

# Imágenes Médicas - Práctica 4: Reconstrucción de Imágenes Tomográficas: Métodos Iterativos

JUAN PABLO MORALES<sup>1a</sup>

<sup>a</sup>*Instituto Balseiro - Centro Atómico Bariloche. Av Bustillo 9500, 8400 Bariloche, Río Negro, Argentina*

25 DE MARZO DE 2024

## 1. Introducción

En el proceso de reconstrucción de imágenes tomográficas, se busca reconstruir una imagen del objeto a partir de proyecciones. Entre los métodos de reconstrucción, los métodos iterativos son utilizados debido a su capacidad para producir imágenes de alta calidad y reducir artefactos de reconstrucción.

Los métodos iterativos como su nombre indica se basan en iterativamente para mejorar una estimación inicial de la imagen mediante la minimización de una función que compara las proyecciones medidas con las proyecciones reconstruidas a partir de la estimación actual de la imagen. Este proceso se repite hasta que se alcanza una convergencia satisfactoria.

## 2. El método ART

El Método de Reconstrucción Algebraica (ART) es una técnica ampliamente reconocida en el campo de la tomografía computarizada. Este método se utiliza para reconstruir imágenes a partir de proyecciones medidas. La idea principal detrás del ART es iterar sobre una estimación inicial de la imagen para mejorarla gradualmente hasta obtener una reconstrucción aceptable.

En el proceso de reconstrucción mediante ART, se realizan las siguientes etapas:

1. **Estimación inicial:** Se comienza con una estimación inicial de la imagen que se desea reconstruir.
2. **Medición de proyecciones:** Se miden las proyecciones de la imagen desde múltiples ángulos.
3. **Iteración:** Se lleva a cabo un proceso iterativo para mejorar la estimación de la ima-

gen. La ecuación fundamental que define este proceso es:

$$f_j^{(k+1)} = f_j^{(k)} + \frac{p_i - \sum_{j=1}^N f_{ji}^{(k)}}{N} \quad (1)$$

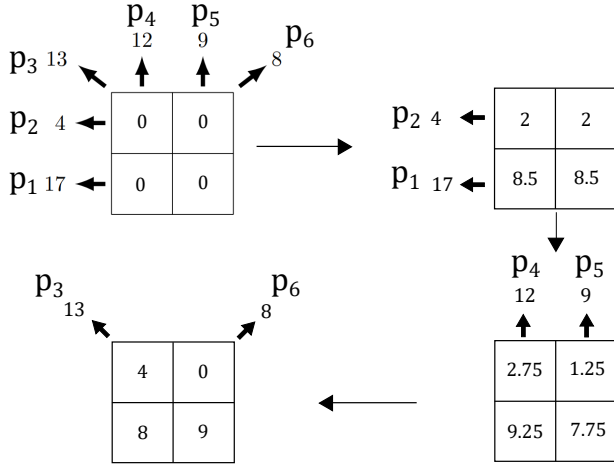
Donde:

- $f_j^{(k)}$  es la estimación actual de la intensidad de píxel en la posición  $j$  en la iteración  $k$ .
- $f_j^{(k+1)}$  es la estimación propuesta para la intensidad de píxel en la siguiente iteración.
- $p_i$  es la proyección medida a lo largo de la dirección  $i$ .
- $N$  es el número total de píxeles en la dirección considerada.
- $\sum_{j=1}^N f_{ji}^{(k)}$  es la suma de las intensidades de píxel a lo largo de la dirección  $i$  en la iteración  $k$ .

Teniendo en cuenta el algoritmo ART, se realizó una reconstrucción de una imagen de tamaño  $2 \times 2$  con un total de 6 mediciones  $p_i$ . Esta reconstrucción se llevó a cabo en 3 pasos, como se muestra en la Figura 1. Inicialmente, se tomaron las mediciones en la dirección horizontal ( $p_1$  y  $p_2$ ). Partiendo de la suposición inicial de que  $f_j^{(0)} = 0 \forall j$ , se aplicó la ecuación (1) para realizar el primer paso de la reconstrucción.

Posteriormente, se llevaron a cabo mediciones en la dirección vertical ( $p_4$  y  $p_5$ ), continuando desde el paso anterior. Finalmente, se realizaron mediciones en la dirección diagonal ( $p_3$  y  $p_6$ ). Se observó que la reconstrucción convergió y la imagen resultante satisface las mediciones en todas las direcciones.

<sup>1</sup>juan.morales@ib.edu.ar



**Figura 1:** Pasos de reconstrucción por el método ART de una imagen de 2x2 con 6 mediciones en las direcciones horizontal, diagonal y vertical.

### 3. Dependencias de las reconstrucciones

Se llevó a cabo un análisis de la influencia del número de ángulos, detectores y nivel de ruido en las reconstrucciones realizadas mediante el método de reconstrucción iterativa ART utilizando MATLAB y los archivos de AIRTools. En primer lugar, se generó un fantoma Shepp-Logan (ver Figura 2) con un tamaño de  $N \times N$  píxeles, donde  $N = 256$ , junto con su correspondiente sinograma. Posteriormente, se realizaron las reconstrucciones utilizando MATLAB, y mediante Python, se calcularon los errores de reconstrucción. En este caso, se utilizaron las siguientes métricas:

- Mean Squared Error (MSE):

$$MSE = \frac{1}{n} \sum_{i=1}^n (x_i - y_i)^2$$

- Peak Signal to Noise Ratio (PSNR):

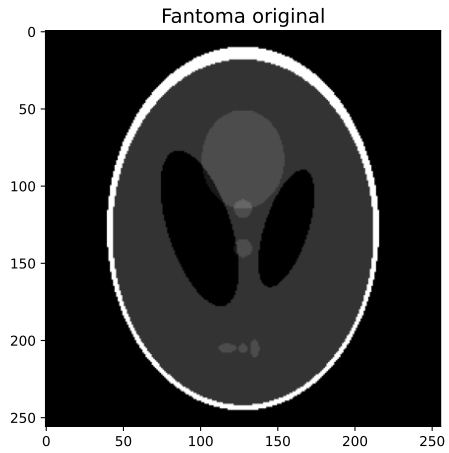
$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX^2}{MSE} \right)$$

, donde MAX es el máximo valor posible de la imagen. En este caso 255.

- Structural Similarity Index (SSIM):

$$SSIM = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

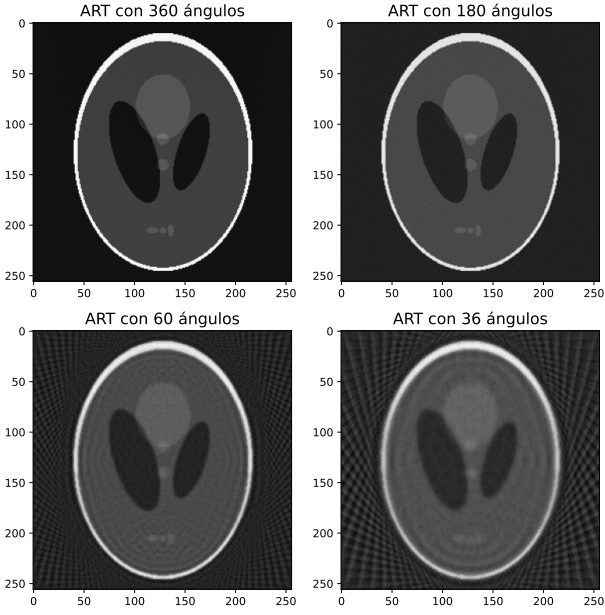
, donde  $\mu_x$  es la media muestral de píxeles de  $x$ ;  $\mu_y$  es la media muestral de píxeles de  $y$ ;  $\sigma_x^2$  es la varianza de  $x$ ;  $\sigma_y^2$  es la varianza de  $y$ ;  $\sigma_{xy}$  es la covarianza de  $x$  e  $y$ ;  $c_1 = (k_1L)^2$ ,  $c_2 = (k_2L)^2$  son dos variables para estabilizar la división con denominador débil;  $L$  es el rango dinámico de los valores de píxeles (típicamente esto es  $2^{\#bits \text{ per pixel}} - 1$ );  $k_1 = 0,01$  y  $k_2 = 0,03$  por defecto.



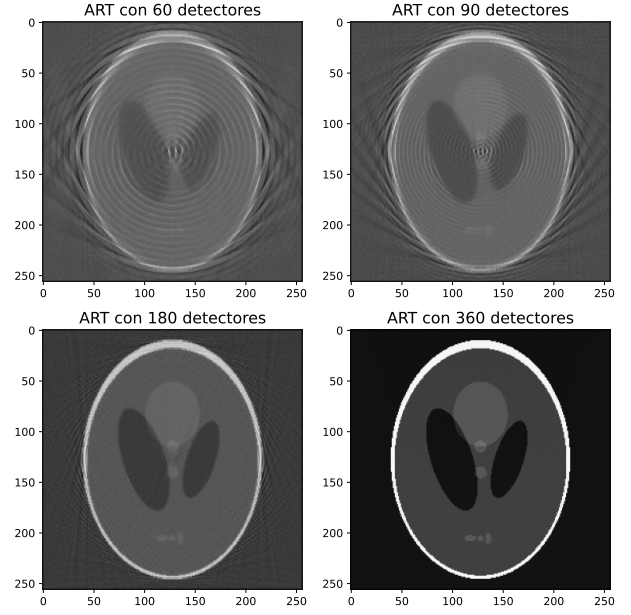
**Figura 2:** Fantoma original shepp-logan utilizado.

En primer lugar, se fijaron el número de detectores en  $p = 360$ , el número de iteraciones en  $k = 100$ , y el error relativo en  $\eta = 1 \times 10^{-4}$ . Este error relativo se define como el error en las mediciones comparado con  $\|p_i\|$ . Luego, se varió la cantidad de ángulos, tomando valores  $\theta \in [0, 180]$  y variando el número de ángulos en 36, 60, 180 y 360. Los resultados obtenidos para las distintas reconstrucciones se muestran en la Figura 3. En esta figura, se puede observar que a medida que se aumenta el número de ángulos medidos, se obtiene una mejor calidad en la reconstrucción de la imagen.

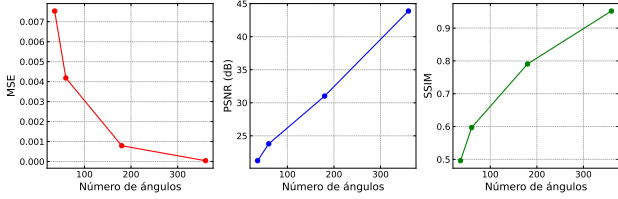
Para tener una medida correcta del error de reconstrucción, se utilizaron las métricas mencionadas anteriormente. Estas métricas se muestran en la Figura 4 en función de la cantidad de ángulos utilizados.



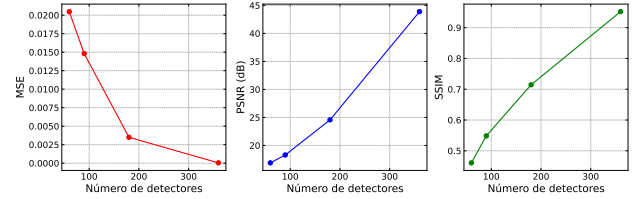
**Figura 3:** Reconstrucciones del fantoma shepp-logan mediante el método ART variando el número de ángulos utilizados.



**Figura 5:** Reconstrucciones del fantoma shepp-logan mediante el método ART variando el número de detectores utilizados.



**Figura 4:** Medidas MSE, PSNR y SSIM del error de reconstrucción del fantoma shepp-logan mediante el método ART variando el número de ángulos utilizados.



**Figura 6:** Medidas MSE, PSNE, SSIM del error de reconstrucción del fantoma shepp-logan mediante el método ART variando el número de detectores utilizados.

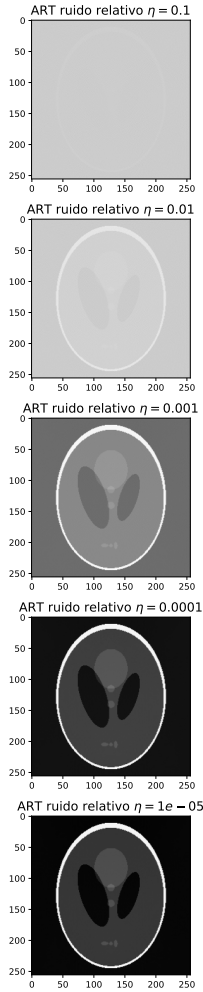
Para el análisis del número de detectores, se varió el número tomando  $p = \{60, 90, 180, 360\}$ . Se mantuvieron los parámetros del caso anterior: el número de iteraciones  $k = 100$  y el error relativo  $\eta = 1 \times 10^{-4}$ , y se fijó el número de ángulos  $\theta \in [0, 180]$  en 360.

Los resultados se muestran en la Figura 5. En esta figura, se puede observar que a medida que se aumenta el número de detectores, se obtiene una mejor reconstrucción, y que su efecto es similar al que ocurre con el número de ángulos, incluyendo artefactos en la imagen reconstruida. Esto se evidencia también en las métricas mostradas en la Figura 6.

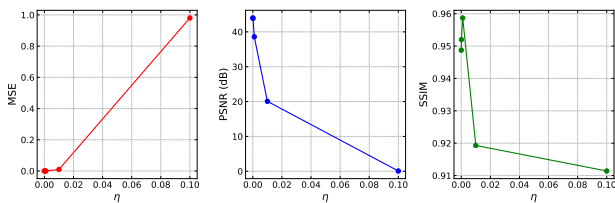
Además, se realizó el mismo estudio variando el nivel de intensidad del ruido. Se dejó el número de detectores en  $p = 360$ , el número de iteraciones en  $k = 100$ , y el número de ángulos  $\theta \in [0, 180]$  en 360. Los valores utilizados para variar el ruido fueron los del ruido relativo  $\eta = \{0,1, 0,01, 0,001, 0,0001, 0,00001\}$ .

Los resultados se muestran en la Figura 7. Se puede observar que, si bien el ruido no añade artefactos a la reconstrucción, sí cambia la información que se puede obtener de las imágenes, ya que se pierden detalles al aumentar el ruido relativo. Al igual que en los casos anteriores, esto se evidencia en las métricas del error de reconstrucción mostradas en la Figura 7, donde a medida que aumenta el ruido, las métricas empeoran. Sin embargo, para valores como 0,0001 y 0,00001, la

evaluación del error de reconstrucción es similar.



**Figura 7:** Reconstrucciones del fantoma shepp-logan mediante el método ART variando el nivel de ruido relativo.



**Figura 8:** Medidas MSE, PSNE, SSIM del error de reconstrucción del fantoma shepp-logan mediante el método ART variando el nivel de ruido relativo  $\eta$ .

## 4. Análisis para distintos métodos

Se realizó el mismo análisis con respecto a las variables que en el caso anterior, pero esta vez para tres métodos alternativos de ART:

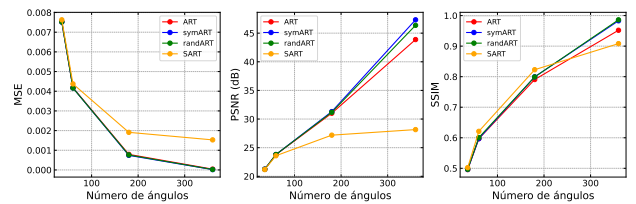
- **Simétrico:** Se realiza una barrida en un sentido seguida de una segunda en el sentido opuesto.
- **Aleatorio:** Se elige el índice  $i$  para las mediciones de manera aleatoria.
- **Simultáneo (SART):** Se actualizan todos los índices simultáneamente usando la siguiente fórmula:

$$f^{(k+1)} = f^{(k)} + \mu T A^T M(p - A f^{(k)}), \quad (2)$$

donde  $T = V^{-1}$  y  $M = W^{-1}$ , con  $V$  y  $W$  siendo matrices diagonales cuyos elementos son las sumas de filas y columnas de  $A$ .

Se utilizó el mismo fantoma Shepp-Logan de tamaño  $256 \times 256$  y los mismos valores utilizados en la exploración de ART anteriormente.

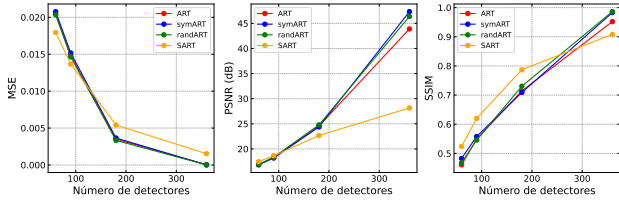
En el análisis de variar la cantidad de ángulos utilizados, se obtuvieron los resultados mostrados en la Figura 9 para las métricas anteriormente analizadas. Se puede observar que, mientras que los métodos ART, ART simétrico y ART aleatorio dan resultados similares, el método simultáneo llega rápidamente a un valor en el cual no mejora la reconstrucción, y según las métricas, esta reconstrucción es peor que la de los otros métodos.



**Figura 9:** Medidas MSE, PSNE, SSIM del error de reconstrucción del fantoma shepp-logan mediante los métodos de reconstrucción ART, ART simétrico, ART aleatorio y SART variando el número de ángulos.

El análisis para el número de detectores se repite y se obtienen los resultados mostrados en la

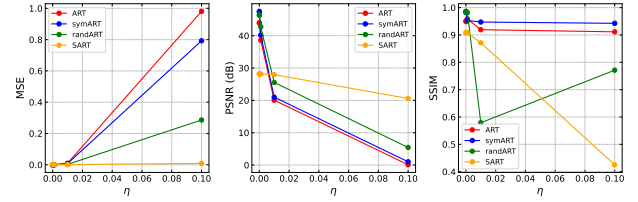
Figura 10. En este caso, se observa un comportamiento similar al anterior, en el cual el método que obtiene los peores resultados es el método SART. Sin embargo, la diferencia entre este y los otros métodos es menor que en el caso de variar el número de ángulos, por lo tanto, afecta menos a esta reconstrucción.



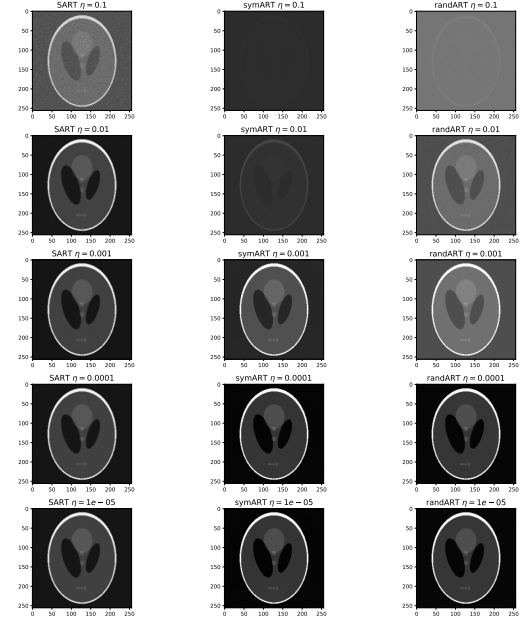
**Figura 10:** Medidas MSE, PSNE, SSIM del error de reconstrucción del fantoma shepp-logan mediante los métodos de reconstrucción ART, ART simétrico, ART aleatorio y SART variando el número de detectores.

El análisis del error de reconstrucción, variando el ruido relativo, se muestra en la Figura 11. Este caso resulta interesante debido a que, según los errores de reconstrucción dados por el MSE y el PSNR, el método SART muestra un mejor rendimiento, seguido por el método ART aleatorio, y luego por los métodos ART simétrico y ART, en contraposición a los análisis realizados anteriormente. Sin embargo, se observa que el método SART produce resultados peores que los otros métodos cuando el ruido relativo es aproximadamente menor a  $\eta = 0,01$ .

Además, es importante destacar que la métrica SSIM no refleja esta última observación, lo que sugiere que se pierden correlaciones con el método SSIM. Sin embargo, esto no implica necesariamente un rendimiento inferior en presencia de ruido (ver Figura 12).



**Figura 11:** Medidas MSE, PSNE, SSIM del error de reconstrucción del fantoma shepp-logan mediante los métodos de reconstrucción ART, ART simétrico, ART aleatorio y SART variando el valor del ruido relativo.



**Figura 12:** Reconstrucciones del fantoma shepp-logan mediante los métodos de reconstrucción ART, ART simétrico, ART aleatorio y SART variando el valor del ruido relativo.

## 5. Tiempos de cálculo

Para comparar los tiempos de cálculo, se realizó la medición del tiempo de ejecución de cada uno de estos métodos en MATLAB, y se compararon también con el método de retroproyección filtrada en Python. Estas mediciones se llevaron a cabo utilizando el mismo conjunto de parámetros: 360 detectores, 100 iteraciones para los métodos iterativos, sin ruido y 360 ángulos entre 0 y 180 grados. Los resultados obtenidos se muestran en la tabla 1.

Se observa que los métodos iterativos, en promedio, requieren mucho más tiempo en comparación con el método directo de la retroproyección filtrada. Esto se debe a que los métodos iterativos implican un mayor esfuerzo computacional, ya que requieren múltiples iteraciones para converger hacia la solución.

Método	Segundos
ART	590
ART Simetrico	1170
ART Aleatorio	598
SART	5.8
Retroproyección filtrada	0.2

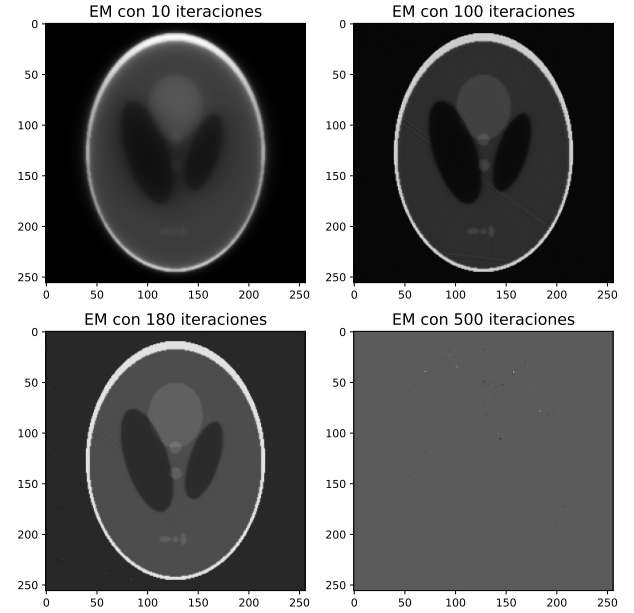
**Tabla 1:** Tabla con los métodos de reconstrucción y el tiempo en segundos que llevó realizarlo.

## 6. Expectation Maximization

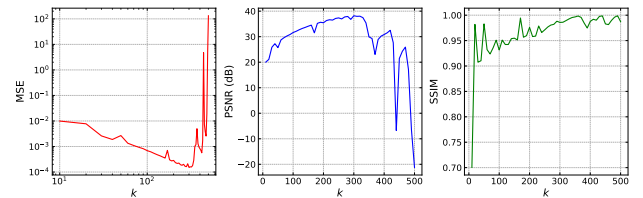
El método de Expectation Maximization (EM) se basa en la reconstrucción de la función  $\hat{f}$  maximizando la probabilidad de obtener las mediciones  $\vec{p}$  dado  $\vec{f}$ . Para lograr esto, se busca una cota inferior de esta probabilidad que pueda ser maximizada mediante iteraciones. De esta manera, se obtiene la reconstrucción de  $\vec{f}$  deseada.

Se llevó a cabo un análisis de la dependencia del error de reconstrucción utilizando el algoritmo EM en relación con el número de iteraciones realizadas. Este error de reconstrucción se evaluó utilizando las mismas métricas utilizadas anteriormente. Algunos resultados de las reconstrucciones se muestran en la Figura 13. Se observa que, al aumentar las iteraciones, la calidad de la reconstrucción mejora; sin embargo, llega un punto en el que aumentar el número de iteraciones empeora la calidad de la reconstrucción. Este efecto se puede observar en la Figura 14, donde se mues-

tran las métricas en función del número de iteraciones realizadas. Esto se observa claramente para las métricas MSE y PSNE, sin embargo, el error de reconstrucción calculado con la métrica SSIM no logra captar este suceso.



**Figura 13:** Reconstrucciones del fantoma shepp-logan mediante el método de reconstrucción expectation maximization variando el número de iteraciones realizado.



**Figura 14:** Medidas MSE, PSNE, SSIM del error de reconstrucción del fantoma shepp-logan mediante el método de reconstrucción expectation maximization variando el número de repeticiones realizado.



## 7. Código

```
1 # # Ejercicio 2
2
3 # %%
4 import os
5 import numpy as np
6 import matplotlib.pyplot as plt
7 from skimage.transform import radon, iradon
8 from scipy.io import loadmat
9 from scipy.stats import poisson
10 from skimage.metrics import mean_squared_error as mse
11 from skimage.metrics import peak_signal_noise_ratio as psnr
12 from skimage.metrics import structural_similarity as ssim
13 import cv2
14
15 # %%
16 path = r"C:/Users/Propietario/Desktop/ib/5-Maestría/Imágenes Médicas/
    Practicas/Practica4/AIRtools/"
17
18 # %%
19 files_angulos = ["ART_angles0.5.mat", "ART_angles1.mat", "ART_angles3.
    mat", "ART_angles5.mat"]
20 num_angulos = [360, 180, 60, 36]
21 fantoma_path = os.path.join(path, "fantoma256.mat")
22 fantoma_original = loadmat(fantoma_path)['fantoma']
23 plt.imshow(fantoma_original, cmap='gray')
24 plt.title('Fantoma original', fontsize='x-large')
25 plt.tight_layout()
26 plt.savefig('fantoma_original.pdf')
27 plt.show()
28
29 # %%
30 fig, ax = plt.subplots(2,2,figsize=(8, 8))
31 for i, (n, file) in enumerate(zip(num_angulos, files_angulos)):
32     path_actual = os.path.join(path, file)
33     reconstruccion_fantoma = loadmat(path_actual)['ART']
34     if i == 0 or i == 1:
35         ax[0,i].imshow(reconstruccion_fantoma, cmap='gray')
36         ax[0,i].set_title('ART con {} ángulos'.format(n), fontsize='x-
            large')
37     else:
38         ax[1,i%2].imshow(reconstruccion_fantoma, cmap='gray')
39         ax[1,i%2].set_title('ART con {} ángulos'.format(n), fontsize='x
            -large')
40 plt.tight_layout()
41 plt.savefig('ART_angulos.pdf')
42 plt.show()
43
44 # %%
45 mse_a = []
46 psnr_a = []
```

```

47 ssim_a = []
48 for n, file in zip(num_angulos, files_angulos):
49     path_actual = os.path.join(path, file)
50     reconstruccion_fantoma = loadmat(path_actual)['ART']
51     mse_a.append(mse(fantoma_original, reconstruccion_fantoma))
52     psnr_a.append(psnr(fantoma_original, reconstruccion_fantoma))
53     ssim_a.append(ssim(fantoma_original, reconstruccion_fantoma,
        data_range=reconstruccion_fantoma.max()-reconstruccion_fantoma.
        min()))
54
55
56 # %%
57 files_detectores = ["ART_detectorsv260.mat", "ART_detectorsv290.mat", "
    ART_detectorsv2180.mat", "ART_detectorsv2360.mat"]
58 num_detectores = [60, 90, 180, 360]
59 fig, ax = plt.subplots(2,2,figsize=(8, 8))
60 for i, (n, file) in enumerate(zip(num_detectores, files_detectores)):
61     path_actual = os.path.join(path, file)
62     reconstruccion_fantoma = loadmat(path_actual)['ART']
63     if i == 0 or i == 1:
64         ax[0,i].imshow(reconstruccion_fantoma, cmap='gray')
65         ax[0,i].set_title('ART con {} detectores'.format(n), fontsize='
            x-large')
66     else:
67         ax[1,i%2].imshow(reconstruccion_fantoma, cmap='gray')
68         ax[1,i%2].set_title('ART con {} detectores'.format(n), fontsize
            ='x-large')
69 plt.tight_layout()
70 plt.savefig('ART_detectores.pdf')
71 plt.show()
72
73 # %%
74 mse_detectors = []
75 psnr_detectors = []
76 ssim_detectors = []
77 for n, file in zip(num_detectores, files_detectores):
78     path_actual = os.path.join(path, file)
79     reconstruccion_fantoma = loadmat(path_actual)['ART']
80     mse_detectors.append(mse(fantoma_original, reconstruccion_fantoma))
81     psnr_detectors.append(psnr(fantoma_original, reconstruccion_fantoma
        ))
82     ssim_detectors.append(ssim(fantoma_original, reconstruccion_fantoma
        , data_range=reconstruccion_fantoma.max()-reconstruccion_fantoma
        .min()))
83
84
85 # %%
86 files_ruido = ["ART_rnoise0.1.mat", "ART_rnoise0.01.mat", "ART_rnoise0
    .001.mat", "ART_rnoise0.0001.mat", "ART_rnoise1e-05.mat"]
87 eta_list = [0.1, 0.01, 0.001, 1e-4, 1e-5]
88 fig, ax = plt.subplots(5,1,figsize=(10, 16))
89 for i, (eta, file) in enumerate(zip(eta_list, files_ruido)):

```



```

90     path_actual = os.path.join(path, file)
91     reconstruccion_fantoma = loadmat(path_actual)['ART']
92     ax[i].imshow(reconstruccion_fantoma, cmap='gray')
93     ax[i].set_title('ART ruido relativo $\eta = {}$'.format(eta),
94                     fontsize='x-large')
95 plt.tight_layout()
96 plt.savefig('ART_rnoise.pdf')
97 plt.show()
98 # %%
99 mse_noise = []
100 psnr_noise = []
101 ssim_noise = []
102 for n, file in zip(eta_list, files_ruido):
103     path_actual = os.path.join(path, file)
104     reconstruccion_fantoma = loadmat(path_actual)['ART']
105     mse_noise.append(mse(fantoma_original, reconstruccion_fantoma))
106     psnr_noise.append(psnr(fantoma_original, reconstruccion_fantoma))
107     ssim_noise.append(ssim(fantoma_original, reconstruccion_fantoma,
108                           data_range=reconstruccion_fantoma.max()-reconstruccion_fantoma.
109                               min()))
110 # %%
111 files_angulos_sym = ["symART_angles0.5.mat", "symART_angles1.mat", "
112                     symART_angles3.mat", "symART_angles5.mat"]
113 files_angulos_rand = ["randART_angles0.5.mat", "randART_angles1.mat", "
114                       randART_angles3.mat", "randART_angles5.mat"]
115 files_angulos_sart = ["SART_angles0.5.mat", "SART_angles1.mat", "
116                       SART_angles3.mat", "SART_angles5.mat"]
117 # path_test = os.path.join(path, "symART_angles0.5.mat")
118 # print(loadmat(path_test).keys())
119 # %%
120 mse_a = []
121 mse_sym = []
122 mse_rand = []
123 mse_sart = []
124 psnr_a = []
125 psnr_sym = []
126 psnr_rand = []
127 psnr_sart = []
128 ssim_a = []
129 ssim_sym = []
130 ssim_rand = []
131 ssim_sart = []
132 # %%
133 for file_art, file_sym, file_rand, file_sart in zip(files_angulos,
134                                                     files_angulos_sym, files_angulos_rand, files_angulos_sart):
135     path_actual = os.path.join(path, file_art)
136     path_sym = os.path.join(path, file_sym)
137     path_rand = os.path.join(path, file_rand)

```

```

135     path_sart = os.path.join(path, file_sart)
136
137     reconstruccion_art = loadmat(path_actual)['ART']
138     reconstruccion_sym = loadmat(path_sym)['symART']
139     reconstruccion_rand = loadmat(path_rand)['randART']
140     reconstruccion_sart = loadmat(path_sart)['SART']
141
142     mse_a.append(mse(fantoma_original, reconstruccion_art))
143     psnr_a.append(psnr(fantoma_original, reconstruccion_art))
144     ssim_a.append(ssim(fantoma_original, reconstruccion_art, data_range
        =reconstruccion_art.max()-reconstruccion_art.min()))
145
146     mse_sym.append(mse(fantoma_original, reconstruccion_sym))
147     psnr_sym.append(psnr(fantoma_original, reconstruccion_sym))
148     ssim_sym.append(ssim(fantoma_original, reconstruccion_sym,
        data_range=reconstruccion_sym.max()-reconstruccion_sym.min()))
149
150     mse_rand.append(mse(fantoma_original, reconstruccion_rand))
151     psnr_rand.append(psnr(fantoma_original, reconstruccion_rand))
152     ssim_rand.append(ssim(fantoma_original, reconstruccion_rand,
        data_range=reconstruccion_rand.max()-reconstruccion_rand.min()))
153
154     mse_sart.append(mse(fantoma_original, reconstruccion_sart))
155     psnr_sart.append(psnr(fantoma_original, reconstruccion_sart))
156     ssim_sart.append(ssim(fantoma_original, reconstruccion_sart,
        data_range=reconstruccion_sart.max()-reconstruccion_sart.min()))
157
158 # %% [markdown]
159 # ### Variando los detectores
160
161 # %%
162 files_detectores = ["ART_detectorsv260.mat", "ART_detectorsv290.mat", "
    ART_detectorsv2180.mat", "ART_detectorsv2360.mat"]
163 files_detectores_sym = ["symART_detectorsv260.mat", "
    symART_detectorsv290.mat", "symART_detectorsv2180.mat", "
    symART_detectorsv2360.mat"]
164 files_detectores_rand = ["randART_detectorsv260.mat", "
    randART_detectorsv290.mat", "randART_detectorsv2180.mat", "
    randART_detectorsv2360.mat"]
165 files_detectores_sart = ["SART_detectorsv260.mat", "SART_detectorsv290.
    mat", "SART_detectorsv2180.mat", "SART_detectorsv2360.mat"]
166 num_detectores = [60, 90, 180, 360]
167
168 # %%
169 mse_a = []
170 mse_sym = []
171 mse_rand = []
172 mse_sart = []
173 psnr_a = []
174 psnr_sym = []
175 psnr_rand = []
176 psnr_sart = []

```

```

177 ssim_a = []
178 ssim_sym = []
179 ssim_rand = []
180 ssim_sart = []
181
182 for file_art, file_sym, file_rand, file_sart in zip( files_detectores,
    files_detectores_sym, files_detectores_rand, files_detectores_sart):
183     path_actual = os.path.join(path, file_art)
184     path_sym = os.path.join(path, file_sym)
185     path_rand = os.path.join(path, file_rand)
186     path_sart = os.path.join(path, file_sart)
187
188     reconstruccion_art = loadmat(path_actual)['ART']
189     reconstruccion_sym = loadmat(path_sym)['symART']
190     reconstruccion_rand = loadmat(path_rand)['randART']
191     reconstruccion_sart = loadmat(path_sart)['SART']
192
193     mse_a.append(mse(fantoma_original, reconstruccion_art))
194     psnr_a.append(psnr(fantoma_original, reconstruccion_art))
195     ssim_a.append(ssim(fantoma_original, reconstruccion_art, data_range
        =reconstruccion_art.max()-reconstruccion_art.min()))
196
197     mse_sym.append(mse(fantoma_original, reconstruccion_sym))
198     psnr_sym.append(psnr(fantoma_original, reconstruccion_sym))
199     ssim_sym.append(ssim(fantoma_original, reconstruccion_sym,
        data_range=reconstruccion_sym.max()-reconstruccion_sym.min()))
200
201     mse_rand.append(mse(fantoma_original, reconstruccion_rand))
202     psnr_rand.append(psnr(fantoma_original, reconstruccion_rand))
203     ssim_rand.append(ssim(fantoma_original, reconstruccion_rand,
        data_range=reconstruccion_rand.max()-reconstruccion_rand.min()))
204
205     mse_sart.append(mse(fantoma_original, reconstruccion_sart))
206     psnr_sart.append(psnr(fantoma_original, reconstruccion_sart))
207     ssim_sart.append(ssim(fantoma_original, reconstruccion_sart,
        data_range=reconstruccion_sart.max()-reconstruccion_sart.min()))
208
209
210 # %%
211 files_ruido = ["ART_rnoise0.1.mat", "ART_rnoise0.01.mat", "ART_rnoise0
    .001.mat", "ART_rnoise0.0001.mat", "ART_rnoise1e-05.mat"]
212 files_ruido_sym = ["symART_rnoise0.1.mat", "symART_rnoise0.01.mat", "
    symART_rnoise0.001.mat", "symART_rnoise0.0001.mat", "symART_rnoise1e
    -05.mat"]
213 files_ruido_rand = ["randART_rnoise0.1.mat", "randART_rnoise0.01.mat",
    "randART_rnoise0.001.mat", "randART_rnoise0.0001.mat", "
    randART_rnoise1e-05.mat"]
214 files_ruido_sart = ["SART_rnoise0.1.mat", "SART_rnoise0.01.mat", "
    SART_rnoise0.001.mat", "SART_rnoise0.0001.mat", "SART_rnoise1e-05.
    mat"]
215 eta_list = [0.1, 0.01, 0.001, 1e-4, 1e-5]
216

```

```

217 # %%
218 mse_a = []
219 mse_sym = []
220 mse_rand = []
221 mse_sart = []
222 psnr_a = []
223 psnr_sym = []
224 psnr_rand = []
225 psnr_sart = []
226 ssim_a = []
227 ssim_sym = []
228 ssim_rand = []
229 ssim_sart = []
230
231 for file_art, file_sym, file_rand, file_sart in zip(files_ruido,
    files_ruido_sym, files_ruido_rand, files_ruido_sart):
232     path_actual = os.path.join(path, file_art)
233     path_sym = os.path.join(path, file_sym)
234     path_rand = os.path.join(path, file_rand)
235     path_sart = os.path.join(path, file_sart)
236
237     reconstruccion_art = loadmat(path_actual)['ART']
238     reconstruccion_sym = loadmat(path_sym)['symART']
239     reconstruccion_rand = loadmat(path_rand)['randART']
240     reconstruccion_sart = loadmat(path_sart)['SART']
241
242     mse_a.append(mse(fantoma_original, reconstruccion_art))
243     psnr_a.append(psnr(fantoma_original, reconstruccion_art))
244     ssim_a.append(ssim(fantoma_original, reconstruccion_art, data_range
        =reconstruccion_art.max()-reconstruccion_art.min()))
245
246     mse_sym.append(mse(fantoma_original, reconstruccion_sym))
247     psnr_sym.append(psnr(fantoma_original, reconstruccion_sym))
248     ssim_sym.append(ssim(fantoma_original, reconstruccion_sym,
        data_range=reconstruccion_sym.max()-reconstruccion_sym.min()))
249
250     mse_rand.append(mse(fantoma_original, reconstruccion_rand))
251     psnr_rand.append(psnr(fantoma_original, reconstruccion_rand))
252     ssim_rand.append(ssim(fantoma_original, reconstruccion_rand,
        data_range=reconstruccion_rand.max()-reconstruccion_rand.min()))
253
254     mse_sart.append(mse(fantoma_original, reconstruccion_sart))
255     psnr_sart.append(psnr(fantoma_original, reconstruccion_sart))
256     ssim_sart.append(ssim(fantoma_original, reconstruccion_sart,
        data_range=reconstruccion_sart.max()-reconstruccion_sart.min()))
257
258 # %%
259 eta_list = [0.1, 0.01, 0.001, 1e-4, 1e-5]
260 fig, ax = plt.subplots(5,3,figsize=(16, 16))
261 for i, (eta, sart, sym, rand) in enumerate(zip(eta_list,
    files_ruido_sart, files_ruido_sym, files_ruido_rand)):
262     path_sart = os.path.join(path, sart)

```

```

263 path_sym = os.path.join(path, sym)
264 path_rand = os.path.join(path, rand)
265 reconstruccion_sart = loadmat(path_sart)['SART']
266 reconstruccion_sym = loadmat(path_sym)['symART']
267 reconstruccion_rand = loadmat(path_rand)['randART']
268 ax[i,0].imshow(reconstruccion_sart, cmap='gray')
269 ax[i,0].set_title('SART $\eta = {}$'.format(eta), fontsize='x-large
    ')
270 ax[i,1].imshow(reconstruccion_sym, cmap='gray')
271 ax[i,1].set_title('symART $\eta = {}$'.format(eta), fontsize='x-
    large')
272 ax[i,2].imshow(reconstruccion_rand, cmap='gray')
273 ax[i,2].set_title('randART $\eta = {}$'.format(eta), fontsize='x-
    large')
274 plt.tight_layout()
275 plt.savefig('noise.pdf')
276 plt.show()
277
278 # %% [markdown]
279 # ## Expectation maximization
280
281 # %%
282 files_em_show = ["EM_k10.mat", "EM_k100.mat", "EM_k180.mat", "EM_k500.
    mat"]
283
284 num_k = list(np.arange(10,510,10))
285 files_em = []
286 for k in num_k:
287     files_em.append("EM_k{}.mat".format(k))
288 num_k_show = [10,100,180,500]
289
290 # %%
291 fig, ax = plt.subplots(2,2,figsize=(8, 8))
292 for i, (k, file) in enumerate(zip(num_k_show, files_em_show)):
293     path_actual = os.path.join(path, file)
294     reconstruccion_fantoma = loadmat(path_actual)['EMr']
295     if i == 0 or i == 1:
296         ax[0,i].imshow(reconstruccion_fantoma, cmap='gray')
297         ax[0,i].set_title('EM con {} iteraciones'.format(k), fontsize='
            x-large')
298     else:
299         ax[1,i%2].imshow(reconstruccion_fantoma, cmap='gray')
300         ax[1,i%2].set_title('EM con {} iteraciones'.format(k), fontsize
            ='x-large')
301 plt.tight_layout()
302 plt.savefig('EM_iteraciones.pdf')
303 plt.show()
304
305 # %%
306 mse_em = []
307 psnr_em = []
308 ssim_em = []

```

```

309 for n, file in zip(num_k, files_em):
310     path_actual = os.path.join(path, file)
311     reconstruccion_fantoma = loadmat(path_actual)['EMr']
312     mse_em.append(mse(fantoma_original, reconstruccion_fantoma))
313     psnr_em.append(psnr(fantoma_original, reconstruccion_fantoma))
314     ssim_em.append(ssim(fantoma_original, reconstruccion_fantoma,
        data_range=reconstruccion_fantoma.max()-reconstruccion_fantoma.
        min()))
315
316 theta = np.linspace(0., 180., 360, endpoint=False)
317 sinogram = radon(img_eq, theta=theta)
318 %timeit iradon(sinogram, theta=theta, filter_name='ramp')
319
320 #
    #####

321 MATLAB
322
323 % Ejercicio 2 - ART, Simetric ART, Random ART, SART
324 close all
325
326 % Set the parameters for the test problem.
327
328 for step = [5, 3, 1, 0.5]
329     N = 256;                % The discretization points.
330     p = 360;                % No. of parallel rays.
331     eta = 0.0001;           % Relative noise level.
332     k = 100;                % No. of iterations.
333     theta = 0:step:179;     % No. of used angles.
334     % Create the test problem.
335     [A,b_ex,x_ex] = paralleltomo(N,theta,p);
336     % Noise level.
337     delta = eta*norm(b_ex);
338     % Add noise to the rhs.
339     randn('state',0);
340     e = randn(size(b_ex));
341     e = delta*e/norm(e);
342     b = b_ex + e;
343     % Show the exact solution.
344     figure
345     imagesc(reshape(x_ex,N,N)), colormap gray,
346     axis image off
347     c = caxis;
348     title('Exact phantom')
349     fantoma = reshape(x_ex,N,N);
350     %fantomaname = ['fantoma' num2str(step) '.mat'];
351     %save(fantomaname, 'fantoma'); % Save the matrix to a .mat file
352     % Perform the kaczmarz iterations.
353     Xkacz = kaczmarz(A,b,k);
354     % Show the kaczmarz solution.
355     figure
356     imagesc(reshape(Xkacz,N,N)), colormap gray,

```



```

357 axis image off
358 caxis(c);
359 title('Kaczmarz reconstruction')
360 ART = reshape(Xkacz,N,N);
361 ARTname = ['ART_angles' num2str(step) '.mat'];
362 save(ARTname, 'ART'); % Save the matrix to a .mat file
363 % Perform the symmetric kaczmarz iterations.
364 Xsymk = symkaczmarz(A,b,k);
365 % Show the symmetric kaczmarz solution.
366 figure
367 imagesc(reshape(Xsymk,N,N)), colormap gray,
368 axis image off
369 caxis(c);
370 title('Symmetric Kaczmarz reconstruction')
371 symART = reshape(Xsymk,N,N);
372 symARTname = ['symART_angles' num2str(step) '.mat'];
373 save(symARTname, 'symART'); % Save the matrix to a .mat file
374 % Perform the randomized kaczmarz iterations.
375 Xrand = randkaczmarz(A,b,k);
376 % Show the randomized kaczmarz solution.
377 figure
378 imagesc(reshape(Xrand,N,N)), colormap gray,
379 axis image off
380 caxis(c);
381 title('Randomized Kaczmarz reconstruction')
382 randART = reshape(Xrand,N,N);
383 randARTname = ['randART_angles' num2str(step) '.mat'];
384 save(randARTname, 'randART'); % Save the matrix to a .mat file
385 % Perform SART
386 Xsart = sart(A,b,k);
387 % Show the SART solution.
388 figure
389 imagesc(reshape(Xsart,N,N)), colormap gray,
390 axis image off
391 caxis(c);
392 title('SART reconstruction')
393 SART = reshape(Xsart,N,N);
394 SARTname = ['SART_angles' num2str(step) '.mat'];
395 save(SARTname, 'SART');
396 end
397
398 for p = [30, 60, 90, 180, 360]
399     N = 256; % The discretization points
400     %p = 360; % No. of parallel rays.
401     eta = 0.0001; % Relative noise level.
402     k = 100; % No. of iterations.
403     theta = 0:0.5:179; % No. of used angles.
404     % Create the test problem.
405     [A,b_ex,x_ex] = paralleltomo(N,theta,p);
406     % Noise level.
407     delta = eta*norm(b_ex);
408     % Add noise to the rhs.

```

```

409     randn('state',0);
410     e = randn(size(b_ex));
411     e = delta*e/norm(e);
412     b = b_ex + e;
413     % Show the exact solution.
414     figure
415     imagesc(reshape(x_ex,N,N)), colormap gray,
416     axis image off
417     c = caxis;
418     title('Exact phantom')
419     fantoma = reshape(x_ex,N,N);
420     %fantomaname = ['fantoma' num2str(step) '.mat'];
421     %save(fantomaname, 'fantoma'); % Save the matrix to a .mat file
422     % Perform the kaczmarz iterations.
423     Xkacz = kaczmarz(A,b,k);
424     % Show the kaczmarz solution.
425     figure
426     imagesc(reshape(Xkacz,N,N)), colormap gray,
427     axis image off
428     caxis(c);
429     title('Kaczmarz reconstruction')
430     ART = reshape(Xkacz,N,N);
431     ARTname = ['ART_detectorsv2' num2str(p) '.mat'];
432     save(ARTname, 'ART'); % Save the matrix to a .mat file
433     % Perform the symmetric kaczmarz iterations.
434     Xsymk = symkaczmarz(A,b,k);
435     % Show the symmetric kaczmarz solution.
436     figure
437     imagesc(reshape(Xsymk,N,N)), colormap gray,
438     axis image off
439     caxis(c);
440     title('Symmetric Kaczmarz reconstruction')
441     symART = reshape(Xsymk,N,N);
442     symARTname = ['symART_detectorsv2' num2str(p) '.mat'];
443     save(symARTname, 'symART'); % Save the matrix to a .mat file
444     % Perform the randomized kaczmarz iterations.
445     Xrand = randkaczmarz(A,b,k);
446     % Show the randomized kaczmarz solution.
447     figure
448     imagesc(reshape(Xrand,N,N)), colormap gray,
449     axis image off
450     caxis(c);
451     title('Randomized Kaczmarz reconstruction')
452     randART = reshape(Xrand,N,N);
453     randARTname = ['randART_detectorsv2' num2str(p) '.mat'];
454     save(randARTname, 'randART'); % Save the matrix to a .mat file
455     % Perform SART
456     Xsart = sart(A,b,k);
457     % Show the SART solution.
458     figure
459     imagesc(reshape(Xsart,N,N)), colormap gray,
460     axis image off

```

```

461     caxis(c);
462     title('SART reconstruction')
463     SART = reshape(Xsart,N,N);
464     SARTname = ['SART_detectorsv2' num2str(p) '.mat'];
465     save(SARTname, 'SART');
466 end
467
468 for eta = [0.1, 0.01, 0.001, 0.0001, 0.00001]
469     N = 256;
470     p = 360;           % No. of parallel rays.
471     k = 100;           % No. of iterations.
472     theta = 0:0.5:179; % No. of used angles.
473     % Create the test problem.
474     [A,b_ex,x_ex] = paralleltomo(N,theta,p);
475     % Noise level.
476     delta = eta*norm(b_ex);
477     % Add noise to the rhs.
478     randn('state',0);
479     e = randn(size(b_ex));
480     e = delta*e/norm(e);
481     b = b_ex + e;
482     % Show the exact solution.
483     figure
484     imagesc(reshape(x_ex,N,N)), colormap gray,
485     axis image off
486     c = caxis;
487     title('Exact phantom')
488     fantoma = reshape(x_ex,N,N);
489     %fantomaname = ['fantoma' num2str(step) '.mat'];
490     %save(fantomaname, 'fantoma'); % Save the matrix to a .mat file
491     % Perform the kaczmarz iterations.
492     Xkacz = kaczmarz(A,b,k);
493     % Show the kaczmarz solution.
494     figure
495     imagesc(reshape(Xkacz,N,N)), colormap gray,
496     axis image off
497     caxis(c);
498     title('Kaczmarz reconstruction')
499     ART = reshape(Xkacz,N,N);
500     ARTname = ['ART_rnoise' num2str(eta) '.mat'];
501     save(ARTname, 'ART'); % Save the matrix to a .mat file
502     % Perform the symmetric kaczmarz iterations.
503     Xsymk = symkaczmarz(A,b,k);
504     % Show the symmetric kaczmarz solution.
505     figure
506     imagesc(reshape(Xsymk,N,N)), colormap gray,
507     axis image off
508     caxis(c);
509     title('Symmetric Kaczmarz reconstruction')
510     symART = reshape(Xsymk,N,N);
511     symARTname = ['symART_rnoise' num2str(eta) '.mat'];
512     save(symARTname, 'symART'); % Save the matrix to a .mat file

```

```

513 % Perform the randomized kaczmarz iterations.
514 Xrand = randkaczmarz(A,b,k);
515 % Show the randomized kaczmarz solution.
516 figure
517 imagesc(reshape(Xrand,N,N)), colormap gray,
518 axis image off
519 caxis(c);
520 title('Randomized Kaczmarz reconstruction')
521 randART = reshape(Xrand,N,N);
522 randARTname = ['randART_rnoise' num2str(eta) '.mat'];
523 save(randARTname, 'randART'); % Save the matrix to a .mat file
524 % Perform SART
525 Xsart = sart(A,b,k);
526 % Show the SART solution.
527 figure
528 imagesc(reshape(Xsart,N,N)), colormap gray,
529 axis image off
530 caxis(c);
531 title('SART reconstruction')
532 SART = reshape(Xsart,N,N);
533 SARTname = ['SART_rnoise' num2str(eta) '.mat'];
534 save(SARTname, 'SART');
535 end
536
537
538 y = 10:10:1000;
539
540 N = 256; % Tamano de imagen
541
542 p = 360; % No. of parallel rays.
543
544 theta = 0:0.5:179; % No. of used angles.
545
546 eta = 0.0001; % Error relativo
547
548 % Create the test problem.
549 [A,b_ex,x_ex] = paralleltomo(N,theta,p);
550
551 % Noise level.
552 delta = eta*norm(b_ex);
553
554 % Add noise to the rhs.
555 randn('state',0);
556 e = randn(size(b_ex));
557 e = delta*e/norm(e);
558 b = b_ex + e;
559
560 for k = y
561 % Perform Expectation Maximization
562 XEM = em(A,b,k);
563 figure
564 imagesc(reshape(XEM,N,N)), colormap gray,

```

---

```
565     axis image off
566     caxis(c);
567     title('Expectation maximization reconstruct')
568     EMr = reshape(XEM,N,N);
569     EMname = ['EM_k' num2str(k) '.mat'];
570     save(EMname, 'EMr');
571 end
```