Information mutuelle pour la construction d'un détecteur quantique optimal

Formulation du problème

Le problème de la détection d'état quantique porte sur un ensemble de m états quantiques représentés par les opérateurs densité $\{\rho_i \ , \ 1 \leq i \leq m\}$ munis des probabilités préalables $\{p_i \geq 0 \ , \ 1 \leq i \leq m\}$. L'objectif est d'obtenir un ensemble de m opérateurs de mesure $\{\Pi_j \ , \ 1 \leq i \leq m\}$ permettant de mesurer le mieux possible par rapport aux probabilités les états d'entrée qui nous arrivent.

Les opérateurs ρ_i et Π_i sont des matrices Hermitiennes semi-définies positives, de la forme $\begin{bmatrix} a & b+ic \\ b-ic & d \end{bmatrix}$.

Plusieurs critères ont été proposés à optimiser afin de construire ces détecteurs optimaux. D'une part, on a la possibilité de travailler sur la minimisation de l'erreur de mesure (l'erreur moyenne ou l'erreur quadratique). D'autre part, et c'est ce sur quoi nous avons travaillé, on peut considérer le critère de l'information mutuelle comme critère à maximiser. Ce critère indique la dépendance de deux variables aléatoires entre elles, il permet dans notre cas de bien caractériser la quantité d'information qu'on peut retirer des états d'entrée en ayant les opérateurs de mesure

L'information mutuelle de deux variables aléatoires X et Y est donnée par :

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p_{(X,Y)}(x,y) \log \Big(\frac{p_{(X,Y)}(x,y)}{p_X(x) p_Y(y)} \Big), \tag{1}$$

mais peut aussi être écrite en fonction des entropies des variables aléatoires :

$$I(X;Y) = H(X) - H(X|Y) \tag{2}$$
$$= H(Y) - H(Y|X) \tag{3}$$
$$= H(X) + H(Y) - H(X,Y). \tag{4}$$

Avec $H(X)$ entropie marginale de X, $H(Y)$ entropie marginale de Y, $H(X|Y)$ entropie conditionelle de X sachant Y et enfin $H(X,Y)$ entropie jointe de X et Y. On peut utiliser indifférement $\log_2$, $\log_{10}$ ou ln pour le logarithme, le changement étant à une constante près. On utilise par la suite le logarithme base exponentielle pour l'ensemble des calculs.

Dans le cas classique, les entropies marginales, conditionnelles et jointes sont définies par :

$$H(X) = -\sum_{x\in X} p(x)\log(p(x)), \tag{5}$$
$$H(Y) = -\sum_{y\in Y} p(y)\log(p(y)), \tag{6}$$
$$H(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log(p(x,y)), \tag{7}$$
$$H(Y|X) = -\sum_{x\in X, y\in Y} p(x,y)\log\Big(\frac{p(x,y)}{p(x)}\Big) \tag{8}$$

Dans le cas quantique, les formules restent les mêmes, mais on exprime les probabilités des variables en fonction des valeurs des états quantiques d'entrée.

En fonction d'un état d'entrée ρ_i de probabilité préalable p_i, et d'un opérateur de sortie Π_i, on peut définir leur probabilité jointe :

$$p(X = \rho_i, Y = \Pi_i) = p_i \operatorname{tr}(\rho_i \Pi_i). \tag{9}$$

On en déduit les probabilités marginales :

$$p(X = \rho_i) = \sum_j p_i \operatorname{tr}(\rho_i \Pi_j) \tag{10}$$
$$p(Y = \Pi_j) = \sum_i p_i \operatorname{tr}(\rho_i \Pi_j), \tag{11}$$

Et les probabilités conditionelles :

$$P(Y = \Pi_j | X = \rho_i) = \frac{\mathrm{tr}(\rho_i \Pi_j)}{\sum_k \mathrm{tr}(\rho_i \Pi_k)} \tag{12}$$

L'information mutuelle pour notre problème peut donc être ré-écrite de la façon suivante, en utilisant $\alpha_{ij} = \mathrm{tr}(\rho_i \Pi_j)$:

$$\begin{aligned} I(\rho; \Pi) &= H(\rho) + H(\Pi) - H(\rho, \Pi) \\ &= -\sum_{i=1}^{m} \Big(\sum_{j=1}^{m} \alpha_{ij} \Big) \log \Big(\sum_{j=1}^{m} \alpha_{ij} \Big) - \sum_{i=1}^{m} \Big(\sum_{j=1}^{m} \alpha_{ji} \Big) \log \Big(\sum_{j=1}^{m} \alpha_{ji} \Big) + \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_{ij} \log(\alpha_{ij}) \end{aligned} \tag{13}$$

On peut aussi exprimer l'information mutuelle en fonction de l'entropie conditionnelle, mais il est plus efficace d'utiliser celle donnée à l'équation 13 pour la résolution numérique.

Le problème se formule comme un problème de maximization de l'information mutuelle : on cherche à maximiser l'information qu'on peut obtenir sur ρ_i quand on a les opérateurs de mesure Π_i :

$$\max_{\Pi} I(\rho, \Pi) \tag{14}$$

tel que :

$$\Pi_j \succeq 0 \quad 1 \leq j \leq m \tag{15}$$

$$\sum_{j=1}^{m} \Pi_j = I \tag{16}$$

La contrainte 15 correspond à la semi-définition positive des opérateurs de mesure Π_j. Enfin, la contrainte 16 permet d'obtenir des opérateurs de mesure cohérents pour que les probabilités de mesure $p(j) = \mathrm{tr}(\rho \Pi_j)$ soient positives et se somment à 1.

On est en présence d'une fonction convexe, et les contraintes engendrent un ensemble admissible convexe. C'est le cas idéal lors d'une minimisation, mais le problème est une maximisation, de même difficulté qu'une minimisation concave, on ne peut donc pas juste faire une descente de gradient pour le résoudre. On peut utiliser un certain nombre de méthodes approximatives, nous utilisons le calcul par intervalle afin d'obtenir un résultat sûr dans un intervalle.

Convexité de l'information mutuelle

Davies considère en 1978 que l'information mutuelle pour ce problème peut être considérée comme étant convexe, simplifiant la résolution du problème en ayant à chercher le maximum sur les bords. On s'intéresse ici à l'étude de cette convexité.

Dans son article, Davies regroupe les traces et probabilités sous une seule variable $P_{ij} = p_i \operatorname{tr}(\rho_i \Pi_j)$. Ces coefficients P_{ij} forment une matrice des probabilités, telle que :

$$\sum_{ij} P_{ij} = 1, \tag{17}$$

$$\sum_{i} P_{ij} = p_i. \tag{18}$$

L'information mutuelle s'écrit donc :

$$I(P) = \sum_i H(\sum_j P_{ij}) + \sum_j H(\sum_i P_{ij}) - \sum_{ij} H(P_{ij}) \tag{19}$$

La fonction $H(x) = -x \log(x)$ est convexe, et donc I est convexe par rapport à la matrice des probabilités P. La figure 1 illustre cette fonction en fixant $p_1 = 0.3$ et $p_2 = 0.7$.

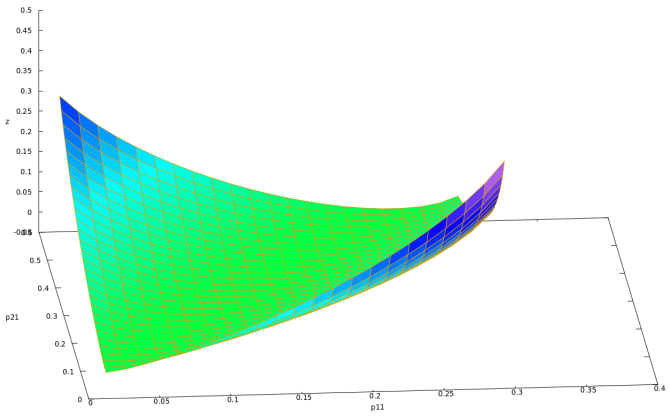


FIGURE 1 – Information mutuelle par rapport à la matrice de probabilités

La convexité semble bien vraie par rapport à P, mais on cherche à optimiser les matrices Π_j. La matrice P comporte les traces de la multiplication

$\rho_i \Pi_j$, qui est linéaire par rapport aux coefficients de Π_j. Si la fonction $I(P)$ est convexe par rapport à P, alors elle l'est par rapport aux Π_j, grace à la linéarité.

Quand on trace la même fonction mais par rapport aux variables $\Pi_{k_{ij}}$, en se fixant dans un espace 2 dimensions, on s'apperçoit que la fonction n'est pas correctement définie sur les bords. Ceci est dû au fait que $x \log(x)$ n'est pas définit pour $x < 0$, ce qui fausse ou bloque les calculs, suivant l'implémentation. Le détail de notre implémentation est expliqué en 3c.

Formulation des contraintes

La définition du problème permet de résoudre notament les cas immédiats des opérateurs de densité orthogonaux, mais la résolution devient très lente lorsqu'on passe à d'autres cas non orthogonaux. On rajoute des conditions au problème pour accélérer la résolution.

Le premier élément à simplifier est l'expression de l'entropie marginale de $X = \rho_i$. En effet, nous l'avons exprimé en fonction de la trace de la multiplication matricielle, mais on peut reprendre la définition donnée lors du cas classique qui dit que $H(X) = -\sum_{x \in X} p(x) \log(p(x))$. Le problème nous indique que nous connaissons les probabilités préalables des états d'entrée, on peut donc directement exprimer cette entropie en fonction de ces données et donc sans les variables de sortie Π_i.

Ensuite, on sait que les opérateurs de mesure se somment à l'identité. Cela signifie d'une part qu'on peut passer d'un problème à m matrices à un problème à $m-1$ matrices pour $m \geq 2$. Les matrices étant carrées de dimension n, on passe de $m \times n^2$ variables à $(m-1) \times n^2$ variables, ce qui est non négligeable.

De plus, le problème et les contraintes sont symmétriques, on peut intervertir les Π_j sans influencer le résultat de la fonction coût. Cela nous permet de couper le problème au moins en deux pour réduire à nouveau le temps de calcul. Du fait de la somme à l'identité, on peut ajouter en contrainte que $\Pi_{1_{1,1}} \leq \frac{1}{m}$ pour m opérateurs de mesure, puis $\Pi_{2_{1,1}} \leq \frac{1}{m-1}$, etc.

Enfin, pour rappel, les opérateurs de mesure sont des opérateurs densité, qui ne sont pas necessairement purs. Pour qu'ils soient purs, il faudrait entre autre que $\mathrm{tr}(\Pi_i) = 1$. On peut considérer qu'on restreint le problème à un cas purs, et dans ce cas rajouter la contrainte que la somme des éléments diagonaux des opérateurs de mesure doit sommer à 1. Cela permet soit de retirer une variable par matrice densité au problème, en l'exprimant par $x_{n+1} = 1 - \sum_{i}^{n} x_i$ avec les x_i éléments diagonaux de l'opérateur densité, ce

qui nécessite une reformulation du problème, soit l'ajout de la contrainte.

Résolution avec `ibex`

Pour la résolution de ce problème, utilisons la librairie `ibex` permettant de faire du calcul par intervale, et possède entre autres un outil d'optimisation, `ibexopt`. Le problème est formulé avec un langage dédié, `minibex`. Nous avons eu besoin de définir un opérateur additionnel à ceux présents, l'opérateur `xlog` permettant d'effectuer l'opération $x \times \log(x)$ en redéfinissant $0 \times \log(0) = 0$ pour que les intervalles ne tombent pas à l'infiniquand ils contiennent 0. De plus, `minibex` ne considère que des problèmes de minimisation, on ré-écrit le problème en prenant la fonction coût opposée : $\max f(x) \Leftrightarrow \min -f(x)$.

Le premier test effectué est sur le cas de deux états d'entrée $|\psi_1\rangle = |0\rangle$ et $|\psi_2\rangle = |1\rangle$ ayant pour probabilité respective $p_1 = 0.1$ et $p_2 = 0.9$. Le résultat théorique est connu : les états étant orthogonaux, on doit obtenir les opérateurs de mesure égaux aux opérateurs densité d'entrée. On obtient bien avec `ibex` les opérateurs suivant :

$$\Pi_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad \Pi_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},$$

qui correspondent bien à deux opérateurs de mesure orthogonaux. Dans ce cas, l'information mutuelle est comprise dans l'intervalle $I(\rho, \Pi) \in [0.3250, 0.3254]$, avec un temps de calcul de 8.6 millisecondes.

Le deuxième cas qu'on peut présenter est le suivant : $|\psi_1\rangle = |0\rangle$ et $|\psi_1\rangle = |+\rangle$ avec comme probabilité respectives $p_1 = 0.1$ et $p_2 = 0.9$. Le résultat théorique n'est pas donné, et on obtient avec `ibex` le résultat suivant :

$$\Pi_1 = \begin{bmatrix} 0.445 & 0.497 \\ 0.497 & 0.555 \end{bmatrix}, \quad \Pi_2 = \begin{bmatrix} 0.555 & -0.497 \\ -0.497 & 0.445 \end{bmatrix},$$

avec une information mutuelle comprise dans l'intervalle $I(\rho, \Pi) \in [0.1348, 0.1349]$, et un temps de calcul de 0.79 secondes.

En revanche, ibex ralentit fortement dès qu'on sort des cas simples présentés si-dessus, et notamment quand on retire la restriction des opérateurs de mesure à des opérateurs densité purs (de trace unitaire). Deux éléments importants sont à l'origine du problème. Tout d'abord, en analysant l'utilisation des ressources cpu lors de la résolution d'un problème, on voit qu'un seul thread est utilisé par `ibexopt`. Les algorithmes d'optimisation peuvent être parallélisés, et dans le cas de processeurs à plusieurs cœurs on pourrait avoir un gain de performance conséquent. Le deuxième élément est en lien avec la convexité de la fonction coût évoqué précédement. On eut alors se limiter aux extremités de la fonction coût pour réduire le nombre de calculs

à effectuer. Il faudrait alors implémenter la condition $0 \in [\texttt{grad}]([f])$, ce qui n'est pas prévu de base dans `ibexopt`.

Algorithme d'optimisation

On met en place un algorithme d'optimisation utilisant le calcul par intervalle pour obtenir un encadrement garanti de la solution à notre problème.

La figure 5 présente l'algorithme utilisé pour ce problème de maximisation (la logique étant la même pour une minimisation). Quatre éléments sont important à comprendre dans cet algorithme.

La première étape est de bisecter l'ensemble des boites composant notre liste. Tout d'abord, la bisection d'un intervalle seul s'effectue simplement en prenant $[x] \longrightarrow [\underline{x}; (\overline{x} + \underline{x}).0.5], [(\overline{x} + \underline{x}).0.5; \overline{x}]$ en coupant l'intervalle voulu au milieu. On peut aussi envisager de le bisecter plus près de la borne inférieure ou de la borne supérieure. Pour une boite - extension d'un intervalle à plusieurs dimensions - on bisecte suivant la dimensions comportant l'intervalle le plus grand. La figure 2 illustre la bisection sur des boites en deux dimensions, avec une alternance de bisection verticale et horizontale.

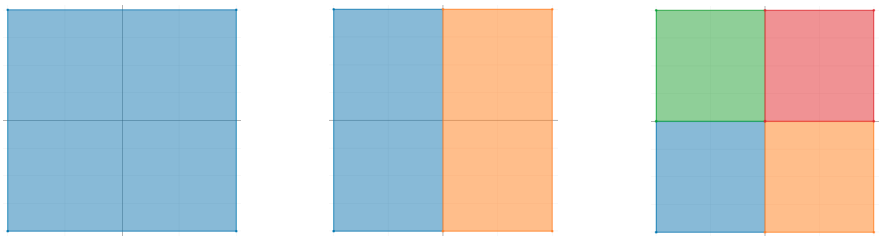


FIGURE 2 – Bisection de boites en deux dimensions

La deuxième étape est d'évaluer un encadrement de la fonction coût pour une boite d'entrée. Dans le cadre du calcul par intervalle, on dispose de la notion de fonction d'inclusion. Le principe est de fournir un encadrement garanti de l'image de la fonction :

Définition 1. *Soit une fonction* $f : \mathbb{R}^n \mapsto \mathbb{R}^m$. *La fonction d'inclusion correspondante est définie par :*

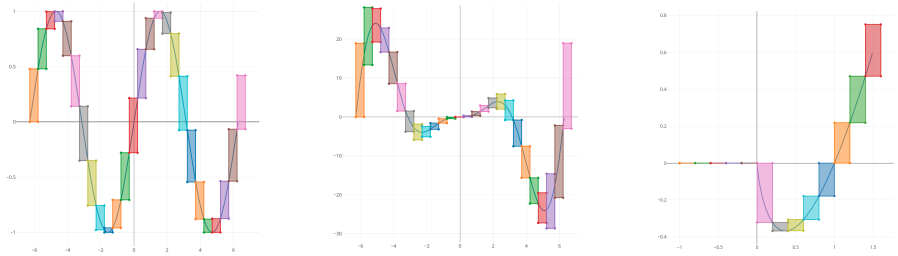
$$\forall [x] \in \mathbb{R}^n, f([x]) \subset [f]([x]) \tag{20}$$

La figure 3a illustre la fonction d'inclusion pour la fonction $f : x \mapsto sin(x)$. On note que l'encadrement peut être plus ou moins optimiste, en

laissant plus ou moins d'espace entre les bornes inférieures ou supérieures et les valeurs réelles.

Les fonctions d'inclusion peuvent aussi être combinées sans perdre la garantie d'encadrement de l'image, par exemple avec la figure 3b pour la fonction $f : x \mapsto sin(x) \times x^2$.

On peut donc construire une fonction d'inclusion pour l'information mutuelle définie à l'équation 13. Il faut noter dans cette fonction la présence récurrente de l'élément $f : x \mapsto x \times \log(x)$. Cette fonction n'est définie que sur $[0, +\infty]$ et mathématiquement $\lim_{x \to 0} x \times \log(x) = 0$. On ne peut donc en théorie pas avoir une entrée négative. En revanche, en pratique le problème peut amener à se retrouver avec des valeurs négatives comme entrées de cette fonction, et les langages informatiques ne définissent pas le logarithme hors des bornes, en renvoyant une exception en général. Pour les librairies de calcul par intervalle, suivant les implémentations on peut se retrouver soit avec une exception, soit avec une borne de l'intervalle infinie. Il est donc nécessaire de ré-implémenter spécifiquement cette fonction avec un opérateur distinct, en gérant les cas de bord. On choisi de définir $\forall x \in [-\infty, 0], x \times \log(x) = 0$ et donc $\forall [x] \leq 0, [x] \times [log]([x]) = [0, 0]$, comme illustré à la figure 3c.



(a) $y = sin(x)$ (b) $y = sin(x) \times x^2$ (c) $y = x \times \log(x)$

FIGURE 3 – Fonctions d'inclusion

Une fois établie l'évaluation de la fonction, on voit que pour récupérer l'optimal de la fonction, il suffit de trouver les plus hautes boites de l'ensemble obtenu.

Propriété 1. *Le maximum d'une fonction d'inclusion peut être trouvé en éliminant les boites suivant la formule suivante :*

$$sup([f]([x])) \leq f(a) \Rightarrow x* \notin [x] \tag{21}$$

Il s'agit alors de trouver un critère a satisfaisant cette propriété. Par exemple, on peut prendre la borne inférieure des intervalles, et donc on considère que tout les intervalles ayant leur borne supérieure strictement inférieure

aux autres peuvent être éliminés. Un exemple plus utile et permettant l'élimination de plus de boites est de prendre $f(a) = f(\frac{x+\overline{x}}{2})$, plus haute image des milieux des intervalles.

Exemple 1. *En reprenant la fonction $f : x \mapsto sin(x) \times x^2$, on voit avec la figure 4 que toutes les boites ayant leur borne supérieure strictement inférieure à la ligne rouge sont garantit comme ne contenant pas l'optimum. On peut donc garder les deux boites verte et rouge, et éliminer le reste des boites.*

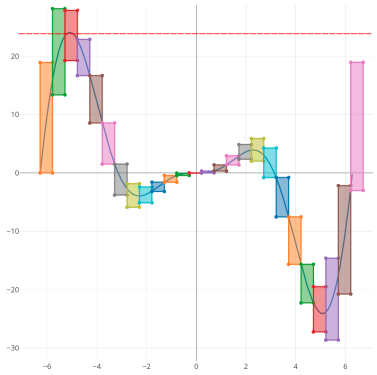


FIGURE 4 – $y = sin(x) \times x^2$

Enfin, le dernier élément à traiter dans cet algorithme est la gestion des contraintes. Le problème qu'on souhaite traiter possède un certain nombre de contraintes, dont les bornes des variables d'entrées. On considère l'ensemble des contraintes $\{g_j([x])\}$ formant un espace admissible A. Pour qu'une boite soit acceptée, il faut qu'elle soit comprise dans cet espace admissible. Spécifiquement, on a vu que la fonction d'information mutuelle est convexe. Cela simplifie le problème puisqu'il nous suffit de chercher le maximum sur les bords de la fonction. Formellement, tout cela correspond à deux conditions :

Propriété 2. *Deux conditions pour éliminer des boites*

$$(0,0,0) \notin [df]([x]) \quad \textbf{\textit{et}} \quad [x] \subset A \Rightarrow x^* \notin [x]; \tag{22}$$

$$\begin{cases} 0 \in g_j([x]) \\ \forall i \neq j, 0 \notin g_i([x]) \\ dg_j \quad \textbf{\textit{et}} \quad df \quad \textbf{\textit{indépendants}} \\ 0 \notin [df]([x]) \end{cases} \Rightarrow x^* \notin [x]. \tag{23}$$

La première condition indique qu'il faut être sur le bord des contraintes : on cherche à optimiser, donc l'intérieur de l'espace admissible ne contiendra pas la solution. La deuxième condition indique que les gradients des fonctions

contraintes et de la fonction coût doivent être indépendant (géométriquement parallèles, algébriquement de produit scalaire nul).

illustration gradient contrainte / coût parallèle ?

On vois qu'il suffit de répéter les étapes précédentes sur les boites restantes, ce qui va nous faire tendre vers une inclusion de plus en plus précise de l'optimum du problème. L'algorithme détaillé est présent avec la figure 5. On peut noter entre autres que la première boucle, pour la bisection, est facilement parallélisable et donc accélère considérablement les calculs. La section de suppression des boites ne l'est en revanche pas facilement.

Data: $[I_{init}]$ initial search box ; ϵ stop criterion ; f cost function ; g constraints function ;
Output: $[f]$ bounds of best solution ; $[I]$ solution box
begin
  `solutions` list of solution boxes;
  Add $[I_{init}]$ to `solutions`;
  $[f_c]$ current bounds of solutions;
  while $\overline{f_c} - \underline{f_c} \geq \epsilon$ **do**
    `currentSolutions` empty list of boxes;
    `/* Bisect, evaluate cost, manage constraints */`
    forall *sol in solutions* **do**
      [`left`], [`right`] $\longleftarrow$ `bisect(sol)`;
      if *g*([*left*]) *is valid* **then**
        Add [`left`] to `currentSolutions`;
      end
      if *g*([*right*]) *is valid* **then**
        Add [`right`] to `currentSolutions`;
      end
    end
    `/* Remove boxes certified not to contain maximum */`
    Evaluate $[f]$ for all [`currentSolutions`];
    f_{best} best f(`sol.mid`) in all $[f]$;
    Remove all [`sol`] in [`currentSolutions`] where
     $sup([f]([sol])) \leq f_{best}$;
    `solutions` $\longleftarrow$ `currentSolutions`
  end
  return *solutions*, $[f_c]$
end

FIGURE 5 – Algorithme de maximisation par le calcul par intervalle

L'algorithme 5 a été implémenté en C# (dotnet 5) sur la base de la librairie IntSharp modifiée pour répondre à nos besoins (rajout de l'intervalle vide, de la fonction $x\log(x)$, des intervalles booléens, ...). Une interface graphique basique utilisant Blazor a été mise en place pour faciliter la visualisation de l'optimisation et des différents problèmes rencontrés lors du développement.

Exemple

Prenons le cas de deux états quantiques :

$$|\psi_1\rangle = \begin{bmatrix} \frac{1}{3} \\ \frac{2\sqrt{2}}{3} \end{bmatrix}, \quad |\psi_2\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \tag{24}$$

avec les probabilités préalables

$$p(|\psi_1\rangle) = 0.1, \quad p(|\psi_2\rangle) = 0.9. \tag{25}$$

Ces deux états peuvent être réécris sous la forme d'opérateurs densité $\rho_1 = \begin{bmatrix} \frac{1}{9} & \frac{2\sqrt{2}}{9} \\ \frac{2\sqrt{2}}{9} & \frac{8}{9} \end{bmatrix}$ et $\rho_2 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$.

Le problème est de trouver les deux opérateurs densité optimaux au sens de l'information mutuelle. On considère le problème comme ne possédant pas de termes complexes sur les antidiagonaux, et l'équation 16 nous permet de réduire le problème à trois variables seulement $\{a_1, b_1, d_1\}$:

$$\Pi_1 = \begin{bmatrix} a_1 & b_1 + ic_1 \\ b_1 - ic_1 & d_1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 \\ b_1 & d_1 \end{bmatrix} \tag{26}$$

$$\Pi_2 = I - \Pi_1 = \begin{bmatrix} 1 - a_1 & -b_1 \\ -b_1 & 1 - d_1 \end{bmatrix} \tag{27}$$

On pose ensuite les contraintes sur ces variables. Tout d'abord, ces variables sont définies sur ces bornes spécifiques : $a_1 \in [0, 0.5]$, $b_1 \in [-1, 1]$, $d_1 \in [0, 1]$. La détermination de la semi-définition positive passe par les diagnonales et le déterminant strictement positifs, d'une part avec les bornes précédentes et d'autre part avec deux nouvelles contraintes sur les 3 variables. Le problème s'écrit donc :

$$\max_{a_1, b_1, d_1} I(a_1, b_1, d_1),$$

tel que :

$$
\begin{aligned}
&a_1 \in [0, 0.5], b_1 \in [-1, 1], d_1 \in [0, 1], \\
&a_1 * d_1 - b_1^2 \geq 0, \\
&(1 - a_1) * (1 - d_1) - b_1^2 \geq 0.
\end{aligned}
$$

La résolution avec `ibex` ou avec notre solveur donne les deux opérateurs de mesure suivants :

$$
\Pi_1 = \begin{bmatrix} 0.456 & -0.498 \\ -0.498 & 0.544 \end{bmatrix}, \quad \Pi_2 = \begin{bmatrix} 0.544 & 0.498 \\ 0.498 & 0.456 \end{bmatrix}, \tag{28}
$$

Avec une information mutuelle $I(a_1, b_1, d_1) = 0.04723$. Notre solveur résout le problème en 6.4 secondes pour une précision sur I de 1×10^{-5}, et `ibexopt` résout en 88.3 secondes pour une précision sur I de 4×10^{-5}.

Inconvénients de la résolution par intervalles

La résolution de ce problème avec le calcul par intervalles permet certes de garantir une solution, mais on se retrouve très rapidement bloqué quand on veut augmenter la dimension du problème. On peut déjà le voir en passant du cas de deux états orthogonaux à deux états non orthogonaux les temps de calculs augmentent considérablement, même si on est avec seulement 3 variables. Quand on veut passer à trois états, donc 6 variables, en revanche, la résolution devient impossible dans un temps et utilisation mémoire raisonnables.