Vol.2020-HPC-175 No.14
2020/7/30

ディープラーニング技術の流体解析への応用

松本 正晴[1,a)]

概要：偏微分方程式は科学技術分野において，自然・社会現象を記述する数理モデルとして定式化され，その数値解法は科学技術計算における最も重要なタスクの一つとなっている．一方，近年様々な分野においてディープラーニング技術が利活用されており，偏微分方程式の求解手法にも応用され始めている．Sirignano らは偏微分方程式の微分演算子，境界条件，初期条件を満たすように損失関数を構成し，ニューラルネットワークに学習させることによって近似解を得る方法（Deep Galerkin Method, DGM）を提案している．この手法は，予め観測値や数値解などのデータを使うことなく，時空間のランダムな座標の値のみから偏微分方程式の解を学習・推論するもので，数学的に偏微分方程式の求解そのものに注目している点が他の手法とは異なる．しかしこの手法による応用例はまだ数が少なく，どの程度の偏微分方程式が解析可能なのか明らかでない部分が多い．そこで本研究では，DGM の実用性を考察するため，これを流体解析，つまり圧縮性 Navier-Stokes 方程式の求解に応用することを目的とする．2 次元 Burgers 方程式による予備計算，1 次元 Navier-Stokes 方程式による衝撃波管問題，2 次元 Navier-Stokes 方程式による鈍頭物体周りの超音速流れの 3 種類の問題に対して，DGM を適用した．DGM による推論を同じ計算条件の差分法による計算と比較した結果，両者とも概ねよく一致しており，DGM の有効性が示される結果を得た．

Application of Deep-Learning Technique to CFD Analysis

1. はじめに

近年，ディープラーニング（Deep Learning，深層学習）[1] が様々な分野で注目を集めている．ディープラーニングとは，機械学習手法の一種であり，多層の人工ニューラルネットワークと大量の学習データを利用することによって，分類・回帰分析やクラスタリングを行うものであり，特に音声・画像認識や自然言語処理の分野において他の手法を圧倒する高い性能が得られたことから 2010 年代より一気に普及した．ディープラーニングに利用されるニューラルネットワークのモデルや最適化アルゴリズム，ディープラーニングを行うためのライブラリやフレームワーク，ディープラーニングに適したハードウェア（CPU や GPU）などは日進月歩で研究開発が行われている．

このディープニューラルネットワークが持つ豊富な関数表現を偏微分方程式の解法へ応用しようとする試みが近年，いくつか提案されている．一般に微分方程式は，様々な分野において，自然・社会現象を記述する数理モデルとして定式化され，その数値解法は科学技術計算における最も重要なタスクの一つとなっている．実は単〜数層のニューラルネットワークを利用して微分方程式の解を得ようとする試みは 1990 年代から提案されている [2], [3], [4], [5]．当時はまだニューラルネットワークの隠れ層を多層化する技術が低く，ニューラルネットワークが持つ表現力に限界があったため，比較的簡単な微分方程式の解を求めるにとどまっていたようである．一方，近年のディープラーニング技術の応用について例を挙げると，リアルタイムで非圧縮性流体のシミュレーションを行うことを目的として，Euler 方程式（非粘性 Navier-Stokes 方程式）の求解に差分法と畳み込みニューラルネットワークを組み合わせる手法が提案されている [6]．非圧縮性流体の数値解法である Marker-And-Cell（MAC）法では，圧力の Poisson 方程式の求解に最も計算時間がかかるため，速度場の発散がゼロという非圧縮性流体の特徴を使い，圧力を畳み込みニューラルネットワークに学習・推論することで，シミュレーション時間を低減させることに成功している．数値流体分野へのディープラーニングの応用は他にも，粒子法に適用したもの [7] や，速度場のみから解を推論するもの [8] などが提案されているが，いずれの手法も CG やゲームなどへの応用が考えられ

[1] 東京大学大学院情報理工学系研究科
The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[a)] matsumoto@is.s.u-tokyo.ac.jp

ている．この他にも Poisson 方程式の解をディープラーニングによって推論する手法は提案されている [9]．これは，誘電率分布と距離の関数として表される電荷密度分布を多数用意し，これらの計算条件下で差分法による数値解を予め求めておき，それを学習データとして畳み込みニューラルネットワークに学習させることで，誘電率分布と電荷密度分布を入力すると出力として静電ポテンシャルが予測される，というものである．また，方程式の解に注目するのではなく，潜在的にノイズの多い観測値から物理の隠れた法則を発見することを目的に，観測データの進展から非線形関数（ニューラルネットワーク）を学習する手法が提案されており，Burgers，KdV，Kramoto-Sivashinsky，Schrödinger，Navier-Stokes の各方程式へ応用した結果について示されている [10]．

これらの手法はいずれも，実験による観測値や差分法などによる数値解など，すでに予め存在する学習データを入力とすることで，ニューラルネットワークを学習・推論する．一方，Sirignano らは偏微分方程式の微分演算子，境界条件，初期条件を満たすように損失関数を構成し，ニューラルネットワークに学習させることによって近似解を得る方法（Deep Galerkin Method, DGM）を提案しており，例として多次元の Hamilton-Jacobe-Bellman 方程式と 1 次元 Burgers 方程式へ応用した結果が示されている [11]．この手法は，予め観測値や数値解などのデータを使うことなく，時空間のランダムな座標の値のみから偏微分方程式の解を学習・推論するもので，数学的に偏微分方程式の求解そのものに注目している点が他の手法とは異なる．DGM の詳細については 2 節で述べるが，同論文中で示された 1 次元 Burgers 方程式の結果では，時間 t，位置座標 x，上流と下流側の境界値 a，b，移流項や拡散項に付く係数 α，ν として，これらの値を入力としてランダムに与えてニューラルネットワークを学習することで，近似解 $f(t,x,a,b,\alpha,\nu)$ をニューラルネットワークの出力として得る．ニューラルネットワークの学習にかかる計算コストは低くないが，一度学習を終えてしまえば，行列の基本線形演算のみで偏微分方程式の近似解を推論できるようになる．しかしこの手法による応用例はまだ数が少なく，どの程度の偏微分方程式が解析可能なのか明らかでない部分が多い．

そこで本研究では，DGM の実用性を考察するため，これを流体解析，つまり圧縮性 Navier-Stokes 方程式の求解に応用することを目的とする．

2. Deep Galerkin Method

2.1 DGM の特徴

DGM はディープラーニングを利用する他の手法と比較して，2 つの特徴を持つ．1 つ目はニューラルネットワークの学習に利用する入力データにラベル付けされた学習データセットを必要としない点，2 つ目は偏微分方程式の数値解法ではほぼ必ず利用される計算格子を必要としないメッシュフリーな点である．一般にディープラーニングでは，入力データとしてラベル付けされた大量のデータセットを用意する必要があるが，DGM では入力データとしてランダムな時空間の座標点のみを与えるので，予めデータを用意しておく必要がない．また，多次元の偏微分方程式では計算格子の形成自体が難しくなることからメッシュフリーであることは多次元計算にとっては重要となる．

2.2 アルゴリズム

以下に，DGM のアルゴリズムについて述べる．ここでは，計算対象として以下に示すような空間 d 次元放物型偏微分方程式を考える．

$$\begin{aligned}\frac{\partial u(t,x)}{\partial t}+\mathcal{L}u(t,x)&=0, &&(t,x)\in[0,T]\times\Omega\\ u(t,x)&=g(t,x), &&(t,x)\in[0,T]\times\partial\Omega\\ u(t=0,x)&=u_0(x), &&x\in\Omega\end{aligned}\tag{1}$$

ここで $x\in\Omega\subset\mathbb{R}^d$，$\mathcal{L}$ は空間微分の演算子（ここでは放物型の方程式としているので，2 階微分項を含む空間微分項となる），g, u_0 はそれぞれ u の境界条件と初期条件を示す．DGM では図 1 に示すように，

(1) 計算領域 $[0,T]\times\Omega$ から (t_n,x_n)，境界条件の領域 $[0,T]\times\partial\Omega$ から (τ_n,z_n)，そして初期条件の領域 $t=0,\Omega$ から w_n のそれぞれの座標点を乱数で生成する．

(2) 乱数で生成した座標点 $s_n=\{(t_n,x_n),(\tau_n,z_n),w_n\}$ をニューラルネットワークの入力データとして，以下の損失関数 $G(\theta_n,s_n)$ を計算する．

$$\begin{aligned}G(\theta_n,s_n)=&\left(\frac{\partial f(t_n,x_n;\theta_n)}{\partial t}+\mathcal{L}f(t_n,x_n;\theta_n)\right)^2\\&+\left(f(\tau_n,z_n;\theta_n)-g(\tau_n,z_n)\right)^2\\&+\left(f(0,w_n;\theta_n)-u_0(w_n)\right)^2\end{aligned}\tag{2}$$

ここで，$f(t_n,x_n;\theta_n)$ は，ニューラルネットワークにより推論（出力）される $u(t,x)$ の近似値，θ_n は最適化すべきニューラルネットワークのパラメータを示す．つまり，損失関数 G を最小化（ゼロ）する θ_n を求めることがこの手法の目的となる．

(3) 確率的勾配降下法に代表される最適化手法により，θ_{n+1} を求める．

$$\theta_{n+1}=\theta_n-\alpha_n\nabla_\theta G(\theta_n,s_n)\tag{3}$$

(4) 以上の (1)〜(3) について，収束条件を満たすまで（G が一定の値以下になるまで）繰り返す．

損失関数 $G(\theta_n,s_n)$ の右辺に現れる 3 つの項ではそれぞれ，計算対象としている偏微分方程式の微分演算子，境界条件，初期条件からの推論値の差分を評価しており，もしこの損失関数の値がゼロになった場合は，それはつまり計算

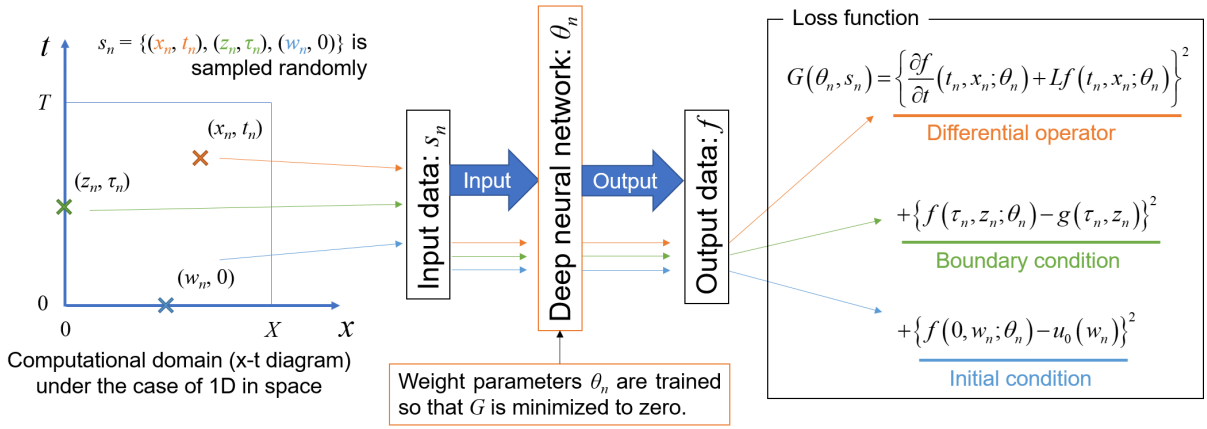


図 1: DGM におけるニューラルネットワークの学習プロセス
Fig. 1 Training process of the neural network in DGM

対象としている偏微分方程式の近似解を求めた（すなわちニューラルネットワーク自体が近似関数である）ことと同義となる．したがって，DGM では他の手法のように，予め用意されている実験結果や数値解析結果などから解の特徴量を抽出してそれを基に推論する，という方法論ではなく，ランダムな座標点で実際に偏微分方程式を評価し解を修正・記憶していく，という方法論によって解を推論する．言い換えると，ニューラルネットワークという巨大な非線形関数を回帰関数として使う重回帰分析を偏微分方程式を実際に評価しながら行っている，と言える．したがって DGM では，予め設定されている計算領域（時空間座標の定義域）の外まで解を予測できるわけではないことに注意が必要である．

式（2）中の右辺第 2 項の境界条件 g と第 3 項の初期条件 u_0 の関数については，式（1）に示すように，計算対象となる偏微分方程式と同時に与えられるが，第 1 項の時間微分項 $\partial f/\partial t$，ならびに空間微分項 $\mathcal{L}f$ については陽に与えられない．しかし，ニューラルネットワークの出力である $f(t_n, x_n; \theta_n)$ は，ニューラルネットワークのアーキテクチャが決定した段階でどのような計算をして出力されるかがわかるので，偏微分の連鎖律（チェインルール）を用いる自動微分法によって求めることが可能である．仮にニューラルネットワークをブラックボックスとして扱う場合でも，差分近似に基づく数値微分法によって上記の項は求めることができる（メッシュフリーなので微小区間 Δt や Δx の与え方は任意となる）．

2.3 DGM ネットワークアーキテクチャ

一般に，ニューラルネットワークのアーキテクチャには順伝播型や再帰型，畳み込み型など，アプリケーションに応じて数多くの種類が存在するが，DGM では，関数 $u(t, x)$ が x に対して急に不連続となるようなプロファイルを上手く再現できるモデル $f(t, x; \theta)$ を作ることが必要となる．このような急峻なプロファイルは 2 階微分などの拡散項を含むことによってある程度平滑化されることが期待されるが，方程式が非線形の場合はその限りではない．そこで Sirignano らは，Long Short-Term Memory（LSTM）アーキテクチャ [12] を参考に DGM に適したネットワークであるとして，以下のようなネットワークアーキテクチャを提案している（図 2 参照）．

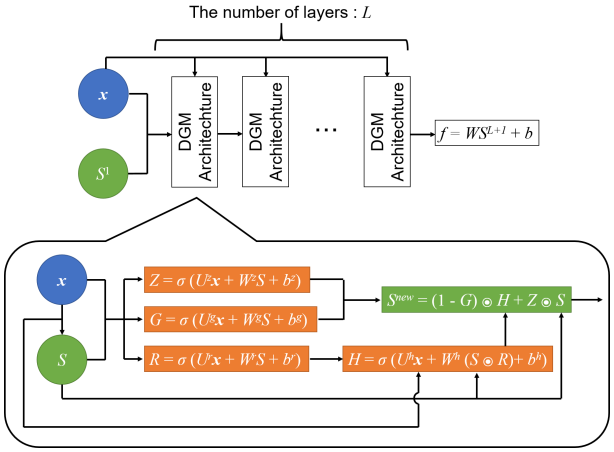


図 2: DGM アーキテクチャ
Fig. 2 Neural network architecture used in DGM

$$
\begin{aligned}
S^1 &= \sigma(W^1 \boldsymbol{x} + b^1), \\
Z^l &= \sigma(U^{z,l} \boldsymbol{x} + W^{z,l} S^l + b^{z,l}), \quad l = 1, ..., L \\
G^l &= \sigma(U^{g,l} \boldsymbol{x} + W^{g,l} S^l + b^{g,l}), \quad l = 1, ..., L \\
R^l &= \sigma(U^{r,l} \boldsymbol{x} + W^{r,l} S^l + b^{r,l}), \quad l = 1, ..., L \\
H^l &= \sigma(U^{h,l} \boldsymbol{x} + W^{h,l} (S^l \odot R^l) + b^{h,l}) \quad l = 1, ..., L \\
S^{l+1} &= (1 - G^l) \odot H^l + Z^l \odot S^l \\
f(t, x; \theta) &= W S^{L+1} + b
\end{aligned}
\tag{4}
$$

ここで，$\boldsymbol{x} = (t, x)$，$L+1$ は隠れ層の数，記号 $\odot$ は Hadamard 積をそれぞれ示す．したがって，ニューラルネットワーク

のパラメータ θ は，まとめると以下のようになる．

$$\theta = \{W^1, b^1, \ \left(U^{z,l}, W^{z,l}, b^{z,l}\right)_{l=1}^{L}, \left(U^{g,l}, W^{g,l}, b^{g,l}\right)_{l=1}^{L}, \left(U^{r,l}, W^{r,l}, b^{r,l}\right)_{l=1}^{L}, \left(U^{h,l}, W^{h,l}, b^{h,l}\right)_{l=1}^{L}, W, b\} \quad (5)$$

また，各層のユニット数を M とすると，$\sigma : \mathbb{R}^M \to \mathbb{R}^M$ は以下に示す要素ごとの非線形変換を示している．

$$\sigma = (\phi(z_1), \phi(z_2), ..., \phi(z_M)) \quad (6)$$

ここで，$\phi : \mathbb{R} \to \mathbb{R}$ は非線形の活性化関数を示しており，一般にはシグモイド関数や Rectified Linear Unit（ReLU），tanh 関数などが使われることが多い．θ の中の変数の次元は以下の通り．$W^1 \in \mathbb{R}^{M\times(d+1)}$，$b^1 \in \mathbb{R}^M$，$U^{z,l} \in \mathbb{R}^{M\times(d+1)}$，$W^{z,l} \in \mathbb{R}^{M\times M}$，$b^{z,l} \times \mathbb{R}^M$，$U^{g,l} \in \mathbb{R}^{M\times(d+1)}$，$W^{g,l} \in \mathbb{R}^{M\times M}$，$b^{g,l} \times \mathbb{R}^M$，$U^{r,l} \in \mathbb{R}^{M\times(d+1)}$，$W^{r,l} \in \mathbb{R}^{M\times M}$，$b^{r,l} \times \mathbb{R}^M$，$U^{h,l} \in \mathbb{R}^{M\times(d+1)}$，$W^{h,l} \in \mathbb{R}^{M\times M}$，$b^{h,l} \times \mathbb{R}^M$，$W \in \mathbb{R}^{1\times M}$，$b \in \mathbb{R}$

ハイパーパラメータとして隠れ層の数 L と，各層のユニット数 M，活性化関数 ϕ を決める必要があるが，Sirignano らによると，$L = 3$，$M = 50$，$\phi(z) = \tanh(z)$ 程度が効果的であるとしており，問題によって値を増減させている．

2.4 DGM の実装

図 2 に示すニューラルネットワークの実装には，機械学習等のフレームワークとしてよく用いられている Tensorflow[13] を利用した．Tensorflow は，幅広く変数の微分値が計算できるトップダウン型自動微分機能を搭載しており，ニューラルネットワークの学習に利用される誤差逆伝播法の実装や，本実装で必要となるニューラルネットワークに対する x や θ の微分を簡単に求めることができる．また，確率的最適化手法についても多くの手法が利用可能であり，本研究ではよく知られている ADAM アルゴリズム [14] を採用した．さらに Tensorflow では GPU を用いた計算を簡単に行うことができ，本研究では GPU（NVIDIA Titan RTX）2 台を利用して同期的に分散並列学習を行った．前述の通り，ニューラルネットワークの学習に必要な入力値は時空間座標をランダムに与えるが，1 回のイタレーションに各 GPU ノードでバッチサイズ 4,096 のバッチ（つまり，1 イタレーションで入力データは $s_n \times 4,096$ の時空間点）を含め，計 100,000 イタレーションで学習を行った．学習時間は計算対象の偏微分方程式や計算条件，ハイパーパラメータにもよるが，数時間〜1 日オーダーとなる．

3. 計算結果

本節では以下に示す 3 つの計算結果について述べる．1）2 次元 Burgers 方程式（予備計算），2）1 次元圧縮性 Navier-Stokes 方程式（衝撃波管問題），3）2 次元圧縮性 Navier-Stokes 方程式（鈍頭物体周りの超音速流れ）

3.1 2 次元 Burgers 方程式（予備計算）

Navier-Stokes 方程式は質量保存則，運動量保存則，エネルギー保存則の 3 式からなる連立偏微分方程式であるが，DGM で連立方程式を解くことができるかどうかは自明ではない．そこで Navier-Stokes 方程式の解析を行う前に，予備計算として 2 次元 Burgers 方程式を対象に計算を行った．Burgers 方程式は非線形の移流拡散方程式であり，非圧縮性 Navier-Stokes 方程式の運動量保存則の圧力項を無視した形で表されるため，Navier-Stokes 方程式の予備計算としては妥当であると考えられる．空間 2 次元以上の Burgers 方程式は連立方程式の形となる．対象とする方程式を以下に示す．

$$\begin{cases} \dfrac{\partial u_x}{\partial t} + u_x\dfrac{\partial u_x}{\partial x} + u_y\dfrac{\partial u_x}{\partial y} = \nu\left(\dfrac{\partial^2 u_x}{\partial x^2} + \dfrac{\partial^2 u_x}{\partial y^2}\right) \\ \dfrac{\partial u_y}{\partial t} + u_x\dfrac{\partial u_y}{\partial x} + u_y\dfrac{\partial u_y}{\partial y} = \nu\left(\dfrac{\partial^2 u_y}{\partial x^2} + \dfrac{\partial^2 u_y}{\partial y^2}\right) \end{cases}$$

$$boundary \quad \begin{cases} \boldsymbol{u}(t,0,y) = \boldsymbol{u}(t,1,y) \\ \boldsymbol{u}(t,x,0) = \boldsymbol{u}(t,x,1) \end{cases}$$

$$initial \quad \begin{cases} u_x(0,x,y) = \sin(4\pi y) \\ u_y(0,x,y) = \dfrac{1}{2}\cos(2\pi x) \end{cases} \quad (7)$$

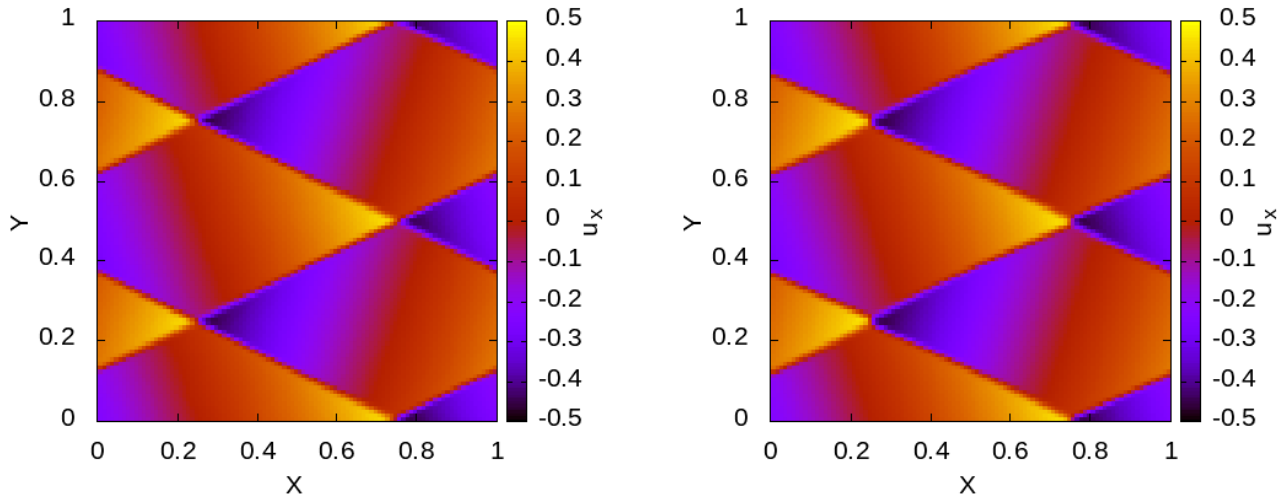
ここで $\boldsymbol{u}(t,x,y) = (u_x(t,x,y), u_y(t,x,y))$, $(0 \le x \le 1, 0 \le y \le 1)$ である．境界条件は周期的境界条件，初期条件は上式の通りとした．拡散項の係数 ν は 0，1×10^{-4}，5×10^{-4}，1×10^{-3}，5×10^{-3}，1×10^{-2} の計 6 パターンの計算を行った．ν について，Sirignano らによる 1 次元 Burgers 方程式の計算では，1 節で述べたように，ニューラルネットワークの入力としていたが，ここでは簡単のために上記の 6 パターン固定として計算を行った．

損失関数 L は計算対象が連立方程式であることを考慮して，以下のように設定した．

$$L = (L_1^2 + L_2^2) + (L_3^2 + L_4^2) + (L_5^2 + L_6^2)$$

$$\begin{cases} L_1 = \dfrac{\partial f_x^1}{\partial t} + f_x^1\dfrac{\partial f_x^1}{\partial x} + f_y^1\dfrac{\partial f_x^1}{\partial y} - \nu\left(\dfrac{\partial^2 f_x^1}{\partial x^2} + \dfrac{\partial^2 f_x^1}{\partial y^2}\right) \\ L_2 = \dfrac{\partial f_y^1}{\partial t} + f_x^1\dfrac{\partial f_y^1}{\partial x} + f_y^1\dfrac{\partial f_y^1}{\partial y} - \nu\left(\dfrac{\partial^2 f_y^1}{\partial x^2} + \dfrac{\partial^2 f_y^1}{\partial y^2}\right) \\ L_3 = f_x(t_2,x_2,y_2) - f_x^b \\ L_4 = f_y(t_2,x_2,y_2) - f_y^b \\ L_5 = f_x(0,x_3,y_3) - \sin(4\pi y_3) \\ L_6 = f_y(0,x_3,y_3) - \dfrac{1}{2}\cos(2\pi x_3) \end{cases} \quad (8)$$

ここで f_x, f_y はニューラルネットワークの出力として推論される u_x, u_y の近似解であり，$f_x^1 = f_x(t_1,x_1,y_1)$，$f_y^1 = f_y(t_1,x_1,y_1)$ を示す．また (t_1,x_1,y_1) は計算領域内



(a) DGM による u_x の推論結果　　(b) FDM による u_x の計算結果

図 3: 2 次元 Burgers 方程式における結果（$\nu = 5 \times 10^{-4}, t = 1$）

Fig. 3 Comparison of the result of 2D Burgers equations

からランダムに抽出した座標，(t_2, x_2, y_2) は境界条件に対応する領域からランダムに抽出した座標，$(0, x_3, y_3)$ は初期条件に対応する領域からランダムに抽出した座標を示す．f_x^b, f_y^b は以下のように境界条件（周期的境界条件）の値を表している．

$$f_n^b(t_2, x_2, y_2) = \begin{cases} f_n(t_2, 0, y_2) & (x_2 = 1) \\ f_n(t_2, 1, y_2) & (x_2 = 0) \\ f_n(t_2, x_2, 0) & (y_2 = 1) \\ f_n(t_2, x_2, 1) & (y_2 = 0) \end{cases} \tag{9}$$

ここで n は x, y を示す．

図 3a に $\nu = 5 \times 10^{-4}, t = 1$ の場合のニューラルネットワークによる u_x の推論結果，図 3b に比較対象として，1 次精度風上差分法（以下，FDM）により計算した u_x の近似解の結果を示す．2 次元 Burgers 方程式は非線形な方程式であり，厳密解を求めるのは困難であるので，その代わりにここでは FDM により計算した結果を示している．DGM，FDM どちらの結果も同じ格子点でプロットした結果を示している（格子点数は $N_x \times N_y = 101 \times 101$）．両者を比較するとよい一致を示しており，連立偏微分方程式の解法にも DGM が有効利用できることがわかる．ここでは示していないが，ν を変えた 6 パターンの計算のうち，ν の値が高い 4 つの条件（$\nu = 5 \times 10^{-4}$，1×10^{-3}，5×10^{-3}，1×10^{-2}）では，DGM と FDM の結果はよく一致するが，ν の値が低い 2 つの条件（$\nu = 0$，1×10^{-4} では，DGM による学習の進展とともに減少する損失関数の値が飽和傾向にあり，学習が収束せず，FDM の結果と比較して大きく異なる推論結果となった．一般に，Burgers 方程式の移流項は時間の経過に対して解を急峻にさせる傾向にある一方で，拡散項は解を滑らかにする傾向にある．拡散項の値がゼロ，もしくは低い場合，移流項の効果により，解は急峻な不連続面を形成する（例えば図 3 で表されているひし形模様の周囲）ようになるが，不連続面が発達すればするほど，そこでの空間微分値は評価が難しくなる（完全な不連続面では微分不可能である）ことから，ν がある一定以上に低い値となった場合，DGM による学習がうまく機能しなくなるものと考えられる．これは差分法において不連続面を扱う場合に数値（人工）粘性が必要になることとよく似た問題であると考えられる．

3.2 1 次元 Navier-Stokes 方程式（衝撃波管問題）

次に，1 次元 Navier-Stokes 方程式による衝撃波管問題を対象に計算を行う．衝撃波管問題では，初期条件に高圧領域と低圧領域を配置し，その後の時間発展を解くことで衝撃波伝播の様子を計算する．この問題は一般に，差分法や有限体積法等の数値（離散化）手法の検証として解かれることが多い．衝撃波を扱う問題では，発生する衝撃波面前後で物理量の急峻な不連続面が現れ，数値的に不安定になりやすい．そこで，圧縮性流体用の数値手法では，衝撃波を捕獲するために数値粘性が付加され，数値的な不安定性を抑える工夫がされているが，このような衝撃波管問題の数値計算では，Euler 方程式，すなわち非粘性 Navier-Stokes 方程式を支配方程式とすることが多い．これは，数値手法の検証から，数値粘性によって衝撃波面の急峻な不連続面がなだらかになる度合いを見るため，物理粘性を排除する必要があることによる．一方，DGM を用いて衝撃波管問題を解く場合に Euler 方程式を支配方程式としてしまうと，粘性が存在しない急峻な不連続面において空間微分が不可能な状況となり，学習がうまく進まなくなる．すなわち，前節の 2 次元 Burgers 方程式における拡散項の係数が低い

場合に学習ができなくなることと同じ問題が現れることとなる．

そこで本研究では，粘性項を無視しない Navier-Stokes 方程式を支配方程式としている．簡単のため，エネルギー方程式の熱伝導項は無視し，粘性係数 μ が計算領域で一定と仮定すると，対象とする支配方程式は以下のように定式化される．

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \dfrac{\partial (\rho u_x)}{\partial x} = 0 \\ \rho \dfrac{\partial u_x}{\partial t} + u_x \dfrac{\partial u_x}{\partial x} + \dfrac{\partial p}{\partial x} = \dfrac{1}{\mathrm{Re}} \dfrac{4}{3} \dfrac{\partial^2 u_x}{\partial x^2} \\ \dfrac{\partial p}{\partial t} + u_x \dfrac{\partial p}{\partial x} + \gamma p \dfrac{\partial u_x}{\partial x} = (\gamma - 1) \dfrac{1}{\mathrm{Re}} \dfrac{4}{3} \left(\dfrac{\partial u_x}{\partial x} \right)^2 \end{cases} \tag{10}$$

ここで，ρ は密度，u_x は x 方向流速，p は圧力，γ は比熱比（ここでは 5/3 で一定）をそれぞれ示している．（10）式に示すようにここでは保存形ではなく，非保存形で Navier-Stokes 方程式を表している．これは，保存形のままでは，ニューラルネットワークの損失関数を評価する際に計算（特に微分値の評価）が複雑になることから，より簡単に評価可能な非保存形で表すこととした．また，（10）式は代表長さ L_c，代表密度 ρ_c，代表速度 u_c，代表時間 $t_c = L_c/u_c$，代表圧力 $p_c = \rho_c u_c^2$ で無次元化されており，以上の代表値を使って（10）式に現れる Reynolds 数 Re は以下のように表される．

$$\mathrm{Re} = \frac{\rho_c u_c L_c}{\mu} \tag{11}$$

この Reynolds 数についてもニューラルネットワークの入力とすることで，指定した範囲の任意の Reynolds 数の流れ場を推論できるようにする．つまり，ニューラルネットワークへの入力データとして，ランダムな時空間座標だけではなく，1/Re の値についても含むこととする．ここでは 1/Re の値は，10^{-2} から 10^{-3} の間のランダムな値とした．

DGM による推論では，式の無次元化がニューラルネットワークの学習結果に対して重要な役割を持つと考えられる．一般に，ニューラルネットワークの学習で用いられる入力データにおいて，パラメータ間のスケールが大きく異なる場合，勾配降下法による更新幅にパラメータ間で偏りが生じてしまい，学習精度の低下や，学習が進まないなどの影響が表れることがある．これは，入力データの前処理として正規化や標準化を行うことで防ぐことができるが，DGM においては式の無次元化がこの入力データの前処理に対応していると考えられ，本研究では，式の無次元化により全てのパラメータが概ね 0〜1 程度の値となるよう計算条件を設定した．

図 4 に計算領域と計算条件を示す．計算領域は $0 \leq x \leq 1$ として，初期条件として $x = 0.5$ を境に，上流側（左）に高圧領域，下流側（右）に低圧領域を配置する．上流側と下流側境界条件は Neuman 条件とし，全ての物理量で勾配

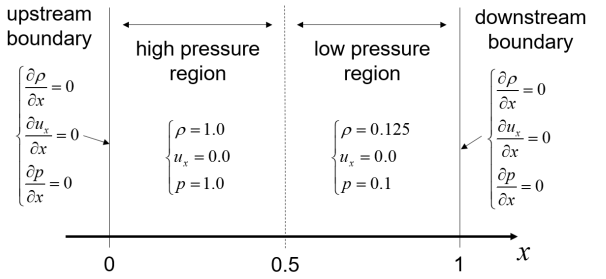


図 4: 衝撃波管問題における計算領域と計算条件

Fig. 4 Computational domain and working conditions for shock tube problem

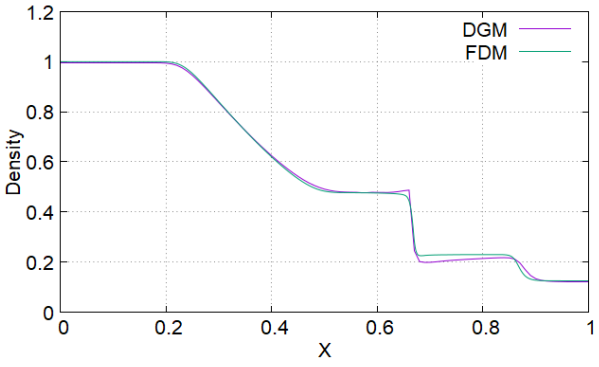
ゼロとした．これらの条件を踏まえ，損失関数 L を以下のように設定した．

$$L = (L_1 + L_2 + L_3) + (L_4 + L_5 + L_6) + (L_7 + L_8 + L_9)$$

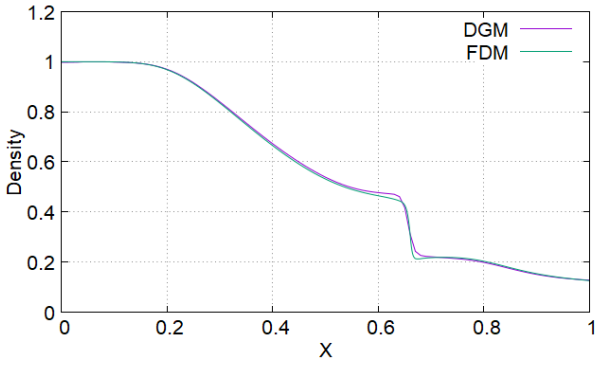
$$\begin{cases} L_1 = \dfrac{\partial f_\rho^1}{\partial t} + \dfrac{\partial (f_\rho^1 f_u^1)}{\partial x} \\ L_2 = f_\rho^1 \dfrac{\partial f_u^1}{\partial t} + f_u^1 \dfrac{\partial f_u^1}{\partial x} + \dfrac{\partial f_p^1}{\partial x} - \dfrac{1}{\mathrm{Re}} \dfrac{4}{3} \dfrac{\partial^2 f_u^1}{\partial x^2} \\ L_3 = \dfrac{\partial f_p^1}{\partial t} + f_u^1 \dfrac{\partial f_p^1}{\partial x} + \gamma f_p^1 \dfrac{\partial f_u^1}{\partial x} - (\gamma - 1) \dfrac{1}{\mathrm{Re}} \dfrac{4}{3} \left(\dfrac{\partial f_u^1}{\partial x} \right)^2 \\ L_4 = f_\rho(t_2, x_2) - \dfrac{\partial f_\rho(t_2, x_2)}{\partial x} \\ L_5 = f_u(t_2, x_2) - \dfrac{\partial f_u(t_2, x_2)}{\partial x} \\ L_6 = f_p(t_2, x_2) - \dfrac{\partial f_p(t_2, x_2)}{\partial x} \\ L_7 = \begin{cases} f_\rho(0, x_3) - 1.0 & (0 \leq x_3 \leq 0.5) \\ f_\rho(0, x_3) - 0.125 & (0.5 < x_3 \leq 1.0) \end{cases} \\ L_8 = f_u(0, x_3) \\ L_9 = \begin{cases} f_p(0, x_3) - 1.0 & (0 \leq x_3 \leq 0.5) \\ f_p(0, x_3) - 0.1 & (0.5 < x_3 \leq 1.0) \end{cases} \end{cases} \tag{12}$$

ここで f_ρ, f_u, f_p はニューラルネットワークの出力として推論される ρ, u_x, p の近似解であり，$f_\rho^1 = f_\rho(t_1, x_1)$，$f_u^1 = f_u(t_1, x_1)$，$f_p^1 = f_p(t_1, x_1)$ を示す．また (t_1, x_1) は計算領域内からランダムに抽出した座標，(t_2, x_2) は境界条件に対応する領域からランダムに抽出した座標，$(0, x_3)$ は初期条件に対応する領域からランダムに抽出した座標を示す．

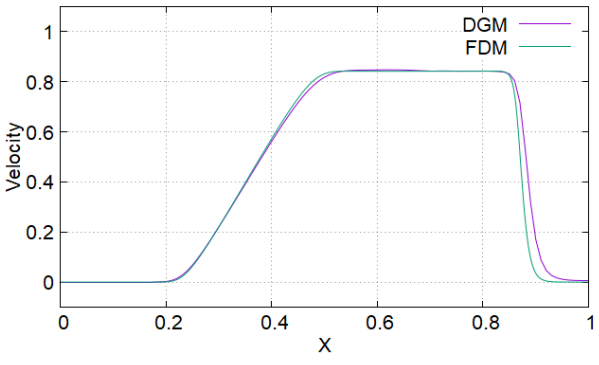
DGM による推論結果の妥当性を評価するために比較対象として，差分法（2 次精度 TVD 法 [15]，以下 FDM）によって計算した結果についても併せて示す．FDM による計算では，x 方向に 1001 点の格子点を利用し，DGM の計算条件（図 4 参照）に合わせて計算を行った．図 5 に $t = 0.2$ 経過時における ρ, u_x, p の各値の DGM による推論結果と FDM による計算結果を併せて示す．図 5a，5c，5e が $1/\mathrm{Re} = 10^{-3}$ の結果，図 5b，5d，5f が $1/\mathrm{Re} = 10^{-2}$ の



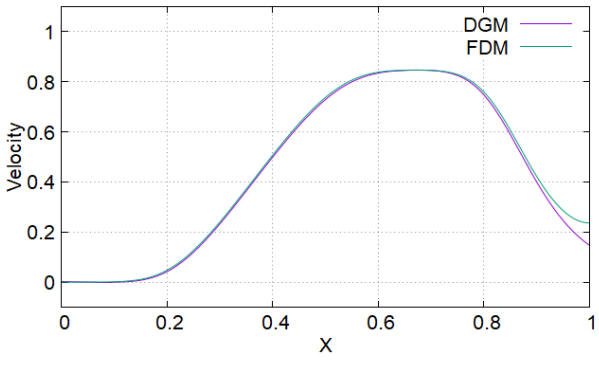
(a) $1/\mathrm{Re} = 10^{-3}, t = 0.2$ での密度（ρ）分布の比較



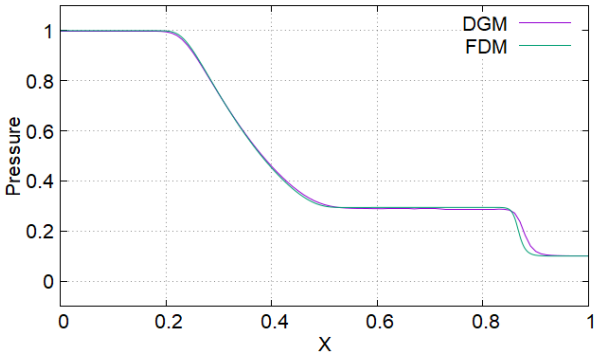
(b) $1/\mathrm{Re} = 10^{-2}, t = 0.2$ での密度（ρ）分布の比較



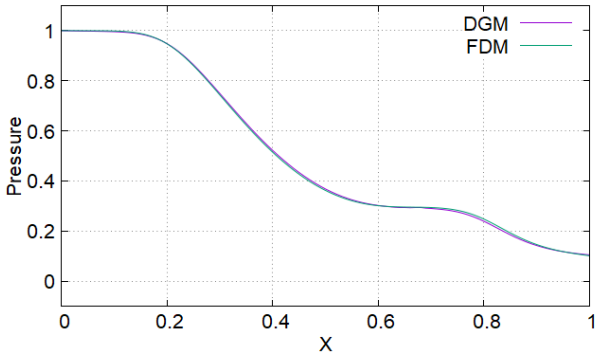
(c) $1/\mathrm{Re} = 10^{-3}, t = 0.2$ での流速（u_x）分布の比較



(d) $1/\mathrm{Re} = 10^{-2}, t = 0.2$ での流速（u_x）分布の比較



(e) $1/\mathrm{Re} = 10^{-3}, t = 0.2$ での圧力（p）分布の比較



(f) $1/\mathrm{Re} = 10^{-2}, t = 0.2$ での圧力（p）分布の比較

図 5: 衝撃波管問題における DGM による推論結果と FDM による計算結果の比較

Fig. 5 Comparisons of the result of shock tube problem

結果をそれぞれ示している．衝撃波管問題では初期条件からの時間経過とともに，下流（右）方向へは衝撃波面と接触不連続面が，反対に上流（左）方向へは膨張波がそれぞれ伝播するが，$1/\mathrm{Re} = 10^{-3}$ での結果（図 5a，5c，5e）を見ると，衝撃波面が $x = 0.9$ 付近，接触不連続面が $x = 0.7$ 付近，そして膨張波が $x = 0.2$〜0.5 付近に現れている様子がわかる．一方，$1/\mathrm{Re} = 10^{-2}$ での結果（図 5b，5d，5f）では，$1/\mathrm{Re} = 10^{-3}$ での結果と比較して粘性の影響が強く表れることから，解が拡散的になっており，特に衝撃波面ははっきりとは見えない状況となっている．

DGM による推論結果と FDM による計算結果を比較すると，概ねよい一致を示しているといえるが，分布が大きく変化する領域では両者にわずかながら差が表れていることがわかる．特に $1/\mathrm{Re} = 10^{-3}$ の結果において，衝撃波面がある $x = 0.9$ 付近の流速（図 5c）と圧力（図 5e），接触不連続面がある $x = 0.7$ 付近の密度（図 5a）など，不連続が現れるような領域で差があることがわかる．一方で，$1/\mathrm{Re} = 10^{-2}$ の結果を見ると，$1/\mathrm{Re} = 10^{-3}$ の結果と比べて，DGM と FDM の結果に差が現れていないことがわかる．これは，解が拡散的になっていることに起因しているものと考えられる．つまり DGM では，急峻なプロファイルの変化に対して，（前節の 2 次元 Burgers 方程式の場合と同様に）ニューラルネットワークの学習がうまく進みずらい傾向にあると考えられ，不連続が現れるような領域では，関数の連続性に関する微分可能性が問題になるものと示唆される．ただしここでは，各種の学習パラメータ（DGM

アーキテクチャ層の数 L，ユニットの数 M，バッチサイズ，イタレーション回数など）について最適化を行っているわけではないため，この DGM と FDM の差がこれ以上縮まらないというわけではなく，上記の学習パラメータを増加・調整・最適化することで，この差は縮まっていくものと考えるのが自然である．したがって，どの程度の学習パラメータ，学習時間であれば実用的に満足ができる結果が得られるのかについては，今後の課題となる．

3.3 2 次元 Navier-Stokes 方程式（鈍頭物体周りの超音速流れ）

次に，2 次元 Navier-Stokes 方程式による鈍頭物体周りの超音速流れを対象に計算を行う．簡単のために，エネルギー方程式の熱伝導項は無視すると，対象とする支配方程式は以下のように定式化される．

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \dfrac{\partial (\rho u_x)}{\partial x} + \dfrac{\partial (\rho u_y)}{\partial y} = 0 \\ \rho \dfrac{\partial u_x}{\partial t} + u_x \dfrac{\partial u_x}{\partial x} + u_y \dfrac{\partial u_x}{\partial y} + \dfrac{\partial p}{\partial x} = \dfrac{\partial \tau_{xx}}{\partial x} + \dfrac{\partial \tau_{yx}}{\partial y} \\ \rho \dfrac{\partial u_y}{\partial t} + u_x \dfrac{\partial u_y}{\partial x} + u_y \dfrac{\partial u_y}{\partial y} + \dfrac{\partial p}{\partial y} = \dfrac{\partial \tau_{xy}}{\partial x} + \dfrac{\partial \tau_{yy}}{\partial y} \\ \dfrac{\partial p}{\partial t} + u_x \dfrac{\partial p}{\partial x} + u_y \dfrac{\partial p}{\partial y} + \gamma p \left(\dfrac{\partial u_x}{\partial x} + \dfrac{\partial u_y}{\partial y} \right) = (\gamma - 1)\Phi \end{cases} \tag{13}$$

ここで，ρ は密度，u_x, u_y は x 方向と y 方向の流速，p は圧力，γ は比熱比（ここでは 5/3 で一定）をそれぞれ示している．(13) 式が非保存形で表されている理由は前節と同じである．また簡単のため，粘性係数 μ が計算領域で一定と仮定すると，粘性応力テンソル τ_{ij}，散逸関数 Φ はそれぞれ以下のように表される．

$$\begin{cases} \tau_{xx} = \dfrac{1}{\mathrm{Re}} \dfrac{2}{3} \left(2 \dfrac{\partial u_x}{\partial x} - \dfrac{\partial u_y}{\partial y} \right) \\ \tau_{yx} = \tau_{xy} = \dfrac{1}{\mathrm{Re}} \left(\dfrac{\partial u_x}{\partial y} + \dfrac{\partial u_y}{\partial x} \right) \\ \tau_{yy} = \dfrac{1}{\mathrm{Re}} \dfrac{2}{3} \left(2 \dfrac{\partial u_y}{\partial y} - \dfrac{\partial u_x}{\partial x} \right) \end{cases} \tag{14}$$

$$\Phi = \tau_{xx} \frac{\partial u_x}{\partial x} + \tau_{yx} \frac{\partial u_x}{\partial y} + \tau_{xy} \frac{\partial u_y}{\partial x} + \tau_{yy} \frac{\partial u_y}{\partial y} \tag{15}$$

(13) 式は各代表値で無次元化されており，全てのパラメータが概ね 0〜1 程度の値となるよう計算条件を設定した（詳細は前節参照）．

図 6 に計算領域と計算条件を示す．計算領域は $0 \leq x \leq 1, 0 \leq y \leq 1$ として，計算領域中央 $x = 0.5, y = 0.5$ を中心に半径 $r = 0.125$ の円形状の鈍頭物体を配置する．流体は $x = 0$ の入口境界から Mach 数 $M = 3$ の超音速で流入するものする．境界条件は入口境界で Dirichlet 条件として与え，物体表面の内部境界では，流速ゼロ，密度と圧力は物体表面の法線方向（n）に対して勾配ゼロとして与えた．ここで示した入口境界と内部境界以外の境界条件は自由流出条件を仮定するため，特に何の条件も与えていない．また，Reynolds 数 $\mathrm{Re} = 100$ で一定とした．損失関数についてはここでは省略するが，上記の諸条件を考慮し構成した．

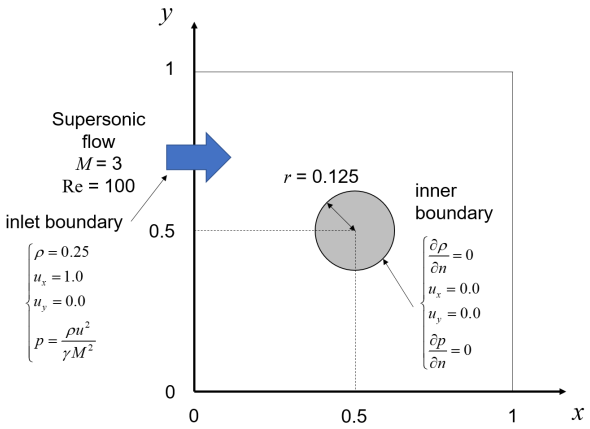


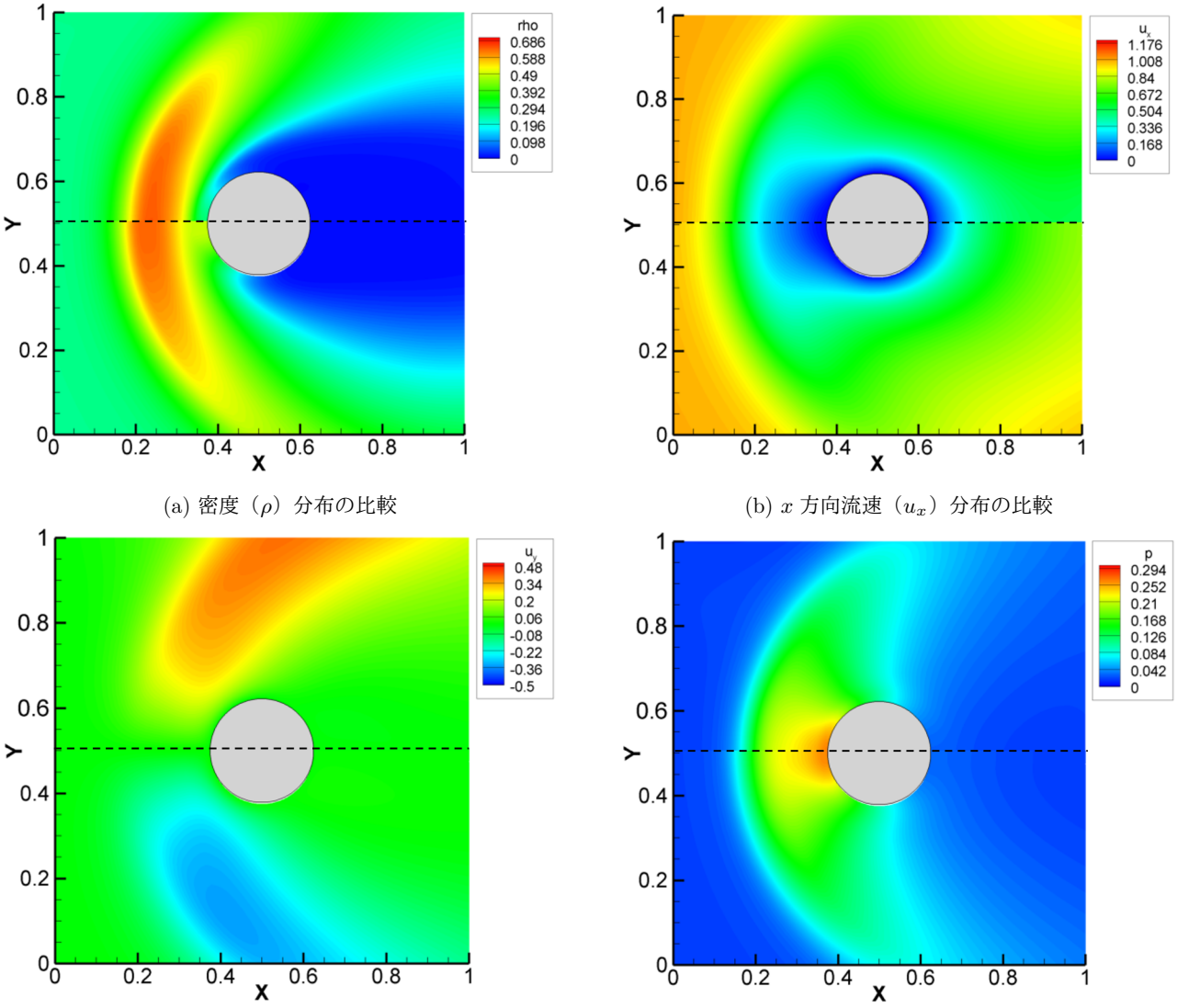
図 6: 鈍頭物体周りの超音速流れの計算領域と計算条件
Fig. 6 Computational domain and working conditions for the supersonic flow around a blunt body

DGM による推論結果の妥当性を評価するために比較対象として，差分法（2 次精度 TVD 法 [15]，以下 FDM）によって計算した結果についても併せて示す．FDM による計算では，支配方程式を一般座標へ変換し DGM の計算条件（図 6 参照）に合わせて計算を行った．図 7 に $t = 1$ 経過時における ρ, u_x, u_y, p の各値の DGM によって推論した結果と FDM による計算結果を示す．図の上半分が DGM の結果，下半分が FDM の結果を示している．鈍頭物体周りの超音速流れでは，物体前方（上流側）で弓型衝撃波が発生し，衝撃圧縮による圧力と密度の上昇が見られるとともに，流速が亜音速まで低下する．一方，物体背後（下流）では，密度は大きく低下し，流れも物体を避けるように上下に分かれて流れていく．図 7a 等を見ると，わずかに衝撃波面の位置や鈍頭物体周りの密度の値などに差が見られるが，いずれの結果も概ねよい一致を示していることがわかる．これらの結果より，DGM が 2 次元 Navier-Stokes 方程式に対しても有効に利用できることがわかる．

4. まとめ

偏微分方程式の微分演算子，境界条件，初期条件を満たすように損失関数を構成し，ディープニューラルネットワークに学習させることによって近似解を得る Deep Galerkin Method（DGM）の実用性を考察するため，2 次元 Burgers 方程式による予備計算，1 次元 Navier-Stokes 方程式による衝撃波管問題，2 次元 Navier-Stokes 方程式による鈍頭物体周りの超音速流れの 3 種類の問題に対して，DGM を適用した．DGM による推論結果を同じ計算条件の差分法による計算と比較した結果，両者とも概ねよく一致しており，DGM が流体解析にも有用であることが示された．

Reynolds 数が大きい条件（つまり粘性項や拡散項の影



(a) 密度（ρ）分布の比較

(b) x 方向流速（u_x）分布の比較

(c) y 方向流速（u_y）分布の比較

(d) 圧力（p）分布の比較

図 7: 鈍頭物体周りの超音速流れの結果比較（$t = 1$，上：DGM による推論結果，下：FDM による計算結果）

Fig. 7 Comparisons of the result of supersonic flow around a blunt body

響が相対的に小さい条件）では，衝撃波面などの急峻な不連続面が顕著に表れるが，そのような流れ場では，DGM による学習が進まなくなる傾向にあるため，今回，鈍頭物体周りの超音速流れの解析では，Reynolds 数が比較的小さい条件（Re = 100）での結果を示した．これは 2 次元 Burgers 方程式による予備解析と 1 次元衝撃波管問題の中で示した通り，衝撃波に起因する不連続面が現れるような流れ場において，そこでの関数の微分可能性に原因があると考えられる．そこで従来の数値手法で使われている数値粘性のように，不連続面だけを局所的に拡散させるような何らかの手段を用いることによって, 高 Reynolds 数流れが DGM によって効率よく学習できるようになる可能性がある．一方，解が十分滑らかになるような低 Reynolds 数の流れ場であれば，DGM による方法は効果的に解を推論できる傾向にあるものと考えられる．また，今回示した 2 次元 Navier-Stokes 方程式の結果では，簡単のためにエネルギー方程式の熱伝導項を無視したが，この項を含んだ式や，粘性係数，熱伝導係数，入口 Mach 数などの細かいパラメータについても，ニューラルネットワークの入力データとした 3 次元解析を行うことによって，3 次元 Navier-Stokes 方程式そのものの近似関数をニューラルネットワークで作ることが可能になると考えられる．さらに，本手法は放物型偏微分方程式以外にも，楕円型方程式にも応用が可能であると考えられるため，非圧縮性 Navier-Stokes 方程式やそれ以外の方程式にも適用できる可能性があり，その有効性を確認する必要がある．

謝辞　本研究を通じて有益な議論をしていただいた東京大学の吉本芳英准教授，石村脩さんに感謝申し上げます．

参考文献

[1] Goodfellow, I., Bengio, Y. and Courville, A.: *Deep Learning*, The MIT Press (2016).
[2] Lee, H. and Kang, I. S.: Neural algorithm for solving differential equations, *Journal of Computational Physics*, Vol. 91, No. 1, pp. 110 – 131 (online), DOI: https://doi.org/10.1016/0021-9991(90)90007-N (1990).
[3] Dissanayake, M. W. M. G. and Phan-Thien, N.: Neural-network-based approximations for solving partial differential equations, *Communications in Numerical Methods in Engineering*, Vol. 10, No. 3, pp. 195–201 (online), DOI: 10.1002/cnm.1640100303 (1994).
[4] Lagaris, I. E., Likas, A. and Fotiadis, D. I.: Artificial neural networks for solving ordinary and partial differential equations, *IEEE transactions on neural networks*, Vol. 9 5, pp. 987–1000 (1998).
[5] Smaoui, N. and Al-Enezi, S.: Modelling the dynamics of nonlinear partial differential equations using neural networks, *Journal of Computational and Applied Mathematics*, Vol. 170, No. 1, pp. 27 – 58 (online), DOI: https://doi.org/10.1016/j.cam.2003.12.045 (2004).
[6] Tompson, J., Schlachter, K., Sprechmann, P. and Perlin, K.: Accelerating Eulerian Fluid Simulation with Convolutional Networks, *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, JMLR.org, p. 3424–3433 (2017).
[7] Ladický, L., Jeong, S., Solenthaler, B., Pollefeys, M. and Gross, M.: Data-Driven Fluid Simulations Using Regression Forests, *ACM Trans. Graph.*, Vol. 34, No. 6 (online), DOI: 10.1145/2816795.2818129 (2015).
[8] Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M. and Solenthaler, B.: Deep Fluids: A Generative Network for Parameterized Fluid Simulations, *Computer Graphics Forum*, Vol. 38, No. 2, pp. 59–70 (online), DOI: 10.1111/cgf.13619 (2019).
[9] Tang, W., Shan, T., Dang, X., Li, M., Yang, F., Xu, S. and Wu, J.: Study on a Poisson's equation solver based on deep learning technique, *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pp. 1–3 (2017).
[10] Raissi, M.: Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations, *J. Mach. Learn. Res.*, Vol. 19, No. 1, p. 932–955 (2018).
[11] Sirignano, J. and Spiliopoulos, K.: DGM: A deep learning algorithm for solving partial differential equations, *Journal of Computational Physics*, Vol. 375, pp. 1339 – 1364 (online), DOI: https://doi.org/10.1016/j.jcp.2018.08.029 (2018).
[12] Hochreiter, S. and Schmidhuber, J.: Long Short-Term Memory, *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780 (online), DOI: 10.1162/neco.1997.9.8.1735 (1997).
[13] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al.: Tensorflow: A system for large-scale machine learning, *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI16)*, pp. 265–283 (2016).
[14] Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Bengio, Y. and LeCun, Y., eds.), (online), available from ⟨http://arxiv.org/abs/1412.6980⟩ (2015).
[15] Yee, H. C.: A Class of High-Resolution Explicit and Implicit Shock Capturing Methods, *NASA TM-101088* (1989).