High Energy Physics and Quantum Computing

2024 IonQ 4th week meeting

Weekly goal

[Week 1] Review the reference [1] thoroughly and understand basic idea.

[Week 2~3] Implement the idea using Qiskit and/or **PennyLane**, and reproduce their results for two examples in the paper.

[Week 4] Then consider a more complex and more realistic example, taking electron-positron production at the Large Hadron Collider ($pp \to e^+e^-$). Effectively, this problem involves four-dimensional integration.

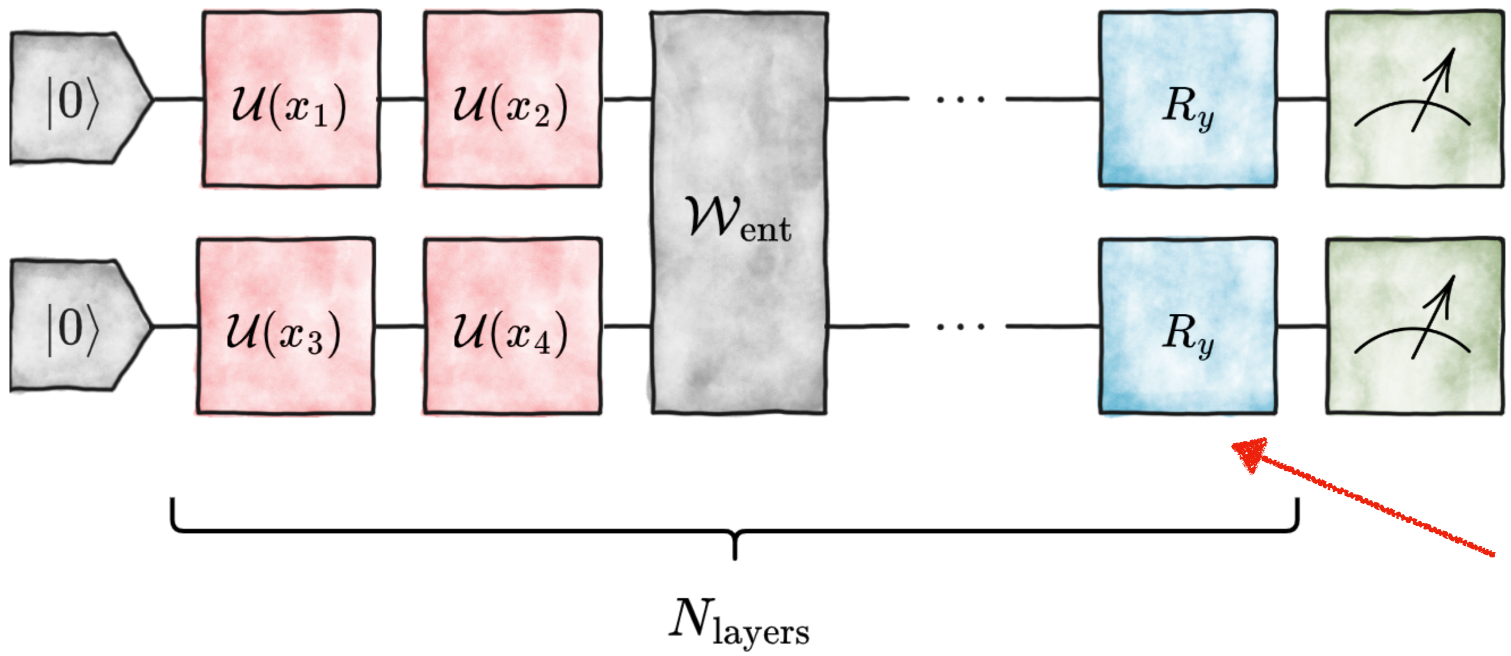
[Week 5] Try to optimize the circuits with different choice of the cost function or variations of quantum circuits.

[Week ?] Study Monte-Carlo sampling with above integration method, and how to implement the importance sampling into a variational quantum circuit. (Check Ref. [2])

Methodology

- Data reuploading model

$$\mathcal{U}(x|\theta) = R_z(\theta_1)R_y(\theta_2)R_z(\theta_3)\boxed{R_z(\theta_4\, x)}R_y(\theta_5)$$



Question? Why R_y?

The $U(x_i)$ quantum channel corresponds to the fundamental Fourier Gate, while the entangling channel W_{ent} is built with a combination of CZ gates.

```
GoodScaling (dims=4, nqubits=2, nlayers=2) running in process id: 1235043
Circuit drawing:

q0: ─RY─RZ─RZ─RY─RZ─RY─RZ─RZ─RY─RZ─o─RY─RZ─RZ─RY─RZ─RY─RZ─RZ─RY─RZ─o─RY─M─
q1: ─RY─RZ─RZ─RY─RZ─RY─RZ─RZ─RY─RZ─Z─RY─RZ─RZ─RY─RZ─RY─RZ─RZ─RY─RZ─Z─RY─M─
Circuit summary:
    Circuit depth = 24
Total number of gates = 46
Number of qubits = 2
Most common gates:
rz: 24
ry: 18
cz: 2
measure: 2
```

Methodology - training

$$I(\overrightarrow{\alpha};\vec{x}) = \int g(\overrightarrow{\alpha};\vec{x})\, d\vec{x}$$

Input data : $[x_1, x_2, x_3, \ldots, \alpha_0]$ ==(train the parameters θ)== Target data : $g = g(\overrightarrow{\alpha};\vec{x})$

After training

Get the integration values I

- **Training**: First, learn the parameters of the Variational Quantum Circuit (VQC) for the given objective function $g(\alpha; x)$.
  The goal is to make the circuit approximate the function g.
  In this process, we aim to minimize the Mean-Squared Error loss function, which finds the parameters of the circuit that provide the optimal prediction for each training data in g.

$$g_{j,\mathrm{est}}(\boldsymbol{\alpha};\boldsymbol{x}_j|\boldsymbol{\theta}) = \left.\frac{\partial G(\boldsymbol{\alpha}, x_1, \ldots, x_n|\boldsymbol{\theta})}{\partial x_1 \ldots \partial x_n}\right|_{\boldsymbol{x}_j}$$

$$J_{\mathrm{mse}} = \frac{1}{N_{\mathrm{train}}} \sum_{j=1}^{N_{\mathrm{train}}} \left[g_{j,\mathrm{meas}} - g_{j,\mathrm{est}}(\boldsymbol{\alpha};\boldsymbol{x}_j|\boldsymbol{\theta})\right]^2$$

Methodology - test (Integration)

Evaluating analytic gradients on quantum hardware

Maria Schuld,* Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran
Xanadu Inc., 372 Richmond St W, Toronto, M5V 1X6, Canada
(Dated: November 29, 2018)

$$\begin{aligned}\partial_\mu f &= r\big(\langle\psi| \mathcal{G}^\dagger(\mu+s)\hat{Q}\mathcal{G}(\mu+s)|\psi\rangle \\ &\quad - \langle\psi| \mathcal{G}^\dagger(\mu-s)\hat{Q}\mathcal{G}(\mu-s)|\psi\rangle \big) \\ &= r\left(f(\mu+s) - f(\mu-s)\right).\end{aligned}$$

a parameter $\mu \in \theta$

$$G(x;\alpha) = \int g(\alpha;x)dx\,.$$

$$g(\mu) = \partial_\mu G(\theta) = r(G(\mu^+) - G(\mu^-))$$

- **Integration**: Using the learned model, calculate the integral of g. Using Parameter Shift Rule (**PSR**) to calculate the derivative of a circuit, which allows us to calculate the indefinite integral of g ($G(\alpha;x)$).
  That is, evaluate G as a circuit, calculate the derivative of g as PSR, and obtain the integral value.

$$I(\alpha) = G(x_b;\alpha) - G(x_a;\alpha) \qquad g_{i,est}(\alpha;x_j|\theta) = \left.\frac{\partial G(\alpha, x_1, \ldots, x_n|\theta)}{\partial x_1 \ldots \partial x_n}\right|_{x_j}$$

How to derive?

$$I_{ab}(\ldots, x_{k-1}, x_{k+1}, \ldots) = \int_{x_{k,a}}^{x_{k,b}} g(x)dx_k = G(x_{k,b};\alpha) - G(x_{k,a};\alpha) \approx g_{est}(x_{k,b}|\theta_{best}) - g_{est}(x_{k,a}|\theta_{best})\,.$$

Methodology - test (Integration) ex: $g(x_1, x_2, x_3, \alpha_0)$

To perform integration over variables x_2 and x_3 and express the result as a function of x_1, we compute the double integral of these variables.

1. Setting up the double integral:

$$I(x_1) = \int_{x_{min}}^{x_{max}} \int_{x_{min}}^{x_{max}} g_{est}(x_1, x_2, x_3, \alpha_0)\, dx_2\, dx_3 \text{ with } \alpha_0 = \# \text{ fixed.}$$

2. Dividing the interval:

For both variables x_2 and x_3, divide the interval $[x_{min}, x_{max}]$ into N small intervals. The width of the interval for each variable is $\Delta x = (x_{max} - x_{min})/N$, which is the same.

3. Calculating the function's value:

To calculate the Riemann sum using the double sum, compute the value of the function at representative points x_{2i} and x_{3j} for each interval.

Methodology - test (Integration)

4. Calculating the double Riemann sum:
Multiply the function's value at each representative point and then sum over all intervals for x_2 and x_3 to calculate the double Riemann sum.

$$I(x_1) = \sum_{i=1}^{N} \sum_{j=1}^{N} g_{est}(x_1, x_{2i}, x_{3j}, \alpha_0)\Delta x^2$$

5. Calculating the integral value and Interpreting the results:
After calculating the Riemann sum, take N sufficiently large so that the approximate integral value $I(x_1)$approximates the actual double integral value.

By calculating $I(x_1)$ for x_1, we can obtain the double integral result as a function of x_1.

Target toy example $g(\vec{x}) = \cos(\vec{\alpha} \cdot \vec{x} + \alpha_0)$

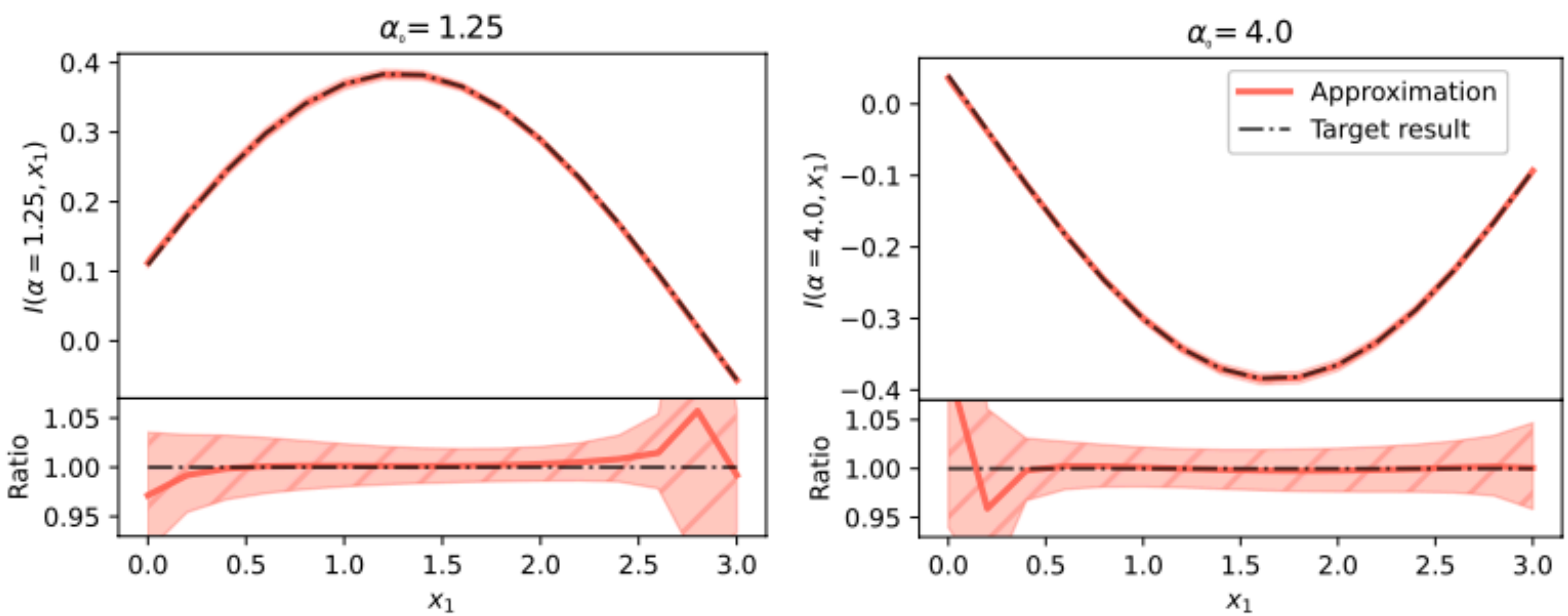
$$I(\vec{\alpha};\vec{x}) = \int g(\vec{\alpha};\vec{x})\, d\vec{x} = \int \cos(x_1 + 2x_2 + 0.5x_3 + \alpha_0) dx_1 dx_2 dx_3$$

Input data : $[x_1, x_2, x_3, \alpha_0]$

Target data : $g = \cos(x_1 + 2x_2 + 0.5x_3 + \alpha_0)$

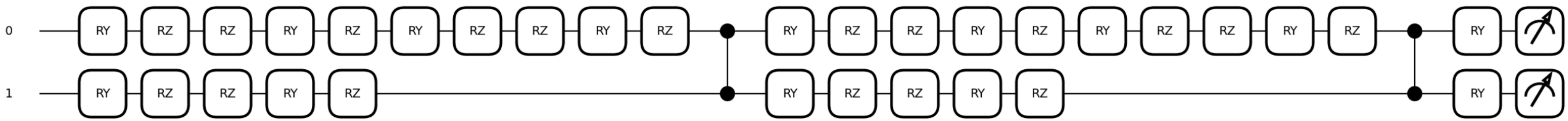
Parameter	Value
$N_{\boldsymbol{x},\mathrm{train}}$	100
$\boldsymbol{\alpha}$	$\{1, 2, 0.5\}$
N_{α_0}	10
N_{layers}	2
N_{params}	20
$\lvert I - \tilde{I} \rvert$	$4.4 \cdot 10^{-3}$
N_{shots}	Exact simulation
Optimizer	L-BFGS

$$I(\alpha_0; x_1)_{target} = [-\cos(\alpha_0 + x_1) + \cos(\alpha_0 + x_1 + 1.5) + \cos(\alpha_0 + x_1 + 6) - \cos(\alpha_0 + x_1 + 7.5)]_{x_1=0}^{x_1=3}$$



Target toy example $g(\vec{x}) = \cos(\vec{\alpha} \cdot \vec{x} + \alpha_0)$

Pennylane reproduce results - train results



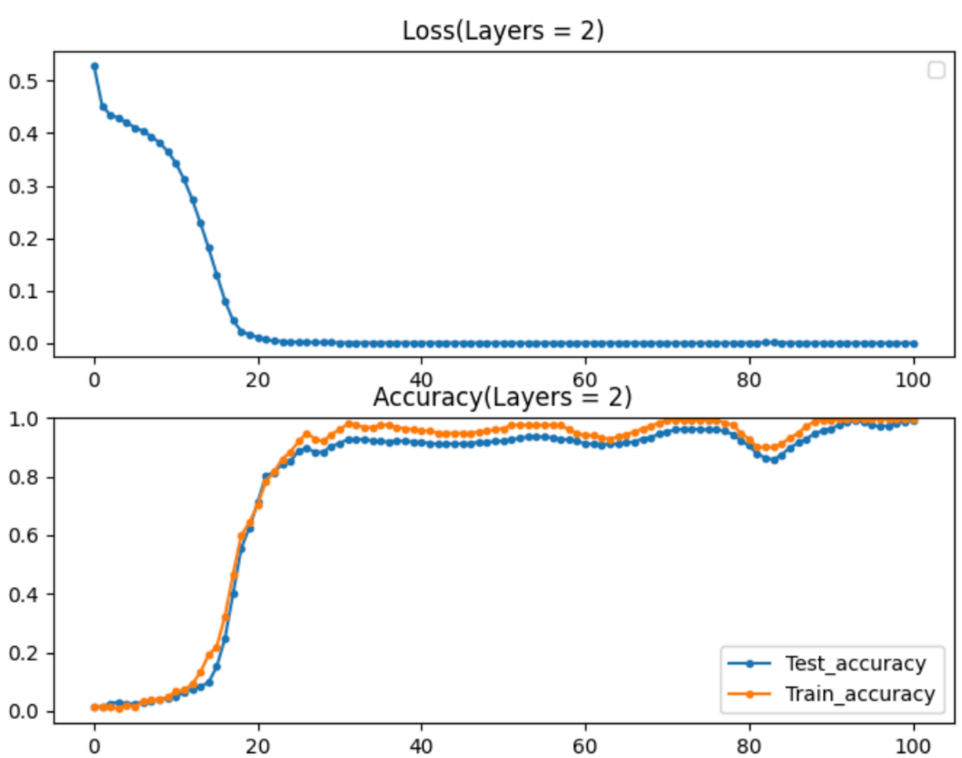

```
tensor([[[0.         , 0.         , 1.25       ],
         [0.03030303, 0.         , 1.25       ],
         [0.06060606, 0.         , 1.25       ],
         [0.09090909, 0.         , 1.25       ],
         ⋮
         [2.93939394, 0.         , 1.25       ],
         [2.96969697, 0.         , 1.25       ],
         [3.         , 0.         , 1.25       ]],
```

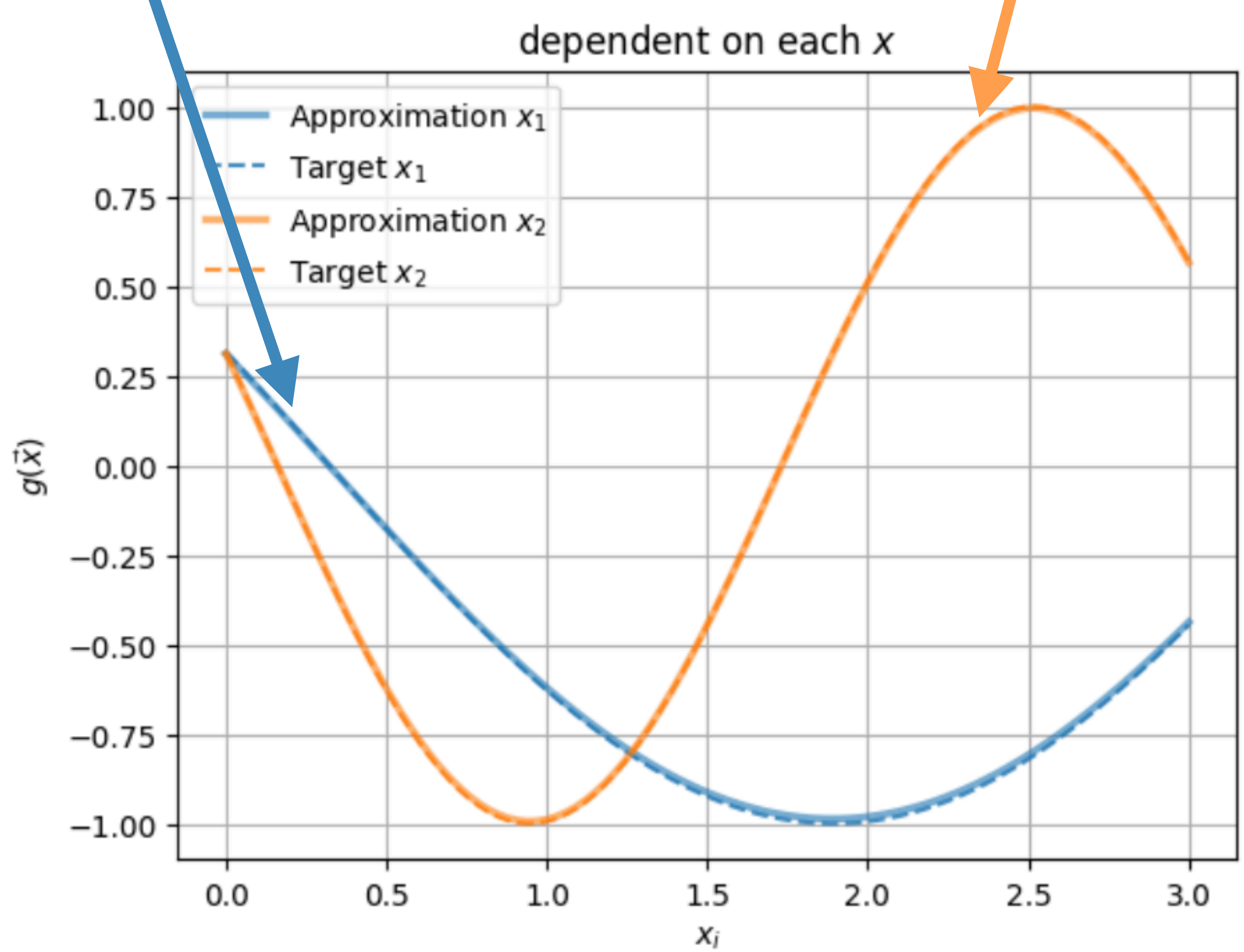
```
tensor([[0.         , 0.         , 1.25       ],
        [0.         , 0.03030303, 1.25       ],
        [0.         , 0.06060606, 1.25       ],
        [0.         , 0.09090909, 1.25       ],
        ⋮
        [0.         , 2.93939394, 1.25       ],
        [0.         , 2.96969697, 1.25       ],
        [0.         , 3.         , 1.25       ]],
```

Input data : $[x_1, x_2, \alpha_0]$

Target data : $g = \cos(x_1 + 2x_2 + \alpha_0)$

alphas = [1, 2]
xdim = len(alphas)
xmin = [0]*xdim
xmax=[3.5]*xdim





Target toy example $g(\vec{x}) = \cos(\vec{\alpha} \cdot \vec{x} + \alpha_0)$

Pennylane reproduce results - test results $x_{\min} = 0.0,\ x_{\max} = 3.0,\ \alpha_0 = 1.25$ (or 4.0)

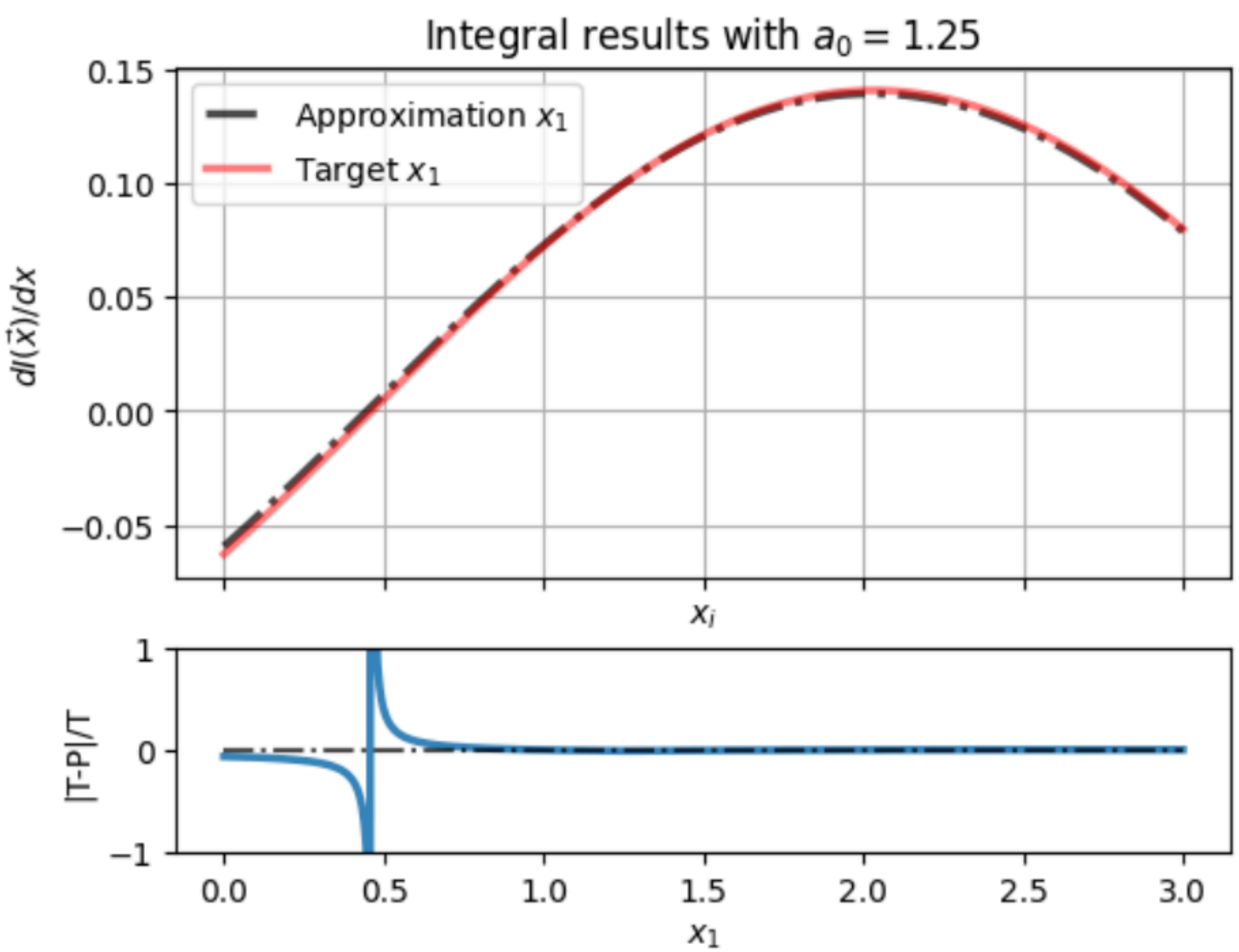
Input data : $[x_1, x_2, \alpha_0]$

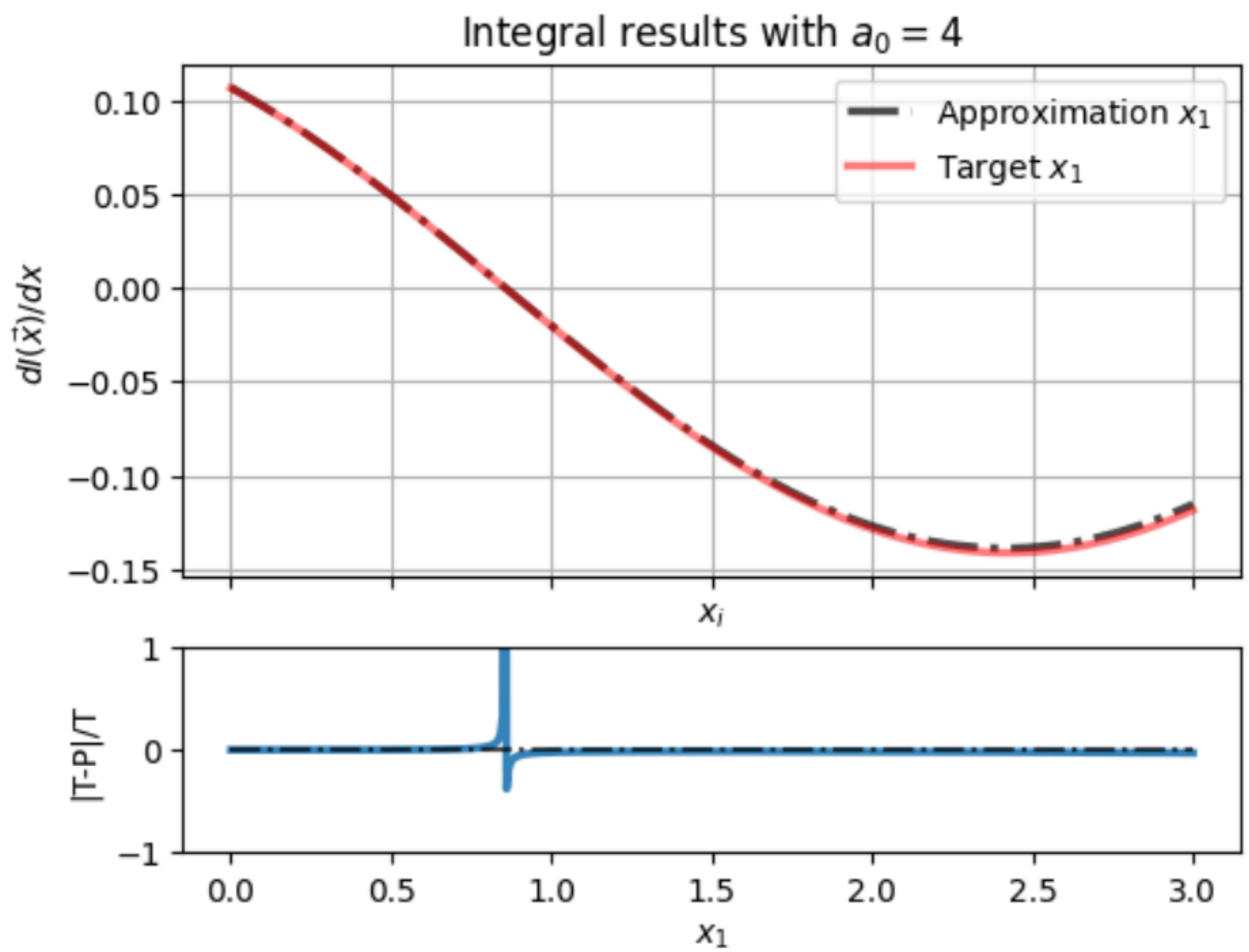
Target data : $g = \cos(x_1 + 2x_2 + \alpha_0)$

$$I(x_1) = \int dx_1 \int_{x_{min}}^{x_{max}} dx_2\ g(x_1, x_2, \alpha_0 = \#)$$

In Fig. 3 we have plotted the differential distribution

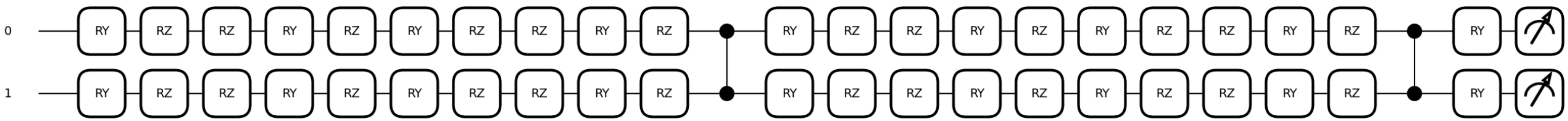
$$\frac{\mathrm{d}I(\boldsymbol{\alpha}; x_1)}{\mathrm{d}x_1}, \tag{19}$$





Target toy example $g(\vec{x}) = \cos(\vec{\alpha} \cdot \vec{x} + \alpha_0)$

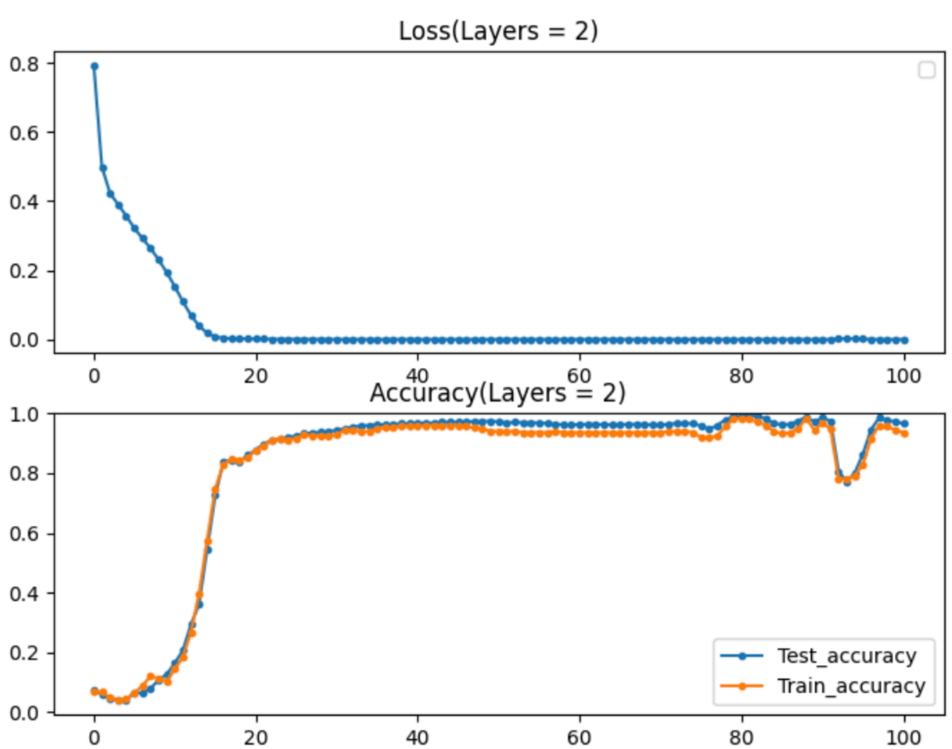
Pennylane reproduce results - train results

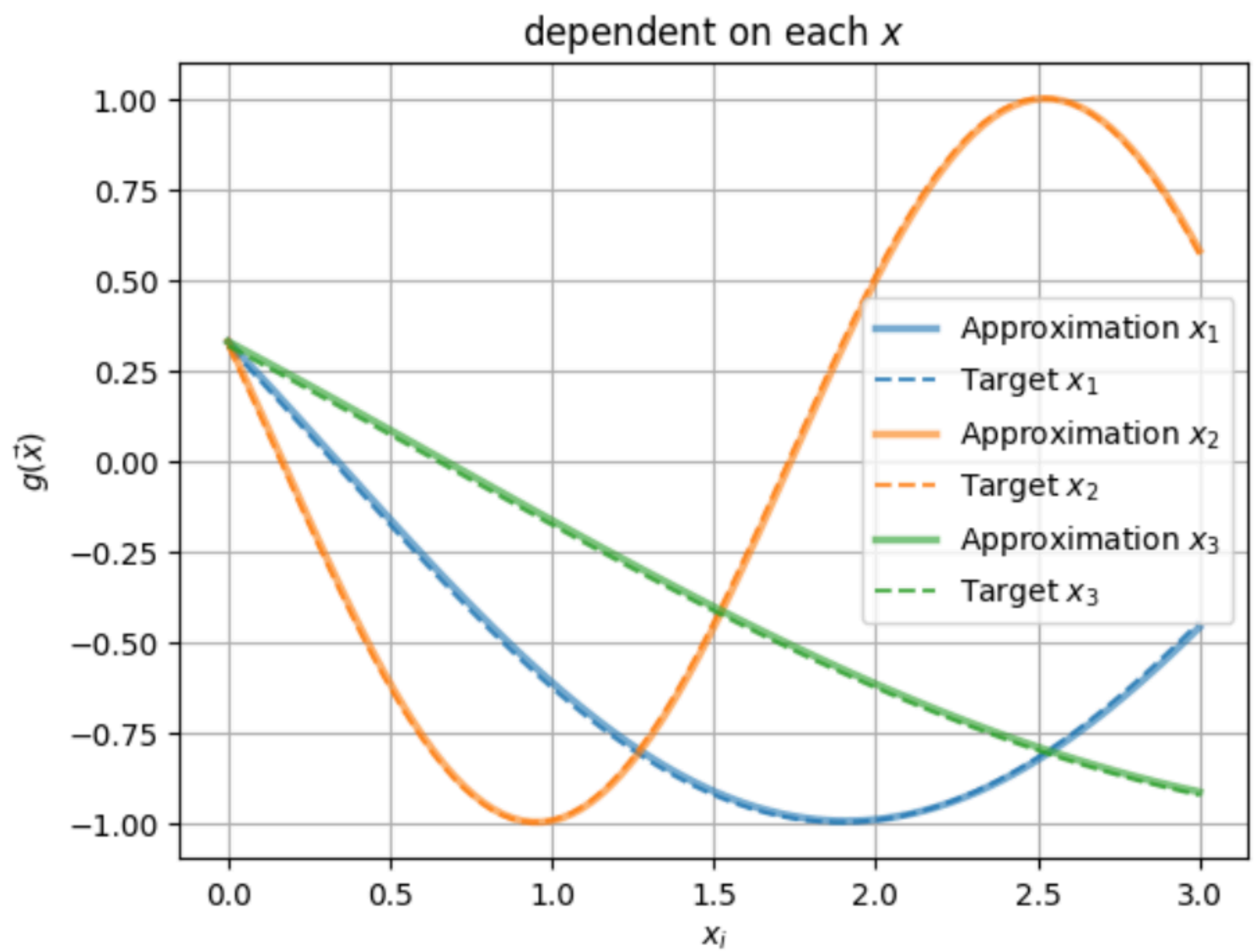


Input data : $[x_1, x_2, x_3, \alpha_0]$

Target data : $g = \cos(x_1 + 2x_2 + 0.5x_3 + \alpha_0)$

alphas = [1, 2, 0.5]
xdim = len(alphas)
xmin = [0]*xdim
xmax=[3.5]*xdim





Target toy example $g(\vec{x}) = \cos(\overrightarrow{\alpha} \cdot \vec{x} + \alpha_0)$

Pennylane reproduce results - test results $x_{\min} = 0.0,\ x_{\max} = 3.0,\ \alpha_0 = 1.25$ (or 4.0)

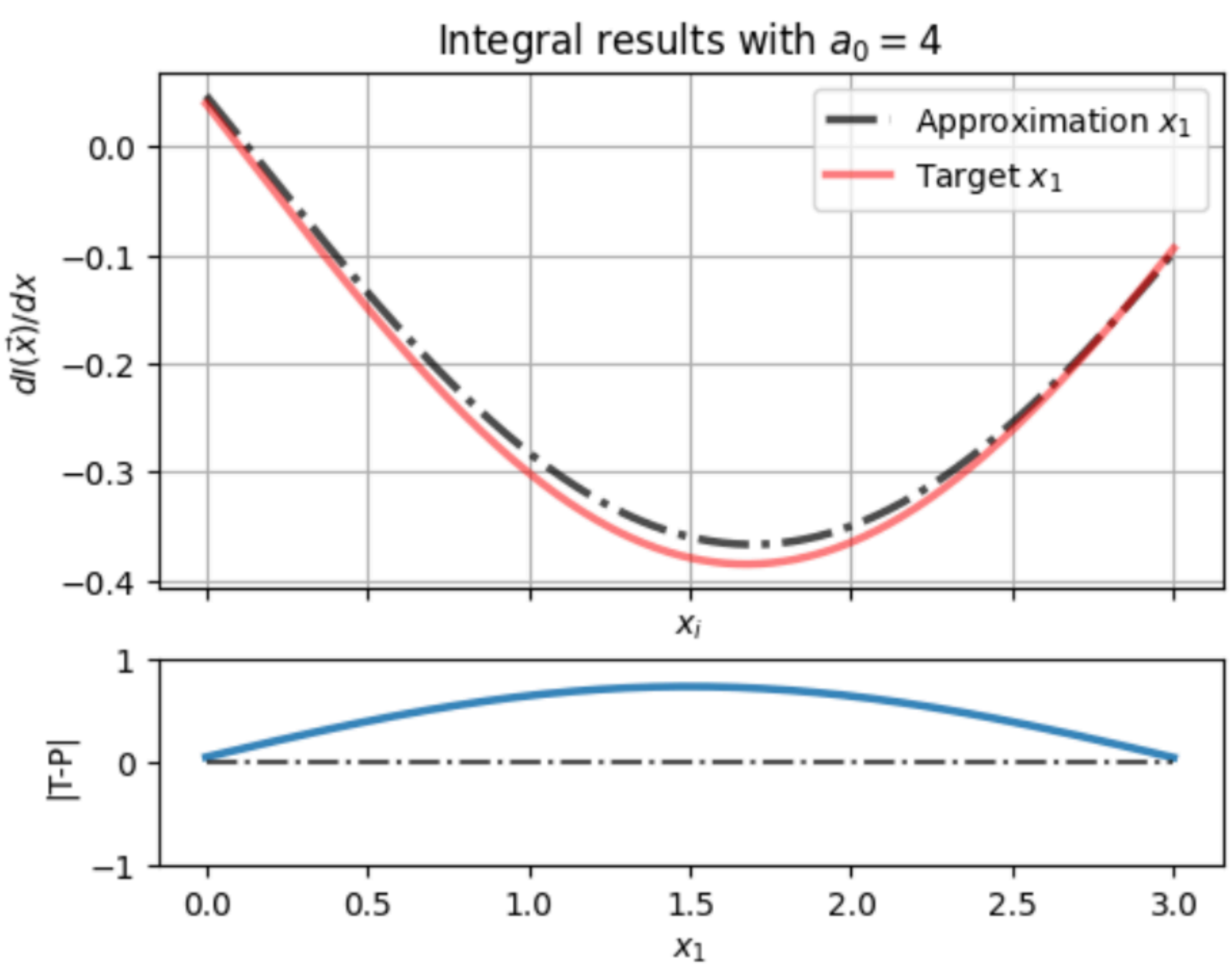
$$I(x_1) = \int dx_1 \int_{x_{min}}^{x_{max}} dx_2 \int_{x_{min}}^{x_{max}} dx_3\ g(x_1, x_2, x_3, \alpha_0 = \#)$$

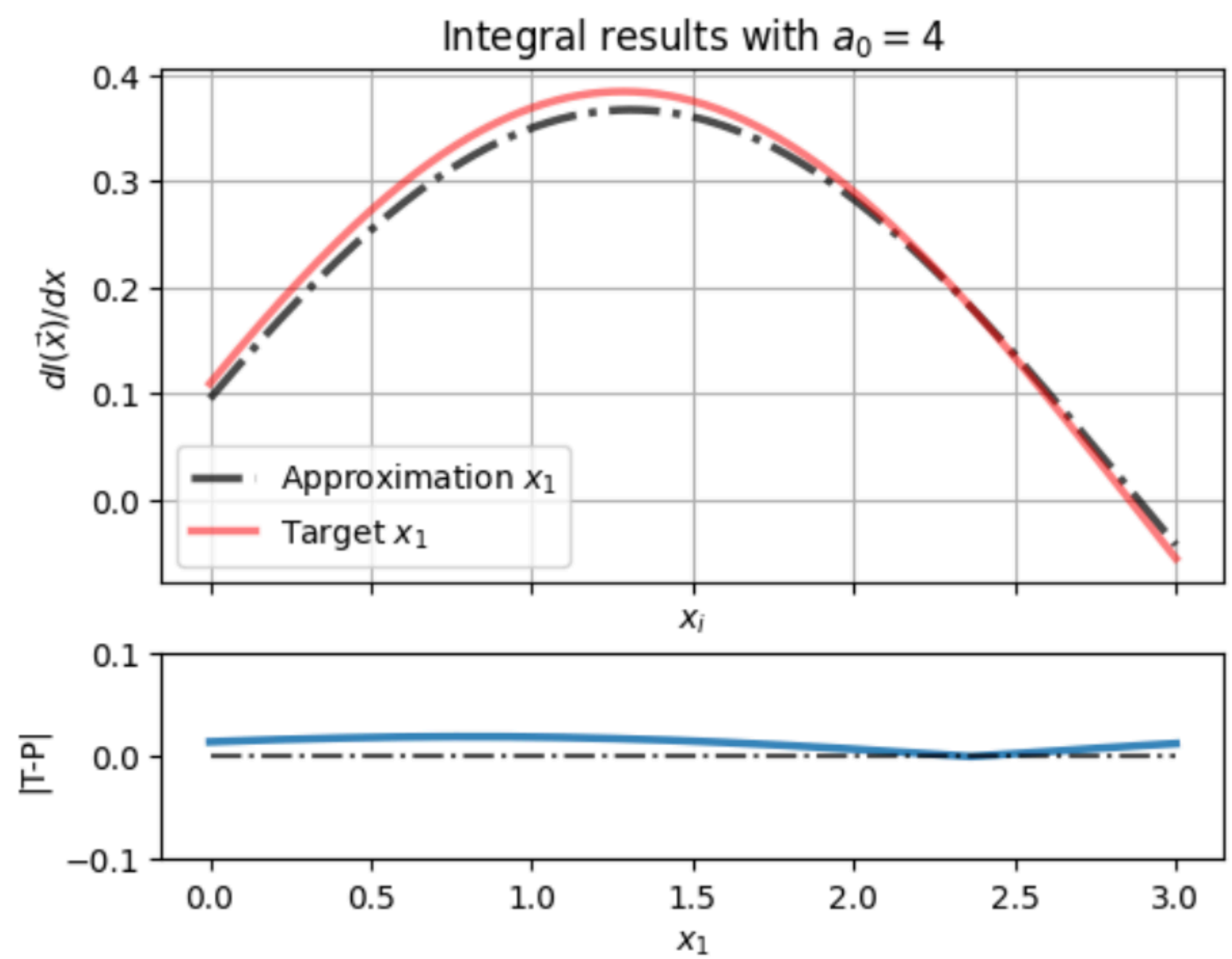
Input data : $[x_1, x_2, x_3, \alpha_0]$

Target data : $g = \cos(x_1 + 2x_2 + 0.5x_3 + \alpha_0)$

In Fig. 3 we have plotted the differential distribution

$$\frac{\mathrm{d}I(\boldsymbol{\alpha}; x_1)}{\mathrm{d}x_1}, \tag{19}$$





Weekly goal

[Week 1] Review the reference [1] thoroughly and understand basic idea.

[Week 2~3] Implement the idea using Qiskit and/or **PennyLane**, and reproduce their results for two examples in the paper.

[Week 4] Then consider a more complex and more realistic example, taking electron-positron Collider ($e^+e^- \to \mu^+\mu^-$). Effectively, this problem involves four-dimensional integration.

[Week 5] Try to optimize the circuits with different choice of the cost function or variations of quantum circuits.

[Week ?] Study Monte-Carlo sampling with above integration method, and how to implement the importance sampling into a variational quantum circuit. (Check Ref. [2])

[15]:

```python
def accuracy(ypred, ydata):

    ratio = []
    score = 0
    for i in range(len(ydata)):
        acc = ypred[i] / ydata[i] - 1
        if np.abs(acc) < 0.2:
            score += 1
        ratio.append(ypred[i] / ydata[i])

    return ratio, score/len(ydata)
```