Mixed-precision Block QR Decomposition on GPU

Jaidon Lybbert, Fulin Li

February 26, 2023

Contents

1 QR Decomposition

The QR decomposition of an m-by-n matrix A with $m > n$, is the matrix product $A = QR$, where Q is an m-by-n unitary matrix, and R is upper triangular.

1.1 Matrix Q

The matrix Q is a transformation which preserves inner products of column vectors of R. If the inner product space is real, the matrix Q is equivalently orthogonal. One possibility of such a transformation is a rotation.

Another possibility of such an orthogonal transformation is a reflection. The matrix Q in general is a combination of rotations and reflections.

1.2 Matrix R

The matrix R is upper triangular, a form which has the following useful properties: (I) the determinant is equal to the product of the diagonal elements, (II) the eigenvalues are equal to the diagonal elements, (III) given the linear system $Rx = b$ it is easy to solve for x by back substitution.

2 Computation

In order to compute the decomposition of A, the matrix is iteratively transformed by unitary matrices $\{U_i : 0 < i < k\}$ until the product is upper triangular. This upper triangular matrix is the matrix R in $A = QR$

$$R = U_k U_{k-i} \dots U_1 A. \tag{1}$$

It follows, that the matrix Q is composed of the set of inverse transformations

$$Q = U_1^T U_2^T \dots U_k^T. \tag{2}$$

The key to solving for R is to choose transformations U_i which produce zeros below the diagonal of the matrix product

$$A^{(i)} = U_i \dots U_1 A, \tag{3}$$

and can iteratively be applied to achieve R. Two choices for U_i are Householder reflections, and Givens rotations.

2.1 Transformations

2.1.1 Householder Reflections

The Householder reflection is a unitary transformation represented by a matrix $H \in \mathbb{R}^{N \times N}$ which reflects a vector $\mathbf{u} \in \mathbb{R}^N$ across a hyperplane defined by its unit normal vector $\{\mathbf{w} \in \mathbb{R}^N : \|\mathbf{w}\| = 1\}$. The transformation matrix is given by

$$H = I - 2\mathbf{w}\mathbf{w}^T \tag{4}$$

where $I \in \mathbb{R}^{N \times N}$ is the identity matrix. [1]

To reflect a vector $\mathbf{u} \in \mathbb{R}^N$ such that it points in the direction of a target vector $\mathbf{v} \in \mathbb{R}^N$, the transformation matrix H can be computed by (4), where $\mathbf{w}$ is given by

$$\mathbf{w} = \mathbf{v} - \mathbf{u}, \tag{5}$$

such that,

$$H\mathbf{u} = \|\mathbf{u}\|\hat{\mathbf{v}}, \tag{6}$$

where $\hat{\mathbf{v}}$ is a unit vector in the direction of the target vector $\mathbf{v}$.

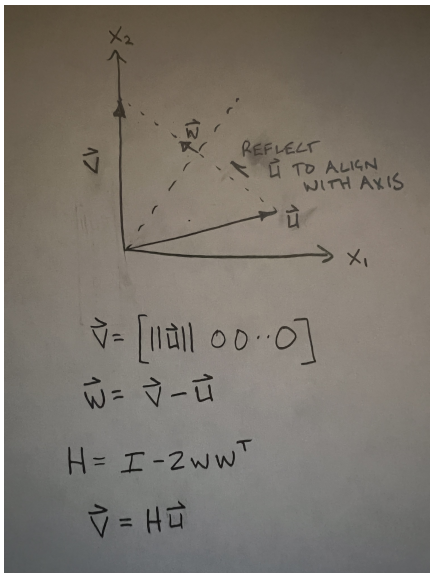

Figure 1: Geometric illustration of the reflection of a vector to an axis. The result of this transformation is that the vector now only has one non-zero component.

2.1.2 Givens Rotations

A Givens rotation is a unitary transformation which rotates a vector x counterclockwise in a chosen plane. For example, possible Givens rotation matrices in $\mathbb{R}^4$ include

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, or \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c & -s \\ 0 & 0 & s & c \end{bmatrix}, \tag{7}$$

where $c = \cos\theta$ and $s = \sin\theta$. Each of these examples have the effect of rotating the vector in different planes.

A Givens rotation can easily be computed to introduce zeros in the matrix P. The scalars c and s can be computed directly from elements in P in order to zero out targeted elements. For example, say we want to zero out element a_{21} in the matrix

$$P = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}. \tag{8}$$

We target the second dimension of the column vector, so we rotate on the plane spanned by the first two dimensions. The Givens rotation to rotate on

this plane is of the form

$$G = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

which will leave the third row of P unmodified. We are aligning the column vector with the axis of the first dimension, making the component of the vector along the second dimension zero. Below is a geometric illustration of the rotation.

The scalars c and s of matrix G are computed directly from the values in matrix P by the equations

$$c = \frac{a_{11}}{r}, \tag{10}$$

$$s = -\frac{a_{21}}{r}, \tag{11}$$

where

$$r = \sqrt{a_{11}^2 + a_{21}^2} \tag{12}$$

The transformation to introduce the zero is then

$$P = GP_{prior} = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{13}$$

$$P = GP_{prior} = \begin{bmatrix} a_{11}/r & a_{21}/r & 0 \\ -a_{21}/r & a_{11}/r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{14}$$

$$P = GP_{prior} = \begin{bmatrix} a_{11}/r & a_{21}/r & 0 \\ -a_{21}/r & a_{11}/r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{15}$$

$$P = \begin{bmatrix} \frac{a_{11}a_{11}+a_{21}a_{21}}{r} & \frac{a_{11}a_{12}+a_{21}a_{22}}{r} & \frac{a_{11}a_{13}+a_{21}a_{23}}{r} \\ \frac{-a_{21}a_{11}+a_{11}a_{21}}{r} & \frac{-a_{21}a_{12}+a_{11}a_{22}}{r} & \frac{-a_{21}a_{13}+a_{11}a23}{r} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{16}$$

$$P = \begin{bmatrix} \frac{a_{11}a_{11}+a_{21}a_{21}}{r} & \frac{a_{11}a_{12}+a_{21}a_{22}}{r} & \frac{a_{11}a_{13}+a_{21}a_{23}}{r} \\ 0 & \frac{-a_{21}a_{12}+a_{11}a_{22}}{r} & \frac{-a_{21}a_{13}+a_{11}a23}{r} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{17}$$

the zero is introduced in the desired location.

2.1.3 WY-representation

For the factored form of $Q \in \mathbb{R}^{M\times M} = Q_1Q_2 \ldots Q_i \ldots Q_n$ where $Q_i = I_m - \beta_i v_i {v_i}^T$ and the factors v_i, b_i are stored as

$$V \in \mathbb{R}^{M\times n} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_n \end{bmatrix} \tag{18}$$

$$B \in \mathbb{R}^{n} = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_n \end{bmatrix} \tag{19}$$

the W and Y factors such that $Q = I_m - WY^T$ can be calculated from V, and B.

Below is an implementation of the algorithm to compute W and Y, from V and B using Numpy in Python.

```
def wy_representation(V, B):
    m = len(V[0])
    r = len(V)

    Y = np.array(V[0]).reshape(m, 1)
    W = np.array(B[0]*V[0]).reshape(m, 1)

    for i in range(1, r):
        z = B[i] * np.dot((np.identity(m) -
                           np.matmul(W, np.transpose(Y))), V[i])
        z = z.reshape(m, 1)
        W = np.concatenate((W, z), axis=1)
        Y = np.concatenate((Y, V[i].reshape(m, 1)), axis=1)

    return W, Y
```

2.2 Algorithms

2.2.1 Householder QR

In order to get the upper triangular matrix $R \in \mathbb{R}^{N\times N}$ given a matrix $A \in \mathbb{R}^{M\times N}$ using householder reflections, we can use (1), where the set of unitary transformations is a set of padded householder matrices $\{U_i \in \mathbb{R}^{M\times M} : 0 < i < N\}$, so that,

$$R = U_{N-1}U_{N-2}\ldots U_1 A. \tag{20}$$

Let

$$A^{(i)} = U_i \ldots U_1 A \tag{21}$$

represent the i-th update of matrix A, so $A^{(N)} = R$ and $A^{(0)} = A$. Then the calculation of U_i depends on the updated matrix $A^{(i-1)}$.

The householder QR algorithm procedure is to sequentially, and iteratively calculate each matrix U_i from $A^{(i-1)}$, then update the matrix $A^{(i)} = U_i A^{(i-1)}$ for the next iteration, until $A^{(N)} = R$ is achieved. At each iteration, U_i is determined such that the i-th column of $A^{(i-1)}$ is transformed so that all elements below the diagonal of the column are zero in the updated matrix $A^{(i)} = U_i A^{(i-1)}$.

For example,

$$A^{(1)} = \begin{bmatrix} \times & \times & \cdots & \times & \times \\ 0 & \times & \cdots & \times & \times \\ 0 & \times & \cdots & \times & \times \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \times & \cdots & \times & \times \end{bmatrix} \tag{22}$$

$$A^{(2)} = \begin{bmatrix} \times & \times & \cdots & \times & \times \\ 0 & \times & \cdots & \times & \times \\ 0 & 0 & \cdots & \times & \times \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \times & \times \end{bmatrix} \tag{23}$$

$$A^{(N-1)} = R = \begin{bmatrix} \times & \times & \cdots & \times & \times \\ 0 & \times & \cdots & \times & \times \\ 0 & 0 & \cdots & \times & \times \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \times \end{bmatrix} \tag{24}$$

Each padded householder transformation matrix $U_i \in \mathbb{R}^{M \times M}$ is created by padding a householder matrix $H_i \in \mathbb{R}^{(M-i)\times(M-i)}$ with ones along the upper diagonal.

$$U_i \in \mathbb{R}^{M \times M} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & H_i \in \mathbb{R}^{(M-i)\times(M-i)} \end{bmatrix} \tag{25}$$

Let $A'^{(i)} \in \mathbb{R}^{(M-i)\times(M-i)}$ be the lower right submatrix of $A^{(i)}$, such that

$$A^{(i)} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & A'^{(i)} \in \mathbb{R}^{(M-i)\times(M-i)} \end{bmatrix} \tag{26}$$

Each householder matrix H_i is calculated by obtaining $\mathbf{w}_i \in \mathbb{R}^{M-i}$ from the submatrix $A'^{(i-1)} \in \mathbb{R}^{(M-i+1)\times(M-i+1)}$, such that

$$A^{(i)} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & A'^{(i)} \in \mathbb{R}^{(M-i)\times(M-i)} \end{bmatrix} \tag{27}$$

and

$$A'^{(i)} = \begin{bmatrix} \mathbf{u_i} & \mathbf{c_2} & \cdots & \mathbf{c_j} & \cdots \mathbf{c}_{M-1} \end{bmatrix}. \tag{28}$$

where $\mathbf{c}_j$ is the j-th column of $A'^{(i)}$, and $\mathbf{u}_i \in \mathbb{R}^{M-i}$ is used as the vector $\mathbf{u}$ in (5) to calculate $\mathbf{w}_i$.

Let

$$\mathbf{v}_i \in \mathbb{R}^{M-i} = \begin{bmatrix} \|\mathbf{u_i}\| \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{29}$$

then $\mathbf{w}_i$ is obtained from $\mathbf{u}_i$ and $\mathbf{v}_i$ according to (5), H_i is determined by (4) from $\mathbf{w}_i$, U_i is obtained by padding H_i, $A^{(i+1)}$ is obtained by (3), and the iterations continue until R is achieved, as in (1).

Q can easily be computed by keeping a running matrix product according to (2) during the iterations of the algorithm.

A simplified Householder QR algorithm is given below, using the Numpy library in Python 3. A more complete implementation, factoring in edge cases is available on Github.

```
# Householder QR Decomposition
def householder_qr(A):
    m,n = A.shape
    Q,H = np.identity(m), np.identity(m)

    for i in range(n):
        u = A[i:,i]
        v = np.zeros_like(u)
        v[0] = np.linalg.norm(u)
        w = (u - v) / np.linalg.norm(u - v)
        H_hat = np.identity(m - i) - 2 * np.outer(w, w)
        H = np.pad(H_hat, (i, 0))
        H[np.diag_indices(i)] = 1
        Q = Q.dot(H)
        A = H.dot(A)
    return Q,A
```

The dependence of U_i on $A^{(i-1)}$ limits the parallelism of the Householder QR algorithm. The matrix update portion $A^{(i)} = U_i A^{(i-1)}$ can be computed by parallel matrix multiply algorithms, however these operations are interspersed with the computation of the padded householder matrix U_i, which is highly sequential. If the parallel portions of this algorithm are implemented on a GPU, and the sequential portions on the host CPU, memory bandwidth and latency become a significant speed and efficiency bottleneck, as the data is passed back and forth between CPU memory and GPU memory.

2.2.2 Block QR

The Block QR algorithm reduces the memory workload by combining multiple householder transformations into a single matrix via the WY-representation of matrix products, before doing the matrix update.

Returning to equation (1), the Block QR algorithm splits the matrix $A \in \mathbb{R}^{M \times N}$ into $b = ceil(\frac{N}{n_b})$ panels $\{P_j \in \mathbb{R}^{M \times n_b} : 0 < j <= b\}$ of width n_b.

$$A = \begin{bmatrix} P_1 & P_2 & \cdots & P_b \end{bmatrix} \tag{30}$$

For each panel, n_b householder vectors $\{\mathbf{w}_k \in \mathbb{R}^M : 0 < k <= n_b\}$ are determined to form a transformation $U_j \in \mathbb{R}^{M \times M} = I - W_j Y_j^T$ such that the set $\{U_j : 0 < j <= b\}$ satisfies (1).

W_j and Y_j are computed using the Householder factors $\mathbf{w}_k$ and β in the general householder equation $H = I - \beta \mathbf{w}\mathbf{w}^T$, where in our case $\beta = 2$ as in (4).

$$W_j, Y_j = wy\_representation(V_j, B_j) \tag{31}$$

where

$$V_j \in \mathbb{R}^{M \times n_b} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_{n_b} \end{bmatrix} \tag{32}$$

and

$$B_j \in \mathbb{R}^{n_b} = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_{n_b} \end{bmatrix} \tag{33}$$

At each iteration j of the block QR algorithm, U_j is computed by the W-Y representation, then the sub-matrix $A'^{(j)} \in \mathbb{R}^{(m-(j*n_b)) \times (n-(j*n_b))}$ is updated by $A'^{(j)} = U_j A'^{(j-1)}$.

When $j = b$ then $(j * n_b) = N$, the width of sub-matrix $A'^{(j)}$ is zero, the matrix $A^{(j)} = A^{(n_b)} = R$, and the decomposition is complete.

2.2.3 Recursive Block QR

2.2.4 Tiled QR

References

[1] Bhaskar Dasgupta. *Applied Mathematical Methods.* Pearson, 1986.

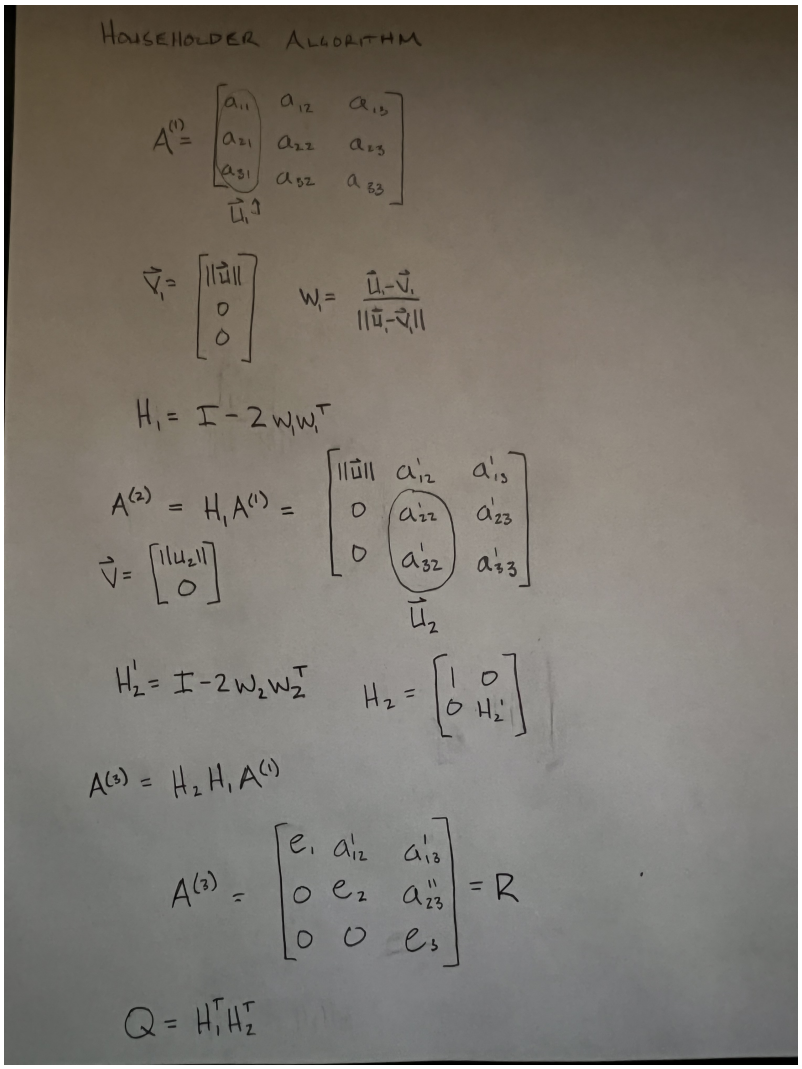

Figure 2: QR factorization algorithm with Householder reflections

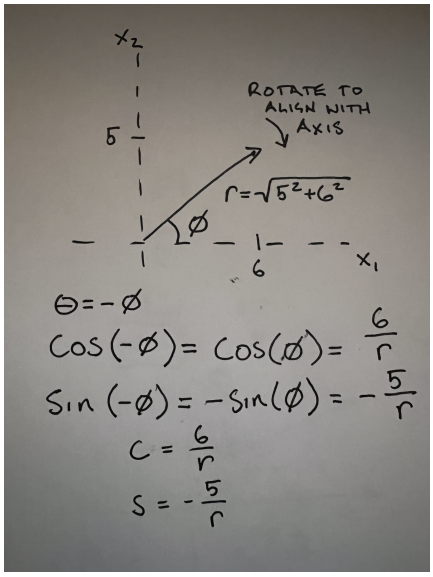

Figure 3: Geometric illustration of the rotation of a vector in $\mathbb{R}^3$ about the axis of basis vector $x3$ to align with the basis vector $x1$. The result of this transformation is that the component of the transformed vector in the direction of the basis vector $x2$ is zero, corresponding to a zero introduced in the transformed matrix.