How can a quantum computers solve PDEs?

Assist Prof. Fabio Pereira dos Santos [1]

Chemical and Biochemical Process Engineering
School of Chemistry/UFRJ

Feb 2022

Outline

Motivation: Where are the intensive partial differential equations (PDE)?

Significance

- Many physical phenomena are fundamentally described by partial differential equations:
  - Quantum mechanics

- Source: https://www.youtube.com/watch?v=3YmeajE-TT8 [a]

$$\mathcal{F}(\mathbf{u}(\mathbf{z});\mathbf{q}) = \mathbf{f}(\mathbf{z}) \quad \mathbf{z} \in \Omega,$$
$$\mathcal{B}(\mathbf{u}(\mathbf{z})) = \mathbf{g}(\mathbf{z}) \quad \mathbf{z} \in \partial\Omega, \quad \mathbf{z} = [\mathbf{x}, t] \quad (1)$$

- source: https://www.youtube.com/watch?v=XNzjjdoz52k

[a] Credit: Seung-Hoon Cha, Sergei Nayakshin

Motivation: Where are the intensive partial differential equations (PDE)?

Significance

- Many physical phenomena are fundamentally described by partial differential equations:
  - Quantum mechanics
  - Particle physics

- Source: https://www.youtube.com/watch?v=3YmeajE-TT8 [a]

$$\mathcal{F}(\mathbf{u}(\mathbf{z});\mathbf{q}) = \mathbf{f}(\mathbf{z}) \quad \mathbf{z} \in \Omega,$$

$$\mathcal{B}(\mathbf{u}(\mathbf{z})) = \mathbf{g}(\mathbf{z}) \quad \mathbf{z} \in \partial\Omega, \quad \mathbf{z} = [\mathbf{x}, t] \quad (1)$$

- source: https://www.youtube.com/watch?v=XNzjjdoz52k

[a] Credit: Seung-Hoon Cha, Sergei Nayakshin

Motivation: Where are the intensive partial differential equations (PDE)?

Significance

- Many physical phenomena are fundamentally described by partial differential equations:
  - Quantum mechanics
  - Particle physics
  - Environmental science

- Source: https://www.youtube.com/watch?v=3YmeajE-TT8 [a]

$$\mathcal{F}(\mathbf{u}(\mathbf{z});\mathbf{q}) = \mathbf{f}(\mathbf{z}) \quad \mathbf{z} \in \Omega,$$

$$\mathcal{B}(\mathbf{u}(\mathbf{z})) = \mathbf{g}(\mathbf{z}) \quad \mathbf{z} \in \partial\Omega, \quad \mathbf{z} = [\mathbf{x}, t] \quad (1)$$

- source: https://www.youtube.com/watch?v=XNzjjdoz52k

[a] Credit: Seung-Hoon Cha, Sergei Nayakshin

Motivation: Where are the intensive partial differential equations (PDE)?

Significance

- Many physical phenomena are fundamentally described by partial differential equations:
  - Quantum mechanics
  - Particle physics
  - Environmental science
  - Fluid mechanics (Navier-Stokes Equations)

- Source: https://www.youtube.com/watch?v=3YmeajE-TT8 [a]

$$\mathcal{F}(\mathbf{u}(\mathbf{z}); \mathbf{q}) = \mathbf{f}(\mathbf{z}) \quad \mathbf{z} \in \Omega,$$

$$\mathcal{B}(\mathbf{u}(\mathbf{z})) = \mathbf{g}(\mathbf{z}) \quad \mathbf{z} \in \partial\Omega, \quad \mathbf{z} = [\mathbf{x}, t] \quad (1)$$

- source: https://www.youtube.com/watch?v=XNzjjdoz52k

[a] Credit: Seung-Hoon Cha, Sergei Nayakshin

Motivation: Where are the intensive partial differential equations (PDE)?

Significance

- Many physical phenomena are fundamentally described by partial differential equations:
  - Quantum mechanics
  - Particle physics
  - Environmental science
  - Fluid mechanics (Navier-Stokes Equations)
  - Astrophysics

- Source: https://www.youtube.com/watch?v=3YmeajE-TT8 [a]

$$\mathcal{F}(\mathbf{u}(\mathbf{z}); \mathbf{q}) = \mathbf{f}(\mathbf{z}) \quad \mathbf{z} \in \Omega,$$

$$\mathcal{B}(\mathbf{u}(\mathbf{z})) = \mathbf{g}(\mathbf{z}) \quad \mathbf{z} \in \partial\Omega, \quad \mathbf{z} = [\mathbf{x}, t] \quad (1)$$

- source: https://www.youtube.com/watch?v=XNzjjdoz52k

[a] Credit: Seung-Hoon Cha, Sergei Nayakshin

Motivation: Where are the intensive partial differential equations (PDE)?

Significance

- Many physical phenomena are fundamentally described by partial differential equations:
  - Quantum mechanics
  - Particle physics
  - Environmental science
  - Fluid mechanics (Navier-Stokes Equations)
  - Astrophysics
  - Biology, to cite a few.

- Source: https://www.youtube.com/watch?v=3YmeajE-TT8 [a]

$$\mathcal{F}(\mathbf{u}(\mathbf{z}); \mathbf{q}) = \mathbf{f}(\mathbf{z}) \quad \mathbf{z} \in \Omega,$$

$$\mathcal{B}(\mathbf{u}(\mathbf{z})) = \mathbf{g}(\mathbf{z}) \quad \mathbf{z} \in \partial\Omega, \quad \mathbf{z} = [\mathbf{x}, t] \quad (1)$$

- source: https://www.youtube.com/watch?v=XNzjjdoz52k

[a] Credit: Seung-Hoon Cha, Sergei Nayakshin

Turbulence from theory to applications

Turbulence Fundamental understanding

Turbulent Flow Simulation at the Exascale: Opportunities and Challenges Workshop

Sponsored by
U.S. Department of Energy
Office of Science
Office of Advanced Scientific Computing Research

Program Committee
Michael Sprague (chair, National Renewable Energy Laboratory)
Stanislav Boldyrev (University of Wisconsin–Madison)
Paul Fischer (Argonne National Laboratory and University of Illinois at Urbana–Champaign)
Ray Grout (National Renewable Energy Laboratory)
William I. Gustafson Jr. (Pacific Northwest National Laboratory)
Robert Moser (University of Texas at Austin)

DOE/ASCR Points of Contact
Michael Martin and Robinson Pino

Abstract

This report details the findings and recommendations from the *Turbulent Flow Simulation at the Exascale: Opportunities and Challenges Workshop*, which was held August, 4–5, 2015, and was sponsored by the U.S. Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR). The workshop objectives were to define and describe the challenges and opportunities that computing at the exascale will bring to turbulent-flow simulations in applied science and technology. The need for accurate simulation of turbulent flows is evident across the DOE applied-science and engineering portfolios, including combustion, plasma physics, nuclear-reactor physics, wind energy, and atmospheric science. The workshop brought together experts in turbulent-flow simulation, computational mathematics, and high-performance computing. Building upon previous

a

[a] F.X.Trias, A. Gorobets, and A. Oliva. "Turbulent flow around a square cylinder Computers Fluids, 123:87–98, 2015.

Intensive PDE application: Climate Science

- Global atmospheric models for large gap in spatial scales between convection (100km) and global motions (1×10^4) km;

GEOPHYSICAL RESEARCH LETTERS, VOL. 40, 4922–4926, doi:10.1002/grl.50944, 2013

Deep moist atmospheric convection in a subkilometer global simulation

Yoshiaki Miyamoto,[1] Yoshiyuki Kajikawa,[1] Ryuji Yoshida,[1] Tsuyoshi Yamaura,[1] Hisashi Yashiro,[1,2] and Hirofumi Tomita[1,2]

Received 19 July 2013; revised 6 September 2013; accepted 6 September 2013; published 20 September 2013.

[1] Deep moist atmospheric convection is a key element of the weather and climate system for transporting mass, momentum, and thermal energy. It has been challenging to simulate convection realistically in global atmospheric models because of the large gap in spatial scales between convection (10^0 km) and global motions (10^4 km). We conducted the first ever subkilometer global simulation and described the features of convection. Through a series of grid-refinement resolution testing, we found that an essential change for convection statistics occurred around 2 km grid spacing. The convection structure, number of convective cells, and distance to the nearest convective cell dramatically changed at this resolution. The convection core was resolved using multiple grids in simulations with grid spacings less than 2.0 km. **Citation:** Miyamoto, Y., Y. Kajikawa, R. Yoshida, T. Yamaura, H. Yashiro, and H. Tomita (2013), Deep moist atmospheric convection in a subkilometer global simulation, *Geophys. Res. Lett.*, *40*, 4922–4926, doi:10.1002/grl.50944.

1. Introduction

[2] Deep moist convection plays an important role in transporting energy in the troposphere. The temporal and spatial scales of the convection are 30 min–1 h and 1–10 km, respectively [*Emanuel*, 1994; *Houze*, 1994]. Convection also contributes to energy transport from equatorial regions to polar regions through meridional circulations. Furthermore, convection-called cumulus parameterization [*Arakawa*, 2004 and references therein]. Recent advances in computer power and development of a new type of model to solve fluid motions on a sphere have made it possible to conduct global simulations without cumulus parameterizations [*Tomita and Satoh*, 2004; *Satoh et al.*, 2008]. Previous studies demonstrated that cloudy atmospheric disturbances can be simulated accurately using such a global model [*Miura et al.*, 2007; *Sato et al.*, 2009; *Fudeyasu et al.*, 2010a; 2010b; *Nasuno and Satoh*, 2011; *Kinter et al.*, 2013].

[4] However, their grid spacings (several kilometers) are still coarser than or comparable to the observed convection scale. It is essentially desired to conduct the simulations with resolution higher than the observed convection scale. Furthermore, although the resolution of several kilometers will be widely used for global simulations in the near future, the resolution dependencies of convections simulated in global models are not yet clear. Previous numerical studies using a limited numerical domain, but without parameterization, have demonstrated the resolution dependence of convection features [*Weisman et al.*, 1997; *Bryan and Fritsch*, 2002; *Petch et al.*, 2002; *Bryan et al.*, 2003].

[5] The scope of this study is to elucidate statistical features of convection in a global model and their resolution dependence, by conducting a series of high-resolution global simulations with resolution as fine as subkilometer that is finer than the convection scale. Section 2 introduces the simulation

- We are on the way to Zettascale computing. However, can Zettascale be the solution?
- it is not enough - Simple aircraift in cruise would take billions of year!!

Fundamental Answer: It is when quantum computing (QC) enters!

Fundamental Questions:

1. ***Is it possible to identify algorithms for PDE solutions that QC can perform effectively?***

Fundamental Answer: It is when quantum computing (QC) enters!

Fundamental Questions:

1. ***Is it possible to identify algorithms for PDE solutions that QC can perform effectively?***
2. ***Can a current QC speed up the solution of these multi-scale nonlinear partial differential equations, such as turbulent Navier-Stokes Equations (NSE)?***

Fundamental Answer: It is when quantum computing (QC) enters!

Fundamental Questions:

1. ***Is it possible to identify algorithms for PDE solutions that QC can perform effectively?***
2. ***Can a current QC speed up the solution of these multi-scale nonlinear partial differential equations, such as turbulent Navier-Stokes Equations (NSE)?***
3. Is it currently possible with ***noisy intermediate-scale quantum (NISQ) architecture?***

Fundamental Answer: It is when quantum computing (QC) enters!

Fundamental Questions:

1. ***Is it possible to identify algorithms for PDE solutions that QC can perform effectively?***
2. ***Can a current QC speed up the solution of these multi-scale nonlinear partial differential equations, such as turbulent Navier-Stokes Equations (NSE)?***
3. Is it currently possible with ***noisy intermediate-scale quantum (NISQ) architecture?***
4. Can we really understand fundamental science with a QA for solving PDE?

Navier-Stokes Equations

- Navier-Stokes equations (NSE)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0; \tag{2}$$

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla p + \nabla \cdot \boldsymbol{\emptyset} + \rho \mathbf{g}; \tag{3}$$

Remarks

- Continuum Hypothesis

Navier-Stokes Equations

- Navier-Stokes equations (NSE)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0; \tag{2}$$

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla p + \nabla \cdot \boldsymbol{\emptyset} + \rho \mathbf{g}; \tag{3}$$

Remarks

- Continuum Hypothesis
- Non-linear coupled PDE system (can be intensive)

Navier-Stokes Equations

- Navier-Stokes equations (NSE)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0; \tag{2}$$

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla p + \nabla \cdot \boldsymbol{\emptyset} + \rho \mathbf{g}; \tag{3}$$

Remarks

- Continuum Hypothesis
- Non-linear coupled PDE system (can be intensive)
- Analytical solution existence and smoothness - unsolved problem

Navier-Stokes Equations

- Navier-Stokes equations (NSE)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0; \tag{2}$$

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla p + \nabla \cdot \boldsymbol{\emptyset} + \rho \mathbf{g}; \tag{3}$$

Remarks

- Continuum Hypothesis
- Non-linear coupled PDE system (can be intensive)
- Analytical solution existence and smoothness - unsolved problem
- Numerical Solution!!!

Research Group Interest

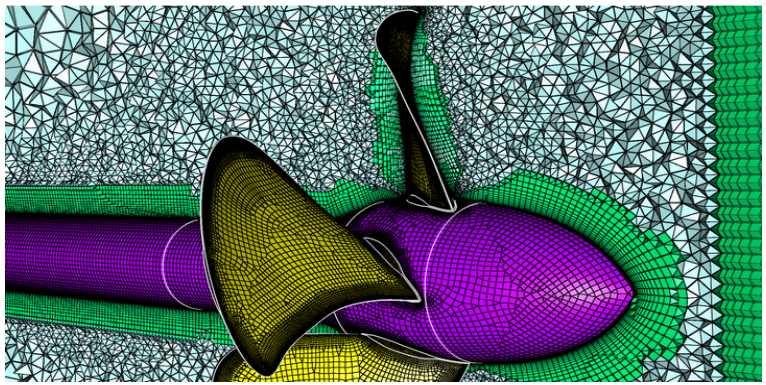
Computational Fluid Dynamics

What is it?

- Simulation of phenomena involving fluid flow;
  1. Domain representation;

Numerical Solution

- Sophisticated algorithms, such as: Finite Elements, Finite Volume and so on;



https://www.pointwise.com/

Research Group Interest

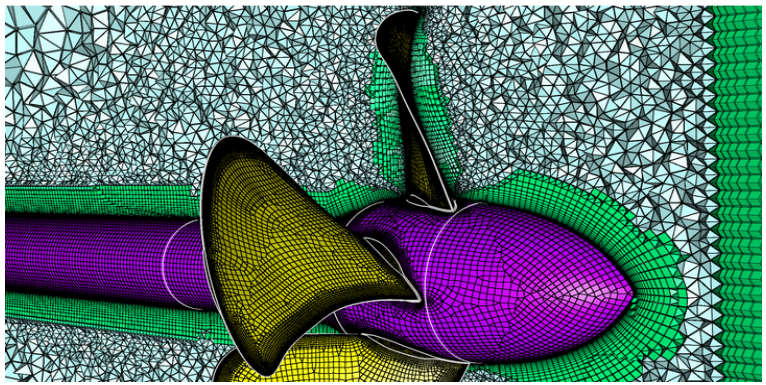
Computational Fluid Dynamics

What is it?

- Simulation of phenomena involving fluid flow;
  1. Domain representation;
  2. Mesh discretization (for numerical methods that requires mesh);

Numerical Solution

- Sophisticated algorithms, such as: Finite Elements, Finite Volume and so on;



https://www.pointwise.com/

Research Group Interest

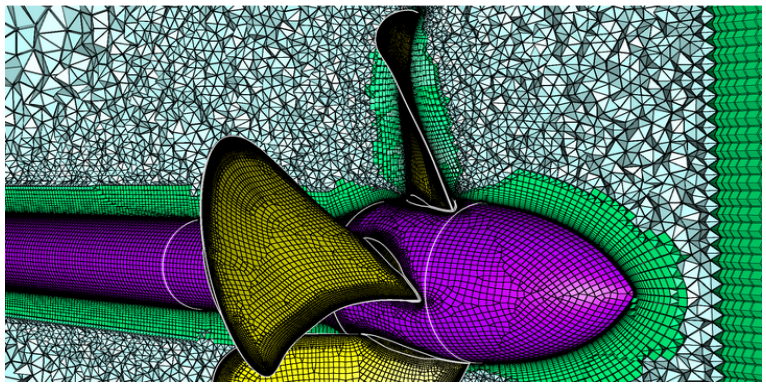
Computational Fluid Dynamics

What is it?

- Simulation of phenomena involving fluid flow;
  1. Domain representation;
  2. Mesh discretization (for numerical methods that requires mesh);
  3. Solution of the discrete equations;

Numerical Solution

- Sophisticated algorithms, such as: Finite Elements, Finite Volume and so on;



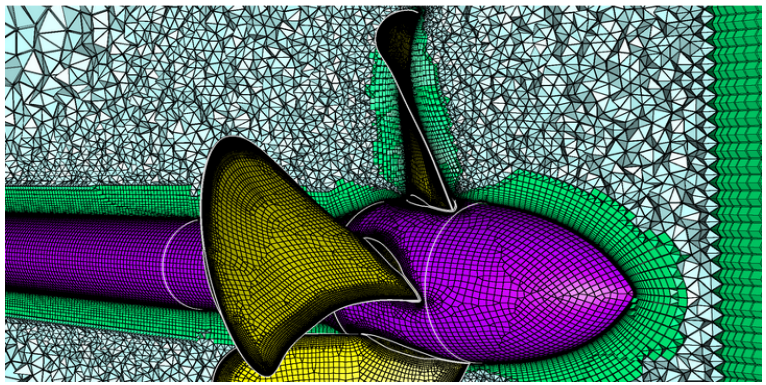
https://www.pointwise.com/

Research Group Interest

Computational Fluid Dynamics

What is it?

- Simulation of phenomena involving fluid flow;
  1. Domain representation;
  2. Mesh discretization (for numerical methods that requires mesh);
  3. Solution of the discrete equations;
  4. Processing results (visualization)



https://www.pointwise.com/

Numerical Solution

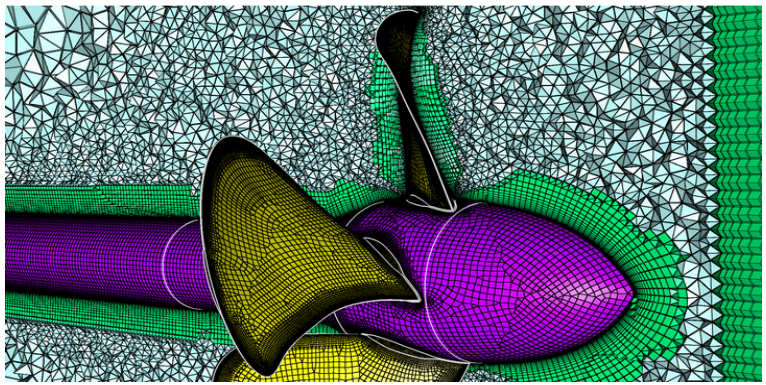
- Sophisticated algorithms, such as: Finite Elements, Finite Volume and so on;

Research Group Interest

Computational Fluid Dynamics

What is it?

- Simulation of phenomena involving fluid flow;
  1. Domain representation;
  2. Mesh discretization (for numerical methods that requires mesh);
  3. Solution of the discrete equations;
  4. Processing results (visualization)



https://www.pointwise.com/

Numerical Solution

- Sophisticated algorithms, such as: Finite Elements, Finite Volume and so on;

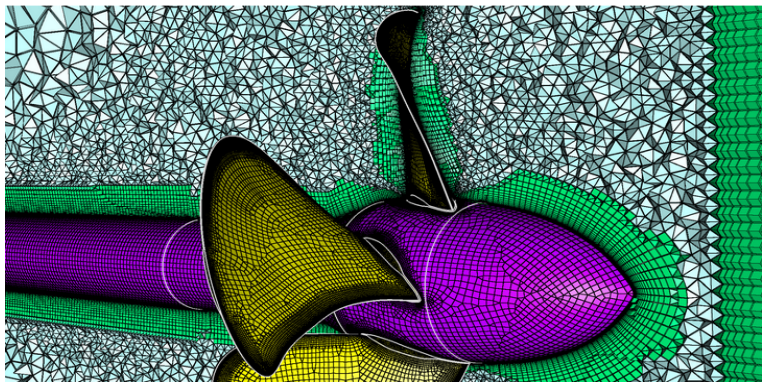
- CFD simulations can be expensive;

Research Group Interest

Computational Fluid Dynamics

What is it?

- Simulation of phenomena involving fluid flow;
  1. Domain representation;
  2. Mesh discretization (for numerical methods that requires mesh);
  3. Solution of the discrete equations;
  4. Processing results (visualization)



https://www.pointwise.com/

Numerical Solution

- Sophisticated algorithms, such as: Finite Elements, Finite Volume and so on;

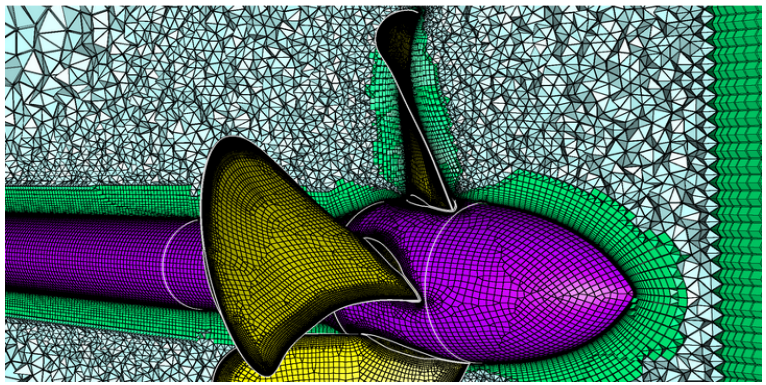
- CFD simulations can be expensive;
- To test, design and analysis, several simulations can be needed;

Research Group Interest

Computational Fluid Dynamics

What is it?

- Simulation of phenomena involving fluid flow;
  1. Domain representation;
  2. Mesh discretization (for numerical methods that requires mesh);
  3. Solution of the discrete equations;
  4. Processing results (visualization)



https://www.pointwise.com/

Numerical Solution

- Sophisticated algorithms, such as: Finite Elements, Finite Volume and so on;

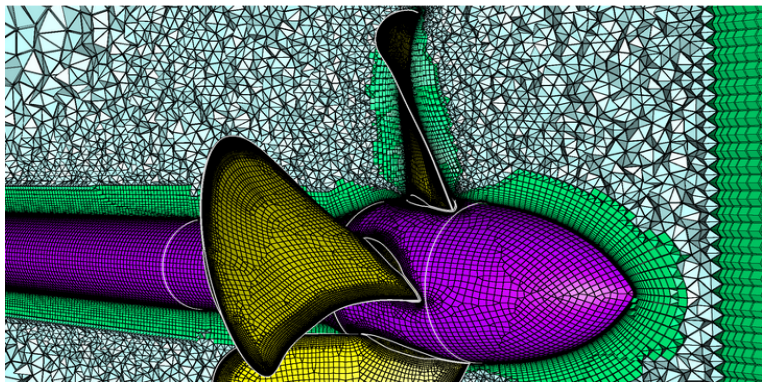
- CFD simulations can be expensive;
- To test, design and analysis, several simulations can be needed;
- How to reduce computational costs, without disobeying the physics of problem?

Research Group Interest

Computational Fluid Dynamics

What is it?

- Simulation of phenomena involving fluid flow;
  1. Domain representation;
  2. Mesh discretization (for numerical methods that requires mesh);
  3. Solution of the discrete equations;
  4. Processing results (visualization)



https://www.pointwise.com/

Numerical Solution

- Sophisticated algorithms, such as: Finite Elements, Finite Volume and so on;

- CFD simulations can be expensive;
- To test, design and analysis, several simulations can be needed;
- How to reduce computational costs, without disobeying the physics of problem?
- How Machine Learning can help?

Neural Network: Brief description

Machine Learning and Computer science

- **AI**: Looking for human-level intelligence;
- **ML**: Learn parttens in a dataset and performs predictions;
- **Data Science**: Find insights from data;

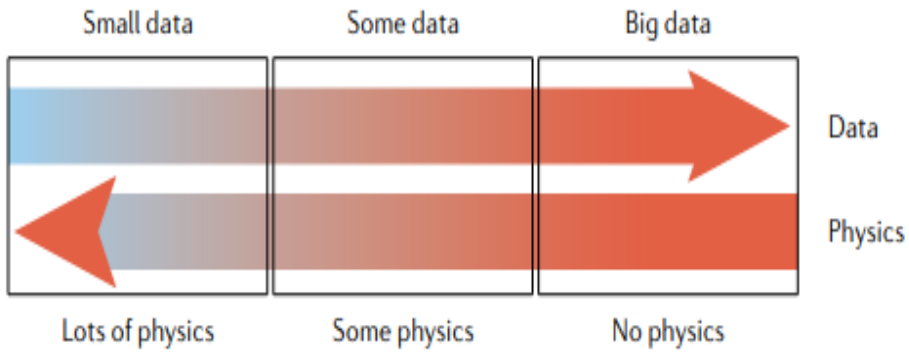
"To make progress in ML in science it is crucial that we incorporate computational models into the learning step."
Prof. Amir Gholaminejad

Types of PINNs

How to introduce physics into a neural network:

- **Method 1**; Where big data is available, but the governing physical law may not be known;

[a] Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Nat Rev Phys 3, 422–440 (2021)

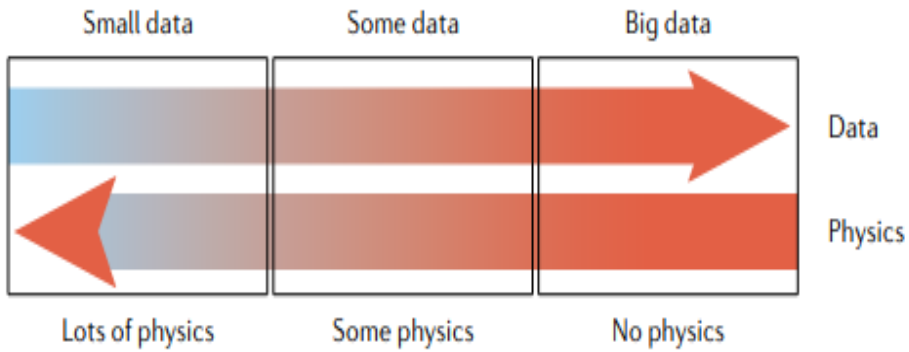


Types of PINNs

How to introduce physics into a neural network:

- **Method 1**; Where big data is available, but the governing physical law may not be known;
- **Method 2**; On the other extreme, where little data is available ("small data regime"), but the describing physics is known (included as a constraint);

[a] Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Nat Rev Phys 3, 422–440 (2021)

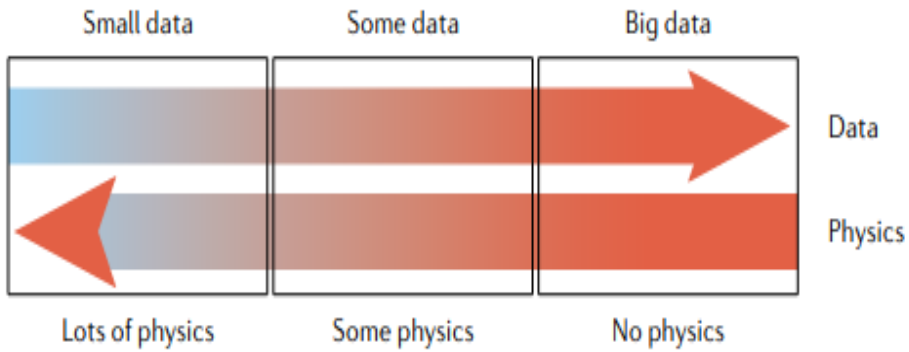


Types of PINNs

How to introduce physics into a neural network:

- **Method 1**; Where big data is available, but the governing physical law may not be known;
- **Method 2**; On the other extreme, where little data is available ("small data regime"), but the describing physics is known (included as a constraint);
- **Method 3**; No data is available but you know the physics
  - PINN actually solves de physics

[a] Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Nat Rev Phys 3, 422–440 (2021)

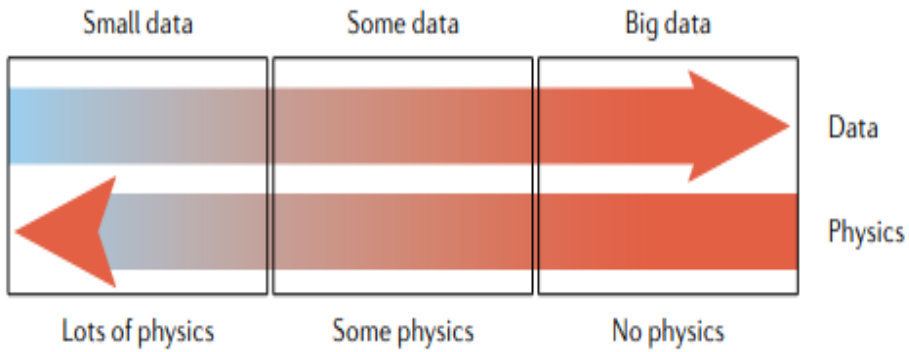


Types of PINNs

How to introduce physics into a neural network:

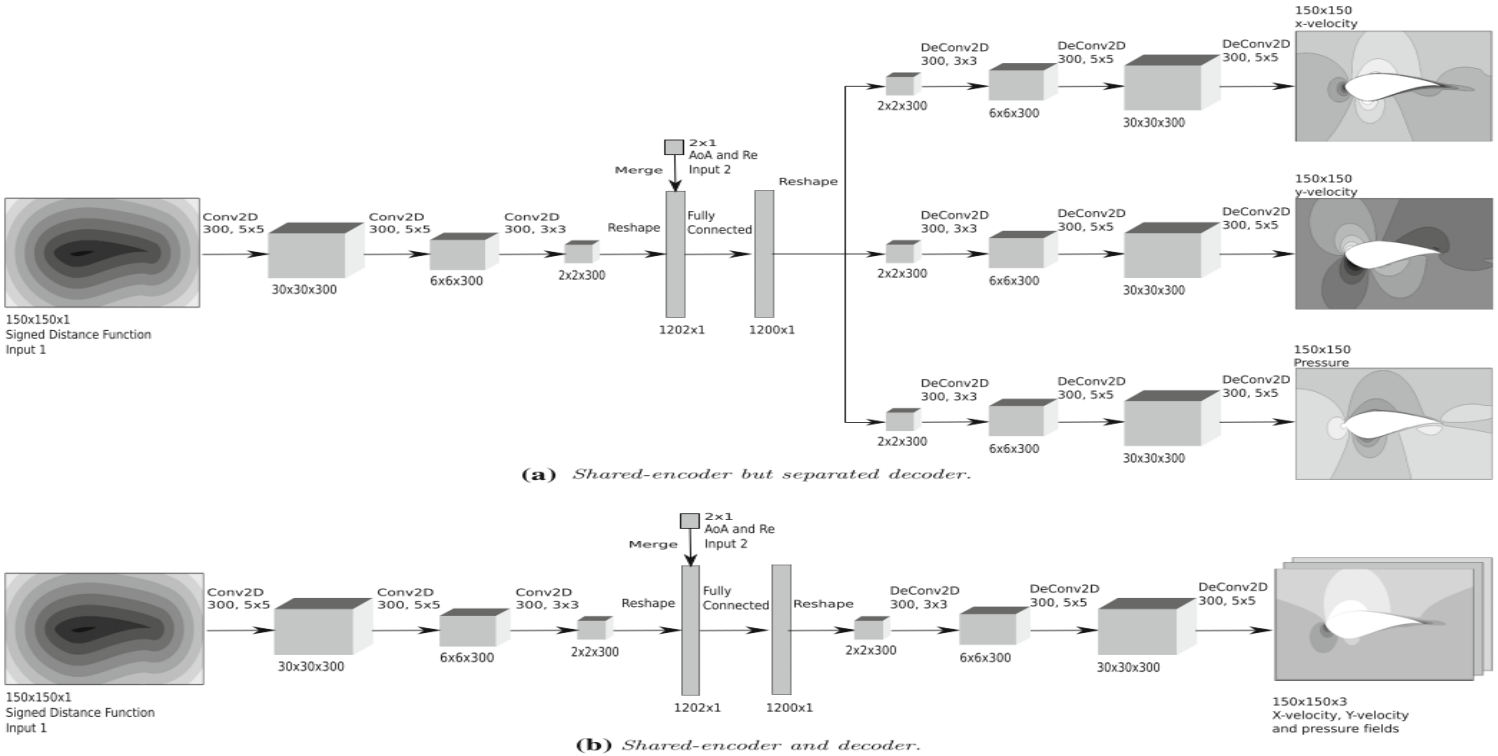
- **Method 1**; Where big data is available, but the governing physical law may not be known;
- **Method 2**; On the other extreme, where little data is available ("small data regime"), but the describing physics is known (included as a constraint);
- **Method 3**; No data is available but you know the physics
  - PINN actually solves de physics
- **Method 4**; where the physics is partially known and several scattered measurements are available (inverse problems) [a]

[a] Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Nat Rev Phys 3, 422–440 (2021)



Physics-informed by a large number of data

Model reduction by Machine Learning, Bhatnagar et 2019

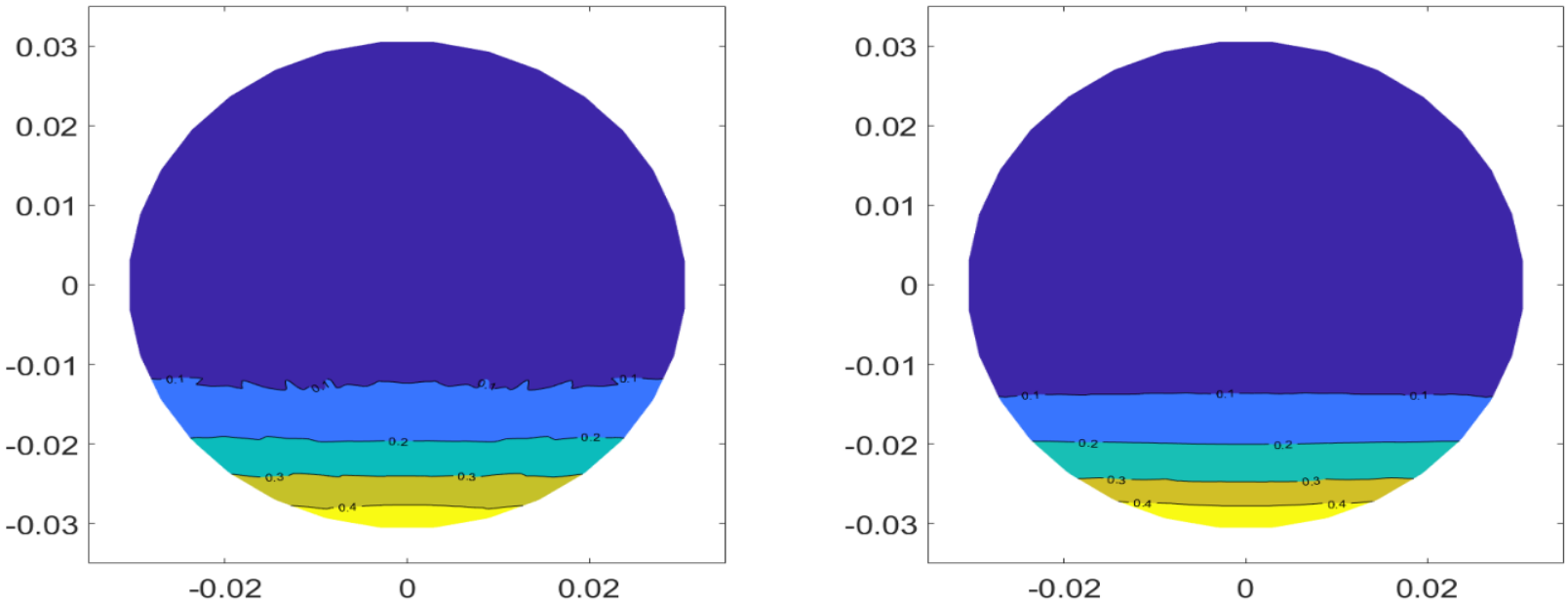


Model reduction by Machine Learning

- Order reduction for aerodynamics;
- Convolutional neural network (ConvNet) with AutoEncoder;

Physics-informed by a large number of data

Model reduction by Machine Learning, Ligang Lu and Kuochen Tsai (SHELL Oil Company), 2019.



Model reduction by Machine Learning

- CFD model and predictions of oil water separation in horizontal pipelines;
- Convolutional neural network (CNN) model;

Physics-informed by constraints

Symmetries and Invariance

- Impose some properties to the trainning, such as: Symmetries and Invarience;
- Symmetry in a shear stress tensor;
- Invariance related to a reference or even conservation laws;

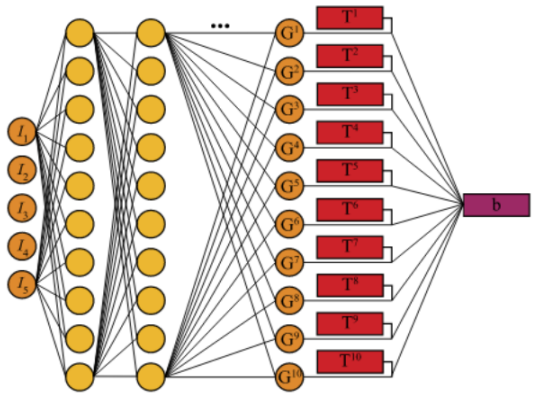


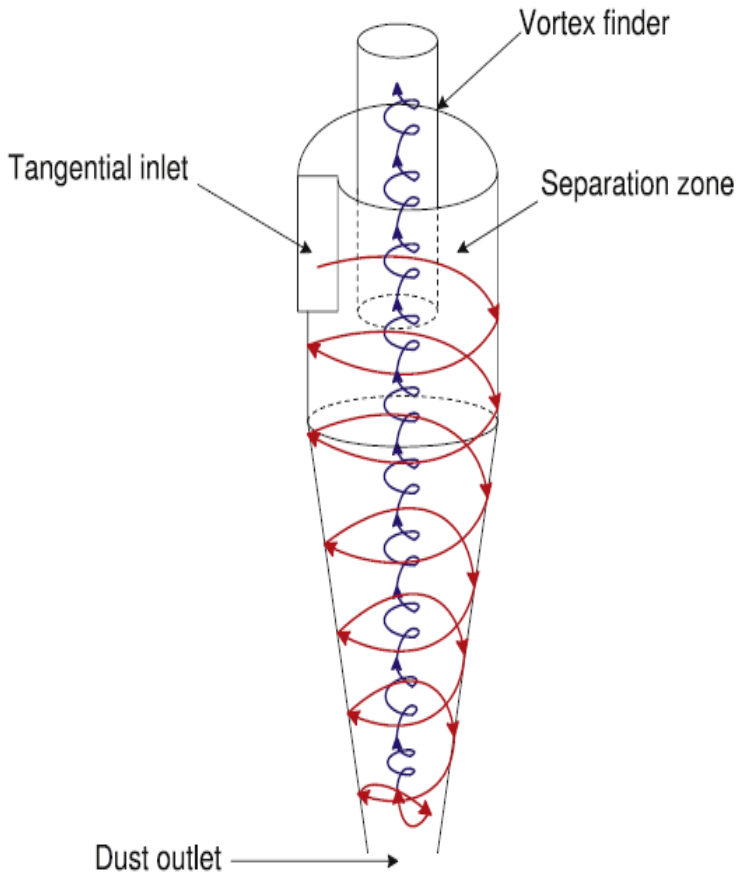
Fig. 2. Invariant, fully-connected (some connections are omitted for clarity), neural network architecture proposed by Ling et al. [17]. The circles indicate scalar values and the rectangles represent 3×3 second-order tensors.

Figure: Symetry and Invariance of reynolds stress tensor based on Invariance expansion;[1]

[1] Ling, J., Kurzawski, A., Templeton, J. (2016). Journal of Fluid Mechanics, 807, 155-166. doi:10.1017/jfm.2016.615

Physics-informed by constraints

Cyclone Separator,Caturwati Ni Ketut et al, 2017



Number of samples

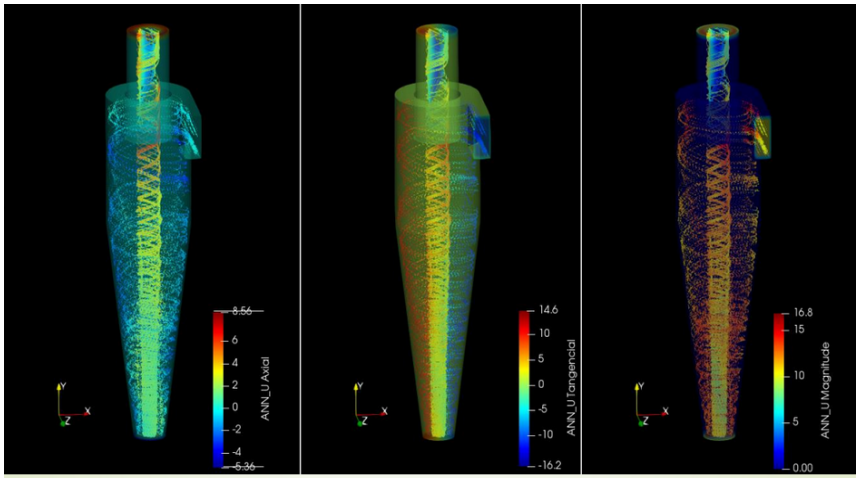
- 300 Simulations with different Re numbers;
- Data extracted from the cell centers of the mesh

Case Setup

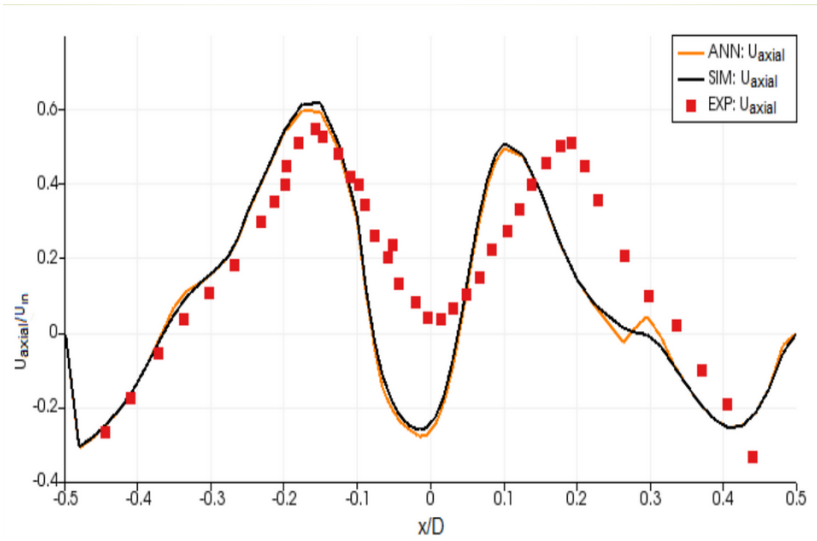
- OpenFOAM (OF) simulation with SimpleFOAM solver;
- $\kappa - \epsilon$ with curvature correction in OF, Spalart-Shur, 2000;

Physics-informed by constraints

Velocity streamlines



Velocity profile versus experimental data



Conservation Equation Table

	$R_{\mathbf{M},x}$	$R_{\mathbf{M},y}$	$R_{\mathbf{M},z}$	$\nabla \cdot \mathbf{U}$
$\overline{V}$	$-4,85 \cdot 10^{-9}$	$-2,31 \cdot 10^{-8}$	$1,65 \cdot 10^{-8}$	$-7,2 \cdot 10^{-7}$
$max(V)$	$2,3 \cdot 10^{-3}$	$4,9 \cdot 10^{-5}$	$1,1 \cdot 10^{-3}$	$0,047$
$min(V)$	$-1 \cdot 10^{-3}$	$-4,10 \cdot 10^{-5}$	$-1,8 \cdot 10^{-3}$	$-0,04$

Physics-informed by constraints

Computational resource for simulation

- Intel Core i7, 12 cores; 32GB RAM;

Computational resource for PIDNN

- Intel Xeon® E5-2640 v4 2,4GHz
- Tesla P100 (Pascal), 3584 CUDA cores, 16GB VRAM;

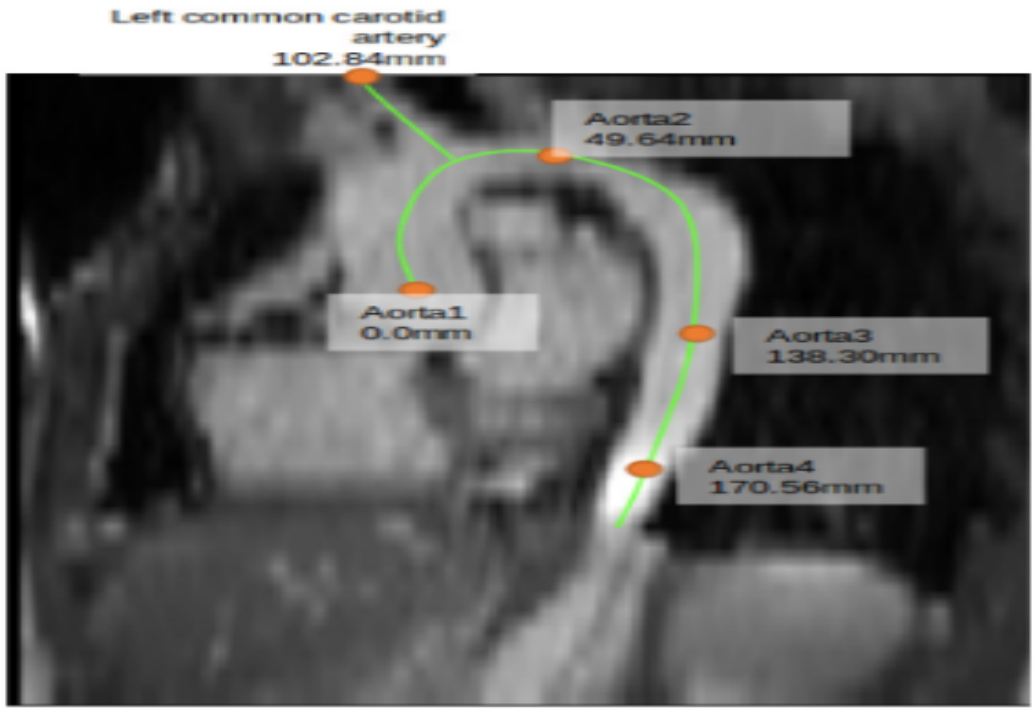
Computational Time

1. CFD Simulation computational Time ≈ 1.6 h
2. Training time ≈ 3 h
3. PIDNN computation ≈ 1 s
4. Computational "speedup“ ≈ 5000;[a]

[a] L.H. Queiroz, F.P. Santos, J.P. Oliveira, M.B. Souza, Digital Chemical Engineering, Volume 1, 2021, 100002.

Physics-informed Neural Network approximation

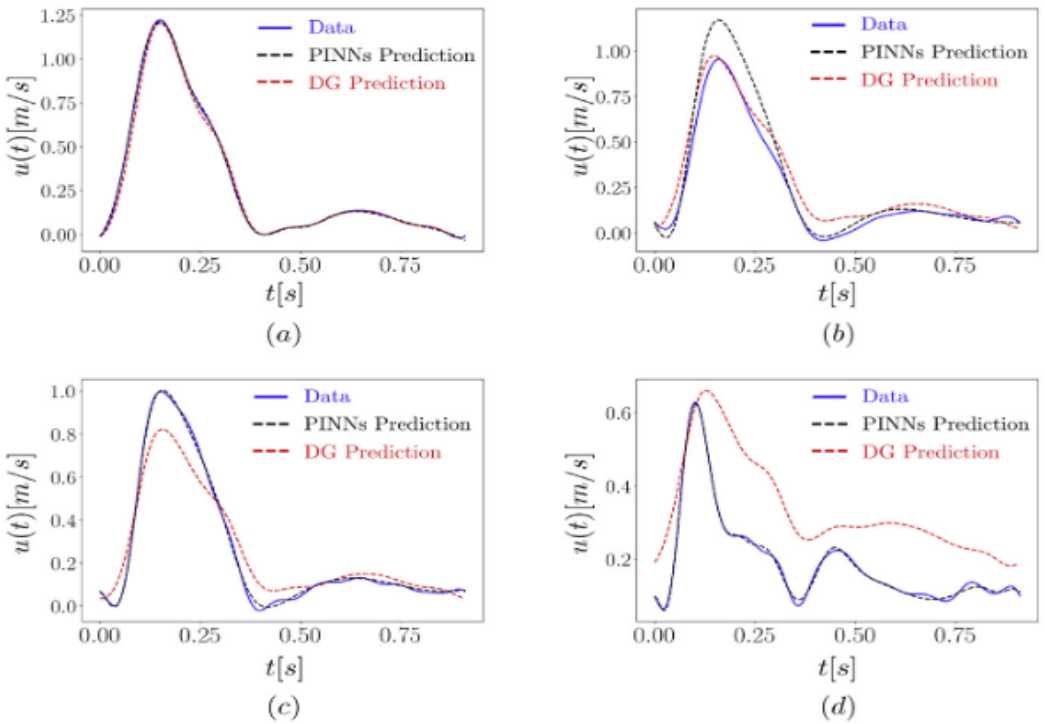
Flow through the aorta/carotid bifurcation of a healthy human subject: Positions of acquired 4D flow MRI measurements in the aorta/carotid bifurcation of a healthy volunteer.[2]



[2] G. Kissas, Y. Yang, E. Hwuang et al. / Computer Methods in Applied Mechanics and Engineering 358 (2020) 112623 21

Physics-informed Neural Network approximation

Comparison of the clinically acquired waveforms of blood velocity versus the predictions of the proposed physics-informed neural networks, and of a conventional Discontinuous Galerkin solver.[3]

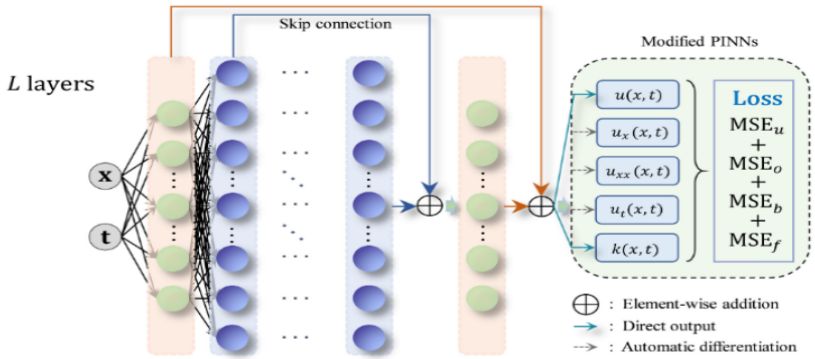


[3] G. Kissas, Y. Yang, E. Hwuang et al. / Computer Methods in Applied Mechanics and Engineering 358 (2020) 112623 21

Physics-informed Neural Network approximation (inverse problem)

A physics-informed deep learning method for solving direct and inverse heat conduction problems of materials.[4]

Inverse problem Setup



$$\rho(u)Cp(u)\frac{\partial u}{\partial t} - k(u)\frac{\partial^2 u}{\partial x^2} = 0 \quad in \quad \Omega, t \in [0, t_{max}] \tag{4}$$

[4] Zhili He, Futao Ni, Weiguo Wang, Jian Zhang. Materials Today Communications, Volume 28, 2021, 102719.

Neural Network: Brief description

Neural Network

- Recognized as fundamental nonlinear function approximators. It can be represented as: $R(\mathbf{w}, b) = \int L[\mathbf{y}, \phi(\mathbf{w}, \mathbf{y}, \mathbf{x}, b)]p(\mathbf{y}, \mathbf{x})d\mathbf{x}d\mathbf{y}$

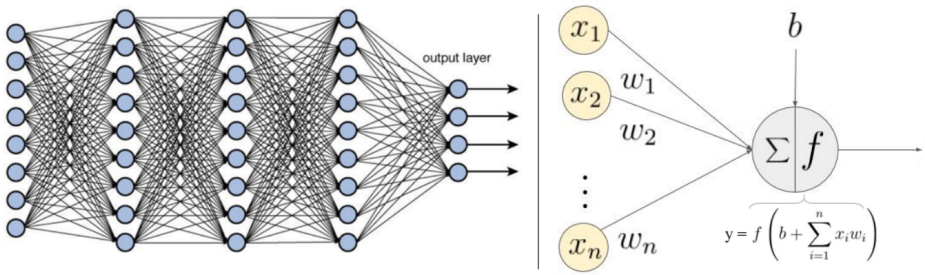


Figure: Deep neural network example

Deep Neural Network is actually a composition function!

Neural Network: Brief description

Supervised neural network

- Depends on what the focus is on....
- We want to find a neural net, $\phi(\mathbf{y}, \mathbf{x}; \theta(\mathbf{w}, b))$, that maps $\mathbf{x}$ to $\mathbf{y}$
- Minimizing $R(\mathbf{w}, b)$, where the data are equiprobable;

$$R(\mathbf{w}, b) = \frac{1}{N} \sum_{i=0}^{N} \left[y_i - \phi(y_i, \mathbf{x}, \mathbf{w}, b) \right]^2 \tag{5}$$

being N the size of trainning data;
How we usually solve? Stochastic gradient decent

$$\theta_{k+1} = \theta_k - \alpha_k \frac{1}{m} \sum_{1}^{m} \nabla_\theta \left[y_j - \phi(y_j, \mathbf{x}, \mathbf{w}, b) \right]^2 \tag{6}$$

Deep Neural Network: Brief description

Let $N^L(\mathbf{x}) : \Re^{d_{in}} \rightarrow \Re^{d_{out}}$. Let us say the weight matrix of the l^{th} layer and the bias vector are $\mathbf{W}^l \in \Re^{N_l \times N_{l-1}}$ and $\mathbf{b}^l \in \Re^{N_l}$, respectively. Then, one can build the deep neural network below:

- $N^0(\mathbf{x}) = \mathbf{x} \quad \in \Re^{d_{in}}$

where ϕ is a composition function of $N^j(\mathbf{x}), \quad j = 0 \cdots L$

$$\mathbf{u}(\mathbf{x}) \approx \mathbf{u}(\mathbf{x}; \theta) = \phi(\theta; \mathbf{x}) \tag{7}$$

where $\theta = [\mathbf{W}, \mathbf{b}]$

Deep Neural Network: Brief description

Let $N^L(\mathbf{x}) : \Re^{d_{in}} \rightarrow \Re^{d_{out}}$. Let us say the weight matrix of the l^{th} layer and the bias vector are $\mathbf{W}^l \in \Re^{N_l \times N_{l-1}}$ and $\mathbf{b}^l \in \Re^{N_l}$, respectively. Then, one can build the deep neural network below:

- $N^0(\mathbf{x}) = \mathbf{x} \quad \in \Re^{d_{in}}$
- $N^l(\mathbf{x}) = \sigma\left(\mathbf{W}^l N^{l-1}(\mathbf{x}) + \mathbf{b}^l\right) \quad \in \Re^{N_l}, \quad 1 < l \leq L-1$

where ϕ is a composition function of $N^j(\mathbf{x}), \quad j = 0 \cdots L$

$$\mathbf{u}(\mathbf{x}) \approx \mathbf{u}(\mathbf{x}; \theta) = \phi(\theta; \mathbf{x}) \tag{7}$$

where $\theta = [\mathbf{W}, \mathbf{b}]$

Deep Neural Network: Brief description

Let $N^L(\mathbf{x}) : \Re^{d_{in}} \rightarrow \Re^{d_{out}}$. Let us say the weight matrix of the l^{th} layer and the bias vector are $\mathbf{W}^l \in \Re^{N_l \times N_{l-1}}$ and $\mathbf{b}^l \in \Re^{N_l}$, respectively. Then, one can build the deep neural network below:

- $N^0(\mathbf{x}) = \mathbf{x} \quad \in \Re^{d_{in}}$
- $N^l(\mathbf{x}) = \sigma\left(\mathbf{W}^l N^{l-1}(\mathbf{x}) + \mathbf{b}^l\right) \quad \in \Re^{N_l}, \quad 1 < l \leq L-1$
- $N^L(\mathbf{x}) = \mathbf{W}^L N^{L-1}(\mathbf{x}) + \mathbf{b}^L \quad \in \Re^{d_{out}}$

where ϕ is a composition function of $N^j(\mathbf{x}), \quad j = 0 \cdots L$

$$\mathbf{u}(\mathbf{x}) \approx \mathbf{u}(\mathbf{x}; \theta) = \phi(\theta; \mathbf{x}) \tag{7}$$

where $\theta = [\mathbf{W}, \mathbf{b}]$

Automatic Differentiation

How to calculate derivatives?

- hand-coded analytical derivative;

In PINNs, it is required to compute the derivatives of the network outputs with respect to the network inputs

Automatic Differentiation

How to calculate derivatives?

- hand-coded analytical derivative;
- finite difference or other numerical approximations;

In PINNs, it is required to compute the derivatives of the network outputs with respect to the network inputs

Automatic Differentiation

How to calculate derivatives?

- hand-coded analytical derivative;
- finite difference or other numerical approximations;
- symbolic differentiation (used in software programs such as Mathematica, Maxima, and Maple); and

In PINNs, it is required to compute the derivatives of the network outputs with respect to the network inputs

Automatic Differentiation

How to calculate derivatives?

- hand-coded analytical derivative;
- finite difference or other numerical approximations;
- symbolic differentiation (used in software programs such as Mathematica, Maxima, and Maple); and
- Automatic differentiation (the derivatives are evaluated using backpropagation);

In PINNs, it is required to compute the derivatives of the network outputs with respect to the network inputs

Automatic Differentiation

How to calculate derivatives?

- a compositional function, then AD applies the chain rule repeatedly to compute the derivatives;
- There are two steps in AD: one forward pass to compute the values of all variables, and one backward pass to compute the derivative

Consider a FNN with one hidden layer two inputs (x_1, x_2) and one output (y);

$$v = -2x_1 + 3x_2 + 0.5 \tag{8}$$

$$h = tahn(v) \tag{9}$$

$$y = 2h - 1 \tag{10}$$

Automatic Differentiation

To calculate the derivative of $\frac{\partial y}{\partial x_1}$ and $\frac{\partial y}{\partial x_2}$ at $(2, 1)$:

Forward pass	Backward pass
$x_1 = 2$ $x_2 = 1$	$\frac{\partial y}{\partial y} = 1$
$v = -2x_1 + 3x_2 + 0.5 = -0.5$ $h = \tanh v \approx -0.462$	$\frac{\partial y}{\partial h} = \frac{\partial (2h-1)}{\partial h} = 2$ $\frac{\partial y}{\partial v} = \frac{\partial y}{\partial h}\frac{\partial h}{\partial v} = \frac{\partial y}{\partial h}\text{sech}^2(v) \approx 1.573$
$y = 2h - 1 = -1.924$	$\frac{\partial y}{\partial x_1} = \frac{\partial y}{\partial v}\frac{\partial v}{\partial x_1} = \frac{\partial y}{\partial v} \times (-2) = -3.146$ $\frac{\partial y}{\partial x_2} = \frac{\partial y}{\partial v}\frac{\partial v}{\partial x_2} = \frac{\partial y}{\partial v} \times 3 = 4.719$

Figure: AD calculation example

How to calculate derivatives?

- AD has two passes one forward pass and one backward;
- Any discretization derivative method has at least ;
- It can be used for n order derivative [a];

[a] Lu, Lu and Meng, Xuhui and Mao, Zhiping and Karniadakis, George Em. SIAM Review, Volume 63,number 1, 208-228, 2021

Universal approximation theorem

Approximation: $Loss \to 0$ [5];?

Let σ be any continuous sigmoidal function. The finite sum of the form:

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma \left(w_j x + b_j \right) \tag{11}$$

are dense in $C(I_d)$

Theorem (Pinks, 1999)

Let $\mathbf{m}^i \in \mathbb{Z}_+^d$, $i = 1, \cdots, s$, and set $m = max_{i=1,\cdots,s} |\mathbf{m}^i|$. Assume $\sigma \in C^m(\mathbb{R})$ and also is not a polynomial. Then the space of single hidden layer neural network:

$$\mathcal{M}(\sigma) = \sigma \left(\mathbf{W}\mathbf{x} + b \right) \tag{12}$$

is dense in $C^{\mathbf{m}^1, \cdots, \mathbf{m}^s}(\mathbb{R}^d) := \bigcap_{i=1}^{s} C^i(\mathbb{R}^d)$.

[5] Kurt Hornik, Maxwell Stinchcombe, Halbert White, Multilayer feedforward networks are universal approximators, Neural Networks, Volume 2, Issue 5, 1989, Pages 359-366.

Universal approximation theorem

Universal approximation theorem

For $\mathbf{m} = (m_1, \cdots, m_d) \in \mathbb{Z}_+^d$, *we set* $|\mathbf{m}| := m_1 + \cdots + m_d$ *and*

$$D^{\mathbf{m}} := \frac{\partial^{|\mathbf{m}|}}{\partial^{m_1} x_d \cdots \partial^{m_d} x_d} \tag{13}$$

We say $f \in C^{\mathbf{m}}(\mathbb{R}^d)$ if $D^{\mathbf{k}} f \in C^{\mathbf{d}}\ \forall\ \mathbf{k} \leq \mathbf{m}$, $\mathbf{k} \in \mathbb{Z}_+^d$, where $C^{\mathbf{d}} = \{f : \mathbb{R}^d \to \mathbb{R}\}$ is a space of continuous functions. We can recall the Pinkus' Theorem and say: *For a any* $f \in C^{\mathbf{m}}(\mathbb{R}^d)$, *any compact* $K \subset \mathbb{R}^d$, *and any* $\epsilon > 0$, *there exists a* $g \in \mathcal{M}(\sigma)$ *satisfying:*

$$max|D^{\mathbf{k}} f - D^{\mathbf{k}} g| < \epsilon, \quad x \in K, \tag{14}$$

Universal approximation theorem

Universal approximation theorem

- Each neuron can be seen as a basis function;
- MLP with non-linear activations are universal functions approximators[a];
- However, it says that MLP can approximate but...

[a] Kurt Hornik, Maxwell Stinchcombe, Halbert White, Multilayer feedforward networks are universal approximators, Neural Networks, Volume 2, Issue 5, 1989, Pages 359-366.

Limitation

- Trainability;

Questions: Does the sequence of neural networks converge to the solution to PDEs?

On convergence of PINN

On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs

Yeonjong Shin[1], Jérôme Darbon[1], and George Em Karniadakis[1]

[1] *Division of Applied Mathematics, Brown University, Providence, RI 02912, USA*

Abstract. Physics informed neural networks (PINNs) are deep learning based techniques for solving partial differential equations (PDEs) encounted in computational science and engineering. Guided by data and physical laws, PINNs find a neural network that approximates the solution to a system of PDEs. Such a neural network is obtained by minimizing a loss function in which any prior knowledge of PDEs and data are encoded. Despite its remarkable empirical success in one, two or three dimensional problems, there is little theoretical justification for PINNs.

As the number of data grows, PINNs generate a sequence of minimizers which correspond to a sequence of neural networks. We want to answer the question: Does the sequence of minimizers converge to the solution to the PDE? We consider two classes of PDEs: linear second-order elliptic and parabolic. By adapting the Schauder approach and the maximum principle, we show that the sequence of minimizers strongly converges to the PDE solution in C^0. Furthermore, we show that if each minimizer satisfies the initial/boundary conditions, the convergence mode becomes H^1. Computational examples are provided to illustrate our theoretical findings. To the best of our knowledge, this is the first theoretical work that shows the consistency of PINNs.

AMS subject classifications: 65M12, 41A46, 35J25, 35K20

Key words: Physics Informed Neural Networks, Convergence, Hölder Regularization, Elliptic and Parabolic PDEs, Schauder approach

Simple PINN-PDE example

Mathematical Formulation[a]

[a] I. Lagaris, A. Likas, D. Fotiadis. Artificial neural networks for solving ordinary and partial differential, 1997.Mathematics, Physics, Computer Science, Medicine IEEE transactions on neural networks

$$\frac{\partial g}{\partial x} = h(x, g), \quad g(0) = A \tag{15}$$

- With NN representation;

$$g(x) \approx g(x; \theta) = A + xNN(x; \theta); \tag{16}$$

$$L(\theta) = \int_{\Omega_X} \left[\frac{\partial g(x; \theta)}{\partial x} - h(x, g(x; \theta)) \right]^2 dx \tag{17}$$

Simple PINN-PDE solution

- In a Riemman point of view, we have:

$$L(\theta) = \lim_{N\to\infty} \sum_{i=0}^{N} \left[\frac{\partial g(x_i;\theta)}{\partial x} - h(x_i, g(x_i;\theta)) \right]^2 \tag{18}$$

- Solving the problem with Stochastic gradient descent

$$\theta_{k+1} = \theta_k - \alpha_k \frac{1}{N} \sum_{1}^{N} \nabla_\theta \left[\frac{\partial g(x_i;\theta)}{\partial x} - h(x_i, g(x_i;\theta)) \right]^2 \tag{19}$$

Finally,

$$g(x;\theta) = A + xNN(x;\theta); \tag{20}$$

Forward PINN-PDE

Generalization PINN-PDE

- Extension for General PDEs[a]

[a] M Raissi, P Perdikaris, GE Karniadakis Journal of Computational Physics 378, 686-707, 2017

PDE general form

Consider the general system PDE as:

$$\mathbf{f}(\mathbf{x}; \frac{\partial u_1}{\partial x_1}, \cdots, \frac{\partial^2 u_m}{\partial x_1 \partial x_d}, \cdots; \lambda) = 0, \quad \mathbf{x} \in \Omega, \tag{21}$$

with boundary conditions:

$$B(\mathbf{x}, \mathbf{u}) = 0 \quad on \quad \partial\Omega \tag{22}$$

Forward Generalization PINN-PDE

Generalization PINN-PDE

$$\Im(\theta, \tau) = \omega_f \Im_f(\theta, \tau_f) + \omega_b \Im_b(\theta, \tau_b) \tag{23}$$

where,

$$\Im_f(\theta, \tau_f) = \frac{1}{|\tau_f|} \sum_{\mathbf{x} \in \tau_f} ||\mathbf{f}(\mathbf{x}; \frac{\partial \hat{u}_1}{\partial x_1}, \cdots, \frac{\partial^2 \hat{u}_m}{\partial x_1 \partial x_d}, \cdots; \lambda)||_2^2 \tag{24}$$

and

$$\Im_b(\theta, \tau_b) = \frac{1}{|\tau_b|} \sum_{\mathbf{x} \in \tau_b} ||B(\mathbf{x}, \mathbf{u})||_2^2 \tag{25}$$

Forward Generalization PINN-PDE

Generalization PINN-PDE

- PINN algorithm[a];

[a] Lu, Lu and Meng, Xuhui and Mao, Zhiping and Karniadakis, George Em. DeepXDE: A deep learning library for solving differential equations.SIAM Review .volume 63, number 1, 208-228, 2021

Procedure 2.1 The PINN algorithm for solving differential equations.

Step 1	Construct a neural network $\hat{u}(\mathbf{x};\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$.
Step 2	Specify the two training sets $\mathcal{T}_f$ and $\mathcal{T}_b$ for the equation and boundary/initial conditions.
Step 3	Specify a loss function by summing the weighted L^2 norm of both the PDE equation and boundary condition residuals.
Step 4	Train the neural network to find the best parameters $\boldsymbol{\theta}^*$ by minimizing the loss function $\mathcal{L}(\boldsymbol{\theta};\mathcal{T})$.

Figure: Physics-informed machine learning for PDE solution;

Quantum State Representation

States: **Finite Hilbert space representation....**

States

- States are vectors;
- Operations are matrices;

1. $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Quantum State Representation

States: **Finite Hilbert space representation....**

States

- States are vectors;
- Operations are matrices;

1. $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
2. **Qubit** - Superposition $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \in \mathbb{C}^2$ - (ket representation) and $\langle\psi|$ dual Hilbert Space (bra representation)

Quantum State Representation

States: **Finite Hilbert space representation....**

States

- States are vectors;
- Operations are matrices;

1. $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
2. **Qubit** - Superposition $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \in \mathbb{C}^2$ - (ket representation) and $\langle\psi|$ dual Hilbert Space (bra representation)
3. $\langle\psi|\phi\rangle$ inner product and $\rangle\psi|\phi\langle$ outer product, where $|\phi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$

Quantum State Representation

States: **Finite Hilbert space representation....**

States

- States are vectors;
- Operations are matrices;

1. $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
2. **Qubit** - Superposition $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \in \mathbb{C}^2$ - (ket representation) and $\langle\psi|$ dual Hilbert Space (bra representation)
3. $\langle\psi|\phi\rangle$ inner product and $\rangle\psi|\phi\langle$ outer product, where $|\phi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$
4. $|\phi|\psi\rangle = \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix}$ - This means that 2^n states if n qubits.

Quantum State Representation

States: **Finite Hilbert space representation....**

States

- States are vectors;
- Operations are matrices;

1. $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
2. **Qubit** - Superposition $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \in \mathbb{C}^2$ - (ket representation) and $\langle\psi|$ dual Hilbert Space (bra representation)
3. $\langle\psi|\phi\rangle$ inner product and $\rangle\psi|\phi\langle$ outer product, where $|\phi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$
4. $|\phi|\psi\rangle = \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix}$ - This means that 2^n states if n qubits.
5. **n-Qubit** state: $|\psi\rangle = \sum_{k\in[0,1]^n} \alpha_k|k\rangle \in \mathbb{C}^{2^n}$

Quantum State Representation

States: **Finite Hilbert space representation....**

States

- States are vectors;
- Operations are matrices;

1. $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
2. **Qubit** - Superposition $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \in \mathbb{C}^2$ - (ket representation) and $\langle\psi|$ dual Hilbert Space (bra representation)
3. $\langle\psi|\phi\rangle$ inner product and $\rangle\psi|\phi\langle$ outer product, where $|\phi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$
4. $|\phi|\psi\rangle = \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix}$ - This means that 2^n states if n qubits.
5. **n-Qubit** state: $|\psi\rangle = \sum_{k\in[0,1]^n} \alpha_k|k\rangle \in \mathbb{C}^{2^n}$
6. The quantum state $|\psi\rangle$, one measures the probability of the state $|k\rangle$ as $|\alpha_k|^2$

Quantum State Operations

Operations: **Logic Gates**

Circuit

- Classical Boolean circuit (**AND**, **OR** and **NOT** gates on an n-bit.
- Quantum circuit uses unitary **quantum gates**

1. NOT-gates

$$NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

2. Hadamard

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

3. Interference:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle \right) = \frac{1}{2} \left(|0\rangle + |1\rangle - |0\rangle + |1\rangle \right)$$

Quantum Circuit

Bloch Sphere: **Graph representation**

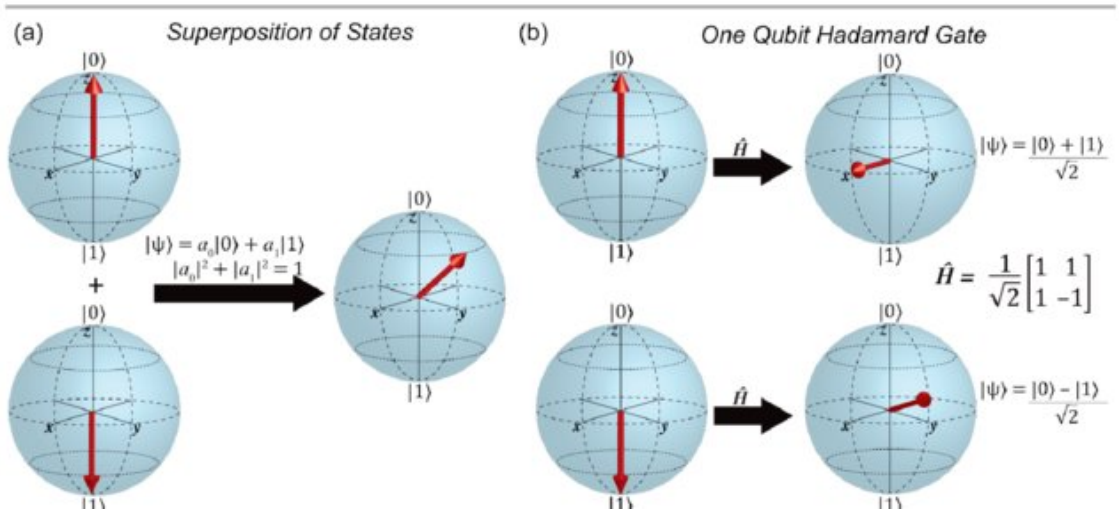


Figure: Graph representation[6]

$$|\psi\rangle = cos(\theta/2)|0\rangle - e^{i\phi} sin(\theta/2)|1\rangle \tag{26}$$

where $0 < \theta \leq 2\pi$ and $0 < \phi \leq \pi$

[6] Moreno-Pineda, Eufemio et al. Molecular spin qudits for quantum algorithms Chemical Society Reviews, 47. 2018.

Quantum Circuit

Quantum Circuit: **A set of Gates operations**

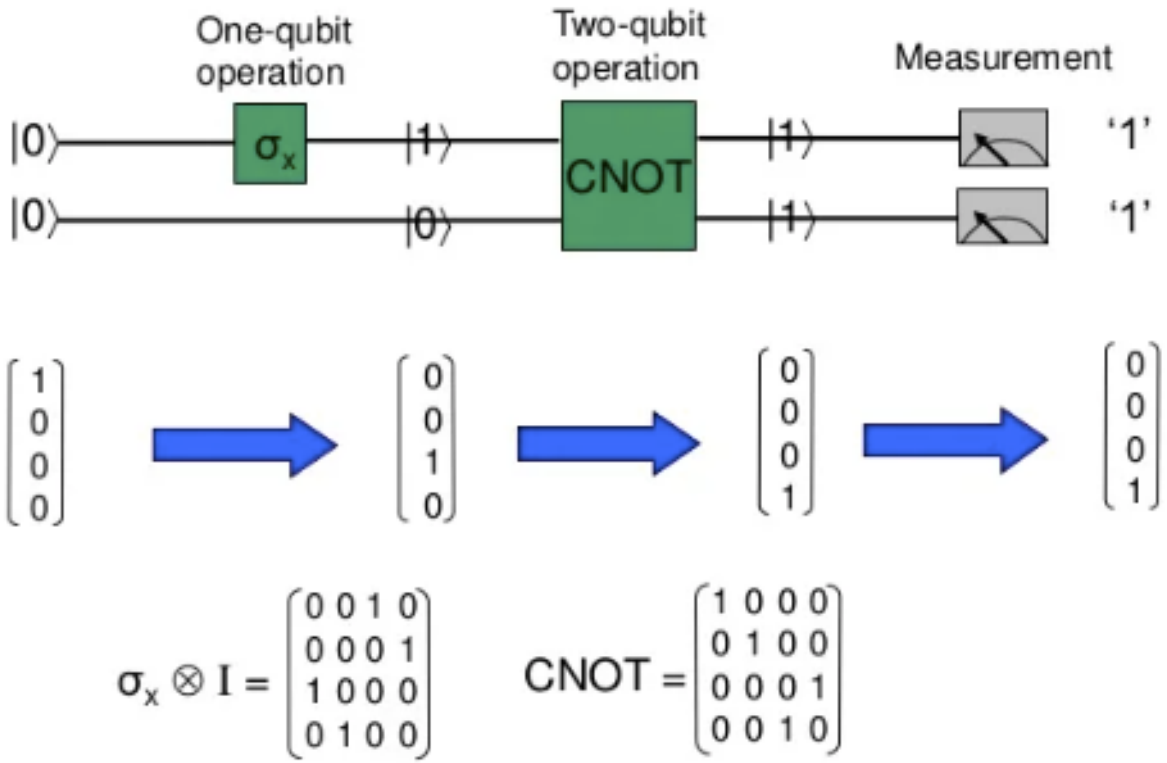


Figure: Quantum Circuit Example[7]

[7] Moreno-Pineda, Eufemio et al. Molecular spin qudits for quantum algorithms Chemical Society Reviews, 47. 2018.

Quantum computer steps

1. Start with all qubits with preparable state (init with all $|0\rangle$)

Quantum parallelism

$$U\left(\frac{1}{2^{n/2}}\sum|x\rangle|0\rangle\right) = \frac{1}{\sqrt{2^n}}\sum|x\rangle|f(x)\rangle \tag{27}$$

- This has all 2^m function values!
- However it produces one random $|x\rangle|f(x)\rangle$: Means that all will be lost!

Quantum computer steps

1. Start with all qubits with preparable state (init with all $|0\rangle$)
2. Run a circuit that produces the desireble interference: Output should interfere constructively or destructively

Quantum parallelism

$$U\left(\frac{1}{2^{n/2}}\sum|x\rangle|0\rangle\right) = \frac{1}{\sqrt{2^n}}\sum|x\rangle|f(x)\rangle \tag{27}$$

- This has all 2^m function values!
- However it produces one random $|x\rangle|f(x)\rangle$: Means that all will be lost!

Quantum computer steps

1. Start with all qubits with preparable state (init with all $|0\rangle$)
2. Run a circuit that produces the desireble interference: Output should interfere constructively or destructively
3. Measurement of final state that gives classical output

Quantum parallelism

$$U\left(\frac{1}{2^{n/2}}\sum|x\rangle|0\rangle\right)=\frac{1}{\sqrt{2^n}}\sum|x\rangle|f(x)\rangle \tag{27}$$

- This has all 2^m function values!
- However it produces one random $|x\rangle|f(x)\rangle$: Means that all will be lost!

Quantum computer steps

1. Start with all qubits with preparable state (init with all $|0\rangle$)
2. Run a circuit that produces the desireble interference: Output should interfere constructively or destructively
3. Measurement of final state that gives classical output

Quantum parallelism

$$U\left(\frac{1}{2^{n/2}}\sum|x\rangle|0\rangle\right)=\frac{1}{\sqrt{2^n}}\sum|x\rangle|f(x)\rangle \tag{27}$$

- This has all 2^m function values!
- However it produces one random $|x\rangle|f(x)\rangle$: Means that all will be lost!

Quantum computer steps

1. Start with all qubits with preparable state (init with all $|0\rangle$)
2. Run a circuit that produces the desireble interference: Output should interfere constructively or destructively
3. Measurement of final state that gives classical output

Quantum parallelism

- Suppose a classical computation $f : [0,1]^n \rightarrow [0,1]^m$

$$U\left(\frac{1}{2^{n/2}}\sum|x\rangle|0\rangle\right) = \frac{1}{\sqrt{2^n}}\sum|x\rangle|f(x)\rangle \tag{27}$$

- This has all 2^m function values!
- However it produces one random $|x\rangle|f(x)\rangle$: Means that all will be lost!

Quantum computer steps

1. Start with all qubits with preparable state (init with all $|0\rangle$)
2. Run a circuit that produces the desireble interference: Output should interfere constructively or destructively
3. Measurement of final state that gives classical output

Quantum parallelism

- Suppose a classical computation $f : [0,1]^n \rightarrow [0,1]^m$
- Convert this to quantum circuit $U : |x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$

$$U\left(\frac{1}{2^{n/2}}\sum|x\rangle|0\rangle\right) = \frac{1}{\sqrt{2^n}}\sum|x\rangle|f(x)\rangle \tag{27}$$

- This has all 2^m function values!
- However it produces one random $|x\rangle|f(x)\rangle$: Means that all will be lost!

General discretization by finite volume

Let's say:

$$\mathbf{U} = [\rho, \rho v_1, \rho v_2, \rho v_3, \epsilon_T]^T, \quad \mathbf{F} = [\rho v_1, \tau_{1,1}, \tau_{1,2}, \tau_{1,1}, \epsilon_1]^T$$

$$\mathbf{B} = [\rho v_2, \tau_{2,1}, \tau_{2,2}, \tau_{2,1}, \epsilon_2]^T, \quad \mathbf{C} = [\rho v_3, \tau_{3,1}, \tau_{3,2}, \tau_{3,1}, \epsilon_3]^T, \quad \mathbf{J} = \mathbf{0}$$

In this sense, the N-S equations can be writen as:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x_1} + \frac{\partial \mathbf{B}}{\partial x_2} + \frac{\partial \mathbf{C}}{\partial x_3} = \mathbf{J} \tag{28}$$

General discretization by finite volume

For example:

$$\frac{\partial \mathbf{F}}{\partial x_1} \approx \frac{\mathbf{F}_{I+1} - \mathbf{F}_I}{\Delta x_1} \tag{29}$$

- one billion of nodes! Means billions degrees of freedom (At least)
- Lets says around 30 qubits!!

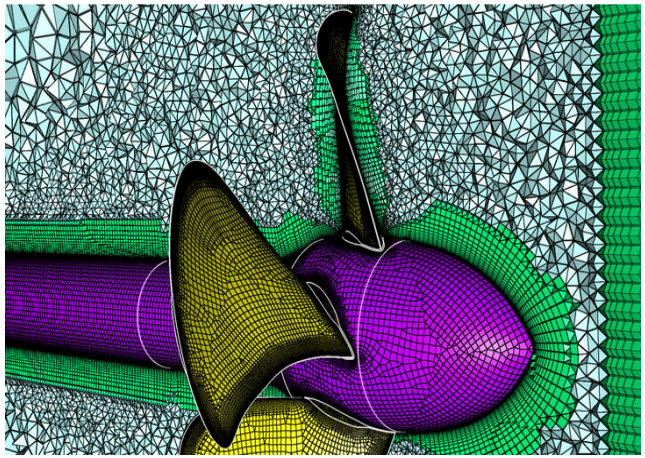


Figure: Volume discretization (mesh)
https://www.pointwise.com/

It can lead to linear system of equation:

$$\mathbf{A}\mathbf{y} = \mathbf{b} \tag{30}$$

Most of the quantum algorithm are based on the solution of this linear system of equations.

Current work applied to fluid dynamics

npj | Quantum Information

www.nature.com/npjqi

ARTICLE OPEN

Check for updates

Finding flows of a Navier–Stokes fluid through quantum computing

Frank Gaitan[1]

There is great interest in using quantum computers to efficiently simulate a quantum system's dynamics as existing classical computers cannot do this. Little attention, however, has been given to quantum simulation of a classical nonlinear continuum system such as a viscous fluid even though this too is hard for classical computers. Such fluids obey the Navier–Stokes nonlinear partial differential equations, whose solution is essential to the aerospace industry, weather forecasting, plasma magneto-hydrodynamics, and astrophysics. Here we present a quantum algorithm for solving the Navier–Stokes equations. We test the algorithm by using it to find the steady-state inviscid, compressible flow through a convergent-divergent nozzle when a shockwave is (is not) present. We find excellent agreement between numerical simulation results and the exact solution, including shockwave capture when present. Finally, we compare the algorithm's computational cost to deterministic and random classical algorithms and show that a significant speed-up is possible. Our work points to a large new application area for quantum computing with substantial economic impact, including the trillion-dollar aerospace industry, weather-forecasting, and engineered-plasma technologies.

npj Quantum Information (2020)6:61 ; https://doi.org/10.1038/s41534-020-00291-0

INTRODUCTION

In one of the earliest papers motivating the development of quantum computers Feynman[1] pointed out that such computers are better suited to simulate the dynamics of a quantum system than existing classical digital computers. This is because classical computers require computational resources that scale exponentially with the size of the simulated quantum system, while quantum computers do not. It is widely expected that quantum simulation of quantum systems will be a major application area for quantum computing.

Quantum systems are not unique in being difficult to simulate. Many important classical systems exist whose dynamics strongly couple many degrees of freedom over multiple length-scales, making their simulation with classical computers hard. An example is a viscous fluid whose flows satisfy the Navier–Stokes nonlinear partial differential equations[2–5] (PDEs). Solving these PDEs is the primary task for such diverse problems as aerospace flight vehicle design, weather-forecasting, probing the dynamics of turbulence, and determining the magneto-hydrodynamics of plasmas in space and in earth-bound technologies. In spite of its importance, little work has been done on finding quantum algorithms for the Navier–Stokes equations. The earliest work we thus extends to a general quantum algorithm for solving PDEs. To test our quantum algorithm, we use it to find the steady-state inviscid, compressible flow through a convergent-divergent nozzle. Such nozzles are used in rocket engines and in jet engines for supersonic flight and display a rich range of flows, from smooth acceleration from subsonic to supersonic speeds, to the appearance of a normal shockwave in the divergent part of the nozzle, which causes supersonic flow to decelerate to subsonic speeds as it crosses the narrow shock region. We find excellent agreement between numerical simulation of our algorithm and the exact flow solution, including shockwave capture when present. We then compare the quantum algorithm's computational cost to deterministic and random classical algorithms, and discuss flow regimes where there is a computational advantage. Lastly, we point to the large new application area our quantum PDE algorithm opens up for quantum computing which includes many economically important problems. The quantum algorithm we present is independent of the previously cited papers, and of a quantum algorithm for elliptic PDEs[19].

RESULTS

Computers and Fluids 173 (2018) 22–28

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Parallel evaluation of quantum algorithms for computational fluid dynamics☆

René Steijl[1,*], George N. Barakos[2]

University of Glasgow, School of Engineering, James Watt South Building, G12 8QQ, UK

ARTICLE INFO

Article history:
Received 24 January 2017
Revised 21 March 2018
Accepted 28 March 2018
Available online 29 March 2018

MSC:
00-01
99-00

Keywords:
Quantum computing
Quantum computer simulator
Vortex-in-cell method

ABSTRACT

The development and evaluation of quantum computing algorithms for computational fluid dynamics is described. A hybrid classical/quantum hardware approach is assumed where selected computationally intensive parts of the solver are implemented as quantum circuits. The vortex-in-cell method is considered as an example where the Quantum Fourier Transform is used to build a Poisson solver. Computational aspects of simulating the required quantum circuits on a classical parallel computer are discussed including an analysis of the required data exchanges for a distributed-memory parallelization using message-passing. The effect of errors and noise in the quantum algorithm on the flow solution is analyzed and it is shown that despite inevitable noise and uncertainties, meaningful flow simulations can be performed using a hybrid classical/quantum hardware approach. An improved version of the vortex-in-cell method with increased resilience to noise is also discussed along with suggestions for future steps. The presented work is limited to a single CFD algorithm. However, building on this work, a broader range of algorithms will be considered in future work.

1. Introduction

In recent years the field of quantum computing (QC) [1] has grown into an active and diverse field of research and significant progress has been made with building quantum computers. For a small number of applications, quantum algorithms have been developed that would lead to a significant speed-up relative to classical methods when executed on a suitable quantum computer. Despite this research effort, progress in defining suitable applications for quantum computers has been relatively limited and two decades after their invention, Shor's algorithm for factoring composite integers and Grover's algorithm for quantum search are still among the main applications. Further applications have been de-

putational fluid dynamics applications. In the absence of the required quantum hardware, large-scale parallel simulations on parallel classical computers are required in developing such algorithms. A hybrid classical/quantum hardware approach is assumed where selected computationally intensive parts of the solver are implemented as quantum circuits.

Quantum computer applications aim to achieve a computational speed-up relative to classical computers by using unique quantum effects. The first and most important for the algorithms considered here is quantum parallelism, based on the use of qubits representing a superposition of infinitely many possible states of a binary system. Quantum entanglement and quantum teleportation are the other main quantum effects used to create computational capabil-

In this two case, the PDEs are discretized[8]. The system of equations produzed is solved either by HHL or a Variational Quantum algorithms[9].

[8] René Steijl, George N. Barakos, Computers & Fluids, Volume 173, 2018, Pages 22-28.

[9] Gaitan, F. npj Quantum Inf 6, 61 (2020)

Current work applied to fluid dynamics

Towards Solving the Navier-Stokes Equation on Quantum Computers

N. Ray,[1]* T. Banerjee[2], B. Nadiga[3],S. Karra[2]
[1]Computer, Computational and Statistical Sciences (CCS-7)
[2]Computational Earth Science Group (EES-16)
[3]Computational Physics and Methods (CCS-2)
Los Alamos National Laboratory, Los Alamos, NM 87545

April 22, 2019

abstract

In this paper, we explore the suitability of upcoming novel computing technologies, in particular adiabatic annealing based quantum computers, to solve fluid dynamics problems that form a critical component of several science and engineering applications. We start with simple flows with well-studied flow properties, and provide a framework to convert such systems to a form amenable for deployment on such quantum annealers. We analyze the solutions obtained both qualitatively and quantitatively as well as the sensitivities of the various solution selection schemes on the obtained solution.

keywords: Quantum computing – quantum annealers – fluid dynamics – turbulence – linear systems.

Adiabatic annealing-based quantum computers (quenching Ising type model)[10]

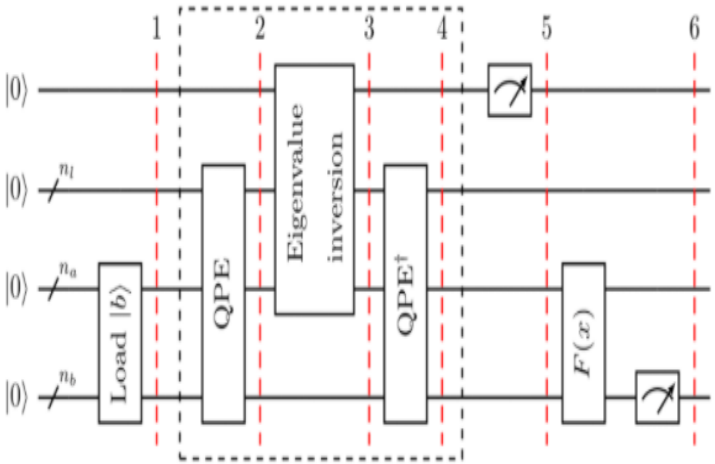
[10] N. Ray, T. Banerjee, B. Nadiga, S. Karra, "Towards Solving the Navier-Stokes Equation on Quantum Computers," arXiv:1904.09033v1 [cs.NA] 16 Apr 2019 (hardware specific)

Harrow-Hassidim-Lloy Quantum Linear Solver

Harrow-Hassidim-Lloy Algorithm [a]

[a] Phys. Rev. Lett. 103.15 (2009), p. 150502

1. $\mathbf{A}|\mathbf{y}\rangle = |\mathbf{b}\rangle$

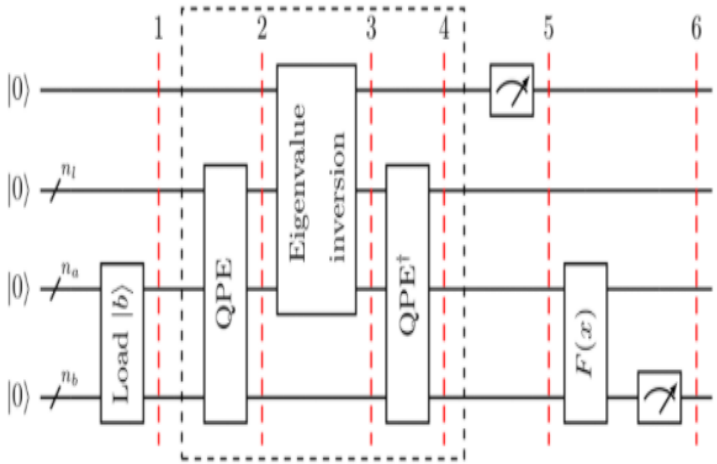


Harrow-Hassidim-Lloy Quantum Linear Solver

Harrow-Hassidim-Lloy Algorithm [a]

[a] Phys. Rev. Lett. 103.15 (2009), p. 150502

1. $\mathbf{A}|\mathbf{y}\rangle = |\mathbf{b}\rangle$
2. $\mathbf{A} = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle\langle u_j|, \quad \lambda_j \in \Re$

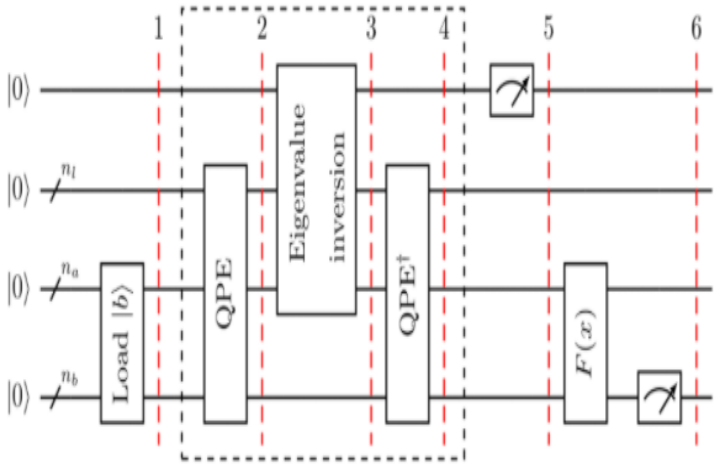


Harrow-Hassidim-Lloy Quantum Linear Solver

Harrow-Hassidim-Lloy Algorithm [a]

[a] Phys. Rev. Lett. 103.15 (2009), p. 150502

1. $\mathbf{A}|\mathbf{y}\rangle = |\mathbf{b}\rangle$
2. $\mathbf{A} = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle\langle u_j|, \quad \lambda_j \in \Re$
3. $\mathbf{U} = e^{i\mathbf{A}t} = \sum_{j=0}^{N-1} e^{i\lambda_j t}|u_j\rangle\langle u_j|$

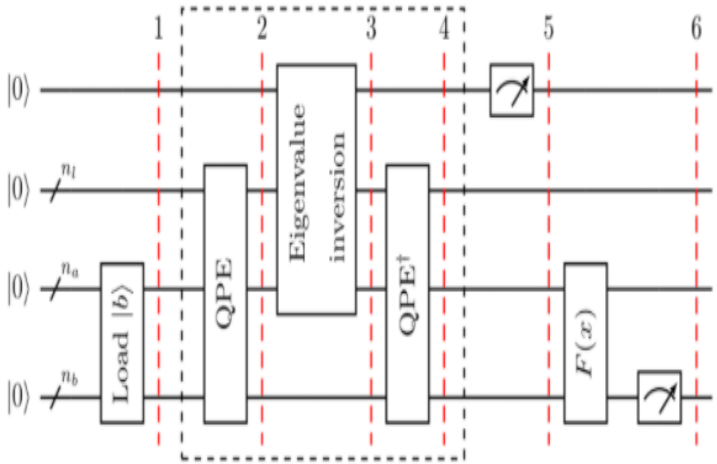


Harrow-Hassidim-Lloy Quantum Linear Solver

Harrow-Hassidim-Lloy Algorithm [a]

[a] Phys. Rev. Lett. 103.15 (2009), p. 150502

1. $\mathbf{A}|\mathbf{y}\rangle = |\mathbf{b}\rangle$
2. $\mathbf{A} = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle\langle u_j|, \quad \lambda_j \in \Re$
3. $\mathbf{U} = e^{i\mathbf{A}t} = \sum_{j=0}^{N-1} e^{i\lambda_j t}|u_j\rangle\langle u_j|$
4. $\mathbf{A}^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1}|u_j\rangle\langle u_j|$

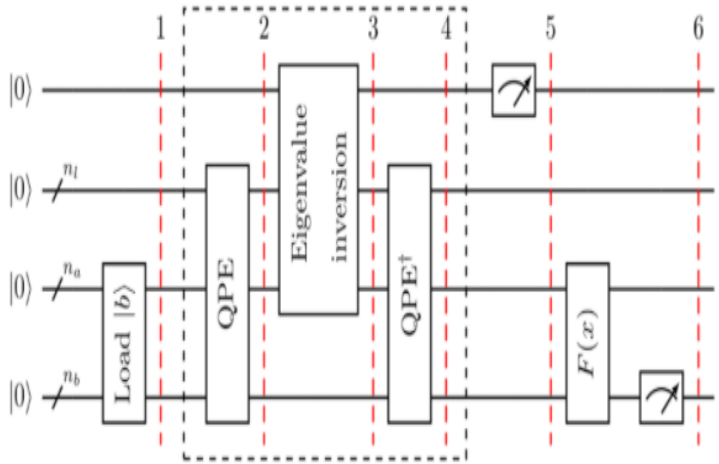


Harrow-Hassidim-Lloy Quantum Linear Solver

Harrow-Hassidim-Lloy Algorithm [a]

[a] Phys. Rev. Lett. 103.15 (2009), p. 150502

1. $\mathbf{A}|\mathbf{y}\rangle = |\mathbf{b}\rangle$
2. $\mathbf{A} = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle\langle u_j|, \quad \lambda_j \in \Re$
3. $\mathbf{U} = e^{i\mathbf{A}t} = \sum_{j=0}^{N-1} e^{i\lambda_j t}|u_j\rangle\langle u_j|$
4. $\mathbf{A}^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1}|u_j\rangle\langle u_j|$
5. $|\mathbf{y}\rangle = \mathbf{A}^{-1}|\mathbf{b}\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} b_j |u_j\rangle$

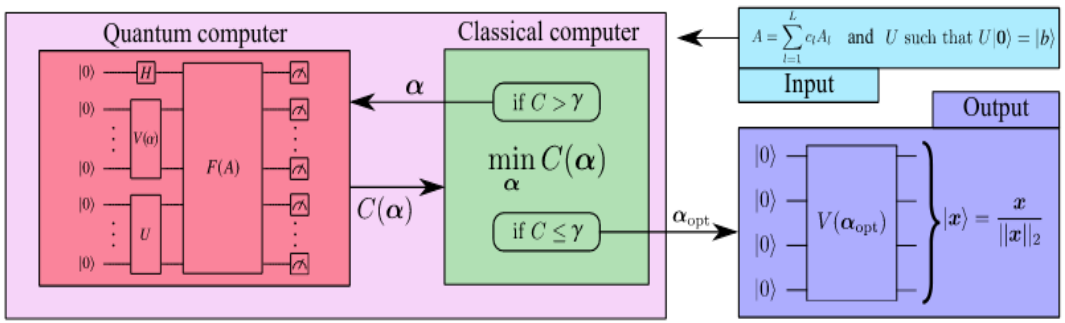


Variational Quantum Linear Solver

Variational Quantum Linear Solver (VQLS)

- Tested with problem size of 1024×1024.[a];
- The input to VQLS is a matrix A written as a linear combination of unitaries $\mathbf{A}$ and a short-depth quantum circuit U which prepares the state $|\mathbf{b}\rangle$
- The hybrid quantum-classical optimization loop until the cost is below a user-specified threshold
- When the loop terminates, one obtains $|\mathbf{y}\rangle = \mathbf{y}/||\mathbf{y}||_2$

[a] Carlos Bravo-Prieto et al. 2020.



Quantum Annealing Linear Solver

Quantum Annealing Linear Solver

- Transform in a binary matrix;
- Quadratic unconstrained binary optimization (QUBO) specific for DWave QPU

$$\frac{\partial u_x}{\partial t} = -\frac{1}{\rho}\frac{\partial \rho}{\partial x} + \mu \frac{\partial^2 u_x}{\partial x^2} \tag{31}$$

After discretization:

$$\mathbf{Ay} = \mathbf{b}, \quad y_j = \sum_{j=1}^{np} 2^{j_0 - j} q_j, \quad \mathbf{A}^q \mathbf{q} = \mathbf{b}$$

$$\bar{q} = min||\mathbf{A}^q \mathbf{q} - \mathbf{b}||_2^2 \tag{32}$$

$$f(\mathbf{q}) = \sum_i \mu_i q_i + \sum_{i<j} \omega_{i,j} q_i q_j, \quad \mu_i = \sum_i A^q_{i,j}(A^q_{i,j} - 2b_i), \quad \omega_{i,k} = 2\sum_i A^q_{i,j} A^q_{i,k}$$

(33)

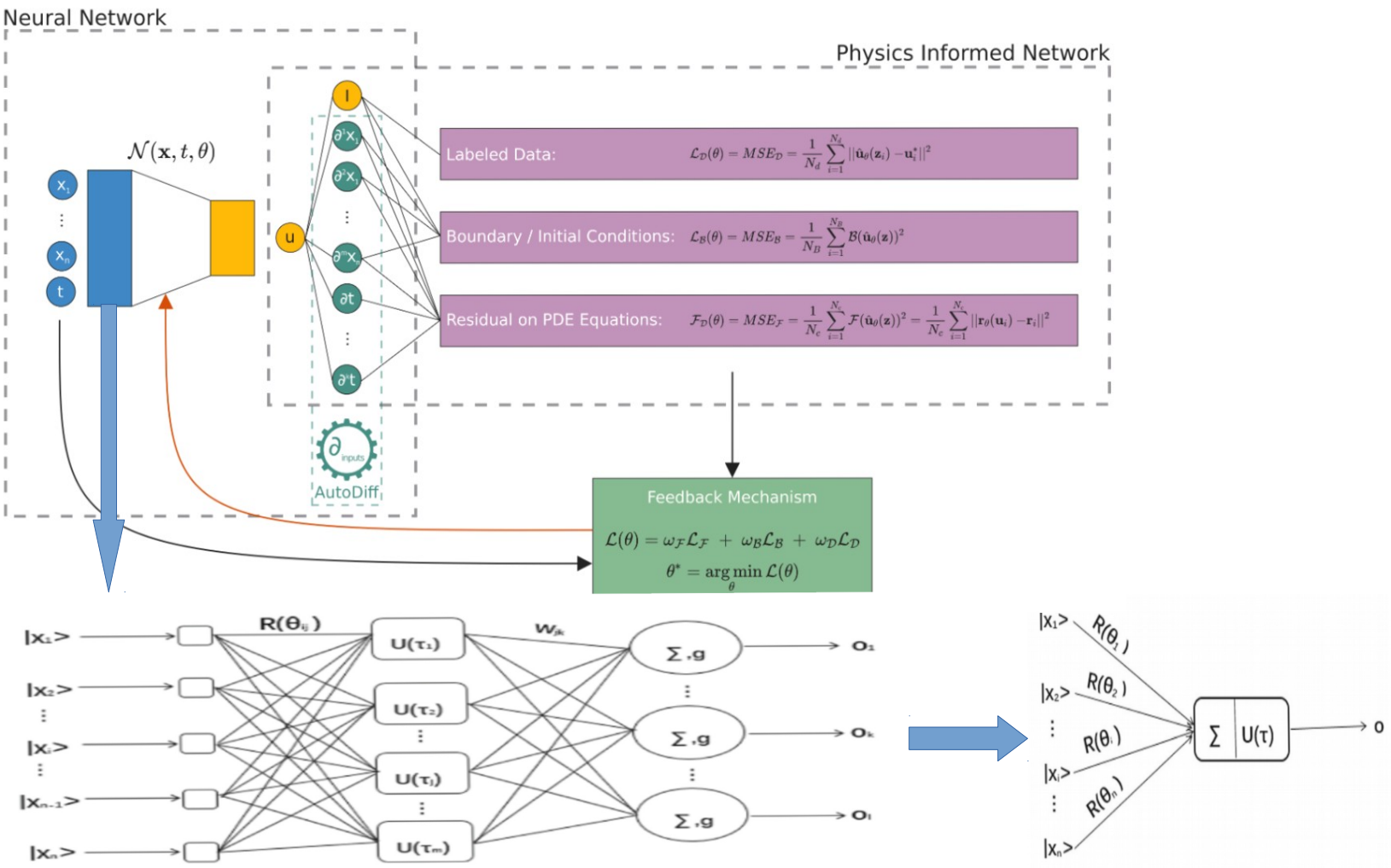
How about Neural network can help?

How about Neural network in quantum computer?

This raises another question: how Neural network can help ?

Methodologies

Quantum Physics-informed Neural Networks Description[11]:



[11] Figure adapt from Chen and Niu, and Cuomo et al.

Quantum Neural Network description

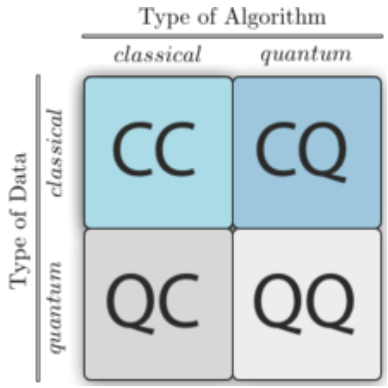
QNN and NN similarities

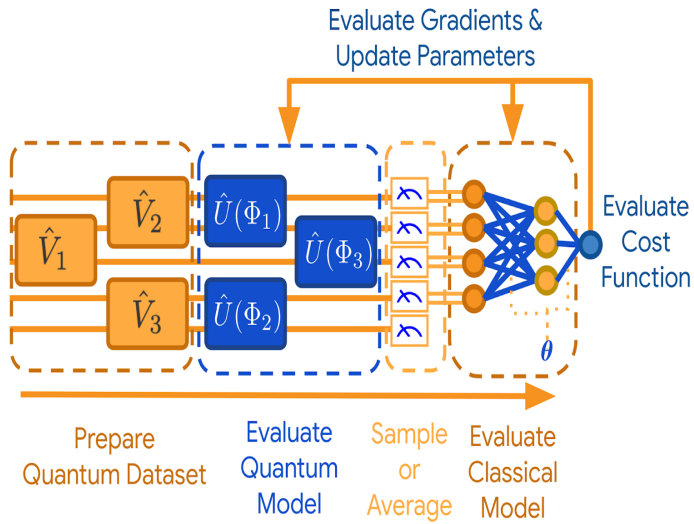
- Input data for trainning and testing - **Can be classical or quantum data**[a].
- Struture - Input Layer, Hidden layers and output layers - **Similar for QNN**

[a] https://www.tensorflow.org/quantum

QNN and NN similarities

- Based on θ - **Similar for QNN**
- Non-linearity and optimization - **Hybrid step - Quantum and classical**
- Backpropagation - **Hybrid step - Quantum and classical**

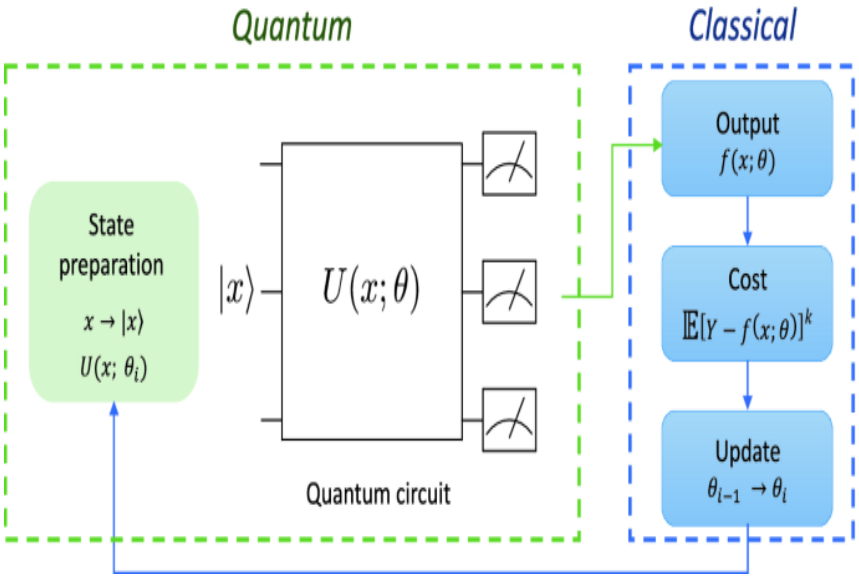




Quantum Neural Network description

Quantum neural network

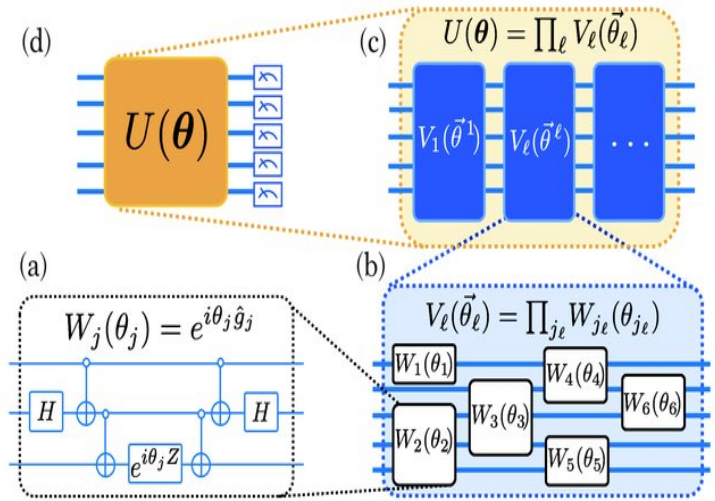
- General form $QNN(x, \theta)$:



Quantum circuit

- Quantum circuit representation[a]:

[a] Abbas, A., Sutter, D., Zoufal, C. et al. The power of quantum neural
https://doi.org/10.1038/s43588-021-00084-1



Quantum PINN formulation

PDE general form

Consider the general system PDE as:

$$\mathbf{f}(\mathbf{x}; \frac{\partial u_1}{\partial x_1}, \cdots, \frac{\partial^2 u_m}{\partial x_1 \partial x_d}, \cdots; \lambda) = 0, \quad \mathbf{x} \in \Omega, \tag{34}$$

with boundary conditions:

$$B(\mathbf{x}, \mathbf{u}) = 0 \quad on \quad \partial\Omega \tag{35}$$

$$I(\mathbf{x}, \mathbf{u}, \lambda), \quad for \quad \mathbf{x} \in \tau_j \tag{36}$$

Can we formulate a Quantum PINN based on Quantum Circuit?

- Let's say:

$$u(\mathbf{x}) \approx QNN(\mathbf{x}; \theta) \tag{37}$$

- We can rewrite the PDE general form as a fuction of the QC because we could compute: $\frac{\partial QNN(\mathbf{x};\theta)}{\partial \mathbf{x}}$ - automatic shifted differentiation quantum circuits

Universal approximation for Quantum Neural Network

Working Hypothesis: *QNN can always be used as an universal approximation.*

Tools

- QNN libraries against their classical counterpart for regression:
  - TensorFlow Quantum (TFQ) based on Cirq.
  - IBM QNN based on Qiskit.
- Implementation of a specialized Quantum Neural Networks:
  - Quantum Convolutional Neural Networks (QCNN) with Bayesian Learning Initialization (BLI).
  - Avoid barren plateau (BP).

Measurement: *(i) Memory capacity (ii) Generalization power (iii) Speed-up ? (iv) Fisher information matrix.*

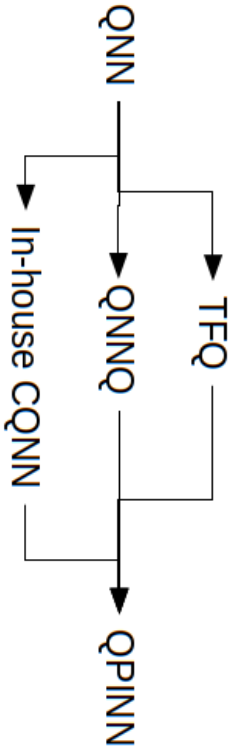
Expected: *A better understanding of QNN for regression and absence of barren plateau.*
Risk Assessment: Medium-high risk
Significance: Overcome the barren plateaus.

A Quantum Physics-Informed Neural Network (QPINN)

Working Hypothesis: *QPNN can solve different type of non-linear PDE.*

Challenges

- Can it be done by QNN libraries based: TFQ and QNNQ.
- Specialized Quantum PINN:
  - Quantum Convolutional Neural Networks (QCNN) with Bayesian Learning Initialization (BLI).
  - Avoid barren plateau (BP).



Expected: *a PDE quantum solver.*
Risk Assessment: Medium-high risk
Significance: This might help to solve several currently impossible problems to answer that demand intensive computation.

Global Risk

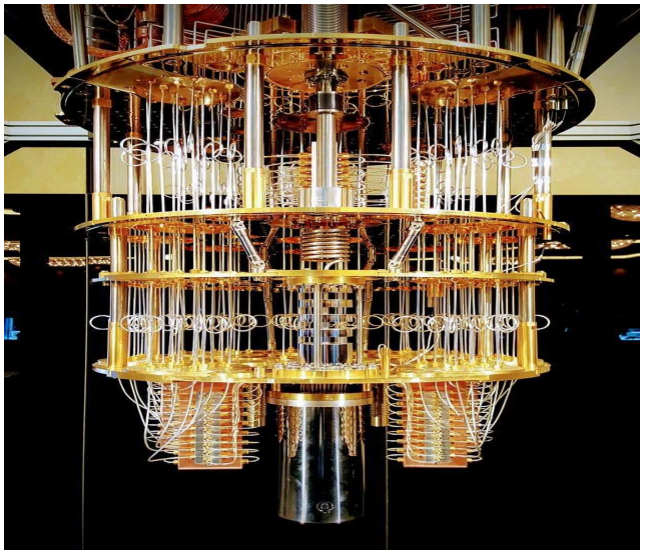
Global Risk

1. These algorithms depend on the development of a more powerful quantum processors.

Noise Intermediate Scale Quantum (NISQ) era:

- Limitation of current quantum computers:
  1. Limitation of number of qubits;

- Google has already proved quantum supremacy on a quantum computer, and several countries, companies, and research centers are investigating this new scientific paradigm.



IBM Q quantum computer.

Global Risk

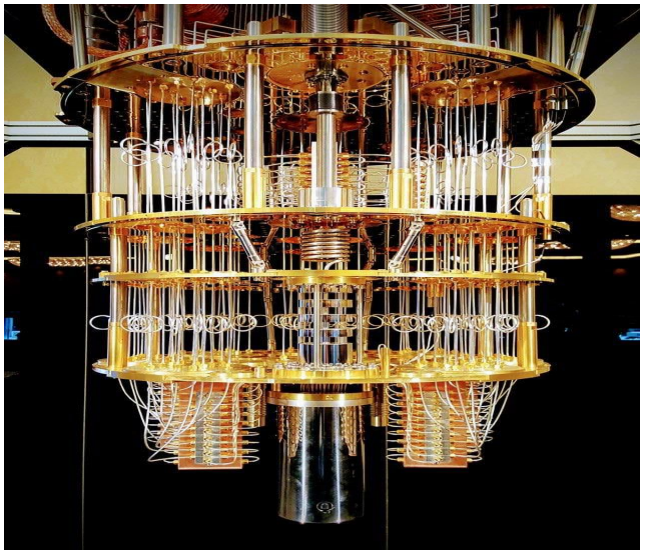
Global Risk

1. These algorithms depend on the development of a more powerful quantum processors.

Noise Intermediate Scale Quantum (NISQ) era:

- Limitation of current quantum computers:
  1. Limitation of number of qubits;
  2. Noise sources limit the number of operations;

- Google has already proved quantum supremacy on a quantum computer, and several countries, companies, and research centers are investigating this new scientific paradigm.



IBM Q quantum computer.

Global Risk

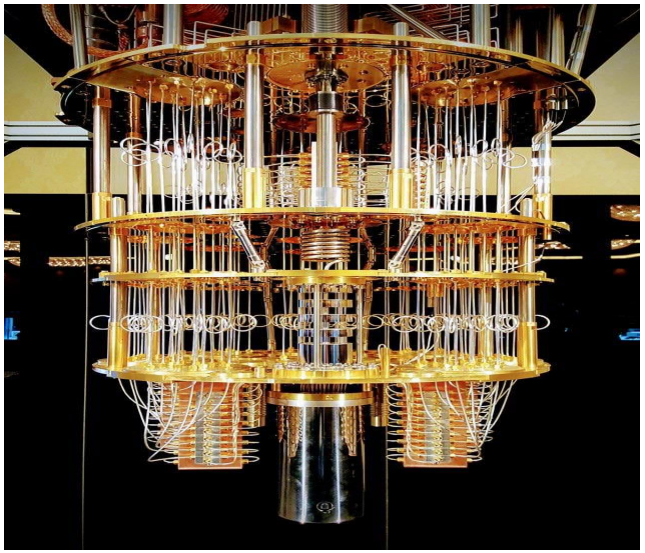
Global Risk

1. These algorithms depend on the development of a more powerful quantum processors.

Noise Intermediate Scale Quantum (NISQ) era:

- Limitation of current quantum computers:
  1. Limitation of number of qubits;
  2. Noise sources limit the number of operations;
  3. Problems of error correction;

- Google has already proved quantum supremacy on a quantum computer, and several countries, companies, and research centers are investigating this new scientific paradigm.



IBM Q quantum computer.

Questions? I have several questions![12]

[12] https://www.linuxfoundation.org/the-linux-mark/