What is the HHL algorithm and how to implement it on a quantum computer?

Otmar Ubbens, Carmina G. Almudever[1], Matthias Möller[2]

[1] Quantum Computer Architecture Lab – Q&CE department

[2] Department of Applied Mathematics

TUDelft

What is the HHL algorithm and how to implement it on a quantum computer?

What is a quantum computer?

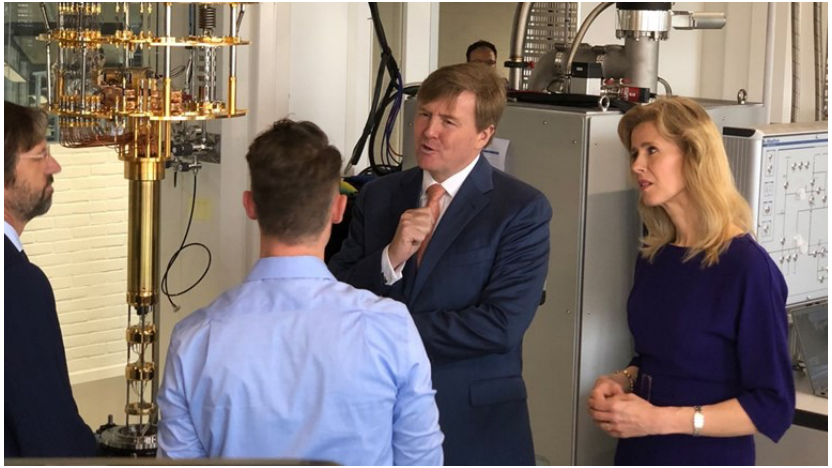
Otmar Ubbens, Carmina G. Almudever[1], Matthias Möller[2]

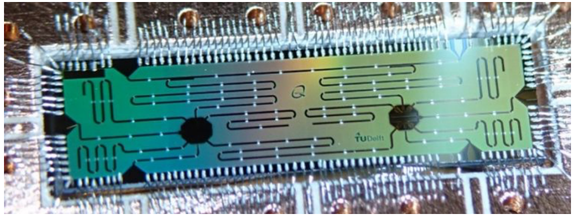
[1] Quantum Computer Architecture Lab – Q&CE department

[2] Department of Applied Mathematics

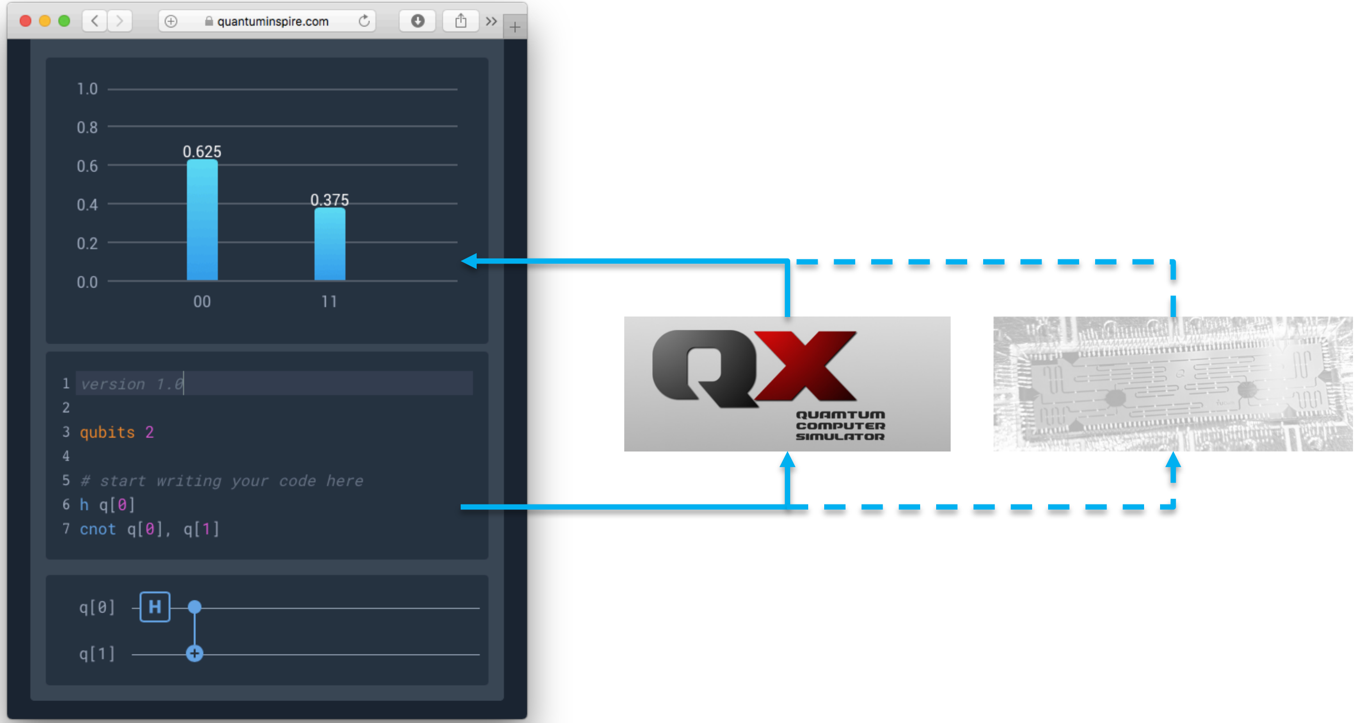
TU Delft

What is a quantum computer?





Quantum computer simulator



Quantum computing in a nutshell

- **Qubit state:** coherent superposition of standard basis states

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle, \qquad \alpha_k \in \mathbb{C}$$

- **Interpretation:**
  - $|\alpha_0|^2$ probability of measuring $|0\rangle$
  - $|\alpha_1|^2$ probability of measuring $|1\rangle$
  - Requirement: $|\alpha_0|^2 + |\alpha_1|^2 = 1$

TU Delft

Quantum computing in a nutshell

- **Qubit register:** coherent superposition of product basis states

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle, \qquad \alpha_k \in \mathbb{C}$$

- **Interpretation:**
  - $|\alpha_0|^2$ probability of measuring $|00\rangle$
  - $|\alpha_1|^2$ probability of measuring $|01\rangle$
  - …
  - Requirement: $|\alpha_0|^2 + |\alpha_1|^2 + |\alpha_2|^2 + |\alpha_3|^2 = 1$

TU Delft

Quantum computing in a nutshell

- **Quantum gate:** unitary operator acting on probability amplitudes

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}\alpha_0\\ \alpha_1\end{bmatrix} = \frac{\alpha_0+\alpha_1}{\sqrt{2}}|0\rangle + \frac{\alpha_0-\alpha_1}{\sqrt{2}}|1\rangle$$

- Example:

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle =: |+\rangle$$

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle =: |-\rangle$$

TU Delft

Quantum computing in a nutshell

- **Bell state:** entangled qubit states

$$C_{\text{NOT}}\left(H|0\rangle_A, |0\rangle_B\right) =$$

$$C_{\text{NOT}}\left(\frac{1}{\sqrt{2}}|00\rangle + 0|01\rangle + \frac{1}{\sqrt{2}}|10\rangle + 0|11\rangle\right) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} =$$

$$\frac{1}{\sqrt{2}}|00\rangle + 0|01\rangle + \frac{1}{\sqrt{2}}|11\rangle + 0|10\rangle$$

TU Delft

Find

$$q = \vec{x}^\dagger M \vec{x}$$

such that

$$A\vec{x} = \vec{b}$$

$N \times N$ Hermitian s-sparse matrix A with low condition number κ and $\mathcal{O}(s)$ access to entries from row index

CG algorithm $\mathcal{O}(Ns\kappa)$
scalar output $\mathcal{O}(N\sqrt{\kappa})$

Exponential speed-up

HHL algorithm
$\mathcal{O}(\kappa^2 \log N / \epsilon)$

PRL **103,** 150502 (2009) PHYSICAL REVIEW LETTERS week ending 9 OCTOBER 2009

Quantum Algorithm for Linear Systems of Equations

Aram W. Harrow,[1] Avinatan Hassidim,[2] and Seth Lloyd[3]

[1]*Department of Mathematics, University of Bristol, Bristol, BS8 1TW, United Kingdom*

[2]*Research Laboratory for Electronics, MIT, Cambridge, Massachusetts 02139, USA*

...boratory for Electronics and Department of Mechanical Engineering, MIT, Cambridge, Massachusetts 02139, USA

2009)

...oth on its own and as a subroutine ... $\vec{x}$ such that $A\vec{x} = \vec{b}$. We consider ... rather an approximation of the expectation value of some operator associated with $\vec{x}$, e.g., $\vec{x}^\dagger M \vec{x}$ for some matrix M. In this case, when A is sparse, $N \times N$ and has condition number κ, the fastest known classical algorithms can find $\vec{x}$ and estimate $\vec{x}^\dagger M \vec{x}$ in time scaling roughly as $N\sqrt{\kappa}$. Here, we exhibit a quantum algorithm for estimating $\vec{x}^\dagger M \vec{x}$ whose runtime is a polynomial of $\log(N)$ and κ. Indeed, for small values of κ [i.e., poly $\log(N)$], we prove (using some common complexity-theoretic assumptions) that any classical algorithm for this problem generically requires exponentially more time than our quantum algorithm.

DOI: 10.1103/PhysRevLett.103.150502 PACS numbers: 03.67.Ac, 02.10.Ud, 89.70.Eg

TU Delft

The math behind HHL

$$A\vec{v} = \lambda \vec{v}$$

A Hermitian

$$\vec{b} = \sum_j \alpha_j \vec{v}_j$$

$$\vec{x} = \sum_j \frac{1}{\lambda_j} a_j \vec{v}_j$$

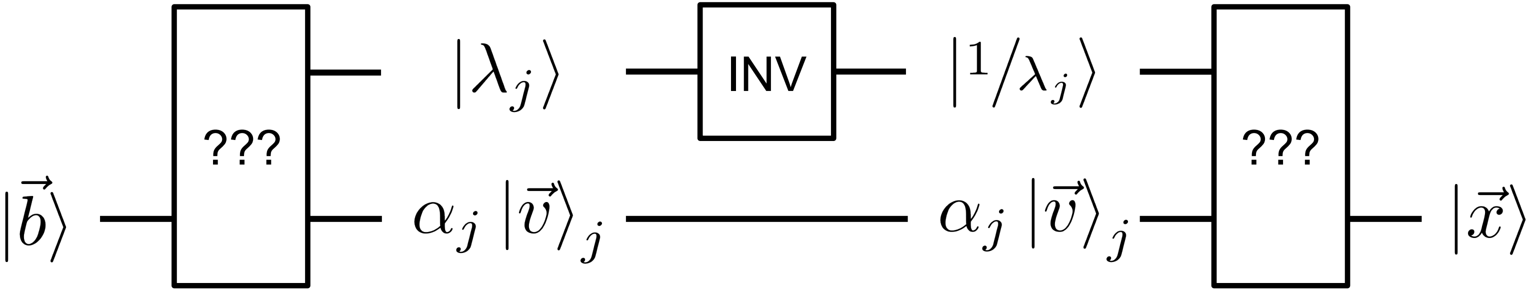
$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix} \implies \begin{matrix} \lambda_0 = 1 \\ \lambda_1 = 2 \\ \lambda_2 = 4 \\ \lambda_3 = 8 \end{matrix} \quad \& \quad \vec{v}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ \vec{v}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \ \vec{v}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \ \vec{v}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

A Hermitian

$$\vec{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \vec{v}_1 + \vec{v}_3 \implies \vec{x} = \frac{1}{2}\vec{v}_1 + \frac{1}{8}\vec{v}_3 = \begin{bmatrix} 0 \\ 1/2 \\ 0 \\ 1/8 \end{bmatrix}$$

HHL algorithm

- Construct $|\vec{b}\rangle$
- Eigen decomposition
- Invert eigenvalues
- Fused-multiply-add



From vectors to qubits

$$\vec{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} \quad \overset{||\vec{b}||_2 = 1}{\Longrightarrow} \quad |\vec{b}\rangle = ?\ \beta_0 |00\rangle + \beta_1 |01\rangle + \beta_2 |10\rangle + \beta_3 |11\rangle$$

$$\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{v}_3 \quad \overset{||\vec{v}_j||_2 = 1}{\Longrightarrow} \quad |\vec{v}_0\rangle, |\vec{v}_1\rangle, |\vec{v}_2\rangle, |\vec{v}_3\rangle$$

$$\Downarrow \text{ Linearity}$$

A Hermitian

$$\vec{b} = \sum_j \alpha_j \vec{v}_j \quad \Longrightarrow \quad |\vec{b}\rangle = \sum_j \alpha_j |\vec{v}_j\rangle$$

TU Delft

Eigenvalue decomposition

"Hamiltonian simulation"

e^{iAt}

Same eigenvectors:

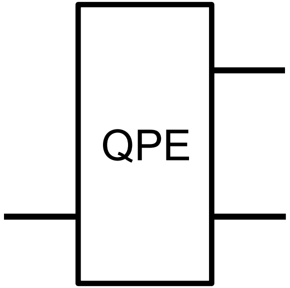
$$\vec{v}_j \rightarrow \vec{v}_j$$

Different eigenvalues:

$$\lambda_j \rightarrow e^{i\lambda_j t}$$

Velocity λ_j

$|\vec{b}\rangle$



$$|\lambda_j\rangle \quad |\lambda_0\rangle + |\lambda_1\rangle + \dots$$

$$|\vec{v}_j\rangle \quad \alpha_0 |\vec{v}_0\rangle + \alpha_1 |\vec{v}_1\rangle + \dots$$

$$= \sum_j \alpha_j |\lambda_j\rangle |\vec{v}_j\rangle$$

$$= \sum_j \alpha_j |\vec{v}_j\rangle$$

"coupled"

Last step

$$|\vec{b}\rangle = \sum_j \alpha_j |\vec{v}_j\rangle$$

$$\xrightarrow{\text{QPE}} \sum_j \alpha_j |\lambda_j\rangle |\vec{v}_j\rangle$$

$$\xrightarrow{\text{INV}} \sum_j \alpha_j |{}^1/\lambda_j\rangle |\vec{v}_j\rangle$$

$$|\vec{x}\rangle = \sum_j \alpha_j \frac{1}{\lambda_j} |\vec{v}_j\rangle$$

→ ?

"Ancilla"

$$|0\rangle$$

Ancilla Rotation

$$\sum_j \alpha_j |0\rangle |{}^1/\lambda_j\rangle |\vec{v}_j\rangle$$

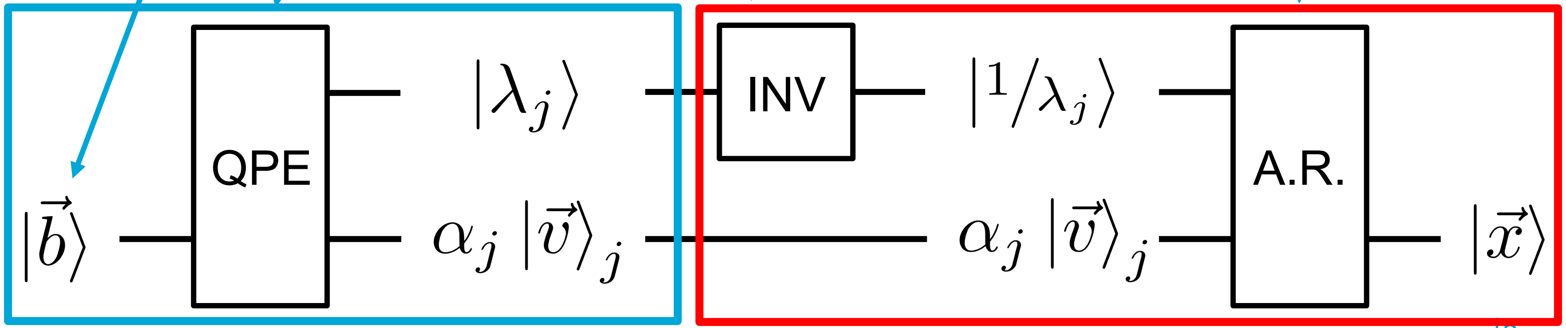
$$\text{“=”} \quad |\vec{x}\rangle$$

Challenges in practice

- Vector implementation
- Hamiltonian simulation
- Eigenvalue inversion
- Ancilla rotation

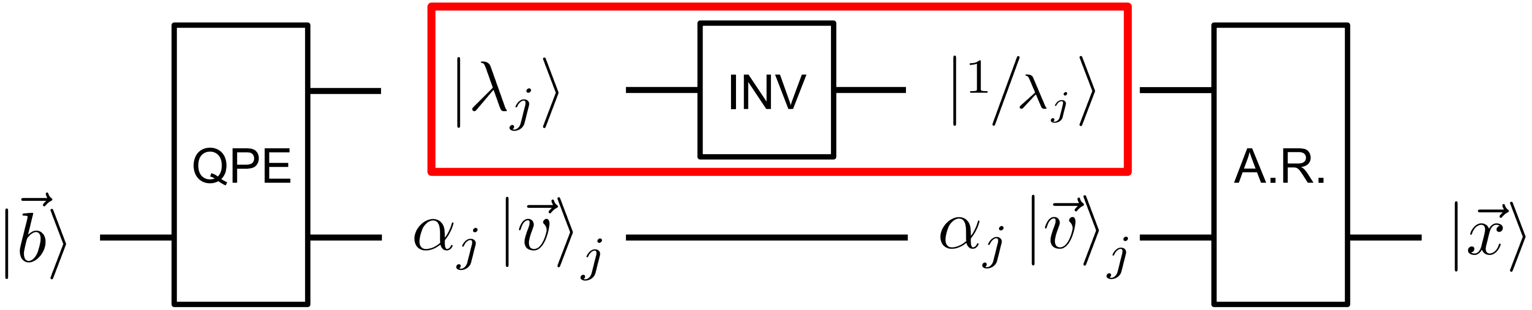
Proof-of-concept implementation

Full implementation



Eigenvalue inversion

- Three methods:
  - ~~Quantum algorithm (Cao et al. 2012)~~ Didn't work
  - ~~Newton Raphson (Sugg. by Cao et al. 2012)~~ Inefficient
  - Long division (Thapliyal et al. 2017)



Thapliyal division

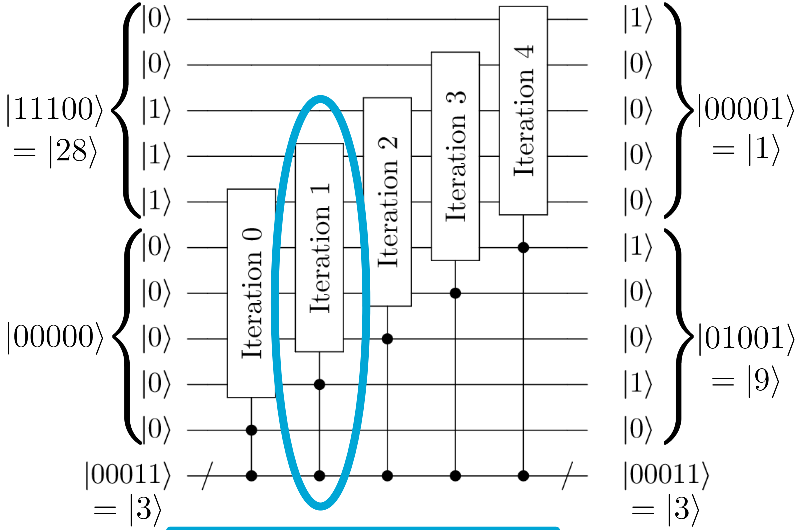
$$\left.\begin{aligned} n &= 28 = 11100 \\ d &= 03 = 00011 \end{aligned}\right\} n/d = ?$$

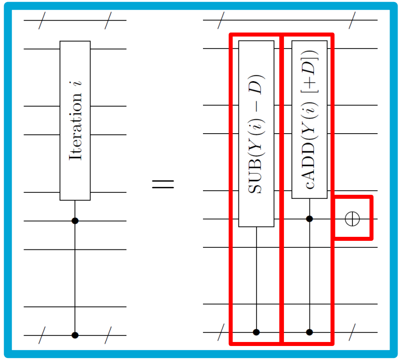
$11 \,/\, 11100 \,\backslash\, 1\,0\,0\,1 = 9$

```
11 / 11100 \ 1 0 0 1  = 9
     11000
   ---------
       100
      1100
   ---------
       100
       110
   ---------
       100
        11
   ---------
         1
```

if $2^k d < n$:
  SUB, ADD 1
else:
  ADD 0

$28 \stackrel{?}{=} 9 \cdot 3 + 1$

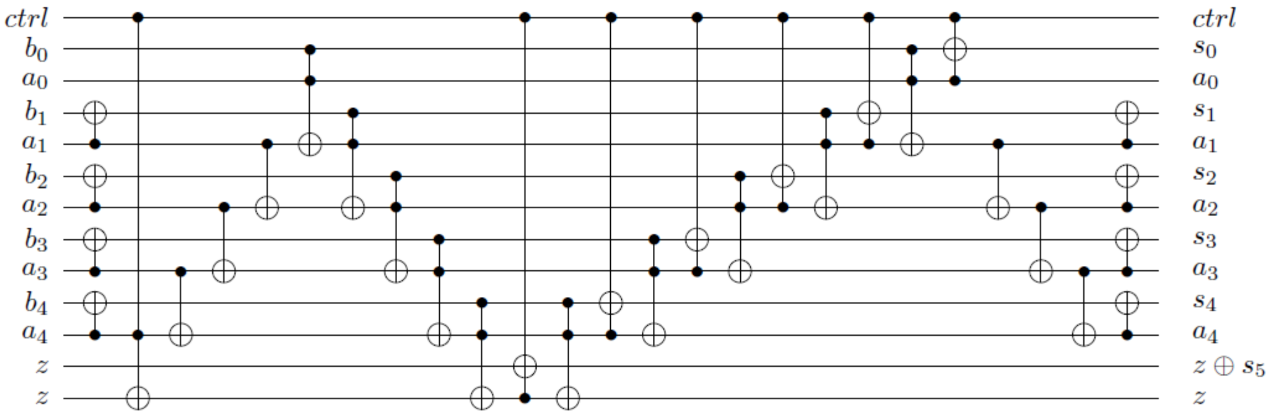




- SUB
- cADD
- FLIP

cADD – How difficult can it be?

Category	Draper	Cuccaro		Muñoz-Coreas		
	Default	Default	Compact	Default	No control	No overflow
Number of gates[1]	$\frac{3}{2}n(n-1)$	$6n+1$	$9n-8$	$7n-4$	$7n-6$	$7n-8$
Circuit Depth	n^2	$6n+1$[4]	$2n+2$	$5n$	$5n-2$	$5n-4$
Number of ancillae	0[2]	1	1	1	0	0
Controlled	Yes[2]	Yes	No	Yes	No	Yes
Overflow	Yes[3]	Yes	Yes	Yes	Yes	No
Unequal register size	Yes	No	No	No	No	No
Only basic gates	No	Yes	Yes	Yes	Yes	Yes



TU Delft

From division to inversion

Division: ${}^{n}/_{d}$

Inversion: ${}^{1}/_{\lambda}$

~~**Take:** $n = 1$ & $d = \lambda$~~

Integers!

${}^{1}/_{\lambda} < 1$ **!!!**

Base 10: ${}^{1}/_{7} = 0.14285714\ldots$ "=" ${}^{1000000}/_{7} = 142857.14\ldots$ $= {}^{10^k}/_{7}$

Binary: ${}^{1}/_{7} = 0.001001001\ldots$ "=" ${}^{2^k}/_{7} = 1001.001\ldots$ **Integers!**

Cut-off at *k*-th decimal!

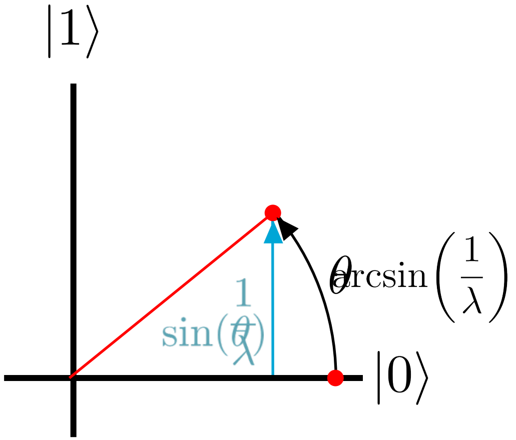
TU Delft

Ancilla rotation

Want: $|0\rangle$ — A.R. — $\frac{1}{\lambda}|1\rangle + |\text{garbage}\rangle$

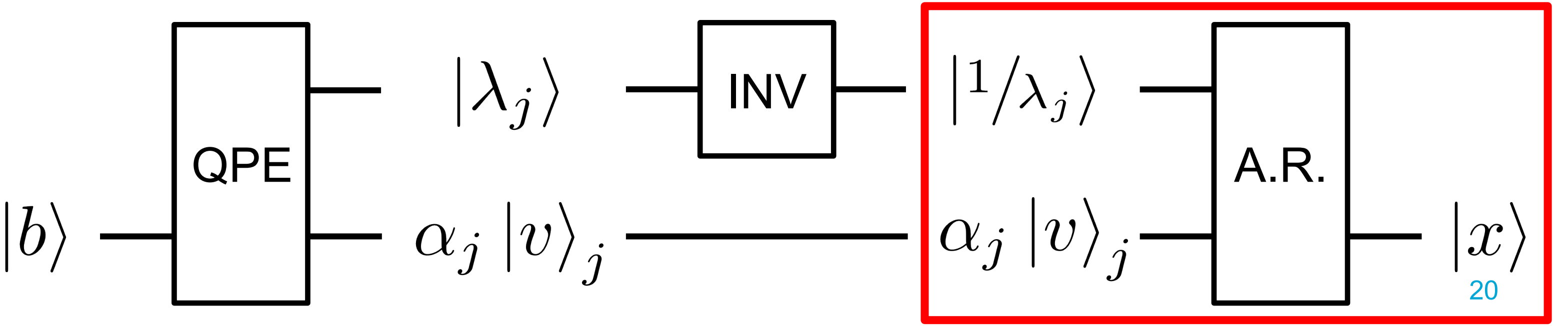
$\theta = ?\arcsin(1/\lambda)$

Have: $|0\rangle$ — $R_y(\theta)$ — $\sin(\theta)|1\rangle + \cos(\theta)|0\rangle$



Taylor expansion:

$$\arcsin(x) = x + \frac{1}{6}x^3 + \frac{3}{40}x^5 + \frac{5}{112}x^7 + \mathcal{O}\left(x^9\right)$$

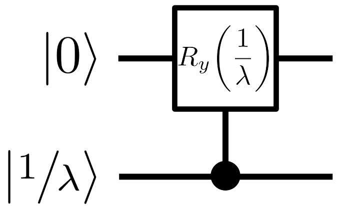


Cao's approximation

$\arcsin(x) \cong x + \frac{1}{6}x^3 + \frac{3}{40}x^5 + \frac{5}{112}x^7 + O(x^9)$

$R_y(a+b) = R_y(a)R_y(b)$

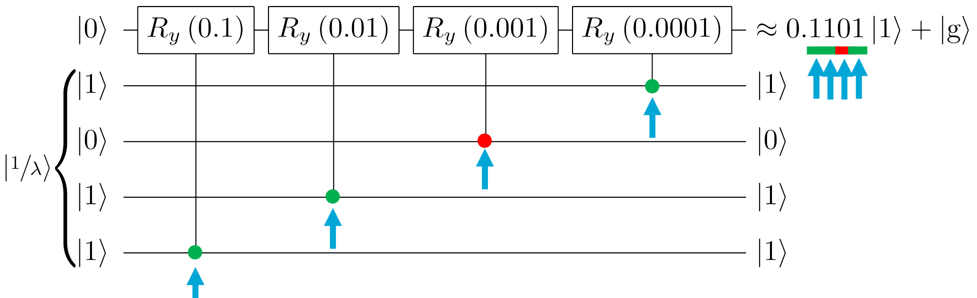
Approximation:

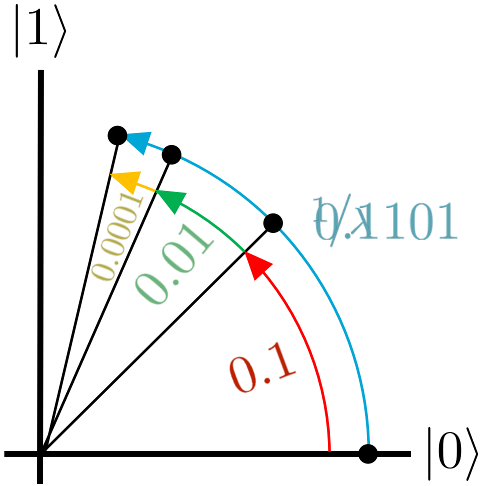


$\approx \frac{1}{\lambda}|1\rangle + |\text{garbage}\rangle$

e.g.

$= |0.1101\rangle = |0.1 + 0.01 + 0.0001\rangle$





Higher-order approximations

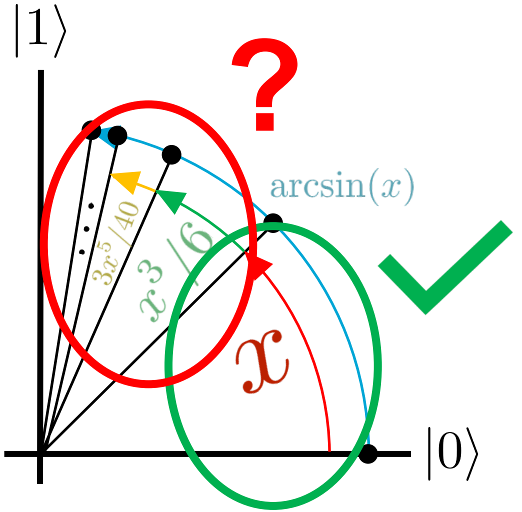
$$\arcsin(x) = x + \frac{1}{6}x^3 + \frac{3}{40}x^5 + \frac{5}{112}x^7 + \mathcal{O}\left(x^9\right)$$

?

&

$$R_y(a+b) = R_y(a)R_y(b)$$

- Reuse Cao's circuit
- Two high-order methods:
  - ~~Calculate higher-power, then rotate~~ Too many qubits!
  - Directly rotate over higher-power

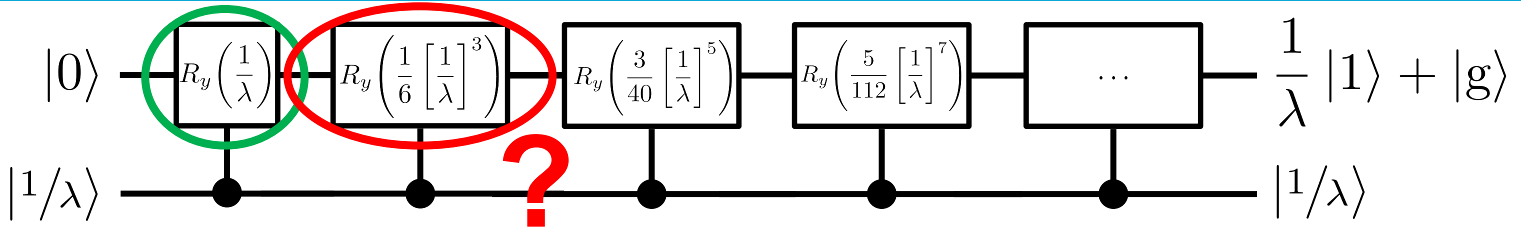


Direct higher-order rotation

$$\arcsin(x) = x + \frac{1}{6}x^3 + \frac{3}{40}x^5 + \frac{5}{112}x^7 + \mathcal{O}\left(x^9\right)$$

&

$$R_y(a+b) = R_y(a)R_y(b)$$



Remember:

e.g. $0.1101 = 0.1 + 0.01 + 0.0001$

$= a + b + c$

Too much detail

$$(a+b+c)^3 = a^3 + b^3 + c^3 + 3a^2b + 3a^2c + 3b^2c + \ldots$$

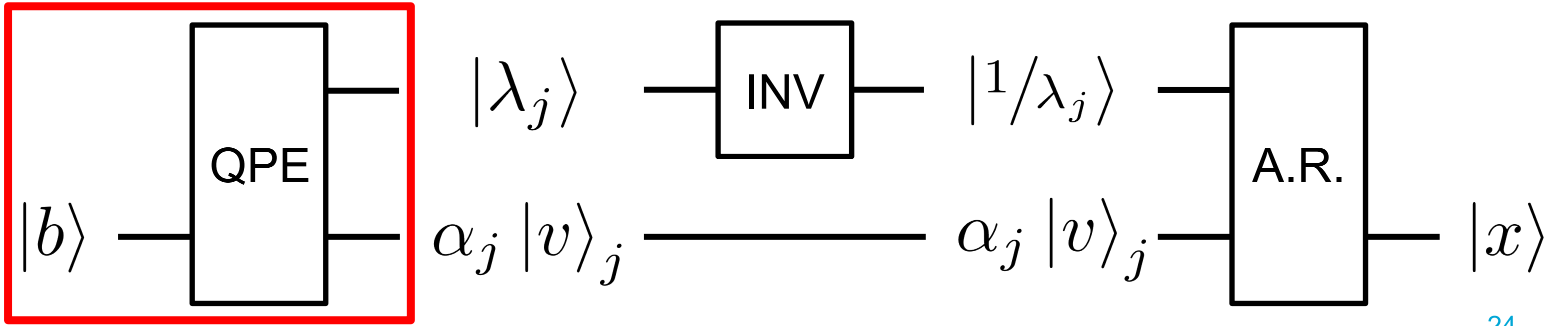
```
Ancilla rotation:

input c                = 1.1
input x                = 0.6875
input k                = 4

output sin(r)          = 0.755004

expectation sin(r)     = 0.7550042148744679
test c*x               = 0.7562500000000001

rel expectation error = 2.8460043640215567e-07
rel output error       = 0.0016476033057852301
```

$= 1/\lambda$

$=$ up to x^9

HLL algorithm in action

$$A = \frac{1}{4}\begin{bmatrix} 15 & 9 & 5 & -3 \\ 9 & 15 & 3 & -5 \\ 5 & 3 & 15 & -9 \\ -3 & -5 & -9 & 15 \end{bmatrix} \quad \vec{b} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \Longrightarrow \quad \vec{x} = \frac{1}{32}\begin{bmatrix} -1 \\ 7 \\ 11 \\ 13 \end{bmatrix}$$



TU Delft

HLL algorithm in action

$$\vec{x} = \frac{1}{32}\begin{bmatrix} -1 \\ 7 \\ 11 \\ 13 \end{bmatrix}$$

QUANTUM COMPUTER SIMULATOR

k	$x_{\text{rel},00}$	$x_{\text{rel},01}$	$x_{\text{rel},10}$	$x_{\text{rel},11}$
0	-1.00000	19.71925	31.69996	37.89948
1	-1.00000	8.81706	14.01056	16.61592
2	-1.00000	7.61822	12.02890	14.23496
3	-1.00000	7.25928	11.43197	13.51838
4	-1.00000	7.11956	11.19926	13.23915
5	-1.00000	7.05842	11.09734	13.11680
6	-1.00000	7.02964	11.04942	13.05929
7	-1.00000	7.01536	11.02557	13.03072
8	-1.00000	7.00796	11.01330	13.01595
9	-1.00000	7.00432	11.00718	13.00860

$=$ up to x^{19}! $< 0.1\%$ error!

Available online: https://github.com/0tmar/BEP_Quantum

TUDelft

Conclusion and outlook

- **Practical** implementation of HHL algorithm on QX simulator
- Generic routines for eigenvalue inversion and ancilla rotation

Ongoing and future work

- Generic vector implementation and Hamiltonian simulation
- Analysis for non-perfect qubits and real quantum hardware
- Integration into LibKet framework

Literature

- **Nielsen & Chuang 2000:** Nielsen, M., & Chuang, I. (2010). Frontmatter. In *Quantum Computation and Quantum Information: 10th Anniversary Edition* (pp. I-Viii). Cambridge: Cambridge University Press
- **HHL 2009:** Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. Physical Review Letters, 103(15):1–15, nov 2009.
- **Cao 2012:** Yudong Cao, Anmer Daskin, Steven Frankel, and Sabre Kais. Quantum Circuit Design for Solving Linear Systems of Equations. Molecular Physics, 110(15-16):1675–1680, oct 2013.
- **Thapliyal 2017:** H. Thapliyal, T. S. S. Varun, and E. Munoz-Coreas, “Quantum Circuit Design of Integer Division Optimizing Ancillary Qubits and T-Count,” arXiv:1609.01241.

TU Delft