Chapter 6
Recycling Krylov Subspaces for Solving Linear Systems with Successively Changing Right-hand Sides Arising in Model Reduction

Peter Benner and Lihong Feng

Abstract We discuss the numerical solution of successive linear systems of equations $Ax = b_i,\ i = 1, 2, \ldots, m$, by iterative methods based on recycling Krylov subspaces. We propose various recycling algorithms which are based on the generalized conjugate residual (GCR) method. The recycling algorithms reuse the descent vectors computed while solving the previous linear systems $Ax = b_j$, $j = 1, 2, \ldots, i-1$, such that a lot of computational work can be saved when solving the current system $Ax = b_i$. The proposed algorithms are robust for solving sequences of linear systems arising in circuit simulation. Sequences of linear systems need to be solved, e.g., in model order reduction (MOR) for systems with many terminals. Numerical experiments illustrate the efficiency and robustness of the proposed method.

Keywords Linear systems of equations · GCR · GMRES · Recycling Krylov subspace

6.1 Introduction

In some fields of circuit design, such as macromodeling of linear networks with many terminals [2], noise analysis of RF circuits [19] etc., successive linear systems of equations $A_i x = b_i, i = 1, 2, \ldots, m$, have to be solved. Here the dimension

P. Benner
Professur Mathematik in Industrie und Technik, Fakultät für Mathematik, Technische Universität Chemnitz, Reichenhainer Str. 41, 01926 Chemnitz, Germany
e-mail: benner@mathematik.tu-chemnitz.de

L. Feng (✉)
Institut für Mikrosystemtechnik, Universität Freiburg, Freiburg, Germany
e-mail: lihong.feng@mathematik.tu-chemnitz.de

© Springer Science+Business Media B.V. 2011

n of the unknown vector $x \in \mathbb{R}^n$ is very large. In general, the A_i, $i = 1, 2, \ldots, m$, form a set of different, nonsymmetric matrices. In some applications [2], only the right-hand sides b_i vary while $A_i \equiv A$ is fixed, which can and should be exploited. In this paper we focus on the latter case, i.e.

$$Ax = b_i, \quad i = 1, 2, \ldots, m. \tag{6.1}$$

If a direct method like a sparse LU or Cholesky decomposition is applicable, then the solution of the linear systems (6.1) can be obtained by sharing the LU/Cholesky decomposition of A and using successive forward/backward solves. Whether or not a sparse direct solver is applicable depends on the size of the matrix, its sparsity pattern and adjacency graph which is highly application-dependent. For details see [3]. Here, we are interested in cases where the coefficient matrix A does not allow the efficient application of a sparse direct solver. In this situation, iterative methods are to be used. For this purpose, any standard solver can be applied, e.g., depending on the structure of A, the generalized conjugate residual (GCR) method, the generalized minimized residual (GMRES) method, the quasi minimal residual method (QMR), the preconditioned conjugate gradient (PCG) method are possible choices (see, e.g., [17] for hints on the choice of method). Doing this in a naive way, the chosen method is started anew for each right-hand side vector b_i. Since the linear systems of equations in (6.1) share the same coefficient matrix A, we try to exploit the information derived from the previous system's solution to speed up the solution of the next system. Recycling is one possible choice here which turns out to be a suitable approach for the applications in MOR we have in mind. Due to a quite easy mechanism for recycling, we focus here on GCR. That is, the recycling algorithms proposed in this paper reuse the descent vectors in GCR for the solution of the previous systems as the descent vectors for the solution of the current system. This way, the computation of many matrix–vector products can be saved as compared to restarting GCR for each subsystem. The algorithm has the same convergence property as the standard GCR method, that is: for each system, in exact arithmetic the approximate solution computed at each iteration step minimizes the residual in the affine subspace comprised of initial guess and descent vectors.

Note that very recently, other Krylov subspace methods have been suggested for recycling. We comment on GMRES-based recycling in Sect. 6.4 and have further investigated this in [9]. CGS- and BiCGStab-recycling is suggested in [1], but the investigation of these new variants for our purposes is beyond the scope of this paper.

However, the number of the recycled descent vectors may become too large if there are many systems in the sequence. If the size of A is very large, then storing and computing all the descent vectors requires very large memory and also slows down the speed of computation due to orthogonalization needs. Therefore, we propose a restarting technique which limits the number of recycled descent vectors so that the required workspace can be fixed a priori.

Conventional block iterative methods (see [12, 18] and references therein) cannot be applied to (6.1) if the b_i are supplied successively (like in the MOR applications to be discussed), since these methods require that the right-hand sides are available simultaneously. A new method for solving systems $A_i x_i = b_i$ is proposed in [16], where the focus is on varying A_i. This approach turns out to be quite involved for systems with the same coefficient matrix A as in (6.1). The proposed algorithms are simpler and easier to implement.

In the next section, we propose various recycling algorithms based on standard GCR. We discuss the convergence properties of these algorithms. In Sect. 6.3, we review the MOR method from [2] for circuits with massive inputs and outputs. We show where sequences of linear systems as in (6.1) arise in this context and how the recycling algorithms are applied. In Sect. 6.4, we show simulation results of the proposed algorithms and the standard GCR method when applied to MOR of a circuit with numerous inputs and outputs. The recycling algorithms are compared with GCR by the number of matrix–vector products needed for solving each system. We also compare recycled GCR with GMRES-based recycling as suggested in [20]. Conclusions are given in the end.

6.2 Methods Based on Recycling Krylov Subspaces

In this section, we first introduce standard GCR and discuss some of its properties. Based on this method, we propose several recycling algorithms for solving sequences of linear systems as in (6.1). In addition, we present a convergence analysis of one of the recycling algorithms in exact arithmetic. (A finite precision analysis is beyond the scope of this paper.)

6.2.1 The Standard Generalized Conjugate Residual Method

The standard GCR method is proposed to solve a single system of linear equations $Ax = b$ in [5]. GCR belongs to a kind of descent methods for nonsymmetric systems, which have the general form as below:

Algorithm 1 Descent method

```
1.  given $x_1$ (initial guess), compute $r_1 = b - Ax_1$;
2.  set $p_1 = r_1$; $k = 1$;
3.  while $\|r_k\| > \text{tol}$ do
4.      $\alpha_k = \frac{(r_k, Ap_k)}{(Ap_k, Ap_k)}$;
5.      $x_{k+1} = x_k + \alpha_k p_k$;
6.      $r_{k+1} = r_k - \alpha_k Ap_k$;
7.      compute $p_{k+1}$;
8.      $k = k + 1$;
9.  end while
```

The parameter α_k is chosen to minimize $\|r_{k+1}\|_2$ as a function of α, so that the residual is reduced step by step. The descent methods differ in computing the new descent vector p_{k+1}. For GCR, the descent vector p_{k+1} is computed so that it satisfies $(Ap_{k+1}, Ap_j) = 0$ for $1 \le j \le k$ (see [5] for details; $(y, z) = z^T y$ denotes the Euclidian inner product in $\mathbb{R}_n$). For completeness, we present a version of standard GCR in the following algorithm.

Algorithm 2 GCR

```
1.   given $x_1$ (initial guess), compute $r_1 = b - Ax_1$; set $k = 1$;
2.   while $\|r_k\|_2 >$ tol do
3.       $\hat{p}_k = r_k; \hat{q}_k = A\hat{p}_k$;
4.       for $j = 1$ to $k - 1$
5.           $\beta_{kj} = (\hat{q}_k, q_j)$;
6.           $\hat{q}_k = \hat{q}_k - \beta_{kj} q_j$;
7.           $\hat{p}_k = \hat{p}_k - \beta_{kj} p_j$;
8.       end for
9.       $q_k = \hat{q}_k / \|\hat{q}_k\|_2, p_k = \hat{p}_k / \|\hat{q}_k\|_2$;
10.      $\alpha_k = (r_k, q_k)$;
11.      $x_{k+1} = x_k + \alpha_k p_k$;
12.      $r_{k+1} = r_k - \alpha_k q_k$;
13.      $k = k + 1$;
14.  end while
```

It is not difficult to prove that the descent vectors p_k in Algorithm 2 satisfy $(Ap_{k+1}, Ap_j) = 0$ for $j = 1, \ldots, k$—this is the effect of the for-loop which serves the purpose of $A^T A$-conjugation of the descent vectors. It is also proven in [5] that x_{k+1} minimizes $R(w) = \|b - Aw\|_2$over the affine space $x_1 + \text{span}\{p_1, p_2, \ldots, p_k\}$, where $\text{span}\{p_1, p_2, \ldots, p_k\}$ constitutes a Krylov subspace, i.e. $\text{span}\{p_1, p_2, \ldots, p_k\} = \text{span}\{p_1, Ap_1, \ldots, A^{k-1} p_1\}$.

In the next subsection, we will show how to employ the descent vectors from the previous systems for solving the current system. This way, the standard GCR method can be accelerated and many matrix–vector products can be saved when solving all linear systems in a sequence as in (6.1).

6.2.2 Recycling Methods

In this subsection, we propose three variants of recycling algorithms based on the standard GCR method. The idea of recycling descent vectors in GCR is motivated by the recycling algorithm in [19], where sequences of linear systems $(I + \sigma_i A) = b_i$, $i = 1, 2, \ldots, m$, are solved. Unfortunately, the algorithm in [19] is not convergent for some cases, because the computed x_k does not minimize $R(w) = \|b - \tilde{A}_i w\|_2$ ($\tilde{A}_i = I + \sigma_i A$) at each iteration step. In the following, we present a convergent recycling algorithm: Algorithm 3. We will further show that for each system $Ax = b_i$, x_k^i generated by Algorithm 3 minimizes $R(w) = \|b_i - Aw\|_2$ at step k.

Algorithm 3 GCR with recycling for $Ax = b_i, i = 1, 2, \ldots m$.

```
1.  $NumDirecs = 0$; (number of descent vectors $p_j$, $j = 1, 2, \ldots$)
2.  for $i = 1$ to $m$
3.      given $x_1^i$ (initial guess for the $i$th linear system), compute $r_1^i = b_i - Ax_1^i$;
4.      set $k = 1$;
5.      while $\|r_k^i\|_2 >$ tol do
6.          if $k > NumDirecs$ then
7.              $\hat{p}_k = r_k^i$;
8.              $\hat{q}_k = A\hat{p}_k$;
9.              for $j = 1$ to $k - 1$
10.                 $\beta_{kj} = (\hat{q}_k, q_j)$;
11.                 $\hat{q}_k = \hat{q}_k - \beta_{kj} q_j$;
12.                 $\hat{p}_k = \hat{p}_k - \beta_{kj} p_j$;
13.             end for
14.             $q_k = \hat{q}_k / \|\hat{q}_k\|_2$, $p_k = \hat{p}_k / \|\hat{q}_k\|_2$;
15.             $NumDirecs = k$;
16.         end if
17.         $\alpha_k^i = (r_k^i, q_k)$;
18.         $x_{k+1}^i = x_k^i + \alpha_k^i p_k$;
19.         $r_{k+1}^i = r_k^i - \alpha_k^i q_k$;
20.         $k = k + 1$;
21.     end while
22. end for
```

In Algorithm 3, every system $Ax = b_i, i = 2, 3, \ldots$, reuses the descent vectors p_j computed from the previous systems. For example, in the beginning of solving system $Ax = b_k$, *NumDirecs* denotes the number of descent vectors computed when solving $Ax = b_i, i = 1, 2, \ldots, k - 1$. These descent vectors are reused when computing the approximate solution of $Ax = b_k$. If these descent vectors cannot produce an accurate enough solution, then new descent vectors are computed, followed by the matrix–vector multiplication $q_k = Ap_k$ in the *if* loop. This way, a considerable number of matrix–vector products together with the corresponding conjugations are saved when solving the current system. Altogether the amount of computations can be significantly reduced as compared to starting an iterative solver from scratch for each new system in the sequence. Furthermore, Algorithm 3 has similar convergence properties as standard GCR as is shown in the following. We first state the results and then comment on the proofs.

Corollary 1 *The descent vectors generated by Algorithm 3 are A^TA-orthogonal, that is,* $(Ap_i, Ap_j) = 0, i \neq j$.

Corollary 2 *For each linear system* $Ax = b_i$ in (6.1), x^i_{k+1} *generated by Algorithm 3 minimizes* $R(x) = \|b_i - Ax\|_2$ *over the subspace* $x^i_1 + \mathrm{span}\{p_1, p_2, \ldots, p_k\}$.

Following the corresponding proof of Theorem 3.1 in [5], the proof of Corollary 1 is straightforward. (Essentially, the proof consists of observing that the `for`-loop in Algorithm 3 is modified Gram-Schmidt A^TA-orthogonalization.) Because x^i_{k+1} in Algorithm 3 is constructed analogously to x_{k+1} in Algorithm 2, and using the proof of Theorem 3.1 in [5], Corollary 2 can also be easily proven if the descent vectors $p_j, j = 1, 2, \ldots k$ are A^TA-orthogonal, which is guaranteed by Corollary 1. Due to space limitation, a complete proof is not provided here.

In Algorithm 3, the search space is actually composed of several Krylov subspaces. For example, when solving the second system, the recycled vectors constitute the Krylov subspace

$$K_{n_1}(A, r^1_1) = \{r^1_1, Ar^1_1, \ldots, A^{n_1-1}r^1_1\}.$$

Furthermore, if new descent vectors are computed to obtain the solution of the second system, they constitute a second Krylov subspace

$$K_{n_2}(A, r^2_{n_1+1}) = \{r^2_{n_1+1}, Ar^2_{n_1+1}, \ldots, A^{n_2-1}r^2_{n_1+1}\}.$$

Here, n_i, $i = 1, 2$ are the number of vectors in the Krylov subspaces. That is, the approximation x^2 of $Ax = b_2$ minimizes the residual over

$$x^2_1 + K_{n_1}(A, r^1_1) + K_{n_2}(A, r^2_{n_1+1}),$$

which, if analyzed further, is a nested Krylov subspace since $r^2_{n_1+1}$ is generated using $K_{n_1}(A, r^1_1)$, for details see [4].

Generally, for the ith system, the recycled vectors are all the linearly independent vectors from the previous Krylov spaces $K_{n_1}(A, r^1_1), K_{n_2}(A, r^2_{n_1+1})$, $K_{n_3}(A, r^3_{n_1+n_2+1}), \ldots$. Thus, in general, the number of recycled Krylov vectors increases with the number of linear systems to be solved. We will see in Sect. 6.4 that this causes a problem as often, for each new system, some new vectors have to be added to attain the required accuracy. Thus, the major drawback of Algorithm 3 is: if there are a large number of systems in (6.1), the number of recycled descent vectors generated may become simply too large. Since all the descent vectors are reused for the next system, they all have to be stored till the final system is solved. For very large-scale systems, the required memory resources will be massive. Moreover, the conjugation part of the algorithm will slow down the algorithm. Therefore, we propose Algorithm 4 in which a restarting technique is used. This keeps the number of recycled descent vectors within a maximum limit which can be based upon the available memory and the dimension of the linear systems.

Algorithm 4 Algorithm 3 with restarts.

```
1. NumDirecs = 0; (number of descent vectors $p_j$, $j = 1, 2, \ldots$)
2. for $i = 1$ to $m$
```

```
3.     if NumDirecs > MaxDirecs
4.         NumDirecs = 0;
5.     end if
6.     perform steps 3.–21. of Algorithm 3;
7. end for
```

Algorithm 4 is equivalent to Algorithm 3, except for the first if-branch in the beginning of the algorithm. In the algorithm, *MaxDirecs* is the maximum number of recycled descent vectors, and the recycling is restarted whenever *MaxDirecs* is reached. When recycling is restarted, the algorithm is identical with standard GCR. This restart technique is somewhat heuristic, because the optimal *MaxDirecs* is difficult to determine. Here “optimal” is meant in the sense that the overall number of matrix–vector products is minimized. However, from experimental experience, the algorithm requires the least matrix–vector products if *MaxDirecs* is set to be a little larger than the actual number of iteration steps needed by standard GCR for solving a single system.

Since Algorithm 4 reduces to the standard GCR method whenever it restarts, no vector is recycled for the current system. As information is wasted this way, this can be considered as a drawback of the algorithm. In the following Algorithm 5, we propose another possibility of controlling the number of recycled vectors in Algorithm 3. Defining M to be the number of recycled vectors used in each (except for the first) system, we then use the M descent vectors which are computed when solving the first system as the recycled vectors for each of the following systems. Thus, each system is guaranteed to be able to recycle certain vectors. The second advantage of Algorithm 5 over Algorithm 4 is that M can be set much smaller than *MaxDirecs*; therefore it can outperform Algorithm 4 by saving computations in the orthogonalization process (Step 10–14 in Algorithm 3).

Algorithm 5 Algorithm 3 with fixed number of recycled vectors.

```
 1. for $i = 1$ to $m$
 2.     if $i = 1$
 3.         NumDirecs = 0;
 4.     else
 5.         NumDirecs = $M$; (number of fixed recycled vectors)
 6.     end if
 7.     given $x_1^i$ (initial guess for the ith linear system), compute $r_1^i = b_i - Ax_1^i$;
 8.     $k = 1$;
 9.     while $\|r_k^i\|_2 >$ tol do
10.         if $k >$ NumDirecs
11.             perform steps 7.–14. of Algorithm 3;
12.         end if
13.         perform steps 17.–20. of Algorithm 3;
14.     end while
15. end for
```

Algorithm 4 is the restarted version of Algorithm 3, therefore the recycled vectors also constitute a union of several Krylov subspaces. Once the algorithm restarts, new recycled Krylov subspaces will be generated. Algorithm 5 always recycles the same search space, which is generated when solving the first linear system(s) (in case M or more vectors are needed for solving $Ax = b_1$, these are taken from $K_{n_1}(A, r_1^1)$, otherwise some Krylov vectors from the next Krylov subspaces are also included).

It is known that GCR may break down in some cases. However, if the coefficient matrix is positive real, i.e., $A + A^T$ is positive definite, then it can be shown that no breakdown occurs [5]. In the following section, we will discuss this issue for those linear systems of equations that we are interested in, that is, system matrices A arising in the context of model reduction algorithms for circuit simulation.

6.3 Application to Model Order Reduction

In this section, we review the model order reduction (MOR) problem for linear systems. Based on this, the method SPRIMA [2] for MOR of circuits with massive inputs and outputs is briefly described. We show that in SPRIMA, successive systems of linear equations as in (6.1) need to be solved.

MOR [7, 15] has been widely applied to the fast numerical simulation of large-scale systems in circuit design. Using MOR, the original large dimensional system is approximated by a system of (much) smaller dimension. If the approximation error is within a given tolerance, only the small system needs to be simulated, which will in general take much less time and computer memory than the original large system would do. Here, we consider linear, time-invariant descriptor systems of the form

$$C\dot{x}(t) + Gx(t) = Bu(t), \quad y(t) = L^T x(t) \tag{6.2}$$

with system matrices $C, G \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times \ell}$, and $L \in \mathbb{R}^{n\times r}$. Here, $x(t) \in R^n$ denotes the unknown descriptor vector of the system, while $u(t)$ and $y(t)$ stand for ℓ inputs and r outputs, n is the order of the system. Conventional moment-matching MOR methods, as, e.g., PRIMA [15] derive the reduced-order model by computing a projection matrix V having orthonormal columns and satisfying

$$\begin{aligned}\text{range}\{V\} = \text{span}\{&(s_0C+G)^{-1}B, -(s_0C+G)^{-1}C(s_0C+G)^{-1}B,\\ &\ldots, (-(s_0C+G)^{-1}C)^p(s_0C+G)^{-1}B\}, \quad p \ll n.\end{aligned} \tag{6.3}$$

The reduced-order model is obtained by using the approximation $x \approx Vz$, leading to the projected system

$$\hat{C}\dot{z}(t) + \hat{G}z(t) = \hat{B}u(t), \quad \hat{y}(t) = \hat{L}^T z(t), \tag{6.4}$$

where $\hat{C} = V^T CV, \hat{G} = V^T GV \in \mathbb{R}^{q\times q}, \hat{B} = V^T B \in \mathbb{R}^{q\times \ell}, \hat{L} = V^T L \in \mathbb{R}^{q\times r}$. The matrix $V \in \mathbb{R}^{n\times q}$ with $q \ll n$ depending on p and ℓ satisfies $V^T V = I$. The new descriptor vector is $z(t) \in \mathbb{R}^q$.

If there are many inputs and outputs in (6.2), conventional MOR methods as the one described above are inefficient in obtaining a reduced-order model. It is not difficult to see from (6.3) that each term in the Krylov subspace is given as a matrix with l columns, where l is the number of inputs as well as the number of columns in B. If the number of linearly independent columns in B is very large, then the number q of columns of the projection matrix V increases very quickly. This may cause that a system fails to be reduced, the dimension of the reduced-order model can become as large as that of the original system.

Several methods have been proposed for MOR of systems with massive inputs/outputs, including the ones discussed in [2, 8, 13, 14]. One of the newly proposed methods is SPRIMA [2]. When there is a large number of inputs, SPRIMA reduces the original system (6.2) by decoupling it into l single-input systems. The basic idea is described in the following. Based on the superposition principle for linear systems, the output response of the multi-input multi-output system (6.2) can be obtained by the sum of the output responses of the following single-input systems:

$$\begin{aligned} C\dot{x}(t) + Gx(t) &= b_i u_i(t), \\ y_i(t) &= L^T x(t), \end{aligned} \quad \text{for } i = 1, 2, \ldots, \ell, \tag{6.5}$$

i.e., $y(t) = y_1(t) + y_2(t) + \cdots + y_\ell(t)$, with b_i being the ith column of B.

Since PRIMA [15] is efficient for MOR of single-input systems, by applying PRIMA to each single-input system in (6.5), we can get approximate output responses $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_l$ of $y_1, y_2, \ldots, y_\ell$ from the reduced-order models of the single-input systems

$$\begin{aligned} \hat{C}\dot{z}(t) + \hat{G}z(t) &= \hat{b}_i u(t), \\ \hat{y}_i(t) &= \hat{L}^T z(t), \end{aligned} \quad \text{for } i = 1, 2, \ldots, \ell. \tag{6.6}$$

The output response of system (6.2) is thus approximated by $y(t) \approx \hat{y}_1(t) + \hat{y}_2(t) + \cdots + \hat{y}_\ell(t)$.

Based on the above considerations, we can see that by decoupling the original system with numerous inputs into systems with a single input, SPRIMA [2] overcomes the difficulties of PRIMA. As can be seen from (6.6), we actually create several reduced-order models, but the final output of the original system can be computed by the superposition of all the reduced-order models. Moreover, the final transfer function of the original system can also be represented by the transfer functions of the reduced models [2].

As has been introduced in (6.3), for the ith single-input system in (6.6), PRIMA requires the matrices V_i to be computed as

$$\begin{aligned} \text{range}\{V_i\} = \text{span}\{&(s_0C + G)^{-1}b_i, -(s_0C + G)^{-1}C(s_0C + G)^{-1}b_i, \\ &\ldots, (-(s_0C + G)^{-1}C)^{p_i}(s_0C + G)^{-1}b_i\}. \end{aligned} \tag{6.7}$$

It is clear from (6.7) that in order to compute V_i, $i = 1, \ldots, \ell$, we have to solve $\sum_{i=1}^{l} p_i + \ell$ systems of linear equations with identical coefficient matrix $s_0C + G$, but different right-hand sides. In [2], it is proposed to use the (sparse) LU decomposition $s_0C + G$ to solve systems of medium dimension. For very large-scale systems, standard iterative solvers are employed to solve them one by one. For circuit systems with numerous inputs/outputs, thousands of such systems are produced during the MOR process, the computational complexity in solving them increases much too fast. Therefore, efficient computational algorithms for solving successive systems of linear equations are very important to improve the whole MOR process. Thus, we propose to use the recycling algorithms proposed in Sect. 6.2 to solve the sequence of systems arising from (6.7).

Please note that for each single-input system in (6.5), MOR methods other than PRIMA can also be employed to derive the reduced single-input systems in (6.6). For example, one-sided rational Krylov methods [11] (which allow multiple expansion points) are possible here.

It remains to discuss the breakdown issue in GCR for the linear systems of equations encountered here. If RLC circuits are modeled by modified nodal analysis (MNA), the matrices in (6.5) have the structure

$$C = \begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix}, \quad G = \begin{bmatrix} G_1 & G_2 \\ -G_2^T & 0 \end{bmatrix},$$

where G_1, C_1 are symmetric positive semidefinite and C_2 is symmetric positive definite [10]. As a consequence, both G, C and thus $A = s_0C + G$ have a positive semidefinite symmetric part for any real s_0. (Similar conclusions can be drawn for complex s_0 using complex conjugate transposes.) Unfortunately, the no-breakdown condition in [5] requires A to be positive real, i.e., its symmetric part to be positive definite. But from the structure, we can only infer that the symmetric part of A is positive semidefinite. Although we have not encountered breakdown in any of the numerical examples performed for RLC circuit equations, the possibility of breakdown cannot be excluded. As we are fairly close to the necessary positive realness condition for A, there is hope that breakdown will only be encountered in pathological cases. Moreover, in many situations (if the circuit neither contains voltage sources nor a cutset consisting of inductances and/or current sources only, see [6]), it can be shown that $s_0C_1 + G_1$ is positive definite; in these cases, $A = s_0\, C + G$ will be positive real and thus, breakdown cannot occur.

6.4 Simulation Results

In this section, we illustrate the efficiency of the proposed recycling algorithms when used within SPRIMA. As a test case, we employ an RLC circuit with $\ell = 854$ inputs and $r = 854$ outputs. The dimension of the system is fairly small,

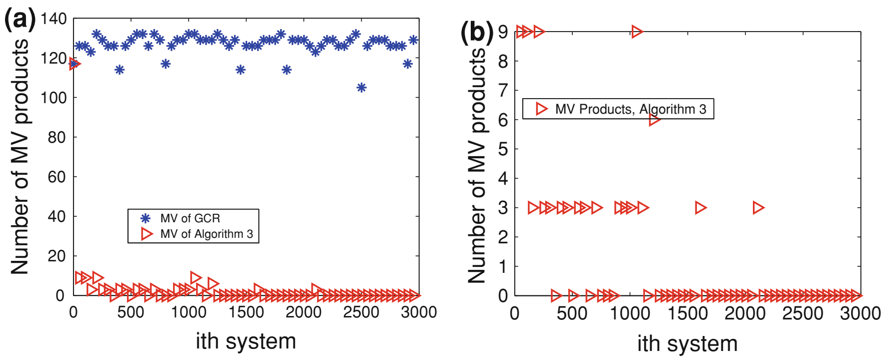

Fig. 6.1 Algorithm 3: ♯ of matrix–vector products. **a** All systems and **b** all but 1st system

$n = 1{,}726$, but it allows to demonstrate the effects of recycling. The model was generated with the circuit simulator TITAN.[1]

We compare Algorithms 3–5 with the standard GCR method (Algorithm 2). If we set $p_i \equiv 19$ in (6.7), then $m = (p_i + 1)\ell = 20 \times 854 = 17{,}080$ systems of linear equations have to be solved in order to obtain the projection matrices $V_i,\ i = 1, 2, \ldots, \ell$. In this case, $p_i = 19$ is necessary to obtain the desired accuracy of the reduced-order model. As for a preconditioner, we employ an incomplete LU factorization with threshold 0.01 (in MATLAB® notation: `ilu(A, 0.01)`) within all algorithms.

We first compare standard GCR with Algorithm 3 by solving the first 3,000 systems. (Similar results for all the systems are also available.) The number of matrix–vector (MV) products for solving each system is plotted in Fig. 6.1. The number of MV products required by Algorithm 3 is much less than that of GCR. Only very few MV products are required after solving the first several systems. Starting from the 1,000 th system, there are almost no MV products necessary.

Figure 6.1 suggests that Algorithm 3 is much more efficient than GCR. However, this would only be true if n would be significantly larger than the number of systems to be solved (which is not the case in the example considered here). Figure 6.2 illustrates that in this example, the number of recycled vectors reaches n before all systems are solved so that storing all recycled vectors need a full $n \times n$ array which is not feasible for larger problems. In order to solve this problem, we propose to use Algorithm 4 or Algorithm 5.

Figures 6.3 and 6.4a compare the MV products required by GCR with the corresponding numbers resulting from Algorithm 4 and Algorithm 5, respectively. Figure 6.3 shows that although Algorithm 4 needs more MV products than Algorithm 3, it is applicable to much larger systems as the number of recycled vectors remains limited. Figure 6.4a illustrates that Algorithm 5 can save MV

[1] TITAN was then developed by Qimonda AG, Neubiberg (Germany); after insolvency of Qimonda in 2009, TITAN is now owned by Infineon Technologies AG, Neubiberg.

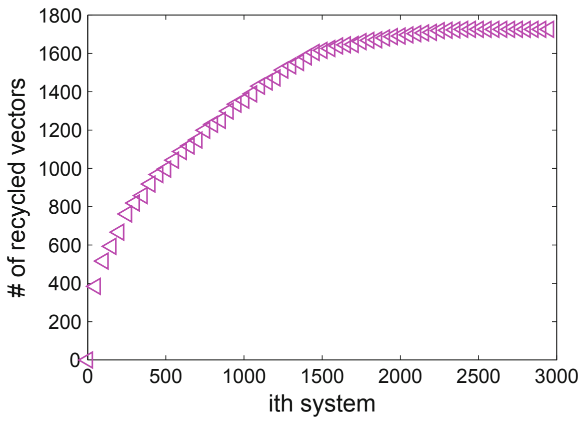

Fig. 6.2 Algorithm 3: ♯ of recycled vectors

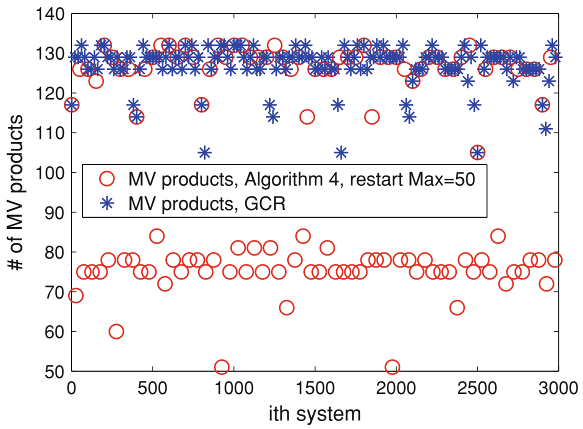

Fig. 6.3 Algorithm 4: ♯ of matrix–vector products

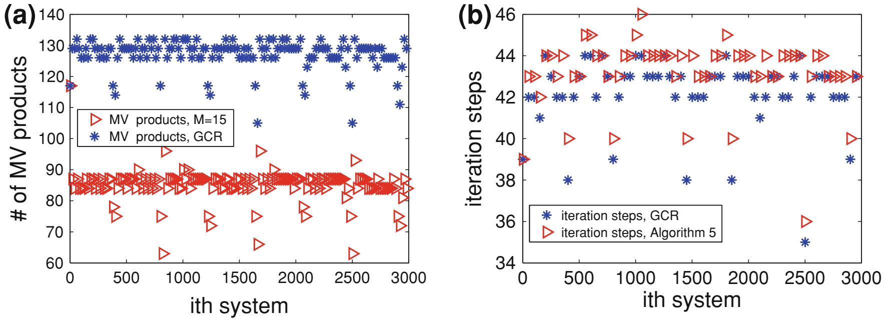

Fig. 6.4 Results for Algorithm 5. **a** ♯ of matrix–vector products **b** ♯ of iteration steps

products for each system, whereas Algorithm 4 cannot save MV products for all the systems as is shown in Fig. 6.3, because Algorithm 4 falls back on GCR whenever it restarts. One can also conclude from Fig. 6.4b that the number of iteration steps used in Algorithm 5 is almost the same as that of GCR for solving each system. Notice that Algorithm 5 uses $M = 15$ recycled vectors for each system; therefore, it actually saves $M = 15$ MV products for each system in the standard form as 1 MV product is computed per iteration. However, if incomplete LU preconditioning is used, there will be three MV products at each iteration step in GCR: one is due to the coefficient matrix itself, the other two are due to the backward/forward solves employed by the preconditioner. Therefore, $M = 15$ iteration steps actually lead to 45 MV products.

In Table 6.1, we list the MV products saved by Algorithms 4 and 5 for solving several groups of systems of linear equations. It is clear that the recycling algorithms can save many MV products when solving long sequences of systems, and therefore can improve the overall efficiency of MOR.

In order to show the accuracy of the reduced-order model obtained by applying the recycling algorithms, we compare the simulation results of the reduced-order model with those of the original model in Fig. 6.5a, b. We use step functions at all the terminals as the input signals and observe the output response at the first terminal. The reduced model is derived by using Algorithm 5. The output responses of the reduced model and the original system at the first terminal are plotted in Fig. 6.5a and the maximum relative error between them is 4.8×10^{-6}.

Table 6.1 Number of MV products saved by Algorithms 4 and 5

No. of systems solved	MV saved by Algorithm 4	MV saved by Algorithm 5
500	13,320	21,288
1,000	26,337	42,381
2,000	52,530	84,354
3,000	78,630	126,039

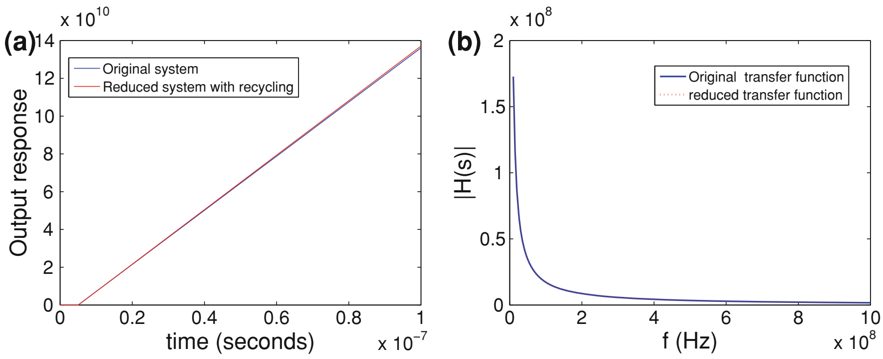

Fig. 6.5 Accuracy of the reduced model. **a** Comparison of the output response and **b** comparison of the transfer function

We have similar accuracy for the output signals at the other terminals. The magnitudes of the transfer functions for both systems are shown in Fig. 6.5b. The transfer function shown here is the mapping from the input signal at the first terminal to the output response at the same terminal. The relative error (computed point-wise at 201 frequency samples) of the magnitude of the reduced transfer function is around 10^{-10}, therefore the two waveforms are indistinguishable in the figure. The accuracy of the transfer functions at any other two terminals is also acceptable, with errors usually smaller than 10^{-4}.

A new recycling method based on GMRES is proposed in [20]. The basic idea of recycling in this method, called MKR_GMRES, is quite similar to Algorithm 3. The main difference is that MKR_GMRES uses GMRES as the basic algorithm for recycling, the recycled vectors are the orthogonal basis vectors generate by the underlying Arnoldi process. When solving the *i*th system, the Arnoldi vectors generated by all the previous linear systems are reused to generate a "better" initial solution. If the initial solution is not accurate, new Arnoldi vectors are computed and the solution is generated in the augmented Krylov subspace spanned by all the Arnoldi vectors. Because all the orthogonal Arnoldi vectors are recycled, therefore after solving a certain number of systems, the number of new Arnoldi vectors becomes less and less. This can be observed from the number of MV products used for solving each linear system (illustrated in Fig. 6.6a). It has a similar behavior as Algorithm 3 which recycles all the previously computed descent vectors. The method also has a similar problem as Algorithm 3, that is the dimension of the recycled Krylov subspace increase fast, see Fig. 6.6b. Since each new Arnoldi vector has to be orthogonalized to all the previous ones, the orthogonalization process becomes slower and slower, which affects the efficiency of the whole algorithm. Although Algorithms 4 and 5 are somewhat heuristic, they can remedy the the weakness of Algorithm 3, and also work quite well for

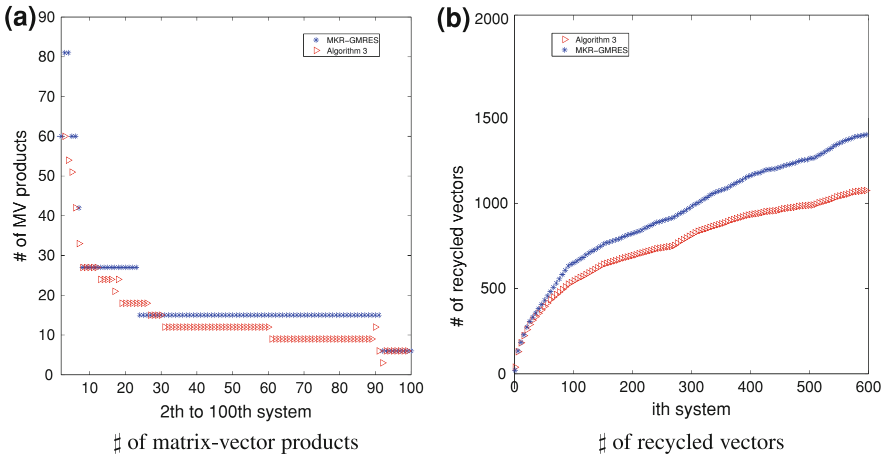

Fig. 6.6 Results of MKR_GMRES. **a** ♯ of matrix–vector products **b** ♯ of recycled vectors

certain systems. It is also possible to construct recycling method based on nested GMRES. Here, "nested" means the GMRES method is nested inside some other iteration solver and contributes to the inner iterations. We have studied this recycling technique in [9]. In this paper we have applied the algorithm to parametric model order reduction in system level simulation for Microelectromechanical systems (MEMS). The outer iteration of the recycling method is GCRO proposed in [4], and the inner iterations are done by GMRES. The recycled subspace is generated by Harmonic Ritz vectors of the coefficent matrix A and does not inflate with the number of linear systems as happens in the algorithms here. The original version of the recycling algorithm in [9] is proposed in [16], where the theoretical background of the method is analyzed in detail.

6.5 Conclusions

We have proposed variants of recycling descent vectors in GCR for solving sequences of linear systems with varying right-hand sides. The recycling methods are efficient for solving sequences of large-scale linear systems successively. The methods discussed here are limited to solving systems with identical coefficient matrix and different right-hand sides. The techniques proposed for controlling the number of recycled vectors are heuristic. Further research will focus on more robust recycling techniques (such as thick restarting) and on solving systems with different coefficient matrices.

Acknowledgements This research was supported by the Alexander von Humboldt-Foundation and by the Research Network *SyreNe—System Reduction for Nanoscale IC Design*, funded by the German Federal Ministry of Education and Science (BMBF), grant no. 03BEPAE1. Responsibility for the contents of this publication rests with the authors. We would like to thank Timo Reis (TU Berlin) for helpful discussions on the positive realness conditions for matrices arising from RLC circuits.

References

1. Ahuja, K.: Recycling Bi-Lanczos algorithms: BiCG, CGS, and BiCGSTAB. Master's thesis, Virginia Tech, Blacksburg, VA (2009)
2. Benner, P., Feng, L.H., Rudnyi, E.B.: Using the superposition property for model reduction of linear systems with a large number of inputs. In: Proceedings of the 18th International Symposium on Mathematical Theory of Networks and Systems, 12 (2008).
3. Davis, T.A.: Direct methods for sparse linear systems. Fundamentals of Algorithms, vol 2. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (2006)
4. de Sturler, E.: Nested Krylov methods based on GCR. J. Comput. Appl. Math. **67**(1), 15–41 (1996)
5. Eisenstat, S.C., Elman, H.C., Schultz, M.H. (1983) Variational iterative methods for nonsymmetric systems of linear equations. SIAM J. Numer. Anal. **20**(2):345–357

6. Estévez-Schwarz, D., Tischendorf, C.: Structural analysis of electrical circuits and consequences for MNA. Int. J. Circuit Theory Appl. **28**(2), 131–162 (2000)
7. Feldmann, P., Freund, R.: Efficient linear circuit analysis by Padé approximation via the Lanczos process. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **14**, 639–649 (1995)
8. Feldmann, P., Liu, F.: Sparse and efficient reduced order modeling of linear subcircuits with large number of terminals. In: Proceedings of the International Conference on Computer-Aided Design, pp. 88–92 (2004)
9. Feng, L., Benner, P., Korvink, J.G.: Parametric model order reduction accelerated by subspace recycling. In: Proceedings of the 48th IEEE CDC and 28th Chinese Control Conf., Shanghai, P.R. China, December 16-18, 2009, pp. 4328–4333 (2009)
10. Freund, R.: SPRIM: structure-preserving reduced-order interconnect macromodeling. In: Tech. Dig. 2004 IEEE/ACM Intl. Conf. CAD, IEEE Computer Society Press, Los Alamitos, pp. 80–87 (2004)
11. Grimme, E.J.: Krylov projection methods for model reduction. Ph.D. thesis, University of Illinois, Urbana-Champaign (1997)
12. Kilmer, M., Miller, E., Rappaport, C.: A QMR-based projection technique for the solution of non-hermitian systems with multiple right hand sides. SIAM J. Sci. Comput. **23**(3), 761–780 (2001)
13. Li, P., Shi, W.: Model order reduction of linear networks with massive ports via frequency-dependent port packing. In: Proceedings of the Design Automation Conference, pp. 267–272 (2006)
14. Nakhla, N., Nakhla, M., Achar, R.: Sparse and passive reduction of massively coupled large multiport interconnects. In: Proceedings of the International Conference on Computer-Aided Design, pp. 622–626 (2007)
15. Odabasioglu, A., Celik, M., Pileggi, L.: PRIMA: passive reduced-order interconnect macromodeling algorithm. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **17**(8), 645–654 (1998)
16. Parks, M.L., de Sturler, E., Mackey, G., Johnson, D.D., Maiti, S.: Recycling Krylov subspaces for sequences of linear systems. SIAM J. Sci. Comput. **28**(5), 1651–1674 (2006)
17. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. SIAM Publications, Philadelphia (2003)
18. Simoncini, V., Gallopoulos, E.: An iterative method for nonsymmetric systems with multiple right-hand sides. SIAM J. Sci. Comput. **16**(4), 917–933 (1995)
19. Telichevesky, R., Kundert, K., White, J.: Efficient AC and noise analysis of two-tone RF circuits. In: Proceedings of the Design Automation Conference, pp. 292–297 (1996)
20. Ye, Z., Zhu, Z., Phillips, J.R.: Generalized Krylov recycling methods for solution of multiple related linear equation systems in electromagnetic analysis. In: Proceedings of the Design Automation Conference, pp. 682–687 (2008)