system

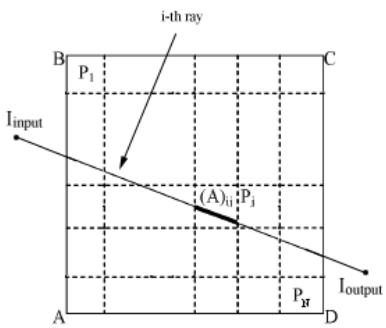


Fig. 2. Construction of A and b.

$$Ax = b^0 \tag{1}$$

we have to consider an inconsistent least squares formulation

$$\| Ax - b \| = \min! \tag{2}$$

If the perturbation vector *pert* is "not too big", then the minimal norm solution of (1), x_{LS} can be an enough good approximation of the corresponding one from (1), x_{LS}^0, whereas in the opposite case, we need to regularize (1) as

$$\| Ax - b \|^2 + \delta^2 \| Lx \|^2 = \min! \Longleftrightarrow \| \begin{bmatrix} A \\ \delta L \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \|^2 = \min! \tag{3}$$

with L an $N \times N$ matrix. In this case, the minimal norm solution of (2), $x_{LS}(\delta)$ will be an enough good approximation of x_{LS}^0, provided δ and L are well chosen (see for details [2]). But, in both the above formulations (1) or (2) the problem is inconsistent, thus appropriate solvers must be used. In this respect, in section 2 of the paper we shall present three such algorithms: Kaczmarz Extended with Relaxation Parameters (KERP), Conjugate Gradient for Normal Equation (CGNE) and a hybrid Kaczmarz Extended-Conjugate Gradient algorithm denoted by KECG. All of them use, not only the matrix A, but also its transpose A^{T}. This aspect is analyzed in section 3, in which an efficient algorithm for saving in a compressed form both matrices A and A^{mT}, during the row-by-row generation of A in the scanning process is presented. This procedure, together with the above mentioned iterative solvers were successfully implemented in an IRP software package, which is described in the second part of section 3. In the last section of the paper we present numerical experiments with this package on real 2D medical images.

2. The iterative solvers

Let A_i, A^j be the i-th row and j-th column of A, respectively.
KERP algorithm: let $x^0 \in \mathbb{R}^n$; $y^0 = b$; for $k = 0, 1, \ldots$ do

$$\begin{aligned} y^{k+1} &= (\varphi_1 \circ \ldots \circ \varphi_N)(\alpha; y^k), \\ b^{k+1} &= b - y^{k+1}, \\ x^{k+1} &= (f_1 \circ \ldots \circ f_M)(\omega; b^{k+1}; x^k), \end{aligned} \tag{4}$$

where

$$\varphi_j(\alpha; y) = y - \alpha \cdot \frac{\langle y, A^j \rangle}{\|A^j\|^2} A^j, \quad f_i(\omega; \beta; x) = x - \omega \cdot \frac{\langle x, A_i \rangle - \beta_i}{\|A_i\|^2} A_i. \tag{5}$$

Theorem 1. ([6]) *If $A_i \neq 0, A^j \neq 0, i = 1, \ldots, M, j = 1, \ldots, N$, for $x^0 = 0$ and any $\alpha, \omega \in (0, 2)$ the sequence $(x^k)_{k \geq 0}$ generated with the KERP algorithm converges to the minimal norm solution of a least squares problem as* (1).

CGNE algorithm: let $x^0 \in \mathbb{R}^N, r^0 = b - Ax^0$; for $k = 0, 1, \ldots$ do
$k = k + 1$
if $k = 1$
$p_1 = A^{\mathrm{T}} r^0$
else

$\beta_k = (A^T r^{k-1})^T (A^T r^{k-1}) / (A^T r^{k-2})^T (A^T r^{k-2})$
$p^k = A^T r^{k-1} + \beta_k p^{k-1}$
endif
$\alpha_k = (A^T r^{k-1})^T (A^T r^{k-1}) / (Ap^k)^T (Ap^k)$
$x^k = x^{k-1} + \alpha_k p^k$
$r^k = r^{k-1} - \alpha_k A p^k$

Theorem 2. ([4]) *If* $x^0 = 0$ *then, the sequence* $(x^k)_{k\geq 0}$ *generated with the CGNE algorithm converges to the minimal norm solution of a least squares problem as* (1).

One iteration of the algorithm CGNE will be denoted by $x^{k+1} = CGNE(A; b; x^k)$, $\forall k \geq 0$.
KECG algorithm: let $x^0 \in R^N, y^0 = b, R^0 = -A^T y^0$; for $k = 0, 1, \ldots$ do

$$\begin{aligned} y^{k+1} &= CGNE(A^T; 0; y^k), & (6)\\ b^{k+1} &= b - y^{k+1}, & (7)\\ x^{k+1} &= (f_1 \circ \ldots \circ f_M)(\omega; b^{k+1}; x^k). & (8) \end{aligned}$$

Note. The above step (6) means the CGNE algorithm applied to the (consistent) problem

$$A^T y = 0 \Longleftrightarrow AA^T y = 0, \tag{9}$$

with $y^0 = b$.

Theorem 3. ([7]) *If* $A_i \neq 0, i = 1, \ldots, M$, *for* $x^0 = 0$ *and any* $\omega \in (0, 2)$ *the sequence* $(x^k)_{k\geq 0}$ *generated with the KECG algorithm converges to the minimal norm solution of a least squares problem as* (1).

3. Computational software package. Storage of A and A^T

The software is an application written in C++ programming language. It contains the implementation of several algorithms in order to create and store A and A^T matrix. It also contains implementation of the image reconstruction algorithms and of the regularization matrixes. The access to this algorithms is made through a graphical interface written with GTK+ and GTKMM library. It allows loading of images from BMP files, converting them to gray scale with values between 0 and 255. It contains a free implementation of Mersenne-Twister pseudorandom number generator that is used when applying perturbation.

3.1. Algorithms

Storage of sparse matrixes. The A matrix is a sparse matrix, every row may contain up two $w + h - 1$ non-zero entries (where $N = w \times h$ is the resolution of the discretisation). As every row stands for the length of the segments obtained by the intersection of X-ray with the grid, and all reconstructions algorithms use rows of A, the best method to store this matrix in a compressed row storage. This means that for every row only non-zero elements are stored with an additional information about their column.

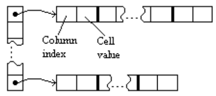


Fig. 3. Compressed row storage.

Construction of A. Construction of A is made by rows. Every ray is intersected first with vertical lines of the grid and the array of intersection points is stored. Then the intersections with horizontal lines are computed. The points in that two arrays are ordered by the distance to the ray source. So a merging of them is possible to obtained all the intersections points between the ray and the grid. Actually, not the array with intersection points is needed, but the length of the segments. So merging algorithm is applied, but instead of keeping the merged points, the length of the segments and the corresponding pixels are saved. It can be easily seen that the algorithm has a linear complexity ($\mathcal{O}((w + h)M)$).

One observation is that a ray may cross a grid through a pixel corner. In this case the intersection point will appear also in horizontal intersections array and also in vertical intersections array. In merging phase, only one of this point is used, the other one being ignored. Another observation is that there can be intersections points between the ray and horizontal lines or vertical lines that do not belong to the grid. This points are eliminated during the intersections computing.

Construction of A^{T}. Construction of A^{T} is tied to the construction of A. During the construction of A, we may easily count how many rays intersects a pixel. Knowing this in advance, give us information about how many non-zero elements every row of A^{T} contains. This allow a more efficient way to allocate memory for storing of A^{T}. The algorithm of transposing the matrix has an $\mathcal{O}((w + h)M)$ complexity. Even if M is very large, computing A^{T} is fast because the matrix is very sparse. Moreover, A^{T} is computed only one time, then it is stored. The above algorithms first allocate the necessary amount of memory needed to store the whole matrix. Then the arrays of pointers to the beginning of the row is initialized because we already know how many elements each row of A^{T} has. Then for every row and for every element in the row, the value is copied in A^{T}.

3.2. Software design and implementation

Below is the class diagram of the software. All iterative solvers are inherited from the abstract class Algorithm. They implement Iterate method and they use an abstract Matrix class which offers only GetRow methods. Some algorithms, like Kaczmarz Extended use TransposeMatrix class. Storage of a matrix is made by MatrixCRS class. MatrixCRS_Compact class is used when the memory block where the matrix is stored is compact (this is the case of transpose matrix). TransposeMatrix groups together a matrix and its transpose. Construction of every matrix row is handled through the implementation of the abstract method CreateRow of ScanningProcedure. It uses methods from Geometry class to perform get the intersections. RegularizationMatrix represent a matrix used for regularization. RegularizedMatrix groups a Matrix and a RegularizationMatrix to provide to the algorithms in a transparent way, a regularized matrix.

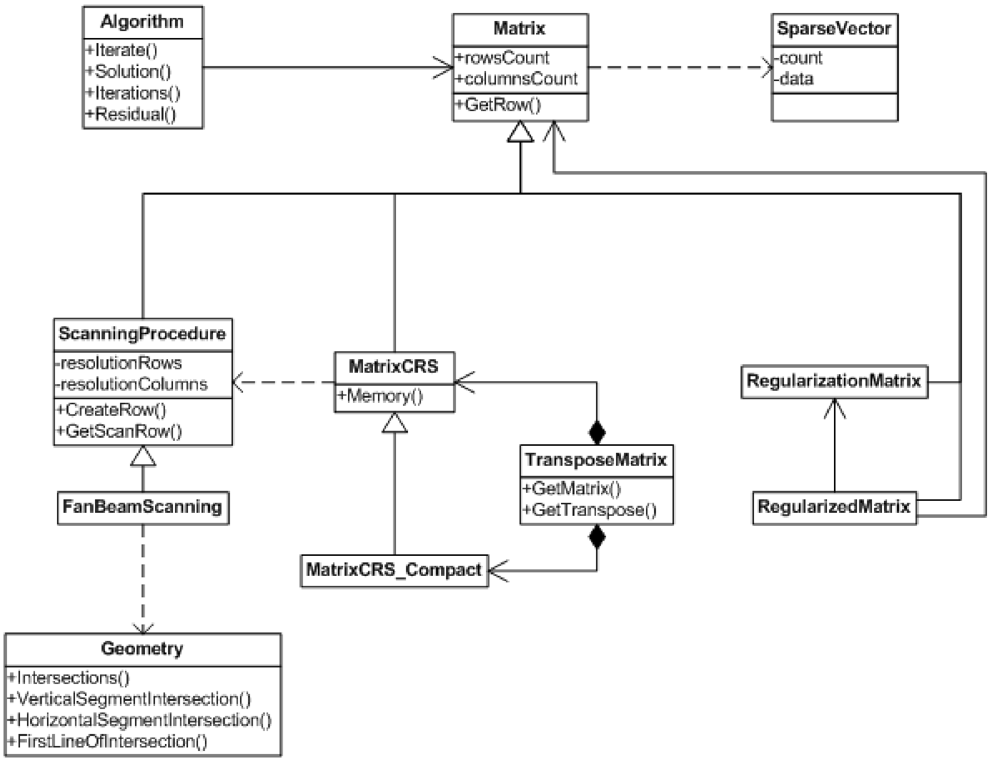


Fig. 4. Class diagram.

4. Numerical experiments

For the 256×256 resolution exact images in figures 5 and 6, we constructed the scanning matrix A as in figures 1 and 2, with 210 sources, 512 detectors, $\alpha_{initial} = 0^o, \alpha_{final} = 210^o$ and a fan-beam angle of 30^o. This gave us the dimensions of A as $M = 210 \times 512 = 107520$ and $N = 256 \times 256 = 65536$. Then, for each of the two exact images, x^{ex} from figures 5 and 6 we constructed a consistent system of the form

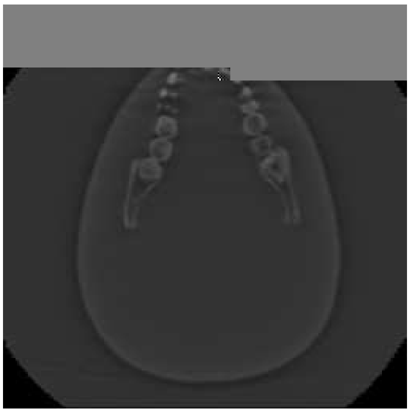


Fig. 5. Original image O1.

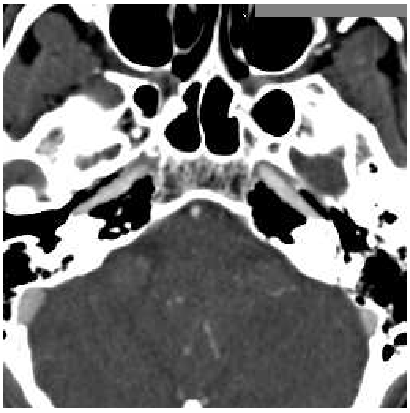


Fig. 6. Original image O2.

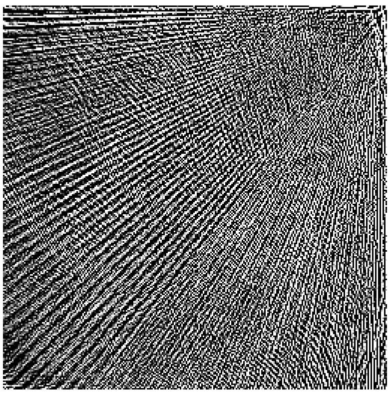


Fig. 7. Kaczmarz for O1.

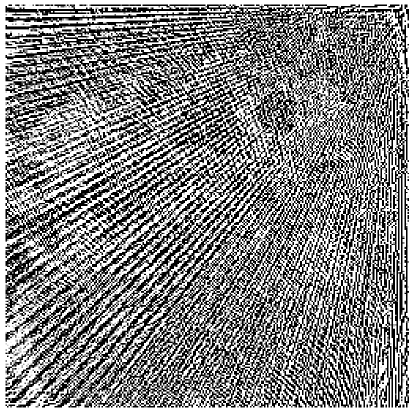


Fig. 8. Kaczmarz for O2.

(1), with b^0 defined as $b^0 = Ax^{ex}$. Then b^0 was perturbed as (see section 1)

$$b = b^0 + pert, \quad pert = 5\% \cdot \parallel b^0 \parallel \cdot rand, \tag{10}$$

with $rand$ a randomly generated vector with unitary norm. The inconsistent least squares problem (1) was solved by applying 20 iterations with the classical Kaczmarz's algorithm (see [1], [6]) and the three methods from section 2. The results for classical Kaczmarz are presented in figures 7 and 8, whereas for the other three algorithms in table 2. We can see that the classical Kaczmarz algorithm doesn't give anymore a good approximation for x^0_{LS}. Then, we regularized the problem (1) as in (2), with the matrix L constructed in the following two different ways. First, we used $L = I$, where I is the identity matrix $N \times N$. Second, we used L constructed as follows: for each $i \in 1 \ldots n$, let H_i be the set of horizontally neighbour pixels of i, V_i the set of vertically neighbour pixels and D_i the set of diagonally neighbour pixels of i.

Table 1. Regularization using $L = I$ and $\alpha = 30$

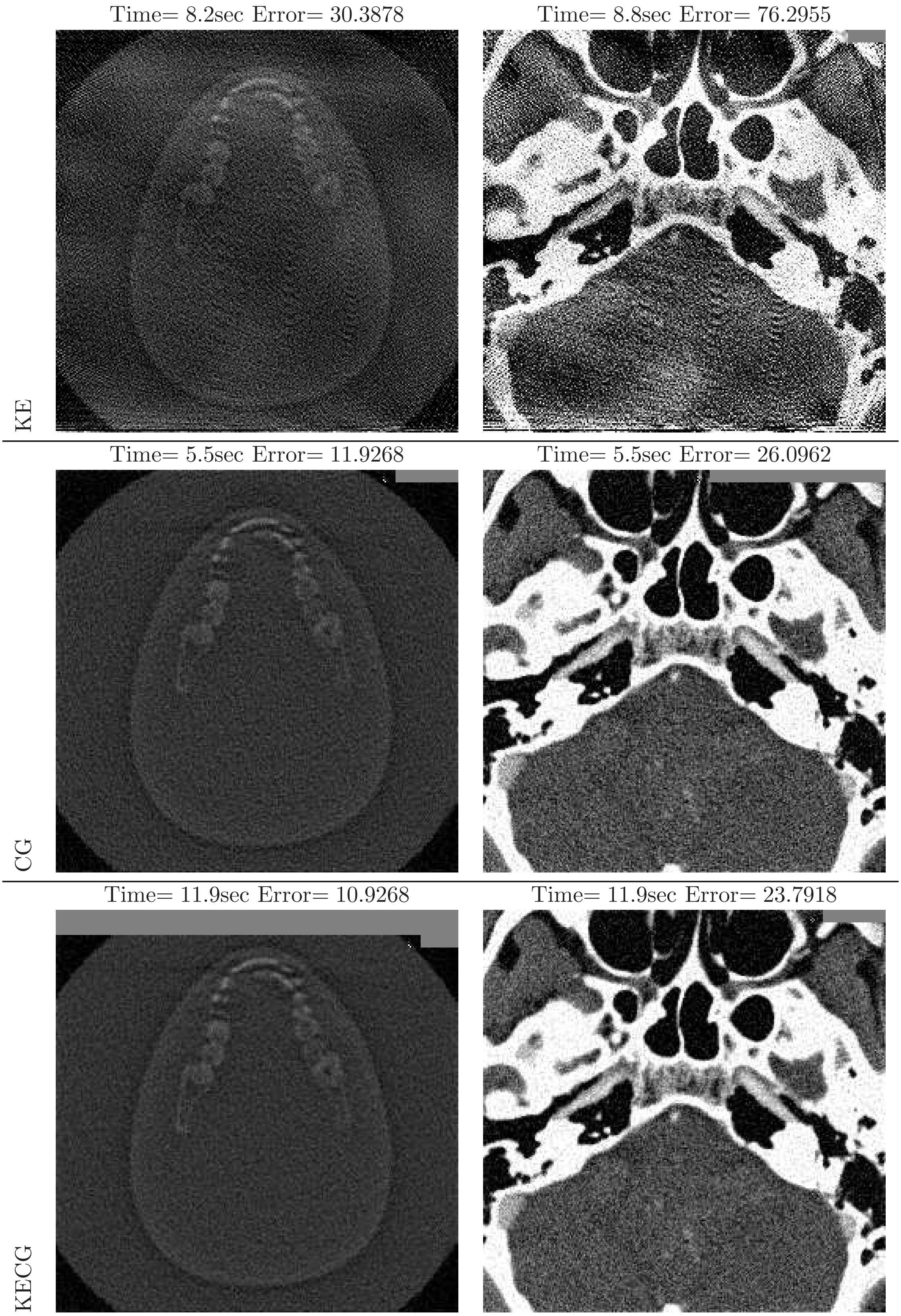


Table 2. No regularization

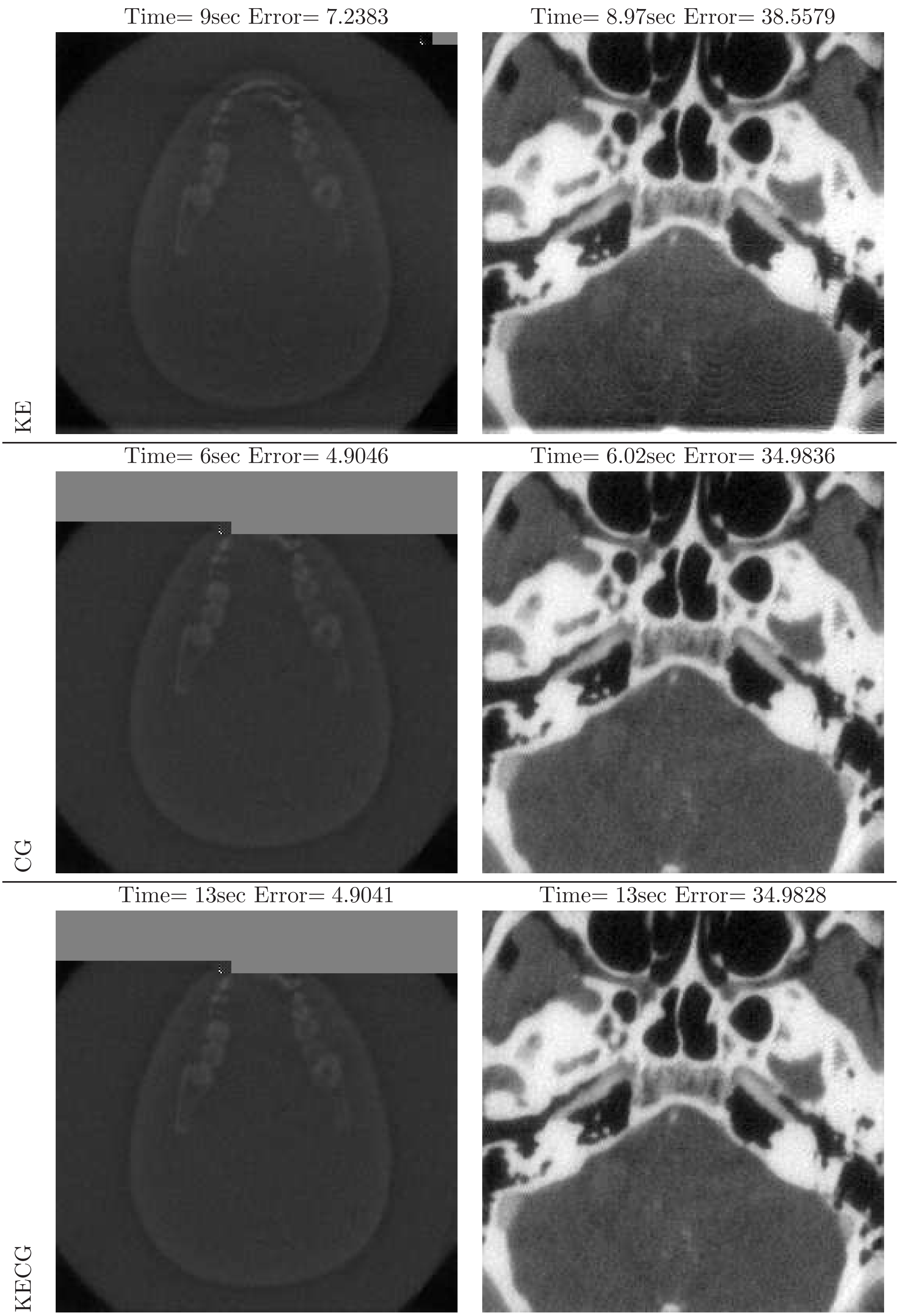
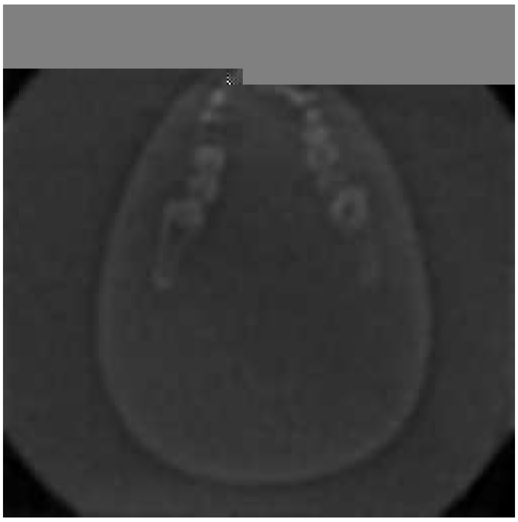

Time= 9sec Error= 7.2383
Time= 8.97sec Error= 38.5579
KE
Time= 6sec Error= 4.9046
Time= 6.02sec Error= 34.9836
CG
Time= 13sec Error= 4.9041
Time= 13sec Error= 34.9828
KECG

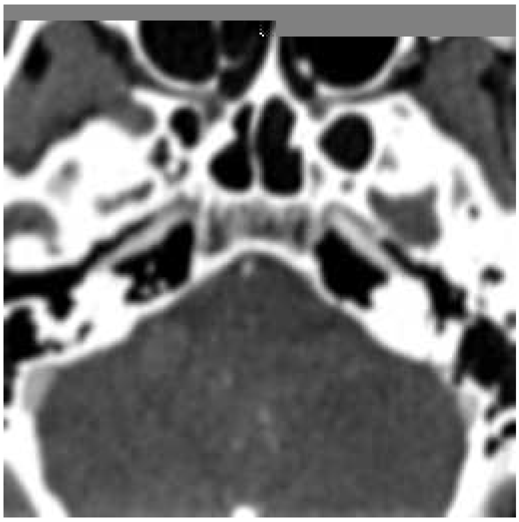
Table 3. Regularization using L from (11) and $\alpha = 20$

KE

Time= 9.5sec Error= 4.6128

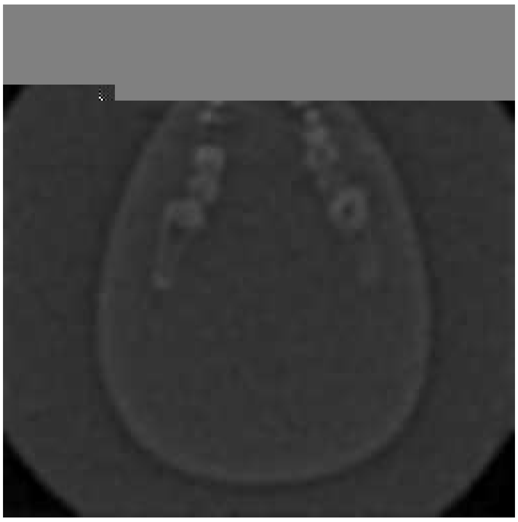


Time= 9.5sec Error= 26.6141

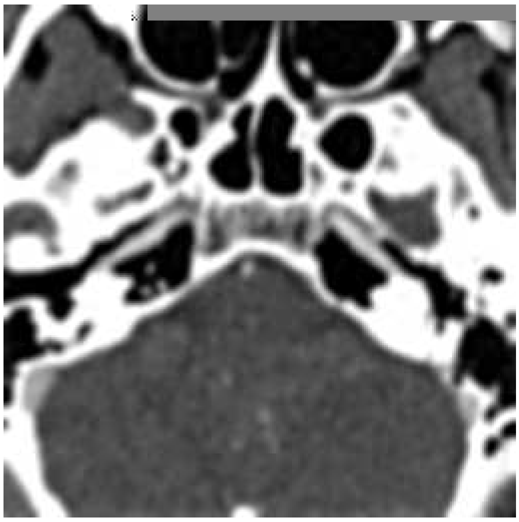


CG

Time= 6.4sec Error= 3.8161

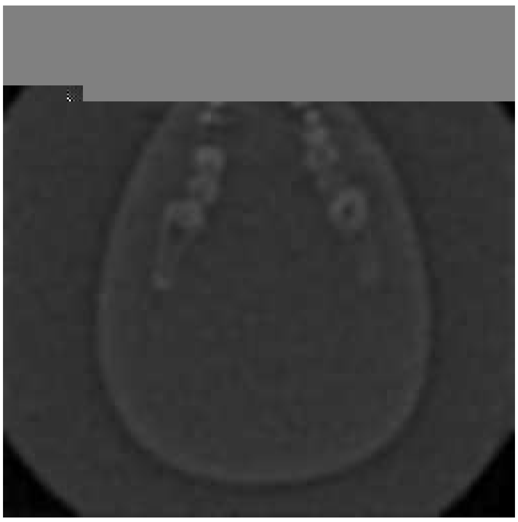


Time= 6.4sec Error= 23.7096

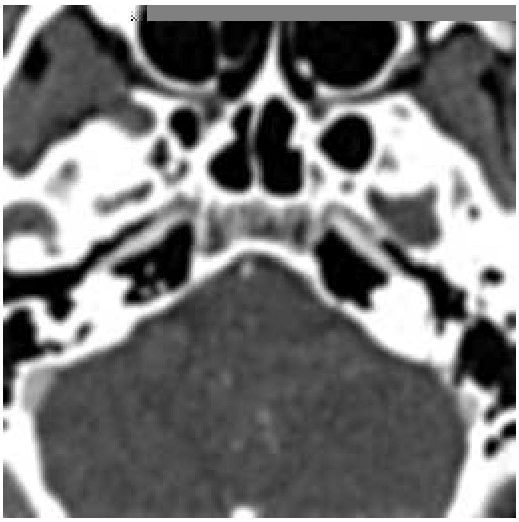


KECG

Time= 13.7sec Error= 3.8097



Time= 13.7sec Error= 23.8274



For each $j \in 1 \dots n$

$$(L)_{ij} = \begin{cases} (L)_{ij} = w_h, & \text{if } j \in H_i \\ (L)_{ij} = w_v, & \text{if } j \in V_i \\ (L)_{ij} = w_d, & \text{if } j \in D_i \\ \sum_{k=1}^{n} |(L)_{ik}|, & \text{if } j = i \text{ and } k \neq i \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

In our experiments we used $w_h = -1, w_v = -1, w_d = -1/\sqrt{2}$. The results are presented in tables 2 and 3.

Acknowledgements. We thank to prof. dr. Ulrich Rüde, dr. Harald Köstler and dr. Marcus Prümmer from the University Erlangen-Nürnberg, Germany for providing us the real medical images used in the numerical experiments.

References

[1] Björck, A., *Numerical methods for least squares problems*, SIAM Philadelphia, 1996.

[2] Engl, H. W., *Regularization methods for the stable solution of inverse problems*, Surv. Math. Ind. **3** (1993), pp. 71–143.

[3] Herman, G., *Image reconstruction from projections. The fundamentals of computerized tomography*, Academic Press, New York, 1980.

[4] Kammerer W. J., Nashed M. Z., *On the convergence of the conjugate gradient method for singular linear operator equations*, SIAM J. Numer. Anal., **9** (1972), pp.165–181.

[5] Popa C., *Extensions of block-projections methods with relaxation parameters to inconsistent and rank-defficient least-squares problems*; *BIT*, **38(1)** (1998), pp. 151–176.

[6] Popa, C., *A mixed Kaczmarz-Conjugate Gradient algorithm for general least squares problems with applications in image reconstruction from projections in Computerized Tomography*; manuscript in preparation.