M A T H E M A T I C S D E P A R T M E N T

A Numerical Method for the Computation of the Analytic Singular Value Decomposition

S.J.G. Bell, N.K. Nichols

Numerical Analysis Report 2/94

U N I V E R S I T Y O F R E A D I N G

Numerical Analysis Report 2/94

[1]Thanks are due to my supervisor Nancy Nichols for support and assistance, and to the Science and Engineering Research Council, who have funded my PhD.

Abstract

A numerical method for calculating a smooth time-varying matrix decomposition based on the singular value decomposition for constant matrices is given. The decomposition is known as the Analytic Singular Value Decomposition (ASVD) and is here calculated by reducing it to a system of ordinary differential equations. These equations are then solved numerically using a method which preserves orthogonality of the left and right singular factors. After the basic method is given, examples are shown which highlight some of the problems. An improved version of the method is then presented.

ontents

1 Introduction

The Singular Value Decomposition (SVD) is a tool of great importance in numerical analysis. It provides a stable, reliable transformation to canonical form which yields a great deal of information about the matrix rank, null space and range space not given by other decompositions. Furthermore, the transformation takes place via orthogonal factors and so the conditioning of the matrix is completely unaffected. In particular, the motivation for this work derives from its use in feedback design for time-invariant descriptor systems (see Bunse-Gerstner et al [3]). The SVD is used to reduce a system of constant coefficient differential algebraic equations to canonical form so that a feedback can be chosen to regularize the system.

With this in mind we present a time-varying version of the SVD, the *Analytic Singular Value Decomposition* (henceforth the ASVD), a version of the singular value decomposition for A(t), a matrix with variable, analytic coefficients - the motivation being, that this may be used for time-varying descriptor systems in the same way that the SVD is used in the time-invariant case (see Kunkel and Mehrmann [5]).

Definition 1 *For a real, analytic, matrix valued function* $E(t) : [a, b] \to \Re^{m\times n}$ *where* $m \geq n$*, an Analytic Singular Value Decomposition is a path of factorisations*

$$E(t) = X(t)S(t)Y(t)^T \tag{1}$$

where

- $X(t) : [a, b] \to \Re^{m\times m}$ *and* $Y(t) : [a, b] \to \Re^{n\times n}$ *are orthogonal,*
- $S(t) : [a, b] \to \Re^{m\times n}$ *is diagonal, and*
- $X(t), Y(t)$ *and* $S(t)$ *are analytic.*

An important point to note is that we must relax the conditions that the singular values be non-negative and ordered for an ASVD to exist. The existence and uniqueness of the ASVD is gone into thoroughly by Bunse-Gerstner et al [2]. We shall concern ourselves with its numerical computation (Section 2), then we present some examples to illustrate the algorithm in action in Section 3. We then say why it does not work and give an improved algorithm which does work in Section 4, as illustrated by the examples of Section 5.

$$\frac{}{2 \quad 2}$$

$$\frac{}{}$$

$$\frac{}{}$$

set to analytic functions). Such a case presents no problem. What is a problem is when two singular values become instantaneously equal in modulus. This may only occur at isolated points, such a point being known as a .

When e1oifIt1 I,ff3SffXwKffeffXe1Kwq(loqXcas1Kwq(ons1 b functionfIt1 $_{j,k}$ I,fffor()ff)1wq anlatf

$$n+1 \qquad n \qquad - \qquad n+1 \qquad n \qquad n+1 \qquad n$$

$$n+1 \qquad n \qquad n \qquad n \qquad -1$$

$$n \qquad \frac{h}{4} \qquad n+1 \qquad n$$

$$n$$

$$0 \qquad n+1$$

0 n

0 0 0 0

0 T 0 0 0

0 0

0 0 0 0

OLD $n-1$

n n

n $n-1$ $n-2$ $n-3$ – – –

n n 1 n 2 n 3

n n 1

n n 1

n n

n n 1

n n 1

1

n n 1

1

n n 1

n n

T

n n n n

— 1 1

2

2 2

2.5 Notes

- Obviously, the forward extrapolation equations do not actually come into play until the third time step. Before this, the value W_{n-1} is used as a first guess for W_n (and similarly for Z).

- A trapezium approximation is used to approximate equation (3). The reason for this is that, since the approximations to equations (4) and (5) are of order h^2, there seems little point in using a higher order scheme for (3).

- A non-generic point is defined as any isolated point where the following condition holds

$$|s_i^2 - s_j^2| < ntol \qquad i \neq j.$$

- Convergence of the fixed point iteration is held to be when the following condition is satisifed:

$$\|S_n - S^{OLD}\|_\infty < itol.$$

  S_n is the latest approximation to the vector of singular values at each step, and S^{OLD} is the previous approximation at the **same** step. The inner loop runs for at most $inmax$ iterations, after which it is assumed that the algorithm has failed to converge. The program then continues from the next time step.

- The two tolerances, $itol$ and $ntol$ are set to the value of h^2 in all of the examples to follow. xperimentally it has been found that tolerances greater than this lead to inaccuracy, and tolerances smaller than this can prevent the algorithm from converging. This seems logical since the schemes used are all of order 2.

- One set of equations for W have been omitted in the above algorithm - these are the extra equations (8) which occur when E is rectangular. Putting them in presents no extra problems - they are omitted merely for neatness.

$$\frac{2}{2}$$

1	1						
1	1			2	2		
				2	2	3	3
						3	3

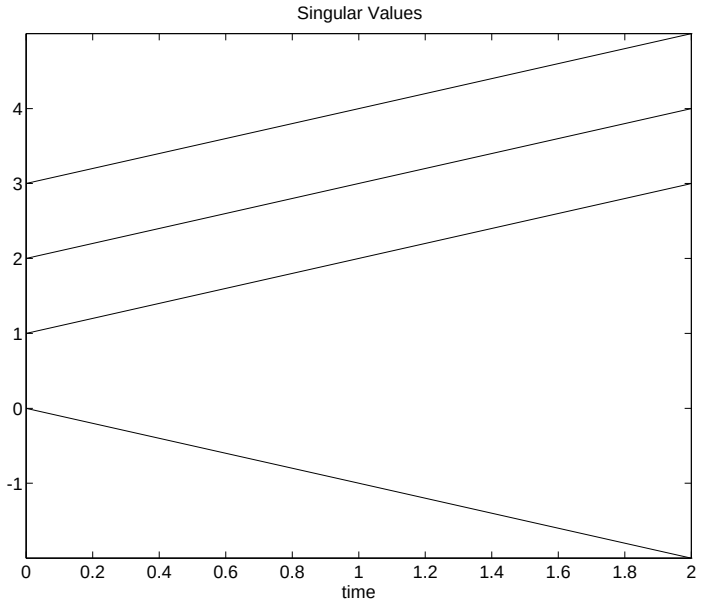


Figure 1: xample 2

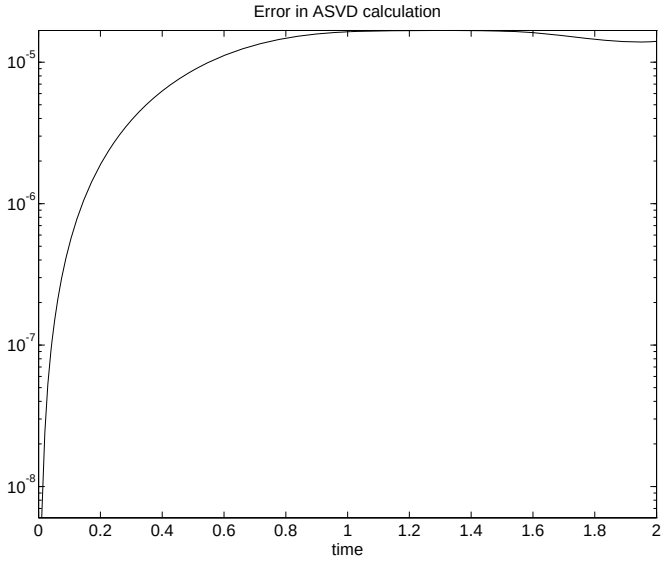
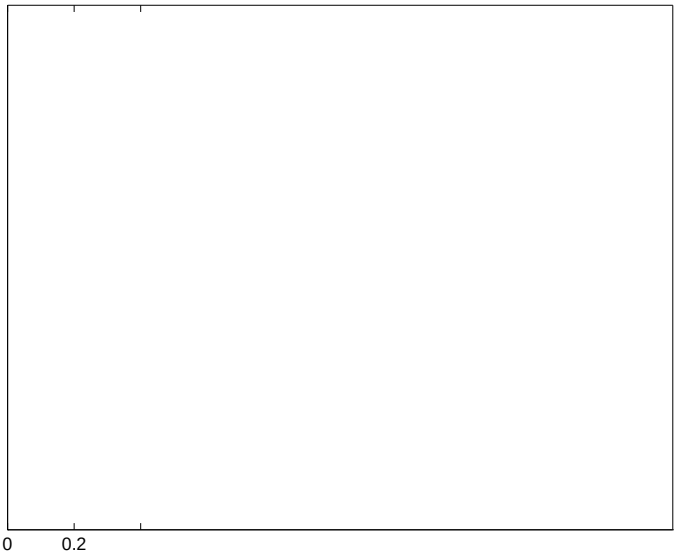


Figure 2: xample 2


0
0.2

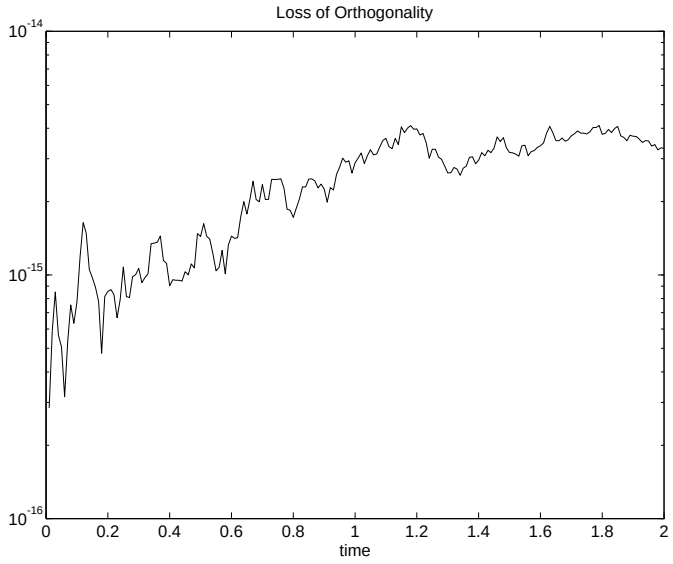


Figure 4: xample 2

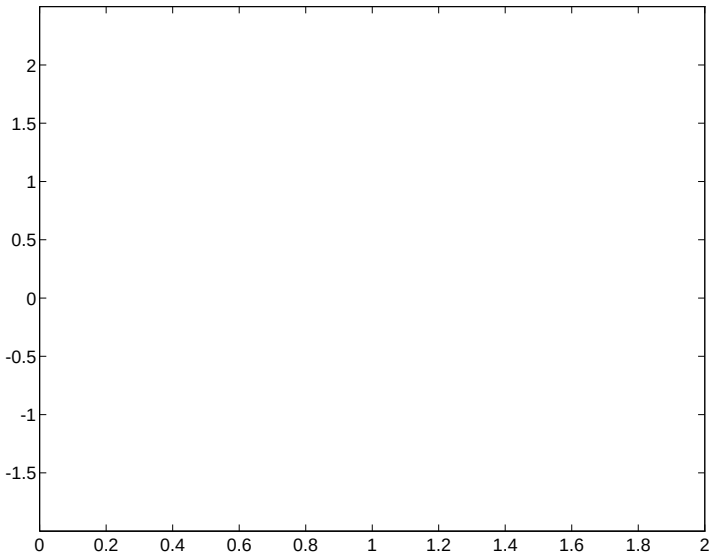
We will not show the same results for any of the following examples, partly for brevity, but mainly because they are all very similar.

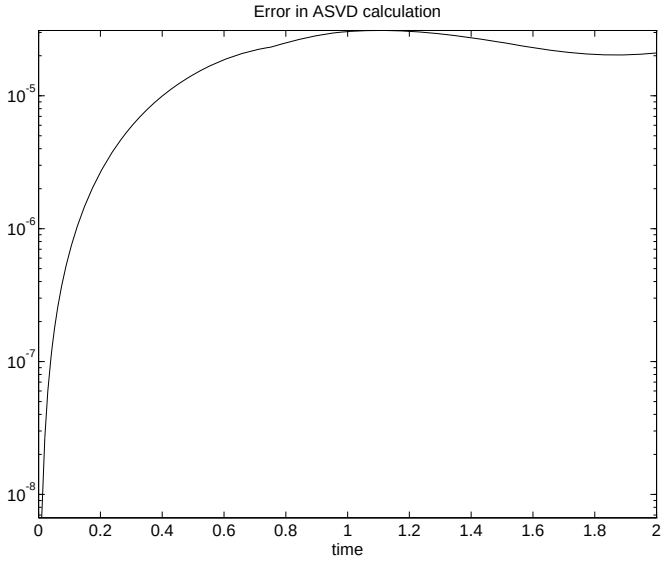
This is of the same form as xample 2, now with

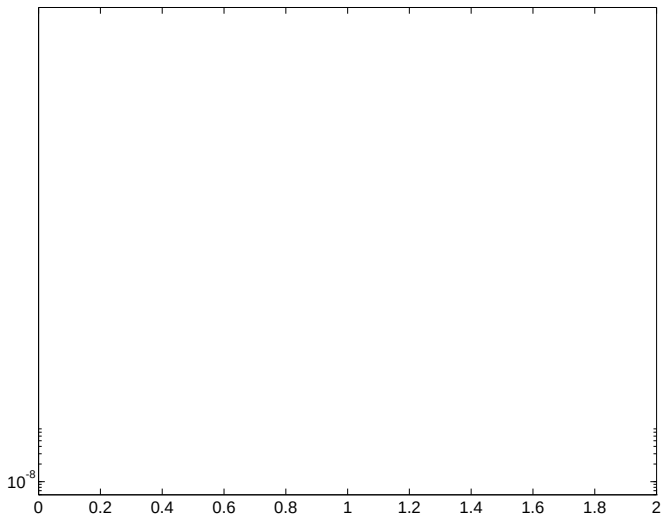
$$( ) = \quad (5 + \quad 2 \quad 1 \quad)$$

and evaluated over the same interval $[0\ 2]$, with the same step-size $= 0\,01$.

This does have non-generic points, occurring at $= 0\,25\ 0\,5\ 0\,75\ 1\ 0\ 1\ 5$. Therefore we might expect worse results, certainly at the non-generic points.


2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Error in ASVD calculation
10^-5
10^-6
10^-7
10^-8
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time


10^{-8}
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

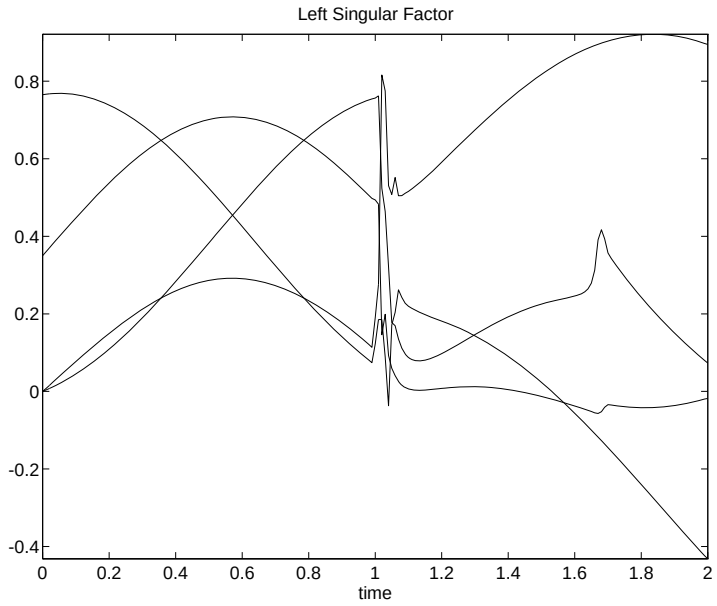


Figure 10: xample 4

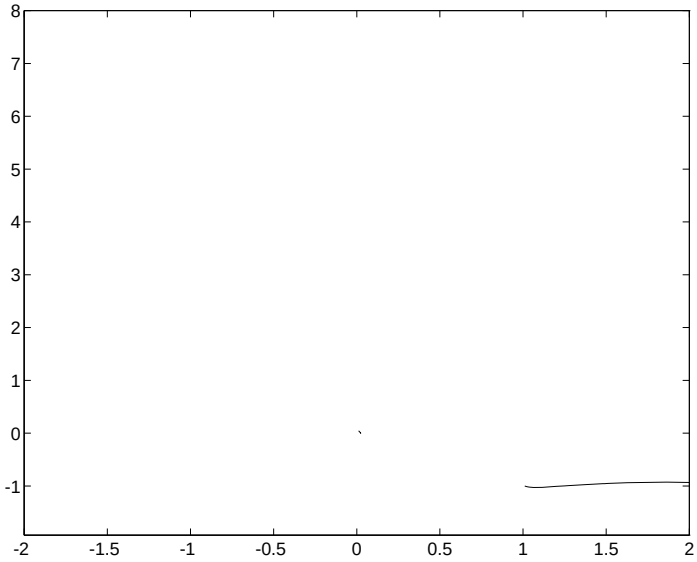
All four singular values have equal modulus at $t = \pm 1$. Further, three of them intersect at $t = 0$. The region of integration is now $t \in [-2, 2]$, and the step-size is still $h = 0.01$.

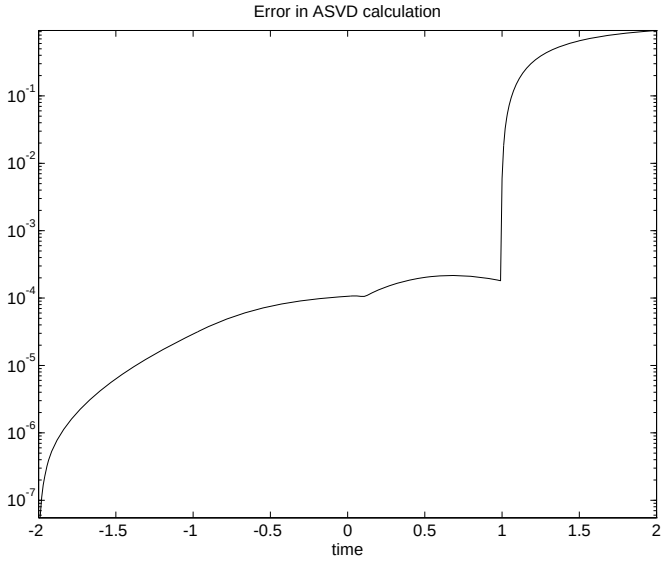
Despite the fact that this should be the worst of the three examples, the algorithm almost manages to cope up until t=1 (see Figure 11). The quadratic contact at t=0 seems to present no problems to the singular values (the error is of the order 10^{-4} at this point), and it is not until t=1 that everything falls apart (Figure 12). Figure 13 shows, once again, a huge jump in the entries of $X(t)$ where the singular values. coincide at $t = 1$.

Again, this version of the algorithm took $inmax = 2$ iterations. xperimentally, if $inmax$ is allowed to be bigger and the step size smaller (10^{-3} or less), then the algorithm will work. The entries of X(t) jump slightly at t=1, but the algorithm stays on course and the error decays back to its previous level beyond $t = 1$ (it has been tested up to t=10). Convergence is attained after no more than two iterations except at t=1, which requires nine.

3.6 Comments

1. Although the errors in the computed singular values give a fair indication of the performance of the algorithm, it should be noted that it is perfectly possible for the singular values to follow the correct path but for the singular


8
7
6
5
4
3
2
1
0
-1
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Error in ASVD calculation
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
time

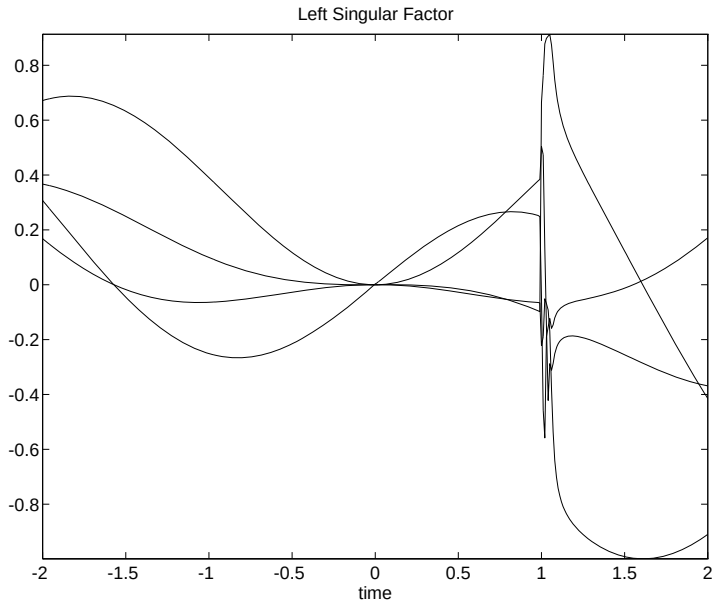
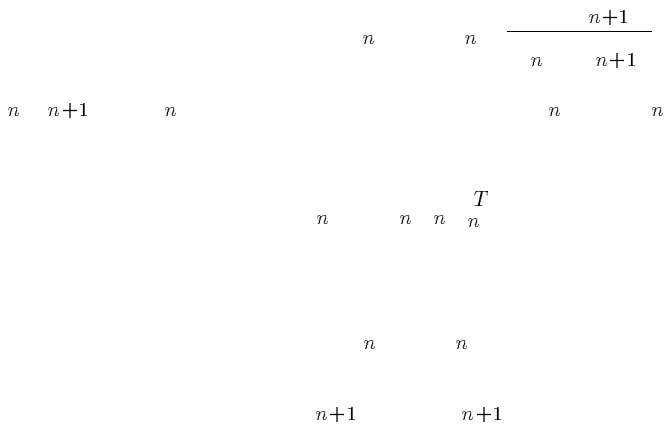


Figure 13: xample 5

matrices to bifurcate. This does not happen in any of the previous examples given, but does in some of the examples in Appendix 1.

2. It should be stressed that we show the last two examples because they cause problems for the method. There are many other examples, all of which give no trouble at all; the algorithm produces results perfectly consistent with a second-order scheme.

A major problem with the algorithm of Section 2 is the gradual build-up of error caused by round-off and truncation error. From experiments run, the bifurcation at some non-generic points seems to be caused by this accumulation of error and not by any step-size criterion (for all the examples tested the step-size condition was always satisfied). At a non-generic point, the ASVD is not fully defined by the equations given and so the problem is very sensitive to error. It seems that if the approximation is still fairly accurate, then the algorithm can cope with this instantaneous singularity and stays on course (in fact, after a non-generic point, the error decays back to its level prior to the non-generic point). If, however, the accumulated error is too great, then the approximated solution may bifurcate and switch to another singular value path. There are several possible ways we could deal with this; one is to obtain extra equations for the entries of W and Z for use at non-generic points. This can be done by differentiation of equations (6)- (8). This idea is impractical, however, in that it requires knowledge of the



Differentiating C, we get

$$\dot{C}(t) = \dot{E}(t) + \frac{(E(t_n) - E_n)}{(t_{n+1} - t_n)}. \tag{15}$$

So, at each node, we replace $\dot{E}(t)$ by $\dot{C}(t)$ and hope that this will remove the accumulated error and hence lead to more accurate results. This "stabilization" does have an effect on the method, as the following results show.

4.1 Stabilized Examples

In the following examples we use exactly the same algorithm as before, but with the derivative of replaced by that of C, where C is our "stabilization matrix". The examples, too, are the same - xample 2 of this section coincides with x-ample 2 of the previous section, etc., and the same step-size ($h = 0.01$) and parameter values are used. Literally the only thing changed is the value of $\dot{E}$. This small change to the algorithm makes a vast difference to the results as we show.

4.2 Example 2

Although this example gave our original algorithm no problems, we can see that there are substantial improvements with our new, stabilized algorithm. If we look at Figure 14 we see what these improvements are specifically, much reduced error. The error is initially order 10^{-8} and then decays rapidly to order 10^{-11}. This is incredible for a second-order method and certainly suggests that stabilization works.

4.3 Example 3

Again, this was dealt with satsifactorily by the original algorithm, even though it had non-generic points. If we look at the error (Figure 15) we see the same drastic improvement; an initial error of order 10^{-8}, decaying away to order $10^{-11} - 10^{-10}$. There is a sharp blip at $t = 1$, corresponding to one of the non-generic points, but this causes no problems later on.

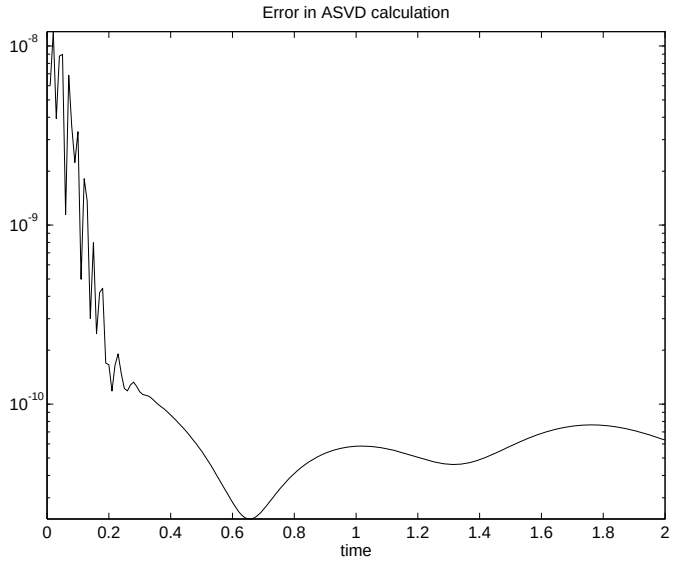


Figure 14: Stabilized xample 2

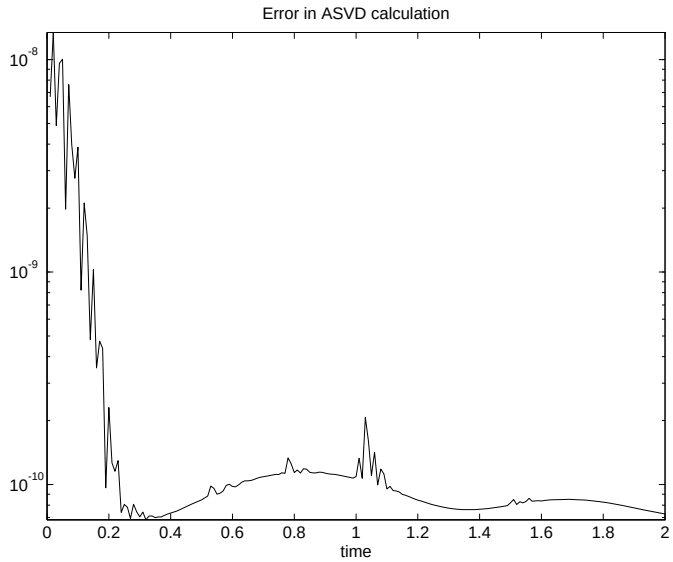
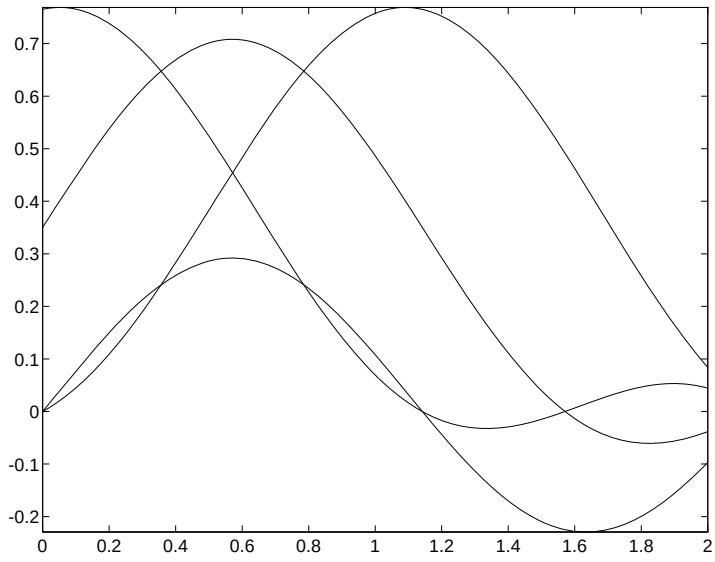


Figure 15: Stabilized xample 3


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

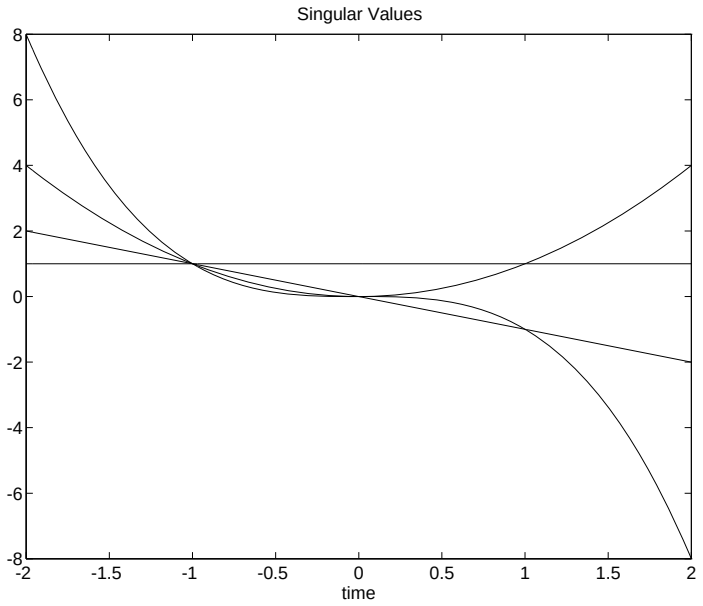
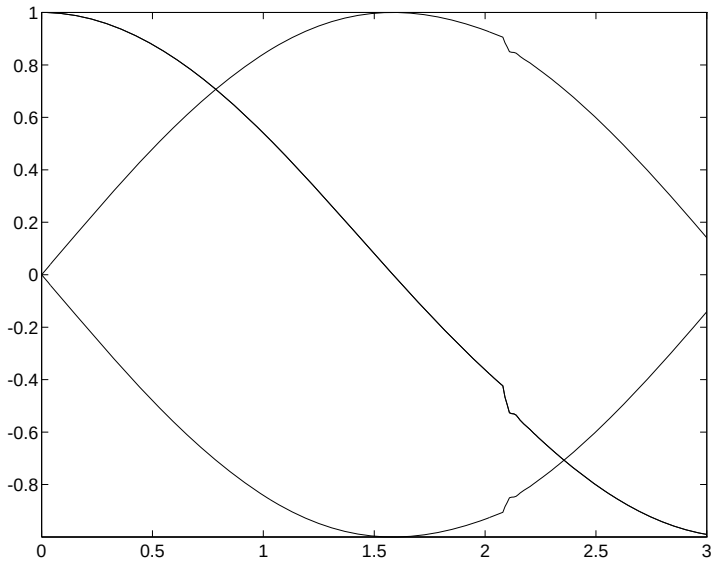


Figure 19: Stabilized xample 5

be so, we do not know, but the error is now of the order 10^{-6} throughout most of the region of integration. There is no sharp decline, but a fairly smooth, constant magnitude error.

Again, inspection of the left singular factor shows perfectly good behaviour (Figure 21).


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.5
1
1.5
2
2.5
3

5 Summary

We have presented a cheap low-order technique for finding the ASVD of an analytic matrix. This method has the very desirable property that the orthogonality of the left and right factors is preserved to machine accuracy. Further, we have modified the ASVD algorithm to increase its accuracy far beyond what could normally be expected for a second order method. We do not know, as yet, why stabilization works as well as it does in some cases, and obviously, we would like to know this and, perhaps more importantly, when it may not work.

What does make the ASVD problem unique is that, although the exact solutions $X(t)$, $S(t)$ and $Y(t)$ are not known explicitly, the product

$$E(t) = X(t)S(t)Y(t)^T$$

is known exactly for all $t \in [a, b]$. This is not information that one normally has when solving an o.d.e. and so it seems logical that use can be made of this extra information to improve the accuracy.

A further modification that has been made is the addition of a step sizer. All of the calculations are carried out by one-step techniques - the only part of the algorithm that requires information from points other than the present node and the previous node is the forward extrapolation used to calculate W_{n+1} and Z_{n+1}. This has been designed for use with a variable step, however, and so it is a fairly simple matter to implement a step sizer. This has been done and results confirm that it works.

The most obvious improvement that could be made is to improve the algorithm's handling of non-generic points.

We said before that obtaining extra equations by differentiating the equations for the ASVD was not a good idea. Perhaps, however, with the extra accuracy of stabilization and forward extrapolation to find an initial estimate for W and Z we can make it a workable idea. If we look at the equations for W and Z at a non-generic point we see that we are left with, at most, one equation

$$s_k W_{j,k} + s_j Z_{k,j} = Q_{j,k}.$$

Using further differentiation Wright [6] shows that we may derive a further two equations for W and Z, thus

$$s_k \dot{W}_{j,k} + s_j \dot{Z}_{k,j} + 2\dot{s}_k W_{j,k} + 2\dot{s}_j Z_{k,j} = -(X^T \ddot{E} Y)_{j,k} - \sum_{i \neq k} W_{j,i} Q_{i,k} - \sum_{i \neq j} Q_{j,i} Z_{k,i},$$

and

$$s_j\dot{W}_{j,k} + s_k\dot{Z}_{k,j} + 2\dot{s}_jW_{j,k} + 2\dot{s}_kZ_{k,j} = \quad (X^T\ddot{E}Y)_{k,j} + \sum_{i\neq j} W_{k,i}Q_{i,j} \quad \sum_{i\neq k} Q_{k,i}Z_{j,i}.$$

At a non-generic point we may subtract the second equation from the first to get

$$2(\dot{s}_k \quad \dot{s}_j)(W_{j,k} \quad Z_{k,j}) = r.h.s.$$

This is our second equation and we may use it to define W and Z (noting that the derivatives of S are known) except in the following special cases.

$_j = \ _k$ and $\dot{}_j = \dot{}_k$.

$_j = \quad _k$ and $\dot{}_j = \quad \dot{}_k$.

It should also be noted that the summations in the above equations may not be known explicitly if there are more than two identical singular values at a single point. Hopefully the "initial guess" should give sufficiently accurate estimates to the missing elements to be able to smooth out any error.

[1] Simon J.G. Bell and Nancy K. Nichols. Numerical solution of orthogonal matrix differential equations. Technical report, University of Reading, 1994.

[2] Angelika Bunse-Gerstner, Ralph Byers, Volker Mehrmann, and Nancy Nichols. Numerical computation of an analytic singular value decomposition. Technical report, University of Bielefeld, 1990.

[3] Angelika Bunse-Gerstner, Volker Mehrmann, and Nancy Nichols. Numerical methods for the regularization of descriptor systems by output feedback. Technical report, University of Minnesota, 1992.

[4] Luca Dieci, Robert Russel, and rik Van Vleck. Unitary integrators and applications to continous orthonormalization techniques. Preprint, 1992.

[5] Peter Kunkel and Volker Mehrmann. Smooth factorisation of matrix valued functions and their derivatives.

, 1990.

We now present a complete list of results obtained for the examples given in this paper and some examples not given. The algorithm ASVD4FU was used, both with and without stabilization. The tolerances $itol$ and $ntol$ were set to be h^2, where h was the step size, and the parameter $inmax$ was set at 2.

$$E(t) = X_1(t)S(t)X_1(t),$$

where

$$X_1(t) = \begin{matrix} c_1 & s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & c_2 & s_2 & 0 \\ 0 & s_2 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c_3 & s_3 \\ 0 & 0 & s_3 & c_3 \end{matrix},$$

$c_k = cos(k \quad 1 + t)$ and $s_k = sin(k \quad 1 + t)$, and

$$S(t) = diag(3 + t, \; diag(W1 + \quad)$$

Region of integration [0 2].

No non-generic points.

Analytic.

	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	3.91e-2	2.34e-2	yes	yes
2	0.2	7.81e-3	5.10e-3	yes	yes
3	0.1	1.60e-3	1.17e-3	yes	yes
4	0.04	2.61e-4	1.90e-4	yes	yes
5	0.02	6.63e-5	4.81e-5	yes	yes
6	0.01	1.68e-5	1.21e-5	yes	yes
7	0.004	2.70e-6	1.95e-6	yes	yes
8	0.002	6.77e-7	4.89e-7	yes	yes
9	0.001	1.69e-7	1.22e-7	yes	yes

	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	2.30e-2	1.35e-2	yes	yes
2	0.2	1.77e-3	6.02e-4	yes	yes
3	0.1	1.17e-4	2.30e-5	yes	yes
4	0.04	3.06e-6	3.12e-7	yes	yes
5	0.02	1.92e-7	1.10e-8	yes	yes
6	0.01	1.20e-8	3.70e-10	yes	yes
7	0.004	3.08e-10	4.00e-12	yes	yes
8	0.002	1.93e-11	1.25e-13	yes	yes
9	0.001	1.20e-12	5.38e-15	yes	yes

$$( ) = \quad _1( ) \quad ( ) \quad _1( )$$

where $_1$ is as before, and

where

$$\sigma_1 = \begin{cases} 1-(t-1)^4 & 0 \le t \le 1 \\ 1 & 1 < t < 2, \\ 1-(t-2)^4 & 2 \le t \le 3 \end{cases}$$

and

$$\sigma_2 = 1.$$

- Region of integration $t \in [0, 3]$.
- The two singular values coalesce at t=1 and remain equal until t=2 where they separate again.
- Non-analytic - four times differentiable.

Unstabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	1.91e+0	3.54e-1	no	no
2	0.2	8.29e-1	9.07e-2	no	no
3	0.1	5.91e-3	4.65e-3	no	no
4	0.04	1.02e-3	7.97e-4	no	no
5	0.02	2.65e-4	2.06e-4	no	no
6	0.01	6.71e-5	5.19e-5	no	no
7	0.004	1.08e-5	8.35e-6	no	no
8	0.002	2.70e-6	2.09e-6	no	no
9	0.001	6.74e-7	5.22e-7	no	no

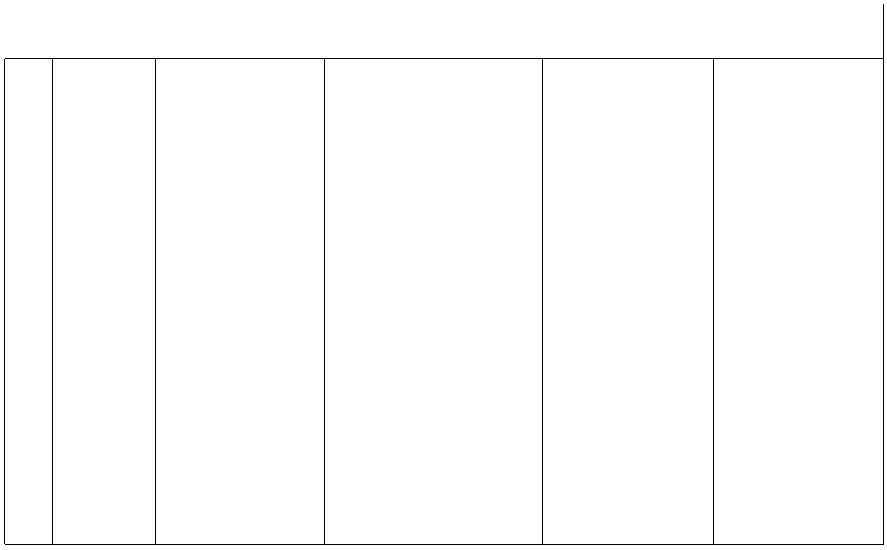
Unstabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	6.17e-1	1.58e-1	no	no
2	0.2	6.74e-1	2.94e-1	no	no
3	0.1	3.83e-1	6.75e-2	no	no
4	0.04	8.41e-1	2.53e-1	no	no
5	0.02	8.37e-1	2.49e-1	no	no
6	0.01	3.10e-5	1.98e-5	yes	yes
7	0.004	1.81e-2	6.35e-3	no	no
8	0.002	1.23e-6	7.86e-7	yes	yes
9	0.001	3.07e-7	1.96e-7	yes	yes

Stabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	5.32e-1	1.51e-1	no	no
2	0.2	5.52e-3	2.52e-3	yes	yes
3	0.1	1.50e-4	6.96e-5	yes	yes
4	0.04	3.53e-6	5.20e-7	yes	yes
5	0.02	2.17e-7	1.40e-8	yes	yes
6	0.01	1.34e-8	4.41e-10	yes	yes
7	0.004	3.41e-10	4.53e-12	yes	yes
8	0.002	2.13e-11	1.42e-13	yes	yes
9	0.001	1.33e-12	5.93e-15	yes	yes

1.6 Example 6

$$E(t) = \begin{cases} 4 \begin{bmatrix} cos(1/t) & -sin(1/t) \\ sin(1/t) & cos(1/t) \end{bmatrix} \begin{bmatrix} exp(-1/t^2) \\ 0 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} & t \neq 0, \\ \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & t = 0. \end{cases}$$

- Region of integration $t \in [-1, 1]$.
- No non-generic points, but a smooth ASVD path does not actually exist.



$$( ) = \quad _1( ) \ ( ) \ _1( )$$

where $( ) = \quad (_1 \ _2 \ _3 \ _4)$, and

$$_1 = \begin{matrix} 1 & (& 1)^4 & 0 & 1 \\ 1 & & & 1 & 2 \\ 1 & (& 2)^4 & 2 & 3 \end{matrix}$$

$$_2 = 1$$

$$_3 = -$$

$$_4 \quad - \quad ^{-10t}$$

1 2 3

4

Unstabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	1.42e+0	5.05e-1	?	?
2	0.2	4.92e-1	1.52e-1	no	no
3	0.1	6.85e-1	1.74e-1	no	no
4	0.04	5.94e-1	3.34e-1	no	no
5	0.02	4.77e-1	1.42e-2	no	no
6	0.01	5.79e-1	1.49e-2	no	no
7	0.004	9.62e-5	6.58e-5	no	no
8	0.002	2.10e-3	7.64e-5	no	no
9	0.001	6.59e-5	1.12e-5	no	no

Stabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	1.45e+0	5.02e-1	no	no
2	0.2	3.91e-1	1.05e-1	no	no
3	0.1	1.03e+0	1.18e-1	no	no
4	0.04	1.15e+0	3.93e-1	no	no
5	0.02	6.39e-1	2.15e-2	no	no
6	0.01	5.54e-2	4.46e-3	no	no
7	0.004	1.62e-2	1.18e-3	no	no
8	0.002	1.18e-2	8.44e-4	yes	yes
9	0.001	4.05e-3	2.92e-4	no	no

Strangely, the results without stabilization are more accurate for the singular values. However, the values for X and Y are absolutely hopeless, whereas for the stabilized results they are much better, even if they are only perfect for one of the step-sizes.