

# Quantum Circuits and Optimization

---

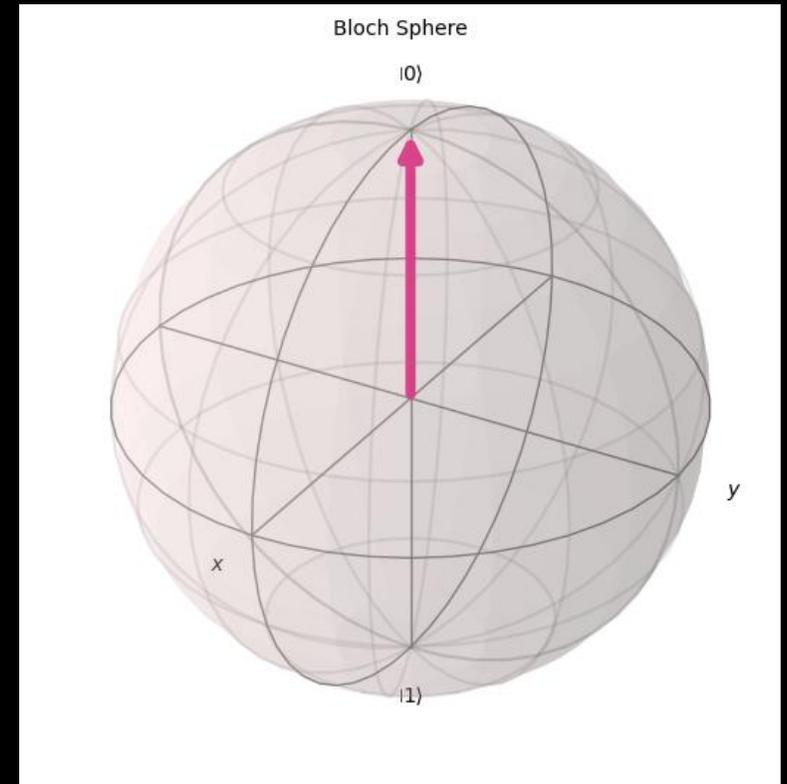
Scaramella, Schillirò, Farsian,  
Sarracino, 11/02/2026



# What is Quantum Computing?

---

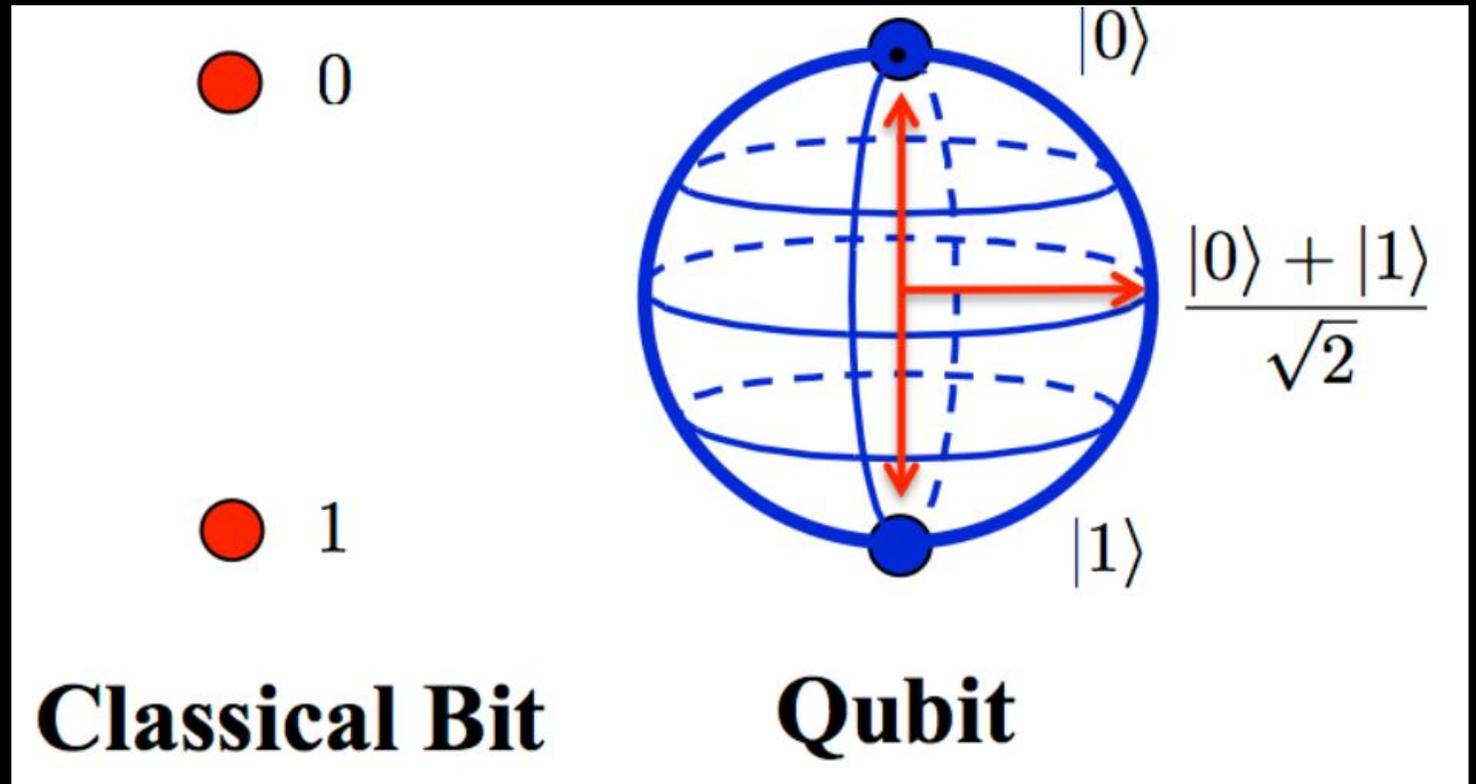
- 1) Quantum Computing Is a novel Scientific Discipline which uses concepts of Quantum Mechanics and Computer Science.
- 2) The fundamental information unit is the Qubit, that can exist as a superposition of states as long as it is not classically measured.
- 3) Quantum Algorithms leverage these objects to perform operations using properties of Quantum Mechanics, looking for advantages on the classical counterparts.
- 4) Quantum Computers have to be built with the idea of being able to manifest these operations and manipulate Qubits.



# The Qubit

The Fundamental Concepts used to build the Qubit:

- 1) Entanglement.
- 2) Superposition.





Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.

— *Richard P. Feynman* —

**AZ** QUOTES

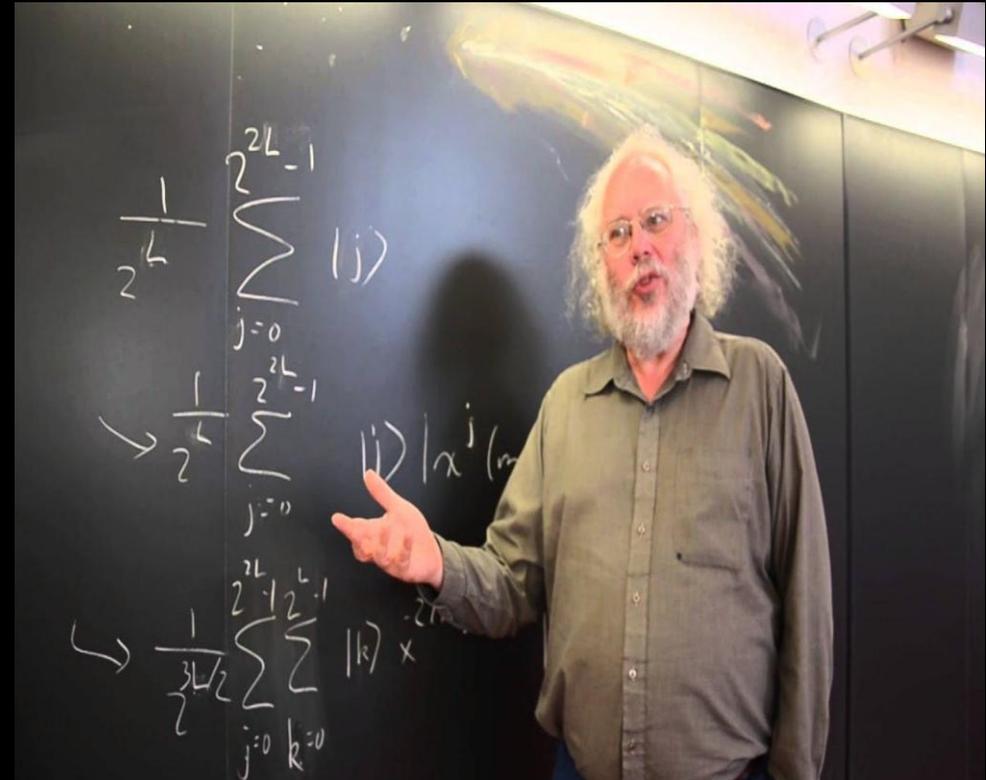
# Quantum History

---

After the first idea from Feynman (1982), theoretically, Quantum Computing has been extensively studied since the 80s.

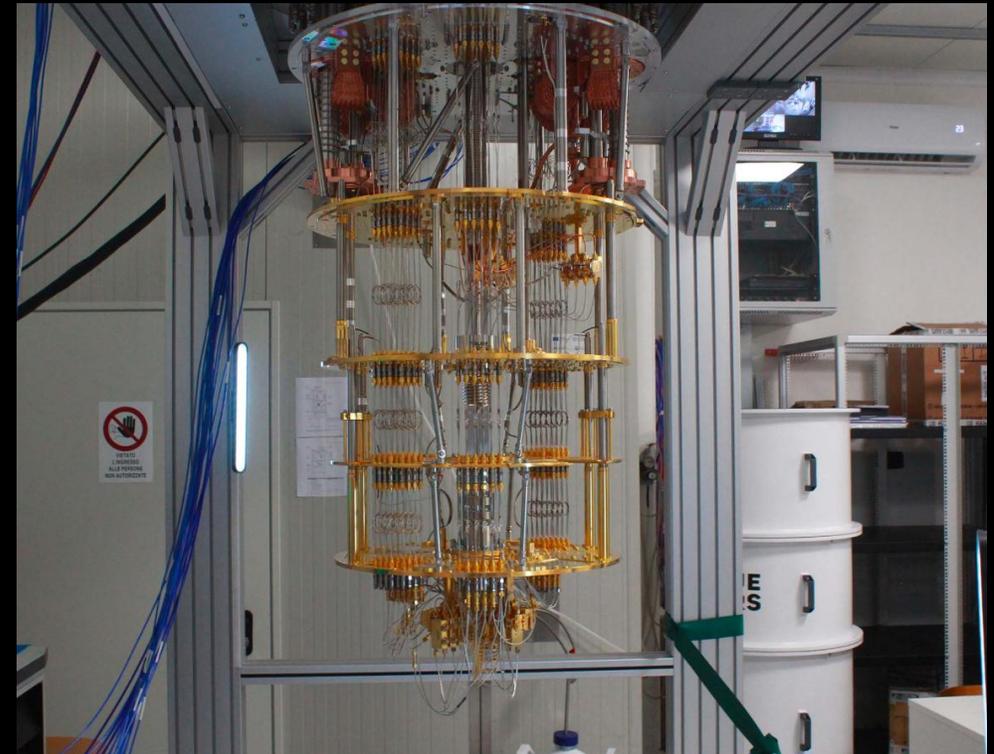
Two very famous Quantum Algorithms are Shor's (1994) for factorization of numbers, and Grover's (1996) for quantum search in a database.

These two algorithms present an exponential and polynomial speed-up with respect to classical counterparts, respectively.



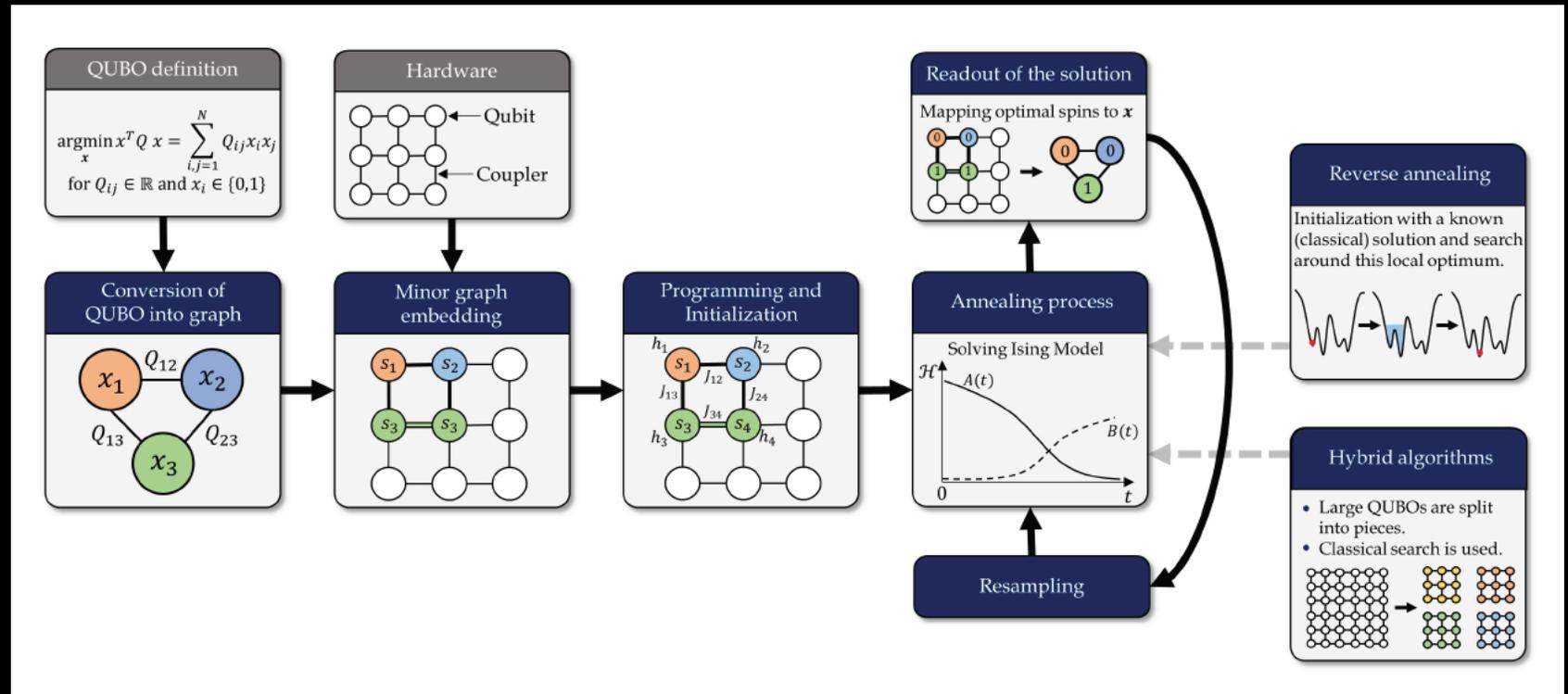
# Quantum Hardware

---



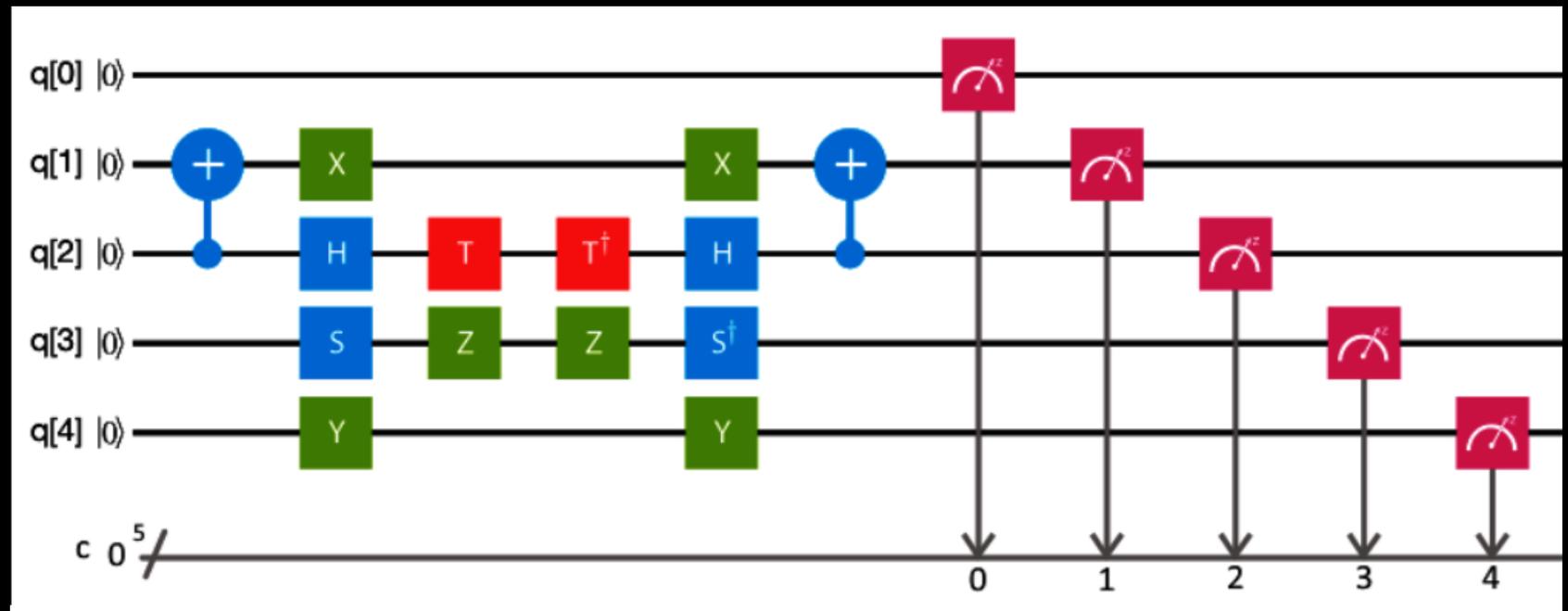
# Quantum Annealing

- 1) Very High number of «Qubits».
- 2) Scope limited to the minimization of Ising-like Hamiltonians
- 3) Very Stable and with low errors, but less user-friendly and sometimes acts as a black box.

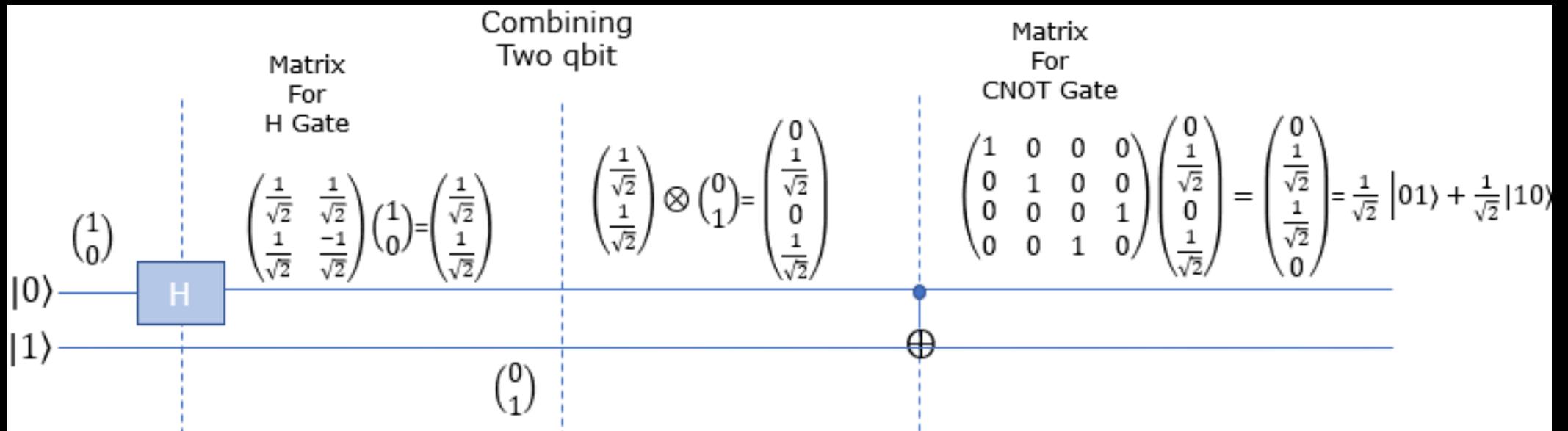


# Quantum Circuits

- 1) Limited number of Qubits.
- 2) Very general scope.
- 3) Less stable, and the errors can become problematic with deep circuits
- 4) User-friendly and clear.

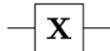
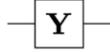
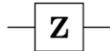
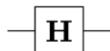
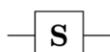
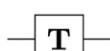
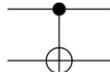
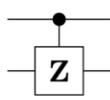
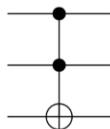


# Quantum Gates: Hadamard and CNOT



# Quantum Gates: Glossary

A Quantum Circuit is a combination of these gates which act on a given number of qubits. Then, a classical measurement is performed, which collapse the Qubits into classical configurations.

Operator	Gate(s)	Matrix
Pauli-X (X)	 	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

# Quantum applications

---

## TOP 9 QUANTUM COMPUTING APPLICATIONS

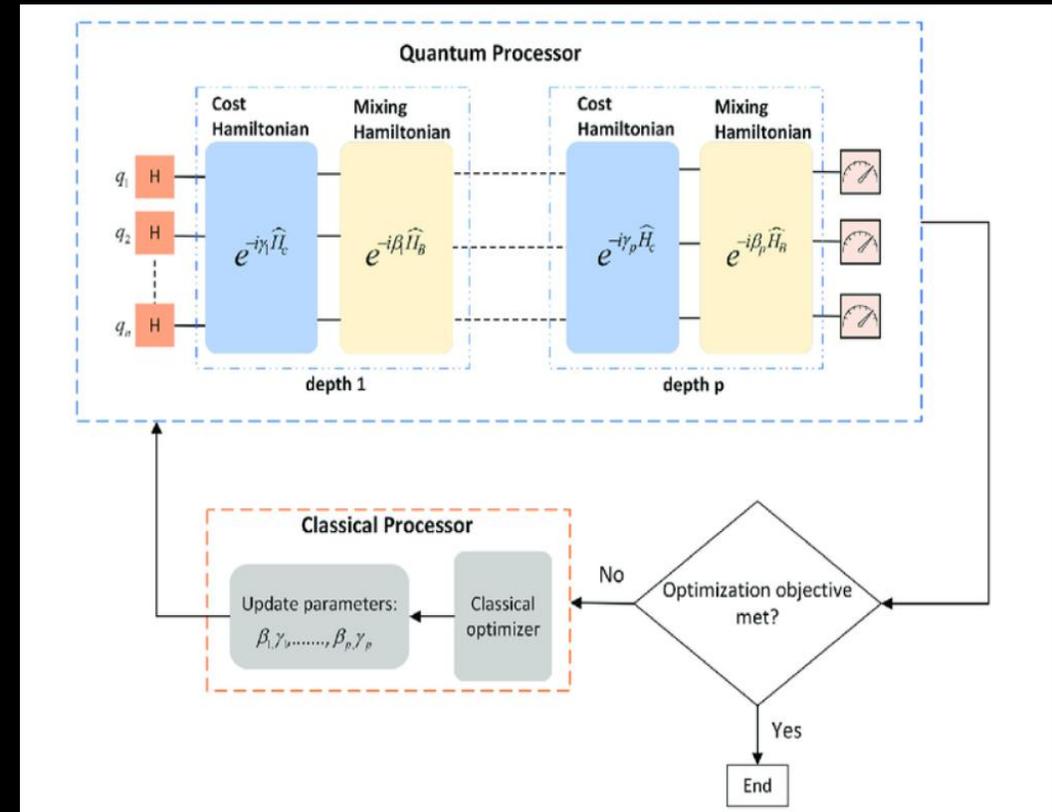
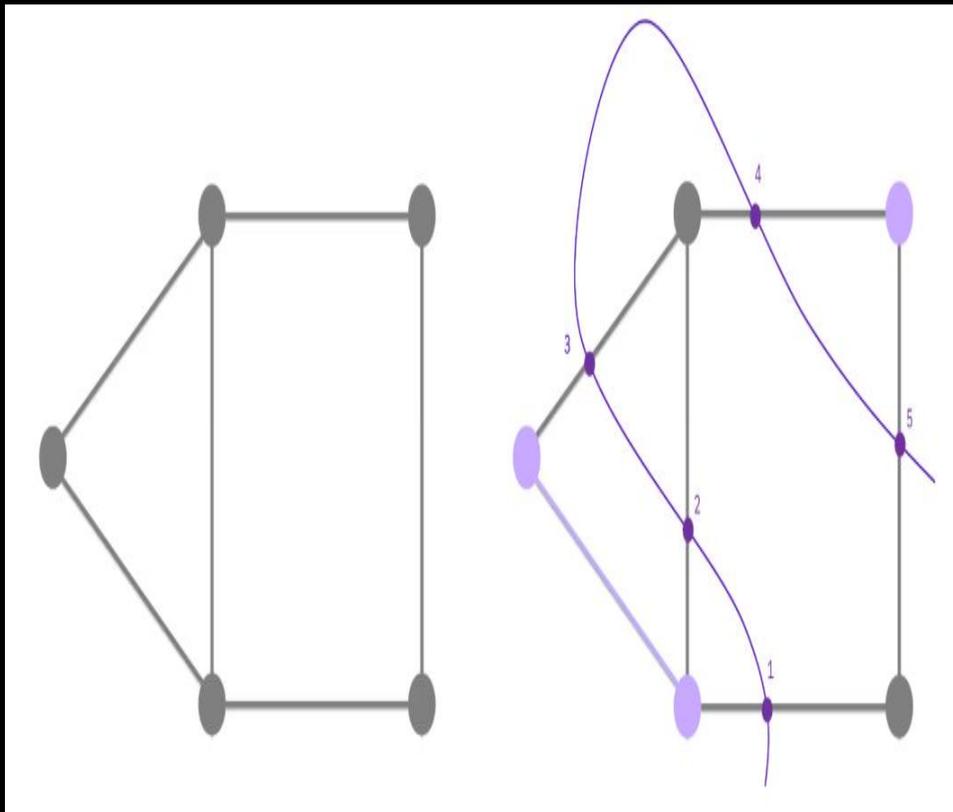


1. Drug discovery
2. Cybersecurity
3. Cryptography
4. Financial modeling and calculations
5. Material science
6. Artificial intelligence and machine learning
7. Manufacturing
8. Logistics
9. Portfolio management

pixelplex

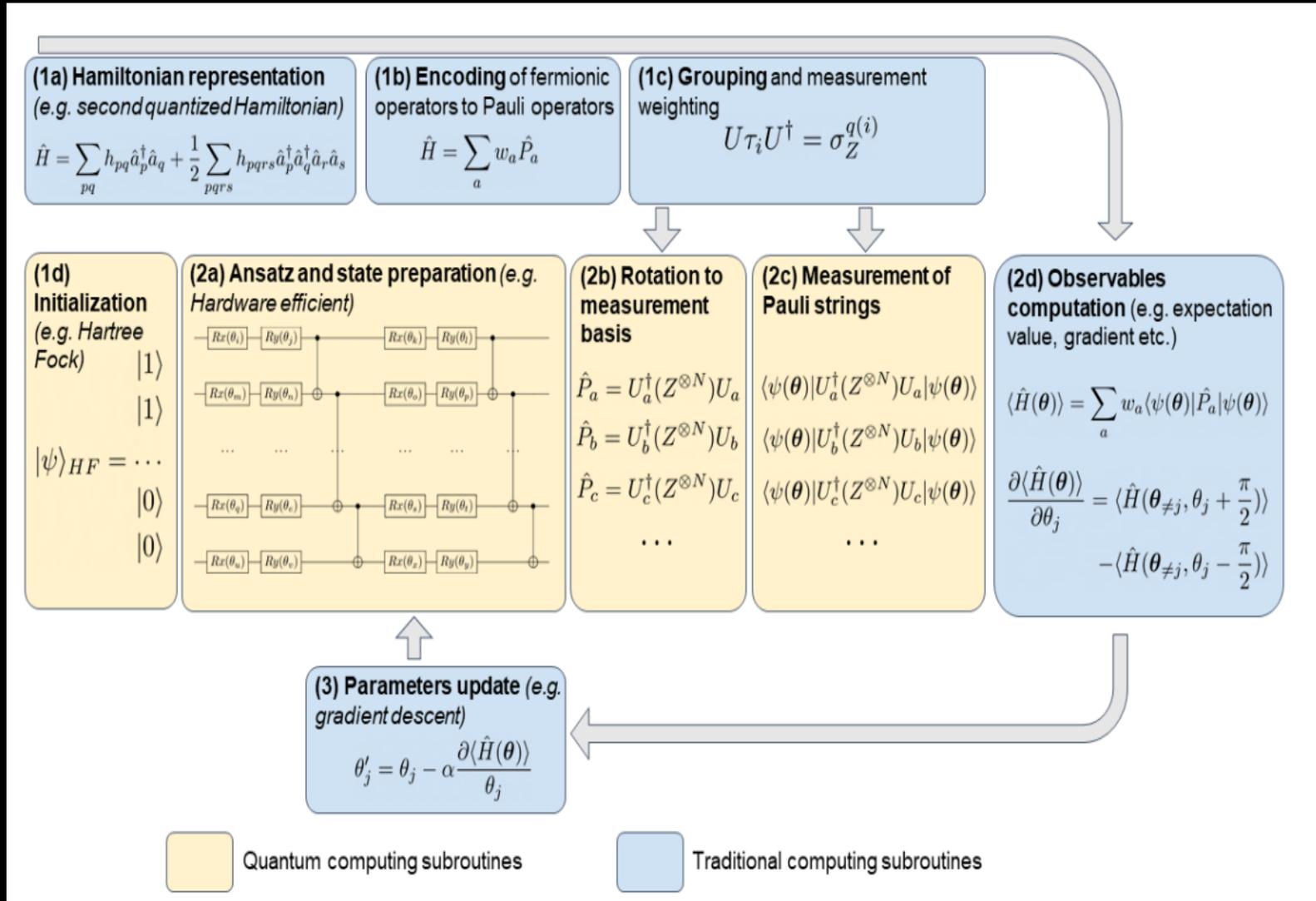
Source:  
<https://pixelplex.io/blog/quantum-computing-applications/>

# Optimization using Quantum Circuits: QAOA, Hybrid quantum-classical framework



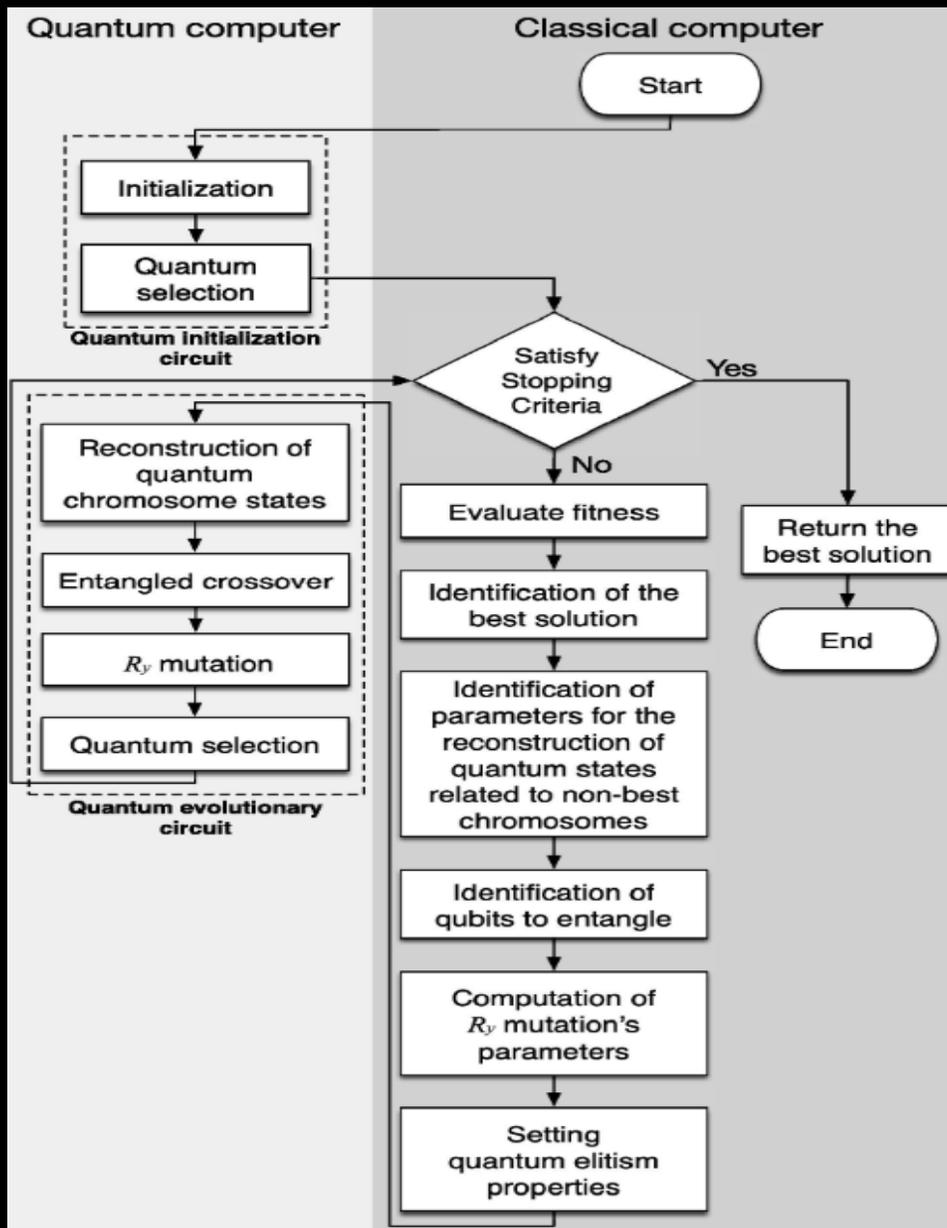
<https://quantum.cloud.ibm.com/docs/it/tutorials/quantum-approximate-optimization-algorithm>

# Variational Quantum Eigensolver (VQE)



<https://arxiv.org/pdf/2111.05176>

<https://quantum.cloud.ibm.com/learning/it/courses/quantum-diagonalization-algorithms/vqe>



# Quantum Genetic algorithms: another hybrid approach

<https://www.sciencedirect.com/science/article/pii/S002002552100640X> (Acampora and Vitiello 2021)

# Quantum Genetic Algorithm: Workflow

---

Merit function Evaluation (Classical), evaluation of the chi-squared for the cosmological functions to find the minimization parameters

---

---

Individual Selection and Repopulation (Classical, duplication of the selected individuals)

---

---

2 quantum circuits, one for the duplicated best data and the other for the rest of the population (random), while keeping the best individuals. Quantum Encoding

---

---

Quantum Superposition (already implemented in the encoding)

---

---

Quantum Crossover + Mutation

---

---

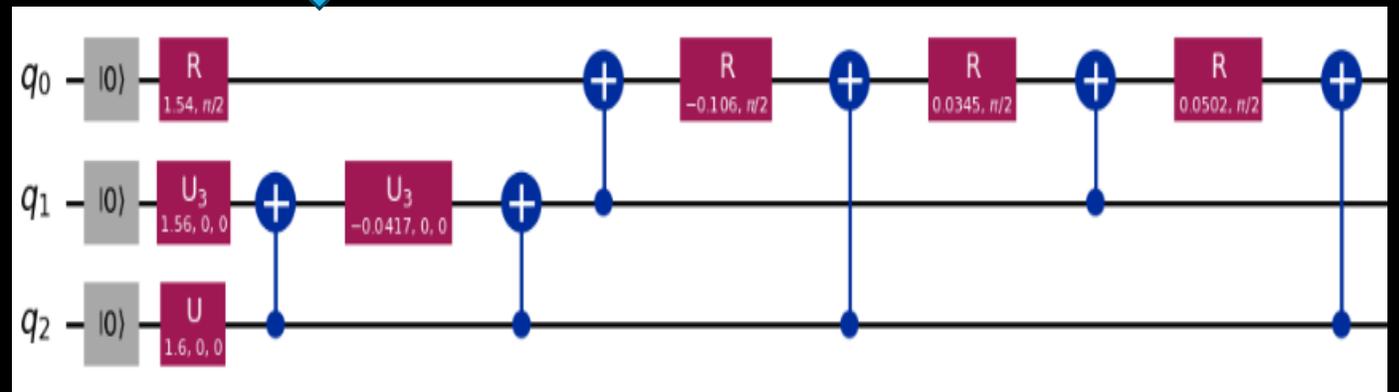
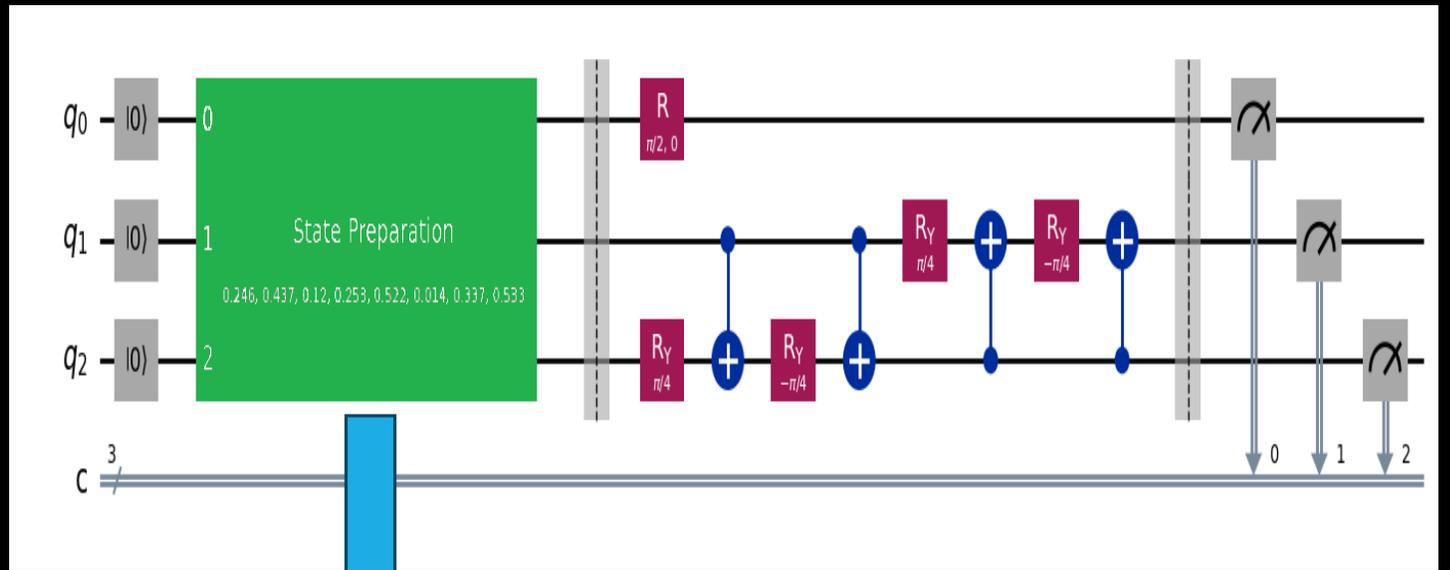
Quantum Decoding

---

# Our Quantum Circuit:

$$U_{\text{cross}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

$$U_{\text{mut}} = R_x\left(\frac{\pi}{2}\right) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$$



# Some more details on our Quantum Genetic Algorithm

---

## Algorithm 1 Amplitude-Encoded Quantum Genetic Algorithm

---

- 1: **Input:** Population size  $n_p$ , number of generations  $n_g$ , crossover probability  $p_c$ , mutation probability  $p_m$
  - 2: Initialize population  $P$  with  $n_p$  individuals
  - 3: **for**  $t = 1$  to  $n_g$  **do**
  - 4:     Evaluate fitness of each individual in  $P$
  - 5:     Select top individuals  $P_{\text{elite}} \subset P$  with size  $n_p/4$ .
  - 6:     Duplication of the top individuals  $P_{\text{elite copy}}$ .
  - 7:     Generate random population  $P_{\text{rand}}$  so that  $P_{\text{rand}} + P_{\text{elite}} + P_{\text{elite copy}} = P$ .
  - 8:     Quantum encode  $P_{\text{elite copy}}$  and  $P_{\text{rand}}$  into two different circuits
  - 9:     **for** each dimension  $d$  **do**
  - 10:         Apply quantum crossover with probability  $p_c$
  - 11:         Apply quantum mutation with probability  $p_m$
  - 12:         Measure circuit to decode updated individuals (two different decoding schemes for  $P_{\text{rand}}$  and  $P_{\text{elite copy}}$ )
  - 13:     **end for**
  - 14:     Combine:  $P \leftarrow P_{\text{elite}} \cup P_{\text{decoded}}$
  - 15: **end for**
  - 16: **Return:** Best individual in final population
- 

---

## Algorithm 2 AEQGA Decoding from Quantum Measurements

---

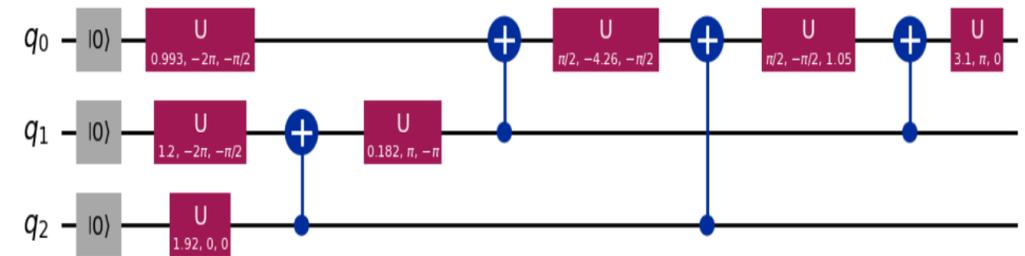
- 1: **Input:** Quantum circuit  $C$  (with measurements), number of qubits  $n_q$ , decoding flag  $\text{flag} \in \{\text{best}, \text{random}\}$ , interval  $[n_{\min}, n_{\max}]$ , shots  $S$
  - 2: Initialize simulator with seed  $s$  and run  $C$  for  $S$  shots to obtain counts
  - 3: **if**  $\text{flag} = \text{best}$  **then**     ▶ Restricted box around elites; probability-based decoding
  - 4:      $C \leftarrow \sum_i c_i$      ▶ total measurements
  - 5:      $p_i \leftarrow c_i / C$
  - 6:      $x_i \leftarrow n_{\min} + (n_{\max} - n_{\min}) \cdot \sqrt{p_i}$      ▶  $\sqrt{p_i}$  focuses samples toward box center
  - 7: **else**     ▶ random: full-range min-max decoding
  - 8:      $c_{\min} \leftarrow \min_i c_i$ ,      $c_{\max} \leftarrow \max_i c_i$
  - 9:      $x_i \leftarrow n_{\min} + (n_{\max} - n_{\min}) \cdot \frac{c_i - c_{\min}}{c_{\max} - c_{\min}}$
  - 10: **end if**
  - 11: **(Post-process):** if any  $x_i = n_{\min}$  exactly, replace it with a draw from  $\mathcal{U}(n_{\min}, n_{\max})$
  - 12: **Output:** Decoded vector  $\mathbf{x}$  of the classical individuals for the populations
-

# The Quantum Genetic Algorithm Step by step on qiskit: Quantum Encoding

```
def Quantum_Encoding(population, n_qubit, dimensions):  
    i=0  
    j=0  
    norm_vectors=np.array([])  
    population_norm=np.array([])  
    while i<dimensions:  
        norm_constant = np.linalg.norm(population[i])  
        pop_norm=population[i]/norm_constant  
        norm_vectors=np.append(norm_vectors,norm_constant)  
        population_norm=np.append(population_norm, pop_norm)  
        i=i+1  
    population_norm=population_norm.reshape(dimensions,2**n_qubit)  
  
    quantum_circuits = []  
    while j<dimensions:  
        circ_encoding=QuantumCircuit(n_qubit)  
        circ_encoding.initialize(population_norm[j])  
        quantum_circuits.append(circ_encoding)  
        j=j+1  
    return quantum_circuits  
Initial_Quantum_pops=Quantum_Encoding(initial_population, n_qubit, dimensions)  
  
Initial_Quantum_pops2=Quantum_Encoding(initial_population, n_qubit, dimensions)  
  
for i, qc in enumerate(Initial_Quantum_pops2):  
    print(f"Circuit {i+1}:")  
    display(qc.decompose(reps = 7).draw('mpl'))  
    plt.show()
```

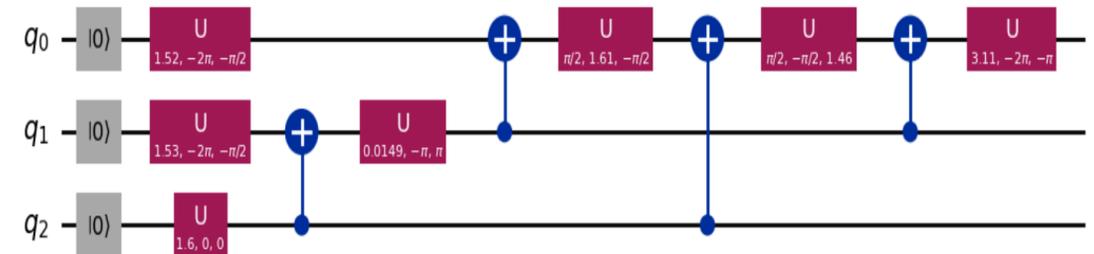
Circuit 1:

Global Phase: 3.177160287378107



Circuit 2:

Global Phase: 0.035940412735186555



# Quantum Genetic Algorithm: Quantum Crossover and mutation

```
def quantum_crossover(quantum_pop, n_qubit, crossover_probability, flag="new"):
    quantum_pop.barrier()
    if flag=="old":
        for q in range(n_qubit):
            if np.random.rand() < crossover_probability:
                qubit2=(q + 1) % n_qubit
                quantum_pop.swap(q, qubit2)

    elif flag=="new":
        if np.random.rand() < crossover_probability:
            q=random.randint(0, n_qubit-2)
            #theta = np.random.uniform(np.pi/8, np.pi)
            quantum_pop.cry(np.pi/2, q, q+1)
            quantum_pop.cry(np.pi/2, q+1, q)

    return quantum_pop
```

```
def quantum_mutation(quantum_pop, n_qubit, mutation_probability, flag="one"):
    #np.random.seed(seed_quantum)
    if flag=="all":
        for q in range(n_qubit):
            if np.random.rand() < mutation_probability:
                # Randomly select a gate
                gate_choice = np.random.choice(['rx', 'ry', 'rz'])

                if gate_choice == 'rx':
                    theta = np.random.uniform(0, np.pi)
                    quantum_pop.rx(theta, q)
                elif gate_choice == 'ry':
                    theta = np.random.uniform(0, np.pi)
                    quantum_pop.ry(theta, q)
                elif gate_choice == 'rz':
                    theta = np.random.uniform(0, np.pi)
                    quantum_pop.rz(theta, q)

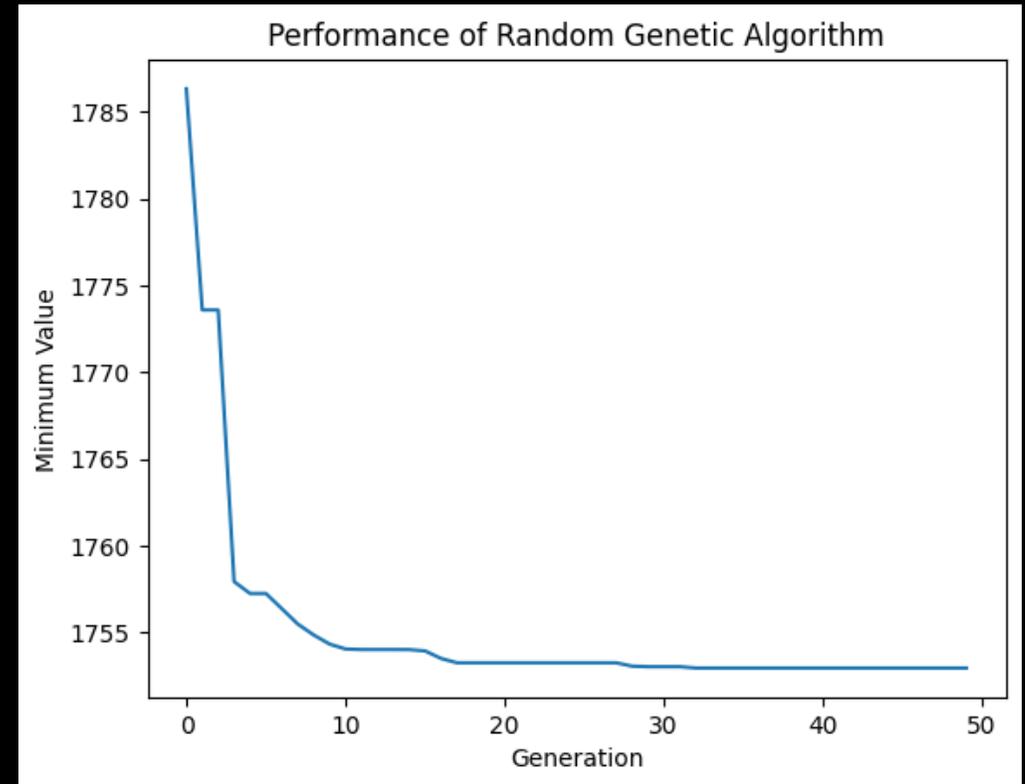
    elif flag=="one":
        if np.random.rand() < mutation_probability:
            q=random.randint(0, n_qubit-1)
            #theta = np.random.uniform(np.pi/4, np.pi)
            quantum_pop.rx(np.pi/2, q)

    quantum_pop.barrier()
    quantum_pop.measure([k for k in range(n_qubit)], [k for k in range(n_qubit)])
    return quantum_pop
```

# Quantum Genetic Algorithm: Quantum Decoding

```
def quantum_decoding(quantum_pop,n_qubit,n_min, n_max, flag, seed,shots = 500*2**n_qubit):
    aersim = AerSimulator(seed_simulator=seed)
    quantum_popt = transpile(quantum_pop, aersim)
    result = aersim.run(quantum_popt, shots=shots).result()
    counts = result.get_counts(quantum_popt)
    if not counts:
        raise ValueError("No counts found. Check the quantum circuit setup.")
    all_outcomes = [format(i, '0' + str(n_qubit) + 'b') for i in range(2**n_qubit)]
    if flag=="random":
        counts_list = [counts.get(outcome, 0) for outcome in all_outcomes]
        norm = np.linalg.norm(counts_list)
        normalized_counts = counts_list / norm
        population = n_min + (n_max - n_min) * (normalized_counts - np.min(normalized_counts)) /
            (np.max(normalized_counts) - np.min(normalized_counts))
    if flag=="best":
        total_measurements = sum(counts.values())
        percentages = [counts.get(outcome, 0) / total_measurements for outcome in all_outcomes]
        population = [n_min + (n_max - n_min) * (percentage**0.5) for percentage in percentages]

    return population
```



# Back up Slides

---

# Talk layout

---

- What is quantum computing?
- Cosmological parameters: probes and open issues for modern cosmology.
- Overview of the problem we aim to solve: Quantum Genetic Algorithm to optimize cosmological parameters.
- Main results and tests with its hyperparameters.
- Comparison with other algorithms.
- Discussions and Conclusions.

# What is MODERN cosmology?

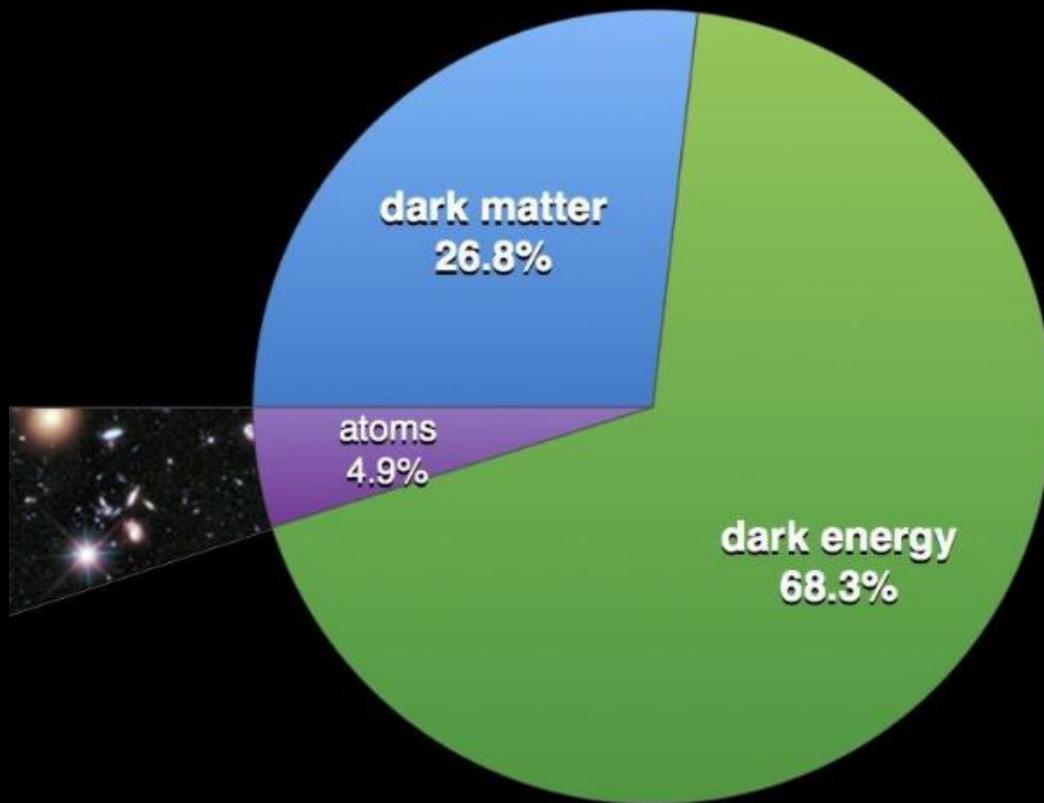
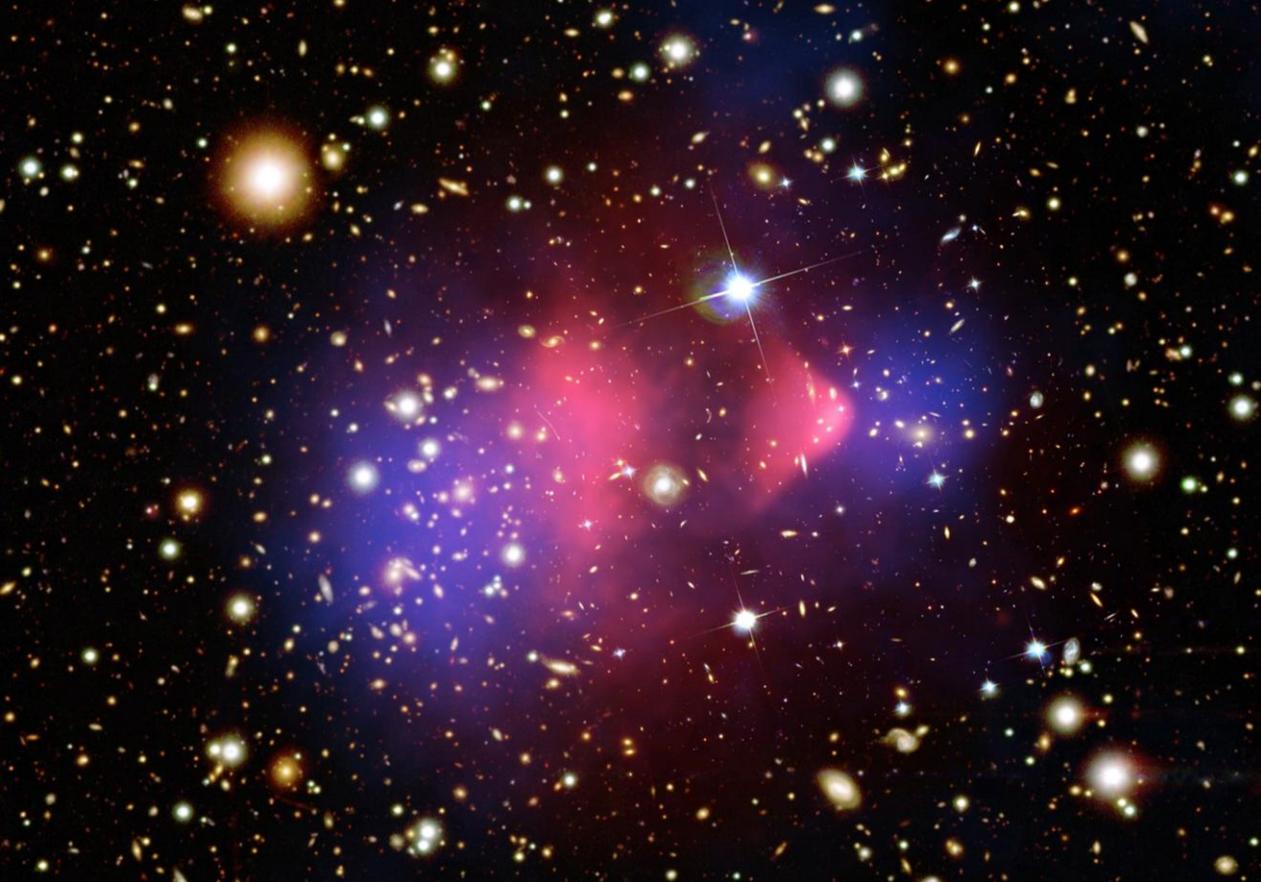
---

The study of the origin, evolution, overall structure, and fate of the Universe.

The standard Cosmological scenario is based on two fundamental assumptions:

- 1) The Cosmological Principle,
- 2) General Relativity.

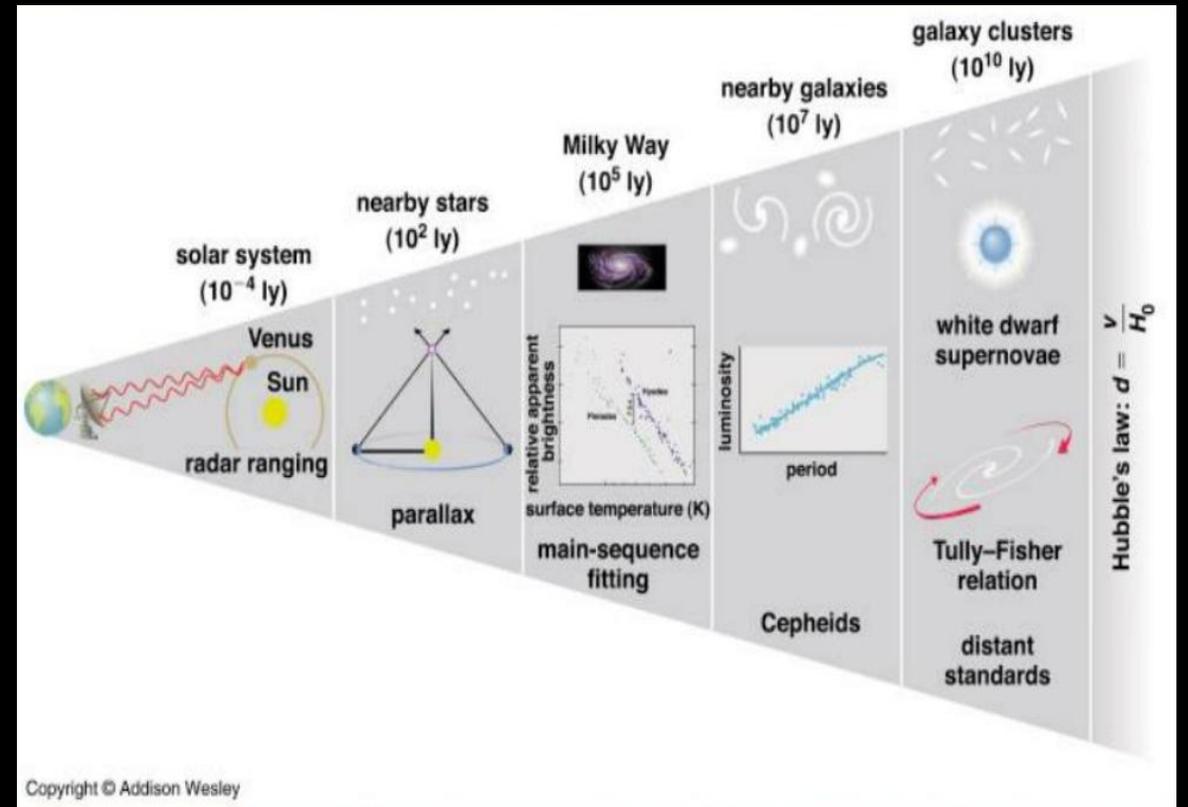
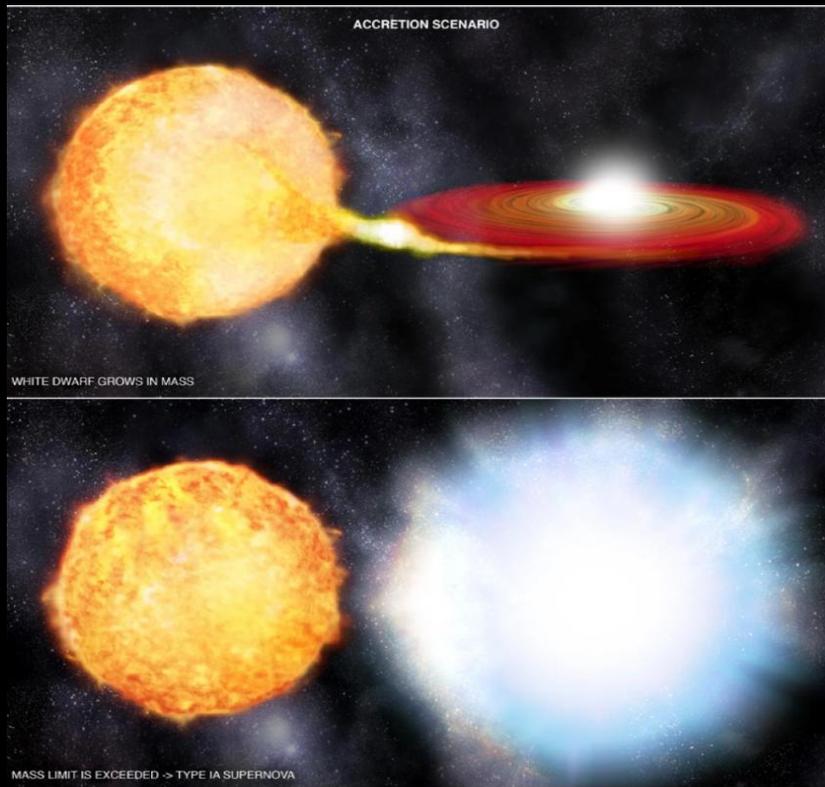
Under these, one can define the so called  $\Lambda$ -Cold Dark Matter ( $\Lambda$ CDM) model, which can allow us to describe the universe using «a small number» of cosmological fundamental parameters.



# The Cosmic Pie

---

# How to measure cosmological parameters: The cosmic ladder



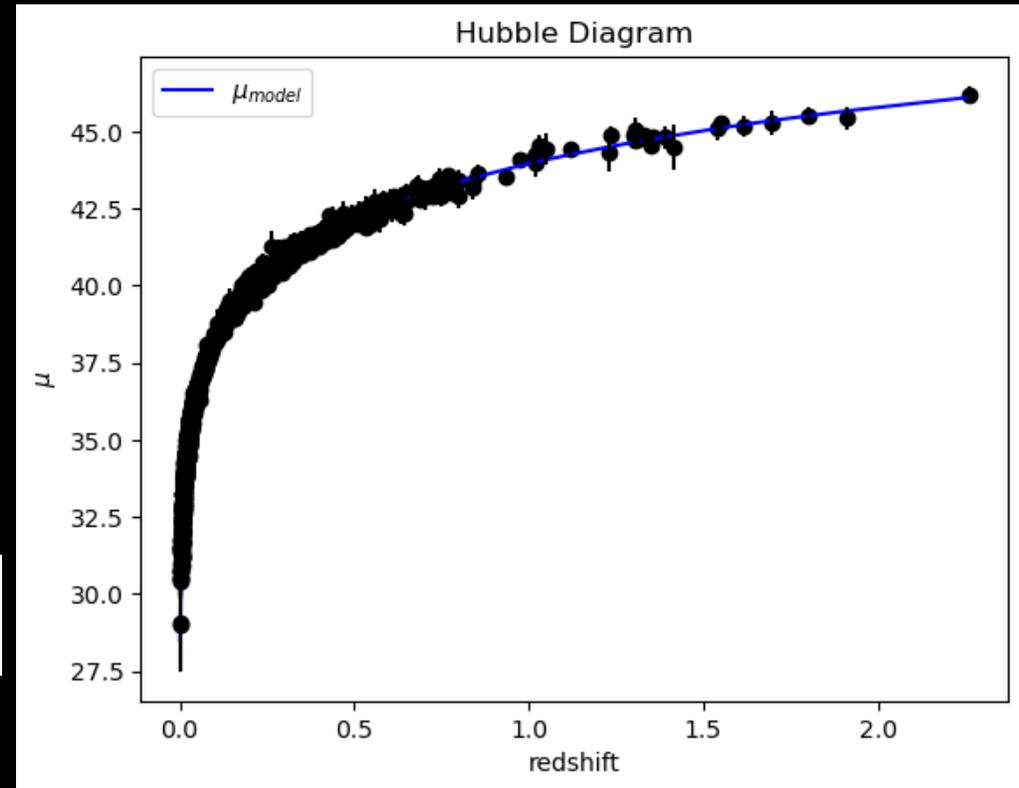
# Objective: the Pantheon+ sample of SNe IA

$$\chi^2 = \Delta \vec{D}^T C_{\text{stat+syst}}^{-1} \Delta \vec{D},$$

$$\mu_{\text{model}}(z_i) = 5 \log(d_L(z_i)/10 \text{ pc}),$$

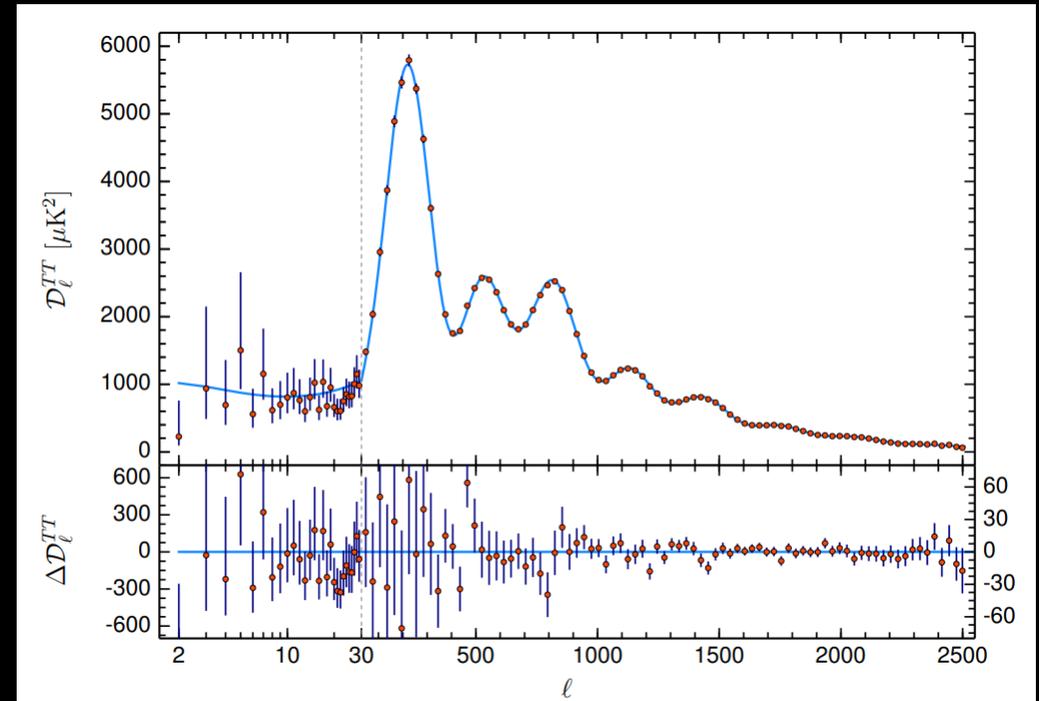
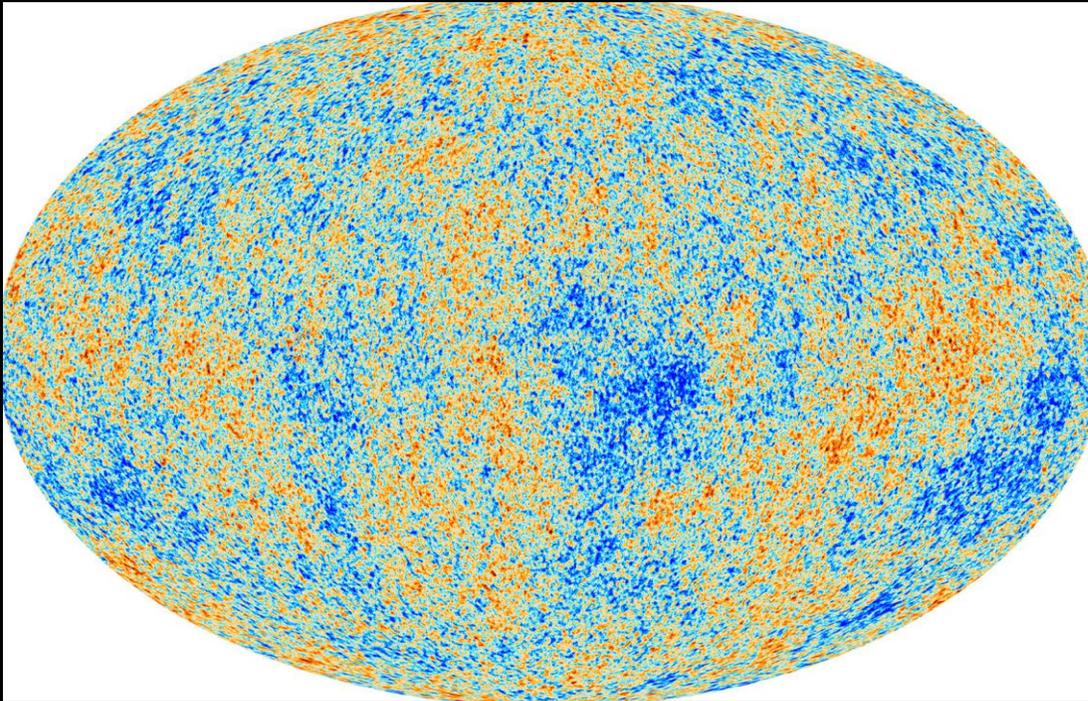
$$d_L(z) = (1+z)c \int_0^z \frac{dz'}{H(z')},$$

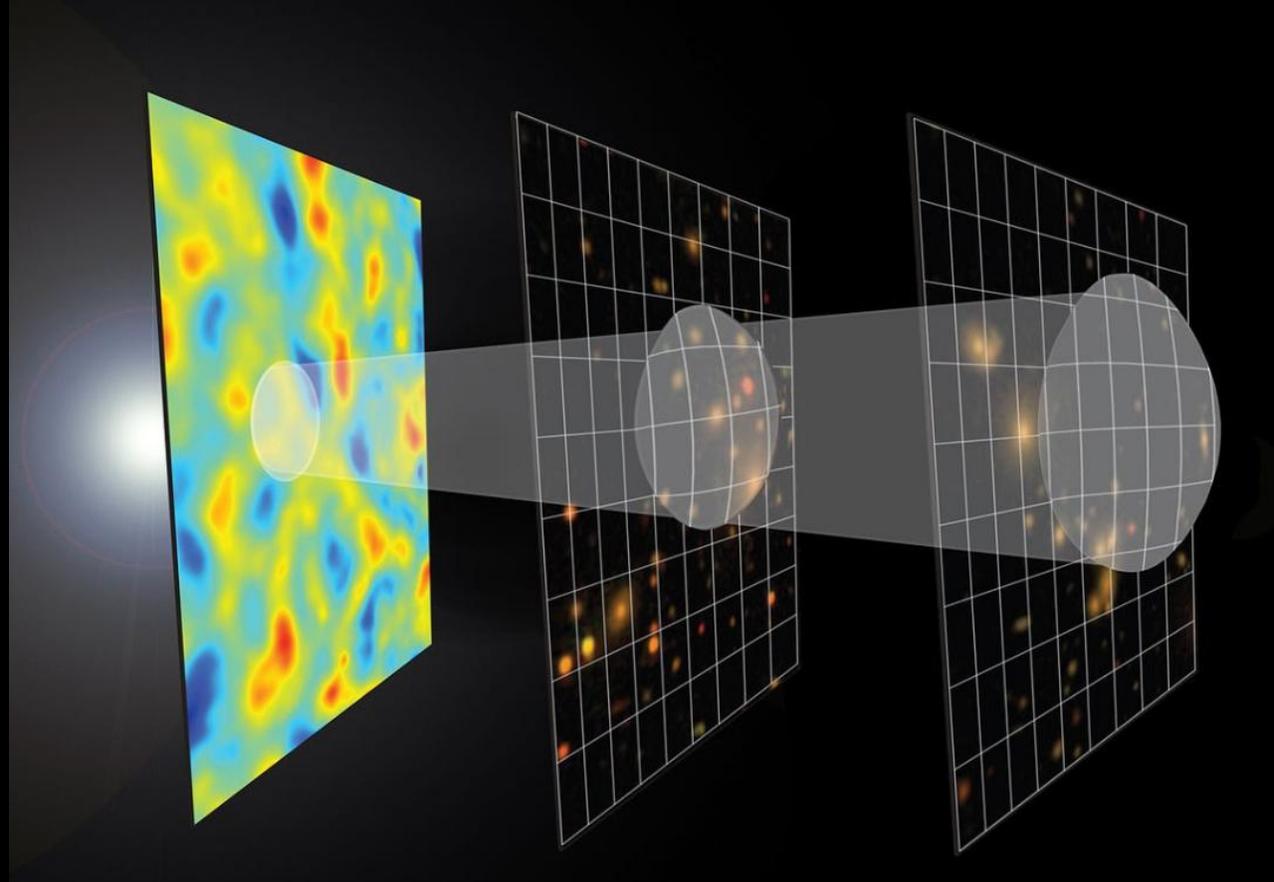
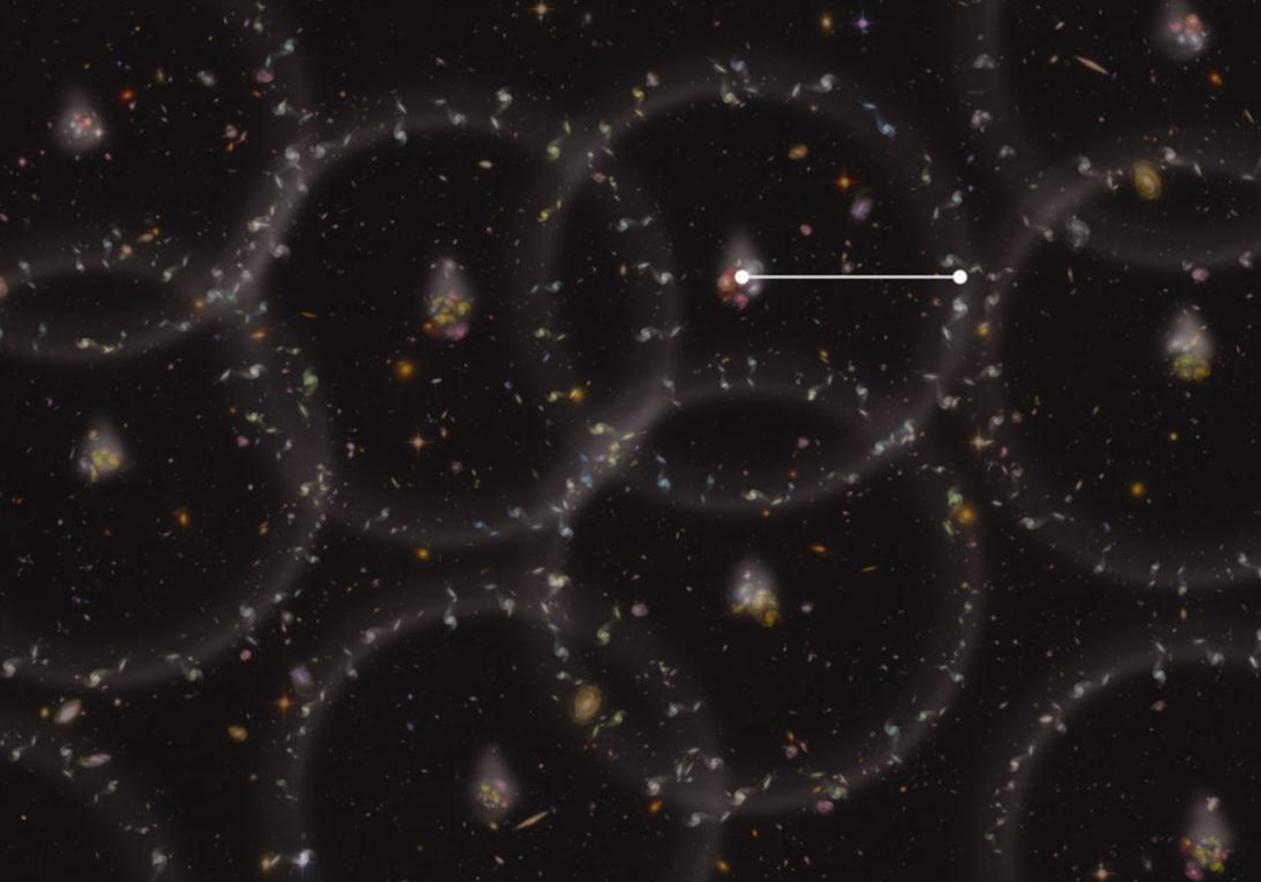
$$H(z) = H_0 \sqrt{\Omega_M(1+z)^3 + \Omega_\Lambda(1+z)^{3(1+w)}}.$$



# How to measure cosmological parameters: The CMB

---





How to measure cosmological parameters: The BAO

---

**CMB with Planck**  
 Pogosian et al. (2020), eBOSS+Planck  $\Omega_m H^2$ :  $69.6 \pm 1.8$   
 Aghanim et al. (2020), Planck 2018:  $67.27 \pm 0.60$   
 Aghanim et al. (2020), Planck 2018+CMB lensing:  $67.36 \pm 0.54$

**CMB without Planck**  
 Aiola et al. (2020), ACT:  $67.9 \pm 1.5$   
 Aiola et al. (2020), WMAP9+ACT:  $67.6 \pm 1.1$   
 Zhang, Huang (2019), WMAP9+BAO:  $68.36^{+0.32}_{-0.32}$   
 Henning et al. (2018), SPT:  $71.3 \pm 1.4$   
 Hinshaw et al. (2013), WMAP9:  $70.0 \pm 2.2$

**No CMB, with BBN**  
 D'Amico et al. (2020), BOSS DR12+BBN:  $68.5 \pm 2.2$   
 Ivanov et al. (2020), DOGS+BBN:  $67.9 \pm 1.1$   
 Alam et al. (2020), BOSS+eBOSS+BBN:  $67.35 \pm 0.97$

**$P_1(k)$  + CMB lensing**  
 Philcox et al. (2020),  $P_1(k)$ +CMB lensing:  $70.6^{+3.2}_{-3.2}$

**Cepheids – SNIa**  
 Riess et al. (2020), R20:  $73.2 \pm 1.3$   
 Breuval et al. (2020):  $72.8 \pm 2.7$   
 Riess et al. (2019), R19:  $74.0 \pm 1.4$   
 Camarena, Marra (2019):  $75.4 \pm 1.7$   
 Burns et al. (2018):  $73.2 \pm 2.3$   
 Dhawan, Jha, Leibundgut (2017), NIR:  $72.8 \pm 3.1$   
 Folini, Knox (2017):  $73.3 \pm 1.7$   
 Feeney, Mortlock, Dalmaso (2017):  $73.2 \pm 1.8$   
 Riess et al. (2016), R16:  $73.2 \pm 1.7$   
 Cardona, Kunz, Pettorino (2016), HPS:  $73.8 \pm 2.1$   
 Freedman et al. (2012):  $74.3 \pm 2.1$

**TRGB – SNIa**  
 Soltis, Casertano, Riess (2020):  $72.1 \pm 2.0$   
 Freedman et al. (2020):  $69.6 \pm 1.9$   
 Reid, Pesce, Riess (2019), SHOES:  $71.1 \pm 1.9$   
 Freedman et al. (2019):  $69.8 \pm 1.9$   
 Yuan et al. (2019):  $72.4 \pm 2.0$   
 Jang, Lee (2017):  $71.2 \pm 2.5$

**Miras – SNIa**  
 Huang et al. (2019):  $73.3 \pm 4.0$

**Masers**  
 Pesce et al. (2020):  $73.9 \pm 3.0$

**Tully – Fisher Relation (TFR)**  
 Kourkchi et al. (2020):  $76.0 \pm 2.6$   
 Schombert, McGaugh, Lelli (2020):  $75.1 \pm 2.8$

**Surface Brightness Fluctuations**  
 Blakeslee et al. (2021) IR-SBF w/ HST:  $73.3 \pm 2.5$   
 Khetan et al. (2020) w/ LMC DEB:  $71.1 \pm 4.1$

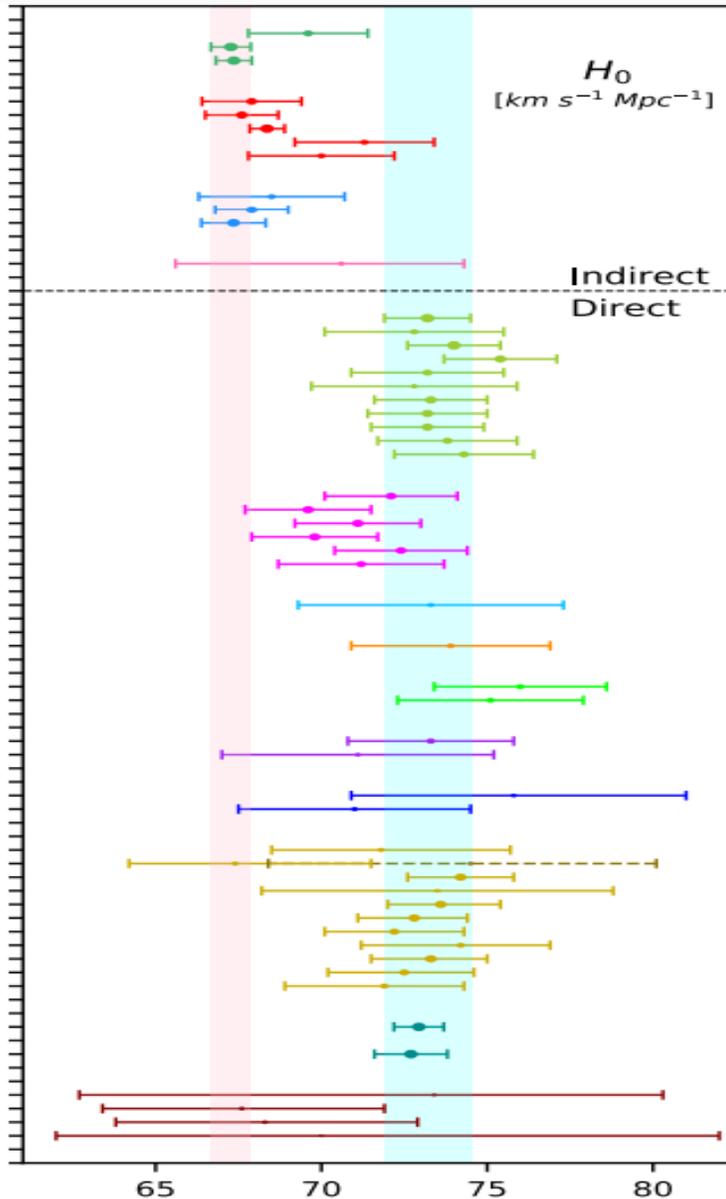
**SNIi**  
 de Jaeger et al. (2020):  $75.8^{+3.5}_{-3.5}$   
 Fernández Arenas et al. (2018):  $71.0 \pm 3.5$

**Lensing related, mass model – dependent**  
 Denzel et al. (2021):  $71.8^{+3.9}_{-3.9}$   
 Birrer et al. (2020), TDCOSMO+SLACS:  $67.4^{+3.3}_{-3.3}$ , TDCOSMO:  $74.5^{+2.4}_{-2.4}$   
 Millon et al. (2020), TDCOSMO:  $74.2 \pm 1.6$   
 Baxter et al. (2020):  $73.5 \pm 5.3$   
 Qi et al. (2020):  $73.6^{+1.9}_{-1.9}$   
 Liao et al. (2020):  $72.8^{+1.8}_{-1.8}$   
 Liao et al. (2019):  $72.2 \pm 2.1$   
 Shajib et al. (2019), STRIDES:  $74.2^{+2.7}_{-2.7}$   
 Wong et al. (2019), HOLICOW 2019:  $73.3^{+1.7}_{-1.7}$   
 Birrer et al. (2018), HOLICOW 2018:  $72.5^{+2.4}_{-2.4}$   
 Bonvin et al. (2016), HOLICOW 2016:  $71.9^{+2.0}_{-2.0}$

**Optimist average**  
 Di Valentino (2021):  $72.94 \pm 0.75$

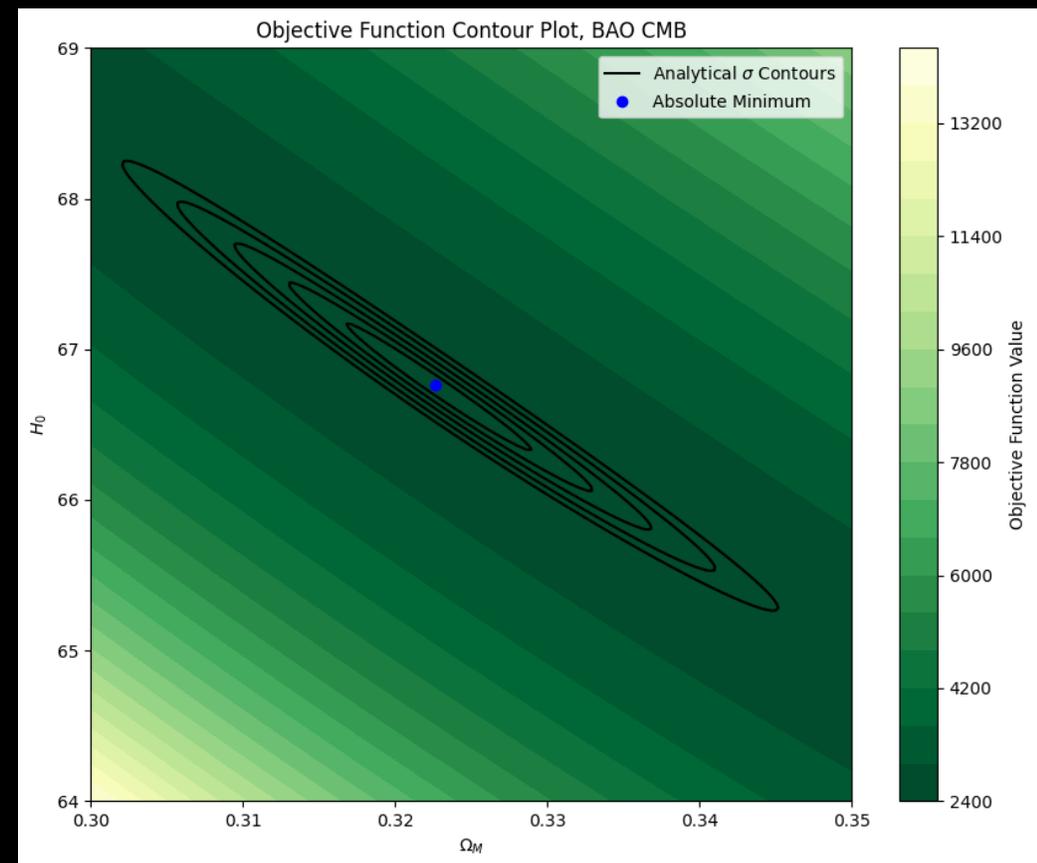
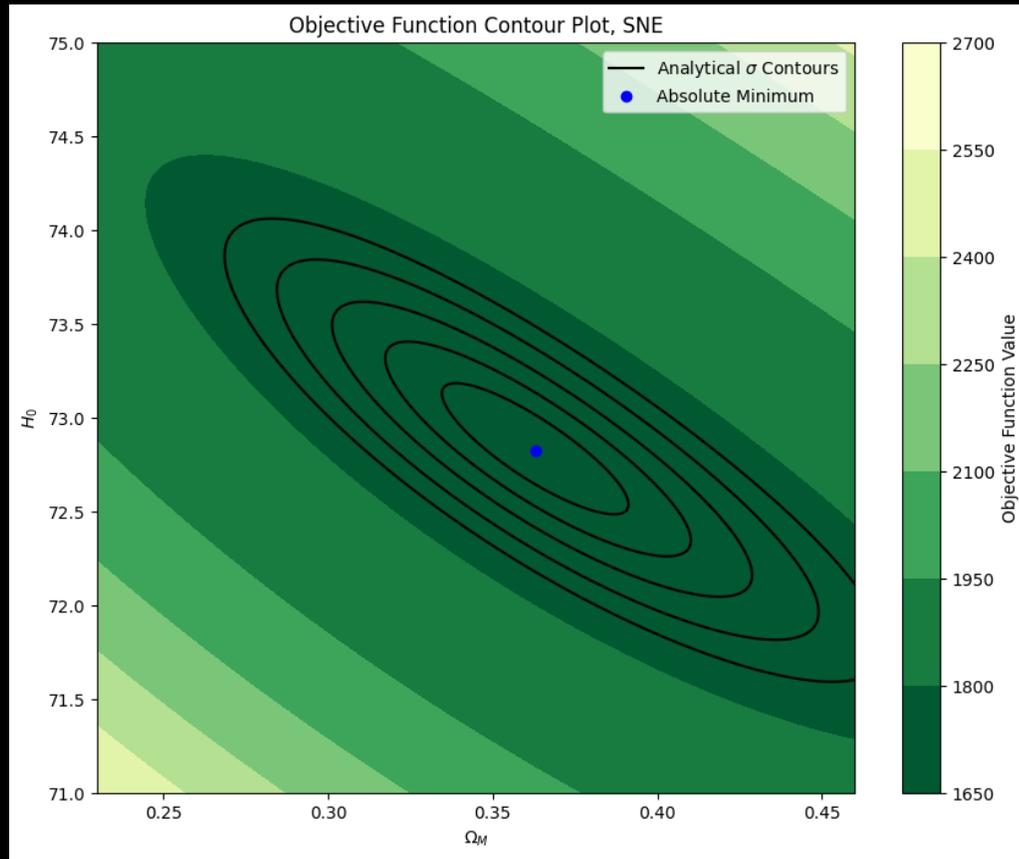
**Ultra – conservative, no Cepheids, no lensing**  
 Di Valentino (2021):  $72.7 \pm 1.1$

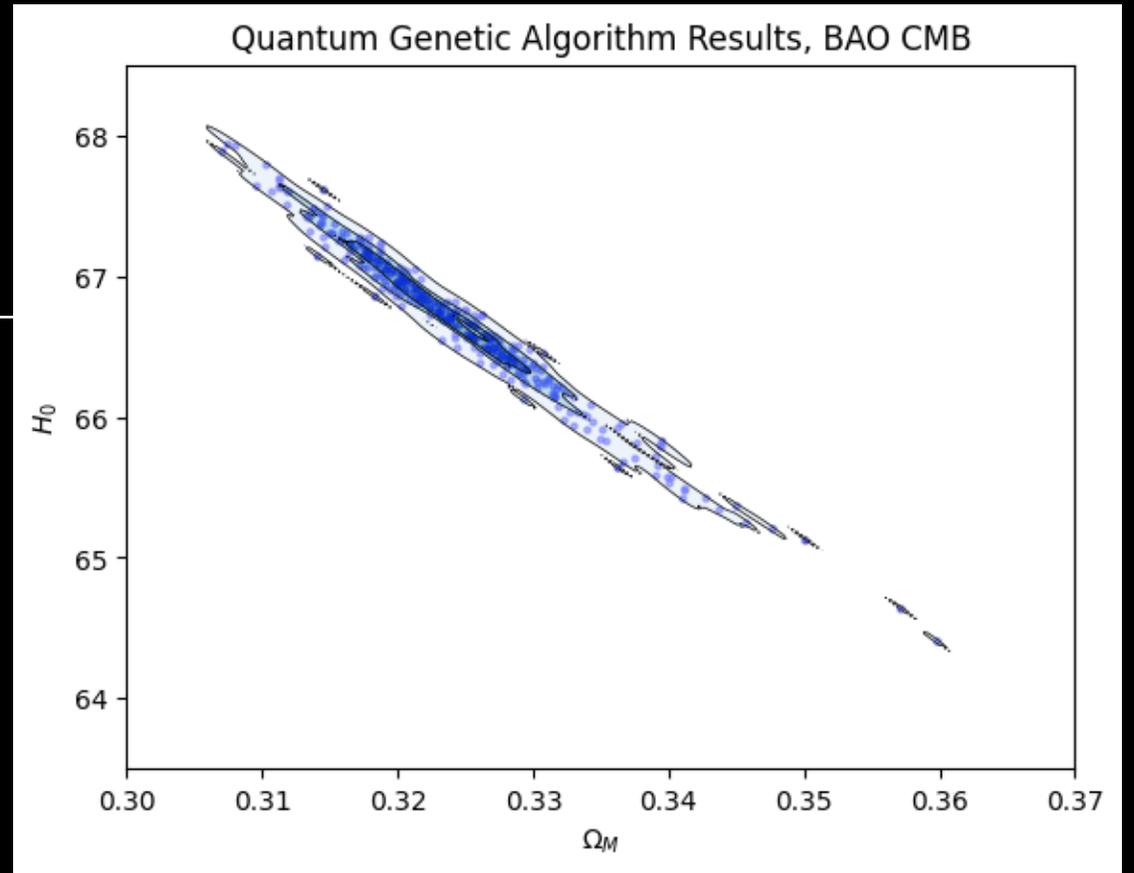
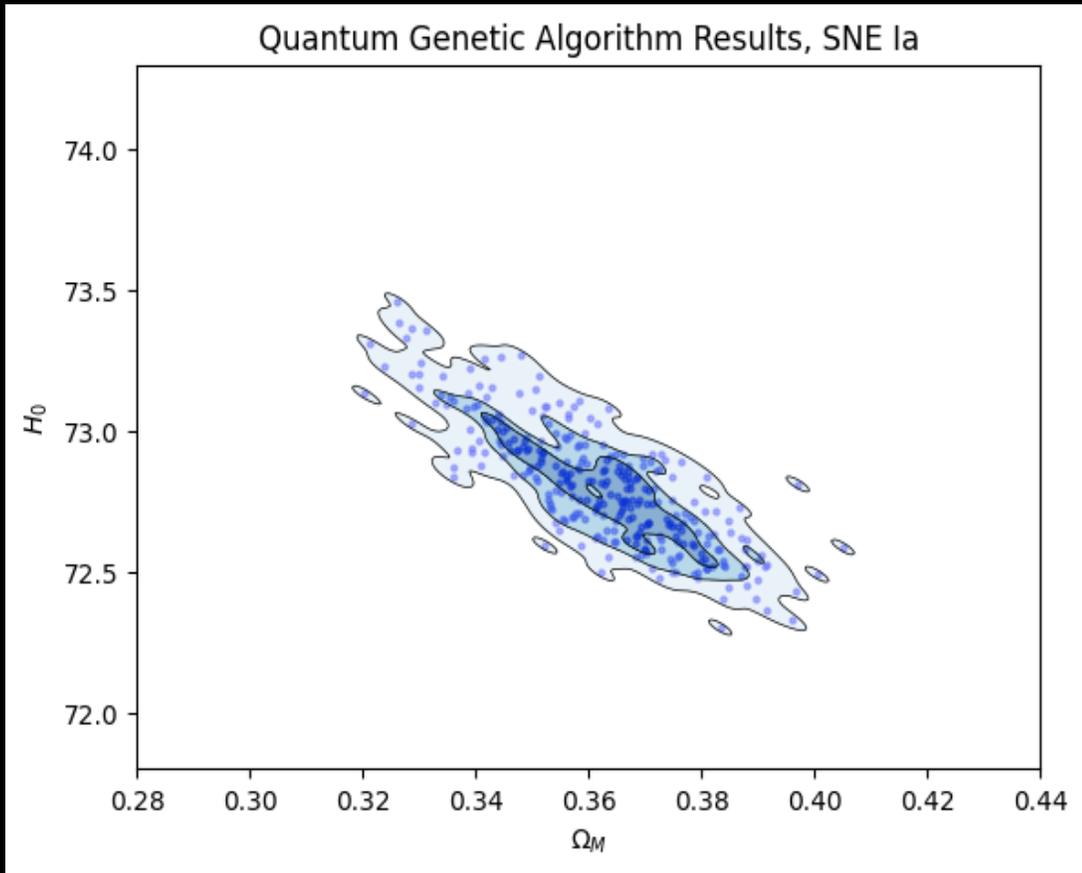
**GW related**  
 Gayathri et al. (2020), GW190521+GW170817:  $73.4^{+1.9}_{-1.9}$   
 Mukherjee et al. (2020), GW170817+ZTF:  $67.6^{+1.9}_{-1.9}$   
 Mukherjee et al. (2019), GW170817+VLBI:  $68.3^{+1.8}_{-1.8}$   
 Abbott et al. (2017), GW170817:  $70.0^{+1.0}_{-1.0}$



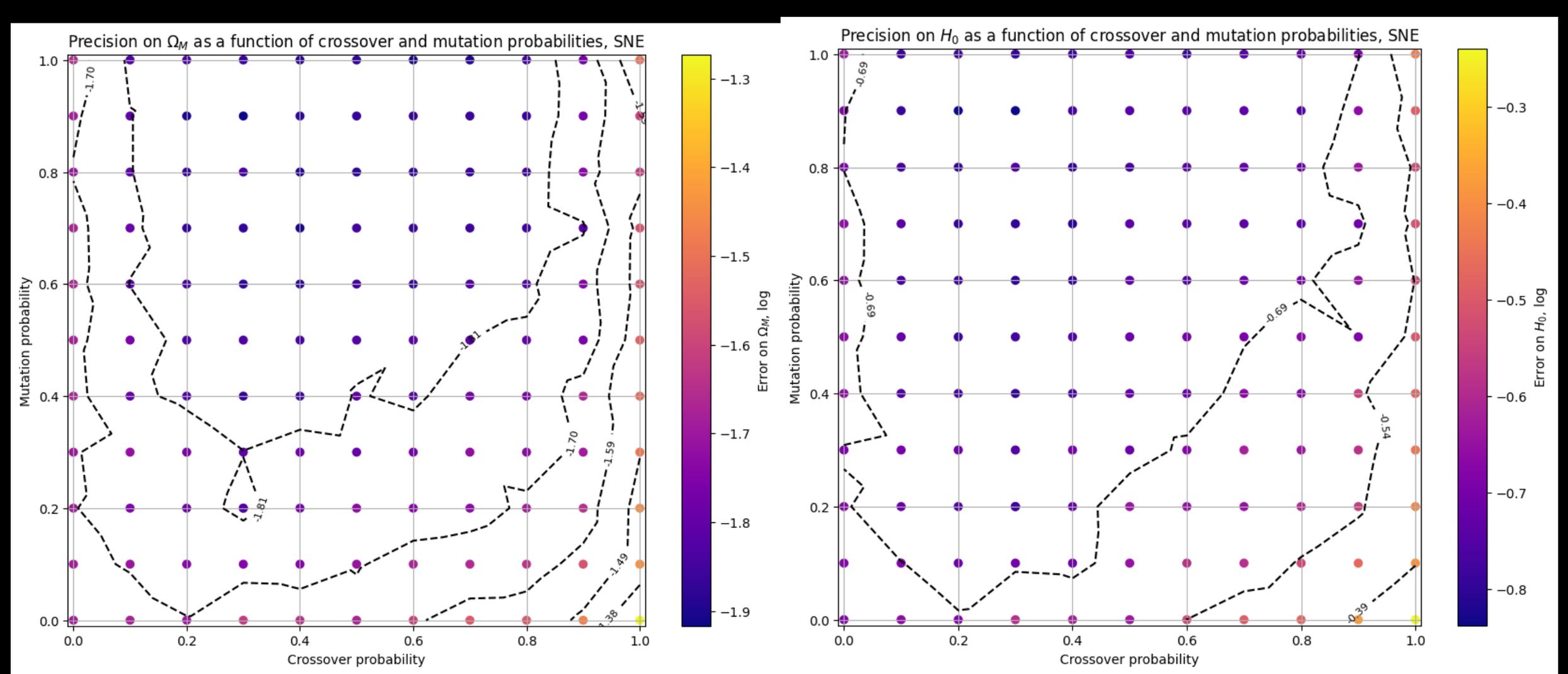
# The Hubble Tension

# Contour Maps for the Cosmological Objective functions





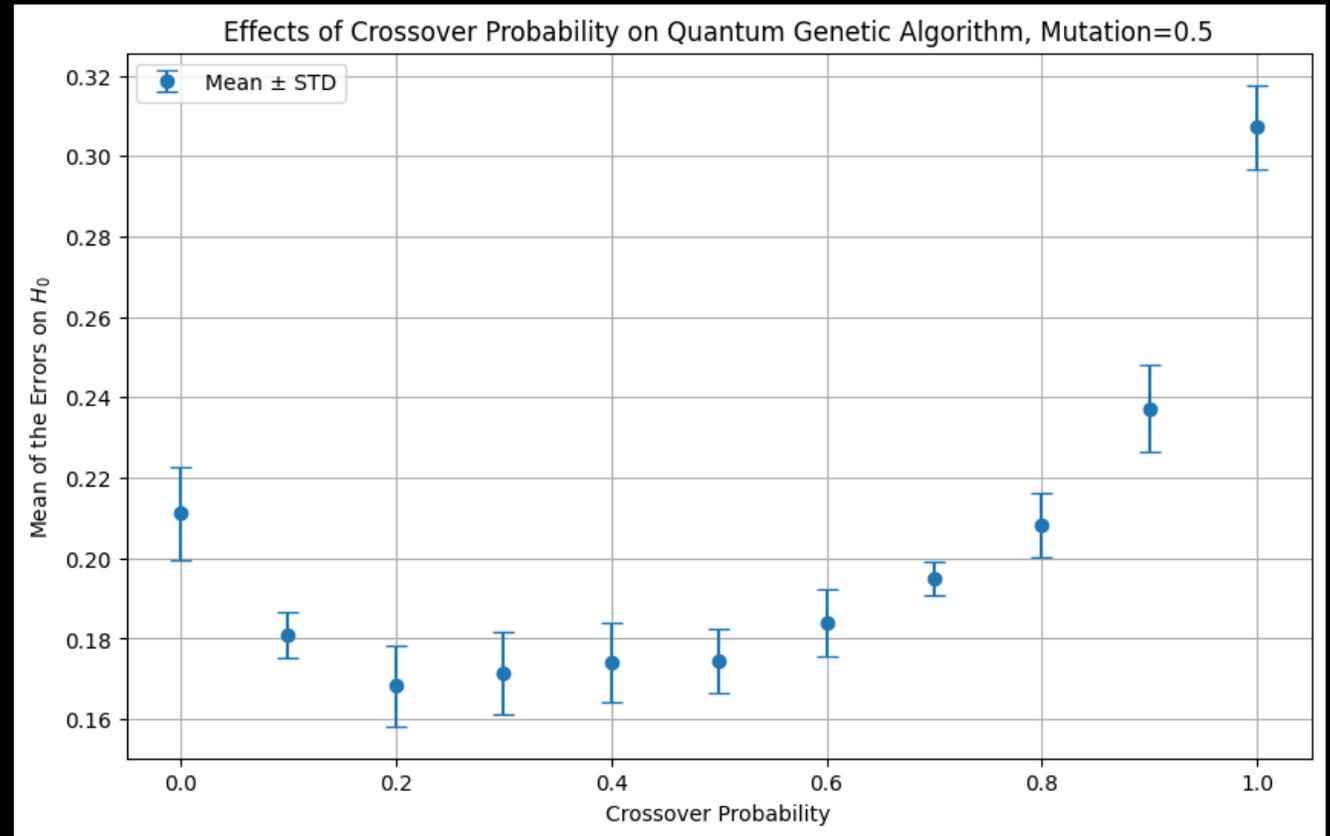
Results (SNe Ia left, CMB+BAO right)

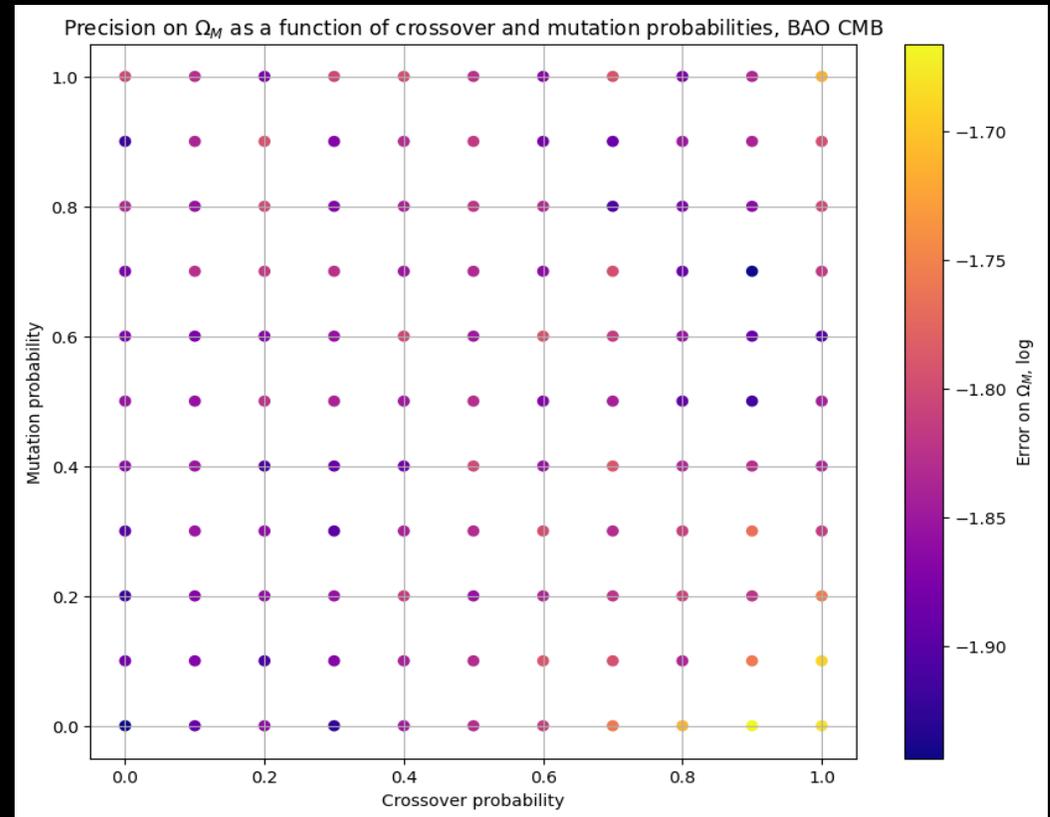
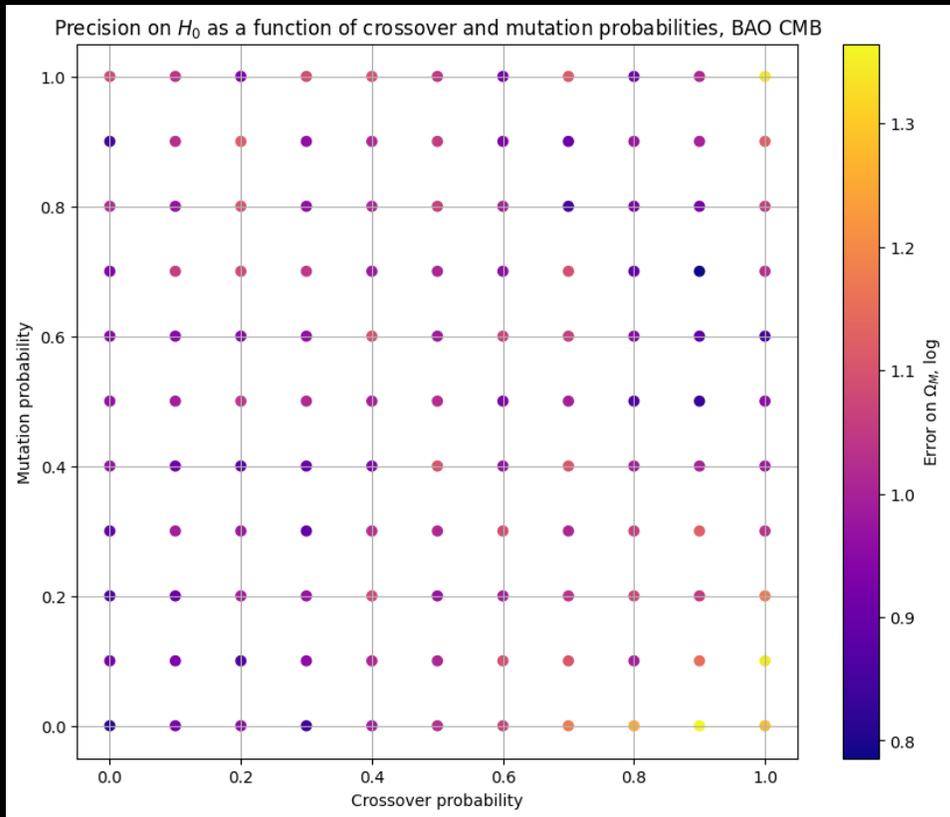


Testing the crossover and mutation effects for the SNe results

# Studying the stability of the Algorithm, SNe Ia, error on $H_0$

---

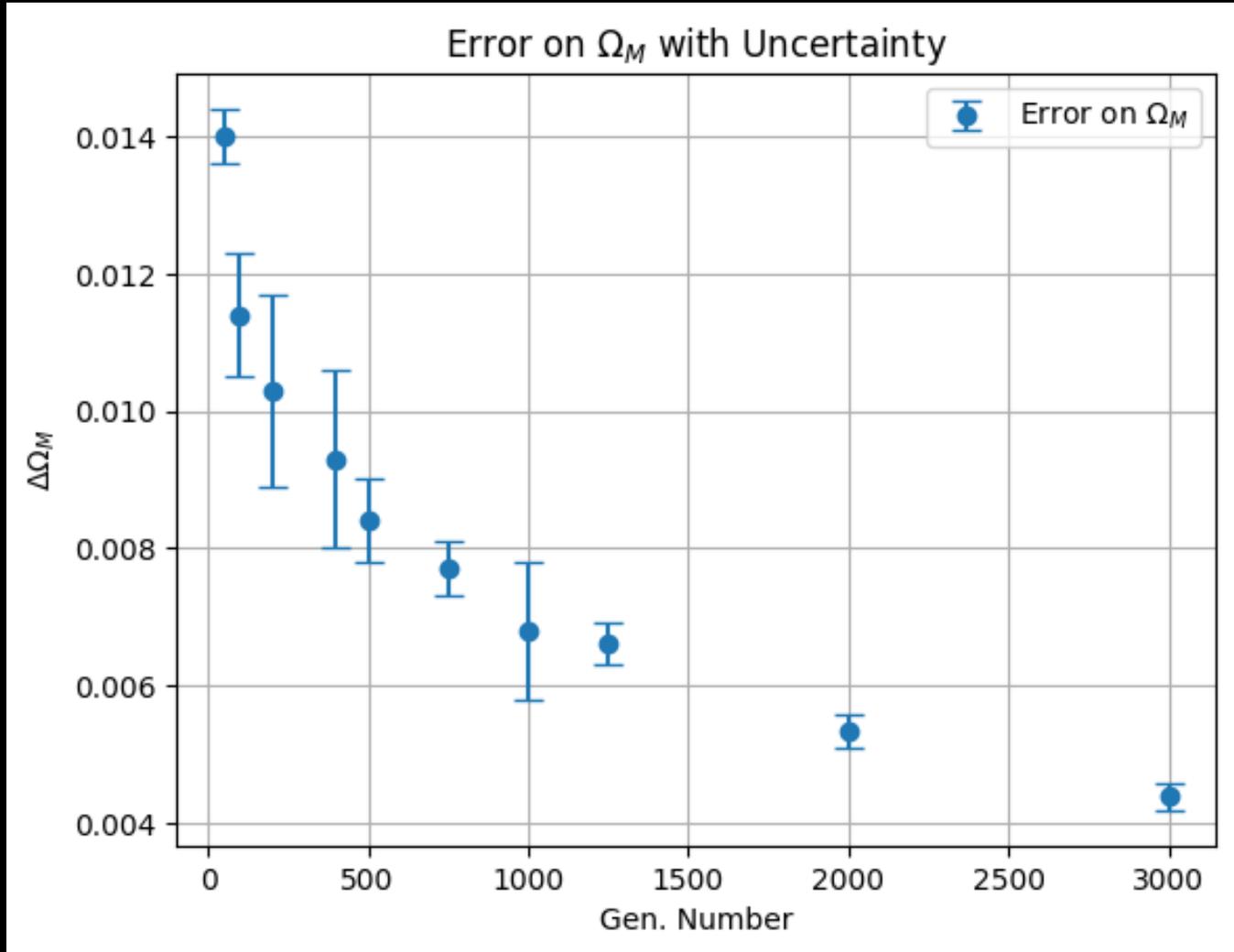




Testing the crossover and mutation effects for the CMB+BAO results

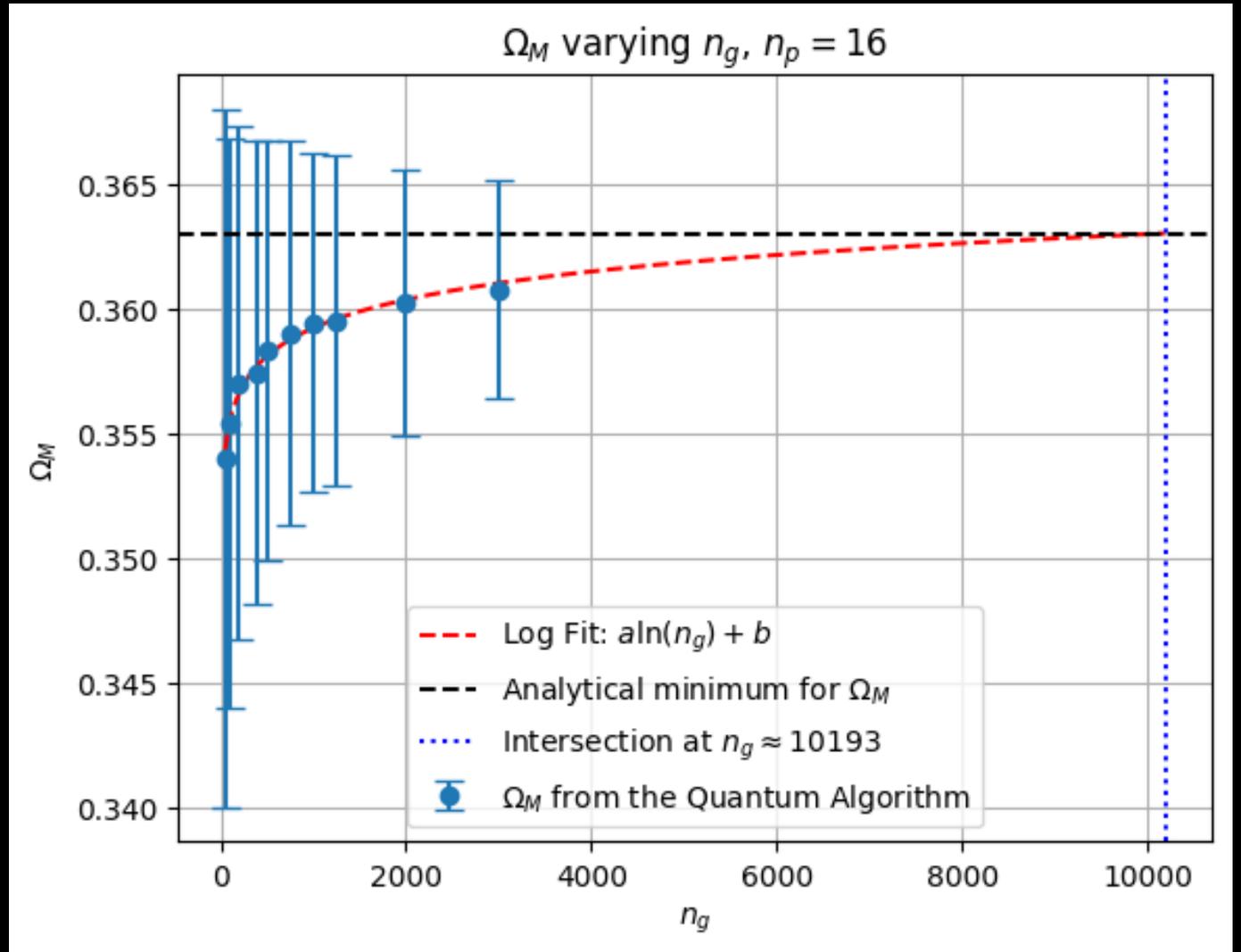
Precision and  
number of  
generations,  
SNE

---

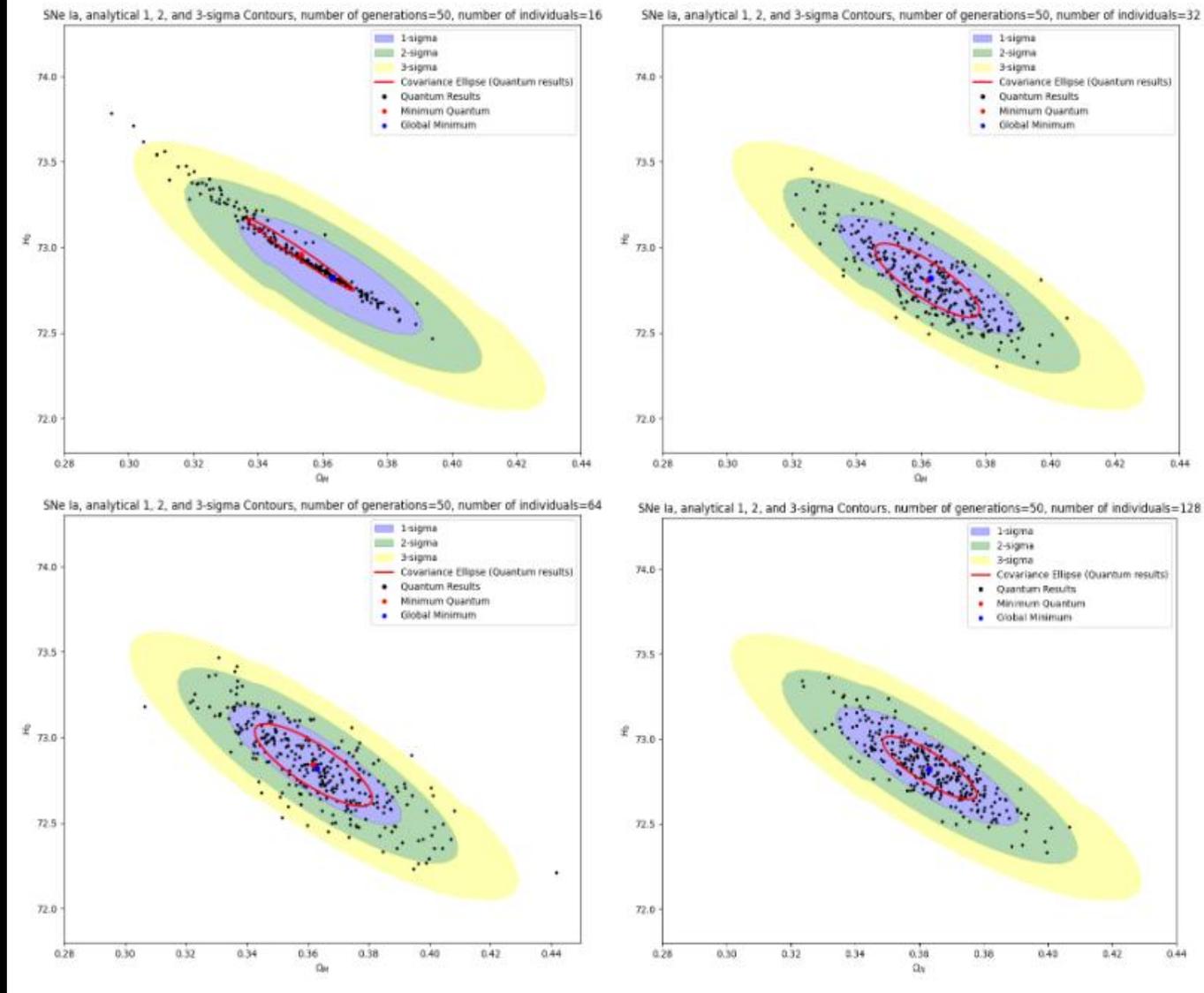


Mean value and number of generations, SNE, with 16 individuals

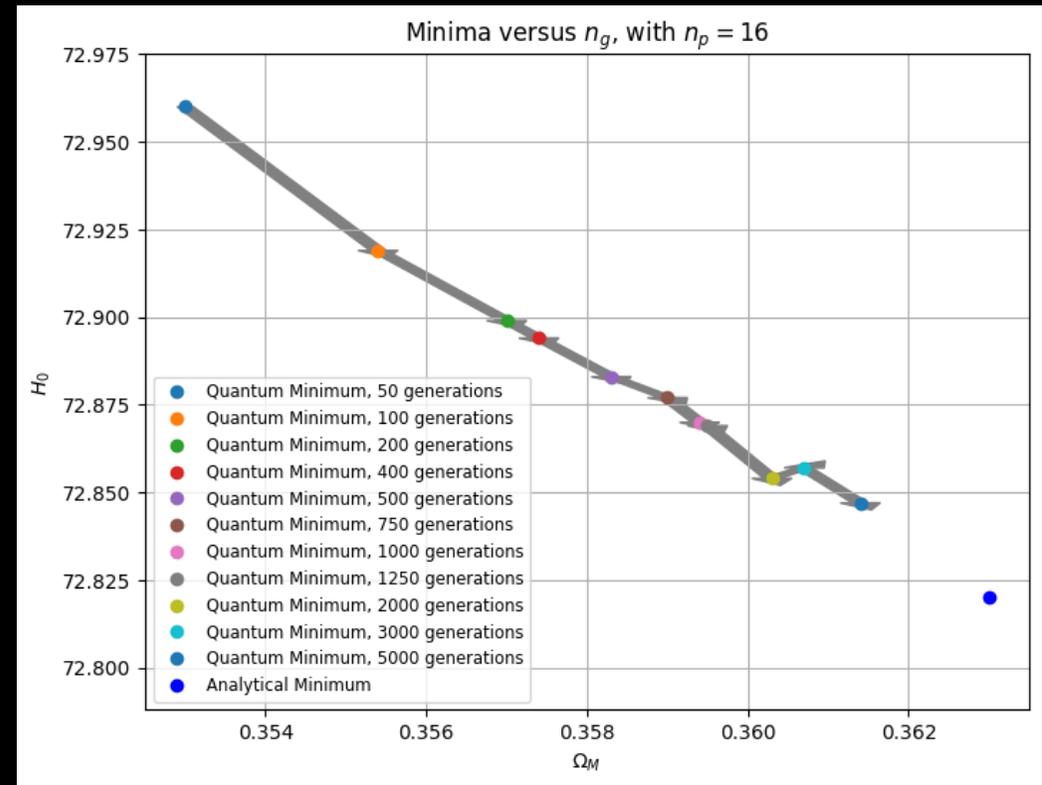
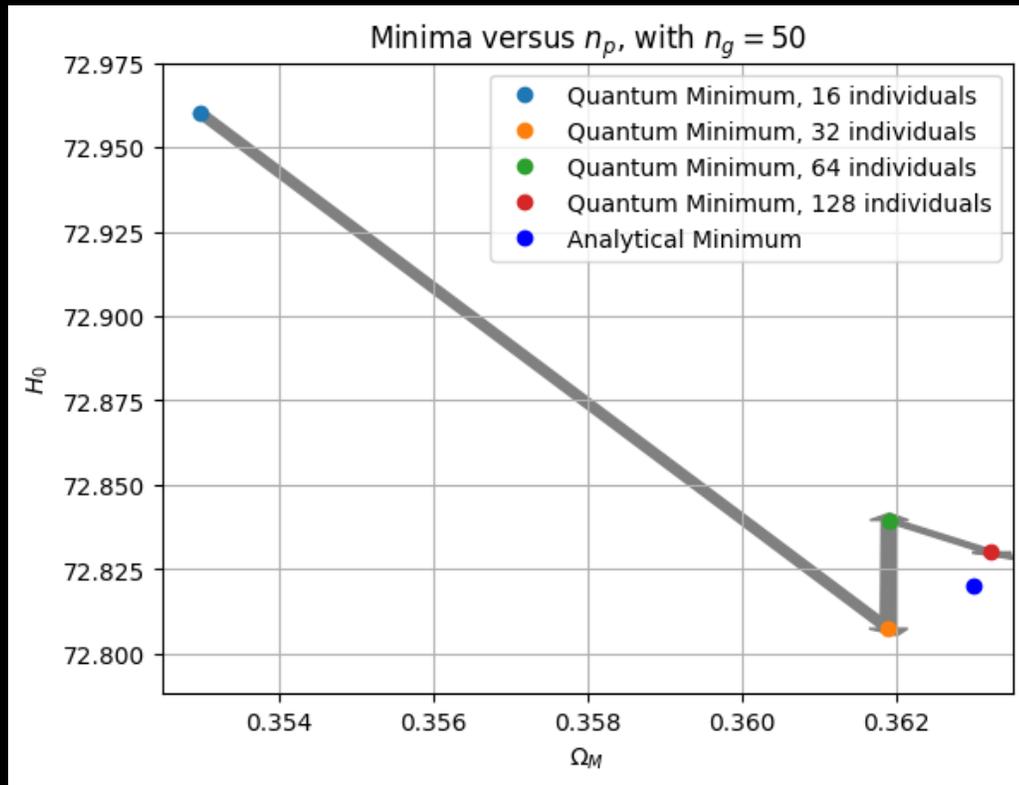
---

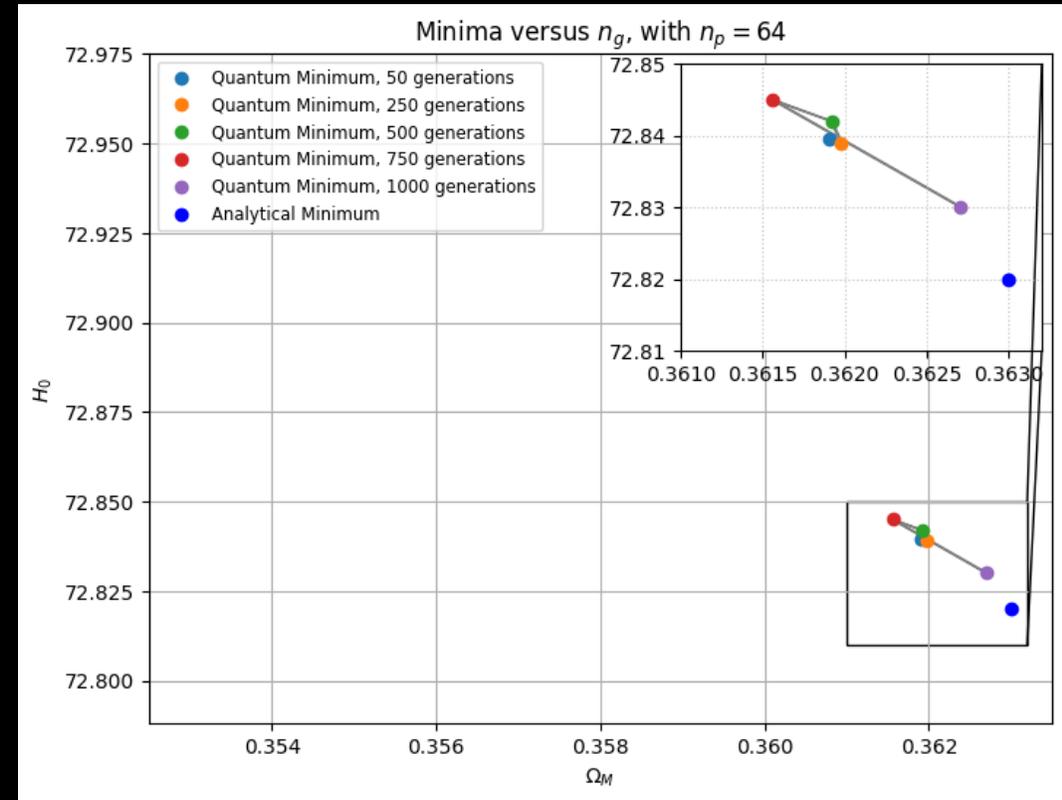
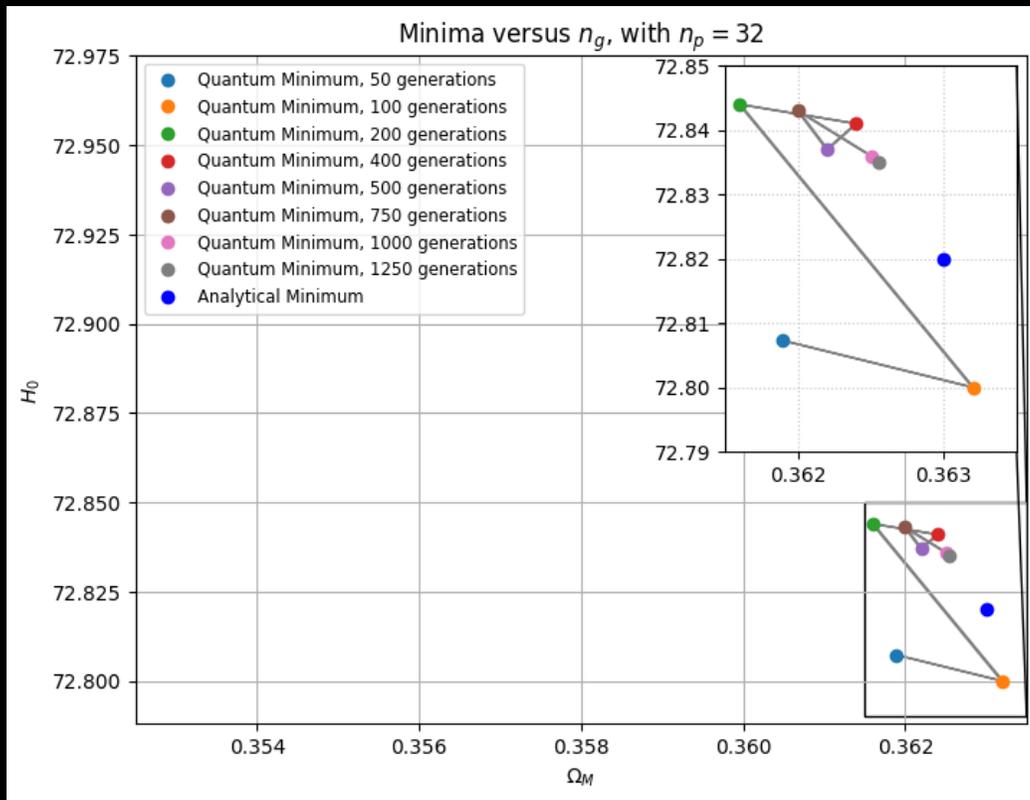


# Varying the number of individuals, SNE

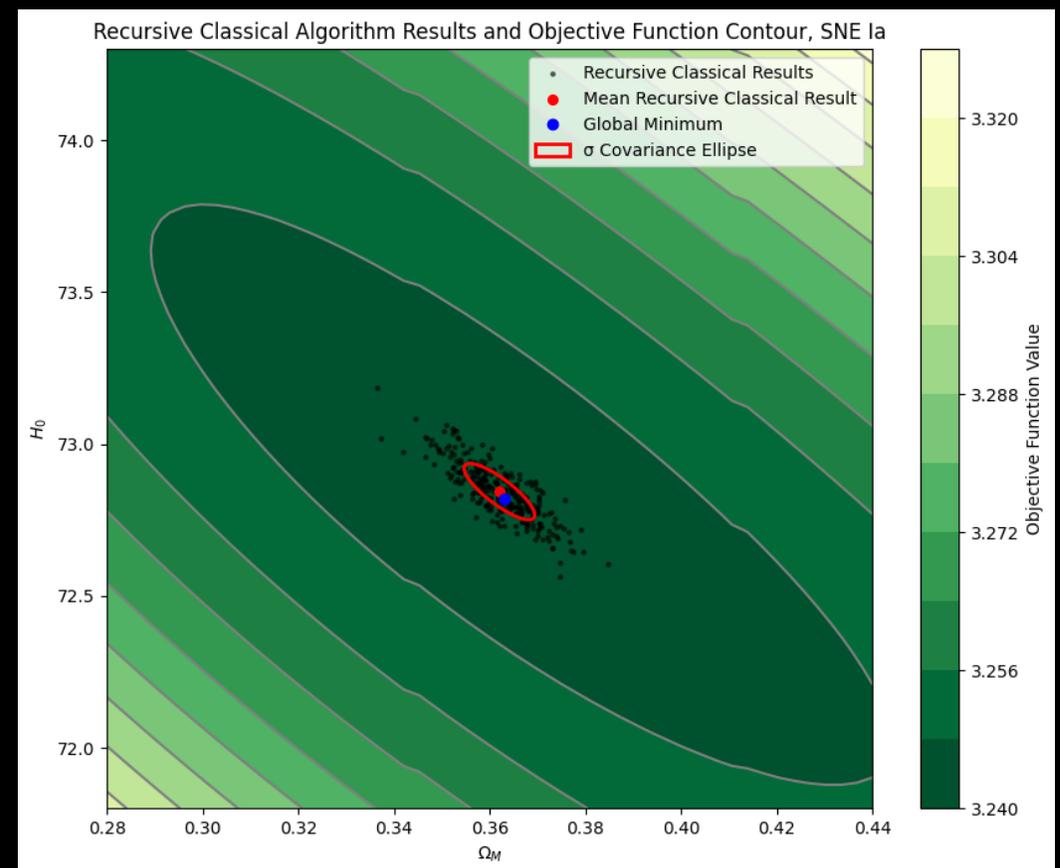
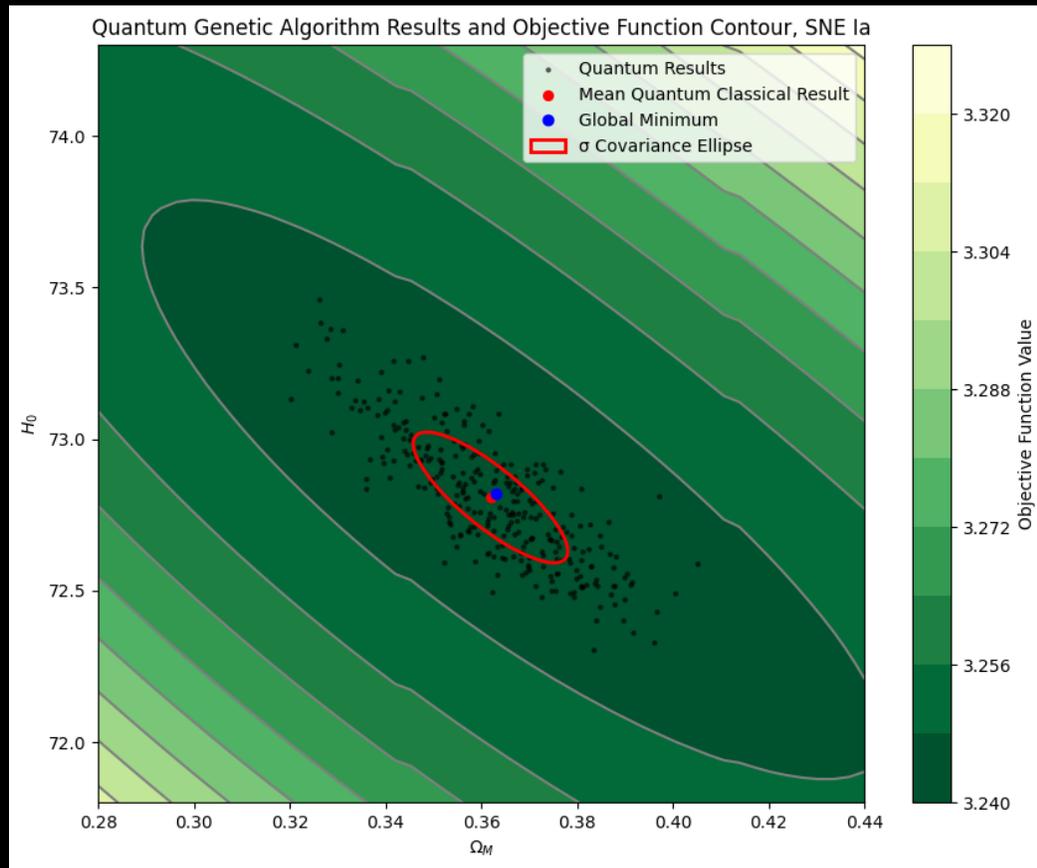


# Minima with generations and individuals

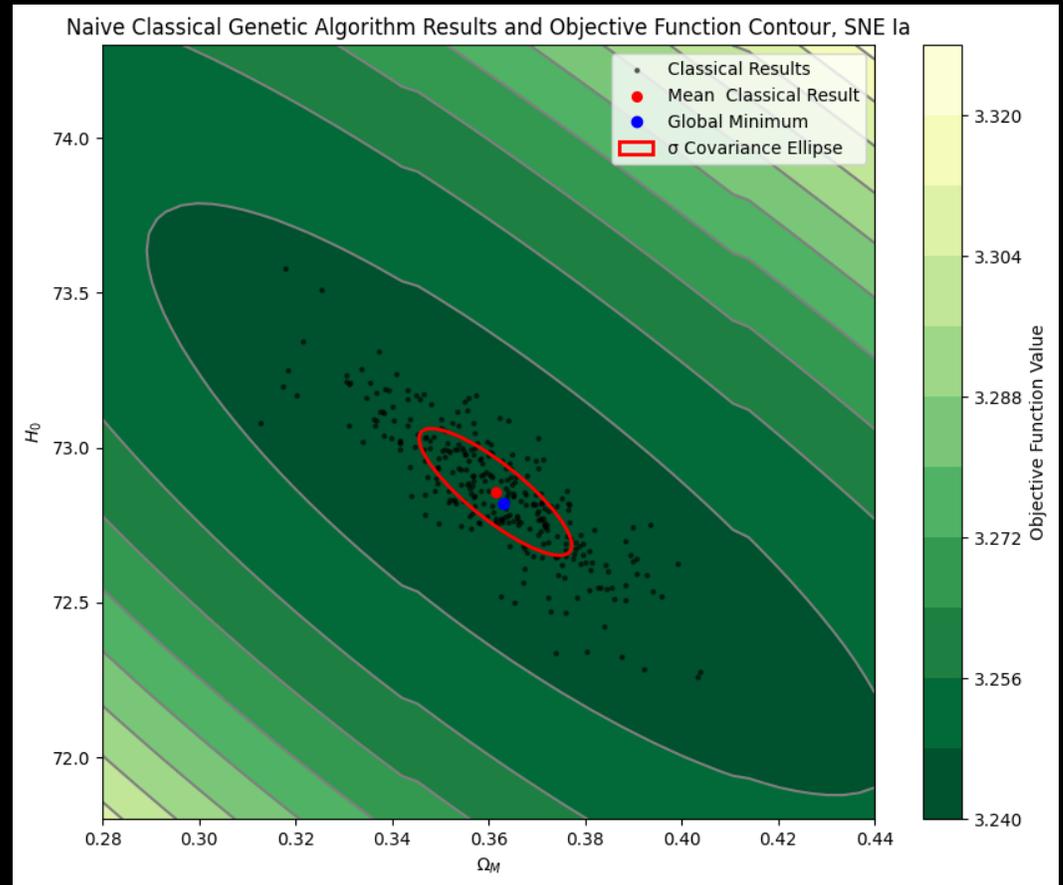
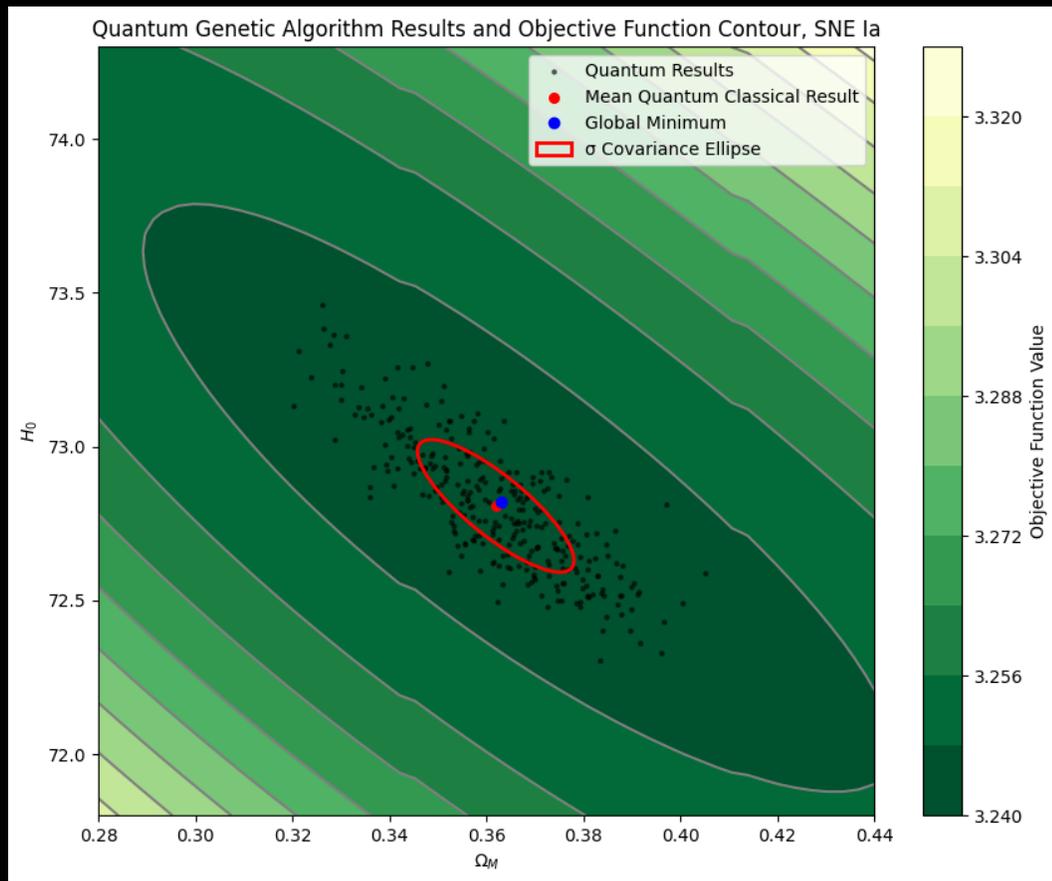




Minima with generations, higher number of individuals



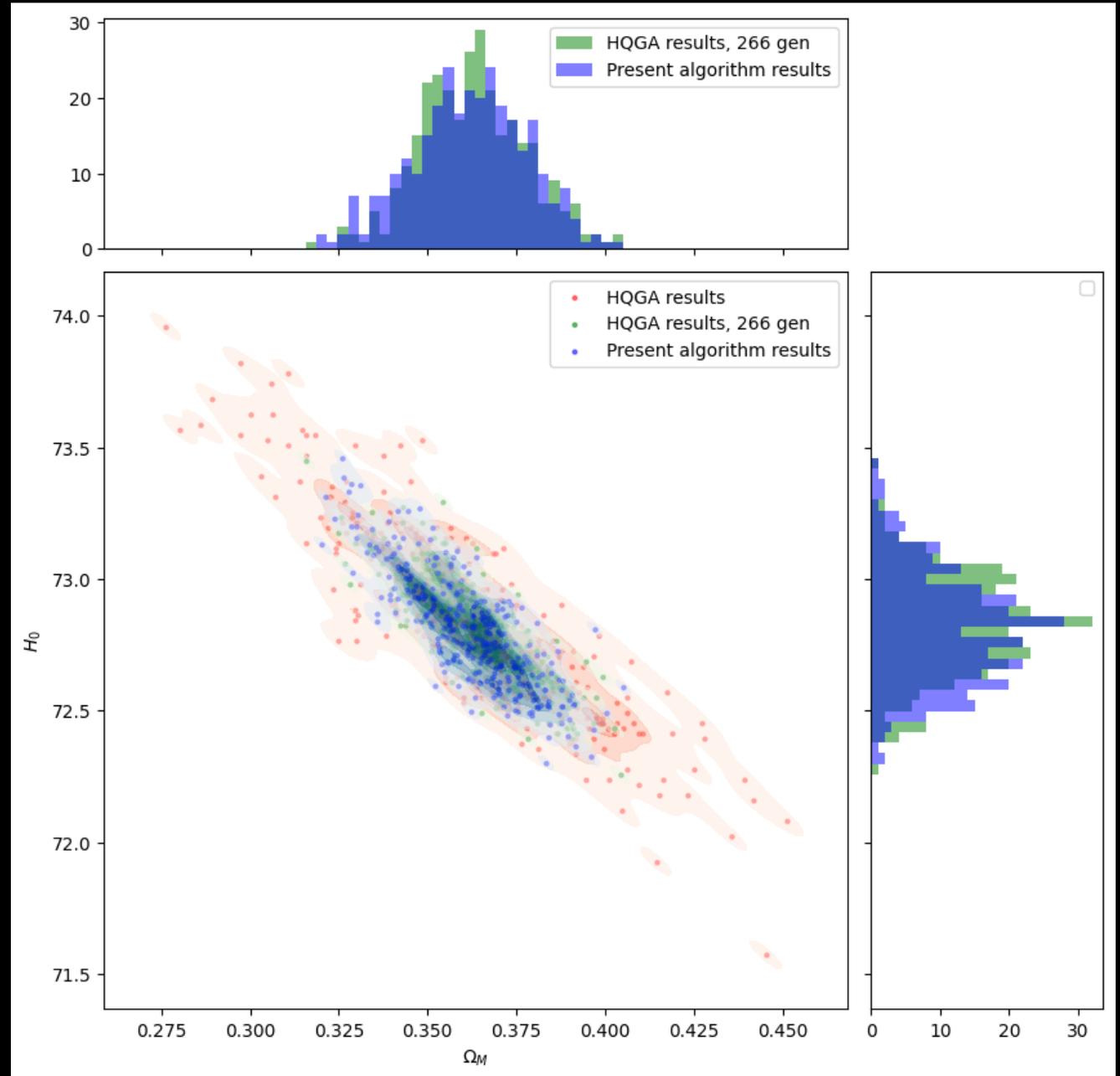
# Comparison with classical algorithms (1)



Comparison with classical algorithms (2)

# Comparison with HQGA

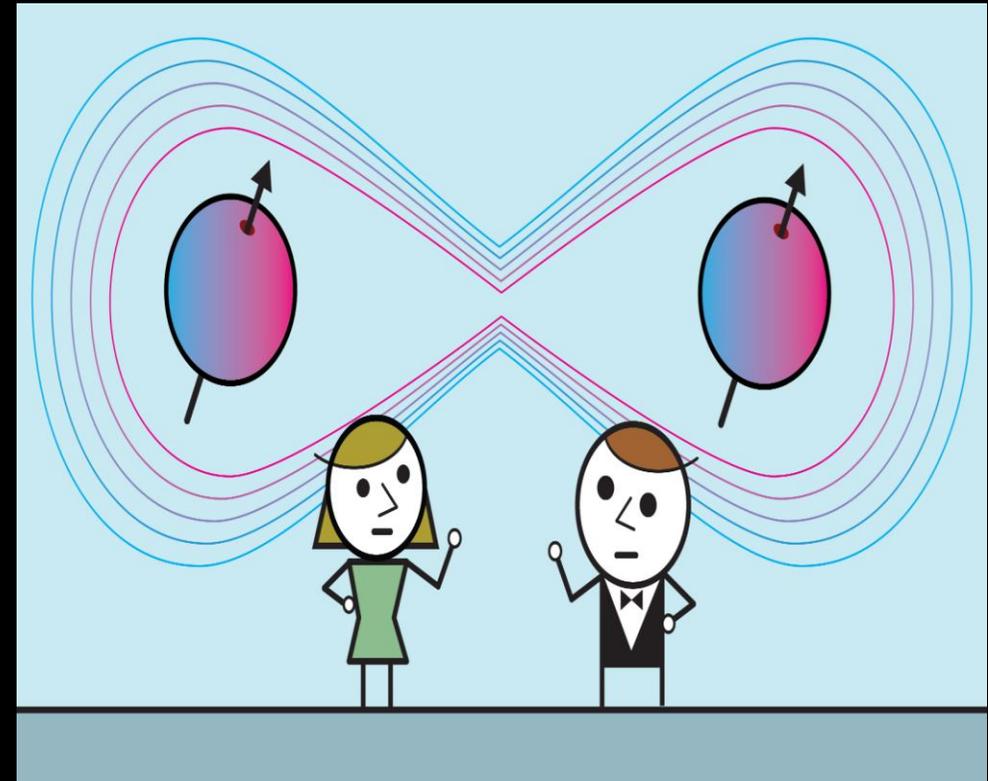
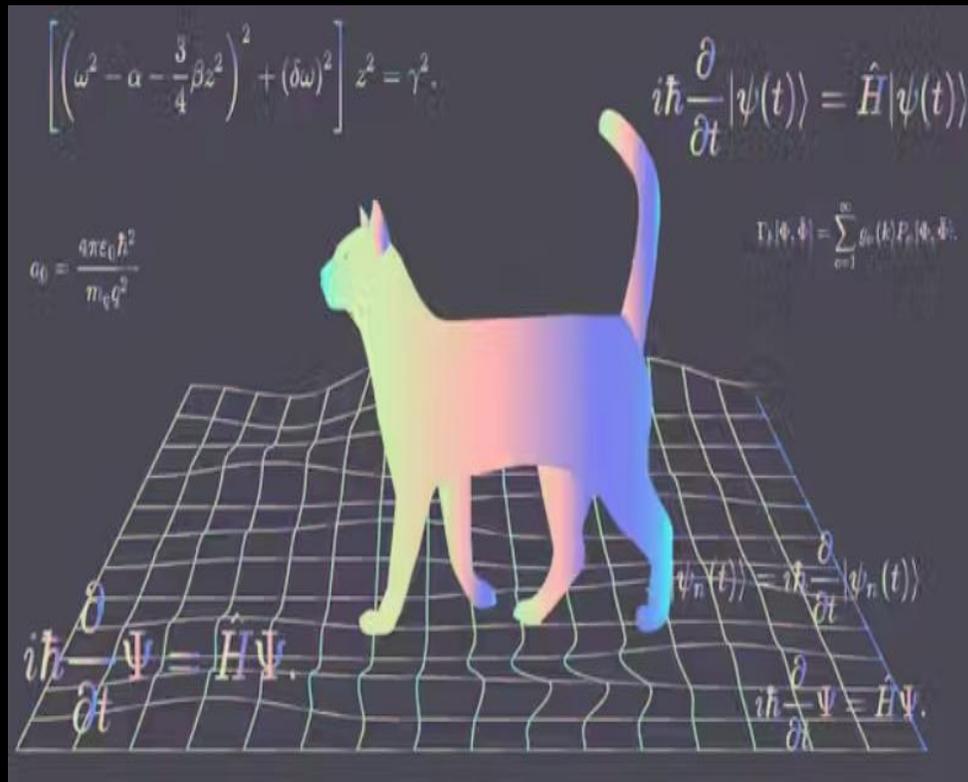
Hybrid Quantum Genetic Algorithm (HQGA, Acampora et al. 2021) is a quantum genetic algorithm following a different philosophy from ours, but even so the results are comparable if the number of merit evaluation is similar.

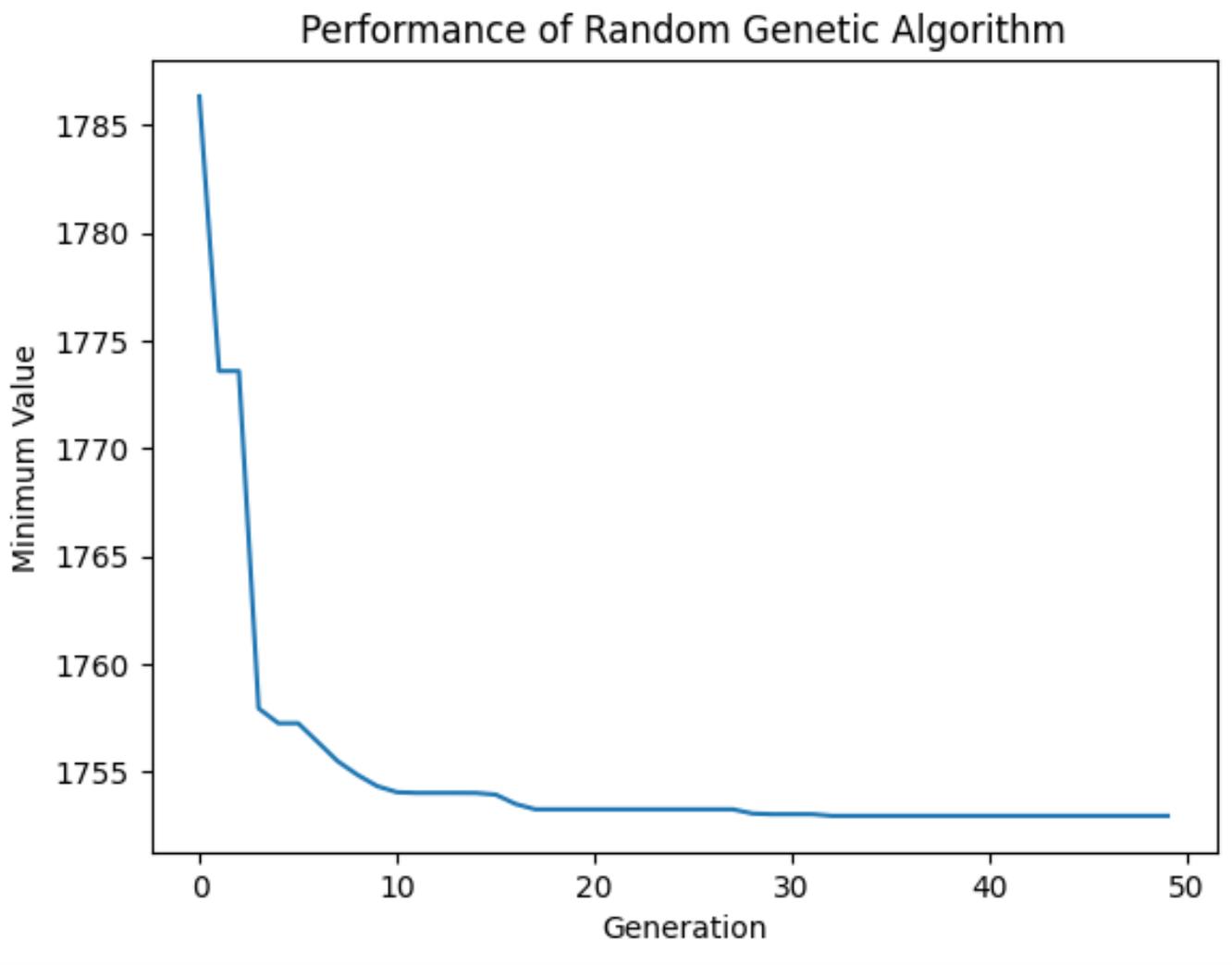


# Conclusions

- 1) The algorithm works.
- 2) An interesting dependence on the crossover and mutation probabilities has been found.
- 3) The first comparisons with classical results give precisions which are comparable (or even better for a naive application for the classical genetic algorithm), similar conclusion also with another quantum genetic algorithm found in the literature.
- 4) Tests on the other hyperparameters (number of population and generation) have also been performed.
- 5) The speed of the algorithm cannot be better than the classical counterparts given the classical fitness evaluation.

# Superposition and Entanglement



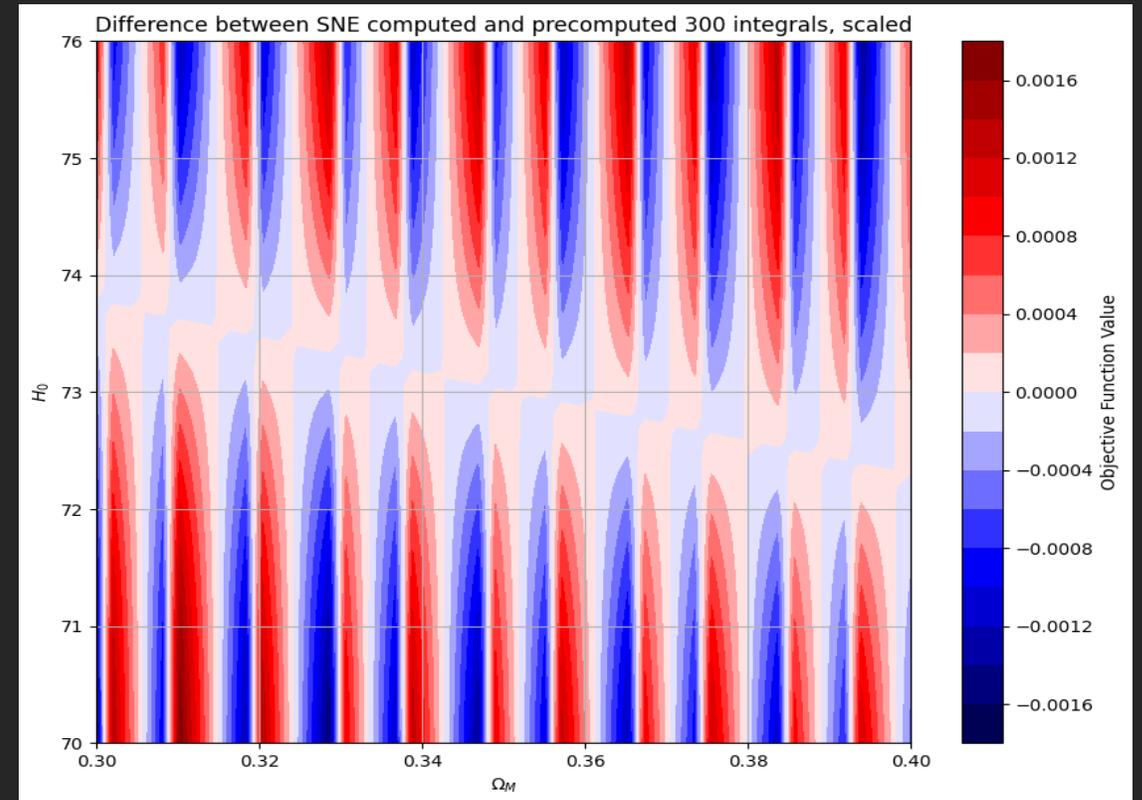
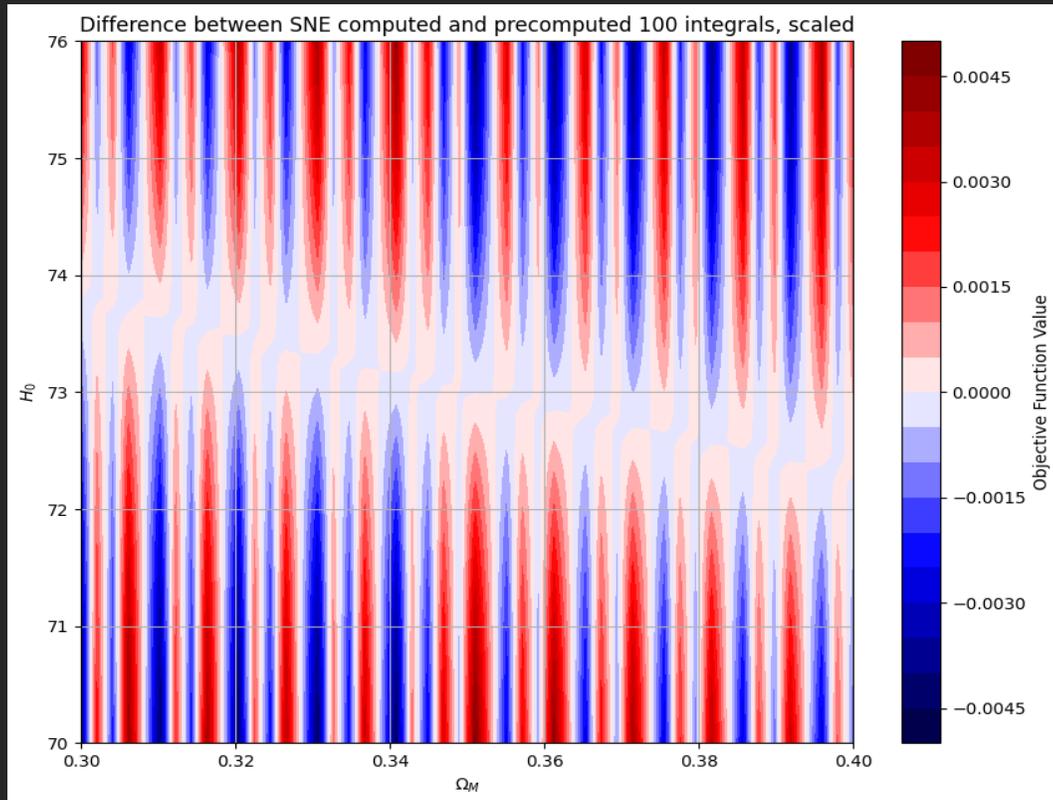


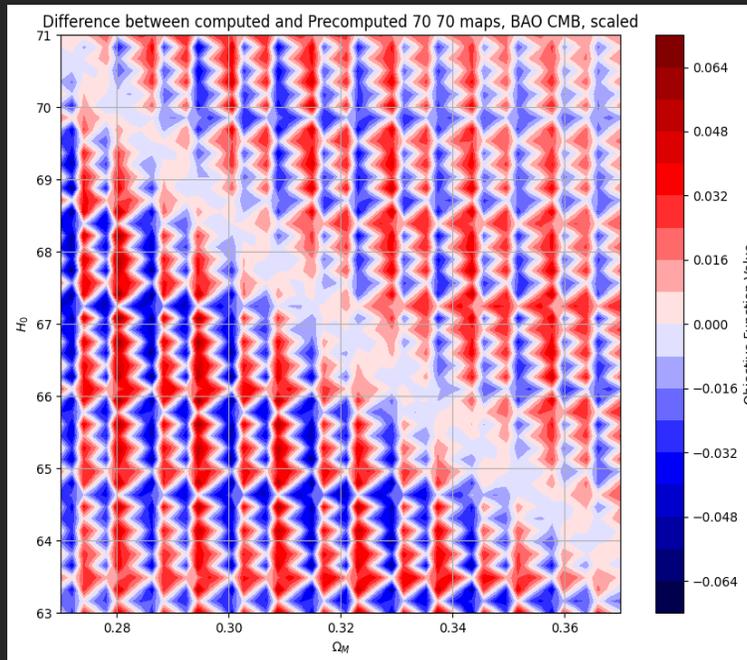
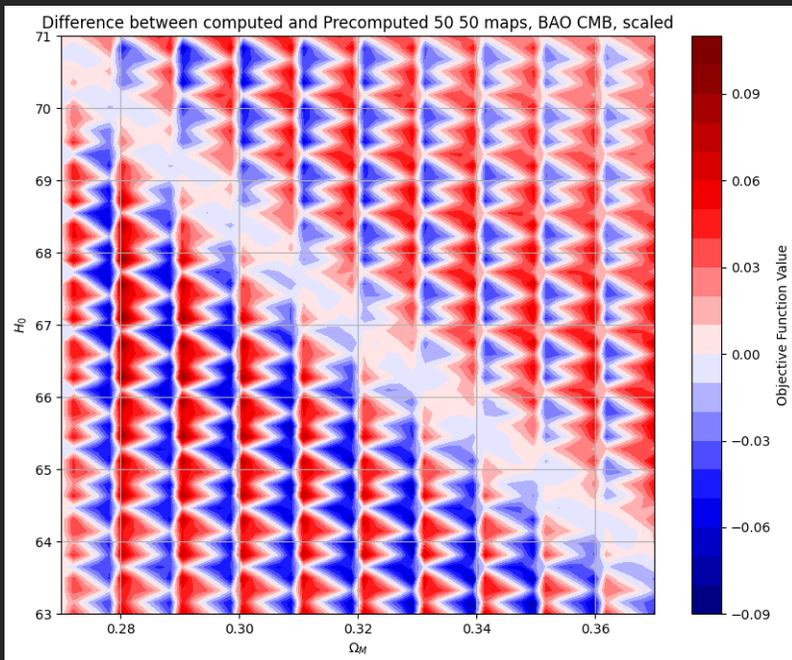
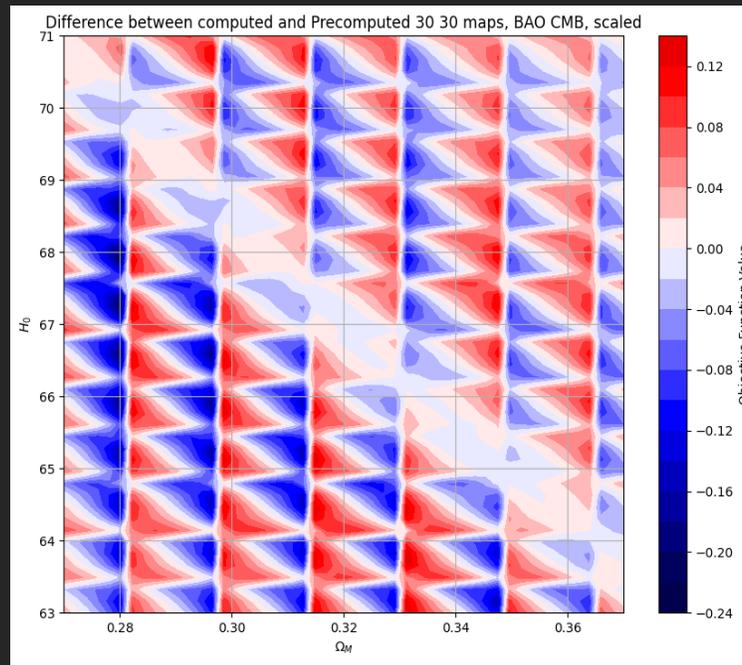
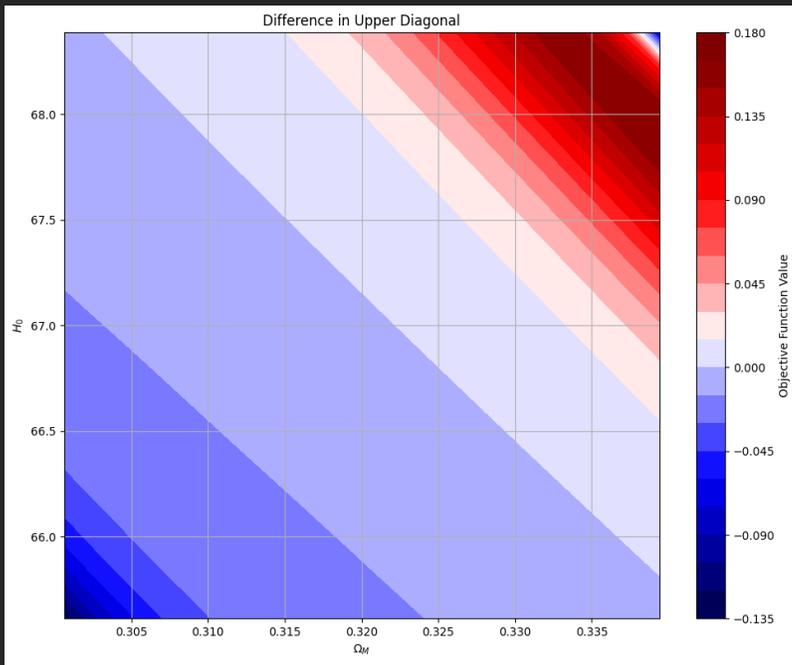
A single iteration of the algorithm for the SNe Ia

---

# Precomputed Integrals, SNE, confronting the Maps

We note the remarkable precision for both the precomputations in the region around the minimum

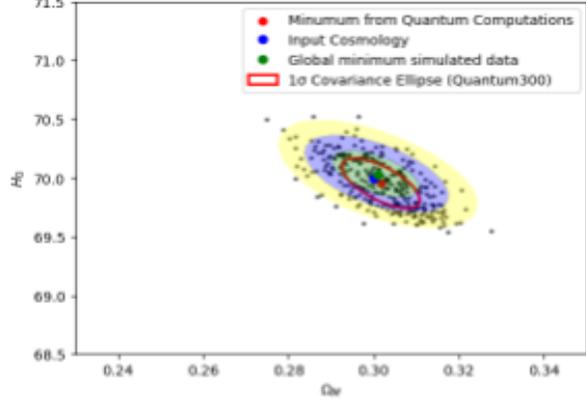




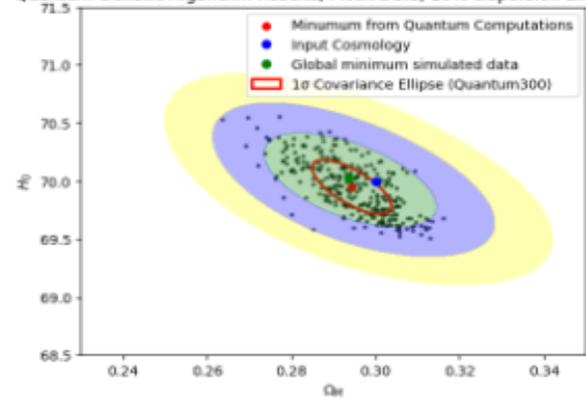
# PICO and precomputed maps, BAO+CMB

Note how the PICO map is more consistent in the region around the minimum of the objective function, while loses accuracy in other zones of the parameter space, which are not represented in the PICO plot

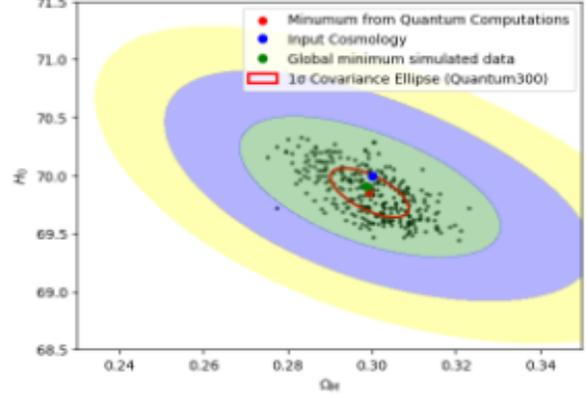
Quantum Genetic Algorithm Results, Mock Data, 5% dispersion and errors



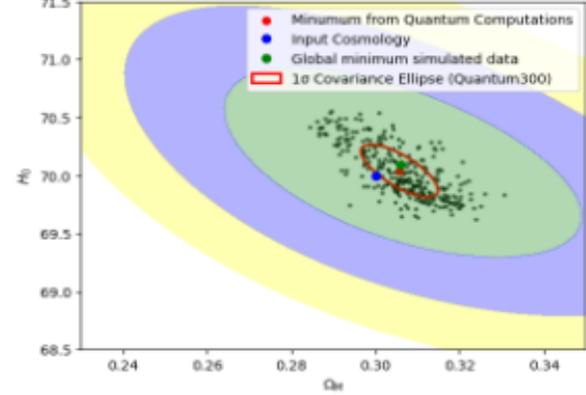
Quantum Genetic Algorithm Results, Mock Data, 10% dispersion and errors



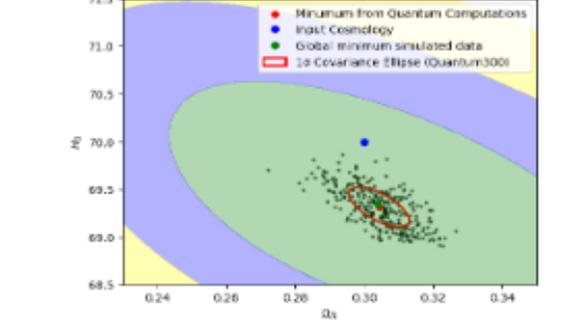
Quantum Genetic Algorithm Results, Mock Data, 15% dispersion and errors



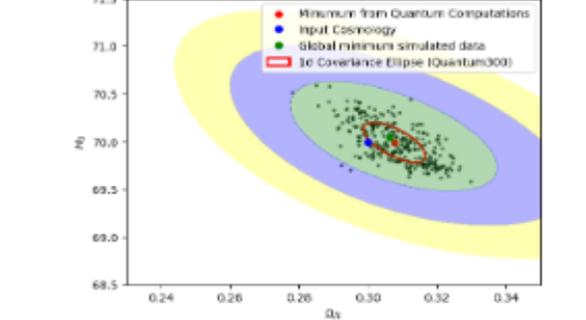
Quantum Genetic Algorithm Results, Mock Data, 20% dispersion and errors



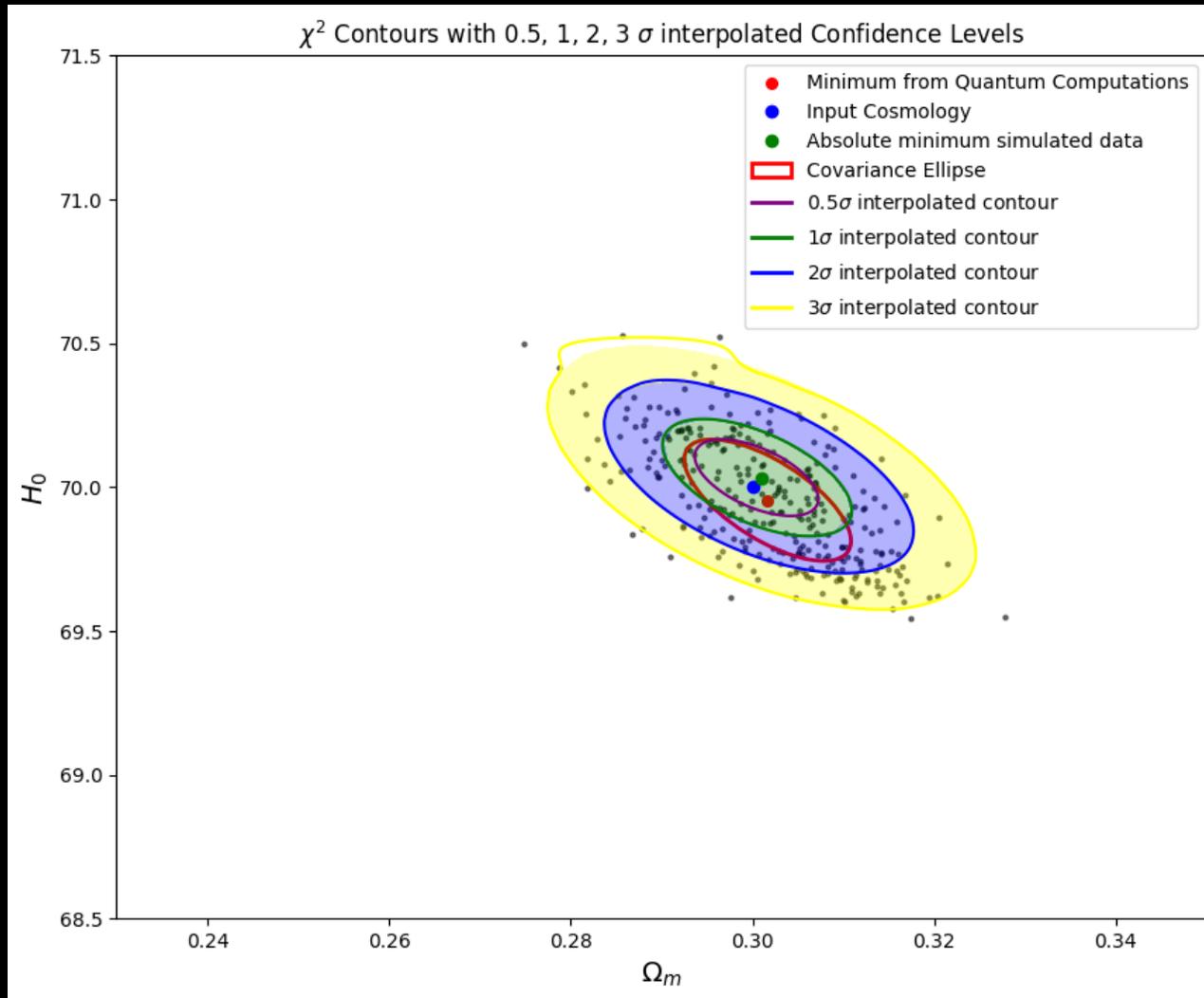
Quantum Genetic Algorithm Results, Mock Data, 10% dispersion and errors, 100 data points



Quantum Genetic Algorithm Results, Mock Data, 10% dispersion and errors, 500 data points



# Mock Data results



# Interpolating the Chi- Squared

---