エンタングルメント変換における触媒効果

工学部 物理工学科 03380411
橋本麻美

2007年2月

目 次

序章

現在、古典的な計算機では、情報はすべて 2 進数で表され、ビット (bit) で表される。量子力学でこれに対応するのが、2 つの状態しか持たない系であり、qubit と呼ばれている。qubit は量子力学の基本的性質である、「量子重ね合わせ」、「観測による射影」を用いることができる。よって、bit は 0 か 1 のどちらか一方の値しかとらないのに対し、qubit は 0 と 1 の重ね合わせの状態もとることができるのである。また、「量子もつれ合い」の状態をとることもでき、量子エンタングルメントと呼ばれている。

近年、このように、原子や素粒子などの微小な世界の力学である量子力学の性質を持った qubit を情報処理に利用しようとする研究が盛んである。

現在の電気や光などの「波」の性質を利用して情報を伝達する古典情報通信では不可能・不得意な情報処理の実行を目指すもので、電子や光などの「粒子」の性質を利用して情報を処理・伝送しようというものである。例えば、超高速で計算できる量子計算機や絶対に盗聴されない量子暗号などがある。このような分野は量子情報と呼ばれており、物理学のなかで最も新しい分野の一つである [参考文献.1,2,3]。

本研究では、量子もつれ合いの状態をとることができる量子エンタングルメントを扱う。この量子もつれ合いの状態を持つ量子の pair は、EPR pair[参考文献.4] と呼ばれ、Einstein, Podolsky, Rosen の頭文字を取っている。

Einstein は、自然を確定的に記述できない量子力学というものに不快感を持って生涯抵抗し、量子力学を攻撃する数々の思考実験を考えては、量子力学の創始者の一人である論敵の Niels Bohr を悩ませた。しかし Einstein ら 3 人が量子力学を攻撃するつもりで考えたその思考実験は、皮肉にも逆に量子力学の理解を深めるためのものになってしまった。この 3 人の実験で用いられた絡み合った量子系が EPR pair である。現在では存在することが確認されており、量子情報には欠かせないものとなっている。

また、将来の量子情報の基礎となると考えられているのが量子テレポーテーション [参考文献 5] である。情報伝達時に送り手が情報をいったん消去するのだが、受け手でいつのまにか復元される。空想科学 (SF) 映画や小説では、人間が姿を消して空間を瞬時に移動し、別の場所に現れるテレポーテーションという場面がよく登場する。話に広がりを持たせているが、実際にはありえないと考えられていた。ところが、情報の伝送なら実現できることが 98 年、米カリフォルニア工科大学の研究グループの実験で実証され [参考文献.6]、今後、量子情報は大きく発展する可能性を秘めている。

この量子情報処理で有効的に使われ、重要な資源である、量子エンタングルメントを遠く離れている 2 人の観測者によって変換できるかどうかは重要な問題である。そのため本研究では、量子エンタングルメントの変換について、どのような場合に変換できるのか考える。また、数学の概念 majorization について紹介し、量子エンタングルメントの変換との関係を明確にする。さらに、変換補助の役割である触媒を使って、触媒効果について考察し、自分自身を触媒として使っても変換できるのかについて検証する。

第1章　古典情報と量子情報

1.1　古典 bit と量子 bit

古典情報の基本概念はビット (bit) である。1 ビットは、0 と 1 で表され、例としては、コインの裏表があり、裏を 0 とすると、表が 1 のようになる。測定すると、0 か 1 のどちらかが測定される。

一方、量子情報でビットに対応するのが、2 つの状態しか持たない系であり、量子ビット (quantum bit) や略して qubit と呼ばれている。qubit と見なせるものとしては、電子のスピンがあり、上向きのスピンを $|\,0\,\rangle$、下向きのスピンを $|\,1\,\rangle$ と表される。重ね合わせの原理によると、重ね合わせの状態もあるはずで、斜め向きのスピンがその例である。

重ね合わせの状態

$$|\,\phi\,\rangle = \alpha|\,0\,\rangle + \beta|\,1\,\rangle \tag{1.1}$$

ここで α、β は複素数であるが、多くの場合、実数と考えてもよい。

この qubit を測定すると、$|\alpha|^2$ の確率で $|\,0\,\rangle$、$|\beta|^2$ の確率で $|\,1\,\rangle$ が測定される。よって、$|\alpha|^2 + |\beta|^2 = 1$ である。

1.2　量子エンタングルメント

1.2.1　EPR pair

qubit が 2 つある場合、1 つ目の qubit の状態が $|\,0\,\rangle$、2 つ目の qubit の状態が $|\,1\,\rangle$ にあるとき、全系の状態は $|\,01\,\rangle$ で表される。従って、量子系は、$|\,00\,\rangle$、$|\,01\,\rangle$、$|\,10\,\rangle$、$|\,11\,\rangle$ の 4 つの状態を持ち、また重ね合わせた状態もとることができる。

重ね合わせの状態

$$|\,\phi\,\rangle = \alpha_{00}|\,00\,\rangle + \alpha_{01}|\,01\,\rangle + \alpha_{10}|\,10\,\rangle + \alpha_{11}|\,11\,\rangle \tag{1.2}$$

ここで α_{00}、α_{01}、α_{10}、α_{11} は複素数であり、その絶対値 2 乗はそれぞれの qubit の存在確率となっている。

$$|0\rangle \rightarrow |0\rangle$$
$$|1\rangle \rightarrow |1\rangle$$

図 1.1　量子エンタングルメント

また qubit が 2 つある場合、EPR pair (Einstein、Podolsky、Rosen 対)[参考文献.4] と呼ばれる重要な状態があり、次のように表される。

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{1.3}$$

この式の具体的意味は、$|00\rangle$ と $|11\rangle$ という状態がともに $\frac{1}{2}$ の確率で測定できることを意味する。つまり、この 2 つの qubit をアリスとボブ、2 人の観測者にそれぞれ持たせたとき、1 つ目の qubit の測定結果が 0 ならば、2 つ目の qubit の測定結果は必ず 0 となる。ここで 2 つの qubit は離れていてもよい。

このように一方の qubit の状態がきまると、自動的にもう一方の qubit の状態も決まるといったもつれのことを量子エンタングルメントと呼ぶ。

1.2.2　量子エンタングルメントが役に立つ例

量子エンタングルメントが役に立つ例として、量子テレポーテーションがあげられる。量子テレポーテーションは、Bennett, Brassard, Crépeau, Jozsa,Peres, Wootters によって発見された [参考文献.5]。

アリスは状態 $|\phi\rangle$ を持っていて、それを遠く離れているボブに送りたいとする。古典的な通信で $|\phi\rangle$ の全情報を送ろうと思うと、無限ビットの情報を送る必要がある。しかし、この量子テレポーテーションでは、アリスとボブは前もって一つの EPR pair を共有しており、その EPR pair を使って、2 ビットの古典情報を送るだけでボブは状態 $|\phi\rangle$ を再生することができるのである。さらに、アリスは状態 $|\phi\rangle$ を全く知らなくてもよいのである。

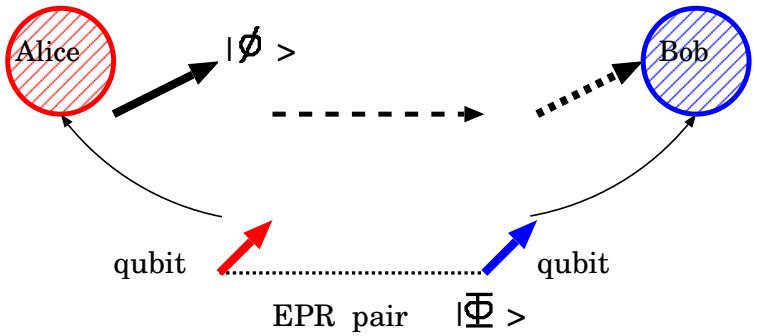

図 1.2　量子テレポーテーション

また、Superdense coding もその例である。Superdense coding は Bennett,Wiesner により発明された [参考文献.7]。量子テレポーテーション同様、アリスとボブは前もって 1 つの EPR pair を共有しているとする。アリスはボブに 2 ビットの古典情報を送りたいのだが、1 つの qubit しか送れない。しかし、EPR pair を使うと、qubit を 1 つ送るだけで、2 ビットの古典情報が送れるのである。

すなわち、量子エンタングルメントは重要な資源であるといえる。本研究では、資源変換、つまり量子エンタングルメントの変換がどのような場合になされるか考察する。

第2章　エンタングルメント変換

ここでは具体的な例をあげ、エンタングルメントの変換方法や条件について考察する。

2.1　LOCC での変換

2.1.1　qubit の測定のみでの変換

2 人の観測者、アリスとボブが EPR 状態 $|\psi\rangle$ を共有しているとして、$|\psi\rangle$ を別のエンタングル状態 $|\phi\rangle$ に変換できるか考える。ここで 2 人は、自分たちの qubit に対して、測定などを含む、局所的演算 (Locol Operations) と、電話などの古典通信 (Classical Communication) はできるものとする。この限られた演算を、それぞれの頭文字をとって LOCC と呼ぶ。

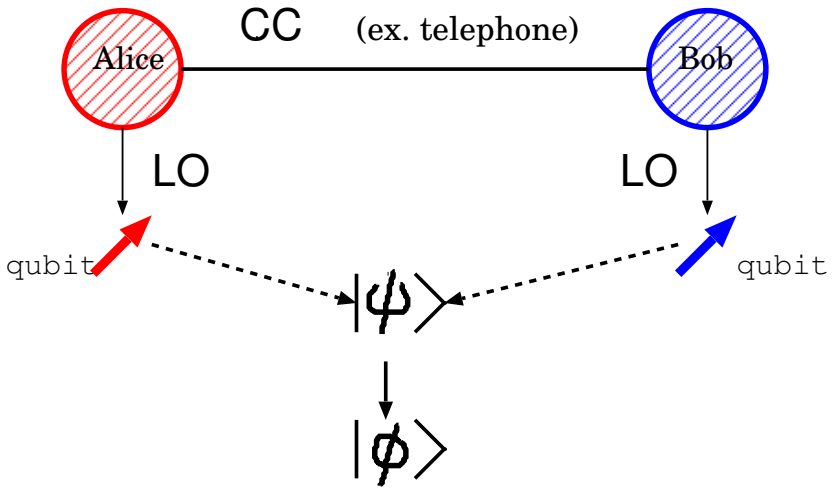

図 2.1　アリスとボブができる LOCC

いま、状態 $|\psi\rangle$、$|\phi\rangle$ は次のようにおく。

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.1}$$

$$|\phi\rangle = \cos\theta|00\rangle + \sin\theta|11\rangle \tag{2.2}$$

このとき、1 つ目の qubit をアリス、2 つ目の qubit をボブが持っているとする。アリスは測定に次の直交した qubit を用いる。

$$|\alpha\rangle = \cos\alpha|0\rangle + \sin\alpha|1\rangle \tag{2.3}$$

$$|\alpha'\rangle = -\sin\alpha|0\rangle + \cos\alpha|1\rangle \tag{2.4}$$

(2.3)、(2.4) 式から、

$$|0\rangle = \cos\alpha|\alpha\rangle - \sin\alpha|\alpha'\rangle \tag{2.5}$$

$$|1\rangle = \sin\alpha|\alpha\rangle + \cos\alpha|\alpha'\rangle \tag{2.6}$$

(2.5)、(2.6) 式を (2.1) 式のアリスの qubit に代入すると、

$$\begin{aligned} & \frac{1}{\sqrt{2}}\{(\cos\alpha|\alpha\rangle - \sin\alpha|\alpha'\rangle)|0\rangle + (\sin\alpha|\alpha\rangle + \cos\alpha|\alpha'\rangle)|1\rangle\} \\ = \ & \frac{1}{\sqrt{2}}\{|\alpha\rangle(\cos\alpha|0\rangle + \sin\alpha|1\rangle) + |\alpha'\rangle(-\sin\alpha|0\rangle + \cos\alpha|1\rangle) \\ = \ & \frac{1}{\sqrt{2}}(|\alpha\rangle|\alpha\rangle + |\alpha'\rangle|\alpha'\rangle) \end{aligned} \tag{2.7}$$

となる。つまり、$\frac{1}{2}$ の確率で $|\alpha\rangle|\alpha\rangle$ と $|\alpha'\rangle|\alpha'\rangle$ が測定される。

測定後、状態は壊れるので、測定後の状態は、それぞれ $|\alpha\rangle|\alpha\rangle$、$|\alpha'\rangle|\alpha'\rangle$ となり、次のようになる。

$$\begin{aligned} |\alpha\rangle|\alpha\rangle &= (\cos\alpha|0\rangle + \sin\alpha|1\rangle)(\cos\alpha|0\rangle + \sin\alpha|1\rangle) \\ &= \cos^2\alpha|0\rangle|0\rangle + \cos\alpha\sin\alpha|0\rangle|1\rangle + \sin\alpha\cos\alpha|1\rangle|0\rangle + \sin^2\alpha|1\rangle|1\rangle \end{aligned} \tag{2.8}$$

いま、$|0\rangle|0\rangle$ と $|1\rangle|1\rangle$ のみ重ね合わせた状態にしたい。しかし、$\cos^2\alpha$ と $\sin^2\alpha$ を残し、$\cos\alpha\sin\alpha$ を 0 にすることは不可能である。よって、(2.3)、(2.4) の qubit では変換できないことが分かる。

では直交する qubit のみの測定では変換はできないのであろうか。上の例と同様にして、直交する qubit を一般的に表して考えてみる。

$$|A\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.9}$$

$$|B\rangle = -\beta^*|0\rangle + \alpha^*|1\rangle \tag{2.10}$$

(2.9)、(2.10) 式より、

$$|0\rangle = \alpha^*|A\rangle - \beta|B\rangle \tag{2.11}$$

$$|1\rangle = \beta^*|A\rangle + \alpha|B\rangle \tag{2.12}$$

となり、(2.1) 式に代入すると、

$$\frac{1}{\sqrt{2}}\{|A\rangle(\alpha^*|0\rangle + \beta^*|1\rangle) + |B\rangle(-\beta|0\rangle + \alpha|1\rangle)\} \tag{2.13}$$

となる。測定後の状態は、

$$\begin{aligned} & |A\rangle(\alpha^*|0\rangle + \beta^*|1\rangle) \\ = \ & |\alpha|^2|0\rangle|0\rangle + \alpha\beta^*|0\rangle|1\rangle + \alpha^*\beta|1\rangle|0\rangle + |\beta|^2|1\rangle|1\rangle \end{aligned} \tag{2.14}$$

となり、先ほどと同様、$|0\rangle|0\rangle$ と $|1\rangle|1\rangle$ のみ重ね合わせた状態にすることは不可能である。よって、直交する qubit のみの測定では LOCC で状態 $|\psi\rangle$ を $|\phi\rangle$ に変換できない。

2.1.2 補助 qubit を使った変換

そこで、アリスが測定する際に自分の持っている qubit の他にもう 1 つの qubit(補助 qubit) を使って測定してみる。いま、補助 qubit を

$$|\psi'\rangle = \cos\theta|0\rangle + \sin\theta|1\rangle \tag{2.15}$$

とする。また、C-NOT gate という演算方法を使用する。C-NOT gate を簡単に説明すると、2 qubit の場合、control bit と target bit が存在し、control bit が 0 ならば、target bit はそのままで、control bit が 1 ならば、target bit は反転する。

control bit	target bit		
0	0	→	00
0	1	→	01
1	0	→	11
1	1	→	10

図 2.2　C-NOT gate

補助 qubit を用いると、全系の状態は、

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)(\cos\theta|0\rangle + \sin\theta|1\rangle) \tag{2.16}$$

であり、アリスの持っている EPR pair の片方を control bit、補助 qubit を target bit として、演算を行うと、

$$\begin{aligned} & \frac{1}{\sqrt{2}}\{|00\rangle(\cos\theta|0\rangle + \sin\theta|1\rangle) + |11\rangle(\cos\theta|1\rangle + \sin\theta|0\rangle)\} \\ = \ & \frac{1}{\sqrt{2}}\{(\cos\theta|00\rangle + \sin\theta|11\rangle)|0\rangle + (\cos\theta|11\rangle + \sin\theta|00\rangle)|1\rangle\} \end{aligned} \tag{2.17}$$

である。補助 qubit の測定を行い、測定結果が $|0\rangle$ であれば、測定後の状態は $\cos\theta|00\rangle + \sin\theta|11\rangle$ となり、測定結果が $|1\rangle$ であれば、測定後の状態は $\cos\theta|11\rangle + \sin\theta|00\rangle$ となる。測定結果が $|1\rangle$ であったとき、アリスは自分の qubit を反転させる NOT gate を使用し、ボブも NOT gate を使用する。すると、アリスが得られる測定結果に関係なく、$\cos\theta|00\rangle + \sin\theta|11\rangle$ を得ることができ、LOCC で状態 $|\psi\rangle$ を $|\phi\rangle$ に変換できることになる。

2.2　$x \prec y$

ここではエンタングルメント変換と深い関係がある数学の概念、majorization について説明する。majorization は d 次元ベクトルの順序づけである。
2 つの d 次元ベクトルを、

$$x = (x_1, \ldots, x_d)$$
$$y = (y_1, \ldots, y_d)$$

とし、また、成分の大きい順から並び換えられているとする。今の場合、x_1、y_1 が最大である。$k = 1, \ldots, d$ に対して、

$$\sum_{j=1}^{k} x_j \leq \sum_{j=1}^{k} y_j \tag{2.18}$$

$$\sum_{j=1}^{d} x_j = \sum_{j=1}^{d} y_j \tag{2.19}$$

が成り立ち、$k = d$ のときに等号が成り立つならば、x は y に majorize(メジャライズ) されるといい、$x \prec y$ と書く。

いま、すべての実数 t に対して、

$$\sum_{j=1}^{d} \max(x_j - t, 0) \leq \sum_{j=1}^{d} \max(y_j - t, 0) \tag{2.20}$$

$$\text{および} \quad \sum_{j=1}^{d} x_j = \sum_{j=1}^{d} y_j \tag{2.21}$$

ならば、またそのときに限り、$x \prec y$ であることを示す。

(2.21) 式は (2.19) 式と同等であるので、(2.20) 式が (2.18) 式と同等であることを示したい。

ここで k と l は、$0 \leq k \leq d$、$0 \leq l \leq d$ の任意の数とし、$x_l - t \geq 0$ $(x_{l-1} - t < 0)$、$y_k - t \geq 0$ $(y_{k-1} - t < 0)$ とする。

i)$k < l$ のとき

$$x_1 + x_2 + \cdots + x_l - lt \leq y_1 + y_2 + \cdots + y_k - kt$$

$$x_1 + x_2 + \cdots + x_k + (x_{k+1} + \cdots + x_l - (l-k)t) \leq y_1 + y_2 + \cdots + y_k$$

$$x_1 + x_2 + \cdots + x_k + (x_{k+1} - t + \cdots + x_l - t) \leq y_1 + y_2 + \cdots + y_k$$

$(x_{k+1} - t + \cdots + x_l - t) > 0$ より

$$x_1 + x_2 + \cdots + x_k \leq y_1 + y_2 + \cdots + y_k$$

よって、

$$\sum_{j=1}^{k} x_j \leq \sum_{j=1}^{k} y_j \tag{2.22}$$

ii)$k = l$ のとき

$$x_1 + x_2 + \cdots + x_k \leq y_1 + y_2 + \cdots + y_k$$

よって、

$$\sum_{j=1}^{k} x_j \leq \sum_{j=1}^{k} y_j \tag{2.23}$$

ⅲ)$k > l$ のとき

$$x_1 + x_2 + \cdots + x_l - lt \leq y_1 + y_2 + \cdots + y_k - kt$$
$$x_1 + x_2 + \cdots + x_k - (x_{l+1} + \cdots + x_k - (k-l)t)$$
$$\leq y_1 + y_2 + \cdots + y_k$$
$$x_1 + x_2 + \cdots + x_k - (x_{l+1} - t + \cdots + x_k - t)$$
$$\leq y_1 + y_2 + \cdots + y_k$$

$-(x_{l+1} - t + \cdots + x_k - t) > 0$ より

$$x_1 + x_2 + \cdots + x_k \leq y_1 + y_2 + \cdots + y_k$$

よって、

$$\sum_{j=1}^{k} x_j \leq \sum_{j=1}^{k} y_j \tag{2.24}$$

i)、ii)、ⅲ) より、(2.20)、(2.21) 式は (2.18)、(2.19) 式と同等であると言え、そのときの x、y は $x \prec y$ である。

2.3 エンタングルメント変換と $x \prec y$ の関係

2.2 で述べたように、$x \prec y$ とエンタングルメント変換には深い関係がある。ここでは、$x \prec y$ とエンタングルメント変換がどのような関係であるか紹介する。

EPR 状態である状態 $|\psi\rangle$ を LOCC で $|\phi\rangle$ に変換したい。いま、$|\psi\rangle$、$|\phi\rangle$ は次のような状態である。

$$|\psi\rangle = \sqrt{x_0}|00\rangle + \sqrt{x_1}|11\rangle \tag{2.25}$$
$$|\phi\rangle = \sqrt{y_0}|00\rangle + \sqrt{y_1}|11\rangle \tag{2.26}$$

2.1 で述べたように、LOCC で変換するには補助 qubit が必要である。よって、使用する補助 qubit は、、

$$\text{補助 qubit} = \cos\alpha|0\rangle + \sin\alpha|1\rangle \tag{2.27}$$

であり、$|\psi\rangle$ に用いると、全系は、

$$(\sqrt{x_0}|00\rangle + \sqrt{x_1}|11\rangle)(\cos\alpha|0\rangle + \sin\alpha|1\rangle)$$
$$= (\sqrt{x_0}\cos\alpha|00\rangle + \sqrt{x_1}\sin\alpha|11\rangle)|0\rangle$$
$$+(\sqrt{x_0}\sin\alpha|00\rangle + \sqrt{x_1}\cos\alpha|11\rangle)|1\rangle$$

となり、補助 qubit を次のように Unitary 変換すると、

$$U_\theta|0\rangle = \cos\theta|0\rangle + \sin\theta|1\rangle \tag{2.28}$$
$$U_\theta|1\rangle = -\sin\theta|0\rangle + \cos\theta|1\rangle \tag{2.29}$$

$$
\begin{aligned}
&(\sqrt{x_0}\cos\alpha|\,00\,\rangle + \sqrt{x_1}\sin\alpha|\,11\,\rangle)(\cos\theta|\,0\,\rangle + \sin\theta|\,1\,\rangle) \\
&\quad +(\sqrt{x_0}\sin\alpha|\,00\,\rangle + \sqrt{x_1}\cos\alpha|\,11\,\rangle)(-\sin\theta|\,0\,\rangle + \cos\theta|\,1\,\rangle) \\
&= (\sqrt{x_0}\cos\alpha\cos\theta|\,00\,\rangle + \sqrt{x_1}\sin\alpha\cos\theta|\,11\,\rangle \\
&\qquad -\sqrt{x_0}\sin\alpha\sin\theta|\,00\,\rangle - \sqrt{x_1}\cos\alpha\sin\theta|\,11\,\rangle)|\,0\,\rangle \\
&\quad +(\sqrt{x_0}\cos\alpha\sin\theta|\,00\,\rangle + \sqrt{x_1}\sin\alpha\sin\theta|\,11\,\rangle \\
&\qquad +\sqrt{x_0}\sin\alpha\cos\theta|\,00\,\rangle + \sqrt{x_1}\cos\alpha\cos\theta|\,11\,\rangle)|\,1\,\rangle
\end{aligned}
$$

ここで、

$$\beta_0(\sqrt{y_0}|\,00\,\rangle + \sqrt{y_1}|\,11\,\rangle)|\,0\,\rangle + \beta_1(\sqrt{y_1}|\,00\,\rangle + \sqrt{y_0}|\,11\,\rangle)|\,1\,\rangle \tag{2.30}$$

$$(|\beta_0|^2 + |\beta_1|^2 = 1) \tag{2.31}$$

の状態ならば、2.1.2 で表したように補助 qubit の測定を行えば、測定後の状態として、状態 $\sqrt{y_0}|\,00\,\rangle + \sqrt{y_1}|\,11\,\rangle$ を得ることができる。

$$\sqrt{x_0}\cos\alpha\cos\theta - \sqrt{x_0}\sin\alpha\sin\theta = \beta_0\sqrt{y_0} \tag{2.32}$$

$$\sqrt{x_1}\sin\alpha\cos\theta - \sqrt{x_1}\cos\alpha\sin\theta = \beta_0\sqrt{y_1} \tag{2.33}$$

$$\sqrt{x_0}\cos\alpha\sin\theta + \sqrt{x_0}\sin\alpha\cos\theta = \beta_1\sqrt{y_1} \tag{2.34}$$

$$\sqrt{x_1}\sin\alpha\sin\theta + \sqrt{x_1}\cos\alpha\cos\theta = \beta_1\sqrt{y_0} \tag{2.35}$$

よって、上式を満たす α、θ が存在すればよい。
(2.32)、(2.33)、(2.34)、(2.35) 式は、和積の公式より、

$$\cos(\alpha+\theta) = \beta_0\frac{\sqrt{y_0}}{\sqrt{x_0}} \tag{2.36}$$

$$\sin(\alpha-\theta) = \beta_0\frac{\sqrt{y_1}}{\sqrt{x_1}} \tag{2.37}$$

$$\cos(\alpha-\theta) = \beta_1\frac{\sqrt{y_0}}{\sqrt{x_1}} \tag{2.38}$$

$$\sin(\alpha+\theta) = \beta_1\frac{\sqrt{y_1}}{\sqrt{x_0}} \tag{2.39}$$

である。(2.36)、(2.39) の右辺をベクトルの成分としたとき内積は、

$$\beta_0^2\frac{y_0}{x_0} + \beta_1^2\frac{y_1}{x_0} \tag{2.40}$$

となり、(2.36) の 2 乗と (2.39) の 2 乗を足すと、

$$x_0 = \beta_0^2 y_0 + \beta_1^2 y_1 \tag{2.41}$$

より (2.40) は 1 となり単位ベクトルをとることが分かる。したがって、角度 $(\alpha + \theta)$ は存在することが証明できた。また同様に、(2.37) の 2 乗と (2.38) の 2 乗を足すと、

$$x_1 = \beta_0^2 y_1 + \beta_1^2 y_0 \tag{2.42}$$

を得、(2.41)、(2.42) 式から、

$$\beta_0^2 = \frac{y_1 x_0 - y_0 x_1}{y_1^2 - y_0^2} \tag{2.43}$$

$$\beta_1^2 = \frac{y_0 x_0 - y_1 x_1}{y_0^2 - y_1^2} \tag{2.44}$$

となる。いま、$x \prec y$ が成り立つとして、

i) $y_0 > y_1$、$x_0 > x_1$ のとき

$$x_0 \leq y_0 \tag{2.45}$$

$$x_0 + x_1 = y_0 + y_1 = 1 \tag{2.46}$$

(2.46) より、

$$\begin{aligned} & y_1 x_0 - y_0 x_1 \\ = \ & (1 - y_0) x_0 - y_0 (1 - x_0) \\ = \ & x_0 - y_0 \end{aligned}$$

(2.45) より $\quad < \ 0 \tag{2.47}$

(2.47) 式、$y_1^2 - y_0^2 < 0$ より、 $\beta_0^2 \geq 0$ であり、同様にして $\beta_1^2 \geq 0$ である。つまり、 $x \prec y$ であれば、 $\beta_0^2 \geq 0$ 、$\beta_1^2 \geq 0$ となり式は成立する。

また、ii) $y_0 > y_1$,$x_1 > x_0$、 iii)$y_1 > y_0$,$x_0 > x_1$、 iv)$y_1 > y_0$,$x_1 > x_0$ のときも同様のことが言える。以上のことから $x \prec y$ であれば、EPR 状態 $|\psi\rangle$ を LOCC で別の状態 $|\phi\rangle$ に変換できることが分かった。

また、2 準位系のみならず、一般の場合にも $x \prec y$ であれば、エンタングルメントを変換できるということが分かっており、さらにエンタングルメント変換が可能ならば、そのとき $x \prec y$ であることも分かっている [参考文献.3 303-310]。

$$|\psi\rangle \underset{\text{LOCC}}{\longrightarrow} |\phi\rangle \quad \Longleftrightarrow \quad x \prec y$$

2.4 まとめ

EPR 状態 $|\psi\rangle$ を $|\phi\rangle$ に変換しようと思うと、自分の qubit の測定や、電話などの LOCC のみではできない。LOCC で変換するには補助 qubit が必要であり、そのことは本章で証明できた。また、エンタングルメントを変換する際に深い関係があると言われている majorization の紹介をし、$x \prec y$ であれば、状態 $|\psi\rangle$ を $|\phi\rangle$ に変換できることを説明した。

次章では $x \prec y$ とエンタングルメント変換の関係を利用して、さらにエンタングルメントの変換について考察する。

第3章　触媒作用

本章では、$x \prec y$ とエンタングルメント変換の関係を利用し、触媒という新しい要素を取り入れて、さらにエンタングルメントの変換について考察する。

3.1　触媒

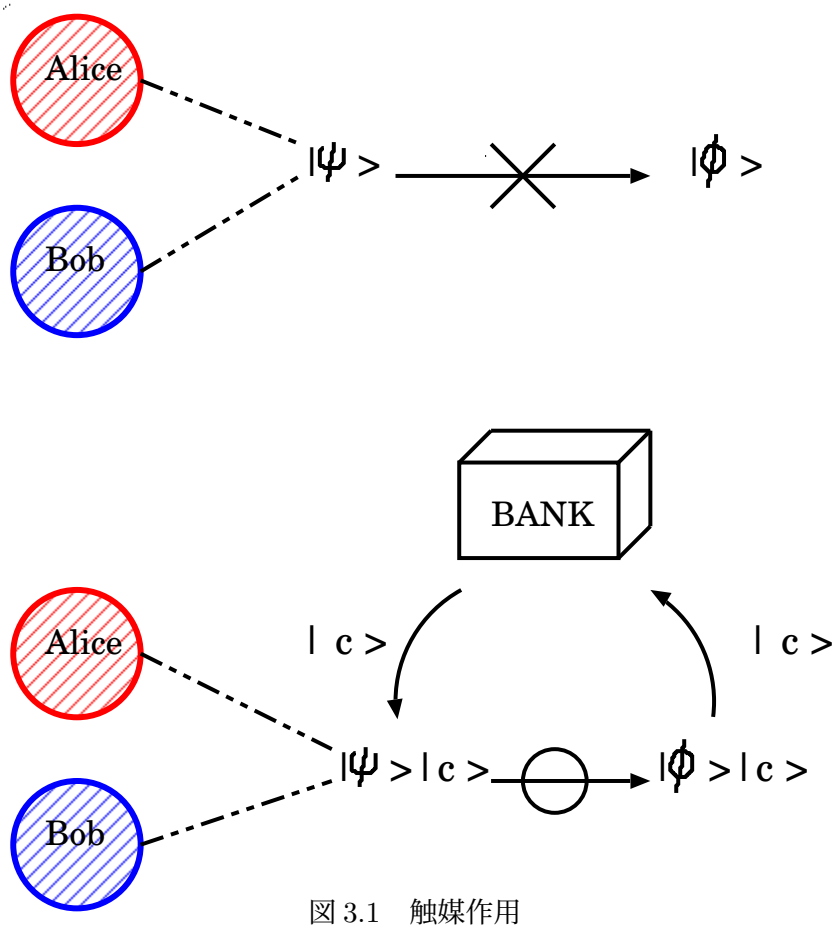

図 3.1　触媒作用

はじめに、本章で扱う触媒作用というおもしろい現象について、紹介する。このエンタングルメントの触媒は、Jonathan,Plenio により発見された。[参考文献.7]

いま、アリスとボブは状態 $|\psi\rangle = \sqrt{0.4}|00\rangle + \sqrt{0.4}|11\rangle + \sqrt{0.1}|22\rangle + \sqrt{0.1}|33\rangle$ にある一対の 4 準位システムを共有しているとする。2 人はこの状態を LOCC で $|\phi\rangle = \sqrt{0.5}|00\rangle + \sqrt{0.25}|11\rangle + \sqrt{0.25}|22\rangle$ に変換したいのだができない。しかし友好的な銀行が エンタングルした状態にある qubit 対 $|c\rangle = \sqrt{0.6}|00\rangle + \sqrt{0.4}|11\rangle$ を彼らに貸してくれたとする。アリスとボブは LOCC によって 状態 $|\psi\rangle|c\rangle$ を $|\phi\rangle|c\rangle$ に変換できて、変換終了後に $|c\rangle$ を銀行に返すことができる。このような $|c\rangle$ は化学反応の触媒作用と似ているため、触媒と呼ば

れている。本章では、この触媒について考察し、どのような場合に触媒作用が起こるのか考える。

3.2 最も簡単な系

Jonathan,Plenio によれば、4 準位システムの状態 $|\psi\rangle$ が触媒によって変換できることが示されていた。そこで、4 準位システムより簡単な系では変換できないのか検証してみることにした。状態 $|\psi\rangle$ が LOCC で $|\phi\rangle$ に変換できない、つまり、$x \prec y$ を満たさない状態 $|\psi\rangle$、$|\phi\rangle$ について考え、さらに、その中の $|\psi\rangle$、$|\phi\rangle$ で触媒を借りると状態 $|\psi\rangle|c\rangle$ が $|\phi\rangle|c\rangle$ に変換できる最も簡単な系を探すこととする。

まず、状態 $|\psi\rangle$、 $|\phi\rangle$ は次のような 2 準位システムの qubit 対であるとする。

$$|\psi\rangle = \sqrt{x_0}|00\rangle + \sqrt{x_1}|11\rangle \qquad (x_0 \geq x_1) \tag{3.1}$$

$$|\phi\rangle = \sqrt{y_0}|00\rangle + \sqrt{y_1}|11\rangle \qquad (y_0 \geq y_1) \tag{3.2}$$

また、触媒 $|c\rangle$ は

$$|c\rangle = \sqrt{\alpha_0}|00\rangle + \sqrt{\alpha_1}|11\rangle \qquad (\alpha_0 \geq \alpha_1) \tag{3.3}$$

とおく。$x \prec y$ の式は、

$$x_0 \leq y_0 \tag{3.4}$$

$$x_0 + x_1 = y_0 + y_1 = 1 \tag{3.5}$$

である。いま。$x \prec y$ を満たさないのだが、(3.5) 式は規格化条件で絶対のものなので、(3.4) 式を満たさないこととなる。よって、最初の条件は、

$$x_0 > y_0 \tag{3.6}$$

$$x_0 + x_1 = y_0 + y_1 = 1 \tag{3.7}$$

である。状態 $|\psi\rangle$、$|\phi\rangle$ に触媒を作用させると、

$$|\psi\rangle|c\rangle = \sqrt{x_0\alpha_0}|0000\rangle + \sqrt{x_1\alpha_0}|1010\rangle + \sqrt{x_0\alpha_1}|0101\rangle + \sqrt{x_1\alpha_1}|1111\rangle \tag{3.8}$$

$$|\phi\rangle|c\rangle = \sqrt{y_0\alpha_0}|0000\rangle + \sqrt{y_1\alpha_0}|1010\rangle + \sqrt{y_0\alpha_1}|0101\rangle + \sqrt{y_1\alpha_1}|1111\rangle \tag{3.9}$$

となる。このとき $x\alpha \prec y\alpha$ を満たしていれば、状態 $|\psi\rangle|c\rangle$ を $|\phi\rangle|c\rangle$ に変換できる。$x\alpha$ 、$y\alpha$ を大きい順に比較してみるとまず、

$$x_0\alpha_0 \leq y_0\alpha_0 \tag{3.10}$$

を得る。しかしこれは (3.6) 式と矛盾している。よって $x \prec y$ を満たさない x、y が同時に $x\alpha \prec y\alpha$ を満たすことはなく、2 準位システムの 状態 $|\psi\rangle$、 $|\phi\rangle$ では LOCC で変換できないことが分かった。

次に、状態 $|\psi\rangle$、 $|\phi\rangle$ を 3 準位システムの qubit 対であるとする。

$$|\psi\rangle = \sqrt{x_0}|00\rangle + \sqrt{x_1}|11\rangle + \sqrt{x_2}|22\rangle \quad (x_0 \geq x_1 \geq x_2) \tag{3.11}$$
$$|\phi\rangle = \sqrt{y_0}|00\rangle + \sqrt{y_1}|11\rangle + \sqrt{y_2}|22\rangle \quad (y_0 \geq y_1 \geq y_2) \tag{3.12}$$

触媒 $|c\rangle$ は (3.3) 式を用いる。このとき $x \prec y$ の式は、

$$x_0 \leq y_0 \tag{3.13}$$
$$x_0 + x_1 \leq y_0 + y_1 \tag{3.14}$$
$$x_0 + x_1 + x_2 = y_0 + y_1 + y_2 = 1 \tag{3.15}$$

である。先ほどと同様、$x \prec y$ を満たさない x、y を考える。2 準位システムの結果から、(3.14) 式のみを満たさないこととなる。よって、3 準位システムの最初の条件は、

$$x_0 \leq y_0 \tag{3.16}$$
$$x_0 + x_1 > y_0 + y_1 \tag{3.17}$$
$$x_0 + x_1 + x_2 = y_0 + y_1 + y_2 = 1 \tag{3.18}$$

である。状態 $|\psi\rangle|c\rangle$、$|\phi\rangle|c\rangle$ の $x\alpha$ 、$y\alpha$ を大きい順に比較する。最大である $x\alpha$ 、$y\alpha$ と、最小である $x\alpha$ 、$y\alpha$ は明らかなので、最後から 2 番目の不等式に注目してみると、

$$\begin{aligned} 1 - \alpha_1 x_2 &\leq 1 - \alpha_1 y_2 \\ x_2 &\geq y_2 \end{aligned} \tag{3.19}$$

を得る。ここで、最初の条件 (3.17)、(3.18) 式より、

$$x_2 < y_2 \tag{3.20}$$

でなければならないので、(3.19) 式は (3.20) 式と矛盾している。よって $x \prec y$ を満たさない x、y が同時に $x\alpha \prec y\alpha$ を満たすことはなく、3 準位システムの状態 $|\psi\rangle$、 $|\phi\rangle$ では LOCC で変換できないことが分かった。

$$\begin{array}{lll} |\psi\rangle \not\rightarrow |\phi\rangle & x \not\prec y & \\ |\psi\rangle|c\rangle \rightarrow |\phi\rangle|c\rangle & x\alpha \prec y\alpha & \Rightarrow \text{ 同時に満たさない} \end{array}$$

よって、4 準位システムの状態 $|\psi\rangle$、 $|\phi\rangle$ が最も簡単な系であると思われる。

では、4 準位システムの状態 $|\psi\rangle$、 $|\phi\rangle$ についても調べてみる。4 準位システムの状態 $|\psi\rangle$、 $|\phi\rangle$ は、

$$|\psi\rangle = \sqrt{x_0}|00\rangle + \sqrt{x_1}|11\rangle + \sqrt{x_2}|22\rangle + \sqrt{x_3}|33\rangle \quad (x_0 \geq x_1 \geq x_2 \geq x_3) \tag{3.21}$$

$$|\phi\rangle = \sqrt{y_0}|00\rangle + \sqrt{y_1}|11\rangle + \sqrt{y_2}|22\rangle + \sqrt{y_3}|33\rangle \quad (y_0 \geq y_1 \geq y_2 \geq y_3) \tag{3.22}$$

となり、触媒 $|c\rangle$ は (3.3) 式を用いる。このとき $x \prec y$ の式は、

$$x_0 \leq y_0 \tag{3.23}$$

$$x_0 + x_1 \leq y_0 + y_1 \tag{3.24}$$

$$x_0 + x_1 + x_2 \leq y_0 + y_1 + y_2 \tag{3.25}$$

$$x_0 + x_1 + x_2 + x_3 = y_0 + y_1 + y_2 + y_3 = 1 \tag{3.26}$$

である。先ほどと同様、$x \prec y$ を満たさない x、y を考える。2 準位、3 準位システムの結果から、(3.24) 式のみを満たさないこととなる。よって、4 準位システムの最初の条件は、

$$x_0 \leq y_0 \tag{3.27}$$

$$x_0 + x_1 > y_0 + y_1 \tag{3.28}$$

$$x_0 + x_1 + x_2 \leq y_0 + y_1 + y_2 \tag{3.29}$$

$$x_0 + x_1 + x_2 + x_3 = y_0 + y_1 + y_2 + y_3 = 1 \tag{3.30}$$

である。先ほどまでと同様、状態 $|\psi\rangle|c\rangle$、$|\phi\rangle|c\rangle$ の $x\alpha$ 、$y\alpha$ を大きい順に比較する。$x\alpha$ 、$y\alpha$ の大小関係は様々な場合が考えられるので、場合分けをし比較してみた。すると、明らかな矛盾は見つからず、3.1 で紹介した例より、4 準位システムの状態 $|\psi\rangle$、 $|\phi\rangle$ であれば、LOCC で変換できるものがあることが確認できた。

$$\begin{array}{ll} |\psi\rangle \not\rightarrow |\phi\rangle & x \not\prec y \\ |\psi\rangle|c\rangle \rightarrow |\phi\rangle|c\rangle & x\alpha \prec y\alpha \qquad \Rightarrow \text{ 同時に満たすものがある} \end{array}$$

しかし、場合分けをし得られた複数の不等式から状態 $|\psi\rangle$ を $|\phi\rangle$ に変換できなくも、状態 $|\psi\rangle|c\rangle$ を $|\phi\rangle|c\rangle$ に変換できる、すなわち、$x \not\prec y$ であり、$x\alpha \prec y\alpha$ を満たす状態 $|\psi\rangle$、$|\phi\rangle$ の新たな条件を導き出すことはできなかった。

したがって、次節では 4 準位システムでの最初の条件 (3.27)、(3.28)、(3.29)、(3.30) 式 ― 今後条件 A とする ― を満たす具体的な x、y から $x\alpha \prec y\alpha$ をも満たす組み合わせを探すこととする。

3.3 触媒を使った変換

3.1 で紹介した状態 $|\psi\rangle$、$|\phi\rangle$、 $|c\rangle$ 以外に、4 準位システムの条件 A を満たす状態 $|\psi\rangle$、$|\phi\rangle$ から状態 $|\psi\rangle|c\rangle$ を $|\phi\rangle|c\rangle$ に変換できる x、y の組み合わせを探す。x、y は自分で適

当に決める。その x、y が $x\alpha \prec y\alpha$ を満たしていれば、状態 $|\psi\rangle|c\rangle$ を $|\phi\rangle|c\rangle$ に変換できるとみなすことができるので調べてみる。ここで、触媒 $|c\rangle$ も自分で決めてみたのだが、自分で決めた状態 $|\psi\rangle$、 $|\phi\rangle$、$|c\rangle$ では $x\alpha \prec y\alpha$ を満たさなかった。

そこで、自分で決めた x、y で $x\alpha \prec y\alpha$ を満たす触媒 $|c\rangle$、つまり α_0 、 α_1 が存在するのか、数値的に調べてみることにした。

すると、α_0 、 α_1 が存在する x、y の組み合わせと、存在しない x、y の組み合わせがあることが確認できた。

α_0 、 α_1 が存在した x、y の組み合わせの例

$$\begin{cases} (x_0, x_1, x_2, x_3) = (0.50,\ 0.40,\ 0.05,\ 0.05) \\ (y_0, y_1, y_2, y_3) = (0.70,\ 0.15,\ 0.15,\ 0.00) \\ \qquad\qquad \text{ex.}\alpha_0 = 0.70, \alpha_1 = 0.30 \end{cases}$$

$$\begin{cases} (x_0, x_1, x_2, x_3) = (0.60,\ 0.30,\ 0.05,\ 0.05) \\ (y_0, y_1, y_2, y_3) = (0.70,\ 0.15,\ 0.15,\ 0.00) \\ \qquad\qquad \alpha_0 = 0.63, \alpha_1 = 0.37 \end{cases}$$

α_0 、 α_1 が存在しなかった x、y の組み合わせの例

$$\begin{cases} (x_0, x_1, x_2, x_3) = (0.45,\ 0.35,\ 0.10,\ 0.10) \\ (y_0, y_1, y_2, y_3) = (0.50,\ 0.25,\ 0.25,\ 0.00) \end{cases}$$

$$\begin{cases} (x_0, x_1, x_2, x_3) = (0.60,\ 0.30,\ 0.05,\ 0.05) \\ (y_0, y_1, y_2, y_3) = (0.70,\ 0.15,\ 0.10,\ 0.05) \end{cases}$$

$$\begin{cases} (x_0, x_1, x_2, x_3) = (0.50,\ 0.40,\ 0.05,\ 0.05) \\ (y_0, y_1, y_2, y_3) = (0.70,\ 0.15,\ 0.05,\ 0.05) \end{cases}$$

以上のことから 4 準位システムの 状態 $|\psi\rangle$、 $|\phi\rangle$ で触媒作用によって、状態 $|\psi\rangle|c\rangle$ を $|\phi\rangle|c\rangle$ に変換できるものが存在することが確認できた。しかし、x、y が条件 A を満たしても、$x\alpha \prec y\alpha$ を満たすとは限らず、状態 $|\psi\rangle|c\rangle$ を $|\phi\rangle|c\rangle$ に変換できるとは限らないことが分かった。すなわち、条件 A は状態 $|\psi\rangle|c\rangle$ を $|\phi\rangle|c\rangle$ に変換する際の必要条件であり、十分条件でないことが分かる。また、計算機の結果から、状態 $|\psi\rangle|c\rangle$ が $|\phi\rangle|c\rangle$ に変換できるときの条件 A 以外の条件がないか調べてみたが、α_0、α_1 が存在した x、y の組み合わせから新しい条件を導き出すことはできず、分かる条件は条件 A のみであった。

3.4 まとめ

本章では、状態 $|\psi\rangle$ を LOCC で $|\phi\rangle$ に変換できないものが触媒によって、状態 $|\psi\rangle|c\rangle$ を LOCC で $|\phi\rangle|c\rangle$ に変換できる触媒作用について考察した。すると、変換できる最も簡単な系は 4 準位システムの状態 $|\psi\rangle$、$|\phi\rangle$ であること分かり、条件 A を導くことができた。また変換できる具体的な x、y を探しだし、状態 $|\psi\rangle|c\rangle$ を LOCC で $|\phi\rangle|c\rangle$ に変換できる触媒 $|c\rangle$ の存在を確認できた。また、導かれた条件 A は必要条件であり、十分条件でないことが分かった。

第4章　自己触媒

第3章では、状態 $|\psi\rangle|c\rangle$ を LOCC で $|\phi\rangle|c\rangle$ に変換できる触媒が存在することが確認できた。そこで、本章ではさらに発展させ、触媒のかわりに自分自身を触媒として用いても変換できるのか考える。つまり、状態 $|\psi\rangle$1 つだけでは $|\phi\rangle$ に変換できないのに、状態 $|\psi\rangle$ を 2 つにすると、 $|\phi\rangle$2 つに変換できるか調べる。ここで、自分自身の触媒のことを『自己触媒』と名付けることとする。

$$
\begin{aligned}
|\psi\rangle &\not\to |\phi\rangle \quad x \not\prec y \\
|\psi\rangle|\psi\rangle &\to |\phi\rangle|\phi\rangle \quad xx \prec yy
\end{aligned}
$$

4.1　最も簡単な系

まず、状態 $|\psi\rangle$ が $|\phi\rangle$ に変換できなく、状態 $|\psi\rangle|\psi\rangle$ を LOCC で $|\phi\rangle|\phi\rangle$ に変換できる状態 $|\psi\rangle|\phi\rangle$ を探したい。そのために変換できる最も簡単な系を探したい。

まず、2 準位、3 準位システムの状態 $|\psi\rangle$、 $|\phi\rangle$ を考える。

2 準位システム

$$
\begin{aligned}
|\psi\rangle &= \sqrt{x_0}|00\rangle + \sqrt{x_1}|11\rangle \qquad (x_0 \geq x_1) && (4.1)\\
|\phi\rangle &= \sqrt{y_0}|00\rangle + \sqrt{y_1}|11\rangle \qquad (y_0 \geq y_1) && (4.2)
\end{aligned}
$$

3 準位システム

$$
\begin{aligned}
|\psi\rangle &= \sqrt{x_0}|00\rangle + \sqrt{x_1}|11\rangle + \sqrt{x_2}|22\rangle \quad (x_0 \geq x_1 \geq x_2) && (4.3)\\
|\phi\rangle &= \sqrt{y_0}|00\rangle + \sqrt{y_1}|11\rangle + \sqrt{y_2}|22\rangle \quad (y_0 \geq y_1 \geq y_2) && (4.4)
\end{aligned}
$$

このとき、先ほどの触媒と同様 $x \not\prec y$ と、 $xx \prec yy$ を同時に満たすことはできない。よって、2 準位、3 準位システムの状態 $|\psi\rangle$、 $|\phi\rangle$ では LOCC で状態 $|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle$ に変換できないことが分かった。また、4 準位システムは、

$$
\begin{aligned}
|\psi\rangle &= \sqrt{x_0}|00\rangle + \sqrt{x_1}|11\rangle + \sqrt{x_2}|22\rangle + \sqrt{x_3}|33\rangle && (4.5)\\
&\qquad\qquad (x_0 \geq x_1 \geq x_2 \geq x_3) \\
|\phi\rangle &= \sqrt{y_0}|00\rangle + \sqrt{y_1}|11\rangle + \sqrt{y_2}|22\rangle + \sqrt{y_3}|33\rangle && (4.6)\\
&\qquad\qquad (y_0 \geq y_1 \geq y_2 \geq y_3)
\end{aligned}
$$

であり、$x \not\prec y$ と、 $xx \prec yy$ を同時に満たそうと思うと矛盾は見つからなかった。すなわち、触媒同様、4 準位システムが状態 $|\psi\rangle$ が $|\phi\rangle$ に変換できなくも、状態 $|\psi\rangle|\psi\rangle$ を LOCC

で $|\phi\rangle|\phi\rangle$ に変換できる可能性のある最も簡単な系であることが分かった。また 4 準位システムの最初の条件も、同じく条件 A となった。

＜条件 A ＞

$$x_0 \leq y_0 \tag{4.7}$$

$$x_0 + x_1 > y_0 + y_1 \tag{4.8}$$

$$x_0 + x_1 + x_2 \leq y_0 + y_1 + y_2 \tag{4.9}$$

$$x_0 + x_1 + x_2 + x_3 = y_0 + y_1 + y_2 + y_3 = 1 \tag{4.10}$$

したがって、次節では 4 準位システムでの＜条件 A ＞を満たす具体的な x、y の中から $xx \prec yy$ をも満たす組み合わせを探すこととする。

4.2 自己触媒を使った変換

4 準位システムの条件 A を満たす状態 $|\psi\rangle$、 $|\phi\rangle$ から状態 $|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle$ に変換できる x、y の組み合わせを探す。x、y は触媒で用いた、自分で適当に決めたものを使う。その x、y が $xx \prec yy$ を満たしていれば、状態 $|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle$ に変換できるとみなすことができるので調べてみる。しかし、複数の x、y の組み合わせで調べてみたのだが、自分で決めた状態 $|\psi\rangle|\phi\rangle$ では $xx \prec yy$ を満たさず、自分で見つけるのは困難であった。

そこで、数値的に乱数を使って x、y を決め、16 の不等式を計算機でチェックし、$xx \prec yy$ であるか調べた。すると、状態 $|\psi\rangle$ が $|\phi\rangle$ に変換できないのに、状態 $|\psi\rangle|\psi\rangle$ なら $|\phi\rangle|\phi\rangle$ に変換できる、つまり $x \not\prec y$ と、 $xx \prec yy$ を同時に満たす、状態 $|\psi\rangle$、 $|\phi\rangle$ を探しだすことができた。

乱数 [通り]	$x \not\prec y$ かつ $xx \prec yy$ である $\|\psi\rangle$ $\|\phi\rangle$ [通り]
1,000	2
100000	233

よって、自分自身が触媒となることはでき、自己触媒は存在するということが証明できた。しかし、上で述べたように、4 準位システムの条件 A を満たす x、y でも $xx \prec yy$ を満たさずに変換できないものも存在した。つまり、触媒同様条件 A は必要条件であり、十分条件ではなかった。

4.3　自己触媒を増やしていったときの関係

前節では、自己触媒が存在することが分かった。自己触媒を 1 つ使うと変換できることから 2 つ以上使った場合でも変換できると思われる。そこで本節では自己触媒を異なる数使った場合にその関係はどうなっているか考察する。

4.3.1　自己触媒 1 つと 2 つ

まず自己触媒を 1 つ使ったときと、2 つ使ったときの関係について考察する。

$$
\begin{array}{ccc}
|\psi\rangle \not\to |\phi\rangle & x \not\prec y & \\
|\psi\rangle|\psi\rangle \to |\phi\rangle|\phi\rangle & xx \prec yy & \\
\Downarrow \Uparrow & & \text{関係は??} \\
|\psi\rangle|\psi\rangle|\psi\rangle \to |\phi\rangle|\phi\rangle|\phi\rangle & xxx \prec yyy &
\end{array}
$$

乱数を使って、x、 y を決め、 $x \not\prec y$ と、 $xx \prec yy$ を満たす x、 y のうち、$xxx \prec yyy$ を満たすものがあるか調べた。すると、すべて満たす結果となったので、別のプログラムを作り、$xxx \prec yyy$ を満たすものと $xxx \prec yyy$ を満たさない組合せの数を調べた。また、反対からのことも調べた。

乱数 [通り]	$xx \prec yy$	$xxx \prec yyy$	$xxx \not\prec yyy$
1,000	2	2	0
10,000	25	25	0
100,000	233	233	0
1,000,000	2,458	2,458	0
10,000,000	24,810	24,810	0
100,000,000	249,033	249,033	0

乱数 [通り]	$xxx \prec yyy$	$xx \prec yy$	$xx \not\prec yy$
1,000	5	2	3
10,000	44	25	19
100,000	404	233	171
1,000,000	4,389	2,458	1,931
10,000,000	44,585	24,810	19,775
100,000,000	448,025	249,033	198,992

すると、$x \not\prec y$ と、 $xx \prec yy$ を満たす x、 y はすべて $xxx \prec yyy$ を満たす結果となった。そこで、x、 y が $xx \prec yy$ を満たすならば、$xxx \prec yyy$ も満たすということを理論的に証明したい。

まず、x、 y が $xx \prec yy$ を満たすときの 16 の不等式と、$xxx \prec yyy$ を満たすときの 64 の不等式から、両方満たすときの条件を導こうとした。しかし、各不等式にはそれぞれ場合分けが必要であることと、不等式が多いことから、導き出すことはできなかった。

次に、第 2 章で紹介した、$x \prec y$ と同等である (2.20)、(2.21) 式を使ってみる。

$$\sum_{j=1}^{d} \max(x_j - t, 0) \quad \leq \quad \sum_{j=1}^{d} \max(y_j - t, 0) \tag{2.20}$$

$$\text{および} \quad \sum_{j=1}^{d} x_j \quad = \quad \sum_{j=1}^{d} y_j \tag{2.21}$$

いま、(2.20) 式を使って $xx \prec yy$ を表すと、

$$\sum_{i,j}^{d} \max(x_i x_j - t, 0) \quad \leq \quad \sum_{i,j}^{d} \max(y_i y_j - t, 0) \tag{4.11}$$

となり、$xxx \prec yyy$ を表すと、

$$\sum_{i,j,k}^{d} \max(x_i x_j x_k - t, 0) \quad \leq \quad \sum_{i,j,k}^{d} \max(y_i y_j y_k - t, 0) \tag{4.12}$$

となる。(4.11) 式が成り立つと (4.12) 式も成り立つことを証明したい。
(4.12) 式の左辺を考えると、

$$\begin{aligned} & \sum_{i,j,k}^{d} \max(x_i x_j x_k - t, 0) \\ = \quad & \sum_{i} x_i \sum_{j,k} \max(x_j x_k - \frac{t}{x_i}, 0) \end{aligned}$$

(4.11) 式より、

$$\leq \quad \sum_{i} x_i \sum_{j,k} \max(y_j y_k - \frac{t}{x_i}, 0) \tag{4.13}$$

$$= \quad \sum_{i,j,k} \max(x_i y_j y_k - t, 0) \tag{4.14}$$

$$\tag{4.15}$$

ここで、(4.12) 式の右辺にもっていくことはできず、(4.11) 式が成り立つと (4.12) 式も成り立つことは証明できなかった。

4.3.2 自己触媒 2 つと 3 つ

次に、自己触媒を 2 つ使ったときと、3 つ使ったときの関係について考察する。

$$
\begin{array}{ll}
|\psi\rangle \not\rightarrow |\phi\rangle & x \not\prec y \\
|\psi\rangle|\psi\rangle|\psi\rangle \rightarrow |\phi\rangle|\phi\rangle|\phi\rangle & xxx \prec yyy \\
\Downarrow \Uparrow & \text{関係は??} \\
|\psi\rangle|\psi\rangle|\psi\rangle|\psi\rangle \rightarrow |\phi\rangle|\phi\rangle|\phi\rangle|\phi\rangle & xxxx \prec yyyy
\end{array}
$$

方法は同様で、 $x \not\prec y$ と、 $xxx \prec yyy$ を満たす x、 y のうち、$xxxx \prec yyyy$ を満たすものと $xxxx \prec yyyy$ を満たさない組合せの数を調べた。また、反対からのことも調べてみた。

乱数 [通り]	$xxx \prec yyy$	$xxxx \prec yyyy$	$xxxx \not\prec yyyy$
1,000	5	5	0
10,000	44	44	0
100,000	404	400	4
1,000,000	4,389	4,340	49
10,000,000	44,585	44,096	489
100,000,000	448,025	443,060	4,965

乱数 [通り]	$xxxx \prec yyyy$	$xxx \prec yyy$	$xxx \not\prec yyy$
1,000	5	5	0
10,000	50	44	6
100,000	494	400	94
1,000,000	5,309	4,340	969
10,000,000	53,464	44,096	9,368
100,000,000	535,060	443,060	92,000

すると、$x \not\prec y$ と、 $xxx \prec yyy$ を満たしても $xxxx \prec yyyy$ を満たすとは限らず、状態 $|\psi\rangle|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle|\phi\rangle$ に変換できても、$|\psi\rangle|\psi\rangle|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle|\phi\rangle|\phi\rangle$ に変換できるとは限らないことが分かった。

4.3.3 自己触媒 1 つと 3 つ

さらに、自己触媒を 1 つ使ったときと、3 つ使ったときの関係について考察する。今回、$xx \prec yy$ を満たすものは $xxxx \prec yyyy$ を満たすのは明らかであるので、$xxxx \prec yyyy$ を満たすとき $xx \prec yy$ を満たすのか調べてみる。

$$
\begin{array}{ccc}
|\psi\rangle \not\rightarrow |\phi\rangle \;\; x \not\prec y & & \\
|\psi\rangle|\psi\rangle \;\rightarrow\; |\phi\rangle|\phi\rangle \;\; xx \prec yy & & \\
\Uparrow & & \text{関係は??} \\
|\psi\rangle|\psi\rangle|\psi\rangle|\psi\rangle \;\rightarrow\; |\phi\rangle|\phi\rangle|\phi\rangle|\phi\rangle \;\; xxxx \prec yyyy &
\end{array}
$$

方法は同様で、$x \not\prec y$ と、$xxxx \prec yyyy$ を満たす x、y のうち、$xx \prec yy$ を満たすものと $xx \prec yy$ を満たさない組合せの数を調べた。

乱数 [通り]	$xxxx \prec yyyy$	$xx \prec yy$	$xx \not\prec yy$
1,000	5	2	3
10,000	50	25	25
100,000	494	233	261
1,000,000	5,309	2,458	2,851
10,000,000	53,464	24,810	28,654
100,000,000	535,060	249,033	286,027

すると、$x \not\prec y$ と、$xxxx \prec yyyy$ を満たしても $xx \prec yy$ を満たすとは限らず、状態 $|\psi\rangle|\psi\rangle|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle|\phi\rangle|\phi\rangle$ に変換できても、$|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle$ に変換できるとは限らないことが分かった。

4.4 まとめ

触媒同様、状態 $|\psi\rangle$ が $|\phi\rangle$ に変換できなくも、状態 $|\psi\rangle|\psi\rangle$ を LOCC で $|\phi\rangle|\phi\rangle$ に変換できる最も簡単な系は 4 準位システムであることが分かり、状態 $|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle$ に変換できる x 、 y の具体的な組合せを得ることができた。導いた条件も、条件 A となったが、同じく必要条件であって、十分条件でないことが分かった。

また、自己触媒の数を変えたとき、その関係はどうなっているか考察した。すると、$xx \prec yy$ を満たす x、y はすべて $xxx \prec yyy$ を満たす結果となり、数値的に調べた範囲では LOCC で状態 $|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle$ に変換できれば $|\psi\rangle|\psi\rangle|\psi\rangle$ も LOCC で $|\phi\rangle|\phi\rangle|\phi\rangle$ に変換できると思われる。しかし、その理論的証明はできなかった。

第5章　結論

本研究では、量子情報処理で有効的に使われ、重要な資源である、量子エンタングルメントの変換について考察し、また変換補助の役割である触媒を使う、触媒効果についても考えた。2 人の観測者が遠く離れていても、共有している状態を変換できることは画期的なことであるといえる。

第 2 章では、EPR 状態 $|\psi\rangle$ を $|\phi\rangle$ に変換する場合、自分の qubit の測定や、電話などの LOCC のみではできないことが確認でき、補助 qubit が必要であることが証明できた。また、エンタングルメントを変換する際に深い関係があると言われている majorization の紹介をし、$x \prec y$ と $|\psi\rangle$ が LOCC で $|\phi\rangle$ に変換できることは同等であることの説明をした。

そして第 3 章では $x \prec y$ とエンタングルメント変換の関係を利用して、状態 $|\psi\rangle$ を LOCC で $|\phi\rangle$ に変換できないものが触媒によって、状態 $|\psi\rangle|c\rangle$ を LOCC で $|\phi\rangle|c\rangle$ に変換できる触媒作用について考察した。すると、変換できる最も簡単な系は 4 準位システムの状態 $|\psi\rangle$、$|\phi\rangle$ であること分かり、条件 A を導くことができた。また変換できる具体的な x、y を探しだし、状態 $|\psi\rangle|c\rangle$ を LOCC で $|\phi\rangle|c\rangle$ に変換できる触媒 $|c\rangle$ の存在を確認できたが、条件 A を満たしたからといって、変換できるとは限らないことが分かった。つまり、導かれた条件 A は必要条件であり、十分条件でないことが分かった。

次に自分自身を触媒として扱うことを考え、自己触媒というものが存在することを発見した。自己触媒は、触媒同様、4 準位システム以上の状態 $|\psi\rangle$、 $|\phi\rangle$ で、状態 $|\psi\rangle|\psi\rangle$ を LOCC で $|\phi\rangle|\phi\rangle$ に変換できる可能性があることが分かった。そして、数値的に探した結果、状態 $|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle$ に変換できる x 、 y の具体的な組合せを得ることができた。導いた条件も、条件 A となったが、同じく必要条件であって、十分条件でないことが分かった。

さらに、自己触媒の数を変えたとき、その関係はどうなっているか考察した。すると、数値的に検証した結果、 $xx \prec yy$ を満たす x、 y はすべて $xxx \prec yyy$ を満たし、LOCC で状態 $|\psi\rangle|\psi\rangle$ を $|\phi\rangle|\phi\rangle$ に変換できれば $|\psi\rangle|\psi\rangle|\psi\rangle$ も LOCC で $|\phi\rangle|\phi\rangle|\phi\rangle$ に変換できると思われることが分かった。しかし、このことは理論的に証明しようと試みたのだが、証明することはできなかった。よって、証明することが今後の課題である。

参考文献

1. Michael A.Nielsen, Isaac L.Chuang 共著／木村達也 訳
   量子コンピュータと量子通信 I － 量子力学とコンピュータ科学 －、オーム社 (2004)

2. Michael A.Nielsen, Isaac L.Chuang 共著／木村達也 訳
   量子コンピュータと量子通信 II － 量子コンピュータとアルゴリズム －、 オーム社 (2005)

3. Michael A.Nielsen, Isaac L.Chuang 共著／木村達也 訳
   量子コンピュータと量子通信 III － 量子通信 · 情報処理と誤り訂正 －、オーム社 (2005)

4. A. Einstein, B.Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *phys.Rev.*, 47 : 777-780, 1935.

5. C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A.Peres, and W. Wootters. Teleporting an unknown quantum state via dual classical and EPR channels. *phys.Rev.Lett.*, 70 : 1895-1899, 1993.

6. http://homepage2.nifty.com/einstein/contents/relativity/contents/relativity104.html

7. C. H. Bennett and S.J. Wiesner. Commnication via one- and two- particle operators on Einstein-Podolsky-Rosen states. *phys.Rev.Lett.*, 69(20) : 2881-2884, 1992.

8. D. Jonathan and M. B. Plenio. Entanglement-assisted local manipulation of pure states. *phys.Rev.Lett.*, 83 : 3566-3569, 1999.

謝辞

本論文を作成するにあたり、物理工学科 林 明久 先生には終始変わらぬ厚い御指導を賜わりましたことを感謝し御礼申し上げ、また、鈴木 敏男 先生、田嶋 直樹 先生にも卒論研究全般で厚い御指導、御意見を賜わり、深く感謝致します。

本論文および研究に対して御指導、御意見を頂いた多くの物理工学科の先生方の皆様にも心から御礼申し上げます。

最後に、有益なご討論を頂いた当研究室の院生、石田 陽介 先輩及び、学生に感謝し、謝辞の言葉とさせて頂きます。

付録 プログラムリスト

1. 条件 A を満たす、自分で決めた x 、y で $x\alpha \prec y\alpha$ を満たす触媒が存在するか調べるプログラム

```
#include<stdio.h>
void order(double x[], int n)
{
        int i, j;
        double c;
        for (i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(x[i]<x[j])
                        {
                            c=x[i]
                            x[i]=x[j];
                            x[j]=c;
                        }
                }
         }
}

int main()
{
        double x[4],y[4],a[2],xa[8],ya[8];
        int p,i,k, n;
        double j,q,m;
        int count;

        x[0]=0.50;
        x[1]=0.40;
        x[2]=0.05;
        x[3]=0.05;
        y[0]=0.70;
        y[1]=0.15;
        y[2]=0.15;
        y[3]=0.00;

        for(n=0;n<50;n++)
        {
                m=0.50+n*0.01;
                a[0]=m;
```

```
                a[1]=1-a[0];


                for(i=0;i<4;i++)
                {
                        for(k=0;k<2;k++)
                        {
                                xa[2*i+k]=x[i]*a[k];
                                ya[2*i+k]=y[i]*a[k];
                        }
                }


                order(xa, 8);
                order(ya, 8);
                for(p=0;p<8;p++)
                {
                       printf("%lf %lf\n", xa[p], ya[p]);
                }

                j=0;
                q=0;
                count=0;
                for(i=0;i<8;i++)
                {
                        j=j+xa[i];
                        q=q+ya[i];

                        printf("%lf %lf\n", j,q);

                        if(j<=q)
                        {
                                printf("OK\n");
                                count++;
                        }
                        if(j>q)
                        {
                                printf("error\n");
                        }
                }
                if(count==8)
                {
                      printf("%lf %f\n", a[0], a[1]);
                }

        }


}
```

2. 条件 A をみたす x 、 y を乱数で決定し、$xx \prec yy$ を満たす x 、 y の組み合わせが存在するか調べるプログラム

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void order(double x[], int n)
{
        int i, j;
        double c;
        for (i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(x[i]<x[j])
                        {
                          c=x[i];
                          x[i]=x[j];
                          x[j]=c;
                        }
                }
        }
}

int futoushiki(double x[], double y[], int n, int job)
{
    int i;
    double j, q, e;
    int  count;

    j=0;
    q=0;
    count=0;
    e=0.000001;

    for(i=0;i<n;i++)
    {
       j=j+x[i];
       q=q+y[i];

       if(job==0)
       {
       }
       if(job==1)
       {
         printf("%lf %lf\n", j,q);
       }

       if(j<=q+e)
       {
```

```
          count++;
        }
        if(j>q)
        {
        }
      }
     if(job==1)
     {
        printf("\n");
     }
     return count;
}

int main()
{
   double x[4],y[4],xx[16],yy[16];
   int p,i,k,j,m,r,w,z,v;
   double a,b,st,su,t,u;
   int count;

         for(i=1;i<=10000;i++)
         {
                 for(m=0;m<4;m++)
                 {
                         a=((double)rand()/(double)(RAND_MAX));
                         b=((double)rand()/(double)(RAND_MAX));
                         x[m]=a;
                         y[m]=b;
                 }

                 t=0;
                 u=0;
                 for(r=0;r<4;r++)
                 {
                         t=t+x[r];
                         u=u+y[r];
                 }
                 st=t;
                 su=u;
                 for(m=0;m<4;m++)
                 {
                         x[m]=x[m]/st;
                         y[m]=y[m]/su;
                 }
                 for(z=0;z<3;z++)
                 {
                   v=100*x[z];
                   w=100*y[z];
                   x[z]=v/100.0;
                   y[z]=w/100.0;
                 }
                 x[3]=1-x[0]-x[1]-x[2];
```

```
                        y[3]=1-y[0]-y[1]-y[2];
                        order(x,4);
                        order(y,4);

                        count=futoushiki(x, y, 4, 0);

                        if(count<4)
                        {
                                for(j=0;j<4;j++)
                                {
                                        for(k=0;k<4;k++)
                                        {
                                                xx[4*j+k]=x[j]*x[k];
                                                yy[4*j+k]=y[j]*y[k];
                                        }
                                }

                                order(xx, 16);
                                order(yy, 16);

                                count=futoushiki(xx, yy, 16, 0);
                                if(count==16)
                                {

                                        for(p=0;p<4;p++)
                                        {
                                                printf("%lf %lf\n", x[p], y[p]);

                                        }
                                        printf("\n");
                                        futoushiki(x, y, 4, 1);
                                        futoushiki(xx, yy, 16, 1);


                                }
                        }
                }
        }
```

3. 条件 A、$xx \prec yy$ を満たす x、y が $xxx \prec yyy$ を満たすのか調べるプログラム

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void order(double x[], int n)
{
        int i, j;
        double c;
        for (i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(x[i]<x[j])
                        {
                          c=x[i];
                          x[i]=x[j];
                          x[j]=c;
                        }
                }
        }
}

int futoushiki(double x[], double y[], int n, int job)
{
    int i;
    double j, q, e;
    int  count;

    j=0;
    q=0;
    count=0;
    e=0.000001;

    for(i=0;i<n;i++)
    {
       j=j+x[i];
       q=q+y[i];

       if(job==0)
       {
       }
       if(job==1)
       {
         printf("%lf %lf\n", j,q);
       }

       if(j<=q+e)
       {
         count++;
```

```
        }
        if(j>q)
        {
        }
      }
     if(job==1)
     {
        printf("\n");
     }
     return count;
}

int main()
{
   double x[4],y[4],xx[16],yy[16],xxx[64],yyy[64];
   int p,i,k,j,m,r,w,z,v;
   double a,b,st,su,t,u;
   int count;

         for(i=1;i<=100000;i++)
         {
                 for(m=0;m<4;m++)
                 {
                         a=((double)rand()/(double)(RAND_MAX));
                         b=((double)rand()/(double)(RAND_MAX));
                         x[m]=a;
                         y[m]=b;
                 }

                 t=0;
                 u=0;
                 for(r=0;r<4;r++)
                 {
                         t=t+x[r];
                         u=u+y[r];
                 }
                 st=t;
                 su=u;
                 for(m=0;m<4;m++)
                 {
                         x[m]=x[m]/st;
                         y[m]=y[m]/su;
                 }
                 for(z=0;z<3;z++)
                 {
                   v=100*x[z];
                   w=100*y[z];
                   x[z]=v/100.0;
                   y[z]=w/100.0;
                 }
                 x[3]=1-x[0]-x[1]-x[2];
                 y[3]=1-y[0]-y[1]-y[2];
```

```
order(x,4);
order(y,4);

count=futoushiki(x, y, 4, 0);

if(count<4)
{
        for(j=0;j<4;j++)
        {
                for(k=0;k<4;k++)
                {
                        xx[4*j+k]=x[j]*x[k];
                        yy[4*j+k]=y[j]*y[k];
                }
        }

        order(xx, 16);
        order(yy, 16);

        count=futoushiki(xx, yy, 16, 0);
        if(count==16)
        {
           printf("2-OK\n");

           for(m=0;m<4;m++)
           {
              for(j=0;j<4;j++)
              {
                for(k=0;k<4;k++)
                {
                        xxx[16*m+4*j+k]=x[m]*x[j]*x[k];
                        yyy[16*m+4*j+k]=y[m]*y[j]*y[k];
                }
              }
           }
           order(xxx, 64);
           order(yyy, 64);

           count=futoushiki(xxx, yyy, 64, 0);
           if(count==64)
           {
                printf("3-OK\n");
                for(p=0;p<4;p++)
                {
                        printf("%lf %lf\n", x[p], y[p]);

                }
                printf("\n");
                  futoushiki(x, y, 4, 1);
                  futoushiki(xx, yy, 16, 1);
                  futoushiki(xxx, yyy, 64, 1);
           }
```

```
                        if(count<64)
                        {
                           printf("3-NG\n");
                        }
                    }
                }
            }
        }
```

4. 条件 A、$xx \prec yy$ を満たす x、y のうち、 $xxx \prec yyy$ を満たす数と、満たさない数を調べるプログラム

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void order(double x[], int n)
{
        int i, j;
        double c;
        for (i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(x[i]<x[j])
                        {
                          c=x[i];
                          x[i]=x[j];
                          x[j]=c;
                        }
                }
        }
}

int futoushiki(double x[], double y[], int n, int job)
{
    int i;
    double j, q, e;
    int  count;

    j=0;
    q=0;
    count=0;
    e=0.000001;

    for(i=0;i<n;i++)
    {
       j=j+x[i];
       q=q+y[i];

       if(job==0)
       {
       }
       if(job==1)
       {
         printf("%lf %lf\n", j,q);
       }

       if(j<=q+e)
       {
```

```
          count++;
        }
        if(j>q)
        {
        }
      }
     if(job==1)
     {
        printf("\n");
     }
     return count;
}

int main()
{
   double x[4],y[4],xx[16],yy[16],xxx[64],yyy[64];
   int p,i,k,j,m,r,w,z,v;
   double a,b,st,su,t,u;
   int count, number1, number2, number3;

   number1=0;
   number2=0;
   number3=0;

        for(i=1;i<=100000000;i++)
        {
                for(m=0;m<4;m++)
                {
                        a=((double)rand()/(double)(RAND_MAX));
                        b=((double)rand()/(double)(RAND_MAX));
                        x[m]=a;
                        y[m]=b;
                }

                t=0;
                u=0;
                for(r=0;r<4;r++)
                {
                        t=t+x[r];
                        u=u+y[r];
                }
                st=t;
                su=u;
                for(m=0;m<4;m++)
                {
                        x[m]=x[m]/st;
                        y[m]=y[m]/su;
                }
                for(z=0;z<3;z++)
                {
                  v=100*x[z];
                  w=100*y[z];
```

```
  x[z]=v/100.0;
  y[z]=w/100.0;
}
x[3]=1-x[0]-x[1]-x[2];
y[3]=1-y[0]-y[1]-y[2];
order(x,4);
order(y,4);

count=futoushiki(x, y, 4, 0);

if(count<4)
{
        for(j=0;j<4;j++)
        {
                for(k=0;k<4;k++)
                {
                        xx[4*j+k]=x[j]*x[k];
                        yy[4*j+k]=y[j]*y[k];
                }
        }

        order(xx, 16);
        order(yy, 16);

        count=futoushiki(xx, yy, 16, 0);
        if(count==16)
        {
           number1++;

           for(m=0;m<4;m++)
           {
              for(j=0;j<4;j++)
              {
                for(k=0;k<4;k++)
                {
                        xxx[16*m+4*j+k]=x[m]*x[j]*x[k];
                        yyy[16*m+4*j+k]=y[m]*y[j]*y[k];
                }
              }
           }
           order(xxx, 64);
           order(yyy, 64);

           count=futoushiki(xxx, yyy, 64, 0);

           if(count==64)
           {
              number2++;
           }


           if(count<64)
```

```
                        {
                            number3++;
                        }
                    }

                }
        }
printf("%d %d %d\n", number1, number2, number3);

}
```

5. 条件 A、$xxx \prec yyy$ を満たす x、y のうち、 $xx \prec yy$ を満たす数と、満たさない数を調べるプログラム

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void order(double x[], int n)
{
        int i, j;
        double c;
        for (i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(x[i]<x[j])
                        {
                          c=x[i];
                          x[i]=x[j];
                          x[j]=c;
                        }
                }
        }
}

int futoushiki(double x[], double y[], int n, int job)
{
    int i;
    double j, q, e;
    int  count;

    j=0;
    q=0;
    count=0;
    e=0.000001;

    for(i=0;i<n;i++)
    {
       j=j+x[i];
       q=q+y[i];

       if(job==0)
       {
       }
       if(job==1)
       {
         printf("%lf %lf\n", j,q);
       }

       if(j<=q+e)
       {
```

```
          count++;
        }
        if(j>q)
        {
        }
      }
     if(job==1)
     {
        printf("\n");
     }
     return count;
}

int main()
{
   double x[4],y[4],xx[16],yy[16],xxx[64],yyy[64];
   int p,i,k,j,m,r,w,z,v;
   double a,b,st,su,t,u;
   int count, number1, number2, number3;

   number1=0;
   number2=0;
   number3=0;

        for(i=1;i<=1000;i++)
        {
                for(m=0;m<4;m++)
                {
                        a=((double)rand()/(double)(RAND_MAX));
                        b=((double)rand()/(double)(RAND_MAX));
                        x[m]=a;
                        y[m]=b;
                }

                t=0;
                u=0;
                for(r=0;r<4;r++)
                {
                        t=t+x[r];
                        u=u+y[r];
                }
                st=t;
                su=u;
                for(m=0;m<4;m++)
                {
                        x[m]=x[m]/st;
                        y[m]=y[m]/su;
                }
                for(z=0;z<3;z++)
                {
                  v=100*x[z];
                  w=100*y[z];
```

```
  x[z]=v/100.0;
  y[z]=w/100.0;
}
x[3]=1-x[0]-x[1]-x[2];
y[3]=1-y[0]-y[1]-y[2];
order(x,4);
order(y,4);

count=futoushiki(x, y, 4, 0);

if(count<4)
{
        for(m=0;m<4;m++)
        {
                for(j=0;j<4;j++)
                {
                     for(k=0;k<4;k++)
                     {
                        xxx[16*m+4*j+k]=x[m]*x[j]*x[k];
                        yyy[16*m+4*j+k]=y[m]*y[j]*y[k];
                     }
                }
        }

        order(xxx, 64);
        order(yyy, 64);

        count=futoushiki(xxx, yyy, 64, 0);
        if(count==64)
        {
           number1++;

           for(j=0;j<4;j++)
           {
              for(k=0;k<4;k++)
              {
                 xx[4*j+k]=x[j]*x[k];
                 yy[4*j+k]=y[j]*y[k];
              }
           }
           order(xx, 16);
           order(yy, 16);

           count=futoushiki(xx, yy, 16, 0);

           if(count==16)
           {
              number2++;
           }


           if(count<16)
```

```
                                {
                                  number3++;
                                }
                            }

                    }
            }
printf("%d %d %d\n", number1, number2, number3);

}
```

6. 条件 A、$xxx \prec yyy$ を満たす x、y のうち、 $xxxx \prec yyyy$ を満たす数と、満たさない数を調べるプログラム

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void order(double x[], int n)
{
        int i, j;
        double c;
        for (i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(x[i]<x[j])
                        {
                          c=x[i];
                          x[i]=x[j];
                          x[j]=c;
                        }
                }
        }
}

int futoushiki(double x[], double y[], int n, int job)
{
    int i;
    double j, q, e;
    int  count;

    j=0;
    q=0;
    count=0;
    e=0.000001;

    for(i=0;i<n;i++)
    {
       j=j+x[i];
       q=q+y[i];

       if(job==0)
       {
       }
       if(job==1)
       {
         printf("%lf %lf\n", j,q);
       }

       if(j<=q+e)
       {
```

```
          count++;
        }
        if(j>q)
        {
        }
      }
     if(job==1)
     {
        printf("\n");
     }
     return count;
}

int main()
{
   double x[4],y[4],xxx[64],yyy[64],xxxx[256],yyyy[256];
   int p,i,k,j,m,r,w,z,v;
   double a,b,st,su,t,u;
   int count, number1, number2, number3;

   number1=0;
   number2=0;
   number3=0;

        for(i=1;i<=100000000;i++)
        {
                for(m=0;m<4;m++)
                {
                        a=((double)rand()/(double)(RAND_MAX));
                        b=((double)rand()/(double)(RAND_MAX));
                        x[m]=a;
                        y[m]=b;
                }

                t=0;
                u=0;
                for(r=0;r<4;r++)
                {
                        t=t+x[r];
                        u=u+y[r];
                }
                st=t;
                su=u;
                for(m=0;m<4;m++)
                {
                        x[m]=x[m]/st;
                        y[m]=y[m]/su;
                }
                for(z=0;z<3;z++)
                {
                  v=100*x[z];
                  w=100*y[z];
```

```
  x[z]=v/100.0;
  y[z]=w/100.0;
}
x[3]=1-x[0]-x[1]-x[2];
y[3]=1-y[0]-y[1]-y[2];
order(x,4);
order(y,4);

count=futoushiki(x, y, 4, 0);

if(count<4)
{
   for(m=0;m<4;m++)
   {
      for(j=0;j<4;j++)
      {
          for(k=0;k<4;k++)
          {
              xxx[16*m+4*j+k]=x[m]*x[j]*x[k];
              yyy[16*m+4*j+k]=y[m]*y[j]*y[k];
          }
      }
   }
    order(xxx, 64);
    order(yyy, 64);

    count=futoushiki(xxx, yyy, 64, 0);

    if(count==64)
    {
      number1++;

      for(r=0;r<4;r++)
      {
        for(m=0;m<4;m++)
        {
          for(j=0;j<4;j++)
          {
            for(k=0;k<4;k++)
            {
                xxxx[64*r+16*m+4*j+k]=x[r]*x[m]*x[j]*x[k];
                yyyy[64*r+16*m+4*j+k]=y[r]*y[m]*y[j]*y[k];
            }
          }
        }
}
      order(xxxx, 256);
      order(yyyy, 256);

      count=futoushiki(xxxx, yyyy, 256, 0);

      if(count==256)
```

```
                        {
                                number2++;

                                for(p=0;p<4;p++)
                                  {
                                    //printf("%lf %lf\n", x[p], y[p]);

                                  }


                         }


                         if(count<256)
                         {
                             number3++;
                         }
                       }

                }
        }
  printf("%d %d\n", number2, number3);

}
```

7. 条件 A、$xxxx \prec yyyy$ を満たす x、y のうち、 $xxx \prec yyy$ を満たす数と、満たさない数を調べるプログラム

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void order(double x[], int n)
{
        int i, j;
        double c;
        for (i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(x[i]<x[j])
                        {
                          c=x[i];
                          x[i]=x[j];
                          x[j]=c;
                        }
                }
        }
}

int futoushiki(double x[], double y[], int n, int job)
{
    int i;
    double j, q, e;
    int  count;

    j=0;
    q=0;
    count=0;
    e=0.000001;

    for(i=0;i<n;i++)
    {
       j=j+x[i];
       q=q+y[i];

       if(job==0)
       {
       }
       if(job==1)
       {
         printf("%lf %lf\n", j,q);
       }

       if(j<=q+e)
       {
```

```
          count++;
        }
        if(j>q)
        {
        }
      }
     if(job==1)
     {
        printf("\n");
     }
     return count;
}

int main()
{
   double x[4],y[4],xxx[64],yyy[64],xxxx[256],yyyy[256];
   int p,i,k,j,m,r,w,z,v;
   double a,b,st,su,t,u;
   int count, number1, number2, number3;

   number1=0;
   number2=0;
   number3=0;

        for(i=1;i<=100000000;i++)
        {
                for(m=0;m<4;m++)
                {
                        a=((double)rand()/(double)(RAND_MAX));
                        b=((double)rand()/(double)(RAND_MAX));
                        x[m]=a;
                        y[m]=b;
                }

                t=0;
                u=0;
                for(r=0;r<4;r++)
                {
                        t=t+x[r];
                        u=u+y[r];
                }
                st=t;
                su=u;
                for(m=0;m<4;m++)
                {
                        x[m]=x[m]/st;
                        y[m]=y[m]/su;
                }
                for(z=0;z<3;z++)
                {
                  v=100*x[z];
                  w=100*y[z];
```

```
  x[z]=v/100.0;
  y[z]=w/100.0;
}
x[3]=1-x[0]-x[1]-x[2];
y[3]=1-y[0]-y[1]-y[2];
order(x,4);
order(y,4);

count=futoushiki(x, y, 4, 0);

if(count<4)
{
   for(r=0;r<4;r++)
   {
        for(m=0;m<4;m++)
        {
          for(j=0;j<4;j++)
          {
            for(k=0;k<4;k++)
            {
                xxxx[64*r+16*m+4*j+k]=x[r]*x[m]*x[j]*x[k];
                yyyy[64*r+16*m+4*j+k]=y[r]*y[m]*y[j]*y[k];
            }
          }
        }
    }

    order(xxxx, 256);
    order(yyyy, 256);

    count=futoushiki(xxxx, yyyy, 256, 0);

    if(count==256)
    {
        number1++;

        for(m=0;m<4;m++)
        {
          for(j=0;j<4;j++)
          {
             for(k=0;k<4;k++)
             {
                xxx[16*m+4*j+k]=x[m]*x[j]*x[k];
                yyy[16*m+4*j+k]=y[m]*y[j]*y[k];
             }
          }
        }
        order(xxx, 64);
        order(yyy, 64);

        count=futoushiki(xxx, yyy, 64, 0);
```

```
                        if(count==64)
                        {
                           number2++;

                           for(p=0;p<4;p++)
                           {
                             //printf("%lf %lf\n", x[p], y[p]);

                           }
                         }

                         if(count<64)
                         {
                            number3++;
                         }
                       }

               }
        }
  printf("%d %d %d\n", number1, number2, number3);

}
```

8. 条件 A、$xxxx \prec yyyy$ を満たす x、y のうち、 $xx \prec yy$ を満たす数と、満たさない数を調べるプログラム

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void order(double x[], int n)
{
        int i, j;
        double c;
        for (i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(x[i]<x[j])
                        {
                          c=x[i];
                          x[i]=x[j];
                          x[j]=c;
                        }
                }
        }
}

int futoushiki(double x[], double y[], int n, int job)
{
    int i;
    double j, q, e;
    int  count;

    j=0;
    q=0;
    count=0;
    e=0.000001;

    for(i=0;i<n;i++)
    {
       j=j+x[i];
       q=q+y[i];

       if(job==0)
       {
       }
       if(job==1)
       {
         printf("%lf %lf\n", j,q);
       }

       if(j<=q+e)
       {
```

```
          count++;
        }
        if(j>q)
        {
        }
      }
     if(job==1)
     {
        printf("\n");
     }
     return count;
}

int main()
{
   double x[4],y[4],xx[16],yy[16],xxx[64],yyy[64],xxxx[256],yyyy[256];
   int p,i,k,j,m,r,w,z,v;
   double a,b,st,su,t,u;
   int count, number1, number2, number3;

   number1=0;
   number2=0;
   number3=0;

        for(i=1;i<=100000000;i++)
        {
                for(m=0;m<4;m++)
                {
                        a=((double)rand()/(double)(RAND_MAX));
                        b=((double)rand()/(double)(RAND_MAX));
                        x[m]=a;
                        y[m]=b;
                }

                t=0;
                u=0;
                for(r=0;r<4;r++)
                {
                        t=t+x[r];
                        u=u+y[r];
                }
                st=t;
                su=u;
                for(m=0;m<4;m++)
                {
                        x[m]=x[m]/st;
                        y[m]=y[m]/su;
                }
                for(z=0;z<3;z++)
                {
                  v=100*x[z];
                  w=100*y[z];
```

```
  x[z]=v/100.0;
  y[z]=w/100.0;
}
x[3]=1-x[0]-x[1]-x[2];
y[3]=1-y[0]-y[1]-y[2];
order(x,4);
order(y,4);

count=futoushiki(x, y, 4, 0);

if(count<4)
{
   for(r=0;r<4;r++)
   {
        for(m=0;m<4;m++)
        {
          for(j=0;j<4;j++)
          {
            for(k=0;k<4;k++)
            {
                xxxx[64*r+16*m+4*j+k]=x[r]*x[m]*x[j]*x[k];
                yyyy[64*r+16*m+4*j+k]=y[r]*y[m]*y[j]*y[k];
            }
          }
        }
      }
      order(xxxx, 256);
      order(yyyy, 256);

      count=futoushiki(xxxx, yyyy, 256, 0);

      if(count==256)
      {
          //number1++;

          for(j=0;j<4;j++)
          {
             for(k=0;k<4;k++)
             {
                xx[4*j+k]=x[j]*x[k];
                yy[4*j+k]=y[j]*y[k];
             }
          }

        order(xx, 16);
        order(yy, 16);

        count=futoushiki(xx, yy, 16, 0);

        if(count==16)
        {
           number2++;
```

```
                            for(p=0;p<4;p++)
                            {
                              //printf("%lf %lf\n", x[p], y[p]);

                            }
                          }

                          if(count<16)
                          {
                             number3++;
                          }
                        }

                }
        }
  printf("%d %d\n", number2, number3);

}
```