基于深度学习的间断有限元求解方法

常芸凡[1]
指导老师：杨顶辉教授[1]、贺茜君教授[2]

[1]清华大学数学科学系

[2]北京工商大学数学科学系

2021 年 9 月 29 日

研究背景

对于物理空间中具有不连续解的方程，间断有限元(DG)方法精度高、数值频散小且容易处理复杂边界问题 [Cockburn and Shu, 1989]，但是由于CFL条件限制，我们的步长不能选得过大，较小的步长极大的增加了我们的计算负担，导致DG方法的计算速度不高。

目前有很多利用神经网络来快速求解高维偏微分方程的思想 [Weinan and Yu, 2017] [Sirignano and Spiliopoulos, 2018]等，这些方法有效的克服了高维问题的维数灾难。我们希望将深度学习引入到DG方法中来，同时结合神经网络和DG的优势，从而实现对DG的加速。

DNN简介

Deep neural network(DNN)包含一系列的层，每一层都有若干个神经元与前层和后层神经元相连。神经元通过仿射变换和非线性激活函数连接。该universal approximation theorem(万能近似定理) [Cybenko, 1989]表明，DNN可以近似任何有限维空间的Borel可测函数。

一般而言DNN输入为 $\boldsymbol{z}^0 = (t, \boldsymbol{x})$ 输出为 $\mathcal{N}(t, \boldsymbol{x})$。每层神经网络之间的关系为：

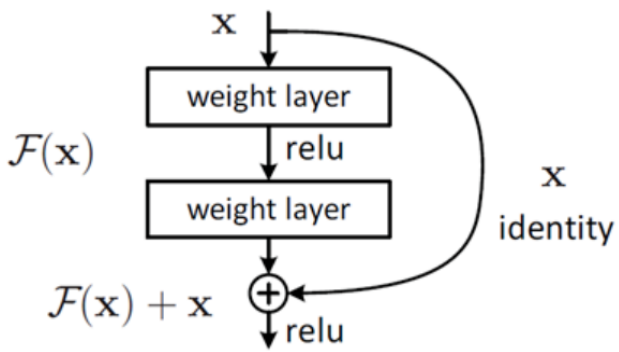
$$\begin{aligned}
&\boldsymbol{z}^0 = (t, \boldsymbol{x}) \quad \text{input} \\
&\boldsymbol{z}_k^{l+1} = \sigma_l \left(\mathbf{w}_k^{l+1} \cdot \boldsymbol{z}^l + b_k^l \right), \quad l = 0, 1, \cdots L-1, \quad 1 \le k \le m_{l+1} \\
&\mathcal{N}(t, \boldsymbol{x}) = \mathbf{w}^{L+1} z^L \quad \text{output}
\end{aligned}$$

其中m_l 是第l层的神经元个数，σ 是非线性激活函数。

两种不同的神经网络结构-ResNet

从经验来看，网络的深度对模型的性能至关重要，当增加网络层数后，网络可以进行更加复杂的特征模式的提取，所以当模型更深时理论上可以取得更好的结果(一些深度网络可表示的函数可能需要浅层网络指数级的隐藏单元才能表示 [Montúfar et al., 2014])。但是在实际应用中，深层网络存在着梯度消失或者爆炸的问题，这使得深度学习模型很难训练，效果也不好。

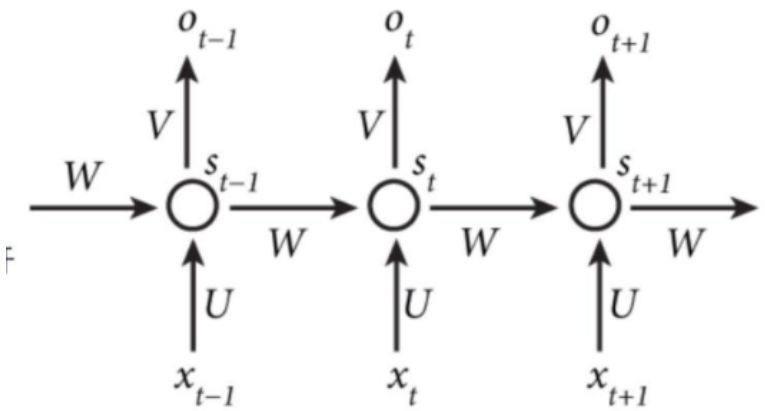
为此 [He et al., 2016]提出了残差学习(ResNet)来解决退化问题



这种网络类似于一种短路连接，保证了深层网络的效果至少不会比浅层网络效果差。

两种不同的神经网络结构-RNN

基础的神经网络只在层与层之间建立了权连接，RNN最大的不同之处就是在层之间的神经元之间也建立的权连接。这种网络结构可以更好的处理序列问题。目前该方法也有一些变种(LSTM,GRU等)



我们尝试的网络结构

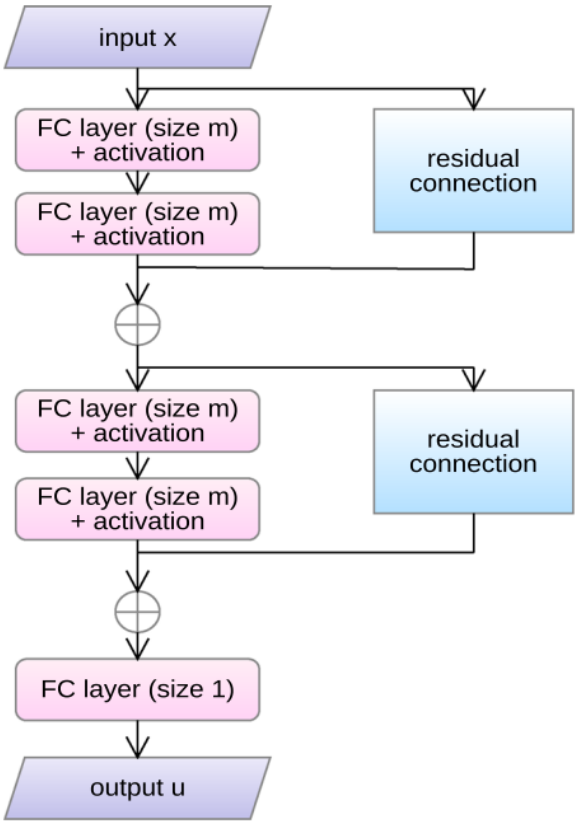


图 1: 残差网络

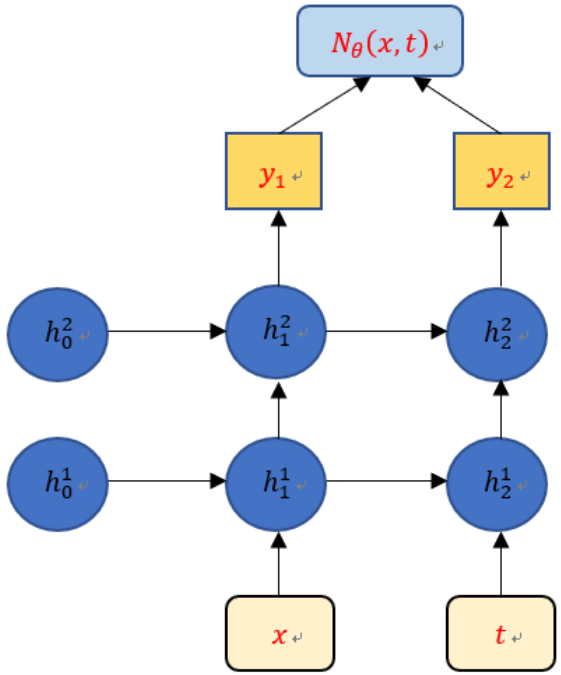


图 2: 循环网络

空间离散

我们考虑如下双曲问题：

$$u_t + \nabla_{\boldsymbol{x}} \cdot \boldsymbol{f}(u) = 0$$
$$u(0, \boldsymbol{x}) = u_0(\boldsymbol{x})$$

由变分法，我们可以将问题转化成

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\left(u_h(t,\boldsymbol{x}), v_h(\boldsymbol{x})\right)_{I_i} - \left(\boldsymbol{f}\left(u_h(t,\boldsymbol{x})\right), \nabla v_h(\boldsymbol{x})\right)_{I_i} \\ + \boldsymbol{f}\left(u_h(t,\boldsymbol{x})\right) \cdot \mathbf{n} v_h(\boldsymbol{x})\big)\big|_{\partial I_i} = 0 \end{aligned} \tag{1}$$

其中v_h为试验函数，I_i为小单元，$\mathbf{n}$为边界∂I_i的外法向量。1维的情形上式可以写成

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\left(u_h(t,x), v_h(x)\right)_{I_i} - \left(f\left(u_h(t,x)\right), v_h'(x)\right)_{I_i} \\ + \hat{f}_{i+1} v_h\left(x_{i+1}^-\right) - \hat{f}_i v_h\left(x_i^+\right) = 0 \end{aligned}$$

由于u_h的间断性，$\hat{f}_i$为数值通量，这是一个定义在界面上的单值函数，通常取决于界面两边的数值解的值。

我们选取Godunov通量：

$$\hat{f}^{\mathrm{God}}\left(u^-, u^+\right) = \begin{cases} \min_{u^- \le u \le u^+} f(u), & \text{if } u^- < u^+ \\ \max_{u^+ \le u \le u^-} f(u), & \text{if } u^+ < u^+ \end{cases}$$

时间离散

由于式(1)中含有时间导数项，我们需要离散相关的时间导数。我们引入时间步$0 = t_0 < t_1 < \cdots < t_N = T, t_{n+1} - t_n = \Delta t$

于是半离散格式变成

$$\left(\frac{u_h\left(t_{n+1}, \boldsymbol{x}\right) - u_h\left(t_n, \boldsymbol{x}\right)}{\Delta t}, v_h(\boldsymbol{x})\right)_{I_i} - \left(\boldsymbol{f}\left(u_h\left(t_n, \boldsymbol{x}\right)\right), \nabla v_h(\boldsymbol{x})\right)_{I_i} + \left(\boldsymbol{f}\left(u_h(t, \boldsymbol{x})\right), \mathbf{n} v_h(\boldsymbol{x})\right)|_{\partial I_i} = 0 \tag{2}$$

一维时我们可以写成

$$\left(\frac{u_h\left(t_{n+1}, x\right) - u_h\left(t_n, x\right)}{\Delta t}, v_h(x)\right)_{I_i} - \left(f\left(u_h\left(t_n, x\right)\right), v_h'(x)\right)_{I_i} + \hat{f}_{i+1} v_h\left(x_{i+1}^-\right) - \hat{f}_i v_h\left(x_i^+\right) = 0 \tag{3}$$

我们选取基函数φ_i^j为单元I_i上的j阶正交Legendre多项式(在其他单元上该函数值为0)，则DG方法最终的解可以写成

$$u_h(t,\boldsymbol{x}) = \sum_{j=0}^{K}\sum_{i=0}^{M} C_i^j(t)\,\varphi_i^j(x)$$

我们可以利用神经网络来近似系数C_i^j，具体而言，我们把解写成如下形式 [Chen et al., 2021]:

$$u_{h,\theta}(t,\boldsymbol{x}) = \sum_{j=0}^{K}\sum_{i=0}^{M} \mathcal{N}_\theta^j\left(t, x_{i+1/2}\right)\varphi_i^j(x) \tag{4}$$

这里我们一共用了$K+1$个神经网络$\mathcal{N}_\theta^0, \mathcal{N}_\theta^1, \cdots, \mathcal{N}_\theta^K$，每个神经网络的输入是时间$t$和空间坐标$x$，我们希望神经网络的输出$\mathcal{N}_\theta^j\left(t, x_{i+1/2}\right) = C_i^j(t)$

一维情形推导

我们把神经网络近似的DG格式(4)代入到方程(3)中，并将试验函数选为$\varphi_i^j(x)$，可以得到

$$L_{i,j,n} := \frac{\mathcal{N}_\theta^j\left(t_{n+1}, x_{i+\frac{1}{2}}\right) - \mathcal{N}_\theta^j\left(t_n, x_{i+\frac{1}{2}}\right)}{\Delta t} \cdot (\varphi_i^j, \varphi_i^j)_{I_i}$$
$$- \left(f\left(u_{h,\theta}\left(t_n, x\right)\right), \frac{\mathrm{d}\varphi_i^j(x)}{\mathrm{d}x}\right)_{I_i} + \hat{f}_{i+1}\varphi_i^j\left(x_{i+1}^-\right) - \hat{f}_i\varphi_i^j\left(x_i^+\right) = 0. \tag{5}$$

从而我们的DNN训练的损失函数可以设为

$$\mathcal{L}(\theta) = \left(h\Delta t \sum_{i,j,n} L_{i,j,n}^2\right)^{1/2} \tag{6}$$

一维情形推导

实际操作中，由于我们一共有$K+1$互不相关的神经网络$\mathcal{N}_\theta^0, \mathcal{N}_\theta^1, \cdots, \mathcal{N}_\theta^K$，从而我们可以将式(6)分成$K+1$个不相关的损失函数$\mathcal{L}^j(\theta) = \left(h\Delta t \sum_{i,n} L_{i,j,n}^2\right)^{1/2}$，这样我们的$K+1$个神经网络几乎可以同时训练，很容易实现并行操作。

常见的边界条件有:

- Dirichlet:

$$u(t, \boldsymbol{x}) = g(t, \boldsymbol{x}) \quad \boldsymbol{x} \in \partial D$$

- Neumann:

$$\frac{\partial u(t, \boldsymbol{x})}{\partial \boldsymbol{\nu}} = g(t, \boldsymbol{x}) \quad \boldsymbol{x} \in \partial D$$

- 初值条件:

$$u(0, \boldsymbol{x}) = u_0(\boldsymbol{x})$$

两种处理方法

- 最直接的想法是把边界条件直接加入到损失函数里进行训练，直接让其收敛于0，比如Dirichlet边界条件我们可以在损失函数中加上$\lambda\|u-g\|_{\partial D}^2$一项，其中$\lambda$是惩罚因子(超参数，可自行设置)
- 另一种思路是构造DNN网络使得其精确的满足边界条件。例如对于初值问题 $u(0,\boldsymbol{x})=u_0(\boldsymbol{x})$, 我们可以构造表达式

$$u_\theta(t,\boldsymbol{x})=t\mathcal{N}_\theta(t,\boldsymbol{x})+u_0(\boldsymbol{x})$$

小批量训练

我们的神经网络$\mathcal{N}_\theta^j$训练的是所有j阶Legendre多项式前的系数，并且相比于传统DG方法$u_h(t,\boldsymbol{x})=\sum_{j=0}^{K}\sum_{i=0}^{M}C_i^j(t)\varphi_i^j(x)$，当阶数$j$固定的时候，$C_i^j(t)$在每个网格上是相同的值，相当于是离散的点。

但是神经网络训练的$\mathcal{N}_\theta^j(t,x)$实际上关于$t$和$x$都是可连续计算的。相比于传统DG方法每个时间步都需要计算所有网格点前基函数的系数，神经网络的方法却不需要每次遍历所有网格，我们每次在$(t,\boldsymbol{x})$组成的网格点中随机抽样(如10000个点)进行训练，这样既减少了我们每次训练的数量，又可知当空间维数变成3维时，由于我们每次依然时随机抽样10000个点，每次训练的计算量其实并不会增加。

并行处理

我们每个神经网络的损失函数$\mathcal{L}^j(\theta)$基本不相关，所以我们可以相对独立的同时训练每个网络，这意味着增加阶数并不会对我们的计算速度产生很多影响。

从数学推导中我们也可以发现，DG格式中，采用n阶正交多项式逼近和采用$n+1$阶正交多项式逼近，其前n阶的系数是相同的。换言之，若$K+1$阶DG格式的解是

$$u_h(t,\boldsymbol{x}) = \sum_{j=0}^{K+1}\sum_{i=0}^{M} C_i^j(t)\varphi_i^j(x)$$

则K阶DG格式的解就是

$$u_h(t,\boldsymbol{x}) = \sum_{j=0}^{K}\sum_{i=0}^{M} C_i^j(t)\varphi_i^j(x)$$

这意味着本身DG格式不同阶数的Legendre多项式前的系数是独立的，而我们的方法正是可以独立的同时近似这些系数。

优势

总得来说，相比于传统方法，神经网络近似有如下的优势

- 如果增加空间维度，由于我们每次训练都是抽样小批量训练，所以每次训练的计算量并不会增加(但是训练次数可能会增加);
- 如果增加近似阶数(使用更高阶的Legendre多项式)，我们可以采用并行处理，同时训练多个神经网络，也不会降低我们的计算速度。

但是我们的方法也只是**近似**DG格式中需要计算的$C_i^j(t)$，所以肯定达不到DG方法的精度。但是神经网络的优势在于，它可以在很少的训练次数内达到DG方法95%的精度，但是如果需要达到99%以上，训练的次数是需要指数式增长的。

Burgers' equation

我们考虑如下Burgers方程(解为间断的):

$$u_t + \left(\frac{u^2}{2}\right)_x = 0$$

初值条件为

$$u(0,x) = \begin{cases} 1 & ,x < 0 \\ 0 & ,x > 0 \end{cases}$$

该方程有解析解: $u(t,x) = \begin{cases} 1 & ,x < t/2 \\ 0 & ,x > t/2 \end{cases}$

神经网络近似

0阶数值解我们如下构造：

$$u_\theta(t,x)=\begin{cases}\mathcal{N}_\theta\left(t,x_{i+\frac{1}{2}}\right)\varphi_i(x) & t>0 \quad x\in[x_i,x_{i+1})\\ u\left(0,x_{i+\frac{1}{2}}\right) & t=0\end{cases}$$

其中$\varphi_i(x)=\begin{cases}1 & x\in[x_i,x_{i+1})\\ 0 & otherwise\end{cases}$ 按照式(5),我们训练的损失函数为

$$\mathcal{L}_\theta=\sum_{i,n}\left(\frac{\mathcal{N}_\theta\left(t_{n+1},x_{i+\frac{1}{2}}\right)-\mathcal{N}_\theta\left(t_n,x_{i+\frac{1}{2}}\right)}{\Delta t}\cdot h+\hat{f}_{i+1}-\hat{f}_i\right)^2$$

1阶数值解我们如下构造：

$$u_\theta(t,x)=\begin{cases}\mathcal{N}_\theta^0\left(t,x_{i+\frac{1}{2}}\right)\varphi_i^0(x)+t\mathcal{N}_\theta^1\left(t,x_{i+\frac{1}{2}}\right)\varphi_i^1(x) & t>0\\ u\left(0,x_{i+\frac{1}{2}}\right) & t=0\end{cases}$$

其

中$\varphi_i^0(x)=\begin{cases}1 & x\in[x_i,x_{i+1})\\ 0 & otherwise\end{cases}$，$\varphi_i^1(x)=\begin{cases}x-x_{i+\frac{1}{2}} & x\in[x_i,x_{i+1})\\ 0 & otherwise\end{cases}$

我们记$U_i^0(t)=\mathcal{N}_\theta^0\left(t,x_{i+\frac{1}{2}}\right)$，$U_i^1(t)=t\mathcal{N}_\theta^1\left(t,x_{i+\frac{1}{2}}\right)$

则我们两个神经网络的损失函数分别为：

$$\mathcal{L}_\theta^0=\sum_{i,n}\left(\frac{U_i^0(t_{n+1})-U_i^0(t_n)}{\Delta t}\cdot h+\hat{f}_{i+1}-\hat{f}_i\right)^2$$

$$\mathcal{L}_\theta^1=\sum_{i,n}\left(\frac{U_i^1(t_{n+1})-U_i^1(t_n)}{\Delta t}\cdot\frac{h^3}{12}-\frac{u_\theta^2 h}{2}+\hat{f}_{i+1}\frac{h}{2}+\hat{f}_i\frac{h}{2}\right)^2$$

0阶格式的结果

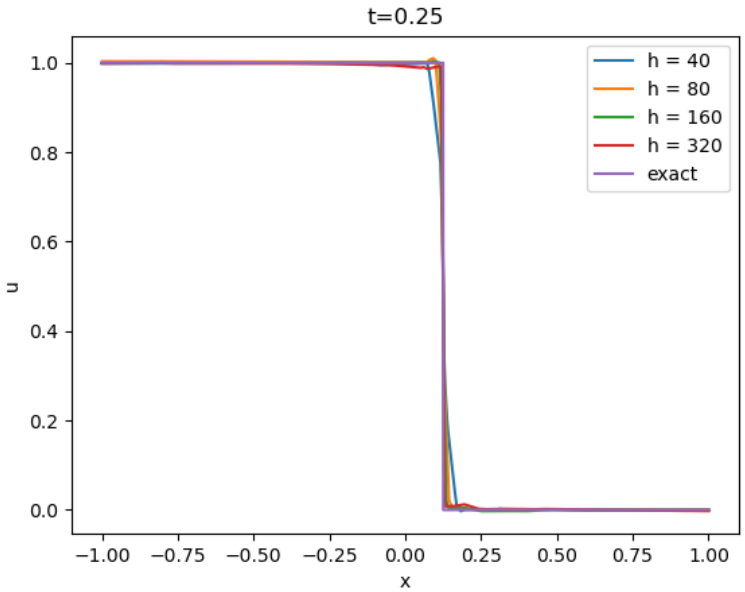


图 3: $t = 0.25$

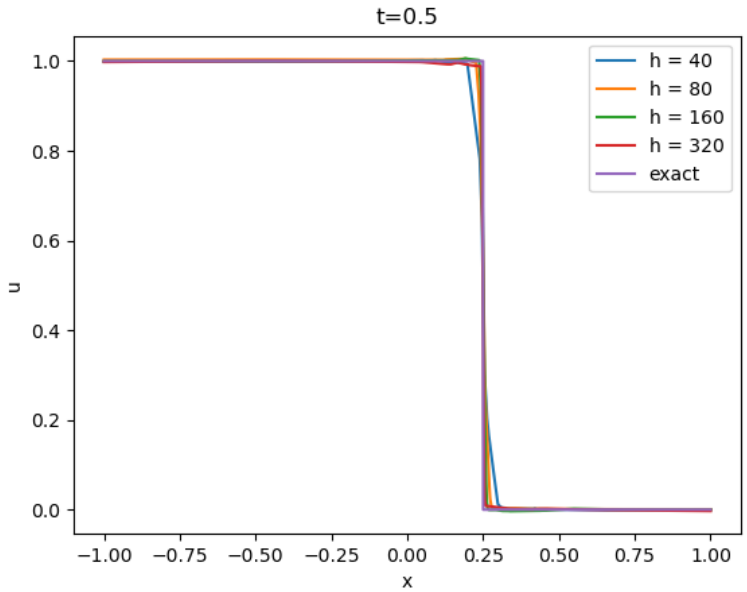


图 4: $t = 0.5$

0阶格式的结果

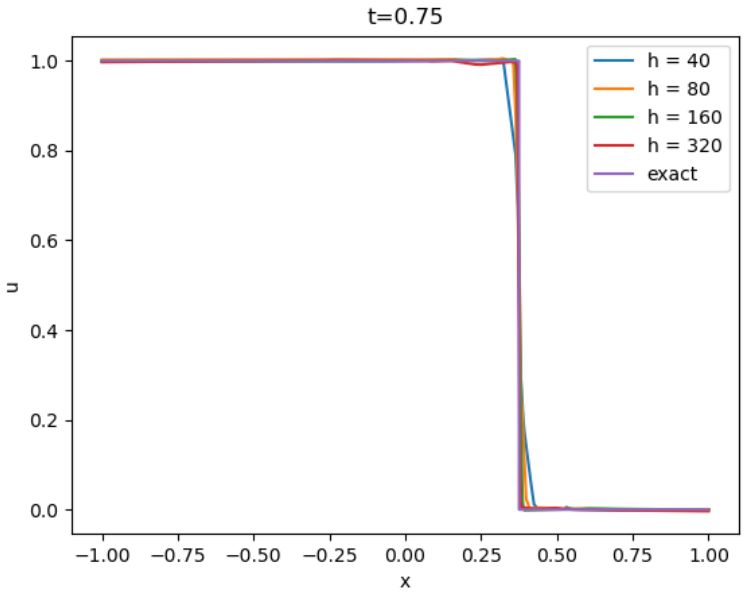


图 5: $t = 0.75$

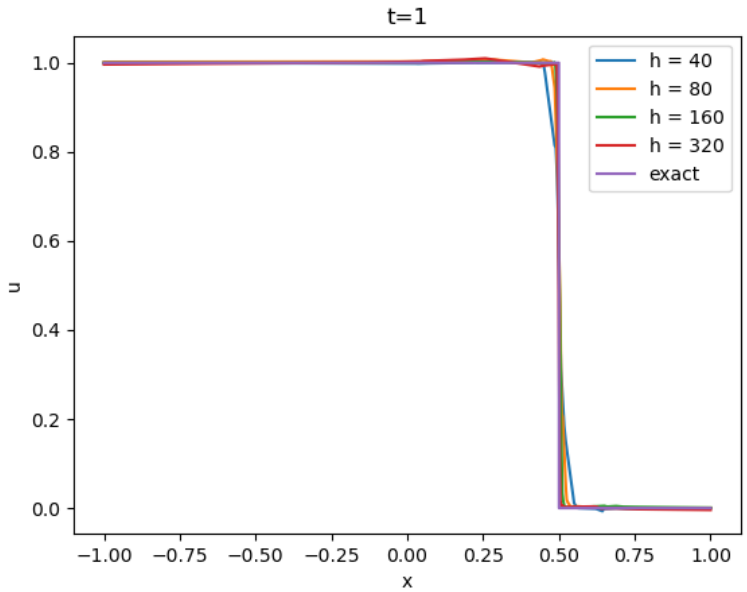


图 6: $t = 1.0$

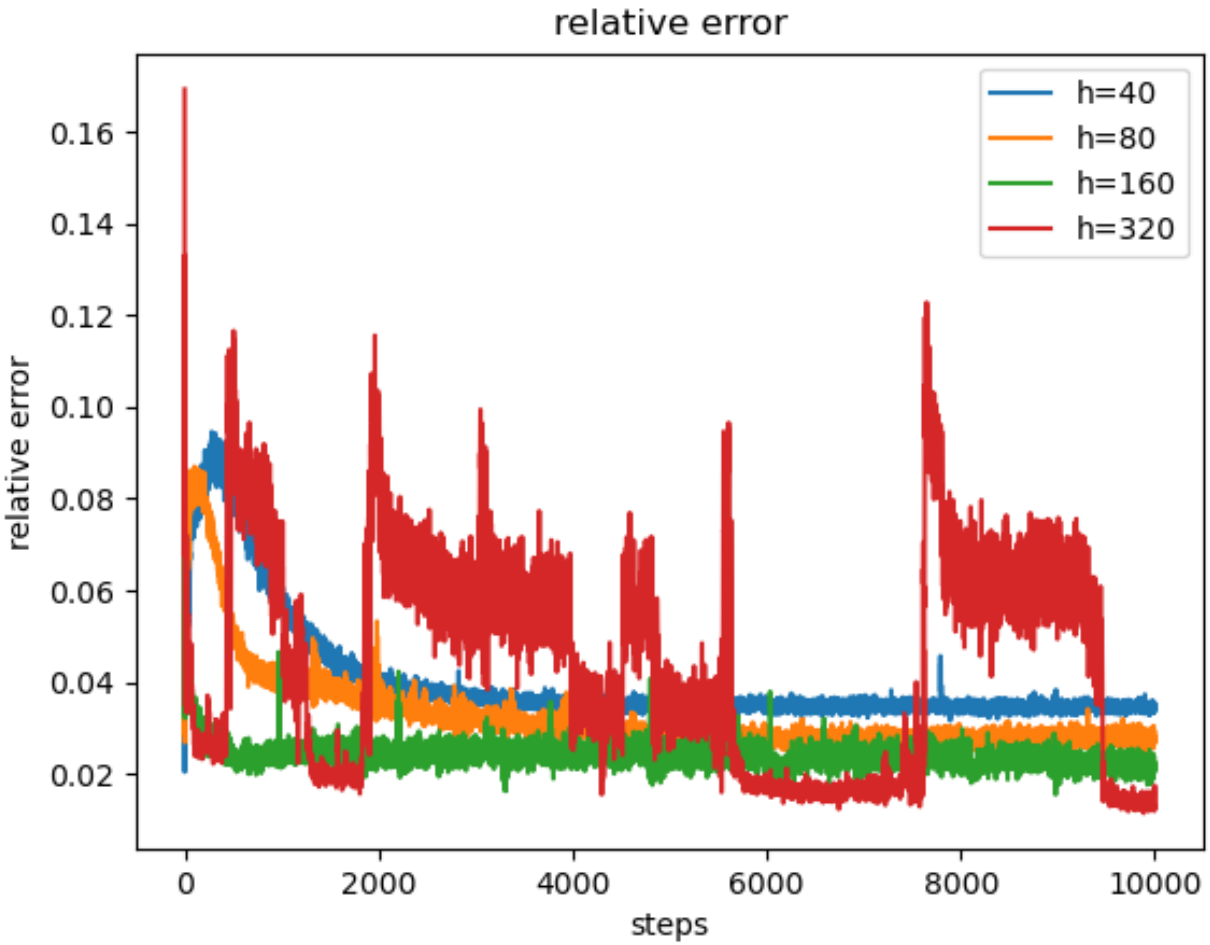

relative error
h=40
h=80
h=160
h=320
relative error
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0
2000
4000
6000
8000
10000
steps

图 7: 相对误差趋势

1阶格式的数值解

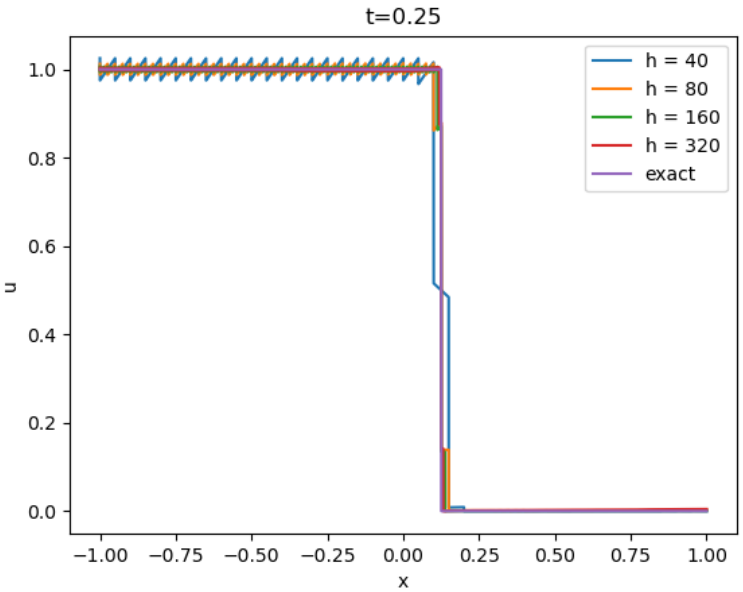


图 8: $t = 0.25$

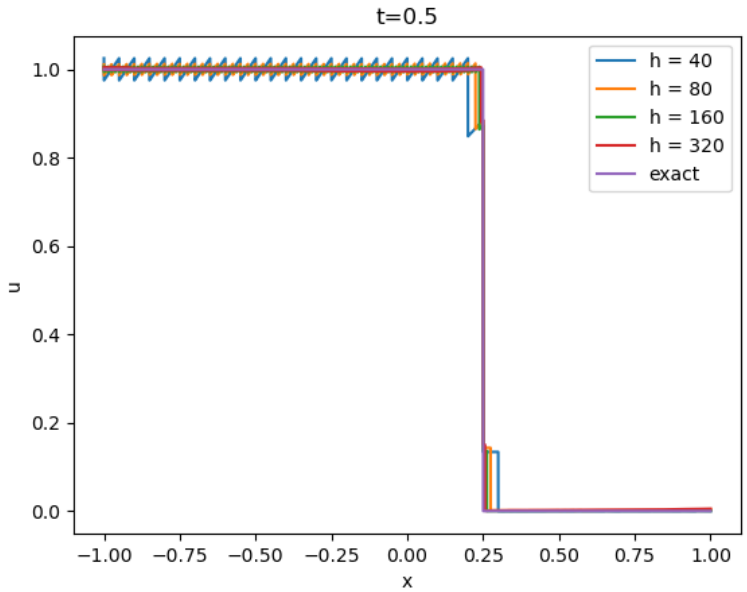


图 9: $t = 0.5$

1阶格式的数值解

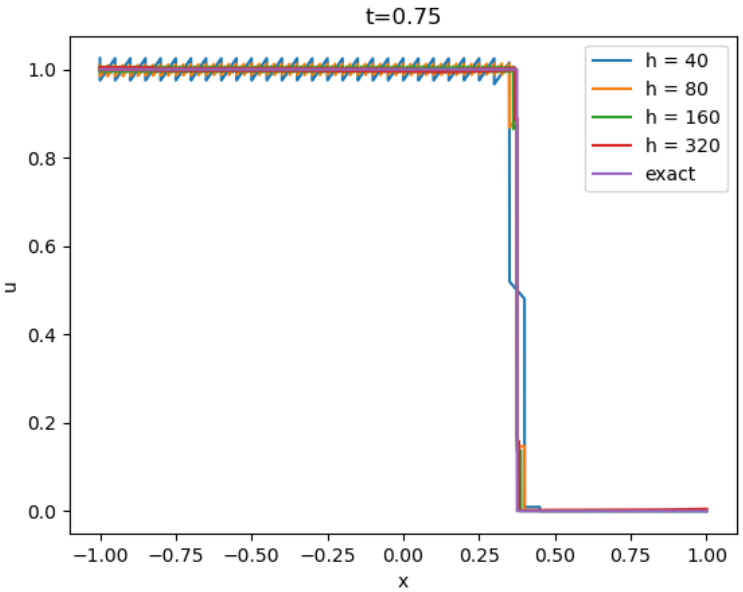


图 10: $t = 0.75$

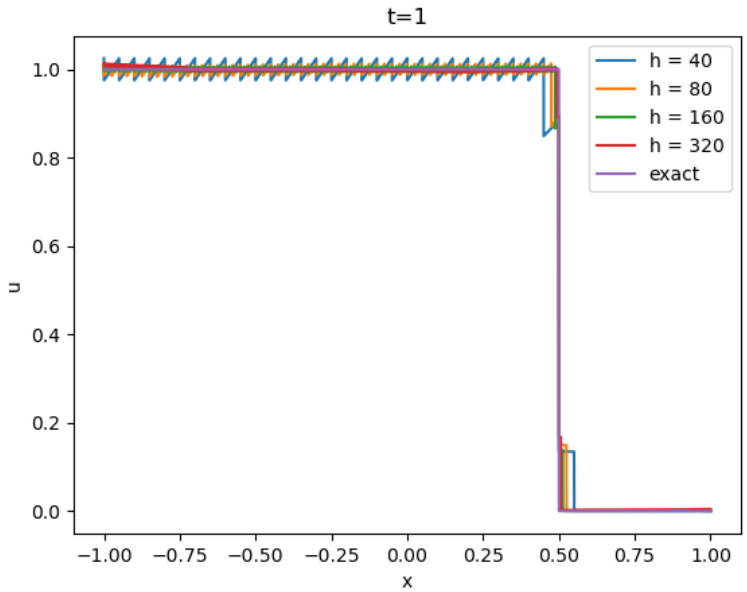


图 11: $t = 1.0$

去掉最大网格

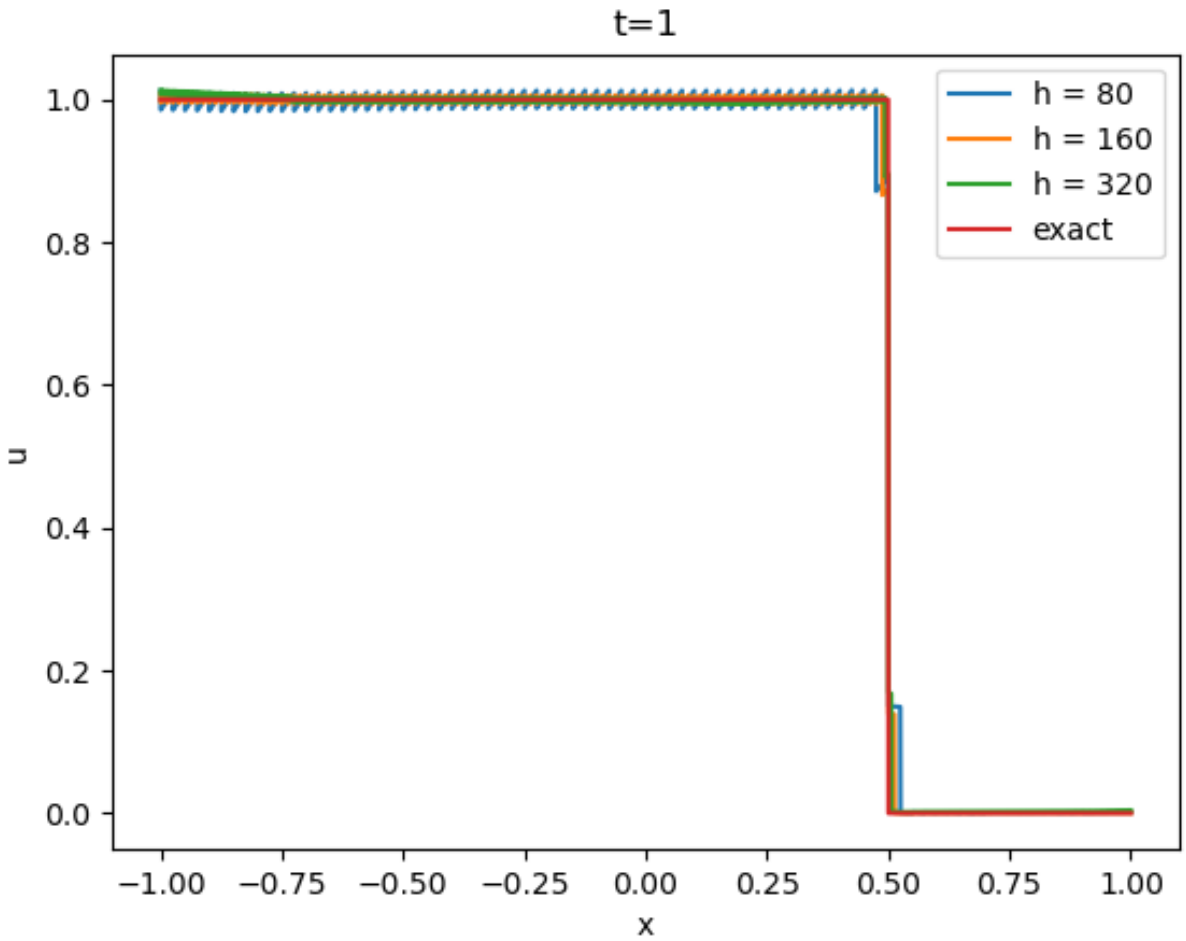


图 12: 去掉h=40的网格

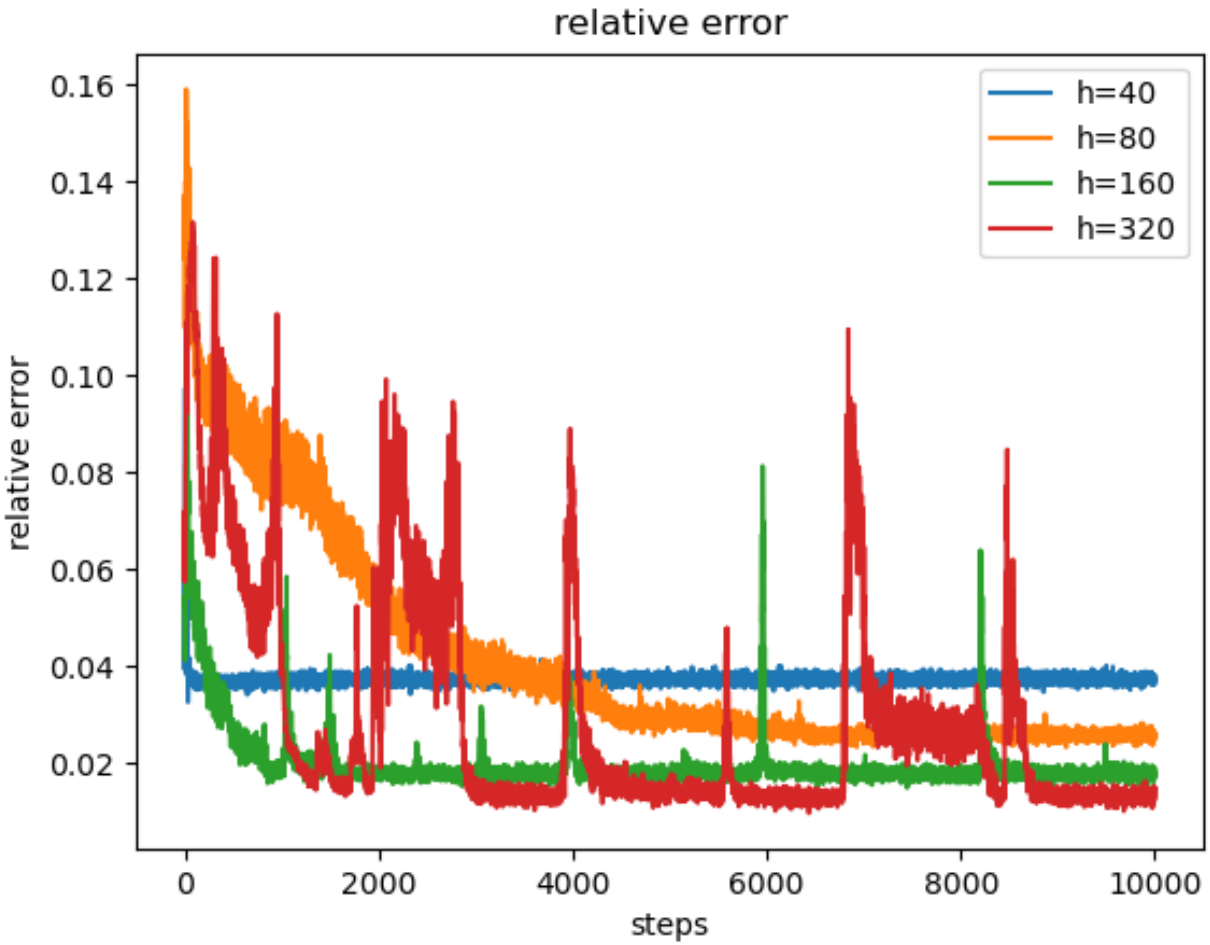

relative error
h=40
h=80
h=160
h=320
relative error
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0
2000
4000
6000
8000
10000
steps

图 13: 相对误差趋势

下一步工作安排

1. 在声波方程中尝试该方法；
2. 至少做到空间2阶格式；
3. 向二维、三维空间尝试

参考文献 I

[Chen et al., 2021] Chen, J., Jin, S., and Lyu, L. (2021).
A deep learning based discontinuous galerkin method for hyperbolic equations with discontinuous solutions and random uncertainties.
arXiv preprint arXiv:2107.01127.

[Cockburn and Shu, 1989] Cockburn, B. and Shu, C.-W. (1989).
Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws ii: General framework.
Mathematics of Computation, 52(186):411–435.

[Cybenko, 1989] Cybenko, G. (1989).
Approximation by superpositions of a sigmoidal function.
Mathematics of control, signals and systems, 2(4):303–314.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016).
Deep residual learning for image recognition.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Montúfar et al., 2014] Montúfar, G., Pascanu, R., Cho, K., and Bengio, Y. (2014).
On the number of linear regions of deep neural networks.
arXiv preprint arXiv:1402.1869.

参考文献 II

[Sirignano and Spiliopoulos, 2018] Sirignano, J. and Spiliopoulos, K. (2018).
Dgm: A deep learning algorithm for solving partial differential equations.
Journal of Computational Physics, 375:1339–1364.

[Weinan and Yu, 2017] Weinan, E. and Yu, T. (2017).
The deep ritz method: A deep learning-based numerical algorithm for solving variational problems.
Communications in Mathematics and Statistics, 6:1–12.

Thanks!