Project 4 (Quantum physics-informed neural networks [QPINN]).

Literature:

- `https://pennylane.ai/qml/glossary/quantum_differentiable_programming/` (*)
- `https://doi.org/10.1103/PhysRevA.103.052416` (**)
- `https://doi.org/10.1103/PhysRevA.104.052417` (***)
- `https://arxiv.org/abs/2306.17026` (****)
- `https://arxiv.org/abs/2308.01827`

Objectives and tasks:

- Learn and understand what a (quantum) physics-informed neural network is.
  - How is the "physics" (e.g., an (initial-)boundary value problem) integrated into the loss function in a physics-informed neural network (PINN)?
  - What does it mean to differentiate the neural network to compute gradients with respect to variables in the physical space, i.e. $(x, y, z, t) \in \mathbb{R}^3 \times [0, T]$?
  - How do you compute gradients with respect to $(x, y, z, t) \in \mathbb{R}^3 \times [0, T]$ in QPINNs?
  - How to you feed inputs $(x, y, z, t) \in \mathbb{R}^3 \times [0, T]$ into the QPINN?
- Provide an implementation of a QPINN for an (initial-)boundary value problem of your choice. **Remark** *It is recommended to start with a simple, e.g., linear ordinary differential equation (ODE) problem and increase the complexity step by step. Beware that training the QPINN becomes more and more challenging if the "physics" becomes more complicated (e.g., nonlinear, vector-valued, multi-dimensional).*
  - Implement one of the variational quantum ansätze (e.g., the hardware-efficient ansatz or the alternating blocks ansatz) suggested in the paper marked (**).
  - Implement the product feature map and its derivatives from (**). You can find additional information on how to compute gradients of quantum circuits in (***).
  - Validate your implementation for a simple ODE problem, e.g.,

    $$\frac{du}{dt}(t) + \lambda u\,(\kappa + \tan(\lambda t)) = 0, \quad u(t = 0) = u_0$$

    for which you know the analytical solution

    $$u(t) = \exp(-\kappa\lambda t)\cos(\lambda x) + c,$$

    where λ and κ are problem parameters, u_0 the initial solution value at time $t = 0$ and c a constant that depends on u_0. Fig. 6 from the paper marked (**) gives some references for problem configurations.

  - How does your QPINN perform for values of t inside the training interval $[0, T]$? How does it perform for $t > T$, i.e., how good can it extrapolate?

- Extend your implementation of the QPINN in one or more of the following aspects
  - Implement Chebyshev feature maps (cf. papers marked (**) and (****) and compare their performance to the product feature maps.
  - Extend your QPINN to a nonlinear ODE problem, e.g., the one from the paper marked (**)