A Note on the Consistent Initialization for Nonlinear Index-2 Differential-Algebraic Equations in Hessenberg form

by

Michael Hanke
Royal Institute of Technology
Department of Numerical Analysis and Computer Science

Parallel and Scientific Computing Institute

Royal Institute of Technology
and
Uppsala University

Report No. 2004:02

May 27, 2004

A Note on the Consistent Initialization for Nonlinear Index-2 Differential-Algebraic Equations in Hessenberg form

Michael Hanke*
Royal Institute of Technology
Department of Numerical Analysis and Computer Science

May 27, 2004

Abstract

In a previous paper, we developed a new algorithm for the consistent initialization of general index-2 differential-algebraic equations arising within the method of lines for solving partial differential equations. In the case of Hessenberg systems, the structural information allows for a lot of simplifications thus allowing for much larger systems to be solved. The crucial point consists of providing sparse projections by the use of sparse approximations to the inverse of the mass matrix. We obtain almost linear computational complexity with respect to the number of degrees of freedom.

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

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

1 Introduction

In the present paper we are interested in the computation of consistent initial values for differential-algebraic equations in Hessenberg form which arise within the method of lines:

$$\begin{aligned} M(x_1,t)x_1'+b_1(x_1,x_2,t) &= 0,\\ b_2(x_1,t) &= 0. \end{aligned} \tag{1}$$

Since the unknown x_2 appears only in non-differentiated form, it is clear that initial conditions are meaningful for x_1, only,

$$x_1(t_0) = x_1^0. \tag{2}$$

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

Because of the algebraic constraint (1) the initial condition must fulfill the condition $b_2(x_1^0, t_0) = 0$. Although necessary, this condition is not sufficient to guarantee the solvability of the initial value problem (1), (2). An initial value x_1^0 is called *consistent* if there exist a solution for the problem (1), (2). Assuming that the mass matrix $M(x_1^0, t_0)$ is invertible, we obtain by differentiating the algebraic constraint in (1) the additional constraint

$$-b_{2,x_1}(x_1^0, t_0)M^{-1}(x_1^0, t_0)b_1(x_1^0, x_2^0, t) + b_{2,t}(x_1^0, t_0) = 0.$$

If x_1^0 is a consistent initial value, this equation must have a solution x_2^0. The latter constraint is called a hidden one because it does not explicitely appear in (1). The real number of degrees of freedom for the initial value (2) can be determined only if all constraints (explicit as well as hidden ones) are known. Therefore, it is a good idea to replace (2) by a requirement

$$Q(x_1(t_0) - \alpha) = 0 \tag{3}$$

where Q is chosen to fix the degrees of freedom that are not already determined by the constraints. The consistent initialization problem for (1) consists of computing (x_1^0, x_2^0) such that (1) possesses a solution with $x_1(t_0) = x_1^0$, $x_2(t_0) = x_2^0$, and $Q(x_1^0 - \alpha) = 0$. For practical reasons, e.g. when initializing an ode solver, it is often a good idea to have $y_1 = x_1'(t_0)$ available, too.

There are a number of different approaches for solving the consistent initialization problem. For an overview see, e.g. [2, 5]. In the present note, we will adapt the algorithm of [5] to the present situation. In that paper, an algorithm for the consistent initialization of index-2 quasilinear differential algebraic equations

$$A(x,t)x' + b(x,t) = 0$$

was developed. We will obtain a considerable speedup compared with [5]. Moreover, the computational complexity of the resulting algorithm is almost linear with respect to the number of degrees of freedom.

The paper is organized as follows. In Section 2, the algorithm is developed. We will derive the necessary equations as well as algorithms for the sparse approximation of certain projectors. The most often used *canonical projection* leads to a full matrix such that its use is impossible. Another problem consists of providing sparse approximations to the inverse of the mass matrix. It is symmetric and positive definite with a full inverse. It turns out that Chebychev approximations can be used efficiently. In Section 3, we provide an example which illustrates the performance of the method.

2 The Initialization Algorithm

In this note, we restrict ourselves to index-2 Hessenberg systems. That is, we require the matrices

$$M(x_1, t) \text{ and } B_{21}(x_1, t)M^{-1}(x_1, t)B_{12}(x_1, x_2, t) \tag{4}$$

to be nonsingular for all arguments (x_1, x_2, t) in a neighborhood of a solution (see [6, 4]). Here, we used the notation $B_{ij} := \frac{\partial}{\partial x_j} b_i$ with $i, j = 1, 2$. If there is no fear of ambiguity, we will omit the arguments. If the condition (4) is fulfilled, the explicit and hidden constraints are given by

$$\begin{aligned} b_2(x_1, t) &= 0, \\ B_{21}(x_1, t) M^{-1}(x_1, t) b_1(x_1, x_2, t) + b_{2,t}(x_1, t) &= 0. \end{aligned} \tag{5}$$

In order to characterize the remaining degrees of freedom it is convenient to use a projection Q_{11}. Let Q_{11} be a projection onto $\mathrm{im}(M^{-1} B_{12})$. Note that, in general, $Q_{11} = Q_{11}(x_1, x_2, t)$. Then, for a given α, a consistent initial value (x_1^0, x_2^0) is uniquely defined by [6, 2]

$$\begin{aligned} (I - Q_{11}(x_1^0, x_2^0, t_0))(x_1^0 - \alpha) &= 0, \\ b_2(x_1^0, t_0) &= 0, \\ B_{21}(x_1^0, t) M^{-1}(x_1^0, t_0) b_1(x_1^0, x_2^0, t_0) + b_{2,t}(x_1^0, t_0) &= 0. \end{aligned} \tag{6}$$

The canonical projection of [6] amounts to using

$$Q_{11} = H = M^{-1} B_{12} (B_{21} M^{-1} B_{12})^{-1} B_{21}.$$

Once a projection is available, it remains to solve the system (6). Taking into account the nonlinearity, some variant of Newton's method must be used. The drawback is that second order derivatives of b_2 appear in the Jacobian. More severely, the projection Q_{11} has a complicated dependence on the arguments and is expensive to compute such that its derivatives are hard to provide. Therefore, we prefer to use a two-stage iteration according to the proposal in [3, 5]. Introduce $y_1^0 = M^{-1}(x_1^0, t_0) b_1(x_1^0, x_2^0, t_0)$. Then, $(x_1, x_2, y_1) = (x_1^0, x_2^0, y_1^0)$ is a solution of the system

$$\begin{aligned} (I - Q_{11}(x_1^0, x_2^0, t_0))(x_1 - \alpha) &= 0, \\ M(x_1^0, t_0) y_1 + b_1(x_1, x_2, t_0) &= 0, \\ b_2(x_1, t_0) &= 0, \\ B_{21}(x_1^0, t_0) y_1 + b_{2,t}(x_1, t_0) &= 0. \end{aligned} \tag{7}$$

The Jacobian of (7) with respect to (x_1, x_2, y_1) is given by (omitting the arguments)

$$J = \begin{pmatrix} I - Q_{11} & 0 & 0 \\ B_{11} & B_{12} & M \\ B_{21} & 0 & 0 \\ B_{21,t} & 0 & B_{21} \end{pmatrix}$$

The inverse of J is explicitely computable. More precisely, the solution of the linear system

$$J \begin{pmatrix} x_1 \\ x_2 \\ y_1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$$

is given by

$$\begin{aligned} x_1 &= \alpha + M^{-1}B_{12}W^{-1}(\gamma - B_{21}\alpha), \\ x_2 &= -W^{-1}(\delta + (B_{21}M^{-1}B_{11} - B_{21,t})x_1 - B_{21}M^{-1}\beta), \\ y_1 &= M^{-1}(\beta - B_{11}x_1 - B_{12}x_2), \\ W &= B_{21}M^{-1}B_{12} \end{aligned} \tag{8}$$

provided that the system is solvable. Once LU decompositions of M and W are available, the solutions in (8) can be easily evaluated. In order to compute the defect of an approximation in (7), additionally a good representation for Q_{11} is necessary. The main problem consists in computing *sparse* quantities. Since M^{-1} is a full matrix, so are W and the canonical projection. Therefore, we will consider their sparse approximation in the following.

2.1 A Sparse Projection

We need to compute a projection Q_{11} onto $\mathrm{im}(M^{-1}B_{12})$. Since W is nonsingular, $T = M^{-1}B_{12}$ has full rank. Assume that S is a generalized inverse of T. Then it is well-known that $Q_{11} = TS$ is a projection onto $\mathrm{im}(T)$ [7]. Therefore, we are looking for a generalized inverse S which is as sparse as possible. There are certain possibilities available.

- A straightforward approach consists in using the Moore-Penrose generalized inverse $S = T^{\dagger}$. In that case, Q_{11} is the orthogonal projection onto $\mathrm{im}(T)$. Because T has full rank, it holds $T^{\dagger} = (T^T T)^{-1}T^T$. This results nearly always in a fully occupied projection Q_{11} such that this alternative should not be used.

- Assume that a QR decomposition of B_{12} is available: $B_{12} = QR$ with an orthogonal matrix Q and an upper triangular matrix R. Then it holds

$$\begin{aligned} T &= M^{-1}QR \\ &=: M^{-1}Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix}. \end{aligned}$$

  Hence, a generalized inverse is given by $S = (R_1^{-1} \vdots 0)Q^T M$. This gives rise to the projection

$$Q_{11} = M^{-1}Q \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} Q^T M.$$

- Let us use an LU decomposition instead: $PB_{12} = LU$ with a permutation matrix P, a lower triangular factor L, and a square upper triangular matrix U. Then it holds

$$\begin{aligned} T &= M^{-1}P^T LU \\ &=: M^{-1}P^T \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} U. \end{aligned}$$

A generalized inverse can be represented by $S = U^{-1}(L_1^{-1}\!:\!0)PM$. We obtain the projection

$$Q_{11} = M^{-1}P^T \begin{pmatrix} I & 0 \\ L_2 L_1^{-1} & 0 \end{pmatrix} PM.$$

The projection Q_{11} is never used explicitely. It is only required that the projector-vector multiplication can be carried out. Therefore, it is sufficient to have a (sparse) LU decomposition of M. Obviously, R^{-1} and L^{-1} need not be formed explicitly either.

2.2 A Sparse Inverse For a Mass Matrix

The most inconvenient part in (8) is doubtlessly the multiplication by W^{-1}. In order to form W the inverse mass matrix M^{-1} is needed. This inverse is a full matrix unless M is diagonal. The latter is the case if a lumped mass matrix is used. This is far from being practical in our case. On the other hand, motivated by mass lumping, we could be tempted to replace M in W by the lumped mass matrix. Practical experiments showed that this approximation does not lead to a convergent iteration. Therefore, we are looking for better sparse approximation to M^{-1}. Since M by itself is a sparse matrix, we try to construct matrix polynomials to approximate the inverse,

$$M^{-1} \approx \sum_{i=0}^{k} \gamma_{ki} M^i.$$

Fortunately, the mass matrix has a nice property. Let D denote the diagonal matrix with diagonal entries from M. Wathens [8] proved that there are very sharp bounds on the eigenvalues of the diagonally preconditioned matrix $D^{-1}M$. He could show that, for triangular elements, the maximal and minimal eigenvalue are independent of the mesh! Moreover, the condition number is in the order of magnitude of 10.

Using this result, one can easily construct Chebychev polynomials which approximate the inverse of M rather well.

3 An Example

The example is taken from [5]. It 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.

The governing equations are

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

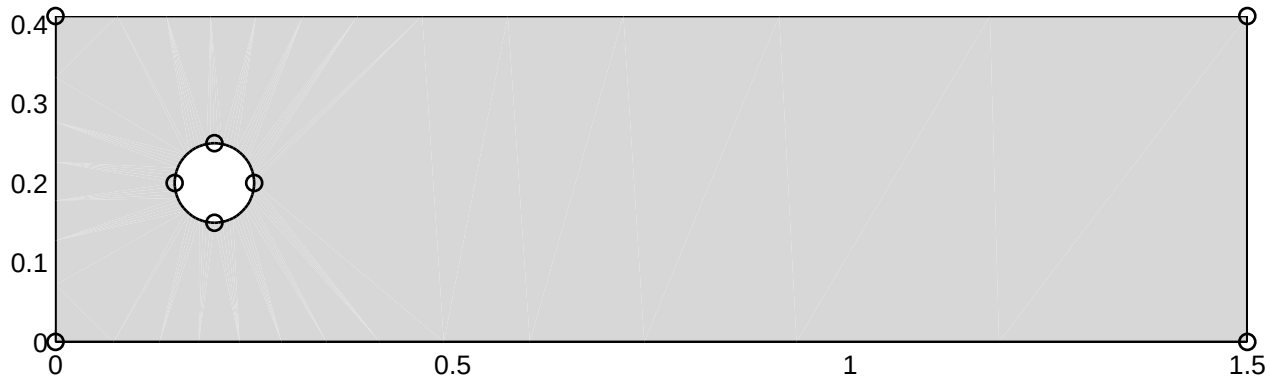

Figure 1: Computational domain for the second example

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

$$\begin{aligned} u(x,y) = 23.8y(0.41-y), v(x,y) = 0, \text{ on } 0 \times (0,0.41) \\ p(x,y) = 0, \text{ on } 1.5 \times (0,0.41) \\ u(x,y) = v(x,y) = 0, \text{ on all other boundary points.} \end{aligned}$$

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]. For these elements, the eigenvalue bounds for the diagonally preconditioned mass matrix are $\lambda_{\min} = 1/2$ an $\lambda_{\max} = 2$ [8]. The tolerance requested was 10^{-10}. The discretization leads to an autonomous Hessenberg system with linear constraints and a linearly appearing x_2 (which represents the discretized pressure). The nonlinearity is quadratic with respect to x_1. As a consequence of these properties, all equations in (7) except for the differential equation are linear. Moreover, the projections are independent of x_1, x_2. So we have one outer iteration, only.

In order to estimate the computational complexity, we used a model

$$t_{\text{CPU}} = O(N^p)$$

where N denotes the number of degrees of freedom and t_{CPU} the CPU time. The order p is estimated by a least squares approximation. Optimal computational complexity is linear one, $p = 1$.

The computations were done on an AMD K7 700 machine running Linux.

As it was expected, we can observe a huge improvement in the performance of the method. The most important observation is that the method attains almost linear computational complexity. Even for relatively low order approximating polynomials, a fast convergence is achieved. The low order makes W in (8) sparse.

The next experiment consists of the same example but in 3 space dimensions. The computational domain is given by $\Omega = (0,1.5) \times (0,0.4) \times (0,0.4) \setminus \{(x,y,z) | (x-0.2)^2 + (y-0.2)^2 +$

k	1613 (1022+591)	2063 (1314+749)	2585 (1654+931)	3563 (2298+1265)	5029 (3262+1767)	Order
lumped	2.18/2	2.75/2	3.46/2	5.17/2	8.20/2	1.2
0	divergent	divergent	divergent	divergent	divergent	
1	divergent	divergent	divergent	divergent	divergent	
2	6.43/12	7.98/12	9.73/12	15.66/13	divergent	
3	4.64/8	6.26/8	7.68/8	11.27/8	15.66/8	1.1
4	4.51/7	5.69/6	7.47/7	11.08/7	17.70/7	1.2
5	4.43/6	5.86/6	7.33/6	11.55/6	18.21/6	1.2
6	4.34/5	5.81/5	7.48/5	11.78/5	18.99/5	1.3
7	4.26/4	5.92/4	7.51/4	12.07/4	20.46/4	1.4
8	4.66/4	6.45/4	8.30/4	13.48/4	22.02/4	1.4
9	5.06/4	7.05/4	9.25/4	14.83/4	25.34/4	1.4
10	5.55/4	7.77/4	10.17/4	16.44/4	27.74/4	1.4
∞	5.23/2	8.14/2	12.15/2	23.04/2	47.77/2	1.9
[5]	32.25/2	59.43/2	105.83/2	245.81/2	639.67/2	2.6

Table 1: Results for the 2-dimensional incompressible Navier-Stokes example. Every column head contains the number of equations. In parenthesis, the number of differential equations and constraints is indicated. k denotes the order of the approximating polynomial for M^{-1}. $k = 0$ represents a simple diagonal approximation. $k = \infty$ is equivalent to using the exact inverse. As a special case, we give comparative figures if the lumped mass matrix is used instead of the full one. The number before the slash is the computation time in seconds while the number after the slash is the number of iterations. The last column contains an estimate of the computational complexity

$(z-0.2)^2 \leq 0.05^2\}$. The boundary conditions are given by

$$u(x,y) = 625yz(0.4-y)(0.4-z), v(x,y,z) = 0, \text{ on } 0 \times (0,0.4) \times (0,0.4)$$
$$p(x,y) = 0, \text{ on } 1.5 \times (0,0.4) \times (0,0.4)$$
$$u(x,y) = v(x,y) = 0, \text{ on all other boundary points.}$$

The problem was discretized using Lagrangian P2-P1 elements. Using the results of [8], estimates for the smallest and the largest eigenvalues of the diagonally scaled mass matrix were computed. The result is $0.24 \leq \lambda_{\min}$ and $\lambda_{\max} \leq 4.3475$. In this example, the number of differential equations is one order of magnitude larger than that of the algebraic constraints.

The present computations were carried out on one node of an IBM SP2 under AIX (Table 2).

In this example, the behavior of the approximation is much worse. The reason is the huge difference in the dimensions of the differential variables and the algebraic variables. This gives rise to a large amount of computations for forming the polynomial approximation for the inverse mass matrix. On the other hand, W is no longer a really sparse matrix. As a conclusion, it would be better to try to find sparse approximations for W *immediately* and not by approximating only M^{-1}.

4 Conclusions

In the present paper, we analyzed the behavior of a general method for computing consistent initial values for index-2 differential-algebraic equations if this method is applied to Hessenberg systems. The special structure of these systems allowed for a considerable simplification of the algorithm. By introducing sparse approximations to the inverse mass matrix we were able to construct an algorithm which showed almost linear complexity for discretized incompressible Navier-Stokes problems. On the other hand, the algorithm loses some of its efficiency if the number of constraints is considerably less than the number of differential equations.

In a future work, other approximations to the inverse mass matrix as well to the matrix W which decides over the overall efficiency of the method will be considered.

References

[1] COMSOL AB. *FEMLAB 2.2*. Tegnérgatan 23, SE–111 40 Stockholm, Sweden, 2001.

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

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

[4] E. Hairer and G. Wanner. *Solving ordinary differential equations II*, volume 14 of *Springer Series in Computational Mathematics*. Springer, Berlin, 2. rev. ed. edition, 1996.

k	6354 (5971+383)				11493 (10811+682)				17895 (16847+1048)				Order	
	Time	Assem	Newton	Iter	Time	Assem	Newton	Iter	Time	Assem	Newton	Iter	Global	Newton
lumped	46.97	38.62	8.35	2	86.03	65.58	20.45	2	164.21	101.90	62.31	2	1.2	1.9
0	divergent				divergent				divergent					
1	195.65	156.04	39.61	19	346.15	273.33	72.82	19	divergent					
2	divergent				divergent				divergent					
3	174.89	120.69	54.20	14	291.66	192.92	98.74	13	510.29	296.05	214.24	13	1.0	1.3
4	172.41	93.46	78.95	10	308.19	158.45	149.74	10	out of memory					
5	195.77	79.86	115.91	8	360.79	134.97	225.82	8	out of memory					
6	237.96	72.84	165.12	6	out of memory				out of memory					
7	279.94	65.64	214.30	6	out of memory				out of memory					
8	323.93	58.92	265.01	5	out of memory				out of memory					
9	366.80	52.13	314.67	4	out of memory				out of memory					
10	421.89	52.44	369.45	4	out of memory				out of memory					
∞	86.57	38.58	47.99	2	194.99	65.68	129.31	2	485.61	101.35	384.26	2	1.6	2.0

Table 2: Results for the 2-dimensional incompressible Navier-Stokes example on an IBM SP2. The column heads are the same as in Table 1. Since the assembly process is rather expensive, we provide the cumulative CPU time (Time) together with that spent in the assembly process (Assem) and the Newton iteration (Newton). Iter denotes the number of iterations

[5] Michael Hanke and René Lamour. Consistent initialization for differential-algebraic equations: Large sparse systems in MATLAB. *Numer. Algorithms*, 32:67–85, 2003.

[6] Roswitha März. Index-2 differential-algebraic equations. *Results in Math.*, 15:149–171, 1989.

[7] M.Z. Nashed and G.F. Votruba. A unified operator theory of generalized inverses. In M.Z. Nashed, editor, *Generalized Inverses and Applications*, pages 1–109. Academic Press, New York, 1976.

[8] A.J. Wathen. Realistic eigenvalue bounds for the Galerkin mass matrix. *IMA J. Numer. Anal.*, 7(4):449–457, 1987.