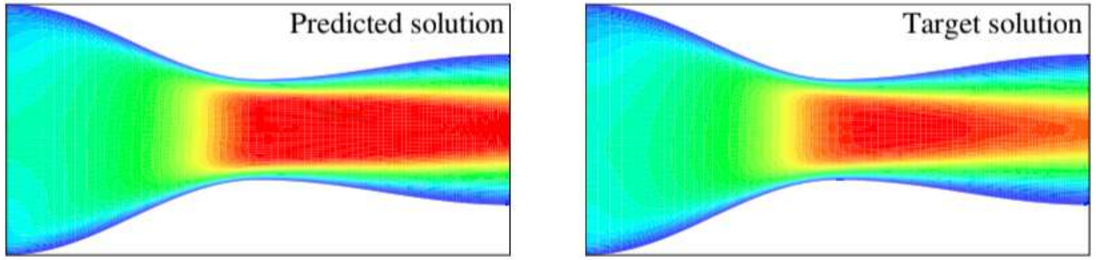
INTERPLAY OF MACHINE LEARNING AND TRADITIONAL NUMERICAL METHODS WITH COMPUTATIONAL FLUID DYNAMIC (CFD) PROBLEMS

HO, Cheuk Hin
AU, Tsz Nga



BACKGOUND

- In science or engineering, time-dependent problems or PDEs are common
- Classical examples ：

Navier –Strokes equation

$$\frac{\partial \mathbf{u}}{\partial t} = -\nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \frac{1}{Re}\nabla^2 \mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}$$

$$\nabla \cdot \mathbf{u} = 0,$$

Burger’s Equation

$$\begin{cases} \phi(x;\boldsymbol{\mu})\dfrac{\partial \phi(x;\boldsymbol{\mu})}{\partial x} - \dfrac{\partial \phi^2(x;\boldsymbol{\mu})}{\partial x^2} = s(x;\boldsymbol{\mu}) & -1 \leq x \leq 1 \\ \phi\left(x = \pm 1;\boldsymbol{\mu}\right) = 0 & \end{cases}$$

Model fluid flow, which can be used in aviation and weather predictions

BACKGROUND

- High-fidelity solutions of these problem by traditional discretization method
- However, it involves solving system of equations with huge dimension or distinct value of parameter
  - Known as the curse of dimensionality

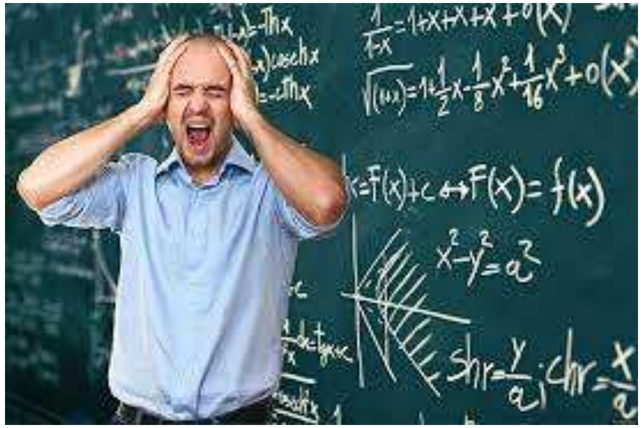
  $\Rightarrow$ huge dimension

  $\Rightarrow$ amount of data needed increases

  $\Rightarrow$ need to generate more high−fidelity solutions

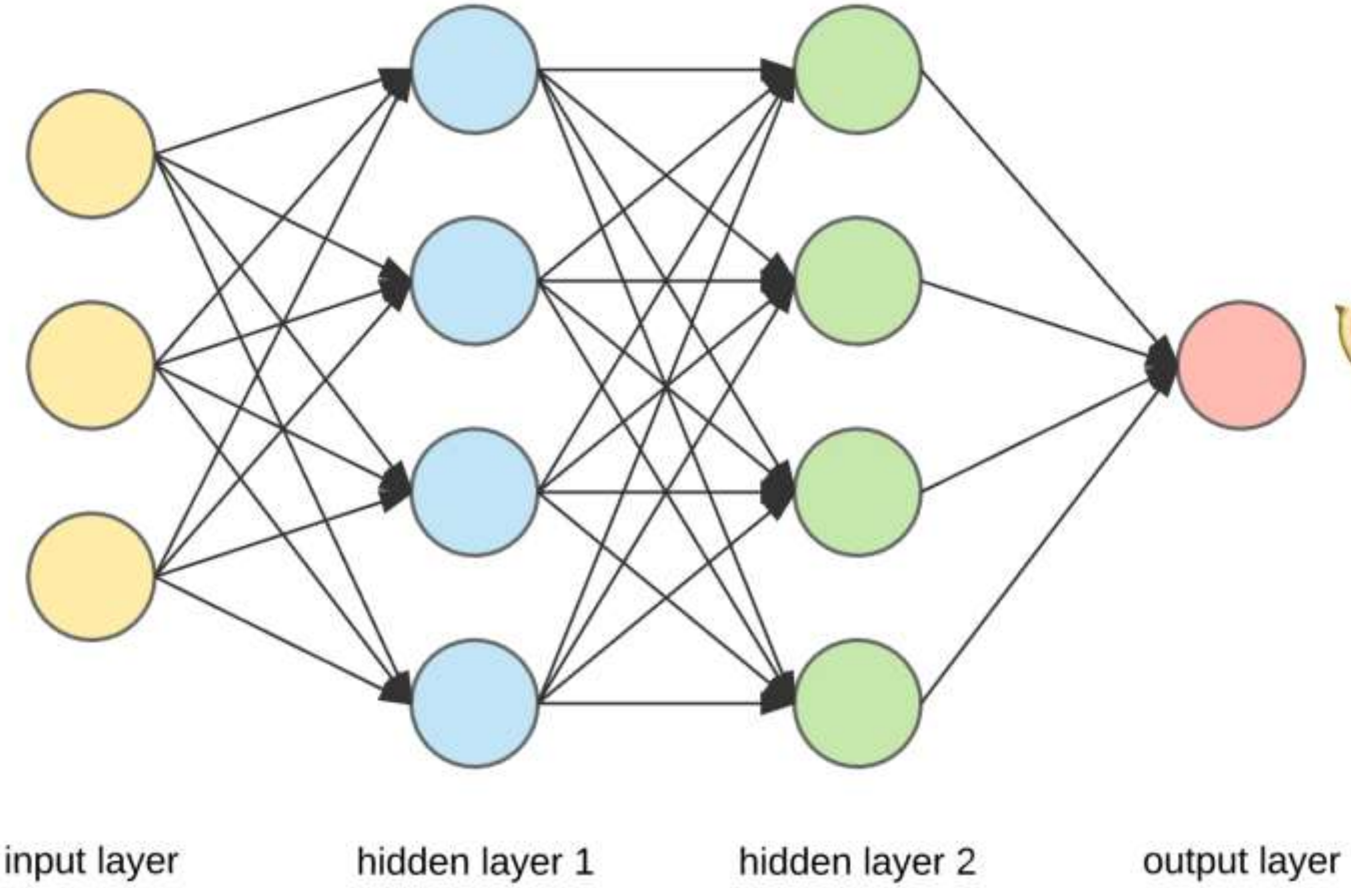
  $\Rightarrow$ computational time and cost increase!

  $\Rightarrow \mathit{Computational\ cost} = O(m^2)$

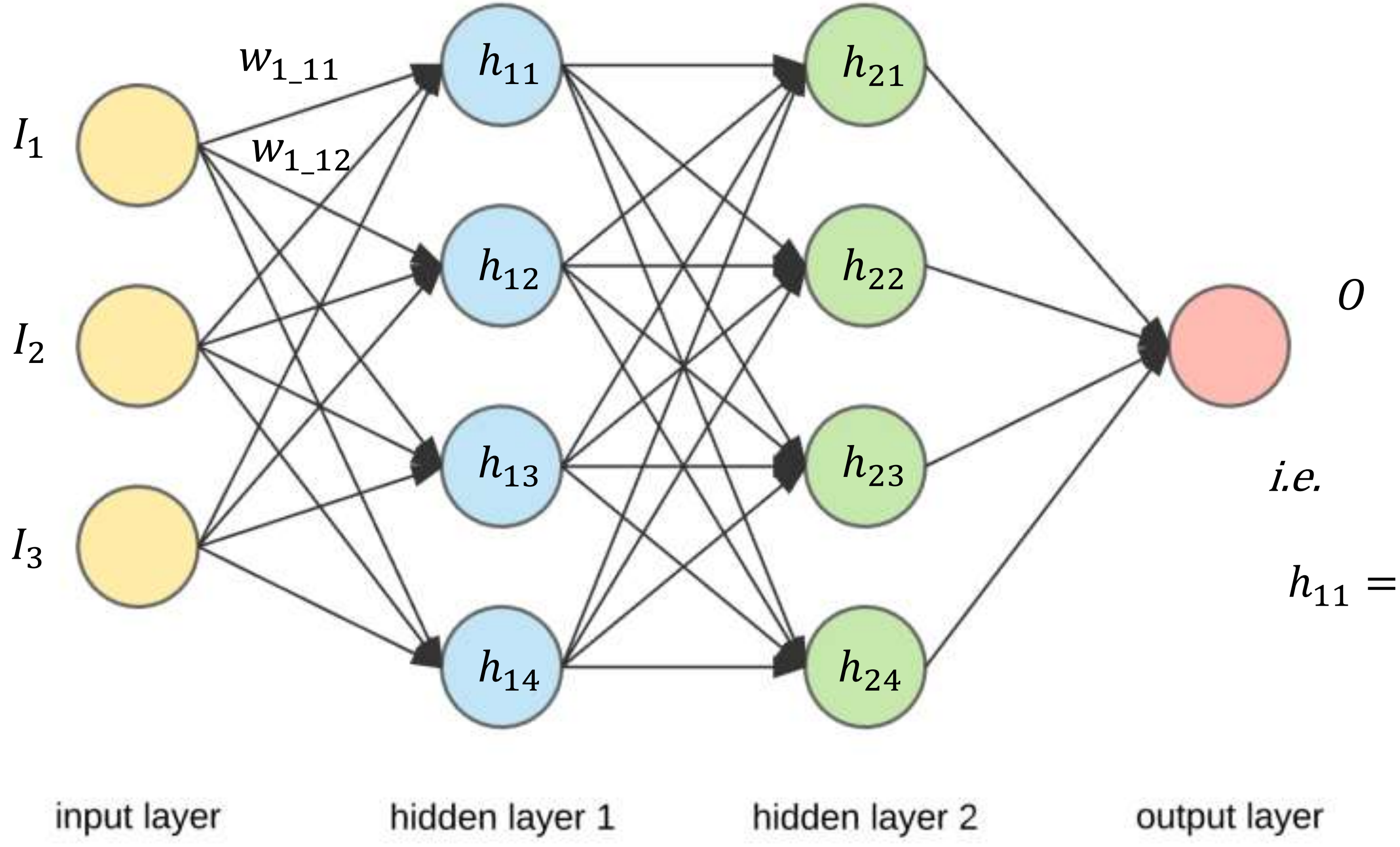


- This problem can be solved by using Model Reduction and Deep Neural Network!!

NEURAL NETWORK



NEURAL NETWORK



i.e.

$$h_{11} = acticvation(\sum_{k=1}^{3} I_k * w_{k\_11})$$

ROLES OF MACHINE LEARNING IN CFD PROBLEMS

- Use Neural Network to substitute some time consuming process in traditional numerical methods

  e.g. approximate non-linear term in non-linear PDE and convert the problem to solving linear system

  (Dmitrii Kochkov, Jamie A. Smith et al, 2021)

- Do prediction directly through the neural network

  e.g. forecasting weather and climate processes

- Act as supporting tools for solving problems

  e.g. detect trouble cell with large derivatives when solving PDEs

  (Deep Ray, Jan S. Hesthaven 2017)

AN EXAMPLE : INTERPLAY OF REDUCED ORDER MODELING AND MACHINE LEARNING

Assume : $\phi(x) = (1 + \mu_1 x) sin(-\mu_2 x/3)(x^2 - 1)$

Consider the following "testing model" for one-dimensional Burger's equation :

$$\phi(x;\mu)\frac{\partial\phi(x;\mu)}{\partial x} - \frac{\partial\phi^2(x;\mu)}{\partial x^2} = s(x;\mu)$$

$$\phi(-1;\mu) = \phi(1;\mu) = 0$$

where $\mu = (\mu_1, \mu_2) \in [1,10] \times [1,10]$ and $s(x;\mu)$ can be determined by the initially assumed $\phi(x;\mu)$ analytically.

PROPER ORTHOGONAL DECOMPOSITION

Obtain an optimal set of orthonormal basis V for the solution space L:

$$L = \{\phi(x;\mu) |\, \mu = (\mu_1, \mu_2) \in [1,10] \times [1,10]\}$$

Let $\mathrm{P} = \{\mu^{(1)}, \mu^{(2)}, \dots, \mu^{(N)}\}$ and

$$S = [\phi_h(\mu^{(1)}) | \phi_h(\mu^{(2)}) \dots |\ \phi_h(\mu^{(n)})]$$

be a matrix consist of corresponding high-fidelity solution.

i.e. V is the solution of $\min_{v \in \mathbb{R}^{N \times m}} ||S - VV^t S||_F$ with $VV^t = I$

PROPER ORTHOGONAL DECOMPOSITION

By Eckart-Young theorem, V can be obtained by Singular Value Decomposition:

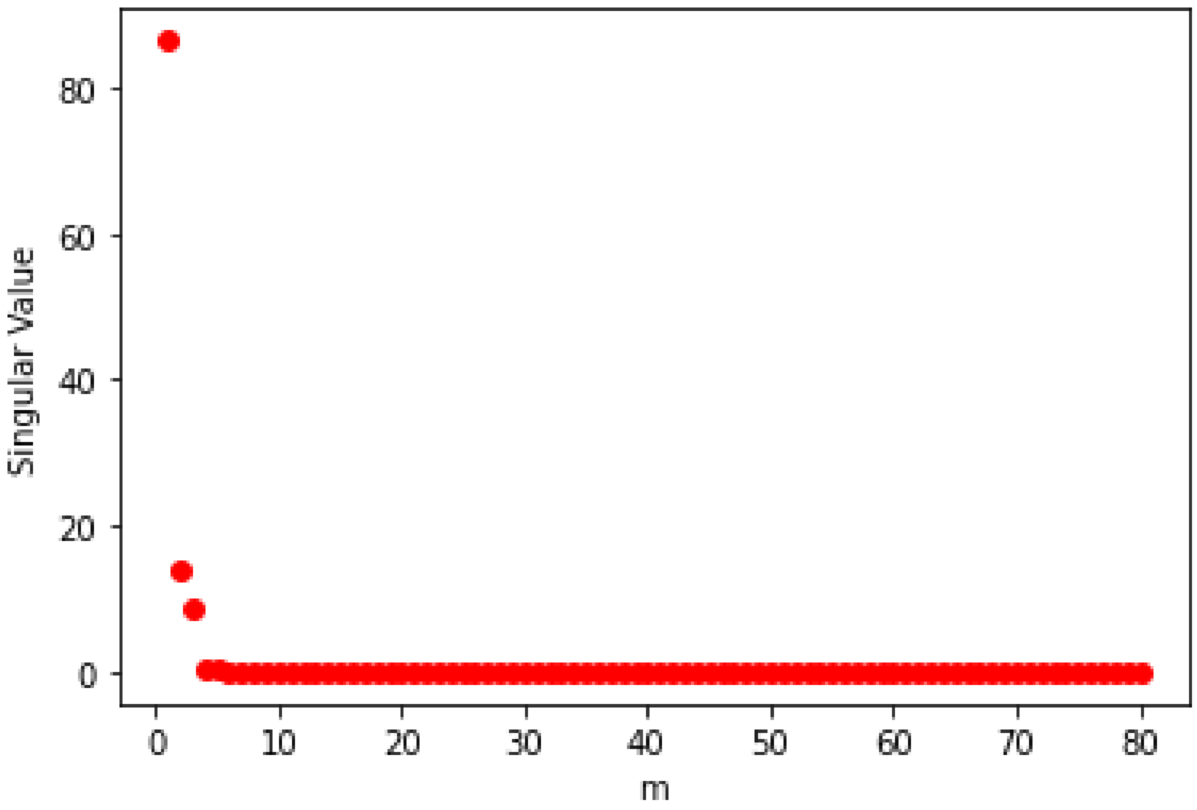
$$S = U_s \Sigma_s V_s$$

where $\Sigma_s = diag(\sigma_i)$, sorted in descending order

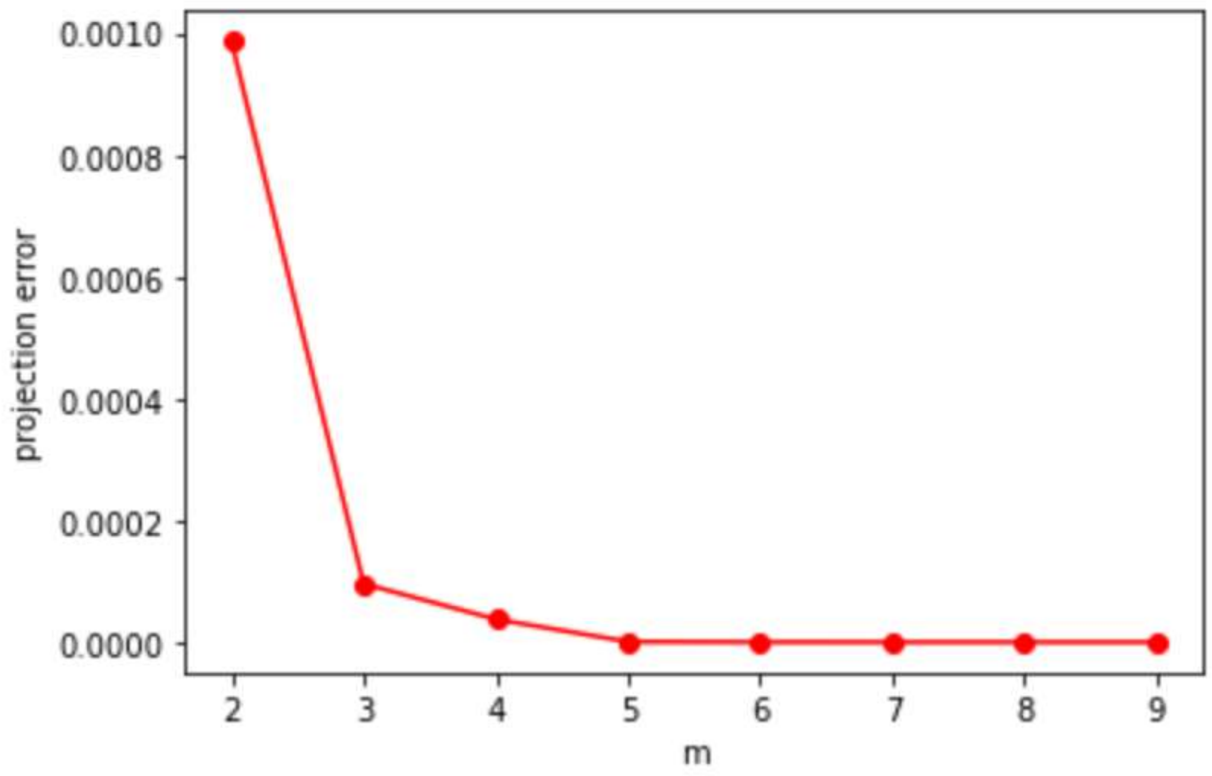
Therefore, we can obtain first m column of U_s as the approximation to the solution base with arbitrary accuracy. i.e.

$$V = U_s[0{:}\, m, :]$$

SINGULAR VALUE OF m^{th} VECTOR



NUMBER OF BASIS USED AND ERROR WITH HIGH FIDELITY SOLUTION AFTER PROJECTION BY V



DATA PREPARATION

After obtaining V, we want to train a neural network $\mathfrak{N}$ such that given any (μ_1, μ_2),

$$\mathfrak{N}: (\mu_1, \mu_2) \rightarrow \alpha \in \mathbb{R}^m$$

with $V\alpha \approx \phi$, where ϕ is the corresponding solution to (μ_1, μ_2).

Therefore, since $VV^t = I$, we can prepare the training and testing data by :

$$\alpha_h = V\phi_h$$

MODEL STRUCTURE

```
model.add(tf.keras.Input(shape=(2)))
model.add(tf.keras.layers.Dense(20,activation='swish',activity_regularizer=regularizers.l2(1e-4)))
model.add(tf.keras.layers.Dense(20,activation='swish',activity_regularizer=regularizers.l2(1e-4)))
model.add(tf.keras.layers.Dense(20,activation='swish',activity_regularizer=regularizers.l2(1e-4)))
model.add(tf.keras.layers.Dense(6,activation='linear',activity_regularizer=regularizers.l2(1e-4)))
```

$swish = \frac{x}{1+e^{-x}}$: a non-linear function interpolate between ReLU and linear function

TRAINING AND TESTING

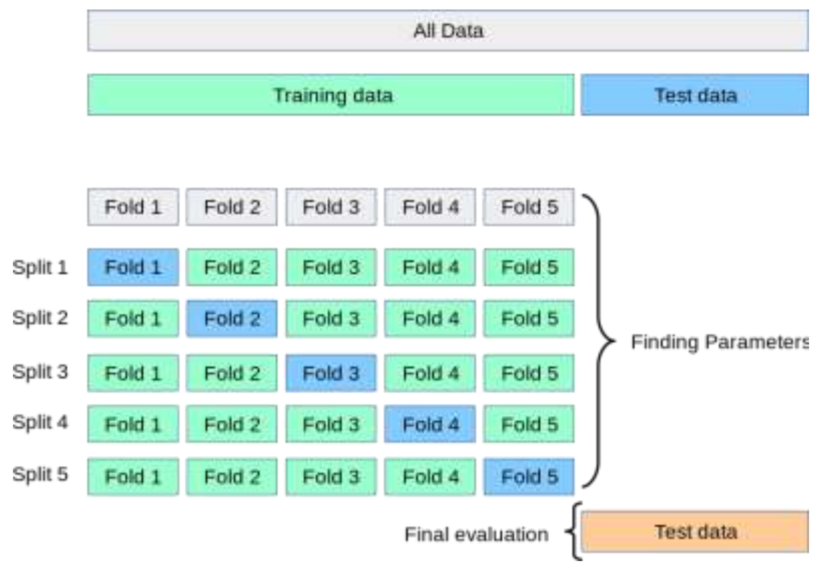
- Number of training data : 448
- Batch size = 1
- Use 101 * 101 uniformly distributed (μ_1, μ_2) for testing
- Optimizer : Adam
- Early stop on validation loss with patience = 6

```
Epoch 25/20000
448/448 [==============================] - 1s 1ms/step - loss: 0.0929 - mean_squared_error: 0.0733 - val_loss: 0.1400
- val_mean_squared_error: 0.1203
```

- % error with analytic solution on testing set = 7.0789%

CROSS VALIDATION

- 10-Fold cross validation
- Use different subset of data for testing
- save the model with best performance
- Error : 7.0789% -> 4.8624%



RECENT WORK

Use Burger's Equation as testing question,

$$u_{xx} + u_x u = 0$$

$$u(-1) = a \qquad u(1) = b$$

where $-1 \leq x \leq 1$ and *a, b* are some constants in [-1, 1].

- Instead of discretize the whole domain and use Neural Network to find the discretized solution
- Cut the domain into n interval (coarse grid) with nodal points $[x_i, x_{i+1}]$ for $i = 0, 1, \ldots, n-1$
  - Let φ be the local solution in $[x_i, x_{i+1}]$ for $i = 0,1, \ldots, n-1$
  - Let $u_i = u(x_i)$ for $i = 0,1, \ldots, n-1$

RECENT WORK

Left derivative

Right derivative

Local residual

Let

$$F = \sum_{i=1}^{n-1}\left(\frac{\partial \varphi_{i-1}}{\partial x} - \frac{\partial \varphi_i}{\partial x}\right)^2\Bigg|_{x=x_i} + \sum_{i=1}^{n-1}\left(\frac{\varphi_{i\_1} - 2u_i + \varphi(x_{i-1\_n^2-1})}{h^2} + u_i\,\frac{\varphi_{i\_1} - \varphi(x_{i-1\_n^2-1})}{2h}\right)^2$$

- Notice that all terms depends on u_i s, i.e. $F = F(u_0, u_1, \dots, u_{n-1})$
- Claim: Minimizer $(u_0, u_1, \dots, u_{n-1})$ of Function F is a high-fidelity solution of the system.
- Reason :
- First term vanishes if left derivative of φ_{i-1} and right derivative of φ_i at x_i is close
- Second term vanishes if $(u_0, u_1, \dots, u_{n-1})$ is indeed a solution and φ_i are solved accurately

**Indeed we have checked the necessity

RECENT WORK

Let $\psi_{i,j} = \frac{\partial \varphi_i}{\partial u_j}$

- By calculation, we need to find out the following terms:

$$(1)\frac{\partial \varphi_i}{\partial x}(x_i) \quad (2)\frac{\partial \varphi_i}{\partial x}(x_{i+1}) \quad (3)\frac{\partial \psi_{i,i}}{\partial x}(x_i) \quad (4)\frac{\partial \psi_{i,i}}{\partial x}(x_{i+1})$$

$$(5)\frac{\partial \psi_{i,i+1}}{\partial x}(x_i) \quad (6)\frac{\partial \psi_{i,i+1}}{\partial x}(x_{i+1}) \quad (7)\varphi(x_{i\_1}) \quad (8)\varphi(x_{i_{n^2}-1})$$

$$(9)\frac{\partial \varphi(x_{i\_1})}{\partial x} \quad (10)\frac{\varphi(x_{i\_n^2-1})}{\partial x}$$

to calculate $\frac{\partial F}{\partial u_i}$

RECENT WORK

- Train a neural network with
  - Input: (u_i, u_j) where u_i, u_j are arbitrary pair from $[a, b]$
  - Find out

$$(1)\frac{\partial \varphi_i}{\partial x}(x_i) \quad (2)\frac{\partial \varphi_i}{\partial x}(x_{i+1}) \quad (3)\frac{\partial \psi_{i,i}}{\partial x}(x_i) \quad (4)\frac{\partial \psi_{i,i}}{\partial x}(x_{i+1})$$

$$(5)\frac{\partial \psi_{i,i+1}}{\partial x}(x_i) \quad (6)\frac{\partial \psi_{i,i+1}}{\partial x}(x_{i+1}) \quad (7)\varphi(x_{i\_1}) \quad (8)\varphi(x_{i_{n^2}-1})$$

$$(9)\frac{\partial \varphi(x_{i\_1})}{\partial x} \quad (10)\frac{\varphi(x_{i\_n^2-1})}{\partial x}$$

REFERENCES

- Physics-informed machine learning: case studies for weather and climate modelling
- https://royalsocietypublishing.org/doi/10.1098/rsta.2020.0093
- Detecting troubled-cells on two-dimensional unstructured grids using a neural network
- https://www.sciencedirect.com/science/article/pii/S0021999119305297
- Machine learning–accelerated computational fluid dynamics
- https://www.pnas.org/content/118/21/e2101784118?__cf_chl_jschl_tk__=pmd_efa773e77381c2f4b475408fc2d512af854c96e1-1627449438-0-gqNtZGzNAeKjcnBszQoO