EPFL

HPC for numerical methods and data analysis

Fall Semester 2023

Prof. Laura Grigori

Assistant: Mariana Martinez

Session 5 – October 17, 2023

TSQR Factorization

For this week you can choose what to do: either these exercises or continue working on your project (or maybe even both).

Exercise 1 CGS and MGS

If we recall what a QR factorization is, given a matrix $W \in \mathbb{R}^{m\times n}$, with $m \geq n$, its QR factorization is

$$W = QR = \begin{bmatrix}\tilde{Q} & \bar{Q}\end{bmatrix}\begin{bmatrix}\tilde{R}\\ 0\end{bmatrix} = \tilde{Q}\tilde{R},$$

where $Q \in \mathbb{R}^{m\times n}$ orthogonal and $R \in \mathbb{R}^{m\times n}$ upper triangular. Note that W can be seen as a map $W : \mathbb{R}^n \to \mathbb{R}^m$. Recall that computing the QR decomposition using CGS is numerically unstable. An alternative algorithm, which is mathematically equivalent is the Modified Gram-Schmidt algorithm (MGS). Define Q_{j-1} as the matrix we get at the j-th step, $Q_{j-1} = \begin{bmatrix}q_1 & q_2 & \dots & q_{j-1}\end{bmatrix}$ and P_j as the projector onto the subspace orthogonal to the column space $\text{col}(Q_{j-1})$. Then we can write this as:

$$P_j = I - Q_{j-1}Q_{j-1}^* = (I - q_{j-1}q_{j-1}^*)\dots(I - q_1q_1^*).$$

a) How many synchronizations do we need for each vector w_j in this case? Why?

b) Why is this more stable than CSG?

c) Make the necessary modifications to your last week's code to implement MGS. Compare both codes by computing $\|I - \tilde{Q}\tilde{Q}^\top\|$.

Exercise 2 TSQR

Remember that communication refers to messages between processors. In the recent years we've seen trends causing floating point to become faster than communication. This is why it's important to minimize communication when dealing with parallelism. The TSQR, "Tall Skinny QR" algorithm is a communication avoiding algorithm for matrices with many more rows than columns. In this

exercise we are going to assume we're using $P = 4$ processors. The computation can be expressed as a product of intermediate orthonormal factors:

$$W = \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} = \begin{bmatrix} Q_{00} & 0 & 0 & 0 \\ 0 & Q_{10} & 0 & 0 \\ 0 & 0 & Q_{20} & 0 \\ 0 & 0 & 0 & Q_{30} \end{bmatrix} \cdot \begin{bmatrix} Q_{01} & 0 \\ 0 & Q_{11} \end{bmatrix} \cdot Q_{02} \cdot R_{02}.$$

a) Let $Q \in \mathbb{C}^{a \times b}$ and $S \in \mathbb{C}^{b \times c}$ be matrices such that their columns form an orthonormal set. Show that the columns of QS also form an orthonormal set.

b) What are the dimensions of Q_{i0}, Q_{i1}, and Q_{02}?

c) Show that the columns of the following matrix are orthonormal. What is the dimension of that matrix?

$$\begin{bmatrix} Q_{00} & 0 & 0 & 0 \\ 0 & Q_{10} & 0 & 0 \\ 0 & 0 & Q_{20} & 0 \\ 0 & 0 & 0 & Q_{30} \end{bmatrix} \cdot \begin{bmatrix} Q_{01} & 0 \\ 0 & Q_{11} \end{bmatrix} \cdot Q_{02}$$

d) Suppose that you are given the implicit representation of $\tilde{Q}$ as a product of orthogonal matrices. This representation is a tree of sets of Householder vectors, $\{Q_{r,k}\}$. Here, r indicates the processor number (both where it is computed and where it is stored), and k indicates the level in the tree. How would you get $\tilde{Q}$ explicitly? We only need the "thin" $\tilde{Q}$, meaning only its first n columns. Note that we can do this by applying the intermediate factors $\{Q_{r,k}\}$ to the first n columns of the $m \times m$ identity matrix. Write a Python script using MPI that does this (assume you are using 4 processors). (*Hint: looking at the graphical representation of parallel TSQR might help.*)