MAX-PLANCK-GESELLSCHAFT

Lihong Feng   Peter Benner   Jan G Korvink

Fast Parametric Macromodeling of MEMS using Subspace Recycling

MAX–PLANCK–INSTITUT
FÜR DYNAMIK KOMPLEXER
TECHNISCHER SYSTEME
MAGDEBURG

Max Planck Institute Magdeburg Preprints

MPIMD/12-08   **March 27, 2012**

Impressum:

Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg

Publisher:
Max Planck Institute for Dynamics of Complex Technical Systems

Address:
Max Planck Institute for Dynamics of Complex Technical Systems
Sandtorstr. 1
39106 Magdeburg

www.mpi-magdeburg.mpg.de/preprints

Lihong Feng Peter Benner Jan G Korvink

Fast Parametric Macromodeling of MEMS using Subspace Recycling

MPIMD/12-08 March 27, 2012

Fast Parametric Macromodeling of MEMS using Subspace Recycling [*]

Lihong Feng[†] Peter Benner[‡] Jan G Korvink [§]

March 27, 2012

Abstract

A fast computational technique is proposed to speed up the process of parametric macromodel extraction of Micro-Electro-Mechanical Systems (MEMS). An efficient technique of parametric macromodel extraction of MEMS is parametric model order reduction (PMOR). The key step and the main computational load of the popular parametric model order reduction methods is the computation of a projection matrix V which requires computing moment matrices of the systems. For computing each moment matrix, the solution of a linear system with multiple right-hand sides is required. Usually, a considerable number of linear systems must be solved when the system includes more than two parameters. If the original system is of very large size, solving all the linear systems is the most computationally expensive step to obtain the reduced model. In this paper, a fast recycling algorithm GCRO-DR is applied to solve the whole sequence of linear systems. In addition, more efficient recycling algorithms G-DRvar1 and G-DRvar2 are proposed. Theoretical analysis and simulation results show that both the GCRO-DR and its variants G-DRvar1, G-DRvar2 are very efficient as compared with the standard solvers. Furthermore, the algorithms in this paper overcome the bottleneck of a recently proposed recycling method MKR-GMRES. By using the recycling algorithms, the PMOR process for extracting the macromodel can be significantly accelerated.

Keywords parametric model order reduction, linear system, recycling algorithm

[*]This is an intensively extended version of "Parametric Model Order Reduction Accelerated by Subspace Recycling," presented at the joint 48th IEEE Conference on Decision Control and 28th Chinese Control Conference, Shanghai, P. R. China, December 16-18, 2009, see [24].

[†]Lihong Feng was with Department of Microsystems Engineering (IMTEK), University of Freiburg, Germany. She is now with Max Planck Institute for Dynamics of Complex Technical Systems, Germany. feng@mpi-magdeburg.mpg.de

[‡]Peter Benner is with Max Planck Institute for Dynamics of Complex Technical Systems, and Faculty of Mathematics, Chemnitz University of Technology, Germany. benner@mpi-magdeburg.mpg.de

[§]Jan G. Korvink is with Department of Microsystems Engineering (IMTEK) and Freiburg Institute of Advanced Studies (FRIAS), University of Freiburg, Germany. korvink@imtek.uni-freiburg.de

1 Introduction

Modeling and numerical simulation are unavoidable for MEMS design due to the very small size and complexity of the devices. To simulate a MEMS model which is modeled by partial differential equations (PDEs), spatial discretization such as finite element discretization is required. A system of ordinary differential equations (ODEs) or differential algebra equations (DAEs) after discretization has to be solved numerically. Currently, in order to reduce the simulation cycles and to shorten the period between design and manufacturing, it is demanding to establish parameterized MEMS models for simulation purpose, meaning the PDEs of MEMS shall include parameters which describe the geometrics of the devices and/or the environmental conditions. By including the parameters in the model, the system of ODEs/DAEs derived from spatial discretization also contains the parameters. With the requirement of design analysis, simulation of the parameterized system of ODEs/DAEs may be implemented many times with different values of parameters. Usually, the number of ODEs/DAEs is very large. Simulating such a system once is already very time consuming, not to mention multiple simulations.

Model order reduction has been recognized as an efficient technique to reduce the simulation time for such a parameterized system. Through model order reduction, a much smaller system with a reduced number of equations can be derived, which is called the reduced model of the original system. The reduced model can be simulated much faster, and the solution of the original system can be approximated by the solution of the reduced model with acceptable accuracy. The reduced model can also replace the original system and be reused many times during the design process, which can further save much time. To date, model order reduction has been widely applied to simulation of MEMS and has achieved much success in simulation and design [4, 5, 6]. The process of model order reduction for MEMS is also known as macromodel extraction in many literatures.

As far as an efficient model order reduction method is concerned, it is best not only the reduced model is as small as possible, but also the process of obtaining it is as fast as possible. Unfortunately, due to the large size of the original system, the process of deriving such a reduced model (macromodel) is still relatively slow.

The recently developed model order reduction methods for parameterized systems include [9, 10, 18, 19, 20, 21] etc. These methods are mostly based on matching moments of the parameterized system. For example, in [9, 21], a modified Gram-Schmidt process is used to orthogonalize the moment matrices and to construct the projection matrix V. During the orthogonalization process, each moment matrix in the subspace has to be computed and orthogonalized against previous moment matrices. However, computing each moment matrix includes several vectors multiplied by the inverse of a matrix. If the dimension of the original system is very large, linear systems of equations must be solved so as to obtain the "inverse of matrix"-vector products. If there are more than two parameters in the system, many linear systems of equations have to be solved in order to match enough moments so that a reduced model with acceptable accuracy can be obtained.

The moment matrices of a parametric model usually are in the forms $E^{-1}E_i, i = 1, 2, \cdots$. When the matrix size is very large, the direct solver such as LU factorization cannot be used to deal with E^{-1} due to the occupation of huge memory storage. One needs to solve a sequence of linear systems iteratively to form the final $E^{-1}E_i,\ i = 1, 2, \ldots$. Such a sequence of linear systems can be

$$Ax = b_i, i = 1, 2, \cdots, l, \tag{1}$$

or

$$A_j x = b_i^j, i = 1, 2, \cdots, l_j, j = 1, 2, \cdots, l_o, \tag{2}$$

where l_o is usually much smaller than each l_j. It is noticed that the index j in (2) is not the power of b_i, but only a superscript of b_i. We have similar notations in the following sections. If the model includes more than two parameters, the systems in the sequence could amount to hundreds. Such a sequence of linear systems can be solved one after another by the standard iterative solvers such as the Generalized Minimal RESidual method (GMRES) [7] and the Conjugate Gradient method (CG) [8] etc..

Subspace recycling in [1] was initially proposed to solve a sequence of linear systems such as $A_i x_i = b_i$, $i = 1, 2, \cdots, l_o$. The coefficient matrices A_i are generally considered to be different from each other. By using the information of the previous linear systems, the convergence rate of the standard iterative solvers such as GMRES can be obviously accelerated.

In this paper we propose to apply the fast subspace recycling algorithm GCRO-DR in [1] to accelerate the solution of the sequences in (1), (2). Furthermore, we have proposed variant versions of GCRO-DR, which have been shown to be more efficient than both GCRO-DR in [1] and SimGCRO-DR in [24] when solving the systems in (2). To the best knowledge of the authors, it is the first time that subspace recycling has been introduced into parametric model order reduction and has been successfully applied to macromodel extraction for MEMS.

Besides introducing the recycling algorithms, we also present a PMOR algorithm which integrates the recycling algorithm with the process of obtaining the reduced model. With the algorithm, one can see that the linear systems actually do not continuously appear as those in (1), (2). However, all the linear systems can be solved with one recycling algorithm, even if they are interrupted by intermediate computations.

Compared with our conference paper [24],

1. more efficient recycling algorithms G-DRvar1/G-DRvar2 are proposed in this paper,

2. we make more detailed explanations of the algorithm GCRO-DR, which can be seen as a complement to the theoretical analysis in [1],

3. we not only introduce the recycling algorithms, but also present an algorithm of parametric model order reduction combined with the recycling algorithm. As a result, more intuition for the application of the recycling algorithm to PMOR is provided,

4. we provide implementation details of PMOR for two more complex MEMS models, a butterfly gyroscope and a microhotplate gas sensor chip, while only a simple 2D example is given in the conference paper [24]. In [24], only recycling algorithm for the sequence in (1) has been considered. However, the sequence of linear systems in (2) which arise from multi-point expansion of the transfer function must be solved for a system with more than two parameters. The newly proposed algorithm G-DRvar1/G-DRvar2 can solve (2) more efficiently than the original version GCRO-DR and the previous version SimGCRO-DR in [24].

It is worth pointing out that the proposed recycling algorithm can also be easily applied to other PMOR methods in [10, 18, 19, 20], where successive linear systems need to be solved when the system dimension is very large. We use the method in [9, 21] as an example to explore the recycling algorithm.

In [27], a recycling algorithm based on BiCG method [28] is applied to model order reduction for non-parametric systems. It is difficult to say which method is more efficient, since the efficiency of the recycling algorithms depends on the sequence solved. In [27], the recycling algorithm is applied to a sequence of linear systems $A_i x_i = b_i$, $i = 1, 2, \cdots, l_o$. In this paper we are concerned with solving different sequences of linear systems in (1) and (2), which can be solved even more efficiently (compared with the original version GCRO-DR) by using the modified algorithms G-DRvar1/G-DRvar2 proposed in this paper.

In Section 2, besides introducing the PMOR method in [9], we describe the linear systems to be solved. In Section 3, we give an overview of various iterative solvers solving the sequence of linear systems. In Section 4, we explain the recycling method GCRO-DR [1] in Subsection 4.1. The algorithm SimGCRO-DR in [24] is analyzed in Subsection 4.2. Two variants G-DRvar1/G-DRvar2 of the original version GCRO-DR are presented in Subscetion 4.3 for solving the sequence in (2). A PMOR algorithm which integrates the recycling algorithm G-DRvar1 with the process of PMOR in [9] is described in Subsection 4.4. The method MKR-GMRES in [2, 3] is also briefly explained for comparison. The simulation results of the standard iterative solvers and the recycling algorithms are presented in Section 5, where they are applied to PMOR of three models of MEMS devices. The accuracy of the reduced model derived from the use of the recycling algorithms is also presented. All the simulation results show the efficiency of the recycling algorithms.

2 A Robust Parametric Model Order Reduction Method

PMOR methods are designed for model order reduction of parameterized systems, where the parameters of the system play an important role in practical applications such as Integrated Circuit (IC) design, MEMS design etc. The parameters could be the variables describing geometrical measurement, material property or damping of the systems. The reduced models are constructed such that all the parameters can be preserved with acceptable accuracy. Usually the time on simulating the reduced models is much shorter than directly simulating the original large system. However, the time on constructing the reduced model increases with the dimension of the original system. If the original system is very large, the process of obtaining the reduced model could become extremely slow. The recycling algorithms considered in this paper attempt to accelerate the above process and reduce the time on deriving the reduced model to a reasonable range.

In the following, we first introduce the PMOR method in [9], then we show how to further improve the efficiency of the method by applying the recycling algorithms proposed in the next section.

For simplicity, we use a linear parameterized system as an example, which has the following form in the frequency domain:

$$\begin{aligned} (E_0 + s_1 E_1 + s_2 E_2 + \ldots + s_p E_p)x &= Bu(s_p), \\ y &= L^{\mathrm{T}} x, \end{aligned} \tag{3}$$

where $s_1, s_2, \ldots, s_p$ are the parameters of the system. $x(t) \in \mathbb{R}^n$ is the vector of (generalized) states (also called descriptor variables), $u \in \mathbb{R}^{d_I}$ and $y \in \mathbb{R}^{d_O}$ are, respectively, the inputs and outputs of the system. To obtain the reduced model in (4), a projection matrix V which is independent from all the parameters must be computed.

$$\begin{aligned} V^T(E_0 + s_1E_1 + s_2E_2 + \ldots + s_pE_p)Vx &= V^TBu(s_p), \\ y &= L^{\mathrm{T}}Vx. \end{aligned} \tag{4}$$

The matrix V is derived from orthogonalizing a number of moment matrices of the system in (3)[9, 10].

By defining $B_M = \tilde{E}^{-1}B$, $M_i = -\tilde{E}^{-1}E_i$, $i = 1, 2, \ldots, p$ and

$$\tilde{E} = E_0 + s_1^0E_1 + s_2^0E_2 + \cdots + s_p^0E_p \tag{5}$$

we can expand x in (3) at $s_1, s_2, \ldots, s_p$ around a set of expansion points $p_0 = [s_1^0, s_2^0, \cdots, s_p^0]$ as below,

$$\begin{aligned} x &= [I - (\sigma_1M_1 + \ldots + \sigma_pM_p]^{-1}B_Mu(s_p) \\ &= \sum_{i=0}^{\infty}(\sigma_1M_1 + \ldots + \sigma_pM_p)^iB_Mu(s_p). \end{aligned} \tag{6}$$

Here $\sigma_i = s_i - s_i^0, i = 1, 2, \ldots, p$. We call the coefficients in the above series expansion moment matrices of the parameterized system, i.e. $B_M,\ M_1B_M,\ \ldots,\ M_pB_M,$
$M_1^2B_M, (M_1M_2 + M_2M_1)B_M,\ \ldots,\ (M_1M_p + M_pM_1)B_M,\ M_p^2B_M,\ M_1^3B_M,\ \ldots$. The corresponding moments are those moment matrices multiplied by L^{T} from the left. The matrix V can be generated by first explicitly computing some of the moment matrices and then orthogonalizing them as is suggested in [10]. The resulting V is desired to expand the subspace:

$$\begin{aligned} \text{range}\{V\} = \ & \text{span}\{B_M,\ M_1B_M, \ldots, M_pB_M,\ M_1^2B_M, \\ & (M_1M_2 + M_2M_1)B_M, \ldots, (M_1M_p + M_pM_1)B_M, \\ & M_p^2B_M, M_1^3B_M, \ldots, M_1^rB_M, \ldots, M_p^rB_M\}. \end{aligned} \tag{7}$$

However, V does not really span the whole subspace, because the latterly computed vectors in the subspace become linearly dependent due to numerical instability. Therefore, with this matrix V one cannot get an accurate reduced model which matches all the moments included in the subspace.

Instead of directly computing the moment matrices in (7), a numerically robust method is proposed in [9] (the detailed algorithm is described in [21]), which combines the recursions in (9) with the modified Gram-Schmidt process to implicitly compute the moment matrices. The computed V is actually an orthonormal basis of the subspace as below,

$$\text{range}\{V\} = \text{span}\{R_0, R_1, \ldots, R_r\}. \tag{8}$$

It can be proved that the subspace in (7) is included in the subspace in (8). Due to the numerical stability properties of the repeated modified Gram-Schmidt process employed in [9, 21], the reduced model derived from V in (8) is computed in a numerically stable and accurate way.

$$\begin{aligned}
&R_0 = B_M,\ R_1 = [M_1R_0, \dots, M_pR_0],\\
&R_2 = [M_1R_1, \dots, M_pR_1],\\
&\vdots,\\
&R_r = [M_1R_{r-1}, \dots, M_pR_{r-1}]\\
&\vdots.
\end{aligned} \tag{9}$$

Furthermore, one can see that each moment matrix is actually several vectors multiplied by $\tilde{E}^{-1}$, and if the dimension of $\tilde{E}$ is very large, it is necessary to solve linear systems such as

$$\tilde{E}x = w_i,\ i = 1, 2, \dots, l \tag{10}$$

to obtain $\tilde{E}^{-1}w_i$, where $\tilde{E}$ is generally nonsymmetric and w_i is a vector. Moreover, if quite a few of the moment matrices need to be computed (this is normal when system (3) contains more than 2 parameters), the number l of the linear systems in (10) will be very large. By looking at the above recursions in (9), it is obvious that the right-hand sides of the linear systems cannot be available simultaneously. Systems in (10) can be simply solved one after another by standard iterative methods such as GMRES. However, it is possible to speed up the standard iterative methods by using more efficient methods.

3 Review of Various Iterative Solvers

It is known that the standard iterative solvers such as GMRES, can only solve one linear system at a time. Therefore, all the linear systems in (1), (2) or (10) must be solved independently by GMRES, or in other words, each of them must be solved from scratch. At the jth iteration, the non-restarted version of GMRES generates the approximate solution x_j from a Krylov subspace. In many cases, the solution with acceptable accuracy cannot be obtained until the dimension of the Krylov subspace grows large, which is terrible for very large scale systems. This is because all the orthogonal vectors in the Krylov subspace have to be stored, the method becomes very slow due to memory occupation and also due to the orthogonalization process where the current vector in the Krylov subspace has to be orthogonalized against all the previous orthogonal vectors. Restarted GMRES(m) was proposed in order to reduce the dimension of the Krylov subspace. Given a small number m, the approximate solution at each iteration step is generated always from a Krylov subspace of dimension m. Therefore GMRES(m) is more efficient than GMRES for very large systems. However a trade-off is that GMRES(m) may need many more iteration steps than the non-restarted version to attain the final solution.

Conventional block methods [11, 12] have been developed to improve the efficiency of standard iterative solvers. Unfortunately, these methods require the right-hand sides of all the linear systems to be available simultaneously, whereas the linear systems considered here can only appear sequentially rather than simultaneously. Different algorithms [13, 1, 17, 2, 3] have also been proposed to deal with linear systems whose right-hand sides are not available simultaneously. However, the method in [17] is limited in solving linear systems with symmetric definite coefficient matrices, which is not applicable to the linear systems with general nonsymmetric coefficient matrices. While the method in [13] can solve linear systems with nonsymmetric coefficient matrices, it requires the solutions of neighboring linear systems to be closely related to each other. The solutions of the linear systems in PMOR, nevertheless, do not possess such a property.

A recycling algorithm GCRO-DR has been proposed in [1] and is used for solving linear systems in the form of $A_i x = b_i,\ i = 1, 2, \ldots, l$. It is suitable for nonsymmetric A_i, and with no obvious limitation on each linear system. Therefore, we propose to apply GCRO-DR to the PMOR method in [9, 21].

4 Solving Linear Systems By Recycling Algorithms

4.1 Analysis of GCRO-DR

In this section, we first summarize the motivation and the theoretical background of GCRO-DR. Moreover, we try to explain GCRO-DR in details, such that one can have a better understanding of the algorithm and be able to apply the algorithm more easily. To make the paper self-contained, we present GCRO-DR in Appendix A.

The algorithm GCRO-DR is developed for the sequential linear systems $A_i x = b_i,\ i = 1, 2, \ldots, l$, with different A_i and b_i, which combines the idea of the GCRO method in [14] and that of the GMRES-DR method in [15].

GMRES-DR was proposed to accelerate the convergence rate of GMRES(m) when solving a single linear system $Ax = b$. By adding the harmonic Ritz vectors of A to the Krylov subspace which generates the approximate solution x_j at each iteration step j, GMRES-DR converges to the real solution more quickly than GMRES(m). The goal of GMRES-DR is to achieve a similar convergence rate as the non-restarted GMRES and to use as little storage space as GMRES(m).

When GCRO-DR is used to solve a single system, it is algebraically equivalent to GMRES-DR. Moreover, a recycling technique is used in GCRO-DR for a sequence of linear systems, i.e. before the current system $A_i x = b_i$ ($i > 1$) is solved, the harmonic Ritz vectors of A_{i-1} are recycled to modify the initial guess x_0 of $A_i x = b_i$. By recycling these harmonic Ritz vectors, GCRO-DR can solve the linear systems with much fewer matrix-vector (MV) products than the standard method GMRES and GMRES-DR. Therefore it is much more efficient than GMRES and GMRES-DR for solving many linear systems.

In the following, we explain the GCRO-DR algorithm given in Appendix A step by step.

Step 1) initializes the algorithm and computes the initial residual r_0 for the current linear system.

Step 2) decides whether the first system or a latter system is being solved. If $\tilde{Y}$ is not yet defined, it means the first system is being solved. In the following we first see what will be implemented if the first linear system is being solved.

If we are solving the first linear system, we go directly to Step 9), from which the first approximate solution x_1 in Step 12) is computed by running m steps of GMRES in Step 11). v_1 and c are the vectors used in the Arnoldi process in GMRES. The vector v_1 is the first column of V_{m+1} produced in Step 11). e_1 in Step 10) is the unit vector $e_1 = (1, 0, \ldots, 0)^T$ with the length of n, the dimension of the coefficient matrix A_i.

In Step 11), two matrices V_{m+1} and $\tilde{H}_m$ are generated by the Arnoldi process, which satisfy

$$\begin{aligned} A_i V_m &= V_m H_m + h_{m+1,m} v_{m+1} e_m^T \\ &= V_{m+1} \tilde{H}_m, \end{aligned} \tag{11}$$

where V_m corresponds to the first m columns of V_{m+1}, and v_{m+1} is the $m+1$th column of V_{m+1}. The matrix H_m is the $m \times m$ upper block in $\tilde{H}_m$, $h_{m+1,m}$ is the $(m+1, m)$ entry in $\tilde{H}_m$, and $e_m \in \mathbb{R}^m$, $e = [0, \ldots, 0, 1]^T$ is the unit vector of length m. The expression of the residual r_1 of x_1 in Step 13) can be obtained using the relation in (11).

The k eigenvectors of $(H_m + h_{m+1,m}^2 H_m^{-H} e_m e_m^H)$ are computed in Step 14) and stored in the matrix P. From [16], we know the k harmonic Ritz vectors of A_i can be computed by $\tilde{Y} = V_m P$ in Step 15). That is, the columns in $\tilde{Y}$ are the harmonic Ritz vectors of A_i, which are the approximate eigenvectors of A_i corresponding to the k smallest eigenvalues. For computation of harmonic Ritz vectors and Ritz values, one can refer to [16].

The matrices C and U are computed in Step 16) and Step 17). From the relation in (11), we get $C = AU$. This relation must be maintained, such that the residual r_1 computed in Step 7) and r_j computed in Step 27) satisfy $r_j = b_i - A_i x_j$, $j = 1, \ldots$.

The relations between U, $\tilde{Y}$ and C will be used to compute the approximate solutions x_j, $j > 1$ in the WHILE loop. Step 4) or Step 17) produces an orthonormal matrix C which satisfies $C^T C = I$. This is the reason why the operator $(I - CC^H)A$ is used in Step 21) such that the matrix V_{m-k+1} computed by the Arnoldi algorithm is automatically orthogonal to C. This relation guarantees that the harmonic Ritz vectors of A or the columns in $\tilde{U}$ are independent from the columns in V_{m-k+1}, so that $\hat{V}_m$ in Step 23) has full column rank. As a result, the columns of $\hat{V}_m$ are the basis of the subspace jointly spanned by the k harmonic Ritz vectors and the $m-k$ Arnoldi vectors. The idea of using the operator $(I - CC^H)A$ in Step 21) comes from the GCRO method [14]. Generally speaking, it ensures that new vectors are generated in $C^\perp$ and thus new information in $\hat{V}_m$ is added to the subspace. The new approximate solution x_j in Step 26) is generated from the subspace spanned by the basis $\hat{V}_m$.

Step 24) to Step 26) compute x_j whose residual is minimized over the subspace spanned by the columns in $\hat{V}_m$. Therefore r_j in Step 27) is actually minimized over the subspace jointly spanned by the k harmonic Ritz vectors of A_i and the $m-k$ vectors (columns in V_{m-k}) computed by Arnoldi.

The k eigenvectors of a small eigenvalue problem is computed in Step 28). From the relation

$$A\hat{V}_m = \hat{W}_{m+1} G_m \tag{12}$$

derived in equation (2.11) in [1], one can prove that the columns in $\tilde{Y}$ computed in Step 29) are the harmonic Ritz vectors of A_i. Step 28) and Step 29) are used to update the harmonic Ritz vectors, such that they approximate the eigenvectors of A more accurately. New C and U are formed in Step 31) and Step 32), which also maintains the relation $C = AU$ and can be proved using (12). The matrix U is given to $\tilde{Y}$ in Step 34) as the initial guess of the Harmonic Ritz vectors of A_{i+1}, which are used to modify the initial guess x_0 for the next linear system $A_{i+1} x = b_{i+1}$.

Now we study what will be implemented if we are solving the ith system for $i > 1$. The matrices C and U which are computed from the previous linear system are reused in Step 3)-Step

5) to generate the new C and U for the current linear system. With the assumption that generally the current coefficient matrix A_i is different from the previous coefficient matrix A_{i-1}, Step 3)-Step 5) must be implemented to keep the relation $C = A_iU$ for the current coefficient matrix A_i. Otherwise,we cannot guarantee that $C = A_iU$ though we have $C = A_{i-1}U$. The relation $C = A_iU$ must be maintained for each linear system $A_ix = b_i,\ i = 1,2,\ldots$ respectively.

The QR factorization of $A\tilde{Y}$ implemented in Step 3) tries to guarantee that the new matrix C satisfies $C^TC = I$, so that r_1 computed in Step 7) is minimized over the subspace spanned by U. Matrix U is modified in Step 5) such that the relation $C = A_iU$ is maintained. From Step 34), $\tilde{Y}$ is actually U computed from solving the previous system $A_{i-1}x = b_{i-1}$. Therefore, Step 5) also tells us that although the matrix U is modified, the subspace spanned by the columns of the new U is the same as the one spanned by the columns of the previous $U = \tilde{Y}$.

The initial guess x_0 is firstly modified to x_1 in Step 6). From Step 5), we see x_1 is in fact generated from the subspace spanned by the columns of the previous $U = \tilde{Y}$, which are the Harmonic Ritz vectors of the previous coefficient matrix A_{i-1}. Therefore, the harmonic Ritz vectors of A_{i-1} are recycled to modify the initial guess of the current system, such that x_1 is expected to be closer to the real solution. If $C^TC = I$, one can prove that r_1 computed in Step 7) is minimized by x_1 over the subspace spanned by the columns of U ([14]). After x_0 is modified to x_1, we go directly to Step 19), and the following steps are the same as for the first system.

The purpose of recycling the harmonic Ritz vectors of A_{i-1} is to make the modified solution x_1 closer to the real solution than x_0. However, if A_{i-1} is very different from A_i, then the harmonic Ritz vectors of A_{i-1} may also be very different from those of A_i. As a result, generating x_1 out of the subspace spanned by the harmonic Ritz vectors of A_{i-1} may produce a x_1 which is even worse than the initial guess x_0. Therefore, to make the recycling of the harmonic Ritz vectors of A_{i-1} meaningful, A_i should not vary much from A_{i-1}, such that the harmonic Ritz vectors of A_{i-1} have small difference from those of A_i, which may produce a x_1 closer to the real solution than x_0. However, this does not mean that, e.g. A_{i+100} should not vary much from A_{i-1} either. Actually, A_{i+100} may be very different from A_{i-1}, but it should be very close to A_{i+99}, because the linear system $A_{i+100}x = b_{i+100}$ recycles the harmonic Ritz vectors of A_{i+99} rather than those of A_{i-1}. Finally, we conclude that for the sequence of linear systems $A_ix = b_i,\ i = 1,2,\ldots,l$, the difference between the neighboring coefficient matrices should be as small as possible, though there might be much difference between the foremost one and the last one. More theoretical results can be found in [1].

The idea of including the harmonic Ritz vectors into the Krylov subspace generated by Arnoldi comes from the method GMRES-DR. As is mentioned above, GMRES-DR is a method more efficient than GMRES especially for linear systems having extreme eigenvalues, e.g. eigenvalues of very small magnitude [15]. Therefore, in GCRO-DR, GMRES-DR rather than GMRES is employed to compute x_j. In this way, GCRO-DR not only speeds up GMRES (by using GMRES-DR) for solving a single system, but also further accelerates GMRES-DR by recycling harmonic Ritz vectors when solving the entire sequence of linear systems.

From the algorithm, we see the Ritz vectors must be computed and modified in solving each linear system. However if the sequence of the linear systems have the same coefficient matrix $A_i = A,\ i = 1,2,\ldots$, the above algorithm can be simplified and more efficiency can be achieved. In the next subsection, we will introduce SimGCRO-DR proposed in [24], which is a simplified version of GCRO-DR, and which is especially suitable for the linear systems in (1) or (10).

4.2 Analysis of SimGCRO-DR

From (6)(9)(10) one can see if the transfer function is expanded around one set of expansion points $p_0 = [s_1^0, s_2^0, \dots, s_p^0]$, the consequential sequence of linear systems will have the same coefficient matrix $\tilde{E}$. For this case, a simplified version SimGCRO-DR is proposed in [24] (see Appendix B).

Generally speaking, when all the linear systems have the same coefficient matrix, Step 3)-Step 5) in GCRO-DR (see Appendix A) are not necessary. Step 28)-Step 32) and Step 34) are only needed for the first linear system. Since we have the same coefficient matrix, the relation $C = AU$ maintained in the previous linear system $Ax = b_{i-1}$ can be naturally maintained in the next linear system $Ax = b_i$. Therefore, we do not have to implement Step 3) and Step 5) to modify C and U, which can save computation time and thus can improve the efficiency of GCRO-DR. The matrices C and U can be directly reused to modify the initial guess x_0.

The harmonic Ritz vectors of A span an invariant subspace of A, which should not change with the different right hand sides b_i, hence once we have computed the harmonic Ritz vectors $\tilde{Y}$ in the course of solving the first system, $\tilde{Y}$ does not have to be modified for the latter linear systems. Therefore, Step 28)-Step 32) need not be repeatedly implemented at each iteration step for each of the linear systems with $i > 1$, which again reduce computation.

By removing the above steps, GCRO-DR can be simplified to SimGCRO-DR. We show in Section 5 that SimGCRO-DR is more efficient than GCRO-DR for the sequence of linear systems in (10) or (1).

4.3 G-DRvar1 and G-DRvar2

For many parametric systems with more than two parameters, expanding the transfer function around only one set of parameters (single-point expansion) is insufficient. Instead, the transfer function needs to be expanded around several sets of parameters (multi-point expansion), such that the accuracy of the reduced model can be improved and the size of the reduced model can be kept small. Once the transfer function is expanded around $l_o > 1$ sets of parameters, e.g. $p_j = [s_1^j, s_2^j, \dots, s_p^j]$, $j = 1, 2, \dots, l_o$, then l_o sequences of linear systems

$$\tilde{E}(p_j)x = w_i(p_j),\ i = 1, 2, \dots, l_j,\ j = 1, 2, \dots, l_o \tag{13}$$

must be solved, which are of similar forms as the linear systems in (2). The jth sequence of linear systems with the same coefficient matrix $\tilde{E}(p_j)$ correspond to the jth set of expansion points p_j, $j = 1, 2, \dots, l_o$.

Since the coefficient matrix only changes every l_j systems, and the number l_j is usually much larger than l_o, SimGCRO-DR can be repeatedly applied for l_o times to get the solutions for all the linear systems. At each time, SimGCRO-DR solves a sequence of linear systems with the same coefficient matrix $\tilde{E}(p_j)$. However for each $j > 1$, when solving $E(p_j) = w_1(p_j)$ (notice here $i = 1$ in (13)), SimGCRO-DR does not reuse the harmonic Ritz vectors from the previous system $\tilde{E}(p_{j-1})x = w_{l_{j-1}}(p_{j-1})$, instead it recomputes the harmonic Ritz vectors of $\tilde{E}(p_j)x = w_1(p_j)$, $i = 1$ and then reuses them until $i = l_j$ for the current j. Therefore, SimGCRO-DR actually overlooks some information which could be recycled to further improve the computational efficiency. It becomes inefficient in solving such sequence of linear systems in (13) or (2), which can be seen from the simulation results.

The l_o sequences of linear systems can also be considered as one sequence, where the coefficient matrices change only every l_j systems. If they are considered as a whole, the original algorithm GCRO-DR can also be applied, and the harmonic Ritz vectors for each system (except for the last one $\tilde{E}(p_{l_o})x = w_{l_o}(p_{l_o})$) can be reused to solve the current linear system. The trade-off is that the harmonic Ritz vectors have to be updated for each system at Step 28)-Step 32), which is sometimes unnecessary, especially for the sequence of linear systems like (2)(13).

In the following we propose a variant of GCRO-DR, which is named G-DRvar1. In this algorithm, the information which is neglected in SimGCRO-DR is reused, and the redundant computation in GCRO-DR is removed to further improve the efficiency of both GCRO-DR and SimGCRO-DR.

For each system in (13), G-DRvar1 recycles the harmonic Ritz vectors of the coefficient matrix of the previous system, but only updates the harmonic Ritz vectors of the coefficient matrices of the l_o linear systems $\tilde{E}(p_j)x = w_1(p_j)$, $j = 1, 2, \ldots, l_o$. That means, once the coefficient matrix changes from $\tilde{E}(p_{j-1})$ to $\tilde{E}(p_j)$, the harmonic Ritz vectors of $\tilde{E}(p_{j-1})$ will be recycled and updated during solving $\tilde{E}(p_j)x = w_1(p_j)$, but they will only be reused and not be updated during solving $\tilde{E}(p_j)x = w_i(p_j)$, $1 < i \leq l_j$. In order to be consistent with GCRO-DR, we use the sequence of linear systems in (2) to describe G-DRvar1. The application to the linear systems in (13) is straightforward simply by replacing A_j with $E(p_j)$.

Algorithm 1 G-DRvar1(*m*, *k*) solving (2)

```
 1. For $j = 1 : l_o$
 2. For $i = 1 : l_j$
 3. If $i = 1$ and $j = 1$
 4.    Implement Step 1) of algorithm GCRO-DR.
 5.    Implement Step 9)-Step 34) of algorithm GCRO-DR.
 6. Else if $i = 1$ and $j > 1$
 7.    Implement Step 2)-Step 7) of algorithm GCRO-DR.
 8.    Implement Step 19)-Step 34) of algorithm GCRO-DR.
 9. Else if $i > 1$ and $j > 1$
10.    Implement Step 2)-Step 7) of algorithm GCRO-DR.
11.    Implement Step 19)-Step 27) and Step 34) of algorithm GCRO-DR.
```

```
12. End If
13. End For (i)
14. End For (j)
```

When solving the current linear system, G-DRvar1 reuses the harmonic Ritz vectors of the previous linear system by implementing Step 2)-Step 7) of algorithm GCRO-DR. Once the harmonic Ritz vectors of A_j is computed while solving the linear system $A_jx = b_1^j$ (notice here $b_i^j = b_1^j$), they will not be recomputed for the latter linear systems with the same coefficient matrix A_j. This can be seen from Step 11) in G-DRvar1. As a result, more information is recycled in G-DRvar1 than in SimGCRO-DR and less computation is implemented in G-DRvar1 than in GCRO-DR. The efficiency of G-DRvar1 can be shown by the MEMS example butterfly gyroscope in Section 5.

In principle, for the cases of $i > 1$ and $j > 1$, Step 10) can be further simplified, because the corresponding linear systems have the same coefficient matrix A_j as the linear system $A_jx = b_1^j$. Since the harmonic Ritz vectors of A_j have been computed during solving $A_jx = b_1^j$, and A_j do not change for b_i^j, $i > 1$, the relation $C = A_jU$ remains for the systems $A_jx = b_i^j$, $i > 1$. Therefore, Step 2)-Step 5) of GCRO-DR implemented in Step 10) of G-DRvar1 is not necessary, and more computation is saved. Here we name the further simplified algorithm as G-DRvar2.

Algorithm 2 G-DRvar2(m, k) solving (2)

```
1.  For $j = 1 : l_o$
2.  For $i = 1 : l_j$
3.  If $i = 1$ and $j = 1$
4.      Implement Step 1) of algorithm GCRO-DR.
5.      Implement Step 9)-Step 34) of algorithm GCRO-DR.
6.  Else if $i = 1$ and $j > 1$
7.      Implement Step 2)-Step 7) of algorithm GCRO-DR.
8.      Implement Step 19)-Step 34) of algorithm GCRO-DR.
9.  Else if $i > 1$ and $j > 1$
10.     Implement Step 6)-Step7) of algorithm GCRO-DR.
11.     Implement Step 19)-Step 27) and Step 34) of algorithm GCRO-DR.
12. End If
13. End For (i)
```

14. End For (j)

We see the only difference of G-DRvar2 from G-DRvar1 is Step 10), where less computation is done by G-DRvar2. However, from the simulation results for the butterfly gyroscope, we find if Step 10) is simplified as in G-DRvar2, the convergence rate of G-DRvar2 is slower than G-DRvar1 for the butterfly gyroscope, whereas G-DRvar2 converges faster than G-DRvar1 for the other two examples, microthruster and the microhotplate gas sensor, which is in agreement with our analysis.

4.4 Integration of the recycling algorithm with PMOR

In this Subsection, we show how the recycling algorithms are applied to PMOR to compute the final projection matrix V for the reduced model. Assume we expand the transfer function around l_o sets of expansion points $p_j,\ j=1,2,\dots,l_o$. For each set p_j, we compute a matrix V_j by

$$\text{range}\{V_j\} = \text{span}\{R_0, R_1, \dots, R_{r_j}\},\ j=1,2,\dots,l_o. \tag{14}$$

From the definition of $R_i,\ i=1,2,\dots,r_j$, $\tilde{E}$ is included in each term of R_i and $\tilde{E}$ actually depends on the set of expansion points p_j. Therefore, it can be written as $\tilde{E}(p_j)$. In order to distinguish the subspace spanned by $R_i,\ i=1,2,\dots,r_j$ corresponding to different p_j, and indicate the dependence of $R_i,\ i=1,2,\dots,r_j$ upon p_j, we rewrite (14) into

$$\text{range}\{V_j\} = \text{span}\{R_0(p_j), R_1(p_j), \dots, R_{r_j}(p_j)\},\ j=1,2,\dots,l_o, \tag{15}$$

where

$$\begin{aligned}
&R_0(p_j) = B_M(p_j),\\
&R_1(p_j) = [M_1(p_j)R_0(p_j), \dots, M_p(p_j)R_0(p_j)],\\
&R_2(p_j) = [M_1(p_j)R_1(p_j), \dots, M_p(p_j)R_1(p_j)],\\
&\vdots,\\
&R_{r_j}(p_j) = [M_1(p_j)R_{r_j-1}(p_j), \dots, M_p(p_j)R_{r_j-1}(p_j)],
\end{aligned} \tag{16}$$

and $B_M = \tilde{E}(p_j)^{-1}B$, $M_i = -\tilde{E}(p_j)^{-1}E_i$, $i=1,2,\dots,p$ and

$$\tilde{E}(p_j) = E_0 + s_1^j E_1 + s_2^j E_2 + \cdots + s_p^j E_p. \tag{17}$$

The final matrix V is computed by orthogonalizing $V_j,\ j=1,2,\dots,l_o$,

$$V = \text{orthogonalize}\{V_1, V_2, \dots, V_{l_o}\}. \tag{18}$$

If some of the expansion points in a set, e.g. p_{j_0}, are complex numbers, such as those expansion points for the variable s in the Laplace domain, then the corresponding matrix V_{j_0} has to be divided into real and imaginary parts and the matrix V is then computed by

$$V = \text{orthogonalize}\{V_1, V_2, \dots, Re(V_{j_0}), Im(V_{j_0}), \dots, V_{l_o}\} \tag{19}$$

such that V is a real matrix and the resultant reduced model has real system matrices. Parametric model reduction combined with the recycling method G-DRvar1 can be described in Algorithm 3.

Algorithm 3 PMOR combined with G-DRvar1

```
1.  For $j = 1 : l_o$ (compute $V_j$)
2.  $V_j = [\,]$, an empty matrix.
3.      If computing the first column of $R_0(p_1) = B_M(p_1)$,
        implement
4.          Solve $\tilde{E}(p_1)x = B(:,1)$ by implementing
            Step 4)-Step 5) of G-DRvar1 and get x.
5.          If $x \neq \mathbf{0}$
6.              $V_j = x/||x||_2$.
7.          EndIf
8.      ElseIf computing the first column of $R_0(p_j)$ and $j \neq 1$
9.          Solve $\tilde{E}(p_j)x = B(:,1)$ by implementing
            Step 7)-Step 8) of G-DRvar1 and get x.
10.         Orthogonalize x with respect to all the columns in
            $V_j$ and get w
11.         If $w \neq \mathbf{0}$
12.             $V_j = [V_j, w/||w||]$.
13.         EndIf
14.     Else
15.         Solve $\tilde{E}(p_j)x = b(p_j, B)$ by implementing
            Step 10)-Step 11) of G-DRvar1 and get x.
16.         Orthogonalize x with respect to all the columns in
            $V_j$ and get w
17.         If $w \neq \mathbf{0}$
18.             $V_j = [V_j, w/||w||]$.
19.         EndIf
20.     End If
21. End For
```

```
22. $V = \text{orthogonalize}\{V_1, V_2, \dots, V_{l_o}\}$.
```

In Algorithm 3, the right hand side vector $b(p_j,B)$ is related to p_j and/or B. For the computation of the tth column of $R_0(p_j)$ (except for the first column), the right hand side vector $b(p_j,B) = B(:,t)$, $t = 2,\dots,d_I$, where d_I is the number of inputs and it is also the number of columns in the input matrix B in (3).

For the computation of the tth column of $M_i(p_j)$, $i = 1,2,\dots p$, $j = 1,2,\dots,l_o$ in $R_q(p_j)$, $0 < q < r_j$, the right hand side vector $b(p_j,B) = E_iV_{R_{q-1}}(:,t)$, $t = 2,\dots,col$; col is the number of columns in $V_{R_{q-1}}$. Either $B(:,t)$ or $V_{R_{q-1}}(:,t)$ is the tth column of B or $V_{R_{q-1}}$. At the step of computing the vectors in $R_q(p_j)$, the vectors in $R_{q-1}(p_j)$ are already orthogonalized to $V_{R_{q-1}}$. Therefore, when generating the orthogonal vectors in $R_q(p_j)$, $R_{q-1}(p_j)$ is replaced by $V_{R_{q-1}}$.

Step 3), Step 8) and Step 15) are automatically implemented according to the PMOR process in [21] which for clarity, is not shown in detail here.

Algorithm 3 indicates that the linear systems involved in PMOR do not appear continuously. In fact, after solving one linear system, the current solution is first orthogonalized with respect to the columns in V, and the next linear system is solved afterwards. Except for the first linear system $\tilde{E}(p_1)x = B(:,1)$, all the other linear systems in the whole sequence have reused the harmonic Ritz vectors generated from solving the previous linear system. The recycling algorithms G-DRvar2, SimGCRO-DR and GCRO-DR can also be integrated with PMOR in a similar way.

4.5 Comparison with MKR-GMRES

The recycling algorithms are more efficient than a recently developed recycling algorithm MKR-GMRES in [2, 3], where the Krylov subspaces generated by Arnoldi are recycled. However, the dimension of the recycled subspace increases rapidly with the number of the linear systems. This is because all the vectors in the previous Krylov subspaces are recycled. If there are many linear systems in the sequence, the dimension of the recycled subspace approaches n (the dimension of the coefficient matrix A_i) quickly. As a result, the algorithm becomes very slow because of the quick inflation of the recycled subspace and the huge memory occupation for storage of all the vectors in the subspace. MKR-GMRES for solving a sequence of linear systems $A_ix = b_i$, $i = 2,\dots,l$ is described in Algorithm 4.

Algorithm 4 *MKR-GMRES solving the ith system in the sequence of linear systems* $A_ix = b_i$, $i = 1,2,\dots,l$.

```
1) Calculate pseudo-inverse $P^+$ of $P^i$. If $i = 1$,
   let $P^1 = 0$ and its $P^+ = 0$.
2) Let $\mathscr{P}_s = P^i P^+$.
3) Calculate $r = (I - \mathscr{P}_s)b_i$ and $x = V^i P^+ b_i$.
4) Generate Krylov subspace $V_m$ using Arnoldi process
     $(I - \mathscr{P}_s)A_i V_m = V_{m+1}\tilde{H}_m$.
5) Solve least squares problem
     $y = \arg\min \|r - A_i[V^i,\ V_m]y\|$.
   If the error is satisfactory, go to Step 6, otherwise
   let $m = m + 1$ and go to Step 4.
6) Save $V^{i+1} = [V^i,\ V_m]$, $P^{i+1} = A_i V^{i+1}$ for the next
   system, and let $x = x + V^{i+1}y$ be the solution.
```

Problems caused by the subspace inflation could be the pseudo-inverse computation and least squares problem in Step 5).

From Algorithm 4, we can see that one need not explicitly compute the pseudo-inverse P^+ of P^i. The only necessary computation related to P^+ is the application of P^+ to a vector w. This can be achieved by solving a least-squares problem, which in a MATLAB implementation is simply achieved by using the command "\". However, when the number of columns in P^i becomes large, the computation of $P^i\ w$ becomes very slow, and becomes even more expensive than Aw. In [3], only detailed analysis on how to reduce the complexity of computing $(P^i)^T P^i$ is presented. However, in order to compute P^+w, the computational difficulty arises from the fact that the matrix P^i has dense columns and a singular value decomposition (SVD) or rank-revealing QR decomposition must be computed to solve the least-squares problem.

Similarly, Step 5) also becomes difficult to implement due to the increase of the columns in V^i. Unfortunately in order to recycle the subspace V^i, the dimensional increase in both P^i and V^i is inevitable. All the above issues make the algorithm not as efficient as expected. We illustrate the above problems by numerical simulations in the next section.

5 Simulation Results

In this section, the recycling algorithms GCRO-DR, SimGCRO-DR, G-DRvar1 and G-DRvar2 are compared with the standard solver GMRES as well as GMRES-DR. The methods G-DRvar1, G-DRvar2 are also compared with the original version GCRO-DR and the previous version SimGCRO-DR. Only SimGCRO-DR is compared with MKR-GMRES proposed in [2, 3], which is sufficient to illustrate the problem of MKR-GMRES. The accuracy of the reduced model by applying SimGCRO-DR, G-DRvar1 and G-DRvar2 is also presented.

5.1 Examples

We use three examples to show the efficiency of the recycling algorithms. The first example is a microthruster. In Fig. 1, the upper-left part[1] is the structure of an array of pyrotechnical

[1]The picture is taken from the paper [25], we acknowledge the author's permission for using the picture.

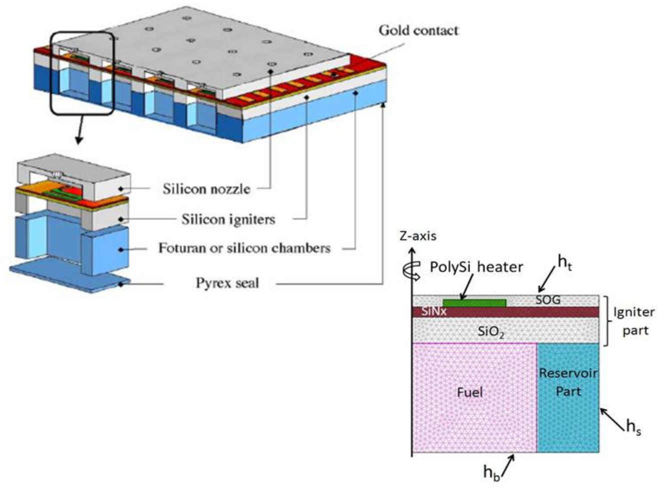

Figure 1: A model of the microthruster unit.

thrusters. The lower-right part of Fig. 1 is the structure of a 2D-axisymmetric model for a single microthruster. When the PolySilicon (green) in the middle is excited by a current, the fuel below is ignited and the explosion will occur through the nozzle. The thermal process can be modeled by a heat transfer partial differential equation, while the heat exchange through device interfaces is modeled by convection boundary conditions with different film coefficients h_t, h_s, h_b. The film coefficients h_t, h_s, h_b respectively describe the heat exchange on the top, side, and bottom of the microthruster with the outside surroundings. The values of the film coefficients can change from 1 to 10^9 [23]. After finite element discretization of the 2D-axisymmetric model, a parameterized system is derived,

$$\begin{aligned} E\dot{x} &= (A - h_t A_t - h_s A_s - h_b A_b)x + B \\ y &= Cx. \end{aligned} \tag{20}$$

Here, h_t, h_s, h_b are the parameters and the dimension of the system is $n = 4,257$. We observe the temperature at the center of the PolySilicon heater changing with time and the film coefficient, which defines the output of the system[2].

The second example is a butterfly gyroscope, and the parameterized system is obtained by finite element discretization of the model for the gyroscope, see Fig. 2 (The details of the model can be found in [22]). The system is of the following form:

$$\begin{aligned} M(d)\ddot{x} + D(\theta,\alpha,\beta)\dot{x} + T(d)x &= Bu(t) \\ y &= Cx. \end{aligned} \tag{21}$$

Here, $M(d) = (M_1 + dM_2)$, $D(\theta,\alpha,\beta) = \theta(D_1 + dD_2) + \alpha M(d) + \beta T(d)$, and $T(d) = (T_1 + \frac{1}{d}T_2 + dT_3)$. The variables d, θ, α, β are the parameters of the system. d is the width of the

[2]Detailed description of the parameterized system can be find at http://simulation.uni-freiburg.de/downloads/benchmark

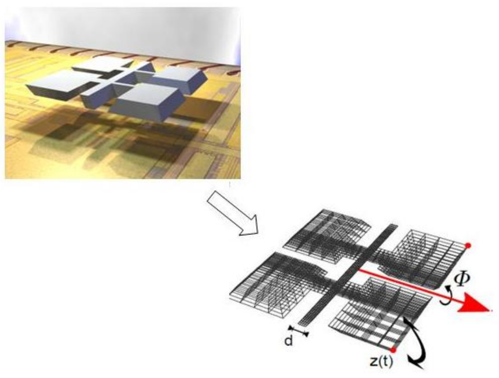

Figure 2: Scheme of the butterfly gyroscope [22].

bearing, and θ is the rotation velocity along the x axis. The parameters α, β are used to form the Rayleigh damping matrices $\alpha M(d), \beta T(d)$ in $D(\theta, \alpha, \beta)$. The interesting output of the system is δz, the difference of the displacement $z(t)$ between the two end nodes depicted as red dots on the same side of the bearing (see Fig. 2). Referring to the explanation in [22], the paddles of the device are excited to a vibration $z(t)$, where all paddles vibrate in phase. With the external rotation ϕ the Coriolis force acts upon the paddles, which causes an out-of-phase movement which is measured as the z-displacement difference δz between the two red dotted nodes. The dimension of the system is $n = 17913$.

The third example is a microhotplate gas sensor chip [29]. The heat transfer within the sensor is illustrated in Fig. 3 [29]. The model of the heat transfer inside the microhotplate gas sensor is a system with four parameters [30], the mass density ρ in kg/m^3, the specific heat capacity c_p in J/kg/K, the thermal conductivity in W/m/K, and the heat transfer coefficient h in W/m^2/K. The dimension of the system is $n = 60020$.

$$\begin{array}{rcl} (E_0 + \rho c_p E_1)\dot{x} + (K_0 + \kappa K_1 + hK_2)x & = & Bu(t) \\ y & = & Cx. \end{array} \tag{22}$$

5.2 Criteria for comparison

We use the same convergence criterion ($tol = 10^{-7}$) for all the methods, i.e. once the residual of the current approximate solution of a linear system is smaller than tol, the algorithm stops and the solution is obtained. We let the solution produced by all the methods be minimized over the subspaces of the same dimension m. We use the same k, the number of harmonic Ritz vectors, for each recycling algorithm as well as for GMRES-DR. GMRES(∞) connotes GMRES method without restarts and GMRES(m) connotes the method with restarts. We implement GMRES method by running the function "gmres.m" in MATLAB® version 2007b. All the other methods are also programmed in the same version of MATLAB.

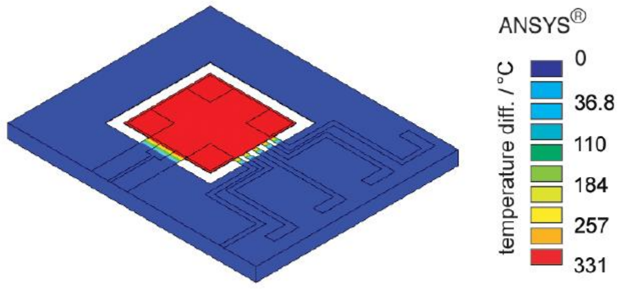

Figure 3: Temperature distribution over the microhotplate gas sensor chip after 5s of heating with constant heating power of 340mW. (Picture taken from [29])

Any iterative methods may become extremely slow if without preconditioning. Therefore, when implementing the recycling algorithms, GMRES-DR and the standard solvers, we also use preconditioning to accelerate the convergence rate. In the following simulation results, incomplete LU factorization is used as the preconditioner for all the algorithms, which is `luinc`$(\cdot, 0.001)$ in MATLAB notation. Once the preconditioner is applied, the number of MV products will be counted based on the preconditioned algorithm. Because the two factors L and U are triangular matrices, it is reasonable to count one time of preconditioning by L and U as two MV products for simplicity.

We employ the same comparison criterion as that used in [1] to compare the recycling algorithms with the standard solver GMRES and GMRES-DR, i.e. the MV products used in the algorithm for solving each linear system. This is a reasonable criterion because the number of MV products constitutes the main computation in all the algorithms, which in some sense, corresponds to the number of iteration steps.

When compare different recycling algorithms, we not only compare the MV products but also compare the CPU time used by each algorithm. It can be justified by the difference between SimGCRO-DR, G-DRvar1, G-DRvar2, and the original version GCRO-DR summarized as below.

- For solving the linear systems in (1), SimGCRO-DR has saved the MV products and QR factorization in Step 3) of GCRO-DR for all the linear systems except for the first. It also has saved the repeated computation of harmonic Ritz vectors and QR factorization in Step 28)-Step 32) of GCRO-DR for these linear systems.

- G-DRvar1 actually saves the repeated computation of the harmonic Ritz vectors of A_j and QR factorization in Step 28)-Step 32) of GCRO-DR for the linear systems $A_jx = b_i^j$ with $i > 1$, and it does not reduce MV products of GCRO-DR nevertheless.

- G-DRvar2 has saved not only the MV products and QR factorization in Step 3) of GCRO-DR for each of the linear systems $A_j x = b_i^j$ with $i > 1$ and $j > 1$, but also the repeated computation of the harmonic Ritz vectors and QR factorization in Step 28)-Step 32) of GCRO-DR for these systems.

Therefore, if we only compare the MV products, we have neglected the repeated computation of the harmonic Ritz vectors and QR factorization saved by these methods. Since the harmonic Ritz vectors are computed by solving a small eigenvalue problem, the CPU time spent on the harmonic Ritz vectors is much less than the CPU time spend on the MV products, especially when the linear systems are of very large dimension. However, methods SimGCRO-DR, G-DRvar1 and G-DRvar2 do not save the computation of one small eigenvalue problem, rather the computation of hundreds of such small eigenvalue problems which may not be neglected.

As is analyzed in Section 4, the main computational complexity in MKR-GMRES is constituted by the computation of the pseudo-inverse of a large matrix and the least squares solution of a large dimensional problem, therefore it is not reasonable to compare MKR-GMRES with SimGCRO-DR with MV products. We compare the two methods by CPU time spent on solving each system. Although the CPU time varies from computer to computer, the simulation results are enough to differentiate the two methods from one another because of the large difference of CPU times between the two methods when they are run on the same computer.

In the next Subsection, we show the simulation results for the thermal process in the microthruster. The simulation results of the butterfly gyroscope and the microhotplate gas sensor are analyzed in Subsection 5.4 and 5.5 .

5.3 Simulation of the microthruster

By applying the Laplace transform to the system in (20), we get

$$\begin{array}{rcl} sEx &=& (A - h_t A_t - h_s A_s - h_b A_b)x + Bu(s) \\ y &=& Cx. \end{array} \tag{23}$$

Here, s is the variable in the frequency domain, and there are totally 4 parameters in (23) including s. Following the PMOR method introduced in Section 2, we may deal with either of the two cases below.

- Case A, if the state vector x is expanded around one set of expansion points, e.g. $p_0 = [s^0, h_t^0, h_s^0, h_b^0]$, the sequence of the linear systems are $\tilde{E}x = w_i,\ i = 1, 2, \ldots, l$. Here, $\tilde{E} = -A + s^0 E + h_t^0 A_t + h_s^0 A_s + h_b^0 A_b$, and $\tilde{E}$ is nonsymmetric. The number of linear systems depends on how many moment matrices of the system are used. For this example, 600 linear systems are solved to match enough moments and guarantee the accuracy of the reduced model. We take expansion points $s^0 = 1, h_t^0 = h_s^0 = h_b^0 = 1$ and the reduced model is of dimension $q = 325$.

- Case B, a reduced model of much smaller dimension can be obtained by using multi-point expansion, i.e. by expanding x in (23) around several sets of expansion points. Here, we take $p_0 = [s^0, h_t^0, h_s^0, h_b^0] = [1, 10, 10, 10]$, $p_1 = [s^1, h_t^1, h_s^1, h_b^1] = [1, 10^2, 10^2, 10^2]$, $p_2 = [s^2, h_t^2, h_s^2, h_b^2] = [1, 10^6, 10^6, 10^6]$ and $p_3 = [s^3, h_t^3, h_s^3, h_b^3] = [1, 5 \times 10^8, 5 \times 10^8, 5 \times$

10^8]. The sequence of linear systems are of the form in (13), with $l_1 = 21, l_2 = 21, l_3 = 21$, $l_4 = 21$ and $l_o = 3$. In total, 84 linear systems have been solved, and the corresponding reduced model is of dimension $q = 82$ which is much smaller than the reduced model obtained by single-point expansion in Case A.

Notice that although the expansion points p_j used here are very different from each other, they have not caused much difference in the corresponding coefficient matrices $\tilde{E}(p_j),\ j = 1,2,\ldots,l_o$. Hence, the neighboring matrices $\tilde{E}(p_{j-1})$ and $\tilde{E}(p_j)$ are still close to each other. This is because that the magnitudes of the entries in A dominate the magnitudes of the entries in $\tilde{E}(p_j) = -A + s^j E + h_t^j A_t + h_s^j A_s + h_b^j A_b,\ j = 1,2,\ldots,l_o$. The maximal magnitude of the entries in A is O(10), while the maximal magnitude of the entries in either A_t, A_s or A_b is O(10^{-7}). Therefore, when we change the expansion points, we only change the part of $\tilde{E}(p_j)$ associated with the parameters, which actually cause relatively small changes in $\tilde{E}(p_j)$. Finally, it can be expected that the recycling algorithms may achieve much efficiency for such a sequence of linear systems.

5.3.1 Results for Case A

For Case A, the sequence of linear systems are in the form of (1); therefore, SimGCRO-DR is used to solve them and is compared with the original version GCRO-DR, as well as GMRES-DR and the standard solver GMRES. In Table 1, we list the number of MV products used by each algorithm corresponding to different groups of k (the number of harmonic Ritz vectors) and m (the number of restarts or equivalently, the dimension of the subspace which generates the approximate solution for each system). In the table, "average MV" is the average number of MV products used for solving a single linear system, and "total MV" is the total number of MV products for solving the whole sequence of linear systems. "total time" is the CPU time used for solving all the linear systems. We see the two recycling algorithms are much

Table 1: SimGCRO-DR vs GCRO-DR, GMRES for Case A

Methods	m	k	average MV	total MV	total time (s)
SimGCRO-DR	30	20	175	105,060	516.61
GCRO-DR	30	20	182	109,197	1053.5
GMRES(m)	30	—	2584	15,501,118	1707.4
Methods	m	k	average MV	total MV	total time (s)
SimGCRO-DR	40	30	145	87,120	534.55
GCRO-DR	40	30	141	84,357	1196.2
GMRES(m)	40	—	1871	1,122,690	1638.8
Methods	m	k	average MV	total MV	total time (s)
SimGCRO-DR	50	30	162	97,290	666.81
GCRO-DR	50	30	157	94,167	1209.8
GMRES(m)	50	—	1378	826,677	1280.9
GMRES(∞)	—	—	264	158,394	581.96

more efficient than the restarted GMRES(m). They also use much fewer MV products than the non-restarted GMRES(∞). However, GCRO-DR uses more CPU time than GMRES(∞). This may be due to our usage of the function "gmres.m" in MATLAB, which is already numerically optimized. What's more important is that it is well-known that GMRES(∞) uses significantly more memory storage, therefore it is actually not applicable for very large systems, which can be seen from the simulation of the other two examples.

Usually the number of MV products of GMRES(m) can be reduced by improving the number of restarts m, and if we take $m = n$ (the dimension of the coefficient matrix), then the MV products are reduced to the MV products used by GMRES(∞), because these two methods are equivalent in this situation. However the memory storage is sacrificed at the same time, which is not suitable for very large systems either.

The two recycling algorithms use more or less the same number of average MV products for each linear system, whereas SimGCRO-DR uses only half of the CPU time used by GCRO-DR. It tells us that SimGCRO-DR has saved much computation of the MV products and QR factorization in Step 3) as well as the computation of the harmonic Ritz vectors and QR factorization in Step 28)-Step 32), which cannot be ignored. We conclude that SimGCRO-DR attains a similar convergence rate as GCRO-DR, while saving much extra computation implemented by GCRO-DR.

The method GMRES-DR begins to stagnate after solving the first linear system, which is not listed in the table.

In Fig. 4, the error of the output response of the reduced model in the time domain is plotted. The reduced model is obtained by applying SimGCRO-DR to PMOR. The error is the 2-norm relative error between the output of the reduced model and that of the original system on the whole range of the time interval. By fixing h_t (here $h_t = 100$), Fig. 4 indicates the error varying with the two parameters h_s, h_b. We take 529 groups of h_s and h_b from the whole range $[1, 10^9]$. The maximum error of the reduced model over all the groups of the parameters is around 10^{-7}, much smaller than the requirement of the maximum error 0.05 for real-world applications.

Typical important sets of the film coefficients h_t, h_s, h_b are given in [23]. In Table 2, Table 3 and Table 4, we show the errors of the reduced model corresponding to various sets of the film coefficients taken from TABLE VI in [23]. From the results in the tables, we see that the reduced model is accurate for all interesting values of the parameters listed there. One can also see that the error at $h_t = h_b = h_s = 10^9$ in Table 4 (though it is acceptable) is much larger than the others. This is mainly because we have used the expansion points $h_t^0 = h_s^0 = h_b^0 = 1$, therefore the reduced model could cause larger error for the values far away from the expansion points. However the accuracy can be improved by the multi-point expansion studied in Case B.

Fig. 5 includes the MV products used in MKR-GMRES. The number of MV products used for the first system is 2337, which is much more than the numbers for the other systems. For clarity, it is therefore not included in the figure. Although the number of MV products for each system in MKR-GMRES is relatively small, the corresponding CPU time becomes larger and larger, which is shown in the figure on the bottom. It is not surprising from the above analysis that the main computational complexity of MKR-GMRES are not the MV products but the high dimensional pseudo-inverse computation and the least squares solution in Step 5). Moreover, MKR-GMRES is also slowed by the large memory requirements due to storage of the recycled subspace, which could amount to dimension n, the dimension of the

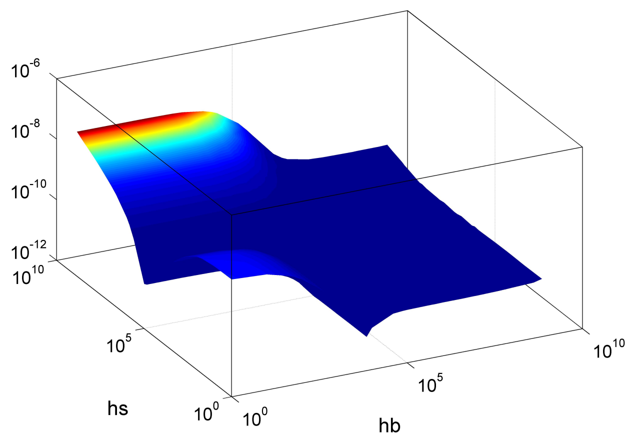

Figure 4: Error of the reduced model produced by SimGCRO-DR.

Table 2: error of the reduced model vs. the film coefficients for the microthruster

No.	h_t	h_b	h_s	error
1	5	1	5	9.239×10^{-9}
2	5	10	5	9.244×10^{-9}
3	5	25	5	8.452×10^{-9}
4	5	50	5	7.834×10^{-9}
5	5	100	5	6.557×10^{-9}
6	15	1	15	8.099×10^{-9}
7	15	10	15	7.745×10^{-9}
8	15	25	15	7.297×10^{-9}
9	15	50	15	6.616×10^{-9}
10	15	100	15	5.540×10^{-9}
11	30	5	30	5.871×10^{-9}
12	30	30	30	5.424×10^{-9}
13	30	50	30	5.102×10^{-9}
14	30	200	30	3.622×10^{-9}
15	80	5	80	3.097×10^{-9}
16	80	30	80	2.952×10^{-9}
17	80	50	80	2.848×10^{-9}
18	80	200	80	2.264×10^{-9}
19	200	5	200	1.373×10^{-9}

Table 3: error of the reduced model vs. the film coefficients for the microthruster

No.	h_t	h_b	h_s	error
20	200	30	200	1.337×10^{-9}
21	200	50	200	1.311×10^{-9}
22	200	200	200	1.149×10^{-9}
23	25	1	5	9.257×10^{-9}
24	25	10	5	8.899×10^{-9}
25	25	25	5	8.380×10^{-9}
26	25	50	5	7.696×10^{-9}
27	25	100	5	6.333×10^{-9}
28	75	1	15	6.361×10^{-9}
29	75	10	15	6.162×10^{-9}
30	75	25	15	5.847×10^{-9}
31	75	50	15	5.374×10^{-9}
32	75	100	15	4.658×10^{-9}
33	150	5	30	4.241×10^{-9}
34	150	30	30	3.949×10^{-9}
35	150	50	30	3.766×10^{-9}
36	150	200	30	2.827×10^{-9}
37	500	5	200	9.926×10^{-10}
38	500	30	200	9.694×10^{-10}

Table 4: error of the reduced model vs. the film coefficients for the micorthruster

No.	h_t	h_b	h_s	error
39	500	50	200	9.523×10^{-10}
40	500	200	200	8.484×10^{-10}
41	10	50	10	6.996×10^{-10}
42	10	1000	10	1.287×10^{-9}
43	1000	5	10	9.062×10^{-9}
44	10000	50	10	7.910×10^{-9}
45	10000	10000	10000	5.106×10^{-11}
46	5000	5000	5000	2.487×10^{-11}
47	1000	1000	1000	1.362×10^{-10}
48	500	500	500	3.829×10^{-10}
49	10^9	10^9	10^9	0.0161
50	50000	50000	50000	1.093×10^{-8}
51	10000	10000	1	2.260×10^{-11}
52	10	10000	1	2.339×10^{-10}
53	10000	10	1	2.980×10^{-11}
54	1	1	1	3.097×10^{-9}

coefficient matrix. The CPU time of MKR-GMRES for solving each system increases very fast with the increase of the linear systems, whereas the CPU time used in SimGCRO-DR is much less (around $1s$) and remains steady.

5.3.2 Results for Case B

If a smaller reduced model is desired, then multi-point expansion in Case B can achieve this. The sequence of linear systems necessary to be solved during PMOR is in the form in (13), which can be solved by GCRO-DR, G-DRvar1, G-DRvar2 or by SimGCRO-DR. In Table 5, we compare the four recycling algorithms with different choices of k and m. To avoid repetition, we do not list the results of GMRES(∞) and GMRES(m), because they are as inefficient as in Case A. For each group of k and m, GMRES-DR behaves similar to Case A, i.e. it stagnates when solving the second linear system.

As can be seen from Table 5, GCRO-DR is least efficient because of the implementation of Step 3) for each linear system (except for the 1st) and Step 28)-Step 32) at each iteration for each linear system. However for the sequence of linear systems in (13), the computations in Step 3) and Step 28)-Step 32) can be saved by SimGCRO-DR, G-DRvar1 and G-DRvar2. The algorithm SimGCRO-DR is not as efficient as G-DRvar1 and G-DRvar2, because it does not recycle the harmonic Ritz vectors for some linear systems, which slows down the convergence and is consistent with our analysis above. By saving the computation in Step 3) of GCRO-DR, G-DRvar2 uses almost the same MV products as G-DRvar1, and uses less CPU time. It implicates that G-DRvar2 behaves better than G-DRvar1.

The accuracy of the reduced model obtained by applying G-DRvar2 to PMOR is shown in

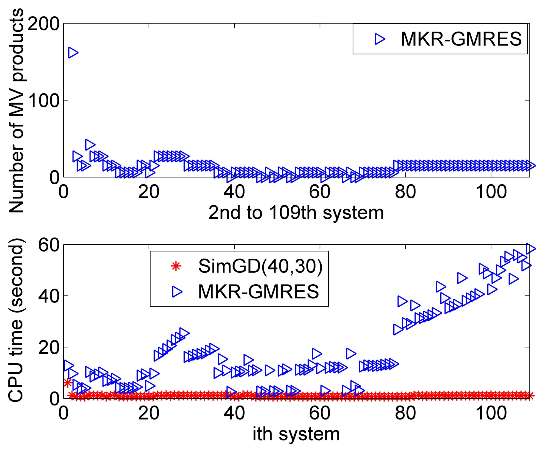

Figure 5: Behavior of MKR-GMRES for Case A

Figure 6. Similar to Figure 4, the error here is the error of the output response of the reduced model at different values of h_s and h_b. Here we fix $h_t = 1e9$ instead of $h_t = 100$ in Figure 4. We can see the error of the reduced model at $h_s = h_b = h_t = 1e9$ is reduced from 0.0161 in Table 4 to $O(10^{-8})$. Similar accuracy can also be obtained by G-DRvar1.

5.4 Simulation of a gyroscope

To apply the PMOR method in Section 2, we also first transform the system into the frequency domain:

$$\begin{aligned} s^2M(d)x + sD(\theta,\alpha,\beta)x + T(d)x &= Bu(s) \\ y &= Cx. \end{aligned} \tag{24}$$

It can be seen that although there are four physical parameters in the original system in (22), there are actually 11 different variables in (24) and they must be considered as individual parameters during PMOR. The parameters are $s_1 = s^2$, $s_2 = s^2d$, $s_3 = s\theta$, $s_4 = s\theta d$, $s_5 = s\alpha$, $s_6 = s\alpha d$, $s_7 = s\beta$, $s_8 = \frac{s}{d}\beta$, $s_9 = s\beta d$, $s_{10} = \frac{1}{d}$, $s_{11} = d$ respectively.

For such a system with many parameters, the dimension of the reduced model will grow quickly if only one set of expansion points are chosen. In order to deal with this problem, we use multi-point expansion. Here we use 4 sets of expansion points $p_j = [s_1^j, s_2^j, \ldots, s_{11}^j]$, $j = 1, 2, \ldots, 4$, and the resulting reduced model is of dimension $q = 289$, which is small versus the original dimension $n = 17,931$.

The physical parameters have their own interesting ranges respectively, which may help us to choose the expansion points. Here $d \in [100\%, 200\%]$ is taken as the percentage of the base value, and $\theta \in [10^{-7}, 10^{-5}]MHz$. It is proven in [26] that the two damping parameters α and β need not be considered during model reduction, and they can be taken as 0.

We also have the range for the frequency $f \in [0.025, 0.25]MHz$. The relation between the Laplace domain variable s and the frequency f is $s = \sigma + 2\pi\sqrt{-1}f$. σ is a real variable, which

Table 5: G-DRvar1, G-DRvar2 vs SimGCRO-DR, GCRO-DR for Case B

Methods	m	k	average MV	total MV	total time (s)
G-DRvar1	30	20	133	11,139	62.44
G-DRvar2	30	20	130	10,899	59.70
SimGCRO-DR	30	20	131	11,010	59.60
GCRO-DR	30	20	129	10,809	108.20
Methods	m	k	average MV	total MV	total time
G-DRvar1	40	30	118	9,939	67.9
G-DRvar2	40	30	115	9,699	64.34
SimGCRO-DR	40	30	116	9,750	65.55
GCRO-DR	40	30	117	9,849	131.64
Methods	m	k	average MV	total MV	total time
G-DRvar1	50	40	106	8,889	74.27
G-DRvar2	50	40	103	8,649	66.59
SimGCRO-DR	50	40	113	9,480	71.92
GCRO-DR	50	40	105	8,829	156.69

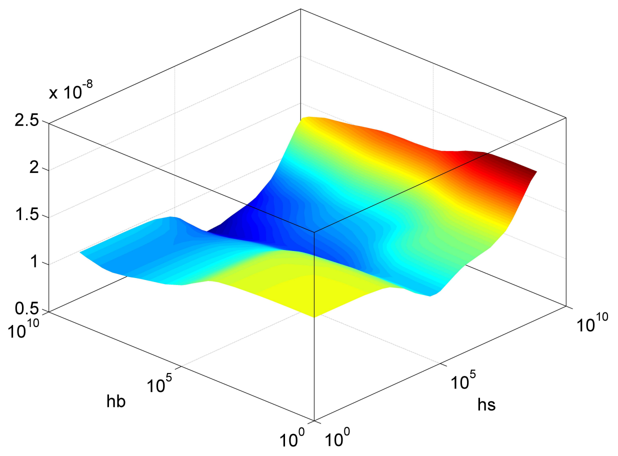

Figure 6: Error of the reduced model produced by G-DRvar2.

is often taken as zero.

We finally have three independent parameters s, d and θ. The system (24) is established based on the unit MHz, therefore the values of f for numerical simulation are in the interval $[0.025, 0.25]$, and the values of θ are in the interval $[10^{-7}, 10^{-5}]$.

We denote $\tilde{p}_j = [s_j, d_j, \theta_j],\ j = 1, 2, \ldots, 4$ the expansion points. Once $\tilde{p}_j$ are given, the corresponding values in $\tilde{p}_j$ are available. Here, $\tilde{p}_j = [s_j, d_j, \theta_j],\ j = 1, 2, \ldots, 4$ are taken as $\tilde{p}_1 = [2\pi\sqrt{-1} \times 0.225, 1, 10^{-6}]$, $\tilde{p}_2 = [2\pi\sqrt{-1} \times 0.225, 2, 10^{-6}]$, $\tilde{p}_3 = [2\pi\sqrt{-1} \times 0.15, 2, 10^{-6}]$, $\tilde{p}_4 = [2\pi\sqrt{-1} \times 0.15, 1.5, 10^{-6}]$.

Next, we check if the expansion points have caused big difference between the coefficient matrices $\tilde{E}(p_j) = T_1 + s_j^2 M_1 + s_j^2 d M_2 + s_j \theta_j D_1 + s_j \theta_j d_j D_2 + \frac{1}{d_j} T_2 + d_j T_3,\ j = 1, \ldots, 4$. If we look at the magnitudes of the individual matrix in $\tilde{E}(p_j)$, we see that the magnitude of the entries in the matrices M_1, M_2 are around O(10^{-11}) and O(10^{-12}) respectively, and the maximal magnitudes of the entries in D_1 and D_2 are both around O(10^{-11}), which are much smaller than the magnitude of s_j. The magnitudes of the entries in T_2 and T_3 are around O(10^{-5}), which are also much smaller than the magnitudes of d_j. Therefore, the change from one expansion point to another expansion point has caused relatively large change in the matrices $s_j^2 M_1 + s_j^2 d M_2 + s_j \theta_j D_1 + s_j \theta_j d_j D_2 + \frac{1}{d_j} T_2 + d_j T_3,\ j = 1, \ldots, 4$. However, the magnitude of T_1 is around O(1), which dominates the whole matrix $E(p_j)$. Therefore, the neighboring $E(p_j),\ j = 1, \ldots, 4$ have just changed slightly.

Notice $\tilde{p}_j$ are complex numbers here, and the resultant projection matrix V_j is also a complex matrix. We should divide V_j into two matrices $[Re\{V_j\}, Im\{V_j\}]$ to form the final projection matrix V as in (19) in Subsection 4.4.

The MV products of the recycling algorithms and the standard solver GMRES are shown in Table 6. The results of GMRES-DR are not listed, because it stagnates after solving the first linear system. We see that with the same usage of memory storage (the same m), the recycling methods uses far less MV products and CPU time than GMRES(m) does. The recycling methods generate the approximate solution from a subspace of small dimension m, much memory storage has been saved compared with GMRES(∞). Actually GMRES(∞) cannot be implemented on the PC (Pentium(R) Dual-Core CPU E5400@ 2.70GHz, 2.69GHz, 2.98GB of RAM) because of limited memory. It makes no sense to implement GMRES(m) for all the linear systems, because it requires too much time. Therefore, we only use GMRES(m) to solve around 50 systems for each $m = 85, 90, 95$ and do not give the total MV products and total time used. From the systems it solves, GMRES(m) behaves almost the same for each system, and we expect that it will produce similar results for the unsolved systems.

Unlike its behavior for Case B of the microthruster, G-DRvar1 is even more efficient than G-Drvar2 for the gyroscope. As is analyzed in Subsection 4.3, G-DRvar2 should save more computation than G-DRvar1 theoretically. However, for $m = 85$ and $m = 90$, although G-DRvar2 has saved around 50 MV products on average, the CPU time spent is even more than G-DRvar1. Especially for the case $m = 95$, G-DRvar1 uses even less MV products than G-DRvar2, which means it converges faster than G-DRvar2, and therefore less MV products have been used. For $m = 110, 120, 130$, G-DRvar1 converges also faster than G-DRvar2.

SimGCRO-DR does not converge for all the linear systems. For the case $m = 85$, it does not converge when it solves for the 130th linear system. Therefore, the algorithm is stopped. For the case $m = 90$, it stops converging from the 87th system. Only the first 44 systems are

Table 6: G-DRvar1, G-DRvar2 vs SimGCRO-DR, GCRO-DR, GMRES for the gyroscope, $k = 30$

Methods	m	average MV	total MV	total time (s)
G-DRvar1	85	668	114,810	22,256
G-DRvar2	85	601	103,485	23,188
SimGCRO-DR	85	795	—	—
GCRO-DR	85	689	118, 440	23,525
GMRES(m)	85	24,758	—	—
Methods	m	average MV	total MV	total time (s)
G-DRvar1	90	633	108, 900	21,106
G-DRvar2	90	589	101, 340	21,931
SimGCRO-DR	90	699	—	—
GCRO-DR	90	638	109,800	23,192
GMRES(m)	90	25,026	—	—
Methods	m	average MV	total MV	total time (s)
G-DRvar1	95	597	102,645	19,812
G-DRvar2	95	631	108, 585	23,754
SimGCRO-DR	95	713	—	—
GCRO-DR	95	601	103,425	23,038
GMRES(m)	95	28,495	—	—
GMRES(∞)	—	—	—	—

solved when $m = 95$, and after, it begins to diverge. Therefore, we cannot count the total MV products and the total CPU time of the method. The average MV products are computed from the results for the converged systems. For the converged systems, SimGCRO-DR uses even more average MV products than GCRO-DR, which means it converges slower than GCRO-DR. This may be due to the fact that for each new different coefficient matrix, SimGCRO-DR does not recycle the harmonic Ritz vectors of the previous matrix, which slows the algorithm.

Table 7: G-DRvar1 vs GCRO-DR for the gyroscope

Methods	m	k	average MV	total MV	total time (s)
G-DRvar1	110	30	633	108,840	23,048
GCRO-DR	110	30	634	109,080	23,418
Methods	m	k	average MV	total MV	total time (s)
G-DRvar1	120	30	668	114,840	24,126
GCRO-DR	120	30	666	114,570	24, 983
Methods	m	k	average MV	total MV	total time (s)
G-DRvar1	120	40	663	114,000	23,004
GCRO-DR	120	40	682	117,360	27,123

In Table 7, when we further compare G-DRvar1 with GCRO-DR with more choices of k and m, we can see without implementing Step 28)-Step 32) for the linear system from the 2nd until the last one in the sequence, G-DRvar1 can still converge with almost the same iteration steps as GCRO-DR because they use almost the same MV products. Since less computation is implemented by G-DRvar1, it uses less CPU time than GCRO-DR.

The output δz of the system in (22), changing with d and the frequency, is plotted in Fig. 7 and that of the reduced model by applying G-DRvar1 to PMOR is given in Fig. 8. Here we show the output in the frequency domain; 4 samples of d and 41 samples of the frequency are taken. The absolute error between the two is plotted in Fig. 9, where the maximum absolute error is 4.6×10^{-8}. The relative error is plotted in Fig. 10, where the maximum relative error is 0.26% which is already much smaller than the usually acceptable error tolerance 1%. To get the output of the original system, one needs more than one week by using an iterative method (e.g. GMRES(m)) to compute the value of δz at each frequency sampling as well as each d-sampling, whereas the output of the reduced model can be obtained in 40 seconds.

5.5 Simulation of a microhotplate gas sensor chip

We have used three groups of expansion points $p_i = [s_i, \kappa_i, c_p^i, \rho_i, h_i]$, $i = 1, \ldots, 3$ to obtain a reduced model with dimension $q = 63$, which are $p_1 = [0, 4, 700, 3100, 11]$, $p_2 = [0, 3, 500, 3100, 10.5]$, $p_3 = [0, 2.5, 439, 3100, 10]$.

In total, 63 linear systems have been solved. In Table 8, we list the results of the methods GCRO-DR, SimGCRO-DR, G-DRvar1 and G-DRvar2 for four groups of k and m. The method GMRES(m) is too slow to give the result for a single linear system after a long time (more

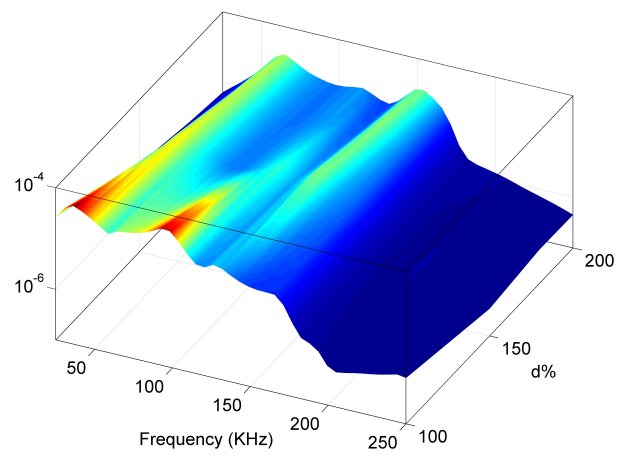

Figure 7: Magnitude of the output δz of the model for the Gyroscope

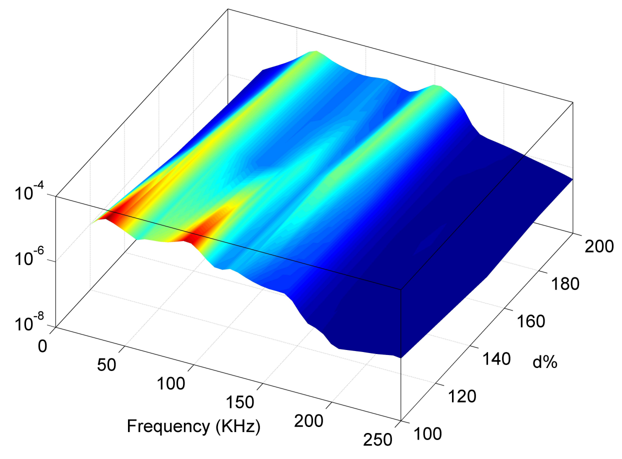

Figure 8: Magnitude of the output δz of the reduced model for the Gyroscope.

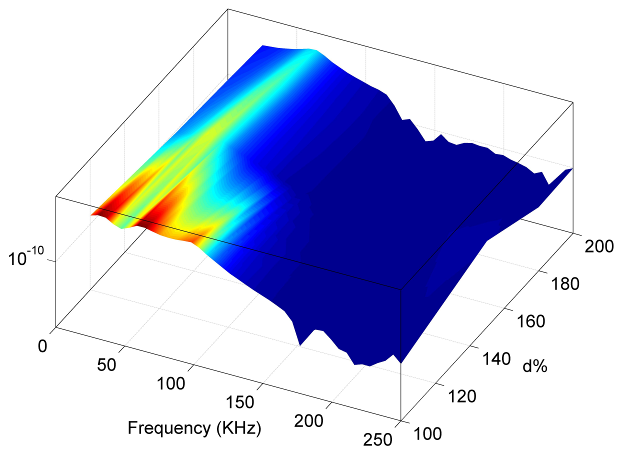

Figure 9: Absolute error plot of the reduced model for for the Gyroscope.

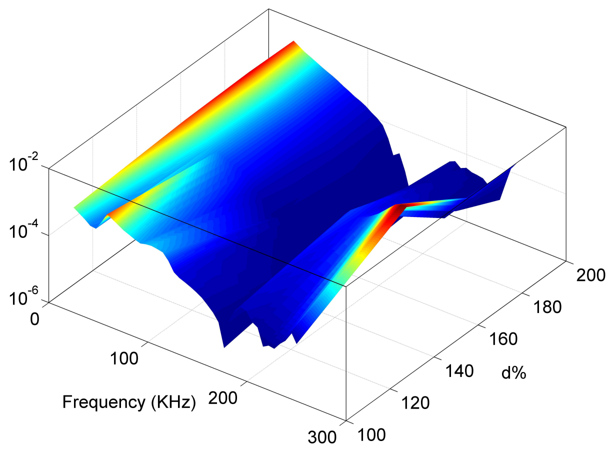

Figure 10: Relative error plot of the reduced model for the Gyroscope.

than 1 hour). GMRES without restarts use too much memory, and the computation cannot be implemented. GMRES-DR still cannot finish solving all the linear systems. For various values of k and m, GMRES-DR starts to stagnate after solving around 10 linear systems.

Table 8: G-DRvar1, G-DRvar2 vs SimGCRO-DR, GCRO-DR for the microhotplate gas sensor chip

Methods	k	m	average MV	total MV	total time (s)
G-DRvar1	50	60	217	13650	2292
G-DRvar2	50	60	83	5220	1777
SimGCRO-DR	50	60	81	—-	—-
GCRO-DR	50	60	217	13650	2602
Methods	k	m	average MV	total MV	total time (s)
G-DRvar1	40	50	187	11760	1814
G-DRvar2	40	50	72	4560	1461
SimGCRO-DR	40	50	71	4500	1444
GCRO-DR	40	50	187	11760	2488
Methods	k	m	average MV	total MV	total time (s)
G-DRvar1	30	40	157	9900	1761
G-DRvar2	30	40	71	4500	1390
SimGCRO-DR	30	40	81	5100	1482
GCRO-DR	30	40	158	9930	1766
Methods	k	m	average MV	total MV	total time (s)
G-DRvar1	20	30	138	8670	1675
G-DRvar2	20	30	80	5070	1462
SimGCRO-DR	20	30	81	5130	1487
GCRO-DR	20	30	141	8910	1689

In general, G-DRvar1, G-DRvar2, SimGCRO-DR are more efficient than the original algorithm GCRO-DR. For this example, G-DRvar2 is much faster than G-DRvar1, which is in agreement with our theoretical analysis. Although SimGCRO-DR is better than G-DRvar1 in many cases, it is unfortunately not convergent for the case $k = 50$, $m = 60$. Among all the methods, G-DRvar2 performs the best.

The relative error of the transfer function of the reduced model relative to the frequency and the parameter κ is plotted in Fig. 11. The transfer function of the original system is shown in Fig. 12. The maximal relative error at the 800 samples for frequency and κ is 1.3×10^{-7}. In Fig. 13, we plot the transfer function of the original system changing with frequency and the heat capacity c_p. The relative error of the transfer function of the reduced model changing with frequency and c_p is plotted in Fig. 14. The maximal error at 1750 samples of frequency and c_p is 2.4×10^{-7}.

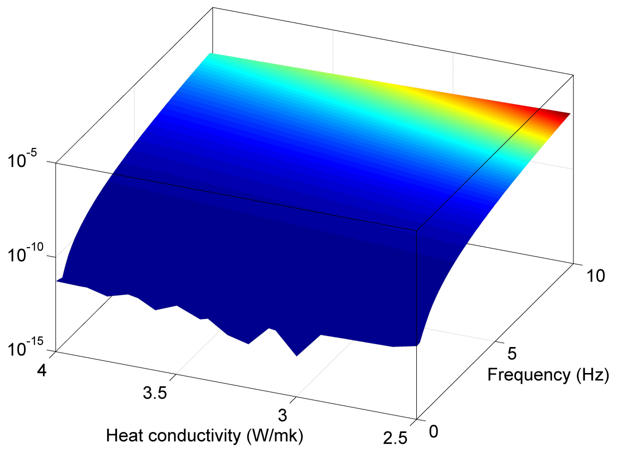

Figure 11: Relative error plot of the reduced model for microhotplate gas sensor chip changing with frequency and κ.

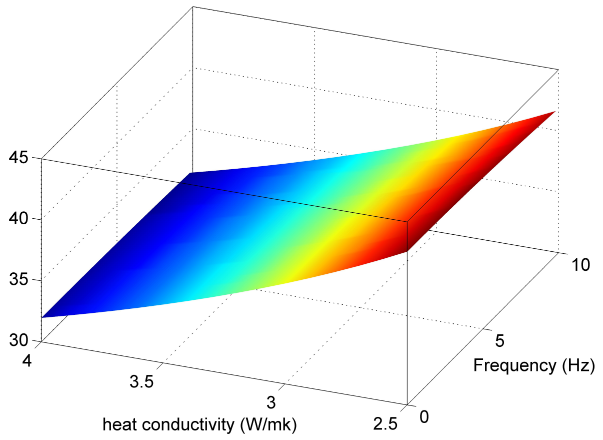

Figure 12: Transfer function of the original system for the microhotplate gas sensor chip changing with frequency and κ.

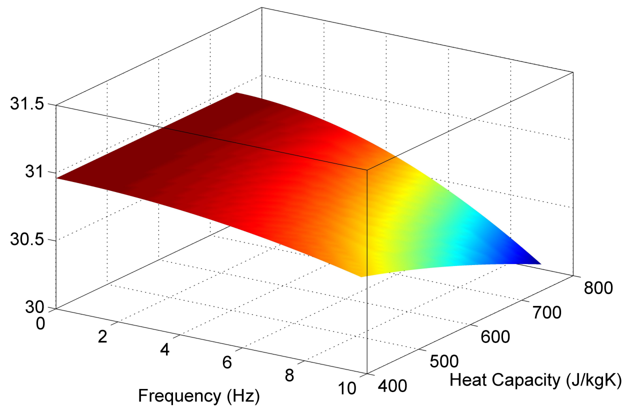

Figure 13: Transfer function of the original system for the microhotplate gas sensor chip changing with frequency and c_p.

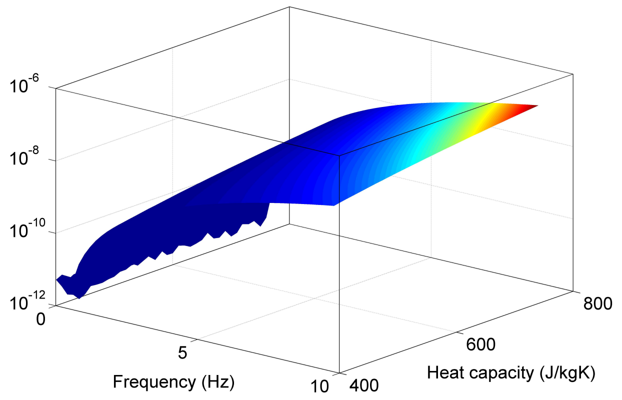

Figure 14: Relative error plot of the reduced model for the microhotplate gas sensor chip changing with frequency and c_p.

6 Conclusions

In this paper, algorithms based on subspace recycling have been successfully applied to the PMOR process, with much efficiency achieved.

During the PMOR process, many linear systems must be solved. When linear systems are of very large scale, direct solvers like LU decomposition are difficult to employ. The most appropriate choices are the iterative methods. The algorithms based on subspace recycling are designed to accelerate the standard iterative methods, and further they speeds up the process of PMOR.

The two variants G-DRvar1 and G-DRvar2 of the recycling algorithm GCRO-DR are proposed in this paper, which require less computation and hence are more efficient than the original version GCRO-DR, especially for the sequences of linear systems arising from PMOR.

It can be seen from the above simulation results that the efficiency of the recycling algorithms depends on the choice of m and k, i.e. different values of m, k leads to different numbers of MV products. How to choose the proper m and k remains an open problem.

Finally, the application of the recycling algorithms is not limited to PMOR. It can be easily extended to many other simulation problems, where computation of a sequence of linear systems is unavoidable for very large scale systems. Furthermore, the right-hand sides of the linear systems do not need to be simultaneously available, which makes the recycling algorithms flexible for various problems.

Appendix A. GCRO-DR in [1]

Algorithm 5 GCRO-DR(m,k) *Solving the ith system in the sequence of linear systems* $A_i x = b_i,\ i = 1,2,\ldots,l$:

```
1. Choose m, the maximum size of the subspace, and k the desired number of the harmonic
   Ritz vectors of A. tol is the convergence tolerance. $x_0$ is the initial guess, and $r_0 = b_i - A_i x_0$ is the initial error. Set $j = 1$.
2. IF $\tilde{Y}$ is defined (from solving a previous system) then
3.    Let $[Q,R]$ be the reduced QR-factorization of $A_i\tilde{Y}$.
4.    $C = Q$
5.    $U = \tilde{Y}R^{-1}$
6.    $x_1 = x_0 + UC^H r_0$
7.    $r_1 = r_0 - CC^H r_0$
8. ELSE
9.    $v_1 = r_0/||r_0||_2$
10.   $c = ||r_0||e_1$
```

```
11.     Perform m steps of GMRES, generating a $(m+1)\times m$ matrix $\tilde{H}_m$ , a $n\times(m+1)$ matrix $V_{m+1}$.
12. $x_1 = x_0 + V_m y$
13. $r_1 = V_{m+1}(c - \tilde{H}_m y)$
14.     Compute the k eigenvectors $\xi_j$ of: $(H_m + h^2_{m+1,m} H_m^{-H} e_m e_m^H)\xi_j = \lambda_j \xi_j$ associated with the k smallest magnitude eigenvalues and store in P.
15.     $\tilde{Y} = V_m P$
16.     Let [Q, R] be the reduced QR-factorization of $\tilde{H}_m P$.
17.     $C = V_{m+1} Q$; $U = \tilde{Y} R^{-1}$.
18. END IF
19. WHILE $\|r_j\|_2 > tol$ do
20. $j = j + 1$
21. run $m-k$ steps of Arnoldi algorithm with the linear operator $(I - CC^H)A$, letting $v_1 = r_{j-1}/\|r_{j-1}\|_2$ and generating $\tilde{H}_{m-k}$, $V_{m-k+1}$, and $B_{m-k}$.
22. Let $D_k$ be a diagonal scaling matrix such that $\tilde{U} = UD_k$, where that columns of $\tilde{U}$ have unit norm.
23. $\hat{V}_m = [\tilde{U}\ V_{m-k}]$; $\hat{W}_{m+1} = [C\ V_{m-k+1}]$;
24. Let $G_m = \begin{bmatrix} D_k & B_{m-k} \\ 0 & \tilde{H}_{m-k} \end{bmatrix}$.
25. Solve $\min\|\hat{W}^H_{m+1} r_{j-1} - G_m y\|_2$ for y.
26. $x_j = x_{j-1} + \hat{V}_m y$;
27. $r_j = r_{j-1} - \hat{W}_{m+1} G_m y$.
28. Compute the k eigenvectors $\xi_t$ of $G_m^H G_m \xi_t = \lambda_t G_m^H \hat{W}^H_{m+1} \hat{V}_m \xi_t$ associated with the k smallest magnitude eigenvalues $\lambda_t$ and store in P.
29. $\tilde{Y} = \hat{V}_m P$.
30. Let [Q,R] be the reduced QR-factorization of $G_m P$;
31. $C = \hat{W}_{m+1} Q$;
32. $U = \tilde{Y} R^{-1}$.
33. END WHILE
34. Let $\tilde{Y} = U$ (for the next system).
```

Appendix B. SimGCRO-DR in [24]

Algorithm 6 SimGCRO-DR(m,k) *Solving the ith system in the sequence of linear systems $Ax = b_i,\ i = 1,2,\ldots,l$:*

```
1. Choose m, the maximum size of the subspace, and k the desired number of the harmonic Ritz vectors of A. tol is the convergence tolerance. $x_0$ is the initial guess, and $r_0 = b_i - Ax_0$ is the initial error. Set $j = 1$.
2. IF C and U are defined (from solving a previous system) then
3.     $x_1 = x_0 + UC^H r_0$;
4.     $r_1 = r_0 - CC^H r_0$.
5. ELSE
6.     run m steps of GMRES, generating an $(m+1) \times m$ matrix $\tilde{H}_m$, an $n \times (m+1)$ matrix $V_{m+1}$ and $x_1$, $r_1$.
7. Compute k smallest harmonic Ritz vectors of A, i.e. k eigenvectors $\xi_j$ of: $(H_m + h^2_{m+1,m} H_m^{-H} e_m e_m^H)\xi_j = \lambda_j \xi_j$ associated with the k smallest magnitude eigenvalues and store in P. Here $H_m$ is the $m \times m$ upper block in $\tilde{H}_m$, $h_{m+1,m}$ is the $(m+1,m)$ entry in $\tilde{H}_m$ and $e_m \in \mathbb{R}^m$, $e = [0,\ldots,01]^T$.
8. $\tilde{Y} = V_m P$; $V_m$ contains the first m columns in $V_{m+1}$.
9. Let [Q, R] be the reduced QR-factorization of $\tilde{H}_m P$;
10. $C = V_{m+1} Q$; $U = \tilde{Y} R^{-1}$.
11. END IF
12. WHILE $\|r_j\|_2 > tol$ do
13. $j = j + 1$;
14. run $m-k$ steps of Arnoldi algorithm with the linear operator $(I - CC^H)A$, letting $v_1 = r_{j-1}/\|r_{j-1}\|_2$ and generating a $(m-k+1) \times (m-k)$ matrix $\tilde{H}_{m-k}$, a $n \times (m-k+1)$ matrix $V_{m-k+1}$, and $B_{m-k} = C^H A V_{m-k}$, likewise, $V_{m-k}$ contains the first $m-k$ columns of $V_{m-k+1}$.
15. Step 16-Step 20 compute $x_j$ whose residual $r_j$ is minimized over the subspace spanned by $[\tilde{U}\ V_{m-k}]$. Here $\tilde{U} = UD_k$, $D_k$ is a diagonal scaling matrix such that each column in $\tilde{U}_k$ has unit 2-norm.
16. Let $G_m = \begin{bmatrix} D_k & B_{m-k} \\ 0 & \tilde{H}_{m-k} \end{bmatrix}$.
```

```
17. $\hat{V}_m = [\tilde{U}\ V_{m-k}]$; $\hat{W}_{m+1} = [C\ V_{m-k+1}]$;
18. Solve $min\|\hat{W}_{m+1}^H r_{j-1} - G_m y\|_2$ for y.
19. $x_j = x_{j-1} + \hat{V}_m y$;
20. $r_j = r_{j-1} - \hat{W}_{m+1} G_m y$.
21. IF solving the first system i.e. $i = 1$, then modify the k harmonic Ritz vectors of A, i.e.,
    compute k eigenvectors $\xi_t$ of $G_m^H G_m \xi_t = \lambda_t G_m^H \hat{W}_{m+1}^H \hat{V}_m \xi_t$ associated with the k smallest
    magnitude eigenvalues $\lambda_t$ and store in P.
22. $\tilde{Y} = \hat{V}_m P$.
23. Let [Q,R] be the reduced QR-factorization of $G_m P$;
24. $C = \hat{W}_{m+1} Q$;
25. $U = \tilde{Y} R^{-1}$.
26. END IF
27. END WHILE.
```

Acknowledgment

This work was supported by DFG grant BE 2174/7-1, KO 1883/8-1 Automatic, Parameter-Preserving Model Reduction for Applications in Microsystems Technology. We kindly thank Dr. Jan Lienemann for the discussions and suggestions on the two tested examples. Dr. Tamara Bechtold is appreciated for providing the data of the third example.

References

[1] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti, "Recycling Krylov subspaces for sequences of linear systems," *SIAM J. Sci. Comput.*, 28 (5): 1651-1674, 2006.

[2] Z. Ye, Z. Zhu, and J. R. Phillips, "Generalized Krylov Recycling Methods for Solution of Multiple Related Linear Equation Systems in Electromagnetic Analysis," *In Proc. Design Automation Conference (DAC08)*, 682-687, 2008.

[3] Z. Ye, Z. Zhu, and J. R. Phillips, "Incremental Large-Scale Electrostatic Analysis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 28(11): 1641-1653, 2009.

[4] E. B. Rudnyi, J. G. Korvink. "Review: Automatic Model Reduction for Transient Simulation of MEMS-based Devices," *Sensors Update* 11: 3-33, 2002.

[5] J. G. Korvink, E. B. Rudnyi, A. Greiner, and Z. Liu. “MEMS and NEMS Simulation. In MEMS: A Practical Guide to Design, Analysis, and Applications,” *ed. Jan G. Korvink, Oliver Paul, William Andrew Publishing, Norwich, NY, 2005* 93-186.

[6] T. Bechtold, E. B. Rudnyi and J. G. Korvink, “Error indicators for fully automatic extraction of heat-transfer macromodels for MEMS,” *Journal of Micromechanics and Mountaineering* 15(3): 430-440, 2005.

[7] Y. Saad and M.H. Schultz, ”GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems,” *SIAM J. Sci. Stat. Comput.*, 7:856-869, 1986.

[8] M. R. Hestenes and E. Stiefel, “Methods of Conjugate Gradients for Solving Linear Systems,” *Journal of Research of the National Bureau of Standards* 49(6):409-436, 1952.

[9] L. Feng and P. Benner, “A Robust Algorithm for Parametric Model Order Reduction,” *In Proc. Applied Mathematics and Mechanics (ICIAM 2007)*, 7(1): 10215.01–02, 2007.

[10] L. Daniel, O. C. Siong, L. S. Chay, K. H. Lee, and J. White. “A multiparameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 22(5): 678–693, 2004.

[11] M. Kilmer, E. Miller, and C. Rappaport, “QMR-Based projection techniques for the solution of non-Hermitian systems with multiple right-hand sides,” *SIAM J. Sci. Comput.*, 23 (3): 761-780, 2001.

[12] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc’h, “A deflated version of the conjugate gradient algorithm,” *SIAM J. Sci. Comput.*, 21 (5): 1909-1926, 2000.

[13] P. F. Fischer, “Projection techniques for iterative solution of $A\underline{x} = \underline{b}$ with successive right-hand sides,” *Comput. Methods Appl. Mech. Engi.*, 163: 193–204, 1998.

[14] E. de Sturler, “Nested Krylov methods based on GCR,” *J. Comput. Appl. Math.*, 67: 15-41, 1996.

[15] R. B. Morgan, “GMRES with deflated restarting,” *SIAM J. Sci. Comput.*, 24: 20-37, 2002.

[16] R. B. Morgan and M. Zeng, “Harmonic projection methods for large non-symmetric eigenvalue problems,” *Numer. Linear Algebra Appl.*, 5: 33-55, 1998.

[17] M. Clemens, M. Helias, T. Steinmets, and G. Wimmer, “Multiple right-hand side techniques for the numerical simulation of quasistatic electric and magnetic fields,” *J. Comput. Appl. Math.*, 215: 328-338, 2008.

[18] L. Feng, E. B. Rudnyi, and J. G. Korvink, “Preserving the film coefficient as a parameter in the compact thermal model for fast electro-thermal simulation,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 24(12):1838–1847, 2005.

[19] X. Li, P. Li, and L. T. Pileggi. “Parameterized interconnect order reduction with explicit-and-implicit multi-parameter moment matching for inter/intra-die variations,” *In Proc. International Conference on Computer-Aided Design*, 806–812, 2005.

[20] Y. Li, Z. Bai, Y. Su, and X. Zeng. “Model order reduction of parameterized interconnect networks via a two-directional Arnoldi process,” *In Proc. International Conference on Computer-Aided Design*, 868–873, 2007.

[21] L. Feng and P. Benner, “A robust algorithm for parametric model order reduction based on implicit moment matching,” *submitted.*

[22] C. Moosmann, “ParaMOR—- Model Order Reduction for parameterized MEMS applications,” *PhD thesis, Department of Microsystems Engineering, University of Freiburg,* 2007.

[23] C. J. M. Lasance, “Two benchmarks to facilitate the study of compact thermal modeling phenomena,” *IEEE Transactions on Components and Packaging Technologies*, 24, 559-565 (2001).

[24] L. Feng, P. Benner and J. G. Korvink, “Parametric Model Order Reduction Accelerated by Subspace Recycling,” *In Proc. Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference Shanghai, P.R. China*, 4328-4333, 2009.

[25] C. Rossi, B. Larangot, D Lagrange and A. Chaalane, “Final characterizations of MEMS-based pyrotechnical microthrusters,” Sensors and Actuators A, 121:508-514, 2005.

[26] B. Salimbahrami, R. Eid, B. Lohmann, “Model Reduction by Second Order Krylov Subspaces: Extensions, Stability and Proportional Damping,” IEEE International Symposium on Intelligent Control, 29973002, 2006.

[27] K. Ahuja, E. D. Sturler, E. R. Chang, and S. Gugercin, “Recycling BiCG for Model Reduction,” Preprint of Cornel University Library, 2010.

[28] H. A. van der Vorst, “Iterative Krylov Methods for Large Linear Systems,” Cambridge University Press, Cambridge, England, UK, 2003.

[29] Tamara Bechtold, “Model Order Reduction of Electro-Thermal MEMS”, PhD thesis, University of Freiburg, 2005.

[30] T. Bechtold, D. Hohlfeld3, E. B. Rudnyi, and M. Guenther, “ Efficient extraction of thin-film thermal parameters from numerical models via parametric model order reduction”, J. Micromech. Microeng. 20 (2010) 045030 (13pp).