European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2006
P. Wesseling, E. Oñate and J. Périaux (Eds)

ACCURACY AND PERFORMANCE ISSUES OF SPECTRAL PRECONDITIONERS IN SEMICONDUCTOR DEVICE SIMULATION

S. Röllin*, W. Fichtner*

*Integrated Systems Laboratory, ETH Zurich, 8092 Zurich, Switzerland
e-mail: roellin@iis.ee.ethz.ch

Key words: iterative solvers, spectral preconditioners, semiconductor device simulation

Abstract. *In most numerical simulations in computational science and engineering, a preconditioner is mandatory to iteratively solve large sparse linear systems. In theory, a good preconditioner clusters all eigenvalues of the iteration matrix around one. However, in practice, there may still be a few outliers. In particular, small eigenvalues deteriorate the convergence speed and may affect the ultimate accuracy. Spectral preconditioners can be used to circumvent these problems. They are constructed with the help of an invariant subspace that contains the eigenvectors corresponding to the small eigenvalues. In this paper, we investigate spectral preconditioners in the field of semiconductor device simulation. We look into different formulations and their influence on the accuracy. Performance issues of spectral preconditioners are the second topic we investigate.*

1 INTRODUCTION

The solution of sparse linear systems is an important task and often the most crucial part in numerical simulations in computational science and engineering. Simulations become more and more important in many fields such as fluid dynamics, chemistry, and electronics. The focus of this paper lies on the iterative solution of the linear systems originating in semiconductor device simulation. In this field, there is a steadily growing demand to carry out large 3D simulations. Only advanced device structures in 3D combined with a fine simulation mesh are able to describe the complex devices and to resolve the relevant physical effects. Such simulations lead to huge sparse linear systems. While direct solvers deliver excellent performance combined with accurate solutions, only iterative solvers are a viable alternative for large 3D simulations due to time and memory constraints.

The investigation of linear systems arising from various semiconductor device problems has revealed that the corresponding matrices might have a few very small eigenvalues, adversely affecting the convergence behaviour of an iterative method. Moreover, they may also lower the accuracy of the iterative solution.

A good preconditioner is usually able to cluster all eigenvalues around one. Unfortunately, we have encountered situations where the preconditioned systems still have a few very small eigenvalues.

It has been shown that spectral preconditioners significantly enhance iterative solvers.[1] Using such a preconditioner markedly increases accuracy and robustness and leads to an iterative solver that is comparable to a direct one in terms of accuracy. The idea of spectral preconditioners is to support an existing preconditioner. They are particularly beneficial, if the preconditioned system has a few small eigenvalues. The corresponding eigenvectors are used to construct a supplemental preconditioner. Unfortunately, the construction of the spectral preconditioners is quite expensive, as the eigenvectors corresponding to the smallest eigenvalues of the iteration matrix have to be available.

The focus of this paper is on two different aspects of spectral preconditioners. First, we will compare different formulations regarding their accuracy and suitability. The second aspect is concerned with the performance of the preconditioners if applied to semiconductor device simulations.

The papers is organised as follows. In the next section, we will introduce spectral preconditioners. Section 3 presents our numerical results. In the last section, we draw some conclusions.

2 SPECTRAL PRECONDITIONERS

In this section, we introduce different kinds of spectral preconditioners. As stated in the introduction, they are beneficial if the iteration matrix has some small eigenvalues. The idea is to shift these small eigenvalues. We can shift them either to zero, to one or into the vicinity of one. This paper mainly focuses on the first possibility, but we also present the other ideas for the sake of completeness. We first introduce some notation.

We are interested in iteratively solving linear systems, i.e.

$$Ax = b, \qquad A \in \mathbb{R}^{n\times n}, \qquad b, x \in \mathbb{R}^n. \tag{1}$$

In many applications, the matrix A is ill-conditioned and, therefore, the linear system is hard to treat with iterative solvers. As a remedy, system (1) is replaced by a *preconditioned* system, i.e.,

$$MAx = Mb \tag{2}$$

that is easier to solve. Here, M is a linear operator, and, since it is applied from the left, it is called a *left preconditioner*. Finding a suitable preconditioner M is of utmost significance for an iterative solver. In general, we require that M is a good approximation of A or, in other words, $MA \approx I$. Furthermore, the construction as well as the application of M should not be too expensive. An example for a very efficient preconditioner is an incomplete LU-factorisation. In this case $M = (LU)^{-1}$ with $LU \approx A$, whereas L and U are lower and upper triangular matrices.

The preconditioner can also be applied from the right — compared to multiplying the given system from the left hand side:

$$AM\bar{x} = b, \qquad x = M\bar{x}. \tag{3}$$

In the following, it is not important whether we have a left or right preconditioned systems or even an unpreconditioned one. Therefore, we use $\widetilde{A}\widetilde{x} = \widetilde{b}$ as an arbitrary linear system in the remainder. In practice, the three cases below are the most important situations:

1. *without preconditioner* $\widetilde{A} := A$ $\widetilde{b} := b$
2. *left preconditioner* $\widetilde{A} := MA$ $\widetilde{b} := Mb$
3. *right preconditioner* $\widetilde{A} := AM$ $\widetilde{b} := b$

We only require that $\widetilde{A}$ is nonsingular.

A suitable preconditioner M is usually able to cluster most of the eigenvalues around one. But even in this case, the iteration matrix $\widetilde{A}$ may have some very small eigenvalues, strongly deteriorating the speed of convergence of an iterative method and lowering the accuracy of the solution. The associated eigenvectors can be used to accelerate the convergence. We denote with $\mathcal{V}$ an invariant subspace of the matrix $\widetilde{A}$ containing the eigenvectors.

Let the matrix $Z \in \mathbb{R}^{n,k}$ be a basis of the invariant subspace $\mathcal{V}$ of $\widetilde{A}$ with $\dim(\mathcal{V}) = k$ (or to be more precise: the columns of Z form a basis). Due to the invariant subspace, we can write $\widetilde{A}Z = ZD$, and it follows from the nonsingularity of $\widetilde{A}$ that D is nonsingular, too. We can extend the basis Z with vectors $\bar{Z}$ such that $[Z\ \bar{Z}]$ is a full basis of $\mathbb{R}^n$. Thus, we can write

$$\widetilde{A}[\,Z\ \ \bar{Z}\,] = [\,Z\ \ \bar{Z}\,] \begin{bmatrix} D & E \\ 0 & \bar{D} \end{bmatrix}, \tag{4}$$

where the matrix $\bar{D}$ is nonsingular due to the nonsingularity of $\widetilde{A}$.

2.1 Deflation-based preconditioners

The preconditioners presented in this part go back to Nicolaides[2] who came up with the idea of deflating the Conjugate Gradient method to accelerate its convergence. While Nicolaides idea was restricted to symmetric positive definite systems, Frank and Vuik[3] generalised it and introduced the so-called *deflation-based preconditioners*. We will define these preconditioners next.

In order to solve the linear system $\widetilde{A}\widetilde{x} = \widetilde{b}$, we define the projectors

$$P_0 := I - \widetilde{A}ZA_c^{-1}Y^T, \qquad Q := I - ZA_c^{-1}Y^T\widetilde{A}, \tag{5}$$

where Y and $Z \in \mathbb{R}^{n,k}$ and $A_c := Y^T\widetilde{A}Z$. For the moment, Y is arbitrary except that A_c is nonsingular. The projector P_0 is singular and thus we cannot use it as a normal

preconditioner. Hence, two steps are required in order to compute the solution of the linear system $\widetilde{A}\widetilde{x} = \widetilde{b}$. First, we solve

$$P_0\widetilde{A}\bar{x} = P_0\widetilde{b} \tag{6}$$

for $\bar{x}$ and compute the solution

$$\widetilde{x} = ZA_c^{-1}Y^T\widetilde{b} + Q\bar{x}. \tag{7}$$

In the case of a right preconditioner we finally set $x = M\widetilde{x}$ and $x = \widetilde{x}$ otherwise to get the solution x of the original linear system $Ax = b$.

It is straightforward to show that P_0 shifts the eigenvalues corresponding to the invariant subspace $\mathcal{V}$ to zero, or in other words, that the spectrum of $P_0\widetilde{A}$ is $\{0, \lambda(\bar{D})\}$. The zero eigenvalue has multiplicity k. This also explains the "0" in the notation of the operator P_0. It is not obvious that System (6) has a solution, since the linear operator $P_0\widetilde{A}$ is singular. But this is not a problem, due to the fact that P_0 is applied on both sides; for further information we refer to another work of the authors.[1]

The construction of P_0 itself is not expensive except the computation of the invariant subspaces Y and Z. We will come back to this issue in the section with the numerical experiments. However, the application of the projector P_0 is expensive if implemented naively. One step of an iterative method becomes roughly twice as expensive in terms of multiplications with $\widetilde{A}$, if $P_0\widetilde{A}$ is the iteration matrix, since $\widetilde{A}$ appears in (5). One remedy is to compute the vectors $W := \widetilde{A}Z$ during an initialisation step and to write $P_0 = I - WA_c^{-1}Y^T$. Another possibility is to rewrite P_0 as $P_0 = I - Z(Y^TZ)^{-1}Y^T$. The former choice is more expensive both regarding computing time and memory requirements, since we have to compute and store the vectors W. Nevertheless, it has some advantages as we will see later on.

2.2 Two-level preconditioners

While the preconditioners presented in the previous section shift the eigenvalues of a given subspace to zero, this section is devoted to preconditioners that shift the eigenvalues to one or into the neighbourhood of one. We use the same conventions for $\widetilde{A}$ and $\widetilde{b}$ as before.

Let us define the two different two-level spectral preconditioners:

$$P_1 := I + Z(I - D)A_c^{-1}Y^T \tag{8}$$

$$P_{1\bullet} := I + ZA_c^{-1}Y^T \tag{9}$$

The name "P_1" was chosen to indicate that some eigenvalues are shifted to one. The notation "$P_{1\bullet}$" expresses that the operator shifts some eigenvalues into the neighbourhood of one. To be more precise, the spectrum of $P_1\widetilde{A}$ is $\{1, \lambda(\bar{D})\}$ and the eigenvalue one appears k times. Therefore, we conclude that P_1 is nonsingular. The spectrum of $P_{1\bullet}$ is

$\{1 + \lambda(D), \lambda(\bar{D})\}$. This preconditioner is only nonsingular, if $-1 \notin \lambda(D)$, but since the small eigenvalues are usually shifted, this is not an issue.

Both preconditioners can be used as a right preconditioner for $\widetilde{A}$ as well as a left preconditioner. Since we do not use these preconditioners in the following and report only results for P_0, we refer the readers for more information to the papers of Carpentieri et al.[4] and the authors[1] of the paper at hand.

2.3 Choice of Y and Z

The choice of the vectors Y and Z is important. We would like to shift the small eigenvalues of $\widetilde{A}$ and hence, we choose the columns of Z as a basis of the subspace corresponding to the same small eigenvalues. The vectors Y can be chosen in different ways. The only restriction is that A_c is nonsingular. Two possibilities come to mind:

1. set $Y = Z$,
2. choose Y as the eigenvectors of $\widetilde{A}^T$ corresponding to the small eigenvalues. We refer to this case as $Y \neq Z$.

Both versions have their advantages, but the first one is much cheaper to compute, since only the right eigenvectors are required in comparison to the second, for which also the left eigenvectors are needed. The computation of the eigenvectors is quite expensive as we will see in the next section. Furthermore, the second choice requires the transpose $\widetilde{A}^T$. This operation may not be available or is hard to parallelise, e.g. the transpose of the preconditioner M. The memory requirements are also higher for the latter.

It is not obvious how to compute the basis Z, especially if $\widetilde{A}$ is a product of linear operators. It turns out that ARPACK[5] can be used to compute the desired extremal eigenvalues as suggested by Carpentieri et al.[4]

3 NUMERICAL RESULTS

In this section, we present some numerical experiments with the spectral preconditioner P_0 for problems arising in semiconductor device simulations. We will show how we can improve this preconditioner. During a semiconductor device simulation, several nonlinear systems with up to six equations have to be solved. These are linearised with the Newton algorithm and thus one typically has to solve dozens of linear systems during one simulation. If the linear systems are not solved accurately enough, the Newton iteration either fails or requires more steps to converge. In the first case, the time step is reduced. We say that the simulation fails, if the time step becomes too small.

The iterative solver we use to generate the results consists of several parts. First, it reorders and scales the matrices with MPS, which is very beneficial in this field.[6] Next, the matrix is reordered using the Nested Dissection algorithm. After that, we compute the preconditioner M, a threshold-based incomplete LU-factorisation with a drop-tolerance of 10^{-3} (used as left preconditioner). Then the necessary information for the spectral

Table 1: Multi-dimensional (2D and 3D) numerical device simulation experiments. Unknowns designate the size of the linear systems.

Simulation	Unknowns	Dim.
Igbt	11'010	2D
SiPbhLaser	14'086	2D
Nmos	18'627	2D
DRAM	32'040	3D
ImSensor	38'415	3D
Barrier2	115'625	3D
Para	155'924	3D
Ohne	183'038	3D
Resistivity	318'026	3D

preconditioner P_0 is generated by using ARPACK. Finally, we use the iterative method BiCGStab. The iteration is stopped either if the preconditioned residual is reduced eight orders of magnitude or if 200 iteration steps are carried out.

Nine different simulations from both 2D and 3D are used to test the preconditioners. They are listed in Table 1. The number of unknowns ranges from about ten thousand up to 320'000. An iterative solver often allows to adjust a broad range of parameters, e.g. choice of preconditioner, drop tolerances, orderings, et cetera. We are looking for parameters that perform well for all simulations and not only for one matrix or one simulation. Therefore, it is important that a novel strategy does not lead to a regression on the average.

3.1 Choice of Y and Z

In the previous section, we saw that there are basically two different choices for the vectors Y. One choice is to set them equal to the right eigenvectors Z. The second possibility is to compute Y as the left eigenvectors of $\widetilde{A}$. It seems that this latter choice has only disadvantages. However, it also has its usefulness. The reason is depicted in Figure 1 in which the relative error of the matrix DRAM-18 (of the original, unpreconditioned linear system) is shown for both choices of Y. The error is much smaller for $Y \neq Z$. In order to explain the differences, we remember that the n-th approximate solution of the linear system $\widetilde{A}\widetilde{x} = \widetilde{b}$ is the following sum:

$$\widetilde{x}_n = \underbrace{Z A_c^{-1} Y^T \widetilde{b}}_{x^1 :=} + \underbrace{Q \bar{x}_n}_{x_n^2 :=}. \tag{10}$$

The differences of the relative error come from the first term. For $Y = Z$, we have $x^1 = 2.1 \cdot 10^{12}$, whereas for $Y \neq Z$ the term is $1.6 \cdot 10^5$. In comparison, the 2-norm of the solution is $1.7 \cdot 10^5$. As a result, the initial relative error is much larger for $Y = Z$ and it is obvious that in this case, the second term x_n^2 must be much more accurate to obtain the same accuracy as for $Y \neq Z$. We therefore have to choose smaller tolerances and permit more iteration steps for the former case.

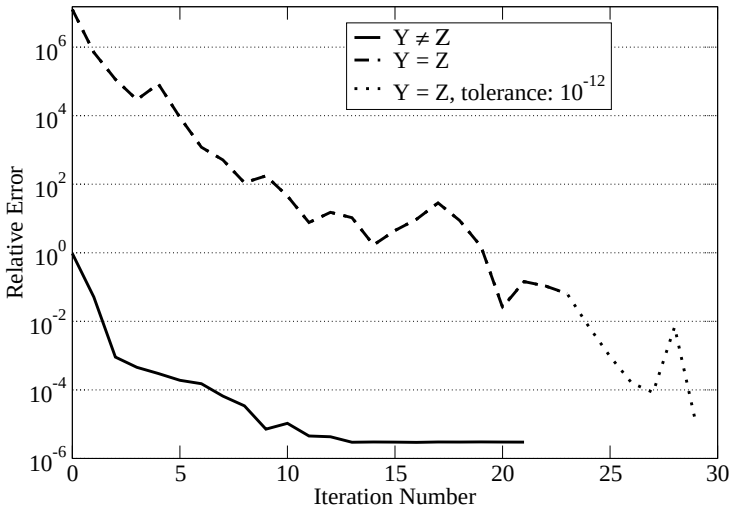

Figure 1: Relative error $\|\widetilde{x} - \widetilde{x}_n\|/\|\widetilde{x}\|$ for two different choices for the vectors Y. In both cases, the iteration is stopped when the relative preconditioned residual is reduced eight orders of magnitude. If left and right eigenvectors are used, the solution is much more accurate at this point. If the iteration is stopped later for $Y = Z$, the accuracy is improved significantly (dotted curve).

Let us analyse the error further. We choose $\bar{x}_0 = 0$ as initial guess in BiCGStab. Therefore, the corresponding error is

$$e_0 = \widetilde{x} - \widetilde{x}_0 = \widetilde{x} - ZA_c^{-1}Y^T\widetilde{b} = \widetilde{x} - (I - Q)\widetilde{x} = Q\widetilde{x}. \tag{11}$$

We see that if the solution $\widetilde{x}$ is a linear combination of the basis Z, the initial error is zero due to $QZ = 0$. On the other hand, let us assume that the solution $\widetilde{x}$ is a linear combination of the eigenvectors of $\widetilde{A}$ not in Z. Then, it is easy to show that $Q\widetilde{x} = \widetilde{x}$ for Y the left eigenvectors. As a result, the initial error has the same norm as the solution itself. However, for $Y = Z$, the norm of $Q\widetilde{x}$ can be much larger than the norm of $\widetilde{x}$. This explains the observation shown above.

From these results, it seems that we should use the left and right eigenvectors, since this case is more accurate and requires fewer iteration steps. However, we achieve the same accuracy with $Y = Z$ by stopping the iteration later. The additional iteration steps are by far less time-consuming than computing the left eigenvectors Y. Therefore, we do not report on $Y \neq Z$ in the following but use $Y = Z$ with the smaller tolerance 10^{-12} instead of 10^{-8} in BiCGStab for the preconditioned residuals.

3.2 Accuracy

In the previous section, we have indicated that there are different equivalent formulations for P_0. Here, we will compare the following four versions and report some results with them:

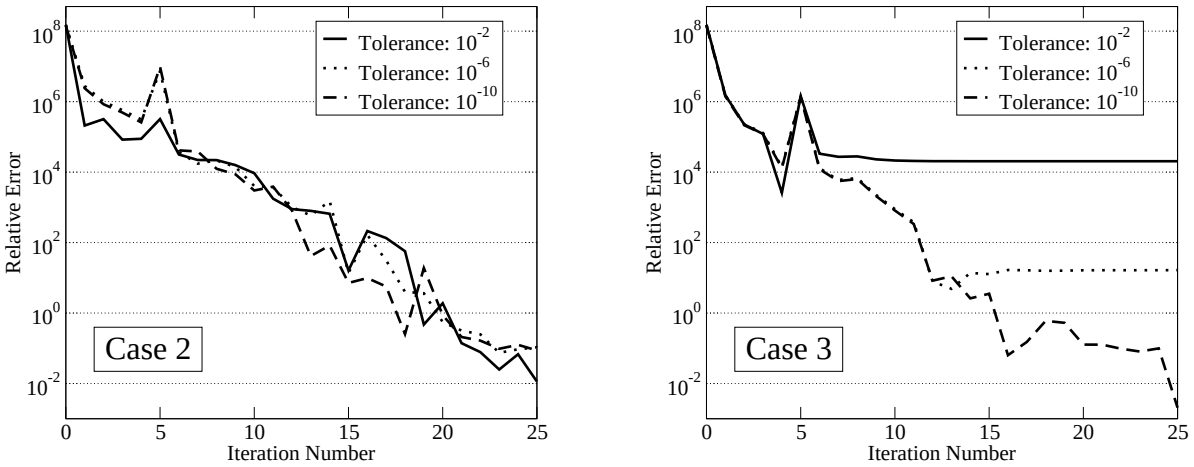

Figure 2: Relative error for different accuracies of the eigenvectors and eigenvalues for matrix Ohne-10. On the left hand side, the results for the second formulation are shown and those on the right correspond to the third version. Results for the last formulation have been omitted, since they are very similar to the third one.

1. $P_0 = I - \widetilde{A}ZA_c^{-1}Y^T \qquad Q = I - ZA_c^{-1}Y^T$
2. $P_0 = I - WA_c^{-1}Y^T \qquad Q = I - ZA_c^{-1}Y^T, \quad$ where $W = \widetilde{A}Z$
3. $P_0 = I - Z(Y^TZ)^{-1}Y^T \quad Q = I - ZA_c^{-1}Y^T$
4. $P_0 = I - Z(Y^TZ)^{-1}Y^T \quad Q = I - ZD^{-1}(Y^TZ)^{-1}Y^T$

The first version is the original version introduced by Frank and Vuik. We use it for theoretical purposes only, since one has to multiply with $\widetilde{A}$, making the application of P_0 unnecessary expensive. We therefore omit results for it. The second formulation requires more memory than the others, since beside the vectors Z, one has to store $W = \widetilde{A}Z$. The last two versions are similar and differ only in the operator Q. The latter is slightly less expensive to construct because $A_c = Y\widetilde{A}Z$ is not required.

The four formulations differ only due to finite precision arithmetic. Giraud and Gratton[7] have shown in a similar context that inexact eigeninformation leads to a degradation of the performance of spectral preconditioners. But the accuracy of the iterative solver is also affected. In order to show this effect, we computed the eigenvectors with different accuracies. Namely, we used different tolerances in ARPACK, ranging from 10^{-2} to 10^{-10}. With each set of eigenvectors, we solved a linear system and plotted the residuals as well as the errors, see Figure 2. The convergence curves slightly differ for different tolerances for the second formulation, but the ultimate accuracy of the iterative solution is not affected by less accurate eigenvectors. This is not true for the third and fourth formulation: only very accurate eigeninformation leads to an accurate solution. If the wrong tolerance is used for ARPACK, the ultimate accuracy is not good enough. In this case, the second formulation is much more robust than the last two formulations.

The differences between the different formulations of P_0 for matrices from other simulations are not as large as for the matrices from the simulation Ohne. However, version two

Simulation	Version 2	Version 3	Version 4
Igbt	**88**	121	151
SiPbhLaser	**541**	582	577
Nmos	148	167	**138**
DRAM	**410**	445	412
ImSensor	**272**	352	328
Barrier2	**295**	313	318
Para	**116**	139	127
Ohne	**455**	†	†
Resistivity	**86**	133	133

Table 2: Number of linear systems to complete semiconductor device simulations for three different formulations of the spectral preconditioner P_0 and Q. A tolerance of 10^{-2} was used to compute the eigenvectors with ARPACK. The numbers in boldface indicate the best results. In two cases, the simulation stops due to a too small step size, shown with "†".

still delivers the best results for complete semiconductor device simulations. In Table 2, the last three formulations for P_0 and Q are compared. In order to clearly see the differences, we have chosen a large tolerance (10^{-2}) for the computation of the eigenvectors. The number of linear systems to complete a simulation is an indication for the accuracy of the spectral preconditioners. On average, the second formulation leads to fewer linear systems. Furthermore, it is for all but one simulation the best choice and the only one that is able to solve all simulations. In the remainder, we use the tolerance 10^{-6} to compute the eigenvectors.

3.3 Reusing eigeninformation

We have seen that eigeninformation must be available in order to define the spectral preconditioners. Unfortunately, computing the eigenvectors is very time consuming. For the simulations used in this paper, between 55% and 85% of the time spent in the linear solver is used for the computation of the eigenvectors Z. Or in other words, most of the time is spent in ARPACK in our implementation. If we would compute also the left eigenvectors in addition to the right eigenvectors, this part would weight even more.

In this section, we therefore analyse the effectiveness of some simple techniques to reduce this part of the solver. If we look at the eigenvalues and eigenvectors of consecutive linear systems during a simulation, we see that these quantities often change only slightly if at all. We will use this observation in the following.

In ARPACK, it is possible to either use a random vector or to use a user-defined vector as starting vector for the Arnoldi method. We expect that a linear combination of the eigenvectors from the previous linear system is a better choice as starting vector than a random vector. And indeed, this is what we observe in practice. In Table 3 we have listed the average number of iterations spent in ARPACK to compute the eigenvectors for complete semiconductor device simulations. We use the sum of the eigenvectors from

Simulation	random	previous
Igbt	58	**50**
SiPbhLaser	310	**223**
Nmos	77	**61**
DRAM	237	**218**
ImSensor	**91**	144
Barrier2	509	**431**
Para	424	**399**
Ohne	224	**217**
Resistivity	135	**102**

Table 3: Average number of iterations to compute the eigenvectors with ARPACK for two different starting vectors: a *random* vector or the sum of the eigenvectors from the *previous* linear system. The numbers in boldface show the version that requires the fewest iterations.

the previous simulation as starting vector. Except for one simulation, the number of iterations is lower, if an appropriate starting vector is used compared to a random one. The average number of iterations in BiCGStab changes only slightly if at all, so that there are no disadvantages by using the proposed starting vector.

There is another possibility to reuse the eigenvectors. As stated above, the eigenvectors change only slightly from one linear system to the next. Therefore, a strategy is to compute the eigenvectors for a linear system and then to use these vectors for the next couple of linear systems. One can imagine several strategies how long the eigenvectors should be kept. We have tested a simple method. The idea is to compute the eigenvectors for the first linear system in a Newton step and to keep these vectors for the remaining linear systems of the Newton step. On the one hand, this significantly reduces the number of ARPACK iterations, but on the other hand, we have to accept that more iterations in BiCGStab will be required to solve the linear systems on average. This is due to the fact that only approximate eigenvectors are used to construct the spectral preconditioner P_0. As a rough measure for the computational work, we take the number of matrix-vector products used to compute the eigenvectors and the solution with BiCGStab. In Table 4, we have listed the results. For more than half of the simulations, more linear systems have to be solved to complete the simulation, if the eigenvectors are not computed for every system, so it seems that there is more work to carry out. On the other hand, the total number of matrix-vector products drops significantly in all but one simulation. The number of matrix-vector products is reduced between 11% and 37%. However, the proposed strategy to reuse the eigenvectors is not suited at all for simulation DRAM for which there is a tenfold increase in the number of linear systems. As a summary, we conclude that reusing the eigenvectors is beneficial regarding performance but is not reliable enough, since it lowers the accuracy of the iterative solver.

Simulation	Linear systems		Matrix-vector products	
	-	reuse first	-	reuse first
Igbt	89	**88**	8451	**6855**
SiPbhLaser	**541**	**541**	190493	**132348**
Nmos	**148**	**148**	16474	**9052**
DRAM	**424**	4386	**124616**	1476833
ImSensor	**284**	288	59658	**52814**
Barrier2	**289**	328	157190	**133872**
Para	**108**	123	50688	**40498**
Ohne	**457**	477	150387	**115434**
Resistivity	**85**	**85**	13868	**8784**

Table 4: Linear systems and total number of matrix-vector products to complete semiconductor device simulations. The numbers in the third and fifth column correspond to the case for which the first eigenvectors within a Newton step are reused.

3.4 Detecting stagnation

The spectral preconditioners significantly improve the convergence of iterative methods. As mentioned above, a good part of the simulation time is spent to compute the eigenvectors. The second time-consuming part is the iteration itself. If we analyse simulation DRAM and look at the number of iterations to reduce the preconditioned residual by 10^{-12}, we see that BiCGStab requires between 20 and 40 iterations for most linear systems. However, for this simulation, there occur a couple of linear systems that lead to 200 iteration steps, the maximum number of iterations. An example of a corresponding convergence curve is given in Figure 3. We see that after 25 iteration steps, the preconditioned residual is reduced almost eleven orders of magnitude and then stagnates. As a result, the remaining iteration steps do not improve the solution anymore, but unnecessarily increase the simulation time.

Therefore, we propose to detect the stagnation of the preconditioned residual in order to reduce the number of (unnecessary) iterations. We say that the iteration stagnates, if the maximum difference of k consecutive residuals divided by the minimum of those residuals is smaller than a given tolerance tol. Or, in other words, we stop at iteration i, if

$$\frac{\max_{j\in S_i^k} \|r_j\| - r_i^{min}}{r_i^{min}} \leq tol, \qquad r_i^{min} := \min_{j\in S_i^k} \|r_j\|, \qquad S_i^k := \{i-k+1,\dots,i\}.$$

Figure 3 shows that quite a number of iteration steps can be avoided with this technique. The parameters used were $k = 10$ and $tol = 10^{-2}$.

Next, we would like to see if it is worth to detect stagnation during a simulation and how much can be saved. Also, we have to ensure that the accuracy of the iterative solver is not hurt by this idea. In order to avoid to stop the iteration too early, we only

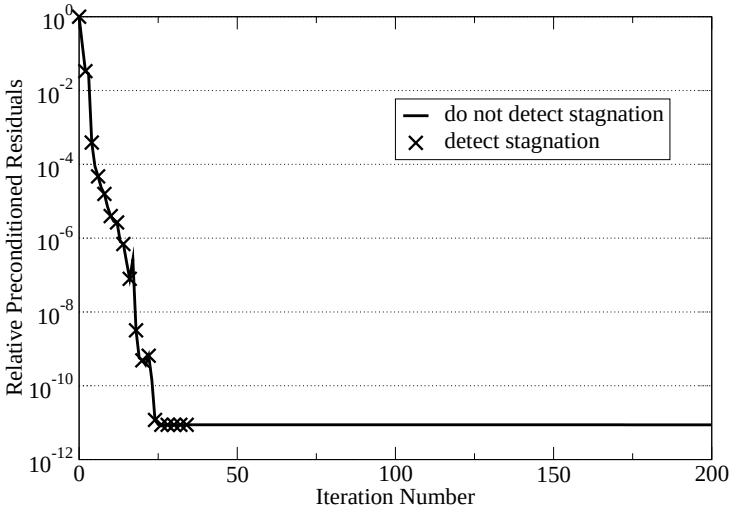

Figure 3: Convergence behaviour of relative preconditioned residual for matrix DRAM-76.

Simulation	without	with
Igbt	22	**12**
SiPbhLaser	**64**	65
Nmos	24	**13**
DRAM	37	**33**
ImSensor	32	**29**
Barrier2	**56**	**56**
Para	**34**	**34**
Ohne	55	**43**
Resistivity	30	**26**

Table 5: Average number of BiCGStab iterations for a complete semiconductor device simulation *with* and *without* detecting stagnation.

check stagnation, if the preconditioned residual is reduced ten orders of magnitude. The results are shown in Table 5. For most simulations, the average number of iterations is reduced. The number of linear systems required to finish the simulation, as well as the average iterations used to compute the eigenvectors does not change. Therefore, for most simulations, the overall computational work is reduced. An exception is the simulation DRAM, for which slightly more linear systems are required, if stagnation of the residuals is detected and therefore the total number of matrix-vector products also increases slightly.

4 CONCLUSIONS

In this paper we have analysed accuracy and performance issues of spectral preconditioners. The focus of the numerical results was on semiconductor device simulations. We

have shown that there are equivalent formulations for the spectral preconditioner P_0 that lead to different results in finite precision arithmetic. The accuracy of eigenvectors has a direct influence on the accuracy of the solution of the iterative solver. We have also shown that one formulation is superior to the others, in the sense that it is less dependent on the accuracy of the eigenvectors.

The computation of the eigenvectors makes up a good part of the time spent in the iterative solver. We have analysed some techniques to reduce this computation. Despite some reduction, it is still the most time consuming part.

REFERENCES

[1] S. Röllin and W. Fichtner. About spectral preconditioners and their application in semiconductor device simulation. Technical Report 19/2005, Integrated Systems Laboratory, ETH Zurich, Switzerland, 2005.

[2] R. A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM J. Numer. Anal.*, 24(2):355–365, April 1987.

[3] J. Frank and C. Vuik. On the construction of deflation-based preconditioners. *SIAM J. Sci. Comput.*, 23(2):442–462, 2001.

[4] B. Carpentieri, I. S. Duff, and L. Giraud. A class of spectral two-level preconditioners. *SIAM J. Sci. Comput.*, 25(2):749–765, 2003.

[5] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, 1998.

[6] O. Schenk, S. Röllin, and A. Gupta. The effects of unsymmetric matrix permutations and scalings in semiconductor device and circuit simulation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 23(3):400–411, March 2004.

[7] L. Giraud and S. Gratton. On the sensitivity of some spectral preconditioners. *SIAM J. Matrix Anal. Appl.*, 27(4):1089–1105, 2006.