B639388 Subcontract Award LLNL-Charles University: Mixed Precision Numerical Linear Algebra

Quarter 3 Report: Preliminary Theoretical Analysis of Mixed Precision Krylov Subspace Methods

Erin Carson and Tomáš Gergelits

1 Summary of activities

The third quarter of the project was spent performing theoretical finite precision analysis of Krylov subspace method variants that use mixed precision. Our focus here is on the Conjugate Gradient (CG) method and the Lanczos method. We have performed an analysis of maximum attainable accuracy for the classical CG method in which 3 precisions are used: a working precision ε, a precision ε_{IP} for the inner product computations, and a precision ε_{MV} for the matrix-vector products. Our results show that performing inner product computations in lower precision does not affect the attainable accuracy. Further, we have performed a complete error analysis of the s-step Lanczos algorithm. In this case, we show that the numerical behavior of the algorithm can be significantly improved by using extra precision in a small part of the computation. We summarize the main theorems in the remainder of the document. Other activities include attending biweekly xSDK meetings.

The subsequent quarter will be spent finalizing these results into technical reports and/or manuscripts for submission to journals, as well as identifying opportunities for future work.

2 Maximum Attainable Accuracy of Mixed Precision HSCG

In this work, we have focused on a mixed precision variant of the Hestenes and Stiefel variant of CG [5], which we call HSCG, shown in Algorithm 1. For simplicity, we assume here an initial value of 0 for the approximate solution vector.

Algorithm 1 Mixed precision Hestenes and Stiefel CG (HSCG)

```
Input: $n \times n$ symmetric matrix $A$, length-$n$ vector $b$, number of steps $k$
Output: Approximate solution $x_{k+1}$
 1: $r_0 = b$, $p_0 = b$, $x_0 = 0$
 2: for $i = 1, \dots, k$ do
 3:     $w_{i-1} = Ap_{i-}$ (precision $\varepsilon_{MV}$)
 4:     $\phi_{i-1} = p_{i-1}^T w_{i-1}$ (precision $\varepsilon_{IP}$)
 5:     $\rho_{i-1} = r_{i-1}^T r_{i-1}$ (precision $\varepsilon_{IP}$)
 6:     $\alpha_{i-1} = \rho_{i-1}/\phi_{i-1}$ (precision $\varepsilon$)
 7:     $x_i = x_{i-1} + \alpha_{i-1}p_{i-1}$ (precision $\varepsilon$)
 8:     $r_i = r_{i-1} - \alpha_{i-1}w_{i-1}$ (precision $\varepsilon$)
 9:     $\rho_i = r_i^T r_i$ (precision $\varepsilon_{IP}$)
10:     $\beta_i = \rho_i/\rho_{i-1}$ (precision $\varepsilon$)
11:     $p_i = r_i + \beta_i p_{i-1}$ (precision $\varepsilon$)
12: end for
```

For our mixed precision variant, we assume all computations in HSCG are done and data stored in working precision ε, except for inner products which are computed in precision ε_{IP} and matrix-vector products which are computed in precision ε_{MV}. We assume that $\varepsilon_{MV} \geq \varepsilon$ and $\varepsilon_{IP} \geq \varepsilon$, so there is negligible error in storing the results. Following the proof of Greenbaum [4], we can prove a bound on maximum attainable accuracy for this mixed precision case.

We use standard rounding error analysis to bound the following errors, where A is an $n \times n$ matrix with at most m nonzeros per row, α is a scalar, and v and w are length-n vectors.

$$\begin{aligned}
fl(\alpha v) &= \alpha v + \delta_1, & \|\delta_1\| &\leq \varepsilon \|\alpha v\| \\
fl(v + w) &= v + w + \delta_2, & \|\delta_2\| &\leq \varepsilon(\|v\| + \|w\|) \\
fl(v^T w) &= v^T w + \delta_3, & \|\delta_3\| &\leq n(\varepsilon_{IP} + O(\varepsilon_{IP}^2))\|v\|\|w\| \\
fl(Av) &= Av + \delta_4, & \|\delta_4\| &\leq mn^{1/2}\varepsilon_{MV}\|A\|\|v\|
\end{aligned}$$

We can write the true residual $b - Ax_k = b - Ax_k - r_k + r_k$, and thus

$$\|b - Ax_k\| \leq \|b - Ax_k - r_k\| + \|r_k\|.$$

Assuming that the method converges (which is another issue to be investigated), we will eventually have $\|r_k\| \to 0$, and thus the true residual (related to the accuracy) will be limited by the size of the residual gap $\|b - Ax_k - r_k\|$. We therefore seek to bound this quantity.

Computing line 7 of Algorithm 1, we have

$$x_i = x_{i-1} + \alpha_{i-1}p_{i-1} + \eta_i,$$

where, using the rounding error bounds, we have

$$\|\xi_i\| \leq \varepsilon\|x_{i-1}\| + (2\varepsilon + \varepsilon^2)\|\alpha_{i-1}p_{i-1}\|. \tag{2.1}$$

Again, using the standard rounding error bounds, in finite precision line 8 becomes

$$r_i = r_{i-1} - \alpha_{i-1}Ap_{i-1} + \eta_i,$$

where

$$\|\eta_i\| \leq \varepsilon\|r_{i-1}\| + (2\varepsilon + \varepsilon^2)\|\alpha_{i-1}Ap_{i-1}\| + \varepsilon_{MV}(1+\varepsilon)^2 mn^{1/2}\|A\|\|\alpha_{i-1}p_{i-1}\|. \tag{2.2}$$

We can then write

$$b - Ax_i - r_i = (b - Ax_{i-1} - r_{i-1}) - A\xi_i - \eta_i = (b - Ax_0 - r_0) - \sum_{j=1}^{i}(A\xi_j + \eta_j).$$

Taking norms and dividing by $\|A\|\|x\|$, we have

$$\frac{\|b - Ax_i - r_i\|}{\|A\|\|x\|} \leq \frac{\|b - Ax_0 - r_0\|}{\|A\|\|x\|} + \sum_{j=1}^{i}\left(\frac{\|\xi_j\|}{\|x\|} + \frac{\|\eta_j\|}{\|A\|\|x\|}\right).$$

For the initial residual gap, we can apply the standard rounding error bounds as well as $\|b\| \leq \|A\|\|x\|$ to write

$$\frac{\|b - Ax_0 - r_0\|}{\|A\|\|x\|} \leq (1+\varepsilon)mn^{1/2}\varepsilon_{MV}\frac{\|x_0\|}{\|x\|} + \varepsilon + e^{\frac{\|x_0\|}{\|x\|}}. \tag{2.3}$$

We define the quantity $\Theta_i \equiv \max_{j\leq i}\|x_j\|/\|x\|$. Then using

$$\alpha_{i-1}p_{i-1} = x_i - x_{i-1} - \xi_i, \tag{2.4}$$

and assuming that $1 - 2\varepsilon - \varepsilon^2 > 0$, we have

$$\|\xi_i\| \leq \varepsilon\|x_{i-1}\| + (2\varepsilon + \varepsilon^2)(\|x_i\| + \|x_{i-1}\|).$$

Therefore we can write

$$\sum_{j=1}^{i}\frac{\|\xi_j\|}{\|x\|} \leq (5\varepsilon + O(\varepsilon^2))i\Theta_i. \tag{2.5}$$

We now turn to bounding the terms in (2.2). Using (2.4), for the third term, we have

$$\varepsilon_{MV}(1+\varepsilon)^2 mn^{1/2}\|A\|\|\alpha_{j-1}p_{j-1}\| \leq mn^{1/2}\varepsilon_{MV}(1 + O(\varepsilon^2))\|A\|(\|x_j\| + \|x_{j-1}\|). \tag{2.6}$$

For the second term, using $\alpha_{j-1}Ap_{j-1} = A(x_j - x_{j-1} - \xi_j)$, we can write

$$(2\varepsilon + \varepsilon^2)\|\alpha_{j-1}Ap_{j-1}\| \leq (2\varepsilon + O(\varepsilon^2))\|A\|(\|x_j\| + \|x_{j-1}\|). \tag{2.7}$$

To finish bounding the terms, we need to use induction. Assume that each $\|\eta_k\|$ is bounded by $O(\varepsilon + \varepsilon_{MV})\|A\|(\|x\| + \max_{\ell\leq k}\|x_{|}ell\|)$ for $k = 1, \ldots, j-1$. This is clearly satisfied for η_1. Now we have that r_{j-1} satisfies

$$\begin{aligned} r_{j-1} &= b - Ax_{j-1} - (b - Ax_0 - r_0) + \sum_{k=1}^{j-1} A\xi_k + \eta_k \\ &= A(x - x_{j-1}) - (b - Ax_0 - r_0) + \sum_{k=1}^{j-1} A\xi_k + \eta_k, \end{aligned}$$

and taking norms and using (2.5) along with the induction hypothesis, we have the bound

$$\|r_{j-1}\| \leq \|A\|\|x - x_{j-1}\| + (1\varepsilon)mn^{1/2}\varepsilon_{MV}\|A\|\|x_0\| + \varepsilon\|A\|\|x\| + \varepsilon\|A\|\|x_0\| + \|A\|\|x\|(5\varepsilon + O(\varepsilon^2))(j-1)\Theta_{j-1} + (j-1)O(\varepsilon + \varepsilon_{MV})\|A\| \left(\|x\| + \max_{\ell \leq j-1} \|x_\ell\|\right).$$

The above along with (2.6) and (2.6) gives

$$\|\eta_j\| \leq (\varepsilon + O(\varepsilon\varepsilon_{MV}))\|A\|\|x\| + (5\varepsilon + 2mn^{1/2}\varepsilon_{MV} + O(\varepsilon\varepsilon_{MV})) \max_{k \leq j} \|x_k\|. \tag{2.8}$$

Since $\|\eta_j\|$ is bounded by $O(\varepsilon + \varepsilon_{MV})\|A\|(\|x\| + \max_{k \leq j} \|x_| k\|)$, the induction hypothesis holds.

We then have

$$\sum_{j=1}^{i} \frac{\|\eta_j\|}{\|A\|\|x\|} \leq i(\varepsilon + O(\varepsilon\varepsilon_{MV}) + \left(5\varepsilon + 2mn^{1/2}\varepsilon_{MV} + O(\varepsilon\varepsilon_{MV})\right)\Theta_i), \tag{2.9}$$

and combining this with (2.5) and (2.3) and ignoring terms of order $O(\varepsilon\varepsilon_{MV})$ and higher, we have the final bound

$$\frac{\|b - Ax_i - r_i\|}{\|A\|\|x\|} \leq (1+i)\varepsilon + \left((10i+1)\varepsilon + (2i+1)mn^{1/2}\varepsilon_{MV}\right)\Theta_i.$$

There are two conclusions that can be drawn from this bound. First, notice that the quantity ε_{IP} appears nowhere in the bound. This means that, assuming that the precisions used as such that the algorithm converges, the precision used for inner products does not contribute to the residual gap and thus does not affect the attainable accuracy. This means that lower precision can be used to compute the inner products with no affect to the accuracy. However, we caution that this can indeed affect the convergence delay; further study is needed here. The second conclusion is that, assuming that $\varepsilon_{MV} \geq \varepsilon$, the error from the matrix vector product will dominate the residual gap. Thus performing the matrix-vector products in lower precision can affect the accuracy, and further, if the matrix-vector products are computed in lower precision, then in terms of accuracy, there is no benefit to using a working precision $\varepsilon < \varepsilon_{MV}$. These theoretical results are supported by numerical experiments which were included in our previous report.

3 Mixed Precision s-step Lanczos

The Lanczos method, shown in Algorithm 2 is a well-known Krylov subspace method for solving large, sparse eigenvalue problems. Compared to the classical Lanczos algorithm, the s-step Lanczos variant has the potential to improve performance by asymptotically decreasing the synchronization cost per iteration. However, this comes at a cost. Despite being mathematically equivalent, the s-step variant is known to behave quite differently in finite precision, with potential for greater loss of accuracy and a decrease in the convergence rate relative to the classical algorithm. It has previously been shown that the errors that occur in the s-step version follow the same structure as the errors in the classical algorithm, but with the addition of an amplification factor that depends on the square of the condition number of the $O(s)-$dimensional Krylov bases computed in each outer loop [1]. As the condition number of these s-step bases grows (in some cases very quickly) with s, this limits the parameter s that can be chosen and thus limits the performance that can be achieved.

Algorithm 2 Lanczos

Input: n-by-n real symmetric matrix A and length-n starting vector v_1 such that $\|v_1\|_2 = 1$

```
1: $u_1 = Av_1$
2: for $m = 1, 2, \ldots$ until convergence do
3:     $\alpha_m = v_m^T u_m$
4:     $w_m = u_m - \alpha_m v_m$
5:     $\beta_{m+1} = \|w_m\|_2$
6:     $v_{m+1} = w_m/\beta_{m+1}$
7:     $u_{m+1} = Av_{m+1} - \beta_{m+1}v_m$
8: end for
```

We have completed an analysis which proves that if a select few computations in s-step Lanczos are performed in double the working precision, the error terms then depend only linearly on the conditioning of the s-step bases. This has the potential for drastically improving the numerical behavior of the algorithm with little impact on per-iteration performance.

3.1 The s-step Lanczos variant

For clarity of the main results, we give a brief derivation of the s-step Lanczos algorithm. Suppose we are beginning iteration $m = sk + 1$ where $k \in \mathbb{N}$ and $0 < s \in \mathbb{N}$. By induction on lines 6 and 7 of Algorithm 2, we can write

$$v_{sk+j}, u_{sk+j} \in \mathcal{K}_{s+1}(A, v_{sk+1}) + \mathcal{K}_{s+1}(A, u_{sk+1}) \tag{3.1}$$

for $j \in \{1, \ldots, s+1\}$, where $\mathcal{K}_i(A, x) = \text{span}\{x, Ax, \ldots, A^{i-1}x\}$ denotes the Krylov subspace of dimension i of matrix A with respect to vector x. Note that since $u_1 = Av_1$, if $k = 0$ we have

$$v_j, u_j \in \mathcal{K}_{s+2}(A, v_1).$$

for $j \in \{1, \ldots, s+1\}$.

For $k > 0$, we then define 'basis matrix' $\mathcal{Y}_k = [\mathcal{V}_k, \mathcal{U}_k]$, where $\mathcal{V}_k$ and $\mathcal{U}_k$ are size n-by-$(s+1)$ matrices whose columns form bases for $\mathcal{K}_{s+1}(A, v_{sk+1})$ and $\mathcal{K}_{s+1}(A, u_{sk+1})$, respectively. For $k = 0$, we define $\mathcal{Y}_0$ to be a size n-by-$(s+2)$ matrix whose columns form a basis for $\mathcal{K}_{s+2}(A, v_1)$. Then by (3.1), we can represent v_{sk+j} and u_{sk+j}, for $j \in \{1, \ldots, s+1\}$, by their coordinates (denoted with primes) in $\mathcal{Y}_k$, i.e.,

$$v_{sk+j} = \mathcal{Y}_k v'_{k,j}, \qquad u_{sk+j} = \mathcal{Y}_k u'_{k,j}. \tag{3.2}$$

Note that for $k = 0$, the coordinate vectors are length $s+2$ and for $k > 0$, the coordinate vectors are length $2s+2$. We can write a similar equation for auxiliary vector w_{sk+j}, i.e., $w_{sk+j} = \mathcal{Y}_k w'_{k,j}$ for $j \in \{1, \ldots, s\}$. We define also the Gram matrix $G_k = \mathcal{Y}_k^T \mathcal{Y}_k$, which is size $(s+2)$-by-$(s+2)$ for $k = 0$ and $(2s+2)$-by-$(2s+2)$ for $k > 0$. Using this matrix, the inner products in lines 3 and 5 can be written

$$\alpha_{sk+j} = v_{sk+j}^T u_{sk+j} = v'^T_{k,j} \mathcal{Y}_k^T \mathcal{Y}_k u'_{k,j} = v'^T_{k,j} G_k u'_{k,j} \quad \text{and} \tag{3.3}$$

$$\beta_{sk+j+1} = (w_{sk+j}^T w_{sk+j})^{1/2} = (w'^T_{k,j} \mathcal{Y}_k^T \mathcal{Y}_k w'_{k,j})^{1/2} = (w'^T_{k,j} G_k w'_{k,j})^{1/2}. \tag{3.4}$$

We assume that the bases are generated via polynomial recurrences represented by the matrix $\mathcal{B}_k$, which is in general upper Hessenberg but often tridiagonal in practice. The recurrence can thus be written in matrix form as

$$A\underline{\hat{\mathcal{Y}}}_k = \hat{\mathcal{Y}}_k \mathcal{B}_k$$

where $\mathcal{B}_k$ is size $(s+2)$-by-$(s+2)$ for $k = 0$ and size $(2s+2)$-by-$(2s+2)$ for $k > 0$, and

$$\underline{\hat{\mathcal{Y}}}_k = \left[\hat{\mathcal{V}}_k [I_s, 0_{s,1}]^T, 0_{n,1}, \hat{\mathcal{U}}_k [I_s, 0_{s,1}]^T, 0_{n,1}\right].$$

Therefore, for $j \in \{1, \ldots, s\}$,

$$Av_{sk+j+1} = A\mathcal{Y}_k v'_{k,j+1} = A\underline{\hat{\mathcal{Y}}}_k v'_{k,j+1} = \mathcal{Y}_k \mathcal{B}_k v'_{k,j+1}. \tag{3.5}$$

Thus, to compute iterations $sk+2$ through $sk+s+1$ in s-step Lanczos, we first generate basis matrix $\mathcal{Y}_k$ such that (3.5) holds, and we compute the Gram matrix G_k from the resulting basis matrix. Then updates to the length-n vectors can be performed by updating instead the length-$(2s+2)$ coordinates for those vectors in $\mathcal{Y}_k$. Inner products and multiplications with A become smaller operations which can be performed locally.

It was shown in [1] that the loss of orthogonality, loss of normality, and error in the s-step Lanczos recurrence can all be bounded by quantities with a factor related to the square of the condition number of the computed s-step bases. The analysis in [1] suggests that a mixed precision approach can potentially have significant benefit. From the results in this work, it is clear that the square of the s-step basis condition number enters the bounds through the formation of the Gram matrix G_k.

We therefore suggest a mixed precision approach as follows. In each outer loop iteration, the Gram matrix G_k should be computed and stored in precision ε^2 (double the working precision). This will double the number of bits moved, but since G_k is modestly-sized (square with dimension $2s+2$), this is not expected to cause a bandwidth bottleneck. This will not affect the number of synchronizations, and further, only needs to occur every s iterations. Within each inner loop, G_k is applied twice to a single vector. The inner products involved in this matrix-vector multiplication should be accumulated in precision ε^2, but the result can be stored in the working precision ε. Again, since G_k is of small dimension, and these matrix-vector multiplies are done locally on each processor, this is an insignificant cost to performance. All other computations are performed in the working precision ε. We summarize this approach in Algorithm 3.

Note that in Algorithm 3 we show the length-n vector updates in each inner iteration (lines 18 and 20) for clarity, although these vectors play no part in the inner loop iteration updates. In practice, the basis change operation (3.2) can be performed on a block of coordinate vectors at the end of each outer loop to recover v_{sk+i} and u_{sk+i}, for $i \in \{2, \ldots, s+1\}$.

Algorithm 3 Mixed precision s-step Lanczos

```
Input: n-by-n real symmetric matrix $A$ and length-$n$ starting vector $v_1$ such that $\|v_1\|_2 = 1$
 1: $u_1 = Av_1$ (precision $\varepsilon$)
 2: for $k = 0, 1, \ldots$ until convergence do
 3:     Compute $\mathcal{Y}_k$ with change of basis matrix $\mathcal{B}_k$ (precision $\varepsilon$).
 4:     Compute and store $G_k = \mathcal{Y}_k^T \mathcal{Y}_k$ in precision $\varepsilon^2$.
 5:     $v'_{k,1} = e_1$
 6:     if $k = 0$ then
 7:         $u'_{0,1} = \mathcal{B}_k e_1$
 8:     else
 9:         $u'_{k,1} = e_{s+2}$
10:     end if
11:     for $j = 1, 2, \ldots, s$ do
12:         Compute $g = G_k u'_{k,j}$ in precision $\varepsilon^2$, store in precision $\varepsilon$.
13:         $\alpha_{sk+j} = v'^T_{k,j} g$ (precision $\varepsilon$)
14:         $w'_{k,j} = u'_{k,j} - \alpha_{sk+j} v'_{k,j}$ (precision $\varepsilon$)
15:         Compute $c = G_k w'_{k,j}$ in precision $\varepsilon^2$, store in precision $\varepsilon$.
16:         $\beta_{sk+j+1} = (w'^T_{k,j} c)^{1/2}$ (precision $\varepsilon$)
17:         $v'_{k,j+1} = w'_{k,j} / \beta_{sk+j+1}$ (precision $\varepsilon$)
18:         $v_{sk+j+1} = \mathcal{Y}_k v'_{k,j+1}$ (precision $\varepsilon$)
19:         $u'_{k,j+1} = \mathcal{B}_k v'_{k,j+1} - \beta_{sk+j+1} v'_{k,j}$ (precision $\varepsilon$)
20:         $u_{sk+j+1} = \mathcal{Y}_k u'_{k,j+1}$ (precision $\varepsilon$)
21:     end for
22: end for
```

3.2 Analysis of mixed precision approach

We will model floating point computation in a precision ε using the following standard conventions (see, e.g., [3]): for vectors $u, v \in \mathbb{R}^n$, matrices $A \in \mathbb{R}^{n\times m}$ and $G \in \mathbb{R}^{n\times n}$, and scalar α,

$$\begin{aligned}
fl_\varepsilon(u - \alpha v) &= u - \alpha v - \delta w, && |\delta w| \le (|u| + 2|\alpha v|)\varepsilon,\\
fl_\varepsilon(v^T u) &= (v + \delta v)^T u, && |\delta v| \le n\varepsilon |v|,\\
fl_\varepsilon(Au) &= (A + \delta A)u, && |\delta A| \le m\varepsilon |A|, \quad \text{and}\\
fl_\varepsilon(A^T A) &= A^T A + \delta E, && |\delta E| \le n\varepsilon |A^T||A|.
\end{aligned}$$

where $fl_\varepsilon()$ represents the evaluation of the given expression in floating point arithmetic with unit roundoff ε and terms with δ denote error terms. We decorate quantities computed in finite precision arithmetic with hats, e.g., if we are to compute the expression $\alpha = v^T u$ in finite precision, we get $\hat{\alpha} = fl_\varepsilon(v^T u)$. We further assume no underflow or overflow, and also assume that no breakdown of the method occurs.

We will use the following lemma, which will be useful in our analysis. The proof is trivial and thus omitted.

Lemma 3.1 *Assume we have rank-r matrix $Y \in \mathbb{R}^{n\times r}$, where $n \ge r$. Let Y^+ denote the pseudoinverse of Y, i.e., $Y^+ = (Y^T Y)^{-1} Y^T$. Then for any vector $x \in \mathbb{R}^r$, we can bound*

$$\| |Y|\, |x| \|_2 \le \| |Y| \|_2 \|x\|_2 \le \Gamma \|Yx\|_2.$$

where $\Gamma = \left\|Y^+\right\|_2 \left\| |Y| \right\|_2 \le \sqrt{r} \left\|Y^+\right\|_2 \left\|Y\right\|_2$.

We note that the term Γ can be thought of as a type of condition number for the matrix Y. In the analysis, we will apply the above lemma to the computed 'basis matrix' $\hat{\mathcal{Y}}_k$; in particular, we will use the definition $\Gamma_k = \left\|\mathcal{Y}_k^+\right\|_2 \left\| |\mathcal{Y}_k| \right\|_2$. We assume throughout that the generated bases $\hat{\mathcal{U}}_k$ and $\hat{\mathcal{V}}_k$ are numerically full rank. That is, all singular values of $\hat{\mathcal{U}}_k$ and $\hat{\mathcal{V}}_k$ are greater than $\epsilon n \cdot 2^{\lfloor \log_2 \theta_1 \rfloor}$ where θ_1 is the largest singular value of A. We further make the key assumption that $\varepsilon n \Gamma_k \ll 1$ for all k. Based on this assumption, throughout the analysis we will drop terms of order $(\varepsilon n \Gamma_k)^2$. Including these terms in the analysis would not change the fundamental structure of the results, but would result in a serious overestimate of the bounds due to growing constant terms.

We state the main theorem that we have proved. We omit the lengthy and technical proof from this report; it will be included in a future publication.

Theorem 3.2 *Assume that Algorithm 3 is implemented in floating point with working precision ε and applied for $m = sk + j$ steps to the n-by-n real symmetric matrix A with at most N nonzeros per row, starting with vector v_1 with $\|v_1\|_2 = 1$. Let $\sigma \equiv \|A\|_2$, $\theta\sigma = \||A|\|_2$ and $\tau_k\sigma = \||\mathcal{B}_k|\|_2$, where $\mathcal{B}_k$ is defined in* (3.5)*, and let*

$$\bar{\Gamma}_k = \max_{i\in\{0,\ldots,k\}} \|\hat{\mathcal{Y}}_i^+\|_2 \||\hat{\mathcal{Y}}_i|\|_2 \geq 1 \quad \textit{and} \quad \bar{\tau}_k = \max_{i\in\{0,\ldots,k\}} \tau_i,$$

where above the superscript '+' denotes the Moore-Penrose pseudoinverse, i.e., $\hat{\mathcal{Y}}_i^+ = (\hat{\mathcal{Y}}_i^T\hat{\mathcal{Y}}_i)^{-1}\hat{\mathcal{Y}}_i^T$. Further, assume that

$$4m\varepsilon\big((N{+}2s{+}5)\theta + (4s{+}9)\bar{\tau}_k{+}n{+}22s{+}37\big)\bar{\Gamma}_k < 1. \tag{3.6}$$

Then $\hat{\alpha}_{sk+j}$, $\hat{\beta}_{sk+j+1}$, and $\hat{v}_{sk+j+1}$ will be computed such that, for $i \in \{1,\ldots,m\}$,

$$A\hat{V}_m = \hat{V}_m\hat{T}_m + \hat{\beta}_{m+1}\hat{v}_{m+1}e_m^T - \delta\hat{V}_m,$$

with

$$\hat{V}_m = [\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_m]$$
$$\delta\hat{V}_m = [\delta\hat{v}_1, \delta\hat{v}_2, \ldots, \delta\hat{v}_m]$$
$$\hat{T}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 & & \\ \hat{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \hat{\beta}_m \\ & & \hat{\beta}_m & \hat{\alpha}_m \end{bmatrix}$$

and

$$\|\delta\hat{v}_i\|_2 \leq \varepsilon_1\sigma, \tag{3.7}$$
$$\hat{\beta}_{i+1}|\hat{v}_i^T\hat{v}_{i+1}| \leq \varepsilon_0\sigma, \tag{3.8}$$
$$|\hat{v}_{i+1}^T\hat{v}_{i+1} - 1| \leq \varepsilon_0/2, \quad \textit{and} \tag{3.9}$$
$$\left|\hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A\hat{v}_i\|_2^2\right| \leq 2i(3\varepsilon_0 + 2\varepsilon_1)\sigma^2, \tag{3.10}$$

where

$$\varepsilon_0 \equiv 2\varepsilon(6s{+}11)\bar{\Gamma}_k \quad \textit{and} \quad \varepsilon_1 \equiv \varepsilon\big((N{+}2s{+}5)\theta + (4s{+}9)\bar{\tau}_k + (10s{+}16)\big)\bar{\Gamma}_k.$$

Furthermore, if R_m is the strictly upper triangular matrix such that

$$\hat{V}_m^T\hat{V}_m = R_m^T + diag(\hat{V}_m^T\hat{V}_m) + R_m,$$

then

$$\hat{T}_m R_m - R_m\hat{T}_m = \hat{\beta}_{m+1}\hat{V}_m^T\hat{v}_{m+1}e_m^T + H_m, \tag{3.11}$$

where H_m is upper triangular with elements η such that

$$\begin{aligned} |\eta_{1,1}| &\leq \varepsilon_0\sigma, \ \textit{and, for } i \in \{2,\ldots,m\}, \\ |\eta_{i,i}| &\leq 2\varepsilon_0\sigma, \\ |\eta_{i-1,i}| &\leq (\varepsilon_0 + 2\varepsilon_1)\sigma, \ \textit{and} \\ |\eta_{\ell,i}| &\leq 2\varepsilon_1\sigma, \ \textit{for } \ell \in \{1,\ldots,i{-}2\}. \end{aligned} \tag{3.12}$$

This theorem has the same structure as that for fixed precision s-step Lanczos appearing in [1], but with the notable exception that the term ε_0 now contains only a factor of $\bar{\Gamma}_k$ rather than $\bar{\Gamma}_k^2$. As Γ_k can potentially grow very quickly with s, this is a significant improvement, and indicates that, among other things, that the Lanczos basis vectors will maintain significantly better orthogonality and normality due to the selective use of higher precision. We note that this theorem also has the same structure as the theorem given by Paige [6] for classical Lanczos.

The bounds in Theorem 3.2 give insight into how orthogonality is lost in the finite precision s-step Lanczos algorithm. Equation (3.7) bounds the error in the columns of the resulting perturbed Lanczos recurrence. How far the Lanczos vectors can deviate from unit 2-norm is given in (3.9), and (3.8) bounds how far adjacent vectors are from being orthogonal. The bound in (3.10) describes how close the columns of $A\hat{V}_m$ and $\hat{T}_m$ are in size. Finally, (3.11) can be thought of as a recurrence for the loss of orthogonality between Lanczos vectors, and shows how errors propagate through the iterations. Note that $\||\mathcal{B}_k|\|_2$ should be $\lesssim \||A|\|_2$ for many practical basis choices. We also note that the assumption (3.6) is needed for the analysis, but is likely to be overly strict in practical scenarios.

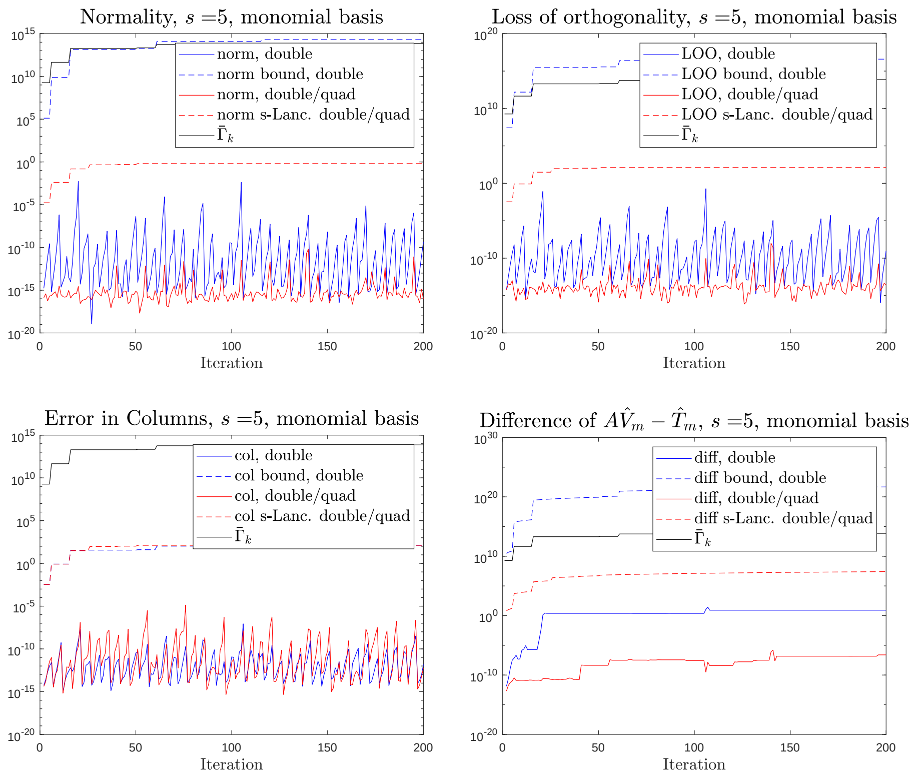

Figure 1: Comparison of fixed and mixed precision s-step Lanczos on the Strakos matrix problem with $n = 100$, $\lambda_1 = 10^2$, $\lambda_n = 10^{-3}$, $\rho = 0.65$, $s = 5$ with monomial basis.

3.3 Numerical experiments

We perform a few experiments in MATLAB (R2020a) in order to demonstrate the validity of the theorem and enable comparison between the fixed finite precision s-step Lanczos algorithm in double precision and the variant that uses a mixed precision approach, with quad precision used for the Gram matrix computations as described and double precision otherwise. For double precision, we use the built-in MATLAB datatype and for quadruple precision we use the Advanpix library. In all tests we use starting vector of norm 1 with equal components. For each test problem, we show the both the computed values and the bounds on the values given in (3.7), (3.8), (3.9), and (3.10). The computed values and bounds for the fixed precision s-step Lanczos algorithm are shown in blue, and the computed values and bounds for the mixed precision s-step Lanczos algorithm are shown in red. We additionally plot the quantity $\bar{\Gamma}_k$, shown in black.

We first test a problem commonly used in studies of the behavior of CG, the diagonal matrix with entries

$$\lambda_i = \lambda_1 + \left(\frac{i-1}{n-1}\right)(\lambda_n - \lambda_1)\rho^{n-i}, \quad i = 1, \ldots, n, \tag{3.13}$$

with parameters $n = 100$, $\lambda_1 = 10^{-3}$, $\lambda_n = 10^2$, and $\rho = 0.65$. The bounds and computed quantities for both fixed precision and mixed precision s-step Lanczos using $s = 5$ with a monomial basis are shown in Figure 1.

A few things are clear from these plots. First, it is clear that the loss of orthogonality, the loss of normality, and the bound (3.10) are much improved by the use of mixed precision. It is also clear that the bounds for both the fixed precision and mixed precision approaches can be large overestimates of the actual quantities. We have tried to stress this in the analysis. The bounds themselves should not be taken as estimates of the errors; rather it is the structure of the accumulation and amplification of errors that tells us something meaningful about the numerical behavior. In this case, it is the fact that using mixed precision we have linear rather than quadratic dependence on the quantity $\bar{\Gamma}_k$, and this is reflected in the computed quantities. We note that the computed quantities in (3.7) are actually sometimes larger for the mixed precision approach than for the fixed precision approach, however, this difference is less significant.

We next perform tests on the matrix `nos4` from the SuiteSparse collection [2]. Here we use $s = 5$ with a monomial basis in Figure 2, $s = 10$ with a monomial basis in Figure 3, and $s = 10$ with a Chebyshev basis in Figure 4. Similar conclusions can be drawn here. It is clear that there is benefit to the mixed precision approach, especially as the quantity $\bar{\Gamma}_k$ grows large. In the $s = 10$ case with the monomial basis, orthogonality and normality are nearly lost for the fixed precision approach, whereas the mixed precision approach dampens

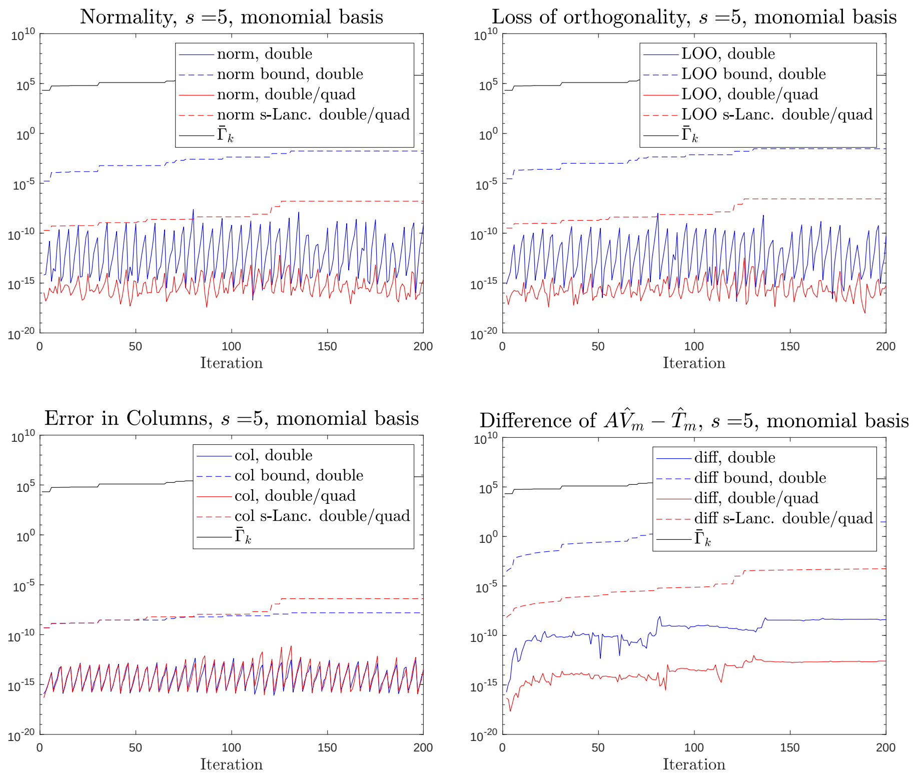

Figure 2: Comparison of fixed and mixed precision s-step Lanczos on the `nos4` problem from SuiteSparse, $s = 5$ with monomial basis.

this effect significantly. As can be seen in Figure 4, using a more well-conditioned basis such as the Chebyshev basis results in lower $\bar{\Gamma}_k$ and thus improved behavior in both cases, although the benefit of mixed precision is still present.

4 Conclusions and Further Outlook

In summary, our analysis shows that there is potential for the use of mixed precision within HSCG and also within s-step variants of Krylov subspace methods. Our results show that inner products within HSCG can be computed at half of the working precision without significantly affecting the attainable accuracy. However, in order to be beneficial, we need a scenario in which the selective use of lower precision is both beneficial to performance and does not significantly affect the rate of convergence so as to negate this performance benefit. The study of convergence behavior is thus crucial; we need to identify properties of the matrix and right-hand side that indicate that the convergence will not deteriorate significantly when low precision is used. A thorough study of the potential performance of low-precision variants will need to involve collaboration with others involved in the xSDK project; we are seeking such collaborations.

We have also proved a result on the loss of orthogonality within a mixed precision s-step Lanczos algorithm. The results show that if the Gram matrix is computed and applied to a vector in double the working precision, the loss of orthogonality (as well as other error terms) depend only linearly on the basis condition numbers instead of quadratically. This can be a significant improvement, as indicated by the included numerical experiments. Because the as in the classical case, the s-step Lanczos process is underlying the s-step CG method, our results suggest that the same mixed precision strategy used here (i.e., using extra precision in the computation and application of the Gram matrix) will improve convergence within the s-step CG method.

We note because we have formulated the mixed precision results in the same format as results previously proved for the fixed precision s-step Lanczos method, all of the further results in [1] (analogous to those proved by Paige for the classical Lanczos method in [7]) also apply to the mixed precision case with the substitution of the new values of ε_0 and ε_1. This paves the way for subsequent study of the convergence and "backward-like stability" of a mixed precision s-step CG algorithm via extension of the work of Greenbaum [4].

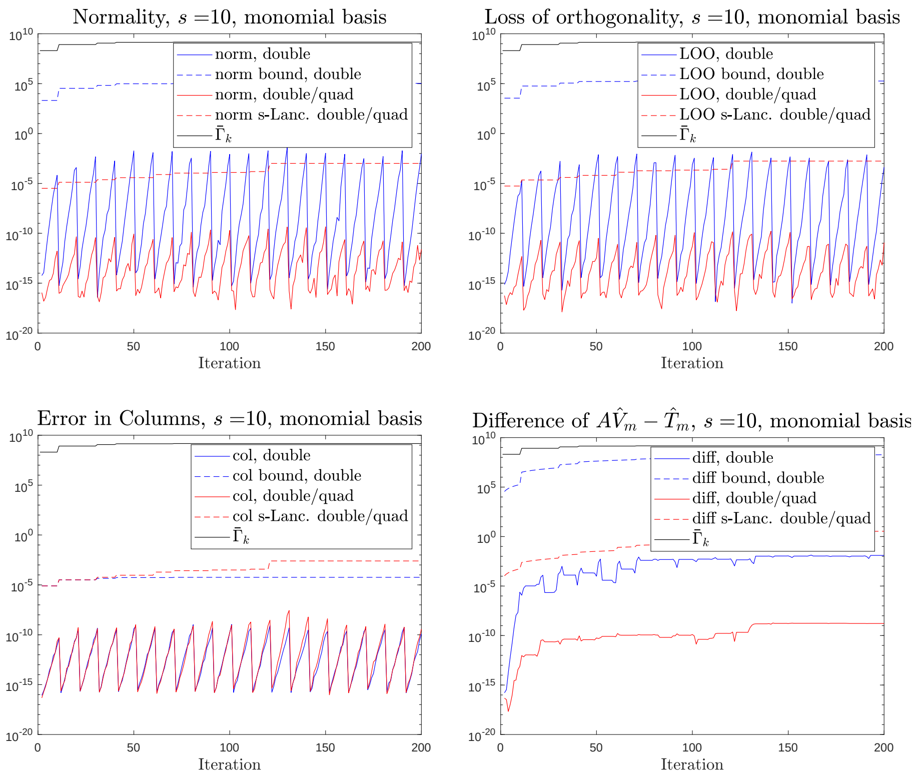

Figure 3: Comparison of fixed and mixed precision s-step Lanczos on the `nos4` problem from SuiteSparse, $s = 10$ with monomial basis.

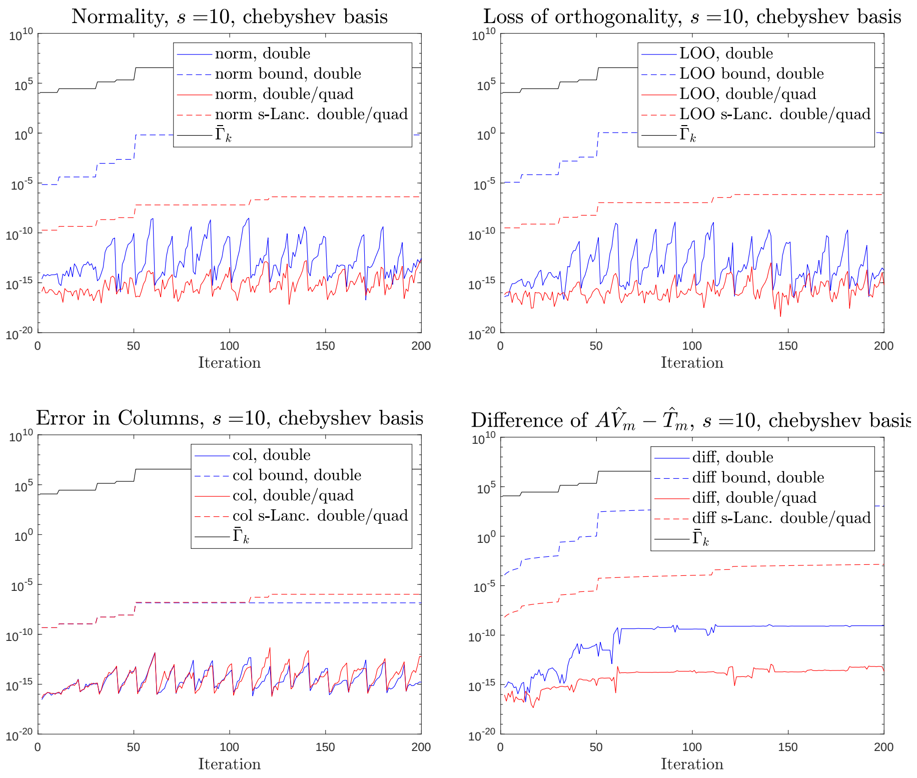

Figure 4: Comparison of fixed and mixed precision s-step Lanczos on the `nos4` problem from SuiteSparse, $s = 10$ with Chebyshev basis.

References

[1] E. Carson and J. W. Demmel. Accuracy of the s-step Lanczos method for the symmetric eigenproblem in finite precision. *SIAM J. Matrix Anal. Appl.*, 36(2):793–819, 2015.

[2] Timothy A Davis and Yifan Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.

[3] G. Golub and C. Van Loan. *Matrix computations.* Johns Hopkins University Press, Baltimore, MD, 3 edition, 1996.

[4] A. Greenbaum. Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences. *Lin. Alg. Appl.*, 113:7–63, 1989.

[5] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards*, 49:409–436, 1952.

[6] C. Paige. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *IMA J. Appl. Math.*, 18(3):341–349, 1976.

[7] C. C. Paige. Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Lin. Alg. Appl.*, 34:235–258, 1980.