Consistent Initialization for Nonlinear Index-2 Differential-Algebraic Equation: Large Sparse Systems in MATLAB

by

Michael Hanke
Royal Institute of Technology
Department of Numerical Analysis and Computer Science
René Lamour
Humboldt-University of Berlin
Institute of Mathematics

Parallel and Scientific Computing Institute

Royal Institute of Technology
and
Uppsala University

Report No. 2001:05

October 22, 2001

Consistent Initialization for Nonlinear Index-2 Differential-Algebraic Equation: Large Sparse Systems in MATLAB

Michael Hanke*
Royal Institute of Technology
Department of Numerical Analysis and Computer Science
René Lamour
Humboldt-University of Berlin
Institute of Mathematics

October 22, 2001

Abstract

An important component of any initial-value solver for higher-index differential-algebraic equations consists in the computation of consistent initial values. In a recent paper [5], an algorithm is proposed which is applicable to a very general class of index-2 systems. Unfortunately, the computational expense is rather high. We present a modification of this approach, which gives rise to a MATLAB implementation capable of handling systems of moderate dimension (several thousands of unknowns). The algorithm is illustrated by examples.

Keywords: differential-algebraic equations, consistent initial values, consistent initialization, method of lines, matlab

AMS MSC(2000): 65L80, 65L05, 65M20

1 Introduction

When solving initial value problems for quasi-linear differential-algebraic equations (dae's)

$$A(x,t)x' + b(x,t) = 0 \tag{1}$$

with a singular coefficient matrix A, the problem of a consistent initialization arises: Because of the singularity of A, derivative-free equations are contained in (1) giving rise to some constraints

*The work of this author was partially supported by the National Network in Applied Mathematics (NTM).

on the initial value x^0 in the initial condition

$$x(t_0) = x^0 \tag{2}$$

in order to ensure the solvability of the initial value problem (1)-(2). Moreover, if (1) represents a so-called higher-index problem,we have additional hidden constraints, which can be obtained explicitly if certain parts of (1) are differentiated. In order to provide a consistent initial value x^0 in (2), a proper description of the hidden constraints becomes necessary. To this end, a number of algorithms have been proposed (e.g. [12, 10, 8, 9, 2, 5]). The most general one is that of Estévez Schwarz and Lamour [5]. Since it makes ample use of a number of projectors, its realization requires either a numerical computation of them (thus restricting the dimension of (1) to small or moderate values) or an analytic expression for them (which is the case for some systems describing electrical networks, for example [6]).

The present paper is motivated by two questions: Is it possible to provide a reliable and efficient implementation of the algorithm proposed in [5] for systems (1) arising within the method of lines? In that case, the dimension of (1) is large but A and the Jacobian matrix of b are sparse matrices. Secondly, is it possible to implement the algorithm within MATLAB® [13]? The latter has important algorithmic consequences.

The paper is organized as follows. In Section 2 we will summarize the algorithm of [5]. Section 3 contains the new implementation. Some examples are presented in Section 4.

2 An algorithm for computing consistent initial values for index-2 dae's

Let us consider the dae

$$A(x,t)x' + b(x,t) = 0 \tag{3}$$

consisting of n equations and $x(t) \in \mathbb{R}^n$. For simplicity we assume that all appearing derivatives exist and that the partial derivatives with respect to x and x' are continuous. We are interested in the case that $A(x,t)$ is singular and assume that

$$\textbf{A1}: \quad N := \ker A(x,t) = \text{const}, \ \operatorname{im} A(x,t) = \text{const}.$$

Let Q be a projector of $\mathbb{R}^n$ onto N, $P = I - Q$, and let W_0 be a projector of $\mathbb{R}^n$ onto a complement of $\operatorname{im} A(x,t)$ with $W_0 A(x,t) \equiv 0$.

In order to derive a system for determining consistent initial conditions the notion of the tractability index [7, 11] is used. Let

$$B(x',x,t) := \frac{\partial}{\partial x}\{A(x,t)x' + b(x,t)\} = \left[A(x,t)x'\right]_x + b_x(x,t).$$

Here, the subindex denotes partial derivatives. From (3) it follows that all solutions lie in

$$M_0(t) := \{z \in \mathbb{R}^n \,|\, W_0 b_x(x,t) = 0\}.$$

The space S, which is closely related to the tangent space of $M_0(t)$, is given by

$$S(x,t) := \{z \in \mathbb{R}^n \mid W_0 B(x',x,t)z = 0\} = \{z \in \mathbb{R}^n \mid W_0 b(x,t)z = 0\}.$$

Eq. (3) with singular $A(x,t)$ is called tractable with index 1 [7] if and only if $N \cap S(x,t) = \{0\}$. The latter condition is equivalent to

$$G_1(x',x,t) := A(x,t) + B(x',x,t)Q$$

being nonsingular.

In the index-1 case, there exists a solution through x^0 for each point $x^0 \in M_0(t_0)$. As we are interested in the index-2 case, we consider the next matrix chain elements, which are given by G_1 and

$$B_1(x',x,t) := B(x',x,t)P.$$

Assume that

$$\textbf{A2}: \quad \operatorname{im} G_1(x',x,t) \text{ and } \ker G_1(x',x,t) \text{ do not depend on } x'.$$

Analogously to W_0, let $W_1(x,t)$ be a projector of $\mathbb{R}^n$ onto a complement of $\operatorname{im} G_1(x',x,t)$ with $W_1(x,t)G_1(x',x,t) \equiv 0$. On this level the relevant spaces are

$$\begin{aligned} S_1(x',x,t) &:= \{z \in \mathbb{R}^n \mid W_1(x,t)B_1(x',x,t)z = 0\}, \\ N_1(x,t) &:= \ker G_1(x',x,t). \end{aligned}$$

Let $Q_1(x,t)$ be a projector of $\mathbb{R}^n$ onto $N_1(x,t)$ and $P_1(x,t) = I - Q_1(x,t)$. Eq. (3) with a singular $A(x,t)$ is called tractable with index 2 [11] if and only if it is not tractable with index 1, $\dim N \cap S(x,t)$ is constant, and $N_1(x,t) \cap S_1(x',x,t) = \{0\}$. Similarly as above, the latter condition is equivalent to

$$G_2(x',x,t) := G_1(x',x,t) + B_1(x',x,t)Q_1(x,t)$$

being nonsingular.

It is important to note that the index definitions introduced above do not depend on the special choice of the different projectors.

For simplicity, we will drop the arguments in the following if there is no danger of ambiguity.

Assume now that (3) is tractable with index 2. In that case, $\mathbb{R}^n = N_1 \oplus S_1$ such that the *canonical projector* $Q_{1,s}$ onto N_1 with $Q_{1,s}S_1 \equiv 0$ is well defined. It holds that $Q_{1,s} = Q_1 G_2^{-1} B_1$ for any projector Q_1 onto N_1 [11]. Consequently, $Q_{1,s} G_2^{-1} B_1 = Q_{1,s}$. Therefore, $Q_1 := Q_{1,s}$ can be chosen from the beginning.

In contrast to the index-1 case, where $M_0(t)$ is filled by solutions, for an index-2 dae the so-called hidden constraints define the manifold

$$M_1(t) \subset M_0(t),$$

which fulfils the requirement that there exists a solution through x^0 for each point $x^0 \in M_1(t_0)$. These hidden constraints arise when differentiating a suitable part of (3). This part can properly be described with the aid of the projectors W_0, W_1. Indeed, let $I_{W_1} \in \mathbb{R}^{n \times n}$ be given by

$$\left(I_{W_1}\right)_{ij} := \begin{cases} 1, & i = j \text{ and } \exists j : (W_1)_{ij} \neq 0, \\ 0, & \text{else.} \end{cases}$$

Note that I_{W_1} is a projector and $W_1 I_{W_1} = W_1$. The equations to be differentiated are now given by $I_{W_1} W_0 b$. This gives rise to the assumption

$$\textbf{A3}: \quad \frac{d}{dt}\big\{I_{W_1} W_0 b(x(t),t)\big\} \text{ exists.}$$

The next assumption states a certain structural property:

$$\textbf{A4}: \quad N \cap S(x,t) \text{ depends on } t \text{ only.}$$

This assumption is fulfilled in many applications. Obviously, it holds true for linear (time-dependent) dae's. Similarly, it is easy to see that it holds for index-2 systems in Hessenberg form. For a further discussion see [4].

Let us now consider an initial condition for (3). An initial condition of the form (2) is *not* the correct one since $x^0 \in M_1(t_0)$ cannot be guaranteed. The number of degrees of freedom for x^0 is exactly $\dim M_1(t_0)$. This dimension can be described by PP_1, which fixes exactly the dynamic variables [11]. Hence, an adequate formulation of an initial condition is given by

$$PP_1(x(t_0),t_0)(x(t_0) - \alpha) = 0. \tag{4}$$

The computation of a consistent initial value amounts, therefore, to a determination of the missing values $(I - PP_1(x(t_0),t_0))x(t_0)$. In order to start an integration method it is convenient to compute $(Px)'(t_0)$ additionally.

Theorem 2.1 ([5]). *Assuming **A1–A4** to hold, $x = x(t_0)$ and $y = Px'(t_0)$ are consistent initial values for (3) fulfilling (4) if and only if the obey the system*

$$\begin{aligned}
A(x,t)y + b(x,t) &= 0,\\
PP_1(x,t)(x-\alpha) &= 0,\\
Qy &= 0,\\
W_1(x,t)\Big[\big(I_{W_1}W_0 b\big)_x(x,t)y + \big(I_{W_1}W_0 b\big)_t(x,t)\Big] &= 0.
\end{aligned} \tag{5}$$

For guaranteeing locally unique solvability, the assumption

$$\textbf{A5}: \quad \big\{PP_1(x,t)(x-\alpha)\big\}_x (I - PQ_1(x,t))z = 0 \Rightarrow PP_1(x,t)z = 0$$

is required.

Theorem 2.2 ([5]). *Assume that **A1–A5** hold and let $Q_1 \equiv Q_{1,s}$ be the canonical projector. Then (5) has a full rank Jacobian matrix with respect to (y,x) in a neighbourhood of a solution.*

Obviously, **A5** is fulfilled if PP_1 depends only on t (e.g. for linear systems). For a discussion of **A5** see [5]. In the original paper, a slightly more general assumption instead of **A4** was used in the latter two theorems. The present version is much easier to verify in practice.

In certain applications, W_1 can be chosen to be constant or of a very simple structure. In those cases, (5) is advantageous even if the Jacobian matrix is rectangular and of a rather high dimension compared to that of (3). On the other hand, the four equations of system (5) can be combined into a system of two equations with a square Jacobian matrix if W_1 is chosen as $W_1 := G_2 P Q_1 G_2^{-1}$, $Q_1 = Q_{1,s}$. The price to be paid is that W_1 has complex dependencies on x now. Some rearrangements provide the system

$$\begin{aligned} A(x,t)y + b(x,t) &= 0, \\ PP_1(x,t)(x-\alpha) + PQ_1(x,t)[y + G_2^{-1}(y,x,t)(A_t(x,t)y + b_t(x,t))] + Qy &= 0. \end{aligned} \tag{6}$$

Because of the difficult dependences of P_1, Q_1, G_2 on (y,x), system (6) is very expensive to evaluate. Therefore, a simplification is proposed in [5]. Assuming that (6) has a solution (y_*,x_*), the system

$$\begin{aligned} A(x,t)y + b(x,t) &= 0, \\ PP_1(x_*,t)(x-\alpha) + PQ_1(x_*,t)[y + G_2^{-1}(y_*,x_*,t)(A_t(x,t)y + b_t(x,t))] + Qy &= 0 \end{aligned} \tag{7}$$

has (y_*,x_*) as a solution. The Jacobian matrix of (7) is easily computable. Moreover, its inverse is explicitly available. The solution of (6) is now accomplished by an inner-outer iteration process: For given (y_*^0,x_*^0) the projectors are computed and (7) is solved for (y,x). Then (y_*^0,x_*^0) is updated and the whole procedure is repeated until convergence. Convergence is ensured if the projectors vary slowly with (y,x)

3 A MATLAB implementation for large systems

The applicability of the approach of Section 2 in the case of sparse systems depends on two crucial facts: The possibility of an efficient computation and representation of the various projectors involved and its efficient realization using MATLAB's tools. This is the content of the present section. Occasionally, some algorithms are conveniently represented by MATLAB expressions. To do so we will use a Courier font.

The first step consists in forming the projectors P,, Q of the matrix chain. In our intended application within the method of lines, (3) is the spatial discretization of a time-dependent partial differential equation (pde). Therefore, the nullspace of the leading coefficient matrix $A(x,t)$ is structurally determined by the underlying pde: If a certain solution component of the pde does not appear time-differentiated, its discrete counterparts will share this property and vice versa. This will lead to our nullspace assumption:

B1: $\ker A(x,t)$ is spanned by a set of unit vectors e^i.

The indices i can easily be determined: e^i belongs to the nullspace of $A(x,t)$ if and only if the i-th column of $A(x,t)$ vanishes. It is natural to assume that this property holds independent of (x,t).

The assumption **B1** is uncritical as long as the pde is concerned. A problem may arise if the boundary conditions are time-dependent.

Once **B1** is given, the projectors P and Q can easily be represented by index vectors:

$$\begin{aligned} \texttt{q} &:= \{i |\ \text{column } i \text{ of } A(x,t) \text{ vanishes}\}, \\ \texttt{p} &:= \{1,\dots,n\} \setminus \texttt{q}. \end{aligned}$$

Let now A and B be given. In practice, these matrices can be computed either analytically or numerically without too much expense. Then G_1 and B_1 are given by

$$\begin{aligned} G_1 &= A + BQ = [\mathtt{A}(:,\mathtt{p}),\mathtt{B}(:,\mathtt{q})], \\ B_1 &= BP = [\mathtt{B}(:,\mathtt{p}),\mathtt{zeros}(\mathtt{n},\mathtt{length}(\mathtt{q}))]. \end{aligned}$$

Note that the above expressions do not contain any arithmetics.

The next step consists in a determining of the nullspace of G_1 and computing of a nullspace projector Q_1. The method of choice is a QR-decomposition of G_1. This method is only reliable if the columns of G_1 are equilibrated and the QR-decomposition is computed with pivotization. In order not to destroy the rank the equilibration must be carried out without any rounding errors. The equilibration can be accomplished by using the properties of the IEEE arithmetic[1]: Arithmetic operations with a result that can be exactly represented in the floating point representation are carried out without rounding errors, the floating point arithmetic uses a binary representation, and there are hardware instructions to extract the exponent of a floating point number. Moreover, MATLAB provides means for accessing the exponent extraction. Then the equilibration can be realized as follows: For every column, its Euclidean norm is evaluated. The dual exponent of its floating point representation is extracted and subtracted from every exponent of the respective column. The latter is a pure integer arithmetic such that there are no rounding errors. After that, all columns have a Euclidean norm between 1/2 and 2. The following MATLAB fragment realizes this approach:[2]

```
d1 = sqrt(sum(G1.^2));
[F,E] = log2(D1);
d = pow2(ones(size(E)),-E);
G1s = A*diag(d);
```

The QR-decomposition of the scaled matrix can now be computed by MATLAB's sparse `qr` command. Unfortunately, this command does not perform any pivotization. In our numerical experiments we observed that equilibration is really a must, but pivotization is not, such that the above algorithm is sufficient for our purposes.

Now let $G_1 P_c = UR$ with an orthogonal matrix U, an upper triangular matrix R, and a column permutation matrix P_c. If **B1** holds true, $\operatorname{rank} R = \operatorname{rank} G1 \geq \operatorname{rank} A$. Therefore, R has a block representation

$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & 0 \end{pmatrix}.$$

The matrices R_{11} and R_{22} are square and nonsingular. The dimension n_u of R_{11} is equal to the number of nonzero columns in A. The dimension of R_{22} may be zero, e.g. in the case of a

[1]Standard IEC 559:1989/IEEE 754-1985

[2]The MATLAB program is much simplified here in order to emphasize the point. Of course, some safeguards must be added. Moreover, a careful memory management is necessary.

Hessenberg system. At this point, the validity of assumption **B1** can easily be verified: The upper left $n_u \times n_u$ block of P_c must be the identity matrix.

For the sake of simplicity assume from now that $P_c = I$. From the representation of R, the nullspace of G_1 is given by

$$\ker G_1 = \{x = (u,v,w) \,|\, w \in \mathbb{R}^{n_w},\ v = -R_{22}^{-1}R_{23}w,\ u = -R_{11}^{-1}(R_{13} - R_{12}R_{22}^{-1}R_{23})w\}. \tag{8}$$

Therefore, a projector Q_1 onto $\ker G_1$ can easily be written down:

$$\begin{pmatrix} 0 & 0 & -R_{11}^{-1}(R_{13} - R_{12}R_{22}^{-1}R_{23}) \\ 0 & 0 & -R_{22}^{-1}R_{23} \\ 0 & 0 & I \end{pmatrix}.$$

In [5], this projector is now used to compute G_2 and $Q_{1,s}$. For larger problems, this approach is much too expensive because all matrices become full ones. Therefore, we take an alternative approach which bases on

Theorem 3.1. *Theorem 2.2 remains true if Q_1 is chosen such that $Q_1 Q = 0$.*

The proof can be given by a careful analysis of that of the original theorem. In order to derive a more compact version of (5) similarly to (6) choose $W_1 := G_2 P Q_1 G_2^{-1}$. We obtain the system

$$\begin{aligned} A(x,t)y + b(x,t) &= 0, \\ PP_1(x,t)(x-\alpha) + PQ_1(x,t)G_2^{-1}(y,x,t)[B(y,x,t)y + (A_t(x,t)y + b_t(x,t))] + Qy &= 0. \end{aligned} \tag{9}$$

If the arguments of the projectors and G_2, B are frozen to be (y_*, x_*), a nonlinear system with the Jacobian

$$J = \begin{pmatrix} A & B \\ PQ_1G_2^{-1}B + Q + PQ_1G_2^{-1}A_t & PP_1 + PQ_1G_2^{-1}B_t \end{pmatrix}$$

arises, where $B_t = b_{tx}(x,t)$. The inverse matrix of J can be computed explicitly:

$$J^{-1} = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix} \begin{pmatrix} G_2^{-1} & \\ & I \end{pmatrix}, \tag{10}$$

where

$$\begin{aligned} J_{11} &= PP_1 - PQ_1Z, \\ J_{21} &= I - PP_1 - QQ_1Z, \\ J_{12} &= I - PP_1 + (PQ_1Z - PP_1)Y - PQ_1G_2^{-1}B_tPP_1, \\ J_{22} &= PP_1 - (I - PP_1 - QQ_1)Y + QQ_1(I + ZY - G_2^{-1}B_tPP_1), \\ Z &= G_2^{-1}\{BPP_1 + [A_t + B_t(I - PP_1)]\}, \\ Y &= G_2^{-1}BPP_1. \end{aligned}$$

Let us now construct a nullspace projector Q_1 with $Q_1Q = 0$. If we use the same block structure for Q_1 as for R, we obtain

$$Q_1 = \begin{pmatrix} Q_{11} & 0 & 0 \\ Q_{12} & 0 & 0 \\ Q_{13} & 0 & 0 \end{pmatrix}.$$

Since $Q_1^2 = Q_1$,

$$Q_{11} = Q_{11}^2, Q_{21} = Q_{21}Q_{11}, Q_{31} = Q_{31}Q_{11}.$$

Let a vector $x = (u,v,w) \in \mathbb{R}^n$ be subdivided according to this block structure and denote $n_u := \dim u$ etc. According to (8), $\dim\ker G_1 = n_w$ and $\operatorname{rank} Q_{31} = n_w$. Hence, $n_w = \dim\ker G_1 = \dim\operatorname{im} Q_1 \geq \operatorname{rank} Q_{11}$. On the other hand, $\dim\operatorname{im} Q_1 = \dim\operatorname{im} Q_1^2 \leq \operatorname{rank} Q_{11} \leq n_u$ such that $\operatorname{rank} Q_{11} = n_w$ and $n_w \leq n_u$.

Again from (8) we see that, for $x = (u,v,w) \in \ker G_1$, it holds that $u = Q_{11}u = -R_{11}^{-1}(R_{13} - R_{12}R_{22}^{-1}R_{23})w$. Because of the restrictions of dimension, $T = -R_{11}^{-1}(R_{13} - R_{12}R_{22}^{-1}R_{23}) \in \mathbb{R}^{n_u \times n_w}$ has full (column) rank. The projector Q_{11} can be easily constructed by using an appropriate generalized inverse of T. This amounts to finding a matrix $S \in \mathbb{R}^{n_z \times n_u}$ such that ST is nonsingular. Once this is available, the various matrices are given by

$$Q_{11} = T(ST)^{-1}S, \quad Q_{21} = -R_{22}^{-1}R_{23}(ST)^{-1}S, \quad Q_{31} = (ST)^{-1}S.$$

There are a number of different possibilities for computing S. Among them are:

1. A straightforward approach is the use of the Moore-Penrose generalized inverse, which amounts to using $S = T^T$. This gives rise to a rather fully occupied projector Q_1.

2. Another idea consists in using an LU-decomposition. Let $T = P_r^T LU$ be an LU-decomposition of T with a permutation matrix P_r, a square upper triangular factor U and an $n_u \times n_w$ lower triangular factor L. Let L be subdivided into

$$L = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix}$$

with a square nonsingular submatrix L_1. Then define

$$S = \begin{pmatrix} U^{-1}L_1^{-1} & \vdots & 0 \end{pmatrix} P_r.$$

This choice of S leads to $ST = I$, which gives rise to

$$\begin{aligned} Q_{31} &= S, \\ Q_{21} &= -R_{22}^{-1}R_{23}S, \\ Q_{11} &= P_r^T \begin{pmatrix} I & 0 \\ L_2L_1^{-1} & 0 \end{pmatrix} P_r. \end{aligned}$$

3. A compromise between the above possibilities is the use of a QR-decomposition. Let $TP_c = VR$ with a column permutation matrix P_c, an orthogonal matrix V, and an upper triangular matrix R. Let those matrices be subdivided into

$$R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}, V = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}.$$

Then define

$$S = P_c R_1^{-1} \begin{pmatrix} V_{11}^T & \vdots & V_{21}^T \end{pmatrix}.$$

As in the previous case, $ST = I$ such that

$$\begin{aligned} Q_{31} &= S, \\ Q_{21} &= -R_{22}^{-1} R_{23} S, \\ Q_{11} &= \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} \begin{pmatrix} V_{11}^T & \vdots & V_{21}^T \end{pmatrix}. \end{aligned}$$

Note that the amount of computations can be reduced if MATLAB's *economy-size QR-factorization* is used.

It should be emphasized at this point that the sparseness of Q_1 decides about the efficiency of the overall method. Numerical experiments have shown that the second alternative is preferable with respect to computation time and memory consumption. We did not experience any stability problems.

With the projector Q_1 computed so far, G_2 can be computed. Instead of inverting this matrix, an LU-decomposition is sufficient. It does not make sense to form the Jacobian matrix J or its inverse (10) explicitly. In order to apply the Newton method it suffices to be able to compute $J^{-1}f$ for any given vector f. And this can be done using the information available so far.

The present approach uses the range projector $W_1 = G_2 P Q_1 G_2^{-1}$ with complicated dependences on its arguments. Therefore, one might expect to obtain a more easily solvable system by using (5) directly with some simple projectors W_0, W_1. Unfortunately, this is not true if one does not have further information on the systems. Even for providing W_0 a property similar to that used in deriving Q does not hold in general. Instead, we are required to use a QR-decomposition of A even at this stage. The resulting orthogonal factor U must be used for computing W_0. This leads to an unacceptable amount of computations and memory requirements.

A nice property of the algorithm is that at various stages, the algorithmic assumptions can be verified rather easily. This concerns the validity of **B1** as well as the decision on the index of the problem at hand.

4 Examples

The algorithm was implemented in MATLAB using the user interface of MATLAB's odesuite and its sparse mode of operation. Derivatives with respect to (y,x) were computed analytically (if available) or numerically using `numjac`. Derivatives with respect to t are always computed by numerical differentiation. The numerical differentiation procedures attain a rather high accuracy. All computations were performed on PC (AMD K7/700) under Linux.

The first example serves as a proof of concept. We choose the trajectory prescribed path control problem used in [5] (see also [3]) for demonstration purposes. The equations read

$$\begin{aligned}
\dot{H} &= V_R \sin\gamma \\
\dot{\zeta} &= V_R \frac{\cos\gamma}{r\cos\lambda} \sin A \\
\dot{\lambda} &= \frac{V_R}{r} \cos\gamma \cos A \\
\dot{V_R} &= -\frac{d}{m} - g\sin\gamma - \Omega^2 r\cos\lambda(\sin\lambda\cos A\cos\gamma - \cos\lambda\sin\gamma) \\
\dot{\gamma} &= l\frac{cos\beta}{mV_R} + \frac{\cos\gamma}{V_R}(\frac{V_R^2}{r} - g) + 2\Omega\cos\lambda\sin A \\
&\quad + \frac{\Omega^2}{V_R} r\cos\lambda(\sin\lambda\cos A\sin\gamma + \cos\lambda\cos\gamma) \\
\dot{A} &= \frac{l}{mV_R\cos\gamma}\sin\beta + \frac{V_R}{r}\cos\gamma\sin A\tan\lambda \\
&\quad - 2\Omega(\cos\lambda\cos A\tan\gamma - \sin\lambda) + \frac{\Omega^2 r}{V_R\cos\gamma}\cos\lambda\sin\lambda\sin A \\
0 &= \gamma + 1 + \frac{t^2}{10000} \\
0 &= A - 45 - \frac{t^2}{1000}
\end{aligned}$$

with the following expressions and constants:

$$\begin{aligned}
a_e &= 20902900 && \text{earth radius,} \\
\mu &= 14076539160000000 && \text{gravity constant,} \\
\Omega &= 0.000072921159 && \text{earth angular rotation rate,} \\
m &= 2.890532728 && \text{mass of the vehicle,} \\
s &= 1 && \text{parameter,} \\
r &= H + a_e, && \\
g &= \frac{\mu}{r^2} && \text{gravity force,} \\
c_l(\alpha) &= \frac{\alpha}{100} && \text{aerodynamic lift coefficient,} \\
\rho &= \frac{0.002378}{23800} e^{-H} && \text{density of the atmosphere,} \\
l &= 0.5\rho c_l(\alpha) s V_R^2 && \text{aerodynamic lift force,} \\
d &= l. &&
\end{aligned}$$

In [3] the following consistent initial value is ascertained by:

$$x = (100000, 0, 0, 12000, -1, 45, 2.6728700742, -0.05220958616134)^T,$$

where the first six components were given by the task and the two latter were computed analytically making use of the special structure of the problem.

For the computation of the initial values we start from the values

$$\alpha = x_0^0 = (100000, 0, 0, 12000, -1, 45, 2, -1)^T, \quad y_0^0 = 0, \quad t_0 = 0.$$

The result with a defect of $3.62 \cdot 10^{-17}$ is computed as

$$x = \begin{pmatrix} 100000 \\ 0 \\ 0 \\ 12000 \\ -1 \\ 45 \\ 2.672870048063419 \\ -0.05220958583937511 \end{pmatrix}, y = \begin{pmatrix} -209.428877247402 \\ 4.039436946700642 \cdot 10^{-4} \\ 4.039436946700642 \cdot 10^{-4} \\ -34.97826035277649 \\ -1.490116119384766 \cdot 10^{-28} \\ -1.490116119384766 \cdot 10^{-27} \\ 0 \\ 0 \end{pmatrix}$$

The relative error of x with respect to the analytical solution provided in [5] is $2.6 \cdot 10^{-13}$. The following table contains a comparison of the behaviour of the iteration procedure of the original version and the new one. The column labels denote the requested tolerance (Tolerance), number of outer iterations (Outer iter.), number of inner iterations (Inner iter.), and the final defect (Defect).

	Version of [5]			Present version		
Tolerance	Outer iter.	Inner iter.	Defect	Outer iter.	Inner iter.	Defect
10^{-2}	3	2	$3.14 \cdot 10^{-8}$	1	1	$4.01 \cdot 10^{-3}$
10^{-4}	2	2	$4.24 \cdot 10^{-5}$	2	2	$3.06 \cdot 10^{-8}$
10^{-8}	3	3	$8.41 \cdot 10^{-9}$	2	3	$3.54 \cdot 10^{-9}$
10^{-12}	5	6	$6.57 \cdot 10^{-15}$	3	4	$6.89 \cdot 10^{-13}$
10^{-14}	5	7	$6.57 \cdot 10^{-15}$	3	5	$3.62 \cdot 10^{-17}$

The difference in performance of the two implementations is mainly due to the numerical differentiation routines used. If the accuracy of the numerical integration in the version of [5] is increased, the behaviour of the two implementations becomes similar.

The second example consists of the initialization problem for an incompressible Navier-Stokes problem in two dimensions, discretized by finite elements with respect to space. The computational domain is sketched in Figure 1. The left-hand border is the inflow region while the flow leaves the region through the right-hand boundary. The hole (in fact, a cylinder) is placed slightly unsymmetric such that the flow becomes turbulent.

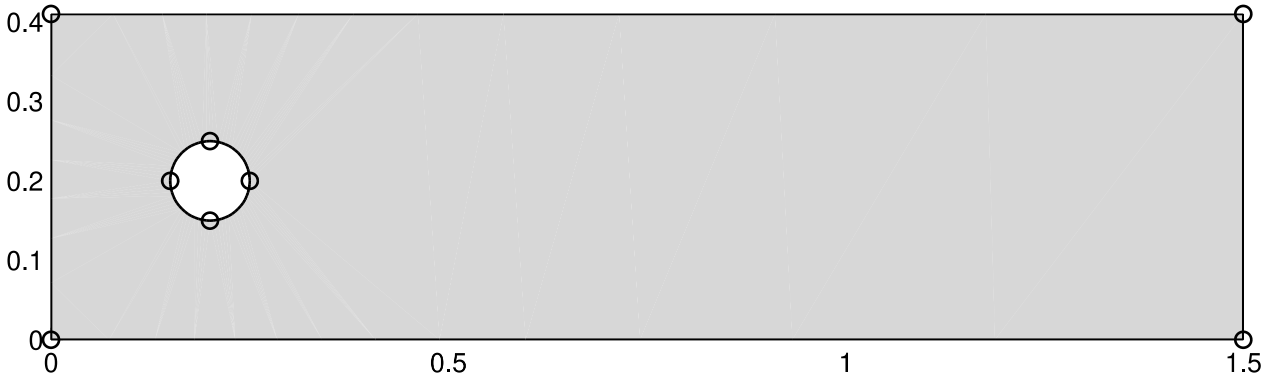

Figure 1: Computational domain for the second example

The governing equations are

$$\frac{\partial \mathbf{u}}{\partial t} - \eta \nabla^2 \mathbf{u} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = 0,$$
$$\nabla \cdot \mathbf{u} = 0$$

where $\mathbf{u} = (u, v)$ denotes the velocity and p the pressure. The boundary conditions are given by

$$u(x,y) = 4y(0.41 - y), v(x,y) = 0, \text{ on } 0 \times (0, 0.41)$$
$$p(x,y) = 0, \text{ on } 1.5 \times (0, 0.41)$$
$$u(x,y) = v(x,y) = 0, \text{ on all other boundary points.}$$

The initial guess for all functions is zero such that even the Dirichlet boundary conditions are not fulfilled.

The problem was discretized by P1-iso-P2 finite elements using FEMLAB® [1]. The performance of the method is summarized in the following table. Each row corresponds to a different triangularization. The tolerance requested was 10^{-10}. The abbreviations within that table denote the number of outer iterations (Outer iter.) and inner iterations (Inner iter.), respectively.

Dimension	Initial defect	Final defect	Outer iter.	Inner iter.	Time [s]
1613	$6.50 \cdot 10^{-2}$	$1.04 \cdot 10^{-12}$	1	2	32.25
2063	$6.42 \cdot 10^{-2}$	$1.07 \cdot 10^{-12}$	1	2	59.43
2585	$6.17 \cdot 10^{-2}$	$1.07 \cdot 10^{-12}$	1	2	105.83
3563	$5.70 \cdot 10^{-2}$	$4.85 \cdot 10^{-12}$	1	2	245.81
5029	$5.05 \cdot 10^{-2}$	$3.48 \cdot 10^{-12}$	1	2	639.67

The computed consistent initial values for the smallest dimension are plotted in the Figures 2–4.

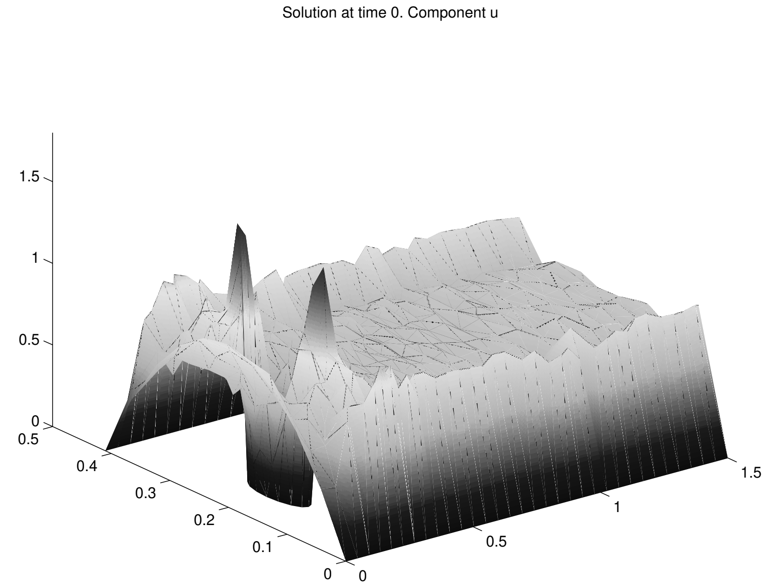

Figure 2: Consistent initial values for P1-iso-P2 elements: u

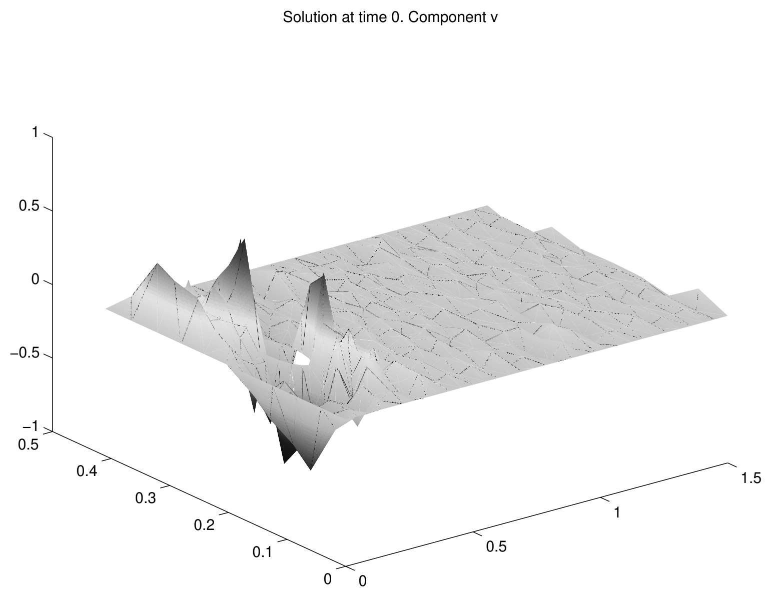

Figure 3: Consistent initial values for P1-iso-P2 elements: v

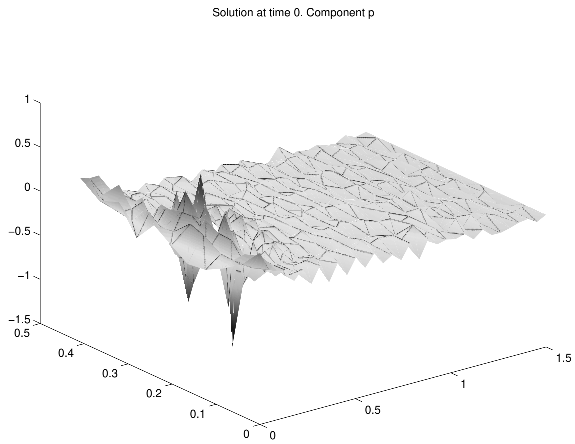

Figure 4: Consistent initial values for P1-iso-P2 elements: p

5 Conclusion

In the present paper, we have presented an implementation of an algorithm for the consistent initialization of index-2 dae's arising within the method of lines. The crucial point is the efficient computation of a nullspace and a sparse projector onto it. The solution proposed is applicable to discrete systems of moderate dimension. An important property of the algorithm consists in the fact that it is realized within MATLAB in a completely portable manner.

A drawback of the approach consists in the use of the initial condition in the form $PP_1(x-\alpha) = 0$. This requirement may give rise to an "unwanted" initial value. Instead, one may try to replace some equations of $PP_1(x-\alpha) = 0$ by conditions $Kx = \beta$, which are better adapted to the physics behind the model. A principal solution is given in [9] but it is not clear how it can be generalized with the aims of the present paper in mind.

References

[1] COMSOL AB. *FEMLAB 2.0.* Tegnérgatan 23, SE–111 40 Stockholm, Sweden.

[2] P. Amodio and F. Mazzia. An algorithm for the computation of consistent initial values for differential-algebraic equations. *Numer. Algorithms*, 19:13–23, 1998.

[3] K. E. Brenan, S. L. Campell, and L. R. Petzold. *Numerical solution of initial-value problems in differential- algebraic equations.* North-Holland, New York, 1989.

[4] Diana Estévez Schwarz. *Consistent initialization for index-2 differential-algebraic equations and its application to circuit simulation.* PhD thesis, Humbold-Univ., Berlin, 2000.

[5] Diana Estévez Schwarz and René Lamour. The computation of consistent initial values for nonlinear index-2 differential-algebraic equations. *Numer. Algorithms*, 26(1), 2001.

[6] Diana Estévez Schwarz and Caren Tischendorf. Structural analysis for electric circuits and consequences for MNA. *J. Circuit Theory Appl.*, 28(2000)2:131–162, 1999.

[7] Eberhard Griepentrog and Roswitha März. *Differential-Algebraic Equations and their Numerical Treatment*, volume 88 of *Teubner-Texte zur Mathematik*. Teubner, Leipzig, 1986.

[8] Beate Hansen. Computing consistent initial values for nonlinear index-2 differential-algebraic equations. In E. Griepentrog, M. Hanke, and R. März, editors, *Berlin Seminar on Differential-Algebraic Equations. Seminar Notes*, volume 92-1 of *Seminarbericht*, pages 142–157. Humboldt-Univ., Fachbereich Math., Berlin, 1992.

[9] R. Lamour. A shooting method for fully implicit index-2 differential-algebraic equations. *SIAM J. Sci. Comput.*, 18:94–114, 1997.

[10] B. J. Leimkuhler. *Approximation methods for the consistent initialization of differential-algebraic equations*. PhD thesis, Univ. of Illinois, Dept. of Computer Science, 1988. Report, UIUC DCS-R-88-1450.

[11] Roswitha März. Numerical methods for differential-algebraic equations. *Acta Numerica*, pages 141–198, 1992.

[12] C.C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Statist. Comput.*, 9:213–231, 1988.

[13] The MathWorks, Inc. *MATLAB 5.3*. 24 Prime Park Way, Natick MA, 1999.