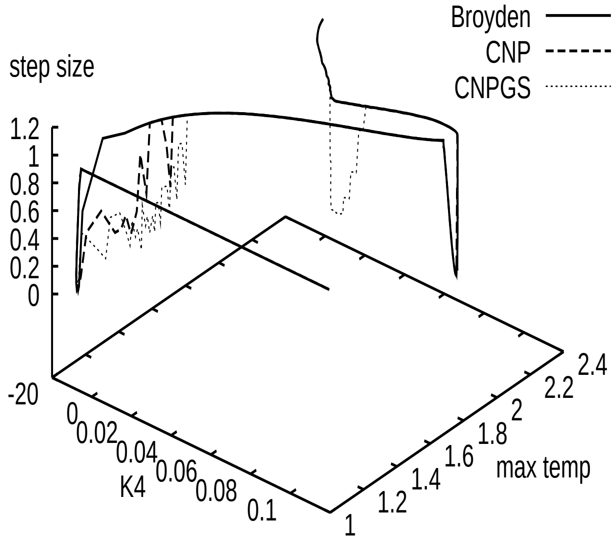

Universiteit Utrecht

Department
of Mathematics

A Broyden Rank $p + 1$ Update Continuation Method with Subspace Iteration

by

T.L. van Noorden, S.M. Verduyn Lunel, and A. Bliek

Preprint

nr. 1275

June, 2003

revised version

A BROYDEN RANK $P+1$ UPDATE CONTINUATION METHOD WITH SUBSPACE ITERATION*

T.L. VAN NOORDEN†, S.M. VERDUYN LUNEL‡, AND A. BLIEK§

Abstract. In this paper we present an efficient branch following procedure that cannot only be used to compute branches of periodic solutions of periodically forced dynamical systems but also to determine the stability of the periodic solutions. The procedure combines Broyden's method with a subspace iteration method to determine the dominant eigenvalues. The method has connections with the hybrid Newton-Picard methods developed by Lust et al. in [SIAM J. Sci. Comp. 19 (1998) pp. 1188–1209]. A convergence analysis of the procedure is presented. The method is applied to the computation of periodic states of a reverse flow reactor and its performance is compared with two variants of the hybrid Newton-Picard method.

Key words. Broyden's method, periodically forced PDE's, reverse flow reactor, continuation of periodic solutions

1. Introduction. This paper is devoted to the efficient computation of periodic solutions of periodically forced partial differential equations and the determination of their stability. The partial differential equations of interest model cyclically operated chemical systems. Examples of such dynamical systems are pressure swing separation [19], the reverse flow reactor [8] and the pressure swing reactor [9].

In order to investigate the behavior of periodic dynamical systems numerically, we first have to discretize the equations. For a shooting method, first the spatial variable of the equations is discretized. This results in a large system of N time periodic ordinary differential equations. This system can abstractly be written as

$$\dot{x}(t) = f(t, x(t), \lambda), \quad x(t) \in \mathbb{R}^N, \quad \lambda \in \mathbb{R}, \quad \text{with } f(t, \cdot, \cdot) = f(t+1, \cdot, \cdot). \tag{1.1}$$

The right hand side, denoted by f, of the ordinary differential equation (1.1) has the following additional structure. If a finite element or finite difference discretization is used to discretize the spatial variable, then the components of x can be ordered in such a way that the Jacobian of f has a band structure. For one dimensional problems the bandwidth will be very small. For two or three dimensional problems the bandwidth can become larger, but in any case the Jacobian of f is a sparse matrix. The assumption in equation (1.1) that the period length equals one is not a limitation. In fact the time variable t can always be rescaled in order to assume that the period of the periodic forcing in f is equal to one.

The map $F : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ that for a given λ assigns to the initial data at time zero, $x(0) = x_0$, the value of the solution after one period, i.e., the state $x(1)$, is called the Poincaré or period map of (1.1),

$$F(x_0, \lambda) = x(1, x_0), \tag{1.2}$$

---

*This is a revised and updated version of: *A Broyden rank p+1 update continuation method with subspace iteration*, T.L. van Noorden, S.M. Verduyn Lunel and A.Bliek, Technical Report MI 2001-30, Mathematical Institute, University of Leiden, 2001.

†Utrecht University, Department of Mathematics, P.O.box 80010, 3508 TA Utrecht, The Netherlands.

‡Universiteit Leiden, Department of Mathematics, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands. Corresponding author. email: verduyn@math.leidenuniv.nl

§Universiteit van Amsterdam, Department of Chemical Engineering, Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands.

where $x(t, x_0)$ denotes the solution of (1.1) with initial condition $x(0) = x_0$. So, evaluating the map F is equivalent to integrating system (1.1) over one time unit starting with a given initial condition x_0.

A 1-periodic state of the chemical system is a 1-periodic solution $x(t)$, i.e., a solution of (1.1) with $x(0) = x(1)$, and therefore the initial condition of a 1-periodic solution of (1.1) is a solution of the fixed point equation

$$x = F(x, \lambda). \tag{1.3}$$

In a similar way we can define 2-periodic, 3-periodic, etc., states of the chemical system.

We define the Jacobian of F at the point $\boldsymbol{x} = (x, \lambda)$ to be the square matrix $J_F(x, \lambda)$ given by $J_F(x, \lambda) := \frac{\partial F}{\partial x}(x, \lambda)$. The stability of a solution x^* of (1.3) is determined by the eigenvalues of the Jacobian $J_F(x^*, \lambda)$ at the fixed point x^*.

If we consider solutions $\boldsymbol{x} = (x, \lambda)$ of the equation $F(x, \lambda) = x$, it is known that these solutions form in general, at least locally, a curve in $\mathbb{R}^{N+1}$, see for example [6]. Pseudo arc-length continuation [10] allows to compute this curve also past turning points. Suppose we have a solution $\boldsymbol{x}_0 = (x_0, \lambda_0)$ of $F(x, \lambda) = x$, as well as the normalized direction vector of the solution curve at (x_0, λ_0), which we will denote by $\dot{\boldsymbol{x}}_0 = (\dot{x}_0, \dot{\lambda}_0)$. Pseudo arc-length continuation consists of solving the following equations for x_1 and λ_1

$$\begin{cases} F(x_1, \lambda_1) - x_1 = 0, \\ \dot{x}_0^T(x_1 - x_0) + \dot{\lambda}_0(\lambda_1 - \lambda_0) - s = 0, \end{cases} \tag{1.4}$$

where s is the step size and $(x_1 - x_0)^T$ is the transpose of $(x_1 - x_0)$. Depending on how the equations (1.4) are solved, it is sometimes expensive to compute the direction vector $\dot{\boldsymbol{x}}_0 = (\dot{x}_0, \dot{\lambda}_0)$. In these cases a fair approximation can be obtained from

$$\dot{\boldsymbol{x}}_0 \approx \frac{(\boldsymbol{x}_0 - \boldsymbol{x}_{-1})}{|(\boldsymbol{x}_0 - \boldsymbol{x}_{-1})|}, \tag{1.5}$$

where $\boldsymbol{x}_{-1}$ is a previously computed solution on the curve.

There exist various ways to obtain numerical solutions to the equations (1.3) or (1.4). When a periodic state $x(t, x^*)$ is stable, i.e., all the eigenvalues of the Jacobian $J_F(x^*, \lambda)$ are inside the open unit disc $D := \{z \in \mathbb{C} : |z| < 1\}$, then a solution to equation (1.3) can be obtained by integrating (1.1) on a long time interval, starting with an initial condition that is sufficiently close to the periodic solution. This approach is equivalent to a Picard iteration of the map F. This method is, in many cases, not feasible, since the convergence to a periodic state of a dynamic simulation may be very slow [19]. Furthermore, only *stable* periodic states can be obtained by this approach.

A Newton-type method, applied directly to the fixed point equation (1.3), would allow to compute both stable and unstable periodic states. However, such an approach involves the computation or approximation of the Jacobian $J_F(x, \lambda)$ at various points x, either by numerical differentiation or by solving the variational form of (1.1). Both approaches are very expensive when N is large [13, 3]. Note that, in contrast to the map f in the right hand side of (1.1), the Jacobian $J_F(x, \lambda)$ does generally not have a band structure.

In numerical literature, a number of hybrid Newton-Picard methods has been developed, see [13, 7, 16]. These methods use the fact that for many physically interesting (infinite dimensional) systems, the interesting dynamics actually occurs only in a low dimensional subspace. Numerical evidence that this is also a feature for many periodic chemical processes is presented in [20]. A vector basis of this low dimensional subspace is obtained from a subspace iteration algorithm. This approach also yields eigenvalues that determine the stability. Furthermore, the action of the Jacobian $J_F(x,\lambda)$ is only computed on the space spanned by this basis. It turns out that the theoretical convergence rate of the Newton-Picard method is linear.

In the chemical engineering literature other methods have been proposed to solve the equations (1.3) or (1.4) more efficiently. It has been shown that Broyden's method is very efficient in most cases [18, 8]. Broyden's method uses only one evaluation of F per iteration and converges q-superlinearly. However, the disadvantage of Broyden's method is that the approximations, that the method produces, to the Jacobian $J_F(x^*,\lambda)$ of at the periodic solution (x^*,λ) are not accurate enough for the determination of the stability of the periodic solution.

In this paper we discuss a Broyden rank $p+1$ update continuation method with subspace iteration (in short the BSI rank $p+1$ method). The method combines the efficiency of Broyden's method with the ability of the subspace iteration method to determine the stability of the computed solutions. The application of both the BSI rank $p+1$ method and the Newton-Picard method to an example system, shows that the BSI rank $p+1$ method is approximately twice as efficient as the Newton-Picard method, in terms of number of evaluations of F. The difference in efficiency lies mainly in the fact that the BSI method uses less evaluations of F per iteration than the Newton-Picard method. The example shows that the convergence rate and robustness of the two methods are comparable. We also consider the theoretical convergence properties of the Broyden rank $p+1$ update method, and show that an adapted version of the method is q-superlinearly convergent.

Since the value $F(x_0,\lambda)$ is obtained by integrating a large system of differential equations over a period of time with initial condition x_0, the evaluation of F is a computationally expensive task. This means that for any method used to find a fixed point of F, most of the computation time will be spent on the evaluation of F. Thus the method that needs the least number of evaluations of F will be the most efficient method. Therefore the comparison between the BSI rank $p+1$ method is made with respect to the number of evaluations of F needed to compute a branch of periodic solutions.

The organization of this paper is as follows. In Section 2 we introduce the BSI rank $p+1$ method and discuss the merits of the method in comparison with the Newton-Picard method. In Section 3 we discuss the implementation of the BSI rank $p+1$ method. In Section 4 we analyze the theoretical convergence properties of the BSI rank $p+1$ method. In Section 5 we present the model equations for the cooled reverse flow reactor and in Section 6 we compute a branch of periodic states using the BSI rank $p+1$ method, Broyden's method and the Newton-Picard method, and compare the efficiency of all methods.

2. BSI Rank $p+1$ Method. In the Introduction we saw that, for the computation of a branch of periodic solutions of (1.1), we have to solve repetitively systems of the form

$$\begin{pmatrix} F(x,\lambda)-x \\ w^T(x-y)+\kappa(\lambda-\mu)-s \end{pmatrix} = 0, \tag{2.1}$$

where $w,\ y \in \mathbb{R}^N$ and $\kappa,\ \mu,\ s \in \mathbb{R}$ are given. For convenience, we define the map $G : \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$ by

$$G(\boldsymbol{x}) = G(x, \lambda) := \begin{pmatrix} F(x,\lambda) - x \\ w^T(x-y) + \kappa(\lambda - \mu) - s \end{pmatrix}, \tag{2.2}$$

where $\boldsymbol{x}$ is the $(N+1)$-dimensional vector (x, λ).

An efficient way to compute solutions to the system $G(\boldsymbol{x}) = 0$ is to use Broyden's "good" method [1] (there also exist other variants of Broyden's method, e.g. see [5]). Broyden's method produces approximations to a zero of G using the following iteration scheme

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - M_i^{-1} G(\boldsymbol{x}_i)\ , \tag{2.3}$$

with $M_i \in \mathbb{R}^{(N+1)\times(N+1)}$ iterative approximations to $\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_i)$ defined by

$$M_{i+1} = M_i + \frac{(\boldsymbol{g}_i - M_i \boldsymbol{p}_i)\boldsymbol{p}_i^T}{\boldsymbol{p}_i^T \boldsymbol{p}_i}\ , \tag{2.4}$$

where $\boldsymbol{g}_i = G(\boldsymbol{x}_{i+1}) - G(\boldsymbol{x}_i)$ and $\boldsymbol{p}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i$. This iteration scheme for M_i is derived from the following constraints on the update M_{i+1} of M_i:

$$M_{i+1}\boldsymbol{p}_i = \boldsymbol{g}_i, \tag{2.5}$$

$$M_{i+1}\boldsymbol{y} = M_i \boldsymbol{y} \quad \text{for all } \boldsymbol{y} \perp \boldsymbol{p}_i. \tag{2.6}$$

Equation (2.5) can be viewed as a secant approximation of $\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_{i+1})$ in the direction $\boldsymbol{p}_i$. Note that the only information that is used in updating the approximation M_i at the new approximation $\boldsymbol{x}_{i+1}$ is the function value $G(\boldsymbol{x}_{i+1})$. The method thus uses only one evaluation of F in each iteration. However, there is no guarantee that, if the approximations $\boldsymbol{x}_i$ converge to a solution $\boldsymbol{x}^*$ of $G(\boldsymbol{x}) = 0$, the approximations M_i converge to $\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}^*)$. This means that the approximations M_i cannot be used to obtain good approximations of the eigenvalues of $J_F(\boldsymbol{x}^*)$, which are needed for the determination of the stability of the solution $\boldsymbol{x}^*$. In this section we discuss an approach to combine Broyden's method with a subspace iteration algorithm [15] so that also approximations to the largest eigenvalues of $J_F(\boldsymbol{x}^*)$ can be computed accurately.

For this approach we do not only need the initial approximations $\boldsymbol{x}_0$ and M_0, required for Broyden's method, but also initial approximations to the p largest eigenvalues of $J_F(\boldsymbol{x}_0)$ together with an initial approximation $V_0 \in \mathbb{R}^{N\times p}$ for the orthonormal basis of the subspace spanned by the p corresponding eigenvectors. In the first step of our new approach, we proceed as in Broyden's method, and compute the next approximation to the solution of (2.1) with the same formula as before

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - M_0^{-1} G(\boldsymbol{x}_0). \tag{2.7}$$

Now we would like to update M_0, V_0 and the approximations of the eigenvalues.

Let us first consider the updating of the eigenvalues and of V_0. We would like the updates of the eigenvalues to approximate the p largest eigenvalues of the Jacobian $J_F(\boldsymbol{x}_1)$ in the new approximation $\boldsymbol{x}_1$ and the update V_1 to approximate the basis of the subspace spanned by the corresponding eigenvectors. Both the updates of the eigenvalues and of V_0 can be obtained by performing one (or more) subspace iteration(s). The version of the subspace iteration algorithm that we use, is the version also used in [13]. The algorithm can be summarized by the following steps.

Subspace iteration with projection

1) Set $\widetilde{V}_0 = V_0$ *and set* $j = 0$
2) Compute $W_j = J_F(\boldsymbol{x}_1)\widetilde{V}_j$.
3) Compute $S_j = \widetilde{V}_j^T W_j$.
4) Compute the Schur vectors $Y_j = [y_1, ..., y_p]$ *of* S_j*, order them according to decreasing modulus of the corresponding eigenvalue.*
5) Compute $\widetilde{V}_{j+1} = W_j Y_j$.
6) Orthonormalize $\widetilde{V}_{j+1}$.
7) If $\widetilde{V}_{j+1}$ *is accurate enough, then stop and return* $\widetilde{V}_{j+1}$ *and* W_j*. Else set* $j = j+1$ *and go to step 2).*

In this procedure the matrices $\widetilde{V}_j$ will converge to a orthonormal basis of the subspace spanned by the eigenvectors and generalized eigenvectors associated with the p largest eigenvalues of $J_F(\boldsymbol{x}_1)$, ordered with respect to the modulus. The convergence properties of the subspace iteration algorithm are discussed in [15]. In each iteration of the BSI rank $p+1$ method, we only perform one iteration of the subspace iteration algorithm. This iteration provides us with new approximations to the p largest eigenvalues of $J_F(\boldsymbol{x}_1)$, with a new orthonormal basis $V_1 = \widetilde{V}_1$ and with the matrix

$$W_0 = J_F(\boldsymbol{x}_1)\widetilde{V}_0 = J_F(\boldsymbol{x}_1)V_0. \tag{2.8}$$

The matrix W_0 is very useful in updating M_0. If we define the matrices

$$\boldsymbol{V} := \begin{pmatrix} V_0 \\ 0 \end{pmatrix} \text{ and } \boldsymbol{Z} := \begin{pmatrix} W_0 - V_0 \\ w^T V_0 \end{pmatrix}, \tag{2.9}$$

then it is easily seen that

$$\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_1)\boldsymbol{V} = \boldsymbol{Z}. \tag{2.10}$$

So, we can use the equation (2.10), in the same way as the constraint (2.5) is used in Broyden's method, to update the matrix M_0. Therefore, analogously to the Broyden rank one update, we would now like to update M_0 in such a way that

$$M_1\boldsymbol{V} = \boldsymbol{Z}, \tag{2.11}$$
$$M_1(\boldsymbol{x}_1 - \boldsymbol{x}_0) = \big(G(\boldsymbol{x}_1) - G(\boldsymbol{x}_0)\big), \tag{2.12}$$
$$M_1\boldsymbol{q} = M_0\boldsymbol{q} \text{ for all } \boldsymbol{q} \text{ with } \boldsymbol{V}^T\boldsymbol{q} = 0 \text{ and } (\boldsymbol{x}_1 - \boldsymbol{x}_0)^T\boldsymbol{q} = 0. \tag{2.13}$$

As long as $(I - \boldsymbol{V}\boldsymbol{V}^T)(\boldsymbol{x}_1 - \boldsymbol{x}_0) \neq 0$, we have that the matrix M_1 characterized by the equations (2.11)-(2.13) is well defined and unique. The matrix M_1 can be computed as follows

$$M_1 = M_0 + \sum_{l=1}^{p}(\boldsymbol{z}_l - M_0\boldsymbol{v}_l)\boldsymbol{v}_l^T + (\boldsymbol{f} - M_0\boldsymbol{p})\boldsymbol{p}^T, \tag{2.14}$$

where we have used the following notation

$$\boldsymbol{p} = \frac{(I - \boldsymbol{V}\boldsymbol{V}^T)(\boldsymbol{x}_1 - \boldsymbol{x}_0)}{||(I - \boldsymbol{V}\boldsymbol{V}^T)(\boldsymbol{x}_1 - \boldsymbol{x}_0)||}, \tag{2.15}$$

$$\boldsymbol{f} = \frac{G(\boldsymbol{x}_1) - G(\boldsymbol{x}_0) - \boldsymbol{Z}\boldsymbol{V}^T(\boldsymbol{x}_1 - \boldsymbol{x}_0)}{||(I - \boldsymbol{V}\boldsymbol{V}^T)(\boldsymbol{x}_1 - \boldsymbol{x}_0)||}. \tag{2.16}$$

It is easily verified that the matrix M_1 computed in (2.14) satisfies the equations (2.11)-(2.13). We use the projected update $\boldsymbol{p}$ and "projected" derivative vector $\boldsymbol{f}$, instead of $\boldsymbol{x}_1 - \boldsymbol{x}_0$ and $G(\boldsymbol{x}_1) - G(\boldsymbol{x}_0)$, which are used in Broyden's method, in order to obtain the orthogonal basis $\boldsymbol{v}_1, ..., \boldsymbol{v}_p, \boldsymbol{p}$. It is this basis that is needed for the construction of M_1. Note that (2.11) and (2.12) are equivalent to (2.11) and $M_1\boldsymbol{p} = \boldsymbol{f}$.

At this point we have computed the new updates $\boldsymbol{x}_1$, V_1, M_1 and the new updates of the p largest eigenvalues. By following the above procedure starting with these updates, we can now obtain the next set of updates, and so on. This iteration of the above described procedure characterizes the BSI rank $p+1$ method. For convenience, we present three variants of this new procedure. The first one, which we call BSI1, can be summarized by the following steps.

BSI rank $p+1$ method (BSI1)

1) *Supply* $M_0 \in \mathbb{R}^{(N+1)\times(N+1)}$, *invertible;* $V_0 \in \mathbb{R}^{N\times p}$, *orthonormal;* $\boldsymbol{x}_0 = (x_0, \lambda_0) \in \mathbb{R}^{(N+1)}$, *and set* $i = 0$.
2) *Solve* $M_i\boldsymbol{s}_i = -G(\boldsymbol{x}_i)$, *(computation of* $G(\boldsymbol{x}_i)$ *costs one evaluation of* F*) and set* $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{s}_i$. *compute* $S := J_F(\boldsymbol{x}_{i+1})V_i$ *(costs p evaluations of* F*).*
3) *Set*
$$\boldsymbol{V} := \begin{pmatrix} V_i \\ 0 \end{pmatrix} \text{ and } \boldsymbol{Z} := \begin{pmatrix} S - V_i \\ w^T V_i \end{pmatrix}.$$
4) *Let* $\boldsymbol{v}_l$ *and* $\boldsymbol{z}_l$ *be the* l*th* ($l = 1, ..., p$) *column of respectively* $\boldsymbol{V}$ *and* $\boldsymbol{Z}$. *Set* $\boldsymbol{v}_{p+1} = \frac{(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i}{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}$, *and* $\boldsymbol{z}_{p+1} = \frac{G(\boldsymbol{x}_{i+1})-G(\boldsymbol{x}_i)-\boldsymbol{Z}\boldsymbol{V}^T\boldsymbol{s}_i}{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}$. *Compute*
$$M_{i+1} = M_i + \sum_{l=1}^{p+1}(\boldsymbol{z}_l - M_i\boldsymbol{v}_l)\boldsymbol{v}_l^T.$$
5) *Subspace iteration: Compute the ordered Schur-factorization* Y^TRY *of* VS. *Compute* $V = SY$. *Orthonormalize the columns of* V *and put the result in* V_{i+1}.
6) *Set* $i = i+1$, *go to 2).*

This algorithm needs $p+1$ evaluations of F per iteration. In Section 6 we will use the algorithm to compute a branch of periodic states of a reverse flow reactor.

Note that if for certain $i = i_0$, the update $\boldsymbol{s}_i$ belongs to the subspace spanned by the columns of $\boldsymbol{V}$, then the vectors $\boldsymbol{v}_{p+1}$ and $\boldsymbol{z}_{p+1}$ in step 4 of the algorithm above are not defined, and in that case we can only update M_i with a rank p matrix instead of a rank $p+1$ matrix. More generally, if the angle between a certain update $\boldsymbol{s}_i$ and subspace spanned by the columns of $\boldsymbol{V}$ becomes small, we might encounter problems.

The sine of the angle between the update $\boldsymbol{s}_i$ and the subspace spanned by the columns of $\boldsymbol{V}$ is given by the quotient $||(I - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||/||\boldsymbol{s}_i||$. It turns out that this quotient is important in the convergence analysis of the BSI rank $p+1$ method. To assure q-superlinear convergence of the method, this quotient has to be bounded away from zero. In the BSI rank $p+1$ method as defined in algorithm BSI1, we cannot assure a priori that this quotient stays bounded away from zero. We can, however, adapt the BSI1 algorithm so that we can avoid problems to occur. We next present two variants of the BSI rank $p+1$ method, called BSI2 and BSI3, for which we can control the situation of a small angle.

In the first approach, BSI2, we use, in case the quotient $||(I - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||/||\boldsymbol{s}_i||$ becomes too small, an additional matrix-vector product in order to approximate the derivative of G in the direction $||(I - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||$, instead of using the secant update. In the second approach, BSI3, we use, when the angle becomes to small, only a rank $p-1$ update on a $(p-1)$-dimensional subspace of the p dimensional subspace $\boldsymbol{V}$, i.e., the subspace spanned by the columns of $\boldsymbol{V}$. This $(p-1)$-dimensional subspace is chosen orthogonal to the update vector $\boldsymbol{s}_i$.

In both algorithms we introduce a parameter κ, $0 < \kappa \leq 1$, in order to control the influence of the sine of the angle between the update $\boldsymbol{s}_i$ and the subspace $\boldsymbol{V}$. In the limit $\kappa = 0$, the BSI2 and BSI3 algorithm reduce to the BSI1 method. Note that we only have to adapt step 4 in the BSI1 algorithm. We now summarize the steps in the BSI2 and BSI3 algorithms.

BSI rank $p+1$ method (BSI2)

1) *Supply $M_0 \in \mathbb{R}^{(N+1)\times(N+1)}$, invertible; $V_0 \in \mathbb{R}^{N\times p}$, orthonormal; $\boldsymbol{x}_0 = (x_0, \lambda_0) \in \mathbb{R}^{(N+1)}$, and set $i = 0$. Fix κ, $0 < \kappa \leq 1$.*
2) *Solve $M_i\boldsymbol{s}_i = -G(\boldsymbol{x}_i)$, (computation of $G(\boldsymbol{x}_i)$ costs one evaluation of F) and set $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{s}_i$. compute $S := J_F(\boldsymbol{x}_{i+1})V_i$ (costs p evaluations of F).*
3) *Set*
$$\boldsymbol{V} := \begin{pmatrix} V_i \\ 0 \end{pmatrix} \text{ and } \boldsymbol{Z} := \begin{pmatrix} S - V_i \\ w^T V_i \end{pmatrix}.$$
4) *Let $\boldsymbol{v}_l$ and $\boldsymbol{z}_l$ be the lth ($l = 1, ..., p$) column of respectively $\boldsymbol{V}$ and $\boldsymbol{Z}$. Set $\boldsymbol{v}_{p+1} = \frac{(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i}{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}$. Set $c = \frac{||\boldsymbol{s}_i||}{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}$. If $1/c < \kappa$ then compute $\boldsymbol{z}_{p+1} = \frac{\partial G}{\partial x}(\boldsymbol{x}_{i+1})\boldsymbol{v}_{p+1}$, else set $\boldsymbol{z}_{p+1} = \frac{G(\boldsymbol{x}_{i+1})-G(\boldsymbol{x}_i)-\boldsymbol{Z}\boldsymbol{V}^T\boldsymbol{s}_i}{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}$. Compute*
$$M_{i+1} = M_i + \sum_{l=1}^{p+1}(\boldsymbol{z}_l - M_i\boldsymbol{v}_l)\boldsymbol{v}_l^T.$$
5) *Subspace iteration: Compute the ordered Schur-factorization Y^TRY of VS. Compute $V = SY$. Orthonormalize the columns of V and put the result in V_{i+1}.*
6) *Set $i = i+1$, go to 2).*

Thus if the sine of the angle between the update $\boldsymbol{s}_i$ and the subspace $\boldsymbol{V}$ becomes smaller than κ, we replace the "secant " update of M_i in the direction perpendicular to $\boldsymbol{V}$ by a directional derivative update. In this situation we need one extra directional derivative evaluation. The second alternative version of the BSI rank $p+1$ method is given by

BSI rank $p+1$ method (BSI3)

1) *Supply $M_0 \in \mathbb{R}^{(N+1)\times(N+1)}$, invertible; $V_0 \in \mathbb{R}^{N\times p}$, orthonormal; $\boldsymbol{x}_0 = (x_0, \lambda_0) \in \mathbb{R}^{(N+1)}$, and set $i = 0$. Fix κ, $0 < \kappa \leq 1$.*
2) *Solve $M_i\boldsymbol{s}_i = -G(\boldsymbol{x}_i)$, (computation of $G(\boldsymbol{x}_i)$ costs one evaluation of F) and set $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{s}_i$. compute $S := J_F(\boldsymbol{x}_{i+1})V_i$ (costs p evaluations of F).*
3) *Set*
$$\boldsymbol{V} := \begin{pmatrix} V_i \\ 0 \end{pmatrix} \text{ and } \boldsymbol{Z} := \begin{pmatrix} S - V_i \\ w^T V_i \end{pmatrix}.$$

4) *Let* $\boldsymbol{v}_l$ *and* $\boldsymbol{z}_l$ *be the* l*th* ($l = 1, ..., p$) *column of respectively* $\boldsymbol{V}$ *and* $\boldsymbol{Z}$. *Set* $\boldsymbol{v}_{p+1} = \frac{(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i}{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}$. *Set* $c = \frac{||\boldsymbol{s}_i||}{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}$. *If* $1/c \geq \kappa$ *then set* $\boldsymbol{z}_{p+1} = \frac{G(\boldsymbol{x}_{i+1})-G(\boldsymbol{x}_i)-\boldsymbol{Z}\boldsymbol{V}^T\boldsymbol{s}_i}{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}$ *and compute*

$$M_{i+1} = M_i + \sum_{l=1}^{p+1} (\boldsymbol{z}_l - M_i\boldsymbol{v}_l)\boldsymbol{v}_l^T.$$

If $1/c < \kappa$ *then set* $v_1 = \boldsymbol{V}^T\boldsymbol{s}_i/||\boldsymbol{V}^T\boldsymbol{s}_i||$ *and compute* $p-1$ *vectors* v_j ($j = 2, ..., p$), *such that* $(v_1, ..., v_p)$ *forms an orthonormal basis for* $\mathbb{R}^p$, *now compute* $\boldsymbol{v}_l = \boldsymbol{V}v_l$ *for* $l = 2, ..., p$ *and compute* $\boldsymbol{z}_l = \boldsymbol{Z}v_l$ *for* $l = 2, ...p$. *Set* $\boldsymbol{v}_1 = \frac{\boldsymbol{s}_i}{||\boldsymbol{s}_i||}$ *and* $\boldsymbol{z}_1 = \frac{G(\boldsymbol{x}_{i+1})-G(\boldsymbol{x}_i)}{||\boldsymbol{s}_i||}$, *so that we can compute*

$$M_{i+1} = M_i + \sum_{l=1}^{p} (\boldsymbol{z}_l - M_i\boldsymbol{v}_l)\boldsymbol{v}_l^T.$$

5) *Subspace iteration: Compute the ordered Schur-factorization* Y^TRY *of* VS. *Compute* $V = SY$. *Orthonormalize the columns of* V *and put the result in* V_{i+1}.
6) *Set* $i = i + 1$, *go to 2).*

In this case, if the sine of the angle between the update vector $\boldsymbol{s}_i$ and the subspace $\boldsymbol{V}$ becomes smaller than κ, we update M_i with a rank p matrix, instead of the rank $p+1$ matrix in the original method.

Note that if, in the BSI1 method, the angle between the update vector $\boldsymbol{s}_i$ and the subspace $\boldsymbol{V}$ stays bounded away from zero, then we can always set a value for κ such that all three the variants, BSI1, BSI2 and BSI3, perform exactly the same steps. In Section 6, we present results for each of the algorithms for various values of κ.

In order to give an idea of the efficiency of the BSI rank $p+1$ method, we compare its performance with the performance of two variants of a Newton-Picard method, the CNP and CNPGS method, introduced in [13]. The CNP method needs $p + 2l + 4$ evaluations of F per iteration, where l is a parameter of the method that specifies the number of Picard iterations used in one full iteration of the Newton-Picard method. The CNPGS method needs $p + l + 2$ evaluations of F per iteration and so is less expensive than CNP. In [13] it is reported that CNP and CNPGS have the same asymptotic convergence rate, but that CNP happens to be more robust, i.e., has a larger domain of attraction, and thus allows larger step sizes in the continuation algorithm.

In [13], the subspace iteration is performed on $p+p_e$ (with $p_e = 2$, 3 or 4) vectors in order to accelerate the convergence of the subspace iteration procedure. In [13], also the number p is varied during the computations in order to ensure that the subspace iteration is only performed on the subspace spanned by eigenvectors of which the corresponding eigenvalues are outside the region $D_\rho := \{z \in \mathbb{C} : |z| < \rho\}$ (usually ρ is taken to be 0.5). Here, in out setting, for both the Newton-Picard methods and the BSI method, we only perform the subspace iteration on p vectors, and keep this number fixed during the computations. Another difference between our approach and that of [13, 12] is that we do not use locking and deflation in the subspace iteration. Our approach with the dimension of $\boldsymbol{V}$ kept fixed, works very well for the cooled reverse flow reactor discussed in Section 5 (see also [20]). However, for arbitrary dynamical systems, a procedure for varying p as used in [16, 7] or in [13] is necessary.

In [13] it is reported that the Newton-Picard method converges linearly. Broyden's method has theoretically better convergence properties. In [2] it is shown that Broyden's method converges locally at least q-superlinearly, and in [5] it is shown that the method enjoys local $2N$-step, q-quadratic convergence. In Section 4 we will show the BSI2 and BSI3 variants of the BSI rank $p+1$ method also converge q-superlinearly.

We also compare the BSI method with Broyden's method followed by subspace iteration.

3. Implementation of the BSI rank $p+1$ method. In this section we consider an efficient implementation of the BSI rank $p+1$ method. In the previous section, we stated the BSI method in terms approximations M_i of the Jacobian of G. If we store the matrices M_i, then we need to solve the linear system $M_i x = -G(\boldsymbol{x}_i)$ in each iteration of the BSI method. For the implementation of the BSI method it is therefore more convenient to store the inverses of the matrices M_i. Just as for Broyden's method we can easily update the inverse of M_i. In this way we do not need to solve a linear system in each iteration of the BSI method. The BSI1 method in terms of the matrices H_i that are approximations of the inverse of the Jacobian of G, can be formulated as follows.

BSI rank $p+1$ method (BSI1 reformulated)

1) Supply $H_0 \in \mathbb{R}^{(N+1)\times(N+1)}$, invertible; $V_0 \in \mathbb{R}^{N\times p}$, orthonormal; $\boldsymbol{x}_0 = (x_0, \lambda_0) \in \mathbb{R}^{(N+1)}$, and set $i = 0$.

2) Compute

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + H_i G(\boldsymbol{x}_i).$$

(costs one evaluation of F)

3) Subspace iteration: compute $S := \frac{\partial F}{\partial x}(\boldsymbol{x}_{i+1})V_i$ (costs p evaluations of F). Compute the ordered Schur-factorization $Y^T RY$ of VS. Compute $V = SY$. Orthonormalize the columns of V and put the result in V_{i+1}.

4) Compute

$$H_{i+1} = \begin{cases} B_p - \dfrac{(\boldsymbol{p} + B_p \boldsymbol{f})\boldsymbol{p}^T B_p}{\boldsymbol{p}^T B_p \boldsymbol{f}} & \text{if } \boldsymbol{p}^T B_p \boldsymbol{f} \neq 0 \\ & \text{and } (I - \boldsymbol{Z}\boldsymbol{Z}^T)\big(G(\boldsymbol{x}_{i+1}) - G(\boldsymbol{x}_i)\big) \neq 0, \\ B_p & \text{otherwise,} \end{cases}$$

where B_p is defined recursively by $B_0 = H_i$ and

$$B_{l+1} = \begin{cases} B_l - \dfrac{(\boldsymbol{v}_l + B_l \boldsymbol{z}_l)\boldsymbol{v}_l^T B_l}{\boldsymbol{v}_l^T B_l \boldsymbol{z}_l} & \text{if } \boldsymbol{v}_l^T B_l \boldsymbol{z}_l \neq 0, \\ B_l & \text{if } \boldsymbol{v}_l^T B_l \boldsymbol{z}_l = 0, \end{cases}$$

with $\boldsymbol{v}_l$ and $\boldsymbol{z}_l$ the lth column of respectively

$$\boldsymbol{V} := \begin{pmatrix} V_i \\ 0 \end{pmatrix} \text{ and } \boldsymbol{Z} := \begin{pmatrix} \big(\frac{\partial F}{\partial x}(\boldsymbol{x}_{i+1}) - I\big)V_i \\ w^T V_i \end{pmatrix}.$$

and where

$$\begin{aligned} \boldsymbol{p} &= (I - \boldsymbol{V}\boldsymbol{V}^T)(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i), \\ \boldsymbol{f} &= G(\boldsymbol{x}_{i+1}) - G(\boldsymbol{x}_i) - \boldsymbol{Z}\boldsymbol{V}^T(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i). \end{aligned}$$

Set $i = i + 1$, go to 2).

This algorithm is derived by using the Sherwood-Morrison formula recursively for the $p+1$ rank one updates of the matrices M_i. It is easily checked that the inverse of matrix H_{i+1} computed in step 4 of the above algorithm satisfies (2.11)-(2.13). This shows that the two formulations of the BSI1 algorithm are indeed mathematically equivalent. Similarly, the BSI2 and BSI3 algorithms can also be stated in terms of approximations to the inverse of the Jacobian of G.

An important issue for a numerical method is the memory usage. The BSI rank $p+1$ method as discussed above needs to store the $N \times N$ matrix H_i. For the test problem presented in the paper, the N is not so large that the storage of H_i becomes a problem. For two dimensional or three dimensional model problems, however, the dimension of the discretized system N may become so large that it is impossible to store a full $N \times N$ matrix. As an alternative to storing the full matrix H_i, one can start with $H_0 = I$, and then storing each rank $p+1$ update in the BSI rank $p+1$ method using $2(p+1)$ vectors. This approach, however, still leads to memory problems when the BSI rank $p+1$ method needs many iterations to converge.

To overcome this problem for Broyden's method, there exist variants of the method with limited memory usage, that in certain cases behave just like Broyden's method [14]. The idea is to approximate H_i, when the number of updates becomes too large, with a lower rank matrix, so that the required storage remains limited. Such an approach might also be possible for the BSI rank $p+1$ method, but we do not pursue this matter further here. See [17] for a limited memory Broyden method that is also applied to computing periodic states of high-dimensional periodically forced dynamical systems.

The largest matrix that the Newton-Picard method requires to be stored, is the $N \times p$ matrix V_i that is an approximation a basis of the subspace spanned by the eigenvectors corresponding to the largest p eigenvalues. Here the value of p is essentially independent of the discretization, so that the storage used by the Newton-Picard method depends only linearly on N.

4. Convergence Analysis of the BSI Rank $p+1$ Method. For Broyden's method, it is known that it converges q-superlinearly, see for example [4]. In this section we investigate the question whether this property also holds for the BSI rank $p+1$ method. We will see that for the BSI1 method presented in Section 2, this question cannot easily be answered. The two alternative variants BSI2 and BSI3, however, do enjoy q-superlinear convergence. Our numerical experiments in Section 6 will show that the original BSI1 method behaves very much the same as the q-superlinearly convergent alternative variants.

For the convergence analysis of the BSI method, we follow the lines of the proof of the q-superlinear convergence of Broyden's method as given in [4]. We assume that $G : \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$ is continuously differentiable in an open convex set $D \subset \mathbb{R}^{N+1}$. We also assume that there exists $\boldsymbol{x}_* \in \mathbb{R}^{N+1}$ and r, $\beta > 0$, such that

$$B_r(\boldsymbol{x}_*) := \{\tilde{\boldsymbol{x}} \in \mathbb{R}^{N+1} : ||\tilde{\boldsymbol{x}} - \boldsymbol{x}_*|| < r\} \subset D,$$

$G(\boldsymbol{x}_*) = 0$, $\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_*)^{-1}$ exists with $||\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_*)^{-1}|| \leq \beta$, and $\frac{\partial G}{\partial \boldsymbol{x}}$ Lipschitz continuous on $B_r(\boldsymbol{x}_*)$ with Lipschitz constant γ.

The first step is to prove that the generated sequence of matrices M_i is of bounded deterioration. This means that the approximation of the Jacobian can get worse, but slowly enough to ensure convergence to a zero of G. This is the contents of the following lemma.

LEMMA 4.1. *Let $D \subseteq \mathbb{R}^{N+1}$ be an open convex set containing $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$, with $\boldsymbol{x}_{i+1} \neq \boldsymbol{x}_*$, where $\boldsymbol{x}_*$ is a solution of $G(\boldsymbol{x}) = 0$. Let G be Lipschitz continuous on D, let $\boldsymbol{x}_* \in D$. There exists a constant C such that*

$$||M_{i+1} - J_*|| \leq ||M_i - J_*|| + C(||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*|| + ||\boldsymbol{x}_i - \boldsymbol{x}_*||),$$

where J_* denotes $\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_*)$, and where the matrices M_i are generated by the BSI2 or BSI3 algorithm.

Proof. From the description of the algorithms for BSI2 and BSI3, we have

$$\begin{aligned} ||M_{i+1} - J_*|| &= ||M_i - J_* + \sum_{l=1}^{p+1} (\boldsymbol{z}_l - M_i \boldsymbol{v}_l)\boldsymbol{v}_l^T|| \\ &= ||M_i - J_* + \sum_{l=1}^{p+1} (J_* \boldsymbol{v}_l - M_i \boldsymbol{v}_l)\boldsymbol{v}_l^T + \sum_{l=1}^{p+1} (\boldsymbol{z}_l - J_* \boldsymbol{v}_l)\boldsymbol{v}_l^T|| \\ &= ||(M_i - J_*)(I - \sum_{l=1}^{p+1} \boldsymbol{v}_l \boldsymbol{v}_l^T) + \sum_{l=1}^{p+1} (\boldsymbol{z}_l - J_* \boldsymbol{v}_l)\boldsymbol{v}_l^T|| \\ &\leq ||(M_i - J_*)|| + ||\sum_{l=1}^{p+1} (\boldsymbol{z}_l - J_* \boldsymbol{v}_l)\boldsymbol{v}_l^T||. \end{aligned} \tag{4.1}$$

Here we have used that, since all the vectors $\boldsymbol{v}_l$ are orthogonal and have norm equal to one, the matrix $(I - \sum_{l=1}^{p+1} \boldsymbol{v}_l \boldsymbol{v}_l^T)$ is an Euclidean projection matrix and thus has norm one as well. The next step is to investigate the term $\sum_{l=1}^{p+1} (\boldsymbol{z}_l - J_* \boldsymbol{v}_l)\boldsymbol{v}_l^T$. First we split this term into two parts

$$\sum_{l=1}^{p+1} (\boldsymbol{z}_l - J_* \boldsymbol{v}_l)\boldsymbol{v}_l^T = \sum_{l=1}^{p} (\boldsymbol{z}_l - J_* \boldsymbol{v}_l)\boldsymbol{v}_l^T + (\boldsymbol{z}_{p+1} - J_* \boldsymbol{v}_{p+1})\boldsymbol{v}_{p+1}^T.$$

For the first part, we have

$$\begin{aligned} ||\sum_{l=1}^{p} (\boldsymbol{z}_l - J_* \boldsymbol{v}_l)\boldsymbol{v}_l^T|| &= ||\sum_{l=1}^{p} (\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_{i+1})\boldsymbol{v}_l - J_* \boldsymbol{v}_l)\boldsymbol{v}_l^T|| \\ &= ||(\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_{i+1}) - J_*) \sum_{l=1}^{p} \boldsymbol{v}_l \boldsymbol{v}_l^T|| \\ &\leq ||\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_{i+1}) - J_*|| \leq \gamma ||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*||. \end{aligned} \tag{4.2}$$

For the second part, we write

$$\begin{aligned} ||(\boldsymbol{z}_{p+1} - J_* \boldsymbol{v}_{p+1})|| &= \frac{||G(\boldsymbol{x}_{i+1}) - G(\boldsymbol{x}_i) - \boldsymbol{Z}\boldsymbol{V}^T \boldsymbol{s}_i - J_*(I - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}{||(I - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||} \\ &= \frac{||G(\boldsymbol{x}_{i+1}) - G(\boldsymbol{x}_i) - J_* \boldsymbol{s}_i - \boldsymbol{Z}\boldsymbol{V}^T \boldsymbol{s}_i - J_* \boldsymbol{V}\boldsymbol{V}^T \boldsymbol{s}_i||}{||(I - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||} \\ &\leq \frac{||G(\boldsymbol{x}_{i+1}) - G(\boldsymbol{x}_i) - J_* \boldsymbol{s}_i||}{||(I - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||} + \frac{||\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_{i+1})\boldsymbol{V}\boldsymbol{V}^T \boldsymbol{s}_i - J_* \boldsymbol{V}\boldsymbol{V}^T \boldsymbol{s}_i||}{||(I - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||} \\ &\leq \frac{\gamma(||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*|| + ||\boldsymbol{x}_i - \boldsymbol{x}_*||)||\boldsymbol{s}_i||}{2||(I - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||} + \frac{\gamma||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*||||\boldsymbol{s}_i||}{||(I - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||} \end{aligned}$$

If we have an upper bound for the quotient $c = \frac{||\boldsymbol{s}_i||}{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}$, we can control this part as well. For the BSI1 algorithm, we cannot assure that this quotient remains bounded and therefore we cannot prove q-superlinear convergence for the the BSI1 algorithm directly. For both the BSI2 and BSI3 algorithm, we observe that as long as

$$1/c = \frac{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}{||\boldsymbol{s}_i||} \geq \kappa,$$

we have the following estimate

$$||(\boldsymbol{z}_{p+1} - J_*\boldsymbol{v}_{p+1})|| \leq \frac{\gamma}{2\kappa}(||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*|| + ||\boldsymbol{x}_i - \boldsymbol{x}_*||) + \frac{\gamma}{\kappa}||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*|| \tag{4.3}$$

If now $1/c = \frac{||(I-\boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{s}_i||}{||\boldsymbol{s}_i||} < \kappa$, then for the BSI2 algorithm we have

$$||(\boldsymbol{z}_{p+1} - J_*\boldsymbol{v}_{p+1})|| = ||\frac{\partial G}{\partial x}(\boldsymbol{x}_{i+1})\boldsymbol{v}_{p+1} - J_*\boldsymbol{v}_{p+1}|| \leq \gamma||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*||, \tag{4.4}$$

where we have used the Lipschitz continuity of $\frac{\partial G}{\partial \boldsymbol{x}}$. Using the inequalities (4.1), (4.2), (4.3) and (4.4), we obtain the following estimate

$$\begin{aligned}
||M_{i+1} - J_*|| &\leq ||M_i - J_*|| + \gamma||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*|| + \frac{\gamma}{2\kappa}(||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*|| + ||\boldsymbol{x}_i - \boldsymbol{x}_*||) \\
&\quad + \frac{\gamma}{\kappa}||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*|| \\
&\leq ||M_i - J_*|| + \gamma(\frac{3}{2\kappa} + 1)(||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*|| + ||\boldsymbol{x}_i - \boldsymbol{x}_*||) \; . \qquad (4.5)
\end{aligned}$$

For the BSI3 algorithm we have instead of the inequality (4.4), the inequality

$$\begin{aligned}
||(\boldsymbol{z}_{p+1} - J_*\boldsymbol{v}_{p+1})|| &= \frac{||G(\boldsymbol{x}_{i+1}) - G(\boldsymbol{x}_i) - J_*\boldsymbol{s}_i||}{||\boldsymbol{s}_i||} \\
&\leq \frac{\gamma(||\boldsymbol{x}_{i+1} - \boldsymbol{x}_*|| + ||\boldsymbol{x}_i - \boldsymbol{x}_*||)||\boldsymbol{s}_i||}{2||\boldsymbol{s}_i||} \; , \qquad (4.6)
\end{aligned}$$

where we have used again the Lipschitz continuity of $\frac{\partial G}{\partial \boldsymbol{x}}$. Now using the inequalities (4.1), (4.2), (4.3) and (4.6), we also arrive for the BSI3 variant of the BSI rank $p+1$ method at the same estimate (4.5). □

Now we are ready to prove the following theorem.

THEOREM 4.2. *Let $G : \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$ be continuously differentiable in an open convex set $D \subset \mathbb{R}^{N+1}$. Assume that there exists $\boldsymbol{x}_* \in \mathbb{R}^{N+1}$ and r, $\beta > 0$, such that $B_r(\boldsymbol{x}_*) \subset D$, $G(\boldsymbol{x}_*) = 0$, $\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_*)^{-1}$ exists with $||\frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_*)^{-1}|| \leq \beta$, and $\frac{\partial G}{\partial \boldsymbol{x}}$ Lipschitz continuous on $B_r(\boldsymbol{x}_*)$ with Lipschitz constant γ. Then there exist positive constants ϵ, δ such that if $||\boldsymbol{x}_0 - \boldsymbol{x}_*||_2 \leq \epsilon$ and $||M_0 - \frac{\partial G}{\partial \boldsymbol{x}}(\boldsymbol{x}_*)|| \leq \delta$, then the sequence $\{\boldsymbol{x}_i\}$ generated by BSI2 or BSI3 algorithm is well defined and converges q-superlinearly to $\boldsymbol{x}_*$.*

Proof. Lemma 4.1 allows us to invoke (part of) Theorem 8.2.2 in [4] to show that there exist positive constants ϵ, δ such that if $||x_0 - x_*||_2 \leq \epsilon$ and $||M_0 - J(x_*)|| \leq \delta$, then the sequence $\{x_i\}$ generated by the BSI2 or the BSI3 algorithm is well defined and converges at least q-linearly to x_*.

For the proof of the q-superlinear convergence, we can follow almost verbatim the appropriate part of the proof of Theorem 8.2.2 in [4] for the q-superlinear convergence of Broyden's method. However, there is one minor modification that we will explain.

Define $E_i = M_i - J_*$. It is not hard to see that the inequality (4.1) also holds for the Frobenius norm

$$||E_{i+1}||_F \le ||E_i(I - \sum_{l=1}^{p+1} \boldsymbol{v}_l \boldsymbol{v}_l^T)||_F + ||\sum_{l=1}^{p+1}(\boldsymbol{z}_l - J_*\boldsymbol{v}_l)\boldsymbol{v}_l^T||_F.$$

Because $\boldsymbol{s}_i$ is in the span of $\boldsymbol{v}_l$, $l = 1, ..., p+1$, and since we use the Frobenius norm, we also obtain the inequality

$$||E_{i+1}||_F \le ||E_i(I - \frac{\boldsymbol{s}_i \boldsymbol{s}_i^T}{\boldsymbol{s}_i^T \boldsymbol{s}_i})||_F + ||\sum_{l=1}^{p+1}(\boldsymbol{z}_l - J_*\boldsymbol{v}_l)\boldsymbol{v}_l^T||_F. \tag{4.7}$$

Similarly as in the proof of Lemma 4.1, we obtain the estimate

$$||\sum_{l=1}^{p+1}(\boldsymbol{z}_l - J_*\boldsymbol{v}_l)\boldsymbol{v}_l^T||_F \le C(||e_{i+1}|| + ||e_i||), \tag{4.8}$$

where e_i denotes $||\boldsymbol{x}_i - \boldsymbol{x}_*||$. Hence, using (4.7) and (4.8),

$$||E_{i+1}||_F \le ||E_i(I - \frac{\boldsymbol{s}_i \boldsymbol{s}_i^T}{\boldsymbol{s}_i^T \boldsymbol{s}_i})||_F + C(||e_{i+1}|| + ||e_i||). \tag{4.9}$$

This inequality also holds for Broyden's method and is used in the proof for the q-superlinear convergence of the method. Using (4.9) in the appropriate place in the proof of Theorem 8.2.2 in [4], the proof also applies to the BSI2 and BSI3 variants of the BSI rank $p+1$ method. $\square$

5. The Cooled Reverse Flow Reactor. A Reverse Flow Reactor (RFR) is a packed bed reactor in which the flow direction is periodically reversed to trap a hot zone within the reactor. In this way exothermic reactions can be operated without preheating the feed stream. We consider a cooled RFR in which a single irreversible, exothermic, first order reaction occurs. We describe the RFR by a one-dimensional, pseudo-homogeneous model that accounts for axial heat and mass dispersion and external mass transfer resistance between the fluid and the catalyst. The basic model assumption is that all the physical properties are independent of the temperature and the concentration. The model we will discuss next, is taken from [8].

The variables in the model are the dimensionless temperature $\theta(t,x) : [0,\infty) \times [0,1] \to [0,\infty)$ and the conversion $\chi(t,x) : [0,\infty) \times [0,1] \to [0,1]$. The dimensionless parameters that appear in the model equations are denoted by K_j for $j = 1, .., 7$, and by $g(\theta(t,x))$. The model specific values are given in Table 5.1. The parameter K_4 will be used as the bifurcation parameter. For the flow from left to right the dimensionless energy and species balances read

$$\theta_t = K_1\theta_{xx} - K_2\theta_x + K_3 g(\theta)(1-\chi) + K_4(1-\theta), \tag{5.1}$$

$$\chi_t = K_5\chi_{xx} - K_6\chi_x + K_7 g(\theta)(1-\chi), \tag{5.2}$$

with boundary conditions

$$\begin{array}{cc} K_1\theta_x(t,0) = K_2(\theta(t,0)-1), & K_5\chi_x(t,0) = K_6\chi(t,0), \\ \theta_x(t,1) = 0, & \chi_x(t,1) = 0. \end{array} \tag{5.3}$$

TABLE 5.1
The dimensionless parameter values for the reverse flow reactor

parameter	value
K_1	$6.9393{\cdot}10^{-4}$
K_2	0.1749
K_3	$1.5577{\cdot}10^{-6}$
K_4	variable
K_5	$2.4038{\cdot}10^{-3}$
K_6	174.06
K_7	0.01
$g(\theta)$	$\dfrac{1.6656\cdot 10^{-5}e^{25.785(\theta-1)/\theta}}{1.6656\cdot 10^{-5}+e^{-25.785/\theta}}$

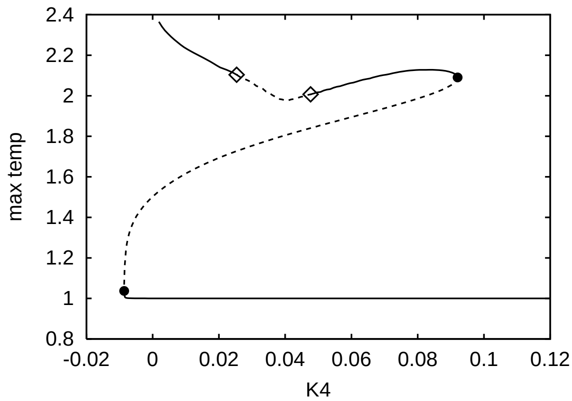

FIG. 6.1. *Branch of periodic solutions for the reverse flow reactor. Along the x-axis the parameter K_4 is plotted and along the y-axis the maximal dimensionless temperature in the bed at flow reversal is plotted. solid line: stable periodic states, dashed line: unstable states. $\diamond$: an eigenvalue crosses the unit circle at -1, $\bullet$: an eigenvalue crosses the unit circle at 1.*

At each integer value of t the flow direction reverses and the evolution equations and the boundary conditions change accordingly. A *cycle* consists of two flow reversals. This means that one cycle has a duration of two time units. The equations (5.1)-(5.2) with the boundary conditions (5.3) describe the model.

We would like to compute periodic solutions of (5.1)-(5.2) with the boundary conditions (5.3) that satisfy

$$\theta(0,x)=\theta(1,1-x) \text{ and } \chi(0,x)=\chi(1,1-x). \tag{5.4}$$

Thus we are interested in periodic solution that consist of two symmetric parts of one time unit. There might of course also exist periodic solutions without this symmetry, and also periodic solutions with period lengths equal to a multiple of the forcing period. Here we will not compute these solutions, but in principle these solutions can also be computed with the methods discussed in this paper, by choosing another period map F (integrating over more than one forcing period).

6. Results and Comparison of the Methods. For the reverse flow reactor we chose to discretize the spatial variable on a grid of 60 nodes using a finite volume

TABLE 6.1
Number of evaluations of F for the different methods

method	# F eval. for $p=5$	# F eval. for $p=7$	# F eval. for $p=9$
BSI1	9242	10000	12255
BSI2 (κ=0.01)	9229	10035	12278
BSI2 (κ=0.1)	10178	11432	11671
BSI2 (κ=0.25)	10038	10374	12774
BSI2 (κ=0.5)	9913	11839	12066
BSI2 (κ=0.75)	10565	12245	13446
BSI3 (κ=0.01)	9182	9992	12225
BSI3 (κ=0.1)	10380	10288	11608
BSI3 (κ=0.25)	10324	10704	13128
BSI3 (κ=0.5)	11057	12372	14700
BSI3 (κ=0.75)	10991	12639	14423
B+SI	8814	9244	9806
CNP (l=1)	40296	22268	20900
CNP (l=2)	32551	19992	20744
CNP (l=3)	26504	20832	21964
CNPGS (l=1)	30784	28428	29032
CNPGS (l=2)	31292	30266	29623
CNPGS (l=3)	27797	28091	31550

approach with upwinding of the first space derivatives in the mass balance. The discretization of the model equations results for the RFR in a system of 120 ordinary differential equations. This system of ODE's is integrated in time using the NAG fortran library routine D02EJF. In this section we present the results for the different methods used to compute a branch of periodic states of the reverse flow reactor. The branch is depicted in Figure 6.1. Here stable periodic states are represented by solid lines and unstable states by dashed lines. The different bifurcation points are denoted in the figure and explained in the caption.

We used three different values of p, i.e., 5, 7 and 9, for all the methods. For the CNP and the CNPGS, we also varied the parameter l. All the methods used the same initial solution on the branch, and the same initial basis V_0. The initial solution and initial basis were computed beforehand.

For all methods we used the same simple step size control procedure. After a convergence failure, the step size is decreased and divided by two and when a point is computed sufficiently fast, the step size is increased and multiplied by 1.6 (up to a maximum step size). This variable step size strategy allows us to observe not only differences in convergence speed, but also differences in the size of the domain of attraction of the different methods. For each point (x, λ) computed on the branch, the three largest eigenvalues of $J_F(x, \lambda)$ (according to absolute value) are computed with four digits of accuracy. For this purpose after the BSI or Newton-Picard iterations in most cases only one subspace iteration was sufficient.

Note that the projection step in the subspace iteration algorithm is strictly speaking not necessary if we keep the dimension of the subspace spanned by $\boldsymbol{V}$ fixed during the iterations in the BSI rank $p+1$ method or the Newton-Picard method. This step is intended for the approach in [13, 12]. The projection step aims to find the p vectors,

TABLE 6.2
Number of continuation points for the different methods

method	# points for $p = 5$	# points for $p = 7$	# points for $p = 9$
BSI1	348	348	370
BSI2 (κ=0.01)	348	348	370
BSI2 (κ=0.1)	361	365	349
BSI2 (κ=0.25)	360	348	365
BSI2 (κ=0.5)	348	365	349
BSI2 (κ=0.75)	350	366	362
BSI3 (κ=0.01)	348	348	370
BSI3 (κ=0.1)	372	348	348
BSI3 (κ=0.25)	360	348	360
BSI3 (κ=0.5)	364	350	374
BSI3 (κ=0.75)	361	362	351
B+SI	352	352	352
CNP (l=1)	464	355	359
CNP (l=2)	420	345	359
CNP (l=3)	356	350	360
CNPGS (l=1)	415	411	396
CNPGS (l=2)	419	438	408
CNPGS (l=3)	403	395	409

out of the $p + p_e$ dimensional subspace, that span the subspace determined by the p eigenvectors corresponding to the p largest eigenvalues. The projection step does help for the additional refining of the approximations to the three largest eigenvalues after convergence of the BSI rank $p + 1$ method, for which subspace iteration on the whole subspace V_i is used [12].

The matrix-vector products $J_F(\boldsymbol{x})v$ that are needed in the subspace iteration and in the Newton-Picard method can be calculated in different ways. A first possibility is to use a finite difference approximation given by

$$J_F(\boldsymbol{x})v = \frac{\partial F}{\partial x}(\boldsymbol{x})v \approx \epsilon^{-1}\big(F(x + \epsilon v, \lambda) - F(x, \lambda)\big),$$

where $(x, \lambda) = \boldsymbol{x}$. This approach costs one extra evaluation of F. Another possibility is to solve the initial value problem

$$\dot{z}(t) = \frac{\partial f}{\partial x}(t, x(t, x_0), \lambda)z(t), \text{ with } z(0) = v,$$

where $(x_0, \lambda) = \boldsymbol{x}$. The matrix vector product is now given by $J_F(\boldsymbol{x})v = z(1)$. This initial value problem is linear, in contrast to the initial value problem that has to be solved when evaluating F. We used both approaches in the tests in this section, for all the methods. We found that, if we counted the number of initial value problem solves for each method, there was not much difference in performance between the two approaches. This means that if we write "evaluation of F", one can also read "initial value problem solve".

The second, variational approach for computing matrix-vector products has the advantage that with integrators such as ODESSA [11], it is possible to integrate the

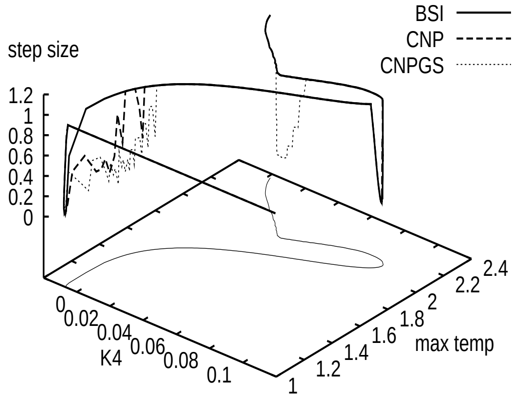

FIG. 6.2. *The step sizes for the different methods along the branch. Along the x-axis the parameter K_4 is plotted, along the y-axis the maximal dimensionless temperature in the bed at flow reversal is plotted and along the z-axis the step size used to compute the point on the branch is plotted. The maximum step size is equal to one. For each method the most efficient run is used; BSI1: $p = 7$, CNP: $l = 2$ and $p = 7$, CNPGS: $l = 3$ and $p = 5$.*

variational equations concurrently with the time integration for the evaluation of F. With this approach matrix-vector products can be computed cheaply compared to evaluating F. We will focus on this issue when comparing the performance of the BSI rank $p+1$ method with that of Broyden's method.

The number of evaluations of F (or number of initial value problem solves) that each method needs to compute the whole branch is listed in Table 6.1. The number of points computed on the curve are listed in Table 6.2.

6.1. Comparison of the BSI method and Newton-Picard methods. We first compare the performance of the BSI1 method with the performance of the Newton-Picard methods. We clearly see that the BSI rank $p+1$ method is much more efficient. Actually, the BSI rank $p+1$ method is most efficient for $p = 5$, but also for the other values of p the method is much more efficient than the either the CNP or the CNPGS method.

The next most efficient method is CNP with $l = 2$ and $p = 7$. Note that there is not much difference between the $p = 7$ and $p = 9$ cases. For $p = 5$, however, the CNP method needs considerably more function evaluations.

The CNPGS method is the least efficient method and needs the most function evaluations. This is clearly due to the fact that it has the smallest domain of attraction. As is seen from Table 6.2, the CNPGS method needs considerably more continuation points than both the BSI and the CNP method. This fact is also illustrated in Figure 6.2. From this figure, we observe that the BSI rank $p+1$ method uses the maximum step size all along the branch except close to the turning points. The CNP method has more or less the same robustness as the BSI rank $p+1$ method. The CNPGS method is the least robust method, since it is the only method that uses smaller step sizes to compute the unstable part of the branch between the two points where an eigenvalues crosses the unit circle at -1.

In [13] it is noted that the performance of the Newton-Picard methods in terms of the number of evaluations of F needed, is essentially independent of the number of nodes used in the discretization. In the order to check whether this is also the case for the BSI rank $p+1$ method, we computed the same branch of solutions using a

TABLE 6.3
Number of calls to the integration code ODESSA for both Broyden's method and the BSI rank $p+1$ method ($p=7$).

method	total	$G(\boldsymbol{x})=0$	eigenvalues
Broyden	4652	3996	656
BSI1	1621	1560	61

discretization of model equations on 100 nodes, so that $N = 200$. In this case the BSI rank $p+1$ method with $p=7$ needed 10397 evaluations of F to compute the whole branch. This number is close to the number of evaluations of F needed for the discretization on 60 nodes.

6.2. Comparison of the BSI method and Broyden's method. If we compare the number of F evaluations for the BSI method with Broyden's method followed by subspace iterations for the computation of the largest eigenvalues (see Table 6.1 and 6.2, lines denoted by B+SI), we see that Broyden's method is slightly more efficient. Because the numbers for the evaluations of F for Broyden's method and the BSI rank $p+1$ method are very similar, it is interesting to compare these two methods more precisely.

The BSI method has one clear advantage over Broyden's method. This advantage is that in one iteration of the BSI method, we are able to compute all the information needed for the updating of the Jacobian with one call to a time integration code such as ODESSA [11]. In this way the directional derivatives that are needed for the updating of the invariant subspace are computed concurrently with the evaluation of F.

The number of calls to ODESSA for both Broyden's method and the BSI rank $p+1$ method are given in Table 6.3. We see that the total number of calls for the BSI rank $p+1$ method is far less than halve the number for Broyden's method. The numbers in the second column denote the number of calls used for the computation of the periodic solutions on the branch. This number is equal to the number of iterations both methods need. We see that the extra rank p update for the BSI rank $p+1$ method speeds up the convergence of Broyden's method more than twice. The last column contains the number of calls to ODESSA for the computation of the three largest eigenvalues after the convergence to a periodic solution. We see that the BSI rank $p+1$ method only needs 61 extra calls.

The fact that the BSI rank $p+1$ method is able to compute almost all the needed matrix-vector product concurrently with the evaluation of F, results in favorable computation times when compared to Broyden's method, even though Broyden's method needs less function evaluations: the timing results for the whole branch are 798 sec. for the BSI method and 899 sec. for Broyden's method. These timing runs were performed on one node of the Beowulf cluster at SARA Computing and Networking Services, Amsterdam.

6.3. The Convergence of the BSI Method. In order to analyze the convergence behavior of the BSI rank $p+1$ method, we plot for the BSI1 method the value of $1/c$ along the computed branch of periodic states of the RFR, see Fig. 6.3. We see that this quantity does not approach to zero, except near the end of the branch, but in this part of the branch the BSI1 method converged in one step up to the desired tolerance. This means that for the largest part of the branch we can choose a value of κ such that the BSI2 and BSI3 methods, for which we have proved q-superlinear

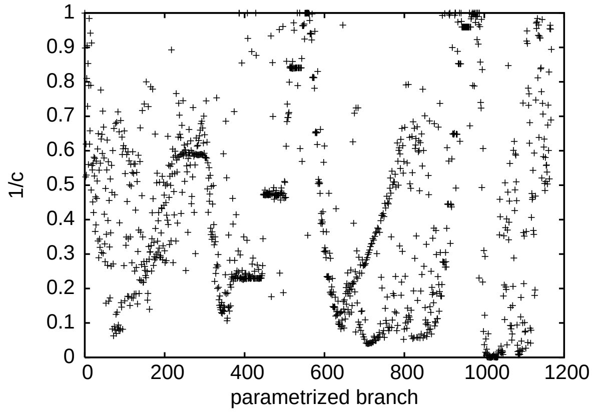

FIG. 6.3. *The sine of the angle between the update and the subspace.*

convergence, perform exactly the same steps as the BSI1 method. It is an interesting question whether it is possible to obtain an a priori lower bound for $1/c$ in certain cases. To answer this question it might be necessary to use the extra structure in the problem: the subspace $\boldsymbol{V}$ is always contained in the N dimensional subspace spanned by the first N standard basis vectors of $\mathbb{R}^{N+1}$ in which the BSI method operates, while the update $\boldsymbol{s}_i$ can be any vector in $\mathbb{R}^{N+1}$. Using this fact and the dependence of the map F on the parameter λ, it might be possible to obtain an a priori lower bound for $1/c$.

In Table 6.1 and 6.2 we give the results for the variants BSI2 and BSI3. We see that there is not much difference between the results for the BSI1 variant. Notice that the methods perform better if κ is smaller, i.e., if the method is "closer" to the BSI1 method.

7. Conclusions. In this paper we have introduced a Broyden rank $p+1$ update continuation method with subspace iteration (in short the BSI rank $p+1$ method) for the computation of branches of periodic solutions of periodically forced PDE's. The method makes efficiently use of the ideas behind Broyden's method and behind the Newton-Picard methods developed by [13]. As the name suggests, the method combines a Broyden method with a subspace iteration procedure for the computation of invariant subspaces. Simultaneously approximations to a periodic solution and to the eigenvalues that determine the stability are computed. In this way bifurcation points can be detected accurately and efficiently.

We have used the method to compute a branch of periodic states of a cooled reverse flow reactor (see Fig. 6.1). We have also used two variants of the Newton-Picard method [13] to compute the same branch. For the dynamical system describing the cooled reverse flow reactor, the BSI rank $p+1$ method proved to be the most efficient method, in terms of number of period map evaluations. In fact, the BSI method needed approximately half the number of evaluations of F as the next most efficient method (CNP with $l=2$ and $p=7$). The main advantage of the BSI rank $p+1$ method, when compared to the Newton-Picard methods, is the small number of period map evaluations per iteration of the method. The method proved also to have the largest domain of attraction. This results in the largest average step size, so that the method needed the fewest continuation points.

Of the two variants of the Newton-Picard method, the CNP and the CNPGS variant, the CNPGS is the least expensive variant in terms of number of evaluations of F per iteration. The CNPGS method, however, is also the least robust method of the two, and this results in the poorest overall performance.

We did not use a state-of-the-art implementation of the variants of the Newton-Picard method. Such an implementation, that incorporates locking and deflation in the subspace iteration, adaptive control of the dimension of the subspace and adaptive control of the number of Picard iterations [12], probably will perform better in tests than the implementation used in this paper. We believe that the BSI rank $p+1$ method will also benefit from a more sophisticated variant of the subspace iteration algorithm. How large the gain in efficiency will be is a topic for future investigations.

We also compared the BSI rank $p+1$ method to Broyden's method followed by subspace iterations for the computation of the largest eigenvalues. The BSI rank $p+1$ method and Broyden's method needed a comparable number of function evaluations for the test problem. When for the matrix-vector products the variational equations are solved (such as is done in the ODESSA code), the BSI rank $p+1$ exploits the possibilities of such a code better than Broyden's method. This resulted in shorter CPU times for the BSI rank $p+1$ method when compared to CPU times for Broyden's method.

We have proven that the variants BSI2 and BSI3 of the BSI rank $p+1$ method enjoy q-superlinear convergence. We also showed that, as long as the angle between the update and the approximate invariant subspace, stays bounded away from zero, there exist values of κ such that the BSI2 and BSI3 method behave exactly as the BSI1 method. For the presented example problem, we could experimentally show that this angle between the update and the approximate invariant subspace stays away from zero for the BSI1 method in the crucial parts of the computations.

We experimentally showed that the performance of the BSI rank $p+1$ method in terms of number of evaluations of F is essentially independent of the number of nodes used in the discretization. This fact is already known for the Newton-Picard methods, see [13].

REFERENCES

[1] C. G. BROYDEN, *A class of methods for solving nonlinear simultaneous equations*, Mathematics of Computation, 19 (1965), pp. 577–593.

[2] C. G. BROYDEN, J. E. DENNIS, AND J. J. MORÉ, *On the local and superlinear convergence of quasi-Newton methods*, J. Inst. Math. Appl., 12 (1973), pp. 223–245.

[3] D. T. CROFT AND M. G. LEVAN, *Periodic states of adsorption cycles–I. Direct determination and stability*, Chemical Engineering Science, 49 (1994), pp. 1821–1829.

[4] J. E. DENNIS, JR. AND R. B. SCHNABEL, *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice-Hall, 1983.

[5] D. M. GAY, *Some convergence properties of Broyden's method*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 623–630.

[6] K. GEORG, *On tracing an implicitely defined cirve by quasi-newton steps and calculating bifurcation by local perturbations*, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 35–50.

[7] H. JARAUSCH AND W. MACKENS, *Solving large nonlinear systems of equations by an adaptive condensation process*, Numerische Mathematik, 50 (1987), pp. 633–653.

[8] J. KHINAST, A. GURUMOORTHY, AND D. LUSS, *Complex dynamic features of a cooled reverse-flow reactor*, AIChE Journal, 44 (1998), pp. 1128–1140.

[9] A. J. KODDE AND A. BLIEK, *Selectivity enhancement in consecutive reactions using the pressure swing reactor*, Stud. Surf. Sci. Catal., 109 (1997), pp. 419–428.

[10] Y. A. KUZNETSOV, *Elements of applied bifurcation theory*, vol. 112 of Appl. Math. Sciences, Springer-Verlag New York, Inc., 1995.

[11] J. R. LEIS AND M. A. KRAMER, *Algorithm 658: ODESSA: An ordinary differential equation solver with explicit simultaneous sensitivity analysis*, ACM Transactions on Mathematical

Software, 14 (1988), pp. 61–67.
[12] K. Lust and D. Roose, *Computation and bifurcation analysis of periodic solutions of large-scale systems*, in Numerical methods for bifurcation problems and large-scaledynamical systems (Minneapolis, MN, 1997), E. Doedel and L. S. Tuckerman, eds., vol. 119 of IMA Vol. Math. Appl., 2000, pp. 265–301.
[13] K. Lust, D. Roose, A. Spence, and A. R. Champneys, *An adaptive Newton-Picard algorithm with subspace iteration for computing periodic solutions*, SIAM Journal on Scientific Computing, 19 (1998), pp. 1188–1209.
[14] J. Nocedal, *Updating Quasi-Newton matrices with limited storage*, Math. Comp., 35 (1980), pp. 773–782.
[15] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Algorithms and Architectures for Advanced Scientific Computing, Manchester University Press, Manchester, 1992.
[16] G. Shroff and H. Keller, *Stabilization of unstable procedures: The recursive projection method*, SIAM Journal on Numerical Analysis, 30 (1993), pp. 1099–1120.
[17] B. Van De Rotten and S. M. Verduyn Lunel, *A limited memory Broyden method to solve high-dimensional systems of nonlinear equations*, Tech. Report MI 2003-06, Mathematical Institute, University of Leiden, 2003.
[18] T. L. Van Noorden, S. M. Verduyn Lunel, and A. Bliek, *Direct determination of cyclic steady states of cyclically operated packed bed reactors*, in Volume 1: Molecular Properties, Reaction Diffusion Problems, Reaction Engineering, Simulation of Reactive Flows, F. Keil, W. Mackens, H. Voss, and J. Werther, eds., Springer-Verlag, 1999, pp. 311–318.
[19] ———, *Acceleration of the determination of periodic states of cyclically operated reactors and separators*, Chemical Engineering Science, 57 (2002), pp. 1041–1055.
[20] ———, *The efficient computation of periodic states of cyclically operated chemical processes*, IMA Journal of Applied Mathematics, 68 (2003), pp. 149–166.