Chapitre IV : Implémentation et application

IV.1. Introduction

IV.2. Environnement de développement

IV.3. Environnement logiciel

IV.4. Simulation quantique

IV.5. Exemples illustratifs

IV.1. Introduction :

Ce chapitre constitue la dernière partie de ce travail. C'est une démonstration numérique de notre correcteur quantique. Pour ce faire, nous introduisons d'abord les différentes fonctions implémentées pour simuler un calculateur quantique. Ensuite, nous présentons quelques études de cas afin de valider l'efficacité des procédures de détection et de correction. Pour plus de clarté, tous les états intermédiaires des registres quantiques manipulés seront affichés.

IV.2. Environnement de développement :

La machine utilisée dans notre implémentation est caractérisée par :

- ✓ Processeur : Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz.
- ✓ Mémoire installée (RAM) : 4,00 GO.
- ✓ Disque dur : 500 GO.
- ✓ Type du système : système d'exploitation 64 bits.

IV.3. Environnement logiciel :

Java : le langage java possède de nombreux points forts qui expliquent pourquoi c'est l'un des principaux langages de programmation utilisés en entreprise aujourd'hui :

- ✓ Langage statique et détaillé - Grâce à la nature statique et robuste de Java, il est facile à maintenir et à lire. Java permet de renvoyer plusieurs types de données et on peut facilement les utiliser dans diverses applications de même domaine.
- ✓ Grande compatibilité avec les outils open source, des applications, des serveurs, etc. (Apache)
- ✓ Portabilité, Facile à exécuter, facile à écrire - une fois un code est écrit en java, il est possible de l'exécuter presque n'importe où et à tout moment. on peut l'écrire sous Windows et l'exécuter sous Linux, sur des serveurs / téléphones mobiles, etc. avec le même comportement. C'est la force de la pierre angulaire de Java.

JDK : nécessaire pour le développement et l'implémentation de notre application. Java Développement Kit, est le kit de développement qui offre un ensemble d'outils permettant de développer des logiciels et applications java et qui est proposé gratuitement par Oracle.

NetBeans: Un environnement de développement intégré avec un ensemble d'outils, venant aider les programmeurs, spécialisé dans le domaine Java [19].

IV.4. Simulation quantique :

Nous avons développé une version simplifiée d'un simulateur quantique. Toutes les fonctionnalités requises dans notre travail sont implémentées. Les classes principales proposées sont les suivantes :

- ✓ **Etat_Base :** Cette classe concerne la création et la manipulation des états élémentaires.

  Chaque objet de cette classe se caractérise par les attributs suivants :

  - Vecteur : chaine de caractère.
  - Param : le coefficient de cet état élémentaire, représenté comme un réel.

  Les méthodes de cette classe sont :

  - Constructeur (Param , Vect).
  - Afficher ( ).

- ✓ **Etat_Quantique :** Cette classe sert à la manipulation des états quantiques. Il s'agit d'un vecteur contenant la combinaison linéaire des états de la base élémentaires. Pour des fins d'optimisation, les éléments avec des coefficients nuls sont ignorés. Cette classe n'a pas d'attributs mais elle est munie des méthodes suivantes :

  - Constructeur (alpha , Beta).
  - ProduitTensoriel (Etat_Quant_01 , Etat_Quant_02).
  - X (PositionQubitCible).
  - Y (PositionQubitCible).
  - Z (PositionQubitCible).
  - H (PositionQubitCible).
  - CZ (PositionQubitControle , PositionQubitCible).
  - Mesure(PositionQubitCible).
  - Afficher ( ).

Ces méthodes sont implémentées en se basant sur les pseudo-codes qui suivent :

```
ProduitTensoriel ( EtatQ1 : EtatQuantique, EtatQt2 : EtatQuantique ) : Etat_Quantique

Début :
    Pour ( i=0 ; i < Taille (EtatQ1 ) ) faire :
        Pour ( j = 0 ; j < Taille ( EtatQ2 ) ) faire :
            Param_R = (EtatQ1 [ i ]. param) * (EtatQ [ j ]. param) ;
            Vect_R = (EtatQ1 [ i ]. vect) + (EtatQ2 [ j ]. vect) ;
            Etat_Base_R= Etat_Base (Param_R, Vect_R) ;
            EtatQ_Resultat .ajouter (Etat_Base_R) ;
        Fin Pour.
    Fin Pour.
    Return ( EtatQ_Resultat) ;
Fin.
```

```
OpérateurX (EtatQ : EtatQuantique , PositionQubitCible :entier ) : EtatQuantique

Début :
  Pos_QubitCible = Pos_QubitCible – 1 ;
  Pour ( i = 0 ; i < Taille (EtatQ)) faire :
    vectCourant = Etat_Quant [i] . vect ;
    QubitCible = vect_courant [Pos_QubitCible] ;
    Si (QubitCible = '0') alors :
      nouv_vect = vect_courant [0, Pos_QubitCible];
      nouv_vect = nouv_vect + '1' ;
      nouv_vect = nouv_vect + vect_courant [ Pos_QubitCible + 1 , Taille (vect_courant) ];
    Sinon :
      nouv_vect = vect_courant [0, Pos_QubitCible];
      nouv_vect = nouv_vect + '0' ;
      nouv_vect = nouv_vect + vect_courant [ Pos_QubitCible + 1 , Taille (vect_courant) ];
    Fin Si.
    Nouv_EtatBase = Etat_Base (Etat_Quant [ i ] .param , nouv_vect ) ;
    EtatQ_Resultat. Ajouter (Nouv_EtatBase);
  Fin Pour.
  Return ( EtatQ_Resultat) ;
Fin.
```

OpérateurZ (EtatQ : EtatQuantique , PositionQubitCible : entier) : EtatQuantique

```
Début:
    Pos_QubitCible = Pos_QubitCible – 1 ;
    Pour ( i = 0 ; i < Taille (EtatQ)) faire :
        VecteurCourant = Etat_Quant [i] . vect ;
        NouveauxVecteur = VecteurCourant ;
        QubitCible = VecteurCourant[PositionQubitCible] ;
        Si (QubitCible = '0') alors :
            Nouveaux Paramaitre = EtatQ [ i ] . param ;
        Sinon :
            Nouveaux Paramaitre = - EtatQ [ i ] . param ;
        Fin Si.
        NouveauxEtatBase = Etat_Base (Nouveaux Paramaitre , NouveauxVecteur ) ;
        EtatQ_Resultat. Ajouter (NouveauxEtatBase);
    Fin Pour.
    Return ( EtatQ_Resultat) ;
Fin.
```

OpérateurCNOT (EtatQ : EtatQuantique , PositionCible : entier , PositionControl : entier) : EtatQuantique

```
Début:
    PositionCible = PositionCible – 1 ;
    PositionControl = PositionControl – 1 ;
    Pour ( i = 0 ; i < Taille (EtatQ)) faire :
        Nouveaux_param = EtatQ [ i ] . param ;
        VecteurCourant = EtatQ [i] . vect ;
        QubitControl = VecteurCourant [PositionControl] ;
        Si (QubitControl = '0') alors :
            NouveauxVecteur = VecteurCourant ;
        Sinon :
            Qubit_cible= vect_courant [Pos_QubitCible] ;
            Si (Qubit_cible= '0') alors :
                nouv_vect = vect_courant [0, Pos_QubitCible];
                nouv_vect = nouv_vect + '1' ;
                nouv_vect = nouv_vect + vect_courant [ Pos_QubitCible + 1 , Taille (vect_courant) ] ;
```

```
        Sinon
            nouv_vect = vect_courant [0, Pos_QubitCible];
            nouv_vect = nouv_vect + '1' ;
            nouv_vect = nouv_vect + vect_courant [ Pos_QubitCible + 1 , Taille (vect_courant) ];
        Fin Si.
    Fin Si.
    Nouv_EtatBase = Etat_Base (nouv_par, nouv_vect) ;
    EtatQ_Resultat. Ajouter (Nouv_EtatBase);
Fin Pour.
Return ( EtatQ_Resultat) ;
Fin.
```

OpérateurCZ (EtatQ : EtatQuantique , PositionCible : entier , PositionControl : entier) : EtatQuantique

```
Début :
    Pos_QubitCible = Pos_QubitCible – 1 ;
    Pour ( i = 0 ; i < Taille (EtatQ)) faire :
        VecteurCourant = Etat_Quant [i] . vect ;
        NouveauxVecteur = VecteurCourant ;
        QubitCible = VecteurCourant[PositionCible] ;
        QubitControl = VecteurCourant[PositionControl] ;
        Si (QubitCible = '1') et (QubitCible = '1') alors :
            NouveauxParamaitre = - EtatQ [ i ] . param ;
        Sinon :
            Nouveaux Paramaitre = EtatQ [ i ] . param ;
        Fin Si.
        NouveauxEtatBase = Etat_Base (Nouveaux Paramaitre , NouveauxVecteur ) ;
        EtatQ_Resultat. Ajouter (NouveauxEtatBase);
    Fin Pour.
    Return ( Etat_Qt_Resultat) ;
Fin.
```

```
OpérateurH (EtatQ : EtatQuantique , PositionCible : entier) : EtatQuantique

Début :
    PositionCible = PositionCible – 1 ;
    Pour ( i = 0 ; i < Taille (EtatQ)) faire :
        vect_courant = Etat_Quant [ i ] . vect ;
        nouv_vect = vect_courant [0, Pos_QubitCible];
        nouv_vect_0 = nouv_vect + '0' ;
        nouv_vect_1 = nouv_vect + '1' ;
        nouv_vect_0 = nouv_vect + vect_courant [ Pos_QubitCible + 1 , Taille (vect_courant) ];
        nouv_vect_1 = nouv_vect + vect_courant [ Pos_QubitCible + 1 , Taille (vect_courant) ];
        Qubit_cible = vect_courant [Pos_QubitCible] ;
        Si (QubitCible = '0') alors :
            Nouv_EtatBase_0 = Etat_Base (Etat_Quant [ i ] . par / sqrt(2), nouv_vect_0) ;
            Nouv_EtatBase_1 = Etat_Base (Etat_Quant [ i ] . par / sqrt(2), nouv_vect_1) ;
        Sinon :
            Nouv_EtatBase_0 = Etat_Base (Etat_Quant [ i ] . par / sqrt(2), nouv_vect_0) ;
            Nouv_EtatBase_1 = Etat_Base ( - Etat_Quant [ i ] . par / sqrt(2), nouv_vect_1) ;
        Fin Si.
        Etat_Qt_Resultat. Ajouter ( Nouv_EtatBase_0 );
        EtatQ_Resultat. Ajouter ( Nouv_EtatBase_1 );
    Fin Pour.
    Return ( EtatQ_Resultat) ;
Fin.
```

✓ Interface graphique :

Afin d'illustrer notre implémentation, on a proposé l'interface graphique suivante :

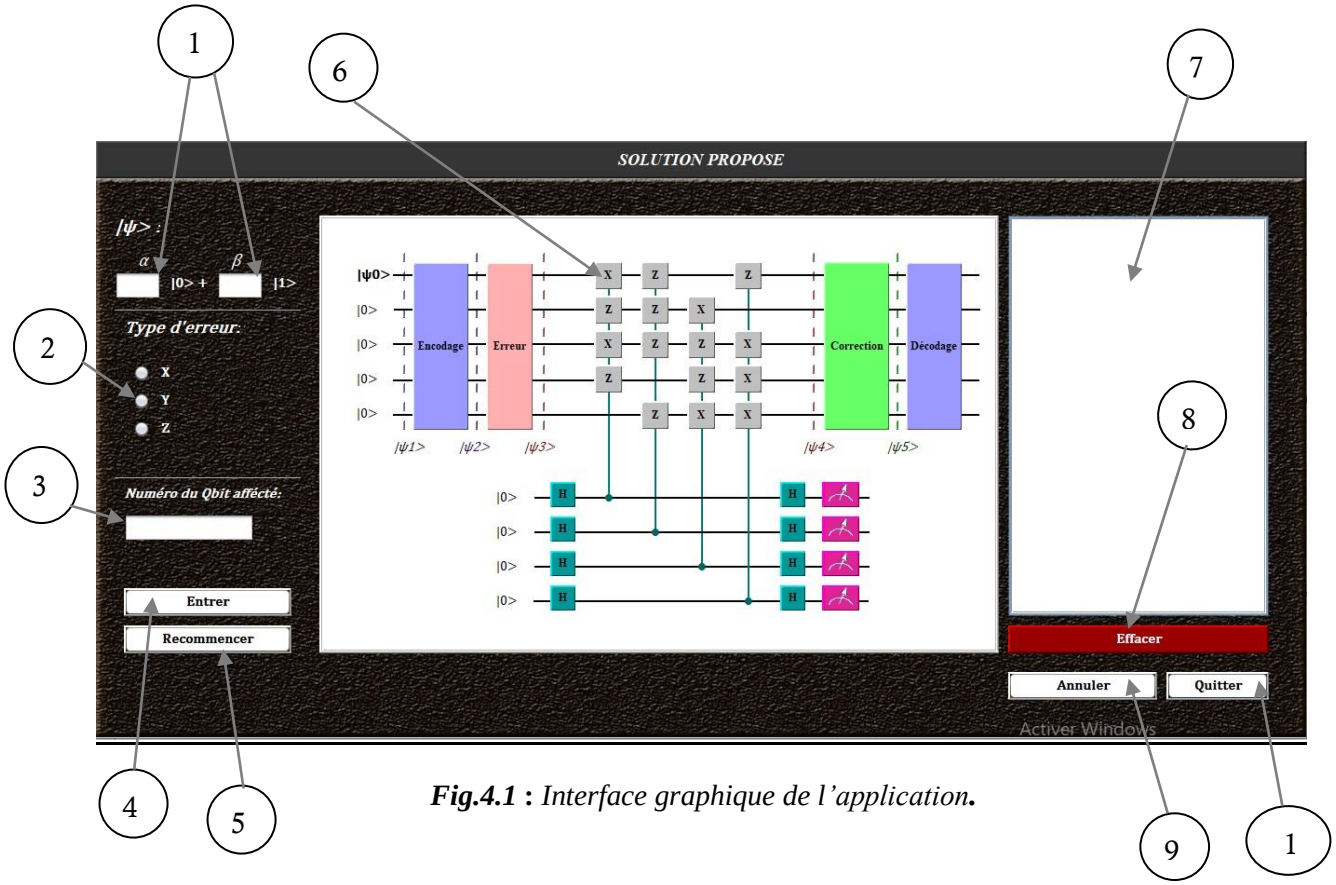

Fig.4.1 **:** *Interface graphique de l'application***.**

1) Champs utilisés pour récupérer les paramètres alpha et beta du Qubit manipulé.

2) Radio Boutons permettant de choisir le type d'erreur traitée : X, Z ou Y.

3) Champs utilisé pour saisir le numéro du Qubit affecté par l'erreur.

4) Bouton pour initialiser tous les paramètres de champs de saisie.

5) Bouton pour réinitialiser les paramètres des champs de saisie.

6) Circuit quantique.

7) Zone d'affichage des résultats.

8) Boutons pour effacer le contenu de la zone d'affichage.

9) Bouton pour arrêter le processus de circuit quantique.

10) Bouton pour quitter l'application.

IV.4. Exemples illustratifs :

Soit le Qubit Psi avec un état initial comme suit :

Psi = alpha | 0 > + beta | 1 >

Tel que : alpha = 0.35 ;

beta = 0.93675 ;

❖ Application du code stabilisateur à cinq Qubits :

- ✓ Création du Qubit à l'état initial:

  $|\psi\rangle$ = 0.35|0> + 0.93675|1>

- ✓ Ajout des quatre Qubits d'intrication:

  $|\psi_1\rangle$ = 0,35|00000> + 0.93675|10000>

- ✓ Etape d'encodage:

  - Application de la porte CNot(1,2):
    0.35|00000> + 0.93675|11000>

  - Application de la porte H sur le deuxième Qubit:
    $|\psi_2\rangle$ = 0.247|00000> + 0.247|01000> + 0.662|10000> - 0.662|11000>

  - Application de la porte CNot(2,3):
    $|\psi_2\rangle$ = 0.247|00000> + 0.247|01100> + 0.662|10000> - 0.662|11100>

  - Application de la porte H sur le quatrième Qubit:
    $|\psi_2\rangle$ = 0.175|00000> + 0.175|00010> + 0.175|01100> + 0.175|01110> + 0.468|10000> + 0.468|10010> - 0.468|11100> - 0.468|11110>

  - Application de la porte CNot(4,1):
    $|\psi_2\rangle$ = 0.175|00000> + 0.468|00010> + 0.175|01100> - 0.468|01110> + 0.468|10000> + 0.175|10010> - 0.468|11100> + 0.175|11110>

  - Application de la porte CNot(4,5):
    $|\psi_2\rangle$ = 0.175|00000> + 0.468|00011> + 0.175|01100> - 0.468|01111> + 0.468|10000> + 0.175|10011> - 0.468|11100> + 0.175|11111>

  - Application de la porte CNot(3,4):
    $|\psi_2\rangle$ = 0.175|00000> + 0.468|00011> - 0.468|01101> + 0.175|01110> + 0.468|10000> + 0.175|10011> + 0.175|11101> - 0.468|11110>

  - Application de la porte CNot(1,2):
    $|\psi_2\rangle$ = 0.175|00000> + 0.468|00011> - 0.468|01101> + 0.175|01110> + 0.175|10101> - 0.468|10110> + 0.468|11000> + 0.175|11011>

- Application de la porte H sur le cinquième Qubit:
  $|\psi_2\rangle$ = 0.124|00000> + 0.124|00001> + 0.331|00010> - 0.331|00011> - 0.331|01100> + 0.331|01101> + 0.124|01110> + 0.124|01111> + 0.124|10100> - 0.124|10101> - 0.331|10110> - 0.331|10111> + 0.331|11000> + 0.331|11001> + 0.124|11010> - 0.124|11011>

- Application de la porte CNot(5,2), on obtient le psi codé comme suit:
  $|\psi_2\rangle$ = 0.124|00000> + 0.331|00010> + 0.331|00101> + 0.124|00111> + 0.124|01001> - 0.331|01011> - 0.331|01100> + 0.124|01110> + 0.331|10001> - 0.124|10011> + 0.124|10100> -0.331|10110> + 0.331|11000> + 0.124|11010> - 0.124|11101> - 0.331|11111>

• Correction d'une erreur de type X (Bit-flip):

✓ Injecter une Erreur X sur le deuxième Qubit :
$|\psi_3\rangle$ = 0.331|00001> + 0.124|00011> + 0.124|00100> + 0.331|00110> - 0.331|01000> + 0.124|01010> + 0.124|01101> - 0.331|01111> + 0.124|10000> - 0.331|10010> + 0.331|10101> - 0.124|10111> -0.124|11001> - 0.331|11011> + 0.331|11100> + 0.124|11110>

✓ Ajout des quatre Qubits du syndrome:
$|\psi_3\rangle$ = 0.331|000010000> + 0.124|000110000> + 0.124|001000000> + 0.331|001100000> -0.331|010000000> + 0.124|010100000> + 0.124|011010000> -0.331|011110000> +0.124|100000000>- 0.331|100100000> + 0.331|101010000> -0.124|101110000> - 0.124|110010000>- 0.331|110110000> + 0.331|111000000> + 0.124|111100000>

✓ Après application du circuit, on obtient :
$|\psi_3\rangle$ = 0.124|00001**1100**> - 0.331|00011**1100**> - 0.331|00100**1100**> + 0.124|00110**1100**> + 0.124|01000**1100**> + 0.331|01010**1100**> + .331|01101**1100**> +0.124|01111**1100**> +0.331|10000**1100**> +0.124|10010**1100**> -0.124|10101**1100**> - 0.331|10111**1100**> +0.331|11001**1100**> -0.124|11011**1100**> +0.124|11100**1100**> - 0.331|11110**1100**>

Résultat de mesure = **1100**, donc erreur de type **X2**.

✓ **La correction :** appliquer la porte X sur le deuxième Qubit :
$|\psi_4\rangle$ = 0.124|000001100> + 0.331|000101100> + 0.331|001011100> + 0.124|001111100> + 0.124|010011100> - 0.331|010111100> - 0.331|011001100> +0.124|011101100> +0.331|100011100>- 0.124|100111100> +0.124|101001100> - 0.331|101101100> +0.331|110001100> +0.124|110101100> -0.124|111011100> - 0.331|111111100>

✓ Suppression des quatre Qubits du syndrome:

$|\psi_4\rangle$ = 0.124|00000> + 0.331|00010> + 0.331|00101> + 0.124|00111> + 0.124|01001> - 0.331|01011> - 0.331|01100> + 0.124|01110> + 0.331|10001> - 0.124|10011> + 0.124|10100> - 0.331|10110> + 0.331|11000> + 0.124|11010> - 0.124|11101> - 0.331|11111>

✓ Etape de décodage :

- Application de circuit de décodage :

$|\psi_5\rangle$ = 0.349999 |00000> + 0.936749 |10000>

- Suppression des Qubits auxiliaires :

$|\psi\rangle$ = 0.349999 |0> + 0.936749 |1>

• Correction d'une erreur de type Z (Phase-flip):

✓ Injecter une Erreur Z sur le quatrième Qubit :

$|\psi_3\rangle$ = 0.124|00000> - 0.331|00010> + 0.331|00101> - 0.124|00111> + 0.124|01001> + 0.331|01011> - 0.331|01100> - 0.124|01110> + 0.331|10001> + 0.124|10011> + 0.124|10100> + 0.331|10110> + 0.331|11000> - 0.124|11010> - 0.124|11101> + 0.331|11111>

✓ Ajout des quatre Qubits du syndrome:

$|\psi_3\rangle$ = 0.124|000000000> - 0.331|000100000> + 0.331|001010000> - 0.124|001110000> + 0.124|010010000> + 0.331|010110000> - 0.331|011000000> -0.124|011100000> +0.331|100010000> +0.124|100110000> +0.124|101000000> +0.331|101100000> +0.331|110000000>- 0.124|110100000> - 0.124|111010000> + 0.331|111110000>

✓ Après application du circuit, on obtient :

$|\psi_3\rangle$ = 0.124|00000**0001**> - 0.331|00010**0001**> + 0.331|00101**0001**> - 0.124|00111**0001**> + 0.124|01001**0001**> + 0.331|01011**0001**> - 0.331|01100**0001**> -0.124|01110**0001**> +0.331|10001**0001**> +0.124|10011**0001**> +0.124|10100**0001**> +0.331|10110**0001**> +0.331|11000**0001**>- 0.124|11010**0001**> - 0.124|11101**0001**> + 0.331|11111**0001**>

Résultat de mesure = **0001**, donc erreur de type **Z4**.

✓ **La correction :** appliquer la porte Z sur le quatrième Qubit :

$|\psi_4\rangle$ = 0.124|000000001> + 0.331|000100001> + 0.331|001010001> + 0.124|001110001> + 0.124|010010001> - 0.331|010110001> - 0.331|011000001> +0.124|011100001> +0.331|100010001>- 0.124|100110001> +0.124|101000001> - 0.331|101100001> +0.331|110000001> +0.124|110100001>- 0.124|111010001> - 0.331|111110001>

✓ Suppression des quatre Qubits du syndrome:

$|\psi_4\rangle$ = 0.124|00000> + 0.331|00010> + 0.331|00101> + 0.124|00111> + 0.124|01001> - 0.331|01011> - 0.331|01100> + 0.124|01110> + 0.331|10001> - 0.124|10011> + 0.124|10100> - 0.331|10110> + 0.331|11000> + 0.124|11010> - 0.124|11101> - 0.331|11111>

✓ Etape de décodage :

- Application de circuit de décodage :

  $|\psi_5\rangle$ = 0.349999 |00000> + 0.936749 |10000>

- Suppression des Qubits auxiliaires :

$|\psi\rangle$ = 0.349999 |0> + 0.936749 |1>

• Correction d'une erreur de type Y (Phase-flip):

✓ Injecter une Erreur Y sur le premier Qubit :

$|\psi_3\rangle$ = 0.331|00001> - 0.124|00011> + 0.124|00100> - 0.331|00110> + 0.331|01000> + 0.124|01010> - 0.124|01101> - 0.331|01111> - 0.124|10000> - 0.331|10010> - 0.331|10101> - 0.124|10111> - 0.124|11001> + 0.331|11011> + 0.331|11100> - 0.124|11110>

✓ Ajout des quatre Qubits du syndrome:

$|\psi_3\rangle$ = 0.331|000010000> - 0.124|000110000> + 0.124|001000000> - 0.331|001100000> + 0.331|010000000> + 0.124|010100000> - 0.124|011010000> - 0.331|011110000> - 0.124|100000000> - 0.331|100100000>- 0.331|101010000> - 0.124|101110000>- 0.124|110010000> +0.331|110110000> +0.331|111000000> - 0.124|111100000>

✓ Après application du circuit, on obtient :

$|\psi_3\rangle$ = 0.331|00001**1101**> - 0.124|00011**1101**> + 0.124|00100**1101**> - 0.331|00110**1101**> + 0.331|01000**1101**> + 0.124|01010**1101**> - 0.124|01101**1101**> - 0.331|01111**1101**> - 0.124|10000**1101**> - 0.331|10010**1101**>- 0.331|10101**1101**> - 0.124|10111**1101**>- 0.124|11001**1101**> +0.331|11011**1101**> +0.331|11100**1101**> - 0.124|11110**1101**>

Résultat de mesure = **1101**, donc erreur de type **Y1**.

✓ **La correction :** appliquer la porte Y sur le premier Qubit :

$|\psi_4\rangle$ = 0.124|000001101> + 0.331|000101101> + 0.331|001011101> + 0.124|001111101> + 0.124|010011101> - 0.331|010111101> - 0.331|011001101> +0.124|011101101> +0.331|100011101>- 0.124|100111101> +0.124|101001101> - 0.331|101101101> +0.331|110001101> +0.124|110101101>- 0.124|111011101> - 0.331|111111101>

✓ Suppression des quatre Qubits du syndrome:

$|\psi_4\rangle$ = 0.124|00000> + 0.331|00010> + 0.331|00101> + 0.124|00111> + 0.124|01001> - 0.331|01011> - 0.331|01100> + 0.124|01110> + 0.331|10001> -

0.124|10011> + 0.124|10100> - 0.331|10110> + 0.331|11000> + 0.124|11010> - 0.124|11101> - 0.331|11111>

✓ Etape de décodage :

- Application de circuit de décodage :

  $|\psi_5\rangle$ = 0.349999 |00000> + 0.936749 |10000>

- Suppression des Qubits auxiliaires :

$|\psi\rangle$ = 0.349999 |0> + 0.936749 |1>

Les énormes difficultés qu'il y a à préserver les Qbits des influences du milieu environnant représentent l'entrave majeure à la réalisation d'un prototype d'ordinateur quantique. Le problème est si aigu que la solution n'est nullement écartée de l'adoption d'un protocole massif de corrections d'erreurs.

La correction automatique d'erreurs quantiques se fait sur le même principe qu'en théorie classique de l'information en recourant à des Qbits supplémentaires qui créent la redondance nécessaire. Mais plusieurs particularités du cas quantique rendent impossible l'adaptation directe des stratégies classiques. Le problème se complique du fait que :

- La structure du Qbit est différente de celle du bit classique.
- Certaines opérations classiques sont impossibles à réaliser de manière quantique, par exemple le clonage pur et simple est impossible.
- La mesure détruit l'état quantique. En plus, c'est une opération irréversible.
- Les erreurs quantiques possibles sont plus nombreuses que celles classiques.

Dans ce travail, nous avons présenté une nouvelle version des codes stabilisateurs. L'idée de base est inspirée de la correction basée sur les matrices de parité. Les valeurs propres des stabilisateurs sont l'analogue du syndrome. Dans un premier temps, nos choix ont été validé manuellement. Ensuite, on a proposé une implémentation en donnant quelques exemples d'application.

Cette solution permet la correction des erreurs de types X, Y et Z sans aucune ambiguïté. Comme ces trois opérateurs, dits groupe de Pauli, constituent une base de l'espace des matrices unitaires, n'importe quelle anomalie est donc corrigeable.

Ce travail peut être étendu selon plusieurs directions. Une des perspectives les plus prometteuses est la recherche des codes stabilisateurs pour un ensemble de Qbits. Le traitement collectif peut diminuer les Qbits supplémentaires nécessaires.

Bibliographie

[1] T. SAID, "Modélisation des systèmes de traitement de l'information quantique", Thèse de Doctorat en Sciences et Technologies de l'Information, soutenue Le 19/11/2016, Faculté des sciences Ben M'sik.

[2] H. Talbi, "Algorithmes évolutionnaires quantiques pour le recalage et la segmentation multi-objectif d'images", Thèse de doctorat en sciences, Université Mentouri Constantine, 2009.

[3] P. Jorrand, "A Programmer's Survey of the Quantum Computing Paradigm", World Academy of Science, Engineering and Technology International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering Vol:1, No:8, 2007

[4] S. Perdrix, "Modèles formels du calcul quantique: ressources, machines abstraites et calcul par mesure", Institut National Polytechnique de Grenoble, Décembre 2006.

[5] J. Gruska, "Quantum computing", McGraw-Hill, 1999.

[6] E. Strubell, "An Introduction to Quantum Algorithms", COS498 – Chawathe, Spring 2011.

[7] B. Lucas, H. Jaffali, I. Nounouh, " Principes fondamentaux de l'information quantique", Université Technologique de Belfort Montbéliard, Printemps 2014.

[8] Y. Leroyer et G. S´enizergues "Introduction à l'information quantique", 1 ENSEIRB- MATMECA Year 2016-2017.

[9] X. Lacour, "Information Quantique par Passage Adiabatique: Portes Quantiques et décohérence", Université de Bourgogne, 03 octobre 2007.

[10] S. MAGNIN-FEYSOT, "Cryptographie : De RSA à l'algorithme de Shor ", 8 janvier 2014.

[11] Dr. Ramazan KOC, "chapter4 Quantum circuits", Université Gaziantep.

[12] G. Brassard "Cryptology column -quantum computing : The end of classical cryptography", ACM SIGACT News 25(4)15-21 décembre 1994.

[13] P. Shor "Algorithms for quantum computation Discrete log and factoring Proceedings of the 35-th Annual Symposium on Foundations of Computer Science".

[14] P. Shor "Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer", SIAM Journal on Computing 26(5) :1484- 1509 octobre 1997.

[15] E. Pelchat, "Décodage viterbi dégénéré", Sherbrooke, Québec,canada,13septembre 2013.

[16] D. Mermin,"Calcul et algorithmes quantique : méthodes et exemples", CNRS EDITIONS.Nice, février 2010.

[17] A. Marin, "Borne inférieure sur la capacité d'un canal quantique", Rapport de stage, Centre de recherche Inria-Rocquencourt, 2009.

[18] M. Ibnouhsein, "Corrélations quantiques et structures causales ", Thèse de Doctorat, Université Paris-Sud, 2014.

[19] M. Abbara. "Turbo-codes quantiques". Théorie de l'information [cs.IT]. Ecole Polytechnique X, 2013. Français.

[20] https://fr.netbeans.org/edi/articles/concours/presentation-netbeans40-1.html.