ÉCOLE NATIONALE DES
PONTS
ET CHAUSSÉES

IP PARIS

PROJET DE PREMIÈRE ANNÉE 2025

ANNEXE

Informatique Quantique

Réalisé par :

Youssef Sahraoui

Pierre Bressollette

Supervisé par :

Henri Pinsolle

Alicia Negre

Contents

1 Quantum Phase Estimation

1.1 Motivation

Après avoir étudié le formalisme mathématique de l'informatique quantique, il est temps de s'intéresser à des algorithmes concrets, comme **l'estimation de phase quantique** `QPE`. Cet algorithme est essentiel car il permet d'estimer avec précision la phase d'un opérateur unitaire, c'est-à-dire ses valeurs propres de module `1`, ce qui est crucial pour de nombreuses applications en calcul quantique.

Pourquoi s'intéresser à `QPE` ? Parce que simplement il est au cœur de plusieurs algorithmes importants, comme :

- **L'algorithme de Shor**, qui permet de factoriser des nombres entiers et pourrait remettre en cause la sécurité des systèmes `RSA`.
- **La résolution des systèmes linéaires quantiques**, notamment avec l'algorithme `HHL`, qui pourrait accélérer la résolution des **EDP** et avoir des applications en optimisation.

1.2 Contexte

Soit $\mathcal{U}$ un opérateur unitaire et $|\psi\rangle$ un vecteur propre de $\mathcal{U}$. Comme les valeurs propres des opérateurs unitaires sont de module 1, l'objectif est de déterminer la phase de cette valeur propre :

$$\mathcal{U}\,|\psi\rangle = e^{i2\pi\phi}\,|\psi\rangle\,, \quad \phi \in [0,1[.$$

1.3 1ère Approche : Test de Hadamard

Connaissant que $\langle\psi|\,\mathcal{U}\,|\psi\rangle = e^{i2\pi\phi}$, le test de Hadamard a pour but principal de trouver $\langle\psi|\,\mathcal{U}\,|\psi\rangle$.

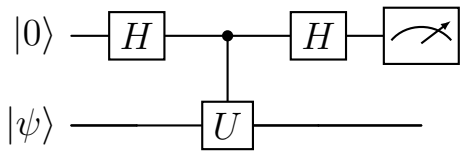


Figure 1. Test de Hadamard pour $\mathrm{Re}(\langle\psi|U|\psi\rangle)$.

Description de ce circuit

Le circuit commence avec l'état initial $|0\rangle|\psi\rangle$. L'application d'une porte Hadamard sur le premier qubit et l'identité sur le second donne :

$$|0\rangle|\psi\rangle \xrightarrow{H\otimes I} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle$$

Ensuite, l'opération c-U (porte contrôlée U) est appliquée, ce qui donne :

$$\xrightarrow{\text{c-}U} \frac{1}{\sqrt{2}}\left(|0\rangle|\psi\rangle + |1\rangle U|\psi\rangle\right)$$

Enfin, une deuxième porte Hadamard est appliquée sur le premier qubit, donnant l'état final :

$$\xrightarrow{H\otimes I} \frac{1}{2}|0\rangle \otimes \left(1 + e^{i2\pi\phi}\right)|\psi\rangle + \frac{1}{2}|1\rangle \otimes \left(1 - e^{i2\pi\phi}\right)|\psi\rangle$$

Ainsi, la probabilité de mesurer $|0\rangle$ dans le premier qubit est donnée par :

$$p(0) = \frac{1}{2}\left(1 + \mathrm{Re}\left(\langle\psi|U|\psi\rangle\right)\right)$$

Reste à mesurer $\mathrm{Im}\left(\langle\psi|U|\psi\rangle\right)$, ce qui peut être fait à l'aide du test suivant, appelé *test de Hadamard imaginaire* :
Notons S la matrice suivante, appelée porte de phase :

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

Le circuit correspondant est le suivant :

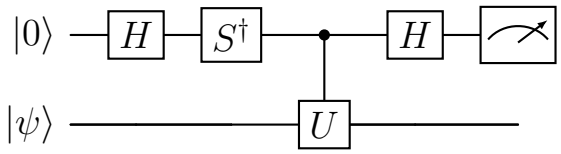


Figure 2. Test de Hadamard pour $\mathrm{Im}(\langle\psi|U|\psi\rangle)$.

Un calcul similaire à celui du *test de Hadamard réel* montre que ce circuit transforme l'état $|0\rangle|\psi\rangle$ en l'état suivant :

$$\frac{1}{2}|0\rangle\left(|\psi\rangle - iU|\psi\rangle\right) + \frac{1}{2}|1\rangle\left(|\psi\rangle + iU|\psi\rangle\right)$$

Ainsi, la probabilité de mesurer le qubit dans l'état $|0\rangle$ est donnée par :

$$p(0) = \frac{1}{2}\left(1 + \mathrm{Im}\left(\langle\psi|U|\psi\rangle\right)\right)$$

En combinant les résultats des deux circuits, on obtient l'estimation de $\langle\psi|U|\psi\rangle$.

Problème :

À première vue, il semble que cette méthode soit efficace.
Cependant, si l'on souhaite obtenir une grande précision sur la phase ϕ, il est nécessaire d'avoir un grand nombre d'échantillons afin que la loi forte des grands nombres nous fournisse un résultat satisfaisant.

Ordre de Grandeur

L'incertitude sur la mesure est : $\Delta p \sim \frac{1}{\sqrt{N_e}}$, où p est la probabilité que l'on souhaite mesurer.
Pour une précision de l'ordre de ϵ sur ϕ, il nous faut $N_e = \mathcal{O}\left(\frac{1}{\epsilon^4}\right)$.

1.4 2ème Approche : Quantum Fourier Transform `QFT`

1.4.1 Définition

Pour tout état $|j\rangle$ dans la base de $\mathcal{H}$, la transformée de Fourier discrète est définie par :

$$U_{\mathrm{FT}}|j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{i2\pi\frac{kj}{N}}|k\rangle\,.$$

L'inverse de la transformée de Fourier discrète est donné par :

$$U_{\mathrm{FT}}^{\dagger}|j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{-i2\pi\frac{kj}{N}}|k\rangle\,.$$

L'opérateur de la transformée de Fourier est unitaire

$$U_{\mathrm{FT}}^{\dagger}U_{\mathrm{FT}} = I$$

Représentation binaire des indices

On peut représenter k et j sous forme binaire :

$$k = (k_{n-1}k_{n-2}\dots k_0)^{(2)}, \quad j = (j_{n-1}j_{n-2}\dots j_0)^{(2)}.$$

On introduit les notations suivantes :

$$(j_{n-1}j_{n-2}\dots j_l \times \dots j_0) = \sum_{k=0}^{n-1} \frac{j_k}{2^{l-k}}$$

$$(\times j_{l-1}\dots j_0) = \sum_{k=0}^{l-1} \frac{j_k}{2^{l-k}}$$

On a ainsi :

$$\frac{kj}{N} = k_0 + k_1 \cdot \frac{j}{2^n} + k_2 \cdot \frac{j}{2^{n-1}} + \dots + k_{n-1} \cdot \frac{j}{2}$$

$$= k_0(\times j_{n-1}j_{n-2}\dots j_0) + k_1(j_{n-1} \times j_{n-2}\dots j_0) + \dots + k_{n-1}(j_{n-1}j_{n-2}\dots j_1 \times j_0).$$

On exprime l'exponentielle comme :

$$e^{i2\pi\frac{kj}{N}} = e^{i2\pi k_0(\times j_{n-1}\dots j_0)} e^{i2\pi k_1(\times j_{n-2}\dots j_0)} \dots e^{i2\pi k_{n-1}(\times j_0)}$$

L'application de la `QFT` sur un état $|j\rangle$ donne :

$$U_{\mathrm{FT}} |j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{i2\pi\frac{kj}{N}} |k\rangle .$$

Décomposons cette expression sous forme de tenseur :

$$|k\rangle = \bigotimes_{m=0}^{n-1} |k_m\rangle = |k_{n-1}\rangle |k_{n-2}\rangle \dots |k_0\rangle$$

$$U_{\mathrm{FT}} |j\rangle = \frac{1}{\sqrt{N}} \bigotimes_{m=0}^{n-1} \left(|0\rangle + e^{i2\pi(\times j_m j_{m-1}\dots j_0)} |1\rangle\right).$$

Cette transformation peut être réalisée par une série de rotations contrôlées de la forme :

$$|0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\times j_{n-1}\dots j_0)} |1\rangle).$$

1.5 Circuit Quantique de la QFT

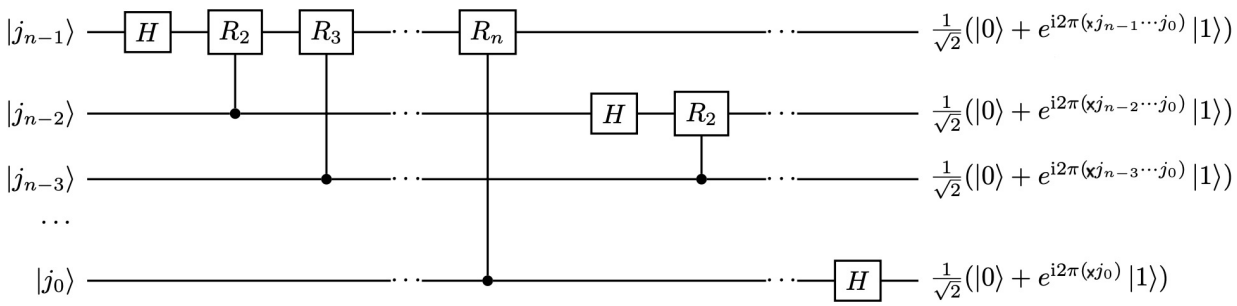


Figure 1: Circuit général pour une QFT sur n *qubits*

Le circuit général pour une QFT sur n qubits suit le schéma :

1. Appliquer une porte de Hadamard H sur le premier qubit.
2. Appliquer des rotations contrôlées $R_k = R_z(2\pi/2^k)$ sur les autres qubits.
3. Répéter pour chaque qubit avec les rotations adéquates.
4. Appliquer des portes SWAP pour inverser l'ordre des qubits.

avec

$$R_z(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$$

1.6 Principe de l'Estimation de Phase avec la QFT

Pour estimer notre phase ϕ associée à $\mathcal{U}$, on suit les étapes suivantes :
Comme $\phi \in [0, 1[$, elle peut être représentée en représentation binaire. Le nombre de bits choisi reflétera la précision souhaitée pour cette valeur propre.

Étapes Clés de l'algorithme :

1. **Préparation de l'état initial** :
   - Un registre de t qubits appelé qubits ancillaires initialisés à $|0\rangle$ est utilisé pour stocker l'estimation de ϕ.
   - L'état initial est $|0^{\otimes t}\rangle|\psi\rangle$.
2. **Application de la QFT inverse** :

- La `QFT` inverse $U_{\mathrm{FT}}^{\dagger}$ est appliquée pour convertir les rotations de phase en une représentation binaire.

3. **Mesure du registre** :
   - La mesure donne une valeur entière $k \approx \phi \cdot 2^t$.
   - La phase estimée est $\widetilde{\phi} = k/2^t$.

Précision et Nombre d'Échantillonnages

La précision $\epsilon \sim 2^{-d}$ de l'estimation de ϕ et la probabilité de succès $1 - \delta$ déterminent les ressources nécessaires pour l'algorithme :

$$t = d + \lceil \log_2(1/\delta) \rceil \qquad \text{(nombre de qubits du registre)}$$

2 Application aux `EDP` : *cas de l'équation de diffusion-advection*

$$\frac{d\phi}{dt} = \frac{\partial \phi}{\partial t} + \mathbf{v} \cdot \nabla \phi = D \nabla^2 \phi \tag{1}$$

- $\frac{d}{dt} = \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla$: désigne ici la dérivée particulaire.
- $\phi(\mathbf{x}, t)$: Grandeur scalaire à laquelle on s'intéresse.
- D : Constante de diffusivité.
- v : Vitesse de convection.

2.1 Généralités

Cette équation décrit le transport d'un scalaire ϕ advecté par un écoulement incompressible de vitesse v, dans un milieu de diffusivité D.

2.2 Discrétisation de l'équation de diffusion

On commence par discrétiser l'équation de diffusion en utilisant la méthode des différences centrées pour l'opérateur de dérivation spatiale, ce qui introduit une erreur d'approximation de l'ordre $\mathcal{O}(\Delta x^2)$.

L'opérateur de dérivation temporelle peut être également modélisé à l'aide

d'une différence avant *(forward difference)* qui introduit une erreur d'approximation de l'ordre $\mathcal{O}(\Delta t)$.

$$\frac{\partial \phi}{\partial x}(x,t) \approx \frac{\phi(x+\Delta x,t) - \phi(x-\Delta x,t)}{2\Delta x}$$

$$\frac{\partial^2 \phi}{\partial x^2}(x,t) \approx \frac{\phi(x+\Delta x,t) - 2\phi(x,t) + \phi(x-\Delta x,t)}{(\Delta x)^2}$$

$$\frac{\partial \phi}{\partial t}(x,t) \approx \frac{\phi(x,t+\Delta t) - \phi(x,t)}{\Delta t}$$

Soit l'intervalle spatial $[a,b]$ dans lequel varie le scalaire ϕ, où a et b sont choisis selon le problème étudié. On discrétise cet intervalle en N_x points espacés uniformément, ce qui donne :

$$x_m = a + m\,\Delta x, \quad \text{avec} \quad \Delta x = \frac{b-a}{N_x - 1}, \quad m = 0, 1, \ldots, N_x - 1.$$

De manière analogue, on considère une discrétisation de l'intervalle temporel $[0, t_f]$ en N_t points également espacés. On définit ainsi les instants discrets par :

$$t_i = i\,\Delta t, \quad \text{où} \quad \Delta t = \frac{t_f}{N_t - 1}, \quad i = 0, 1, \ldots, N_t - 1.$$

On adopte la notation suivante pour les valeurs discrètes de ϕ :

$$\phi_{m,i} = \phi(x_m, t_i)$$

où x_m et t_t sont les points de discrétisation spatiale et temporelle respectivement.

Pour chaque instant t, on introduit le vecteur colonne des valeurs spatiales de ϕ :

$$\phi_t = \begin{bmatrix} \phi_{0,t} \\ \phi_{1,t} \\ \vdots \\ \phi_{N_x-1,t} \end{bmatrix} \in \mathbb{R}^{N_x}$$

En utilisant ces notations, on discrétise l'équation (1).

$$\forall m, t \quad \frac{\phi_{m,t+1} - \phi_{m,t}}{\Delta t} + v \cdot \frac{\phi_{m+1,t} - \phi_{m-1,t}}{2\Delta x} = D \cdot \frac{\phi_{m+1,t} - 2\phi_{m,t} + \phi_{m-1,t}}{(\Delta x)^2}$$

En isolant $\phi_{m,t+1}$, on obtient :

$$\phi_{m,t+1} = \left(r_h + \frac{r_a}{2}\right)\phi_{m-1,t} + (1-2r_h)\phi_{m,t} + \left(r_h - \frac{r_a}{2}\right)\phi_{m+1,t}$$

avec les paramètres adimensionnels suivants :

$$r_h = \frac{D\Delta t}{(\Delta x)^2}, \quad r_a = \frac{v\Delta t}{\Delta x}$$

Ce schéma peut être exprimé en notation vectorielle sous la forme :

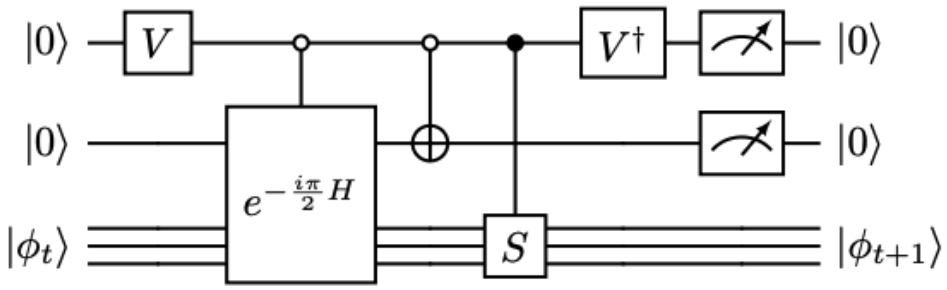
$$\phi_{t+1} = A\phi_t$$

où $A \in \mathbb{R}^{N_x \times N_x}$ est une matrice *circulante* définie par :

$$A = \begin{bmatrix} 1-2r_h & r_h - \frac{r_a}{2} & 0 & \cdots & r_h + \frac{r_a}{2} \\ r_h + \frac{r_a}{2} & 1-2r_h & r_h - \frac{r_a}{2} & \ddots & \vdots \\ 0 & r_h + \frac{r_a}{2} & 1-2r_h & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & r_h - \frac{r_a}{2} \\ r_h - \frac{r_a}{2} & \cdots & 0 & r_h + \frac{r_a}{2} & 1-2r_h \end{bmatrix}$$

3 Implémentation des portes du circuit

3.1 Schéma du circuit quantique



3.2 Implémentation de la porte S

Décomposition de la permutation circulaire via la QFT

Soit $N = 2^n$, pour un entier n, et soit $\omega = e^{2\pi i/N}$, une racine N-ième de l'unité.

On considère la matrice de permutation circulaire S, définie par son action sur la base computationnelle :

$$S|j\rangle = |(j+1) \bmod N\rangle, \quad \text{pour } j = 0, 1, \ldots, N-1.$$

Nous allons montrer que cette matrice peut s'écrire comme :

$$P = F^\dagger D F,$$

où :

- F est la Quantum Fourier Transform (QFT), unitaire, définie par :

$$F|x\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{xk} |k\rangle.$$

- $F^\dagger$ est l'inverse de F, donc :

$$F^\dagger|k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega^{-jk} |j\rangle.$$

- D est une matrice diagonale dans la base de Fourier, donnée par :

$$D = \sum_{k=0}^{N-1} \omega^k |k\rangle\langle k|.$$

Appliquons la `QFT` à $|j\rangle$:

$$F|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{jk} |k\rangle.$$

On applique ensuite D au vecteur obtenu :

$$DF|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{jk} \cdot \omega^k |k\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{k(j+1)} |k\rangle = F\,|j+1\rangle$$

Comme F est ***unitaire*** il suffit de multiplier par $F^\dagger$ à gauche.
Donc :

$$F^\dagger D F|j\rangle = |(j+1) \bmod N\rangle.$$

Conclusion

On a montré que :

$$F^\dagger D F|j\rangle = S|j\rangle,$$

donc :

$$S = F^\dagger D F,$$

avec :

$$D = \sum_{k=0}^{N-1} \omega^k |k\rangle\langle k|, \quad \omega = e^{2\pi i/N}.$$

Ainsi, la permutation circulaire S peut être réalisée en trois étapes quantiques :

1. Appliquer la QFT,
2. Appliquer D qu'on sait construire à l'aide de porte de rotation.
3. Appliquer la QFT inverse.

3.3 Implémentation de la porte $e^{-i\frac{\pi}{2}H}$

Dans de nombreux algorithmes quantiques, en particulier en simulation de systèmes physiques, on souhaite implémenter une porte de la forme :

$$U(t) = e^{-iHt},$$

où H est un opérateur hermitien (souvent un Hamiltonien) agissant sur n qubits. Cette opération est une évolution unitaire, correspondant à l'évolution d'un système quantique selon l'équation de Schrödinger.

L'objectif est de construire un circuit quantique réalisant cette transformation, en utilisant uniquement des portes quantiques élémentaires.

Décomposition de Pauli

Tout opérateur hermitien $A \in \mathbb{C}^{N\times N}$ peut s'exprimer comme combinaison linéaire de produits tensoriels de matrices de Pauli :

$$A = \sum_j \alpha_j P_j,$$

où :

- chaque $\alpha_j \in \mathbb{R}$,
- $P_j \in \mathcal{P}_n = \{I, X, Y, Z\}^{\otimes n}$ est un *Pauli string* (produit tensoriel de matrices de Pauli).

Par exemple, pour 2 qubits :

$$A = \alpha_0 I \otimes I + \alpha_1 X \otimes Z + \alpha_2 Z \otimes Z + \dots$$

Exponentielle d'une somme : Trotterisation

Puisque les P_j ne commutent généralement pas entre eux, on ne peut pas directement écrire :

$$e^{-iAt} = \prod_j e^{-i\alpha_j P_j t}.$$

Cependant, on peut utiliser une approximation connue sous le nom de ***Trotter-Suzuki*** [1]:

Formule de Trotter d'ordre 1

Soit :

$$A = \sum_{j=1}^{m} A_j,$$

alors pour $r \in \mathbb{N}$, on a l'approximation suivante :

$$e^{-iAt} \approx \left(\prod_{j=1}^{m} e^{-iA_j t/r} \right)^r,$$

qui devient exacte dans la limite $r \to \infty$. Cette approximation est appelée *Trotterisation d'ordre 1.*

En appliquant cela à la décomposition de Pauli :

$$e^{-iAt} = e^{-i\sum_j \alpha_j P_j t} \approx \left(\prod_j e^{-i\alpha_j P_j t/r} \right)^r.$$

Chaque facteur $e^{-i\alpha_j P_j t/r}$ peut être implémenté efficacement en circuit quantique.

4 Partie numérique

4.1 Préparation des états

$$\forall t, \quad \phi_{t+1} = A\phi_t$$

Cependant, lors de l'implémentation, il est nécessaire que ϕ_t soit normalisé. En effet, on a :

$$\| |00\rangle |\phi_t\rangle \|^2 = \langle 0| \langle 0| \langle \phi_t| |\phi_t\rangle |0\rangle |0\rangle = \|\phi_t\|^2 = 1$$

À la sortie du circuit (comme indiqué dans le *rapport final*), on obtient l'état suivant :

$$|0\rangle |0\rangle A |\phi_t\rangle + |1\rangle |0\rangle *$$

Après l'étape de post-sélection, on parvient à extraire le vecteur $A|\phi_t\rangle$, avec $|\phi_t\rangle$ de norme unitaire.

Ainsi, pour récupérer le vecteur ϕ_{t+1} correctement normalisé, on écrit :

$$\phi_{t+1} = \|\phi_t\| \cdot A \left(\frac{\phi_t}{\|\phi_t\|} \right) = \|\phi_t\| \cdot A |\phi_t\rangle$$

Il est donc crucial de ne pas oublier de multiplier par la norme de ϕ_t à la fin.

4.2 Implémentation Python

4.2.1 Librairies utilisées

```python
import numpy as np
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.circuit.library import UnitaryGate, XGate
from qiskit.quantum_info import Statevector
from scipy.linalg import expm
import matplotlib.pyplot as plt
import math
```

Figure 2: Import des librairies nécessaires

4.2.2 Paramètres initiaux

```python
# Paramètres

N_x = 2**9
N_t = 100000
velocity = 1
D = 2.2e-5
delta_T = 1 / (N_t - 1)
delta_x = 1 / (N_x - 1)
rh = D * delta_T / (delta_x**2)
ra = velocity * delta_T / delta_x
alpha = (rh + ra / 2) / (1 - 2 * rh)
```

Figure 3: Définition des paramètres

4.2.3 Structure du circuit

```python
# Matrices - Portes
V = np.array([
    [np.sqrt(1 - 2 * rh), -np.sqrt(2 * rh)],
    [np.sqrt(2 * rh), np.sqrt(1 - 2 * rh)]
], dtype=complex)
V_gate = UnitaryGate(V, label="V")
V_dag = UnitaryGate(V.conj().T, label="V†")

hat_A = np.eye(N_x, dtype=complex)
for i in range(N_x):
    if i > 0:
        hat_A[i, i - 1] = -alpha
    if i < N_x - 1:
        hat_A[i, i + 1] = alpha
hat_A[N_x - 1, 0] = -alpha
hat_A[0, N_x - 1] = alpha

H = np.block([
    [np.zeros((N_x, N_x), dtype=complex), -1j * hat_A.conj().T],
    [1j * hat_A, np.zeros((N_x, N_x), dtype=complex)]
])
U = expm(-1j * np.pi / 2 * H)
U_gate = UnitaryGate(U, label="exp(-iπ/2·H)")

S = np.roll(np.eye(N_x), 1, axis=1)
S_gate = UnitaryGate(S, label="S")

# Registres
n_data = math.ceil(np.log2(N_x))
anc = QuantumRegister(2, 'anc')
data = QuantumRegister(n_data, 'phi_t')
cl = ClassicalRegister(2, 'c')

# ajout des composantes
qc = QuantumCircuit(anc, data, cl)
qc.append(V_gate, [anc[0]])
qc.append(U_gate.control(1, ctrl_state='0'), [anc[0], anc[1]] + data[:])
qc.append(XGate().control(1, ctrl_state='0'), [anc[0], anc[1]])
qc.append(S_gate.control(), [anc[0]] + data[:])
qc.append(V_dag, [anc[0]])
```

Figure 4: Construction du circuit quantique

4.2.4 Évolution temporelle

```python
# État initial
x_values = np.linspace(0, 1, N_x)
phi_0 = np.sin(2*np.pi * x_values)
phi_t_list = [phi_0]
# Évolution
for t in range(1, 2): #On pourra remplacer 2 par N_t pour une évolution complète
    phi = phi_t_list[-1]
    norm_phi = np.linalg.norm(phi)
    state_ancilla = [1] + [0]*(2**2 - 1)
    state_total = np.kron(state_ancilla, phi / norm_phi)
    sv = Statevector(state_total)

    # évolution par le circuit
    full_reg = QuantumRegister(2 + n_data)
    sv = sv.evolve(qc)
    # post-sélection sur les états avec ancilla = |00⟩
    phi_next = []
    for i, amp in enumerate(sv.data):
        bin_str = format(i, f'0{2 + n_data}b')
        if bin_str[:2] == '00':
            phi_next.append(amp)
    phi_next = np.array(phi_next[:N_x])
    phi_t_list.append(phi_next*norm_phi)
```

Figure 5: Boucle d'évolution du système

4.2.5 Solution classique

```python
t_values = np.linspace(0, 1, N_t)
phi0 = np.sin(2*np.pi * x_values)
# Solution dns le cas classique
solution_classique = [phi0]
for t in range(1, 2):
    phi_new = np.roll(phi0, -1) * (rh + ra/2) + phi0 * (1 - 2*rh) + np.roll(phi0, 1) * (rh - ra/2)
    solution_classique.append(phi_new)
    phi0 = phi_new
```

Figure 6: Implémentation de référence

4.2.6 Visualisation

```python
# Tracé
normL2 = []
normLinf = []
plt.figure(figsize=(10, 6))
for t in range(1, 2):
    plt.plot(x_values, solution_classique[t], linestyle='-', label="solution classique")
    plt.plot(x_values, phi_t_list[t], linestyle=':', label="solution quantique")
    normL2.append(np.linalg.norm(solution_classique[t]-phi_t_list)/np.linalg.norm(solution_classique[t]))
    normLinf.append(np.max(np.abs(solution_classique[t]-phi_t_list))/np.max(np.abs(solution_classique[t])))

print("Les erreurs relatives en norme L2 : ", normL2)
print("Les erreurs relatives en norme Linf : ", normLinf)
plt.title("Équation de convection-diffusion 1D avec conditions de périodiques")
plt.xlabel("x")
plt.ylabel(r"$\phi(x,t)$")
plt.grid(True)
plt.legend()
plt.show()
```

Figure 7: Code de tracé des résultats

Résultats numériques

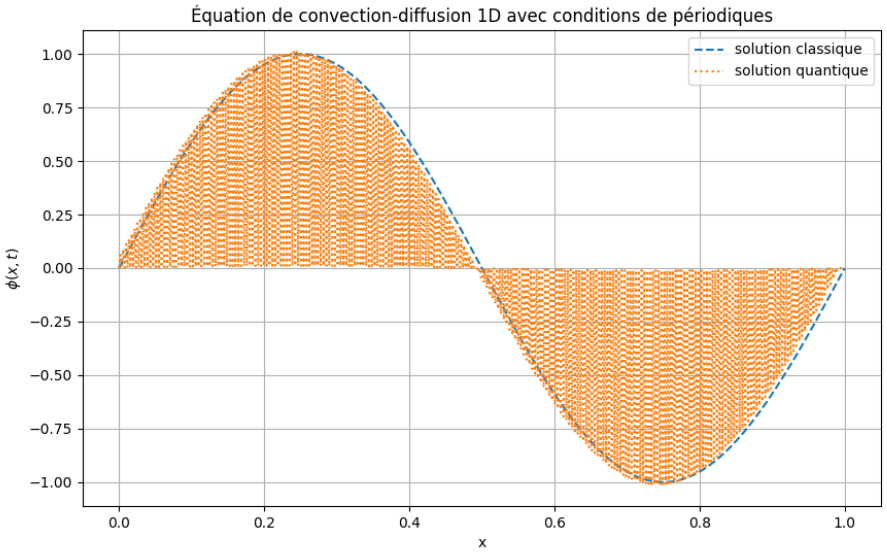


Figure 8: Comparaison des solutions quantique et classique

References

[1] V.Londe, *Trotterisation*, December 2020. https://vivienlonde.github.io/blog/trotterisation#:~:text=Les%20formules%20de%20Trotter%2DSuzuki,apparaissent%20dans%20un%20ordre%20diffÃľrent.