Project 1 on *Orthogonalization techniques for a set of vectors*
HPC for numerical methods and data analysis

1 Goal of the project

The goal of this project is to study three different algorithms for the orthogonalization of a set of vectors. Some of the algorithms for orthogonalizing a set of vectors, as CGS, MGS, CholeskyQR, TSQR, are presented during the Lecture from October 3, 2023. Such an algorithm will be needed in the second project where you will need to orthogonalize a set of vectors in parallel. Hence this project should allow you to identify an orthogonalization algorithm that is numerically stable and scales well in parallel.

2 Grading

The grade of the project counts for 1/3 of the final grade of the course. Each student works individually on the project and must submit a report.

3 Deadline

Deadline for submitting the report: October 31st 2023
Strict deadline, no extension provided.

4 Orthogonalization algorithms

Consider a matrix $W \in \mathbb{R}^{m\times n}$, where $m \gg n$. The goal is to compute the thin QR factorization of W, that is $W = QR$, where $Q \in \mathbb{R}^{m\times n}$ is orthogonal and $R \in \mathbb{R}^{n\times n}$ is upper triangular. The matrix W is available at once. In this project, you should choose three different algorithms that allow to compute the QR factorization of W and thus orthogonalize n vectors stored as columns of W. Typically the number of vectors is in between 50 and a few hundreds, while m can be much larger.

5 Content of the report (10 pages maximum)

The report should contain the following elements, that will guide the approach to use in the project.

1. A short description of the three algorithms chosen for computing the QR factorization.
2. A presentation of their parallelization. For the parallelization, the matrix W should be distributed among processors by using a block row distribution. That is for example for 4 processors, the matrix is decomposed into 4 block rows as $W = \begin{pmatrix} W_0 \\ W_1 \\ W_2 \\ W_3 \end{pmatrix}$ and each processor i owns a block W_i with $i = 0 : 3$. The Python and MPI code used for the implementation of the algorithms should be included as well.
3. An investigation of their numerical stability. For a theoretical investigation, one could use the bounds on the loss of orthogonality provided in the Lecture from October 3. For the numerical investigation, you should provide plots measuring the loss of orthogonality as $\|I - Q^TQ\|_2$ and the condition number of the basis Q. When it is possible, these quantities, along with the condition number of the input vectors, should be provided at each iteration of the algorithm when a new vector is orthogonalized. However this will not be possible for

CholeskyQR or TSQR for example. In this case you should present only the results obtained at the end of the algorithm.

This investigation should be done on at least 3 different matrices. One of these matrices will be generated by uniformly discretizing a parametric function, also used in [1], that we denote $C \in \mathbb{R}^{m \times n}$. For all floating (and thus rational) numbers $0 \leq x, \mu \leq 1$, the function is defined as

$$f(x, \mu) = \frac{\sin(10(\mu + x))}{\cos(100(\mu - x)) + 1.1}$$

and the associated matrix is

$$C \in \mathbb{R}^{m \times n}, \quad C_{i,j} = f\left(\frac{i-1}{m-1}, \frac{j-1}{n-1}\right), \; i \in 1, \ldots, m, j \in 1, \ldots, n. \tag{1}$$

You could use for example $m = 50000$ and $n = 600$.

Other matrices could be obtained for example from the SuiteSparse Matrix Collection, `https://sparse.tamu.edu/about` .

4. A presentation of the sequential runtimes obtained by the three algorithms.

5. A discussion of their parallel performance by using one of the following two options:

   (a) a presentation of the parallel runtimes obtained by the three algorithms on a few processors.

   (b) a theoretical investigation of their parallel scalability by estimating the number of floating point operations performed by the algorithms, as well as their communication cost. For this you should use the α, β, γ model seen during the course, with which the time required to exchange a message containing n words is estimated as $\alpha + n\beta$. By using this model you can estimate the parallel scalability of the algorithms by giving some realistic values to α, β, γ.

References

[1] O. Balabanov and L. Grigori. Randomized gram–schmidt process with application to gmres. *SIAM Journal on Scientific Computing*, 44(3):A1450–A1474, 2022.