Università della Svizzera italiana | Institute of Computing CI

Numerical Computing **2023**

Student: Jeferson Morales Mariciano
Discussed with: Leonardo Birindelli, Michele Dalle Rive

Solution for Project 4

Due date: Wednesday, 6 December 2023, 11:59 PM

1. General Questions [10 points]

1. What is the size of the matrix A?

In the context of Image Deblurring, where the problem defintion is

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

matrix $A \in \mathbb{R}^{n^2 \times n^2}$ is the tranformation matrix of an image of size $n \times n$ pixels, resulting from application of image kernel.
Given the original image matrix $X \in \mathbb{R}^{n \times n}$, It is then vectorized to $x \in \mathbb{R}^{n^2}$, where entries of X are stacked either row or column wise.
The blurred image is $B \in \mathbb{R}^{n \times n}$, and its vectorized form is denoted as $b \in \mathbb{R}^{n^2}$.

Matrix A can be very large as its growth is exponential to the image size, thus is important to use sparse data structures.
In particular, the matrix in *blur_data/A.mat* path is defined as: $A \in \mathbb{R}^{62500 \times 62500}$, where $n = 250$ visibly resulting from blured image $B \in \mathbb{R}^{250 \times 250}$ from *blur_data/B.mat*.

```
load('blur_data/A.mat');
size(A)
```

Listing 1: script to calculate size of matrix A

2. How many diagonal bands does A have?

Transformation matrix A is d^2-banded with $d \ll n$, hence $d \ll 250$.
The blurred image pixel is the weighted average of the surrounding pixels, such weights are defined by the kernel matrix $K \in \mathbb{R}^{d \times d}$. The bigger transformation matrix A is constructed from the kernel matrix, such that the non-zero elements of each row of A correspond to the values of K.

$$K = \frac{1}{605} \begin{bmatrix} 100 & 9 & 9 & 9 & 9 & 9 & 100 \\ 9 & 1 & 1 & 1 & 1 & 1 & 9 \\ 9 & 1 & 1 & 1 & 1 & 1 & 9 \\ 9 & 1 & 1 & 1 & 1 & 1 & 9 \\ 9 & 1 & 1 & 1 & 1 & 1 & 9 \\ 9 & 1 & 1 & 1 & 1 & 1 & 9 \\ 100 & 9 & 9 & 9 & 9 & 9 & 100 \end{bmatrix} \in \mathbb{R}^{7 \times 7}$$

Hence, A has $d^2 = 7^2 = 49$ diagonal bands.

3. What is the length of the vectorized blurred image b?

From Exercise 1, problem defintion equation 1, the length of vectorized blurred image b is n^2.
Remember that the vectorization order for this assignment is row-wise, though Matlab default is column-wise.
The length of b is 62500.

```
B = B';
b = B(:);
length(b)
```

Listing 2: script to calculate length of vectorized blurred image b

2. Properties of A [10 points]

1. If A is not symmetric, how would this affect $\tilde{A}$?

To use the conjugate gradient method, the matrix A must be symmetric positive-definite.
Even if initially A is not symmetric, after solving the augmented equation:

$$\underbrace{A^{\mathsf{T}}A}_{\tilde{A}}\,x = \underbrace{A^{\mathsf{T}}b}_{\tilde{b}}$$

$\tilde{A}$ will be symmetric since the product of any matrix and its transpose always produces a symmetric matrix.

2. Explain why solving $Ax = b$ for x is equivalent to minimizing $\frac{1}{2}x^{\mathsf{T}}Ax - b^{\mathsf{T}}x$ over x, assuming that A is symmetric positive-definite.

Instead of solving $Ax = b$, assume precondition: matrix A is symmetric and positive definite (SPD), meaning

$$\langle Ax, x\rangle > 0, \text{ if } x \neq \underline{0}$$

Then, solve the minimization problem:

$$f(x) := \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$$

Now, let's find the derivative of $f(x)$ with respect to x.

$$\begin{aligned} f(x) &= \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle \\ &= \frac{1}{2}x^T Ax - b^T x \end{aligned}$$

The derivative of the quadratic form $x^T Ax$ with respect to x is given by

$$\frac{d}{dx}(x^T Ax) = (A + A^T)x$$

Therefore, the derivative of $f(x)$ with respect to x is

$$\begin{aligned} \frac{df}{dx} &= \frac{1}{2}\frac{d}{dx}(x^T Ax) - \frac{d}{dx}(b^T x) \\ &= \frac{1}{2}(A + A^T)x - b \end{aligned}$$

So, the result is

$$\frac{df}{dx} = \frac{1}{2}(A + A^T)x - b$$

Remember, since matrix A is symmetric, then $A^{\mathsf{T}} = A$, thus:

$$\begin{aligned} &= \frac{1}{2}(A + A^{\mathsf{T}})x - b \\ &= \frac{1}{2}(A + A)x - b \\ &= Ax - b \end{aligned}$$

Finally, it becomes clear why solving $Ax = b$ is equivalent to minimizing $\frac{1}{2}x^{\mathsf{T}}Ax - b^{\mathsf{T}}x$ over x with assumptions on A being SPD.

3. Conjugate Gradient [30 points]

1. Write a function for the conjugate gradient solver [x,rvec]=myCG(A,b,x0,max itr,tol), where x and rvec are, respectively, the solution value and a vector containing the residual at every iteration.

The file containing the solution can be found in *myCG.m* matlab script file.

2. In order to validate your implementation, solve the system defined by A_test.mat and b_test.mat. Plot the convergence (residual vs iteration).

Hereby, it is shown the relation between residual and iterations during convergence in both bar chart in Figure 1 and logarithmic chart in Figure 2.

Figure 1: Convergence, residual vs iteration

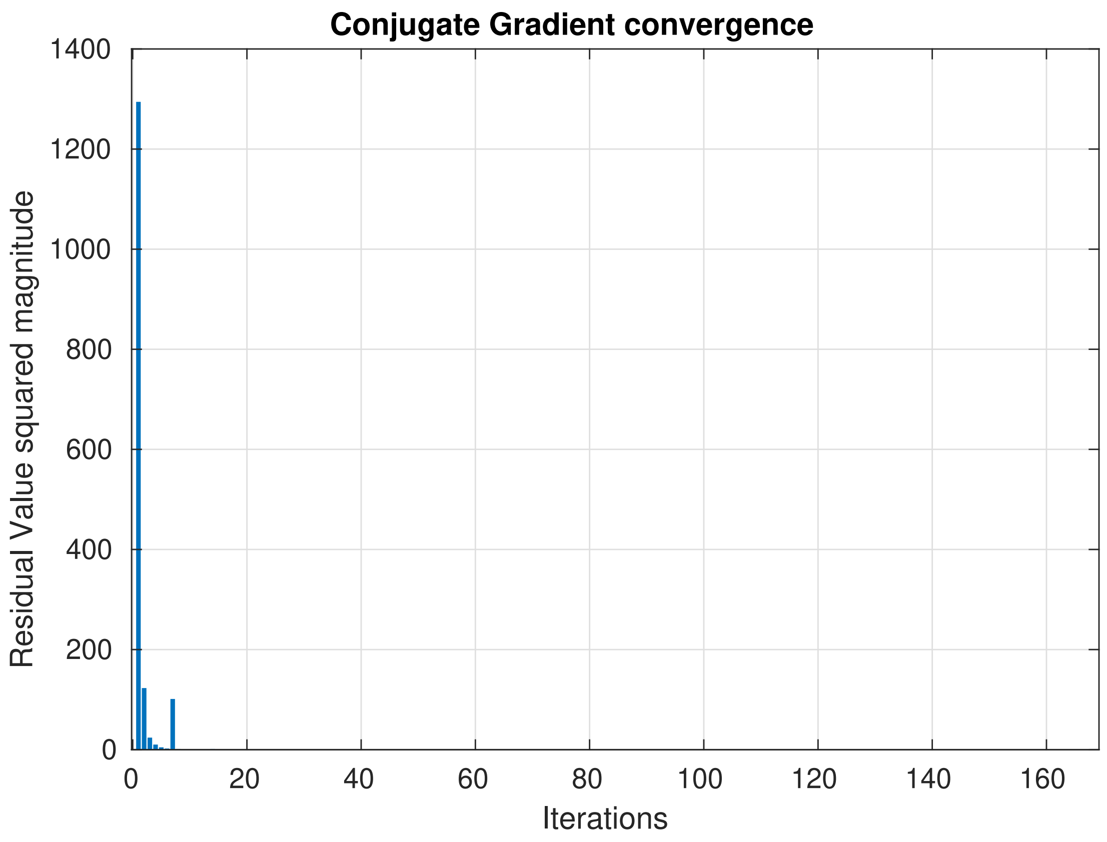
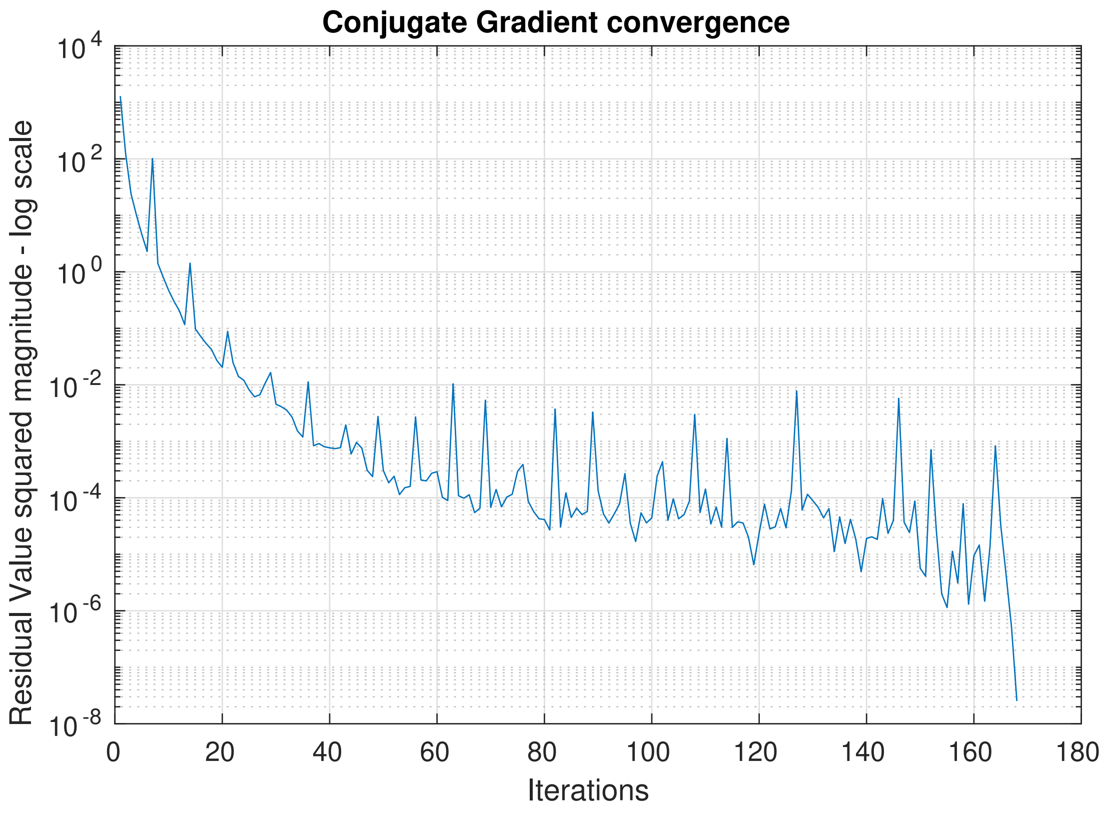


Figure 2: Convergence, residual vs iteration - logarithmic view



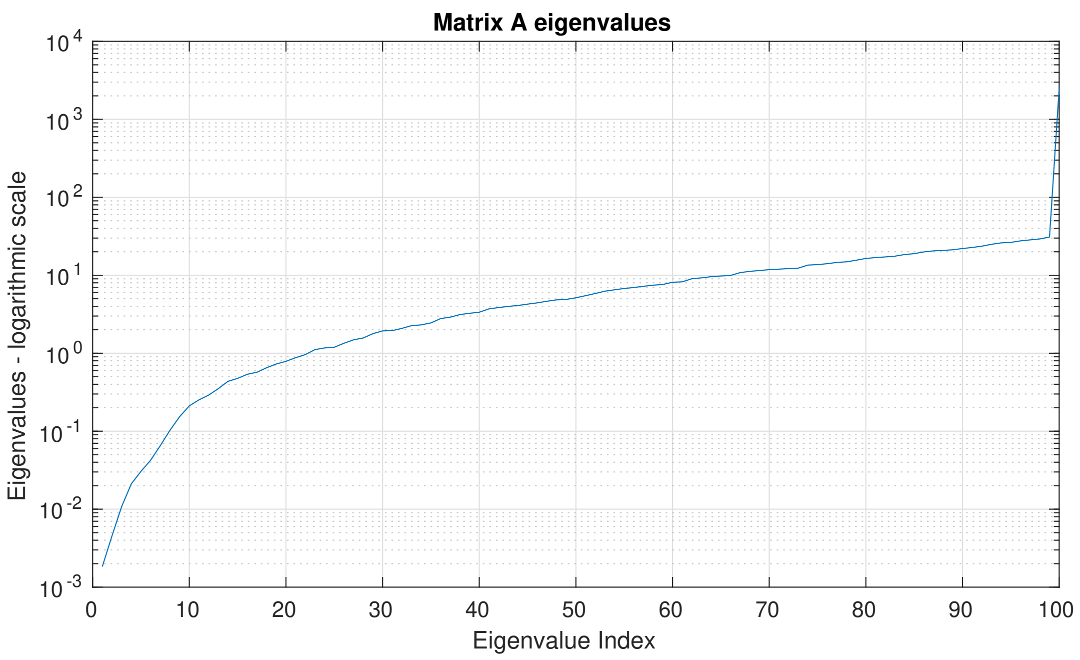
3. Plot the eigenvalues of A_test.mat and comment on the condition number and convergence rate.

The eigenvalues of matrix A are plotted in Figure 3.
Recall that condition number of a matrix is defined as the ratio of the largest to the smallest singular value, and if very big, it indicates the matrix is ill-conditioned.

$$k(A) = \frac{\sigma_{max}}{\sigma_{min}}$$

where σ indicates singular values of A.
The condition number of A_{test} at start is computed with matlab with the *cond(A_{test})* command. The results is $k(A) = 1.3700e + 06 = 1.37 * 10^6$, a huge number explaining why that convergence of the Conjugate Gradient method is slow paced since the convergence is hinderend with such a high number.

Figure 3: Eingenvalues of matrix A



4. Does the residual decrease monotonically? Why or why not?

Figures 1, 2 strongly suggest the residuals are not monotonically decreasing by presenting peaks along the chart, despite the Conjugate Gradient method supposed to monotonically improving approximations of x_k to the exact solution. The relation with the CG method is explainable because of its inner instability with respect to small pertubations, particularly considering the fact that its speed is determined by the condition number $k(A)$ of the system matrix, meaning: the larger $k(A)$, the slower the improvement.

Apporting small perturbations to decrease the condition number, e.g. preconditioning, should denote approximations of x_k being monotonically more accurate during the iterative method.

4. Deblurring problem [35 points]

1. Solve the deblurring problem for the blurred image matrix B.mat and transformation matrix A.mat using your routine myCG and Matlab's preconditioned conjugate gradient pcg. As a preconditioner, use ichol to get the incomplete Cholesky factors and set routine type to nofill with $\alpha = 0.01$ for the diagonal shift (see Matlab documentation). Solve the system with both solvers using $max\_iter = 200$, $tol = 10^{-6}$. Plot the convergence (residual vs iteration) of each solver and display the original and final deblurred image. Comment on the results that you observe.

Figure 4, 5 show respectively the blurred image and the deblurred one using the assignment own implementation of Conjugate Gradient method.
The plot in Figure 6 show the relation of the residual squared magnitude using the Conjugate Gradient method.

Figure 7, 8 show respectively the blurred image and the deblurred one using the Preconditioned Conjugate Gradient method.
Such method uses the incomplete Cholesky factorization as preconditioner, already implemented in matlab API with function signatures *ichol(...)*. The plot in Figure **??** show the relation of the

residual squared magnitude using the Preconditioned Conjugate Gradient method.

The precondiotioned method converge way faster as it is visible from the plots. Such phenomenon is due to the fact that the condition number, computed with $condest(\tilde{A}) = 2.0501e + 11 = 2.0501 * 10^{11}$ is very high and thus the preconditioner method is capable to show off its potential by reducing the condition number and reach convergence in a very smaller number of iterations than simple Conjugate Gradient Method.

Figure 4: Blurred Image

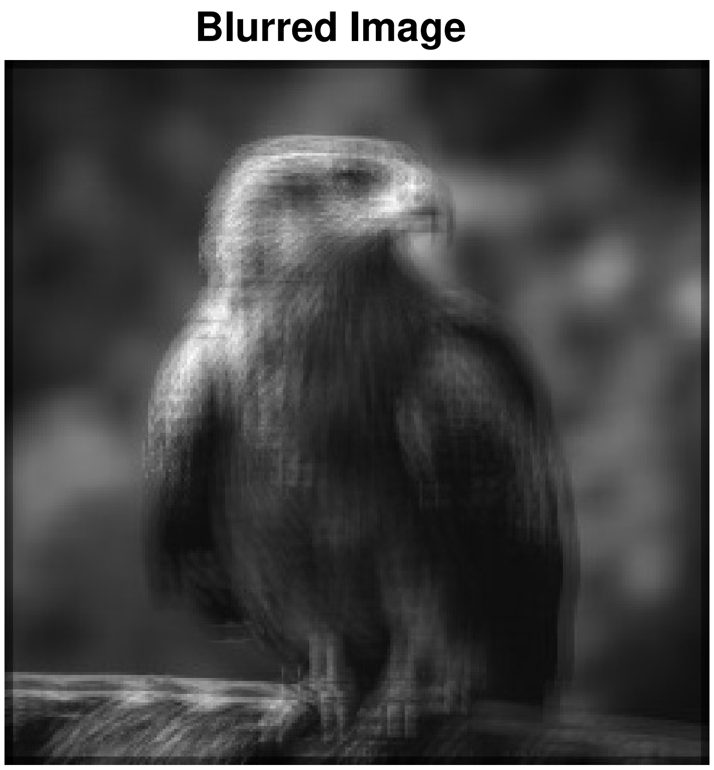


Figure 5: Deblurred Image using CG method

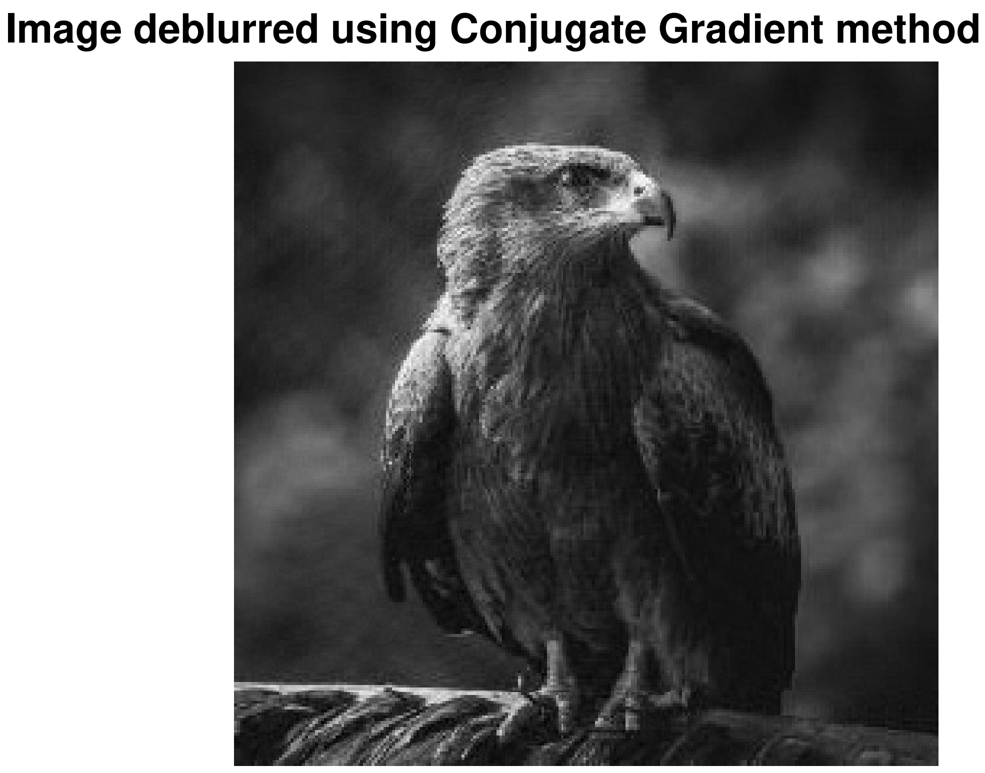


Figure 6: Relation between residual squared magnitude and iterations with CG method

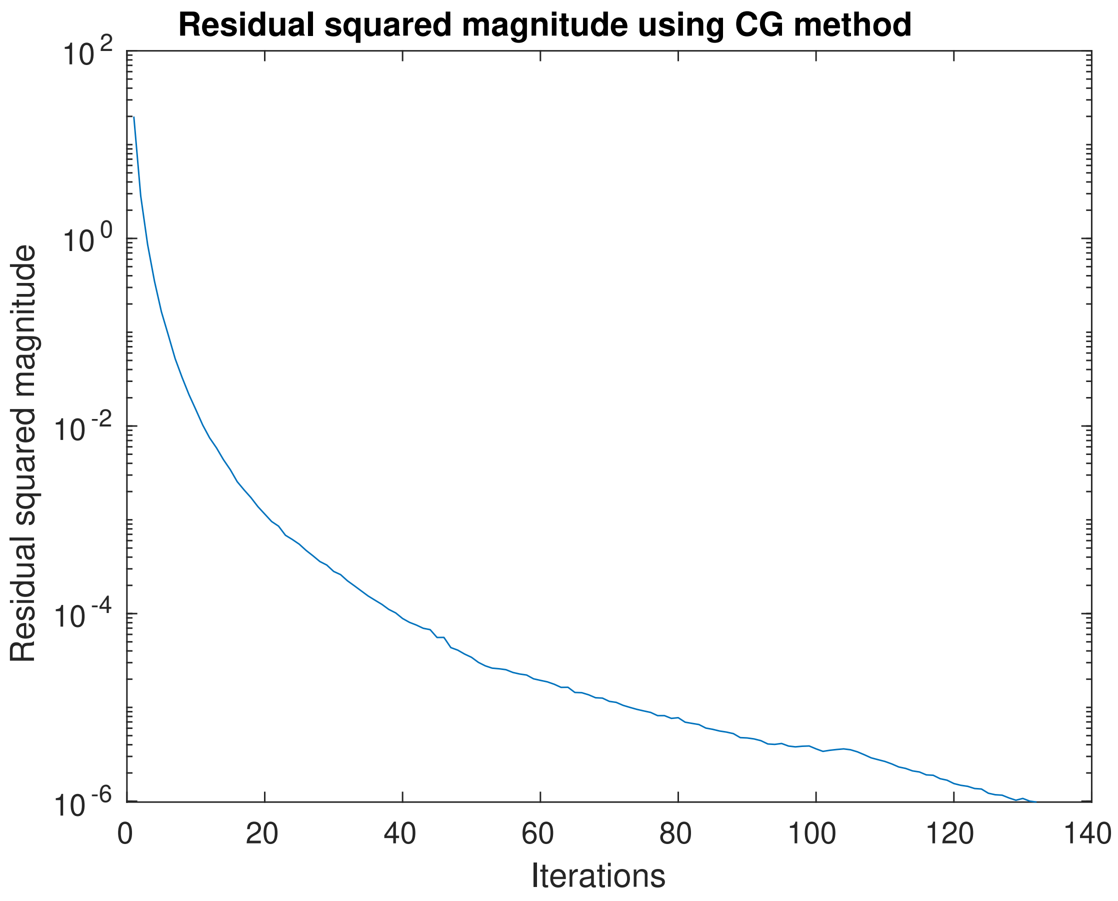


Figure 7: Blurred Image

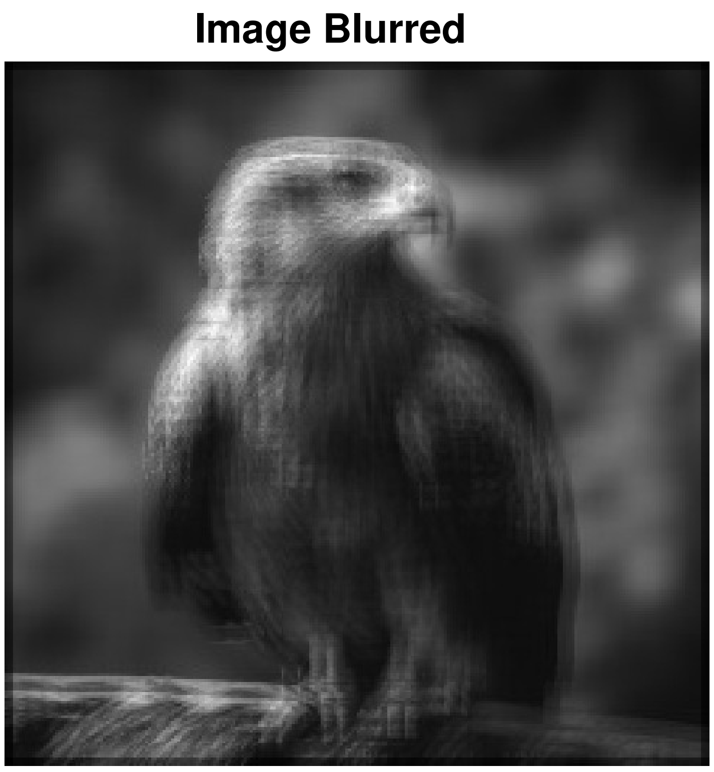


Figure 8: Deblurred Image using PCG preconditioning

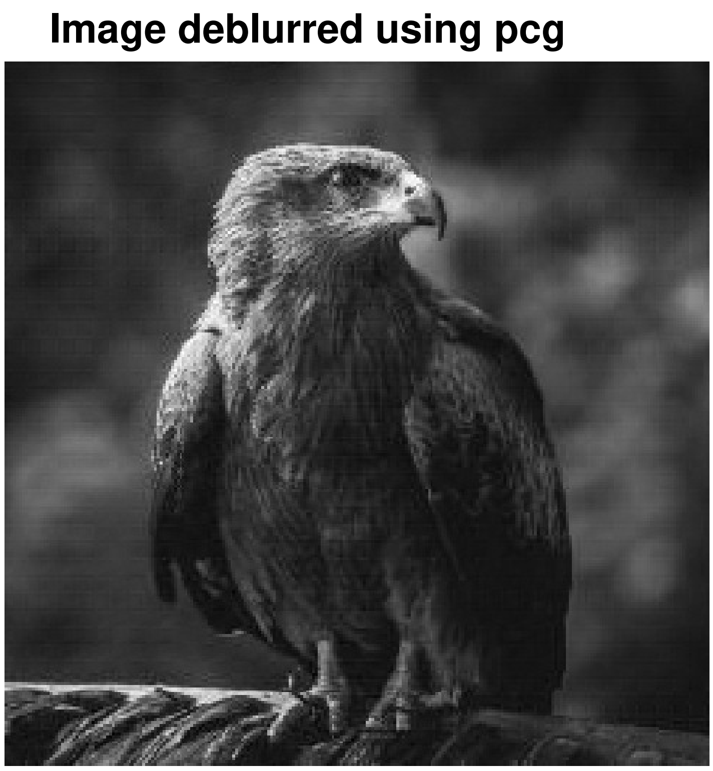
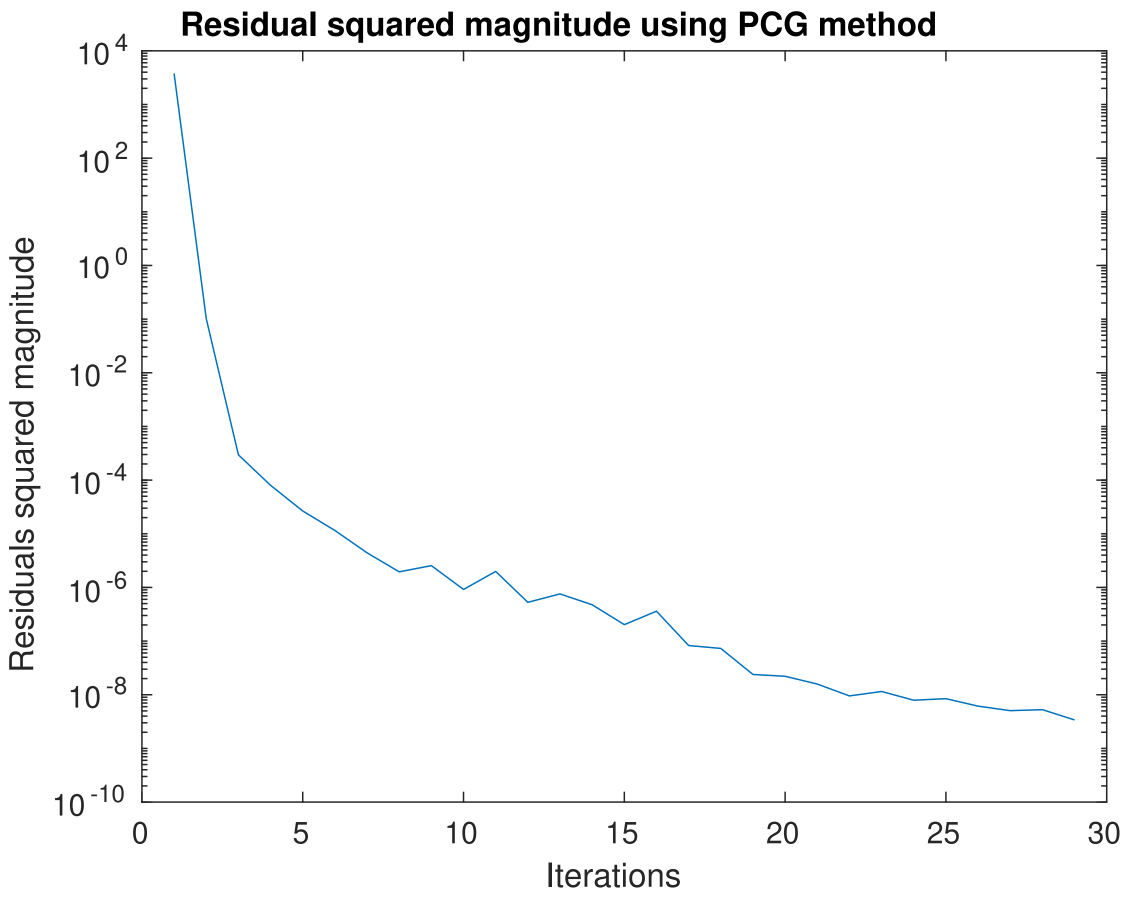


Figure 9: Relation between residual squared magnitude and iterations with PCG preconditioning



2. When would pcg be worth the added computational cost? What about if you are deblurring lots of images with the same blur operator?

The preconditioning must be stored in memory and for very large matrix it cannot be feasible. Such method have indeed strong advantages but when the memory size is impractical are even impossible to reach if it is in the magnitude of Terabytes, then is discorauged.
Keep in mind that if the condition number is indeed extremely large, such method even for large matrices can be taken into consideration.
Moreover, the computational cost of the preconditioning could be distributed into multiple systems to increase feasibility, and can be a great advantages if the matrix is reused multiple times since it is enough to compute it once.