Numerical Algorithms - Homework Sheet 3

Summer Semester 2018

June 17, 2018

1 PR1: Effects of Preconditioners on Conjugate Gradient (12 Points)

1.1 Task Description & Introduction

We implement the Conjugent Gradient[1] algorithm and evaluate its performance in terms of number of iterations and run time needed until it converges to a relative residual of $\frac{||b_{approx}-b||}{||b||} = 10^{-8}$ where $b = Ax$ and $x = ones(dim(A), 1$.

We followed the pseudocode as presented in the lecture slides. The case of standard CG is implemented in the same fashion as CG with preconditioning, but with $M^{-1} = I^{-1} = I$.

Furthermore we investigated the parameter space for

1. the size of the blocks in case of the block-diagonal preconditioner,
2. α, i. e. the factor used as offset to avoid non-positive pivots during the Cholesky factorization (see here),
3. the dropping threshold that determines whether fill-in should be set to zero or not in the incomplete Cholesky factorization with threshold dropping.

We averaged three CG runs with each matrix for each hyperparameter and selected the values yielding the fastest convergence in terms of total run time. As requested in the task description, the dropping threshold is explored in greater detail (see the next subsection 1.2).

Notes regarding the implementation:

- M^{-1} was calculated using `inv(M)` for preconditioning methods diagonale and block-diagonale and using $(L^{-1})^T L^{-1}$ for the based on the Incomplete Cholesky Factorization (where L is the lower triangular matrix produced by ICF).
- We tried to represent the different preconditioning methods as different line types in our chart. In this, we did not succeed due to an apparent bug in Octave's plotting integration - line style arguments are ignored.

[1] In the following referred to as *CG*.

Instead we had to do with different marker styles. In order not to overplot and make the plot unreadable, we decided to plot only every x-th datapoint, where x depends on the number of iterations necessary in this run until convergence was reached and was chosen manually. That allows for the trend to be recognizable and to distinguish the different data series by their marker symbols (and colors, if not viewed in monochrome).

- Matrices are loaded and stored in a sparse format throughout the computation process by using the functions available here.

1.2 Investigation of Hyperparameter Values

There are three free hyperparameters that might be tuned for faster convergence - see 1.1 for more information. In order to find decent, if not necessarily strictly optimal values, we varied the values of all three and measured time and number of iterations to convergence. Time to convergence was used as decisive criterion to pick the values ultimately used.

The ideal value for α w.r.t. convergence time was zero or close to zero for the smaller files, `nos5.mtx` and `nos6.mtx`, and around 10^3 for the considerably larger `s3rmt3m3.mtx`. This an effect the latter not being diagonally dominant - a shifted incomplete Cholesky factorization can be constructed using this α in order to achieve a IC of this matrix.

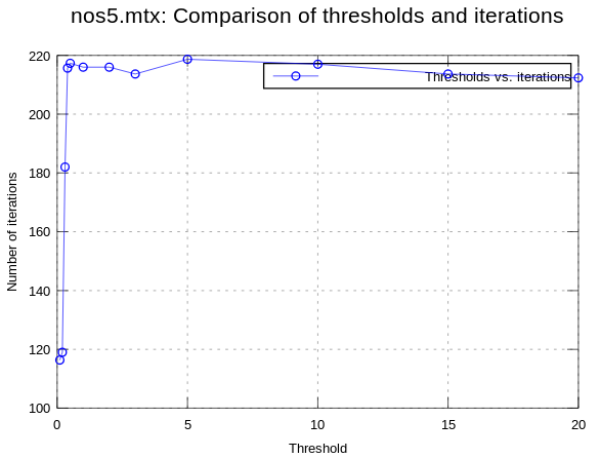
The optimal values for the size of the blocks used in the block-diagonale matrix preconditioner were approximately 2, 4 and 64 (same sequence as before: `nos5.mtx`, `nos6.mtx`, `s3rmt3m3.mtx`).

The development of the numbers of iterations and run time needed to converge to 10^{-8} in relation to the dropping threshold are shown in Fig. 1. Interestingly, while the number of iterations increases with rising threshold values for all investigated matrices, the curve exhibits a different behaviour for the first two: A sharp increase first, after which a plateau stage is reached. The bigger matrix follows a more erratic pattern, but also seems to follow a positive correlation of threshold value and number of iterations up to a degree. The run time pattern w.r.t. threshold values is similar in all matrices in so far as it seems erratic for low threshold values and then stabilizes on a comparably low level - with the exception of `nos5.mtx`: Here, higher threshold values are less desirable than those close to 0.

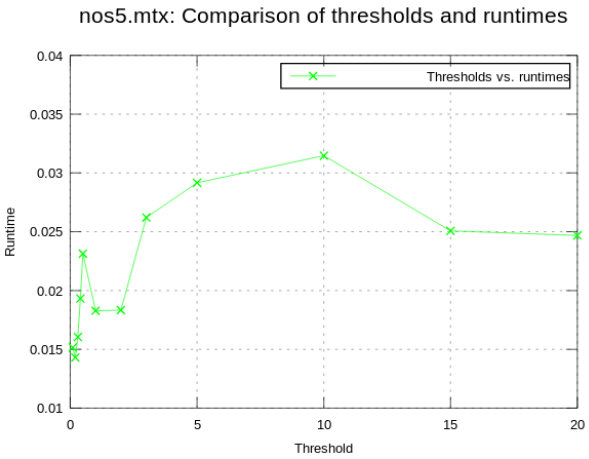
It's hard to extract generalizable recommendations from such a limited set of matrices and within the scope of this work - a more thorough exploration of the parameter space might be beneficial. For this exercise, we deduct that there is a significant bit of variance for threshold values close to 0 and that picking a decent threshold value can speed up the convergence process noticeably to drastically. Furthermore, we used the threshold values associated with the lowest amount of time to convergence for our final run (see next subsection).

1.3 Evaluation of Results

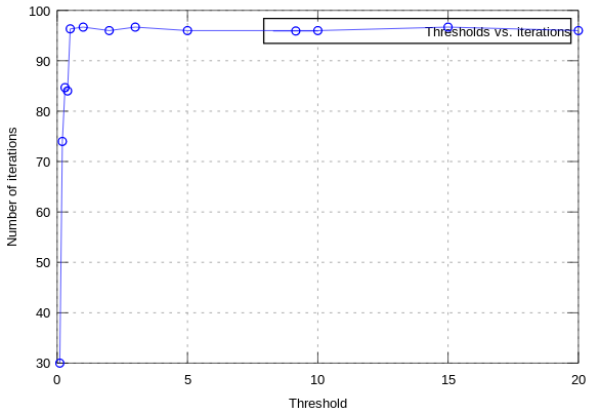
The most unexpected result was that the Incomplete Cholesky Factorization with no fill-in ($IC(0)$), while taking fewer iterations than all other methods regardless of their precondi-



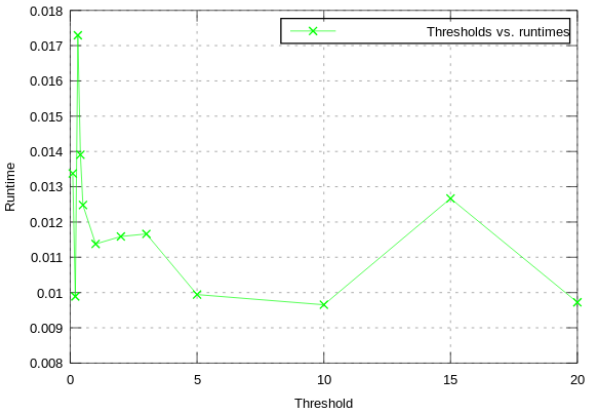
(a) `nos5.mtx`, numbers of iterations.



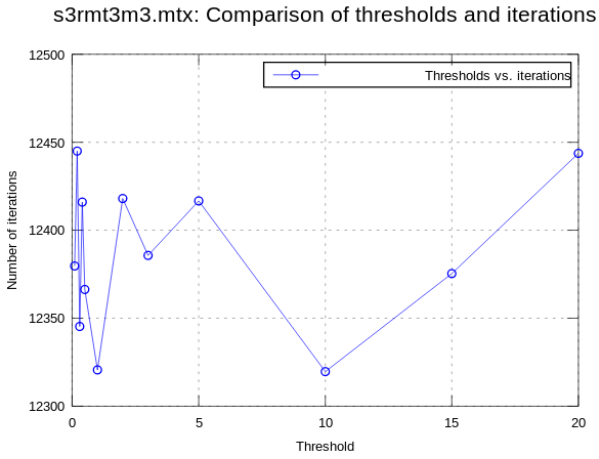
(b) `nos5.mtx`, run time.



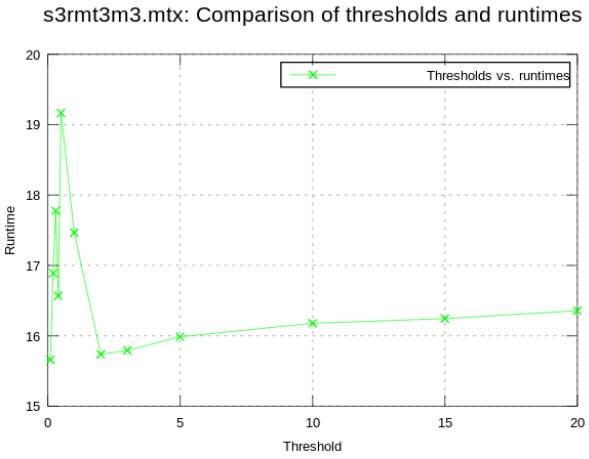
(c) `nos6.mtx`, numbers of iterations.



(d) `nos6.mtx`, run time.



(e) `s3rmt3m3.mtx`, numbers of iterations.



(f) `s3rmt3m3.mtx`, run time.

Figure 1: A comparison of numbers of iterations and run time until convergence for different threshold values on all investigated matrices.

tioning, took drastically longer to converge. In fact, attempting to reduce the residual to 10^{-8} took so long we had to cancel the run. Thus we present the following results:

1. A comparison of all run times, numbers of iterations and convergence histories for all preconditioning methods as well as standard CG for a relative residual of 10^{-4}.

2. A comparison of all run times, numbers of iterations and convergence histories for all preconditioning methods except $IC(0)$ as well as standard CG for a relative residual of 10^{-8}.

The value of 10^{-4} was chosen since it allowed us to run multiple computations with $IC(0)$ within a reasonable time frame (i. e. less than 30 minutes). 10^{-8} was mentioned as a desirable threshold in the lecture.

Since we weren't successful in finding an error in our implementation of $IC(0)$, our assumption is that the complete lack of fill-in naturally leads to a slower convergence due to the structure of the matrices investigated. Research into this topic didn't yield insights beyond the rather general statement that allowing more fill-in might lead to a better preconditioning matrix M^{-1}.

1.3.1 Discussion of results with a relative residual of 10^{-4}

Fig. 2 shows a comparison of the numbers of iterations and run times for a relative residual of 10^{-4}, Fig. 3 the histories for the relative residual. Quite obviously, $IC(0)$ always converges last - in the case of the largest matrix, dramatically so. Interestingly it has the lowest or one of the lowest number of iterations. The other preconditioning methods follow our initial naive hypotheses: The more "sophisticated" the preconditioning method, the better on average we expect the results to be[2].

Other valuable finding are that *(a)* standard CG fluctuates quite a lot: The relative residual often falls just to rise again a few iterations later and *(b)* for this termination criterion, standard PG is almost comparable to PG with preconditioning w.r.t. number of iterations needed, where it becomes obvious how much more efficient PG can be with preconditioners once we investigate more demanding termination criteria in the next subsection.

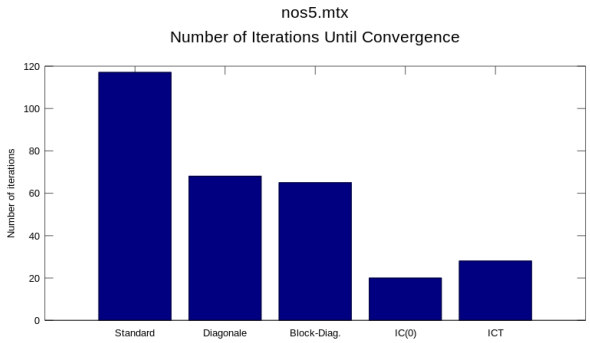
Note that a useful comparison of running times for `s3rmt3m3.mtx` is not possible with this chart, since the runtime for $IC(0)$ is so out of proportion that it dwarfs the other approaches' results. An intepretation of the running times for this matrix follows in the discussion of the results for a convergence at 10^{-8}.

1.3.2 Discussion of results with a relative residual of 10^{-8}

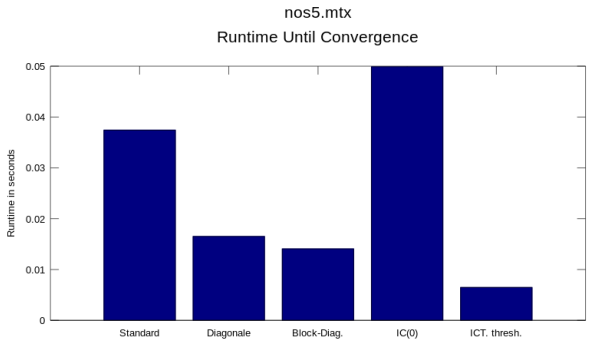
For a comparison at the actually desired threshold of 10^{-8} for the relative residual, we excluded the $IC(0)$ preconditioner due to the aforementioned performance problems.

Fig. 4 displays information on the numbers of iterations and the run times required until convergence, Fig. 5 shows the development of the relative residual over time (i. e. iteration number).

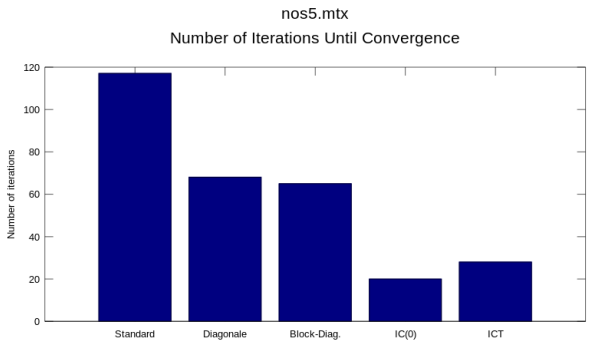
[2]This does not necessarily apply to all situations, but is meant to reflect our original subjective personal expectations.



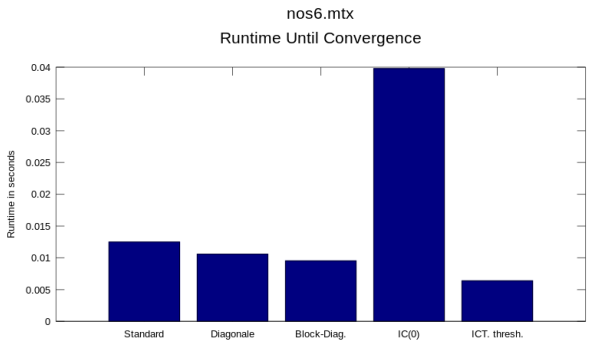
(a) `nos5.mtx`, num. of iterations.



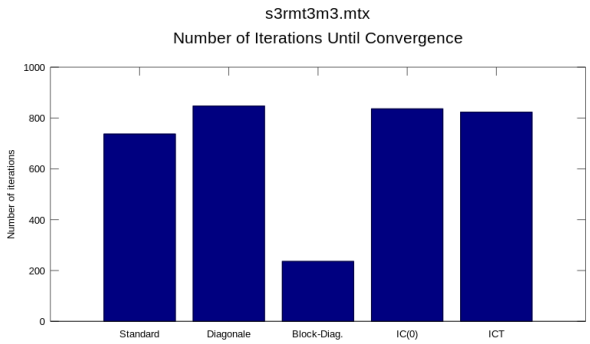
(b) `nos5.mtx`, run time.



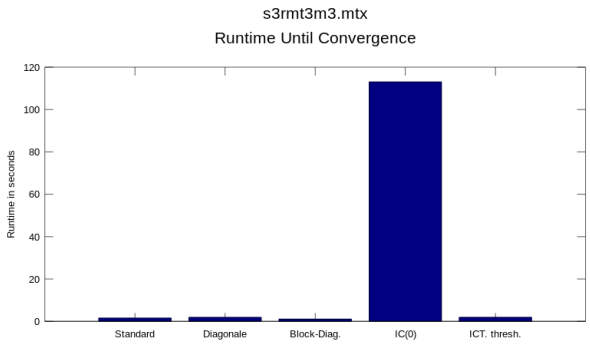
(c) `nos6.mtx`, num. of iterations.



(d) `nos6.mtx`, run time.



(e) `s3rmt3m3.mtx`, num. of iterations.



(f) `s3rmt3m3.mtx`, run time.

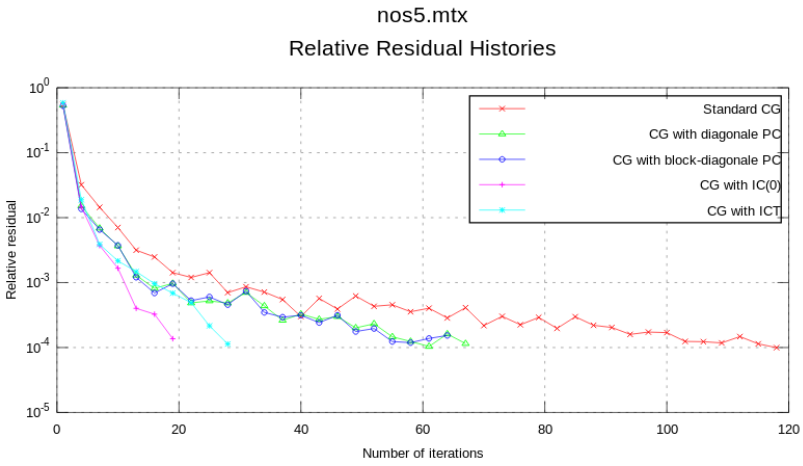
Figure 2: A comparison of numbers of iterations, run times and relative residual histories until convergence with a relative residual of 10^{-4} for different preconditioning methods and standard Conjugate Gradient.

Surprisingly enough, *ICT* is outperformed by the diagonale and block-diagonale approach for `nos6.mtx` and `s3rmt3m3.mtx`. Particularly the biggest matrix is processed very quickly using the block-diagonale preconditioner, which undermines the aforementioned naive hypothesis of a correlation of "sophistication" and performance. Especially favorably chosen hyperparameters might also contribute to the fast convergence.

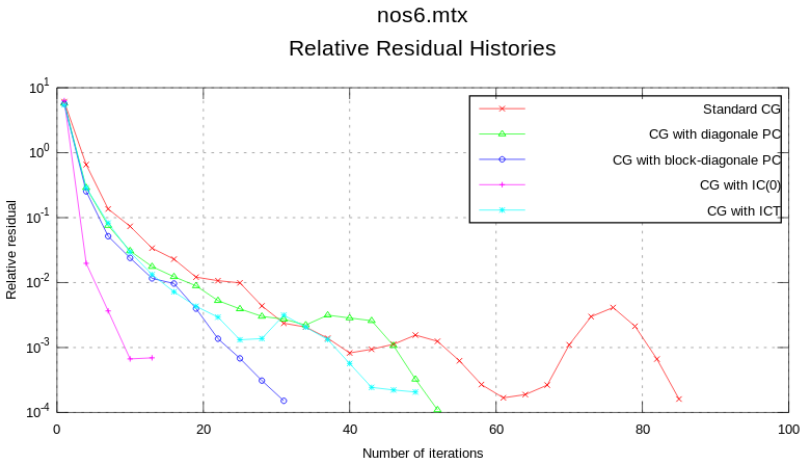
As in the previous discussion on the results with a relative residual of 10^{-4} as convergence criterion, a correlation between number of iterations and run time exists, but not in all circumstances. I. e. the number of iterations can't be used as an sole reliable indicator of a

preconditioner's performance.

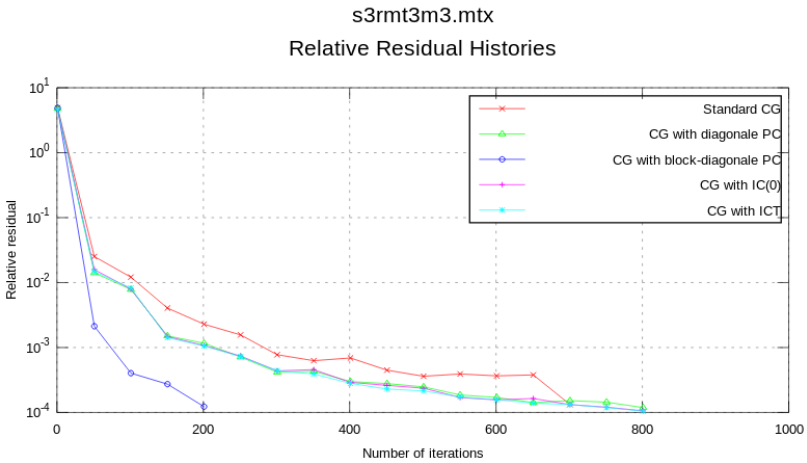
The patterns visible in the relative residual histories are rather similar to the ones in Fig. 3: The standard CG approach without preconditioners fluctuates a lot; the preconditioning methods allow CG to converge much faster. Compellingly, the diagonale and the block-diagonale approach follow very similar trends for `nos5.mtx` and `nos6.mtx`, whereas the results for `s3rmt3m3.mtx` differ quite a bit.



(a) `nos5.mtx`, relative residual history.

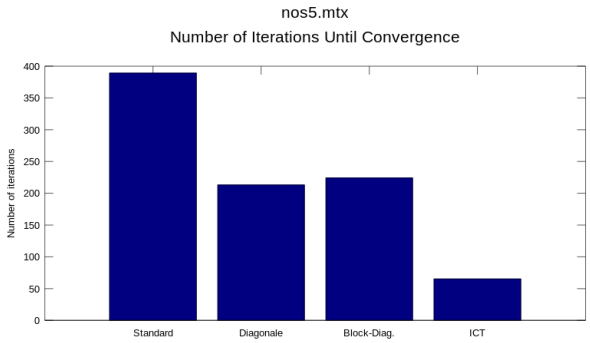


(b) `nos6.mtx`, relative residual history.

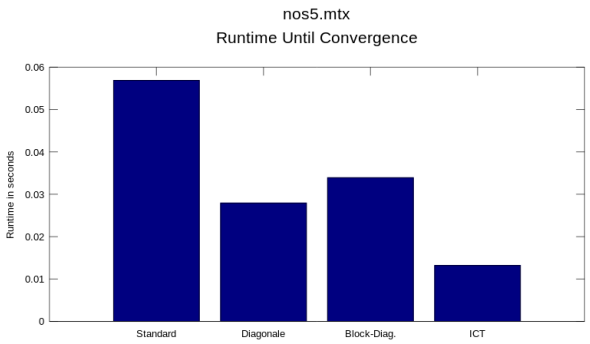


(c) `s3rmt3m3.mtx`, relative residual history.

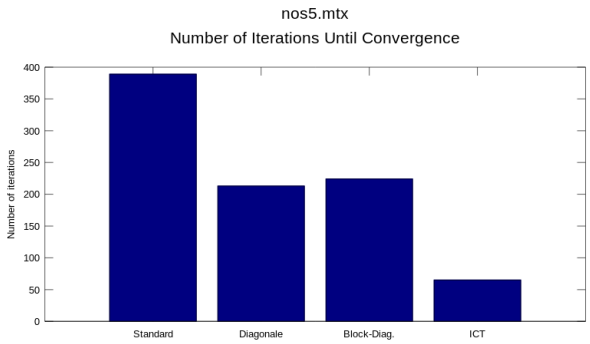
Figure 3: Relative residual histories until convergence with a relative residual of 10^{-4} for different preconditioning methods and standard Conjugate Gradient.



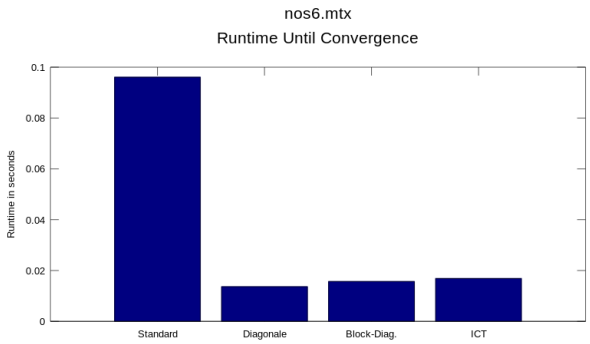
(a) `nos5.mtx`, num. of iterations.



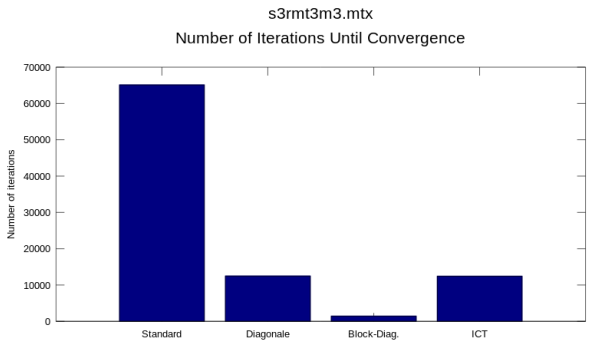
(b) `nos5.mtx`, run time.



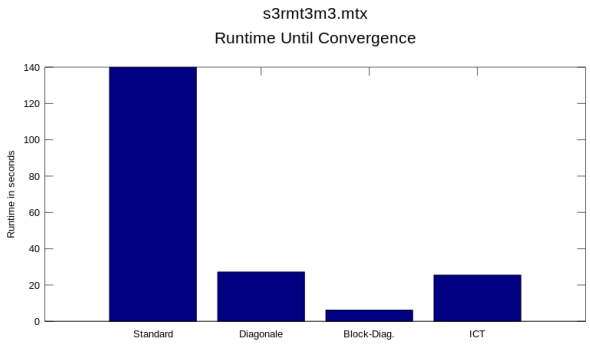
(c) `nos6.mtx`, num. of iterations.



(d) `nos6.mtx`, run time.

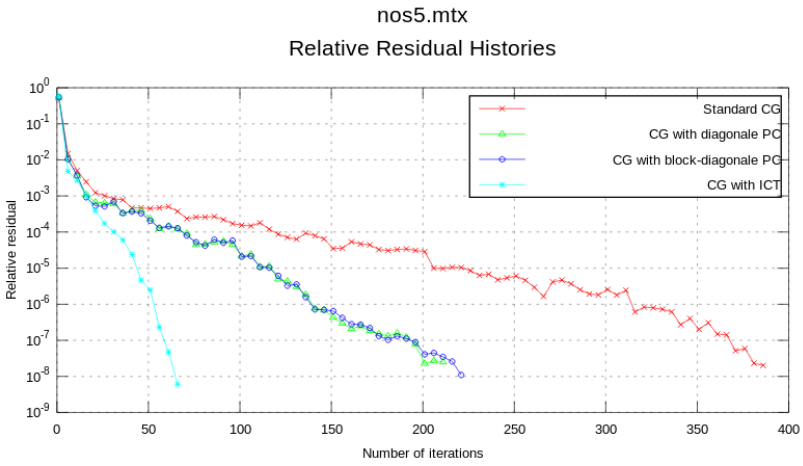


(e) `s3rmt3m3.mtx`, num. of iterations.

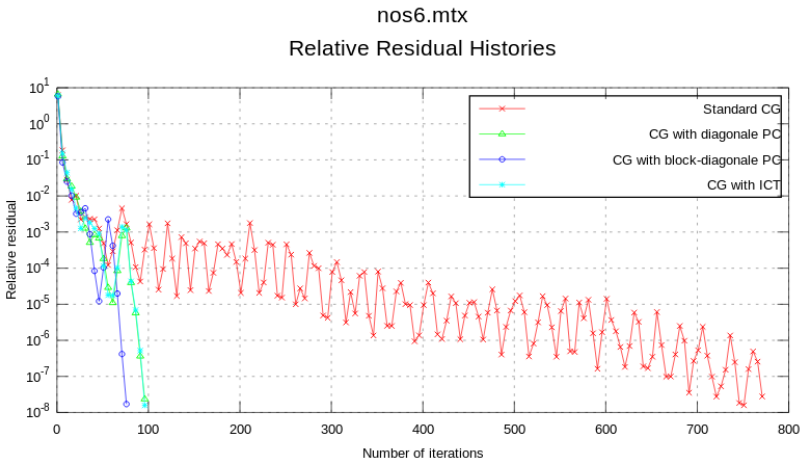


(f) `s3rmt3m3.mtx`, run time.

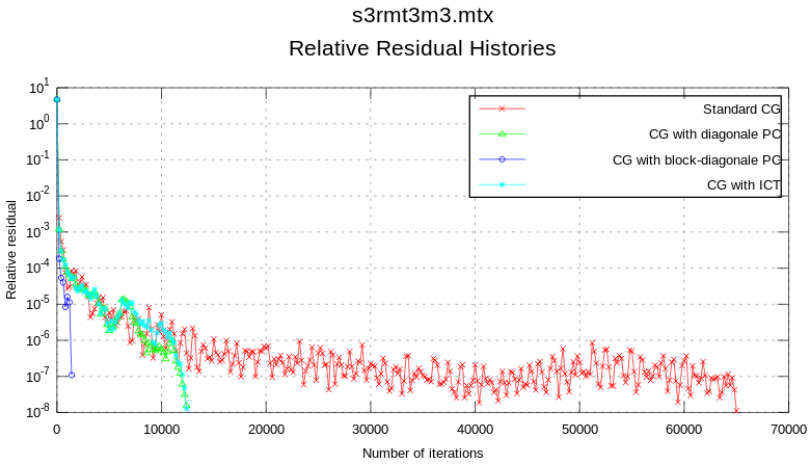
Figure 4: A comparison of numbers of iterations, run times and relative residual histories until convergence with a relative residual of 10^{-8} for different preconditioning methods (no $IC(0)$) and standard Conjugate Gradient.



(a) `nos5.mtx`, relative residual history.



(b) `nos6.mtx`, relative residual history.



(c) `s3rmt3m3.mtx`, relative residual history.

Figure 5: Relative residual histories until convergence with a relative residual of 10^{-8} for different preconditioning methods (no $IC(0)$) and standard Conjugate Gradient.