Path following by SVD

Luca Dieci[1], Maria Grazia Gasparo[2], and Alessandra Papini[3]

[1] School of Mathematics, Georgia Institute of Technology, Atlanta GA 30332, USA,
`dieci@math.gatech.edu`,
[2] Università di Firenze, Dip. Energetica "S. Stecco", via C. Lombroso 6/17,
I-50134 Firenze, Italia,
`mariagrazia.gasparo@unifi.it`,
[3] Università di Firenze, Dip. Energetica "S. Stecco", via C. Lombroso 6/17,
I-50134 Firenze, Italia,
`alessandra.papini@unifi.it`

Acknowledgement. This work was supported in part under NSF-DMS Grant 0139895, INDAM-GNCS Rome-Italy, and MIUR Rome-Italy.

Subject Classifications: 65F15, 65F99

Abstract. In this paper, we propose a path-following method for computing a curve of equilibria of a dynamical system, based upon the smooth Singular Value Decomposition (SVD) of the Jacobian matrix. Our method is capable of detecting fold points, and continuing past folds. It is also able to detect branch points and to switch branches at such points. Algorithmic details and examples are given.

1 Introduction

One of the most important and recurring problems in applications is that of finding solutions of overdetermined nonlinear systems, $f(u) = 0$, where f is a C^k map, $k \geq 1$, from (part of) $R^{n+1} \to R^n$. Standard occurrences of this situation are homotopy techniques for solving nonlinear systems, see [15], and also techniques to find equilibria or periodic solutions of parameter dependent dynamical systems, see [11]. In such case, the system is usually written as

$$f(x, \alpha) = 0, \quad f : R^n \times R \to R^n. \tag{1}$$

As it is well understood, assuming that 0 is a regular value for f, the solution set of $f(u) = 0$, is a C^k curve (e.g., see [10]); we stress that, for (1), this does not mean that the curve can be globally parametrized in α. Computation of this regular curve of equilibria is the task of path-following algorithms, or continuation methods. The successful continuation methods are of *predictor-corrector* type: From knowledge of a point on the curve, they seek a new point on the curve at a certain (arc-length) distance from the present one, by iteratively *correcting* an initial *prediction* of this new point. The standard prediction stage is carried out by a tangent (or Euler) approximation, and the standard correction is performed by Newton's method or one of its variants. We refer to [1] for an excellent overview of continuation techniques. Now, while continuing the curve of equilibria, a robust algorithm must also be able to adaptively choose the steps to be taken, and to detect *bifurcation* values (e.g., points where two solution curves intersect, or points where –for (1)– the curve fails to be parametrizable in α). Predictor-corrector algorithms are well understood and reliable implementations exist; see [9, 15]. In practical terms, the largest cost of prediction-correction methods is given by the need for frequent factorizations of the Jacobian matrices involved.

One aspect which has not been satisfactorily resolved in previous works on predictor-corrector algorithms is that the dynamical point of view, inherent in the continuation context, gets lost at the linear algebra level: the linear algebra is done in a *static* way, namely canned linear algebra software is used to factor the Jacobians. Our approach, which we present in this paper, is to view the Jacobians as smooth functions and to **use** the **underlying smoothness** of the factors in their decompositions to device better continuation strategies; at the same time, we will monitor variation of the factors to determine how to choose the continuation steps. Motivated by their relevance in detecting bifurcation phenomena, and by the provable smoothness in situations of practical interest (see [6] for the C^k case, and [4] for the analytic case), our particular emphasis here is on SVD decompositions. We must point out right away that a smooth SVD will ordinarily require a singular value to cross zero when the Jacobian becomes singular (one obtains a *signed* SVD), and two singular values will exchange ordering when they coalesce (one obtains an *unordered* SVD).

2 Path following methods via SVD

In this work, we focus on computing curves of equilibria of (1): $f(x, \alpha) = 0$. By using arc-length parametrization, the problem is rewritten as

$$f(x(s), \alpha(s)) = 0, \;\; \dot{x}(s)^T \dot{x}(s) + \dot{\alpha}(s)^2 = 1,$$

where $(\dot{x}(s), \dot{\alpha}(s))^T$ is the tangent vector and satisfies

$$f_x \dot{x} + f_\alpha \dot{\alpha} = 0. \tag{2}$$

A point (x, α) on the curve is a regular point if $f_x(x, \alpha)$ is invertible, is a fold (turning) point if $f_x(x, \alpha)$ is singular and $f_\alpha(x, \alpha) \notin \text{range}(f_x(x, \alpha))$, is a branch point if $f_x(x, \alpha)$ is singular and the enlarged matrix $M = \begin{bmatrix} f_x & f_\alpha \\ \dot{x}^T & \dot{\alpha} \end{bmatrix}$ has rank equal to n. At a regular point there is a unique tangent; this also occurs at a fold point with $\dot{\alpha} = 0$; at a branch point instead, there are two tangents and two branches of equilibria crossing at the point. A branch point may occur with $f_\alpha \in \text{range}(f_x)$ and $\text{rank}(f_x) = n - 1$, and also with $\text{rank}(f_x) = n - 2$.

The pseudo arc-length path following approach (see e.g. [10], [13]) can be sketched as follows. Given a point $(x_0, \alpha_0) = (x(s_0), \alpha(s_0))$ on the path (i.e.

$f(x_0,\alpha_0)=0$), a tangent $(\dot{x}_0,\dot{\alpha}_0)$ and $s_1=s_0+h$ for some steplength h, we seek $(x_1,\alpha_1)=(x(s_1),\alpha(s_1))$ by solving

$$F(x,\alpha)\equiv\begin{bmatrix} f(x,\alpha) \\ \dot{x}_0^T(x-x_0)+\dot{\alpha}_0(\alpha-\alpha_0)-h \end{bmatrix}=0. \tag{3}$$

To solve (3), we can use an iterative method starting from the tangent predictor

$$(x_1^{(0)},\alpha_1^{(0)})=(x_0,\alpha_0)+h(\dot{x}_0,\dot{\alpha}_0).$$

Typically, the stationary Newton method is used as corrector: for $k=0,1,\dots$

$$\begin{bmatrix} f_x(x_1^{(0)},\alpha_1^{(0)}) & f_\alpha(x_1^{(0)},\alpha_1^{(0)}) \\ \dot{x}_0^T & \dot{\alpha}_0 \end{bmatrix}\begin{bmatrix} \Delta x \\ \Delta\alpha \end{bmatrix}=-F(x_1^{(k)},\alpha_1^{(k)}) \tag{4}$$

$$(x_1^{(k+1)},\alpha_1^{(k+1)})=(x_1^{(k)}+\Delta x,\alpha_1^{(k)}+\Delta\alpha)\,.$$

The idea to use SVD for computing bifurcations of vector fields appeared in [5] several years ago. Afterwards, as far as we could determine, it has not been considered any more. Why? Surely, one reason is that existing techniques typically work just fine. But, we believe that a more cogent reason is that by using the standard linear algebra SVD, whereby singular values are kept always positive and ordered, it is nearly impossible to locate exactly a fold or a branch point: One would need to step exactly at such points! Simply monitoring small singular values, as done in [5], is at best inefficient and potentially misleading. The answer to this impasse is beautifully provided by smoothness: If the Jacobian f_x becomes singular, a singular value will go through 0, and monitoring the signs of the singular values will suffice. As we will make clear below, our smooth SVD method provides not only a theoretically sound, but also a computationally interesting, way to compute curves of equilibria. As a matter of fact, we know of no other technique which allows **at once** to compute the points on the curve, provide tangents to form the predictor, detects folds and branch points, and also allows easily to switch branches at branch points.

Motivated by the work [7], where we studied several methods to compute smooth curves of SVD, we have written a `Matlab` code implementing a SVD-based path following method for tracking smooth path of equilibria. Our code computes a smooth SVD of f_x at points on the curve and uses this SVD to locate folds and generic branch points (switching branches and continuing new branches). Moreover the code avoids to compute f_x and f_α at the predictor and solves (3) by a Newton-type method, where $f_x(x_0,\alpha_0)$ and $f_\alpha(x_0,\alpha_0)$ are used instead of $f_x(x_1^{(0)},\alpha_1^{(0)})$ and $f_\alpha(x_1^{(0)},\alpha_1^{(0)})$ (cfr. (4)). From now on, the symbols f_x and f_α will be used to denote $f_x(x_0,\alpha_0)$ and $f_\alpha(x_0,\alpha_0)$ respectively.

Notice that if $f_x=U\Sigma V^T$, then the Newton-type system

$$\begin{bmatrix} f_x & f_\alpha \\ \dot{x}_0^T & \dot{\alpha}_0 \end{bmatrix}\begin{bmatrix} \Delta x \\ \Delta\alpha \end{bmatrix}=\begin{bmatrix} c \\ d \end{bmatrix}$$

becomes simply

$$\begin{bmatrix} \Sigma & U^T f_\alpha \\ \dot{x}_0^T V & \dot{\alpha}_0 \end{bmatrix} \begin{bmatrix} V^T \Delta x \\ \Delta\alpha \end{bmatrix} = \begin{bmatrix} U^T c \\ d \end{bmatrix}$$

which has a very simple structure and can be solved with a computational cost of $O(n)$ flops, by ad hoc substitution techniques tailored for the two cases Σ invertible and Σ singular. So, taking into account the matrix-vector products, each iteration costs $O(n^2)$ flops.

Analogously, the SVD of f_x is easily exploited in the computation of the tangents. Indeed, problem (2) becomes

$$\Sigma V^T \dot{x} + U^T f_\alpha \dot{\alpha} = 0.$$

If Σ is invertible, we set $\dot{\alpha} = \pm 1$ (the sign is chosen so that we do not trace back the same piece of the curve), $\dot{x} = -\dot{\alpha} V \Sigma^{-1} U^T f_\alpha$ and then we normalize. If we are at a fold point (Σ singular and $f_\alpha \notin \text{range}(f_x)$), the solution is $(\dot{x}, \dot{\alpha}) = (V e_n, 0)$. Finally, at a branch point ($\text{rank}[f_x \ , \ f_\alpha] = n-1$, but $\text{rank}(M) = n$) we have two tangents in the kernel of $[f_x \ , \ f_\alpha]$, and we proceed as follows: (1) We approximate the tangent, call it u, to the branch that we are following, by normalizing the secant approximation obtained using the last two points on the curve; (2) we seek a point on the other branch by looking for solutions lying on a hyperplane parallel to the tangent line we have, and at a distance h from it (this we can do using a vector v in the null space of $[f_x \ , \ f_\alpha]$ orthogonal to u).

3 Continuation of the SVD

In [7], we proposed several techniques to compute a smooth block-SVD for a smooth matrix-valued function $A(s) : [a, b] \to R^{n\times n}$. In the context of path following for equilibria, we have $A(s) = f_x(x(s), \alpha(s)) \in R^{n\times n}$ and want to smoothly compute its complete signed SVD. The theory developed in [7] guarantees that one of the algorithms there studied, namely Algorithm BSVD, can be successfully used to this scope as long as the singular values remain distinct. We refer to [7] for details. Here we only recall that this method is essentially based on the solution of suitable algebraic Riccati equations at each continuation step. These equations are solved by Newton's method, for which we can construct a second order approximation as initial guess. Convergence is ensured as long as the continuation steplength h is sufficiently small.

If two singular values coalesce within a continuation step, the Newton iteration for the Riccati equations may either converge to a wrong solution or fail to converge. Now, having two singular values coalescing is a non-generic occurrence for one-parameter problems (see [6]), and thus one should not witness it in practice; naturally, even less likely (and thus not very interesting) is the case of three or more singular values coalescing at once, a case which we can thus safely disregard. However, special symmetries in the problem make the occurrence of two singular values coalescing (and crossing) possible, and we want methods robust enough to handle this case. Futhermore, singular values nearing each other can cause numerical difficulties. To overcome this critical situation, in the present paper, we implemented a novel technique, which we call the **accordion** strategy. The idea is to adaptively switch from a complete SVD to a block SVD in order to bypass coalescing (or just close) singular values. In practice, at each step we monitor the singular values to detect if we are nearing a possible crossing. If this is the case, we group into a 2×2 block the two singular values which seem to coalesce and apply Algorithm BSVD to smoothly continue a block-SVD with one block of dimension 2 and the others of dimension 1. This structure is possibly maintained over several continuation steps, until the crossing has been safely passed, and we can split the 2×2 block and proceed with a complete SVD. Of course, as the continuation proceeds, the 2×2 block must be further decomposed to get always a smooth complete SVD. To do this, we solve a simple Procrustes problems (see [12] for analogous refining methods in the context of computing analytic path of SVD). In the context of continuation methods for large bifurcation problems, strategies where one dynamically chooses the block sizes have also been considered in [2, 3].

The *accordion* strategy works very well in practice so long as coupled with a reliable criterion to select the steplength at each continuation step. In the continuation context, the stepsize control is generally convergence-dependent through the number of performed iterations ([8], [9],[10]). In our code, instead, we implemented a technique based on the distance between the predictors and the converged values. This is similar in spirit to the stepsize control in codes for solving initial value problems of differential equations.

Finally, the code allows to locate fold and branch points accurately. Given an interval $[s_0, s_1]$ such that $\sigma_n(s_0) \cdot \sigma_n(s_1) < 0$, we use the secant method to find s^* s.t. $\sigma_n(s^*) = 0$ and compute the point $(x(s^*), \alpha(s^*))$ on the equilibria curve. Obviously, each secant iteration involves a continuation step from s_0 to the current iterate.

4 Algorithmic details

We now discuss in more details the three key algorithmic choices we have adopted in our code.

1. *When and how to group/split singular values.* Given s_0, assume we have the decomposition $A(s_0) = U(s_0)\Sigma(s_0)V(s_0)^T$ with $\sigma_1(s_0) > \sigma_2(s_0) > \ldots > \sigma_n(s_0) \geq 0$. We predict the singular values at $s_1 = s_0 + h$ by [6]

$$\sigma_i^{(\mathrm{pred})} = \sigma_i(s_0) + (U^T(s_0)(A(s_1) - A(s_0))V(s_0))_{ii} = \sigma_i(s_0) + h\dot{\sigma}_i(s_0) + O(h^2)$$

and declare a possible crossing if

$$\text{either} \quad \frac{\sigma_{i+1}^{(\mathrm{pred})} - \sigma_i^{(\mathrm{pred})}}{\sigma_{i+1}(s_0) - \sigma_i(s_0)} < 0 \quad \text{or} \quad \frac{\sigma_{i+1}^{(\mathrm{pred})} - \sigma_i^{(\mathrm{pred})}}{\sigma_i^{(\mathrm{pred})} + 1} \leq \eta$$

for some small $\eta > 0$ ($\eta = 10^{-4}$ is the default value). In this case, we group σ_i, σ_{i+1} into a 2×2 block.

Once the singular values $\sigma_{i+1}(s_1)$ and $\sigma_i(s_1)$ have been computed, the 2×2 block is unrolled if

$$\frac{\sigma_{i+1}(s_1) - \sigma_i(s_1)}{\sigma_i(s_1) + 1} > \eta.$$

2. *Refinement of a* 2×2 *block to get a complete SVD.* Given a block C, we get a standard decomposition $U_C^T C V_C = \text{diag}(\sigma_1, \sigma_2)$ with $\text{sign}(\sigma_i) = \text{sign}(\sigma_i^{(\text{pred})})$ and then possibly modify it to recover smoothness in the complete SVD. To do this, we solve a Procrustes problem to minimize the distance (in Frobenius norm) between U_C, V_C and suitable reference orthogonal factors. If $\sigma_1 \neq \sigma_2$, U_C and V_C are correct up to permutation and/or changes of signs of their columns. If $\sigma_1 = \sigma_2$, then U_C and V_C may have to be changed into $U_C Q$ and $V_C Q$ for a suitable orthogonal matrix Q.
3. *Stepsize control.* Denote by $x^{(\text{pred})}$, $\alpha^{(\text{pred})}$, $\Sigma^{(\text{pred})}$ the predictors used for $x(s_1)$, $\alpha(s_1)$ and $\Sigma(s_1)$, respectively. Our steplength choice tries to perform a mixed absolute/relative error control, monitoring the errors $\|x^{(\text{pred})} - x(s_1)\|$, $\|\alpha^{(\text{pred})} - \alpha(s_1)\|$, $\|\Sigma^{(\text{pred})} - \Sigma(s_1)\|$, in such a way that the iterative procedure involved in a continuation step will converge in a few iterations. Moreover, starting from the second continuation step, we try to control also $\|U^{(\text{pred})} - U(s_1)\|$ and $\|V^{(\text{pred})} - V(s_1)\|$, where $U^{(\text{pred})} = U(s_0) + h\dot{U}(s_0)$, $V^{(\text{pred})} = V(s_0) + h\dot{V}(s_0)$, and the derivatives are approximated by differences using the orthogonal factors obtained at the last two steps. At the end of a continuation step of size h, we compute the weighted norm $\rho_\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{\sigma_i^{(\text{pred})} - \sigma_i}{\epsilon_r|\sigma_i| + \epsilon_a})^2}$, where ϵ_r and ϵ_a are relative and absolute error tolerances, and analogously we compute ρ_x, ρ_α, ρ_U, ρ_V. Then, we set $\rho = \max(\rho_\sigma, \rho_x, \rho_\alpha, \rho_U, \rho_V)$ and $h_{\text{new}} = \frac{h}{\sqrt{\rho}}$. If $\rho \leq 1.5$, h_{new} is used to proceed the continuation; otherwise the step just completed is rejected and retried with the new steplength.

A continuation step may also fail because of lack of convergence either when solving a Riccati equation or when computing a new point on the curve. In both cases, the steplength is halved and the step retried.

5 Some numerical results

We refer on some experiments on two standard test problems. The results have been obtained with the following data:
- Initial steplength: 10^{-3}; minimum steplength: 10^{-8};
- Tolerances for solving algebraic Riccati equations: 10^{-8};
- Tolerances for computing fold and branch points: 10^{-12};
- Tolerances for computing points on the curve: 10^{-14}.

In the tables we give:

`Nsteps`: required number of continuation steps to complete the path;
`hmax`: maximum used steplength;
$\texttt{Nits}_1$: average number of iterations required to solve a Riccati equation (this is needed to update the SVDs);

Nits$_2$: average number of iterations per continuation step required to compute the point on the curve. The numbers Nits$_1$ and Nits$_2$ take into account all performed iterations, on both successful and unsuccessful steps.

Example 1. This is known as the aircraft problem [14]. The dimension is $n = 5$. The curve has to be followed from $(x_0, \alpha_0) = (0, 0)$ until α is out of $[-1, 1]$. Starting with $\dot{\alpha}_0 = 1$, two folds are found at $\alpha = 0.18608332702306$ and $\alpha = -0.50703056119994$. The figure on the right shows the first component of the solution vs. α. Table 1 shows the results obtained with several choices of the parameters ϵ_r and ϵ_a for the steplength control. Starting with $\dot{\alpha}_0 = -1$, the code detects two folds at $\alpha = -0.18690833269959$ and $\alpha = 0.51015853464868$; the performance is identical to that in Table 1.

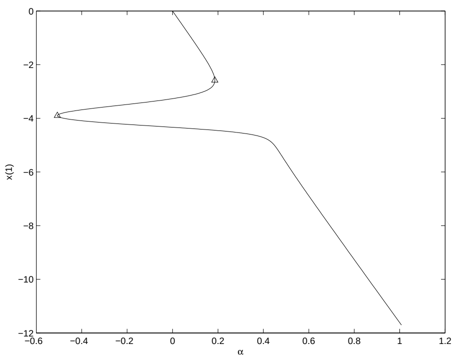

Table 1. Example 1.				
$\epsilon_a (= \epsilon_r)$	Nsteps	Nits$_1$	Nits$_2$	hmax
10^{-4}	694	2	4	0.24
10^{-3}	226	3	6	0.70
10^{-2}	82	4	11	1.72

Example 2. This is a test problem from the AUTO manual [9]:

$$f(x, \alpha) = \begin{bmatrix} x_1(1 - x_1) - 3x_1x_2 \\ -\frac{1}{4}x_2 + 3x_1x_2 - 3x_2x_3 - \alpha(1 - \mathrm{e}^{-5x_2}) \\ -\frac{1}{2}x_3 + 3x_2x_3 \end{bmatrix}.$$

The continuation process starts from $(x_0, \alpha_0) = (1, 0, 0, 0)$ and stops when α is out of $[0, 0.6]$. The code detects two branch points, at $\alpha = 0.55$ and $\alpha = 0.36846953170021$, and a fold at $\alpha = 0.56459590997250$. In the figure on the right, we show the first solution component vs. α. At each branch point we started new runs to follow the branches. In Table 2 we give a summary of performance of the code relatively to completion of the three paths in the figure.

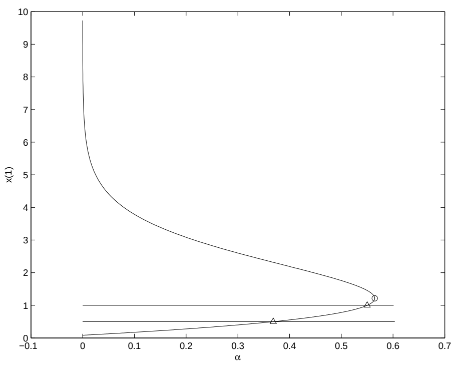

Table 2. Example 2.

$\epsilon_a (= \epsilon_r)$	`Nsteps`	$\texttt{Nits}_1$	$\texttt{Nits}_2$	`hmax`
10^{-4}	792	2	2	0.76
10^{-3}	281	2	3	2.25
10^{-2}	121	3	3	5.27

In both examples, $3 \div 5$ secant iterations were needed to locate either a fold or a branch point.

References

1. Allgower, E., Georg, K.: Numerical Continuation Methods, Springer-Verlag, New York (1990)
2. Beyn, W.J., Kleß, W., Thümmler, V.: Continuation of low-dimensional invariant subspaces in dynamical systems of large dimension. In: Fiedler, B. (ed.): Ergodic Theory, Analysis and Efficient Simulation of Dynamical Systems. Springer (2001) 47–72
3. Bindel, D., Demmel, J., Friedman, M.: Continuation of Invariant Subspaces for Large Bifurcation Problems. In: Proceedings of the SIAM Conference on Applied Linear Algebra. The College of William and Mary, Williamsburg, VA (2003) http://www.siam.org/meetings/la03/proceedings.
4. Bunse-Gerstner, A., Byers, R., Mehrmann, V., Nichols, N. K.: Numerical Computation of an Analytic Singular Value Decomposition by a Matrix Valued Function, Numer. Math. **60** (1991) 1–40
5. Chow, S., Shen, Y.: Bifurcations via Singular Value Decompositions. Appl. Math. Comput. **28** (1988) 231–245
6. Dieci, L., Eirola, T.,: On Smooth Decomposition of Matrices. SIAM J. Matrix. Anal. Appl. **20** (1999) 800–819
7. Dieci, L., Gasparo, M.G., Papini, A.: Continuation of Singular Value Decompositions. Mediterr. j. math. **2** (2005) 179–203
8. Dhooge, A., Govaerts, W., Kuznetsov, Y.A.: A MATLAB Package for Numerical Bifurcation Analysis of ODE. ACM Trans. on Math. Software **29** (2003) 141–164
9. Doedel, E.J., et al.: AUTO97: Continuation and Bifurcation Software for Ordinary Differential Equations (with HomCont), User's Guide. Concordia University, Montreal, P.Q, Canada (1997) http://indy.cs.concordia.ca
10. Keller, H.B.: Numerical Methods in Bifurcation Problems. Springer Verlag, Tata Institute of Fundamental Research, Bombay (1987)
11. Kuznetsov, Y.A.: Elements of Applied Bifurcation Theory. Springer-Verlag, New York (1995)
12. Mehrmann, V., Rath., W.: Numerical Methods for the Computation of Analytic Singular Value Decompositions. Electron. Trans. Numer. Anal. **1** (1993) 72–88
13. Rheinboldt, W.C.: Numerical Analysis of Parametrized Nonlinear Equations. Wiley, New York (1986)
14. Rheinboldt, W.C., Burkardt, J.V.: A locally parametrizied continuation process. ACM Trans. on Math. Software **9** (1983) 215–235
15. Watson, L.T., Sosonkina, M., Melville, R.C., Morgan, A.P., Walker, H.F.: Algorithm 777: HOMPACK 90: a suite of Fortran 90 codes for globally convergent homotopy algorithms. ACM Trans. Math. Software **23** (1997) 514–549