AIMS Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 5(1): 603–618.
DOI:10.3934/math.2020040
Received: 03 October 2019
Accepted: 18 November 2019
Published: 16 December 2019

Research article

Acceleration of implicit schemes for large systems of nonlinear differential-algebraic equations

Mouhamad Al Sayed Ali[1] **and Miloud Sadkane**[2,*]

[1] Université Libanaise, Faculté des Sciences 1, Beyrouth, Liban
[2] Université de Brest, CNRS - UMR 6205, Mathématiques, 6, Av. Le Gorgeu. 29238 Brest Cedex 3, France

* **Correspondence:** Email: miloud.sadkane@univ-brest.fr.

Abstract: When solving large systems of nonlinear differential-algebraic equations by implicit schemes, each integration step requires the solution of a system of large nonlinear algebraic equations. The latter is solved by an inexact Newton method which, in its turn, leads to a set of large linear systems commonly solved by a Krylov subspace iterative method. The efficiency of the whole process depends on the initial guesses for the inexact Newton and the Krylov subspace methods. An inexpensive approach is proposed and justified that computes good initial guesses for these methods. It requires a subspace of small dimension and the use of line search and trust region for the inexact Newton method and Petrov-Galerkin for the Krylov subspace method. Numerical examples are included to illustrate the effectiveness of the proposed approach.

Keywords: nonlinear equations; nonlinear DAE systems; inexact Newton; GMRES; line search; trust region
Mathematics Subject Classification: 65H10, 65L80, 65F10, 65K10

1. Introduction

Consider the system of nonlinear differential-algebraic equations (DAEs)

$$\begin{cases} f(t, y(t), y'(t)) = 0, & \forall t \in [t_0, T], \\ y(t_0) = y^{(0)}, \end{cases} \tag{1.1}$$

where $y(t) \in \mathbb{R}^n$ is the unknown solution and n is large. We assume that the function $f : [t_0, T] \times \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is nonlinear and sufficiently smooth. We also assume that the solution exists and refer to [5, chap. 2] and [14, chap. 4] and references therein for the theory of DAEs. For an overview, see the surveys in [8, 15].

Integration schemes provide natural ways to solve (1.1) for large n. An important and general class of integration schemes is given by (see, e.g., [5, p. 71])

$$f\left(\sum_{j=0}^{q} a_j t_{i-j}, \sum_{j=0}^{q} b_j y_{i-j}, \frac{1}{h}\sum_{j=0}^{q} c_j y_{i-j}\right) = 0, \text{ for } i = q+1, \ldots, N, \tag{1.2}$$

where $y_0 = y^{(0)}$, N and $y_1, \ldots, y_q$ are given, $q \ll N$, and y_i is an approximation of $y(t_i)$ where $t_i = t_0 + ih$, $h = \frac{T-t_0}{N}$ is the step size, and a_j, b_j, c_j are given scalars. We assume throughout the note that the scheme (1.2) is stable, see, e.g., [5, p. 70] for a precise definition.

Let

$$x_i = \frac{1}{h}\sum_{j=0}^{q} c_j y_{i-j} \text{ and } \varphi_i = \sum_{j=1}^{q} c_j y_{i-j}. \tag{1.3}$$

Then

$$y_i = (hx_i - \varphi_i)/c_0 \tag{1.4}$$

and (1.2) becomes

$$G_i(x_i) = 0, \ i = q+1, \ldots, N, \tag{1.5}$$

where

$$G_i(x) = f\left(\sum_{j=0}^{q} a_j t_{i-j}, \sum_{j=1}^{q} b_j y_{i-j} - \frac{b_0}{c_0}\varphi_i + \frac{b_0}{c_0}hx, x\right). \tag{1.6}$$

Since n is large and the equation $G_i(x_i) = 0$ in (1.5) must be solved $N - q$ times, the natural way for finding x_i is the inexact Newton method, which is sketched in Algorithm 1.

Algorithm 1 Inexact Newton for solving $G_i(x_i) = 0$

Input: initial guess $x_i^{(0)}$.
Output: approximation of x_i

1: **for** $k = 0, 1, \ldots,$ **do**
2: Solve the system $G_i'(x_i^{(k)})\tilde{s}_i^{(k)} = -G_i(x_i^{(k)})$ inexactly, so that

$$\|G_i'(x_i^{(k)})\tilde{s}_i^{(k)} + G_i(x_i^{(k)})\| \le \epsilon_i^{(k)}\|G_i(x_i^{(k)})\|; \tag{1.7}$$

3: $x_i^{(k+1)} = x_i^{(k)} + \tilde{s}_i^{(k)}$;
4: **end for**

For a fixed i, Algorithm 1 starts with $x_i^{(0)}$ and computes the sequence $(x_i^{(k)})_{k\ge 0}$ which converges, under some conditions, to the solution x_i.

In (1.7) and throughout this note, the symbol $\|\ \|$ denotes the 2-norm. The matrix $G_i'(x_i^{(k)})$ is the Jacobian of G_i at $x_i^{(k)}$, the approximate solution $\tilde{s}_i^{(k)}$ is obtained by an appropriate iterative method, and $\epsilon_i^{(k)}$ is the required convergence tolerance threshold. In [10] it is shown that local convergence is guaranteed under the condition that the sequence $\epsilon_i^{(k)}$ is uniformly less than one.

Inexact Newton type methods are particularly suitable for solving large-scale nonlinear equations or series of nonlinear systems as those arising when large-scale ordinary differential equations (ODEs) are solved by implicit integration schemes, see, e.g., [2,4,6,7,10,12,13,16]. Krylov subspace iterative methods [17] are generally used to solve the linear systems arising in step 2 of Algorithm 1. They have the advantage of requiring only the action of the Jacobian matrix on a given vector and the Jacobian need not be stored. However, unless good and cheap preconditioners are available, the efficiency of these methods strongly depends on the initial guesses. In the present note, this will be obtained by the Petrov-Galerkin method (see [17]), which has proved to be effective.

The choice of the initial guess $x_i^{(0)}$ in Algorithm 1 is crucial for convergence. The standard choice in Newton-like methods is the zero vector or the previous iterate. This, however, does not necessarily lead to convergence. In this note we will show that the line search and trust region algorithms (see, e.g., [11]) offer a cheap and efficient way for the computation of $x_i^{(0)}$ through convergent sequences. The line search and trust region algorithms involve linear systems similar to the one in step 2 and are solved in the same way.

Thus, for a fixed i, computing a good initial $x_i^{(0)}$ and good initial guesses for each linear system to be solved in step 2 will result in an acceleration of the scheme (1.2), which is the purpose of the present work. For this reason, issues specific to numerical methods for DAEs are not considered (the interested readers may usefully refer to [1,5,14]).

The special case when f is of the form $f(t,y,z) = z - g(t,y)$ where g is a (linear) non linear function leads to (linear) nonlinear ODEs. This case has been addressed in a previous work [2,3] where some subspaces were used to provide good initial guesses of the underlying implicit schemes. The approach taken in the present work extends to DAEs the investigations started in [2,3] for ODEs. However, the approximation subspaces used for ODEs are no longer valid here since y' is not given explicitly in terms of t and y. Nevertheless, our goal is still to construct a low-dimensional subspaces (see (2.7), (2.11)) that contain good approximations of each initial guess in Algorithm 1 and for the corresponding linear systems.

The note is organized as follows. In Section 2 we explain the construction of a low-dimensional subspace that contains useful information on the desired initial guess for Algorithm 1 and provide related error estimates. In Section 3 we briefly review the Petrov-Galerkin process and the line search and trust region algorithms. The algorithmic aspect related to the acceleration of the numerical solution of (1.1) is discussed in section 4. Section 5 is devoted to numerical tests and a conclusion is given in Section 6

2. Acceleration of implicit scheme

When implemented, Algorithm 1 can only provide an approximate solution $\tilde{x}_i$ to x_i, such that

$$\|\tilde{G}_i(\tilde{x}_i)\| \leq \varepsilon, \quad i = q+1, \ldots, N, \tag{2.1}$$

where ε is a tolerance threshold and $\tilde{G}_i$ is the approximate analogue of G_i, defined by

$$\tilde{G}_i(x) = f\left(\sum_{j=0}^{q} a_j t_{i-j}, \sum_{j=1}^{q} b_j \tilde{y}_{i-j} - \frac{b_0}{c_0}\tilde{\varphi}_i + \frac{b_0}{c_0} hx, x\right), \tag{2.2}$$

with

$$\tilde{\varphi}_i = \sum_{j=1}^{q} c_j \tilde{y}_{i-j} \text{ and } \tilde{y}_i = \frac{h\tilde{x}_i - \tilde{\varphi}_i}{c_0}. \tag{2.3}$$

Assume that $\tilde{y}_0 = y_0 = y^{(0)}$ and that for $i = 1, \ldots, q$, $\tilde{y}_i$ is computed, for example, with an i-step scheme, such that

$$\max_{0\le i\le q} \|y_i - \tilde{y}_i\| = O(\varepsilon). \tag{2.4}$$

Then, since the scheme (1.2) is stable, there exists a constant $C > 0$ such that

$$\max_{q+1\le i\le N} \|y_i - \tilde{y}_i\| \le C\left(\max_{0\le i\le q} \|y_i - \tilde{y}_i\| + \max_{q+1\le i\le N} \|\tilde{G}_i(\tilde{x}_i)\|\right).$$

Using (2.1), (2.4) and (1.4), (2.3) we get

$$\max_{0\le i\le N} \|y_i - \tilde{y}_i\| = O(\varepsilon), \tag{2.5}$$

$$\max_{0\le i\le N} \|x_i - \tilde{x}_i\| = O(\varepsilon/h). \tag{2.6}$$

The following theorem shows that a good approximation to x_i (and therefore to $y_i + \varphi_i/c_0$) can be obtained from the subspace spanned by the previous approximate solutions $\tilde{x}_{i-1}, \tilde{x}_{i-2}, \ldots$.

Theorem 2.1. *Assume that $y \in C^r([t_0, T])$, $r \ll n$ and let*

$$\mathcal{V}_i = \text{span}\{\tilde{x}_{i-1}, \tilde{x}_{i-2}, \ldots, \tilde{x}_{i-r}\}. \tag{2.7}$$

Then, there exist a $\tilde{x} \in \mathcal{V}_i$ and a $\tilde{y} \in (-\tilde{\varphi}_i/c_0) + \mathcal{V}_i$ such that for $i = q+1, \ldots, N$,

$$\|\tilde{x} - x_i\| = O(h^{p-1}) + O(h^{r-1}) + O(\varepsilon/h), \tag{2.8}$$

$$\|\tilde{y} - y_i\| = O(h^p) + O(h^r) + O(\varepsilon), \tag{2.9}$$

where p is the order of the scheme (1.2).

Proof. The Lagrange interpolation formula (see, e.g., [9]) ensures the existence of constants $\alpha_l, 1 \le l \le r$ such that

$$\|y(t_i) - \sum_{l=1}^{r} \alpha_l y(t_{i-l})\| = O(h^r). \tag{2.10}$$

Let $\tilde{x} = \sum_{k=1}^{r} \alpha_k \tilde{x}_{i-k} \in \mathcal{V}_i$. Then we have

$$\begin{aligned} \tilde{x} - x_i &= \frac{1}{h}\left(\sum_{k=1}^{r} \alpha_k \sum_{j=0}^{q} c_j \tilde{y}_{i-k-j} - \sum_{j=0}^{q} c_j y_{i-j}\right) \\ &= \frac{1}{h}\left(\sum_{j=0}^{q} c_j \left(\sum_{k=1}^{r} \alpha_k \tilde{y}_{i-j-k} - y_{i-j}\right)\right). \end{aligned}$$

The estimate (2.8) follows by noticing that

$$\sum_{k=1}^{r} \alpha_k \tilde{y}_{i-k-j} - y_{i-j} = (a) + (b) + (c) + (d),$$

where

$$\begin{aligned}(a) &= \sum_{k=1}^{r} \alpha_k(\tilde{y}_{i-j-k} - y_{i-j-k}),\ (b) = \sum_{k=1}^{r} \alpha_k(y_{i-j-k} - y(t_{i-j-k})),\\ (c) &= \sum_{k=1}^{r} \alpha_k y(t_{i-j-k}) - y(t_{i-j}),\ (d) = y(t_{i-j}) - y_{i-j},\end{aligned}$$

and using (2.10) in (c) and the stability and order p properties in (a), (b) and (d), i.e., (2.5) and $\|y_i - y(t_i)\| = O(h^p)$ for $i = 0, \ldots, N$.

Now let $\tilde{y} = -\tilde{\varphi}_i/c_0 + h\frac{\tilde{x}}{c_0} \in -\tilde{\varphi}_i/c_0 + \mathcal{V}_i$. Then we have

$$\begin{aligned}\tilde{y} - y_i &= \frac{h}{c_0}(\tilde{x} - x_i) + \frac{1}{c_0}(\varphi_i - \tilde{\varphi}_i)\\ &= \frac{h}{c_0}(\tilde{x} - x_i) + \frac{1}{c_0}\sum_{j=1}^{q} c_j(y_{i-j} - \tilde{y}_{i-j})\end{aligned}$$

and the estimate (2.9) follows then from (2.8) and (2.5). □

The computation of the initial guess for Algorithm 1 will be based on the subspace $\mathcal{V}_i$. However, in order to further reduce the computational cost, we will modify this subspace by keeping only the last active vectors, that is, the approximate solutions which require the use of Algorithm 1 to satisfy (2.1). For example, if it happens that no iterative method is needed in Inexact Newton, or in other words, that the initial guess $x_i^{(0)}$ already satisfies (2.1), then we set $\tilde{x}_i = x_i^{(0)}$ and use the same subspace $\mathcal{V}_i$ for the next iteration i of the scheme. This leads to the following simplification.

Corollary 1. *Let*

$$\mathcal{V}_i = \text{span}\{\tilde{x}_{i-l_1}, \tilde{x}_{i-l_2}, \ldots, \tilde{x}_{i-l_r}\} \tag{2.11}$$

be the subspace spanned by the last r vectors whose computations require the use the inexact Newton method, where $l_1 < l_2 < \ldots < l_r$ and $r \ll n$. Then there exist a $\tilde{x} \in \mathcal{V}_i$ and a $\tilde{y} \in (-\tilde{\varphi}_i/c_0) + \mathcal{V}_i$ such that for $i = q + 1, \ldots, N$,

$$\|\tilde{x} - x_i\| = O(h^{p-1}) + O(h^{r+l_1-2}) + O(\varepsilon/h), \tag{2.12}$$

$$\|\tilde{y} - y_i\| = O(h^p) + O(h^{r+l_1-1}) + O(\varepsilon). \tag{2.13}$$

Proof. The hypothesis means that the vectors $\tilde{x}_{i-k}, k = 1, \ldots, l_1 - 1$ are already in $\mathcal{V}_i$. Therefore the subspaces (2.7) and (2.11) coincide and the proof of Theorem 2.1 can be repeated to show the existence of $\tilde{x} \in \mathcal{V}_i$ and $\tilde{y} \in (-\tilde{\varphi}_i/c_0) + \mathcal{V}_i$ that satisfy (2.12) and (2.13). □

3. Petrov-Galerkin, line search and trust region

Using the subspace $\mathcal{V}_i$ defined in (2.7) or (2.11), we seek the initial guess $\hat{s}_i^{(k)} \in -x_i^{(k)} + \mathcal{V}_i$ to the linear system $\tilde{G}_i'(x_i^{(k)})\tilde{s}_i^{(k)} = -\tilde{G}_i(x_i^{(k)})$ that satisfies the Petrov-Galerkin condition

$$\tilde{G}_i'(x_i^{(k)})\hat{s}_i^{(k)} + \tilde{G}_i(x_i^{(k)}) \perp \tilde{G}_i'(x_i^{(k)})\mathcal{V}_i,$$

where $\perp$ denotes orthogonality with respect to the Euclidean inner product.

This is equivalent to the least-squares problem

$$\min_{s\in -x_i^{(k)}+\mathcal{V}_i} \|\tilde{G}_i'(x_i^{(k)})s + \tilde{G}_i(x_i^{(k)})\|$$

whose solution provides the initial guess

$$\hat{s}_i^{(k)} = -x_i^{(k)} + V_i v_i^{(k)}, \tag{3.1}$$

where V_i is a matrix whose columns form an orthonormal basis of $\mathcal{V}_i$ and $v_i^{(k)}$ solves the small-size linear system

$$\left[(\tilde{G'}_i(x_i^{(k)})V_i)^T(\tilde{G'}_i(x_i^{(k)})V_i)\right] v_i^{(k)} = -(\tilde{G'}_i(x_i^{(k)})V_i)^T\left(\tilde{G}_i(x_i^{(k)}) - \tilde{G'}_i(x_i^{(k)})x_i^{(k)}\right). \tag{3.2}$$

Recall that the initial guess $x_i^{(0)}$ has to be chosen to guarantee convergence of the sequence $(x_i^{(k)})$ to x_i. To this end, the line search and trust region algorithms will be employed. These algorithms compute a sequence of vectors $u_i^{(k)}$ that converges, under some conditions, to a local minimum or a saddle point of the function

$$h_i(x) = \frac{1}{2}\|\tilde{G}_i(x)\|^2. \tag{3.3}$$

We refer to [11] for more details on these algorithms. Here, we present a brief description useful for understanding Algorithms 2, 3 and 4.

The sequence of vectors computed by the line search algorithm is given by

$$u_i^{(k+1)} = u_i^{(k)} + \lambda_k p_i^{(k)},\ k \geq 0, \lambda_k > 0, \tag{3.4}$$

where $u_i^{(0)}$ is an initial guess and $p_i^{(k)}$ is a descent direction of $\tilde{G}_i$ at $u_i^{(k)}$ (that is, $\nabla h_i(u_i^{(k)})^T p_i^{(k)} < 0$) and the scalar λ_k is chosen so that the sequence $u_i^{(k)}$ satisfies the Wolf condition

$$h_i(u_i^{(k+1)}) \leq h_i(u_i^{(k)}) + \alpha\lambda_k \nabla h_i(u_i^{(k)})^T p_i^{(k)}, \tag{3.5}$$

where $\alpha \in (0, 1/2)$ is a parameter typically set to 10^{-4}.

The computation of λ_k uses the backtracking method, which starts with $\lambda_k = 1$ and repeatedly reduces it until an acceptable iterate $u_i^{(k+1)}$ satisfying (3.5) is found. The descent direction

$$p_i^{(k)} = -\left\{\tilde{G}_i'(u_i^{(k)})\right\}^{-1}\tilde{G}_i(u_i^{(k)}), \tag{3.6}$$

which is referred to as the Newton direction, ensures that (3.5) is satisfied. In practice, an iterative method is used to compute an approximation $\tilde{p}_i^{(k)}$ to $p_i^{(k)}$ that satisfies

$$\|\tilde{G}_i'(u_i^{(k)})\tilde{p}_i^{(k)} + \tilde{G}_i(u_i^{(k)})\| \leq \eta_i^{(k)}\|\tilde{G}_i(u_i^{(k)})\|, \tag{3.7}$$

where $\eta_i^{(k)} \ll 1$ is some convergence tolerance threshold.

The trust region algorithm generates a sequence of vectors of the form

$$u_i^{(k+1)} = u_i^{(k)} + d_i^{(k)}, \tag{3.8}$$

where

$$d_i^{(k)} = \underset{\|d\| \le \delta_i^{(k)}}{\operatorname{argmin}} \Psi_i^{(k)}(d), \tag{3.9}$$

and where $\delta_i^{(k)} > 0$ and $\Psi_i^{(k)}$ is defined by

$$\Psi_i^{(k)}(d) = \frac{1}{2}\|\tilde{G}_i(u_i^{(k)}) + \tilde{G}_i'(u_i^{(k)})d\|^2. \tag{3.10}$$

It is easy to verify that

$$d_i^{(k)} = \begin{cases} -(B_i^{(k)})^{-1}\nabla h_i(u_i^{(k)}) & \text{if } \|(B_i^{(k)})^{-1}\nabla h_i(u_i^{(k)})\| \le \delta_i^{(k)}, \\ -(B_i^{(k)} + \mu_i^{(k)} I_n)^{-1}\nabla h_i(u_i^{(k)}) & \text{if } \|(B_i^{(k)})^{-1}\nabla h_i(u_i^{(k)})\| > \delta_i^{(k)}, \end{cases} \tag{3.11}$$

where $B_i^{(k)} = \tilde{G}_i'(u_i^{(k)})^T \tilde{G}_i'(u_i^{(k)})$ and $\mu_i^{(k)} \ge 0$. The matrix $B_i^{(k)}$ and the scalar $\mu_i^{(k)}$ are related by

$$\|(B_i^{(k)} + \mu_i^{(k)} I_n)^{-1}\nabla h_i(u_i^{(k)})\| = \delta_i^{(k)}. \tag{3.12}$$

Note that $d_i^{(k)}$ is a descent direction of $\tilde{G}_i$ at $u_i^{(k)}$ and coincides with the Newton direction (3.6) when $\|(B_i^{(k)})^{-1}\nabla h_i(u_i^{(k)})\| \le \delta_i^{(k)}$.

Two methods are commonly used to compute $\mu_i^{(k)}$: the hook step and the dogleg step. The former finds $\mu_i^{(k)}$ which provides an approximation of (3.12) and takes $u_i^{(k+1)} = u_i^{(k)} - (B_i^{(k)} + \mu_i^{(k)} I_n)^{-1}\nabla h_i(u_i^{(k)})$, while the latter makes a piecewise linear approximation to the curve $\mu \to u_i^{(k)} - (B_i^{(k)} + \mu I_n)^{-1}\nabla h_i(u_i^{(k)})$ and takes $u_i^{(k+1)}$ as the point on this approximation such that $\|u_i^{(k+1)} - u_i^{(k)}\| = \delta_i^{(k)}$. Both methods require the solution of large linear systems.

To reduce the cost of computing $d_i^{(k)}$, we replace $d_i^{(k)}$ in (3.8) by

$$d_i^{(k)} = V_i c_i^{(k)}, \tag{3.13}$$

where, as in (3.1), V_i is a matrix whose columns form an orthonormal basis of $\mathcal{V}_i$, and $c_i^{(k)}$ solves the lower-dimensional minimization problem

$$c_i^{(k)} = \underset{\|c\| \le \delta_i^{(k)}}{\operatorname{argmin}} \Psi_i^{(k)}(V_i c). \tag{3.14}$$

Note that $c_i^{(k)}$ is cheap to compute and that $V_i c_i^{(k)}$ is a descent direction of G_i at $u_i^{(k)}$. The condition for accepting $u_i^{(k+1)}$ is still the Wolf condition (3.5), which can now be written

$$h_i(u_i^{(k+1)}) \le h_i(u_i^{(k)}) + \alpha\nabla h_i(u_i^{(k)})^T V_i c_i^{(k)}. \tag{3.15}$$

4. The main algorithm

In this section we describe an inexact Newton-type algorithm to compute the sequence $\{\tilde{x}_i\}$ that satisfies (2.1). A template is given in Algorithm 2 and for the sake of clarity the line search and trust region steps are given in Algorithms 3 and 4.

Algorithm 2 Inexact Newton with Line Search or Trust Region

Input: parameters r, $\varepsilon_{\mathrm{LS}}^{(i)}$, $\varepsilon_{\mathrm{TR}}^{(i)}$, i=1,2, ε, $k_{\max\mathrm{LS}}$, $k_{\max\mathrm{TR}}$, $k_{\max\mathrm{IN}}$, and initial solutions $\tilde{y}_1, \ldots, \tilde{y}_q$. It is assumed that the $\tilde{y}_l$'s are given or computed with an appropriate scheme such that $\max_{1\le l\le q} \|\tilde{y}_l - y_l\| = O(\varepsilon)$ (see (2.4)).

Output: sequence $(\tilde{x}_i)$ that satisfies (2.1).

$k = 0$, $i = q + 1$, $k_0 = \min(q + 1, r)$.
$R_i = [\tilde{x}_{i-1}, \ldots, \tilde{x}_{i-k_0}]$, $V_i = \mathrm{orth}(R_i)$
while $i \le N$ **do**
  Compute an initial solution $u_i^{(0)}$
  Line Search - Trust Region:
  Compute $u_i^{(k+1)}$ using Algorithms 3 or 4;
  $k := k + 1$, $x_i^{(0)} = u_i^{(k)}$;
  if $\|\tilde{G}_i(x_i^{(0)})\| \le \varepsilon$ **then**
    $\tilde{x}_i = x_i^{(0)}$, $i := i + 1$, go to 3;
  end if
  Inexact Newton:
  $k = 0$;
  while $\|\tilde{G}_i(x_i^{(k)})\| > \varepsilon$ and $k \le k_{\max\mathrm{IN}}$ **do**
    Compute $\hat{s}_i^{(k)}$ via Petrov-Galerkin (see (3.1));
    if $\|\tilde{G}_i(x_i^{(k)} + \hat{s}_i^{(k)})\| \le \varepsilon$ **then**
      $\tilde{x}_i = x_i^{(k)} + \hat{s}_i^{(k)}$, $i := i + 1$, go to 3;
    else if $\hat{s}_i^{(k)}$ satisfies (1.7) **then**
      $x_i^{(k+1)} = x_i^{(k)} + \hat{s}_i^{(k)}$, $k := k + 1$, go to 11;
    else
      Compute $\tilde{s}_i^{(k)}$ by GMRES starting with $\hat{s}_i^{(k)}$;
      $\hat{s}_i^{(k)} = \tilde{s}_i^{(k)}$, go to 13;
    end if
  end while
  $\tilde{x}_i = x_i^{(k)}$;
  $k_0 = \mathrm{rank}(R_i)$;
  if $k_0 < r$ **then**
    $R_{i+1} = [R_i, \tilde{x}_i]$;
  else
    $R_i = [R_i(:, 2 : r), \tilde{x}_i]$;
  end if
  $V_{i+1} = \mathrm{orth}(R_{i+1})$; $i := i + 1$;
end while

Algorithm 2 takes as input the desired number of columns r of $\mathcal{V}_i$, convergence tolerance thresholds and maximum number of iterations, $\varepsilon_{\mathrm{LS}}^{(i)}$, $\varepsilon_{\mathrm{TR}}^{(i)}$, i=1,2, ε, $k_{\max\mathrm{LS}}$, $k_{\max\mathrm{TR}}$, $k_{\max\mathrm{IN}}$, for line search, trust region and inexact Newton methods, and approximations $\tilde{y}_1, \ldots, \tilde{y}_q$ of $y_1, \ldots, y_q$ (recall that the $\tilde{y}_l$'s and y_l's are related by (2.4)). The initial phase computes the matrix V_i whose columns form an orthonormal basis of $\mathrm{span}\{\tilde{x}_{i-1}, \ldots, \tilde{x}_{i-\min(q+1,r)}\}$. The notation orth denotes a column orthonormalization process.

The main phase (steps 3 – 30) computes $\tilde{x}_{q+1}, \ldots, \tilde{x}_N$. The line search or trust region methods provides (steps 5–6) the initial guess $x_i^{(0)}$. Then we check if $x_i^{(0)}$ satisfies the condition (2.1). If so, we set $\tilde{x}_i = x_i^{(0)}$. Otherwise, we perform inexact Newton iterations (steps 11–21) until and acceptable $x_i^{(k)}$ is found. Then, we set $\tilde{x}_i = x_i^{(k)}$ and append it to the previous approximations to form the new matrix V_{i+1}.

Algorithm 3 Line Search

Input: vector $u_i^{(0)}$ and matrix V_i provided by Algorithm 2 at iteration i, $k = 0$.
parameters $\varepsilon_{\text{LS}}^{(1)}$, $\varepsilon_{\text{LS}}^{(2)}$, $k_{\max\text{LS}}$.
Output: vector $u_i^{(k)}$ obtained by Line Search at iteration k.

1: **while** $\|\tilde{G}_i(u_i^{(k)})\| > \varepsilon_{\text{LS}}^{(1)}$ & $k \le k_{\max\text{LS}}$ **do**
2: Compute the initial guess $\hat{p}_i^{(k)}$ for the system (3.6) by applying the Petrov-Galerkin process on range(V_i);
3: **if** $\hat{p}_i^{(k)}$ satisfies (3.7) **then**
4: $\tilde{p}_i^{(k)} = \hat{p}_i^{(k)}$;
5: **else**
6: Starting GMRES with $\hat{p}_i^{(k)}$, compute an approximation $\tilde{p}_i^{(k)}$ to (3.6) such that (3.7) holds;
7: **end if**
8: Compute the scalar λ_k by the backtracking method such that (3.5) holds;
9: $u_i^{(k+1)} = u_i^{(k)} + \lambda_k \tilde{p}_i^{(k)}$;
10: Stop if $\left|\|\tilde{G}_i(u_i^{(k+1)})\| - \|\tilde{G}_i(u_i^{(k)})\|\right| \le \varepsilon_{\text{LS}}^{(2)}$;
11: $k := k + 1$;
12: **end while**

Algorithm 4 Trust Region

Input: vector $u_i^{(0)}$ and matrix V_i provided by Algorithm 2 at iteration i, $k = 0$.
parameters $\varepsilon_{\text{TR}}^{(1)}$, $\varepsilon_{\text{TR}}^{(2)}$, $k_{\max\text{TR}}$.
Output: vector $u_i^{(k)}$ obtained by Trust Region at iteration k.

1: **while** $\|\tilde{G}_i(u_i^{(k)})\| > \varepsilon_{\text{TR}}^{(1)}$ & $k \le k_{\max\text{TR}}$ **do**
2: Compute an approximation $c_i^{(k)}$ to (3.14) using hook step or dogleg step;
3: Compute $u_i^{(k+1)} = u_i^{(k)} + V_i c_i^{(k)}$ satisfying (3.15);
4: Stop if $\left|\|\tilde{G}_i(u_i^{(k+1)})\| - \|\tilde{G}_i(u_i^{(k)})\|\right| \le \varepsilon_{\text{TR}}^{(2)}$;
5: $k := k + 1$;
6: **end while**

5. Numerical tests

We present numerical results that illustrate the efficiency of the proposed approach. We consider the system of PDEs with unknowns $u = u(x, t)$ and $v = v(x, t)$

$$\begin{cases} \frac{\partial u}{\partial t} &= -\sin(2uv)\frac{\partial u}{\partial x} + \mu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial x^2})u^2 v, \\ 0 &= t\frac{\partial v}{\partial x} + \frac{\partial^2 v}{\partial x^2} + t^2 v + t^2 \sin(tx), \end{cases}$$

for $0 < x < 1, 0 < t < 1$ with the initial conditions

$$\begin{aligned} u(0,x) &= u(1,x) = \pi - 2\pi x, \quad u(t,0) = \pi, \quad u(t,1) = -\pi, \\ v(0,x) &= 1, \quad v(1,x) = \cos(x), \quad v(t,0) = 1, \quad v(t,1) = \cos(t). \end{aligned}$$

The second partial derivatives are discretized using centered finite differences with step size $\Delta x = 1/(N+1)$ so that the size of the resulting system of nonlinear DAEs equals $2N$. The first partial derivatives are discretized using right finite differences. We present numerical results with $N = 5000$, two values of the parameter μ ($\mu = 10^{-2}$ and $\mu = 1$) and when the differential equation is solved by the implicit Euler scheme (see, e.g., [5, p. 41])

$$f\left(t_i, y_i, \frac{y_i - y_{i-1}}{h}\right) = 0 \tag{5.1}$$

and the Crank-Nicolson scheme(see, e.g., [5, p. 72])

$$f\left(\frac{t_{i-1} + t_i}{2}, \frac{y_{i-1} + y_i}{2}, \frac{y_i - y_{i-1}}{h}\right) = 0, \tag{5.2}$$

with $h = 1/100$.

The nonlinear systems arising in these schemes are solved by Inexact Newton (Algorithm 1), and the initial guesses are obtained via line-search and trust-region (Algorithms 2, 3,4).

The parameters used in Algorithm 1 are as follows: the number of vectors in the matrix R_i is given by $r = 20$; the tolerance thresholds ε and $\eta_i^{(k)}$ (used in (2.1) and (3.7)) are respectively fixed at 10^{-5} and 10^{-2}; the initial guess $u_i^{(0)}$ for line search and trust region algorithms is chosen so that

$$\begin{aligned} \|G_0(u_0^{(0)})\| &= \min(\|G_0(0)\|, \|G_0(-y^{(0)}/h)\|), \\ \|G_i(u_i^{(0)})\| &= \min(\|G_i(0)\|, \|G_i(x_{i-1})\|), \ i \geq 1. \end{aligned}$$

Tests with line search. The parameters used in Algorithm 3 are as follows: $\varepsilon_{\text{LS}}^{(1)} = 1$, $\varepsilon_{\text{LS}}^{(2)} = 10^{-6}$, $k_{\max\text{LS}} = 15$, and the parameter α in the Wolfe condition (3.5) is fixed at 10^{-4}.

Tables 1 and 2 show the residual norm $\|G_i(x_i^{(k)} + \hat{s}_i^{(k)})\|$ of the approximate solution $\hat{s}_i^{(k)}$ and the relative residual norm $\|G_i'(u_i^{(k)})\hat{p}_i^{(k)} + G_i(u_i^{(k)})\|/\|G_i(u_i^{(k)})\|$ of the initial guess $\hat{p}_i^{(k)}$ using implicit Euler and Crank-Nicolson schemes. The tables also show the residual norm $\|G_i(x_i^{(k+1)})\|$. These results are illustrated for some iterations k of Algorithm 3. From iterations $i = 10, 20, 75, 99$, we see that only one iteration of line search is used.

Figures 1, 2 show the nonlinear residual norm $\|G_i(x_i^{(0)})\|$, the run time in seconds required at each iteration of implicit euler and Crank-Nicolson schemes, and the number of GMRES iterations required at each iteration of the scheme for computing x_i. Figure 1 shows that $\|G_i(x_i^{(0)})\| \leq 10^{-4}$, which means that a good descent direction is obtained from the subspace $\mathcal{V}_i$. Also, it shows that, as the iteration i increases, the run time decreases from about 1000 seconds to approximately 5 seconds. The Figure 2 shows that the number of GMRES iterations required at each iteration of the scheme for computing x_i. Here also we see that as the iteration i increases, the number of GMRES iterations decreases. These results show that the subspace $\mathcal{V}_i$ leads to a significant acceleration of implicit Euler and Crank-Nicolson schemes.

Table 1. Algorithm 2 with line-search (implicit Euler scheme is used).

Iteration (i,k)	$\|\tilde{G}_i(x_i^{(k)} + \hat{s}_i^{(k)})\|$	$\|\tilde{G}_i'(u_i^{(k)})\hat{p}_i^{(k)} + \tilde{G}_i(u_i^{(k)})\|/\|\tilde{G}_i(u_i^{(k)})\|$	$\|\tilde{G}_i(x_i^{(k+1)})\|$
(0,0)	1.772×10^3	9.866×10^{-1}	10.0176
(0,1)	1.772×10^3	1	1.0007×10^{-1}
(0,2)	1.772×10^3	1.7542	1.0007×10^{-3}
(0,3)	1.772×10^3		1.0025×10^{-5}
(0,4)	1.772×10^3		9.9857×10^{-6}
(1,0)	8.0885×10^3	2.2826×10^{-1}	8.8481
(1,1)	8.0885×10^3	9.9999×10^{-1}	8.8544×10^{-2}
(1,2)	8.0885×10^3	9.9999×10^{-1}	8.8543×10^{-4}
(1,3)	8.0885×10^3		9.9874×10^{-6}
(10,0)	7.8958×10^{-5}	2.1328×10^{-8}	7.8958×10^{-5}
(20,0)	1.9882×10^{-5}	5.7646×10^{-9}	1.9882×10^{-5}
(20,1)	1.9908×10^{-5}		9.8338×10^{-6}
(75,0)	2.4432×10^{-5}	9.1764×10^{-9}	2.4432×10^{-5}
(75,1)	2.4170×10^{-5}		9.9932×10^{-6}
(99,0)	1.5471×10^{-5}	8.0693×10^{-9}	1.5471×10^{-5}
(99,1)	1.5482×10^{-5}		9.9691×10^{-6}

Table 2. Algorithm 2 with line-search (Crank Nicolson scheme is used).

Iteration (i,k)	$\|\tilde{G}_i(x_i^{(k)} + \hat{s}_i^{(k)})\|$	$\|\tilde{G}_i'(u_i^{(k)})\hat{p}_i^{(k)} + \tilde{G}_i(u_i^{(k)})\|/\|\tilde{G}_i(u_i^{(k)})\|$	$\|\tilde{G}_i(x_i^{(k+1)})\|$
(0,0)	5.3741×10^2	9.9085×10^{-1}	3.9087
(0,1)	5.3741×10^2	1	3.9078×10^{-2}
(0,2)	5.3741×10^2	1.3592	3.9078×10^{-4}
(0,3)	5.3741×10^2		9.9861×10^{-6}
(1,0)	1.7559×10^3	6.5815×10^{-1}	1.8753×10^1
(1,1)	1.7559×10^3	9.9999×10^{-1}	1.8763×10^{-1}
(1,2)	1.7559×10^3	9.9999×10^{-1}	1.8763×10^{-3}
(1,3)	1.7559×10^3		1.8773×10^{-5}
(1,4)	1.7559×10^3		9.9840×10^{-6}
(20,0)	2.7249×10^{-5}	3.2071×10^{-8}	2.7249×10^{-5}
(20,1)	2.7248×10^{-5}		9.9286×10^{-6}
(75,0)	1.7721×10^{-5}	8.1385×10^{-9}	1.7721×10^{-5}
(75,1)	1.7687×10^{-5}		9.8056×10^{-6}
(99,0)	2.2422×10^{-5}	1.2082×10^{-8}	2.2422×10^{-5}
(99,1)	2.2378×10^{-5}		9.9807×10^{-6}

Tests with trust region. The parameters used in Algorithm 4 are (the same as in Algorithm 3) : $\varepsilon_{\mathrm{TR}}^{(1)} = 1$, $\varepsilon_{\mathrm{TR}}^{(2)} = 10^{-6}$, $k_{\max\mathrm{LS}} = 15$.

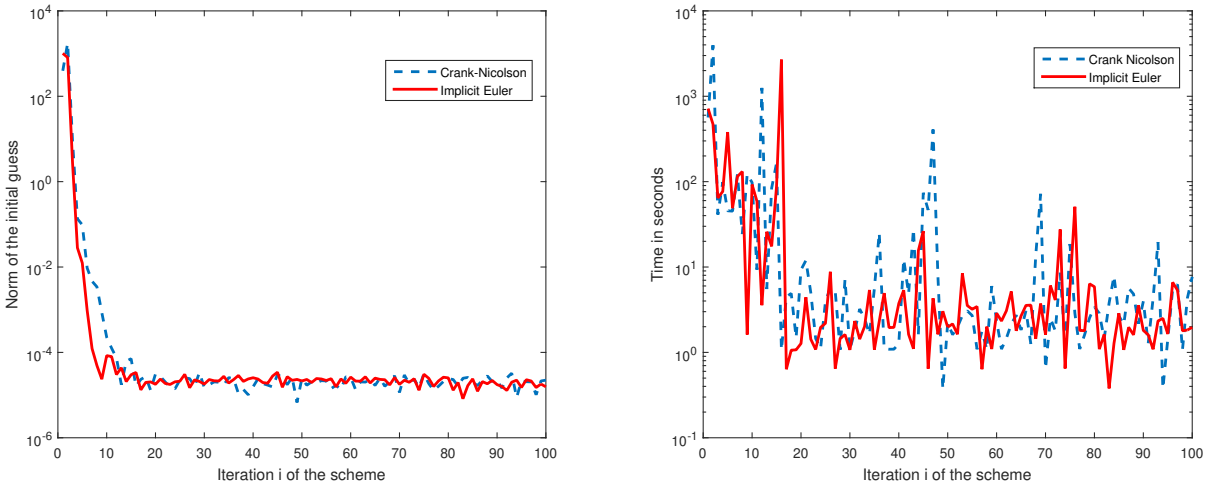

Figure 1. Residual norm of initial guess (left) and run time (right) - Algorithm 2 with line search.

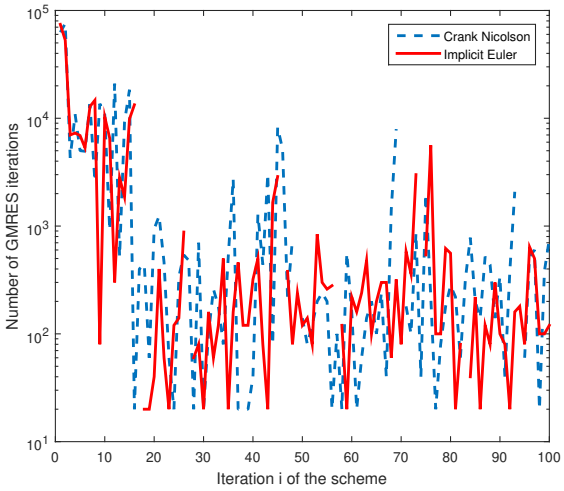

Figure 2. Number of GMRES iterations - Algorithm 2 with line search.

Tables 3 and 4 show the residual norm $\|G_i(x_i^{(k)} + \hat{s}_i^{(k)})\|$ of the approximate solution $\hat{s}_i^{(k)}$ and the residual norm $\|G_i(x_i^{(k+1)})\|$ using implicit Euler and Crank-Nicolson schemes with trust-region (hook step and dogleg step methods). The results are illustrated for some iterations k of the trust region algorithm. Figures 3 shows the nonlinear residual norm $\|G_i(x_i^{(0)})\|$ and Figure 4 shows the run time in seconds required at each iteration of the used schemes. Figure 5 shows the number of GMRES iterations required at each iteration of the scheme for computing x_i. Similar comments apply as in the case of line search.

Table 3. Algorithm 2 with trust region (implicit Euler is used).

iteration	Inexact Newton & dogleg step		Inexact Newton & hook step	
(i,k)	$\|\|\tilde{G}_i(x_i^{(k)} + \hat{s}_i^{(k)})\|\|$	$\|\|\tilde{G}_i(x_i^{(k+1)})\|\|$	$\|\|\tilde{G}_i(x_i^{(k)} + \hat{s}_i^{(k)})\|\|$	$\|\|\tilde{G}_i(x_i^{(k+1)})\|\|$
(0,0)	9.3906×10^2	9.1889	9.3906×10^2	9.1889
(0,1)	9.3906×10^2	9.2035×10^{-2}	9.3906×10^2	9.2031×10^{-2}
(0,2)	9.3906×10^2	9.2035×10^{-4}	9.3906×10^2	9.2031×10^{-4}
(0,3)	9.3906×10^2	9.9958×10^{-6}	9.3906×10^2	9.9873×10^{-6}
(1,0)	6.0649×10^2	9.4309	6.0649×10^2	9.4310
(1,1)	6.0649×10^2	9.4096×10^{-2}	6.0649×10^2	9.4097×10^{-2}
(1,2)	6.0649×10^2	9.4087×10^{-4}	6.0649×10^2	9.4086×10^{-4}
(1,3)	6.0649×10^2	9.9998×10^{-6}	6.0649×10^2	9.9790×10^{-6}
(20,0)	7.7952×10^{-5}	7.7952×10^{-5}	8.3291×10^{-5}	8.3291×10^{-5}
(20,1)	7.7952×10^{-5}	9.9559×10^{-6}	8.3292×10^{-5}	9.9779×10^{-6}
(75,0)	2.9284×10^{-5}	2.9284×10^{-5}	5.3112×10^{-5}	5.3112×10^{-5}
(75,1)	2.1296×10^{-5}	9.9642×10^{-6}	4.0032×10^{-5}	9.9035×10^{-6}
(99,0)	4.532×10^{-5}	4.5320×10^{-5}	8.7240×10^{-5}	8.7240×10^{-5}
(99,1)	4.3849×10^{-5}	9.9973×10^{-6}	7.1705×10^{-5}	9.9775×10^{-6}

Table 4. Algorithm 2 with trust region (Crank Nicolson is used).

iteration	Inexact Newton & dogleg step		Inexact Newton & hook step	
(i,k)	$\|\|\tilde{G}_i(x_i^{(k)} + \hat{s}_i^{(k)})\|\|$	$\|\|\tilde{G}_i(x_i^{(k+1)})\|\|$	$\|\|\tilde{G}_i(x_i^{(k)} + \hat{s}_i^{(k)})\|\|$	$\|\|\tilde{G}_i(x_i^{(k+1)})\|\|$
(0,0)	384.2226	3.7436	3.8422×10^2	3.7437
(0,1)	384.2226	3.7425×10^{-2}	3.8422×10^2	3.7420×10^{-2}
(0,2)	384.2226	3.7424×10^{-4}	3.8422×10^2	3.7420×10^{-4}
(0,3)	384.2226	9.985×10^{-6}	3.8422×10^2	9.9966×10^{-6}
(1,0)	1.0915×10^3	1.1607×10^1	1.0915×10^3	1.1606×10^1
(1,1)	1.0915×10^3	1.1623×10^{-1}	1.0915×10^3	1.1611×10^{-1}
(1,2)	1.0915×10^3	1.1623×10^{-3}	1.0915×10^3	1.1611×10^{-3}
(1,3)	1.0915×10^3	1.1624×10^{-5}	1.0915×10^3	1.1613×10^{-5}
(1,4)	1.0915×10^3	9.9837×10^{-6}	1.0915×10^3	9.9936×10^{-6}
(20,0)	1.9537×10^{-4}	1.9537×10^{-4}	1.9529×10^{-4}	1.9529×10^{-4}
(20,1)	1.39537×10^{-4}	9.9821×10^{-6}	1.9529×10^{-4}	9.9962×10^{-6}
(75,0)	4.6947×10^{-5}	4.6947×10^{-5}	4.0956×10^{-5}	4.0956×10^{-5}
(75,1)	4.641×10^{-5}	9.9951×10^{-6}	3.9277×10^{-5}	9.6282×10^{-6}
(99,0)	5.0128×10^{-5}	5.0128×10^{-5}	6.4960×10^{-5}	6.4960×10^{-5}
(99,1)	4.9969×10^{-5}	9.8921×10^{-6}	6.3409×10^{-5}	9.9078×10^{-6}

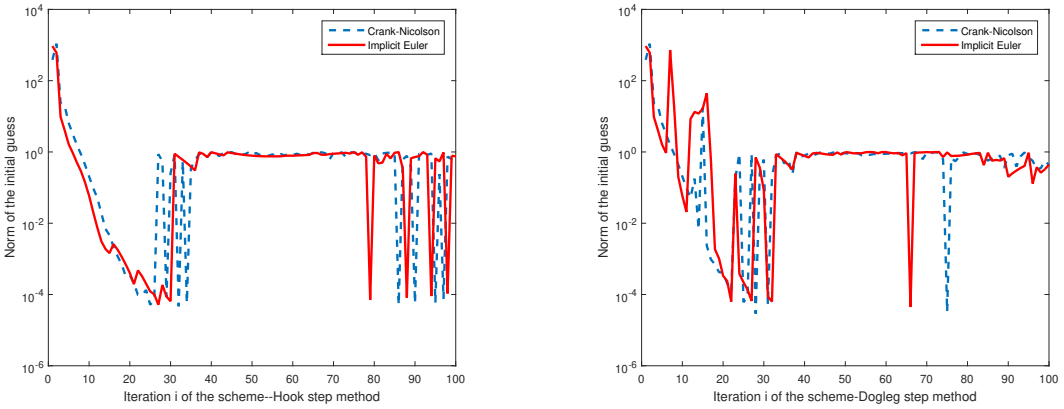

Figure 3. Residual norm of initial guess - Algorithm 2 with trust-region.

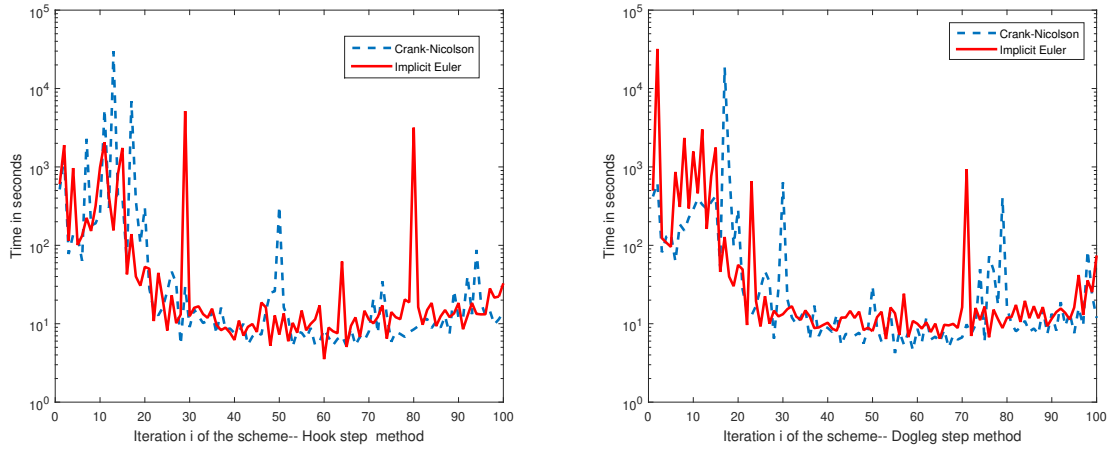

Figure 4. Run time - Algorithm 2 with trust-region.

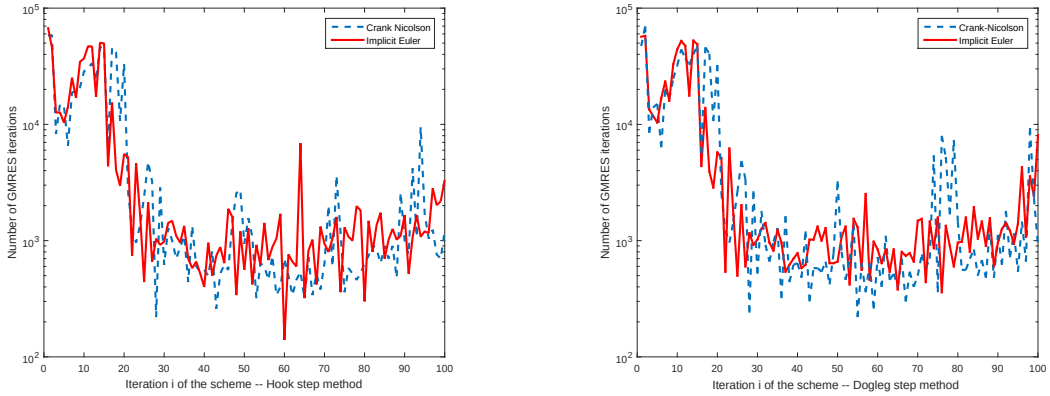

Figure 5. Number of iterations required for each iteration of the scheme - Algorithm 2 with trust-region.

6. Conclusion

We have shown theoretically and numerically that the subspace $\mathcal{V}_i$ (see (2.7), (2.11)) leads to a significant acceleration of the nonlinear systems that occur when solving large systems of nonlinear DAEs by a general class of implicit schemes. Our take focused only on finding good initial solutions for these systems. For further acceleration, a suitable preconditioning technique should be used in step 18 of algorithm 2. The potential user can adapt the present work to more specific schemes.

Conflict of interest

The authors declare that there is no conflict of interest in this paper.

References

1. U. M. Ascher, L. R. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*, SIAM, Philadelphia, PA, 1998.
2. M. Al Sayed Ali, M. Sadkane, *Acceleration of implicit schemes for large systems of nonlinear ODEs*, Electron. T. Numer. Anal., **35** (2009), 104–117.
3. M. Al Aayed Ali, M. Sadkane, *Improved predictor schemes for large systems of linear ODEs*, Electron. T. Numer. Anal., **39** (2012), 253–270.
4. H. Asgharzadeh, I. Borazjani, *A Newton-Krylov method with an approximate analytical Jacobian for implicit solution of Navier-Stokes equations on staggered overset-curvilinear grids with immersed boundaries*, J. Comput. Phys., **331** (2017), 227–256.
5. K. E. Brenan, S. L. Campbell, L. R. Petzold, *Numerical solution of initial-value problems in differential-algebraic equations*, SIAM, Philadelphia, PA, 1989.
6. P. N. Brown and Y. Saad, *Convergence theory of nonlinear Newton-Krylov algorithms*, SIAM J. Optimiz., **4** (1994), 297–330.
7. P. N. Brown, A. C. Hindmarsh, L. R. Petzold, *Using Krylov Methods in the solution of Large-Scale Differential-Algebraic Systems*, SIAM J. Sci. Comput., **15** (1994), 1467–1488.
8. M. Burger, M. Gerdts, *A survey on numerical methods for the simulation of initial value problems with DAEs*. In: Ilchmann A., Reis T. (eds) Surveys in differential-algebraic equations IV, Differ.-Algebr. Equ. Forum, Springer, Cham, (2017), 221–300.
9. P. J. Davis, *Interpolation and approximation*, Blaisdell, New York, 1963.
10. R. S. Dembo, S. C. Eisenstat and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., **19** (1982), 400–408.
11. J. E. Dennis and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, SIAM, Philadelphia, PA, 1996.
12. S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optimiz., **4** (1994), 393–422.
13. C. T. Kelley, *Iterative methods for optimization*, Frontiers in Applied Mathematics, 1999.

14. P. Kunkel and V. Mehrmann, *Differential-Algebraic Equations: Analysis and Numerical Solution*, EMS Publishing House Zurich, 2006.

15. R. Lamour, R. März, E. Weinmüler, *Boundary value problems for differential-algebraic equations: a survey*. In: Surveys in differential-algebraic equations III, Differ.-Algebr. Equ. Forum, Springer, Cham, (2015), 177–309.

16. H. Podhaisky, R. Weiner and B. A. Schmitt, *Numerical experiments with Krylov integrators*, Appl. Numer. Math., **28** (1998), 413–425.

17. Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

AIMS Press