Lehrstuhl für
Prozesstechnik
Prof. Dr.-Ing. W. Marquardt
RWTH Aachen

Technical Report LPT–progress

Sensitivity Analysis of Linearly–implicit Differential–algebraic Systems by One–step Extrapolation

M. Schlegel[1], W. Marquardt[1], R. Ehrig[2], U. Nowak[2]

November 2002

Submitted to:
Applied Numerical Mathematics, 2002.

[1] Lehrstuhl für Prozesstechnik
RWTH Aachen, D–52056 Aachen

[2] Konrad–Zuse–Institut für Informationstechnik
Takustr. 7, D–14195 Berlin

Enquiries should be addressed to:

Lehrstuhl für Prozesstechnik
RWTH Aachen
Templergraben 55
D–52056 Aachen

Tel.: +49 / 241 / 80 94668
Fax: +49 / 241 / 80 92326
E–Mail: secretary@lfpt.rwth–aachen.de

Sensitivity analysis of linearly-implicit differential-algebraic systems by one-step extrapolation

Martin Schlegel [a], Wolfgang Marquardt [a,*], Rainald Ehrig [b], Ulrich Nowak [b]

[a] *Lehrstuhl für Prozesstechnik, RWTH Aachen, D-52056 Aachen, Germany*

[b] *Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, D-14195 Berlin, Germany*

Abstract

In this work we present an approach for the sensitivity analysis of linearly-implicit differential-algebraic equation systems. Solutions for both, states and sensitivities are obtained by applying an extrapolated linearly implicit Euler discretization scheme. This approach is compared to the widely used sensitivity extensions of multi-step BDF methods by means of case studies. Especially, we point out the benefit of this method in the context of dynamic optimization using the sequential approach.

Key words: sensitivity evaluation, differential-algebraic systems, extrapolation methods, dynamic optimization, optimal control

1 Introduction

Sensitivity analysis of differential-algebraic equation systems (DAEs) plays an important role in many engineering and science disciplines. Applications include parameter estimation, optimization, process sensitivity studies, model reduction and experimental design [12,17]. Consequently, there is the need for algorithms which perform sensitivity computations in an efficient and rapid manner.

* Corresponding author: marquardt@lfpt.rwth-aachen.de

A common way of obtaining sensitivity information for dynamic systems is the numerical integration of the sensitivity equation systems, which emerge from a differentiation of the DAE system with respect to the parameters of interest. Various well established techniques make use of the fact that the sensitivity integration depends on the solution of the DAE system. The algorithms differ in the way they reuse information from the DAE integration for the sensitivity integration. Many of the existing algorithms for a combined state and sensitivity integration have been developed as extensions of the multi-step BDF code DASSL or its variants. Caracotsios and Stewart [5] have suggested a scheme in which the linear systems for the sensitivity corrector steps are solved directly after convergence of the nonlinear corrector step. Later, this method has been termed the *staggered direct* approach. Maly and Petzold [17] have proposed a *simultaneous corrector* method, where the DAE and sensitivity systems are solved simultaneously. A *staggered corrector* method was developed by Feehery et al. [12]. Here, instead of solving the linear systems arising in the sensitivity integration directly, they are solved by a Newton iteration which uses the same iteration matrix as in the preceding state integration. Comparisons of these different BDF based methods are given in [12,16].

For specific applications, such as dynamic optimization, also other methods for sensitivity evaluation are available. Alternatives to the integration of the sensitivity equations are backward integration of the adjoint system (e.g. [4]), and perturbation approaches (e.g. [20] and references cited therein).

In this paper, we consider a one-step extrapolation method based on the linearly implicit Euler discretization as an alternative to the BDF based methods for the integration of the combined state and sensitivity system. The basic method has been developed and implemented as the code LIMEX for DAE integration by Deuflhard et al. [9,8]. This paper describes an extension of LIMEX for sensitivity evaluation and compares its computational performance with the established BDF based techniques.

Besides looking into the use for pure sensitivity evaluation we also consider the special application in the context of dynamic optimization using the so-called sequential approach [2,14,21]. The extrapolation-based sensitivity integration appears to be especially beneficial for this class of application.

2 Problem formulation

In this work, we consider a system of linearly-implicit differential-algebraic equations (DAEs) of the form

$$B(t,y)\,\dot{y} = f(t,y,p) \tag{1}$$

$$y(t_0) = y_0 \tag{2}$$

where $y \in \mathbb{R}^{n_y}$ are the states and $p \in \mathbb{R}^{n_p}$ denotes the parameters of the original DAE system. In principle, the matrix B may also depend on the parameters p. However, in order to simplify the presentation, we have dropped this possible dependency. The matrix B may be singular, but the DAE system is assumed to have a differential index up to one. $y_0 \in \mathbb{R}^{n_y}$ are consistent initial conditions. For ease of notation the arguments t, y and p are sometimes left out in the following. Sensitivity analysis aims at finding the derivatives of the states y with respect to the parameters p. Differentiating (1) with respect to each parameter p_i yields additional n_p sensitivity equation systems of the form

$$B\,\dot{s}_i = \left(\frac{\partial f}{\partial y} - \Gamma\right) s_i + \frac{\partial f}{\partial p_i}\,, \qquad i = 1,\ldots,n_p\,, \tag{3}$$

where $s_i = \frac{dy}{dp_i}$ denotes the sensitivity of y with respect to the parameter p_i. The symbol Γ has been introduced as

$$\Gamma_{i,l} := \sum_{j=1}^{n_y} \frac{\partial B_{i,j}}{\partial y_l}\,\dot{y}_j\,, \qquad i = 1,\ldots,n_y\,, \quad l = 1,\ldots,n_y\,, \tag{4}$$

for simpler notation. The sensitivity systems (3) for each parameter p_i are DAEs themselves, which are independent from each other, but obviously depending on the solution of the state system (1). Two special properties of the sensitivity systems have to be mentioned: They are linear in s_i, and, with respect to s_i, they possess the same Jacobian matrix as the state system. The algorithms available for combined state and sensitivity integration [5,12,17] all exploit these special properties in some way.

In the following, we first take a look at the solution of the state system itself, before the sensitivity evaluation is addressed.

3 Linearly implicit extrapolation for state integration

This section summarizes the derivation of the linearly-implicit extrapolation algorithm. The reader is referred to [9,7,8,11] for more details.

Applying the linearly-implicit Euler discretization with a stepsize h to the DAE system (1) yields

$$y_{k+1} = y_k + (B_k - hA_0)^{-1} h f(y_k, p), \tag{5}$$

where

$$A = \frac{\partial}{\partial y}(f - B\dot{y}) = \left(\frac{\partial f}{\partial y} - \Gamma\right) \tag{6}$$

is the Jacobian matrix of the residual of (1) and

$$A_0 = A(t, y)|_{t=t_0} \ .$$

It is important to note that the matrix A_0 could also be replaced by an approximation as long as the relevant parts are retained [18,10].

If we denote the iteration matrix with $M := (A - \frac{B_k}{h})$, we can also write

$$y_{k+1} = y_k - M_0^{-1} f(y_k, p). \tag{7}$$

The expression (7) (or (5)) corresponds to one Newton iteration for the nonlinear system which arises in the implicit Euler discretization [8].

The extrapolation method allows to construct approximations of higher order based on this discretization. If H denotes a basic stepsize, approximations $T_{j,1}$ for $y(t_0+H)$ can be obtained by using the described discretizations with stepsizes $h_j = H/n_j, j = 1, \ldots, j_{max}$. The harmonic sequence $\{n_j\} = \{1, 2, 3, \ldots\}$ has proven to be a reliable choice. The extrapolation tableau

$$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{n_j/n_{j-k+1} - 1}, \qquad k = 1, \ldots, j \tag{8}$$

then defines higher order approximations $T_{j,k}$. The so-called subdiagonal differences $\varepsilon_j = ||T_{j,j} - T_{j,j-1}||$ are taken as error estimates. $T_{j,j}$ can be accepted as an approximation for $y(t_0+H)$, if the error estimate is less then a predefined tolerance.

As mentioned above, this algorithm has been implemented into the code LIMEX, including an advanced adaptive stepsize and order control mechanism developed by Deuflhard [6]. The extensions for sensitivity evaluation, which are presented in the following section, have been implemented based on LIMEX.

A simplified algorithmic structure of the linearly-implicit extrapolation code for one basic integration step H looks as follows [11]:

Algorithm 1: State integration

```
Compute Jacobian $A_0 = \frac{\partial}{\partial y}(f(y_0,p) - B\dot{y})$
for $j = 1,\dots,j_{max}$ while convergence criterion not satisfied
    $h_j = H/n_j$
    for $k = 0,\dots,j-1$
        $y_{k+1} = y_k + (B_k - h_j A_0)^{-1} h_j f(y_k,p)$
    $T_{j,1} = y_j$
if $j > 1$ compute $T_{j,j}$ and check convergence
$y_{new} = T_{j,j}$
```

In order to make the solution of the linear system in the innermost loop most efficient, an iterative procedure has been suggested [9,7,11], which utilizes the fact that in most real problems the matrix B contains only relatively few non-constant entries. In case of a constant matrix B this is not required, rather a standard LU factorization can be used.

4 Extensions for sensitivity evaluation

4.1 Simultaneous state and sensitivity integration

For extending the linearly-implicit extrapolation algorithm for sensitivity evaluation we consider a combined system of the original DAE system (1) and the associated sensitivity systems (3):

$$\begin{bmatrix} B & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{bmatrix} \cdot \begin{bmatrix} \dot{y} \\ \dot{s}_1 \\ \vdots \\ \dot{s}_{n_p} \end{bmatrix} = \begin{bmatrix} f(t,y,p) \\ \left(\frac{\partial f}{\partial y} - \Gamma\right) s_1 + \frac{\partial f}{\partial p_1} \\ \vdots \\ \left(\frac{\partial f}{\partial y} - \Gamma\right) s_{n_p} + \frac{\partial f}{\partial p_{n_p}} \end{bmatrix}. \qquad (9)$$

By using (6), introducing the combined vector $Y := [y, s_1, \dots, s_{n_p}]$ and defining

$$\mathbf{B} := \begin{bmatrix} B & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{bmatrix}, \qquad \mathbf{f} := \begin{bmatrix} f(t,y,p) \\ As_1 + \frac{\partial f}{\partial p_1} \\ \vdots \\ As_{n_p} + \frac{\partial f}{\partial p_{n_p}} \end{bmatrix},$$

we can rewrite (9) as

$$\mathbf{B}\dot{Y} = \mathbf{f}(t, Y, p)\,. \tag{11}$$

We assume that consistent initial conditions for this system are provided. $s_{i,0}$ and $\dot{s}_{i,0}$ can be calculated from solving (3) at t_0, using the knowledge of y_0 and the fact that $s_{k,i,0} = \frac{dy_k}{dp_i}(t_0) = 0$ for those y_k, which have been chosen as dynamic degrees of freedom by fixing their initial conditions in (2). In principal, the combined system (11) then could be integrated using the method described in the previous section.

If we write the residual Jacobian matrix (6) for the combined system (9), we obtain

$$\mathbf{A} = \frac{\partial}{\partial Y}\left(\begin{bmatrix} f(t,y,p) \\ As_1 + \frac{\partial f}{\partial p_1} \\ \vdots \\ As_{n_p} + \frac{\partial f}{\partial p_{n_p}} \end{bmatrix} - \begin{bmatrix} B & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{bmatrix} \cdot \begin{bmatrix} \dot{y} \\ \dot{s}_1 \\ \vdots \\ \dot{s}_{n_p} \end{bmatrix}\right). \tag{12}$$

This can be reformulated as

$$\mathbf{A} = \frac{\partial}{\partial Y}\begin{bmatrix} f(t,y,p) - B\dot{y} \\ As_1 + \frac{\partial f}{\partial p_1} - B\dot{s}_1 \\ \vdots \\ As_{n_p} + \frac{\partial f}{\partial p_{n_p}} - B\dot{s}_{n_p} \end{bmatrix} = \begin{bmatrix} A & & & \\ A_1 & A & & \\ \vdots & & \ddots & \\ A_{n_p} & & & A \end{bmatrix}, \tag{13}$$

where $A_i = \frac{\partial}{\partial y}\left(As_i + \frac{\partial f}{\partial p_i} - B\dot{s}_i\right)$. This leads to the following expression for the iteration matrix $\mathbf{M}$ of (11):

$$\mathbf{M} = \mathbf{A} - \frac{\mathbf{B}}{h} = \begin{bmatrix} A - \frac{B}{h} & & & \\ A_1 & A - \frac{B}{h} & & \\ \vdots & & \ddots & \\ A_{n_p} & & & A - \frac{B}{h} \end{bmatrix} \tag{14}$$

Consequently, the extension of (5) for sensitivity evaluation becomes

$$Y_{k+1} = Y_k - \mathbf{M}_0^{-1}\mathbf{f}(Y_k, p). \tag{15}$$

An implementation based on this would require expensive factorizations of the matrix $\mathbf{M}$. However, by replacing $\mathbf{M}$ by its block-diagonal part

$$\hat{\mathbf{M}} = \begin{bmatrix} M & & & \\ & M & & \\ & & \ddots & \\ & & & M \end{bmatrix} = \begin{bmatrix} A - \frac{B}{h} & & & \\ & A - \frac{B}{h} & & \\ & & \ddots & \\ & & & A - \frac{B}{h} \end{bmatrix} \tag{16}$$

the problem can be simplified significantly. In [17] a similar strategy has been used to simplify the iteration matrix required by Newton's method applied to the system obtained by the implicit Euler discretization. Here, using $\hat{\mathbf{M}}$ instead of $\mathbf{M}$ can be seen as a result of approximating $\mathbf{A}$ by its block-diagonal part. As mentioned before, the linear-implicit extrapolation algorithm allows such an approximation, as long as the characteristics of the dynamic system are preserved [10,18]. The method is a well-defined one-step method even for any choice of the iteration matrix M or $\mathbf{M}$, respectively. Our numerical experiments (cf. Section 5) have shown that the choice of $\hat{\mathbf{M}}$ leads to very satisfactory results.

Since the blocks M in the main diagonal of $\hat{\mathbf{M}}$ are identical, the factorizations $LU = M$, which are required for the solution of the DAE states anyway, can be reused for the integration of the sensitivities. This is also indicated in the algorithmic structure for the extended extrapolation code:

Algorithm 2: Simultaneous state and sensitivity integration

```
Compute Jacobian $A_0 = \frac{\partial}{\partial y}(f(y_0, p) - B\dot{y})$
for $j = 1, \dots, j_{max}$ while convergence criterion not satisfied
    $h_j = H/n_j$
    $LU = A_0 - \frac{B}{h_j}$
    for $k = 0, \dots, j-1$
        $y_{k+1} = y_k - (LU)^{-1} f(y_k, p)$
        $s_{i,k+1} = s_{i,k} - (LU)^{-1} \left( A(y_k) s_{i,k} + \frac{\partial f}{\partial p_i}(y_k) \right), \quad i = 1, \dots, n_p$
    $T_{j,1} = Y_j$
if $j > 1$ compute $T_{j,j}$ and check convergence
$Y_{new} = T_{j,j}$
```

Note that the above algorithm has been written for the case $B = const.$ for simplicity. However, the procedure is not restricted to this case, rather the same iterative method mentioned above (see [9,11]) can be applied.

From the algorithm it follows that the additional effort required for sensitivity evaluation as compared to pure state integration comprises n_p additional evaluations of the sensitivity residuals and n_p additional back substitutions with the already available $(LU)^{-1}$. Also, the effort required for the extrapolation itself will be n_p times larger.

Since in this approach the states and sensitivities are integrated simultaneously, the algorithm can be seen as an analog to the simultaneous corrector extension for the BDF code DDASSL/DDASPK presented by Maly and Petzold [17].

An important issue is the error control applied to the sensitivities. In [15] a partial error control has been suggested which excludes the sensitivities from the error tests and only controls the error on the DAE states. This partial error control has been reported to possibly speed up computations without significant loss of accuracy. With the same reasoning, the extension to LIMEX described above has been equipped with the option to switch off the error control for the sensitivities. Using partial error control could be confirmed to be a reasonable approach by numerical experiments.

4.2 Staggered state and sensitivity integration

Algorithm 2 has a disadvantage in the case, where no convergence could be achieved at j_{max}. In such cases the size of the basic stepsize H is reduced and the entire extrapolation procedure has to be repeated. Since the sensitivities depend on the results from the state integration, calculating them before a potential stepsize reduction would waste computational resources. However, the potential need for a stepsize reduction cannot be known beforehand.

Therefore, a modification of *Algorithm 2* is conceivable. As suggested in [12] for the BDF-based approaches, a staggered approach can also be used here. The idea is to first converge the solution of y for the stepsize H (including potential stepsize reductions) and then the sensitivities.

The algorithmic structure (again written for the case $B = const.$) looks as follows:

Algorithm 3: Staggered state and sensitivity integration

```
    Compute Jacobian $A_0 = \frac{\partial}{\partial y}(f(y_0, p) - B\dot{y})$
    for $j = 1, \ldots, j_{max}$ while convergence criterion not satisfied
        $h_j = H/n_j$
        $LU_j = A_0 - \frac{B}{h_j}$
        for $k = 0, \ldots, j-1$
            $y_{k+1} = y_k - (LU_j)^{-1} f(y_k, p)$
        $T^y_{j,1} = y_j$
    if $j > 1$ compute $T^y_{j,j}$ and check convergence
    $y_H = T^y_{j,j}$
    for $j = 1, \ldots, j^{sens}_{max}$ while convergence criterion not satisfied
        for $k = 0, \ldots, j-1$
            $s_{i,k+1} = s_{i,k} - (LU_j)^{-1} \left( A(y_k) s_{i,k} + \frac{\partial f}{\partial p_i}(y_k) \right), \quad i = 1, \ldots, n_p$
        $T^s_{j,1} = s_{i,j}$
    if $j > 1$ compute $T^s_{j,j}$ and check convergence
    $s_{i,H} = T^s_{j,j}$
```

This algorithm would overcome the problem of superfluous sensitivity evaluations in the case of stepsize reductions. However, a major drawback of this modification is that the LU decompositions for different h_j, which are used during the state integration, have to be stored intermediately for later reuse during the sensitivity integration. The same holds for the intermediate state vectors y_k.

If the initial stepsize H for the integration is chosen properly, in practical applications of LIMEX there is usually no need for frequent stepsize reductions. Therefore the expected benefit of *Algorithm 3* is limited and probably does not justify the large bookkeeping and storage effort required for the intermediate LU decompositions and state vectors. Therefore, an implementation of *Algorithm 3* has not been considered, so far.

4.3 Sensitivity integration with numerical differentiation

In cases where the derivatives $\frac{\partial f}{\partial p_i}$ are not available (or hard to implement) one may use an alternative to the direct solution of the sensitivity equation (3), either with *Algorithm 2* or *Algorithm 3*. However, the classical numerical differentiation approach, i.e., perturbing p_i, repeating the state integration, and forming the difference approximation

$$s_i \approx \frac{y(p_i + \delta p_i) - y(p_i)}{\delta p_i} \,, \tag{17}$$

is known to be inefficient and inaccurate, cf. [13]. Therefore, one should use the so-called internal numerical differentiation. In this approach the derivatives $\frac{\partial f}{\partial p_i}$ in the discretization are replaced by finite difference approximations. This leads, e.g. for *Algorithm 2*, to the discretization

$$s_{i,k+1} = s_{i,k} - (LU)^{-1} \left(A(y_k) s_{i,k} + \frac{\tilde{f}_k - f_k}{\Delta p_i} \right) \tag{18}$$

where

$$f_k = f(y_k, p_i), \quad \tilde{f}_k = f(y_k + s_{i,k} \Delta p_i, p + \Delta p_i) \,. \tag{19}$$

In contrast to the classical numerical differentiation approach, where the perturbation should be chosen according to $\delta p_i \approx \sqrt{tol}$ (tol = prescribed integration tolerance), the second approach allows the choice $\Delta p_i \approx \sqrt{epmach}$ ($epmach$ = relative machine precision). In both cases the error in the sensitivities is known to be proportional to the magnitude of the perturbation.

5 Case studies

In this section we investigate the algorithmic performance of the proposed algorithm by means of four case studies. The sensitivity-extended version of LIMEX (*Algorithm 2*) has been compared with DDASPK [15]. In DDASPK, the three previously mentioned different algorithmic variants for the BDF based sensitivity computation are implemented: the simultaneous corrector method [17], the staggered corrector method [12] and the staggered direct method [5]. In the case studies, the following abbreviations will be used:

LIMEX	simultaneous (*Algorithm 2*)	LSIM
DDASPK	sim. corrector	DSIC
	stag. corrector	DSTC
	stag. direct	DSTD

5.1 Gas-Oil Cracking Problem

The Gas-Oil Cracking Problem is a two dimensional ODE, which has been used as test example e.g. in [12,17]. The problem formulation results in

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} -(p_1+p_3)y_1^2 \\ p_1 y_1^2 - p_2 y_2 \end{pmatrix}, \qquad \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The problem has three parameters with respect to which the sensitivities are to be calculated. The values are $p_1 = 0.9875$, $p_2 = 0.2566$, $p_3 = 0.3323$. The combined state and sensitivity system (9) for this example contains $2\cdot(3+1) = 8$ equations and reads as

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{s}_1 \\ \dot{s}_2 \\ \dot{s}_3 \\ \dot{s}_4 \\ \dot{s}_5 \\ \dot{s}_6 \end{pmatrix} = \begin{pmatrix} -(p_1+p_3)y_1^2 \\ p_1 y_1^2 - p_2 y_2 \\ -2s_1(p_1+p_3)y_1 - y_1^2 \\ 2s_1 p_1 y_1 - s_2 p_2 + y_1^2 \\ -2s_3(p_1+p_3)y_1 \\ 2s_3 p_1 y_1 - s_4 p_2 - y_2 \\ -2s_5(p_1+p_3)y_1 - y_1^2 \\ 2s_5 p_1 y_1 - p_2 s_6 \end{pmatrix}, \qquad \begin{pmatrix} y_1(0) \\ y_2(0) \\ s_1(0) \\ s_2(0) \\ s_3(0) \\ s_4(0) \\ s_5(0) \\ s_6(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

This problem has been solved using LIMEX and DDASPK with its three variants. The comparison has been done for different error tolerances. Since in this problem scaling is not an issue, the relative and absolute tolerances for all states and sensitivities have been set to the same value. The error control is enabled for both, state and sensitivity values.

Table 1 shows computational statistics for this problem. Hereby, the number of function evaluations refers to the evaluation of $f(t, y, p)$, whereas the number of sensitivity evaluations counts the computations of $As_i + \frac{\partial f}{\partial p_i}$. Jacobian evaluations are required not only for the iteration matrix of the integration, but also for the evaluation of the sensitivity residuals. Therefore, in the Table we show separate numbers for those due to iteration matrices and for the total number of Jacobian evaluations. Finally, the number of LU decompositions and back substitutions in the linear algebra are reported. Due to the small size of the problem, computation times are negligible and almost not comparable and therefore not reported, here.

Table 1
Gas-Oil Cracking Problem – computational statistics

	function eval.	sensitiv. eval.	Jacob. (iter.)	Jacob. (tot.)	LU decomp.	back subst.
tolerance: 10^{-5}						
LSIM	81	81	7	88	34	432
DSIC	93	93	14	107	14	372
DSTC	143	64	14	78	14	271
DSTD	141	64	64	128	64	269
tolerance: 10^{-6}						
LSIM	125	125	9	134	48	656
DSIC	139	139	16	155	16	556
DSTC	220	100	17	117	17	420
DSTD	191	87	87	174	87	365
tolerance: 10^{-7}						
LSIM	183	183	11	194	65	948
DSIC	149	149	26	175	26	596
DSTC	295	136	29	165	29	567
DSTD	271	122	124	246	124	515

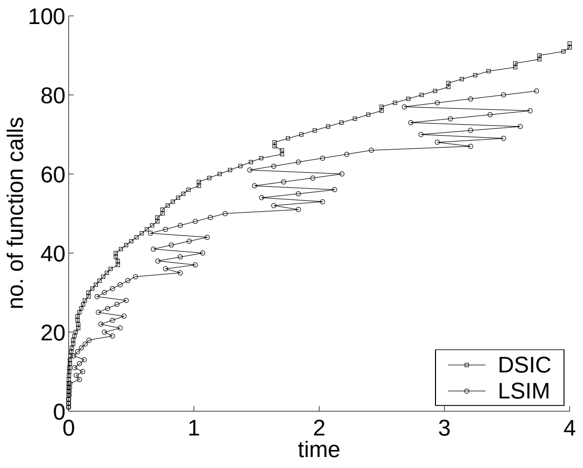


Fig. 1. No. of function calls vs. integration time for tolerance = 10^{-5}

The LSIM algorithm could solve this problem successfully. A graphical representation of the solution trajectories for states and sensitivities is identical to those presented in [17] and is therefore not included in this paper.

Looking on the statistics, various observations can be made. LSIM always needs much fewer Jacobian matrix evaluations for the integration iteration, a typical feature of an extrapolation algorithm. LSIM also always needs less function evaluations but more sensitivity evaluations than DSTC and DSTD. Due to sensitivity evaluations, LSIM needs in total more Jacobian calls than DSTC, but less than DSTD. Comparing LSIM and DSIC, which are most comparable, it can be stated that for lower tolerances LSIM needs less function, sensitivity and total Jacobian calls. For the tolerance 10^{-7}, DSIC needs fewer calls to all these tasks than LSIM. Finally, LSIM needs always significantly more LU decompositions than DSIC and DSTD, but less than DSTD. LSIM needs more back substitutions than all other methods.

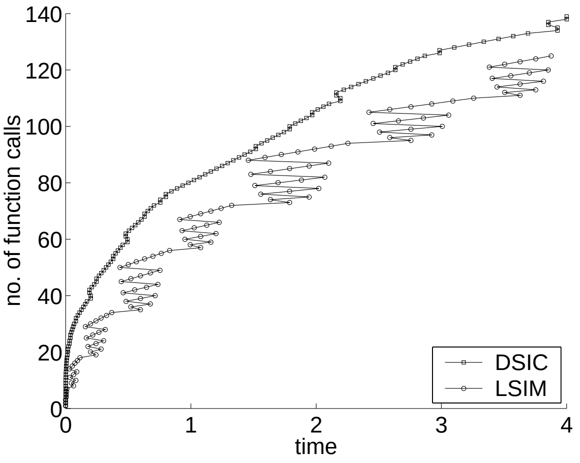


Fig. 2. No. of function calls vs. integration time for tolerance = 10^{-6}

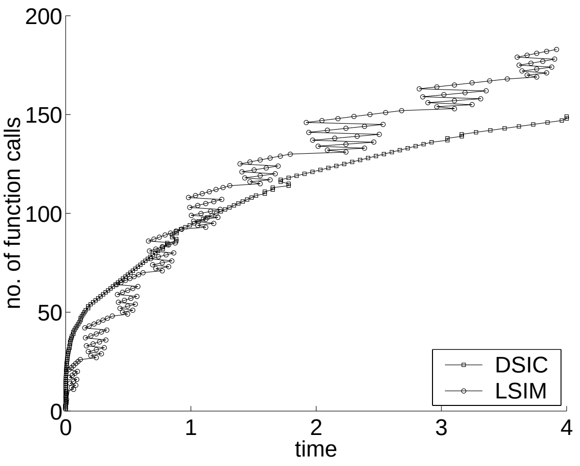


Fig. 3. No. of function calls vs. integration time for tolerance = 10^{-7}

Figures (1),(2) and (3) show the number of function calls done by the integrator versus the integration time for each function call, plotted for LSIM and DSIC for the different integration tolerances. Since both are simultaneous approaches, this number also corresponds to the number of sensitivity evaluations (including the sensitivity part of the Jacobian evaluations).

The curves for LSIM nicely show the extrapolation behavior, i.e. that for each large stepsize H the functions are evaluated on an increasingly finer mesh corresponding to the small stepsizes h_j. Also, it can be seen that for tolerances of 10^{-5} and 10^{-6} the curve for LSIM lies below the one for DSIC, which means that for reaching a certain time point within integration tolerance, fewer function calls are required. For 10^{-7}, LSIM is finally higher than DSIC, which corresponds to the results reported above. Interesting in Figure (3) is the fact, that in the initial phase of the integration up to a time of about 0.8, LSIM needs fewer function evaluations, but then crosses the line for DSIC. This is due to the fact that DSIC, being a multi-step algorithm, initially needs more steps to build up higher order information, but then runs quite smoothly, where as the one-step extrapolation code does not have this start-up behavior, but also does not benefit like a multi-step integrator in the later part of the integration horizon. This fact becomes especially interesting in the context of dynamic optimization, where many but short integration intervals have to be solved, as we will show in subsection 5.3.

5.2 Batch-Reactor Problem

The Batch-Reactor Problem is a very stiff DAE system, which has been used as test example e.g. in [5,17]. The model equations are

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \\ \dot{y}_5 \\ \dot{y}_6 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -p_3 y_2 y_8 \\ -p_1 y_2 y_6 + p_2 y_{10} - p_3 y_2 y_8 \\ p_3 y_2 y_8 + p_4 y_4 y_6 - p_5 y_9 \\ -p_4 y_4 y_6 + p_5 y_9 \\ p_1 y_2 y_6 - p_2 y_{10} \\ -p_1 y_2 y_6 - p_4 y_4 y_6 + p_2 y_{10} + p_5 y_9 \\ -0.0131 + y_6 + y_8 + y_9 + y_{10} - y_7 \\ p_7 y_1 - y_8(p_7 + y_7) \\ p_8 y_3 - y_9(p_8 + y_7) \\ p_6 y_5 - y_{10}(p_6 + y_7) \end{pmatrix}$$

with the following initial conditions and parameter values:

$$\begin{pmatrix} y_1(0) \\ y_2(0) \\ y_3(0) \\ y_4(0) \\ y_5(0) \\ y_6(0) \\ y_7(0) \\ y_8(0) \\ y_9(0) \\ y_{10}(0) \end{pmatrix} = \begin{pmatrix} 1.5776 \\ 8.32 \\ 0 \\ 0 \\ 0 \\ 0.0131 \\ 0.79735 \cdot 10^{-5} \\ 0.79735 \cdot 10^{-5} \\ 0 \\ 0 \end{pmatrix}, \qquad \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \end{pmatrix} = \begin{pmatrix} 21.893 \\ 2.14 \cdot 10^{9} \\ 32.318 \\ 21.893 \\ 1.07 \cdot 10^{9} \\ 7.65 \cdot 10^{-18} \\ 4.03 \cdot 10^{-11} \\ 5.32 \cdot 10^{-18} \end{pmatrix}.$$

For this example, the same test studies as for the Gas-Oil Problem have been carried out. Because DDASPK failed for high tolerances, if the full error control on states and sensitivities is applied, the computations have been done using partial error control on the state variables only. Since within the Newton iteration of DDASPK an error check is always applied to all variables, the tolerances for the sensitivities have to be carefully chosen. Proper scaling is an important issue for this problem, because the problem parameters vary quite significantly in order of magnitude. The absolute tolerances for the sensitivities have been set according to the rule $ATOL_{j,i} = RTOL_{j,i}/O(p_i)$, where $RTOL_{j,i}$ has been set to same value for all components j of all sensitivities s_i. The LSIM approach appeared to be less sensitive to proper scaling.

This stiff DAE problem could be solved successfully by the LSIM algorithm. When the sensitivities are plotted as in [5,17], i.e. the sensitivity values have been multiplied by the corresponding parameter value in order to get the results in a comparable scale, then the same graphical representation is obtained as in those references. Therefore, these plots are not shown in this paper.

The computational statistics for this case study are shown in Table 2. Since the solution for this problem is numerically more involved than in the previous problem, here also the computation time in CPU seconds is given.

Again, various observations can be made by evaluating the results given in Table 2. The computation times have to be interpreted carefully, because DDASPK uses a dense solver for the linear algebra, whereas the LIMEX version used for this test contains a sparse solver. Since the problem size is very small, the performance of sparse solvers will not be significantly different than

Table 2
Batch-Reactor Problem – computational statistics

	function eval.	sensitiv. eval.	Jacob. (iter.)	Jacob. (tot.)	LU decomp.	back subst.	CPU sec.
tolerance: 10^{-5}							
LSIM	234	234	20	254	113	2943	0.169
DSIC	264	264	43	307	43	2376	0.231
DSTC	425	196	34	230	34	1797	0.195
DSTD	238	104	104	208	104	966	0.124
tolerance: 10^{-6}							
LSIM	360	360	28	388	166	4482	0.272
DSIC	390	390	47	437	47	3510	0.340
DSTC	684	323	37	360	37	2945	0.323
DSTD	343	152	153	305	153	1407	0.182
tolerance: 10^{-7}							
LSIM	591	591	38	629	246	7191	0.430
DSIC	502	502	52	554	52	4518	0.445
DSTC	963	465	48	513	48	4218	0.461
DSTD	491	217	221	438	221	2010	0.263

the one of dense solvers. Due to this uncertainties a comparison of the given CPU times is not rigorous. Nevertheless, it shows that the performance of the LSIM implementation is comparable to the BDF based approaches. DSTD is clearly fastest for this problem.

The observations regarding the various tasks called by the integration routines are similar to the Gas-Oil Example. Again, for low tolerances, LSIM uses fewer function evaluations than the other methods. It always needs far fewer Jacobian for the integration iteration, however not necessarily a smaller total number Jacobian calls, due to the sensitivity evaluation. Again, LSIM requires more LU decompositions and back substitutions than the other approaches.

The Batch-Reactor Problem has been also used to test the numerical integration approach described in Section 4.3. The problem has been solved for different prescribed tolerances, while comparing the accuracy ϵ of the sensitivities at the final integration point using a highly accurate reference solution. Note that all tolerances and perturbations have to be understood as properly scaled values. In Table 3 these results and the results when doing this experiment with the direct solution approach are collected. Obviously, the

Table 3
Batch-Reactor Problem - accuracy of sensitivities

tol	$\epsilon_{num.diff.}$	ϵ_{direct}
10^{-3}	4.8×10^{-5}	4.8×10^{-5}
10^{-4}	2.8×10^{-6}	2.7×10^{-6}
10^{-5}	1.3×10^{-6}	1.4×10^{-6}
10^{-6}	4.1×10^{-7}	3.0×10^{-7}
10^{-7}	4.7×10^{-7}	1.2×10^{-8}

internal numerical differentiation gives very satisfactory results until the expected limit, $tol \leq \Delta p_i$, $\Delta p_i = 10^{-7}$, is reached. Concerning computing time, the required computing times for the both approaches are nearly identical for this example.

5.3 Application to dynamic optimization

An interesting application of the one-step extrapolation method for state and sensitivity integration is dynamic optimization. A detailed summary of the theory, problem formulation and possible solution approaches for dynamic optimization problems is beyond the scope of this paper. The reader is referred to the relevant literature.

One important approach to dynamic optimization is the so-called sequential approach [2,14,21], where the control variables u_i are approximated by piecewise defined discretizations, such as

$$u_i(t) \approx \sum_{k \in \Lambda_i} c_{i,k} \phi_{i,k}(t). \tag{24}$$

Here, $c_{i,k}$ are degrees of freedom for a nonlinear optimization problem (NLP) solver. $\phi_{i,k}(t)$ defines the discretization functions, for example piecewise constant or linear splines. The resolution of the control profiles depends on the chosen discretization grid Λ_i. The differential-algebraic model equations have to be solved repeatedly as function evaluations for the NLP solver, and, in order to provide the gradient information for the optimizer, sensitivities of the model states with respect to $c_{i,k}$ have to be computed, as well. The sensitivity evaluation belongs to the most expensive tasks of solving dynamic optimization problems with the sequential approach.

It is a known fact that one-step methods such as extrapolation have an inherent advantage over BDF methods when applied to DAEs with frequent discontinuities [3,10]. This is in line with the observed advantage of the LSIM algorithm in the startup phase of the integration (cf. subsection 5.1).

Due to the discretization of the control variables, the number of parameters for the sensitivity integration increases at each mesh point of Λ_i. Therefore, by running through the time horizon the integrator has to cope with a growing combined state and sensitivity system. Also, the integration has to be restarted at each mesh point, because discontinuities in the states and/or their derivatives might occur. In this context, the aforementioned advantage of the extrapolation method, also when applied to sensitivity equations, becomes especially important.

To underline these aspects, we consider the Crane Example Problem [1]. The problem formulation is as follows:

$$
\begin{array}{rl}
 & \min\limits_{u_1,u_2} \Phi(t_f) \\
\text{s.t.} & \\
 & \dot{y}_1 = y_4 \\
 & \dot{y}_2 = y_5 \\
 & \dot{y}_3 = y_6 \\
\text{ODE system} & \dot{y}_4 = u_1 + 17.2656\, y_3 \\
 & \dot{y}_5 = u_2 \\
 & \dot{y}_6 = -1.0/y_2(u_1 + 27.0756\, y_3 + 2\, y_5 y_6) \\
 & \dot{\Phi} = 0.5\,(y_3^2 + y_6^2 + 0.01\,(u_1^2 + u_2^2)) \\
\text{initial conditions} & y(0.0) = [0.0, 22.0, 0.0, 0.0, -1.0, 0.0]^T \\
\text{endpoint constraints} & y(t_f) = [10.0, 14.0, 0.0, 2.5, 0.0, 0.0]^T \\
\text{state path constraints} & |y_4(t)| \le 2.5, \quad |y_5(t)| \le 1 \\
\text{control constraints} & |u_1(t)| \le 2.83374, \quad -0.80865 \le u_2(t) \le 0.71265 \\
\text{final time} & t_f = 9.0
\end{array}
$$

This is a typical dynamic optimization problem involving two control variables (u_1,u_2), an ODE system of dimension 7, and various path and endpoint constraints on state and control variables. In our case study, this problem has been solved using the dynamic optimization software ADOPT [19], which optionally uses LIMEX or DDASPK in the variants described earlier.

Results for a discretization of u_1 and u_2 with 8 equidistant piecewise constant functions for integration tolerances of 10^{-5}, 10^{-6} and 10^{-7} are reported in

Table 4. The control profiles for this case are shown in the Appendix. It was verified that each optimization run required the same number of calls to the integrator. For example, in the case of an 8 interval discretization, 6 integrator calls were required.

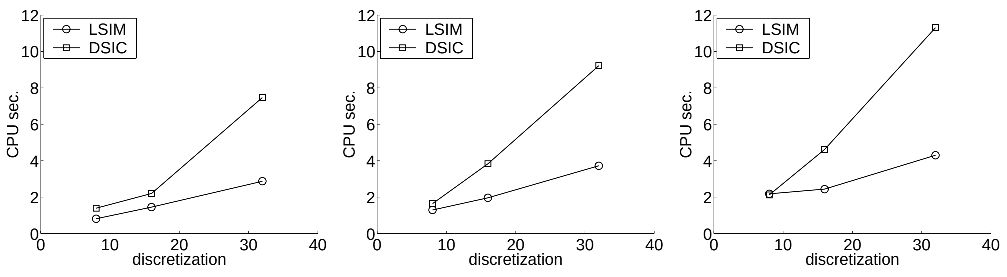
The results in Table 4 show an advantage of the LSIM algorithm as compared to the other methods. As observed in the previous examples, the relative performance of LSIM is better for low integration tolerances.

Table 4
Crane dynamic optimization problem – computational statistics for *8* discretization intervals

	function eval.	sensitiv. eval.	Jacob. (iter.)	Jacob. (tot.)	CPU sec.
tolerance: 10^{-5}					
LSIM	1510	1510	96	1606	0.81
DSIC	1845	1845	683	2528	1.39
DSTC	1715	1654	680	2334	1.73
DSTD	1694	1412	1369	2781	1.89
tolerance: 10^{-6}					
LSIM	2408	2408	125	2533	1.29
DSIC	2285	2285	835	3120	1.64
DSTC	2286	2142	812	2954	2.54
DSTD	2097	1768	1720	3488	2.43
tolerance: 10^{-7}					
LSIM	4408	4408	221	4629	2.18
DSIC	2935	2935	954	3889	2.12
DSTC	2818	2676	943	3619	2.89
DSTD	2635	2241	2193	4434	3.03

In order to investigate the effect of the control discretization grid, the problem has been solved also for control discretizations of 16 and 32 intervals. Since the results in Table 4 indicate that DSIC performs best among the DDASPK variants on this problem, all further comparisons have been done using LSIM and DSIC, only.

Tables 5 and 6 show the results for 16 and 32 intervals, respectively. The corresponding optimal control profiles are displayed in the Appendix. It can



(a) 10^{-5} (b) 10^{-6} (c) 10^{-7}

Fig. 4. CPU time vs. discretization for crane example for different tolerances

be seen that the relative performance of LSIM as compared to DSIC increases significantly with an increasingly refined mesh. Since the integration intervals become shorter, the BDF approach can exploit the multi-step advantages less and less and loses performance in the start-up phase. This is also illustrated by Figure 4, which shows plots of the CPU time versus the Λ_i-discretization for different tolerances. It can be concluded that the integration of sensitivity equations using one-step extrapolation is usually more efficient than BDF based approaches for an application in dynamic optimization.

Table 5
Crane dynamic optimization problem – computational statistics for *16* discretization intervals

	function eval.	sensitiv. eval.	Jacob. (iter.)	Jacob. (tot.)	CPU sec.
tolerance: 10^{-5}					
LSIM	2292	2292	224	2516	1.45
DSIC	3404	3404	1568	4972	2.20
tolerance: 10^{-6}					
LSIM	3154	3254	224	3478	1.96
DSIC	4470	4470	1904	6374	3.83
tolerance: 10^{-7}					
LSIM	3782	3782	224	4006	2.44
DSIC	5337	5337	2194	7531	4.62

Table 6
Crane dynamic optimization problem – computational statistics for *32* discretization intervals

	function eval.	sensitiv. eval.	Jacob. (iter.)	Jacob. (tot.)	CPU sec.
tolerance: 10^{-5}					
LSIM	3527	3527	512	4039	2.87
DSIC	6444	6444	3501	9945	7.47
tolerance: 10^{-6}					
LSIM	4728	4728	512	5240	3.72
DSIC	8548	8548	4126	12674	9.22
tolerance: 10^{-7}					
LSIM	5747	5747	512	6259	4.30
DSIC	10322	10322	4940	15262	11.31

6 Conclusions

In this paper, an alternative approach for combined state and sensitivity integration of differential-algebraic equations systems has been suggested. The method uses one-step extrapolation of the linearly implicit Euler discretization. It could be demonstrated with various case studies that this approach is a clear alternative to the widely used BDF based approaches. Our method is restricted to the class of linearly-implicit differential-algebraic equation systems. Potential savings can be expected especially in cases, where only mild error tolerances are required, and in those cases, where many discontinuous integration problems have to be solved, as it has been shown for dynamic optimization. Furthermore, it is possible to apply extensions of LIMEX such as dynamic sparsing of the iteration matrix [18] and parallelization [11] in the framework of sensitivity computation.

The extrapolation strategy for the sensitivities allows additional flexibility with respect to the desired accuracy of the sensitivities. For applications, which do not require sensitivities up to the same accuracy as the states, a separate error control for the sensitivities offers a simple way to adjust their accuracy. This is because the depth of the extrapolation tableau influences the approximation order of the variables [8]. Therefore it is possible to exclude sensitivities from further extrapolation as soon as their error criterion is met. This allows to save a significant amount of computation time in those cases, where the requirements on the accuracy of the sensitivities are modest, because many sensitivity residual (and thereby Jacobian) evaluations as well as back substitutions can be saved.

Acknowledgments

This work has been partially funded by the European Commission under grant G1RD-CT-1999-00146 in the "INCOOP" project.

References

[1] D. Augustin and H. Maurer. Second order sufficient conditions and sensitivity analysis for the optimal control of a container crane under state constraints. Technical Report 00–17, Westfälische Wilhelms-Universität, Münster, 2000.

[2] J.T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming.* SIAM, Philadelphia, 2001.

[3] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* Number 14 in Classics in Applied Mathematics. SIAM, Philadelphia, 1996.

[4] A.E. Bryson and Y.-C. Ho. *Applied Optimal Control.* Taylor & Francis, 1975.

[5] M. Caracotsios and W.E. Stewart. Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations. *Comp. Chem. Eng.*, 9(4):359–365, 1985.

[6] P. Deuflhard. Order and stepsize control in extrapolation methods. *Numerische Mathematik*, (41):399–422, 1983.

[7] P. Deuflhard and F. Bornemann. *Numerische Mathematik II.* De Gruyter, Berlin, 1994.

[8] P. Deuflhard, E. Hairer, and J. Zugck. One–step and extrapolation methods for differential–algebraic systems. *Numerische Mathematik*, (51):501–516, 1987.

[9] P. Deuflhard and U. Nowak. Extrapolation integrators for quasilinear implicit ODEs. In Deuflhard, P., Engquist, B., editor, *Large Scale Scientific Computing. Progress in Scientific Computing,* 7:37–50, 1987.

[10] P. Deuflhard, U. Nowak, and M. Wulkow. Recent developments in chemical computing. *Comp. Chem. Eng.*, 14(11):1249–1258, 1990.

[11] R. Ehrig, U. Nowak, and P. Deuflhard. Highly scalable parallel linearly–implicit extrapolation algorithms. Technical Report TR 96–11, Konrad–Zuse–Zentrum für Informationstechnik, Berlin, 1996.

[12] W. Feehery, J. Tolsma, and P.I. Barton. Efficient sensitivity analysis of large–scale differential–algebraic systems. *Appl. Numer. Math.*, 25:41–54, 1997.

[13] E. Hairer, S.P. Norsett, G. Wanner, *Solving Ordinary Differential Equations I, Nonstiff Problems*, Springer Series in Computational Mathematics **8**, (2nd rev. ed.), Springer-Verlag, Berlin, 1993.

[14] D. Kraft. On converting optimal control problems into nonlinear programming problems. *Comput. Math. Prog.*, 15:261–280, 1985.

[15] S. Li and L. Petzold. Design of new DDASPK for sensitivity analysis. Technical report, Dept. of Mech. and Env. Eng., University of California, Santa Barbara, 1999.

[16] S. Li, L. Petzold, and W. Zhu. Sensitivity analysis of differential-algebraic equations: A comparison of methods on a special problem. *Appl. Num. Math.*, 32:161–174, 2000.

[17] T. Maly and L. Petzold. Numerical methods and software for sensitivity analysis of differential–algebraic systems. *Appl. Numer. Math.*, 20, 1996.

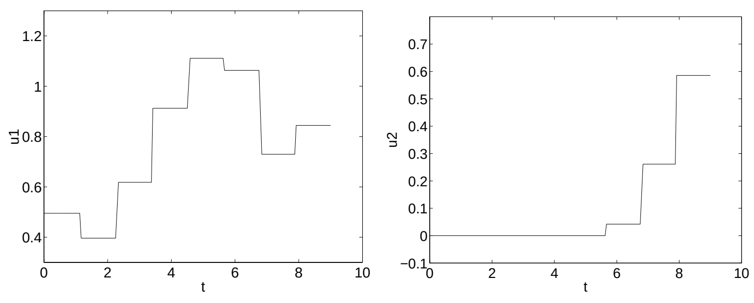
[18] U. Nowak. Dynamic sparsing in stiff extrapolation methods. *IMPACT Comput. Sci. Engrg.*, 5:53–74, 1993.

[19] M. Schlegel, Th. Binder, A. Cruse, J. Oldenburg, and W. Marquardt. Component–based implementation of a dynamic optimization algorithm using adaptive parameterization. In R. Gani and S.B. Jørgensen, editors, *European Symposium on Computer Aided Process Engineering – 11*, pages 1071–1076. Elsevier, 2001.

[20] S. Støren and T. Hertzberg. Obtaining sensitivity information in dynamic optimization problems solved by the sequential approach. *Comp. Chem. Eng.*, 23:807–819, 1999.

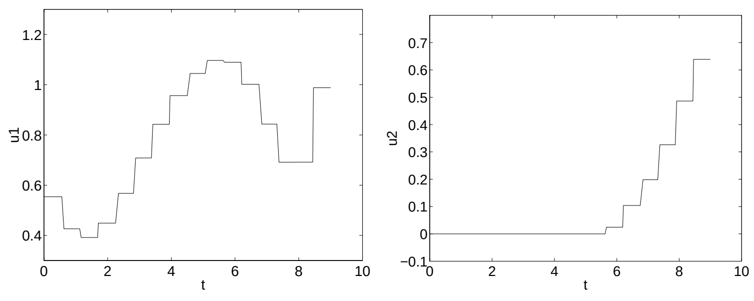
[21] V.S. Vassiliadis, R.W.H. Sargent, and C.C. Pantelides. Solution of a class of multistage dynamic optimization problems: 1. problems without path constraints. *Ind. Eng. Chem. Res.*, 33(9):2111–2122, 1994.

A Appendix: Solution plots for Crane dynamic optimization problem



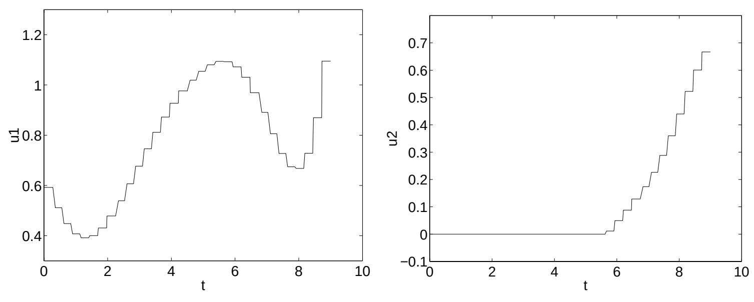
(a) $u1$ (b) $u2$

Fig. A.1. Optimal control profiles for 8 discretization intervals.



(a) $u1$ (b) $u2$

Fig. A.2. Optimal control profiles for 16 discretization intervals.



(a) $u1$ (b) $u2$

Fig. A.3. Optimal control profiles for 32 discretization intervals.