September 2017

GPU-Accelerated and Storage-Efficient Implementation of the QR Decomposition

Peter BENNER [a], Martin KÖHLER [a] and Carolin PENKE [a,1]

[a] *Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, Germany*

Abstract. The LAPACK routines `GEQRT2` and `GEQRT3` can be used to compute the QR decomposition of a matrix of size $m \times n$ as well as the storage-efficient representation of the orthogonal factor $Q = I - VTV^\mathsf{T}$. A GPU-accelerated algorithm is presented that expands a blocked CPU-GPU hybrid QR decomposition to compute the triangular matrix T. The storage-efficient representation is used in particular to access blocks of the matrix Q without having to generate all of it. The algorithm is presented in two variants using one and two GPUs, respectively. To avoid redundant computations or communication between devices, a scheme is developed to additionally compute TV^T during the iteration. As a result the algorithm outperforms the standard LAPACK routine by a factor of 3 for square matrices on a single GPU and by a factor of 5 on two GPUs.

Keywords. QR Decomposition, Numerical Linear Algebra, GPU Acceleration, Storage Efficiency, Block Algorithm, BLAS Level-3

1. Introduction

Along with the LU decomposition, the QR decomposition [1] is one of the basic matrix factorizations used in many numerical linear algebra algorithms. Especially when orthonormal bases come into play, e.g. in least-squares problems, the QR decomposition is commonly involved. During the last decades many different strategies to compute the orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and the upper triangular matrix $R \in \mathbb{R}^{m\times n}$ from a general matrix $A \in \mathbb{R}^{m\times n}$ fulfilling $QR = A$ were developed. The most common ones are based on Householder reflections [2]. These algorithms represent the matrix Q as a product of orthonormal Householder matrices $H_i \in \mathbb{R}^{m\times m}$:

$$Q = H_1 \cdots H_n, \tag{1}$$

where $H_i = I_m - 2\frac{v_i v_i^\mathsf{T}}{v_i^\mathsf{T} v_i}$ and v_i is the i-th Householder vector. Typically, Q is not stored explicitly but implicitly in factored-form representation, where the scaled Householder vectors v_i and the scalar factors $\tau_i = \frac{2}{v_i^\mathsf{T} v_i}$ are stored explicitly [1,3].

[1] Corresponding Author: Carolin Penke, Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtor-Str. 1, 39106 Magdeburg, Germany; E-mail: penke@mpi-magdeburg.mpg.de.