European Options Pricing

Put - Call

CS 405: Quantum Computation

Prof. Jaideep Mulherkar

201701073 - Purvil Mehta

201701074 - Bhargey Mehta

201701435 - Ruchit Shah

201701438 - Niharika Dalsania

Assumptions

- A payoff function v(x) is available which provides the financial derivative to be priced.
- $v(x) : \{0,1\}^n \rightarrow [0,1]$
- A: produces a state vector with each component state $|x\rangle$ with a probability $|a_x|^2$, captures the probability distribution of the underlying asset i.e. stocks $A\left|0^{\otimes n}\right\rangle = \sum a_x \left|x\right\rangle$
- Goal: Estimate $E[v(x)] = \sum v(x) |a_x|^2$

Algorithm

- Assume we can implement a rotation R onto a ancilla qubit to get

$$R\,|x\rangle\,|0\rangle = |x\rangle\,(\sqrt{1-v(x)}\,|0\rangle + \sqrt{v(x)}\,|1\rangle)$$

- Perform R on $A\left|0^{\otimes n}\right\rangle|0\rangle$. Call this operation F

$$|\chi\rangle := \Sigma_{x=0}^{2^n-1} a_x\,|x\rangle\,(\sqrt{1-v(x)}\,|0\rangle + \sqrt{v(x)}\,|1\rangle)$$

$$F\left|0^{\otimes n+1}\right\rangle = R(A\otimes I_2)\left|0^{\otimes n+1}\right\rangle \equiv |\chi\rangle$$

Algorithm

- Consider observable $I_{2^n} \otimes |1\rangle \langle 1| \text{ on } \chi$

$$\left\langle \chi \middle| (I_{2^n} \otimes |1\rangle \langle 1|) \middle| \chi \right\rangle = \Sigma |a_x|^2 v(x) =: \mu$$

- Now we will use phase estimation method to estimate the stock price which would lead us to the QMC estimation. Consider an another observable

$$V = I_{2^{n+1}} - 2 I_{2^n} \otimes |1\rangle \langle 1|$$

Algorithm

- Note that $\langle \chi | V | \chi \rangle = \langle \chi | I_{2^{n+1}} - 2I_{2^n} \otimes |1\rangle \langle 1 | \chi \rangle = 1 - 2\mu$
- Any quantum state in the (n + 1)-qubit Hilbert space can be expressed as a linear combination of $|\chi\rangle$ and an orthogonal complement $|\chi^{\perp}\rangle$.

$$V|x\rangle = \cos\frac{\theta}{2}|\chi\rangle + e^{\iota\phi}\sin\frac{\theta}{2}|\chi^{\perp}\rangle$$

$$1 - 2\mu = \cos\frac{\theta}{2}$$

Now our task is to find θ

Phase Estimation

- We will now define the transformation **Q** that encodes $\boldsymbol{\Theta}$ in its eigenvalues. Before that, we will define two unitary operators.
- Unitary Reflection **U which acts as** $U|\chi\rangle = -|\chi\rangle$ **and** $U|\chi^{\perp}\rangle = |\chi^{\perp}\rangle$ **.**
- **-U** reflects across $|\chi\rangle$ and leaves $|\chi\rangle$ itself unchanged.

$$U = I_{2^{n+1}} - 2|\chi\rangle\langle\chi|$$

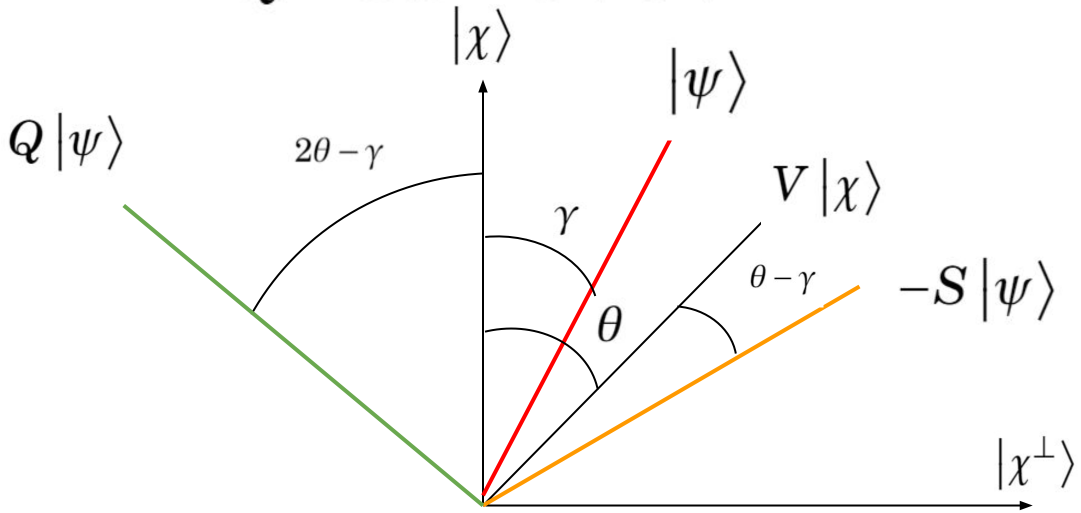
- **Similarly, we can define -S which reflects across** $V|x\rangle$

$$S = I_{2^{n+1}} - 2V|\chi\rangle\langle\chi|V \equiv VUV$$

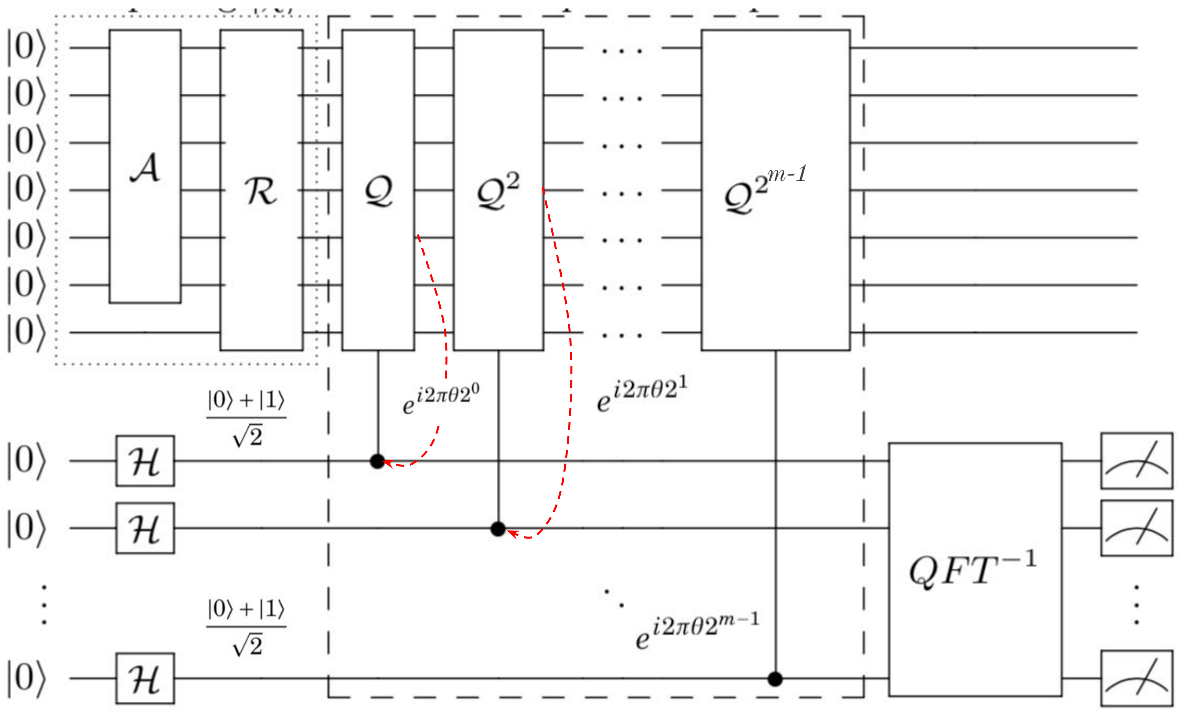
Phase Estimation

- Hence transformation **Q** can be defined as

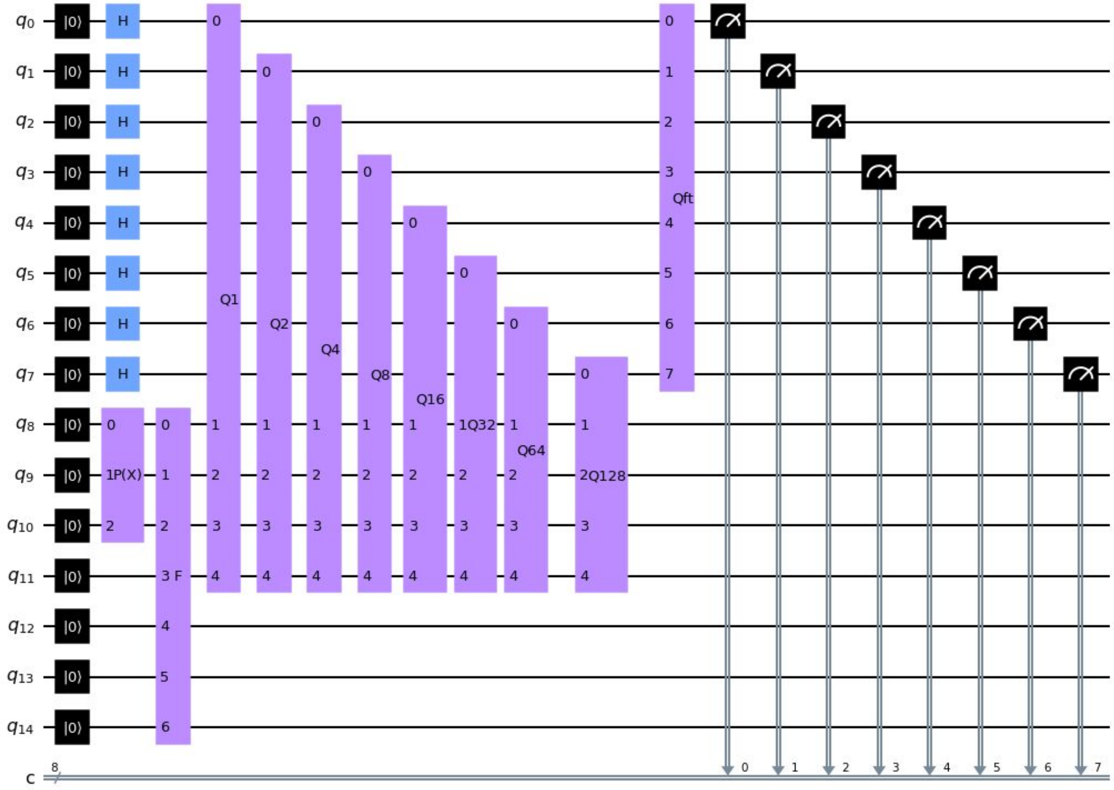
$$Q = US = UVUV$$



Phase Estimation Circuit



Custom Phase Estimation Circuit



Phase Estimation

- Phase estimation requires the conditional application of the operator Q.

$$Q^c : |j\rangle \left|\psi\right\rangle \rightarrow |j\rangle Q^j \left|\psi\right\rangle$$

- Applying hadamard gate on first m qubits, we got,

$$H^{\otimes m} \left|0^m\right\rangle \left|\chi\right\rangle = \frac{1}{\sqrt{2^m}} \Sigma_0^{2^m-1} |j\rangle \left|\chi\right\rangle$$

Notice that $0.\theta_1 ... \theta_{m-1}\theta_m = \theta$

Phase Estimation

- Finally applying Q^c operator and inverse QFT,

$\theta_1 ... \theta_{m-1}.\theta_m =$ $0.\theta_m$

$0.\theta_{m-1}\theta_m$

$$\frac{1}{\sqrt{2^m}}\Sigma_0^{2^m-1}|j\rangle Q^j|\chi\rangle = \boxed{(|0\rangle + e^{i2\pi 2^{m-1}\theta}|1\rangle)}\boxed{(|0\rangle + e^{i2\pi 2^{m-2}\theta}|1\rangle)} \times$$

$$\times \cdots \times \boxed{(|0\rangle + e^{i2\pi\theta}|1\rangle)}$$

$$= \sum_{y=0}^{2^m-1} e^{i2\pi\theta y}|y\rangle.$$

$0.\theta_1 ... \theta_{m-1}\theta_m = \theta$

Option Pricing

- $v(x)$ is the pay off function when considering some distribution
- $E[v(x)]$ is the price of the option at time T
- Then the price of the option would be the discounted value

$$\Pi = e^{-rT}\mathbb{E}_{\mathbb{Q}}[v(W_T)]$$

- The probability density for this random variable is given by $P_T(x)$
- To prepare an approximate superposition of these probabilities, we take the support of this density form and discretize the interval $[-x_{max}, x_{max}]$.
- $$p_j = \frac{p_T(x_j)}{C} \qquad C = \sum_{j=0}^{2^n-1} p_T(x_j)$$

Option Pricing

- We now define the algorithm G which takes on the role of A described earlier, generates the superposition of n bits.

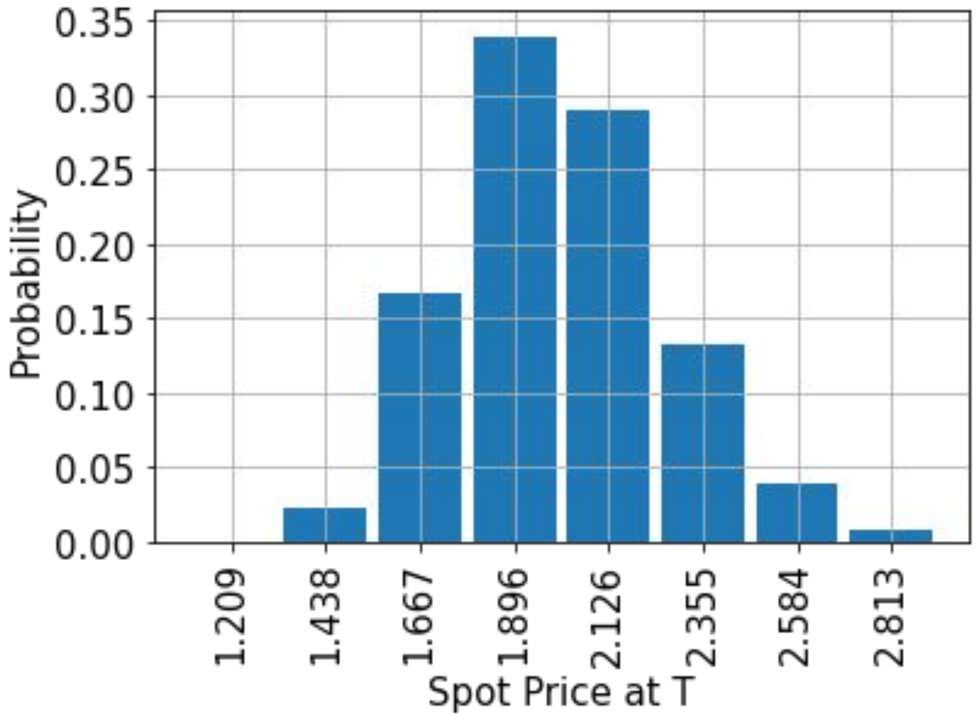
$$\mathcal{G}|0^n\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_j}|j\rangle$$

- Similarly v(x) function can be defined as

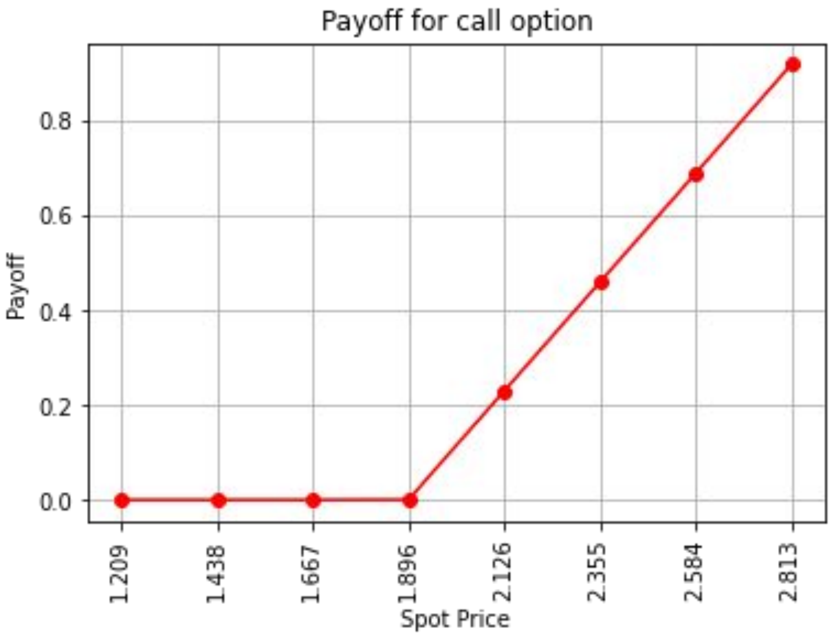
$$v_{\text{euro}}(x) = \max\{0, S_0 e^{\sigma x + (r - \frac{1}{2}\sigma^2)T} - K\}$$

Experiments - Lognormal

- Parameters
  - Spot price = 2
  - Volatility = 0.4
  - r = 0.05
  - T = 40 days
  - Strike price = 1.896



Experiments - Lognormal (Call)



Exact expected value: 0.1623

Estimated expected value: 0.1681

Error percentage: 3.57%

Experiments - Lognormal (Put)

Exact expected value: 0.0490

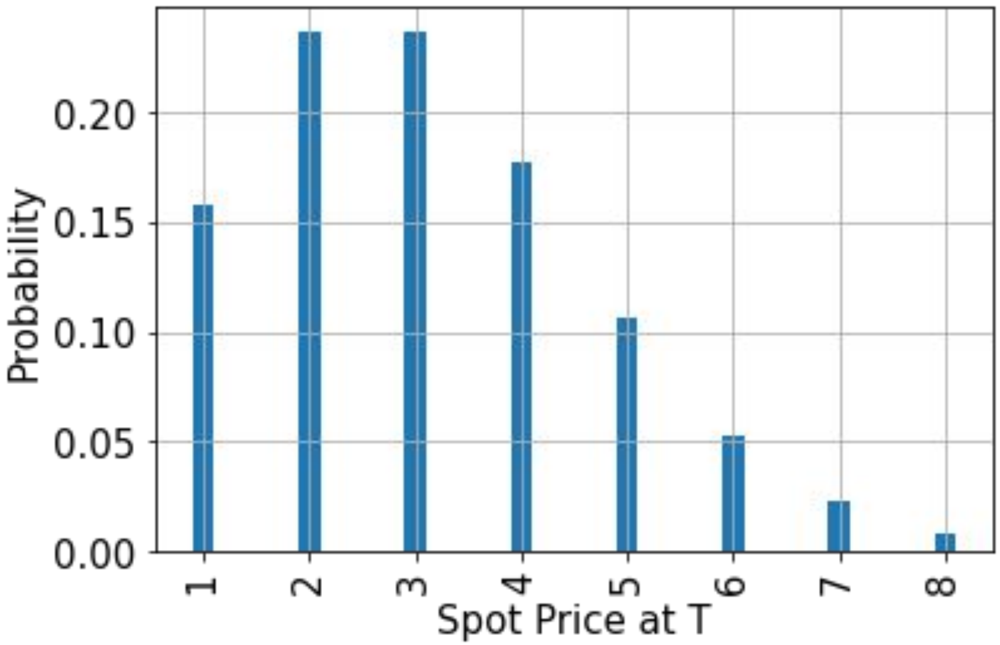
Estimated expected value: 0.0565

Error percentage: 15.3%

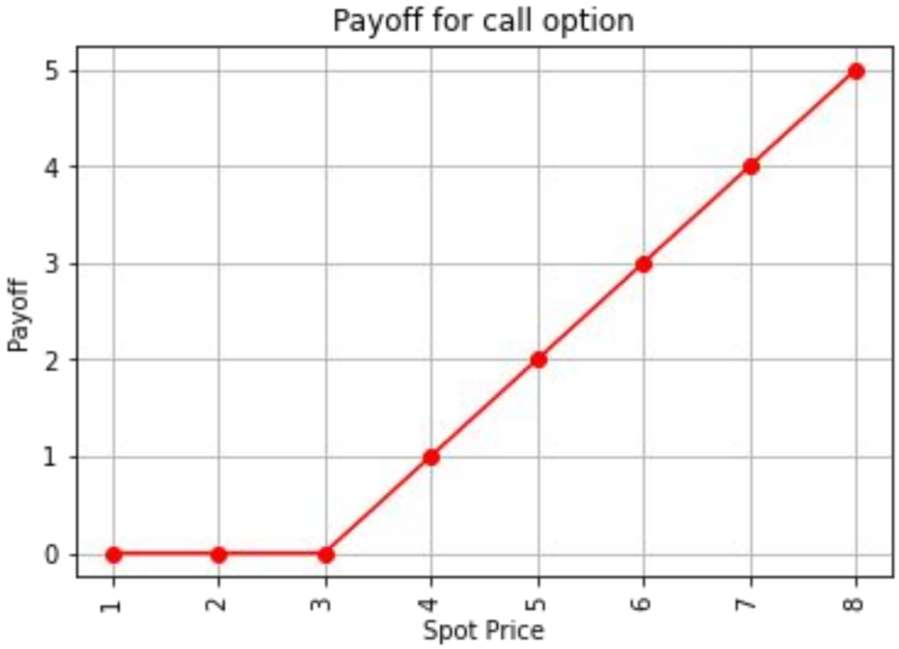
Experiments - Poisson

- Binomial → approximation → Poisson

- Parameters
  - Mean = 3
  - Standard deviation = 2



Experiments - Poisson (Call)



Exact expected value: 0.6845

Estimated expected value: 0.7198

Error percentage: 5.0%

Experiments - Poisson (Put)

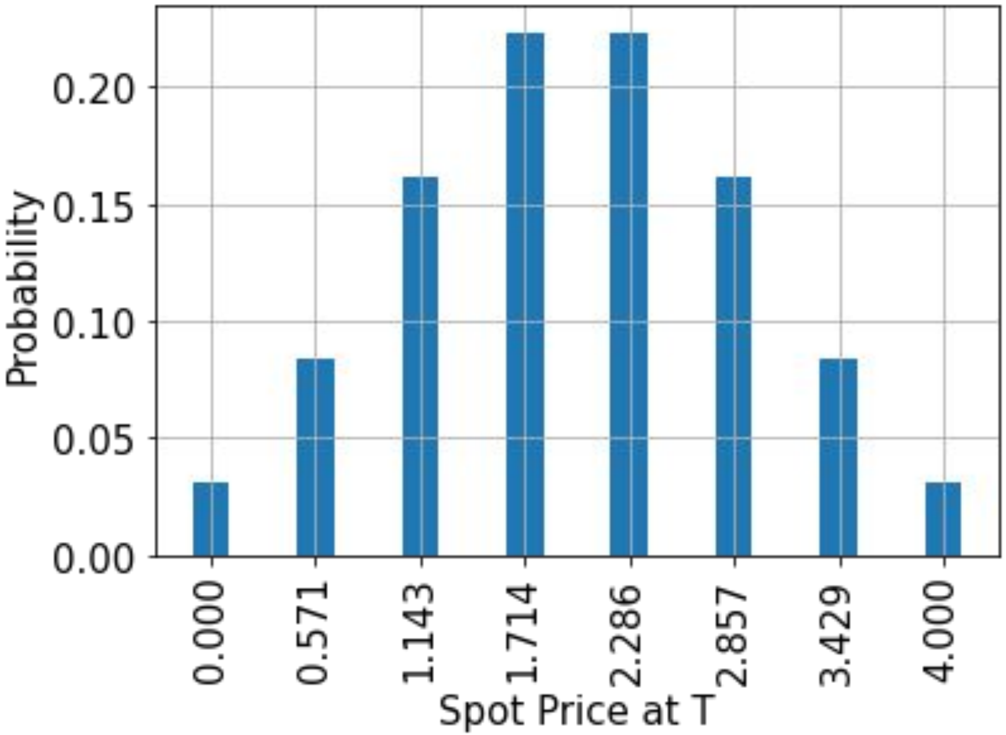
Exact expected value: 0.5524

Estimated expected value: 0.5565

Error percentage: 0.7%

Experiments - Gaussian

- Parameters
    - Mean = 2
    - Standard Deviation = 1



Experiments - Gaussian (Call)

Exact expected value: 0.3848

Estimated expected value: 0.3992

Error percentage: 3.7%

Experiments - Gaussian (Put)

Exact expected value: 0.3848

Estimated expected value: 0.3881

Error percentage: 0.8%

Conclusion

- For classical Monte Carlo:
  - Each sample is independent of other
  - By Central Limit Theorem, variance = $\sigma^2_{\hat{x}} = \sigma^2_x \,/\, t$
  - **For accuracy ε, t=O(1/ε^2) iterations.**
- For Quantum Monte Carlo:
  - Theorems (Amplitude estimation, Mean estimation for bounded functions, Mean estimation for bounded variance) show - **For accuracy ε, t=O(1/ε) iterations**.
- Hence, quadratic speedup!

Thank you!