Optical computing or **photonic computing** uses photons produced by lasers or diodes for computation. For decades, photons have shown promise to enable a higher bandwidth than the electrons used in conventional computers (see optical fibers).

Most research projects focus on replacing current computer components with optical equivalents, resulting in an optical digital computer system processing binary data. This approach appears to offer the best short-term prospects for commercial optical computing, since optical components could be integrated into traditional computers to produce an optical-electronic hybrid. However, optoelectronic devices consume 30% of their energy converting electronic energy into photons and back; this conversion also slows the transmission of messages. All-optical computers eliminate the need for optical-electrical-optical (OEO) conversions, thus reducing electrical power consumption.[1]

Application-specific devices, such as synthetic-aperture radar (SAR) and optical correlators, have been designed to use the principles of optical computing. Correlators can be used, for example, to detect and track objects,[2] and to classify serial time-domain optical data.[3]

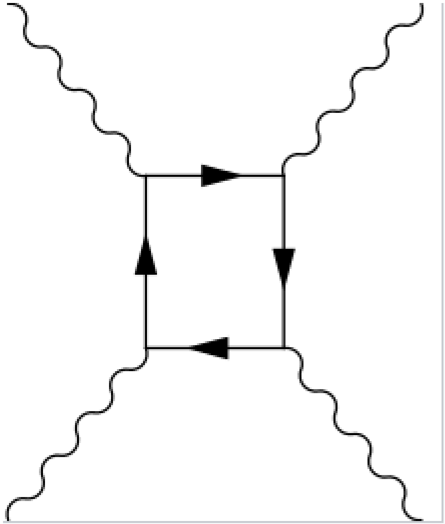
Continuous Variable Quantum Computing: CV-QC

Soronzonbold Otgonbaatar

DLR Oberpfaffenhofen

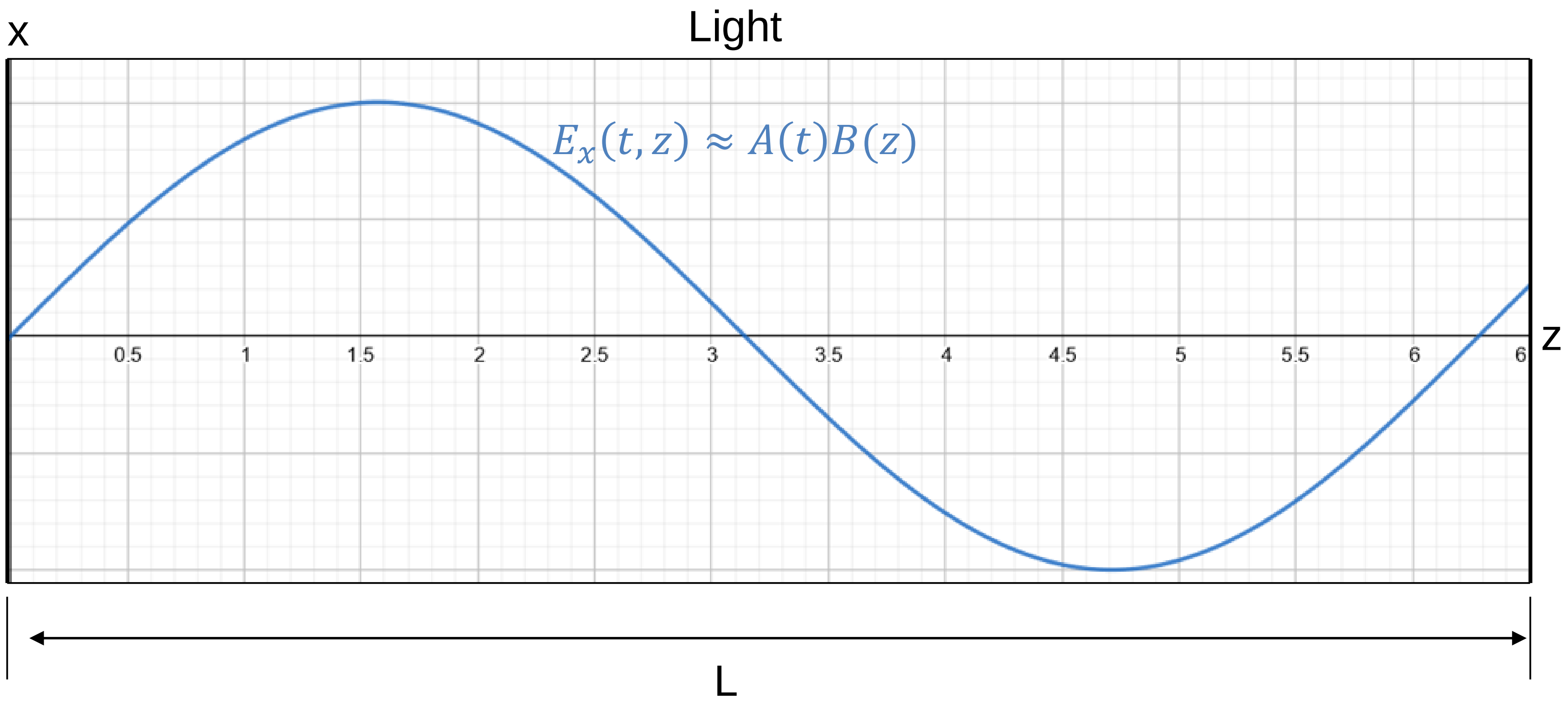
DLR

Star Wars: CV-QC



A Feynman diagram (*box diagram*) for photon–photon scattering, one photon scatters from the transient vacuum charge fluctuations of the other

CV-QC

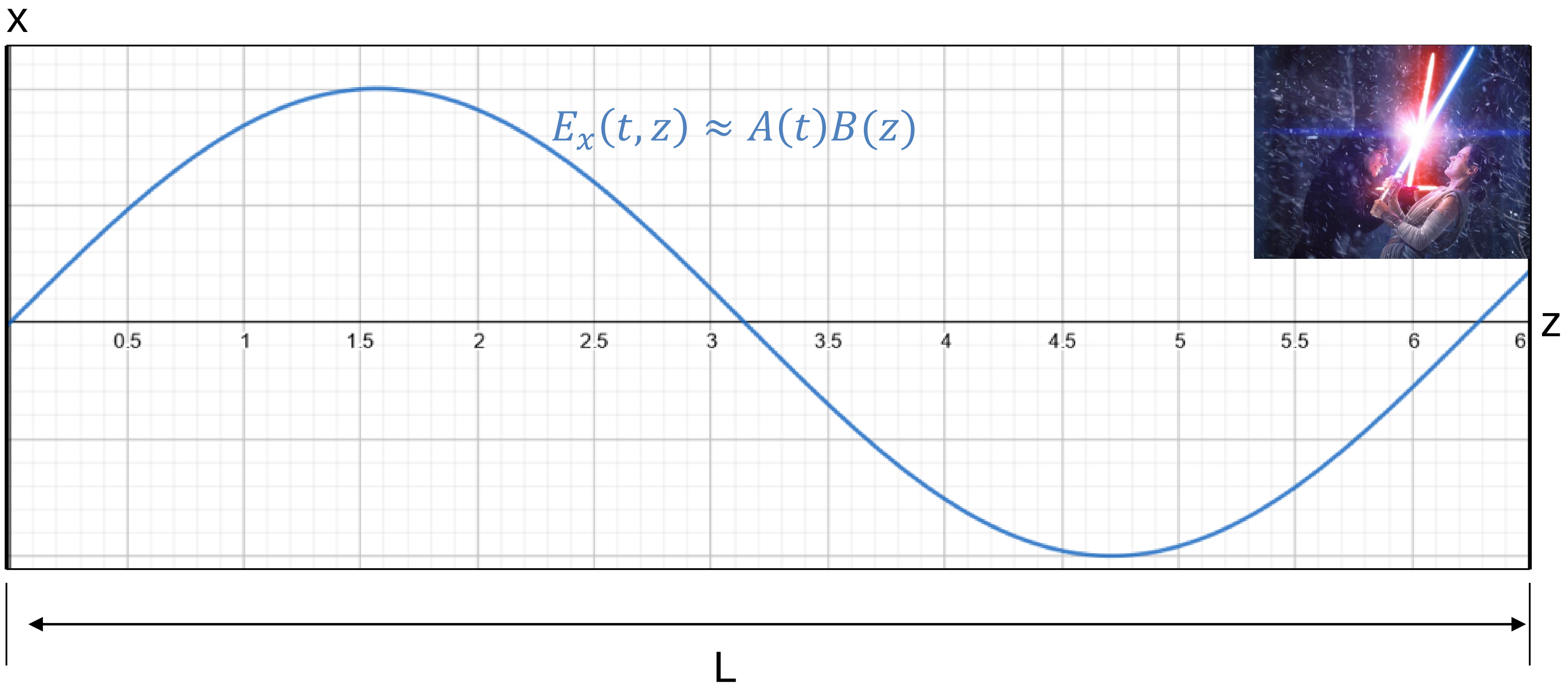


The field E is the emergent physical property of **many photons** and **creations**.

God said: let there be Maxwell, then there is light.

CV-QC: Star Wars

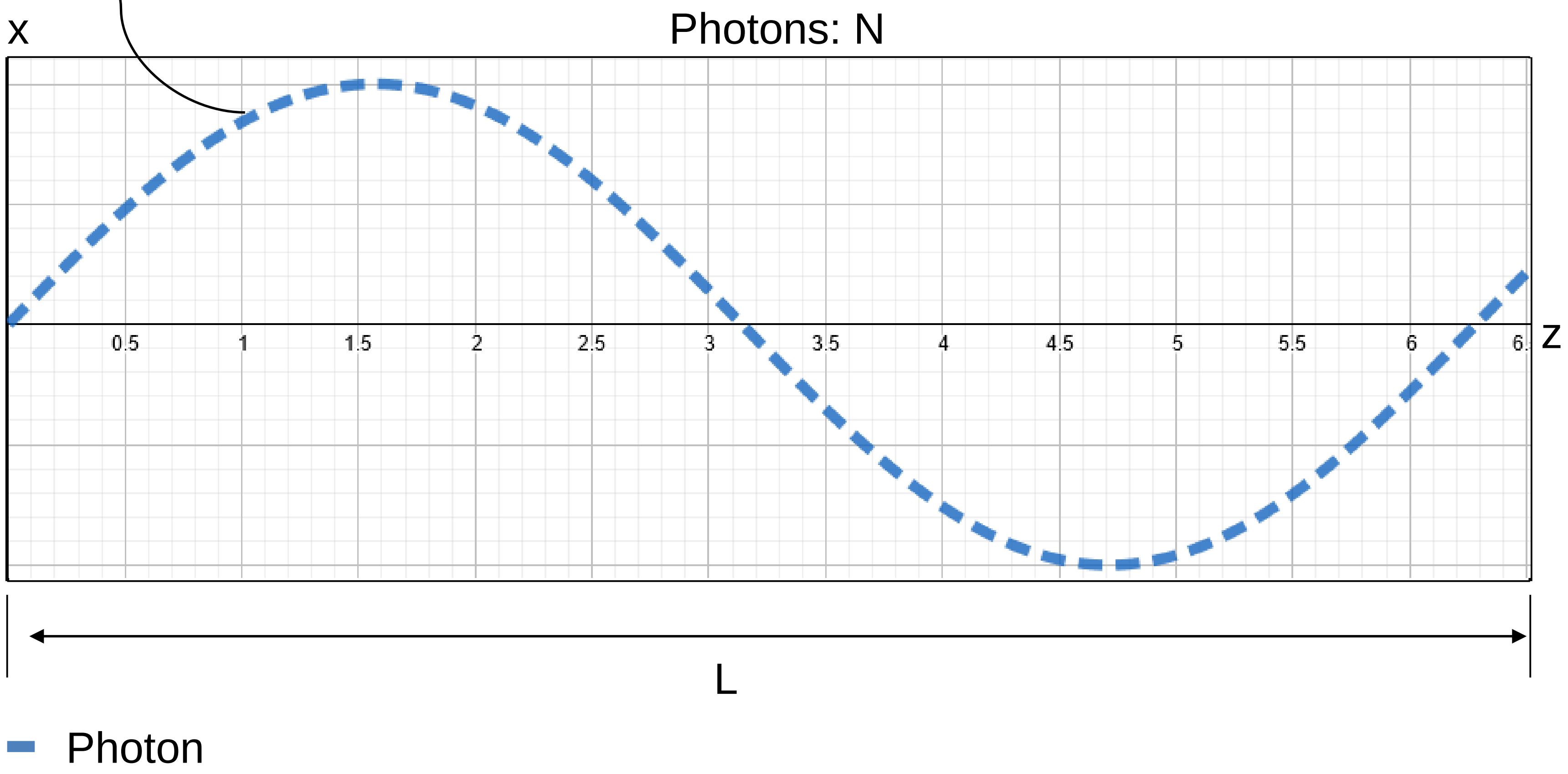
CV-QC



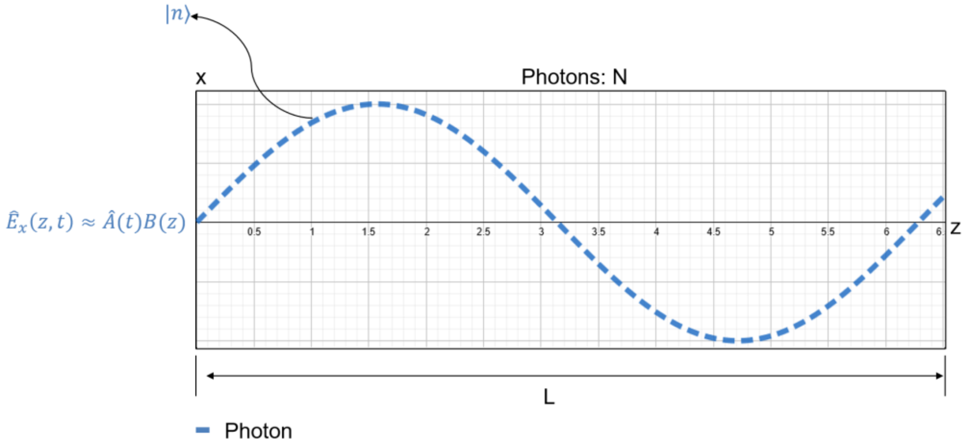
CV-QC

$|n\rangle$

$\hat{E}_x(z,t) \approx \hat{A}(t)B(z)$



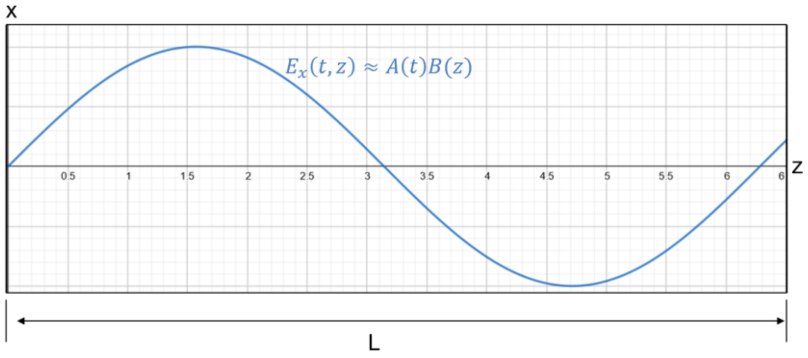
CV-QC



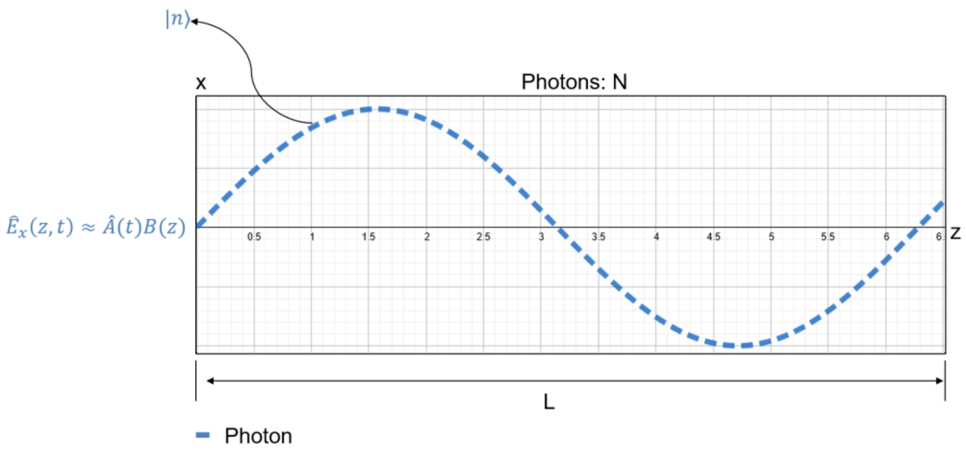
$$\langle n|\hat{E}|n\rangle \neq E_x(t,z)$$

$$\langle ?\,|\hat{E}|\,?\,\rangle$$

What is $|?\,\rangle$

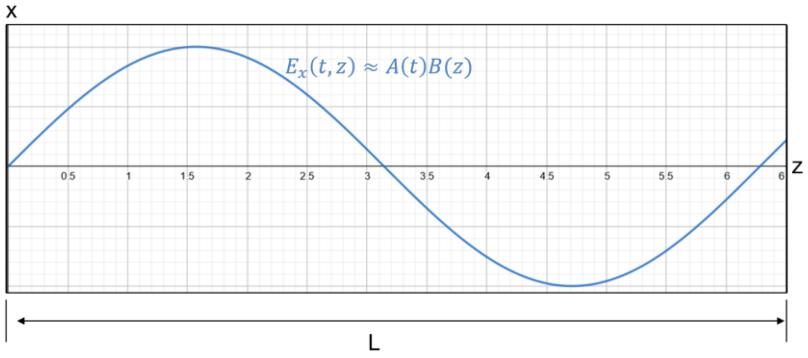


CV-QC



$$\langle\alpha|\hat{E}|\alpha\rangle = E_x(t,z)$$

Coherent state: $|\alpha\rangle \approx \sum_i f(n_i)$



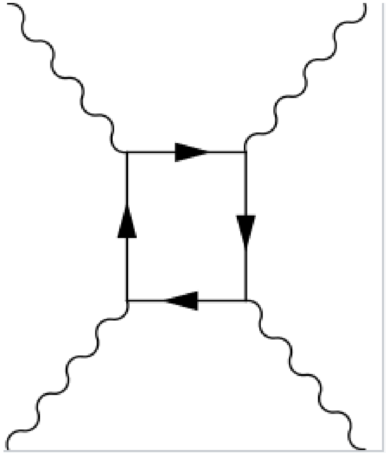
Photonic quantum computing

Classic photonic computing was here (for decades) even before photonic quantum computing. Photons are charge-less and spin 0. **It does not interact with environment**. **Photons even donot interact with each other**, and only via nonlinear optical medium.

Recently (around 2001), linear optical medium is discovered. Hence, they become promising candidates for building **room temperature** CV-QCs. **No-self interaction**, and so you do not see many papers for computing optimization problems.

Photonic quantum computing

~1980



A Feynman diagram (*box diagram*) for photon–photon scattering, one photon scatters from the transient vacuum charge fluctuations of the other

Today-Future

Shut up and Calculate - „David Mermin“

Strawberry and Pennylane

DLR

Strawberry and Pennylane

Gates As PQC and Operations

Rotation(phi, wires)	Phase space rotation
Squeezing(r, phi, wires)	Phase space squeezing.
Displacement(a, phi, wires)	Phase space displacement.
Beamsplitter(theta, phi, wires)	Beamsplitter interaction.
TwoModeSqueezing(r, phi, wires)	Phase space two-mode squeezing.
QuadraticPhase(s, wires)	Quadratic phase shift.
ControlledAddition(s, wires)	Controlled addition operation.
ControlledPhase(s, wires)	Controlled phase operation.
Kerr(kappa, wires)	Kerr interaction.
CrossKerr(kappa, wires)	Cross-Kerr interaction.
CubicPhase(gamma, wires)	Cubic phase shift.
Interferometer(U, wires)	A linear interferometer transforming the bosonic operators according to the unitary matrix U.

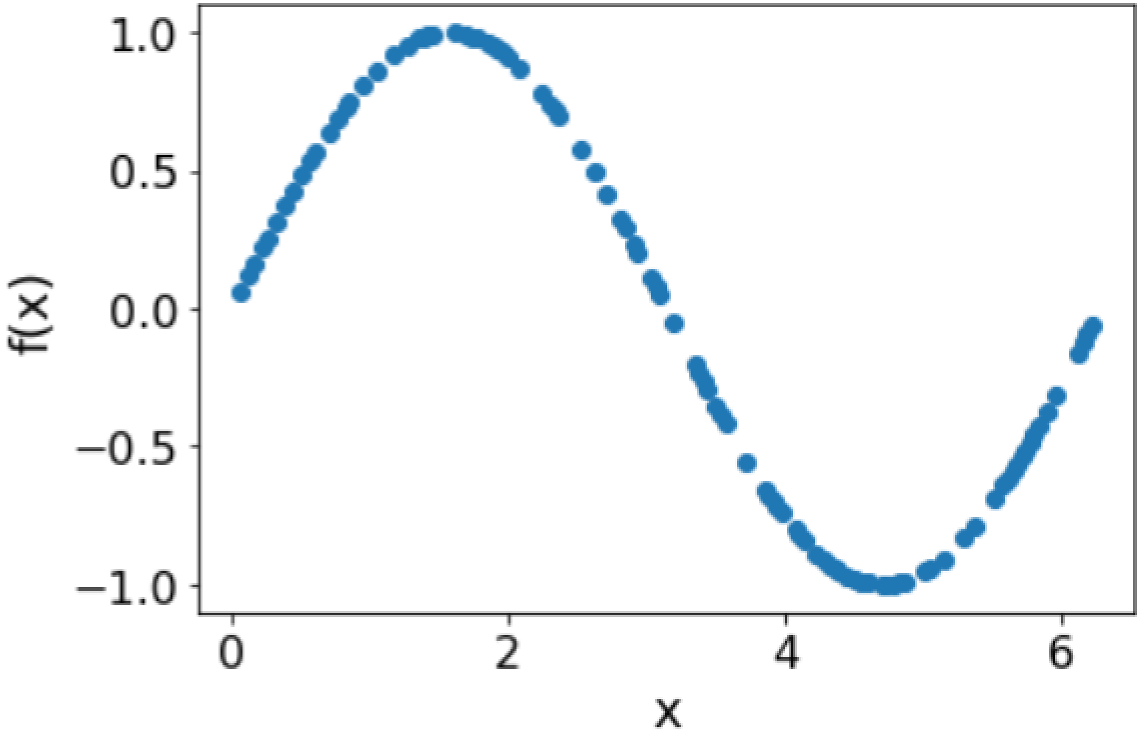
State preparation As encoding classical data

CoherentState(a, phi, wires)	Prepares a coherent state.
SqueezedState(r, phi, wires)	Prepares a squeezed vacuum state.
DisplacedSqueezedState(a, phi_a, r, phi_r, wires)	Prepares a displaced squeezed vacuum state.
ThermalState(nbar, wires)	Prepares a thermal state.
GaussianState(r, V, wires)	Prepare subsystems in a given Gaussian state.
FockState(n, wires)	Prepares a single Fock state.
FockStateVector(state, wires)	Prepare subsystems using the given ket vector in the Fock basis.
FockDensityMatrix(state, wires)	Prepare subsystems using the given density matrix in the Fock basis.
CatState(a, phi, p, wires)	Prepares a cat state.

Observables As outputs of class label or functions

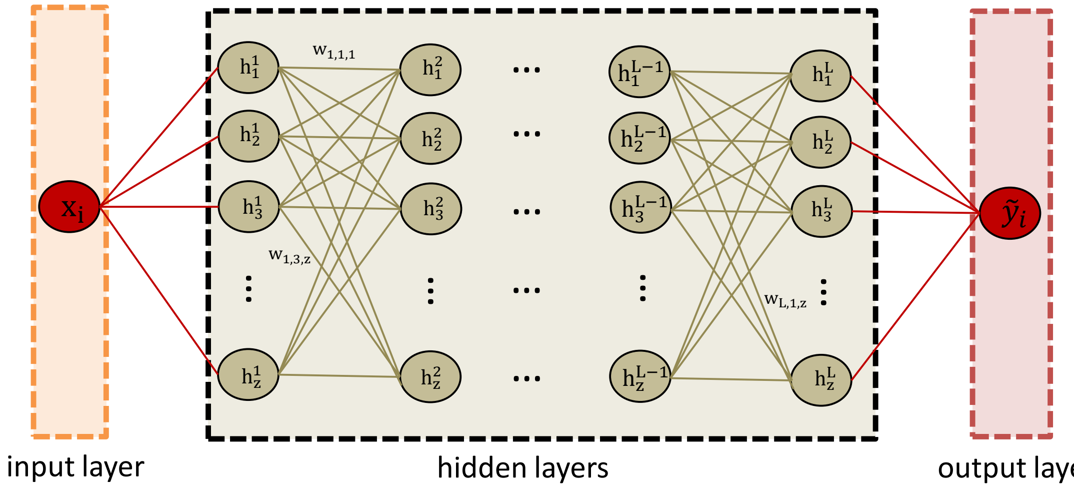
NumberOperator(wires)	The photon number observable $\langle \hat{n} \rangle$.
X(wires)	The position quadrature observable $\hat{x}$.
P(wires)	The momentum quadrature observable $\hat{p}$.
QuadOperator(phi, wires)	The generalized quadrature observable $\hat{x}_\phi = \hat{x} cos\phi + \hat{p} \sin\phi$.
PolyXP(q, wires)	An arbitrary second-order polynomial observable.
FockStateProjector(n, wires)	The number state observable $\lvert n \rangle \langle n \rvert$.

Regression on CNN: f(x)=sin(x)



Regression on CNN: sin(x) with sigmoid

L=1, z=10

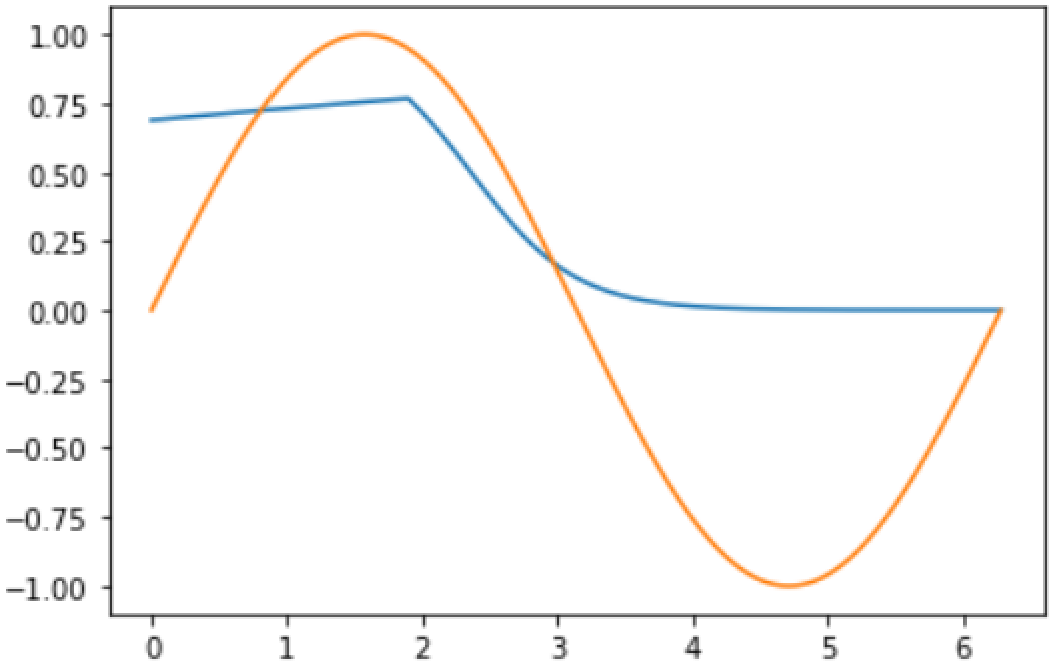


$$\mathrm{L} = \frac{1}{N} \sum_i (y_i - \tilde{y}_i)^2$$

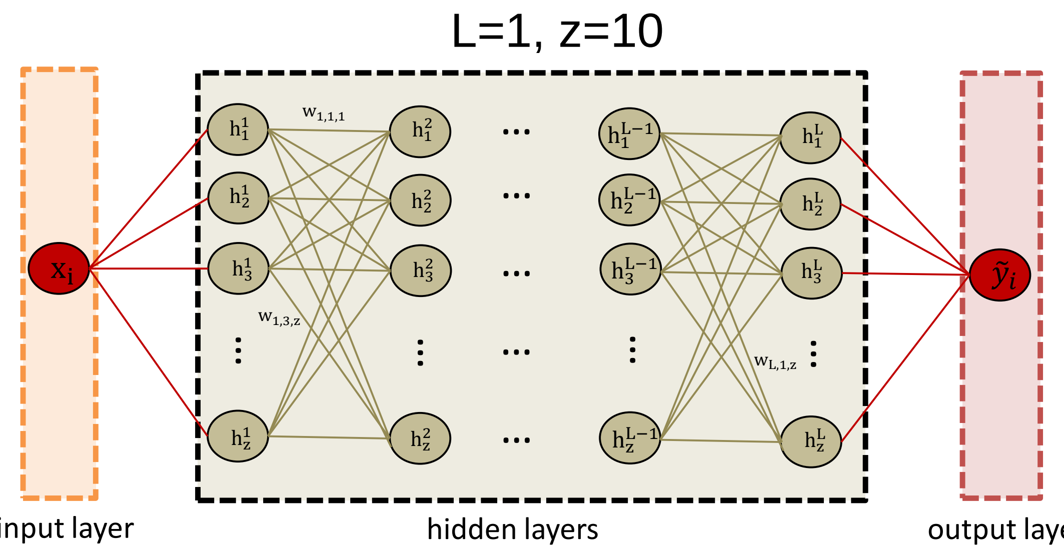
$$y_i = \sin(x_i)$$

Regression on CNN: Nonlinear functions at the output neuron

$sigmoid(y_i)$



Regression on CNN: sin(x) with tanh

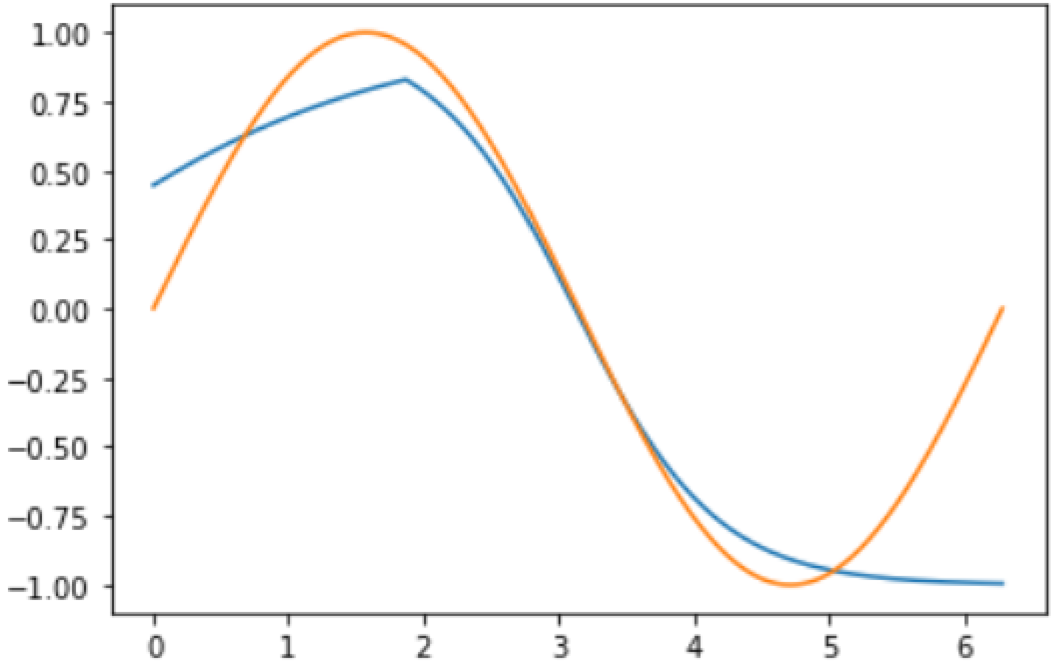


$$\mathrm{L} = \frac{1}{N} \sum_i (y_i - \tilde{y}_i)^2$$

$$y_i = \sin(x_i)$$

Regression on CNN: Nonlinear functions at the output neuron

$tanh(y_i)$



Regression on CV-QC: sin(x)

class **`Rotation`(*phi, wires*)** [source]

Phase space rotation

$$R(\phi) = \exp\left(i\phi\hat{a}^\dagger\hat{a}\right) = \exp\left(i\frac{\phi}{2}\left(\frac{\hat{x}^2+\hat{p}^2}{\hbar} - \hat{\mathbb{1}}\right)\right).$$

Details:

- Number of wires: 1
- Number of parameters: 1
- Gradient recipe: $\frac{d}{dr}f(R(r)) = \frac{1}{2}[f(R(\phi+\pi/2)) - f(R(\phi-\pi/2))]$ where f is an expectation value depending on $R(r)$.
- Heisenberg representation:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

Parameters: **phi** (*float*) – the rotation angle

class **`Squeezing`(*r, phi, wires*)** [source]

Phase space squeezing.

$$S(z) = \exp\left(\frac{1}{2}(z^*\hat{a}^2 - z\hat{a}^{\dagger 2})\right).$$

where $z = re^{i\phi}$.

Details:

- Number of wires: 1
- Number of parameters: 2
- Gradient recipe: $\frac{d}{dr}f(S(r,\phi)) = \frac{1}{2\sinh s}[f(S(r+s,\phi)) - f(S(r-s,\phi))]$, where s is an arbitrary real number (0.1 by default) and f is an expectation value depending on $S(r,\phi)$.
- Heisenberg representation:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cosh r - \cos\phi\sinh r & -\sin\phi\sinh r \\ 0 & -\sin\phi\sinh r & \cosh r + \cos\phi\sinh r \end{bmatrix}$$

Parameters:
- **r** (*float*) – squeezing amount
- **phi** (*float*) – squeezing phase angle ϕ
- **wires** (*Sequence[int] or int*) – the wire the operation acts on

class **`Displacement`(*a, phi, wires*)** [source]

Phase space displacement.

$$D(a,\phi) = D(\alpha) = \exp(\alpha\hat{a}^\dagger - \alpha^*\hat{a}) = \exp\left(-i\sqrt{\frac{2}{\hbar}}(\mathrm{Re}(\alpha)\hat{p} - \mathrm{Im}(\alpha)\hat{x})\right).$$

where $\alpha = ae^{i\phi}$ has magnitude $a \geq 0$ and phase ϕ. The result of applying a displacement to the vacuum is a coherent state $D(\alpha)\left|0\right\rangle = \left|\alpha\right\rangle$.

Details:

- Number of wires: 1
- Number of parameters: 2
- Gradient recipe: $\frac{d}{da}f(D(a,\phi)) = \frac{1}{2s}[f(D(a+s,\phi)) - f(D(a-s,\phi))]$, where s is an arbitrary real number (0.1 by default) and f is an expectation value depending on $D(a,\phi)$.
- Heisenberg representation:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 2a\cos\phi & 1 & 0 \\ 2a\sin\phi & 0 & 1 \end{bmatrix}$$

Parameters:
- **a** (*float*) – displacement magnitude $a = |\alpha|$
- **phi** (*float*) – phase angle ϕ
- **wires** (*Sequence[int] or int*) – the wire the operation acts on

class **`Kerr`(*kappa, wires*)** [source]

Kerr interaction.

$$K(\kappa) = e^{i\kappa\hat{n}^2}.$$

Details:

- Number of wires: 1
- Number of parameters: 1
- Gradient recipe: None (uses finite difference)

Parameters:
- **kappa** (*float*) – parameter
- **wires** (*Sequence[int] or int*) – the wire the operation acts on

class **`X`(*wires*)** [source]

The position quadrature observable $\hat{x}$.

When used with the `expval()` function, the position expectation value $\langle\hat{n}\rangle$ is returned. This corresponds to the mean displacement in the phase space along the x axis.

Details:

- Number of wires: 1
- Number of parameters: 0
- Observable order: 1st order in the quadrature operators
- Heisenberg representation:

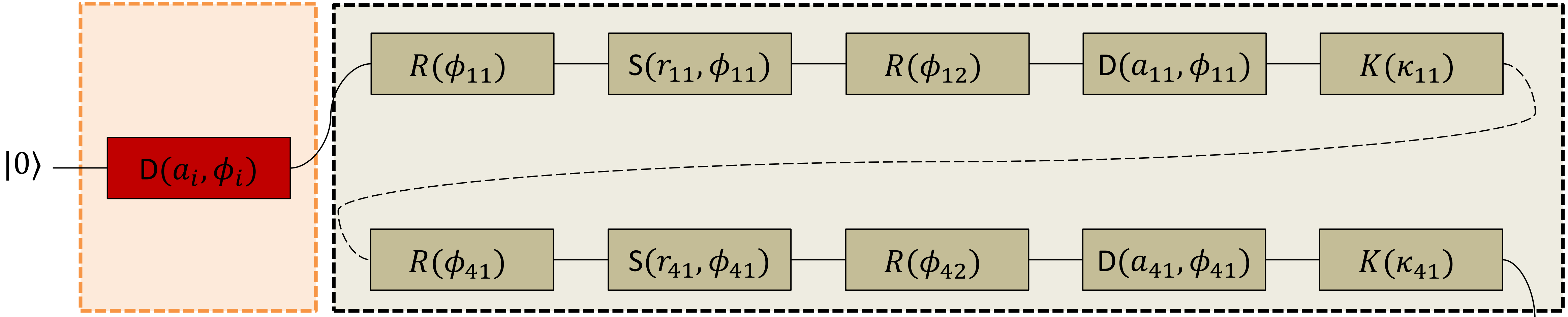
$$d = [0, 1, 0]$$

Parameters: **wires** (*Sequence[int] or int*) – the wire the operation acts on

DLR

Regression on CV-QC: sin(x)

A number of hidden layers: L=4

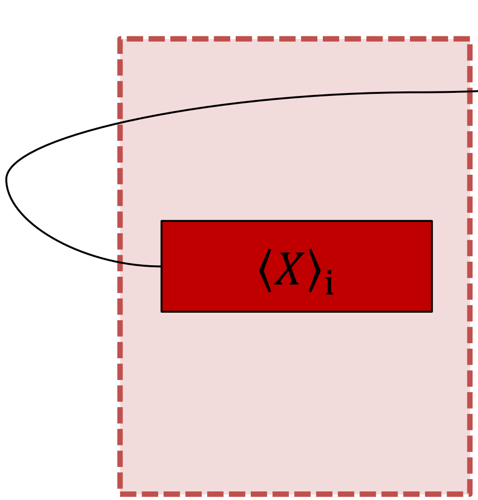


$$R(\phi) = \exp\left(i\phi \hat{a}^\dagger \hat{a}\right) = \exp\left(i\frac{\phi}{2}\left(\frac{\hat{x}^2+\hat{p}^2}{\hbar} - \hat{1}\right)\right)$$

$$S(z) = \exp\left(\frac{1}{2}(z^* \hat{a}^2 - z\hat{a}^{\dagger 2})\right) \qquad \alpha = a e^{i\phi}$$

$$D(a,\phi) = D(\alpha) = \exp(\alpha\hat{a}^\dagger - \alpha^*\hat{a}) = \exp\left(-i\sqrt{\frac{2}{\hbar}}(\mathrm{Re}(\alpha)\hat{p} - \mathrm{Im}(\alpha)\hat{x})\right)$$

$$K(\kappa) = e^{i\kappa \hat{n}^2}$$

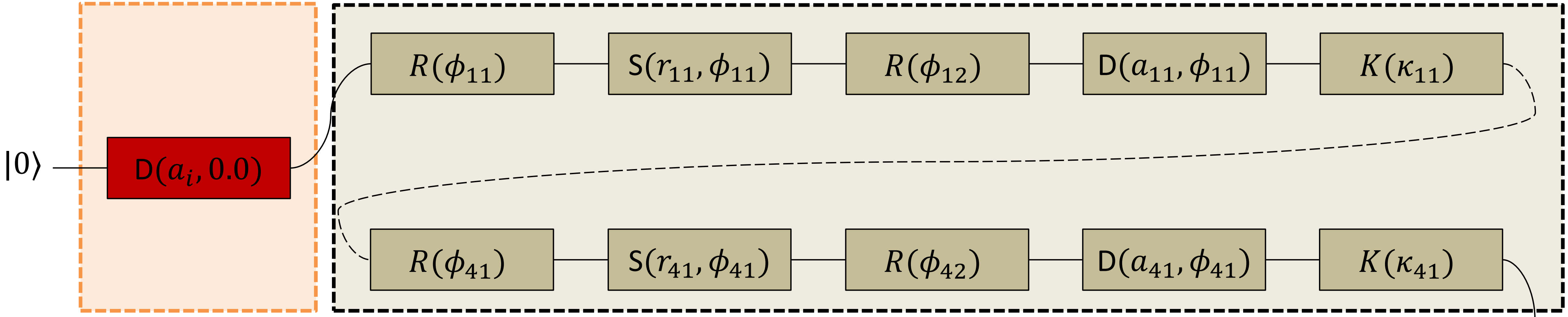


$$\mathrm{L} = \frac{1}{N}\sum_i (y_i - \langle X \rangle_i)^2$$

$$y_i = \sin(x_i)$$

Regression on CV-QC: sin(x)

A number of hidden layers: L=4

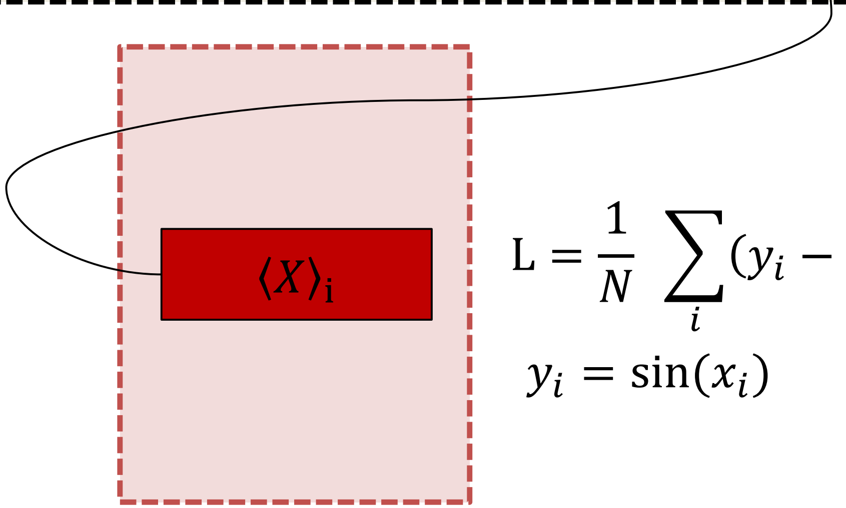


$$R(\phi) = \exp\left(i\phi\hat{a}^\dagger\hat{a}\right) = \exp\left(i\frac{\phi}{2}\left(\frac{\hat{x}^2+\hat{p}^2}{\hbar} - \hat{1}\right)\right)$$

$$S(z) = \exp\left(\frac{1}{2}(z^*\hat{a}^2 - z\hat{a}^{\dagger 2})\right) \qquad \alpha = ae^{i\phi}$$

$$D(a,\phi) = D(\alpha) = \exp(\alpha\hat{a}^\dagger - \alpha^*\hat{a}) = \exp\left(-i\sqrt{\frac{2}{\hbar}}(\mathrm{Re}(\alpha)\hat{p} - \mathrm{Im}(\alpha)\hat{x})\right)$$

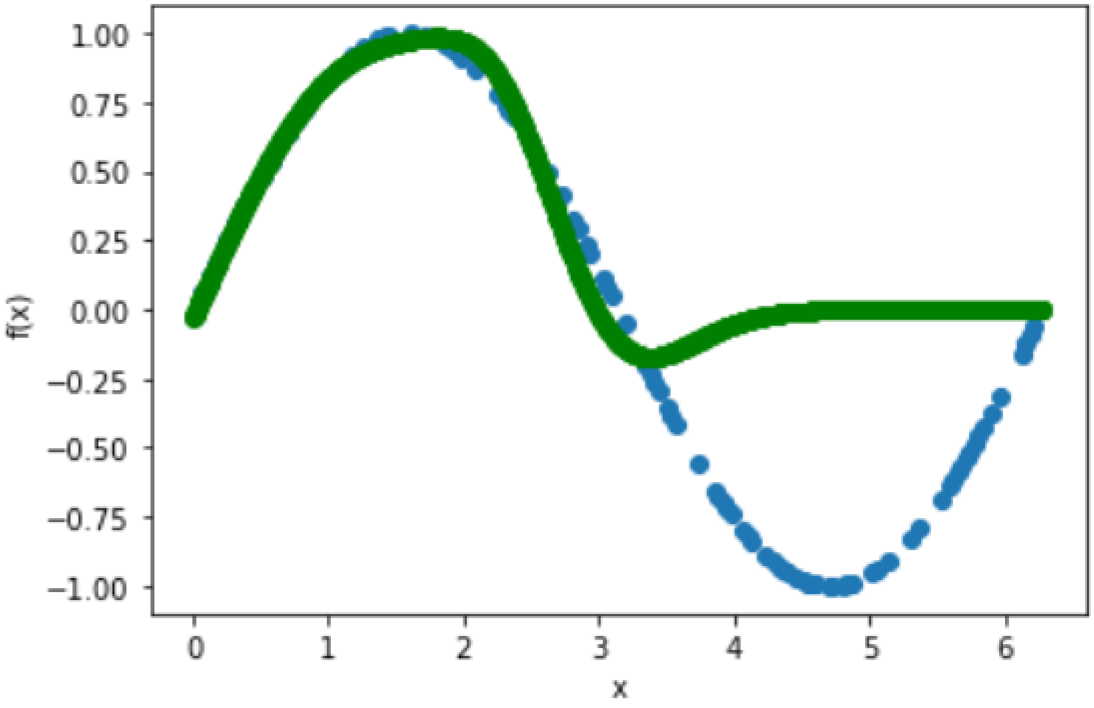
$$K(\kappa) = e^{i\kappa\hat{n}^2}$$



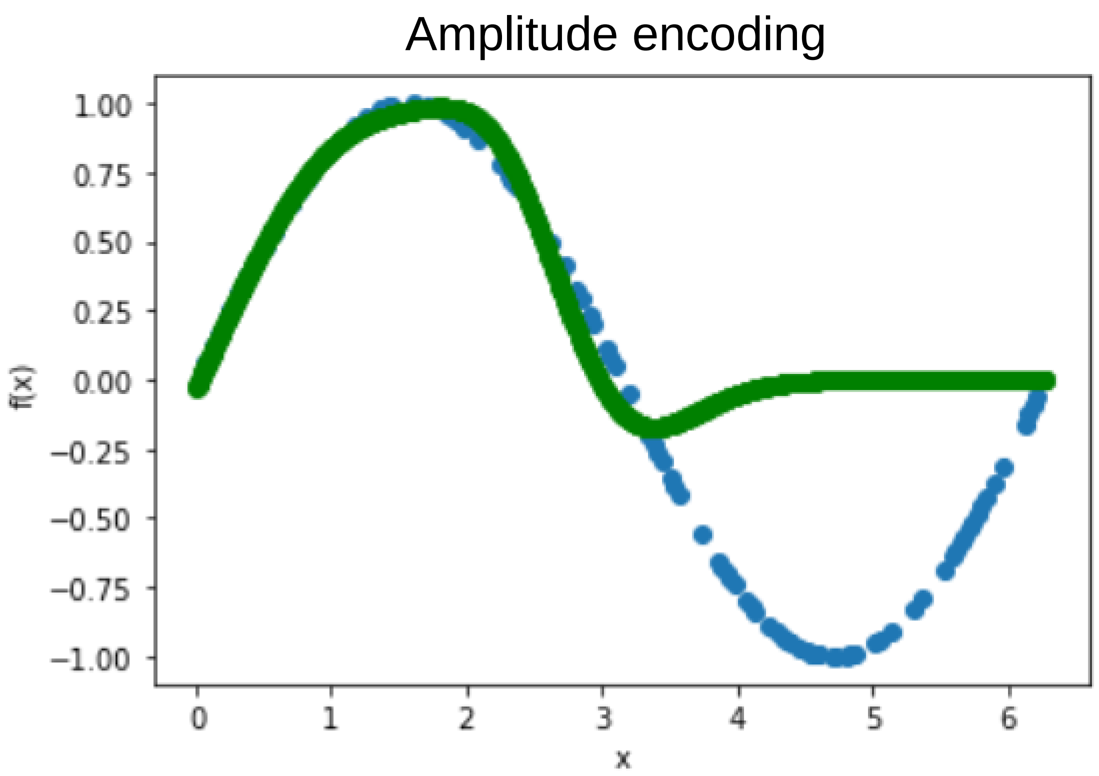
$$L = \frac{1}{N}\sum_i (y_i - \langle X\rangle_i)^2$$

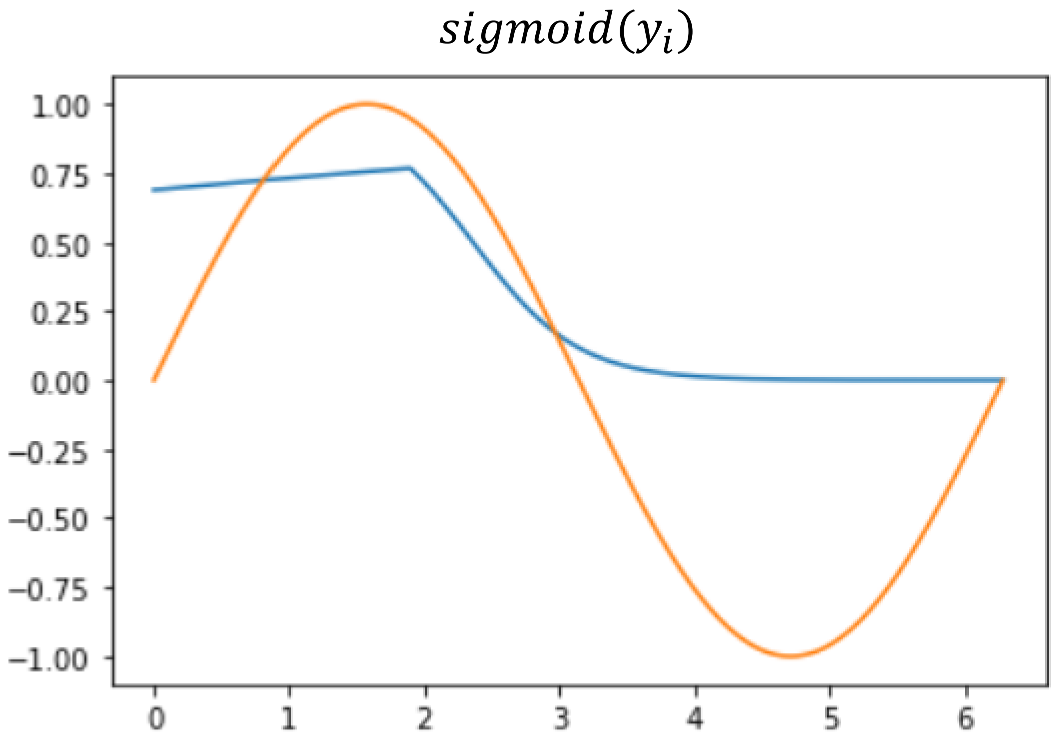
$$y_i = \sin(x_i)$$

Regression on CV-QC: amplitude encoding



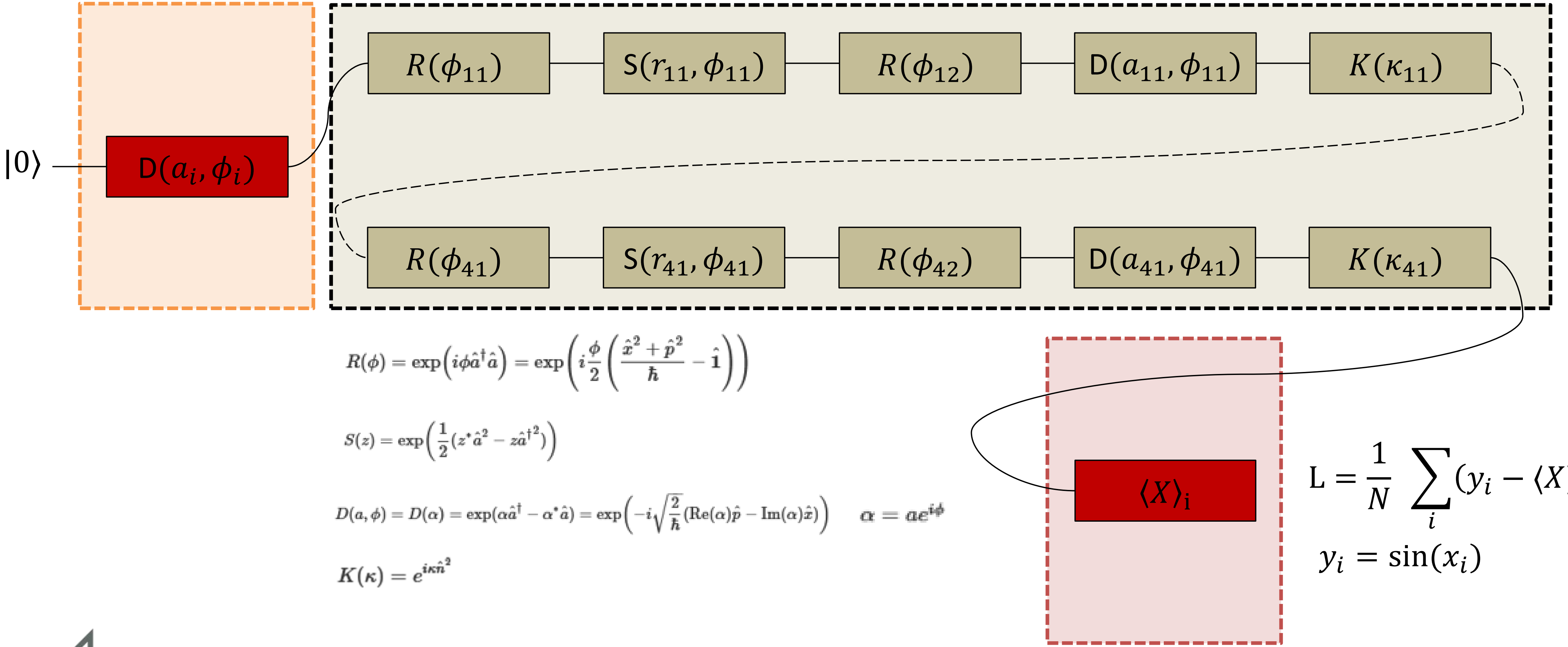
Regression on CV-QC: amplitude encoding





Regression on CV-QC: sin(x)

A number of hidden layers: L=4



$$R(\phi) = \exp\left(i\phi\hat{a}^\dagger\hat{a}\right) = \exp\left(i\frac{\phi}{2}\left(\frac{\hat{x}^2+\hat{p}^2}{\hbar} - \hat{1}\right)\right)$$

$$S(z) = \exp\left(\frac{1}{2}(z^*\hat{a}^2 - z\hat{a}^{\dagger 2})\right)$$

$$D(a,\phi) = D(\alpha) = \exp(\alpha\hat{a}^\dagger - \alpha^*\hat{a}) = \exp\left(-i\sqrt{\frac{2}{\hbar}}(\mathrm{Re}(\alpha)\hat{p} - \mathrm{Im}(\alpha)\hat{x})\right) \qquad \alpha = ae^{i\phi}$$

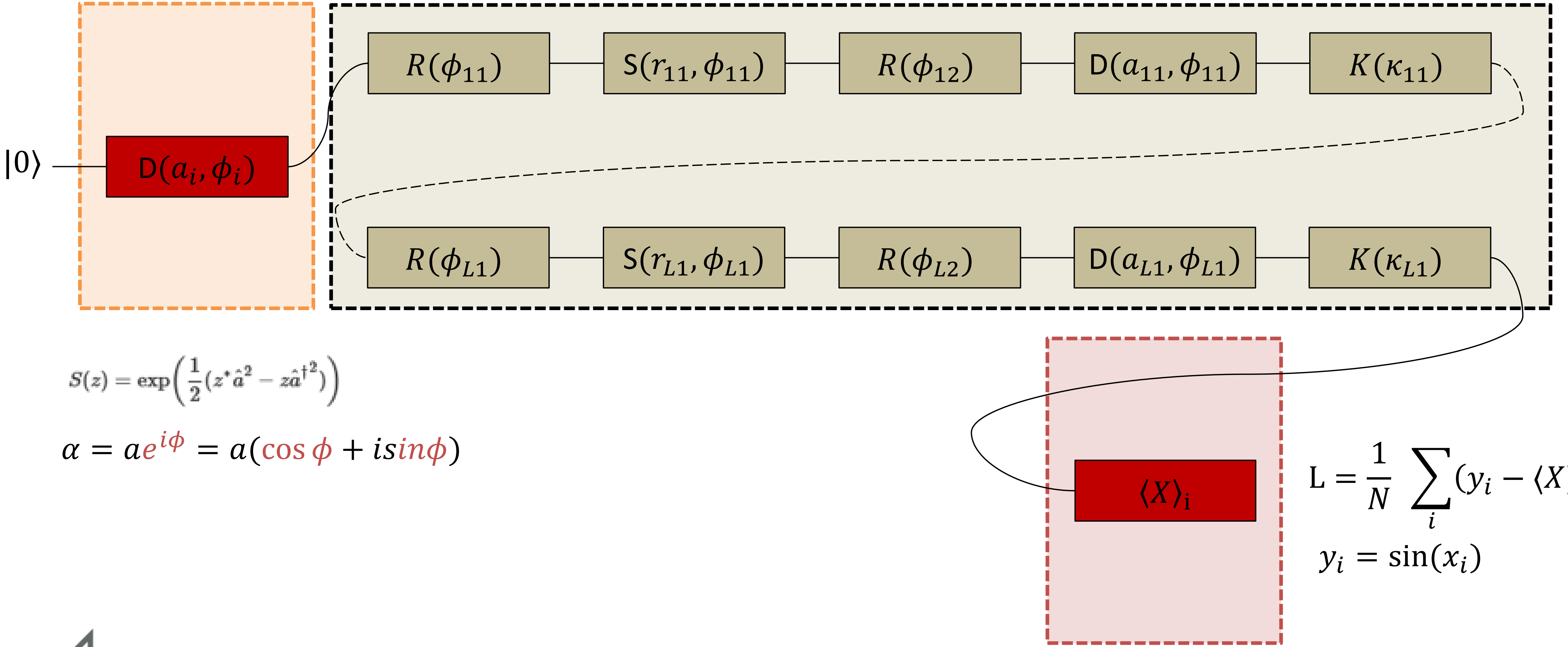
$$K(\kappa) = e^{i\kappa\hat{n}^2}$$

$$\mathrm{L} = \frac{1}{N}\sum_i (y_i - \langle X\rangle_i)^2$$

$$y_i = \sin(x_i)$$

Regression on CV-QC: sin(x)

A number of hidden layers: L=4



$$S(z) = \exp\left(\frac{1}{2}(z^* \hat{a}^2 - z\hat{a}^{\dagger 2})\right)$$

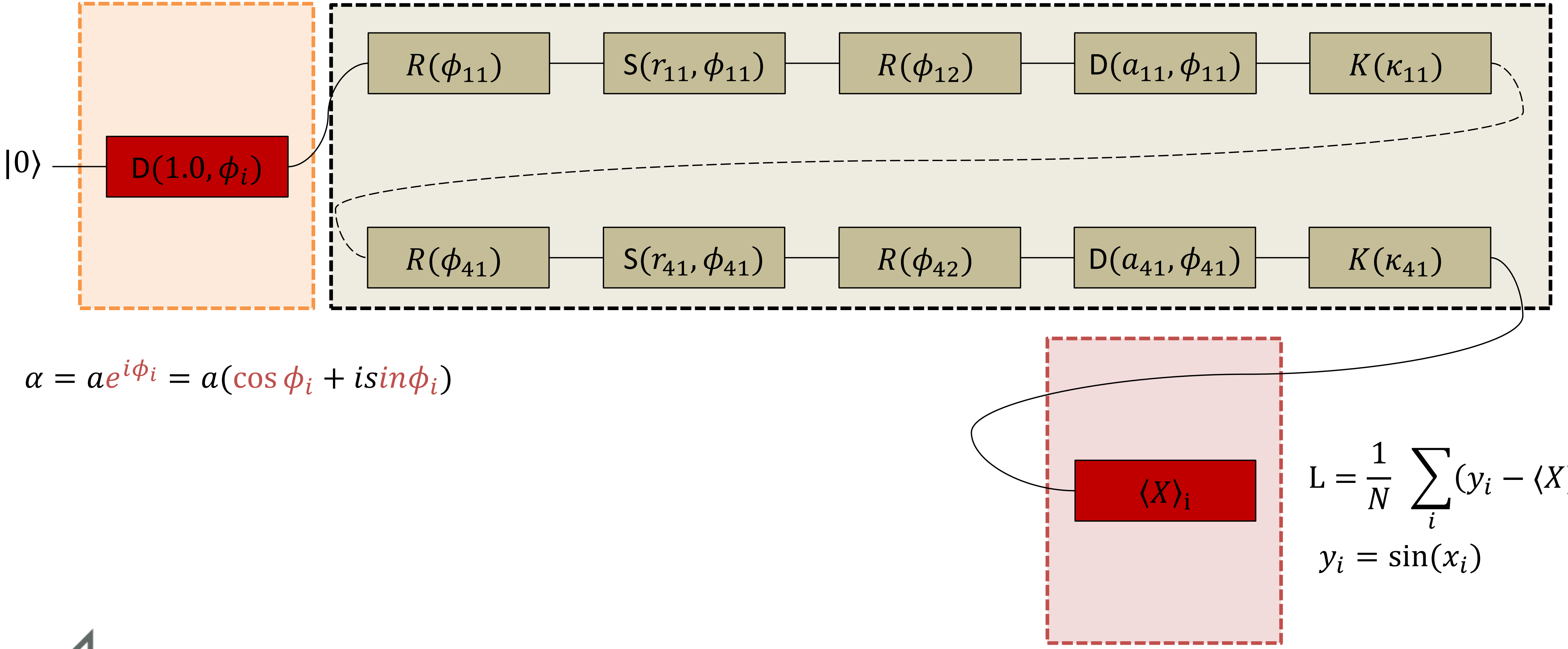
$$\alpha = a e^{i\phi} = a(\cos\phi + i sin\phi)$$

$$L = \frac{1}{N}\sum_i (y_i - \langle X \rangle_i)^2$$

$$y_i = \sin(x_i)$$

Regression on CV-QC: sin(x)

A number of hidden layers: L=4

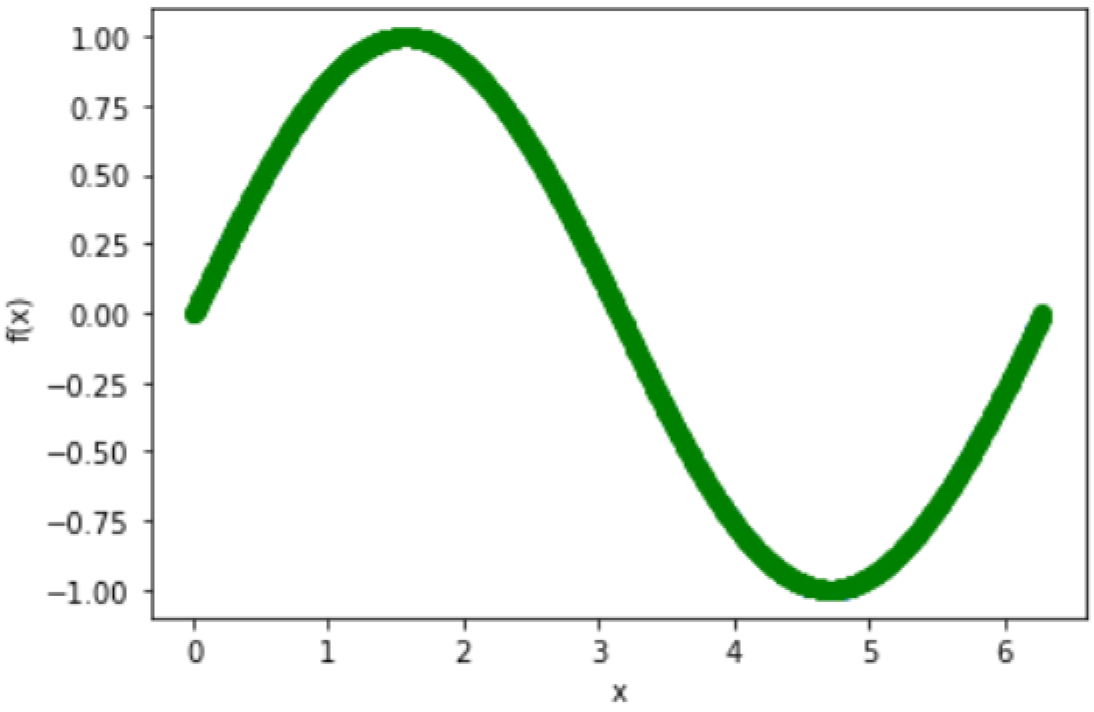


$$\alpha = a e^{i\phi_i} = a(\cos\phi_i + i sin\phi_i)$$

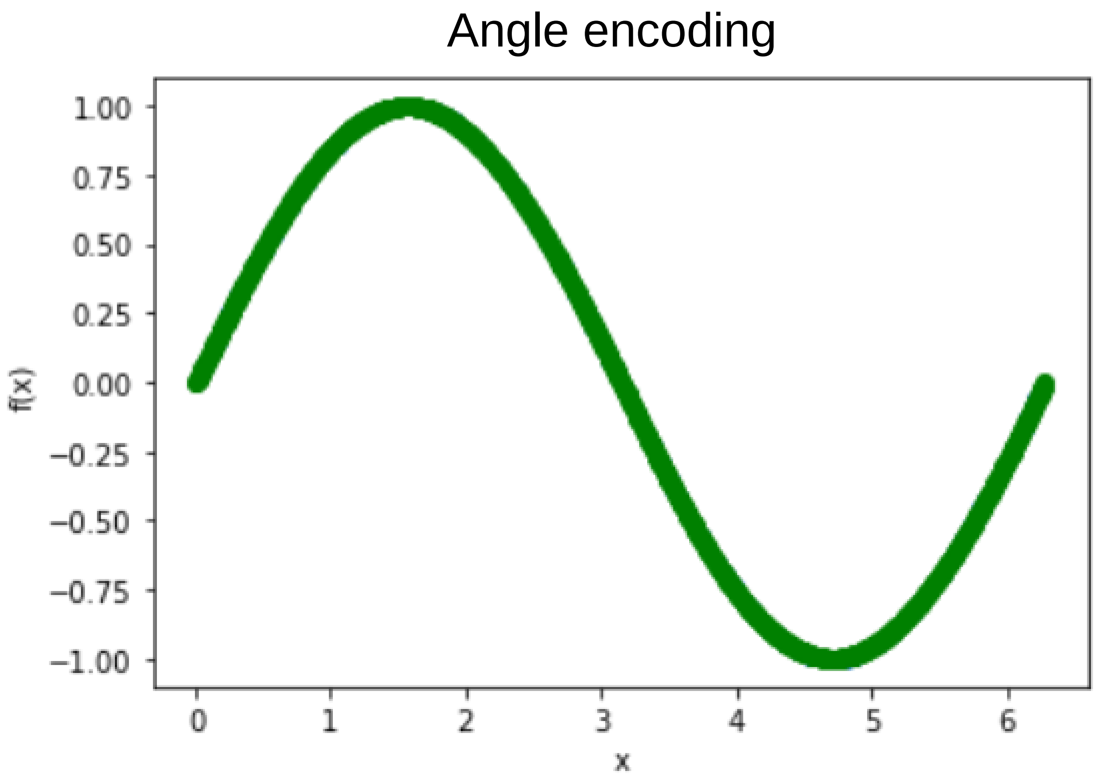
$$\mathrm{L} = \frac{1}{N} \sum_i (y_i - \langle X \rangle_i)^2$$

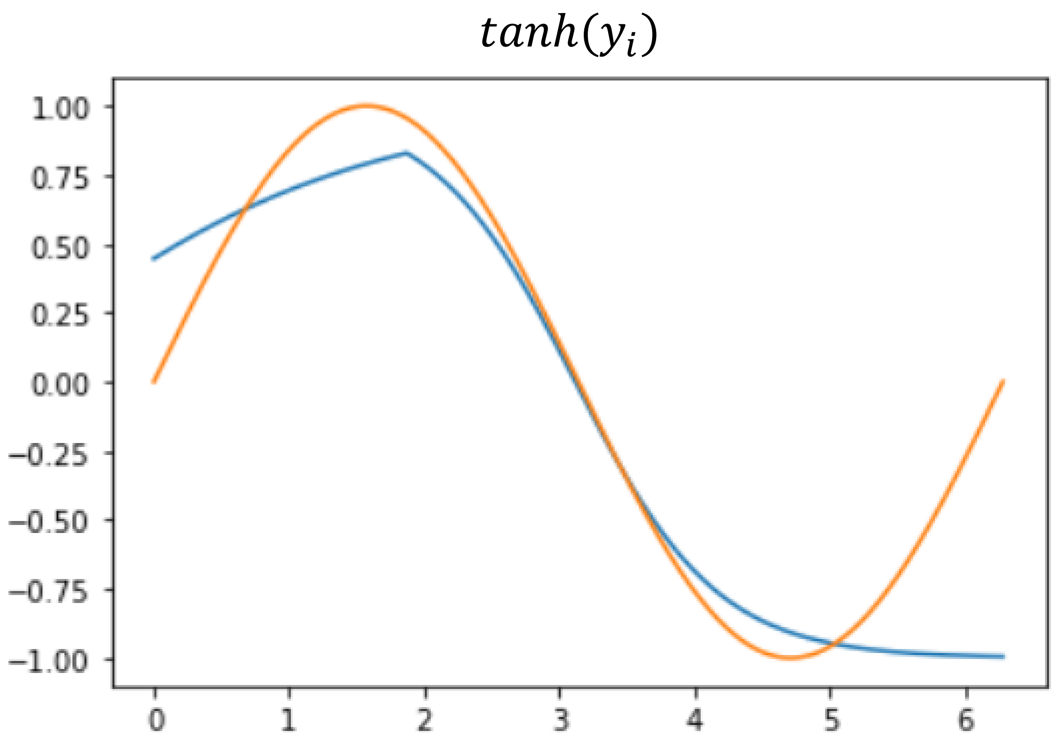
$$y_i = \sin(x_i)$$

Regression on CV-QC: angle encoding



Regression on CV-QC: angle encoding





Conclusion

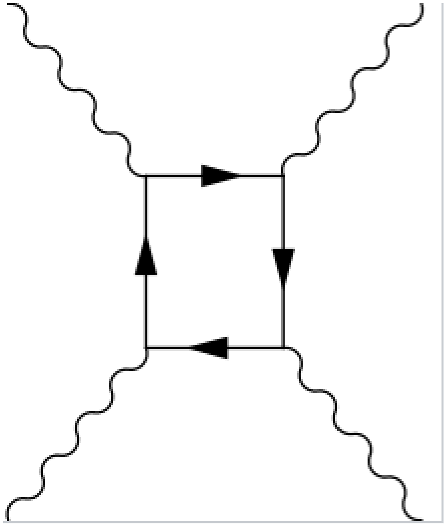
- Regression function: CV-QC works perfectly well (e.g., Rosenbrock function given x and y data)
- Optimization problems: Probably hard to use CV-QC since photons interaction is limited but we do need interactions (e.g., Rosenbrock function).
- Classification: CV-QC maybe not that much beneficial since **we do not need continous values and we do have integer labels for classification tasks.** Even encoding classical datasets to quantum circuit.

$$U(\theta)|0\rangle \approx D(a,\phi)|0\rangle$$

DV-QC CV-QC

- SAR satellites: Probably implementing CV-QC on cubesats is a very cool idea since satellites are optical computing devices.

Dream of photonic scientists



A Feynman diagram (*box diagram*) for photon–photon scattering, one photon scatters from the transient vacuum charge fluctuations of the other