Parallel solution of reaction-diffusion equations with a stabilization method

Uwe Kleis

Section of Mathematics, TU Hamburg-Harburg, 21071 Hamburg, Germany,
kleis@tu-harburg.de

Abstract. We present a method for the computation of stable and unstable solutions of a discretized reaction-diffusion equation. Fixed point solvers are frequently used for this problem. But they diverge for many applications. A method for the stabilization of these solvers has been introduced in [3] and [6].

This (sequential) method is now modified so that the algorithm allows for parallel execution. The parallelization is done on a high level and enables the simultaneous execution of the main algorithmic blocks.

1 Introduction

We consider a standard problem in Chemical Engineering: The solution of a stationary reaction-diffusion equation. A satisfactory numerical treatment of these boundary value problems requires a fine-grain discretization of the differential equation and one gets large-dimensional nonlinear systems of equations of the form

$$Au = F(u), \quad u \in \mathbb{R}^n. \tag{1}$$

Here A is a discrete diffusion operator and $F(u)$ describes the reaction rate of the process. The matrix A will normally be sparse as it describes diffusive processes with interactions only between neighbouring points on the grid.

To simplify the presentation, we assume in this work that A is symmetric positive definite and $F'(u)$ is symmetric. This models the treatment of a single reaction-diffusion equation without convection. Notice, however, that the proposed method can be extended to the nonsymmetric case including convective reaction-diffusion systems.

Picard's method

$$u_{k+1} = A^{-1}F(u_k) \tag{2}$$

is an easily implemented iteration for solving (1). It does not require Jacobians or the storage of any full matrix of dimension n. It allows to incorporate elaborate PDE-Software for the computation of A^{-1}.

Unfortunately, iteration (2) may not converge or converges only slowly if the spectral radius $\rho(A^{-1}F'(u))$ is greater than or close to one at the solution u^*. This can happen when simulating parameter dependent problems

$$Au = F(u, \mu)$$

with varying Thiele-modulus μ (cf. [1]), e.g. A small change in the parameter μ can render a currently stable iteration unstable as the spectral radius crosses the value one. In such a case, only a few unstable modes prevent the solver from converging.

For symmetric systems, Jarausch and Mackens developed in [3] the 'Condensed-Newton-Supported-Picard' (CNSP-) Method to stabilize fixed point iterations of type (2) when there are only a few unstable modes. They split the system into two parts by projection onto the subspace of unstable modes and its orthogonal complement, respectively. The small and unstable part (of dimension $m \ll n$) is processed by a Newton solver while for the remaining large part the fixed point solver is rendered stable through the extraction of the unstable modes.

Shroff and Keller introduced in [6] the 'Recursive Projection Method' (RPM) for the unsymmetric case. But since both methods are based on the same idea and the symmetric case is easier to explain, we shall demonstrate our parallel modification for the symmetric method here. We modify this method in such a way that the main algorithmic blocks are executed in parallel. A similar parallelization of the unsymmetric approach is under development.

In the following section we give a short outline of the CNSP-Method. In section 3 we present the new variant of the method which allows for parallelization. Section 4 demonstrates this approach for a reaction-diffusion equation and shows the new algorithmic properties. We conclude with a short outlook in chapter 5.

2 CNSP-Method

Let X be $\mathbb{R}^n$ with the inner product $\langle x, y \rangle_A := x^T A y$. For an iterate u_k and a given value $\gamma_{max} \in [0, 1)$ we let

$$\begin{aligned} |\lambda_1(u_k)| \geq |\lambda_2(u_k)| \geq \ldots \geq |\lambda_m(u_k)| \geq \quad & \gamma_{max} \\ > |\lambda_{m+1}(u_k)| \geq \ldots \geq |\lambda_n(u_k)| & \end{aligned}$$

be the eigenvalues of the matrix $A^{-1}F'(u_k)$ with the approximate and A-orthogonal eigenvectors $z_1, z_2, \ldots, z_n$.

We define a matrix $Z := (z_1, \ldots, z_m)$ by the eigenvectors corresponding to the eigenvalues with modulus larger than γ_{max}. We call these dominant eigenvectors "supports". With Z, we define the orthogonal projectors

$$P := ZZ^T A, \quad Q := I - P,$$

which project the problem

$$R(u) := u - A^{-1}F(u) = 0 \tag{3}$$

onto two A-orthogonal subspaces: $X = PX \oplus QX$. This splits (3) into the two subsystems

$$f_P(c,q) := c - Z^T F(Zc+q) = 0$$
$$f_Q(q,c) := QA^{-1}F(Zc+q) = q$$

with the variables q and c from $u = Pu + Qu$ and

$$Zc := p := Pu, \quad c \in \mathbb{R}^m,$$
$$q := Qu.$$

The small, m-dimensional "P-system" $f_P(c,q) = 0$ is iterated for the variable c (with q fixed) by Newton's method. This iteration reduces the "P-residuum" $r_p(u) := \|PR(u)\|_A^2$ where the A-norm is defined as

$$\|u\|_A := \sqrt{\langle u,u\rangle_A}.$$

The large "Q-system" $f_Q(q,c) = q$ can be processed (c is fixed) by the original (parallel) fixed point solver which is contractive now. A Q-step reduces the Q-residuum (the "local residuum" of the Q-system) $r_q(u) := \|QR(u)\|_A^2$ by a contraction factor of γ_{max}. This turns out to be the overall linear convergence rate, too.

The total residuum $r(u) := \|R(u)\|_A^2$ is connected to the local residua by

$$r(u) = r_p(u) + r_q(u).$$

If the supports $(z_1, \ldots, z_m)$ approximate the dominant eigenvectors sufficiently well, the reduction of $r_p(u)$ does not increase $r_q(u)$ too much, and vice versa. Thus, a step in a subsystem results in an absolute decrease of the total residuum $r(u)$ comparable to the absolute decrease of the currently treated local residuum.

3 Algorithm

First, we give a very short outline of the sequential method as presented in [3]. In the following subchapter we introduce our new variant of the method. The new parallel variant takes advantage of the inherent parallel nature of the algorithm.

3.1 Sequential Algorithm

The sequential algorithm starts with a simultaneous power iteration to determine an approximate matrix Z and to split the system into the two subsystems. Then, the step which promises more gain in its residuum is executed. This is usually the one with the larger local residuum. Now, as shown in

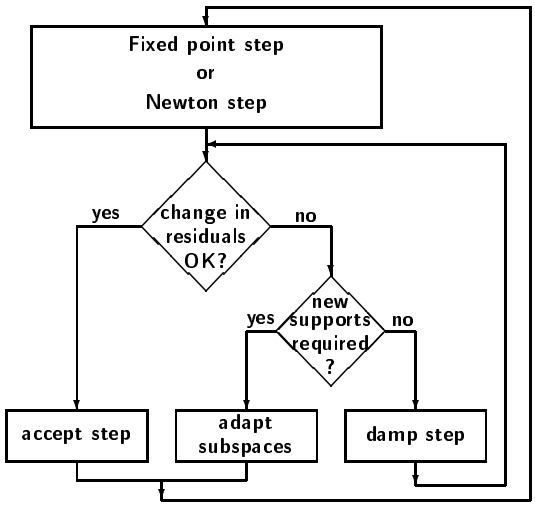

Fig. 1. Flowchart for sequential method (simplified)

Fig. 1, the total residuum $r(u)$ is checked for this step. The step is accepted if the decrease of the total residuum is comparable to the decrease of the local residuum. Otherwise, it is concluded that the step is too large or that the subspaces are not sufficiently invariant. Based on additional criteria, it is decided whether the step has to be damped or whether new approximate eigenvectors $z_1(u_k), \ldots, z_s(u_k)$ have to be calculated.

Notice, that the supporting vectors as well as their number can change as the iteration proceeds and the iterate u_k changes.

3.2 Parallel Algorithm

The sequential algorithm inheres an obvious approach to parallelization. We modify the method, so that the tasks

- Newton type steps on the small unstable subspace,
- fixed point steps on the large stable subspace,
- adaptation of the support vectors

are executed in parallel. The three tasks are shown as dark grey blocks in the flowchart in Fig. 2.

As in the sequential case we start with the calculation of a splitting of the subspaces. With these supports, the Newton and the fixed point solver start running and execute steps in their subsystems without communication. Different numbers of substeps can be executed in the solvers. The individual local iterations cease for synchronisation due to a sufficient reduction of their local residua or due to too high an increase of the complementary residuum. We enter the light grey block in Fig. 2, then. At this point, we combine

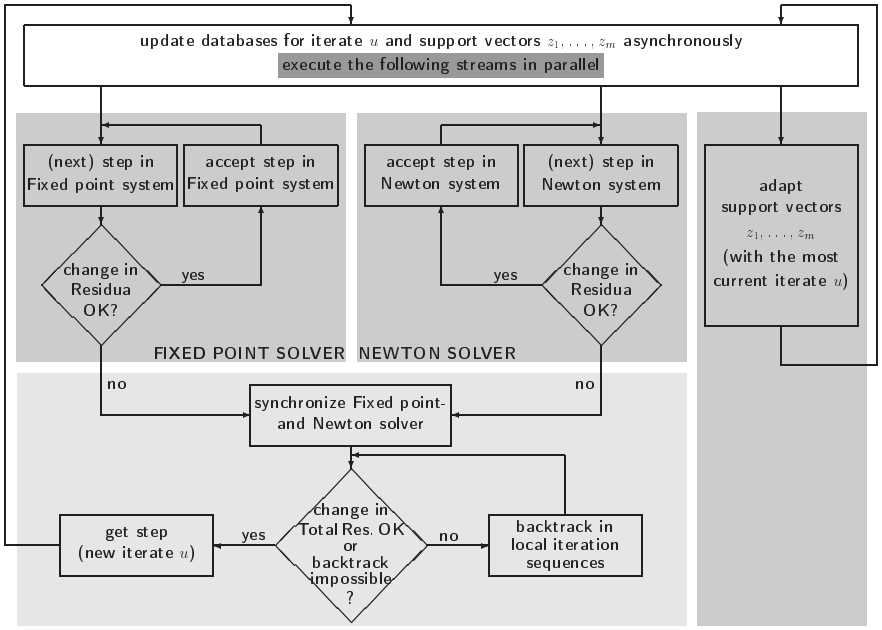

Fig. 2. Flowchart for parallel method (simplified)

the substeps and get a step for the full-space system. This step can be (and normally is) a combination of different numbers of steps in the subsolvers.

Next, we check if the total residuum decreases sufficiently for this full-space step. If it does, we accept the step. Otherwise, we backtrack using the stored local iteration history, i.e. we damp the step by reduction of the number of substeps in the two solvers which we combine for a full-space step. We carry out the test on the total residuum for these steps too. As worst (and rare) possibility, none of the full-space steps can be accepted. Then, we have to start iterating with the old iterate u again. But another process has been running simultaneously and improved the support vectors constantly. So, we have better decoupled subspaces for new iterations in the subsolvers.

Parallel to all the formerly described calculations we adjust the subspaces. The renewed supports $z_1, \ldots, z_m$ are calculated without intermediate communication with the subsolvers. We adapt them by a simultaneous power iteration which is parallelisable by itself, too. The only information needed for this block (the third one on the right in Fig. 2), is the most current iterate u. This block returns an improved subsystem splitting.

So, we automatically improve the subspace splitting if just a small or even no full-space step is accepted. Consequently, the more problems the iterations encounter, the better the supporting subspaces get: thus the algorithm automatically reacts to quickly changing unstable modes.

4 Example

To illustrate the algorithm we compute a stationary solution of the two-dimensional heat and mass transfer equations as described in [2] and [5]. Starting with $y = 1$ as a trivial mass distribution in the domain $\Omega = [-1, 1]^2$, we solve the problem for the heat distribution:

$$\Delta\theta = -\delta y \cdot \exp\left(\frac{\theta}{1 + \theta/\gamma}\right), \text{ on } \Omega = (-1, 1)^2$$

for $\delta = 1, \gamma = 20$ and with the boundary condition

$$\theta = 0, \text{ on } \partial\Omega.$$

The equation is discretized by the common 5-point difference scheme on a regular (65×65)–grid. The resulting nonlinear system of equations has the dimension 4225.

We start the iteration with a random vector of adequate size. The parallel computations have been simulated under MATLAB on a HP9000. The porting of the code onto a parallel machine is currently under progress.

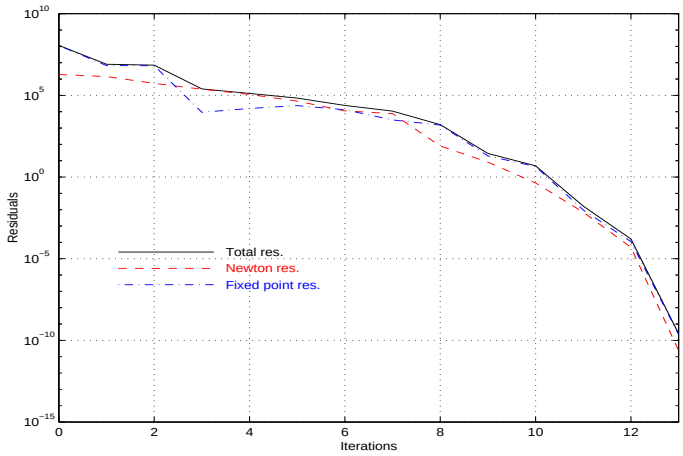

Fig. 3. Iteration history

The iteration history for the parallel program is shown in Fig. 3. Notice here that each step on the 'Iterations'–axis stands for one full-space step. Each full-space step may be a combination of several steps in the subsystems. The number of substeps per full-space step is shown in Table 1. Note that the progress of the iteration history is quite similar for many other examples (cf. [4]).

We begin iterating with a "not too good" starting vector u. Here, the Newton subsystem has the dimension 11, which is less than 1% of the size of

full-space step	1	2	3	4	5	6	7	8	9	10	11	12	13
Fixed point steps	1	0	1	0	0	1	1	0	2	1	7	5	16
Newton steps	2	2	1	3	1	1	1	4	2	1	1	1	1
Dimension of Newton system	11	11	9	9	5	4	3	3	2	2	1	1	1

Table 1. Number of substeps per full–space step

the fixed point system. This ratio of size is found for many applications. See [3], [6] for a discussion why.

As we start our iteration far away from the solution, we try to perform large steps in the subsystems. But we can not perform many large substeps independently, as the subspaces change remarkably for these steps due to the nonlinearity of the problem. The fixed point solver soon stops for synchronisation, because there is too much exchange between the residuals. The Newton solver can calculate more steps, because it performs well according to its local residual criterion. But it is also stopped after a few steps.

The solvers synchronize and a full space step is calculated. This step can consist of several Newton steps, but usually less than two fixed point steps, as we can see in Table 1. Possibly, it is best to accept a full space step consisting of Newton substeps only. Note that at this stage, a fair amount of computation must be done to guarantee the approximate invariance of the solvers. That is, we have to invest quite some work to correct the splitting of the subspaces by adaptation of the supports.

The computational effort changes as we approach the solution. There are less eigenvalues with modulus larger than one near to the solution point. Hence, the fixed point system increases and the Newton system decreases. The steplength decreases, too. Consequently, the subspaces are sufficiently decoupled for more steps in the two subsolvers. We can perform increasing numbers of substeps without synchronization. We exploit this by executing several fixed point steps per full-space step. Usually, we execute just one Newton step simultaneously, as the Newton solver converges quadratically and consequently rather fast close to the solution. We combine the iterates of the subsolvers as discussed to receive a full-space step.

In this phase of the algorithm most computations are done in the fixed point solver. The importance of the adaptation of the supports decreases substantially and the Newton solver is not time critical due to its small dimension.

The convergence speed of the full system is driven by the linearly convergent fixed point solver. We combine increasing numbers of fixed point steps for the full-space steps. Consequently, the iteration history shows increasing residual gain per full-space step. This results in an iteration history looking superlinearly convergent. But notice, that the solver is of course still linearly convergent, when measured per step in the fixed point system.

5 Outlook

We have been successful in computing a solution for the original heat and mass transfer problem ([2], [5]) with both species as unknowns, too. We switched iterating for the two species and found a solution for the full heat and mass transfer problem. But here, a future implementation for the non-symmetric case will be clearly of advantage.

Currently, we investigate an enhancement of the algorithm by load balancing. We shift the computational power for the processes by dynamic allocation of processors. Besides a parallel fixed point solver, we also need a parallel subspace iteration, which exploits the inherent parallelism and which utilizes the parallel fixed point solver for the calculation of new supports.

References

1. Aris, R. (1975) The Mathematical Theorie of Diffusion and Reaction in Permeable Catalysts (Volume 1). Clarendon Press, Oxford.
2. Hlavacek, V., Kubicek, M., Marek, M. (1969) Analysis of Nonstationary Heat and Mass Transfer in a Porous Catalyst Particle I. Journal of Catalysis, **15**, 17–30.
3. Jarausch, H., Mackens, W. (1987) Solving Large Nonlinear Systems of Equations by an Adaptive Condensation Process. Numerische Mathematik, **50**, 633–653.
4. Kleis, U. A Parallel Stabilization for Nonlinear Equation Solvers. Accepted for Publication by ZAMM.
5. Roose, D., Hlavacek, V. (1983) Numerical Computation of Hopf Bifurcation Points for Parabolic Diffusion-Reaction Differential Equations. SIAM J. Appl. Math., **43**(5), 1075–1085.
6. Shroff, G. M., Keller, H. B. (1993) Stabilization of Unstable Procedures: The Recursive Projection Method. SIAM J. Numer. Anal., **30**(4), 1099–1120.

Index