Note on Trace Maximization Correction to the Multi-precision Polar Decomposition

Thomas Seleiro[*]

11 January 2021

1 Polar Decomposition

For $A \in \mathbb{C}^{m \times n}$ with $m \geq n$, we can find a polar decomposition $A = UH$, where $U \in \mathbb{C}^{m \times n}$ has orthonormal columns and $H \in \mathbb{C}^{n \times n}$ is Hermitian positive semi-definite. The unitary polar factor U for a non-singular $n \times n$ matrix A can be computed via the scaled Newton iteration defined by the recursive step

$$X_{k+1} = \frac{1}{2}(\mu_k X_k + \mu_k^{-1} X_k^{-*}), \qquad X_0 = A.$$

Throughout the experiments performed, we used the $1, \infty$-norm scaling factor

$$\mu_k = \left(\frac{\|X_k^{-1}\|_1 \|X_k^{-1}\|_\infty}{\|X_k\|_1 \|X_k\|_\infty} \right)^{1/4},$$

and we used a mixture of the stopping conditions $\|X_k - X_{k-1}\|_\infty / \|X_k\|_\infty \leq nu$ and $\|I - X_k^* X_k\|_\infty \leq nu$ suggested in [1, §8.4].

We try to evaluate the effectiveness of using this method for computing the polar decomposition of a matrix in multiple precision. The iterates converge quadratically to the unitary polar factor. Therefore once the iteration has converged to a lower precision, only one further iteration in the desired higher precision would be needed for convergence.

The computed matrix U_1 will be unitary to the desired precision, but the corresponding Hermitian factor $H_1 = U_1^* A$ need not be Hermitian positive semi-definite. Table 1. shows that in multi-precision H_1 is only Hermitian to single precision and the calculated matrices are inaccurate. We try to compensate for this inaccuracy by calculating the polar decomposition $H_1 = WH$. We then have $A = U_1 H_1 = (U_1 W)H =: UH$, where U is unitary to double precision and H is Hermitian positive semi-definite.

[*] Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK (thomas.seleiro@postgrad.manchester.ac.uk)

A	$\|H_{mp} - H_{mp}^*\|_\infty/2$	$\|H - H^*\|_\infty/2$	$\|U - U_{mp}\|_\infty$
`rand(20)`	2.52e-06	4.72e-15	1.14e-05
`rand(40)`	1.09e-05	2.35e-14	3.35e-05
`rand(60)`	2.84e-05	4.92e-14	6.56e-05
`rand(80)`	4.07e-05	8.39e-14	9.33e-05
`rand(100)`	6.84e-05	1.42e-13	1.38e-04

Table 1: Table comparing results between the multi-precision polar decomposition Newton iteration (without correcting the resulting Hermitian polar factor), and the corresponding iteration in double precision. Note "mp" refers to multi-precision.

In general, H_1 is not unitary ($\|A\|_2 = \|U_1 H_1\|_2 = \|H_1\|_2$). Therefore we try to avoid using the Newton method to compute this polar decomposition, since the iterates converge to a unitary matrix.

We instead consider the property that for all unitary $W \in \mathbb{C}^{n\times n}$, trace($W^*A$) is maximised if and only if W is a unitary polar factor of A (see [1, Prob. 8.13]).

An algorithm for computing the polar decomposition using the trace maximisation property is proposed in [2, p.84]. We repeatedly loop through every 2×2 principal submatrix $A_{ij} = A([i,j],[i,j])$ and apply Givens transformations that make A_{ij} symmetric and maximise its trace. We do so until the matrix is symmetric. If the resulting matrix $\tilde{A}$ is indefinite, it has a smallest negative eigenvalue $\lambda_{min}(A)$ and associated eigenvector x. Applying a Householder transformation W^* where $W = I - 2xx^*$ makes the resulting matrix Hermitian positive semi-definite and increases the trace of $\tilde{A}$.

We directly implemented this algorithm in the function `maxtracePoldec`. Our implementation differs from [2] by adding a relaxation term to the symmetric condition.

```
symmDist = norm(A - A', inf) / norm(A, inf);
while(symmDist > u*n)
```

We added this term since for random dense matrices of moderate size, the routine remains stuck in the while loop.

As a method for computing a general polar decomposition, Table 2. shows that the Newton method is more efficient and accurate than the trace maximization algorithm. Looking at the output of `maxtracePoldec` in Fig. 1, we see that the convergence of the trace maximization algorithm is linear and thus unusable to efficiently correct the multi-precision result. One observation that can be made when looking at the output is that the trace varies very little after relatively few sweeps of the algorithm. This could leave space for using a two step method, one which rapidly maximises the trace, and another which focusses on making the matrix symmetric rapidly. Such a method could allow for the partial use of this algorithm, over a restricted number of sweeps to maximise the trace of H_1.

A	t_G	t_N	s_G	$\|H_G - H_G^*\|/2$	$\|H_N - H_N^*\|/2$
`rand(25)`	0.27	1.67e-3	245	1.91e-14	8.12e-15
`rand(50)`	4.88	1.83e-3	713	7.88e-14	3.90e-14
`rand(75)`	35.96	3.11e-3	1364	1.64e-13	9.17e-14
`rand(100)`	140.53	4.49e-3	2110	3.14e-13	1.46e-13

Table 2: Table showing t_G and t_N, the calculation times using `maxtracePoldec` and a double precision Newton iteration; s_G the number of sweeps of `maxtracePoldec`, and the norm skew-Hermitian parts of the computed Hermitian polar factors.

```
Sweep     |A-A'|/|A|      trace_Diff
====================================
1         1.9541e-07      6.2148e-14
2         1.3377e-07      1.0236e-14
3         8.0454e-08      2.4372e-15
4         5.7389e-08      1.5842e-15
5         3.9764e-08      9.7488e-16
6         3.1497e-08      8.5302e-16
7         2.5357e-08      4.8744e-16
8         2.2810e-08      1.2186e-16
...
453       3.8064e-15      0.0000e+00
454       3.6931e-15      0.0000e+00
455       3.5531e-15      1.2186e-16
456       3.4020e-15      1.2186e-16
```

Figure 1: Output of `maxtracePoldec` when used to correct the multi-precision polar decomposition of `A = rand(32)`

	Runtime			Sweeps		$\|H - H^*\|_\infty/2$		
A	t_G	t_P	t_N	s_G	s_P	H_G	H_P	H_N
`rand(20)`	0.12	0.12	9.85e-04	176	176	1.25e-14	1.27e-14	1.70e-15
`rand(40)`	0.66	0.64	1.32e-03	182	183	4.92e-14	4.91e-14	5.06e-15
`rand(60)`	6.41	6.69	2.07e-03	420	421	1.16e-13	1.16e-13	8.80e-15
`rand(80)`	16.68	16.58	3.56e-03	446	447	1.97e-13	1.94e-13	1.20e-14
`rand(100)`	41.50	40.30	4.65e-03	552	552	3.13e-13	3.16e-13	1.63e-14

Table 3: Table showing the runtime, number of sweeps, and norm of the skew-Hermitian component of the computed correction to the multi-precision Newton method of A. The subscripts G, P and N correspond respectively to corrections using `maxtracePoldec`, `twobytwoPoldec` and a double precision Newton method on H_1

A formula for the polar decomposition of a matrix $B \in \mathbb{R}^{2\times 2}$ is known [1, Prob. 8.2]. A variant of the algorithm considered involves directly computing the polar decomposition of the 2×2 submatrices A_{ij}. The polar decomposition maximises the trace (over all unitary matrices), thus we could not find any better unitary matrices to use in the algorithm. Note that such a method still requires a final check for positive semi-definiteness and potentially a Householder correction like in `maxtracePoldec`.

We implemented this variant of the algorithm in the function `twobytwoPoldec`. We then compared its performance computing the multi-precision correction against that of the Givens rotation based algorithm, and a simple Newton iteration on H_1. As Table 3. shows, there is no significant difference in runtime or accuracy of the computed correction with either trace maximisation method. More importantly, these methods fail in both these metrics when compared to calculating the correction with a Newton method. We also note that `twobytwoPoldec` exhibits similar behaviour to `maxtracePoldec` in Fig 1. However, the trace difference doesn't drop as low as `maxtracePoldec` and stays around 10^{-15}.

2 Implementation of the methods

All the MATLAB code used can be found in the `code` folder of the project.

The implementation of a flexible single-double precision Newton Iteration for the polar decomposition is contained in `multiPoldec.m`.

The implementation of the trace maximisation algorithm using Givens matrices in [2] is contained in `maxtracePoldec.m`.

The implementation of the trace maximisation algorithm using the 2×2 polar decomposition is contained in `twobytwoPoldec.m`.

References

[1] N. J. Higham, *Functions of Matrices : Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, 2008.

[2] M. I. Smith, *Numerical Computation of Matrix Function*, PhD thesis, University of Manchester, Manchester, England, 2002.