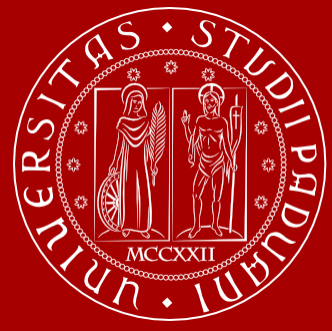


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Two-qubit CZ gate implementation with trapped neutral atoms: a numerical simulation

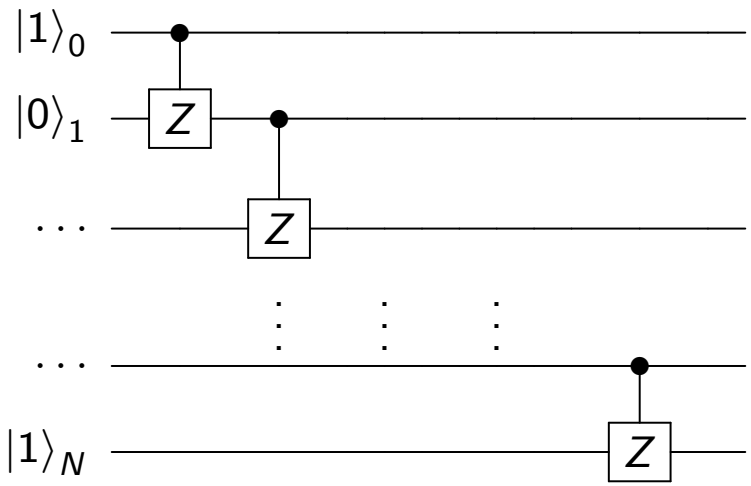
Alice Pagano	**Mat. 1236916**	alice.pagano@studenti.unipd.it
Michele Puppin	**Mat. 1227474**	michele.puppin@studenti.unipd.it

Quantum Information and Computing
(a.y. 2020/21)

22 March 2021

Chain of N-qubits
Numerical simulation

- Numerical simulation of a **chain of N-qubit**
- The **CZ gate** is applied between consequent qubits, assuming **perfect blockade regime**
- One-dimensional array of atoms $\rightarrow$ each time the **two-qubit CZ gate** acts on a **pair of atoms**, the interaction with the other atoms is neglected



- Obtain the **behavior** of the map in Eq. between the **first** and **last qubits**

$$|00\rangle \rightarrow |00\rangle$$
$$|01\rangle \rightarrow |01\rangle\, e^{i\phi}$$
$$|10\rangle \rightarrow |10\rangle\, e^{i\phi}$$
$$|11\rangle \rightarrow |11\rangle\, e^{i(2\phi-\pi)}$$

- **First** and **last qubits** can be **initialized** either in state $|0\rangle$ or $|1\rangle$
- All the other qubits in the chain are **initialized** in $|0\rangle$
- The state vector of the N-**qubit** system is the **tensor product** of the **subsystems** as:

$$|\psi\rangle = |q_1\rangle \otimes \cdots \otimes |q_N\rangle$$

- Starting from the first qubit, we **apply** the two-qubit CZ gate **iteratively**.
- In the i-th iteration, the total **Hamiltonian** of the chain H_{chain} is:

$$H^{(N)}_{chain,i} = \underbrace{\mathbb{I}_3 \otimes \cdots \otimes \mathbb{I}_3}_{i-1} \otimes H_{CZ} \otimes \underbrace{\mathbb{I}_3 \otimes \cdots \otimes \mathbb{I}_3}_{N-i-1}.$$

- Hamiltonian of **dimension** $3^N \times 3^N$

- As **longer chains** are considered, the dimension of the total Hamiltonian of the system **scales** as $3^N \times 3^N$
- Given the Hamiltonian H of the system, the **time-dependent Schrödinger equation** can be solved as:

$$|\psi(t)\rangle = U(t)\,|\psi(0)\rangle\,, \qquad U(t) = e^{-iHt/\hbar}$$

- The **time evolution** of the system can become **computationally challenging** for large N.
- We consider several time-dependent Schrödinger **equation solvers**, along with some optimizations, in order to test their performances
- We implement these algorithms using **different coding environment** and **libraries implementation** both on CPU and GPU

NumPy implementation

- Support for large, multi-dimensional arrays
- Collection of high-level mathematical functions
- Coded in well-optimized C code
- SciPy provides sparse array libraries
- Plays well with parallel computing

TensorFlow implementation

- Optimized to deal with large matrices
- Can be easily executed on GPU thanks to CUDA

Fortran implementation

- One of the most powerful language for scientific computation with LAPACK routines for linear algebra
- Its design allows the compiler to perform stronger optimizations

- We compute the **unitary time evolution** of the system as

$$|\psi(t)\rangle = U(t)\,|\psi(0)\rangle\,, \qquad U(t) = e^{-iHt/\hbar}$$

- The system Hamiltonian can be diagonalized $H = PDP^{-1}$, where D is the diagonal matrix and P is the eigenvectors matrix.
- The exponential matrix can be computed as:

$$e^{-iHt} = Pe^{-iDt}P^{-1}$$

- **NumPy** $\rightarrow$ matrix diagonalization with `numpy.linalg.eigh`, matrix inversion with `numpy.linalg.inv`
- **TensorFlow** $\rightarrow$ matrix exponential with `tensorflow.linalg.expm`
- **Fortran** $\rightarrow$ matrix diagonalization with `_heev`, matrix inversion with `_getrf` and `_getri` LAPACK routines

- Given a discrete time step Δt, solving time-dependent Schrödinger equation is equivalent to solve the linear system:

$$\left(1 + \frac{iH\Delta t}{2}\right)\psi(x, t + \Delta t) = \left(1 - \frac{iH\Delta t}{2}\right)\psi(x, t).$$

- By iterating the procedure n_{iter} times, we obtain the time evolution for time $T = \Delta t \times n_{iter}$

- **NumPy** $\rightarrow$ linear system solved with `numpy.linalg.solve`
- **TensorFlow** $\rightarrow$ analogous resolution using `tensorflow.linalg` module

- An invertible matrix A can be decomposed into two factors LU, where L is a lower and a U an upper triangular matrix.
- The linear system $A\boldsymbol{x} = \boldsymbol{y}$ can be recasted as:

$$\begin{cases} L\boldsymbol{z} = \boldsymbol{y} \\ U\boldsymbol{x} = \boldsymbol{z} \end{cases}$$

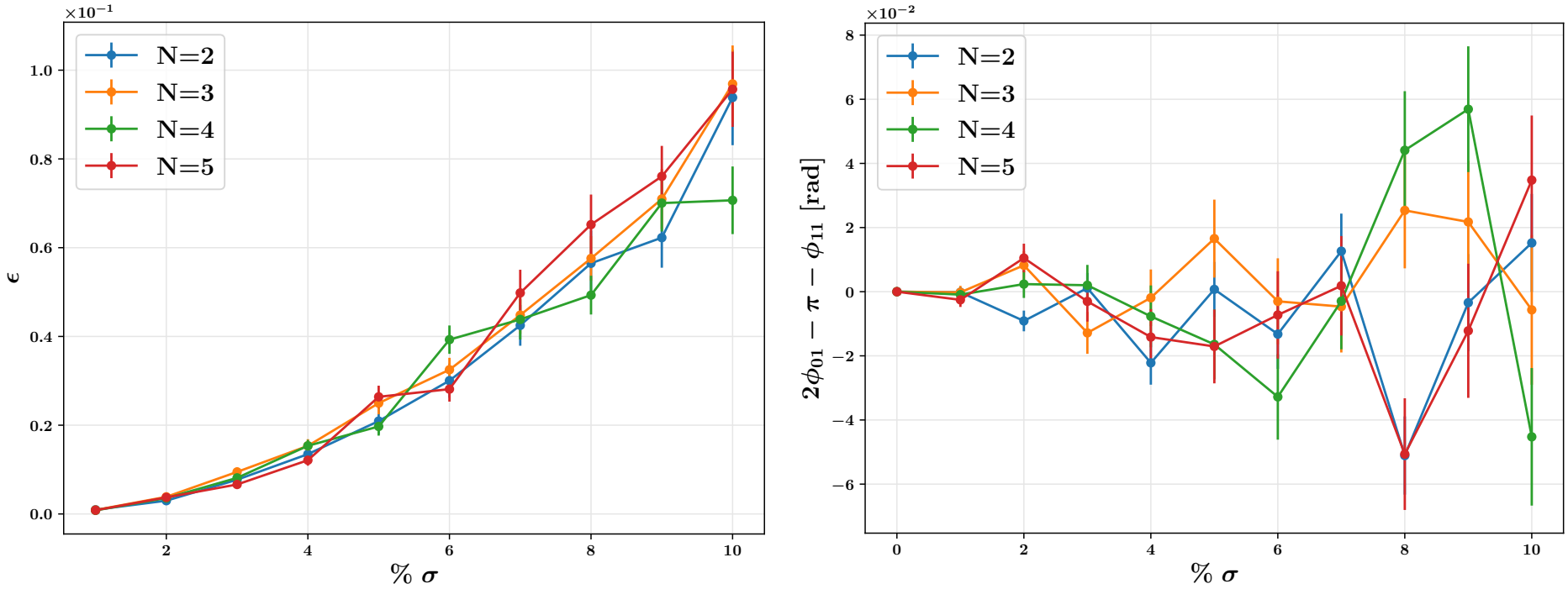
- In our case the matrix A is fixed and the system has to be solved many times for different $\boldsymbol{x}$ and $\boldsymbol{y}$

- **NumPy** $\rightarrow$ compute the LU decomposition with `scipy.linalg.lu` function and solve the systems with the `scipy.linalg.solve_triangular` function
- **TensorFlow** $\rightarrow$ analogous resolution using `tensorflow.linalg` module

- The total Hamiltonian of the N-qubits system is a *sparse matrix*
- **NumPy** $\rightarrow$ matrices are transformed into `csc_matrix`
- optimized functions are used for LU decomposition and triangular systems solving

Noise effects

Gaussian noise on $\Omega\tau$ and Δ/Ω



(a) $\Omega\tau$ (b) Δ/Ω

Figure: Noise effects on the CZ gate implementation in a N-qubit chain. A Gaussian noise with zero mean and standard deviation σ is introduced on $\Omega\tau$ and Δ/Ω. In particular, $\%\,\sigma$ refers to the value of the standard deviation as a percentage of the optimal parameters. The error ϵ for the state $|11\rangle$ and the phase difference are computed as the mean of 100 iterations.

Timing analysis
Number of iterations

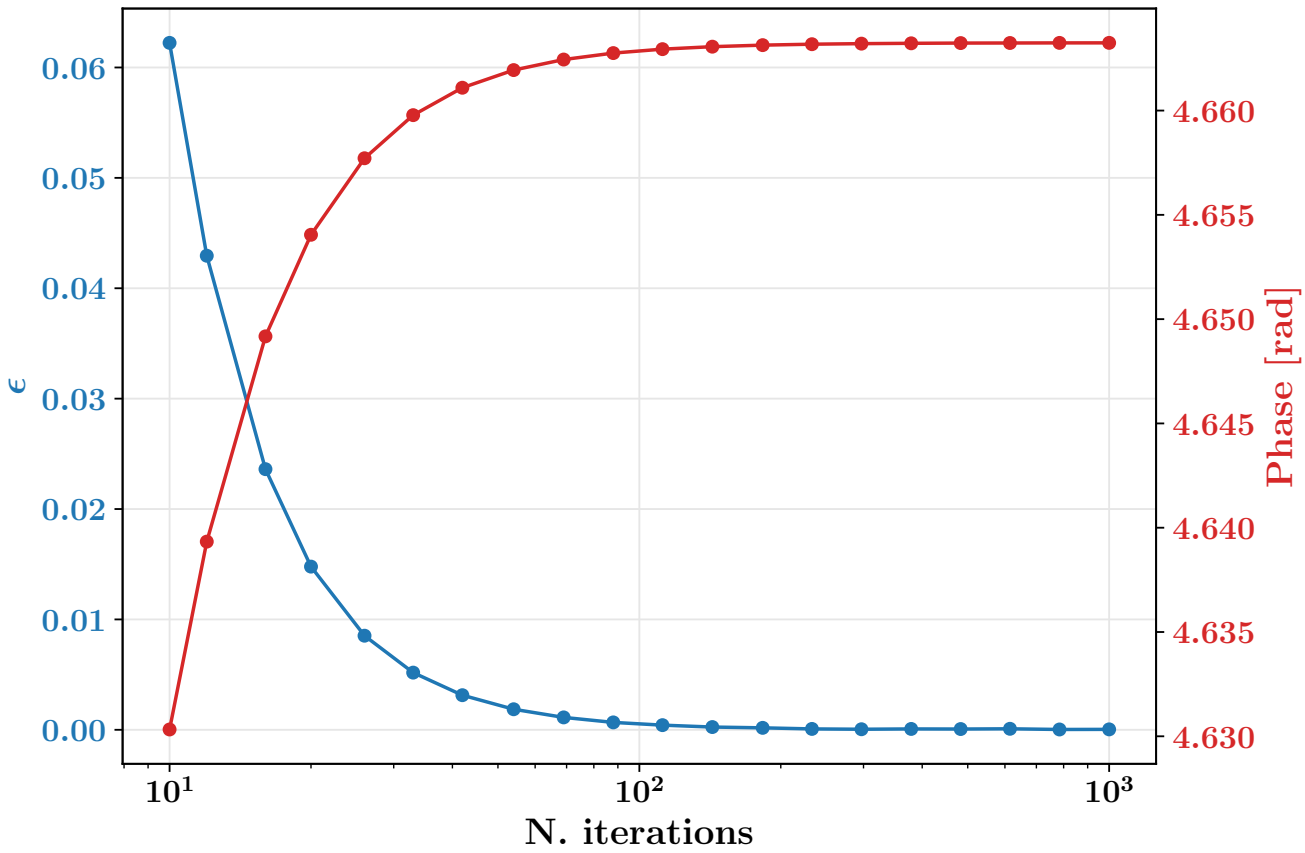


Figure: Phase and error of state $|11\rangle$ as a function of the number of iterations for two-qubits CZ gate. We consider NumPy implementation with Crank-Nicolson with LU decomposition method.

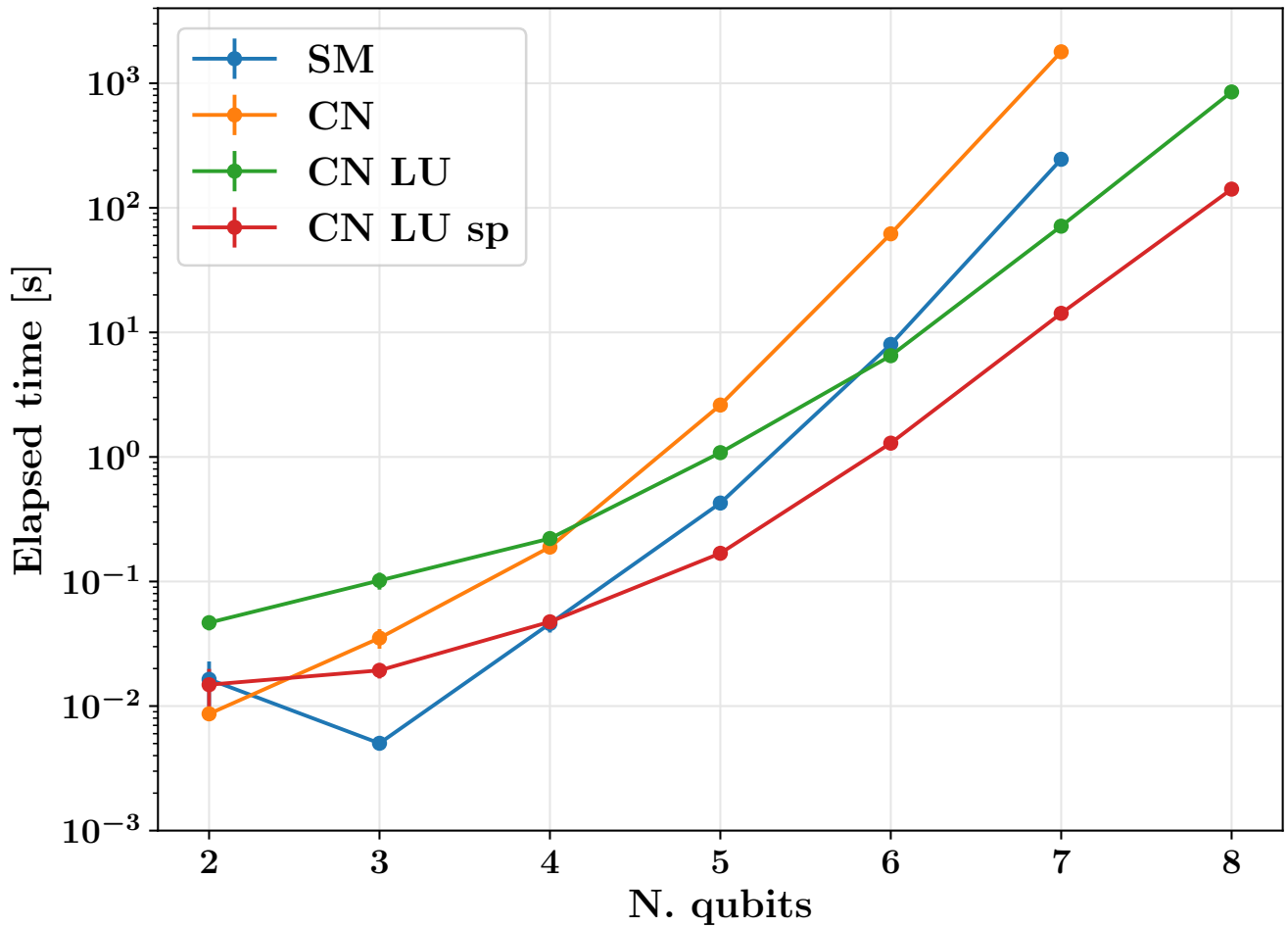
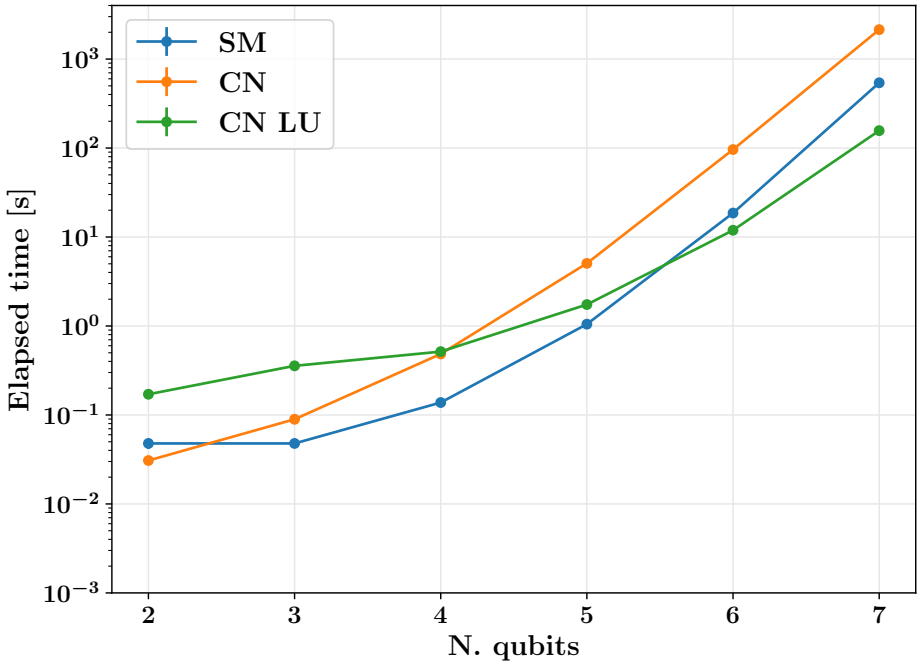
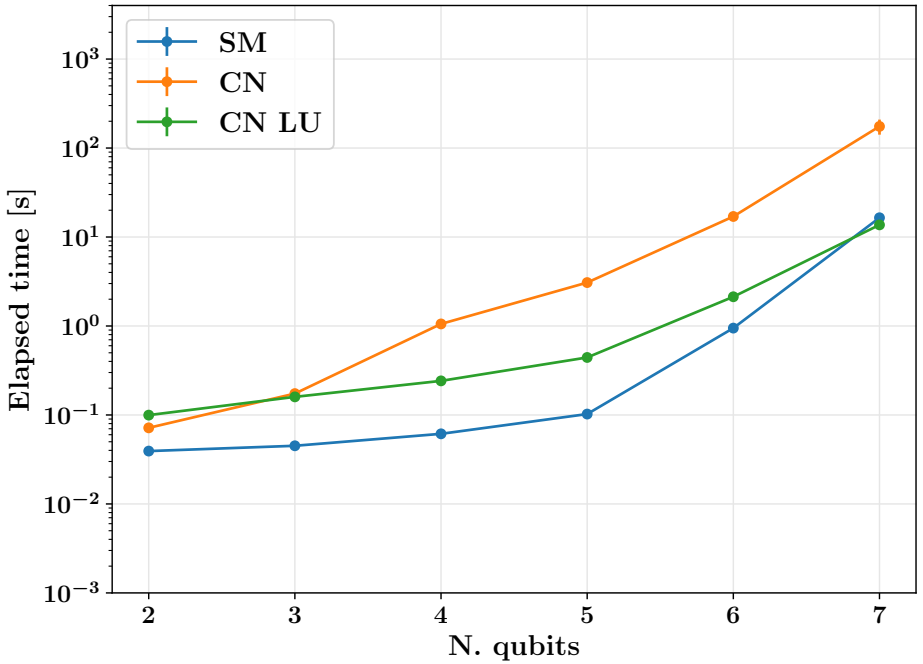


Figure: Mean elapsed time in seconds as a function of the number of qubits in the chain. Spectral Method (SM), Crank-Nicolson method (CN), Crank-Nicolson with LU decomposition method (CN LU) and Crank-Nicolson with LU decomposition with sparse matrix method (CN LU sp) are compared.

Timing analysis
TensorFlow CPU and GPU



(a) TensorFlow CPU



(b) TensorFlow GPU

Figure: Mean elapsed time in seconds as a function of the number of qubits in the chain. Spectral Method (SM), Crank-Nicolson method (CN) and Crank-Nicolson with LU decomposition method (CN LU) are compared.

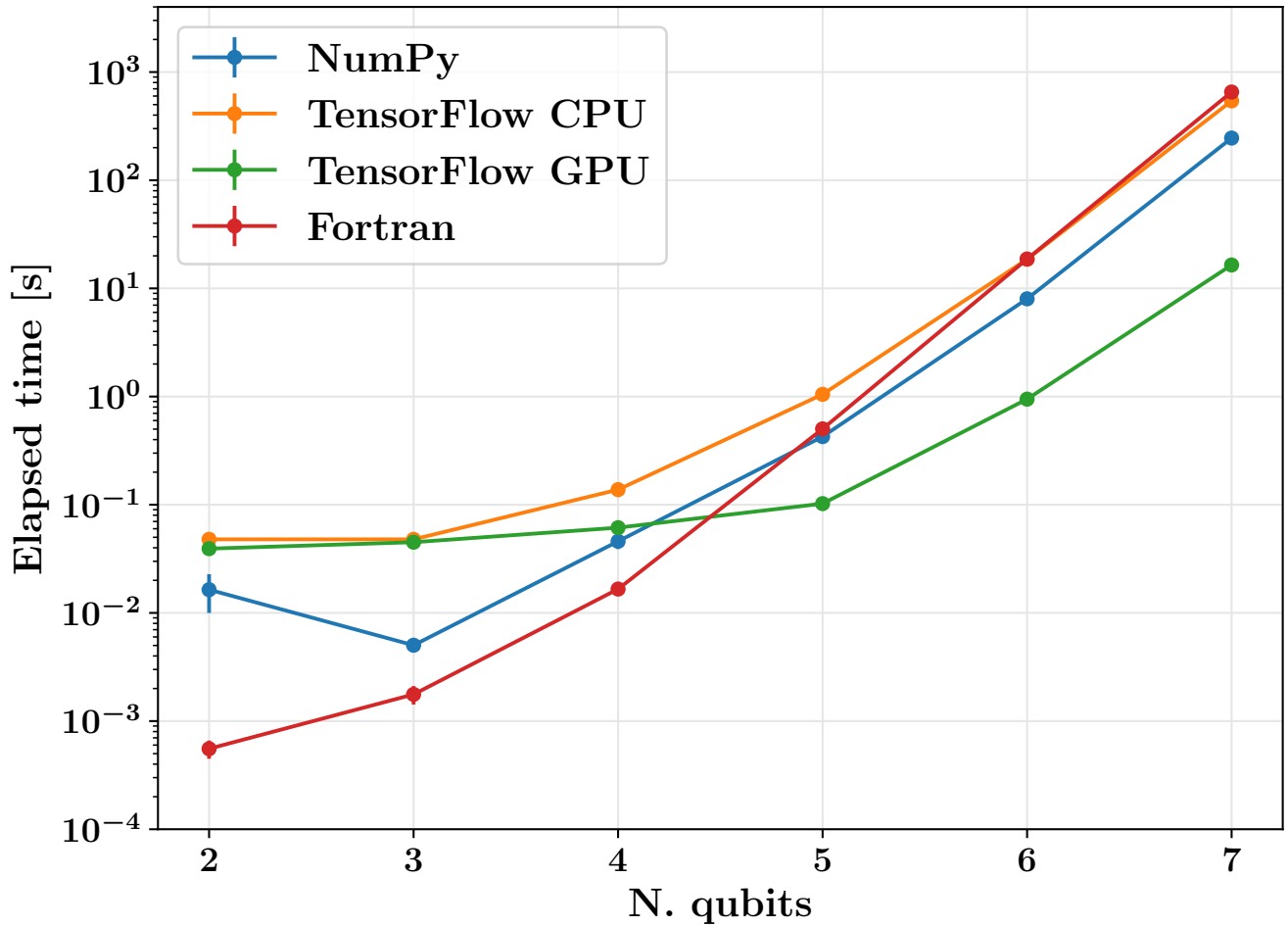


Figure: Mean elapsed time in seconds as a function of the number of qubits in the chain.

Conclusions

- **Gaussian noise** is introduced to perturb separately the parameter $\Omega\tau$ and $\Delta/\Omega \rightarrow$ no significant differences as a function of the standard deviation of the normal noise for different number of qubits.
- **Timing analysis** $\rightarrow$ for a lower number of qubits the Spectral Method have good performances but it is not feasible for large number of qubits. Approximation are needed in order to handle such big matrices $\rightarrow$ Crank-Nicolson

Thank you for the attention!