信州大学
大学院総合理工学研究科

修士論文

パケット間隔で情報を送る通信路に対する Sum-Product 復号法の検証

指導教員　西新 幹彦 准教授

専攻　工学専攻
分野　電子情報システム工学分野
学籍番号　21W2033H
氏名　佐藤 奏杜

2023 年 2 月 17 日

目次

1　はじめに

情報伝達方法の一つとしてパケット通信がある．パケット通信とはコンピュータ通信において，データを小さなまとまりに分割してその一つ一つを送受信するという通信方法である．分割されたデータはパケットと呼ばれ，多くの場合はメッセージを符号化し，その符号語をパケットに収める．パケット通信を利用すると通信の送受信間で途中の回線が占有されることがなくなり，通信回線を効率よく使用することが出来る．また，通信経路の選択をすることが出来るため，ネットワーク上の一部に障害が生じてしまった場合でも，他の経路を利用し通信を行うことが出来るという利点もある．

通常のパケット通信ではパケットの中に伝えるべき情報が収められている．それに加えて，Anantharam and Verdú [1] は，パケット間隔も情報を運ぶことができると指摘している．つまりパケットの送信間隔の長さに意味を与えることで，受信者は到着するパケットの間隔で情報を得ることができるのである．パケット通信を行う際はネットワーク内では様々な処理時間の変化や転送経路の移動などが発生してしまい，パケットの時間間隔が送信時と受信時とで異なってしまうため復号器での誤り訂正が必要とされる．そして，誤り訂正された間隔から元となる情報を復号することで，正しく情報を伝達することが可能とされているのである．

大きさのない空のパケットの送信間隔に情報を与えた時の単一サーバ待ち行列システムの通信路容量は，文献 [1] で既に求められている．具体的な符号化法に関して，Coleman and Kiyavash [2] はパケット間隔を用いた 4 元の離散時間通信路に対して，記憶のある通信路モデルを復号器に内蔵することで，通信路容量に近い符号化レートを達成する実用性のある符号器，復号器を設計しており，低密度パリティ検査符号（Low Density Parity Check Code，以下では LDPC 符号と略記する）と Sum-Product 復号法を用いる方法を提案している．しかし，追実験を行うために必要な詳細の記述は省略されている．これに対し文献 [3] では，省略されていた手法の詳細を合理的に特定している．そのもとで筆者は，文献 [4] にて符号語長が短い場合における復号性能の評価を行った．

本研究では，文献 [3] で示された手法をもとに符号語長が長い場合における復号誤り確率の測定を行い，復号性能を検証した．その際，比較対象として，通信路を無記憶とみなした時の Sum-Product 復号法の誤り確率を測定することによって，通信路の記憶を考慮することによる復号性能の変化を確認した．その結果，符号語長が長い場合においては丸め誤差が発生し，通信路を無記憶とみなした時の Sum-Product 復号法が性能が高い様子が確認された．

2　通信路モデルと符号化定理

本章では，本研究で対象とする通信路を連続時間の数理モデルを用いて表現し，概要を述べる．このモデルは，文献 [5] で扱われているモデルと同じである．

2.1　通信路モデル

インターネットのようなネットワークでは複数の送信者がそれぞれの受信者にパケットを不規則に送信する．各パケットは複数のサーバを通り最終的な目的地に到着する．パケットが送信者から出発してから目的地に到着するまでの時間は，他のユーザが送信しているパケットの数，すなわちネットワーク上での混み具合などに依存する．ここで，一組の送受信者に着目して考えるとネットワーク全体は複雑な待ち行列システムとして表される．本研究では簡単化のため複雑な待ち行列システムをシンプルな単一サーバ FIFO 待ち行列システムとみなす．そのため，サーバが稼働中に通信路に到着したパケットの順序を崩さないためにサーバの前に十分大きなキューがあると仮定する．また，サーバはパラメータ μ の指数分布に従う処理時間でパケットを処理する．このような待ち行列システムによる通信路モデルを図 1 に示す．

通常，パケット通信はパケットの中に情報を入れて通信を行う．しかし，本研究ではパケット間隔で情報を送ることに重点を置くため，パケットには情報を入れずパケットの送信間隔のみで通信を行う．すなわち，符号器では，送信されるメッセージを時間間隔の列に変換し通信路へと送信する．そして，復号器では到着したパケットの時間間隔からメッセージを復号する．

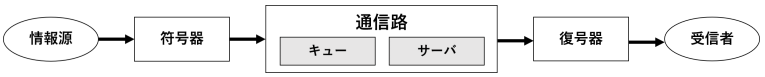

図 1　通信路モデル

2.2 通信路の記憶

通信路が無記憶であるとは，通信路の出力がそれ以前の入力・出力とは独立であることをいう．そうでない通信路は記憶があると言われる．これを踏まえ，本研究の通信路には記憶があることを以下で示す．

まず，パケット間隔を用いて情報を送る通信路における時刻と時間間隔の関係を図 2 に示す．横軸は時間経過を表している．時間軸に向かっている下からの矢印は通信路に到着したパケットの時刻を示し，時間軸から上に出ている矢印は通信路で処理されパケットが出力される時刻を示している．$x_i, d_i (i = 1, \ldots, 5)$ をパケット送信の時間間隔とし，$\hat{x}_i$ と $\hat{d}_i$ をそれぞれパケットの到着時刻，出発時刻とする．パケットが時刻 $\hat{x}_i$ で通信路に入力され，サーバで処理されてから時刻 $\hat{d}_i$ に通信路から出力される．

ここで，時刻 $\hat{d}_2$ に着目する．時刻 $\hat{d}_2$ を時間間隔で表すと d_1+d_2 となる．ここで，時間間隔 d_1 は，パケットが時刻 $\hat{d}_1$ に通信路から出力されたことを示す．時刻 $\hat{d}_1$ で通信路から出力されたパケットは，時刻 $\hat{x}_1$ で通信路に入力されたパケットがサーバで処理されてから出力されたものである．従って，時刻 $\hat{d}_2$ で通信路から出力されるパケットは，時刻 $\hat{x}_1$ で入力されたパケット及び時刻 $\hat{d}_1$ で通信路が出力されたパケットと独立でないことが分かる．これが通信路に記憶があるということである．

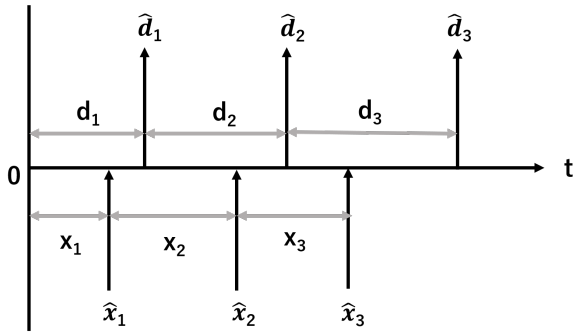

図 2　時刻と時間間隔の関係

2.3 通信路符号化定理

形式的には，符号語は非負実数の並びである．本研究では文献 [1] と同じく，パケットの間隔を用いた符号を次のように定義する．

定義 1 符号語は非負実数ベクトル $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ として表される．符号器が符号語 $\boldsymbol{x}$ を送信するとき，時刻 0 で空のキューに対して時刻 $\sum_{i=1}^{k} x_i$ に k 番目 $(k \geq 1)$ の到着が起こる．符号器は M 個の符号語を持っているとし，それらは等確率で選ばれて送信されるとする．復号器にとっての受信語とは，サーバからの出発間隔である．復号器の復号誤り確率を ϵ とおく．最後の出発が起こる時刻の平均を T とおく．このような符号を (n, M, T, ϵ) 符号という．この符号の符号化レートを $(\log\ M)/T$ と定める．符号化レートとは単位時間あたりに符号器が送信する情報量を表している．なお本論文では log は自然対数を表す．

通信において，符号化レートが大きいことが求められると同時に復号器で正しく復号されることも重要とされている．したがって，与えられた符号化レートに対して正しく復号できるような復号器を設計しなければならない．この性質ついて文献 [1] に基づいて文献 [4] で次のように定義されている．

定義 2 符号化レート R が**達成可能**とは

$$\liminf_{n\to\infty} \frac{\log M_n}{T_n} \geq R \tag{1}$$

$$\lim_{n\to\infty} \frac{n}{T_n} > 0 \tag{2}$$

$$\lim_{n\to\infty} \epsilon_n = 0 \tag{3}$$

となるような符号の列 $\{(n, M_n, T_n, \epsilon_n)\}_{i=1}^{\infty}$ が存在することである．通信路容量を

$$C \triangleq \sup\{R|\ \text{符号化レート } R \text{ は達成可能 }\}$$

と定める．

本研究では，出力レートを変えて復号性能を比較検証するため，出力レート λ を考慮した場合における定義を以下に示す．

定義 3 符号化レート R が出力レート λ で **達成可能** とは

$$\liminf_{n\to\infty} \lambda\frac{\log M_n}{n} \geq R \tag{4}$$

$$\lim_{n\to\infty} \frac{n}{T_n} \geq \lambda \tag{5}$$

$$\lim_{n\to\infty} \epsilon_n = 0 \tag{6}$$

となるような符号の列 $\{(n, M_n, T_n, \epsilon_n)\}_{i=1}^{\infty}$ が存在することである．$0 < \epsilon < 1$ となる任意の ϵ において R が出力レート λ で **達成可能** という．出力レート λ における通信路容量と呼び，$C(\lambda)$ と表す．すなわち

$$C(\lambda) \triangleq \sup\{R| \text{ 符号化レート } R \text{ は出力レート}\lambda\text{で達成可能 }\}$$

と定める．

上記の定義のもとで次のような 2 つの定理が成り立つことが文献 [6] に示されている．

定理 1

$$C(\lambda) = \lambda \log \frac{\mu}{\lambda},\ \lambda \leq \mu \tag{7}$$

定理 2

$$C = \max_{\lambda \leq \mu} C(\lambda) = e^{-1}\mu \tag{8}$$

3 Sum-Prodcut 復号法の基礎

記憶のある通信路において Sum-Product 復号法をそのまま適用することは難しい．これを受け，本章では無記憶通信路における基本的な Sum-Product 復号法を示し，4 章で記憶を考慮した場合に発生する問題点について明らかにする．

3.1 条件付き確率

定常無記憶通信路における基本的な Sum-Product 復号の計算手順について，具体的な符号を用いて説明していく．

符号器から通信路へ送信されたシンボルを符号シンボル x，通信路から送信され復号器で受信されたシンボルを受信シンボル y と置く．このとき，符号器から通信路へ x が送信された際に，通信路から送信され復号器で y を受信する条件付き確率 $W(y|x)$ は，

$$W(y|x) = \Pr\{Y = y|X = x\} \tag{9}$$

と表すことが出来る．

この通信路に対して長さ 5 の符号語を送信することを考える．ここで，$\boldsymbol{x}$= $(x_1, x_2, \ldots, x_5)$, $\boldsymbol{y}$= $(y_1, y_2, \ldots, y_5)$ とすると，符号語 $\boldsymbol{x}$ を通信路に入力した際に受信語 $\boldsymbol{y}$ が出力される確率は

$$W^5(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1,2,3,4,5} W(y_i|x_i) \tag{10}$$

と表される．

通信路に入力される符号語は一定の制約を満たしたものの中から選ばれる．ここでは例として，符号語長が 5 で

$$x_1 + x_2 = 0 \tag{11}$$
$$x_2 + x_3 + x_4 = 0 \tag{12}$$
$$x_4 + x_5 = 0 \tag{13}$$

の制約を満たすものを符号語とする．この制約は行列を用いて

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = 0 \tag{14}$$

と表すことができる．このとき式 (14) 左辺の 3×5 行列をパリティ検査行列という．

一般に通信路の入力と出力は異なるため，復号器では受信語を観測したもとで符号語を推定する必要がある．受信語 $\boldsymbol{y}$ を受信したもとで条件付き確率 $\Pr\{X^5 = \boldsymbol{x}|Y^5 = \boldsymbol{y}\}$ が最大となるような符号語 $\boldsymbol{x}$ を選択する復号方法を最大事後確率復号法 (MAP 復号法) という．この条件付き確率は,

$$\Pr\{X^5 = \boldsymbol{x}|Y^5 = \boldsymbol{y}\} = \frac{\Pr\{X^5 = \boldsymbol{x}, Y^5 = \boldsymbol{y}\}}{\Pr\{Y^5 = \boldsymbol{y}\}} \tag{15}$$
$$= \frac{\Pr\{X^5 = \boldsymbol{x}\} \cdot \Pr\{Y^5 = \boldsymbol{y}|X^5 = \boldsymbol{x}\}}{\Pr\{Y^5 = \boldsymbol{y}\}} \tag{16}$$

と表すことができる．すると，$\boldsymbol{y}$ は観測された値であるから $\Pr\{Y^5 = \boldsymbol{y}\}$ は定数となる．また，符号語は等確率で送信されるとすれば，$\Pr\{X^5 = \boldsymbol{x}\}$ も定数となる．

MAP 復号を実際に行おうとするとすべての符号語に対して条件付き確率を計算しなければならない．この計算量を削減するために，シンボル単位で MAP 復号する方法が知られている．この方法はシンボル MAP 復号と呼ばれている．例えば受信語 $\boldsymbol{y}$ に対して送信シンボル x_1 を推定する際には条件付き確率 $\Pr\{X_1 = x_1|Y^5 = \boldsymbol{y}\}$ を最大化する x_1 を求めればよい．

ここで，$\boldsymbol{C}$ を符号語の集合とすると条件付き確率は，

$$\Pr\{X_1 = x_1 | Y^5 = \boldsymbol{y}\} \tag{17}$$

$$= \sum_{x_2,x_3,x_4,x_5} \Pr\{X^5 = x_1 \dots x_5 | Y^5 = \boldsymbol{y}\} \tag{18}$$

$$= \sum_{x_2,x_3,x_4,x_5} \mathbb{1}\{\boldsymbol{x} \in \boldsymbol{C}\} \Pr\{X^5 = x_1 \dots x_5 | Y^5 = \boldsymbol{y}\} \tag{19}$$

$$= \sum_{x_2,x_3,x_4,x_5} \mathbb{1}\{\boldsymbol{x} \in \boldsymbol{C}\} \frac{\Pr\{X^5 = \boldsymbol{x}\} \Pr\{Y^5 = \boldsymbol{y} | X^5 = \boldsymbol{x}\}}{\Pr\{Y^5 = \boldsymbol{y}\}} \tag{20}$$

$$= \frac{1}{|\boldsymbol{C}|} \frac{1}{\Pr\{Y^5 = \boldsymbol{y}\}} \sum_{x_2,x_3,x_4,x_5} f_1(x_1,x_2) f_2(x_2,x_3,x_4) f_3(x_4,x_5) \prod_{i=1}^{5} W(y_i|x_i) \tag{21}$$

となる．ただし，$f_1(x_1,x_2) \triangleq \mathbb{1}\{x_1 + x_2 = 0\}$，$f_2(x_2,x_3,x_4) \triangleq \mathbb{1}\{x_2 + x_3 + x_4 = 0\}$，$f_3(x_4,x_5) \triangleq \mathbb{1}\{x_4 + x_5 = 0\}$ とおいた．また，$|\boldsymbol{C}|$ と $\Pr\{Y^5 = \boldsymbol{y}\}$ は定数なので最大化問題を解く際には関係がない．したがって，

$$\sum_{x_2,x_3,x_4,x_5} f_1(x_1,x_2) f_2(x_2,x_3,x_4) f_3(x_4,x_5) \prod_{i=1}^{5} W(y_i|x_i) \tag{22}$$

を最大化するような x_1 を求めればよい．その意味で，以降ではこの式のことも条件付き確率と呼ぶことにする．この式はさらに，

$$\begin{aligned}
&\sum_{x_2,x_3,x_4,x_5} f_1(x_1,x_2) f_2(x_2,x_3,x_4) f_3(x_4,x_5) \prod_{i=1}^{5} W(y_i|x_i) \\
&= W(y_1|x_1) \\
&\times \sum_{x_2} f_1(x_1,x_2) W(y_2|x_2) \\
&\times \sum_{x_3,x_4} f_2(x_2,x_3,x_4) W(y_3|x_3) W(y_4|x_4) \\
&\times \sum_{x_5} f_3(x_4,x_5) W(y_5|x_5)
\end{aligned} \tag{23}$$

と書き換えることができる．このとき，各総和の変数と W 内の変数は一致している．

3.2 ファクターグラフ

式 (22) について計算の流れを視覚的に捉えられるようにすることを目的に，積計算を行う部分を $M_{x_k \to f_i}(x_k)$，和計算を行う部分を $M_{f_i \to x_k}(x_k)$ と置く．具体的には，

$$
\begin{aligned}
M_{x_1}(x_1) &\triangleq W(y_1|x_1)M_{f_1\to x_1}(x_1)\\
M_{f_1\to x_1}(x_1) &\triangleq \sum_{x_2} f_1(x_1,x_2)M_{x_2\to f_1}(x_2)\\
M_{x_2\to f_1}(x_2) &\triangleq W(y_2|x_2)M_{f_2\to x_2}(x_2)\\
M_{f_2\to x_2}(x_2) &\triangleq \sum_{x_3,x_4} f_2(x_2,x_3,x_4)M_{x_3\to f_2}(x_3)M_{x_4\to f_2}(x_4)\\
M_{x_3\to f_2}(x_3) &\triangleq W(y_3|x_3)\\
M_{x_4\to f_2}(x_4) &\triangleq W(y_4|x_4)M_{f_3\to x_4}(x_4)\\
M_{f_3\to x_4}(x_4) &\triangleq \sum_{x_5} f_3(x_4,x_5)M_{x_5\to f_3}(x_5)\\
M_{x_5\to f_3}(x_5) &\triangleq W(y_5|x_5)
\end{aligned}
$$

とおく．これにより，式 (22) を図 3 のようなツリー構造で表すことが可能となる．これは，ツリーの葉から根に向かって計算結果を次々に伝えることにより式 (22) が求まる構造となっている．ツリー構造で表現することにより，式 (22) を視覚的に捉えることが可能となる．$x_2,\ldots,x_5$ についても同様の操作を行うと，それぞれ図 4〜7 のようなツリー構造を描くことができる．

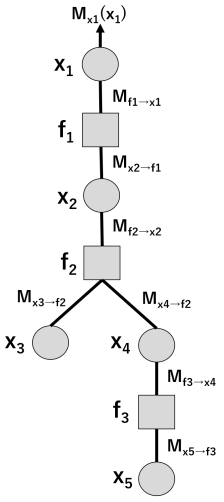

図 3　x_1 を根とするツリー

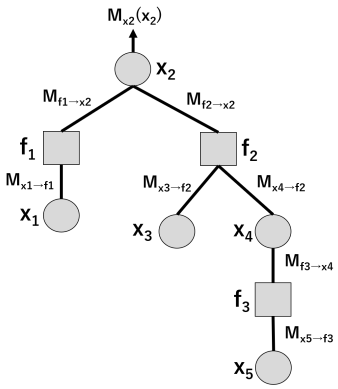

図 4　x_2 を根とするツリー

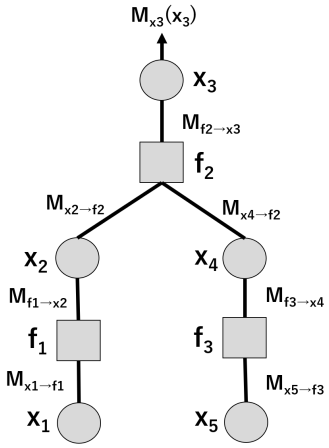

図 5　x_3 を根とするツリー

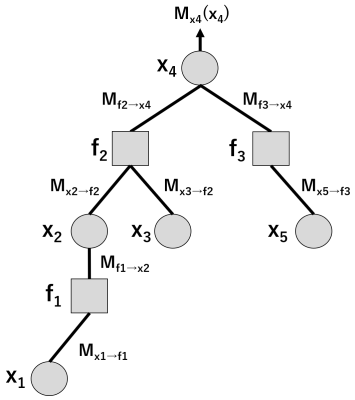

図 6　x_4 を根とするツリー

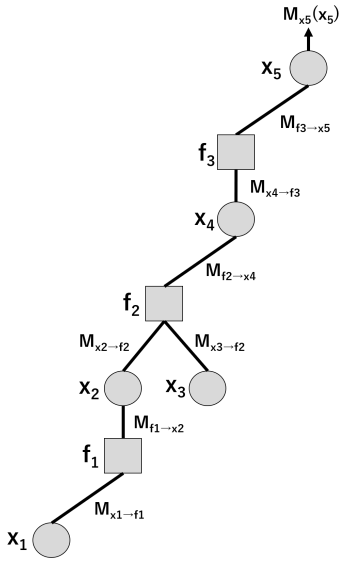

図 7　x_5 を根とするツリー

以上で示された 5 つのツリーにおいて，葉から根方向に向かって計算していくことにより条件付き分布が求まる．実は，図 3〜7 のツリーはすべて同じ構造を持っている．具体的には，変数ノードと関数ノードをそれぞれ上と下に配置して描き直すとどのツリーも図 8 のようになる．このグラフをファクターグラフという． 変数ノードから関数ノード方向への計算と関数ノードから変数ノード方向への計算を繰り返し行うことにより，5 つ同時に条件付き分布を求めることが出来る．得られた条件付き分布を用いてシンボル MAP 復号を行い，これによって

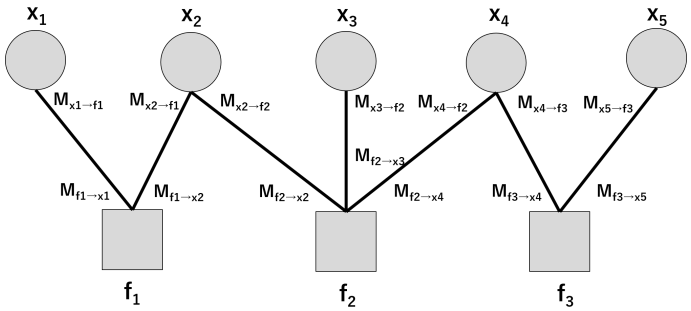

図 8　ファクターグラフ

得られたシンボル列を一時推定語とする．一時推定語とパリティ検査行列との積をとり，もしその結果が 0 であれば一時推定語は符号語であることになり，復号器から受信者に出力される．そうでなければ，一時推定語は符号語とはならず再度上下方向の計算を行う．

3.3　Sum-Product アルゴリズム

続いて，以上で示された処理手順を一般化する．Sum-Product 復号は，LDPC 符号と相性の良い反復復号を代表する復号方法とされている．以下に，一般的な Sum-Product 復号の処理手順をステップに沿って示していく．

ステップ 1

各ノードごとでの処理を行う．図 8 で示したファクターグラフの上部は変数ノード，下部は関数ノードと呼ばれ，パリティ検査行列の行と列に対応している．変数ノード x_k は，関数ノード f_i にメッセージを伝達し，関数ノード f_i は変数ノード x_k にメッセージを伝達する．以下に，変数ノードと関数ノードにおける処理手順を計算式で示す．

変数ノード処理

変数ノード x_k から関数ノード f_i へのメッセージを

$$M_{x_k \to f_i}(x_k) = \prod_{a \in N(x_k) \setminus f_i} M_{a \to x_k}(x_k) \tag{24}$$

として計算する．式中の $M_{x_k \to f_i}(x_k)$ は，$x_k \to f_i$ 方向に伝達されるメッセージを表しており，x_k の関数になっていることを表している．加えて，ファクターグラフに関して，求めたいメッセージのやり取りを行っているもの以外の枝の集合を $N(f_i)$ と表す．

関数ノード処理

関数ノード f_i から変数ノード x_k へのメッセージを

$$M_{f_i \to x_k}(x_k) = \prod_{N(f_i)\setminus x_k} f_i(B_i) \prod_{b \in N(f_i)\setminus x_k} M_{b \to f_i}(b) \tag{25}$$

として計算する．関数ノード f_i においては，f_i のノードから送られてきたメッセージと関数ノードの積を取る．B_i は，パリティ検査行列の i 列目において，1 が立っている行の集合を表している．

ステップ 2

それぞれの変数ノードに入ってきたメッセージの積を計算し，条件付き分布を求める．

条件付き確率の計算

変数ノード x_k に対して，

$$M_{x_k}(x_k) = \prod_{a \in N(x_k)} M_{a \to x_k}(x_k) \tag{26}$$

を計算する．

一時推定シンボル $\hat{x}_k$ を

$$\hat{x}_k = \arg\max M_{x_k}(x_k) \tag{27}$$

と定める．一時推定シンボルを並べたものが一時推定語である．

ステップ 3

ステップ 2 で求められた一時推定語とパリティ検査行列の積をとる．もしその結果が 0 であれば一時推定語は符号語であることになり，復号器から受信者に出力される．そうでなければ，一時推定語は符号語とはならず，再度ステップ 1 に戻り同様の計算を繰り返す．

4　通信路の記憶を考慮した Sum-Product 復号法

3 章では無記憶通信路に対する Sum-Product 復号法の手順を説明した．本章ではこの手順に沿って，記憶のある通信路における Sum-Product 復号法を活用した際の問題点とその解決方法を提示する．

4.1　記憶の影響

本研究で扱う記憶のある通信路に対しては Sum-Product 復号法をそのまま適用することはできない．その理由を説明する．

シンボル x_i を時間間隔にして足し合わせた時刻を $\hat{a}_i(x_1,\ldots,x_i)$ $(i=1,\ldots,5)$ とする．さらに，パケットが通信路から出力された時刻を $\hat{d}_i(i=1,\ldots,5)$ とおく．ここで，符号語 $\boldsymbol{x}$ を通信路に入力したもとで，受信語 $\boldsymbol{y}$ が出力される確率 W を考える．時刻 $\hat{a}_1(x_1)$ に符号器から通信路へパケットが入力されたもとで，時刻 $\hat{d}_1$ でパケットが通信路から出力される確率は $W(\hat{d}_1|\hat{a}_1(x_1))$ となる．さらにその後，符号器から通信路に時刻 $\hat{a_2}(x_1,x_2)$ にパケットが入力されたもとで，時刻 $\hat{d}_2$ で通信路から復号器へ出力される確率は $W(\hat{d}_2|\hat{d}_1\hat{a}_2(x_1,x_2))$ となる．$W(\hat{d}_3|\hat{d}_2\hat{a}_3(x_1,x_2,x_3))$, $W(\hat{d}_4|\hat{d}_3\hat{a}_4(x_1,x_2,x_3,x_4))$, $W(\hat{d}_5|\hat{d}_4\hat{a}_5(x_1,x_2,x_3,x_4,x_5))$ についても同様に考えることが出来る．

以下，このことを踏まえると，x_1 のシンボル MAP 復号のための条件付き確率は，

$$
\begin{aligned}
&\sum_{x_2,x_3,x_4,x_5} f_1(x_1,x_2)f_2(x_2,x_3,x_4)f_3(x_4,x_5)W^5(\hat{d}_1,\ldots,\hat{d}_5|\hat{a}_1,\ldots,\hat{a}_5)\\
&= W(\hat{d}_1|\hat{a}_1(x_1))\\
&\times \sum_{x_2} f_1(x_1,x_2)W(\hat{d}_2|\hat{d}_1\hat{a}_2(x_1,x_2))\\
&\times \sum_{x_3,x_4} f_2(x_2,x_3,x_4)W(\hat{d}_3|\hat{d}_2\hat{a}_3(x_1,x_2,x_3))W(\hat{d}_4|\hat{d}_3\hat{a}_4(x_1,x_2,x_3,x_4))\\
&\times \sum_{x_5} f_3(x_4,x_5)W(\hat{d}_5|\hat{d}_4\hat{a}_5(x_1,x_2,x_3,x_4,x_5)) \qquad (28)
\end{aligned}
$$

と表される．Sum-Product 復号を行うには式 (22) で示したように，総和を計算するために各総和の変数と W 内の変数が一致していなければならない．しかし，式 (27) ではいずれの総和でも一致しておらず，条件付き確率の計算過程をツリー構造で描くことができない．従って図 8 のようなファクターグラフも存在せず，従来の Sum-Product 復号法をそのまま利用することが出来ない．

4.2 提案法

前節で指摘した問題に対して，文献 [3] では変数をひとつ前の反復処理でシンボル MAP 推定されたシンボルに置き換えることで，ツリー構造を描くことを提案している．変数を置き換えた条件付き確率は，

$$
\begin{aligned}
&W(\hat{d}_1|\hat{a}_1(x_1))\\
&\times \sum_{x_2} f_1(x_1,x_2)W(\hat{d}_2|\hat{d}_1\hat{a}_2(\hat{x}_1,x_2))\\
&\times \sum_{x_3,x_4} f_2(x_2,x_3,x_4)W(\hat{d}_3|\hat{d}_2\hat{a}_3(\hat{x}_1,\hat{x}_2,x_3))W(\hat{d}_4|\hat{d}_3\hat{a}_4(\hat{x}_1,\hat{x}_2,\hat{x}_3,x_4))\\
&\times \sum_{x_5} f_3(x_4,x_5)W(\hat{d}_5|\hat{d}_4\hat{a}_5(\hat{x}_1,\hat{x}_2,\hat{x}_3,\hat{x}_4,x_5)) \qquad (29)
\end{aligned}
$$

と表される．

ここでは，各総和において $x_1, \ldots, x_5$ を異なる変数 $\hat{x}_1, \ldots, \hat{x}_4$ に置き換えた．これにより，各総和の変数と W 内の変数が一致し，ツリー構造を描くための条件が成り立つ．以上より，無記憶通信路における Sum-Prodcut 復号法と同様に，式 (28) の各項を以下のように一部の変数を定数に置き換えたことで，各総和を計算できる式として定義できるようになった．

$$
\begin{aligned}
M_{x_1}(x_1) &\triangleq W(\hat{d}_1|\hat{a}_1(x_1))M_{f_1\to x_1}(x_1)\\
M_{f_1\to x_1}(x_1) &\triangleq \sum_{x_2} f_1(x_1,x_2)M_{x_2\to f_1}(x_2)\\
M_{x_2\to f_1}(x_2) &\triangleq W(\hat{d}_2|\hat{d}_1\hat{a}_2(\hat{x}_1,x_2))M_{f_2\to x_2}(x_2)\\
M_{f_2\to x_2}(x_2) &\triangleq \sum_{x_3,x_4} f_2(x_2,x_3,x_4)M_{x_3\to f_2}(x_3)M_{x_4\to f_2}(x_4)\\
M_{x_3\to f_2}(x_3) &\triangleq W(\hat{d}_3|\hat{d}_2\hat{a}_3(\hat{x}_1,\hat{x}_2,x_3))\\
M_{x_4\to f_2}(x_4) &\triangleq W(\hat{d}_4|\hat{d}_3\hat{a}_4(\hat{x}_1,\hat{x}_2,\hat{x}_3,x_4))M_{f_3\to x_4}(x_4)\\
M_{f_3\to x_4}(x_4) &\triangleq \sum_{x_5} f_3(x_4,x_5)M_{x_5\to f_3}(x_5)\\
M_{x_5\to f_3}(x_5) &\triangleq W(\hat{d}_5|\hat{d}_4\hat{a}_5(\hat{x}_1,\hat{x}_2,\hat{x}_3,\hat{x}_4,x_5))
\end{aligned}
$$

これにより，ツリー構造を描くことが出来るようになるため，Sum-Product 復号法の手順に基づき復号することが可能となった．文献 [3] では置き換える変数として，ひとつ前の反復処理でシンボル MAP 推定されたシンボルを利用することが提案されている．そこで，本研究でもこの提案手法を採用し復号性能の検証を行った．

5　復号性能の検証

本研究では，記憶を考慮した Sum-Product 復号法と記憶を考慮しない Sum-Product 復号法 の復号性能を比較検証する．ここで，記憶を考慮しない Sum-Product 復号法とは，通信路を無記憶とみなしたもとで実験的に求めた通信路行列に対して，Sum-Product 復号法を適用したものを指す．

第一に符号語長が短い符号を用いて，シンボル MAP 復号と Sum-Product 復号の復号性能の比較検証と Sum-Product 復号において記憶を考慮する場合としない場合の復号性能の比較検証を行い，傾向を掴む．第二に，符号語長を伸ばした際の問題点を浮き彫りにする．第三に，問題設定を実際の通信路に近づけるため，符号語長が長い場合における復号誤り確率を測定する．

5.1　符号語長 5

まず，符号語長が短い場合においてシンボル MAP 復号と Sum-Product 復号の復号誤り確率を比較したものを図 9 に示す．ここでは，5 列 3 行のパリティ検査行列をもつ符号を使用した．この図では，符号化レートが大きく復号誤り確率が低ければ復号性能が良いとされる．そこで，符号化レートが同じ場合における復号誤り確率を比較した際に，Sum-Product 復号の復号誤り確率がシンボル MAP 復号より高くなっている様子が確認できる．これより，シンボル MAP 復号と比べて Sum-Product 復号の性能が低くなっていることが分かる．原因として，4 章で示したように Sum-Product 復号を行う際に式を別の値で置き換え，本来の式とは異なるものを計算したことが考えられる．

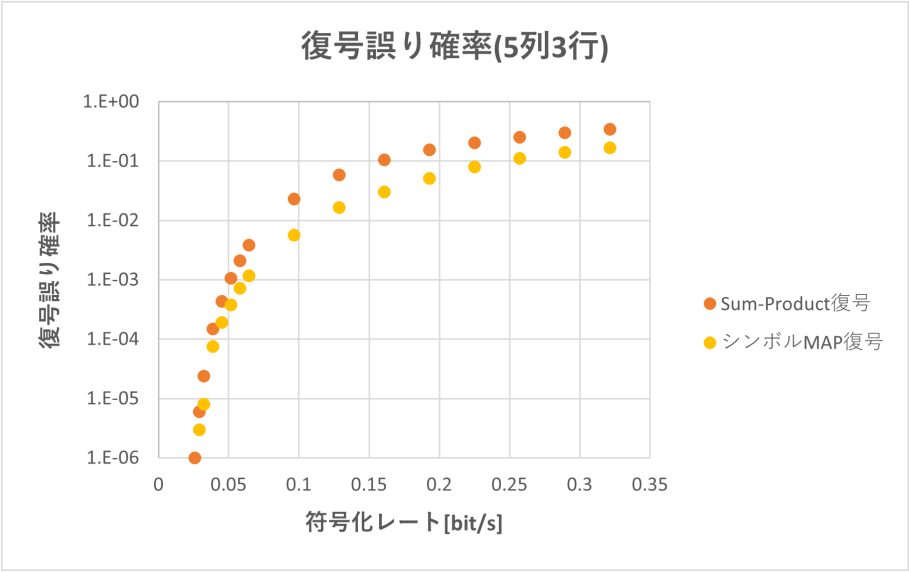

図 9　符号語長 5 記憶あり通信路

さらに，符号語長が短い場合に，Sum-Product 復号において記憶を考慮する場合と考慮しない場合の復号性能を比較したものを図 10 に示す．符号化レートが同じ場合における復号誤り確率を比較した際に，記憶を考慮する場合が記憶を考慮しない場合より低くなっている様子が確認できる．これより，記憶を考慮することで復号性能が高くなっていることが分かる．これは，前に受信したシンボルを考慮する記憶を考慮した Sum-Product 復号 の性質が現れたと考えられる．

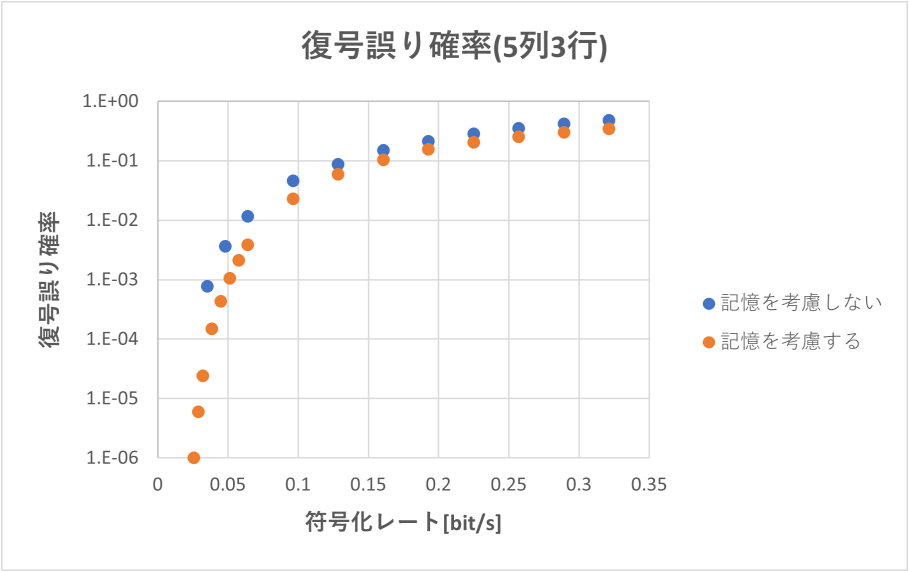

図 10　符号語長 5 Sum-Product 復号 比較

5.2 符号語長 20

符号語長を大きくした際に，Sum-Product 復号において記憶を考慮する場合と考慮しない場合の復号性能を比較したものを図 11 に示す．ここでは，20 列 8 行のパリティ検査行列をもつ符号を使用した．符号化レートが同じ場合における復号誤り確率を比較した際に，記憶を考慮した Sum-Product 復号 が記憶を考慮しない Sum-Product 復号より低くなっている様子が確認できる．これより，符号語長 5 の時と同様に記憶を考慮することで復号性能が高くなっていることが分かる．しかし，シミュレーション実験において出力結果を分析したところ， Sum-Product 復号の反復回数が増えた際に丸め誤差が発生してしまうことが確認された．これにより，反復回数が多くなる傾向にある記憶を考慮した Sum-Product 復号 においては，記憶を考慮しない場合より復号誤り確率が高くなり，復号性能が低くなってしまっていると考えられる．

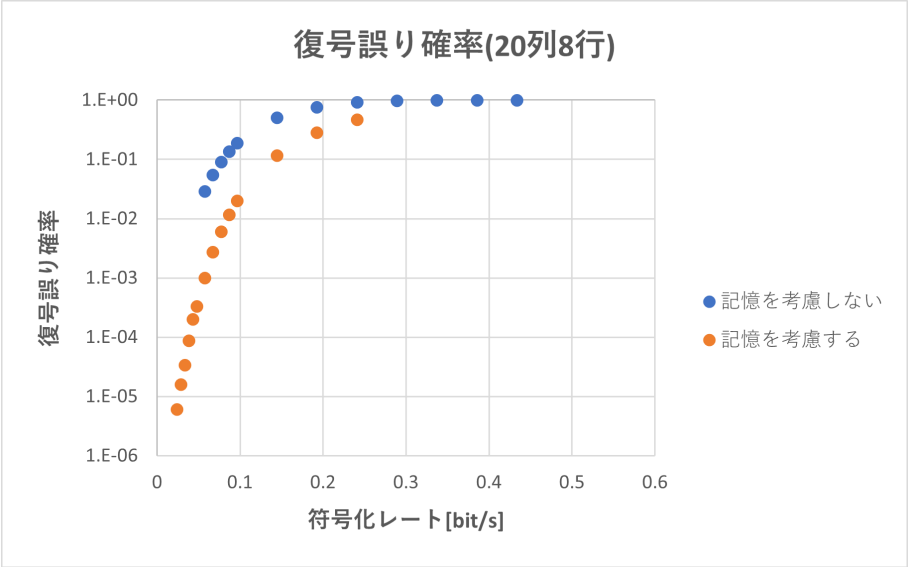

図 11　符号語長 20 Sum-Product 復号 比較

5.3　符号語長 900

最後に，符号語長が長い場合に，Sum-Product 復号の記憶を考慮する場合と考慮しない場合の復号性能を比較したものを図 12 に示す．ここでは，900 列 540 行のパリティ検査行列を活用する．記憶を考慮しない Sum-Product 復号が記憶を考慮する場合よりも高い復号性能を示している．これは，符号語長 20 で示したように， Sum-Product 復号の反復回数がさらに増え，丸め誤差が大きく表れてしまったためだと考えられる．

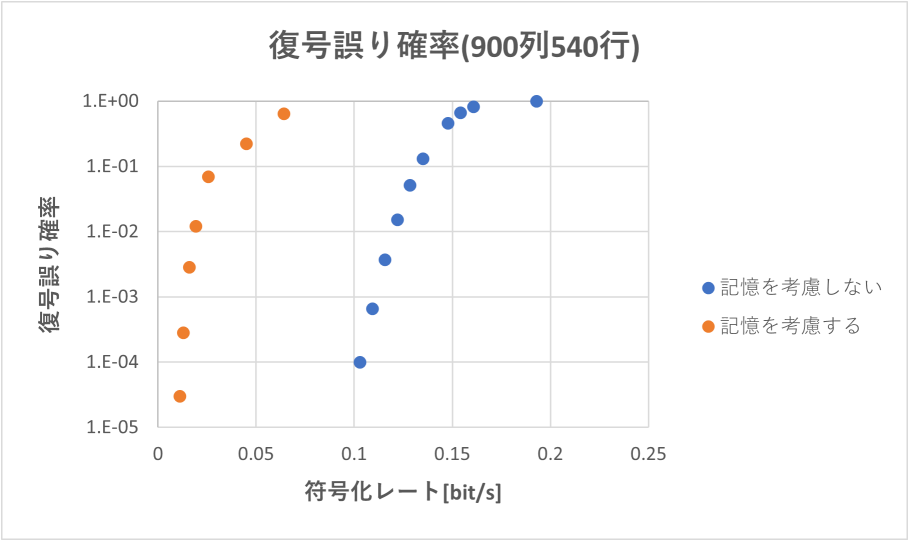

図 12　符号語長 900 Sum-Product 復号 比較

6 まとめ

本研究では，パケット間隔で情報を送る通信路に対して，記憶を考慮した上で Sum-Product 復号法を適用し，符号語長が長い場合における復号性能を検証した．その際，比較対象として記憶を考慮しない Sum-Product 復号法の復号性能も測定した．その結果，符号語長が長くなるにつれて丸め誤差が大きくなり，正確な値を出力することが出来ず，記憶を考慮する場合における復号性能の方が低く示されてしまった．今後の課題として，より良いパリティ検査行列を用いて Sum-Product 復号の反復計算の回数を減らすことで丸め誤差を減らし，復号性能の向上を図ることが挙げられる．

謝辞

本研究を行うにあたり，数多くの助言，指導をしてくださった指導教員西新幹彦准教授に感謝の意を表する．

参考文献

[1] Venkat Anantharam, Sergio Verdú, “Bits Through Queues,” IEEE Transactions on Information Theory, vol.42, no.1, pp.4–18, Jan. 1996.

[2] Todd P. Coleman, Negar Kiyavash, “Practical Codes for Queueing Channels: An algebraic, state-space, message-passing approach,” IEEE Information Theory Workshop, pp.318–322, 2008.

[3] 市川謙吾，「パケット間隔で情報を送る通信路の記憶を考慮した Sum-Product 復号法の提案」，信州大学大学院総合理工学研究科修士論文（指導教員：西新幹彦），2020 年 3 月．

[4] 佐藤奏杜，「パケット間隔で情報を送る通信路の記憶を考慮した Sum-Product 復号法の検証」，信州大学工学部卒業論文（指導教員：西新幹彦），2021 年 3 月．

[5] 飯島一貴，「パケット間隔で情報を送る通信路に対する誤り訂正符号の構成に関する考察」，信州大学大学院総合理工学研究科修士論文（指導教員：西新幹彦），2017 年 3 月．

[6] 藤山直貴，「パケット間隔で情報を送る通信路に対する悲観的符号化定理」，信州大学大学院総合理工学研究科修士論文（指導教員：西新幹彦），2012 年 3 月．

[7] 萩原学，符号理論，日本評論社，2012 年 8 月．

[8] 和田山正，誤り訂正技術の基礎，森北出版，2010 年 7 月．

付録 A　ソースコード

乱数生成に活用したプログラム
Mersenne Twister with improved initialization,
http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/MT2002/mt19937ar.html,
(参照 2023-01-30)

A.1　正則 LDPC 符号の構成

```
//正則 LDPC 符号の構成：20 列 8 行
#include <stdio.h>
#include <math.h>
#include <stdarg.h>
#include <float.h>
#include <stdlib.h>

//(λ , ρ)=(縦, 横) ,r=個数
int l = 2;
int p = 5;
int r = 4;

//(p*r)*(p*l)
int n = 160;
char pcm[160];

int matrix(){
    FILE *fp;
    fp = fopen("test1.txt","w");

    int column = p*r;
    int i = 0;

    //0 を p 個× a セット目
    for (int a = 0; a < r; a++){
        //111... の前の 000 が何セット入るか？
        for (int b = 0; b < l; b++){
            for (int c = 0; c < a; c++){
                //111... の前の 000... を記述
                for (int d = 0; d < p; d++){
                    pcm[i] = 0;
                    i += 1;
                }
            }
            //000... の後の 111... を記述
            for (int e = 0; e < p; e++){
                pcm[i] = 1;
                i += 1;
            }

            //列数 - ( 0 の数 + 1 の数 )
            for(int f = 0; f < column - (p*a + p); f++){
                pcm[i] = 0;
                i += 1;
            }
        }
    }


    /* 行列表示 */
    for (int k = 0; k < n; k++){
```

```
        //行列改行
        if((k % (p * r)) == 0){
            //fprintf(fp,"\n");
        }
        //fprintf(fp, "%d,", pcm[k]);
    }


    /* 行列内の 1 の位置を示す配列 */
    int mat1[p*r*l];
    int m = 0;
    for (int l = 0; l < n; l++){
        if(pcm[l] == 1){
            mat1[m] = l;
            m += 1;
        }
    }


    /* 行列内の 1 の位置を表示 */
    for (int i = 0; i < p * r * l; i++){
        if((i % 10) == 0){
            //fprintf(fp,"\n");
        }
        //fprintf(fp,"%d,",mat1[i]);
    }


    /* 行列内の 0 の位置を示す配列 */
    int mat0[n - p*r*l];
    int o = 0;
    for (int p = 0; p < n; p++){
        if(pcm[p] == 0){
            mat0[o] = p;
            o += 1;
        }
    }

    /* 行列内の 0 の位置を表示 */
    for (int i = 0; i < (n - p*r*l); i++){
        if((i % 10) == 0){
            //fprintf(fp, "\n");
        }
        //fprintf(fp, "%d,", mat0[i]);
    }
    fclose(fp);

}

int main(){
    matrix();
}
```

A.2 シャッフルを用いた構成

```
//シャッフルを用いた構成：20 列 8 行
#include <stdio.h>
#include "random.h"//乱数生成

/* 『正則 LDPC 符号の構成』で作成した行列 */
int row = 8;
int column = 20;
int count = 160;
int pcm[160] = {
```

```
    1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1
};

//0 の位置
int mat0[120] = {
    5,6,7,8,9,10,11,12,13,14,
    15,16,17,18,19,25,26,27,28,29,
    30,31,32,33,34,35,36,37,38,39,
    40,41,42,43,44,50,51,52,53,54,
    55,56,57,58,59,60,61,62,63,64,
    70,71,72,73,74,75,76,77,78,79,
    80,81,82,83,84,85,86,87,88,89,
    95,96,97,98,99,100,101,102,103,104,
    105,106,107,108,109,115,116,117,118,119,
    120,121,122,123,124,125,126,127,128,129,
    130,131,132,133,134,140,141,142,143,144,
    145,146,147,148,149,150,151,152,153,154
};

//1 の位置
int mat1[40] = {
    0,1,2,3,4,20,21,22,23,24,
    45,46,47,48,49,65,66,67,68,69,
    90,91,92,93,94,110,111,112,113,114,
    135,136,137,138,139,155,156,157,158,159
};

//行列内の 0,1 の数
int num0 = 120;
int num1 = 40;


/* 行列をシャッフルする */
int exchange(int a[]){
int a11;//1 回目の 1 の mat1 における要素番号
int a21;//2 回目の 1 の mat1 における要素番号

    int a10;//1 回目の 0 の mat1 における要素番号
    int a20;//2 回目の 0 の mat1 における要素番号

    int x11;//1 回目の 1 の pcm における要素番号
    int x21;//1 回目の 1 の pcm における要素番号

    int x10;//1 回目の 0 の pcm における要素番号
    int x20;//2 回目の 0 の pcm における要素番号

    int row11;//『1 つ目の 1』の行
    int column11;//『1 つ目の 1』の列

    int row21;//『2 つ目の 1』の行
    int column21;//『2 つ目の 1』の列

int x;
int y;

    int l;
    int m;

    //【1 つ目】行列の中から『1』をランダムで選ぶ
a11 = (double)genrand_real2() * num1;
    l = mat1[a11];
    row11 = l / column;
```

```
    column11 = l % column;
    x11 = l;

    //【2 つ目】行列の中から『1』をランダムで選ぶ
a21 = (double)genrand_real2() * num1;
    m = mat1[a21];
    row21 = m / column;
    column21 = m % column;
    x21 = m;

    //【2 つ目】の 1 が【1 つ目】の 1 と同じ行じゃなければ操作を続ける
    if((row11 != row21) && (column11 != column21)){

        x10 = row21*column + column11;
        x20 = row11*column + column21;

        //x10 と x20 がいずれも 0 の場合、0 と 1 を入れ替える
        if ((pcm[x10] == 0) && (pcm[x20] == 0)){
            int p = pcm[x11];
            int q = pcm[x21];
            pcm[x11] = pcm[x10];
            pcm[x21] = pcm[x20];
            pcm[x10] = p;
            pcm[x20] = q;

            //1 と 0 のそれぞれの要素番号の配列を入れ替える
            //0 の要素番号を探す
            for (int i = 0;i < num0;i++){
                if(mat0[i] == x10){
                    a10 = i;
                }
                if(mat0[i] == x20){
                    a20 = i;
                }
            }

            int s = mat1[a11];
            int t = mat1[a21];

            //1 と 0 の要素番号を入れ替える
            mat1[a11] = mat0[a10];
            mat1[a21] = mat0[a20];
            mat0[a10] = s;
            mat0[a20] = t;
        }
    }

//関数引き渡し用
for (int k = 0; k < count; k++){
a[k] = pcm[k];
    }

    return 0;
}

int main(){
    FILE *fp;
    fp = fopen("test2.txt","w");

    unsigned long init[4] = {0x234, 0x456, 0x345, 0x123}, length = 4;
init_by_array(init, length);

int x;
int a[160];

for (int i = 0; i < 1000000; i++){
exchange(a);
}
```

```
    for (int k = 0; k < count; k++){
        if((k % column) == 0){
            fprintf(fp, "\n");
            printf("\n");
        }
        fprintf(fp, "%d,", pcm[k]);
        printf("%d,", pcm[k]);
    }
    fclose(fp);
}
```

A.3 記憶を考慮しない Sum=Product 復号のメタプログラム

```
//【メタプログラム】記憶を考慮しない Sum-Product 復号（対数）：5 列 3 行
#include <stdio.h>
#include <stdarg.h>
#include "random.h"

#define ALPHABETSIZE 4//シンボル数

int row = 3;
int column = 5;
char pcm[15] = {
    1,1,0,0,0,
    0,1,1,1,0,
    0,0,0,1,1
};

/* インデント生成 */
int tprintf(int n, char *fmt, ...)
{
va_list arg;

while (n-- > 0){
putchar('\t');
}
va_start(arg, fmt);
int ret = vprintf(fmt, arg);
va_end(arg);
return(ret);
}


/* 符号器設計 &&  復号誤り判定設計 */
int encode(){
    FILE *fp;
    fp = fopen("test2.txt","w");

    /* 符号器設計 */
    int j = 0;
    int rank = row;
int index[5] = {0,1,2,3,4};//手動で変更する

    for (int i = 0; i < row; i++){
        //行入れ替え
        if (pcm[i*column + j] == 0){
            //0 だったら j+1,j+2.. 列の中から 1 を見つけて交換
            int l;
            for (l = 1 ; l < column; l++){
                if (pcm[i*column + l] != 0){
                    break;
                }
            }
```

```
            //見つからなかった場合
            if(l >= column){
                rank--;
                continue;
            }

            //列入れ替え
            for(int k = 0;k < row;k++){
                int tmp = pcm[k*column + l];
                pcm[k*column + l] = pcm[k*column + j];
                pcm[k*column + j] = tmp;
            }
            int tmp = index[l];
            index[l] = index[j];
            index[j] = tmp;
        }


        for(int k = 0; k < row;k++){
            if (k == i) continue;
            if (pcm[k*column + j] == 0) continue;
            for (int l = 0; l < column;l++){
                pcm[k*column + l] ^= pcm[i*column + l];//行の引き算
            }
        }

        j++;
    }

    //符号器出力
    fprintf(fp,"\n");
    fprintf(fp,"void encode(");

for (int j = rank; j < column ; j++){
        fprintf(fp,"int x%d, ",index[j]);
}
    fprintf(fp,"int x[]){\n");

    /* 復号誤り判定設計 */
int h;
j = 0;
for (int i = 0; i < row; i++){
if (pcm[i*column + j] == 0) continue;
        fprintf(fp,"    x[%d] = 0",index[j]);

for(int j0 = rank; j0 < column;j0++){
if (pcm[i*column + j0] == 0) continue;
            fprintf(fp," ^ x%d",index[j0]);
}
j++;
        fprintf(fp,";\n");
}

for (int i = rank; i < column;i++){
        fprintf(fp,"    x[%d] = x%d;\n", index[i], index[i]);
}

    fprintf(fp,"    return;\n");
    fprintf(fp,"}\n");
    fprintf(fp,"\n");
    fprintf(fp,"int count(){\n");
    fprintf(fp,"    int x[%d];\n", column);
    fprintf(fp,"    int xh[%d];\n", column);
    fprintf(fp,"    double a[%d];\n", column);
fprintf(fp,"    double d[%d];\n", column + 1);
    fprintf(fp,"    int num;\n");

for(int i = rank; i < column;i++){
        fprintf(fp,"    int x%d =  genrand_real2() * ALPHABETSIZE;\n", index[i]);
```

```
}

fprintf(fp,"\n");
    fprintf(fp,"    encode(");

for(int i = rank; i < column;i++){
        fprintf(fp,"x%d, ",index[i]);
}

    fprintf(fp,"x");
    fprintf(fp,");\n");
    fprintf(fp,"    code2interval(x,a);\n");
    fprintf(fp,"    transmit(a,d);\n");
fprintf(fp,"    decode(d,xh);\n");
    fprintf(fp,"\n");
    fprintf(fp,"    for (num = 0;num < %d;num++){\n",column);
    fprintf(fp,"        if(xh[num] != x[num]){\n");
    fprintf(fp,"            break;\n");
    fprintf(fp,"        }\n");
    fprintf(fp,"    }\n");
    fprintf(fp,"    return (num < %d);\n", column);
    fprintf(fp,"}\n");

    fclose(fp);
    return (0);
}

/* 符号器 → 通信路 */
int code2interval(){
    printf("double code2interval(int x[],double a[]){\n");
    tprintf(1,"double u[%d];\n",column);
    tprintf(1,"for(int k = 0; k < %d; k++){\n",column);
    tprintf(2,"a[k] = expdistinv(rvariable(x[k]), SRCRATE);\n");
    tprintf(1,"}\n");
    printf("}\n");
}

/* 通信路 */
int interval(){
    printf("void transmit(double a[],double rcvtime[]){\n");
    tprintf(1,"int i = 0;\n");
    tprintf(1,"int j = 1;\n");
    tprintf(1,"int k;\n");
    tprintf(1,"double current_time = 0.0;\n");
    tprintf(1,"int customer_num = 0;\n");
    tprintf(1,"double sendtime;\n");
    tprintf(1, "double service_end = 0.0;\n");
    tprintf(1, "sendtime = a[0];\n");
    tprintf(1, "rcvtime[0] = 0.0;\n");
    tprintf(1, "while (i < %d || customer_num > 0){\n",column);
    tprintf(2, "if (customer_num == 0 || (i < %d && sendtime < service_end)){\n",column);
    tprintf(3, "current_time = sendtime;\n");
    tprintf(3, "i++;\n");
    tprintf(3, "if (i < %d){\n",column);
    tprintf(4, "sendtime += a[i];\n");
    tprintf(3, "}\n");
    tprintf(3, "if (customer_num == 0){\n");
    tprintf(4, "service_end = current_time +   expdistinv(genrand_real2(), SERVICERATE);\n");
    tprintf(3, "}\n");
    tprintf(3, "customer_num++;\n");
    tprintf(2, "}\n");
    tprintf(2, "else {\n");
    tprintf(3, "current_time = service_end;\n");
    tprintf(3, "rcvtime[j] = service_end;\n");
    tprintf(3, "j++;\n");
    tprintf(3, "customer_num--;\n");
    tprintf(3, "if (customer_num >= 1){\n");
    tprintf(4, "service_end = current_time + expdistinv(genrand_real2(), SERVICERATE);\n");
    tprintf(3, "}\n");
```

```
    tprintf(2, "}\n");
    tprintf(1, "}\n");
    tprintf(1, "return;\n");
    printf("}\n");
}

/* 復号器設計 */
int decode(){
    FILE *fp;
    fp = fopen("test3.txt","w");
    fprintf(fp,"void decode(double dep[], int xh[]){\n");

    /* 通信路行列 (確率) */
    fprintf(fp,"    double prob[");
    fprintf(fp,"%d][%d];\n",ALPHABETSIZE,ALPHABETSIZE);

    /* 受信語 */
    fprintf(fp,"    int y[");
    fprintf(fp,"%d",column);
    fprintf(fp,"] = {0");
    for(int i = 0;i < column - 1;i++){
        fprintf(fp,",0");
    }
    fprintf(fp,"};\n");

    /* 通信路に入れた後の間隔を整数に */
    fprintf(fp,"    for (int i = 0; i < ");
    fprintf(fp,"%d",column);
    fprintf(fp,"; i++){\n");

    int i = ALPHABETSIZE;
    double d;
    fprintf(fp,"        if ((dep[i + 1] - dep[i]) < -(log(%f)/SRCRATE)){\n",1 - (double)1 / (double)ALPHABETSIZE);
    fprintf(fp,"            y[i] = 0;\n");

    for (int i = 1; i < ALPHABETSIZE - 1; i++){
        d = 1 - (double)(i + 1) / (double)ALPHABETSIZE;
        fprintf(fp,"        }else if ((dep[i + 1] - dep[i]) < -(log(%f)/SRCRATE)){\n",d);
        fprintf(fp,"            y[i] = %d;\n", i);
    }

    fprintf(fp,"        }else{\n");
    fprintf(fp,"            y[i] = %d;\n", ALPHABETSIZE - 1);
    fprintf(fp,"        }\n");
    fprintf(fp,"    }\n");
    fprintf(fp,"\n");

    /*通信路行列 (頻度) ⇒ (確率)*/
    fprintf(fp,"    gen_prob_mat(sample, prob);\n");
    fprintf(fp,"\n");

/* ○ ⇒ □の配列準備 */
for (int i = 0; i < row; i++){
for (int j = 0; j < column; j++){
if (pcm[i*column + j] == 0) continue;
//tprintf(1,"double Mx%df%d[%d];\n",j, i,ALPHABETSIZE);//tab 別の書き方//☆
            fprintf(fp,"    double Mx%df%d[%d];\n",j, i,ALPHABETSIZE);//☆
}
    }
    fprintf(fp,"\n");

/* □ ⇒ ○の配列準備 */
for (int i = 0; i < row; i++){
for (int j = 0; j < column; j++){
if (pcm[i*column + j] == 0) continue;
            fprintf(fp,"    double Mf%dx%d[%d] = {0", i, j,ALPHABETSIZE);//☆☆☆☆
            for(int k = 0 ;k < ALPHABETSIZE - 1;k++){
                fprintf(fp,",0");
            }
```

```
            fprintf(fp,"};\n");
        }
}
    fprintf(fp,"\n");

/*繰り返し処理*/
    fprintf(fp,"    for (int n = 1; n < 100; n++){\n");
    /*○ ⇒ □*/
    fprintf(fp,"        ");
    fprintf(fp,"/* ○ ⇒ □ */");
    fprintf(fp,"\n");
for (int i = 0; i < row; i++){
for (int j = 0; j < column; j++){
if (pcm[i*column + j] == 0) continue;
            fprintf(fp,"        for (int x=0; x < %d;x++){\n",ALPHABETSIZE);////☆
            fprintf(fp,"            Mx%df%d[x] = prob[y[%d]][x]", j, i,j );
for (int k = 0; k < row; k++){
if (k == i) continue;
if (pcm[k*column + j]){
                    fprintf(fp," + Mf%dx%d[x]", k, j);
}
}
            fprintf(fp,";\n");
            fprintf(fp,"        }\n");
}
}
    fprintf(fp,"\n");

    /* □ ⇒ ○ */
    fprintf(fp,"        ");
    fprintf(fp,"/* □ ⇒ ○ */\n");
    for (int i = 0; i < row; i++){
        for (int j = 0; j < column; j++){
            if (pcm[i*column + j] == 0) continue;
            fprintf(fp,"        ");
            fprintf(fp,"for (int x = 0; x < %d ;x++){\n",ALPHABETSIZE);//☆
            fprintf(fp,"            ");
            fprintf(fp,"Mf%dx%d[x] = 0;\n", i, j);
            fprintf(fp,"        ");
            fprintf(fp,"}\n");

            for (int j0 = 0; j0 < column; j0++){
                if (j0 == j) continue;
                if (pcm[i*column + j0] == 0) continue;
                fprintf(fp,"        ");
                fprintf(fp,"for (int x%d = 0; x%d < %d; x%d++){\n", j0, j0, ALPHABETSIZE,j0);//☆
            }
            fprintf(fp,"            ");
            fprintf(fp,"int x = 0");

            for (int j0 = 0; j0 < column; j0++){
                if (j0 == j) continue;
                if (pcm[i*column + j0] == 0) continue;
                fprintf(fp," ^ x%d",j0);
            }

            fprintf(fp,";\n");
            fprintf(fp,"            ");
            fprintf(fp,"Mf%dx%d[x] += 0", i, j);

            for (int j0 = 0; j0 < column; j0++){
                if (j0 == j) continue;
                if (pcm[i*column + j0] == 0) continue;
                fprintf(fp," + Mx%df%d[x%d]", j0, i, j0);
            }
            fprintf(fp,";\n");
            for (int j0 = 0; j0 < column; j0++){
                if (j0 == j) continue;
                if (pcm[i*column + j0] == 0) continue;
```

```
                fprintf(fp,"        ");
                fprintf(fp,"}\n");

            }
            fprintf(fp,"\n");
        }
    }

    /* 最尤推定 */
    fprintf(fp,"        ");
    fprintf(fp,"/* 最尤推定 */\n");
    fprintf(fp,"        ");
    fprintf(fp,"double max;\n");
for (int j = 0;j<column;j++){
            fprintf(fp,"        ");
            fprintf(fp,"max = -DBL_MAX;\n");
            fprintf(fp,"        ");
fprintf(fp,"for (int x = 0; x < %d; x++){\n",ALPHABETSIZE);//☆
            fprintf(fp,"            ");
            fprintf(fp,"double lh =  prob[y[%d]][x]",j);
            for (int i = 0; i < row;i++){
if (pcm[i*column + j] == 0) continue;
                fprintf(fp," + Mf%dx%d[x]", i, j);
}
            fprintf(fp,";\n");
            fprintf(fp,"            ");
            fprintf(fp,"if (max < lh){\n");
            fprintf(fp,"            ");
fprintf(fp,"max = lh;\n");
            fprintf(fp,"            ");
            fprintf(fp,"xh[%d] = x;\n",j,j);

            fprintf(fp,"            ");
            fprintf(fp,"}\n");
            fprintf(fp,"        ");
fprintf(fp,"}\n");
            fprintf(fp,"\n");
}

//符号語かどうかの判断
    fprintf(fp,"        ");
    fprintf(fp,"/*符号語の判定*/\n");
    fprintf(fp,"        ");
    fprintf(fp,"if (1");

for (int i = 0; i < row; i++){
        fprintf(fp," && ((0");
for (int j = 0;j < column; j++){
if (pcm[i*column + j] == 0) continue;
                fprintf(fp," ^ xh[%d]", j);
}
        fprintf(fp,") == 0)");
}
    fprintf(fp,"){\n");
fprintf(fp,"            break;\n");
    fprintf(fp,"        }\n");
    fprintf(fp,"    }\n");
fprintf(fp,"    return;\n");
    fprintf(fp,"}\n");
    fprintf(fp,"\n");


    fclose(fp);
return (0);

}

int main(){
    //encode();
```

```
    //code2interval();
    //interval();
    decode();
}
```

A.4 記憶を考慮しない Sum-Product 復号

```
//記憶を考慮しない Sum-Product 復号 (対数)：5 列 3 行
#include <stdio.h>
#include <math.h>
#include <stdarg.h>
#include <float.h>
#include <stdlib.h>
#include "random.h"//乱数生成

//出力レート [symbol/s]
#define SRCRATE (1/2.71828182845904523536028747135 27)
//#define SRCRATE (0.9/2.71828182845904523536028747135 27)
//#define SRCRATE (0.8/2.71828182845904523536028747135 27)
//#define SRCRATE (0.7/2.71828182845904523536028747135 27)
//#define SRCRATE (0.6/2.71828182845904523536028747135 27)
//#define SRCRATE (0.5/2.71828182845904523536028747135 27)
//#define SRCRATE (0.4/2.71828182845904523536028747135 27)
//#define SRCRATE (0.3/2.71828182845904523536028747135 27)
//#define SRCRATE (0.2/2.71828182845904523536028747135 27)
//#define SRCRATE (0.1/2.71828182845904523536028747135 27)


#define ALPHABETSIZE 4  //シンボル数
#define SERVICERATE 1.0 //サービスレート
#define TRIAL 1000000   //試行回数

/* 確率変数 */
double rvariable(int x){
    return ((2 * x + 1) * 0.125);
}

/* 分布関数の逆関数 */
double expdistinv(double x, double rate){
    if ((1-x) < 0){
        printf("%d %f\n", __LINE__, 1 - x);
        exit(1);
    }
    return(-log(1 - x) / rate);
}

/* 間隔変換 */
double sym2interval(int sym){
    return(expdistinv((sym + 0.5) / ALPHABETSIZE, SRCRATE));
}

/* 指数分布の分布関数 */
double expdistfunc(double x, double rate){
    return(1 - exp(-x * rate));
}

/* 大小比較 */
double max(double a, double b) {
    if (a > b)
        return a;
    else
        return b;
}
```

```
/* 通信路容量（頻度）：メタプログラムより出力 */
int sample[ALPHABETSIZE][ALPHABETSIZE] = {
    21815697,685869,445,0,
    450802,21856432,195569,1,
    58,69810,22424155,7259,
    0,0,55,22493848
};


/* 通信路行列（頻度）⇒（確率） */
void gen_prob_mat(int sample[ALPHABETSIZE][ALPHABETSIZE], double prob[ALPHABETSIZE][ALPHABETSIZE])
{
    int i, j;
    double total[ALPHABETSIZE];

    for (i = 0; i<ALPHABETSIZE; i++){
        total[i] = 0.0;
    }
    for (i = 0; i<ALPHABETSIZE; i++){
        for (j = 0; j<ALPHABETSIZE; j++){
            total[i] += sample[i][j];
        }
    }
    for (i = 0; i<ALPHABETSIZE; i++){
        for (j = 0; j<ALPHABETSIZE; j++){
            prob[i][j] = log((double)sample[i][j] / total[i]);//通信路行列（頻度）⇒（確率）⇒（対数）
        }
    }
    return;
}


//メタプログラム出力結果（ここから）

/* 符号器 */
void encode(int x2, int x4, int x[]){
    x[0] = 0 ^ x2 ^ x4;
    x[1] = 0 ^ x2 ^ x4;
    x[3] = 0 ^ x4;
    x[2] = x2;
    x[4] = x4;
    return;
}

/* 符号器 → 通信路 */
double code2interval(int x[],double a[]){
        double u[5];
        for(int k = 0; k < 5; k++){
                a[k] = expdistinv(rvariable(x[k]), SRCRATE);
        }
}

/* 通信路 */
void transmit(double a[],double rcvtime[]){
        int i = 0;
        int j = 1;
        int k;
        double current_time = 0.0;
        int customer_num = 0;
        double sendtime;
        double service_end = 0.0;
        sendtime = a[0];
        rcvtime[0] = 0.0;
        while (i < 5 || customer_num > 0){
                if (customer_num == 0 || (i < 5 && sendtime < service_end)){
                        current_time = sendtime;
                        i++;
                        if (i < 5){
                                sendtime += a[i];
```

```
                        }
                        if (customer_num == 0){
                                service_end = current_time +   expdistinv(genrand_real2(), SERVICERATE);
                        }
                        customer_num++;
                }
                else {
                        current_time = service_end;
                        rcvtime[j] = service_end;
                        j++;
                        customer_num--;
                        if (customer_num >= 1){
                                service_end = current_time + expdistinv(genrand_real2(), SERVICERATE);
                        }
                }
        }
        return;
}

/* 復号器 */
void decode(double dep[], int xh[]){
    double prob[4][4];
    int y[5] = {0,0,0,0,0};
    for (int i = 0; i < 5; i++){
        if ((dep[i + 1] - dep[i]) < -(log(0.750000)/SRCRATE)){
            y[i] = 0;
        }else if ((dep[i + 1] - dep[i]) < -(log(0.500000)/SRCRATE)){
            y[i] = 1;
        }else if ((dep[i + 1] - dep[i]) < -(log(0.250000)/SRCRATE)){
            y[i] = 2;
        }else{
            y[i] = 3;
        }
    }

    gen_prob_mat(sample, prob);

    double Mx0f0[4];
    double Mx1f0[4];
    double Mx1f1[4];
    double Mx2f1[4];
    double Mx3f1[4];
    double Mx3f2[4];
    double Mx4f2[4];

    double Mf0x0[4] = {0,0,0,0};
    double Mf0x1[4] = {0,0,0,0};
    double Mf1x1[4] = {0,0,0,0};
    double Mf1x2[4] = {0,0,0,0};
    double Mf1x3[4] = {0,0,0,0};
    double Mf2x3[4] = {0,0,0,0};
    double Mf2x4[4] = {0,0,0,0};

    for (int n = 1; n < 100; n++){
        /* ○ ⇒ □ */
        for (int x=0; x < 4;x++){
            Mx0f0[x] = prob[y[0]][x];
        }
        for (int x=0; x < 4;x++){
            Mx1f0[x] = prob[y[1]][x] + Mf1x1[x];
        }
        for (int x=0; x < 4;x++){
            Mx1f1[x] = prob[y[1]][x] + Mf0x1[x];
        }
        for (int x=0; x < 4;x++){
            Mx2f1[x] = prob[y[2]][x];
        }
        for (int x=0; x < 4;x++){
            Mx3f1[x] = prob[y[3]][x] + Mf2x3[x];
```

```
}
for (int x=0; x < 4;x++){
    Mx3f2[x] = prob[y[3]][x] + Mf1x3[x];
}
for (int x=0; x < 4;x++){
    Mx4f2[x] = prob[y[4]][x];
}

/*□ ⇒ ○*/
for (int x = 0; x < 4 ;x++){
    Mf0x0[x] = 0;
}
for (int x1 = 0; x1 < 4; x1++){
    int x = 0 ^ x1;
    if(Mf0x0[x] == 0.00){
        Mf0x0[x] = 0 + Mx1f0[x1];
    }else{
        if(Mf0x0[x] - (0 + Mx1f0[x1]) < 100){
            Mf0x0[x] = log(exp(Mf0x0[x] - (0 + Mx1f0[x1])) + 1) + (0 + Mx1f0[x1]);
        }else{
            Mf0x0[x] = Mf0x0[x];
        }
    }
}

for (int x = 0; x < 4 ;x++){
    Mf0x1[x] = 0;
}
for (int x0 = 0; x0 < 4; x0++){
    int x = 0 ^ x0;
    if(Mf0x1[x] == 0.00){
        Mf0x1[x] = 0 + Mx0f0[x0];
    }
    if(Mf0x1[x] - (0 + Mx0f0[x0]) < 100){
        Mf0x1[x] = log(exp(Mf0x1[x] - (0 + Mx0f0[x0])) + 1) + (0 + Mx0f0[x0]);
    }else{
        Mf0x1[x] = Mf0x1[x];
    }
}

for (int x = 0; x < 4 ;x++){
    Mf1x1[x] = 0;
}
for (int x2 = 0; x2 < 4; x2++){
    for (int x3 = 0; x3 < 4; x3++){
        int x = 0 ^ x2 ^ x3;
        if(Mf1x1[x] == 0.00 && x2 == 0){
            Mf1x1[x] = 0 + Mx2f1[x2] + Mx3f1[x3];
        }else{
            if(Mf1x1[x] - (0 + Mx2f1[x2] + Mx3f1[x3]) < 100){
                Mf1x1[x] = log(exp(Mf1x1[x] - (0 + Mx2f1[x2] + Mx3f1[x3])) + 1) + (0 + Mx2f1[x2] + Mx3f1[x3]);
            }else{
                Mf1x1[x] = Mf1x1[x];
            }
        }
    }
}

for (int x = 0; x < 4 ;x++){
    Mf1x2[x] = 0;
}
for (int x1 = 0; x1 < 4; x1++){
    for (int x3 = 0; x3 < 4; x3++){
        int x = 0 ^ x1 ^ x3;

        if(Mf1x2[x] == 0.00 && x1==0){
            Mf1x2[x] = 0 + Mx1f1[x1] + Mx3f1[x3];
        }else{
            if(Mf1x2[x] - (0 + Mx1f1[x1] + Mx3f1[x3]) < 100){
```

```
                Mf1x2[x] = log(exp(Mf1x2[x] - (0 + Mx1f1[x1] + Mx3f1[x3])) + 1) + (0 + Mx1f1[x1] + Mx3f1[x3]);
            }else{
                Mf1x2[x] = Mf1x2[x];
            }
        }
    }
}

for (int x = 0; x < 4 ;x++){
    Mf1x3[x] = 0;
}
for (int x1 = 0; x1 < 4; x1++){
    for (int x2 = 0; x2 < 4; x2++){
            int x = 0 ^ x1 ^ x2;
            if(Mf1x3[x] == 0.00 && x1 == 0){
                Mf1x3[x] = 0 + Mx1f1[x1] + Mx2f1[x2];
            }else{
                if(Mf1x3[x] - (0 + Mx1f1[x1] + Mx2f1[x2]) < 100){
                    Mf1x3[x] = log(exp(Mf1x3[x] - (0 + Mx1f1[x1] + Mx2f1[x2])) + 1) + (0 + Mx1f1[x1] + Mx2f1[x2]);
                }else{
                    Mf1x3[x] = Mf1x3[x];
                }
            }
        }
    }

for (int x = 0; x < 4 ;x++){
    Mf2x3[x] = 0;
}
for (int x4 = 0; x4 < 4; x4++){
    int x = 0 ^ x4;
    if(Mf2x3[x] == 0.00){
        Mf2x3[x] = 0 + Mx4f2[x4];
    }
    if(Mf2x3[x] - (0 + Mx4f2[x4]) < 100){
        Mf2x3[x] = log(exp(Mf2x3[x] - (0 + Mx4f2[x4])) + 1) + (0 + Mx4f2[x4]);
    }else{
        Mf2x3[x] = Mf2x3[x];
    }
}

for (int x = 0; x < 4 ;x++){
    Mf2x4[x] = 0;
}
for (int x3 = 0; x3 < 4; x3++){
    int x = 0 ^ x3;
    if(Mf2x4[x] == 0.00){
        Mf2x4[x] = 0 + Mx3f2[x3];
    }
    if(Mf2x4[x] - (0 + Mx3f2[x3]) < 100){
        Mf2x4[x] = log(exp(Mf2x4[x] - (0 + Mx3f2[x3])) + 1) + (0 + Mx3f2[x3]);
    }else{
        Mf2x4[x] = Mf2x4[x];
    }
}

/* 最尤推定 */
double max;
max = -DBL_MAX;
for (int x = 0; x < 4; x++){
    double lh =  prob[y[0]][x] + Mf0x0[x];
    if (max < lh){
    max = lh;
    xh[0] = x;
    }
}

max = -DBL_MAX;
for (int x = 0; x < 4; x++){
```

```
            double lh =  prob[y[1]][x] + Mf0x1[x] + Mf1x1[x];
            if (max < lh){
            max = lh;
            xh[1] = x;
            }
        }

        max = -DBL_MAX;
        for (int x = 0; x < 4; x++){
            double lh =  prob[y[2]][x] + Mf1x2[x];
            if (max < lh){
            max = lh;
            xh[2] = x;
            }
        }

        max = -DBL_MAX;
        for (int x = 0; x < 4; x++){
            double lh =  prob[y[3]][x] + Mf1x3[x] + Mf2x3[x];
            if (max < lh){
            max = lh;
            xh[3] = x;
            }
        }

        max = -DBL_MAX;
        for (int x = 0; x < 4; x++){
            double lh =  prob[y[4]][x] + Mf2x4[x];
            if (max < lh){
            max = lh;
            xh[4] = x;
            }
        }

        /*符号語の判定*/
        if(n > 5){
            if (1 && ((0 ^ xh[0] ^ xh[1]) == 0) && ((0 ^ xh[1] ^ xh[2] ^ xh[3]) == 0) && ((0 ^ xh[3] ^ xh[4]) == 0)){
                break;
            }
        }
    }
    return;
}


/* 復号誤り判定 */
int count(){
    int x[5];
    int xh[5];
    double a[5];
    double d[6];
    int num;
    int x2 =  genrand_real2() * ALPHABETSIZE;
    int x4 =  genrand_real2() * ALPHABETSIZE;

    encode(x2, x4, x);
    code2interval(x,a);
    transmit(a,d);
    decode(d,xh);

    for (num = 0;num < 5;num++){
        if(xh[num] != x[num]){
            break;
        }
    }
    return (num < 5);
}

//メタプログラム出力結果（ここまで）
```

```
int main(){
    unsigned long init[4] = {0x123, 0x234, 0x345, 0x456};
    init_by_array(init, 4);

    //復号誤りした数をカウント
    int error = 0;
    for(int j = 0;j < TRIAL ;j++){
        error += count();
    }
    printf("%f\n", (double)error/TRIAL);
}
```

A.5 記憶を考慮した Sum-Product 復号のメタプログラム

```
//【メタプログラム】記憶を考慮した um-Product 復号（対数）：5 列 3 行
#include <stdio.h>
#include <stdarg.h>

int row = 3;
int column = 5;
char pcm[15] = {
    1,1,0,0,0,
    0,1,1,1,0,
    0,0,0,1,1
};

/* インデント生成 */
int tprintf(int n, char *fmt, ...){
va_list arg;
while (n-- > 0){
putchar('\t');
}
va_start(arg, fmt);
int ret = vprintf(fmt, arg);
va_end(arg);
return(ret);
}



/* 符号器設計 &&  復号誤り判定設計 */
int encode(){
    FILE *fp;
    fp = fopen("test2.txt","w");

    /* 符号器設計 */
    int j = 0;
    int rank = row;
int index[5] = {0,1,2,3,4};//手動で変更する

    for (int i = 0; i < row; i++){
        //行入れ替え
        if (pcm[i*column + j] == 0){
            //0 だったら j+1,j+2.. 列の中から 1 を見つけて交換
            int l;
            for (l = 1 ; l < column; l++){
                if (pcm[i*column + l] != 0){
                    break;
                }
            }

            //見つからなかった場合
            if(l >= column){
```

```
                rank--;
                continue;
            }

            //列入れ替え
            for(int k = 0;k < row;k++){
                int tmp = pcm[k*column + l];
                pcm[k*column + l] = pcm[k*column + j];
                pcm[k*column + j] = tmp;
            }

            int tmp = index[l];
            index[l] = index[j];
            index[j] = tmp;
        }

        for(int k = 0; k < row;k++){
            if (k == i) continue;
            if (pcm[k*column + j] == 0) continue;
            for (int l = 0; l < column;l++){
                pcm[k*column + l] ^= pcm[i*column + l];
            }
        }
        j++;
    }

    //符号器出力
    fprintf(fp,"\n");
    fprintf(fp,"void encode(");

for (int j = rank; j < column ; j++){
        fprintf(fp,"int x%d, ",index[j]);
}
    fprintf(fp,"int x[]){\n");


    /* 復号誤り判定設計 */
int h;
j = 0;
for (int i = 0; i < row; i++){
if (pcm[i*column + j] == 0) continue;
        fprintf(fp,"    x[%d] = 0",index[j]);

for(int j0 = rank; j0 < column; j0++){
if (pcm[i*column + j0] == 0) continue;
            fprintf(fp," ^ x%d",index[j0]);
}
j++;
        fprintf(fp,";\n");
}

for (int i = rank; i < column; i++){
        fprintf(fp,"    x[%d] = x%d;\n", index[i], index[i]);
}

    fprintf(fp,"    return;\n");
    fprintf(fp,"}\n");
    fprintf(fp,"\n");
    fprintf(fp,"int count(){\n");
    fprintf(fp,"    int x[%d];\n", column);
    fprintf(fp,"    int xh[%d];\n", column);
    fprintf(fp,"    double a[%d];\n", column);
fprintf(fp,"    double d[%d];\n", column + 1);
    fprintf(fp,"    int num;\n");

for(int i = rank; i < column; i++){
        fprintf(fp,"    int x%d =  genrand_real2() * ALPHABETSIZE;\n", index[i]);
}
```

```
fprintf(fp,"\n");
    fprintf(fp,"    encode(");

for(int i = rank; i < column; i++){
        fprintf(fp,"x%d, ", index[i]);
}

    fprintf(fp,"x");
    fprintf(fp,");\n");
    fprintf(fp,"    code2interval(x,a);\n");
    fprintf(fp,"    transmit(a,d);\n");
fprintf(fp,"    decode(d,xh);\n");
    fprintf(fp,"\n");
    fprintf(fp,"    for (num = 0;num < %d;num++){\n", column);
    fprintf(fp,"        if(xh[num] != x[num]){\n");
    fprintf(fp,"            break;\n");
    fprintf(fp,"        }\n");
    fprintf(fp,"    }\n");
    fprintf(fp,"    return (num < %d);\n", column);
    fprintf(fp,"}\n");

    fclose(fp);
    return (0);
}

/* 符号器 → 通信路 */
int code2interval(){
    printf("double code2interval(int x[],double a[]){\n");
    tprintf(1,"double u[%d];\n", column);
    tprintf(1,"for(int k = 0; k < %d; k++){\n", column);
    tprintf(2,"a[k] = expdistinv(rvariable(x[k]), SRCRATE);\n");
    tprintf(1,"}\n");
    printf("}\n");
}

/* 通信路 */
int interval(){
    printf("void transmit(double a[],double rcvtime[]){\n");
    tprintf(1,"int i = 0;\n");
    tprintf(1,"int j = 1;\n");
    tprintf(1,"int k;\n");
    tprintf(1,"double current_time = 0.0;\n");
    tprintf(1,"int customer_num = 0;\n");
    tprintf(1,"double sendtime;\n");
    tprintf(1, "double service_end = 0.0;\n");
    tprintf(1, "sendtime = a[0];\n");
    tprintf(1, "rcvtime[0] = 0.0;\n");
    tprintf(1, "while (i < %d || customer_num > 0){\n",column);
    tprintf(2, "if (customer_num == 0 || (i < %d && sendtime < service_end)){\n",column);
    tprintf(3, "current_time = sendtime;\n");
    tprintf(3, "i++;\n");
    tprintf(3, "if (i < %d){\n",column);
    tprintf(4, "sendtime += a[i];\n");
    tprintf(3, "}\n");
    tprintf(3, "if (customer_num == 0){\n");
    tprintf(4, "service_end = current_time +   expdistinv(genrand_real2(), SERVICERATE);\n");
    tprintf(3, "}\n");
    tprintf(3, "customer_num++;\n");
    tprintf(2, "}\n");
    tprintf(2, "else {\n");
    tprintf(3, "current_time = service_end;\n");
    tprintf(3, "rcvtime[j] = service_end;\n");
    tprintf(3, "j++;\n");
    tprintf(3, "customer_num--;\n");
    tprintf(3, "if (customer_num >= 1){\n");
    tprintf(4, "service_end = current_time + expdistinv(genrand_real2(), SERVICERATE);\n");
    tprintf(3, "}\n");
    tprintf(2, "}\n");
    tprintf(1, "}\n");
```

```
    tprintf(1, "return;\n");
    printf("}\n");
}

/* 復号器設計 */
int decode(){
    FILE *fp;
    fp = fopen("test3.txt","w");

    fprintf(fp,"void decode(double dep[], int xh[]){\n");
    fprintf(fp,"    double ah[%d];\n", column + 1);
    fprintf(fp,"    ah[0] = 0;\n");
    fprintf(fp,"    for (int i = 0; i < %d; i++){\n", column);
    fprintf(fp,"        xh[i] = expdistfunc(dep[i + 1] - dep[i], SRCRATE) * ALPHABETSIZE;\n");
    fprintf(fp,"        ah[i + 1] = ah[i] + sym2interval(xh[i]);\n");
fprintf(fp,"    }\n");
    fprintf(fp,"\n");

/* ○ ⇒ □の配列準備 */
for (int i = 0; i < row; i++){
for (int j = 0; j < column; j++){
if (pcm[i*column + j] == 0) continue;
            fprintf(fp,"    double Mx%df%d[4];\n",j, i);
}
    }
    fprintf(fp,"\n");

/* □ ⇒ ○の配列準備 */
for (int i = 0; i < row; i++){
for (int j = 0; j < column; j++){
if (pcm[i*column + j] == 0) continue;
            fprintf(fp,"    double Mf%dx%d[4] = {0,0,0,0};\n", i, j);
}
}
    fprintf(fp,"\n");
    fprintf(fp,"    for (int n = 1; n < 100; n++){\n");


    /* ○ ⇒ □ */
for (int i = 0; i < row; i++){
for (int j = 0; j < column; j++){
if (pcm[i*column + j] == 0) continue;
            fprintf(fp,"        for (int x=0; x < 4;x++){\n");
            fprintf(fp,"            Mx%df%d[x] = w(dep[%d], dep[%d] ,ah[%d] + sym2interval(x))", j, i, j + 1, j, j);
for (int k = 0; k < row; k++){
if (k == i) continue;
if (pcm[k*column + j]){
                    fprintf(fp," + Mf%dx%d[x]", k, j);
}
}
            fprintf(fp,";\n");
            fprintf(fp,"        }\n\n");
}
}
    fprintf(fp,"\n\n\n");

    /* □ ⇒ ○ */
    for (int i = 0; i < row; i++){
        for (int j = 0; j < column; j++){
            if (pcm[i*column + j] == 0) continue;
            fprintf(fp,"        for (int x = 0; x < 4 ;x++){\n");
            fprintf(fp,"            Mf%dx%d[x] = 0;\n", i, j);
            fprintf(fp,"        }\n");

            int c = 0;
            for (int j0 = 0; j0 < column; j0++){
                if (j0 == j) continue;
                if (pcm[i*column + j0] == 0) continue;
                c += 1;
```

```
    for(int p = 0;p < c + 1;p++){
        fprintf(fp,"    ");
    }
    fprintf(fp,"for (int x%d = 0; x%d < 4; x%d++){\n", j0, j0, j0);
}
fprintf(fp,"                    int x = 0");

for (int j0 = 0; j0 < column; j0++){
    if (j0 == j) continue;
    if (pcm[i*column + j0] == 0) continue;
    fprintf(fp," ^ x%d",j0);
}
fprintf(fp,";\n");
fprintf(fp,"                    if(Mf%dx%d[x] == 0.00){\n",i,j);
fprintf(fp,"                        Mf%dx%d[x] = 0",i,j);

for (int j0 = 0; j0 < column; j0++){
    if (j0 == j) continue;
    if (pcm[i*column + j0] == 0) continue;
        fprintf(fp," + Mx%df%d[x%d]", j0, i, j0);
}
fprintf(fp,";\n");
fprintf(fp,"                    }\n");

for(int p = 0;p < c + 2;p++){
    fprintf(fp,"    ");
}

fprintf(fp,"if(Mf%dx%d[x] - (0",i,j);
for (int j0 = 0; j0 < column; j0++){
    if (j0 == j) continue;
    if (pcm[i*column + j0] == 0) continue;
    fprintf(fp," + Mx%df%d[x%d]", j0, i, j0);
}
fprintf(fp,") < 100){\n");


for(int p = 0;p < c + 3;p++){
    fprintf(fp,"    ");
}

fprintf(fp,"Mf%dx%d[x] = log(exp(Mf%dx%d[x] - (0", i, j, i, j);

for (int j0 = 0; j0 < column; j0++){
    if (j0 == j) continue;
    if (pcm[i*column + j0] == 0) continue;
    fprintf(fp," + Mx%df%d[x%d]", j0, i, j0);
}
fprintf(fp,")) + 1) + (0");

for (int j0 = 0; j0 < column; j0++){
    if (j0 == j) continue;
    if (pcm[i*column + j0] == 0) continue;
    fprintf(fp," + Mx%df%d[x%d]", j0, i, j0);
}
fprintf(fp,");\n");
for(int p = 0;p < c + 2;p++){
    fprintf(fp,"    ");
}

fprintf(fp,"}else{\n");

for(int p = 0;p < c + 3;p++){
    fprintf(fp,"    ");
}

fprintf(fp,"Mf%dx%d[x] = Mf%dx%d[x];\n",i,j,i,j);
```

```
            for(int p = 0;p < c + 3;p++){
                fprintf(fp,"    ");
            }

            fprintf(fp,"}\n");

            for (int j0 = 0; j0 < column; j0++){
                if (j0 == j) continue;
                if (pcm[i*column + j0] == 0) continue;

                for(int p = c + 2;p > 0;p--){
                    fprintf(fp,"    ");
                }
                fprintf(fp,"}\n");
                c-=1;
            }
        fprintf(fp,"\n");
        }

    }

    /* 最尤推定 */
    fprintf(fp,"\n\n\n");
    fprintf(fp,"        double max;\n");

for (int j = 0 ;j < column; j++){
        fprintf(fp,"        max =  -DBL_MAX;\n");
fprintf(fp,"        for (int x = 0; x < 4; x++){\n");
fprintf(fp,"            double lh = w(dep[%d], dep[%d] ,ah[%d] + sym2interval(x))", j + 1, j, j);

        for (int i = 0; i < row;i++){
if (pcm[i*column + j] == 0) continue;
            fprintf(fp," + Mf%dx%d[x]", i, j);
}
        fprintf(fp,";\n");
        fprintf(fp,"            if (max < lh){\n");
fprintf(fp,"                max = lh;\n");
        fprintf(fp,"                xh[%d] = x;\n",j,j);
        fprintf(fp,"            }\n");
fprintf(fp,"        }\n");
        fprintf(fp,"        ah[%d] = ah[%d] + sym2interval(xh[%d]);\n",j + 1, j, j);
        fprintf(fp,"\n");
}

    fprintf(fp,"        if (1");

for (int i = 0; i < row; i++){
        fprintf(fp," && ((0");

for (int j = 0;j < column; j++){
if (pcm[i*column + j] == 0) continue;
                fprintf(fp," ^ xh[%d]", j);
}

        fprintf(fp,") == 0)");
}

    fprintf(fp,"){\n");
fprintf(fp,"            break;\n");
    fprintf(fp,"        }\n");
    fprintf(fp,"    }\n");
fprintf(fp,"    return;\n");
    fprintf(fp,"}\n");
    fprintf(fp,"\n");

    fclose(fp);
return (0);
}
```

```
int main(){
    //encode();
    code2interval();
    //interval();
    //decode();
}
```

A.6 記憶を考慮した Sum-Product 復号

```
//記憶を考慮した Sum-Product 復号（対数）：5 列 3 行
#include <stdio.h>
#include <math.h>
#include <stdarg.h>
#include <float.h>
#include <stdlib.h>
#include "random.h"//乱数生成

//出力レート [symbol/s]
#define SRCRATE (1/2.71828182845904523536028747135 27)//レート
//#define SRCRATE (0.9/2.71828182845904523536028747135 27)
//#define SRCRATE (0.8/2.71828182845904523536028747135 27)
//#define SRCRATE (0.7/2.71828182845904523536028747135 27)
//#define SRCRATE (0.6/2.71828182845904523536028747135 27)
//#define SRCRATE (0.5/2.71828182845904523536028747135 27)
//#define SRCRATE (0.4/2.71828182845904523536028747135 27)
//#define SRCRATE (0.3/2.71828182845904523536028747135 27)
//#define SRCRATE (0.2/2.71828182845904523536028747135 27)
//#define SRCRATE (0.1/2.71828182845904523536028747135 27)


#define ALPHABETSIZE 4  //シンボル数
#define SERVICERATE 1.0 //サービスレート
#define TRIAL 100   //試行回数



/* 確率変数 */
double rvariable(int x){
    return ((2 * x + 1) * 0.125);
}

/* 分布関数の逆関数 */
double expdistinv(double x, double rate){
    if ((1 - x) < 0){
        printf("%d %f\n", __LINE__, 1 - x);
        exit(1);
    }
    return(-log(1 - x) / rate);
}

/* 間隔変換 */
double sym2interval(int sym){
    return(expdistinv((sym + 0.5) / ALPHABETSIZE, SRCRATE));
}

/* 指数分布の分布関数 */
double expdistfunc(double x, double rate){
    return(1 - exp(-x * rate));
}

/* 大小比較 */
double max(double a, double b) {
    if (a > b)
        return a;
```

```
    else
        return b;
}

/* w(受信時刻, 受信時刻のひとつ前, 送信時刻) */
double w(double dep, double pdep, double arr){
    double servicetime;
    if(dep < arr){
        return (-1.0e+100);//限りなく-∞に近い値を設定（桁落ちも考慮）
        //return (-10000);
    }
    servicetime = dep - max(pdep,arr);

    if (servicetime < 0){
        printf("終了");
        exit(1);
    }
    return(-servicetime);
}

/* メタプログラム出力結果（ここから） */

/* 符号器 */
void encode(int x2, int x4, int x[]){
    x[0] = 0 ^ x2 ^ x4;
    x[1] = 0 ^ x2 ^ x4;
    x[3] = 0 ^ x4;
    x[2] = x2;
    x[4] = x4;
    return;
}

/* 符号器 → 通信路 */
double code2interval(int x[],double a[]){
        double u[5];
        for(int k = 0; k < 5; k++){
                a[k] = expdistinv(rvariable(x[k]), SRCRATE);
        }
}

/* 通信路 */
void transmit(double a[],double rcvtime[]){
        int i = 0;
        int j = 1;
        int k;
        double current_time = 0.0;
        int customer_num = 0;
        double sendtime;
        double service_end = 0.0;
        sendtime = a[0];
        rcvtime[0] = 0.0;
        while (i < 5 || customer_num > 0){
                if (customer_num == 0 || (i < 5 && sendtime < service_end)){
                        current_time = sendtime;
                        i++;
                        if (i < 5){
                                sendtime += a[i];
                        }
                        if (customer_num == 0){
                                service_end = current_time +   expdistinv(genrand_real2(), SERVICERATE);
                        }
                        customer_num++;
                }
                else {
                        current_time = service_end;
                        rcvtime[j] = service_end;
                        j++;
                        customer_num--;
                        if (customer_num >= 1){
```

```
                                service_end = current_time + expdistinv(genrand_real2(), SERVICERATE);
                        }
                }
        }
        return;
}

/* 復号器 */
void decode(double dep[], int xh[]){
    double ah[6];
    ah[0] = 0;
    for (int i = 0; i < 5; i++){
        xh[i] = expdistfunc(dep[i + 1] - dep[i], SRCRATE) * ALPHABETSIZE;
        ah[i + 1] = ah[i] + sym2interval(xh[i]);
    }

    double Mx0f0[4];
    double Mx1f0[4];
    double Mx1f1[4];
    double Mx2f1[4];
    double Mx3f1[4];
    double Mx3f2[4];
    double Mx4f2[4];

    double Mf0x0[4] = {0,0,0,0};
    double Mf0x1[4] = {0,0,0,0};
    double Mf1x1[4] = {0,0,0,0};
    double Mf1x2[4] = {0,0,0,0};
    double Mf1x3[4] = {0,0,0,0};
    double Mf2x3[4] = {0,0,0,0};
    double Mf2x4[4] = {0,0,0,0};


    for (int n = 1; n < 100; n++){
        /* ○ ⇒ □ */
        for (int x=0; x < 4;x++){
            Mx0f0[x] = w(dep[1], dep[0] ,ah[0] + sym2interval(x));
        }

        for (int x=0; x < 4;x++){
            Mx1f0[x] = w(dep[2], dep[1] ,ah[1] + sym2interval(x)) + Mf1x1[x];
        }

        for (int x=0; x < 4;x++){
            Mx1f1[x] = w(dep[2], dep[1] ,ah[1] + sym2interval(x)) + Mf0x1[x];
        }

        for (int x=0; x < 4;x++){
            Mx2f1[x] = w(dep[3], dep[2] ,ah[2] + sym2interval(x));
        }

        for (int x=0; x < 4;x++){
            Mx3f1[x] = w(dep[4], dep[3] ,ah[3] + sym2interval(x)) + Mf2x3[x];
        }

        for (int x=0; x < 4;x++){
            Mx3f2[x] = w(dep[4], dep[3] ,ah[3] + sym2interval(x)) + Mf1x3[x];
        }

        for (int x=0; x < 4;x++){
            Mx4f2[x] = w(dep[5], dep[4] ,ah[4] + sym2interval(x));
        }


        /* □ ⇒ ○ */
        for (int x = 0; x < 4 ;x++){
            Mf0x0[x] = 0;
        }
        for (int x1 = 0; x1 < 4; x1++){
```

```
    int x = 0 ^ x1;
    if(Mf0x0[x] == 0.00){
        Mf0x0[x] = 0 + Mx1f0[x1];
    }else{
        if(Mf0x0[x] - (0 + Mx1f0[x1]) < 100){
            Mf0x0[x] = log(exp(Mf0x0[x] - (0 + Mx1f0[x1])) + 1) + (0 + Mx1f0[x1]);
        }else{
            Mf0x0[x] = Mf0x0[x];
        }
    }
}

for (int x = 0; x < 4 ;x++){
    Mf0x1[x] = 0;
}
for (int x0 = 0; x0 < 4; x0++){
    int x = 0 ^ x0;
    if(Mf0x1[x] == 0.00){
        Mf0x1[x] = 0 + Mx0f0[x0];
    }
    if(Mf0x1[x] - (0 + Mx0f0[x0]) < 100){
        Mf0x1[x] = log(exp(Mf0x1[x] - (0 + Mx0f0[x0])) + 1) + (0 + Mx0f0[x0]);
    }else{
        Mf0x1[x] = Mf0x1[x];
    }
}

for (int x = 0; x < 4 ;x++){
    Mf1x1[x] = 0;
}
for (int x2 = 0; x2 < 4; x2++){
    for (int x3 = 0; x3 < 4; x3++){
        int x = 0 ^ x2 ^ x3;
        if(Mf1x1[x] == 0.00 && x2 == 0){
            Mf1x1[x] = 0 + Mx2f1[x2] + Mx3f1[x3];
        }else{
            if(Mf1x1[x] - (0 + Mx2f1[x2] + Mx3f1[x3]) < 100){
                Mf1x1[x] = log(exp(Mf1x1[x] - (0 + Mx2f1[x2] + Mx3f1[x3])) + 1) + (0 + Mx2f1[x2] + Mx3f1[x3]);
            }else{
                Mf1x1[x] = Mf1x1[x];
            }
        }
    }
}


for (int x = 0; x < 4 ;x++){
    Mf1x2[x] = 0;
}
for (int x1 = 0; x1 < 4; x1++){
    for (int x3 = 0; x3 < 4; x3++){
        int x = 0 ^ x1 ^ x3;

        if(Mf1x2[x] == 0.00 && x1==0){
            Mf1x2[x] = 0 + Mx1f1[x1] + Mx3f1[x3];
        }else{
            if(Mf1x2[x] - (0 + Mx1f1[x1] + Mx3f1[x3]) < 100){
                Mf1x2[x] = log(exp(Mf1x2[x] - (0 + Mx1f1[x1] + Mx3f1[x3])) + 1) + (0 + Mx1f1[x1] + Mx3f1[x3]);
            }else{
                Mf1x2[x] = Mf1x2[x];
            }
        }
    }
}

for (int x = 0; x < 4 ;x++){
    Mf1x3[x] = 0;
}
for (int x1 = 0; x1 < 4; x1++){
```

```
    for (int x2 = 0; x2 < 4; x2++){
        int x = 0 ^ x1 ^ x2;
        if(Mf1x3[x] == 0.00 && x1 == 0){
            Mf1x3[x] = 0 + Mx1f1[x1] + Mx2f1[x2];
        }else{
            if(Mf1x3[x] - (0 + Mx1f1[x1] + Mx2f1[x2]) < 100){
                Mf1x3[x] = log(exp(Mf1x3[x] - (0 + Mx1f1[x1] + Mx2f1[x2])) + 1) + (0 + Mx1f1[x1] + Mx2f1[x2]);
            }else{
                Mf1x3[x] = Mf1x3[x];
            }
        }
    }
}

for (int x = 0; x < 4 ;x++){
    Mf2x3[x] = 0;
}
for (int x4 = 0; x4 < 4; x4++){
    int x = 0 ^ x4;
    if(Mf2x3[x] == 0.00){
        Mf2x3[x] = 0 + Mx4f2[x4];
    }
    if(Mf2x3[x] - (0 + Mx4f2[x4]) < 100){
        Mf2x3[x] = log(exp(Mf2x3[x] - (0 + Mx4f2[x4])) + 1) + (0 + Mx4f2[x4]);
    }else{
        Mf2x3[x] = Mf2x3[x];
    }
}

for (int x = 0; x < 4 ;x++){
    Mf2x4[x] = 0;
}
for (int x3 = 0; x3 < 4; x3++){
    int x = 0 ^ x3;
    if(Mf2x4[x] == 0.00){
        Mf2x4[x] = 0 + Mx3f2[x3];
    }
    if(Mf2x4[x] - (0 + Mx3f2[x3]) < 100){
        Mf2x4[x] = log(exp(Mf2x4[x] - (0 + Mx3f2[x3])) + 1) + (0 + Mx3f2[x3]);
    }else{
        Mf2x4[x] = Mf2x4[x];
    }
}



/* 最尤推定 */
//x0 を推定
double max;
max =  -DBL_MAX;
for (int x = 0; x < 4; x++){
    double lh = w(dep[1], dep[0] ,ah[0] + sym2interval(x)) + Mf0x0[x];
    if (max < lh){
        max = lh;
        xh[0] = x;
    }
}
ah[1] = ah[0] + sym2interval(xh[0]);

//x1 を推定
max =  -DBL_MAX;
for (int x = 0; x < 4; x++){
    double lh = w(dep[2], dep[1] ,ah[1] + sym2interval(x)) + Mf0x1[x] + Mf1x1[x];
    if (max < lh){
        max = lh;
        xh[1] = x;
    }
}
ah[2] = ah[1] + sym2interval(xh[1]);
```

```
        //x2 を推定
        max =  -DBL_MAX;
        for (int x = 0; x < 4; x++){
            double lh = w(dep[3], dep[2] ,ah[2] + sym2interval(x)) + Mf1x2[x];
            if (max < lh){
                max = lh;
                xh[2] = x;
            }
        }
        ah[3] = ah[2] + sym2interval(xh[2]);

        //x3 を推定
        max =  -DBL_MAX;
        for (int x = 0; x < 4; x++){
            double lh = w(dep[4], dep[3] ,ah[3] + sym2interval(x)) + Mf1x3[x] + Mf2x3[x];
            if (max < lh){
                max = lh;
                xh[3] = x;
            }
        }
        ah[4] = ah[3] + sym2interval(xh[3]);

        //x4 を推定
        max =  -DBL_MAX;
        for (int x = 0; x < 4; x++){
            double lh = w(dep[5], dep[4] ,ah[4] + sym2interval(x)) + Mf2x4[x];
            if (max < lh){
                max = lh;
                xh[4] = x;
            }
        }
        ah[5] = ah[4] + sym2interval(xh[4]);


        if (n > 5){
            if (1 && ((0 ^ xh[0] ^ xh[1]) == 0) && ((0 ^ xh[1] ^ xh[2] ^ xh[3]) == 0) && ((0 ^ xh[3] ^ xh[4]) == 0)){
                break;
            }
        }
    }
    return;
}


/* 復号誤り判定 */
int count(){
    int x[5];
    int xh[5];
    double a[5];
    double d[6];
    int num;
    int x2 =  genrand_real2() * ALPHABETSIZE;
    int x4 =  genrand_real2() * ALPHABETSIZE;

    encode(x2, x4, x);
    code2interval(x,a);
    transmit(a,d);
    decode(d,xh);

    for (num = 0; num < 5; num++){
        if(xh[num] != x[num]){
            break;
        }
    }
    return (num < 5);
}

//メタプログラム出力結果（ここまで）
```

```
int main(){
    unsigned long init[4] = {0x123, 0x234, 0x345, 0x456};
    init_by_array(init, 4);

    //復号誤りした数をカウント
    int error = 0;
    for(int j = 0;j < TRIAL ;j++){
        error += count();
    }
    printf("%f\n", (double)error/TRIAL);
}
```

A.7 シンボル MAP 復号

```
//シンボル MAP 復号：5 列 3 行
#include <stdio.h>
#include <math.h>
#include <float.h>
#include <stdlib.h>
#include "random.h"

//出力レート [symbol/s]
#define SRCRATE (1/2.71828182845904523536028747135 27)
//#define SRCRATE (0.9/2.71828182845904523536028747135 27)
//#define SRCRATE (0.8/2.71828182845904523536028747135 27)
//#define SRCRATE (0.7/2.71828182845904523536028747135 27)
//#define SRCRATE (0.6/2.71828182845904523536028747135 27)
//#define SRCRATE (0.5/2.71828182845904523536028747135 27)
//#define SRCRATE (0.4/2.71828182845904523536028747135 27)
//#define SRCRATE (0.3/2.71828182845904523536028747135 27)
//#define SRCRATE (0.2/2.71828182845904523536028747135 27)
//#define SRCRATE (0.1/2.71828182845904523536028747135 27)


#define ALPHABETSIZE 4  //シンボル数
#define SERVICERATE 1.0 //サービスレート
#define TRIAL 1000000   //試行回数

#define N 624
#define M 397
#define MATRIX_A 0x9908b0dfUL   /* constant vector a */
#define UPPER_MASK 0x80000000UL /* most significant w-r bits */
#define LOWER_MASK 0x7fffffffUL /* least significant r bits */

/* 確率変数 */
double rvariable(int x){
    return ((2 * x + 1) * 0.125);
}

/* 分布関数の逆関数 */
double expdistinv(double x, double rate){
    if ((1 - x) < 0){
        printf("%d %f\n", __LINE__, 1 - x);
        exit(1);
    }
    return(-log(1 - x) / rate);
}

/* 間隔変換 */
double sym2interval(int sym){
    return(expdistinv((sym + 0.5) / ALPHABETSIZE, SRCRATE));
}
```

```
/* 指数分布の分布関数 */
double expdistfunc(double x, double rate){
    return(1 - exp(-x * rate));
}

/* 大小比較 */
double max(double a, double b) {
    if (a > b)
        return a;
    else
        return b;
}

/* w(受信時刻, 受信時刻のひとつ前, 送信時刻) */
double w(double dep, double pdep, double arr){
    double servicetime;
    if(dep < arr){
        return (0.0);
    }
    servicetime = dep - max(pdep,arr);

    if (servicetime < 0){
        printf("終了");
        exit(1);
    }
    return(exp(-servicetime));
}


/* メタプログラム出力結果（ここから） */

/* 符号器 */
void encode(int x2, int x4, int x[]){
        x[0] = 0 ^ x2 ^ x4;
        x[1] = 0 ^ x2 ^ x4;
        x[3] = 0 ^ x4;
        x[2] = x2;
        x[4] = x4;
        return;
}


/* 符号器 → 通信路 */
double code2interval(int x[],double a[]){
        double u[5];
        for(int k = 0; k < 5; k++){
                a[k] = expdistinv(rvariable(x[k]), SRCRATE);
        }
}

/* 通信路 */
void transmit(double a[],double rcvtime[]){
        int i = 0;
        int j = 1;
        int k;
        double current_time = 0.0;//現在時刻
        int customer_num = 0;     //システム内のパケット数
        double sendtime;          //次の到着予定時刻
        double service_end = 0.0; //サービス終了時刻
        sendtime = a[0];          //到着予定時刻
        rcvtime[0] = 0.0;         //出発時刻

        //『符号語の数だけ繰り返す』 or 『通信路にパケットが残っている時』
        while (i < 5 || customer_num > 0){
            //『システム内にパケットがない』 or 『次の到着時間よりサービス終了時刻が小さい時』
            if (customer_num == 0 || (i < 5 && sendtime < service_end)){
                //パケットがシステム内に入る場合
                current_time = sendtime;
                i++;
```

```
                //次の到着時刻を更新
                if (i < 5){
                    sendtime += a[i];
                }

                //システム内にパケットがない場合
                if (customer_num == 0){
                    service_end = current_time +   expdistinv(genrand_real2(), SERVICERATE);
                }

                customer_num++;
            }
            else {
                //パケットがシステムから出ていく場合
                current_time = service_end; //現在時刻をサービス終了時刻に更新
                rcvtime[j] = service_end;   //出発時刻をサービス終了時刻に更新
                j++;
                customer_num--;

                //システム内にパケットがある場合
                if (customer_num >= 1){
                    service_end = current_time + expdistinv(genrand_real2(), SERVICERATE);
                }
            }
        }
    return;
}


/* 復号器 */
void decode(double dep[],int xh[]){

    /* シンボル MAP 復号 */
    double ah[6];

    //x0 を推定
    double max0 = -1.0;
    for (int x0 = 0; x0 < 4; x0++){
        double pra = 0;
        ah[0] = sym2interval(x0);
        for (int x1 = 0; x1 < 4; x1++){
            ah[1] = ah[0] + sym2interval(x1);
            for (int x2 = 0; x2 < 4; x2++){
                ah[2] = ah[1] + sym2interval(x2);
                for (int x3 = 0; x3 < 4; x3++){
                    ah[3] = ah[2] + sym2interval(x3);
                    for (int x4 = 0; x4 < 4; x4++){
                        ah[4] = ah[3] + sym2interval(x4);
                        if ((x0^x1) != 0 || (x1^x2^x3) != 0 || (x3^x4) != 0){
                            continue;
                        }
                        pra += w(dep[1],dep[0],ah[0])
                             * w(dep[2],dep[1],ah[1])
                             * w(dep[3],dep[2],ah[2])
                             * w(dep[4],dep[3],ah[3])
                             * w(dep[5],dep[4],ah[4]);
                    }
                }
            }
        }
        if (max0 < pra){
            max0 = pra;
            xh[0] = x0;
        }
    }


    //x1 を推定
```

```
double max1 = -1.0;
for (int x1 = 0;x1 <4 ;x1++){
    double prb = 0;
    for (int x0 = 0;x0 < 4;x0++){
        ah[0] = sym2interval(x0);
        ah[1] = ah[0] + sym2interval(x1);
        for (int x2 = 0; x2 < 4; x2++){
            ah[2] = ah[1] + sym2interval(x2);
            for (int x3 = 0;x3 < 4;x3++){
                ah[3] = ah[2] + sym2interval(x3);
                for (int x4 = 0;x4 < 4;x4++){
                    ah[4] = ah[3] + sym2interval(x4);
                    if ((x0^x1) != 0 || (x1^x2^x3) != 0 || (x3^x4) != 0){
                        continue;
                    }
                    prb += w(dep[1],dep[0],ah[0])
                         * w(dep[2],dep[1],ah[1])
                         * w(dep[3],dep[2],ah[2])
                         * w(dep[4],dep[3],ah[3])
                         * w(dep[5],dep[4],ah[4]);
                }
            }
        }
    }
    if (max1 < prb){
        max1 = prb;
        xh[1] = x1;
    }
}

//x2 を推定
double max2 = -1.0;
for (int x2 = 0; x2 < 4 ;x2++){
    double prc = 0;
    for (int x0 = 0; x0 < 4; x0++){
        ah[0] = sym2interval(x0);
        for (int x1 = 0; x1 < 4; x1++){
            ah[1] = ah[0] + sym2interval(x1);
            for (int x3 = 0; x3 < 4; x3++){
                ah[2] = ah[1] + sym2interval(x2);
                ah[3] = ah[2] + sym2interval(x3);
                for (int x4 = 0;x4 < 4;x4++){
                    ah[4] = ah[3] + sym2interval(x4);
                    if ((x0^x1) != 0 || (x1^x2^x3) != 0 || (x3^x4) != 0){
                        continue;
                    }
                    prc += w(dep[1],dep[0],ah[0])
                         * w(dep[2],dep[1],ah[1])
                         * w(dep[3],dep[2],ah[2])
                         * w(dep[4],dep[3],ah[3])
                         * w(dep[5],dep[4],ah[4]);
                }
            }
        }
    }
    if (max2 < prc){
        max2 = prc;
        xh[2] = x2;
    }
}

//x3 を推定
double max3 = -1.0;
for (int x3 = 0;x3 <4 ;x3++){
    double prd = 0;
    for (int x0 = 0;x0 < 4; x0++){
        ah[0] = sym2interval(x0);
        for (int x1 = 0; x1 < 4; x1++){
            ah[1] = ah[0] + sym2interval(x1);
```

```
                for (int x2 = 0; x2 < 4; x2++){
                    ah[2] = ah[1] + sym2interval(x2);
                    ah[3] = ah[2] + sym2interval(x3);
                    for (int x4 = 0;x4 < 4; x4++){
                        ah[4] = ah[3] + sym2interval(x4);
                        if ((x0^x1) != 0 || (x1^x2^x3) != 0 || (x3^x4) != 0){
                            continue;
                        }
                        prd += w(dep[1],dep[0],ah[0])
                             * w(dep[2],dep[1],ah[1])
                             * w(dep[3],dep[2],ah[2])
                             * w(dep[4],dep[3],ah[3])
                             * w(dep[5],dep[4],ah[4]);
                    }
                }
            }
        }
        if (max3 < prd){
            max3 = prd;
            xh[3] = x3;
        }
    }

    //x4 を推定
    double max4 = -1.0;
    for (int x4 = 0; x4 < 4 ; x4++){
        double pre = 0;
        for (int x0 = 0; x0 < 4; x0++){
            ah[0] = sym2interval(x0);
            for (int x1 = 0; x1 < 4; x1++){
                ah[1] = ah[0] + sym2interval(x1);
                for (int x2 = 0; x2 < 4; x2++){
                    ah[2] = ah[1] + sym2interval(x2);
                    for (int x3 = 0; x3 < 4; x3++){
                        ah[3] = ah[2] + sym2interval(x3);
                        ah[4] = ah[3] + sym2interval(x4);
                        if ((x0^x1) != 0 || (x1^x2^x3) != 0 || (x3^x4) != 0){
                            continue;
                        }
                        pre += w(dep[1],dep[0],ah[0])
                             * w(dep[2],dep[1],ah[1])
                             * w(dep[3],dep[2],ah[2])
                             * w(dep[4],dep[3],ah[3])
                             * w(dep[5],dep[4],ah[4]);
                    }
                }
            }
        }
        if (max4 < pre){
            max4 = pre;
            xh[4] = x4;
        }
    }
    return;
}

/* 復号誤り判定 */
int count(){
    int x[5];
    int xh[5];
    double a[5];
    double d[6];
    int num;
    int x2 =  genrand_real2() * ALPHABETSIZE;
    int x4 =  genrand_real2() * ALPHABETSIZE;

    encode(x2, x4, x);
    code2interval(x,a);
    transmit(a,d);
```

```
    decode(d,xh);

    for (num = 0; num < 5; num++){
        if(xh[num] != x[num]){
            break;
        }
    }
    return (num < 5);
}

//メタプログラム出力結果（ここまで）

int main(){
    unsigned long init[4] = {0x123, 0x234, 0x345, 0x456};
    init_by_array(init, 4);

    //復号誤りした数をカウント
    int error = 0;
    for(int j = 0; j < TRIAL; j++){
        error += count();
    }
    printf("%f\n", (double)error/TRIAL);
}
```

A.8 符号化レート出力

```
//符号化レート
#include <stdio.h>
#include <math.h>
#include <stdarg.h>
#include <float.h>
#include <stdlib.h>


//出力レート [symbol/s]
#define SRCRATE (1/2.71828182845904523536028747135 27)
//#define SRCRATE (0.9/2.71828182845904523536028747135 27)
//#define SRCRATE (0.8/2.71828182845904523536028747135 27)
//#define SRCRATE (0.7/2.71828182845904523536028747135 27)
//#define SRCRATE (0.6/2.71828182845904523536028747135 27)
//#define SRCRATE (0.5/2.71828182845904523536028747135 27)
//#define SRCRATE (0.4/2.71828182845904523536028747135 27)
//#define SRCRATE (0.3/2.71828182845904523536028747135 27)
//#define SRCRATE (0.2/2.71828182845904523536028747135 27)
//#define SRCRATE (0.1/2.71828182845904523536028747135 27)


//シンボル数
#define ALPHABETSIZE 4

//符号語長 [symbol]
//int length = 900;
//int length = 20;
int length = 5;

//乱数の数（列 - 行）
//int random = 360;//540 行の時
//int random = 12;//8 列の時
int random =2;//3 行の時


/* 分布関数の逆関数 */
double expdistinv(double x, double rate){
    if ((1-x) < 0){
```

```
        printf("%d %f\n", __LINE__, 1 - x);
        exit(1);
    }
    return(-log(1 - x) / rate);
}

/* シンボルから間隔に変換 */
double sym2interval(int sym){
    return(expdistinv((sym + 0.5) / ALPHABETSIZE, SRCRATE));
}

int main(){
    //各シンボルを間隔になおす
    double inter;
    for(int i = 0;i < 4;i++){
        inter += sym2interval(i);
    }
    //0~3 のシンボルそれぞれにかかる時間の平均 [s/symbol]
    double ave = inter / ALPHABETSIZE;

    //1 つの符号語を送信するのにかかる時間 [s]
    double time = ave * length;

    //符号語 1 つの情報量 [bit]
    int info = random * 2;

    //符号化レート [bit/s]
    double code_rate = info / time;
    printf("%f\n",code_rate);
}
```

A.9 通信路行列出力

```
//通信路行列 (頻度)
#include <stdio.h>
#include <math.h>
#include <stdarg.h>
#include <float.h>
#include <stdlib.h>
#include "random.h"

/* 出力レート */
#define SRCRATE (1/2.71828182845904523536028747135 27)
//#define SRCRATE (0.9/2.71828182845904523536028747135 27)
//#define SRCRATE (0.8/2.71828182845904523536028747135 27)
//#define SRCRATE (0.7/2.71828182845904523536028747135 27)
//#define SRCRATE (0.6/2.71828182845904523536028747135 27)
//#define SRCRATE (0.5/2.71828182845904523536028747135 27)
//#define SRCRATE (0.4/2.71828182845904523536028747135 27)
//#define SRCRATE (0.3/2.71828182845904523536028747135 27)
//#define SRCRATE (0.2/2.71828182845904523536028747135 27)
//#define SRCRATE (0.1/2.71828182845904523536028747135 27)


#define ALPHABETSIZE 4 //シンボル数
#define SERVICERATE 1  //サービスレート
#define TRIAL 900      //個数

/* 確率変数 */
double rvariable(int x){
    return ((2 * x + 1) * 0.125);//1/8
}

/* 分布関数の逆関数 */
```

```
double expdistinv(double x, double rate){
    if ((1-x) < 0){
        printf("%d %f\n", __LINE__, 1 - x);
        exit(1);
    }
    return(-log(1 - x) / rate);
}

/* 間隔に変換 */
double sym2interval(int sym){
    return(expdistinv((sym + 0.5) / ALPHABETSIZE, SRCRATE));
}

/* 通信路 */
void transmit(double a[],double rcvtime[]){
        int i = 0;
        int j = 1;
        int k;
        double current_time = 0.0;
        int customer_num = 0;
        double sendtime;
        double service_end = 0.0;
        sendtime = a[0];
        rcvtime[0] = 0.0;
        while (i < TRIAL || customer_num > 0){
                if (customer_num == 0 || (i < TRIAL && sendtime < service_end)){
                        current_time = sendtime;
                        i++;
                        if (i < TRIAL){
                                sendtime += a[i];
                        }
                        if (customer_num == 0){
                                service_end = current_time +   expdistinv(genrand_real2(), SERVICERATE);
                        }
                        customer_num++;
                }
                else {
                        current_time = service_end;
                        rcvtime[j] = service_end;
                        j++;
                        customer_num--;
                        if (customer_num >= 1){
                                service_end = current_time + expdistinv(genrand_real2(), SERVICERATE);
                        }
                }
        }
        return;
}

int main(void)
{
    unsigned long init[4] = {0x234, 0x456, 0x345,0x123}, length = 4;
    init_by_array(init, length);

    char x[TRIAL];       //通信路前
    double a[TRIAL];     //間隔
    double d[TRIAL + 1]; //通信路のあとの時刻

    char xh[TRIAL];
    int count[ALPHABETSIZE][ALPHABETSIZE] = {0,0,0,0,
                                             0,0,0,0,
                                             0,0,0,0,
                                             0,0,0,0};

    for (int j = 0; j < 100000; j++){
        for (int i = 0; i < TRIAL; i++) {
            int rand = (double)genrand_real2() * ALPHABETSIZE;

            x[i] = rand;
```

```
            a[i] = sym2interval(x[i]);

            transmit(a,d);

            if ((d[i + 1] - d[i]) < -(log(0.75)/SRCRATE)){
                xh[i] = 0;
            }else if((d[i + 1] - d[i]) < -(log(0.50)/SRCRATE)){
                xh[i] = 1;
            }else if((d[i + 1] - d[i]) < -(log(0.25)/SRCRATE)){
                xh[i] = 2;
            }else{
                xh[i] = 3;
            }

            /* 通信路行列生成（頻度）*/
            for(int k = 0; k < ALPHABETSIZE; k++){
                for(int l = 0; l < ALPHABETSIZE; l++){
                    if(x[i] == k && xh[i] == l){
                        count[k][l] += 1;
                    }
                }
            }
        }
    }


    for(int k = 0; k < ALPHABETSIZE; k++){
        for(int l = 0; l < ALPHABETSIZE; l++){
            printf("%d ",count[k][l]);
        }
        printf("\n");
    }
    return 0;
}
```