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

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

S.J.G. Bell , N.K. Nichols [1]

Numerical Analysis Report 2/94

Department of Mathematics
P.O. Box 220
University of Reading
Whiteknights
Reading
RG6 2AX
United Kingdom

[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.

Contents

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.

2 How Can the ASVD be Found?

It might be thought that to calculate the ASVD numerically, all one need do, would be to compute an SVD pointwise. This is not the case - such a sequence of SVD's would not in general be smooth, as for a given matrix $E(t)$ there is generally more than one possible smooth SVD path. A sequence of SVD's would tend to jump from path to path. This can be easily seen from the following example.

2.1 Example 1

Define

$$A(t) = \begin{bmatrix} 1-t & 0 \\ 0 & 1+t \end{bmatrix}.$$

Computing the SVD pointwise we get the following sequence of decompositions.

$$X(t)S(t)Y(t)^T = \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1-t & 0 \\ 0 & -1-t \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & t \in (-\infty, -1], \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1-t & 0 \\ 0 & 1+t \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & t \in (-1, 0], \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1+t & 0 \\ 0 & 1-t \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & t \in (0, 1], \\ \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1+t & 0 \\ 0 & -1+t \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & t \in (1, \infty). \end{cases}$$

So, the singular values are continuous but not differentiably so, and the elements of the singular factors X and Y are not even continuous. This is due to the SVD forcing the singular values to be non-negative and ordered, and illustrates the fact that we must allow these two conditions to be relaxed in the time-varying case if we are to get any sort of smooth decomposition. In many applications we wish to track these values in an (at least) continuous way. This is clearly not possible by a pointwise SVD calculation.

The fact that we are constraining our matrix to be analytic is no accident. Unless this condition is met, there is no guarantee that *any* smooth singular value

path exists, even for infinitely differentiable matrix functions. Bunse-Gerstner et al [2] discuss the non-uniqueness of the ASVD at length. They also derive an "Euler-type" numerical method for calculating the ASVD. The method works by calculating a standard SVD at each time-step and then calculating the permutation matrices needed to move it onto the correct path - the correct path in their case being that for which the norm of the derivative of $X(t)$ varies the least. We now show how the ASVD may be computed numerically by another, simpler method.

2.2 Differential Equations for the ASVD

Wright [6] proposed the following method for finding the ASVD by reducing it to a set of ordinary differential-algebraic equations and then solving these.

Define the ASVD

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

where $E(t) \in \Re^{m \times n}$. If we differentiate this we get

$$\dot{E} = \dot{X}SY^T + X\dot{S}Y^T + XS\dot{Y}^T.$$

Now, if we premultiply both sides of the above equation by X^T, then postmultiply by Y we get

$$X^T\dot{E}Y = X^T\dot{X}S + \dot{S} + S\dot{Y}^TY,$$

which, if we write $W = X^T\dot{X}$, $Z = Y^T\dot{Y}$ and $Q = X^T\dot{E}Y$ may be expressed more neatly as the following equation

$$Q = WS + \dot{S} + SZ^T. \tag{2}$$

This equation has a great deal of structure which is not immediately obvious. We can show that W and Z are skew-symmetric, since

$$\begin{aligned} 0 = \frac{dI}{dt} &= \frac{d}{dt}(X^TX), \\ &= \dot{X}^TX + X^T\dot{X}, \\ &= W^T + W. \end{aligned}$$

Hence, $W^T = -W$, and so W is skew-symmetric. Since Z is defined similarly to W, it too is skew-symmetric. An important property of skew-symmetric matrices is that they have zero diagonals. Thus, since S is a diagonal matrix, the only

contribution to the matrix $\dot{S}$ in equation (2) comes from the diagonal of Q. Therefore, using the definitions of W and Z we may express the ASVD in terms of the following set of equations

$$diag(\dot{S}(t)) = diag(Q(t)), \tag{3}$$

$$\dot{X}(t) = X(t)W, \tag{4}$$

$$\dot{Y}(t) = Y(t)Z, \tag{5}$$

where the elements of Z and W are given by the algebraic equations

$$s_k W_{j,k} + s_j Z_{k,j} = Q_{j,k}, \tag{6}$$

$$s_j W_{j,k} + s_k Z_{k,j} = -Q_{k,j}, \tag{7}$$

where $j > k$. If $m > n$, ie. $E(t)$ is rectangular, we have an extra set of equations thus

$$s_k W_{j,k} = Q_{j,k}, \tag{8}$$

where $j = n+1, \ldots, m$, $k = 1, \ldots, n$. These, rearranged, give

$$W_{j,k} = \frac{s_k Q_{j,k} + s_j Q_{k,j}}{s_k^2 - s_j^2}, \tag{9}$$

$$Z_{k,j} = \frac{s_j Q_{j,k} + s_k Q_{k,j}}{s_j^2 - s_k^2}, \tag{10}$$

$$W_{j,k} = \frac{Q_{j,k}}{s_k}. \tag{11}$$

Then, setting $W_{k,j} = -W_{j,k}$ and $Z_{j,k} = -Z_{k,j}$ determines the other entries and guarantees anti-symmetry.

2.3 Some Points of Interest

2.3.1 A Problem

It is obvious from looking at equations (6) and (7) in the form of (9) and (10) that we have a problem when $s_j^2 = s_k^2$. If this is so, then the two equations (6) and (7) become linearly dependent, ie. only one equation exists for every two unknowns $W_{j,k}$ and $Z_{k,j}$. If $s_j^2 \equiv s_k^2$, ie. the two singular values are identical for all t, then there is a degree of freedom in the ASVD. One of the two unknowns may be chosen arbitrarily if the two singular values are non-zero, both may be set arbitrarily if the two singular values are identically zero (although they must be

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 *non-generic point*. When this happens we lose one or both of the equations for $W_{j,k}$ and $Z_{k,j}$, but there is no corresponding degree of freedom in the ASVD - the path is still uniquely defined, assuming analyticity. We are therefore left with a system for which the solution is perfectly well-defined and unique [6], but does not give us sufficient information to find the solution at such points.

Similarly, from (11), when E(t) is rectangular, points at which a singular value becomes instantaneously zero are defined as non-generic points, whilst identically zero singular values allow the corresponding W elements to be set to any analytic function.

These non-generic points represent the only real problem in finding the ASVD For all other points, provided that the initial value of t is *generic*, the equations (3) - (8) uniquely define an ASVD path for a given initial SVD.

2.3.2 An Observation

The differential equations (4) and (5) are of the form

$$\dot{X} = X(t)W(X,t),$$

where W is skew symmetric which implies that X is orthogonal. All matrix functions are analytic, so all matrices are differentiable and bounded. We would like to solve these equations in such a way that the orthogonality of the left and right factors is preserved, without having to reorthogonalise them at each step. Most standard methods will not do this (see [4]). We, therefore, look to a non-standard method and apply the following scheme from [1] to solve these equations.

$$X_{n+1} - X_n = \frac{h}{4}(X_{n+1} + X_n)(W_{n+1} + W_n), \tag{12}$$

but rewritten in the semi-explicit form

$$X_{n+1} = X_n(I + B_n)(I - B_n)^{-1}, \tag{13}$$

where $B_n = \frac{h}{4}(W_{n+1} + W_n)$.

This scheme has the great advantage that, provided B_n is skew symmetric and X_0 is orthogonal, (13) always returns an estimate X_{n+1} which is orthogonal

to machine accuracy. Equation (13) is thus used in conjunction with the forward extrapolation technique described in [1] as a predictor-corrector pair. The theory on convergence of [1] gives sufficient conditions on the step size h for convergence of this method and non-generic points are dealt with by leaving the initial estimates for the relevant W or Z entries in place and only recalculating those for which equations exist. We illustrate this with a basic version of our ASVD algorithm for a constant step-size (ASVD4FU).

2.4 ASVD Algorithm ASVD4FU

1. Input $t_0 = a, t_n = b, h, itol, ntol, inmax$

2. Calculate initial SVD
$$X_0 S_0 Y_0 = E(t_0)$$

3. Calculate
$$Q_0 = X_0^T \dot{E}(t_0) Y_0$$

4. Calculate W_0 and Z_0
call wzcal(W_0, Z_0, S_0, Q_0)

5. Outer loop $n = 1, 2, \ldots$

   - Set $S^{OLD} = S_{n-1}$
   - Inner loop $incn = 1, 2, \ldots, inmax$
     - Use forward extrapolation on W_n and Z_n
       If incn=1 then $W_n = 3W_{n-1} - 3W_{n-2} + W_{n-3}$
       $Z_n = 3Z_{n-1} - 3Z_{n-2} + Z_{n-3}$
       else $W_n = W_{n-1}$
       $Z_n = Z_{n-1}$
       endif
     - Calculate X_n and Y_n
$$B = h(W_n + W_{n-1})/4$$
$$C = h(Z_n + Z_{n-1})/4$$
$$X_n = X_{n-1}(I + B)(I - B)^{-1}$$
$$Y_n = Y_{n-1}(I + C)(I - C)^{-1}$$

– Calculate Q_n and S_n

$$Q_n = (X_n)^T \dot{E}(t_n) Y_n$$

$$diag(S_n) = \frac{h}{2} * diag(Q_{n-1} + Q_n) + diag(S_{n-1})$$

– Calculate W_n and Z_n

call wzcal(W_n, Z_n, S_n, Q_n)

– Decide whether to exit inner loop

If $\|S_n - S^{OLD}\|_2 > itol$ then $S^{OLD} = S_n$

next incn

Endif

- Write X_n, S_n, Y_n

  $t_n = t_n + h$

  if $t_n < t_f$ then next n

- End

subroutine wzcal(W, Z, S, Q)

do 5, j=2,n

do 10, k=1,j-1

$sqd = S^2(j) - S^2(k)$

if $sqd \geq ntol$ then

$W = (S(k)Q(j,k) + S(j)Q(k,j))/sqd$

$Z = -(S(j)Q(j,k) + S(k)Q(k,j))/sqd$

endif

$W(k,j) = -W(j,k)$

$Z(j,k) = -Z(k,j)$

10 next k

5 next j

return

end

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. Experimentally 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.

3 Examples

We now present a selection of interesting examples. They do not represent all of the examples tested so far (a full list of all examples tested with various step-sizes is given in Appendix 1), but they exhibit most of the important possible characteristics. The errors we give are the relative errors in the approximated singular values, ie.

$$error(t_n) = \frac{\|S(t_n) - S_n\|_2}{\|S(t_n)\|_2}.$$

Where graphs are shown, those depicting approximations to singular values are labelled "Singular Values" and show all of the values on the one graph; those depicting approximations to the singular factors are labelled "Left/Right Singular Factor" and show the diagonal elements of the relevant matrix.

3.1 Example 2

This is of the form

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

where $X(t)$ is given by

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

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

$$S(t) = diag(t, 1 + t, 2 + t, 3 + t).$$

This is a nice, analytic matrix with no non-generic points anywhere. If we apply our algorithm to it over the interval $t \in [0, 2]$ with a step-size $h = 0.01$, we get the singular value paths shown in Figure 1. As we would expect, there are no problems anywhere, although a point of minor interest is that the lower singular value is $-t$, as opposed to t. This is perfectly acceptable though: since we are starting at $t = 0$, either function is perfectly correct - in other words, there is a certain amount of (trivial) non-uniqueness at this point. Overall error (Figure 2) is of the order 10^{-5}, and as the step-size is changed is seen to be consistent with the algorithm being second order (See Appendix 1).

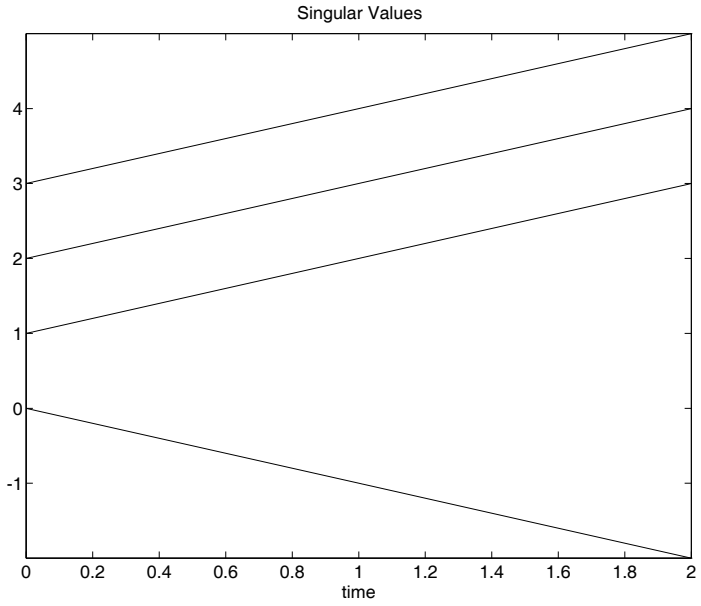

Figure 1: Example 2

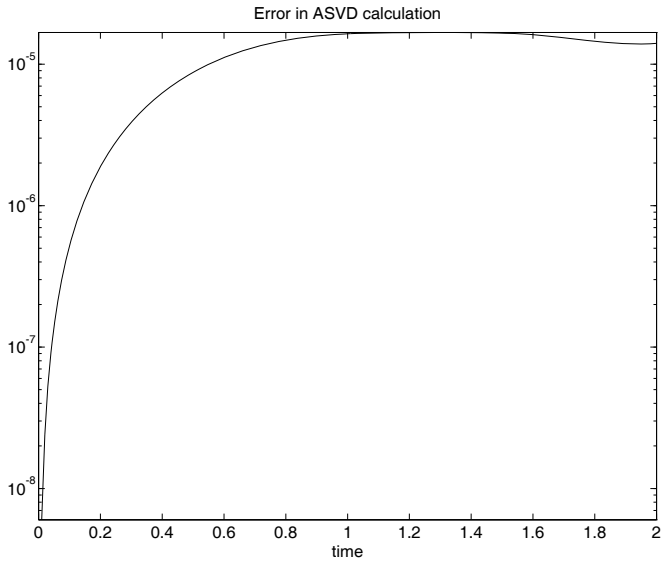

Figure 2: Example 2

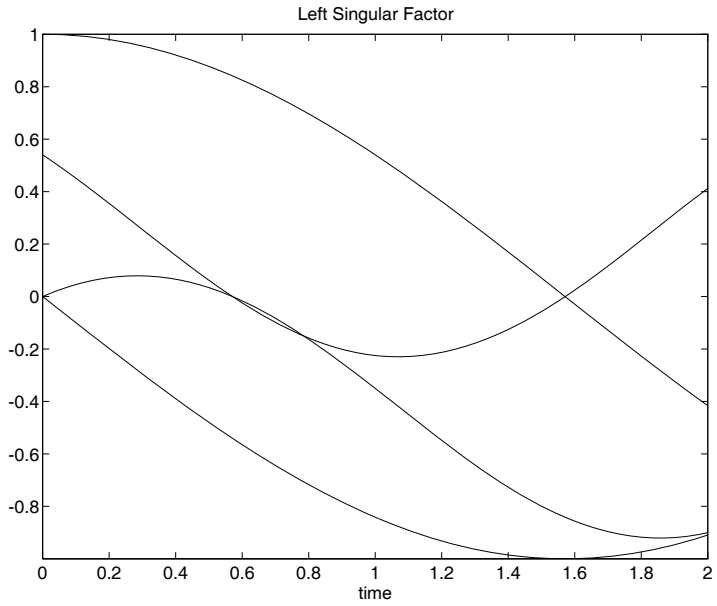

Figure 3: Example 2

Inspection of the approximation to $X(t)$ (Figure 3) shows a perfectly smooth (at least to the naked eye) set of functions. There are no surprises here.

3.2 Orthogonality of Left and Right Factors

One of the greatest assets of our algorithm is the fact that the orthogonality of the left and right factors is preserved - perfectly in theory, to machine error in practice. It would be good, therefore, before going any further, to demonstrate that this is indeed the case.

We use the previous example, but this time the graph (Figure 4) shows the quantity

$$\|X_n X_n^T - I\|_2,$$

in other words, it gives a measure of the loss of orthogonality as we step forwards in time.

Using standard difference techniques, the loss of orthogonality is of the same order as the global truncation error [1]. Using our algorithm with the orthogonality preserving difference scheme, we get the results shown in Figure 4. These are very pleasing - the calculations were carried out in Matlab using twelve point precision, the loss of orthogonality is of the order 10^{-14} to 10^{-15}. Shortening the step size seems to produce no dramatic effect on this loss of orthogonality.

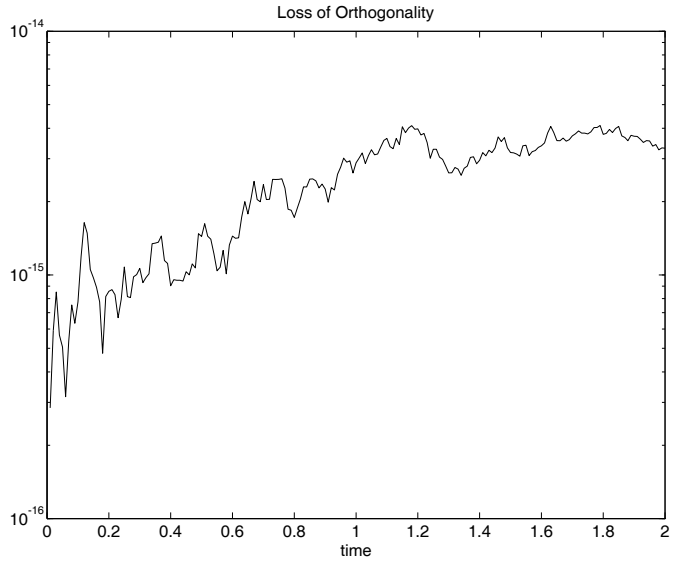

Figure 4: Example 2

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

3.3 Example 3

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

$$S(t) = diag(.5 + t, 2 - t, 1 - t, t),$$

and evaluated over the same interval $t \in [0, 2]$, with the same step-size $h = 0.01$.

This does have non-generic points, occurring at $t = 0.25, 0.5, 0.75, 1.0, 1.5$. Therefore we might expect worse results, certainly at the non-generic points. Examination of the singular values calculated seems to suggest that everything is fine; they are nice and smooth, and look accurate enough to the naked eye (Figure 5). When we look at the error (Figure 6) we see that this is indeed the case. The results are comparable with the previous example, giving an error of order 10^{-5} away from the initial value. As before, the left factor seems perfectly smooth (Figure 7).

3.4 Example 4

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

$$S(t) = diag(1, t, 2 - t, 3 - 2t).$$

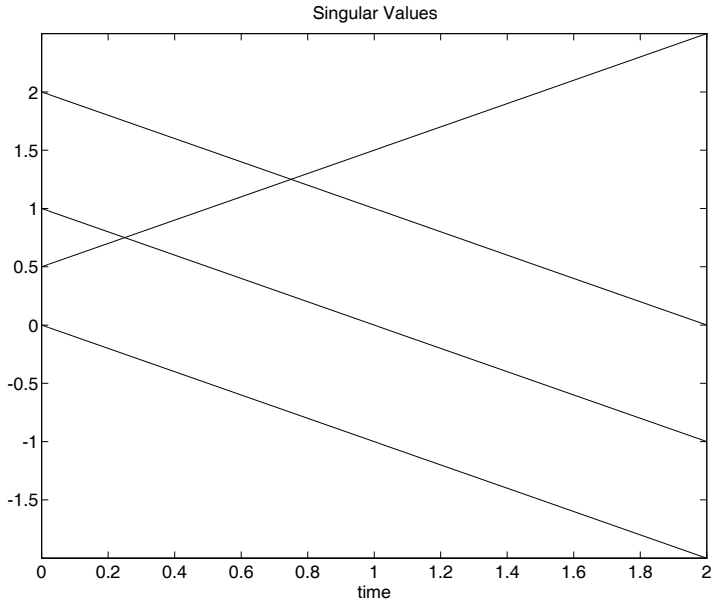

Figure 5: Example 3

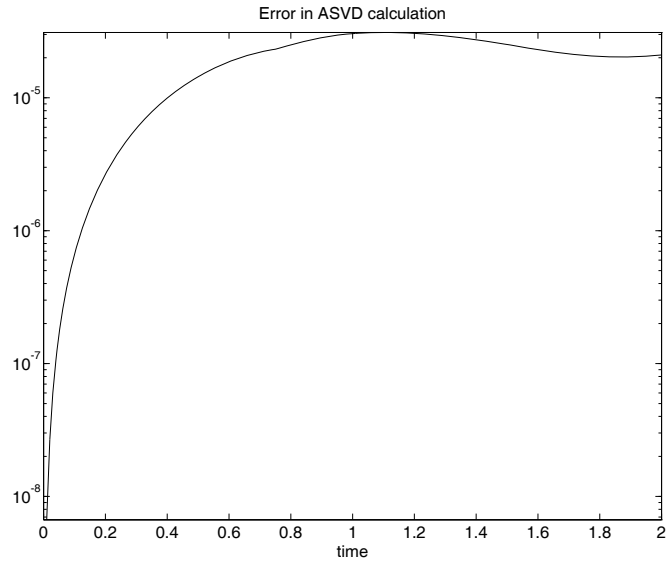

Figure 6: Example 3

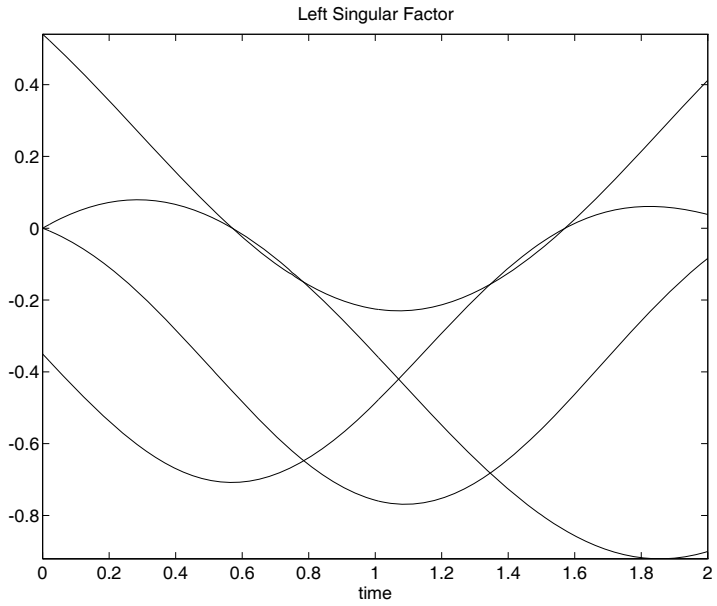

Figure 7: Example 3

All four singular values intersect at $t = 1$, and there are further non-generic points at $t = \frac{5}{3}$ and $t = 2$.

Evaluated over $t \in [0, 2]$ with a step-size $h = 0.01$, this example poses a real problem. The singular values (Figure 8) happily follow the correct path until t=1, where they receive a knock which pushes them onto new paths. This is confirmed by the error obtained (Figure 9) which is reasonable (order 10^{-5}) until the first non-generic point. Inspection of the entries of the left singular matrix X(t) (Figure 10) shows that the values of the singular factors are fine up to this point, and hopeless afterwards. These are results obtained using only two iterations in the inner loop of the algorithm ($inmax = 2$). Slightly better results can be obtained if more are allowed. If, say $inmax = 100$, then the method converges more satisfactorily at t=1, but fails at t=5/3. The only way to resolve this problem is to go down to a smaller step-size (say $O(10^{-3})$).

3.5 Example 5

This is similar to the previous examples. However, whilst the matrix $X(t)$ has the same structure as before, its entries are now $c_1 = cos(t)$, $c_2 = cos(t/2)$ and $c_3 = cos(t/4)$, etc. The singular values are given by

$$S(t) = diag(1, t, t^2, t^3).$$

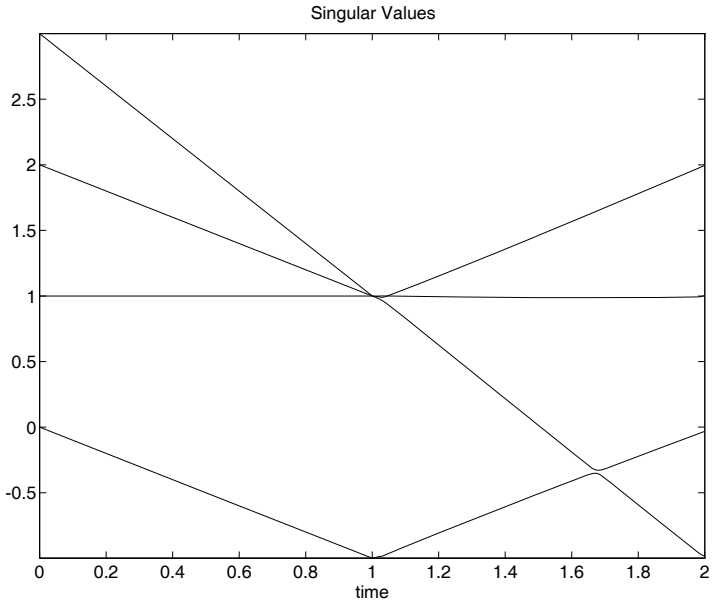

Figure 8: Example 4

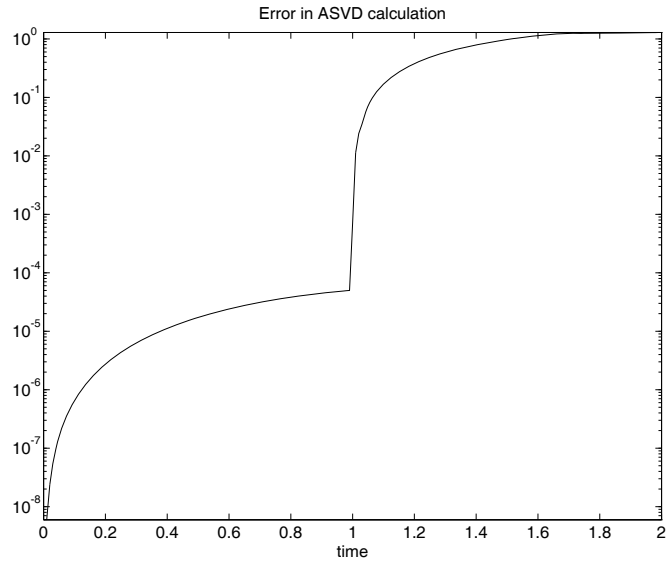

Figure 9: Example 4

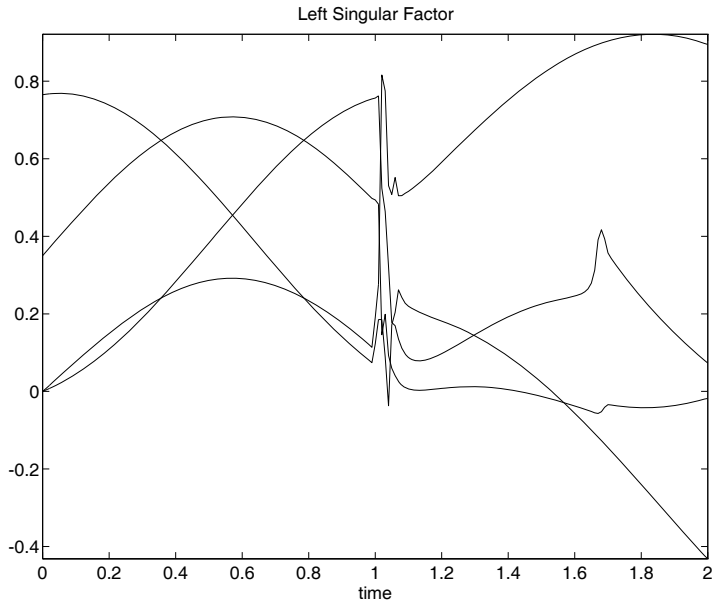

Figure 10: Example 4

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. Experimentally, 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

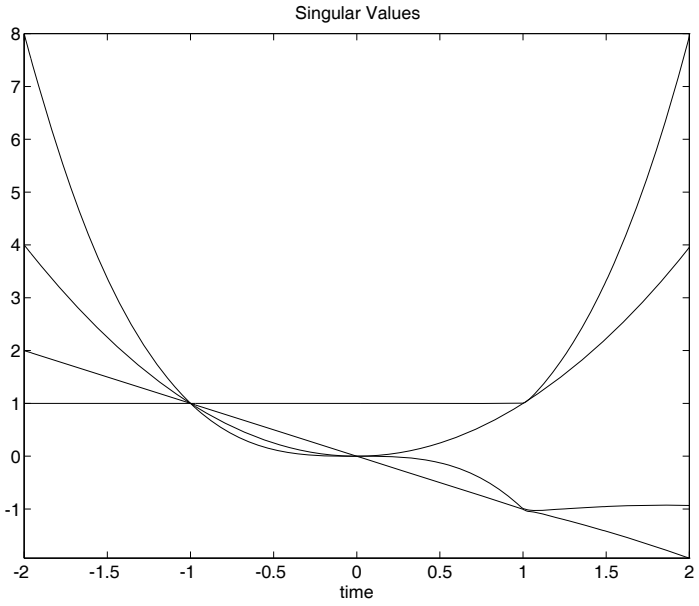

Figure 11: Example 5

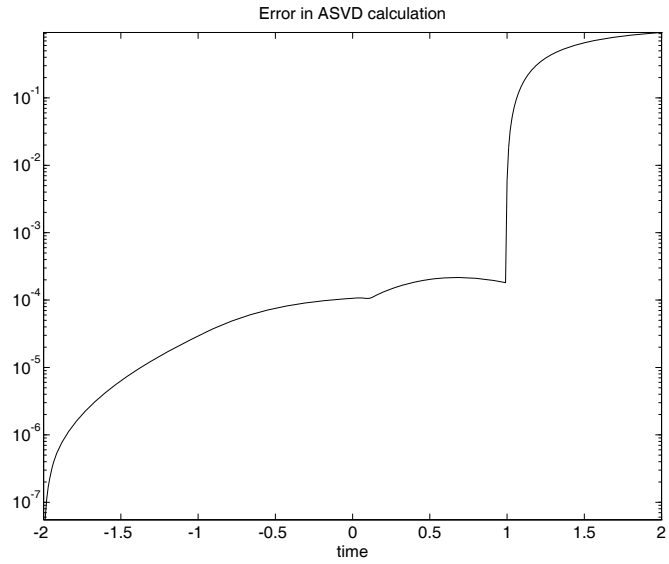

Figure 12: Example 5

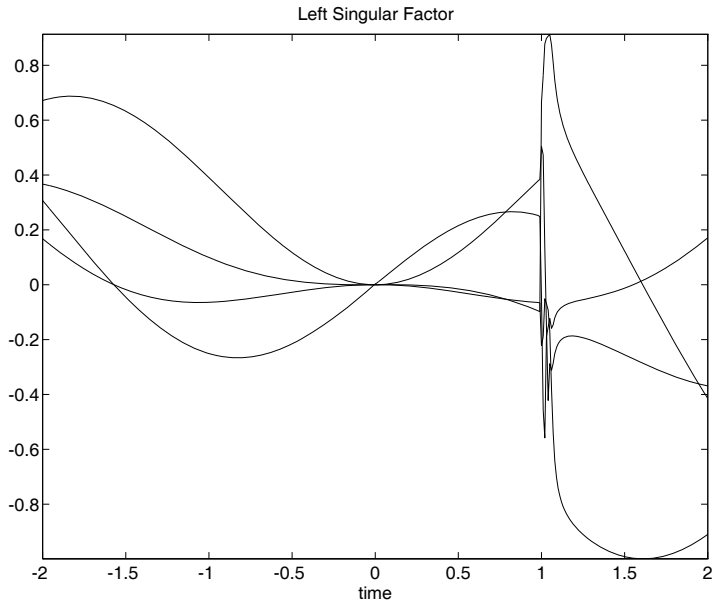

Figure 13: Example 5

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.

4 Stabilization

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 derivatives of Q. Further, if the first derivatives of two singular values should also happen to be equal in modulus, then the equations for W and Z become linearly dependent once again; the next derivative has to be taken, and so on. All in all, with the extra complicated terms involved (the derivative of Q can be replaced by other "known" terms, but it is messy), and with the extra special cases involving derivatives of singular values, this does not seem to be a viable option.

Another idea is due to Wright [6] and this involves reducing the accumulated error by replacing the exact matrix E(t) at each step by a perturbed one, C(t) say, of the form

$$C(t) = E(t) + (E_n - E(t_n))\frac{(t - t_{n+1})}{(t_n - t_{n+1})}, \tag{14}$$

where $t \in [t_n, t_{n+1}]$, $E(t_n)$ is the exact value of E at t_n and E_n is our approximated value, given by

$$E_n = X_n S_n Y_n^T.$$

This has the properties

$$C(t_n) = E_n,$$

$$C(t_{n+1}) = E(t_{n+1}).$$

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 E replaced by that of C, where C is our "stabilization matrix". The examples, too, are the same - Example 2 of this section coincides with Example 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.

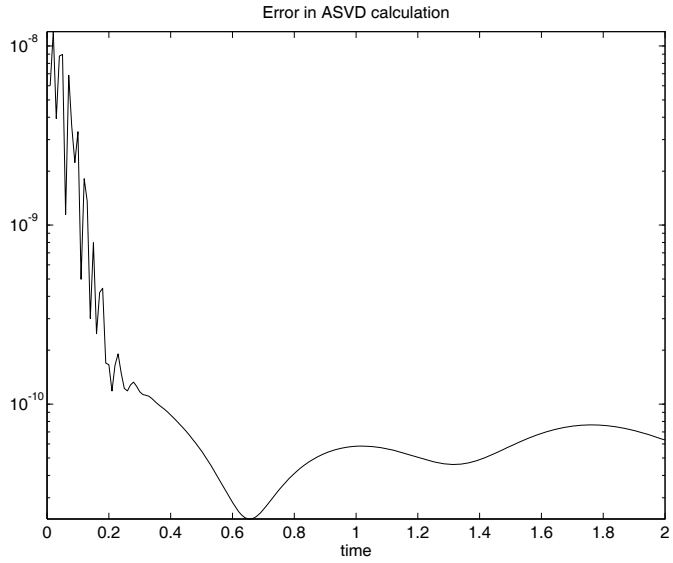

Figure 14: Stabilized Example 2

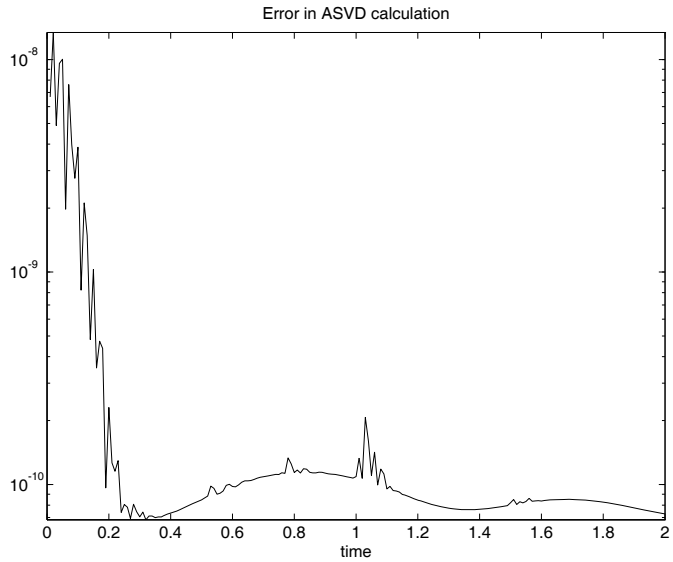

Figure 15: Stabilized Example 3

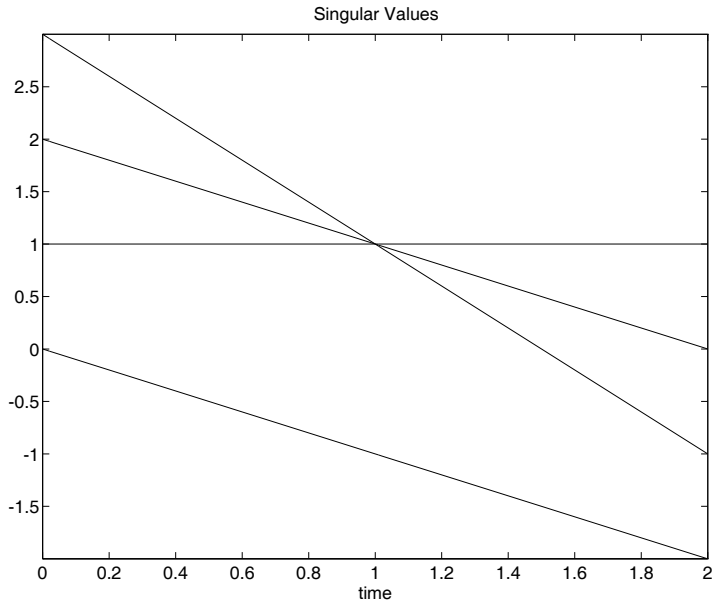

Figure 16: Stabilized Example 4

4.4 Example 4

This is one of the "problem" examples from the previous section. For a step greater than $h = 0.001$ the algorithm failed entirely, and the step-size had to be reduced for it to converge at the non-generic points. Figure 16 shows the singular values that we obtain - all on the correct paths, and Figure 17 shows the error - initially order 10^{-8}, decaying away (though not so rapidly as before) to order 10^{-11}. However, good singular values do not imply good singular factors. We must therefore look to Figure 18 which displays $X(t)$. Once again, there are no problems. Without stabilization, the method failed on this example until the step-size was made as small as 0.001. Now, the method works for a step as large as $h = 0.1$.

These results, and many others, suggest that, somehow, stabilization raises the order of the method to four. This is exciting, as, it puts the method on a par with a good implicit Runge-Kutta scheme [7] for these examples, with orthogonality preserved to machine accuracy, but with far fewer calculations.

4.5 Example 5

Now the results are slightly worse. Figure 19 shows the computed singular values, and Figure 20 the error. Everything is in the correct place, but the error for this example is slightly worse than for the two previous examples. Why this should

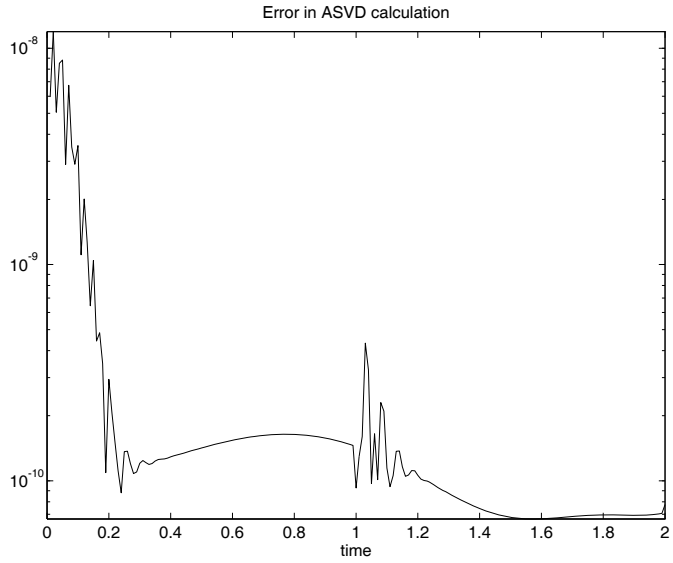

Figure 17: Stabilized Example 4

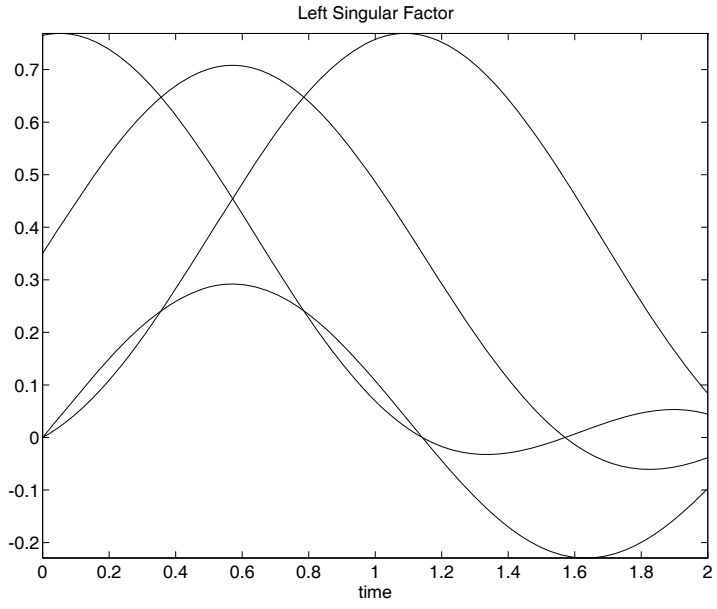

Figure 18: Stabilized Example 4

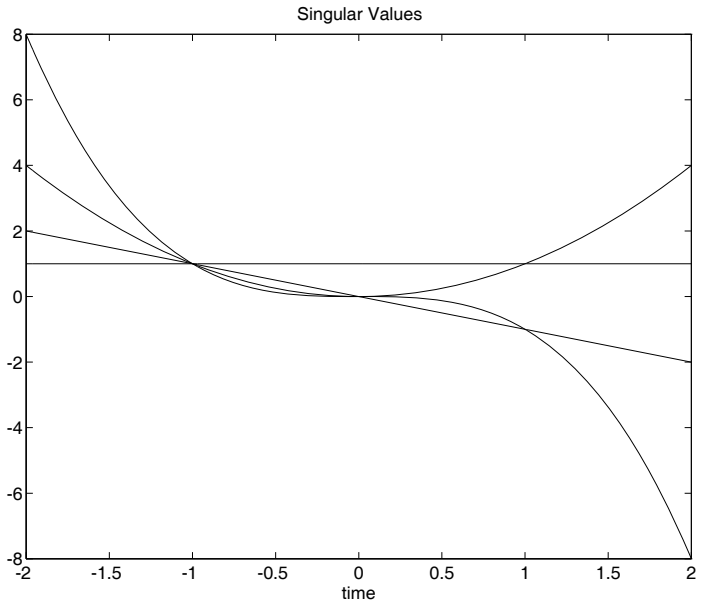

Figure 19: Stabilized Example 5

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).

4.6 Thoughts on these Examples

It has been seen that stabilization has an enormous effect on the accuracy of the solutions, raising the order of the algorithm by at least one, sometimes more. What is more, the stabilization procedure is extremely cheap to implement, involving only a few divisions and matrix additions at each step. And, very importantly, none of the examples tested required more than two fixed point iterations at any step. This means that the fixed point iteration can be disposed of; all that is required is the "initial guess" step and then one more run through the calculations to correct the values. What we do not know is why stabilization works as well as it does.

It does, however, seem to work extremely well, and, with this in mind, we shall apply the method to one last problem. This is of interest because it is not analytic.

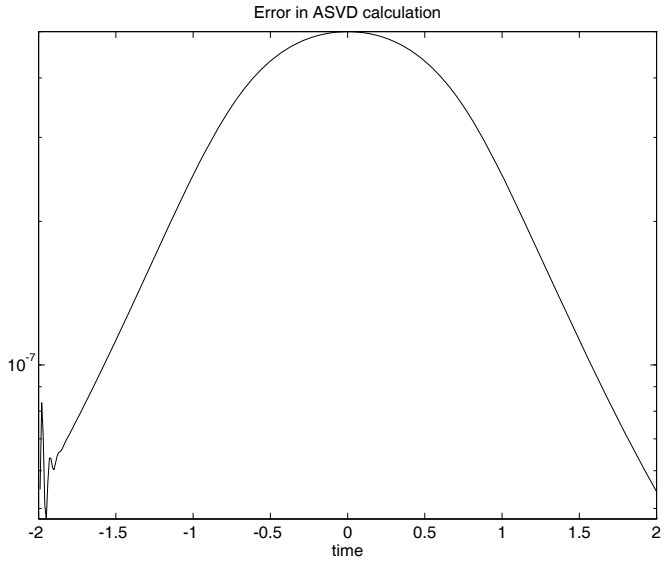

Figure 20: Stabilized Example 5

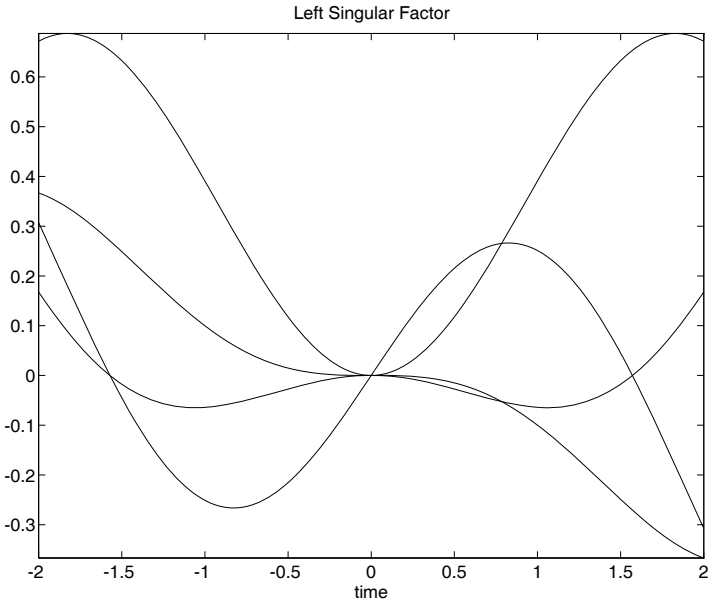

Figure 21: Stabilized Example 5

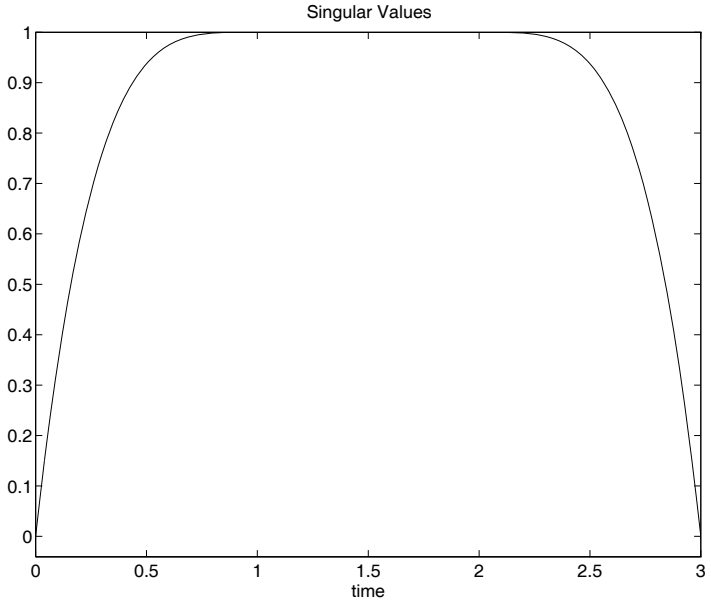

Figure 22: Stabilized Example 6

4.7 Non-Analytic Example 6

This is of the form

$$E(t) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix},$$

where

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

$$\sigma_2 = 1 \quad t \in [0,3],$$

and $c = cos(t), s = sin(t)$.

Despite the fact that this example is C^3, the method without stabilization fails completely. The singular values are tracked very accurately, but the subspaces bifurcate at t=1 and t=2. Most of the theory of this paper, however, relies on all matrix functions being no more than three-times differentiable [1], so with the aid of the stabilization, it should be hoped that the method will cope. Figures 22 - 24 show the results with a step size of 0.01. The error in this case has a strange symmetry: it remains between 10^{-7} and 10^{-6} everywhere except in the interval $[1,2]$ where it plummets to effectively zero. This is probably something

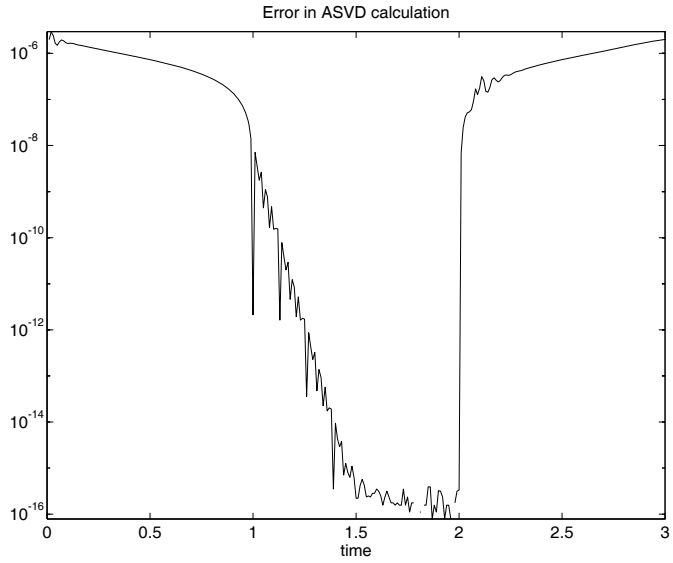

Figure 23: Stabilized Example 6

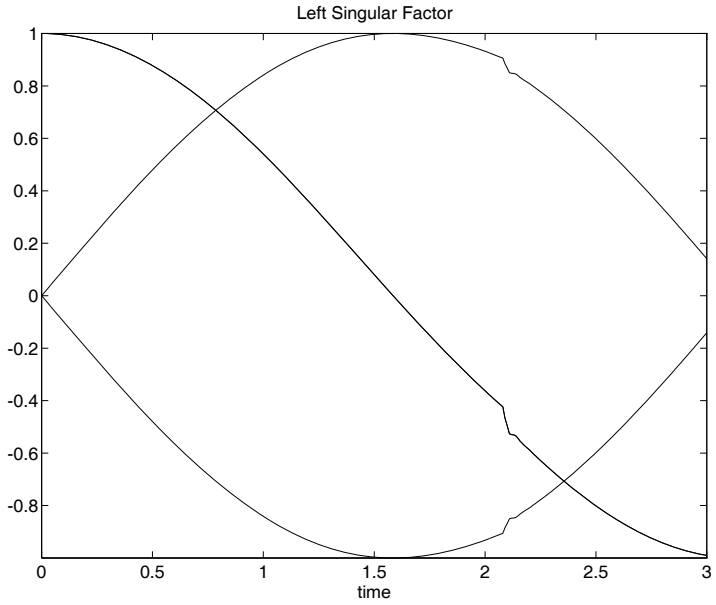

Figure 24: Stabilized Example 6

to do with the simplicity of the system over this interval, as it reduces to just

$$E(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The left and right singular factors are not perfect - there is a slight blip after $t = 2$. They are not knocked off course, however, but settle back down again. Reducing the step size irons out these blips.

Although these results are less accurate than for previous analytic examples, what is exciting and important is that the algorithm works at all. The accuracy is certainly good enough for many practical purposes, and the fact that everything works suggests that the algorithm as it stands has excellent potential as a method for finding the ASVD of a matrix.

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

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

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

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

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.

- $s_j = s_k$ and $\dot{s}_j = \dot{s}_k$.
- $s_j = -s_k$ and $\dot{s}_j = -\dot{s}_k$.

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.

6

References

[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 Erik 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. *Institut fur Geometrie und Praktische Mathematik*, 1990.

[6] K.Wright. Differential equations for the analytic singular value decomposition of a matrix. Technical report, University of Newcastle, 1991.

[7] K. Wright. Numerical solution of differential equations for the analytic singular value decomposition. Technical report, University of Newcastle, 1993.

A Appendix 1

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.

1.1 Example 1

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

where

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

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

$$S(t) = diag(3+t, 2+t, 1+t, t).$$

- Region of integration $t \in [0, 2]$.
- No non-generic points.
- Analytic.

Unstabilized					
	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

Stabilized					
	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

1.2 Example 2

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

where X_1 is as before, and

$$S(t) = diag(1, t, 2-t, 3-2t).$$

- Region of integration $t \in [0, 2]$.
- Non-generic points: all four singular values coincide at t=1, other crossings at t=5/3 and t=2.
- Analytic.

Unstabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	4.15e-1	1.71e-1	no	no
2	0.2	7.44e-1	2.71e-1	no	no
3	0.1	1.33e+0	4.01e-1	no	no
4	0.04	7.58e-1	1.85e-1	no	no
5	0.02	1.03e+0	3.45e-1	no	no
6	0.01	1.29e+0	4.25e-1	no	no
7	0.004	8.10e-1	1.87e-1	no	no
8	0.002	5.74e-1	5.34e-2	no	no
9	0.001	5.76e-1	5.28e-2	no	no

Stabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	2.96e-1	6.91e-2	?	?
2	0.2	3.08e-3	1.66e-3	yes	yes
3	0.1	1.30e-4	4.88e-5	yes	yes
4	0.04	3.15e-6	4.67e-7	yes	yes
5	0.02	1.93e-7	1.43e-8	yes	yes
6	0.01	1.19e-8	4.39e-10	yes	yes
7	0.004	3.03e-10	4.46e-12	yes	yes
8	0.002	1.89e-11	1.39e-13	yes	yes
9	0.001	1.18e-12	6.73e-15	yes	yes

The first table shows how totally the unstabilized algorithm fails on this example. This failure is due to the X and Y values jumping at each non-generic point. With a larger value of $inmax$, slightly better results are possible for small step-sizes, but, in general, we may say that the unstabilized algorithm fails on this example.

The stabilized algorithm, however, gives excellent results - the question marks in the first two rows refer to the fact that insufficient data points were available for us to be able to tell whether the X and Y matrices were smooth or not.

1.3 Example 3

$$E(t) = X_2(t)S(t)X_2(t)$$

where X_2 has the same structure as X_1, but now $c_1 = cos(t)$, $c_2 = cos(t/2)$, $c_3 = cos(t/4)$, $s_1 = sin(t)$, $s_2 = sin(t/2)$ and $s_3 = sin(t/4)$, and

$$S(t) = diag(1, t, t^2, t^3).$$

- Region of integration $t \in [-2, 2]$.
- Non-generic points: all four singular values coincide at t=$\pm$1, three coincide at t=0.
- Analytic.

Unstabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	1.62e+0	7.88e-1	no	no
2	0.2	9.41e-1	3.21e-1	no	no
3	0.1	6.29e-1	1.06e-1	no	no
4	0.04	6.57e-1	1.04e-1	no	no
5	0.02	1.01e+0	1.58e-1	no	no
6	0.01	9.35e-1	1.49e-1	no	no
7	0.004	1.07e-1	1.71e-1	no	no
8	0.002	3.06e-1	5.25e-2	no	no
9	0.001	1.07e-1	1.71e-1	no	no

Stabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	1.09e+0	4.18e-1	no	no
2	0.2	1.63e+0	5.09e-1	no	no
3	0.1	5.25e-4	2.65e-4	yes	yes
4	0.04	3.20e-5	1.71e-5	yes	yes
5	0.02	4.00e-6	2.14e-6	yes	yes
6	0.01	5.00e-7	2.69e-7	yes	yes
7	0.004	3.20e-8	1.72e-8	yes	yes
8	0.002	4.00e-9	2.15e-9	yes	yes
9	0.001	5.00e-10	2.69e-10	yes	yes

For all of the unstabilized results and the first two stabilized results, the elements of X and Y jumped at t=1.

1.4 Example 4

$$E(t) = \begin{bmatrix} cos(t) & sin(t) \\ -sin(t) & cos(t) \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} cos(t) & sin(t) \\ -sin(t) & cos(t) \end{bmatrix},$$

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

Stabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	1.26e+0	2.17e-1	no	no
2	0.2	2.15e-2	6.22e-3	yes	yes
3	0.1	2.76e-3	6.12e-4	yes	yes
4	0.04	1.76e-4	3.57e-5	yes	yes
5	0.02	2.32e-6	4.82e-6	yes	yes
6	0.01	2.96e-6	5.57e-7	yes	yes
7	0.004	1.91e-7	3.52e-8	yes	yes
8	0.002	2.39e-8	4.41e-9	yes	yes
9	0.001	3.00e-9	5.51e-10	yes	yes

The unstabilized results demonstrate the fact that the singular values can be perfectly okay, even if the X and Y values are going haywire.

1.5 Example 5

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

where

$$S(t) = diag(t + \frac{1}{2}, 2 - t, 1 - t, t).$$

- Region of integration $t \in [0, 2]$.
- Non-generic points at $t = 0.25$, $t = 0.5$, $t = 0.75$, $t = 1.0$ and $t = 1.5$.
- Analytic.

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.

- Non-analytic, infinitely differentiable.

Unstabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	1.05e+0	3.32e-1	?	yes
2	0.2	1.02e+0	2.26e-1	?	yes
3	0.1	4.01e-1	9.17e-2	no	yes
4	0.04	1.77e-1	3.99e-2	yes	yes
5	0.02	6.08e-3	1.54e-3	yes	yes
6	0.01	2.08e-1	4.33e-2	no	yes
7	0.004	3.96e-3	6.90e-4	yes	yes
8	0.002	3.98e-3	5.37e-4	yes	yes
9	0.001	2.33e-3	3.24e-4	yes	yes

Stabilized					
	h	Max error	Average error	Smooth X	Smooth Y
1	0.4	1.11e+0	3.78e-1	?	yes
2	0.2	3.62e-2	1.64e-2	?	yes
3	0.1	9.31e-2	1.49e-2	no	yes
4	0.04	1.08e-1	9.09e-3	yes	yes
5	0.02	2.05e-2	7.60e-4	yes	yes
6	0.01	1.66e-2	3.86e-4	yes	yes
7	0.004	3.87e-3	6.02e-5	yes	yes
8	0.002	1.89e-3	2.50e-5	yes	yes
9	0.001	1.26e-3	1.49e-5	yes	yes

As before, the question marks in the above tables indicate that it was impossible to say whether the X entries were smooth.

It would be somewhat optimistic to expect sensible results for X in a neighbourhood of zero, so "Smooth X" for this example implies that the entries of X either side of a small neighbourhood of $t = 0$ were smooth (typically entries outside $t \approx \pm 0.2$).

1.7 Example 7

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

where $S(t) = diag(s_1, s_2, s_3, s_4)$, and

$$\begin{aligned} s_1 &= \begin{cases} 1-(t-1)^4 & 0 \le t \le 1, \\ 1 & 1 < t < 2, \\ 1-(t-2)^4 & 2 \le t \le 3, \end{cases} \\ s_2 &= 1, \\ s_3 &= \frac{1}{2}, \\ s_4 &= \frac{1}{2} + e^{-10t}. \end{aligned}$$

- Region of integration $t \in [0, 3]$.
- Non-generic points at

$$\begin{aligned} &(i) && t = 0.06931471805599, \\ &(ii) && t = 0.15910358474629, \\ &(iii) && t = 0.21378427872548, \\ &(iv) && t = 2.84089641525352, \\ &(v) && t = 2.84089641525371. \end{aligned}$$

  Singular values s_1 and s_2 are identical for $t \in [1, 2]$. Singular values s_3 and s_4 are almost identical from $t = 0.5$ onwards (hence the two very close non-generic points).

- Non-analytic, four times differentiable.

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.