Sensitivity Analysis of Differential-Algebraic Systems using the GMRES method-Application to Power Systems

Dimitrios Chaniotis, M. A. Pai, Ian Hiskens
Department of Electrical & Computer Engineering
University of Illinois at Urbana-Champaign
1406 W. Green St. Urbana, IL 61801

Abstract

This paper outlines a technique which solves the sensitivity as well as the system equations in an efficient manner. Instead of solving the two sets of equations simultaneously, we propose integrating the system equations and then solving the sensitivity equations in a "staggered fashion", using the preconditioned Generalized Minimum Residual (GMRES) method. This reduces the computation time. The method is directly applicable to hybrid systems, of which power systems form an important application area.

1. Introduction

Trajectory sensitivities provide valuable insights into the influence of parameters on the dynamic behavior of systems. Properties, which are not obvious from the actual system response, are often evident in the sensitivities. Applications are numerous and include parameter estimation, optimal control and stability analysis of periodic behavior [1, 2, 3]. However, the additional cost of obtaining the sensitivities may be prohibitive. Currently, methods such as the *Simultaneous Corrector* method [4], the *Staggered Direct* method [5] and the *Staggered Corrector* method [6] are used for this task. In this paper, we introduce a variation called the *Staggered Hybrid* method, which involves using an iterative procedure such as the *Generalized Minimum Residual* (GMRES) method [7] along with the traditional LU decomposition to solve for the sensitivities. We compare these methods for a small, 50-machine, 145-bus power system.

2. Problem Formulation

The multi-machine model of an *m*-machine, *n*-bus power system is of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{y}) \quad , \quad \mathbf{x}(0) = \mathbf{x_0} \tag{1}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x},\mathbf{y}) \quad , \quad \mathbf{y}(0) = \mathbf{y_0} \tag{2}$$

where **x** is the vector of dynamic state variables and **y** is the vector of algebraic network variables. Vector functions **f** and **g** represent the machine and the network equations respectively. The dimension of **x** depends on the level of modeling of the generators. We use the two-axis machine model with the IEEE type-I exciter, resulting in 7 differential equations for each generator, and therefore dim(**x**)=dim(**f**)=7*m*. The dimension of **y** and **g** is 2*n* [8].

In order to keep a compact notation, we study the trajectory sensitivities with respect to initial conditions rather than with respect to the system parameters. Any system parameter of interest can be modeled as a constant dynamic state variable whose initial condition is equal to its nominal value. Therefore, **x** becomes an augmented vector to include the system parameters [9].

Differentiating (1) and (2) with respect to the initial conditions $\mathbf{x_0}$ gives the sensitivity equations

$$\dot{\mathbf{x}}_{\mathbf{x_0}} = \mathbf{f_x}(t)\mathbf{x}_{\mathbf{x_0}} + \mathbf{f_y}(t)\mathbf{y}_{\mathbf{x_0}} \tag{3}$$

$$\mathbf{0} = \mathbf{g_x}(t)\mathbf{x}_{\mathbf{x_0}} + \mathbf{g_y}(t)\mathbf{y}_{\mathbf{x_0}} \,. \tag{4}$$

Equations (3) and (4) are a linear, time varying set of differential-algebraic equations. Each trajectory sensitivity computation involves the integration of 7*m* differential equations constrained by 2*n* algebraic equations. This task is comparable, in terms of cost, to the problem of solving (1) and (2).

A number of methods have been applied to this problem. Here, we examine and compare the *Simultaneous Corrector method* [4], the *Staggered Direct method* [5], the *Staggered Corrector method* [6] and the *Staggered Hybrid method* proposed in this paper. For all the methods, the trapezoidal rule was used to algebraize the differential equations, which are then solved simultaneously with the algebraic equations. This scheme is known to be very reliable and fast for the problems arising in power systems. Nevertheless, most of the results also apply for any general backward differentiation formula.

3. The Simultaneous Corrector method (SICM) [4]

Applying the trapezoidal rule to (1)-(4) we obtain the following set of algebraic equations

$$\begin{aligned}\mathbf{F_1}(\mathbf{x}_{t+1},\mathbf{y}_{t+1},\mathbf{x}_t,\mathbf{y}_t) = \mathbf{x}_{t+1} - \mathbf{x}_t - \\ \frac{h}{2}\big(\mathbf{f}(\mathbf{x}_t,\mathbf{y}_t) + \mathbf{f}(\mathbf{x}_{t+1},\mathbf{y}_{t+1})\big) = \mathbf{0}\end{aligned} \tag{5}$$

$$\mathbf{F_2}(\mathbf{x}_{t+1},\mathbf{y}_{t+1}) = \mathbf{g}(\mathbf{x}_{t+1},\mathbf{y}_{t+1}) = \mathbf{0} \tag{6}$$

$$\begin{aligned}\mathbf{F_3}(\mathbf{x}_{t+1},\mathbf{y}_{t+1},\mathbf{x}_t,\mathbf{y}_t,\mathbf{x}_{\mathbf{x_0}t+1},\mathbf{y}_{\mathbf{x_0}t+1},\mathbf{x}_{\mathbf{x_0}t},\mathbf{y}_{\mathbf{x_0}t}) = \mathbf{x}_{\mathbf{x_0}t+1} - \mathbf{x}_{\mathbf{x_0}t} - \\ \frac{h}{2}\big(\mathbf{f_x}(\mathbf{x}_t,\mathbf{y}_t)\mathbf{x}_{\mathbf{x_0}t} + \mathbf{f_y}(\mathbf{x}_t,\mathbf{y}_t)\mathbf{y}_{\mathbf{x_0}t} + \\ \mathbf{f_x}(\mathbf{x}_{t+1},\mathbf{y}_{t+1})\mathbf{x}_{\mathbf{x_0}t+1} + \mathbf{f_y}(\mathbf{x}_{t+1},\mathbf{y}_{t+1})\mathbf{y}_{\mathbf{x_0}t+1}\big) = \mathbf{0}\end{aligned} \tag{7}$$

$$\begin{aligned}\mathbf{F_4}(\mathbf{x}_{t+1},\mathbf{y}_{t+1},\mathbf{x}_{\mathbf{x_0}t+1},\mathbf{y}_{\mathbf{x_0}t+1}) = \\ \mathbf{g_x}(\mathbf{x}_{t+1},\mathbf{y}_{t+1})\mathbf{x}_{\mathbf{x_0}t+1} + \mathbf{g_y}(\mathbf{x}_{t+1},\mathbf{y}_{t+1})\mathbf{y}_{\mathbf{x_0}t+1} = \mathbf{0}\end{aligned} \tag{8}$$

where *t* is the time instant and *h* is the integration step. In order to make the notation easier, assume that we compute the trajectory sensitivities with respect to only one initial condition x_0. The extension to sensitivity computations with respect to more parameters is straightforward. Equations (5) through (8) are solved using the Newton-Raphson (NR) method

0-7803-6685-9/01/$10.00©2001 IEEE

$$\begin{bmatrix}\mathbf{J} & \mathbf{0}\\ \mathbf{J_1} & \mathbf{J}\end{bmatrix}\begin{bmatrix}\Delta\mathbf{x}_{t+1}\\ \Delta\mathbf{y}_{t+1}\\ \Delta\mathbf{x}_{x_0 t+1}\\ \Delta\mathbf{y}_{x_0 t+1}\end{bmatrix}^{(k)} = -\begin{bmatrix}\mathbf{F_1}\\ \mathbf{F_2}\\ \mathbf{F_3}\\ \mathbf{F_4}\end{bmatrix}^{(k)} \tag{9}$$

and

$$\begin{aligned}\mathbf{x}_{t+1}^{(k+1)} &= \mathbf{x}_{t+1}^{(k)} + \Delta\mathbf{x}_{t+1}^{(k)}\\ \mathbf{y}_{t+1}^{(k+1)} &= \mathbf{y}_{t+1}^{(k)} + \Delta\mathbf{y}_{t+1}^{(k)}\\ \mathbf{x}_{x_0 t+1}^{(k+1)} &= \mathbf{x}_{x_0 t+1}^{(k)} + \Delta\mathbf{x}_{x_0 t+1}^{(k)}\\ \mathbf{y}_{x_0 t+1}^{(k+1)} &= \mathbf{y}_{x_0 t+1}^{(k)} + \Delta\mathbf{y}_{x_0 t+1}^{(k)}\end{aligned} \quad , \quad \mathbf{J} = \begin{bmatrix}\mathbf{I} - \frac{h}{2}\mathbf{f_x} & -\frac{h}{2}\mathbf{f_y}\\ \mathbf{g_x} & \mathbf{g_y}\end{bmatrix}$$

$$\mathbf{J_1} = \begin{bmatrix}\mathbf{f_{xx}}\mathbf{x}_{x_0 t+1} + \mathbf{f_{xy}}\mathbf{y}_{x_0 t+1} & \mathbf{f_{xy}}\mathbf{x}_{x_0 t+1} + \mathbf{f_{yy}}\mathbf{y}_{x_0 t+1}\\ \mathbf{g_{xx}}\mathbf{x}_{x_0 t+1} + \mathbf{g_{xy}}\mathbf{y}_{x_0 t+1} & \mathbf{g_{xy}}\mathbf{x}_{x_0 t+1} + \mathbf{g_{yy}}\mathbf{y}_{x_0 t+1}\end{bmatrix}.$$

Index k refers to the NR iteration count. It is shown in [4] that the Jacobian in (9) can be approximated by its block diagonal part without jeopardizing convergence. Therefore, the factorization of the Jacobian is significantly simplified. Additionally, the Jacobian may be kept constant for a number of time steps to avoid further factorizations. This technique is widely known as the *Very Dishonest Newton's method* and is frequently used in dynamic simulation (see for example [10]).

4. The Staggered Direct method (SDM) [5]

Rearrangement of (7) and (8) reveals that

$$\begin{bmatrix}\mathbf{I} - \frac{h}{2}\mathbf{f_x} & -\frac{h}{2}\mathbf{f_y}\\ \mathbf{g_x} & \mathbf{g_y}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{x_0 t+1}\\ \mathbf{y}_{x_0 t+1}\end{bmatrix} = \begin{bmatrix}\frac{h}{2}\left(\mathbf{f_x}\mathbf{x}_{x_0 t} + \mathbf{f_y}\mathbf{y}_{x_0 t}\right) + \mathbf{x}_{x_0 t}\\ \mathbf{0}\end{bmatrix}. \tag{10}$$

Once the states are computed for a new time step, the sensitivity equations can be computed from (10). The efficiency of this method comes from the fact that in (10) the coefficient matrix is already factored, as it happens to be the Jacobian at the current step of the integration of the system equations. Therefore, only a forward/backward substitution is required for the solution of the sensitivity equations. However, this means that the Jacobian must be computed at every step of the integration. Developments in sparse-matrix computations, especially in sparse-matrix factorization and forward-backward substitution, indicate that this trade-off is crucial for the overall performance of the algorithm [11].

5. The Staggered Corrector method (SCM) [6]

To address the problem of having to factor the Jacobian at every time step, Feehery et. al. [6] proposed the use of a quasi-NR method to compute the sensitivities. Specifically, the Jacobian for the trajectory computation is not updated at every time step, but rather the (approximate) Jacobian $\hat{\mathbf{J}}$ computed in a previous time step is used

$$\hat{\mathbf{J}}\begin{bmatrix}\Delta\mathbf{x}_{t+1}\\ \Delta\mathbf{y}_{t+1}\end{bmatrix}^{(k)} = -\begin{bmatrix}\mathbf{F_1}\\ \mathbf{F_2}\end{bmatrix}^{(k)}. \tag{11}$$

Since the exact Jacobian LU-factors for the current step are not available, (10) is solved using the iterative scheme

$$\begin{aligned}&\hat{\mathbf{J}}\left(\begin{bmatrix}\mathbf{x}_{x_0 t+1}^{(k+1)}\\ \mathbf{y}_{x_0 t+1}^{(k+1)}\end{bmatrix} - \begin{bmatrix}\mathbf{x}_{x_0 t+1}^{(k)}\\ \mathbf{y}_{x_0 t+1}^{(k)}\end{bmatrix}\right) =\\ &-\begin{bmatrix}\mathbf{I} - \frac{h}{2}\mathbf{f_x} & -\frac{h}{2}\mathbf{f_y}\\ \mathbf{g_x} & \mathbf{g_y}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{x_0 t+1}^{(k)}\\ \mathbf{y}_{x_0 t+1}^{(k)}\end{bmatrix} + \begin{bmatrix}\frac{h}{2}\left(\mathbf{f_x}\mathbf{x}_{x_0 t} + \mathbf{f_y}\mathbf{y}_{x_0 t}\right) + \mathbf{x}_{x_0 t}\\ 0\end{bmatrix}\end{aligned}. \tag{12}$$

This quasi-NR scheme usually converges very quickly provided the Jacobian is well conditioned.

6. The Staggered Hybrid method (SHM)

We now propose another method to overcome the drawbacks of the SDM. In order to solve (10) when the current Jacobian is not factored, we note that the LU-factors of the approximate Jacobian $\hat{\mathbf{J}}$ are available. Using these factors as a *preconditioner* for an iterative method (i.e. GMRES) we can expect the solution of (10) in a very small number of iterations, since these factors are a very good approximation to the current system Jacobian. Therefore, instead of (10) one effectively solves

$$\hat{\mathbf{J}}^{-1}\begin{bmatrix}\mathbf{I} - \frac{h}{2}\mathbf{f_x} & -\frac{h}{2}\mathbf{f_y}\\ \mathbf{g_x} & \mathbf{g_y}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{x_0 t+1}\\ \mathbf{y}_{x_0 t+1}\end{bmatrix} = \hat{\mathbf{J}}^{-1}\begin{bmatrix}\frac{h}{2}\left(\mathbf{f_x}\mathbf{x}_{x_0 t} + \mathbf{f_y}\mathbf{y}_{x_0 t}\right) + \mathbf{x}_{x_0 t}\\ \mathbf{0}\end{bmatrix}$$

though $\hat{\mathbf{J}}^{-1}$ is never explicitly formed.

The closer $\hat{\mathbf{J}}$ is to the current system Jacobian $\mathbf{J}$ the better GMRES performs, because the product $\hat{\mathbf{J}}^{-1}\mathbf{J}$ resembles the identity matrix. Within GMRES, the current system Jacobian appears only in sparse matrix-vector multiplications, while the action of preconditioning requires the inversion of $\hat{\mathbf{J}}$, which is equivalent to a forward-backward substitution. Additionally, GMRES can be initiated with a very good initial guess, which is the value of the sensitivities from the previous time step. This method gains from the fact that the system Jacobian *is not factored at every step* of the integration but loses from the fact that *an iterative method (with an excellent preconditioner) replaces the very fast forward/backward substitution* in the solution for the sensitivities.

7. Comparison of the Methods

For typical, large power systems the cost of factoring the Jacobian usually dominates the computations. We also consider the cost of forward/backward substitution, the cost of updating the Jacobian and the cost of computing the mismatches for the sensitivities. The computational complexity analysis for the SICM, the SDM and the SCM is undertaken in [12]. The authors in [12] considered the option of avoiding matrix-vector products (and subsequently Jacobian updates) via directional derivative computations. However, for power systems this option does not yield significant improvement, as the equations are quite complex. Therefore this option is not examined here. The analysis of [12] is repeated here to compare the four methods, using the notation shown in Table 1.

Table 1

C_{eval}	Cost of evaluating the Jacobian
C_{fac}	Cost of factoring the Jacobian
C_{fbs}	Cost of Forward/Backward substitution
C_{mvp}	Cost of Sparse matrix-vector product
C_{srhs}	Cost of evaluating equations (7) and (8)
C_{srhs2}	Cost of computing the right-hand-side of Eq. (10)
C_{trhs}	Cost of evaluating equations (5) and (6)

Assume that the NR method for the SICM requires N_0 iterations to converge and that the Jacobian is evaluated and factored every N_1 time steps. Note that in this case C_{srhs} can be approximated by $C_{eval}+4C_{mvp}$. Therefore the average cost per time step is

$$C_{SICM} = \frac{C_{eval} + C_{fac}}{N_1} + N_0\left(C_{trhs} + 2C_{fbs} + C_{eval} + 4C_{mvp}\right).$$

Note that the quasi-NR scheme is used for the trajectory computation. (Only one evaluation/factorization of the Jacobian is performed per time step). This is also true for the other three methods.

Assume that the NR method for the SDM requires N_2 iterations to converge. Then the average cost per time step is

$$C_{SDM} = C_{eval} + C_{fac} + N_2 C_{trhs} + C_{fbs} + C_{srhs2}\ .$$

Assume that the NR method for the SCM requires N_3 iterations to converge and that the Jacobian is evaluated and factored every N_1 time steps. Further, assume that the quasi-NR scheme for the sensitivity computation requires N_4 iterations to converge. Then

$$C_{SCM} = \frac{C_{fac}}{N_1} + C_{eval} + N_3\left(C_{trhs} + C_{fbs}\right) + N_4\left(C_{fbs} + C_{mvp}\right) + C_{srhs2}\ .$$

Assume that the NR method for the SHM requires N_3 iterations to converge and that the Jacobian is evaluated and factored every N_1 steps. Further, assume that GMRES requires N_5 iterations to converge. The cost of GMRES is dominated by the preconditioning steps and by a matrix-vector product. The cost of orthonormalization using the Gram-Schmidt method can be overlooked for small N_5. Then, the cost of GMRES is approximated by $(N_5+1)(C_{fbs}+C_{mvp})$, and

$$C_{SHM} = \frac{C_{fac}}{N_1} + C_{eval} + N_3\left(C_{trhs} + C_{fbs}\right) + \left(N_5 + 1\right)\left(C_{fbs} + C_{mvp}\right) + C_{srhs2}\ .$$

Evidently, N_4 and N_5 determine the relative efficiency of SCM and SHM.

The previous analysis involves the trajectory sensitivity computation with respect to only one initial condition. When more sensitivities are required SHM may become more efficient with the use of block-Krylov GMRES [13].

8. Numerical Results

The methods described earlier were applied to the sensitivity analysis of a 50-machine, 145-bus system. The system was simulated for 3 seconds, for a self-clearing fault on bus 58 at 0.1 seconds, cleared at 0.3 seconds. The initial integration step size was 0.001 seconds. This was increased to 0.01 seconds after a lapse of 0.1 seconds following the fault clearance. All trajectory sensitivities with respect to the clearing time were computed. The computations were performed in Matlab, on a Pentium III processor using Linux 6.2. Only floating point operations (flops) are reported here. It should be noted that the cost of the re-ordering stage of the LU factorization does not show in flops even though it consumes almost 1/3 of the total time required by the LU method. However, the re-ordering scheme is computed infrequently, because the structure of the Jacobian only changes at events such as reactive power limit violations or controller saturation.

Table 2 shows the comparison between the four methods. The simulation required 661 steps. Each column shows the number of times each task was performed and the average cost per time step. The last column shows the total cost for the simulation for each method. The cost for the right-hand side evaluation is either $C_{trhs} + C_{srhs}$ or $C_{srhs2} + C_{srhs}$ depending on the method.

On average, the NR method for the trajectory computation required 5.23 iterations for SICM, 4.42 iterations for SDM and 4.85 iterations for SCM and SHM per time step. This is an expected result [12] because SDM uses the most updated version of the Jacobian, while SICM approximates the Jacobian by its block diagonal part in addition to the fact that it keeps it constant for a number of time steps.

Table 2

$\text{Number of times the operation is executed} \big/ \text{Cost of operation (Kflops) per time step}$

Method	Jacobian evaluation	Jacobian factorization	Forward-Backward substitution	Right-hand side evaluation	GMRES or Quasi-NR	Total Cost (Mflops)
SICM	$70/17.6$	$70/188$	$5882/25.5$	$3602/49.9$		326.18
SDM	$564/17.6$	$564/188$	$3143/25.5$	$3804/13.8$		243.3
SCM	$565/17.6$	$31/188$	$2885/25.5$	$4050/13.9$	$534/157.3$	228.96
SHM	$565/17.6$	$31/188$	$2885/25.5$	$4050/13.9$	$534/152.4$	226.3

Table 3
Number of times the operation is executed / Cost of operation (Kflops) per time step

Method	Jacobian evaluation	Jacobian factorization	Forward-Backward substitution	Right-hand side evaluation	GMRES or Quasi-NR	Total Cost (Mflops)
SCM	661/17.5	78/188	4649/25.8	5995/14.4	586/163.8	328.86
SHM	661/17.5	78/188	4649/25.8	5995/14.4	586/146.2	318.53

It is evident from Table 2 that SCM and SHM outperform SDM and SICM. This difference will be more obvious for larger systems, where the cost ratio between C_{fac} and C_{fbs} will be greater. (For this system the ratio was 7.1:1.)

It was shown earlier in the computational complexity analysis that the difference in performance between SCM and SHM depends on the number of iterations that the quasi-NR scheme and GMRES respectively, require to converge. For this example, the quasi-NR scheme of SCM required on average 1.69 more iterations to converge, making SHM slightly more efficient.

It is shown in Table 2 that the difference in performance between GMRES (SHM) and the quasi-NR method (SCM) is about 5 Kflops per time step. This difference may grow if the Jacobian becomes ill-conditioned as in the case of an unstable system. The quasi-NR method is more sensitive than GMRES to the ill-conditioning of the Jacobian (unless of course, arithmetic errors are introduced). To illustrate this fact for the 50-machine system, we set the clearing time to 0.31 seconds in order to simulate an unstable system. Table 3 shows the results for the SCM and SHM.

From Table 3 we see that the difference between GMRES and the quasi-NR method is now 17.6 Kflops per time step. On average $N_4 - N_5 = 2.71$, thus the difference in the performance.

9. Conclusions

In this paper we applied three well-established methods for the sensitivity calculations of differential-algebraic equations to power systems. We also proposed a new method that involves using an iterative linear solver (GMRES) for the sensitivity computations. The results show that this new method is slightly better than the other methods when the sensitivity of the trajectories with respect to only one parameter is required. Future work will focus on computing a full set of sensitivities through the use of the block-GMRES method, and on testing the techniques on larger systems.

Acknowledgements

We acknowledge the support of National Science Foundation through its grant NSF-ECS 00-00474 and the Complex Interactive Networks/Systems Initiative of EPRI-DoD.

References

[1] P.M. Frank, *Introduction to System Sensitivity Theory*, Academic Press, New York, 1978.

[2] I.A. Hiskens, “Stability of limit cycles in hybrid systems,” *Proceedings 34th Hawaii International Conference on System Sciences*, Maui, HI, January 2001.

[3] I.A. Hiskens, “Nonlinear Dynamic Model Evaluation from Disturbance Measurements,” to appear *IEEE Trans. On Power Systems*.

[4] T. Maly and L. R. Petzold, “Numerical methods and software for sensitivity analysis of differential-algebraic systems,” *Applied Numerical Mathematics*, 20, (1996), pp. 57-79.

[5] M. Caracotsios and W.E. Stewart, “Sensitivity analysis of initial value problems with mixed ODEs and algebraic constraints,” *Comput. Chem. Engrg*. 9 (1985) pp. 359-365.

[6] W.F. Feehery, J.E. Tolsma, P.I. Barton, “Efficient sensitivity analysis of large-scale differential-algebraic systems,” *Applied Numerical Mathematics*, 25, (1997), pp. 41-54.

[7] Y. Saad, *Iterative Methods For Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.

[8] P. W. Sauer and M. A. Pai, *Power System Dynamics and Stability*, Prentice Hall, Englewood Cliffs, NJ, 1998.

[9] I.A.Hiskens and M.A.Pai, “Trajectory sensitivity analysis of Hybrid Systems,” *IEEE Transactions On Circuits and Systems Part I Fundamental Theory and Applications*, vol 47, no 2, Feb 2000, pp. 204-220.

[10] Dembart B., et al, “Power System Dynamic Analysis – Phase 1,” *Final Report on EPRI Project RP670-1, EL484*, Electric Power Research Institute, Palo Alto, California, July 1997.

[11] Fernardo L. Alvarado, William F. Tinney, Mark K. Enns, “Sparsity in Large-Scale Network Computation,” Control and Dynamic Systems, vol. 41, 1991, pp. 207-272.

[12] S. Li, L. Petzold, W. Zhu, “Sensitivity analysis of differential-algebraic equations: A comparison of methods on a special problem,” *Applied Numerical Mathematics*, 32, (2000), pp. 161-174.

[13] V. Simoncini and E. Gallopoulos, “Convergence properties for block GMRES and matrix polynomials,” *Lin. Alg. Appl.* 247, Nov. 1996, pp. 97-119.