Using Krylov-Subspace Iterations in Discontinuous Galerkin Methods for Nonlinear Reaction-Diffusion Systems*

Donald J. Estep[1] and Roland W. Freund[2]

[1] School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA
[2] Bell Laboratories, Room 2C–420, 700 Mountain Avenue, Murray Hill, NJ 07974–0636, USA

Abstract. We consider discontinuous in time and continuous in space Galerkin finite-element methods for the numerical solution of reaction-diffusion differential equations. These are implicit methods that require the solution of a system of nonlinear equations at each time node. In this paper, we explore the use of Krylov-subspace techniques for the iterative solution of the linear systems that arise when these nonlinear systems are solved by means of Newton-type methods. It is shown how these linear systems depend on the choice of the basis functions used for the time discretization. We demonstrate that Krylov-subspace methods can be sped up considerably by employing an orthogonal basis for the time discretization and by combining the Krylov iteration with a suitable block preconditioner. Results of numerical experiments are reported.

1 Introduction

Many fundamental models in science take the form of time-dependent nonlinear reaction-diffusion differential equations. Examples include the modeling of domain walls in ferromagnetic materials, predator-prey models, the description of superconductivity in liquids, the modeling of the famous Belousov-Zhabotinsky reaction in chemical kinetics, the modeling of flame propagation, and the modeling of the spread of rabies in foxes. As well as being important for physical modeling purposes, solutions of reaction-diffusion equations can also exhibit complex and beautiful behavior arising primarily from the competition between reaction and diffusion and the nonlinear nature of the equations. However, the numerical solution of reaction-diffusion equations is correspondingly difficult.

In this paper, we consider systems of D reaction-diffusion equations consisting of d, $1 \le d \le D$, parabolic equations and $D - d$ ordinary equations

* Numerical Analysis Manuscript No. 99–3–12, Bell Laboratories, Murray Hill, New Jersey, May 1999. Available on WWW at `http://cm.bell-labs.com/cs/doc/99`.

for the $\mathbb{R}^D$-valued function $\mathbf{u} = [u_i]_{1 \le i \le D}$:

$$\begin{aligned} \frac{\partial u_i}{\partial t} - \nabla \cdot (\epsilon_i(\mathbf{u},\mathbf{x},t)\,\nabla u_i) = f_i(\mathbf{u},\mathbf{x},t), &\quad (\mathbf{x},t) \in \Omega \times \mathbb{R}^+,\ 1 \le i \le D, \\ u_i(\mathbf{x},t) = 0, &\quad (\mathbf{x},t) \in \partial\Omega \times \mathbb{R}^+,\ 1 \le i \le d, \\ \mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}), &\quad \mathbf{x} \in \Omega. \end{aligned} \tag{1}$$

Here, Ω is a convex polygonal domain in $\mathbb{R}^N$, where $N \in \{1,2,3\}$, with boundary $\partial\Omega$. We assume that there is a constant $\epsilon > 0$ such that

$$\epsilon_i(\mathbf{u},\mathbf{x},t) \ge \epsilon \quad \text{for} \quad 1 \le i \le d \quad \text{and} \quad \epsilon_i(\mathbf{u},\mathbf{x},t) \equiv 0 \quad \text{for} \quad d < i \le D.$$

Note that, in (1), there is no parabolic smoothing in the $D-d$ "singular" equations. To compute numerical solutions of systems of the form (1), we employ the finite-element space-time discretization of (1) called the *discontinuous Galerkin method* (dG method); see, e.g., [2,3,10] and the references given there. The reason for employing space-time finite element methods is to take advantage of the new approach to computational error estimation based on residuals and variational analysis; see [3,4]. Let $q \ge 1$ be an integer. The dG(q) method uses discontinuous (in time) approximations that are piecewise polynomials of degree at most q in time and piecewise linear polynomials in space. The approximations are allowed to be discontinuous at the time nodes, but for fixed time, they are continuous in space. It is well known that dG(q) methods offer a number of advantages over their continuous counterparts. These advantages include an increased convergence order at the time nodes, better behavior for long-time integration of parabolic problems, and the easy implementation of adaptive spatial grids that are changed with time. On the other hand, dG(q) methods can be significantly more expensive than the corresponding continuous Galerkin methods, especially when no attention is paid to the linear algebra problems that actually dominate the computational costs. In this paper, we explore the use of Krylov-subspace methods for the linear algebra problems arising in discontinuous Galerkin methods.

2 Discontinuous Galerkin Discretizations

For simplicity, from now on we write $\epsilon_i(\mathbf{u}) = \epsilon_i(\mathbf{u},\mathbf{x},t)$ and $f_i(\mathbf{u}) = f_i(\mathbf{u},\mathbf{x},t)$. We partition $[0,\infty)$ as $0 = t_0 < t_1 < t_2 < \cdots < t_n < \cdots$, denoting each time interval by $I_n := (t_{n-1}, t_n]$ and time step by $k_n := t_n - t_{n-1}$. To each interval I_n, we associate a triangulation $\mathcal{T}_n$ of Ω. Note that $\mathcal{T}_n$ is allowed to change across time nodes.

The dG(q) approximations are polynomials of degree at most q in time and piecewise polynomials in space on each space-time "slab" $S_n = \Omega \times I_n$. In space, we let $\mathbf{V}_n \subset (H_0^1(\Omega))^d \times (H^1(\Omega))^{D-d}$ denote the space of piecewise linear continuous vector-valued functions $\mathbf{v}(\mathbf{x}) \in \mathbf{R}^D$ defined on $\mathcal{T}_n$, where the first d components of $\mathbf{v}$ are zero on $\partial\Omega$. Then on each slab, we define

$$\mathbf{W}_n^q := \{\, \mathbf{w}(\mathbf{x},t) \mid \mathbf{w}(\mathbf{x},t) = \sum_{j=0}^{q} t^j \mathbf{v}_j(\mathbf{x}),\, \mathbf{v}_j \in \mathbf{V}_n,\, (\mathbf{x},t) \in S_n \,\}. \tag{2}$$

Furthermore, we let $\mathbf{W}^q$ denote the space of functions defined on the space-time domain $\Omega \times \mathbf{R}^+$ such that $\mathbf{v}|_{S_n} \in \mathbf{W}_n^q$ for $n \geq 1$. Note that functions in $\mathbf{W}^q$ are generally discontinuous across the discrete time levels, and we denote the jump across t_n by $[\mathbf{w}]_n = \mathbf{w}_n^+ - \mathbf{w}_n^-$ where $\mathbf{w}_n^\pm = \lim_{s \to t_n \pm} \mathbf{w}(s)$. Finally, $\mathbf{P}_n : L^2(\Omega) \to \mathbf{V}_n$ denotes the L^2-projection operator onto $\mathbf{V}_n$, and $(\cdot,\cdot)$ is the $L_2(\Omega)$-inner product.

The dG(q) approximation to the solution of (1) is then defined as the function $\mathbf{U} = [U_i]_{1\leq i \leq D} \in \mathbf{W}^q$ that satisfies $\mathbf{U}_0^- = \mathbf{P}_0\, \mathbf{u}_0$ and for $n \geq 1$,

$$\int_{t_{n-1}}^{t_n} \big((\dot{U}_i, v_i) + (\epsilon_i(\mathbf{U})\nabla U_i, \nabla v_i)\big)\, dt + \big([U_i]_{n-1}, v_i^+\big) = \int_{t_{n-1}}^{t_n} (f_i(\mathbf{U}), v_i)\, dt$$
$$\text{for all} \;\; \mathbf{v} = [v_i]_{1\leq i\leq D} \in \mathbf{W}_n^q,\;\; 1 \leq i \leq D. \tag{3}$$

In practice, some of the integrals in (3) are computed using quadrature. In particular in space, we evaluate the integrals involving some form of mass matrix, i.e., $(\dot{U}_i, v_i)$, $\big([U_i]_{n-1}, v_i^+\big)$, and $(f_i(\mathbf{U}), v_i)$, using the lumped mass, or composite trapezoidal rule, quadrature. The choice of this quadrature rule is dictated by stability considerations; see [4].

Equations (3) defining the dG(q) approximation result in a large sparse system of nonlinear equations that needs to be solved on each space-time slab S_n. For this task, we use an inexact Newton method combined with preconditioned Krylov-subspace iterations to obtain an approximate solution of the large sparse linear system for the Newton direction that needs to be solved at each Newton iteration. In the next section, we discuss the structure of these linear systems for the special case of the dG(1) method.

3 Structure of the Linear Systems for the Case $q = 1$

For simplicity, from now on, we assume that the ϵ_i's in (1) are all constant, and that the triangulation $\mathcal{T} = \mathcal{T}_n$ of Ω is the same for all time nodes. The usual nodal basis associated with $\mathcal{T}$ is used for the spatial functions in the ansatz space (2). In the sequel, we consider only the dG(1) method. Extensions to dG(q) methods with $q \geq 2$ will be described elsewhere.

For each space-time slab S_n, we need to solve the nonlinear system

$$\mathbf{F}_n(\mathbf{U}) = \mathbf{0}, \quad \text{where} \quad \mathbf{U} = \begin{bmatrix} \mathbf{U}_n^- \\ \mathbf{U}_{n-1}^+ \end{bmatrix}, \tag{4}$$

for the approximate solutions $\mathbf{U}_n^-$ and $\mathbf{U}_{n-1}^+$ corresponding to the right and left end point of the interval $I_n = (t_{n-1}, t_n]$. Note that (4) is a system of $2Dm$

equations for a total of $2Dm$ unknowns, where m denotes the the number of grid points in the triangulation $\mathcal{T}$ of Ω.

Each step of a Newton method applied to (4) requires the solution of a linear system with the Jacobian matrix $\mathbf{J} = D\mathbf{F}_n(\mathbf{U})$ of $\mathbf{F}_n(\mathbf{U})$. It turns out that the structure of $\mathbf{J}$ depends on the choice of the basis function for the time discretization in the ansatz space (2). Recall that for dG(1), the approximation is linear on each time interval I_n so there are two basis functions for each I_n.

First, we consider the usual nodal basis functions that are 1 at one end point and 0 at the other end point. In this case, the Jacobian matrix $\mathbf{J}$ is of the form

$$\mathbf{J}_{\mathcal{N}} = \frac{1}{2}\begin{bmatrix} \mathbf{B} & -\mathbf{B} \\ \mathbf{B} & \mathbf{B} \end{bmatrix} + \frac{k_n}{3}\begin{bmatrix} \boldsymbol{\epsilon}\,\mathbf{A} & \boldsymbol{\epsilon}\,\mathbf{A}/2 \\ \boldsymbol{\epsilon}\,\mathbf{A}/2 & \boldsymbol{\epsilon}\,\mathbf{A} \end{bmatrix} - k_n\,\mathbf{R}_{\mathcal{N}}. \tag{5}$$

Here, $\mathbf{A}$ and $\mathbf{B}$ are the stiffness and mass matrix associated with the spatial discretization, and $\boldsymbol{\epsilon} := \operatorname{diag}(\epsilon_1, \epsilon_2, \ldots, \epsilon_D)$. Note that, in (5), the first term corresponds to the time-derivative part in (1), the second term to the diffusion part in (1), and the third term to the reaction part in (1). In practice, mass lumping is used, so that $\mathbf{B}$ is a diagonal matrix. Moreover, for equidistant regular spatial grids, $\mathbf{B} = \mathbf{I}$ is the identity matrix. The matrix $\mathbf{R}_{\mathcal{N}}$ has nonzero elements only in a total of $4D - 1$ diagonals.

A second standard choice for the time basis functions is an orthogonal (on I_n) basis where one function is constant on I_n and the other is the linear function with the values 1 and -1 at the end points t_{n-1} and t_n of I_n. For the general dG(q) method, the orthogonal basis is given by the first $q+1$ Legendre polynomials (translated from $[-1, 1]$ to I_n). In this case, the Jacobian matrix is of the form

$$\mathbf{J}_{\mathcal{O}} = \begin{bmatrix} \mathbf{B} & -\mathbf{B} \\ \mathbf{B} & \mathbf{B} \end{bmatrix} + k_n\begin{bmatrix} \boldsymbol{\epsilon}\,\mathbf{A} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\epsilon}\,\mathbf{A}/3 \end{bmatrix} - k_n\,\mathbf{R}_{\mathcal{O}}. \tag{6}$$

Finally, a third choice is to use a "mixed" basis with the nodal basis for the solution $\mathbf{U}$ and the orthogonal basis for the test functions $\mathbf{v}$ in (3). The resulting Jacobian is now of the form

$$\mathbf{J}_{\mathcal{M}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix} + \frac{k_n}{6}\begin{bmatrix} \boldsymbol{\epsilon}\,\mathbf{A} & -\boldsymbol{\epsilon}\,\mathbf{A} \\ 3\,\boldsymbol{\epsilon}\,\mathbf{A} & 3\,\boldsymbol{\epsilon}\,\mathbf{A} \end{bmatrix} - k_n\,\mathbf{R}_{\mathcal{M}}. \tag{7}$$

Note that the mixed basis makes the time-derivative part of $\mathbf{J}_{\mathcal{M}}$ block-diagonal. The mixed basis is used in the reaction-diffusion solver Cards [5,6]. The motivation for this choice is that in (7), $J_{\mathcal{M}}$ becomes a very simple matrix as the time step k_n is reduced. In fact, for equidistant regular spatial grids, $\mathbf{B} = \mathbf{I}$ and thus $\mathbf{J}_{\mathcal{M}} \to \mathbf{I}$ as $k_n \to 0$. This is exploited in Cards, in the sense that the time step is reduced every time the Krylov-subspace linear solver does not converge or converges too slowly, so that the linear system becomes easier to solve. However, this strategy often results in the use of exceedingly tiny time steps that are dictated solely by the linear algebra, but not by the dG(1) discretization. We now demonstrate that such an "artificial" reduction of the time step can be easily avoided by employing the orthogonal basis for the time discretization and by combining the Krylov iteration with a suitable block preconditioner.

The convergence of Krylov-subspace methods depends on the spectral properties of the coefficient matrix of the linear system to be solved; see, e.g., [9] and the references given there. For the numerical results presented in this paper, we use the TFQMR Krylov-subspace method [8]. Convergence results for TFQMR can be found in [7]. It turns out that the spectral properties of the three matrices (5)–(7), which result from the three described choices of the basis functions for the time discretization, are vastly different. In particular, the matrix (6) corresponding to the orthogonal basis usually has a spectrum that is most amenable to Krylov-subspace iterations. Moreover, we note that the sparsity of the matrices (5)–(7) is dominated by the stiffness matrix $\mathbf{A}$ in the diffusion part. Thus the orthogonal basis also leads to the sparsest Jacobian matrix (6) with roughly half as many nonzero entries as the two other matrices (5) and (7). This helps to further reduce the computational costs of the Krylov-subspace method. Therefore, we recommend the choice of the orthogonal basis.

4 Preconditioning

In order to speed up Krylov-subspace methods, the basic iteration is combined with preconditioning. The key idea is that instead of the original linear system, $\mathbf{J}\,\mathbf{u} = \mathbf{b}$, the Krylov-subspace iteration is applied to the preconditioned system

$$\left(\mathbf{M}_1^{-1}\,\mathbf{J}\,\mathbf{M}_2^{-1}\right)\ \mathbf{u}' = \mathbf{M}_1^{-1}\,\mathbf{b},$$

and the solution of the original system is then obtained as $\mathbf{u} = \mathbf{M}_2^{-1}\,\mathbf{u}'$. The matrix $\mathbf{M} = \mathbf{M}_1 \cdot \mathbf{M}_2$ is called the preconditioner. It needs to be such that systems with $\mathbf{M}_1$ and $\mathbf{M}_2$ are "easy" to solve and that the spectrum of the preconditioned matrix, $\mathbf{M}_1^{-1}\,\mathbf{J}\,\mathbf{M}_2^{-1}$, is more amenable to the employed Krylov-subspace method. The latter requirement usually means that the eigenvalues of $\mathbf{M}_1^{-1}\,\mathbf{J}\,\mathbf{M}_2^{-1}$ should be clustered around the point 1 and bounded away from the origin 0 in the complex plane.

We now discuss two simple preconditioners for the Jacobian matrices arising in the dG(1) method. As explained, in the previous section, we use the orthogonal basis. Furthermore, we scale the second block row and the second block column of $\mathbf{J}_{\mathcal{O}}$ in (6) by $\sqrt{3}$ so that the stiffness matrix is equally weighted. The resulting Jacobian matrix is

$$\mathbf{J} = \begin{bmatrix} \mathbf{B} & -\sqrt{3}\,\mathbf{B} \\ \sqrt{3}\,\mathbf{B} & 3\,\mathbf{B} \end{bmatrix} + k_n \begin{bmatrix} \boldsymbol{\epsilon}\,\mathbf{A} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\epsilon}\,\mathbf{A} \end{bmatrix} - k_n\,\mathbf{R}. \tag{8}$$

A simple, but often efficient preconditioner is SSOR. It has the advantage that due to the so-called Eisenstat trick [1], this preconditioner can be implemented so that it requires only little extra computational work, compared with the work per iteration for the unpreconditioned system. The idea is to split the matrix $\mathbf{J} = \mathbf{L} + \boldsymbol{\Delta} + \mathbf{U}$ into its strictly lower part $\mathbf{L}$, its diagonal part $\boldsymbol{\Delta}$, and its strictly upper part $\mathbf{L}$. The SSOR preconditioner is then given by

$$\mathbf{M}_1 = (\mathbf{L} + \boldsymbol{\Delta})\ \boldsymbol{\Delta}^{-1}, \quad \mathbf{M}_2 = \mathbf{U} + \boldsymbol{\Delta}.$$

However, for the linear systems arising in the dG(1) method applied to reaction-diffusion systems (1), SSOR only leads to marginal speed-ups.

The key to obtain an efficient preconditioner is to first reorder the linear system and then to use a block variant of SSOR. Note that in $\mathbf{J}\,\mathbf{u} = \mathbf{b}$ with $\mathbf{J}$ given in (8), there are $2D$ unknowns per spatial grid point. However, in (8), these unknowns are not grouped together, but rather they are spaced $m-1$ apart, where m is the number of spatial grid points. We now reorder $\mathbf{J}$ so that all unknowns associated with each spatial grid point are grouped together. This reordering is done by means of a permutation matrix $\mathbf{P}$. We then consider the following block-wise splitting of the resulting reordered matrix:

$$\mathbf{P}^{\mathrm{T}}\,\mathbf{J}\,\mathbf{P} = \mathbf{L}_{\mathcal{B}} + \boldsymbol{\Delta}_{\mathcal{B}} + \mathbf{U}_{\mathcal{B}}.$$

Here, $\boldsymbol{\Delta}_{\mathcal{B}}$ is a block-diagonal matrix with $2D \times 2D$ diagonal blocks, $\mathbf{L}_{\mathcal{B}}$ is strictly lower block-triangular, and $\mathbf{U}_{\mathcal{B}}$ is strictly upper block-triangular. The block-SSOR preconditioner is then given by

$$\mathbf{M}_1 = \mathbf{P}^{\mathrm{T}}\ (\mathbf{L}_{\mathcal{B}} + \boldsymbol{\Delta}_{\mathcal{B}})\ \boldsymbol{\Delta}_{\mathcal{B}}^{-1}, \quad \mathbf{M}_2 = \mathbf{P}\ (\mathbf{U}_{\mathcal{B}} + \boldsymbol{\Delta}_{\mathcal{B}})\,.$$

5 Numerical Examples

In this section, we report numerical results for two representative examples. In both cases, the spatial domain is the two-dimensional unit square $\Omega = [0,1] \times [0,1]$, and a regular equidistant mesh is used for Ω. We stress that the preconditioners discussed in this paper can also be employed in the more general situation when the mesh points change across time nodes.

The first example is a predator-prey model; see, e.g., [11,12]. More precisely, we use a model that is analyzed in [12]. This is a system of the form (1) with $D = d = 2$ and functions

$$f_1(\mathbf{u}) = u_1\,(-\,(u_1 - a_1)\,(u_1 - 1) - a_2 u_2)\,,\ f_2(\mathbf{u}) = u_2\,(-a_3 - a_4 u_2 + a_2 u_1)\,,$$

where $a_1, a_2, a_3, a_4 > 0$ are real parameters. For our numerical tests, we set $a_1 = 0.25$, $a_2 = 2$, $a_3 = 1$, and $a_4 = 3.4$. Furthermore, in (1), the diffusion coefficients are the constants $\epsilon_1 = \epsilon_2 = 0.01$. A 64×64 grid is used, resulting in $m = 63^2 = 3969$ spatial grid points and a total matrix size of $2Dm = 15876$. In Figure 1, we show the relative residuals for four TFQMR runs with the same matrix. The first run is without preconditioning, the other three runs are with SSOR, block-diagonal ($\mathbf{M}_1 = \mathbf{I}$, $\mathbf{M}_2 = \boldsymbol{\Delta}_{\mathcal{B}}$), and block-SSOR preconditioning. Clearly, block-SSOR results in the most efficient preconditioner.

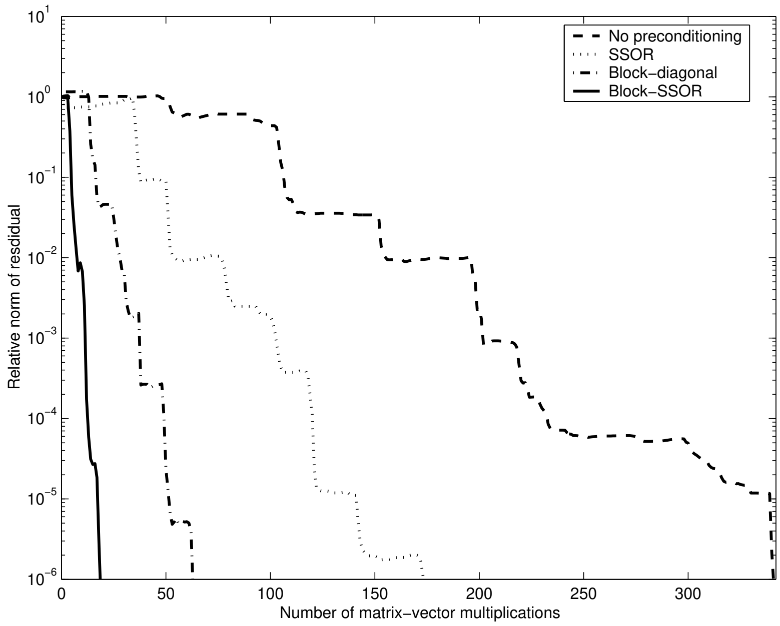

Fig. 1. TFQMR residual history for predator-prey example

The second example arises form the bistable equation; see, e.g., [4] and the references given there. This is an equation of the form (1) with $D = d = 1$ and $f_1(u_1) = \alpha\left(u_1 - u_1^3\right)$. For our tests, we chose the constant $\alpha = 1111.111$ and the constant diffusion coefficient $\epsilon_1 = 1$. We use a small 16×16 grid with $m = 15^2 = 225$ grid points, resulting in a total matrix size of $2Dm = 450$. The small grid was chosen so that we can compute the complete spectrum of the unpreconditioned and the block-SSOR preconditioned matrix. In Figure 2, we show the resulting spectra. Note that the unpreconditioned matrix has eigenvalues on both sides of the origin, causing TFQMR to converge slowly, while block-SSOR nicely clusters the eigenvalues about 1. In Figure 3, we show the relative residuals for TFQMR runs with the unpreconditioned matrix, SSOR, block-diagonal, and block-SSOR preconditioning.

Acknowledgment. The research of the first author was partially supported by the National Science Foundation, DMS 9805748.

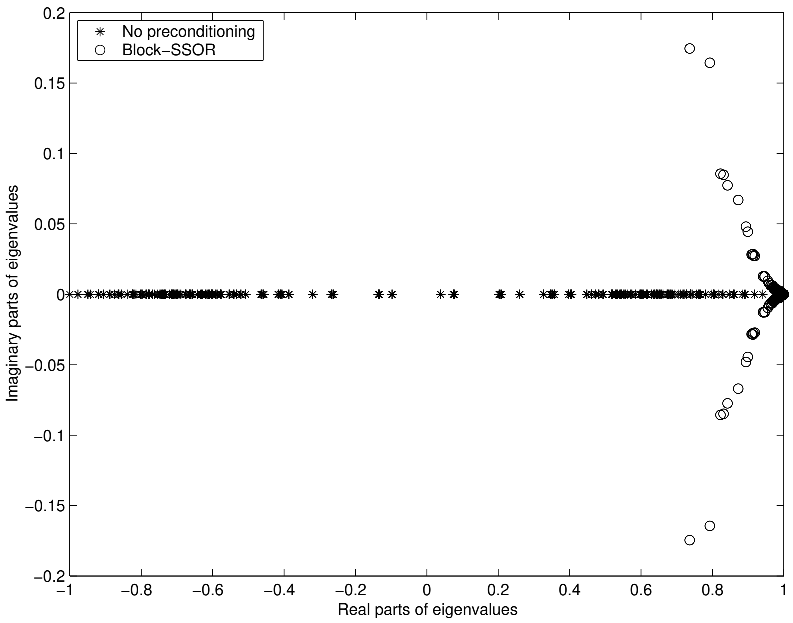

Fig. 2. Spectra for bistable example

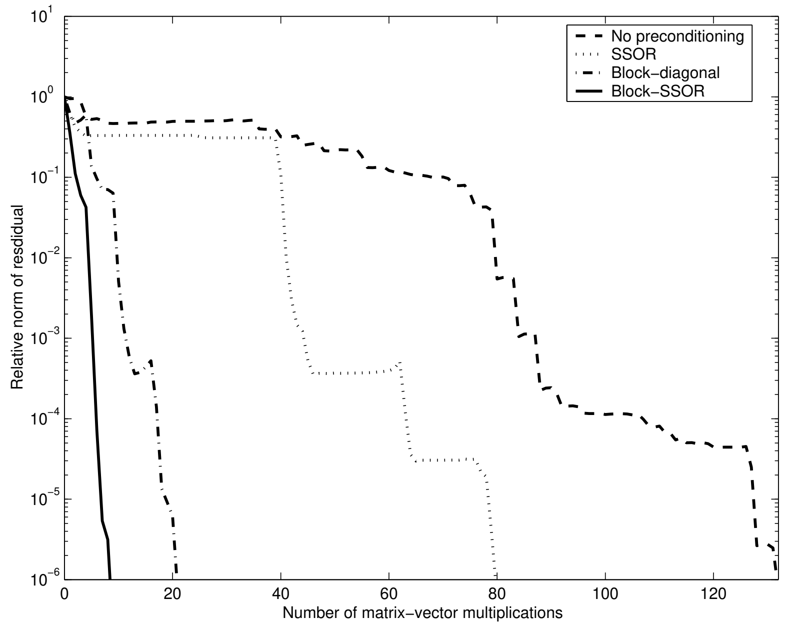

Fig. 3. TFQMR residual history for bistable example

References

1. Eisenstat, S.C.: Efficient implementation of a class of preconditioned conjugate gradient methods. SIAM J. Sci. Statist. Comput. **2** (1981) 1–4
2. Eriksson, K., Estep, D., Hansbo, P., Johnson, C.: Introduction to adaptive methods for differential equations. Acta Numerica **4** (1995) 105–158
3. Eriksson, K., Estep, D., Hansbo, P., Johnson, C.: Computational Differential Equations. Cambridge University Press, New York, 1996
4. Estep, D., Larson, M., Williams, R.: Estimating the error of numerical solutions of systems of reaction-diffusion equations. Mem. Amer. Math. Soc., 1999, to appear
5. Estep, D., Williams, R.: Accurate parallel integration of large sparse systems of differential equations. Math. Models Meth. Appl. Sci. **6** (1996) 535–568
6. Estep, D., Williams, R.: Cards: Concurrent Adaptive Reaction-Diffusion Solver, Version 2.0 (1997)
7. Freund, R.W.: Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations. In: Numerical Methods of Approximation Theory (D. Braess and L. L. Schumaker, eds.). Birkhäuser, Basel, 1992, pp. 77–95
8. Freund, R.W.: A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. SIAM J. Sci. Comput. **14** (1993) 470–482
9. Freund, R.W., Golub, G.H., Nachtigal, N.M.: Iterative solution of linear systems. Acta Numerica **1** (1992) 57–100
10. Makridakis, Ch.G., Babuška, I.: On the stability of the discontinuous Galerkin method for the heat equation. SIAM J. Numer. Anal. **34** (1997) 389–401
11. Murray, J.D.: Mathematical Biology, 2nd edition. Springer-Verlag, New York, 1993
12. Smoller, J.A.: Shock Waves and Reaction-Diffusion Equations, 2nd edition. Springer-Verlag, New York, 1994