Initial Note on Multiprecision Algorithms

Thomas Seleiro*

January 23, 2021

1 Sign Function

I have implemented the base version of the scaled sign function Newton iteration. This only uses norm scaling with the ∞-norm for ease of computation (ie $\mu_k = \sqrt{\|X_k^{-1}\|_\infty / \|X_k\|_\infty}$).

We measure the accuracy of a multi-precision iteration from single to double without corrections. To do so we use the iteration in double precision only, as a best measure of the sign function. The results of these comparisons are presented in Table 1.

Some notable results include that the non-corrected multi-precision iteration is only accurate up to single precision as expected. Note we can ignore the cases of `eye(8)` and `hilb(6)` since $\text{sign}(I) = \text{sign}(\texttt{hilb(6)}) = I$. Therefore a correction is needed for accuracy.

We also observe that the number of iterations in multiple precision is higher than that of the double precision algorithm. Looking at the extreme case for `hadamard(8)`,

```
k    |X_k-X_{k-1}|/|X_k|          |I - X_k^2|
--- --------------------   --------------------
1     1.82842743397e+00       8.08498299421e-08
2     2.00197519007e-07       7.09627158813e-07
3     2.21270937573e-07       4.36684103988e-07
...
19    9.48304048620e-08       3.62178212754e-07
20    7.37569791909e-08       2.96339692341e-07
----------CONVERTING TO DOUBLE PRECISION---------
21    3.52261672986e-08       5.23955030229e-15
22    1.19719549975e-15       2.10231470958e-15
23    3.33644647470e-16       6.19480657439e-16
```

we see that the convergence criterion is not weak enough to catch instances when the iteration has indeed converged, and in some cases due to rounding errors will

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

		Iterations		Comp. time (in ms)	
A	$\|S_{mp} - S\|_\infty$	mp	double	mp	double
`rand(8)`	3.0536e-06	15	9	0.5	0.3
`rand(16)`	8.2193e-06	20	10	0.9	0.4
`eye(8)`	0	0	0	1.6	1.1
`hilb(6)`	8.7086e-19	7	7	2.7	1.5
`magic(6)`	4.1246e-07	10	7	1.6	0.8
`hadamard(8)`	5.9486e-07	23	1	1.0	0.2

Table 1: Table comparing results between the non-corrected multi-precision sign function Newton iteration, and the same iteration in double precision. Note "mp" refers to multi-precision.

take a long time to do so. Note that this likely explains why the multi-precision iteration takes longer to compute than the higher precision iteration.

I am still using the convergence criterion from my `poldec` function:

```
iterDist >= n*roundoff(type) && involDist >= n*roundoff(type)
```

where `iterDist` stores $\|X_k - X_{k-1}\|_\infty / \|X_k\|_\infty$, `involDist` stores $\|I - X_k^2\|_\infty$ and `roundoff(type)` returns the unit roundoff u for the precision currently in use. I will try to fix this by testing the criterion

$$\frac{\|X_k - X_{k-1}\|_\infty}{\|X_k\|_\infty} \leq \|X_k\|_\infty^p \eta, \qquad p = 0, 1, 2$$

proposed in [1, p.123]. Note this might also prevent the need for more than one iteration after convergence in lower precision, which is what we would normally expect since the method is quadratically convergent.

We also computed the least-squares solution to the problem

$$\min \{\|E\| : A(X + E) = (X + E)A\}$$

after lower precision convergence. We use the SVD of $C = I \otimes A - A^T \otimes I$ to solve the least-squares problem $C \operatorname{vec}(E) = -C \operatorname{vec}(X_k)$ where X_k is the converged lower precision iterate. The results using the same matrices as for the case with no correction are presented in Table 2.

We first note that the correction seems to have given a final iterate that is accurate in double precision.

We also observe similar numbers of iterations between both multi-precision methods. Seeing as the first iterations in lower precision are the same for both methods, this again relates to the convergence condition in low and high precision, which need to be loosened to avoid excess iterations.

Most importantly, we witness a significant increase in the computation time, due to the time it takes to compute the correction. Indeed, since comutation of the SVD of C is $O((n^2)^3) = O(n^6)$, the correction ends up dominating the

		Iterations		Comp. time (in ms)	
A	$\|S_{corrected} - S\|_\infty$	mp	double	mp	double
`rand(8)`	4.7992e-15	15	9	1.1	0.3
`rand(16)`	1.3514e-14	20	10	13.1	0.5
`eye(8)`	0	0	0	2.1	0.3
`hilb(6)`	2.0624e-19	7	7	2.0	0.4
`magic(6)`	1.3194e-15	11	7	1.3	1.6
`hadamard(6)`	3.0725e-16	22	1	2.6	0.3

Table 2: Table comparing results between the multi-precision sign fucntion Newton iteration (with a correction found by using the SVD), and the same iteraiton in double precision. Note "mp" refers to multi-precision.

overall computation with larger matrices. This is best illustrated here for the matrix `rand(16)`, where computing the correction increases the computation time tenfold (compared to the multi-precision iterations without correction). For any dense matrix, [2] exploits the structure of C and uses triangular matrices to bring the complexity of the computation down to $O(n^4)$. I will try to see if the structure of the sign function of a matrix could allow for quicker computation of this procedure.

2 Polar Decomposition

I have also implemented a basic multi-precision scaled Newton method to calculate the polar decomposition of a matrix. Note that it does not yet contain a correction to the Hermitian polar factor. The algorithm uses the $1, \infty$-norm scaling, ie

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

Some main observations of the behaviour of the multi-precision algorithm are presented in Table 3.

We first note that the polar decomposition implementation suffers from the same convergence issues as the sign function implementation discussed previously. Therefore, the efficiency that should be gained by lower precision iterations is not apparent in current observations.

The multi-precision iteration does indeed succeed in producing a unitary factor U_{mp} to double precision. Thus when forming the computed unitary polar factor `H = U'*A`, the matrices form an accurate factorization of A. However, Table 3. shows that the computed unitary polar factor U_{mp} is only close to the actual unitray polar factor U of A to single precision. And similarly, we see that H_{mp} is only Hermitian to single precision, and not in double precision as desired.

A	$\|(H - H^*)/2\|_\infty$	$\|(H_{mp} - H_{mp}^*)/2\|_\infty$	$\|U - U_{mp}\|_\infty$
`rand(8)`	8.51e-16	4.98e-07	4.82e-07
`rand(16)`	3.27e-15	1.96e-06	1.20e-06
`eye(8)`	0	0	0
`hilb(6)`	0	0	1.05e-18
`magic(6)`	1.60e-14	6.04e-06	1.74e-07
`hadamard(8)`	0	3.14e-06	9.91e-07

Table 3: Table comparing reulsts 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.

Therefore, in cases where a polar decomposion in high-precision is required, we indeed need to provide a correction to the computed matrices.

References

[1] N. J. Higham, *Functions of matrices : theory and computation*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, 2008.

[2] H. Zha and Z. Zhang, *Computing the optimal commuting matrix pairs*, BIT Numerical Mathematics, 37 (1997), pp. 202, 220.