Review on the SISC manuscript M129636 "Mixed-Precision Analysis of Householder QR Algorithms" by L. Yang, A. Fox, and G. Sanders

February 2, 2020

The Householder QR (HQR) factorization is an important algorithm notably appreciated for its numerical stability. This article presents its rounding error analysis in a mixed-precision framework, assuming that the inner products arising in the algorithm are accumulated in a higher precision and then rounded to low precision. Mixed-precision computations are an important and timely topic, and this article's contribution is clearly relevant to SISC. However, I feel that the article could be significantly improved in several key aspects as explained below. Notably, the conclusions of the analysis should be clarified, more thoroughly discussed and interpreted, and supported by additional experiments. I think these changes would better highlight the main contributions of this article, which to my understanding are the following two: (1) proving that mixed-precision HQR produces an error bound with a weaker dependence on m; (2) comparing HQR with TSQR and identifying a key difference between the uniform and mixed precision settings.

Choice of mixed-precision assumptions The article uses the following mixed-precision framework. For an inner product $s = x^T y$ (with $x, y \in \mathbb{R}^n$), multiplications are performed in precision u_p, additions in precision u_s, and x, y, and s are stored in precision u_w. The model can be fine per se, but I think the following key points should be clarified:

- Relation with the GPU tensor cores units: the authors mention these units several times (e.g., P8L232), but I do not think their framework is directly applicable to them. GPU tensor cores perform the operation

$$D \leftarrow C + AB, \qquad A, B, C, D \in \mathbb{R}^{4\times 4} \tag{1}$$

  using full precision for the multiplications and fp32 precision for the additions. A and B are stored in fp16 but C and D may be stored either in fp16 or fp32. The

key difference with the authors' model lies with the dimension of the matrices. Tensor cores can only be used for matrix–matrix products $S = XY^T$, with X and Y with at least 4 *rows*. Inner products therefore cannot use GPU tensor cores. The authors analyze the BLAS-2 HQR, which is based on inner products and so their analysis does not apply to tensor cores. Considering instead the BLAS-3 HQR would allow their use, but that would be a very different analysis leading to different error bounds.

- The assumptions on the storage precision types of x, y and s should be better justified. In particular, why is it assumed that the result s is stored in the same precision as x and y, and why should this precision be the low one? Consider GPU tensor cores: while A and B must be stored in fp16 precision, C and D need not be. This point is crucial because of the dimensions of the matrices: if X and Y have more than 4 *columns*, computing $S = XY^T$ with tensor cores requires several calls to the kernel (1). Then, storing C and D in fp16 produces a rounding to fp16 every 4 additions. This produces a much larger bound than if C and D are kept in fp32, as observed in [1]. Then, going back to the authors' model: I think it is fine to assume that the result of inner products $s = x^T y$ is stored in low precision, but only when s is the final result of interest and is not reused by some other computations. For example, if an inner product is broken down in several smaller inner products of constant size, these smaller inner products should *not* use the same framework! This point needs clarification: in particular, is it reasonable to systematically round to fp16 every inner product in the HQR factorization? What would change in the bound if the results were kept in fp32 and only rounded, say, at the end of the factorization?

- The distinction between Lemma 2.4 and Corollary 2.5 is confusing. The authors state Corollary 2.5 has "additional constraints: the vectors are being stored in a lower precision than the precision types being used for multiplications and additions", but it seems to me this is not really the main difference with Lemma 2.4, since $u_w > u_s$ is already assumed in Lemma 2.4? Instead, the key difference is the assumption that the multiplications are being performed exactly (i.e., $u_p = 0$), which explains the improvement by 1 of the bound? I suggest clarifying and, perhaps, unifying both results (with $z = 1$ or 2 depending on u_p).

Rounding error analysis framework Rounding error analysis is a difficult topic to write about, as the technical complications can often obfuscate the main message and conclusions of the analysis. I feel this is happening in this article, and I assume this is the reason why most proofs were moved to the appendix in an attempt to make the manuscript more "digestible". The problem with this solution is that it makes all the intermediate results from sections 3.2.1 and 3.2.2 *seem* irrelevant, since these results

are only required for the proof of the main result. Besides, if the main contribution of this article is to perform the first error analysis of mixed-precision HQR, the proofs are an important part of this contribution, and the reader should not only be able to understand them and check their correctness, but even learn something from them.

In order to achieve this (admittedly difficult) objective, I suggest the authors move the proofs in the main body of the article, and implement the following changes.

- Add sytematically after each result a comment or intepretation. In particular, answer the following questions: what has changed compared with the uniform precision result? What improvements has the use of mixed-precision brought? Should we be happy with the bound obtained? It is important not to lose sight of the big picture (more on this in the next paragraph).

- Use the $\tilde{\gamma}_n$ notation from [2] to avoid keeping track of all constants: $\tilde{\gamma}_n = cnu/(1-cnu)$ for any constant c which is not relevant.

- Keep different precisions in the bound rather than trying to merge them as proposed in Lemma 2.3 with the introduction of the constant d: that is, write $c_1 u_{low} + c_2 u_{high}$ rather than $(c_1 + d)u_{low}$, with $d = \lfloor c_2 u_{high}/u_{low} \rfloor$. I do not think introducing this d constant is needed, as it complicates both the analysis and its interpretation: indeed, keeping different precisions separately allows for an intuitive understanding of the bounds by discarding the terms proportional to u_{high}.

Conclusions from the mixed-precision HQR analysis (section 3) What conclusions can we draw from the mixed-precision HQR analysis? The end of section 3 feels underwhelming and could better highlight the main result. The authors compare their bound $\gamma_w^{(6d+6z+13)} = \tilde{\gamma}_w^{(d)}$, where $d \approx mu_s/u_w$, to the bound $\tilde{\gamma}^{(m)}$. For this comparison it is however crucial to specify the precision of the latter bound! It seems to me that in this context the most relevant comparison is with $\tilde{\gamma}_w^{(m)}$, which is then always greater than the mixed-precision bound by a factor about $\min(m, u_w/u_s)$[1]. Therefore, the mixed-precision bound is much better than its uniform precision counterpart for large m and for $u_w \gg u_s$.

This conclusion is not clearly stated or emphasized, while it seems to me it is one of the main results of the article (which should be highlighted in the abstract, introduction, and conclusion?). Moreover, this raises many interesting questions:

- Are we happy about this result? The mixed-precision bound still grows with n: is that OK? In the case of LU factorization (see [1]), the mixed-precision bound is $2u_{low} + nu_{high}$ instead of nu_{low}, so the use of mixed-precision allows to drop the

[1] Rather than using this d constant, rewriting the bounds as suggested in the previous paragraph: $\tilde{\gamma}_w^{(1)} + \tilde{\gamma}_s^{(m)}$ and $\tilde{\gamma}_w^{(m)}$ makes this a lot more apparent.

dependence on n to first order. Why is it not the case for QR? Can we modify the algorithm or the way mixed-precision is used to drop a factor n instead of m (or both!) in the bound?

- Do any of the answers to the questions above change from an experimental standpoint? What are the practical benefits of mixed-precision? Do we actually need to worry about m? Or n? Can probabilistic analyses help?[2]

I suggest that the authors clarify the conclusions by answering the above questions and by adding experiments to compare the uniform and mixed precision HQR for a range of sizes m and n.

Conclusion from the HQR vs TSQR comparison (section 4) Section 4 then analyzes TSQR and also leads to conclusions that I feel could be stated more clearly or be better emphasized (e.g., in the abstract and conclusion). To summarize, taking the backward error as an example (the forward error case is similar), these are the four bounds of interest:

$$\|\Delta A\| \lesssim \begin{cases} n^{3/2}\tilde{\gamma}_w^{(m)} & \text{(uniform-precision HQR)} \\ n^{3/2}\big(\tilde{\gamma}_w^{(1)} + \tilde{\gamma}_s^{(m)}\big) & \text{(mixed-precision HQR)} \\ n^{3/2}\big(\tilde{\gamma}_w^{(2^{-L}m)} + \tilde{\gamma}_w^{(Ln)}\big) & \text{(uniform-precision TSQR)} \\ n^{3/2}\big(\tilde{\gamma}_w^{(1)} + \tilde{\gamma}_s^{(2^{-L}m)} + \tilde{\gamma}_s^{(Ln)}\big) & \text{(mixed-precision TSQR)} \end{cases}$$

I suggest adding a table in the article to summarize these four bounds as done here (the different types of errors may be unified with some parameter ξ_n which is equal to either 1 or $\sqrt{n}$).

HQR and TSQR are first compared in a uniform precision setting. In this setting, the conclusion is simply that there need to exist an L such that $\max(2^{-L}m, Ln) \ll m$ for TSQR to be beneficial w.r.t HQR. This is more or less explained in section 4.2.1, although I was initially confused by the fact that the ratio $(L+1)/2^L$ should be inversed and by the sentence "if we assume $m/2^L = 2n$", which actually is not an assumption but is taken as an example, as far as I understand. I think this section needs a rewrite to simplify/clarify the discussion.

Then, HQR and TSQR are compared in a mixed precision setting. In this setting, the conclusion seems to be that in the regime where $\tilde{\gamma}_w^{(1)}$ dominates the error, there is no accuracy benefit in using TSQR over HQR. I think this result should be better emphasized and not overshadowed by Figure 3, which shows that in practice TSQR

[2] To be clear, I am not suggesting that the authors perform a probabilistic analysis, which is an entirely different work. But perhaps a quick look at [3] (which does not directly cover QR) may reveal that the dependence on m is more important than that on n in practice (that would be really good news for this article, which reduces the dependence on m!) or conversely (bad news?).

may still deliver a smaller error. In particular, the conclusion (section 6) mentions only this latter experimental observation, and seems to suggest that TSQR is therefore more adapted to mixed precision than HQR. I disagree with this conclusion. TSQR may reduce the error in both uniform and mixed precision settings, but to conclude that is is more adapted to mixed-precision than HQR one would need to reproduce the same experiment in Figure 3 in a uniform precision setting: I suggest the authors add this experiment. Perhaps the conclusion will then be that the error reduction achieved by TSQR is actually larger in uniform precision, which would then support the theoretical bounds! (and lead to a different conclusion: TSQR is *less* relevant in a mixed-precision context). Moreover, rather than varying the condition number, I suggest varying m and n. Indeed it seems natural that the result of this HQR vs TSQR comparison will be very dependent on m and n. I am unsure how relevant the right plot of Figure 3 is, given the extreme zoom on the y-axis: essentially all methods yield an error of the same order.

Missing references to relevant work Some relevant work should be acknowledged.

- Most importantly, [1] develops a rounding error analysis of mixed-precision matrix–matrix products and LU factorization, in particular applied to GPU tensor cores. There are many similarities between the two works although I don't think the originality of either is under question: LU and QR are sufficiently different algorithms to warrant two analyses.

- The present manuscript also discusses the sharpness of the bounds at relatively great length in section 2.1 and mention probabilistic analyses. The articles [3, 5] should be cited.

- Finally, the simulation of low precision arithmetic is also discussed in section 2.1. The authors propose Algorithm 1, which corresponds to Simulation 3.1 in [4]. The proof that Algorithm 1 results in a faithful simulation is actually not immediate and is done in [4].

Other inaccuracies

- There seems to be a confusion between forward and backward error in several places throughout the article. Equation (2.8) is *not* a forward error bound, nor is it a backward error, because of the division by $|x|^T|y|$, which can be much bigger than $|x^Ty|$ if cancellation occurs. A similar confusion takes place P8, around Lemma 2.4 and Corollary 2.5 (L217,230,243): the text keeps referring to the "forward error" but these results actually bound the *backward* error. A forward error bound for inner products is $|fl(x^Ty) - x^Ty| = |\Delta x^Ty| \leq \gamma_w^{(d+1)}|x|^T|y|$. Note the dependence on the condition number $\kappa = \frac{|x|^T|y|}{|x^Ty|}$.

- P1L15: "standard algorithms may no longer be numerically stable when using half precision": I understand what the authors mean, but it sounds strange that the numerical stability of an algorithm should depend on the precision used (by definition, an algorithm is numerically stable if it produces an error that is close to the machine precision, so in absence of overflow/underflow numerical stability should be independent of the precision used). I suggest rewording, perhaps something in the line of "standard algorithms yield insufficient accuracy when using half precision".
- P2L44: "QR factorization is known to provide a backward stable solution and thus is ideal for mixed-precision": I do not follow the causal link between backward stability and potential to use mixed-precision.
- P2L49: fp16 *is* the IEEE half precision type so should be removed in this context. bfloat should be changed to bfloat16.
- P5L125: "k represents the number of FLOPs". There is actually no link between k in $\gamma^{(k)}$ and the FLOPs. For example, for $x, y \in \mathbb{R}^n$, the inner product $s = x^T y$ performs $2n-1$ FLOPs and leads to an error bound $\gamma^{(n)}$, while the outer product $s = xy^T$ performs n^2 FLOPs but leads to an error bound $\gamma^{(1)}$. Rewording is needed.
- P20L622: "FLOPs are now considered free": I suggest softening this unnecessarily strong statement.

Minor comments

- Everywhere: the constant $\gamma^{(k)}$ is usually written γ_k instead [2]. The superscript is then used to distinguish precisions, e.g., $\gamma_k^{(w)}$. In this review I have used the same notation as the authors to avoid confusions but perhaps they could switch to the more standard notation.
- Title: I suggest moving "mixed-precision" in front of "Householder", as it is the algorithm rather than the analysis that is mixed-precision. Also, should TSQR be added to title, since it is an important contribution?
- P1L20: what does "weight" refer to in this context?
- P2L57: "can successfully" → "can be successfully".
- P5L134: $\gamma_p^{(d+2)}$ has not been defined yet.
- P12L329: the middle term should be $(1+\delta_w)(x_1 - \sigma - \Delta\sigma)$, rather than $(1+\delta_w)(\sigma + \Delta\sigma)$. Moreover, the last equality is only true because no cancellation can happen, since x_1 and σ have the same sign: this should be commented on.

- Equations (4.6) and (4.7): isn't the $\sqrt{n}$ factor on the wrong equation?
- P18L527: "for the a meaningful".
- P18531: as mentioned above, the $(L+1)/2^L$ factor is inversed.
- P20L599: I find it very strange that the backward error depends on the condition number of the matrix! Is it rather the forward error that is being plotted?
- Section 5: given the relatively theoretical nature of this article, section 5 felt slightly out of place to me. Given that the article is quite long, perhaps the authors could consider including section 5 in another piece of work?

References

[1] Pierre Blanchard, Nicholas J. Higham, Florent Lopez, Theo Mary, and Srikara Pranesh. Mixed precision block fused multiply-add: Error analysis and application to GPU tensor cores. MIMS EPrint 2019.18, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, September 2019. 15 pp.

[2] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. xxx+680 pp. ISBN 0-89871-521-0.

[3] Nicholas J. Higham and Theo Mary. A new approach to probabilistic rounding error analysis. *SIAM J. Sci. Comput.*, 41(5):A2815–A2835, 2019.

[4] Nicholas J. Higham and Srikara Pranesh. Simulating low precision floating-point arithmetic. *SIAM J. Sci. Comput.*, 41(5):C585–C602, 2019.

[5] Ilse C. F. Ipsen and Hua Zhou. Probabilistic error analysis for inner products. arXiv:1906.10465, June 2019.