Introduction to the FRIB-TA Summer School: Quantum Computing and Nuclear Few- and Many-Body Problems with teaching plan and learning outcomes

Facility for Rare Isotope Beams, Michigan State University, USA

June 20

Introduction

Recent developments in quantum information systems and technologies offer the possibility to address some of the most challenging large-scale problems in science, whether they are represented by complicated interacting quantum mechanical systems or classical systems. The last years have seen a rapid and exciting development in algorithms and quantum hardware. The emphasis of this summer school is to highlight, through a series of lectures and hands-on exercises and practice sessions, how quantum computing algorithms can be used to study nuclear few- and many-body problems of relevance for low-energy nuclear physics. And how quantum computing algorithms can aid in studying systems with increasingly many more degrees of freedom compared with more classical few- and many-body methods. Several quantum algorithms for solving quantum-mechanical few- and many-particle problems with be discussed. The lectures will start with the basic ideas of quantum computing. Thereafter, through examples from nuclear physics, we will elucidate how different quantum algorithms can be used to study these systems. The results from various quantum computing algorithms will be compared to standard nuclear few- and many-body methods

Organizers: Alexei Bazavov (MSU), Scott Bogner (MSU), Heiko Hergert (MSU), Matthew Hirn (MSU), Morten Hjorth-Jensen (MSU), Dean Lee (MSU), Huey-Wen Lin (MSU), and Andrea Shindler (MSU) Morten Hjorth-Jensen, hjensen@msu.edu

Aims and Learning Outcomes

The following topics will be covered.

1. Basic elements of quantum computing (first day) with introduction to relevant software, including
   (a) Introduction to quantum computing, qubits and systems of qubits
   (b) Measurements, Superposition and Entanglement
   (c) Gates, unitary transformations and quantum circuits
   (d) Quantum algorithms and implementation on a real quantum computer
2. Simulating quantum-mechanical few- and many-body systems
   (a) Quantum algorithms for quantum mechanical systems
   (b) Quantum simulation of the Schroedinger equation
   (c) Quantum computing and nuclear few- and many-body systems
   (d) Quantum state preparation and Quantum simulations
   (e) Quantum simulations on a real quantum computer
3. Quantum field theory and quantum computing
4. Noise, error correction and mitigation

All of the above topics will be supported by examples, hands-on exercises and project work.

Practicalities

1. Lectures Monday through Wednesday, starting at 830am, see schedule below
2. Hands-on sessions before lunch and in the afternoons till 6pm
3. For all lecture days we provide relevant jupyter-notebooks you can work on

Learning material and resources

1. Qiskit textbook, free online, see https://qiskit.org/textbook/preface.html
2. Scherer, The Mathematics of Quantum Computing, see https://link.springer.com/book/10.1007/978-3-030-12358-1
3. Chuang and Nielsen, Quantum Computation and Quantum Information, https://www.cambridge.org/highereducation/books/quantum-computation-and-quantum-informa 01E10196D0A682A6AEFFEA52D53BE9AE#overview
4. Hundt, Quantum Computing for programmers, https://www.cambridge.org/core/books/quantum-computing-for-programmers/BA1C887BE4AC0D0D5653E71FFBEF61C6

Good resources. With the hands-on programming component we strongly recommend that you install Qiskit on your computer before the school starts.

1. For Qiskit, follow the instructions at https://qiskit.org/documentation/getting_started.html
2. We strongly recommend using the Jupyter notebook environment at https://quantum-computing.ibm.com/. This environment has Qiskit already set up and is free, just requires an email to register. It has built in support for Jupyter notebooks and should be sufficient for everything needed.
3. See also Ryan Larose's (from 2019) Quantum computing bootcamp with Qiskit - https://github.com/rmlarose/qcbq.
4. See also https://www.ryanlarose.com/external-resources.html

Detailed lecture plan

The duration of each lecture is approximately 45-50 minutes and there is a small break of 10-15 minutes between each lecture. Longer breaks at 1030am-11am and 3pm-330pm, except for Monday where there is also the possibility fora guided FRIB tour. In-person attendance is the main teaching modus, but lectures and hands-on sessions will be broadcasted via zoom for those who cannot attend in person. The zoom link will be sent to those who have expressed that they cannot attend in person. The lectures will also be recorded.

Teachers.

- AB = Alexei Bazavov
- BH = Benjamin Hall
- DL = Dean Lee
- JW = Jacob Watkins
- JB = Joey Bonitati
- MHJ = Morten Hjorth-Jensen
- RL = Ryan Larose
- QZ - Zhenrong Qian

Monday June 20.

- 8am-830am: Welcome and registration
- 830am-930am: Introduction to quantum computing, qubits, systems of qubits, gates and quantum circuits (AB)
- 930am-1030am: Measurements, Superposition, Entanglement (AB)
- 1030am-11am: Break, coffee, tea etc
- 11am-12pm: Hands-on session with applications and introduction to software libraries (AB, JW, RL)
- 12pm-1pm: Lunch (shorter lunch, else 1h30m lunches)
- 1pm-2pm: Quantum algorithms (Time evolution) for solving quantum mechanical problems (DL, JB, JW, and ZQ), simple problems
- 2pm-3pm: Quantum algorithms (Phase estimation) for solving quantum mechanical problems (DL, JB, JW, and ZQ), simple problems
- 3pm-4pm: Break, coffee, tea or tour for FRIB for those interested. Please let us know if you are interested in a tour of FRIB.
- 4pm-6pm: Hands-on sessions and problem solving (AB+all)

Tuesday June 21.

- 830-930am: Time evolution, Suzuki-Trotter approximation and quantum simulations (DL and JW) (advanced topic)
- 930am-1030am: Introduction to VQE and simple model (BH)
- 1030am-11am: Break, coffee, tea etc
- 11am-12pm: Many-body theory and nuclear few- and many-body systems (BH and MHJ)
- 12pm-130pm: Lunch
- 130pm-230pm: Quantum algorithms (VQE) and nuclear physics with applications (BH and MHJ), part 1
- 230pm-330pm: Quantum algorithms (VQE) and nuclear physics with applications (BH and MHJ), part 2
- 330pm-4pm: Break, coffee, tea etc
- 4pm-6pm: Hands-on sessions and problem solving (BH and JW+all)

Wednesday June 22.

- 830am-930am: Noise, error correction and mitigation, part I (RL)
- 930am-1030am: Noise, error correction and mitigation, part II (RL)
- 1030am-11am: Break, coffee, tea etc
- 11am-12pm: Practicing error correction and mitigation, hands-on part (RL)
- 12pm-130pm: Lunch
- 130pm-230pm: Wrapping up and defining nuclear many-body system to study for hands-on session (All)
- 230pm-330pm: Start hands-on session (RL)
- 330pm-4pm: Break, coffee, tea etc
- 4pm-6pm: Hands-on sessions and problem solving (all)

Prerequisites

You are expected to have operating programming skills in programming languages like Python (preferred) and/or Fortran, C++, Julia or similar and knowledge of quantum mechanics at an intermediate level (senior undergraduate and/or beginning graduate). Knowledge of linear algebra is essential. Additional modules for self-teaching on Python and quantum mechanics are also provided.

Software and needed installations

We will make extensive use of Python as programming language and its myriad of available libraries. You will find Jupyter notebooks invaluable in your work.

If you have Python installed (we strongly recommend Python3) and you feel pretty familiar with installing different packages, we recommend that you install the following Python packages via **pip** as

1. pip install numpy scipy matplotlib ipython scikit-learn mglearn sympy pandas pillow

For Python3, replace **pip** with **pip3**.

For OSX users we recommend, after having installed Xcode, to install **brew**. Brew allows for a seamless installation of additional software via for example

1. brew install python3

For Linux users, with its variety of distributions like for example the widely popular Ubuntu distribution, you can use **pip** as well and simply install Python as

1. sudo apt-get install python3 (or python for pyhton2.7)

etc etc.

Python installers

If you don't want to perform these operations separately and venture into the hassle of exploring how to set up dependencies and paths, we recommend two widely used distrubutions which set up all relevant dependencies for Python, namely

- Anaconda,

which is an open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system **conda**.

- Enthought canopy

is a Python distribution for scientific and analytic computing distribution and analysis environment, available for free and under a commercial license.

Furthermore, Google's Colab is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. Try it out!

Useful Python libraries

Here we list several useful Python libraries we strongly recommend (if you use anaconda many of these are already there)

- NumPy is a highly popular library for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays
- The pandas library provides high-performance, easy-to-use data structures and data analysis tools
- Xarray is a Python package that makes working with labelled multi-dimensional arrays simple, efficient, and fun!
- Scipy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering.
- Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
- Autograd can automatically differentiate native Python and Numpy code. It can handle a large subset of Python's features, including loops, ifs, recursion and closures, and it can even take derivatives of derivatives of derivatives
- SymPy is a Python library for symbolic mathematics.

- scikit-learn has simple and efficient tools for machine learning, data mining and data analysis
- TensorFlow is a Python library for fast numerical computing created and released by Google
- Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano
- And many more such as pytorch, Theano etc

All learning material and teaching schedule pertinent to the course is avaliable at this GitHub address. A simple **git clone** of the material gives you access to all lecture notes and program examples. Similarly, running a **git pull** gives you immediately the latest updates.