Newton-like Iteration for Determinantal Systems and Structured Low Rank Approximation

Éric Schost and Pierre-Jean Spaenlehauer
Western University, Department of Computer Science
London, Ontario, Canada
eschost@uwo.ca, pierre-jean.spaenlehauer@m4x.org

Problem statement. Let $\mathcal{M}_{p,q}(\mathbb{R})$ be the space of $p \times q$ matrices with real entries, $r \in \mathbb{N}$ be an integer, $V_r \subset \mathcal{M}_{p,q}(\mathbb{R})$ be the determinantal variety of matrices of rank at most r and E be a linear (or affine) subspace of $\mathcal{M}_{p,q}(\mathbb{R})$ (*e.g.* Toeplitz, Hankel, Sylvester matrices). Given a matrix $M \in E$, the goal is to compute a close matrix in $E \cap V_r$. More precisely, we want a numerical algorithm computing a function $\varphi : E \longrightarrow E$ such that, if M is close enough to $E \cap V_r$, then the sequence defined by $M_0 = M$, $M_{i+1} = \varphi(M_i)$ converges quadratically towards a matrix $M_\infty \in E \cap V_r$. As shown in [5], this problem which is also known as *Structured Low-Rank Approximation* (SLRA) is central in data fitting or in numerical analysis. It is also underlying classical symbolic-numeric problems.

Main results. We propose a Newton-like algorithm (`NewtonSLRA`) which answers the above specification and appears to converge quadratically. The main principle of this algorithm is close to Cadzow's algorithm [1] which proceeds by a sequence of Singular Value Decompositions (SVD) and orthogonal projections on E. However, we choose a direction of projection which is tangent to the determinantal variety in order to ensure quadratic convergence. Each iteration of the algorithm `NewtonSLRA` computes a function $\varphi(M)$ in three main steps: (1) compute a rank r approximation $\widetilde{M}$ of M; (2) from the left and right kernels of $\widetilde{M}$, compute a set of generators of the tangent space $T_{\widetilde{M}}V_r$; (3) compute the point in $E \cap T_{\widetilde{M}}V_r$ which minimizes the distance to $\widetilde{M}$ (this is achieved by solving a linear least squares problem). Computing the best rank r approximation with respect to the Frobenius norm is achieved by the SVD. It also provides an orthonormal basis (for the scalar product $\langle M_1, M_2 \rangle = \mathtt{tr}(M_1 \cdot {}^T M_2)$) of the normal space $N_{\widetilde{M}}V_r = \mathrm{Ker}_L(\widetilde{M}) \otimes \mathrm{Ker}_R(\widetilde{M})$ which is used for computing the projection on E. The most expensive step is the SVD which is achieved in $O(pq \min(p,q))$ operations in fixed precision. The main theoretical result lies in the following theorem which ensures the local quadratic convergence towards a matrix $M_\infty \in V_r \cap E$ near the optimal solution, under conditions on the dimensions of $\dim(E)$ and $\dim(V_r)$. To the best of our knowledge, this is the first proof of quadratic convergence of an iterative method for the SLRA problem:

Theorem 1. *If* $\dim(E) = \dim(V_r)$ *and* $\dim(E) + \dim(V_r) > pq$*, then the algorithm* `NewtonSLRA` *computes a function* $\varphi : E \to E$ *verifying the following property: for all* $\mu > 1$ *and for all* $\hat{M} \in V_r \cap E$ *such that* V_r *and* E *verify mild transversality conditions at* $\hat{M}$*, there exists* $\epsilon > 0$ *such that for all* M_0 *with* $\|M_0 - \hat{M}\| < \epsilon$*, the sequence* $M_{i+1} = \varphi(M_i)$ *converges towards a matrix* $M_\infty \in V_r \cap E$ *and*

$$\|M_i - M_\infty\| \leq \left(\frac{1}{2}\right)^{2^i - 1} \|M_0 - M_\infty\| \quad \textit{and} \quad \|M_0 - M_\infty\| \leq \mu \|M_0 - \hat{M}\|.$$

The proof relies on tools from Smale's α-theory, slightly modified to take into account the properties of this Newton-like iteration.

Application to univariate approximate GCD. Approximate GCD computation is a symbolic-numeric example of SLRA problem: a degree condition on the GCD of univariate polynomials amounts to

a rank condition on their Sylvester matrix. In this setting, the algorithm takes as input two floating-point polynomials f, g of degrees m and n, and an integer $d \in \mathbb{N}$; it outputs three floating-point polynomials a, b, h of respective degrees $m-d$, $n-d$, d such that $\|f-ah\|^2 + \|g-bh\|^2$ is small. Here, E is the linear space of truncated Sylvester matrices (see *e.g.* [6]) and V_r is the variety of rank deficient matrices of sizes $(m+n-d+1) \times (m+n-2d+2)$. We compare in Table 1 our `Maple` implementation of `NewtonSLRA` with the `Maple` implementation of `GPGCD` [6], which is a state-of-the art algorithm dedicated to the computation of approximate GCDs. Instances are constructed by generating two random polynomials $\widetilde{f}$, $\widetilde{g}$ such that $\deg(\mathrm{GCD}(\widetilde{f},\widetilde{g})) = d$ and by adding a random error polynomial f_ϵ, g_ϵ such that the relative noise $\sqrt{\|f_\epsilon\|^2 + \|g_\epsilon\|^2}/\sqrt{\|\widetilde{f}\|^2 + \|\widetilde{g}\|^2}$ is equal to a fixed parameter ϵ. The column "perturbation" gives the relative distance between the output and the input of the algorithms. Notice that `NewtonSLRA` performs almost as well as `GPGCD`, which relies on optimization techniques to minimize the function $\|f-ah\|^2 + \|g-bh\|^2$. In comparison, `NewtonSLRA` does not converge to the minimum of this function, but we see in Table 1 that the distance to the optimum is small. Also, experimental results indicate that `NewtonSLRA` converges quadratically (although $\dim(E)$ and $\dim(V_r)$ do not verify the assumptions of theorem 1), whereas `GPGCD` converges linearly (see the right part of table 1 for an example). We also tried to use directly the `QRGCD` routine from the package `SNAP` in `Maple` [3] but it failed to find an approximate GCD in our examples because of the high level of noise in the coefficients of the input polynomials.

	`NewtonSLRA`		`GPGCD`	
(m, n, d, ϵ)	time	perturbation	time	perturbation
(100, 100, 50, 0.001)	0.803s	4.838e-4	0.806s	4.742e-4
(500, 500, 250, 0.001)	37.5s	5.127e-4	45.4s	4.923e-4
(1000, 1000, 500, 0.001)	282s	5.781e-4	317s	5.155e-4
(2000, 2000, 1000, 0.0001)	1567s	5.104e-5	1161s	5.088e-5

	sizes of iteration steps	
iteration	`NewtonSLRA`	`GPGCD`
1	0.9e-1	0.9e-1
2	0.5e-3	0.5e-3
3	0.6e-8	0.2e-5
4	0.1e-17	0.8e-8
5	0.1e-36	0.4e-10

Table 1: Comparison between `GPGCD` [6] and `NewtonSLRA` for computing approximate GCDs

Other applications of SLRA in symbolic-numeric computations and future work. Several other algebraic problems are characterized by rank conditions on structured matrices, for which these techniques could lead to symbolic-numeric algorithms, *e.g.* solving bilinear systems, computing the minimal polynomial of algebraic power series or computing low degree Pade approximants. Moreover, there is still room for improvement: the most computationally-intensive step of this algorithm is the computation of the SVD, but the algorithm converges quadratically even when less precise rank-approximation techniques are used. Also, we plan to compare our method and implementation with other algorithms for SLRA (see *e.g.* [2] and references therein) and for computing approximate GCDs (see *e.g* [4], which relies on the *Structured Total Least Norm* approach). The main challenge is to extend theorem 1 by relaxing the restrictions on $\dim(E)$ and $\dim(V_r)$.

References

[1] J. A. Cadzow. Signal enhancement-a composite property mapping algorithm. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(1):49–62, 1988.

[2] M. T. Chu, R. E. Funderlic, and R. J. Plemmons. Structured Low Rank Approximation. *Linear algebra and its applications*, 366:157–172, 2003.

[3] R. M. Corless, S. M. Watt, and L. Zhi. QR factoring to compute the gcd of univariate approximate polynomials. *Signal Processing, IEEE Transactions on*, 52(12):3394–3402, 2004.

[4] E. Kaltofen, Z. Yang, and L. Zhi. Structured low rank approximation of a Sylvester matrix. In *Symbolic-numeric computation*, pages 69–83. Springer, 2007.

[5] I. Markovsky. Structured low-rank approximation and its applications. *Automatica*, 44(4):891–909, 2008.

[6] A. Terui. An iterative method for calculating approximate gcd of univariate polynomials. In *ISSAC 2009*, pp. 351–358.