5L-05

Tensorコアを用いたBatched QR分解

大友広幸† 横田理央‡

†東京工業大学情報理工学院 ‡東京工業大学学術国際情報センター

1 はじめに

QR分解は固有値問題や最小二乗問題の解法として広く用いられている．QR分解の手法の1つにHouseholder変換[1]を用いた手法があり，行列積計算が使われる．本研究ではNVIDIA Voltaアーキテクチャ[1]などに搭載されている混合精度行列計算回路であるTensorコア[2]を用いて複数のQR分解を行うBatched QR分解をHouseholder変換による方法で実装し，その計算速度と精度について調査を行った．

2 背景

2.1 QR分解

QR分解は行列 $\mathbf{M} \in \mathbb{R}^{m\times n}$ を直交行列 $\mathbf{Q} \in \mathbb{R}^{m\times m}$ と上三角行列 $\mathbf{R} \in \mathbb{R}^{m\times n}$ により

$$\mathbf{M} = \mathbf{QR} \tag{1}$$

に分解する．QR分解により $\mathbf{Q}, \mathbf{R}$ は一意に定まる．

2.2 Householder変換を用いたQR分解

Householder変換では単位行列 $\mathbf{I} \in \mathbb{R}^{m\times m}$，ベクトル $\mathbf{u} \in \mathbb{R}^m$ を用いて

$$\mathbf{H} = \mathbf{I} - 2\frac{\mathbf{u}\mathbf{u}^T}{|\mathbf{u}|^2} \tag{2}$$

で得られるHouseholder行列 $\mathbf{H}$ により原点を通る $\mathbf{u}$ に直交する直線での鏡映を得る．$i \in \mathbb{N}$ に対し任意のベクトル $\mathbf{x} \in \mathbb{R}^i$ をベクトル $\mathbf{y} = (\pm|\mathbf{x}|\ 0\ 0\cdots0) \in \mathbb{R}^i$ に映すHouseholder行列 $\mathbf{H}$ は

$$\mathbf{u} = \mathbf{x} \mp \mathbf{y} \tag{3}$$

とすることで得る(複号同順)．これにより鏡映

$$\mathbf{y} = \mathbf{Hx} \tag{4}$$

が成り立つ．$\mathbf{R}$ が上三角行列となるようにHouseholder変換を行うことで $\mathbf{Q}, \mathbf{R}$ を得る．アルゴリズムをAlgorithm 1に示す．

2.3 Tensorコア

Tensor コアは NVIDIA Volta アーキテクチャより利用可能な混合精度行列計算回路である．CUDA[3]ではWMMA(Warp Matrix Multiply Accumulate) APIを通して利用することができる．WMMA APIではTensorコアを用いて $m, n, k \in \mathbb{N}, \mathbf{A} \in \mathbb{R}^{m\times k}, \mathbf{B} \in \mathbb{R}^{k\times n}, \mathbf{C}, \mathbf{D} \in \mathbb{R}^{m\times n}$ に対し積和計算

$$\mathbf{D} \Leftarrow \mathbf{A} \times \mathbf{B} + \mathbf{C} \tag{5}$$

を行う．$\mathbf{A}, \mathbf{B}$ は半精度[4]，$\mathbf{C}, \mathbf{D}$ は半精度または単精度[5]に対応している．また (m, n, k) は $(16, 16, 16), (32, 8, 16), (8, 32, 16)$ の組み合わせのいずれかである必要がある．WMMA API では fragment と呼ばれるレジスタにメモリから行列をコピーし，1 Warp (32 Threads) が協調して式 (5) を行う．

Algorithm 1 Householder変換を用いたQR分解

Require: $m, n \in \mathbb{N}, \mathbf{M} \in \mathbb{R}^{m\times n}$
Ensure: $\mathbf{Q} \in \mathbb{R}^{m\times m}, \mathbf{R} \in \mathbb{R}^{m\times n}$
1: $\mathbf{Q}' \Leftarrow \mathbf{I}$
2: $\mathbf{R} \Leftarrow \mathbf{M}$
3: **for** $i \Leftarrow 0$ to $(\min(n, m) - 1)$ **do**
4: $\quad \mathbf{u} \Leftarrow [0\ \cdots\ 0\ \mathbf{R}_{i,i} \cdots \mathbf{R}_{m-1,i}]^T$
5: $\quad \mathbf{u}_i \Leftarrow \mathbf{u}_i \pm |\mathbf{u}|$
6: $\quad \mathbf{H} \Leftarrow \mathbf{I} - 2\frac{\mathbf{u}\mathbf{u}^T}{|\mathbf{u}|^2}$ // Householder行列
7: $\quad \mathbf{R} \Leftarrow \mathbf{HR}$
8: $\quad \mathbf{Q}' \Leftarrow \mathbf{HQ}'$
9: **end for**
10: $\mathbf{Q} \Leftarrow \mathbf{Q}'^T$

2.3.1 Tensorコアの特長

Tensorコアには次の特長がある．

- ハードウェア的に行列積が実装されているため高速である．
- 入力行列 $\mathbf{A}, \mathbf{B}$ は半精度であるが，内部の足し込み処理を単精度で行うため精度の劣化が抑えられる．

3 実装

行と列の大きさが16以下の行列に対しTensorコアを用いてQR分解を行う関数を実装した．Algorithm 1の6，7行目の行列積にTensorコアを用いた．比較のため入出力型，ノルム計算型，Tensorコアを用いたか否かで表1の6通りの実装を行った．Tensorコアを用いなかった場合，半精度ではSIMD(f16x2)を用いて行列積計算を行った．

計算モード名	入出力型	ノルム計算型	Tensor コア
HH-TC	half	half	YES
HH	half	half	NO
HF-TC	half	float	YES
HF	half	float	NO
FF-TC	float	float	YES
FF	float	float	NO

表 1: 計算モード一覧：型の half は半精度，float は単精度を表す．ノルム計算型は Algorithm1 の 5，6 行目のノルム計算の足し込みの際の型を表す．

4 実験

4.1 Tensorコアを用いたQR分解の精度と計算速度の調査

表1のそれぞれの計算モードで乱数で初期化した 16×16 の行列のQR分解を行い，その精度と計算性能を調査した．精度の調査では入力行列 $\mathbf{M}$ と出力行列 $\mathbf{Q}, \mathbf{R}$ に対しフロベニウスノルム $|\cdot|_F$ を用いて算出した誤差率 R_{qr}

$$R_{qr} = \frac{|\mathbf{M} - \mathbf{QR}|_F}{|\mathbf{M}|_F} \tag{6}$$

を用いて精度を評価した (図1)． WMMA API では入力 $\mathbf{A}, \mathbf{B}$ が半精度である必要があるため，計算モードの入出力型が単精度であっても行列積部分では半精度に変換する必要がある．このためFF-TCではFFに対して誤差率が大きくなっていると考えられる．一方HH-TC，HF-TCではTensorコアを用いることでHH，HFに対し誤差率が小

Batched QR decomposition using TensorCore

Ootomo Hiroyuki†and Yokota Rio‡
†School of Computing, Tokyo Institute of Technology
152-8550, Tokyo, Japan
‡Global Scientific Information and Computing Center, Tokyo Institute of Technology
152-8550, Tokyo, Japan
ootomo.h@rio.gsic.titech.ac.jp, rioyokota@gsic.titech.ac.jp

[1]https://www.nvidia.com/ja-jp/data-center/volta-gpu-architecture
[2]https://www.nvidia.com/ja-jp/data-center/tensorcore
[3]https://docs.nvidia.com/cuda
[4]IEEE 754 binary16
[5]IEEE 754 binary32

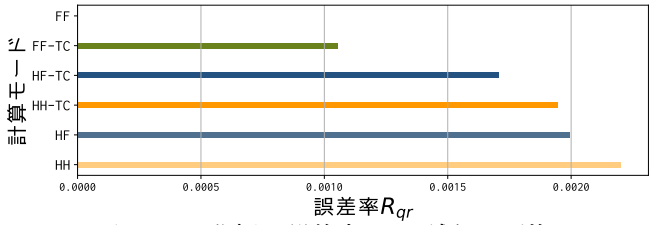

図 1: QR 分解の誤差率 (128 試行の平均)

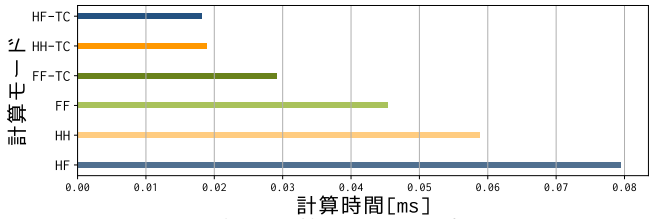

図 2: QR 分解の計算時間 (128 試行の平均)

さく抑えられている．これは Tensor コアが内部の足しこみ処理を単精度で行っていることによると考えられる．また，ノルム計算の精度を半精度から単精度に上げることで誤差率を低く抑えられていることが分かる．

次に QR 分解にかかる時間を調査した (図 2)．計算時間の面では Tensor コアを用いた実装の方が高速に動作していることが分かる．

4.2 Tensor コアを用いた QR 分解による固有値計算の精度

固有値の計算手法の 1 つである QR 法では QR 分解が用いられる．Tensor コアを用いた QR 分解の精度劣化が実アプリケーションにどの程度の影響を及ぼすかを調査するため，表 1 の各計算モードで固有値を計算し，その精度を調査した (図 3)．固有値の正答は Eigen[6]の EigenSolver を用いた．精度の評価は計算した固有値の絶対値を降順に並べた数列 λ_i と Eigen で計算した正答の固有値の絶対値を降順に並べた数列 λ_{ci} を用いて計算した誤差率 R_λ，

$$R_\lambda = \sqrt{\frac{\sum_i (\lambda_i - \lambda_{ci})^2}{\sum_i \lambda_{ci}^2}} \tag{7}$$

を用いた．Tensor コアを用いた場合は用いなかった場合

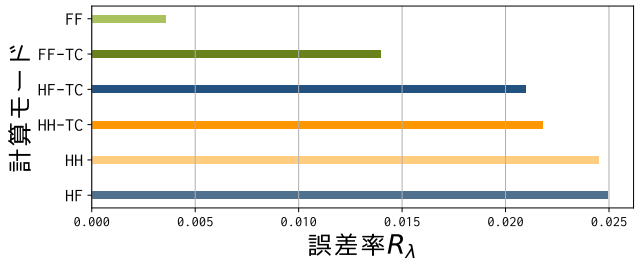

図 3: 固有値の誤差率 (128 試行の平均)

に比べ固有値の精度の劣化が抑えられることが分かる．

4.3 Batched QR 分解の計算時間と計算性能の調査

表 1 の各計算モードの QR 分解を複数の行列に対して並列して行う Batched QR 分解を実装し，複数個 (バッチサイズ) の 16×16 の行列に対し QR 分解を行い，その実行時間及び計算性能を調査した (図 4，5)． バッチサイズが 2^9 程度までは実行時間は増加せず，2^{10} 以上でバッチサイズの増加とともに実行時間が増加していることが分かる．これは 1 度に並列実行可能な処理の数が 2^9 程度であるからであると考えられる．また，バッチサイズの増加とともに計算性能は上昇しており，バッチ処理により計算機を効率よく利用できていると考えられる．

次に cuBLAS[7]の cublasSgeqrfBatched 関数との計算時間の比較を行った (図 6)．cublasSgeqrfBatched 関数は入

[6]http://eigen.tuxfamily.org
[7]https://developer.nvidia.com/cublas

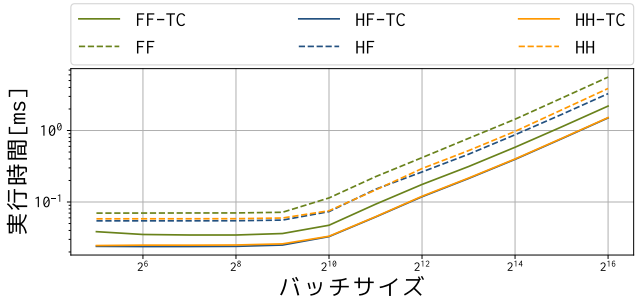

図 4: Batched QR 分解の計算時間 (128 試行の平均)

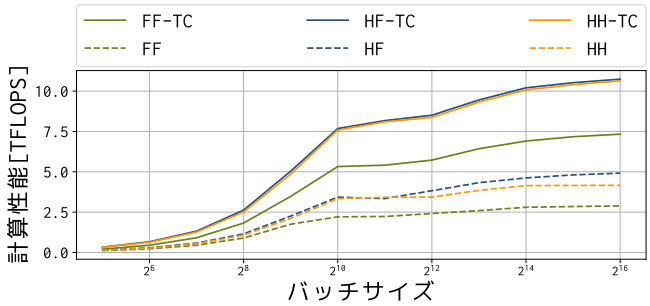

図 5: Batched QR 分解の計算性能 (128 試行の平均)

出力が単精度であるため，表 1 の FF，FF-TC とのみ比較を行った．また，cublasSgeqrfBatched 関数は上三角行列 **R** と Householder 行列 **H** の生成に使用するために必要な値 τ (Algorithm 1 の 6 行目:$\frac{2}{|\mathbf{u}|^2}$ に相当) のみを計算し，直交行列 **Q** の計算を行わない．そこで直交行列 **Q** を生成しない実装 (*-noQ) との比較も行った．Tensor コアを用い

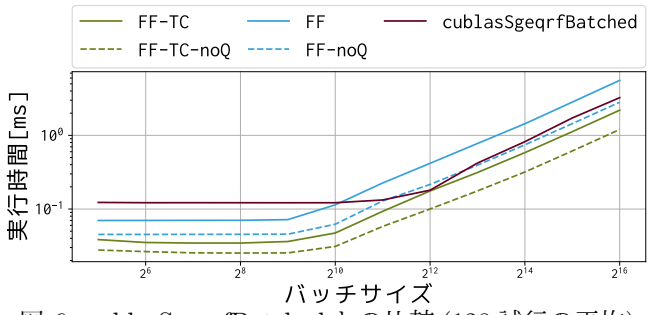

図 6: cublasSgeqrfBatched との比較 (128 試行の平均)

た Batched QR 分解実装の場合，バッチサイズが大きく，また **Q** の計算を行う場合でも cublasSgeqrfBatched 関数より高速であることが分かる．

5 結論

Tensor コアを用いた Batched QR 分解では cuBLAS より高速に複数の行列に対する QR 分解を計算できることを確認した．また，Tensor コアでは精度の劣化を抑えながら半精度行列積を計算するが，QR 分解に用いた場合でも単純な半精度演算に比べ精度の劣化が抑えられることを確認した．

実験環境

Intel Core i9-7940X, NVIDIA Titan V

今後の課題

- 行と列の大きさが 16 以上の行列に対する Batched QR 分解を実装し，その精度と計算時間を調査する．

謝辞

本研究は JST CREST JPMJCR1687 の支援を受けたものである．

参考文献

[1] Alston S. Householder Oak Ridge National Laboratory, Oak Ridge, Tennessee : *Unitary Triangularization of a Nonsymmetric Matrix*, Journal of the ACM (JACM) Volume 5 Issue 4, Oct. 1958