Vol.2022-HPC-183 No.2
2022/3/17

圧縮性乱流直接数値シミュレーションコードのDFTを用いた高速化

竹上 諒[1,a)]　横川 三津夫[2]　櫻井 幹記[3]　石原 卓[4]

概要：乱流の直接数値計算 (DNS) は自然現象や科学技術の諸問題解決のために広く用いられている．周期境界条件の圧縮性乱流の有限差分法に基づく DNS では統計的準定常状態を実現するため速度の低波数成分に外力を与えることが多く，その際には高速フーリエ変換（FFT）がよく利用されている．しかし，3 次元 FFT のプロセス並列実行では分割されたデータの並べ替え操作により実行時間が大きくなることが分かっており，本研究で高速化の対象としている DNS コードの時間発展 1 ステップにかかる実行時間の約 65%が低波数成分導出に用いる FFT の実行時間が占めている．一方，外力の計算に必要な速度の低波数成分は数十点であることから，特定の波数成分を求めることができ，データの並び替えが不要な離散フーリエ変換（DFT）が計算時間の観点で有利である．本研究では，外力注入処理に用いる速度の低波数成分を求める操作を DFT を用いて実装，FFT による実装との実行時間を比較することで，新たな手法の有効性を検証した．その結果，FFT と同程度の計算精度を保ったまま，最大で FFT の 20 倍の速度で，必要な低波数成分を求められることが分かった．また，このプログラムを DNS コードに組み込むことで時間発展 1 ステップを約 2 倍の速度で実行できることが分かった．

1. はじめに

乱流は自然現象や科学技術の諸問題などさまざまな場面に存在するが，流体の支配方程式である Navier-Stokes 方程式は非線形性が強く解析的に解くことは困難であることが知られている．そのため，現在ではコンピュータを用いた数値シミュレーションを行い，乱流のデータを得ることが一般的である．その中でも直接数値シミュレーション（Direct Numerical Simulation; DNS）は支配方程式をモデル化せず直接解き，最小スケールの渦の運動まで解像するため，乱流の性質解明のために広く用いられている．

乱流 DNS を行う場合，レイノルズ数の約 9/4 乗に比例した格子点数が必要となるが，格子点数の増加に伴い必然的にシミュレーションコードの実行時間は増加する．そのため，乱流の DNS で高レイノルズを実現するためにはプログラム全体の高速化が重要となる．

[1] 神戸大学工学部
Faculty of Engineering, Kobe University
[2] 神戸大学大学院システム情報学研究科
Graduate school of System Informatics, Kobe University
[3] 名古屋大学大学院工学研究科
Graduate school of Engineering, Nagoya University
[4] 岡山大学大学院生命科学研究科
Graduate school of Environmental and Life Science, Okayama University
[a)] r-takegami@stu.kobe-u.ac.jp

また，乱流 DNS では，全エネルギーをほぼ一定に保つ統計的準定常状態のシミュレーションが行われることが多い．これは流体速度の低波数成分に外力を注入することによって実現されるため，速度の低波数成分を求める必要がある．周期境界条件を課した 3 次元領域の有限差分法に基づく圧縮性乱流 DNS コードでは，しばしば 3 次元高速フーリエ変換（Fast Fourier Transform; FFT）により低波数成分を求めている．しかし，3 次元 FFT をプロセス並列で実行する場合，3 次元の各軸に FFT を適用させるため，各プロセスが分割して持っているデータの集積と並べ替えの操作が発生し，大規模な並列システムではこれらの操作が 3 次元 FFT の実行時間の多くを占め [9]，DNS の実行時間の約 65%を外力注入処理で用いられる FFT の実行時間が占めることが分かっている．一方，外力を注入するフーリエ成分は数十点であることから，特定のフーリエ係数を求めることができ，かつ並列化時にデータの並び替えが不要な離散フーリエ変換（Discrete Fourier Transform; DFT）が計算時間の観点で有利であると考えられるため，DFT と FFT の性能比を明らかにする必要がある．

本稿では，櫻井らによって開発された圧縮性乱流 DNS コード [8] 内の外力注入処理で速度の低波数成分を導出するために用いられている FFT を DFT に置き換えることを考慮し，必要な流速のフーリエ係数を求める操作を DFT を

用いて実装したプログラムを作成し，スーパーコンピュータ「富岳」を用いて実行時間の計測を行う．また，FFT を用いて実装した場合の実行時間と比較することで，新たな手法の有効性を検証する．

2. 乱流 DNS

2.1 支配方程式

本稿では，等温を仮定した 3 次元圧縮性流体を考える．支配方程式は，連続性の式 (1)，運動量保存式 (2)，状態方程式 (3) から成っている．

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0 \tag{1}$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial (\rho u_i u_j + p\delta_{ij})}{\partial x_j} = \frac{\partial \tau_{ij}}{\partial x_j} + F_i \tag{2}$$

$$p = \rho c^2 \tag{3}$$

ここで，$x_i, \rho, u_i, p, F_i, c$ はそれぞれ直交座標，密度，速度，圧力，外力，音速である，添え字については Einstein の総和規約に従う．計算領域は一辺 2π の立方体で境界条件は周期境界条件を適用する．

また，粘性応力テンソル τ_{ij} は粘性係数 μ を用いて以下のように表される．

$$\tau_{ij} = \mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_m}{\partial x_m}\delta_{ij}) \tag{4}$$

2.2 離散化手法

(1)〜(4) 式を有限差分法で離散化する．支配方程式の粘性項は 8 次精度中心差分法，移流項には波数空間の解像度が高い 8 次精度コンパクト差分 [1] を用いる．これらの式は，1 次元物理量 ($f_i = f(x_i), x_i = 2\pi i/N, N$ は 1 次元の格子点数)，に対して以下の式で表される．

$$f'_i = a\frac{f_{i+1} - f_{i-1}}{h} + b\frac{f_{i+2} - f_{i-2}}{h} + c\frac{f_{i+3} - f_{i-3}}{h} + d\frac{f_{i+4} - f_{i-4}}{h} \tag{5}$$

$$a = \frac{4}{5}, b = -\frac{1}{5}, c = \frac{4}{105}, d = -\frac{1}{280}, \tag{6}$$

$$\alpha f'_{i-1} + f'_i + \alpha f'_{i+1} = a\frac{f_{i+1} - f_{i-1}}{2h} + b\frac{f_{i+2} - f_{i-2}}{4h} + c\frac{f_{i+3} - f_{i-3}}{6h} \tag{7}$$

$$\alpha = \frac{3}{8}, a = \frac{25}{16}, b = \frac{1}{5}, c = -\frac{1}{80} \tag{8}$$

ここで $h = 2\pi/N$ とする．空間離散化によって得られた時間 t に関する乱流場の時間発展は，3 段 3 次 TVD (Total Variation Diminishing) Runge–Kutta 法 [2] を用いる．またプロセス並列化に伴い，3 次元領域を 2 軸分割により並列化している．

2.3 外力を注入するフーリエ係数

準定常状態をとらえるため，運動エネルギーが準定常を保つように波数の小さい（渦のスケールの大きい）領域に外力を加える [3]．低波数成分に注入する外力は以下の式で与える．

$$F_i = c_s\sqrt{\rho}\tilde{w}_{is} + c_d\sqrt{\rho}\tilde{w}_{id} \tag{9}$$

ここで，乱流場の Helmholtz 分解によりソレノイダル成分 (solenoidal) と圧縮成分 (dilatational) に分けることとし，$w_{i\alpha} \equiv (\sqrt{\rho}u_i)_\alpha$，$\alpha = s$ が solenoidal，$\alpha = d$ が dilatation とする．c_s, c_d は以下の式で求める．

$$c_s = \frac{\epsilon_{s_{target}}}{\langle \tilde{w}_{is} w_i \rangle} \tag{10}$$

$$c_d = \frac{\epsilon_{d_{target}} - PD}{\langle \tilde{w}_{id} w_i \rangle} \tag{11}$$

ここで，$\epsilon_{\alpha_{target}}$ はそれぞれソレノイダル成分と圧縮成分の散逸率の目標値であり，$\epsilon = -\langle u_i \partial \tau_{ij}/\partial x_j \rangle$ と任意に与えるソレノイダル成分と圧縮成分の比 ϵ_r を用いて，以下の式を満たすように決定される．

$$\epsilon = \epsilon_{s_{target}} + \epsilon_{d_{target}} \tag{12}$$

$$\frac{\epsilon_{d_{target}}}{\epsilon_{s_{target}}} = \epsilon_r \tag{13}$$

また，$PD = -\langle (\partial p/\partial x_i)\, w_i/\rho^{1/2} \rangle$ であり，$\langle X \rangle$ は X の平均である．ここで波数空間での Helmholtz 分解は全領域のフーリエ係数 $\hat{w}$ を用いて以下のように行う．

$$\hat{w}_{id}(\boldsymbol{k}) = k_i(\boldsymbol{k}\hat{w})/|\boldsymbol{k}|^2 \tag{14}$$

$$\hat{w}_{is}(\boldsymbol{k}) = \hat{w}_i(\boldsymbol{k}) - \hat{w}_{id}(\boldsymbol{k}) \tag{15}$$

そして，低波数領域 ($|\boldsymbol{k}| < K_c$) のみフーリエ変換で実空間へと変換を行う．

$$\tilde{w}_{i\alpha}(\boldsymbol{x}) = \sum_{\boldsymbol{k}} \hat{w}_{i\alpha}(\boldsymbol{k}) e^{-2\pi \mathrm{i} \boldsymbol{k}\cdot\boldsymbol{x}}, |\boldsymbol{k}| < K_c \tag{16}$$

ここで，K_c は低波数領域の閾値であり，また安定的な外力とするため，$\hat{w}_{i\alpha}(\boldsymbol{0}) = 0$ とする．すなわち，式 (16) と $\hat{w}_{i\alpha}(\boldsymbol{0}) = 0$ から，外力注入の計算のために必要なフーリエ係数は $0 < |\boldsymbol{k}| < K_c$ の領域内に存在するもののみでよいことがわかる．

$$\hat{w}_{i\alpha}(i, j, k), 0 < \sqrt{i^2 + j^2 + k^2} < K_c \tag{17}$$

3. 高速フーリエ変換による低波数計算

3.1 3 次元高速フーリエ変換の概要

DNS コードでは，低波数のフーリエ係数を 3 次元 FFT を用いて求めている．3 次元 FFT の並列実行では，1 次元 FFT を行う軸のデータをメモリ内に連続配置とするため，並列に分割したデータを一つのプロセスに集めることが計算効率の点で重要である．入力データが y 軸と z 軸で分割されている 2 軸分割 FFT では，**図 1** のように各プロセス

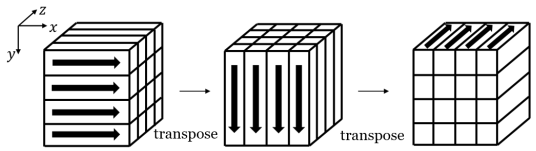

図 1 2 軸分割 3 次元 FFT の計算手順のイメージ

に対して x 方向のみ連続な領域が割り当てられるため，y 方向，z 方向の FFT を行うために，データの転置を行う必要がある．したがって，2 軸分割 3 次元 FFT の計算手順は以下のようになる．

1．x 方向の FFT
2．プロセス内データの並べ替え
3．全対全通信（MPI_alltoall）によるプロセス間通信
4．プロセス内データの並べ替え
5．y 方向の FFT
6．プロセス内データの並べ替え
7．全対全通信（MPI_alltoall）によるプロセス間通信
8．プロセス内データの並べ替え
9．z 方向の FFT

このように，各方向の 1 次元 FFT（1D-FFT）の後に行われる MPI_alltoall によるデータの転送と転送の前後に発生する各プロセス内でのデータの並べ替え操作が，並列化時に新たに生じる処理であり並列化による高速化の妨げとなっている．以降では，MPI_alltoall によるデータの転送を転送，転送前後の各プロセス内でのデータの並べ替え操作を並べ替え，これら二つを合わせて転置と呼ぶ．

3.2 圧縮性乱流 DNS コード内の FFT 実行時間内訳

圧縮性乱流 DNS コードの外力注入処理で行われている FFT の実行時間がプログラム全体の実行時間の中でどの程度の割合を占めているのかを確認する．富岳で計測したコードはコンパイラ Fusitu Fortran Ver 4.7.0(mpifrtpx) を用い，オプションとして-Kfarst，-Kopenmp を指定してコンパイルした．

外力注入部で用いられている FFT は 2 軸分割 3 次元 FFT の一次元方向の FFT に FFT ライブラリ FFTE[6] の関数 fft235 を利用し，並べ替え部をオリジナルに実装したプログラムである．外力注入処理では，n^3 個の 3 次元実空間データの実空間から波数空間への変換が 3 回，波数空間から実空間への変換が 6 回行われ，1 ステップの中で外力注入処理は 3 回行われるため，順方向の変換は計 9 回，逆方向の変換は計 18 回行われる．

データサイズ $n^3 = 256^3$ の圧縮性乱流 DNS コードを富岳上で 64 ノード，256 並列（y=16，z=16）にて 1 ステップ実行し，1 ステップの時間発展の実行時間と外力計算部の FFT の実行時間を MPI_wtime により計測したものを**図 2** に示す．

図 2 から，時間発展 1 ステップの実行時間の中で外力注入部の FFT の計算時間が約 65%を占めていることが確認できた．したがって，外力注入部のフーリエ係数を高速に計算することができれば，プログラム全体の実行時間の大幅な削減が期待できる．

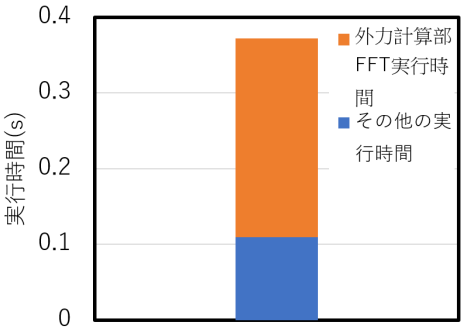

図 2 乱流 DNS 内の外力注入部 FFT の実行時間

4. 3 次元離散フーリエ変換による外力注入処理

4.1 離散フーリエ変換

離散フーリエ変換（Discrete Fourier Transform; DFT）は離散的なデータを波数成分に分解する変換方法である．1 次元の複素離散フーリエ変換は以下の式で定義される．

$$\hat{f}(k) = \frac{1}{n}\sum_{x=0}^{n-1} f(x) w_n^{xk} \quad k = -n/2, \ldots, n/2-1 \qquad (18)$$

(x, k：整数，$f(x)$：実空間データ，$\hat{f}(k)$：フーリエ係数，n：データ長，$w_n = e^{-\frac{2\pi\sqrt{-1}}{n}}$：ひねり係数)

4.2 2 軸分割 3 次元順離散フーリエ変換

3 次元複素離散フーリエ変換をプロセス並列で分割する場合，以下のように書ける．

$$\hat{f}(i,j,k) = \frac{1}{n^3}\sum_{allprocess}\left(\sum_{x\in P_x}\sum_{y\in P_y}\sum_{z\in P_z} f(x,y,z) w_{n_x}^{xi} w_{n_y}^{yj} w_{n_z}^{zk}\right) \qquad (19)$$

ここで $f(x,y,z), (x,y,z = 0,\ldots,n-1)$ は 3 次元実データ，$\hat{f}(i,j,k), (i,j,k = -n/2,\ldots,n/2-1)$ はフーリエ係数，P_x, P_y, P_z は，それぞれ各プロセスが持つ領域内の x, y, z の集合である．このことから，DFT では並列化により領域を分割したとしても，自らのプロセスが持つ範囲のデータとひねり係数を乗算したのち MPI_Allreduce により各プロセス内で計算した値の合計を取る reduction 演算によりフーリエ係数を求めることができる．したがって，FFT の際に発生していたデータの転置操作が不要となる．

入力データの各方向のデータ長が $n_x = n_y = n_z = n$ のとき，DFT のアルゴリズムでは，それぞれの i, j, k に対して，n^3 回の乗算と加算が必要となるため計算量は $\mathrm{O}\left(n^6\right)$ である．しかし，2.4 節，式 (17) より，DNS 内で必要となるフーリエ係数 $\hat{f}(i,j,k)$ は $0 < \sqrt{i^2+j^2+k^2} < K_c$ を満たす格子点のもののみであり，$K_c = 3.0$ の場合，その数は

n に関係なく 92 個である．DFT では，不必要な波数格子点の計算を一切行わずに特定の波数のフーリエ係数を導出することが可能であるため，領域内のフーリエ係数の数を K とした場合その計算量は $\mathrm{O}\left(Kn^3\right)$ である．

2 軸分割 3 次元 DFT の計算手順を以下に示す．

`step.1` プロセス内のデータのみでフーリエ係数の部分和を計算

`step.2` 集合通信（`MPI_Allreduce`）によるプロセス間通信により総和を取る

K_c の値は最大で 3，フーリエ係数の数は 92 個以下であることから，逆離散フーリエ変換の際の計算を通信なしで行えるようにするため，すべてのプロセスがこれらのフーリエ係数を持つよう MPI_Allreduce によるプロセス間通信を行った．

4.3 3 次元逆離散フーリエ変換

低波数領域のフーリエ係数を利用した 3 次元逆 DFT について述べる．まず，K_{in}, K_{out} を以下のように定義する．

$$K_{in} = \{(i,j,k) \mid 0 < \sqrt{i^2+j^2+k^2} < K_c\}$$

$$K_{out} = \{(i,j,k) \mid \sqrt{i^2+j^2+k^2} = 0, \sqrt{i^2+j^2+k^2} \geq K_c\}$$

$\hat{f}(i,j,k)$ は，以下のようにフーリエ係数を低波数領域に限定する．

$$\hat{f}(i,j,k) = \begin{cases} \hat{f}(i,j,k) & ((i,j,k) \in K_{in}) \\ 0 & ((i,j,k) \in K_{out}) \end{cases} \quad (20)$$

このとき，3 次元逆離散フーリエ変換は以下のように表される．

$$\begin{aligned} f(x,y,z) &= \sum_{i=-n/2}^{n/2} \sum_{j=-n/2}^{n/2} \sum_{k=-n/2}^{n/2} \hat{f}(i,j,k) w_{n_x}^{-xi} w_{n_y}^{-yj} w_{n_z}^{-zk} \\ &= \sum_{(i,j,k)\in K_{in}} \hat{f}(i,j,k) w_{n_x}^{-xi} w_{n_y}^{-yj} w_{n_z}^{-zk} \\ &\quad + \sum_{(i,j,k)\in K_{out}} \hat{f}(i,j,k) w_{n_x}^{-xi} w_{n_y}^{-yj} w_{n_z}^{-zk} \end{aligned}$$

ここで式 (20) より，

$$f(x,y,z) = \sum_{(i,j,k)\in K_{in}} \hat{f}(i,j,k) w_{n_x}^{-xi} w_{n_y}^{-yj} w_{n_z}^{-zk} \quad (21)$$

となる．ここで $x,y,z = 0,\ldots,n-1$ である．したがって逆方向の変換も順方向の変換と同様に，領域内の格子点の数 K に対して $\mathrm{O}\left(Kn^3\right)$ となる．

逆方向の 2 軸分割 3 次元 DFT は，順変換の際にそれぞれのプロセスが $0 < \sqrt{i^2+j^2+k^2} < K_c$ 内のすべてのフーリエ係数を持っているので，領域分割された点の $f(x,y,z)$ の値は式 (21) によりすべて自らのプロセス内で並列に計算できる．

表 1 スーパーコンピュータ「富岳」の仕様

Total peak performance		488Flops
Number of Nodes		158,976
node	interconnect	Tofu Interconnect D
	CPU	FUJITSU Processor A64FX
	Performance	3.072TFlops
	Number of cores	48
	Memory	32GiB
	Memory bandwidth	1024GB/s
	L1D/core cache	64KiB, 4way 256GB/s(load), 128GB/s(store)
	L2/CMG cache	8MiB, 16way
	L2/node cache	4 TB/s (load), 2 TB/s (store)
	L2/core cache	128 GB/s (load), 64 GB/s (store)

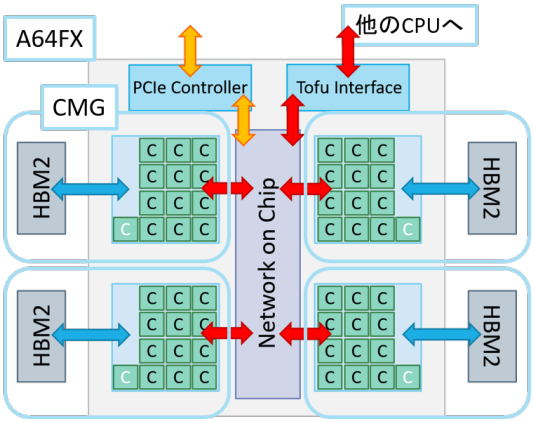

図 3 富岳ノード内のイメージ図

5. 実行環境

富岳の仕様を**表 1** に示す [7]．富岳の 1 ノードにおける内部構成のイメージ図が**図 3** である．富岳の 1 ノードは 4 つの CMG (Core Memory Group) から構成されている．CMG は 12 のコアと 1 つのメモリそして OS の実行などを行うユーザーからは見えないアシスタントコア 1 つからなっており，1 ノードに合計 48 コア 4 メモリ 4 アシスタントコアが存在する．そのため 1 つの CMG に 1 つのプロセスを割り当てることを推奨しており，またスレッド並列が利用できる場合 1 つの CMG 内のある 12 のコアを利用し，1 プロセス当たり 12 スレッド並列で実行することでノード内のコアを有効に利用することができる．そのため今回の計測でも，1 ノード当たり 4 プロセスを割り当てそれぞれを 12 スレッド並列で実行した．

コンパイラは，Fusitu Fortran Ver 4.7.0（mpifrtpx）．コンパイルオプションは，-Kfarst，-Kopenmp とした．

6. DFT による計算手法の性能評価

6.1 計測方法

作成した DFT プログラム及び FFT によるプログラムの実行時間をスーパーコンピュータ富岳上にて計測した．今

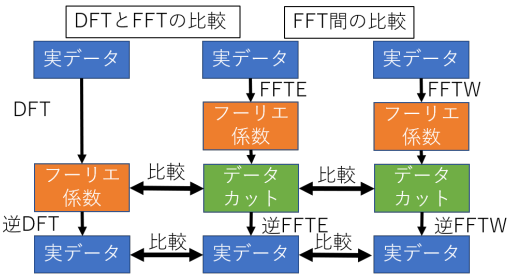

図 4 テストプログラムの流れ

回のプログラムの流れを**図 4** に示す．

データ長 n^3 の実数データをフーリエ変換を用いて $0 < \sqrt{i^2+j^2+k^2} < K_c$ を満たす格子点のフーリエ係数を計算，計算したフーリエ係数を用いて逆フーリエ変換を行い，低周波数成分のみを持つ実データを作成する処理を実行時間の計測範囲とする．DFT を用いた順方向の変換では領域内の係数のみを導出できる．一方，実 FFT では $n^3/2$ 個のすべてのフーリエ係数が求められてしまうため，K_{out} に属するフーリエ係数を 0 にする操作（データカット）が必要である．

6.2 計算精度

今回作成した DFT プログラムが正しく変換を行っているか調べるため，FFTE による計算結果との比較を行った．また，DFT と FFTE 間の出力の差が妥当なものであるのかを判断するため FFTE と FFTW 間での出力の差を計測する．計測に用いるフーリエ変換には，自作した DFT によるプログラム，比較用のデータ作成に用いたフーリエ変換は DNS コード内で実際に使われている FFT ライブラリ FFTE を利用し二軸分割に対応させたものによるプログラム．また，FFT ライブラリ FFTW を用いて二軸並列に対応させたものによるプログラムを用いる．比較は K_{in} に属するフーリエ係数の値，及び実空間に逆変換した値に対して調べた（図 4）．

フーリエ係数の誤差は，FFTE により作成した比較データを用いて，次の式で計算した．

$$err_{Fourier} = \max_{(i,j,k)\in K_{in}} \Big((\hat{f}_{ans_{re}}(i,j,k) - \hat{f}_{re}(i,j,k))^2 + (\hat{f}_{ans_{im}}(i,j,k) - \hat{f}_{im}(i,j,k))^2 \Big)^{\frac{1}{2}}$$

ここで，i,j,k：整数，$\hat{f}_{ans}(K)$：FFTE による比較データ，$\hat{f}(i,j,k)$：DFT，及び FFTW によるフーリエ係数である．また K_{in} 領域内のフーリエ係数を利用し，実データへと逆変換したデータの誤差も同様に比較データとの差の絶対値の中で最大の値を求めた．式は以下の通り．

$$err_{redata} = \max_{(x,y,z)=0,\ldots,n-1} (|f_{ans}(x,y,z) - f(x,y,z)|) \quad (22)$$

表 2 タイプ 0 での出力誤差

	FFTE-DFT		FFTE-FFTW	
n^3	$err_{Fourier}$	err_{redata}	$err_{Fourier}$	err_{redata}
2048^3	4.506E-13	3.865E-12	9.692E-14	2.274E-12
4096^3	5.020E-12	1.910E-11	2.195E-13	3.638E-12
8192^3	3.480E-11	9.822E-11	5.684E-13	9.094E-12

表 3 タイプ 1 での出力誤差

	FFTE-DFT		FFTE-FFTW	
n^3	$err_{Fourier}$	err_{redata}	$err_{Fourier}$	err_{redata}
2048^3	6.206E-16	7.105E-15	3.554E-16	3.997E-15
4096^3	1.024E-15	7.550E-15	4.718E-16	3.553E-15
8192^3	5.177E-15	2.376E-14	6.335E-16	4.441E-15

ここで，x,y,z：整数，$f_{ans}(x,y,z)$：FFTE による比較データ，$f(x,y,z)$：DFT，及び FFTW による逆変換後の実データである．

入力データとして 2 種類のデータを用いた，タイプ 0 として $f(x,y,z) = x+y+z$，タイプ 1 として，$R_{x1}(l)\sin(lx) + R_{x2}(l)\cos(lx)$，$R_{y1}(l)\sin(ly) + R_{y2}(l)\cos(ly)$，$R_{z1}(l)\sin(lz) + R_{z2}(l)\cos(lz)$ を足し合わせたものを用いる．ここで，$R_{x1}(l), R_{x2}(l) (l = 0,\ldots,n/2-1)$ などは，$(0,1)$ の一様乱数を用いて設定した．それぞれのデータでの誤差を**表 2**，**表 3** に示す．

結果から，入力にタイプ 1 よりタイプ 0 を用いたほうが，比較データとの差が大きくなることが分かる．これはフーリエ変換時の変換誤差が各アルゴリズム間の出力の差として出力されたと考えられる．FFTW と DFT の結果を見てみると，比較データとの差はどちらも大きな違いがなく，DFT を用いた順変換，逆変換により出力されるデータは FFT を用いたものと同程度の精度を持っていることが確認できた．

6.3 実行時間

次に，DFT，FFTW，FFTE それぞれの計算手順を**図 5** に示す．図中の通信，転置の操作がプロセス並列化に伴い追加された実行時間となり，その他の操作が並列化しない場合のフーリエ変換に掛かる実行時間である．ただし FFT の計測の際は分割されているそれぞれの操作の実行時間の和を取るものとする．また DNS コードに合わせ FFT の計算時間にはデータカットの時間を含めなかった．

1024 ノード，4096 プロセス (y=64, z=64)，12 スレッドでのハイブリッド並列により，データ長 n^3 を変えながら，実行時間を計測したものを，**図 6**，**図 7** に示す．DFT が求める係数の閾値は $K_c = 3$ とした．図中 calc がフーリエ変換の計算時間，trans が並べ替えの操作，comm が mpi 通信による転送，通信の操作の実行時間である．この結果から 2 軸並列の場合 DFT は FFT に比べ高速に K_{in} 内のフーリエ係数を求められることが分かる．DFT，FFTE そ

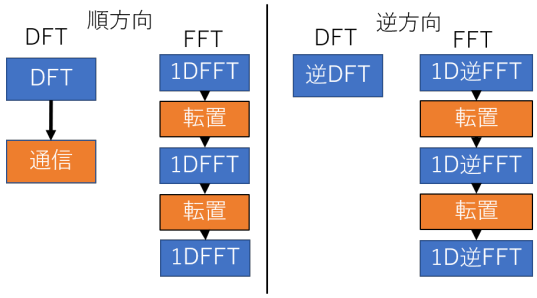

図 5 DFT，FFT の計算手順

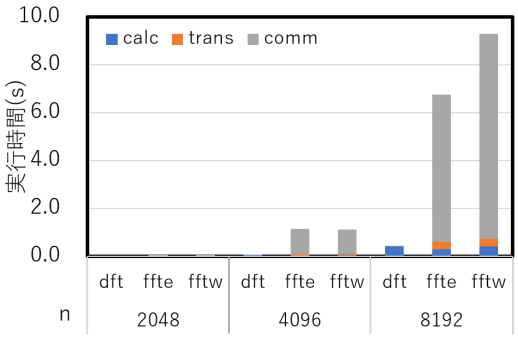

図 6 順方向フーリエ変換の実行時間内訳

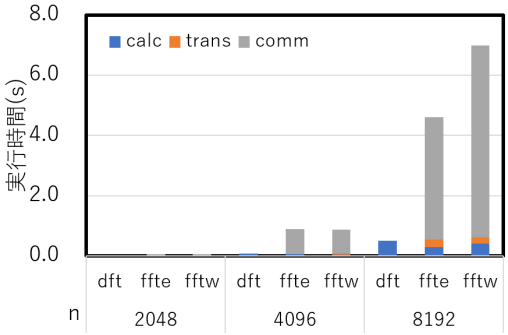

図 7 逆方向フーリエ変換の実行時間内訳

れぞれの順方向のフーリエ変換と逆方向のフーリエ変換の実行時間を加算し，計測範囲全体の実行時間を比較した．1024 ノード，4096 プロセス (y=64, z=64)，12 スレッド，$n^3 = 1024^3$ の実行結果では全体で約 6.6 倍，順方向のみでは約 4.5 倍，逆方向のみでは約 3 倍となった．また，$n^3 = 8192^3$ のとき実行時間の高速化率が最大となり，DFT を用いることで一連の操作を FFTE を用いた場合よりも約 20 倍高速に計算できる．順方向のみでは約 24 倍，逆方向では 19 倍の高速化が見られた．

FFT の計測では，転送時間のばらつきが大きく同条件の計測であっても FFTE，FFTW の実行時間が変動した．これについては富岳内でのプロセス配置が関係していると考えているが，DFT の計算時間が最も短いことが分かった． FFT に注目すると，転置時間が全体の実行時間の中で大きな割合を占めていることが確認できる．計算時間のみに注目すると，FFTE が最も早く，DFT が最も遅い結果となった．しかし DFT は，2 軸並列の 3 次元 FFT を行う際に問題であった転置操作の実行時間が無いため，全体の実行時間では最も早い結果となった．

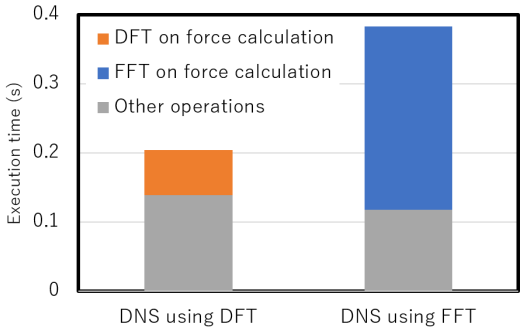

図 8 乱流 DNS 内の外力注入部フーリエ変換の実行時間

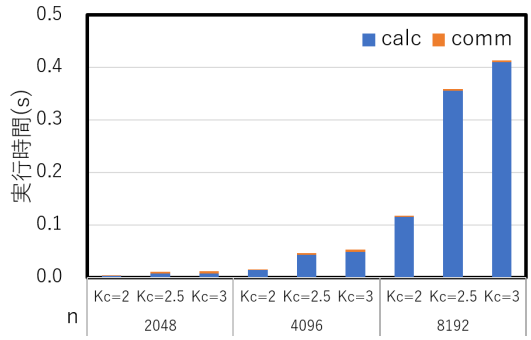

図 9 $K_c = 2, 2.5, 3$ での順方向 DFT の実行時間

最後に今回の DFT を乱流 DNS コードに組み込み $n^3 = 256^3$，64 ノード，256 並列 (y=64, z=64)，$K_c = 3$ にて実行時間の計測を行った結果を**図 8** に示す．図 8 から，DFT を用いて低波数成分を高速に求めることができたため，時間発展 1 ステップの実行時間を約 46%削減することができた．また外力注入部のフーリエ変換だけの実行時間では約 75%削減することができた．

6.4 K_c 値との関係

K_c の値と実行時間の関係について調べた．DNS コードでは，$K_c = 3$ が用いられている．しかし，流速の低波数領域とそれ以外の領域を分ける閾値であるこの値は，2 から 3 の間で複数の候補が考えられている．ここでは K_c の値を変えながら DFT による低波数成分導出の実行時間を計測することにより，K_c 値を変更した際の計算時間の変化を調べた．ここで，K_{in} に含まれる波数格子点数を K とすると，$K_c = 2, 2.5, 3$ に対し，それぞれ $26, 80, 92$ である．

1024 ノード，4096 プロセス (y=64, z=64)，12 スレッドでのハイブリッド並列により，$K_c = 2, 2.5, 3$ のそれぞれの場合にデータサイズ n^3 を変えながら DFT プログラムの実行時間を計測したものを以下の**図 9**，**図 10** に示す．

それぞれの n に対し，K_c の値が小さくなるにつれ実行時間も減っていることが分かる，これは K_c を小さくすることで低波数領域が狭まり計算するフーリエ係数の数が減

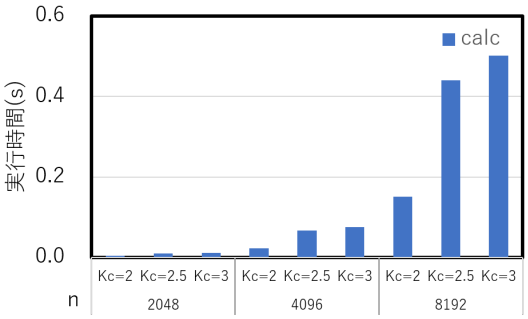

図 10 $K_c = 2, 2.5, 3$ での逆方向 DFT の実行時間

るためである．それぞれのフーリエ変換の計算時間を対応する K の値で割るとほぼ一定の値が得られたため，実行時間は $\mathrm{O}\left(Kn^3\right)$ に従い増加することが分かった．

FFT の実行時間は K_c の値に影響を受けないため，実行時間が最も短くなる $K_c = 2$ の場合，さらに実行時間の高速化率を上げることができる．1024 ノード，4096 プロセス (y=64, z=64)，12 スレッド，$n^3 = 8192^3$ のとき高速化率が最大で，計測範囲全体で FFT の約 67 倍の高速化となり，順方向のみでは約 68 倍，逆方向のみでは約 66 倍の高速化となった．

7. おわりに

本研究では，これまで 2 軸並列 3 次元 FFT を用いて実装していた 3 次元圧縮性乱流 DNS コードの外力注入処理で必要となるフーリエ係数を導出する操作を，DFT を用いて実装したプログラムを作成，スーパーコンピュータ「富岳」においてその実行時間の計測を行い，同様の処理を FFT を用いて行った際の実行時間との比較を行った．

FFT ライブラリ FFTW，FFTE との計測結果の比較により，DFT を用いることで，FFT を用いた場合と同程度の計算精度で FFT よりも高速に必要なフーリエ係数を求められることが分かった．具体的には，プロセス数 4096 (y=64,z=64)，スレッド数 12 のハイブリッド並列，データ長 $n^3 = 8192^3$ での実行で最も実行時間が短縮でき，順方向の変換では約 24 倍，逆方向では約 19 倍，計測範囲全体の実行時間では，約 20 倍の高速化が見られた．この高速化は 2 軸分割 3 次元 FFT での実行の際に問題であったデータの転置部の実行時間が削減されたことに起因しており，当初の予想通りの結果となった．

また，今回の DFT による方法を乱流 DNS コードに組み込み $n^3 = 256^3$，64 ノード，256 並列　(y=64, z=64)，$K_c = 3$ にて実行時間の計測を行った結果，1 ステップの時間発展で実行時間を約 46%削減することができた．また外力注入部のフーリエ変換だけの実行時間では約 75%削減することが確認できた．

謝辞　本研究の一部は，HPCI システム利用研究課題（課題番号：hp200184，hp210138，hp210164）を通じて，理化学研究所のスーパーコンピュータ「富岳」の計算資源の提供を受け実施しました．また，文部科学省「次世代領域研究開発」（高性能汎用計算機高度利用事業費補助金）量子アニーリングアシスト型次世代スーパーコンピューティング基盤の開発の助成を受けて実施したものです．

参考文献

[1] S. K. Lele, ”Compact finite difference schemes with spectral-like resolution,” Journal of Computational Physics, Vol.103,pp.16-42(1992)
[2] S. Gottlieb, C. W. shu, ”Total variation diminishing Runge-Kutta schemes,” Math. Comput. 67(1998), 73-85
[3] M. R. Petersen, and D. Livescu, ”Forcing for statistically stationary compressible isotropic turbulence,” Physics of Fluids , 22.11(2010)：116101.
[4] J. W. Cooley and J. W. Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series,” Mathematics of Computation, Vol. 19, No. 90, pp. 297–301 (1965)
[5] M. Frigo and S. G. Johnson, “The Design and Implementation of FFTW3,” Proceedings of the IEEE, vol. 93, pp. 216-231 (2005).
[6] D. Takahashi, “FFTE: A Fast Fourier Transform Package http://www.ffte.jp/,” (2022/1/12 閲覧)
[7] “Fugaku Portal https://www.fugaku.r-ccs.riken.jp/,” (2022/1/12 閲覧)
[8] Y. Sakurai, T. Ishihara, H. Furuya, M. Umemura, and K. Shiraishi, ”Effects of the Compressibility of Turbulence on the Dust Coagulation Process in Protoplanetary Disks,” The Astrophysical Journal, 911, April 20, pp.140-155 (2021)
[9] 武中　裕次郎, ”非圧縮乱流直接数値シミュレーションコードの開発と大規模計算機向け最適化,” 神戸大学大学院システム情報学研究科　修士論文 (2020)