解説

完璧にサンプリングしよう！ —第三話　終わりある未来—

来嶋　秀治，松井　知己

1. 消せない記憶

前回は，単調CFTPアルゴリズムを紹介した．これにより，高速なパーフェクトサンプリングの設計が可能となった．しかし，いざ（単調）CFTPアルゴリズムを実装しようとすると，思わぬ問題点が存在する．それは乱数の記憶領域である．

CFTPアルゴリズムではcoalesceを確認することによって，パーフェクトサンプリングを実現する．coalesceが確認できなかった場合には，更に過去に遡ってシミュレーションを行わなければならないが，このとき，生成された乱数を記憶しておかなければならない．例えば，単調CFTPアルゴリズムならば，計算時間に比例する記憶領域が要求されるのである．

Wilsonはこの点について改良したRead Onceアルゴリズムを提案している[4]．今回はそのRead Onceアルゴリズムを説明しよう．

2. 準備

有限の状態空間Ωを持ち，更新関数$\phi:\Omega\times[0,1)\to\Omega$で推移を定義されたエルゴード的なマルコフ連鎖$\mathcal{M}$を考える．

CFTPアルゴリズムの重要なアイデアは，「coalesceを確認する」ことである．これまでは時刻$t<0$から時刻0の間の推移だけを考えていたが，少し一般化して，時刻t_1から時刻$t_2$$(t_1<t_2)$の間の推移を考える．時刻$t_1$から時刻$t_2$までの推移において，（全状態からの）colaesceとは，（乱）数列$\boldsymbol{\lambda}\in[0,1)^{t_2-t_1}$に対して，$\exists y\in\Omega$, $\forall x\in\Omega$, $\Phi_{t_1}^{t_2}(x,\boldsymbol{\lambda})$が成り立っている事と定義する．次に，マルコフ連鎖の推移を定義している更新関数は，coalesceする確率が正（非零）であると仮定する[1]．

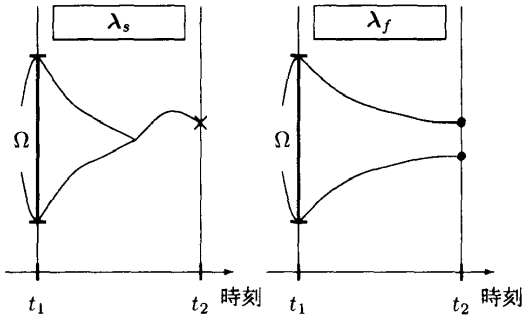

図1　時刻t_1から時刻t_2までの推移の模式図

図1はt_1からt_2の間の推移を模式的に表したものである．図1の左側は，全状態について数列$\boldsymbol{\lambda}_s$を用いて時刻t_1から時刻t_2まで推移を行い，時刻t_2でcoalesceしている様子を表している．右側は，数列$\boldsymbol{\lambda}_f$を用いると時刻t_2ではcoalesceしていない[2]様子である．図1は単調マルコフ連鎖の最大元と最小元からの推移を表していると思って見るとイメージしやすい．ただし次の議論では，計算量の議論の一部を除いて，更新関数の単調性[3]の仮定は必要としない．

Read Onceアルゴリズムを紹介する前に二つの準備を行う．

2.1 a遡るCFTP

まず，特殊なCFTPアルゴリズムを導入する．

アルゴリズム1　（a遡るCFTPアルゴリズム）

Step 0：正整数$a>0$を決める．

Step 1：シミュレーションの開始時刻を$T:=0$とする．空列$\boldsymbol{\lambda}=()$を用意する．反復回数$i:=1$とする．

Step 2：シミュレーションの開始時刻を$T:=T-a$とする．

Step 3：乱数列$\hat{\boldsymbol{\lambda}}^{-i}=(\hat{\lambda}[T],\ldots,\hat{\lambda}[T+a-1])$を生成

きじま　しゅうじ，まつい　ともみ
東京大学 大学院情報理工学系研究科
〒113-8656 文京区本郷7-3-1

[1] 第一話の節4.1参照．

[2] $\exists x_1, x_2\in\Omega, \Phi_{t_1}^{t_2}(x_1,\boldsymbol{\lambda}_f)\neq\Phi_{t_1}^{t_2}(x_2,\boldsymbol{\lambda}_f)$.

[3] 第二話参照．単調性を仮定すると，「全状態からのcoalesce」⇔「最大元と最小元からのcoalesce」が成り立つ．

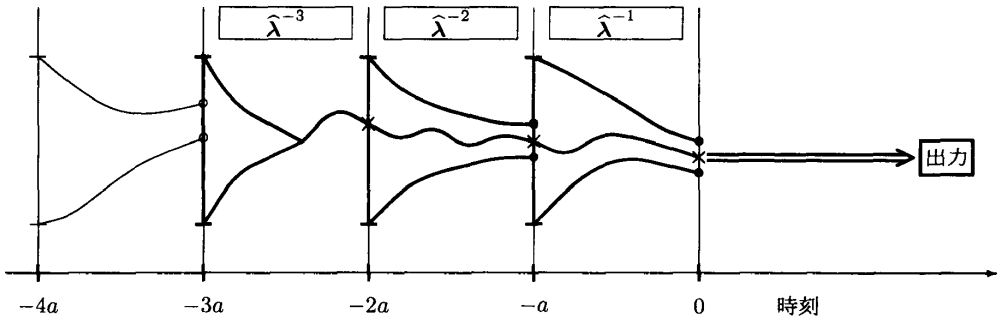

図2　a 遡る CFTP アルゴリズムの概念図

し，数列 $\hat{\boldsymbol{\lambda}}$ の先頭に挿入する．すなわち，$\hat{\boldsymbol{\lambda}}:=(\hat{\boldsymbol{\lambda}}^{-i};\ \hat{\boldsymbol{\lambda}}^{-i+1};\ \cdots;\ \hat{\boldsymbol{\lambda}}^{-1})=(\hat{\lambda}[T],\ \hat{\lambda}[T+1],\ ...,\ \hat{\lambda}[-1])$ とする．

Step 4：Ω のすべての状態について，共通の乱数列 $\hat{\boldsymbol{\lambda}}^{-i}$ を用いて時刻 T から時刻 $T+a$ までマルコフ連鎖を推移させる．このとき，

(a)　もし，時刻 $T+a$ で coalesce していれば（すなわち，$\exists z\in\Omega,\ \forall x\in\Omega,\ z=\Phi_T^{T+a}(x,\hat{\boldsymbol{\lambda}}^{-i})$），時刻 0 の状態 $y=\Phi_T^0(x,\hat{\boldsymbol{\lambda}})$ を出力し[4]，停止する．

(b)　そうでなければ，反復回数を $i:=i+1$ として Step 2 に戻る．

図2はアルゴリズム1が3反復（$i=3$）で終了した様子を表している．

アルゴリズム1は，特に二つの点において，これまでに紹介した CFTP アルゴリズムとは異なるように見える．一つは「ステップバックする時間が a」である点，もう一つは，「第 i 反復での終了条件が時刻 0 における coalesce ではなく，時刻 $-ia+a$ での coalesce」という点である．しかし，アルゴリズム1はまぎれもなく CFTP アルゴリズムの一種である．そのことを説明しよう．

一つ目の点について，CFTP アルゴリズムでは毎回 coalesce を確認する必要はなく，したがって，ステップバックする時間はどのように決めてもよい[5]．二つ目の点は少しだけトリッキーである．時刻 $-ia$ からのシミュレーションを考えた時，時刻 $-ia+a$ で coalesce していれば，時刻 0 でも当然 coalesce している．CFTP アルゴリズムにおいて，coalesce を確認するためのシミュレーション開始時刻はアルゴリズムの出力に影響を与えない[6]ので，アルゴリズム1の出力は無限の推移の実現値になっている．

アルゴリズム1において正整数 a の決め方は重要である．アルゴリズム1の計算時間について考えると，時間 a で coalesce する確率を p とすると，アルゴリズムの期待計算時間は a/p である．すなわち，coalesce するような乱数列が存在しない[7]ほど a が小さいと，アルゴリズムは停止しない．また，a を coalescence 時間の期待値 $\mathrm{E}[T_*]$ 程度にすると p は定数となる[8]ので，例えば単調 CFTP ならば期待計算時間は $\mathrm{O}(\mathrm{E}[T_*])$ となり，標準単調 CFTP と同程度の計算時間となる．

2.2　前向き coupling の coalescence 時間

ここで，前向き coupling の coalescence 時間を議論しておく．前向き coupling の coalescence 時間とは，全状態について時刻 0 からマルコフ連鎖を推移させて，初めて coalesce する時刻 T^* のことである[9]．時刻 T^* での状態は，定常分布に従っているとは限らないことに，改めて注意されたい[10]．

前向き coupling の coalescence 時間 T^* は確率変数である．一方，CFTP アルゴリズムの coalescence 時間 T_* も確率変数である[11]．そして，この二つの確率変数は同一の分布に従う．もう少し正確に言うと，

補題1　確率変数 T^* と T_* に対して次が成り立つ．

$$\forall t>0,\ \Pr(T_*>t)=\Pr(T^*>0).$$

証明：CFTP アルゴリズムで，coalescence 時間が t 以下となるような数列 $\boldsymbol{\lambda}\in[0,1)^t$ 全体の集合を $U[t]$

[4] coalesce しているので，y は x に依存しない．
[5] 第一話の節 4.3 参照．
[6] 第一話の節 4.3 参照．
[7] つまり $p=0$.
[8] 来：submultiplicativity から導けますね．松：劣乗法性ね．
[9] $\exists z\in\Omega,\ \forall x\in\Omega,\ z=\Phi_0^{T^*}(x,\boldsymbol{\lambda})$，かつ $\exists x_1, x_2\in\Omega, \Phi_0^{T^*-1}(x_1,\boldsymbol{\lambda})\neq\Phi_0^{T^*-1}(x_2,\boldsymbol{\lambda})$.
[10] 第一話の節 4.2 参照．
[11] 第一話の節 3.1 参照．

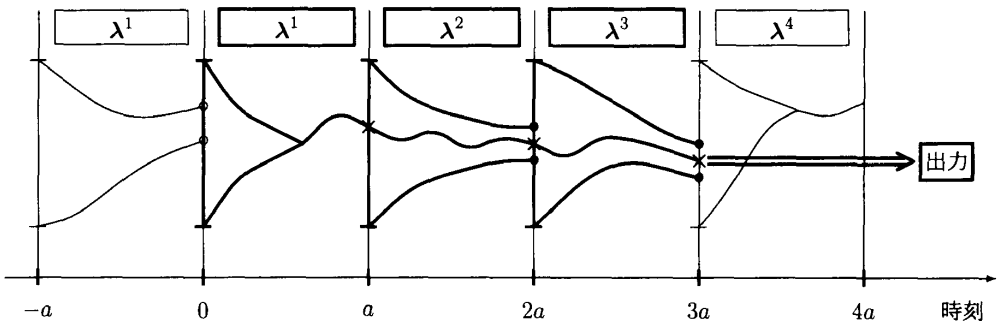

図3　Read Once アルゴリズムの概念図

とする．すなわち，任意の数列 $\boldsymbol{\lambda}\in U[t]$ は $\exists y\in\Omega$, $\forall x\in\Omega, y=\Phi_{-t}^{0}(x,\boldsymbol{\lambda})$ を満たす．このとき，任意の $\boldsymbol{\lambda}\in U[t]$ は $\exists y\in\Omega, \forall x\in\Omega, y=\Phi_{0}^{t}(x,\boldsymbol{\lambda})$ も満たす．また逆も成り立つ．確率変数 Λ は $[0,1)^t$ 上の一様分布に従うとすると，

$$\Pr(T^*\le t)=\Pr(\Lambda\in U[t])=\Pr(T_*\le t). \qquad \square$$

3. Read Once アルゴリズム

3.1 アルゴリズムと定理

以上の準備をふまえて，Read Once アルゴリズムを紹介しよう．

アルゴリズム 2　(Read Once アルゴリズム)

Step 0：正整数 $a>0$ を与える．

Step 1：乱数列 $\boldsymbol{\lambda}^1=(\lambda[0],\lambda[1],\ldots,\lambda[a-1])$ を生成する[12]．Ω のすべての状態について，共通の乱数列 $\boldsymbol{\lambda}^1$ を用いて，時刻 0 から時刻 a までマルコフ連鎖を推移させる．このとき，

(a)　もし時刻 0 と時刻 a の間で coalesce していれば (すなわち，$\exists z\in\Omega, \forall x\in\Omega, z=\Phi_0^a(x,\boldsymbol{\lambda}^1)$)，変数 y に時刻 a の状態を代入する．すなわち $y:=\Phi_0^a(x,\boldsymbol{\lambda}^1)$ とする．反復回数を $i:=1$ とする．

(b)　そうでなければ，Step 1 へ戻る．

Step 2：乱数列 $\boldsymbol{\lambda}^{i+1}=(\lambda[ia],\lambda[ia+1],\ldots,\lambda[(i+1)a-1])$ を生成する[13]．Ω のすべての状態について，共通の乱数列 $\boldsymbol{\lambda}^{i+1}$ を用いて，時刻 ia から時刻 $(i+1)a$ まで推移させる．特に $y'=\Phi_{ia}^{(i+1)a}(y,\boldsymbol{\lambda}^{i+1})$ とする．

(a)　時刻 ia と時刻 $(i+1)a$ の間で coalesce していれば ($\exists z\in\Omega, \forall x\in\Omega, z=\Phi_{ia}^{(i+1)a}(x,\boldsymbol{\lambda}^{i+1})$)，時刻 ia の状態 y を出力し，停止する．

(b)　そうでなければ，$y:=y'$ と更新する．反復回数を $i:=i+1$ とし，Step 2 に戻る．

図 3 は Read Once アルゴリズム (アルゴリズム 2) が 3 反復 ($i=3$) で終了した様子を表している．

定理 2　エルゴード的なマルコフ連鎖に対して，ある a が存在して，アルゴリズム 2 は確率 1 で有限停止し，アルゴリズム 2 の返す値は定常分布に厳密に従う．

3.2 Read Once アルゴリズムのアイデア

ずばり Read Once アルゴリズム (アルゴリズム 2) のアイデアは，a 遡る CFTP アルゴリズム (アルゴリズム 1) の模倣である．このことを説明しよう．

Read Once アルゴリズムが r 回の反復で終了した場合を考える．すなわち，アルゴリズム 2 が停止した時の反復回数 $i=r$ だったとする．このとき，アルゴリズム 2 は $y^*\in\Omega$ を出力し，アルゴリズム 2 で y^* を決定するのに用いた乱数ベクトルは $\boldsymbol{\lambda}=(\boldsymbol{\lambda}^1;\boldsymbol{\lambda}^2;\cdots;\boldsymbol{\lambda}^r)$ であったとする ($\boldsymbol{\lambda}^{r+1}$ は y^* の決定には無関係！)．

出力された状態 y^* について考える．アルゴリズム 2 の中で，時刻 0 から時刻 a の間の推移は coalesce している (すなわち，$\exists z\in\Omega, \forall x\in\Omega, z=\Phi_0^a(x,\boldsymbol{\lambda}^1)$) ので，時刻 0 から時刻 ra の間の推移も当然 coalesce していて，$y^*=\Phi_0^{ra}(x,\boldsymbol{\lambda})\ (\forall x\in\Omega)$ である．

用いられた乱数列 $\boldsymbol{\lambda}$ は，a 遡る CFTP アルゴリズム (アルゴリズム 1) でも，反復回数が r 回で終了した場合に，そしてその場合だけ現れ，同一の状態 y を出力する．このことは Read Once シミュレーションの時刻を $-ra$ ほどずらして見ると分かりやすい (図 3 および図 2 参照)．

まず，$\hat{\boldsymbol{\lambda}}^{-i}=\boldsymbol{\lambda}^{-i+r+1}\ (i\in\{1,\ldots,r\})$ とすると，$\hat{\boldsymbol{\lambda}}=(\hat{\boldsymbol{\lambda}}^{-r};\hat{\boldsymbol{\lambda}}^{-r+1};\cdots;\hat{\boldsymbol{\lambda}}^{-1})=\boldsymbol{\lambda}$ である．このとき，$\forall i\in\{1,\ldots,r-1\}$ について，$\hat{\boldsymbol{\lambda}}^{-i}$ を用いて時刻 $-ia$ から時刻 $-ia+a$ まで推移を行っても coalesce せず[14]，$i=r$ については $\hat{\boldsymbol{\lambda}}^{-r}$ を用いて時刻 $-ra$ から時刻 $-ra$

[12] 便宜上 $\boldsymbol{\lambda}^1$ と記述するが，この数列を記憶する必要はない．

[13] 脚注 12 と同様，数列 $\boldsymbol{\lambda}^{i+1}$ を記憶する必要はない．

$+a$まで推移を行うとcoalescenceする[15]．したがって，$\hat{\boldsymbol{\lambda}}=\boldsymbol{\lambda}$はアルゴリズム1で，反復回数が$r$回で終了した場合に，そしてその場合だけ出現する．また，そのときアルゴリズム1が出力する値は$y^*=\Phi_{-ra}^{0}(x,\hat{\boldsymbol{\lambda}})$ $(\forall x\in\Omega)$となっている．

最後に数列の出現確率を考える．集合$U[r]$をアルゴリズム2がr回の反復で終了するような数列$\boldsymbol{\lambda}=(\boldsymbol{\lambda}^1;\cdots;\boldsymbol{\lambda}^r)$全体の集合とし，集合$\hat{U}[r]$をアルゴリズム1が$r$回の反復で終了するような数列$\hat{\boldsymbol{\lambda}}=(\hat{\boldsymbol{\lambda}}^{-r};\cdots;\hat{\boldsymbol{\lambda}}^{-1})$全体の集合としよう．これまでの議論から，$U[r]=\hat{U}[r]$である．さらに，アルゴリズム2が$r$ステップで終了する確率は$\Pr(\boldsymbol{\lambda}\in U[r])=p(1-p)^{1-r}$である．一方，アルゴリズム1が$r$ステップで終了する確率は$\Pr(\hat{\boldsymbol{\lambda}}\in\hat{U}[r])=p(1-p)^{1-r}$である[16]．すなわち，Read Onceアルゴリズムで得られる状態$y\in\Omega$の出現確率はa遡るアルゴリズムで$y\in\Omega$が得られる確率に等しく，したがって，アルゴリズム2はパーフェクトサンプリングを実現する．

3.3 計算時間

以上の議論で，Read Onceアルゴリズム（アルゴリズム2）がa遡るアルゴリズム（アルゴリズム1）を模倣したものであることが分かった．アルゴリズムからも分かるとおり，Read Onceアルゴリズムでは乱数を記憶する必要がない．

Read Onceアルゴリズムの期待計算時間も，a遡るアルゴリズムと同様であることが分かる．ただ，気をつけなければならないのは，Read Onceアルゴリズムの Step 1で，coalescenceした状態を得るための前処理が行われていることから，正確には，Read Onceアルゴリズムで一つのサンプルを得るのに必要な期待計算時間はa遡るCFTPアルゴリズムの2倍程度であることが分かる．

さらに，Read Onceの良いところは，アルゴリズムの出力する状態yと，アルゴリズム2の終了判定のために行った推移のcoalescenceした状態（Step 2(a)で得られた状態z）は互いに独立，となる点である．したがって，サンプルが複数個必要な時に，アルゴリズム2を連続して実行することができる．具体的にはサンプルがN個必要な場合にはアルゴリズム2の Step 2(a)を次のように置き換えれば良い．

Step 2: (a)′ もし時刻$(i+1)a$でcoalescenceしていればyを出力する．もしN個のサンプルが出力されていたら停止する．そうでなければ，$y:=\Phi_{ia}^{(i+1)a}(x,\boldsymbol{\lambda}^{i+1})$とし，$i:=1$と更新して Step 2に戻る．

このように置き換えられたRead OnceアルゴリズムでN個のサンプルを生成するのに必要な期待計算時間は$(N+1)a/p$となるので，a遡るCFTPアルゴリズムとほぼ同じ計算時間で，乱数を記憶することなくサンプルが得られることが分かる．

Read Onceアルゴリズムの計算時間はaの決め方に大きく依存する．特にaが小さいと，アルゴリズムは全く止まらなくなってしまう[17]．しかし，aをどのように決めたらよいものだろうか．次の節ではパラメータaを全く与えないでRead Onceアルゴリズムを実現する，驚愕のTwin Runアルゴリズムを紹介しよう．

3.4 Twin Runアルゴリズム

Twin Runアルゴリズムのアイデアは，サンプリングとは別の乱数列を生成してステップバックする時間を決めようというものである．まず準備として，アルゴリズム中で呼び出す二つの手続き，*-successfulと*-failingを導入する．

手続き A（*-successful）

Step 1：シミュレーションの終了時刻Tを$T:=1$に設定する．

Step 2：二つの乱数$\lambda_1[T-1]$, $\lambda_2[T-1]$を生成する．便宜上，$\boldsymbol{\lambda}_1=(\lambda_1[0],\lambda_1[1],\ldots,\lambda_1[T-1])$, $\boldsymbol{\lambda}_2=(\lambda_2[0],\lambda_2[1],\ldots,\lambda_2[T-1])$と表す[18]．数列$\boldsymbol{\lambda}_1$と$\boldsymbol{\lambda}_2$を用いて，それぞれ推移を実行する．このとき，

(a) 数列$\boldsymbol{\lambda}_1$および$\boldsymbol{\lambda}_2$について，ともに時刻Tでcoalescenceしている[19]場合，先にcoalescenceしていた乱数列の添え字を$j\in\{1,2\}$とする[20]．同時の場合は1/2の確率でjを1または2とする．数列$\boldsymbol{\lambda}_j$に対する時刻Tの状態$z_j=\Phi_0^T(x,\boldsymbol{\lambda}_j)$を返して終了する．

(b) それ以外の場合，$T:=T+1$とし Step 2に戻る．

---

[14] $\exists x_1,x_2\in\Omega,\ \Phi_{-ia}^{-ia+a}(x_1,\hat{\boldsymbol{\lambda}}^{-i})\neq\Phi_{-ia}^{-ia+a}(x_2,\hat{\boldsymbol{\lambda}}^{-i})$.

[15] $\exists z\in\Omega,\ \forall x\in\Omega,\ z=\Phi_{-ra}^{-ra+a}(x,\hat{\boldsymbol{\lambda}}^{-r})$.

[16] 来：一種のPASTAですね（文献[8]参照）．松：美味しい性質だな．

[17] 節2.1参照．

[18] 例えば単調マルコフ連鎖ならば，この乱数列を保存せずにアルゴリズムを実行することができる．

[19] $\exists z_1,z_2\in\Omega,\ \forall x\in\Omega,\ z_1=\Phi_0^T(x,\boldsymbol{\lambda}_1),\ z_2=\Phi_0^T(x,\boldsymbol{\lambda}_2)$.

[20] 例えば，$\exists z\in\Omega,\ \forall x\in\Omega,\ z=\Phi_0^{T-1}(x,\boldsymbol{\lambda}_1)$なら$j=1$.

手続き B (*-failing)

Step 0：入力された状態を $y\in\Omega$ とする．

Step 1：シミュレーションの終了時刻 T を $T:=1$ に設定する．

Step 2：二つの乱数 $\lambda_1[T-1]$, $\lambda_2[T-1]$ を生成する．便宜上，$\boldsymbol{\lambda}_1=(\lambda_1[0], \lambda_1[1], ..., \lambda_1[T-1])$, $\boldsymbol{\lambda}_2=(\lambda_2[0], \lambda_2[1], ..., \lambda_2[T-1])$ と表す．数列 $\boldsymbol{\lambda}_1$ と $\boldsymbol{\lambda}_2$ を用いて，それぞれ推移を実行する．特に $y_1:=\Phi_0^T(y, \boldsymbol{\lambda}_1)$, $y_2:=\Phi_0^T(y, \boldsymbol{\lambda}_2)$ とする．このとき，

(a) 数列 $\boldsymbol{\lambda}_1$ と $\boldsymbol{\lambda}_2$ について，いずれか一方でも時刻 T で coalesce している場合，coalesce していない乱数列の添え字を $j\in\{1, 2\}$ とする[21]．同時の場合は 1/2 の確率で j を 1 または 2 とする．状態 $y_j\in\Omega$ を返して終了する．

(b) それ以外の場合，$T:=T+1$ とし Step 2 に戻る．

以上の準備の下，Twin Run アルゴリズムを記述する．

アルゴリズム 3 (Twin Run アルゴリズム)

Step 1：手続き A (*-successful) を実行し，出力された値を x とする．1/2 の確率で次の(a), (b)のいずれかを実行する．

(a) $x\in\Omega$ を出力しアルゴリズムを終了する．

(b) Step 2 に進む．

Step 2：入力 $x\in\Omega$ を与えて手続き B (*-failing) を実行し，変数 $x\in\Omega$ を出力された値に更新する．1/2 の確率で次の(a), (b)のいずれかを実行する．

(a) $x\in\Omega$ を出力しアルゴリズムを終了する．

(b) Step 2 にもどる．

Twin Run アルゴリズム (アルゴリズム 3) は，次の CFTP アルゴリズムを模倣する．

アルゴリズム 4 (Twin Run CFTP アルゴリズム)

Step 1：シミュレーションの開始時刻を $T:=0$ とする．空列 $\boldsymbol{\lambda}=()$ を用意し，反復回数を $i:=1$ にする．

Step 2：前向き coupling を実行し，その coalescence 時間を $T^{-i}>0$ とする．シミュレーションの開始時刻を $T:=T-T^{-i}$ とする．

Step 3：乱数列 $\boldsymbol{\lambda}^{-i}=(\lambda[T], ..., \lambda[T+T^{-i}-1])$ を生成し，数列 $\boldsymbol{\lambda}$ の先頭に挿入する．すなわち，$\boldsymbol{\lambda}:=(\boldsymbol{\lambda}^{-i}; \boldsymbol{\lambda}^{-i+1}; \cdots; \boldsymbol{\lambda}^{-1})=(\lambda[T], \lambda[T+1], ..., \lambda[-1])$ とする．

Step 4：Ω のすべての状態について，共通の乱数列 $\boldsymbol{\lambda}^{-i}$ を用いて時刻 T から時刻 $T+T^{-i}$ までマルコフ連鎖を推移させる．このとき，

(a) もし，時刻 $T+T^{-i}$ で coalesce していれば[22] (すなわち，$\exists z\in\Omega, \forall x\in\Omega, z=\Phi_T^{T+T^{-i}}(x, \boldsymbol{\lambda}^{-i})$)，時刻 0 の状態 $y=\Phi_T^0(x, \boldsymbol{\lambda})$ を出力し，停止する．

(b) そうでなければ，反復回数を $i:=i+1$ として Step 2 に戻る．

アルゴリズム 3 と 4 が本質的に等しい事の確認は，Read Once アルゴリズムの場合と同様なので読者に任せる事とし，次ではアルゴリズム 4 の基本的なアイデアを簡単に記す．

図 4 の上半分は，Twin Run アルゴリズムが 4 反復 ($i=4$) で終了した様子を，下半分はアルゴリズム 4 が 4 反復 ($i=4$) で終了した様子を表している．図 4 の Twin Run CFTP (下半分) の上段は，アルゴリズム 4 で出力する状態を決定する推移を示しており，これは本質的に Twin Run (上半分) の上段と同じものである．図 4 最下段の時間軸が右から左へ向かっているのは，値 T^{-i} を決定するのに，(別の乱数列を用いて) 前向き coupling を行っている事を表している．

実はアルゴリズム 4 は，a 遡るアルゴリズム (アルゴリズム 1) では固定されていた遡る時間 (a) を，反復ごとに Step 2 で決定するように変更したものである (ぜひ，アルゴリズム 1 と 4 を直接比較していただきたい)．アルゴリズム 4 は本質的に 2 本の乱数列を用いているが，出力を決定するのに用いているのはそのうち 1 本 (上段) だけであり，2 本目の乱数列 (下段) は遡る幅を決定する時計の役割にしか用いていない．

さらに，確率変数 T^{-i} の分布関数は前向き coupling の coalescence 時間 T^* の分布関数に等しく，これは，1 本目の乱数列 (上段) を使った CFTP の coalescence 時間 T_* の分布関数にも等しい．したがってアルゴリズム 4 の各反復において，(Step 4 (a) で) アルゴリズムが終了する確率はいつも 1/2 である．

すなわちアルゴリズム 4 は，coalescence 時間 T_* を知ることなく，アルゴリズム 1 を毎反復 1/2 の確率で終了させる驚くべき仕掛けとなっている．このこと

[21] 例えば，$\exists x_1, x_2\in\Omega, \Phi_0^T(x_1, \boldsymbol{\lambda}_2)\neq\Phi_0^T(x_2, \boldsymbol{\lambda}_2)$ なら $j=2$.

[22] 正確には，coalescence 時間がちょうど T^{-i} の場合は確率 1/2 で(b)に進む．

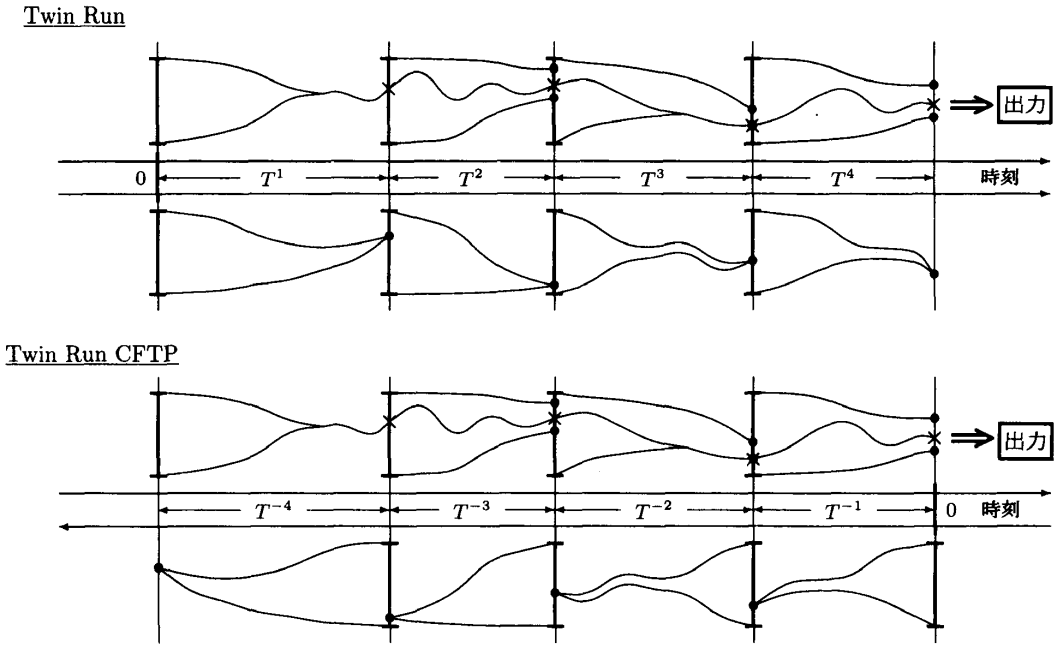

図4　Twin Run アルゴリズムの概念図

から明らかなように，アルゴリズム4の反復回数の期待値はたったの2である．

アルゴリズム3と4が本質的に等しいことから，次の定理が成り立つ．

定理3　エルゴード的な有限マルコフ連鎖に対して，Twin Run アルゴリズム（アルゴリズム3）が確率1で有限停止し，アルゴリズムの返す値は定常分布に厳密に従う．

Twin Run アルゴリズムの驚愕すべき点は，パラメータ a の設定を不要とする巧妙なアイデアに加えて，手続きA，Bを持ち込むことで，記憶領域を殆ど必要としないという成果を同時に達成している点であろう[23]．

4.　まとめ

3回の連載で，CFTP に基づくパーフェクトサンプリングについて述べてきた．CFTP アルゴリズムは非常に面白いアイデアであるが，「パーフェクト」という性質は非常に繊細で，安易な改訂は危険である[24]．

マルコフ連鎖を用いたパーフェクトサンプリング法として，Fill は interruptible アルゴリズム（Fill のアルゴリズム）を提案している[1]．Fill のアルゴリズムも coalesce の確認を利用してパーフェクトサンプリングを実現する．連続状態のマルコフ連鎖に対するパーフェクトサンプリング法としては，Murdoch and Green が2段階カップリングのアイデアを用いたアルゴリズムを提案している[3]．

今回，パーフェクトサンプリング法について紹介したが，マルコフ連鎖を用いたサンプリングを行う際には，マルコフ連鎖の収束時間が大変重要となる．様々なマルコフ連鎖の収束時間の算定に関する研究が，現在も行われている．

[23] 来：どうやってこういう事考えつくんでしょう？　松：確率論とアルゴリズムの両方のセンスが少なくとも必要だねぇ．来：発想も天才的ですよね．松：初めて読んだ時ホント感動したよ．

[24] 来：以前，僕も改良を考えたんですけど，よく考えたら間違えてました．松：発表する前に気づいてよかったね．

参考文献

[1] J. Fill: "An interruptible algorithm for perfect sampling via Markov chains," The Annals of Applied Probability, 8 (1998), 131-162.

[2] O. Häggström: "Finite Markov chains and algorithmic application," London Mathmatical Society, Student Texts 52, Cambridge Univesity Press, 2002

[3] D. J. Murdoch and P. J. Green: "Exact sampling from a continuous state space," Scandinavian Journal of Statistics, 25 (1998), 483-502

[4] D. Wilson: "How to couple from the past using a read-once source of randomness," Random Structures Algorithms, 16 (2000), 85-113