Solving Matrix Equations

Jarod Klion

November 16, 2022

1 Executive Summary

In this report, we consider the solutions x of symmetric positive definite (SPD) matrix problems, $Ax = b$, using two methods: 1.) preconditioned steepest descent (PSD) and 2.) preconditioned conjugate gradient (PCG) with one of three preconditioners: 1.) no preconditioner, P_I, 2.) Jacobi's preconditioner, P_{Jacobi}, and 3.) symmetric Gauss-Seidel precondioner, P_{SGS}. Correctness of the analytic derivations is evaluated by relative error calculations depending on whether the solution is known beforehand or not. Results are obtained and discussed for 4 randomly generated SPD matrices of increasing size, and the given linear system: $A_{test} = \begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix}$, and $b_{test} = \begin{bmatrix} 23 \\ 32 \\ 33 \\ 31 \end{bmatrix}$.

2 Statement of the Problem

The numerical solution to the linear system $Ax = b$ is important in nearly all scientific- and mathematical-computing applications as matrices exist and are used everywhere. Preconditioned solving methods can be used to solve the linear system more computationally efficiently using no preconditioner, Jacobi's preconditioner, and the symmetric Gauss-Seidel preconditioner given by:

- $P_I = I$
- $P_{Jacobi} = \text{diag}(A)$
- $P_{SGS} = CC^T \Rightarrow C = \begin{bmatrix} \frac{a_{11}}{\sqrt{a_{11}}} & 0 & \cdots & 0 \\ \frac{a_{21}}{\sqrt{a_{11}}} & \frac{a_{22}}{\sqrt{a_{22}}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}}{\sqrt{a_{11}}} & \frac{a_{n2}}{\sqrt{a_{22}}} & \cdots & 0 \end{bmatrix}$,

and the accuracy of the factorization should be the same as if the system was solved directly with inverses. We analyze the accuracy of the methods at each iteration using the relative error, which is calculated differently depending on if the solution, x^*, is known beforehand or not, respectively:

- relative error $= \frac{\|x_k - x^*\|_2}{\|x^*\|_2}$
- relative error $= \frac{\|r_k\|_2}{\|b\|_2}$

3 Description of the Algorithms and Implementation

The preconditioned solving methods, PSD and PCG, are translated directly into their respective functions in **Python**, which accept five arguments: (1) an SPD matrix, A, (2) a vector, b, (3) a number denoting the preconditioner type, P, (4) the solution vector, x^*, if known/desired, and (5) a float for desired accuracy level. For example, if an SPD matrix, **A**, and vector b are previously defined, then **PSD(A, b, 1, None, 10e-6)** or **PCG(A, b, 1, None, 10e-6)** performs the respective method with no preconditioning and returns the numerical solution of the matrix equation as well as the relative error at each iteration until the desired accuracy, $\|r_k\|_2 < 10^{-6}$, is reached. For the generation of the first random 10x10 and 100x100 SPD matrices, the **make_spd_matrix** function in scikit-learn's *dataset* library was used. Additionally, another 10x10 SPD matrix was generated using a lower triangular matrix multiplied by its tranpose in order to further test the solving methods.

4 Description of the Experimental Design and Results

For each given matrix and solving method, the accuracies of the solution vector are calculated directly by the formulas given in section 2. This leads us to six values for each matrix as we calculate the accuracies for each of the three preconditioning methods with either of the solving methods. These results can be seen in Tables 1, 2, 3, 4, and 5. Then, the relative error at each iteration step k for each matrix type is plotted for all 3 preconditioning methods as well as the same plots but in logscale. The associated histograms of the errors are plotted in a similar fashion after.

In Figures 1, 4, and 7, the relative error for both methods, PSD and PCG, are plotted against the increasing iteration step for no preconditioner, Jacobi's preconditioner, and the SGS preconditioner, respectively. For the case of a 2x2 matrix, we see that applying a preconditioner decreases the amount of steps it takes for either method to reach the desired accuracy with the effect being much more prominent for PSD - bringing down the steps required from $\sim$13 to 1. For the case of the 4x4 given test matrix, PSD takes several hundred steps regardless of the preconditioner applied while PCG takes $\sim$4 steps regardless. For the case of solving a randomly generated 10x10 SPD matrix with PSD, Jacobi's preconditioner took the most iterations at $\sim$278 while the SGS preconditioner

took the fewest at ∼49. Solving that same matrix with PCG instead yields much lower results with all preconditioners taking ∼10 iterations before reaching the desired accuracy. Lastly, for the case of solving a randomly generated 100x100 SPD matrix with PSD, using no preconditioner takes upwards of 100000 iterations while applying either preconditioner cuts that to a third or fourth of the iterations needed. Solving that same matrix with PCG instead yields drastically lower iteration values and Jacobi's preconditioner returns to taking the most steps at ∼100 and no preconditioner taking the fewest.

Similarly to the other figures, in Figures 10, 11, 12, the relative error achieved when solving the matrix generated by the lower triangular matrix with either PSD and PCG is plotted against the iteration step. In the case of solving this type of matrix with PSD, one can see from both figure 10 and table 4 that the SGS preconditioner takes far fewer iterations - an order of magnitude less - than the cases of no preconditioner or Jacobi's preconditioner. One obvious difference between the 10x10 matrices is that regardless of the preconditioner used when solving this matrix with PSD, the iteration count is much higher than it was for the 10x10 matrix generated by the scikit-learn library. However, when solving this matrix with PCG, the resulting plots look very similar to the other 10x10 matrix in terms of both relative errors at each step and the total iteration steps required.

As can be seen from the discussion above as well as some analysis of the histograms shown in Figures 3, 6, 9, PCG takes far fewer steps to solve these matrix systems than PSD, often by an entire order of magnitude. Focusing on the idea of steepest descent, as we get closer to the actual solution values, the descent becomes less and less steep, leading to much smaller increases in accuracy per iteration which can be seen by the many occurences of near 0 error values in the histograms. As can also be expected, as the size of the matrix increasing, a larger and larger amount of steps is required to reach the desired accuracy. The outlier is the given 4x4 test matrix, where there is instantly a massive increase in accuracy for the first few steps that tapers off, taking several times more steps than the randomly generated 10x10 matrix. One explanation for this is that the given test matrix is not diagonally dominant and has a condition number, $K(A_{test})$, of about 30, leading to a 'zig-zag' between solution values and, thus, slow convergence.

Additionally, the actual solution and the numerically achieved final solutions for the test system for each method and preconditoner are shown below:

Actual Solution:

$$x^* = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

No Preconditioner Results:

$$\mathrm{x}_{PSD} = \begin{bmatrix} 1.046 \\ 0.972 \\ 0.988 \\ 1.006 \end{bmatrix}, \quad \mathrm{x}_{PCG} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$$

Jacobi's Preconditioner Results:

$$\mathrm{x}_{PSD} = \begin{bmatrix} 1.030 \\ 0.982 \\ 0.992 \\ 1.005 \end{bmatrix} \quad \mathrm{x}_{PCG} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$$

SGS Preconditioner Results:

$$\mathrm{x}_{PSD} = \begin{bmatrix} 1.041 \\ 0.975 \\ 0.989 \\ 1.006 \end{bmatrix} \quad \mathrm{x}_{PCG} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$$

5 Conclusions

Our results illustrate the accuracy of the preconditioned steepest descent and preconditioned conjugate gradient method solutions to matrix systems. We found that for the case of the preconditioned steepest descent method, the SGS preconditioner gave the desired solution accuracy in the fewest iteration steps regardless of the matrix size. Similar findings were obtained for the preconditioned conjugate gradient method although the differences between preconditioners was less severe in this case. This raises the question of how to most efficiently apply the SGS preconditioner to PSD as a dense storage approach was used for the calculations as well as raising the question if SGS preconditioner is actually the best to use practically speaking when taking into account possible time-constraints. A topic not considered in this report is the timing of each method. It is possible that despite taking the fewest overall iteration steps, the calculation time per iteration step in SGS makes the preconditioner method take the longest time-wise, which would become more and more problematic as the matrix size increases. Answers to this question and related ones could be studied further in the future or, perhaps, found in the literature.

6 Tables and Figures

6.1 No Preconditioner Plots

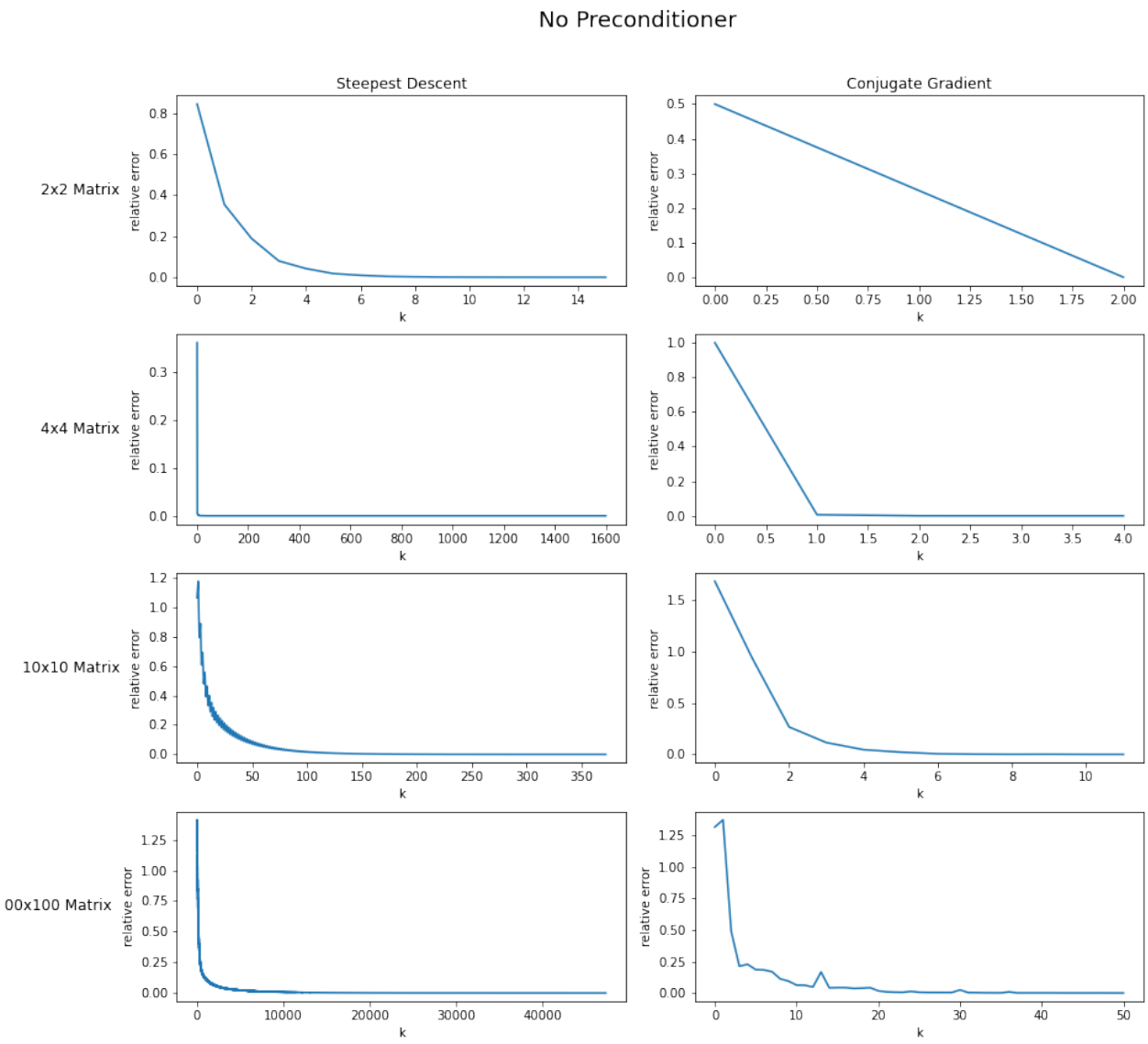

Figure 1: Relative error versus the iteration number when using no preconditioner with either steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

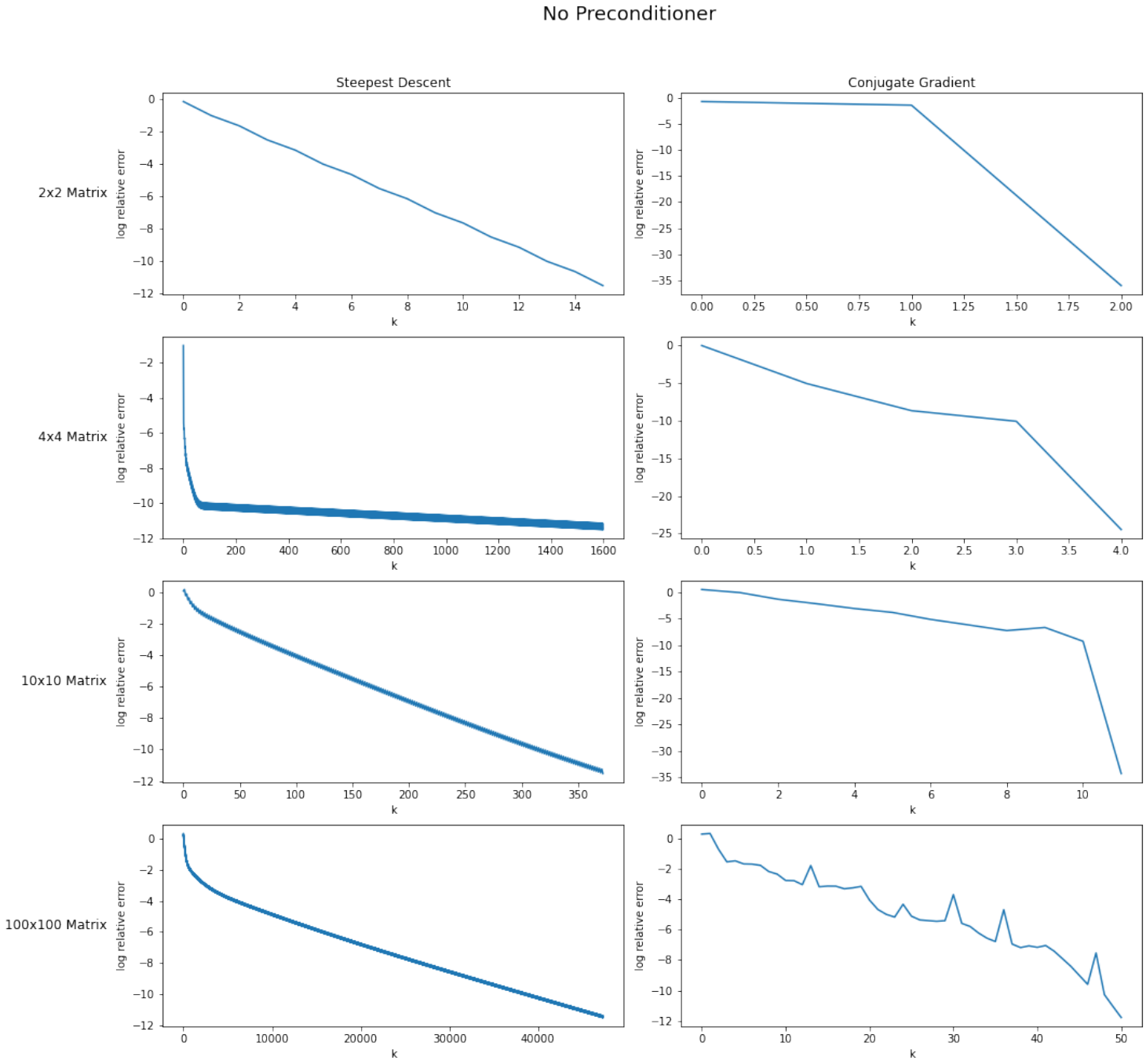

Figure 2: Log of the relative error versus the iteration number when using no preconditioner with either steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

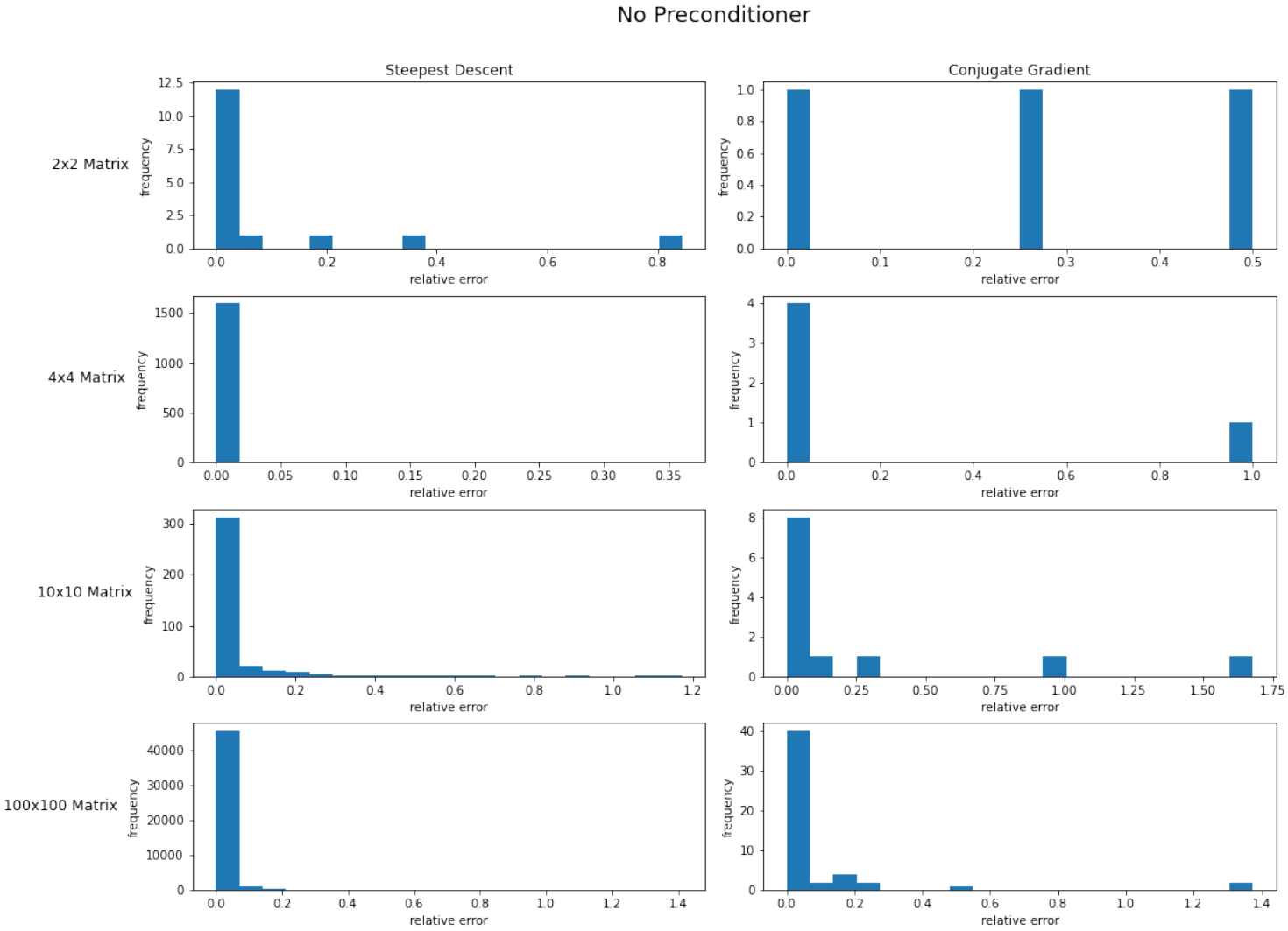

Figure 3: Histogram of relative errors when using no preconditioner with either steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

6.2 Jacobi's Preconditioner Plots

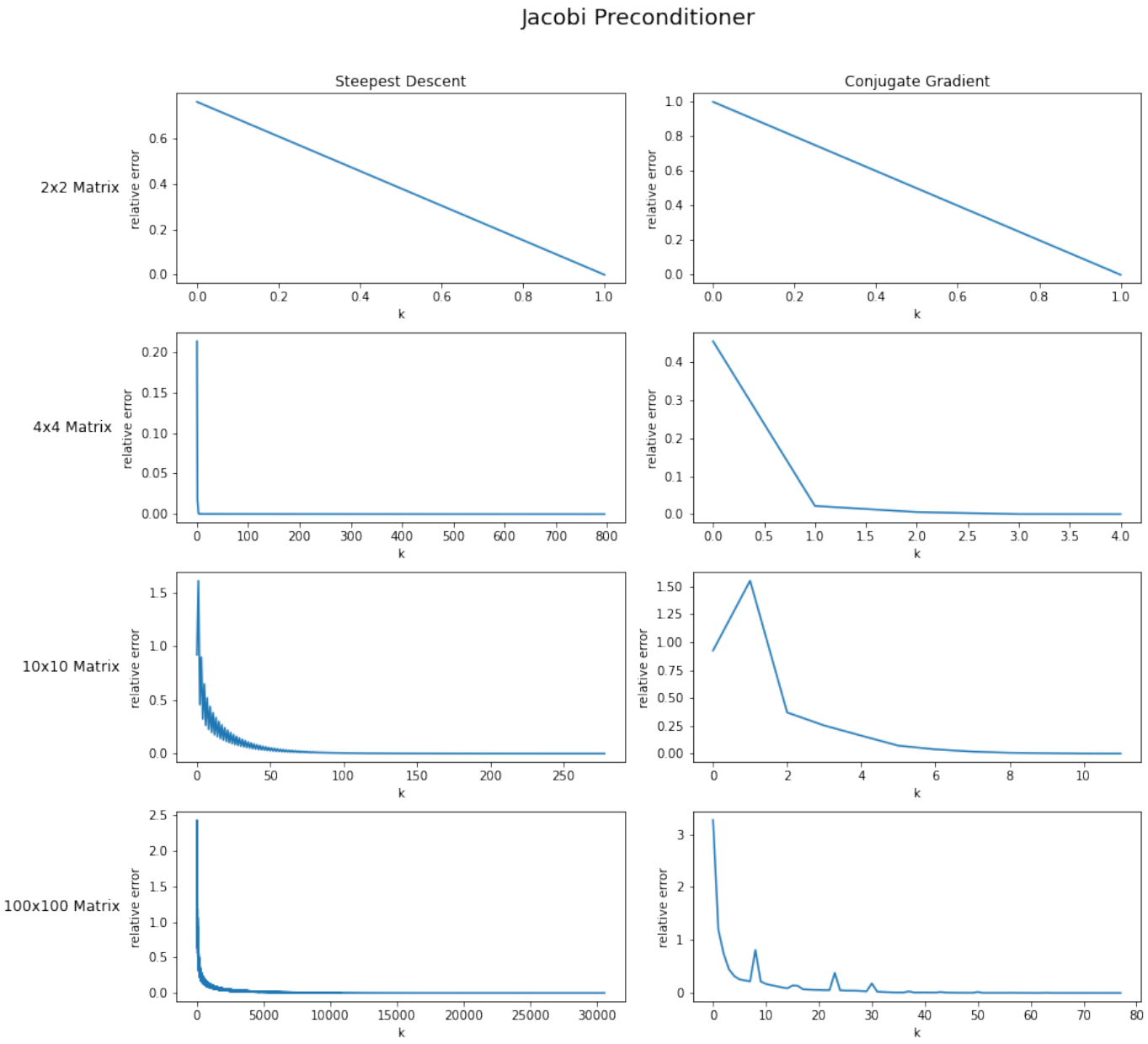

Figure 4: Relative error versus the iteration number when using Jacobi's preconditioner with steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

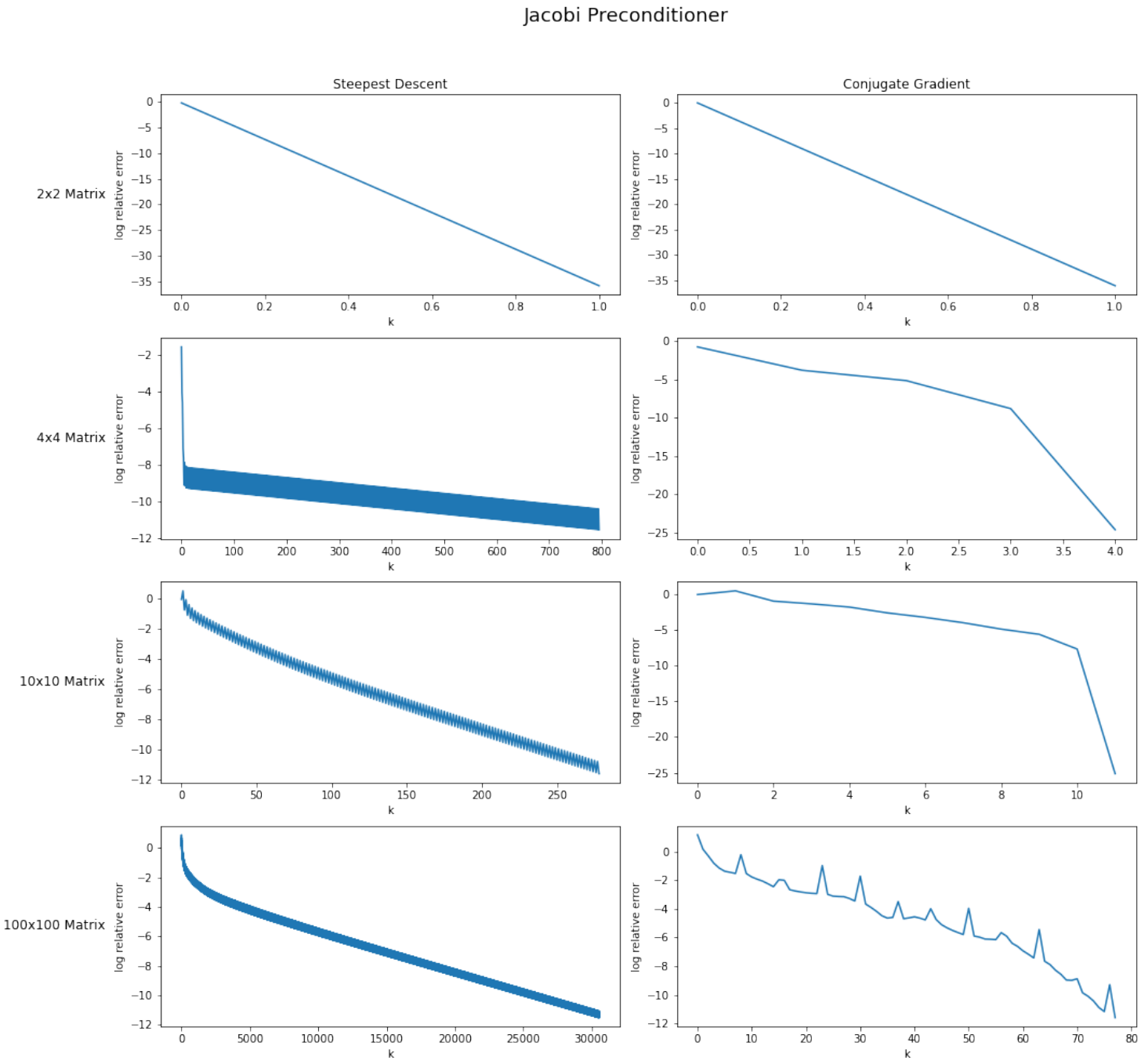

Figure 5: Log of the relative error versus the iteration number when using Jacobi's preconditioner with steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

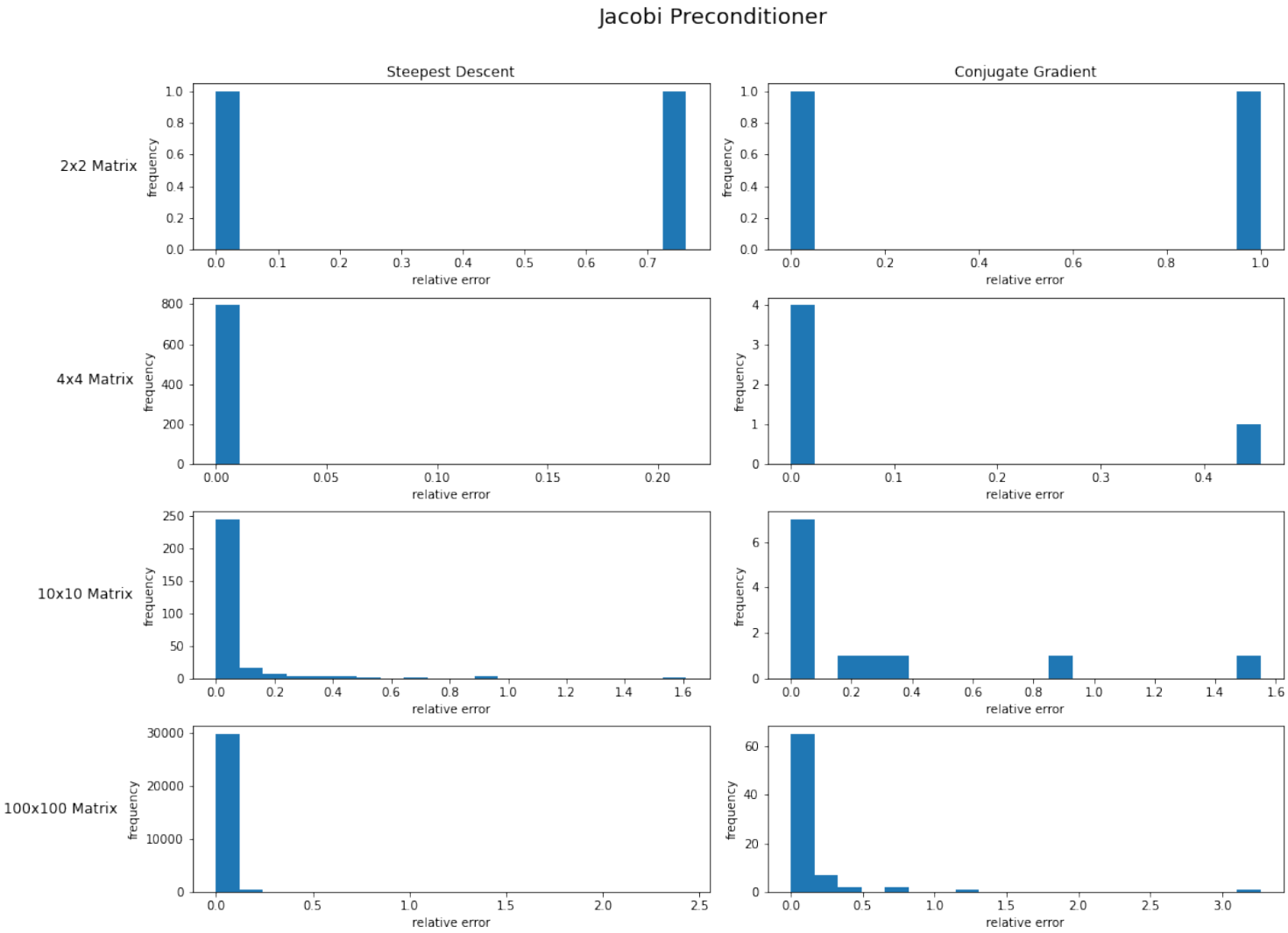

Figure 6: Histogram of relative errors when using Jacobi's preconditioner with steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

6.3 Symmetric Gauss-Seidel Preconditioner Plots

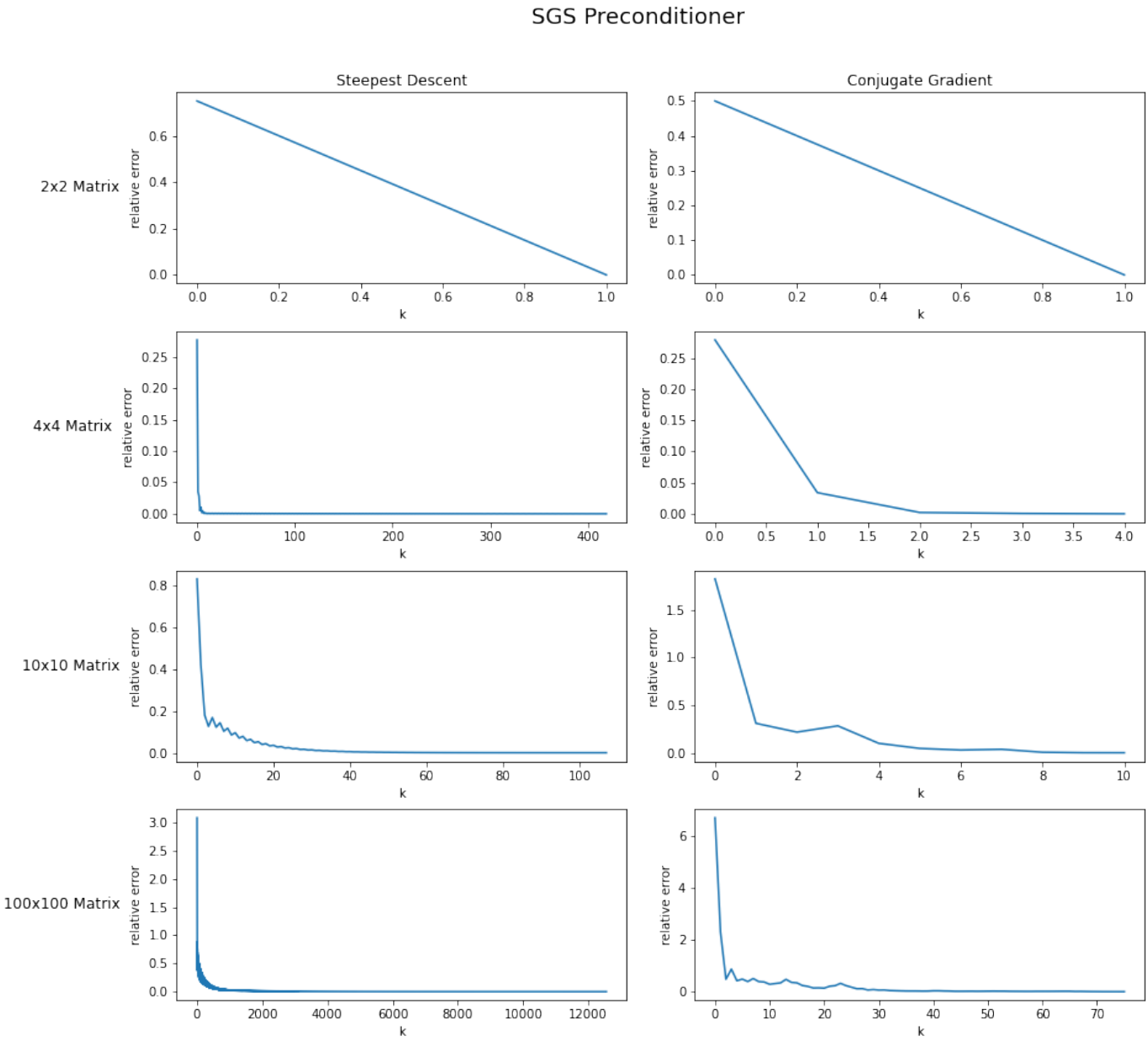

Figure 7: Relative error versus the iteration number when using the SGS preconditioner with steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

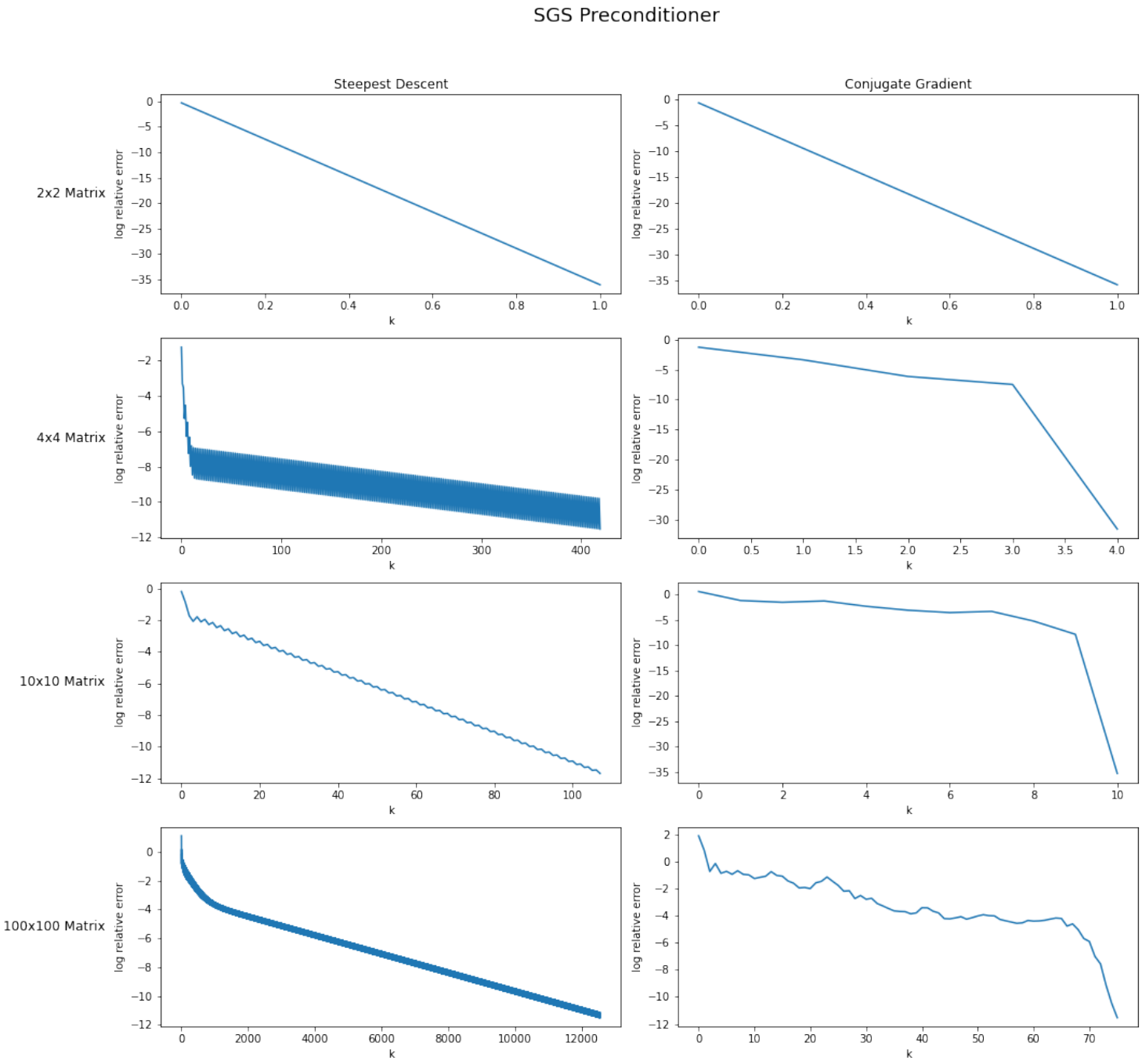

Figure 8: Log of the relative error versus the iteration number when using the SGS preconditioner with steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

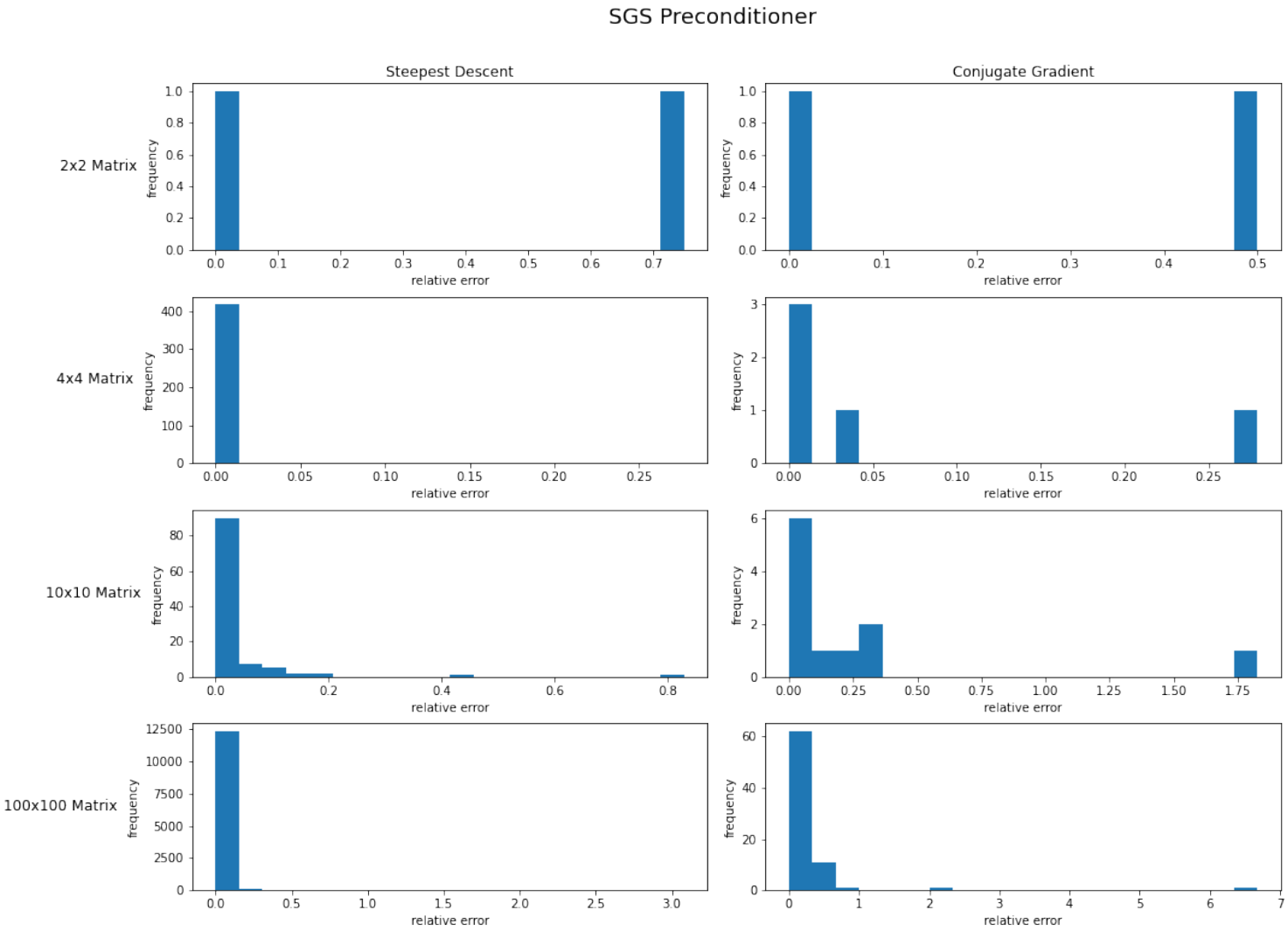

Figure 9: Histogram of relative errors when using the SGS preconditioner with steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

6.4 SPD Matrix From L_1

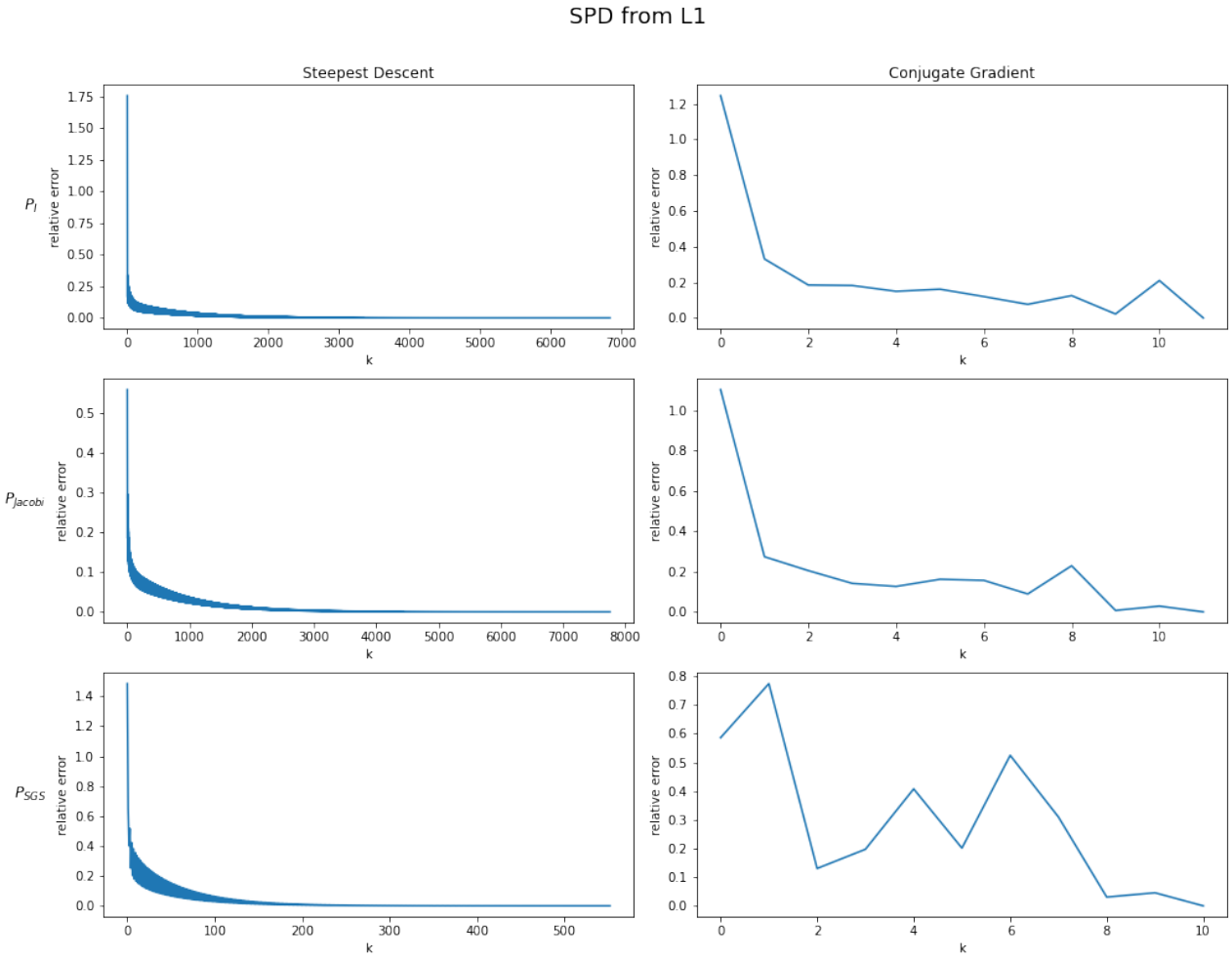

Figure 10: Relative error versus the iteration number when using no preconditioner (row 1), Jacobi's preconditioner (row 2), or the SGS preconditioner (row 3) with steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

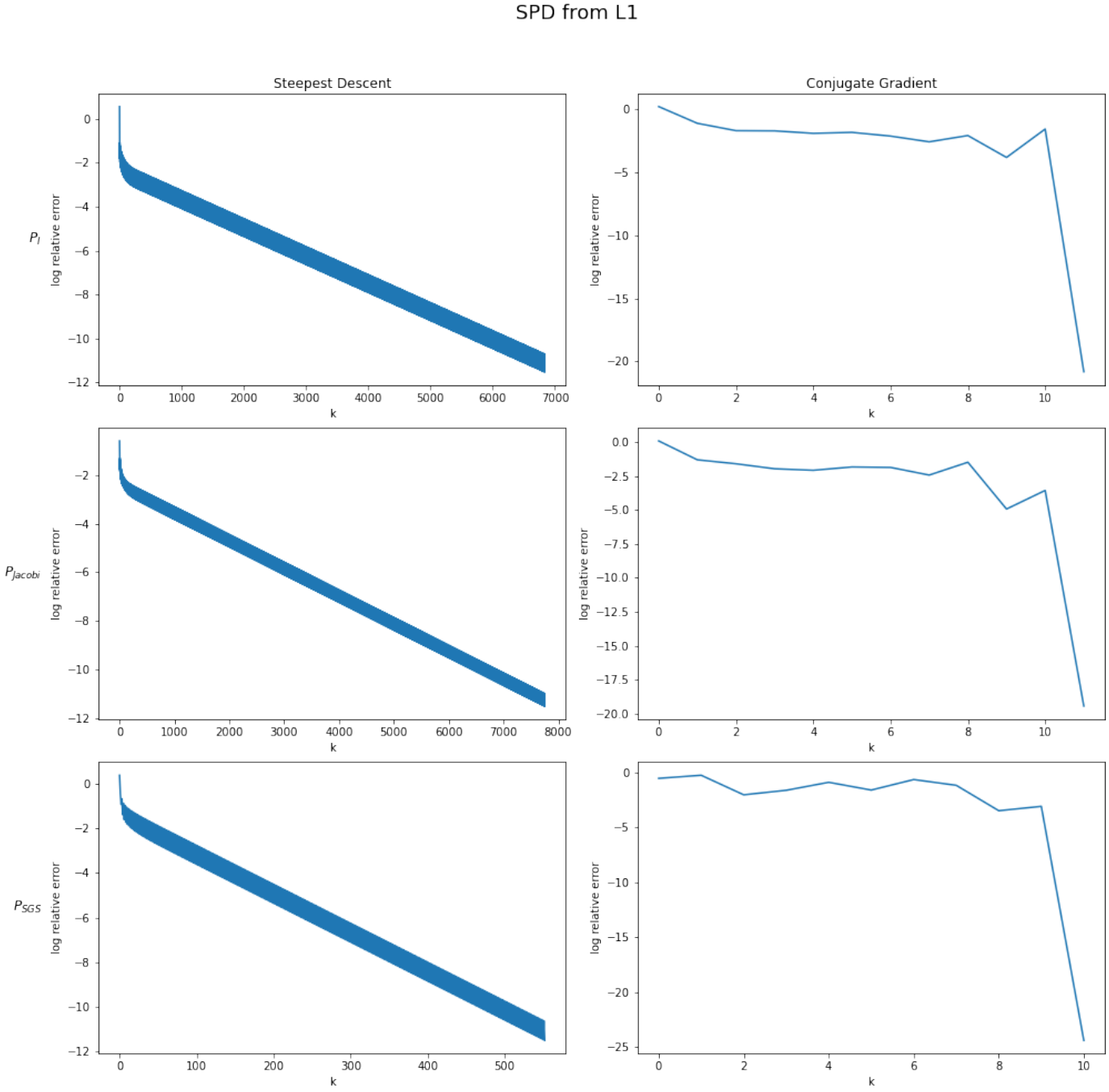

Figure 11: Log of the relative error versus the iteration number when using no preconditioner (row 1), Jacobi's preconditioner (row 2), or the SGS preconditioner (row 3) with steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

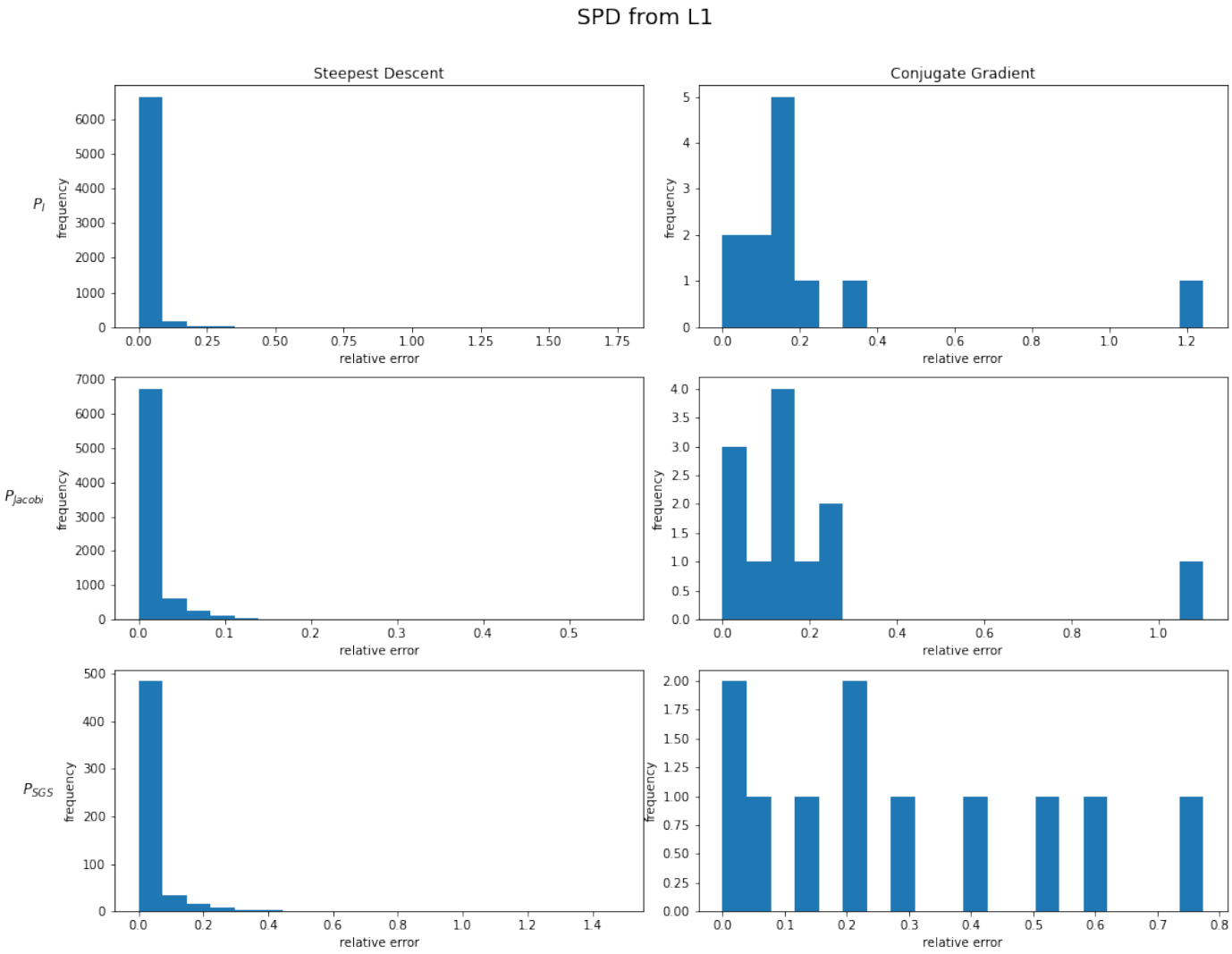

Figure 12: Histogram of relative errors when using no preconditioner (row 1), Jacobi's preconditioner (row 2), or the SGS preconditioner (row 3) with steepest descent (column 1) or conjugate gradient (column 2) as the solving method.

2x2 Matrix	PSD	PCG
k_I	13	2
k_{Jacobi}	1	1
k_{SGS}	1	1

Table 1: Iteration counts (k) for each preconditioner type for preconditioned steepest descent (column 1) and preconditioned conjugate gradient (column 2) for a 2x2 matrix system.

4x4 Matrix	PSD	PCG
k_I	15085	11
k_{Jacobi}	16656	11
k_{SGS}	1363	10

Table 2: Iteration counts (k) for each preconditioner type for preconditioned steepest descent (column 1) and preconditioned conjugate gradient (column 2) for the given 4x4 matrix test system.

10x10 Matrix	PSD	PCG
k_I	172	11
k_{Jacobi}	278	10
k_{SGS}	49	9

Table 3: Iteration counts (k) for each preconditioner type for preconditioned steepest descent (column 1) and preconditioned conjugate gradient (column 2) for a 10x10 matrix system.

L1 10x10 Matrix	PSD	PCG
k_I	783	4
k_{Jacobi}	1192	4
k_{SGS}	721	4

Table 4: Iteration counts (k) for each preconditioner type for preconditioned steepest descent (column 1) and preconditioned conjugate gradient (column 2) for a 10x10 matrix system where A is generated by a lower triangular matrix such that $A = L_1 L_1^T$.

100x100 Matrix	PSD	PCG
k_I	103742	65
k_{Jacobi}	35147	104
k_{SGS}	25269	83

Table 5: Iteration counts (k) for each preconditioner type for preconditioned steepest descent (column 1) and preconditioned conjugate gradient (column 2) for a 100x100 matrix system.