■5 群（通信・放送）- 1 編（待ち行列理論とシミュレーション）

6 章 シミュレーション

（執筆者：山田博司）［2008 年 9 月 受領］

■概要■

通信／コンピュータシステムの処理動作を模擬するモデルを作成し，コンピュータ上で評価実験を行うことをシミュレーションという．シミュレーションは，通信／コンピュータシステムの性能予測を行う有用な方法論のひとつである．一方で，ハードウエア，ファームウエアなどを用いた評価実験をエミュレーションという．いずれもシステム評価手段として有用な方法であるが，ここでは，コンピュータプログラムを用いて，検討対象とする通信／コンピュータシステムのモデルを記述し，コンピュータ上に，仮想的なシステムを構築，様々な実験シナリオを評価するコンピュータシミュレーションについて述べる．

シミュレーションモデルにおいて，システムの状態を示す変数を「状態変数」という．状態変数が連続した値をとる場合は，連続型シミュレーションといい，微分方程式，差分方程式で表現されたモデルの解析評価でよく利用される．一方，状態変数が離散値をとる場合は，離散型シミュレーションという．通信／コンピュータシステムでは，現在処理中のタスク数，バッファ内のパケット数など，状態変数が離散値となるシステムを扱うことが多い．

シミュレーションには，様々なタイプがある．目的に応じて，どのようなタイプを用いるべきかを考える必要がある．時間軸を考慮しないシミュレーションに，モンテカルロシミュレーション（Monte Carlo simulation）がある．時間の経過による影響がない確率事象を扱う場合に利用される．トレース駆動型シミュレーション（trace-driven simulation）は，実システムで発生したイベント（event）レコード（これをトレース（trace）という）を入力情報として用いたシミュレーションである．このとき，トレース情報はシステム動作とは独立の関係である必要がある．キャッシュアルゴリズム，メモリアロケーション管理といった，リソース管理の解析やチューニングの際に利用されることが多い．ここで，イベントとは，システム状態の変化を意味する．離散イベント駆動型シミュレーション（discrete event-driven simulation: DES）は，システムの状態変数が離散値を取るモデルを扱う．通信／コンピュータシステムでは，状態変数としてシステム内パケット数を用いるなど，状態変数として離散値を扱うことが多い．イベントごとに状態が変化するため，イベントを管理するスケジューラ，シミュレーション時間進捗メカニズム，状態変数の管理や初期化処理，出力に関するコンポーネントを必要とする．

シミュレーションを構築するプログラム言語として，一般的な C/C++，Java 言語などを用いることもあるが，通信／コンピュータシステムを扱うシミュレーション専用のツールも広く利用されている．特に IP（Internet protocol）ネットワーク上のプロトコルやネットワーク全体の性能特性を分析する際，RFC（request for comment）で定義された様々なプロトコルでネットワークシステムが構成されていることから，シミュレーションモデル作成においても，これらプロトコルのシミュレーションライブラリを備えたシミュレータが有用である．OPNET[1)]，Qualnet[2)]，ns2[3)]などが，研究機関，企業，大学において広く利用されている．

検討対象である実システムの特徴・動作特性に着目し，これを抽象化したものをモデルという．モデリングとは，このようなモデルを構築することを示す．図 6･1 では，IP ネットワークシステムのモデリングについて示す．モデリングでは，まず実ネットワークから必要な情

報を入手する．IP ネットワークを構成するルータの物理的接続状況だけでなく，ルータに設定されているルーティングプロトコル，アドレスなどの設定パラメータ（コンフィグレーション: configuration）情報を得る．様々なプロトコルが連係動作して IP ネットワークは動作することから，プロトコル処理動作についても十分に分析する必要がある．また，大きなネットワークの場合，すべてのルータから構成されるネットワークモデルを構築するか，ネットワークのエッジ部分を構成するルータを集約したモデルにするかなど，評価の目的に応じて，モデルの詳細度について検討しなければいけない．また，設計・管理に必要な評価尺度を定義し，シミュレーション実験中に計測できるようにする必要がある．コンピュータ上にモデルを構成したら，シミュレーション実験を行う．様々なシナリオを想定し，比較検討を行うことを What-if test という．シナリオでは，ルータやリンクの設定を変更する場合，トラヒック量を変更する場合，ルータ，リンクの障害が生じた場合など，様々な状況を想定する必要がある．得られた性能結果とノード，リンクのコストなどその他の要因を含めて総合的に判断し，最終の意思決定を行う．採用されたシナリオの設定情報は，実ネットワークの設定にフィードバックされる．

シミュレーション実験は，モデルを利用することで経済的な費用・リスクをかけないで，システムの設計・評価分析が可能であるという優位性をもつ．しかし，モデルが対象システムの特徴をうまく表現できていない場合，得られた結果は有用でない，または，誤った結果を導く原因となる．そのため，モデリングでは，対象システムの特性をよく分析し，必要な機能を盛り込んだ適切なモデルを作成すること，モデリングに用いるコンピュータプログラムを正確に作成すること，適切な乱数と統計処理方法を適用して得られた結果を分析することが大変重要になる．

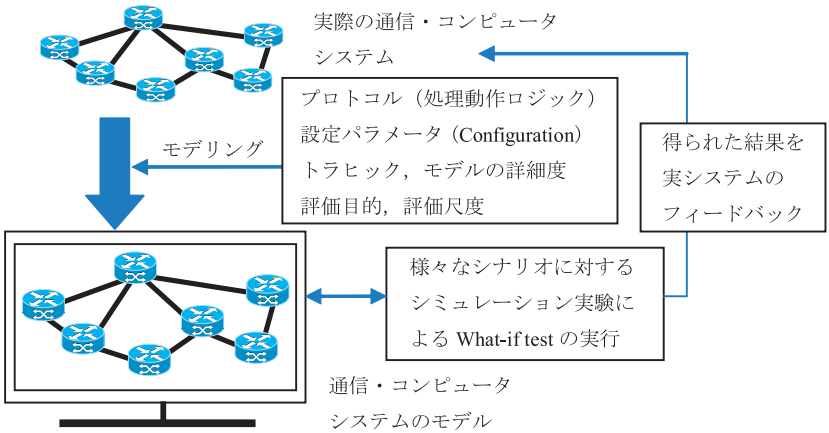

図 6·1　シミュレーションモデリング

■参考文献

1) OPNET Technologies: http://www.opnet.com/
2) Scalable Network technologies: http://www.qualnet.com/
3) ns2: http://www.isi.edu/nsnam/ns/

※ URL は，執筆時点（2008.9.30）のものである．また，文章中に記載されているシステム名，商標名は，その開発元の商標または登録商標であり，この資料では，解説説明を行う目的でのみ，商品名などを記載している．その商標権を侵害する意思，目的のないことをここに，申し述べておく．

【本章の構成】

本章では，6-1 節でモンテカルロシミュレーション，6-2 節で離散イベント駆動型シミュレーションについて説明する．その事例として市販シミュレーション製品の OPNET の例を示す．6-3 節では，擬似乱数の生成と確率分布に従うランダム変数の生成について，6-4 節では，シミュレーション実験によって得られた結果の統計処理，信頼区間の算出に関して説明する．

■5 群 - 1 編 - 6 章

6-1 モンテカルロシミュレーション

（執筆者：山田博司）[2008 年 9 月 受領]

モンテカルロシミュレーション（Monte Carlo Simulation）は，$[0,1]$ 上の一様分布 $U(0,1)$ に従うランダム変数を用いて，時間の経過による影響のない確率的，または確定的な問題を解決する手段として用いられるシミュレーション方法である．確定的な問題解決手段の一例として，多重積分値の計算に応用されている[1]．簡単な例として，式 (6･1) の積分を考える．

$$I = \int_a^b g(x)dx \tag{6･1}$$

ここで，$g(x)$ は，実数値を取る関数とする．この積分値 I をモンテカルロシミュレーションにより以下のように推定する．X を $[a,b]$ 上の一様分布 $U(a,b)$ に従うランダム変数とし，$(b-a)g(X)$ で定義されるランダム変数 Y を考える．このとき，ランダム変数 Y の平均は，式 (6･2) に示すように，積分値 I に等しくなる．N 回の試行によるランダム変数 X の実現値を $\{X_i\}$ $(i = 1, \cdots, N)$ とすると，変数 Y の平均値 $\overline{Y}(N)$ は，式 (6･3) であらわされる．従って，$\overline{Y}(N)$ の平均 $E[\overline{Y}(N)]$ は，I と等しい．$\overline{Y}(N)$ は，式 (6･4) のように I の不偏推定量となる．

$$E[Y] = (b-a)E[g(X)] = (b-a)\frac{\int_a^b g(x)dx}{b-a} = I \tag{6･2}$$

$$\overline{Y}(N) = \frac{\sum_{i=1}^{N} Y_i}{N} = (b-a)\frac{\sum_{i=1}^{N} g(X_i)}{N} \tag{6･3}$$

$$E[\overline{Y}(N)] = (b-a)\frac{NE[g(X)]}{N} = I \tag{6･4}$$

このように，ランダム変数 X を生成し，式 (6･3) を適用することで，積分値 I を推定する確定的な問題に適用できる．

■参考文献

1) A.M. Law and W. David Kelton, “Simulation Modeling and Analysis, 3rd ed.,” McGraw-Hill, New York, 2000.

■5 群 - 1 編 - 6 章

6-2 離散イベント駆動型シミュレーション

（執筆者：山田博司）[2008 年 9 月 受領]

システムの状態変数が離散的な値をとるモデルを対象とするシミュレーションを離散イベント駆動型シミュレーション（discrete event-driven simulation: DES）という．シミュレーションで利用する時間は，連続時間，または，離散時間のいずれの場合でもかまわない．通信／コンピュータシステムでは，状態変数として，ノード内のパケット数，CPU（central processing unit）が現在処理しているジョブ数など，離散的な値をとる場合が多く，離散イベント駆動型シミュレーションは性能評価・設計において広く利用されている．

離散イベント駆動型シミュレーションは，時間の経過にともない発生するイベントを管理する必要があるため，様々なコンポーネントから構成される[1, 2]．なお，イベントとは，システム状態変数の変化を意味する．主要なコンポーネントを以下に示す．

(1) イベントスケジューラ

現在のシミュレーション時点より先の時点に起きるイベントから構成されるイベントリストの管理を行う．新規イベントのリストへの登録，登録済みイベントのキャンセルなど，状態変化にともない，イベントリストを管理する．また，(2) で示すメカニズムに従い，シミュレーション時間進捗管理も行う．

(2) シミュレーション時間進捗メカニズム

イベントが生じた時点で，シミュレーション時間の進捗を行う必要がある．一般に時間進捗の方法には，単位時間（unit time）ごとに時間進捗を行う方法と現シミュレーション時点から，最も早く起きるイベント（イベントリストの最初にあるイベント）のイベント発生時間まで時間を進捗する，イベント駆動（event-driven）による時間進捗を行う方法がある．コンピュータシミュレーションの多くの場合，イベント駆動型の時間進捗メカニズムを採用する．

(3) システム状態変数

グローバル変数としてシステムの状態変数を定義する．

(4) イベント処理

イベントが発生した時点でシステム状態変数を更新し，この結果を踏まえて次に生ずるイベントのスケジュールを行う．この新規イベントは，イベントリストに登録される．

(5) 入力処理

シミュレーションに必要な各種パラメータを取り込む．

(6) レポート処理

シミュレーション終了後に，シミュレーション中に計測した統計量を整理し，表やグラフを作成する．

(7) 初期化処理

システムの初期状態を設定し，シミュレーションで用いられる各種変数の初期化を行う．

(8) トレース処理

シミュレーション途中の様々な変数の値を管理する．シミュレーションプログラムのデバッグにおいて必要な機能である．デバッグ後には，トレース処理は必要なく，トレース処理の有無を開始時点に設定できることが必要である．

(9) ダイナミックメモリ管理

シミュレーション実行時には，必要に応じてメモリの確保・解放が行われる．定期的なガーベージコレクション（garbage collection）機能が必要となる．

(10) メインプログラム

上記で示した各種処理を統合し，シミュレーションを実行する．

重要なコンポーネントは，(1) イベントスケジューラと (2) シミュレーション時間進捗メカニズムである．図 6･2 に，イベント駆動によるシミュレーション時間進捗時におけるイベントリストの事例を示す．

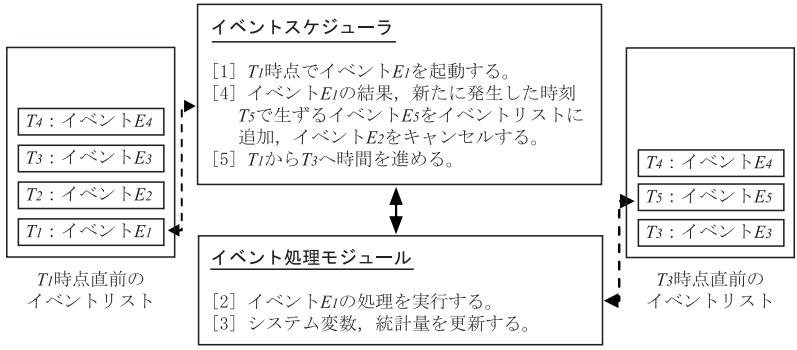

図 6･2　イベントリストの管理

図 6･2 に示すように T_1 時点直前のイベントリストには，E_1，E_2，E_3，E_4 の四つのイベントが待機している．なお，図 6･2 における時間順序は，$T_1 < T_2 < T_3 < T_5 < T_4$ であり，時間順序で，イベントリストに並べられている．T_1 時点のときに，メインプログラムはイベント処理モジュールをコールしてイベント E_1 を実行する．そのとき，変化するシステム状態変数，測定値の更新を行う．イベント E_1 は，新たなイベントを発生する場合やイベントリストにある待機状態のイベントをキャンセルする場合がある．例えば，イベント E_1 をシステムへのパケットの到着とする．システムは，タイマを設定しており，T_2 時点でタイムアウトのイベント E_2 が生ずるよう設定しているとする．タイムアウト時点 T_2 より前にパケットが到着した場合，システムは，タイムアウトイベント E_2 をキャンセルし，時点 T_5 に次のタイムアウトのイベント E_5 を定義する．このような処理は，通信システムでは頻繁に生ずる．これより，イベント処理モジュールは，イベントリストから，処理をしたイベント E_1，キャンセルとなったイベント E_2 を削除し，新たに発生したイベント E_5 を追加する．このとき，

イベントリストに存在する他のイベントとの時間順序に従い，イベント E_5 はイベント E_4 の前になっている．

一般に，通信プロトコル動作は複雑であり，通信ネットワークシステムでは様々なプロトコルにより，パケット転送やアプリケーション処理などを行うイベントが発生する．離散イベント駆動型シミュレーションは，このようなプロトコル，ネットワークシステムの性能評価に適した方法論であり，離散イベント駆動型シミュレーションを用いた多くのシミュレーションツールがある．OPNET, Qualnet, ns2 などが代表的なシミュレーションツールであり，米国を中心に，研究機関，企業，政府，軍事関係，大学などで使用されている．様々な RFC により定義されたプロトコルの処理動作をシミュレーションできるシミュレーションライブラリが用意されている．コンピュータ上に，ルータやスイッチなど，パケットの転送動作をシミュレーションできるノードオブジェクトや有線／無線のリンク特性を模擬できるリンクオブジェクトを用いて，ネットワークを構成し，シミュレーションを実行できる．シミュレーション結果を表形式やグラフによる表示が可能なレポート機能，シミュレーション中のパケットの疎通を示すアニメーションなど表示処理機能も充実している．

シミュレーションツールのプロトコルライブラリに存在する標準的なプロトコルで構成されたノード，リンクモデルを用いてネットワークを構成する場合，プログラミングの知識は必要としないが，ネットワーク，プロトコル及びコンフィグレーション設定の知識が不可欠となる．単純にノード間をリンクで接続してネットワークを構成するだけでは，正しくモデル構築はできない．例えば，ルータオブジェクトの設定をする場合，通常のルータの設定と同様，インタフェースに IP アドレスを付与し，ルーティングプロトコルの設定を行う必要がある．プロトコルのパラメータ設定には，正しく動作させるためのコンフィグレーションのルールが存在する．シミュレーションでは，プロトコル動作を維持管理する制御パケットもデータパケットと同様にネットワークを流れる．ルールに従ったパラメータ設定をしないと，対象システムとは異なる動作をするシミュレーションモデルを構築する危険性がある．

研究開発では，既存プロトコルの改良やプロトコルやノードシステムの新規開発に対し，これらのプロトコルが動作するネットワークシステムの性能評価を行う必要がある．この場合，シミュレーションモデル開発のために，プログラミングを必要とする．シミュレーションツールには，モデル開発用のエディタや API（application program interface）が用意されており，C/C++などのプログラミング言語を利用して，シミュレーションモデルの構築が可能である．

図 6･3 に，OPNET のシミュレーション開発環境を示す．OPNET は，階層的オブジェクト指向のシミュレーションツールで，ネットワーク，ノード，プロセスの三つの階層に対してモデリングを行う．それぞれの階層におけるモデルを図 6･3 に示す．ネットワークモデルでは，ノード，リンクオブジェクトを用いて，ネットワークを構成する．ノードオブジェクトは，そのノードで行われるプロトコル処理にあわせてプロトコルスタックを構成する．図 6･3 の例は，ルータオブジェクトであり，point-to-point，Ethernet のインタフェースに ARP（address resolution protocol），MAC（medium access control）プロトコル，IP，TCP（transmission control protocol），UDP（user datagram protocol），及び各種ルーティングプロトコルスタックが構成されている．プロセスモデルでは，RFC に記述された状態遷移図をモデル化し，各状態におけるプロトコル動作を C/C++で記述する．作成されたプロセス，ノー

ドモデルはオブジェクトとして再利用が可能である．シミュレーション実行環境では，シード（seed：本章 6-3 で記述）やシミュレーション時間，測定する統計量を設定して，シミュレーションを実行する．ルーティングテーブル，使用率など，統計量は表形式，グラフなどで表示される．パケットの転送動作のアニメーション表示も可能である．

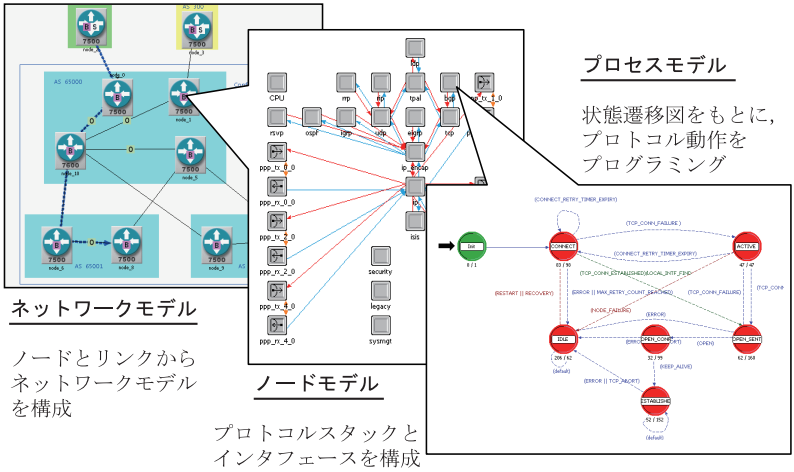

図 6·3　OPNET によるモデリング

■参考文献

1) R. Jain, “The art of computer systems performance analysis – Techniques for Experimental Design, measurement, Simulation, and Modeling,” John Wiley & Sons, Inc., New York, 1991.
2) A.M. Law and W. David Kelton, “Simulation Modeling and Analysis, 3rd ed.,” McGraw-Hill, New York, 2000.

■5 群 - 1 編 - 6 章

6-3 乱数の発生

（執筆者：山田博司）[2008 年 9 月 受領]

ランダムな変動要素がないためシミュレーション結果を確実に予測できる場合，これを確定的モデル（deterministic models）という．一方，ランダムな変動要素の影響を受けて，シミュレーション実験のたびに結果が変化するモデルを確率的モデル（probabilistic model）という．コンピュータシミュレーション実験では，このランダムな変動を，確定的アルゴリズムを用いた擬似乱数（pseudo-random number）を生成して実現する．ここでは，擬似乱数の生成方法について説明する．

最も良く利用されるアルゴリズムは，式 (6･5) に示すように，次に生成する乱数をそれ以前に生成された乱数の値の関数とする漸化式を用いる方法である[1)]．

$$x_n = f(x_{n-1}, x_{n-2}, \cdots) \tag{6･5}$$

初期値 x_0 をシードという．生成される乱数の系列を変更するためには，シードを変更する必要がある．関数 f は，確定的であるため，式 (6･5) により生成される数列は，確率 1 で決まる．図 6･4 に示すように，生成される数列は，数列の初期段階は値を繰り返すことはないが，ある点から同じ値をもつ数列を繰り返すという欠点をもつ．この初期期間（tail）と繰り返し期間（cycle length）をあわせてピリオド（period）という．

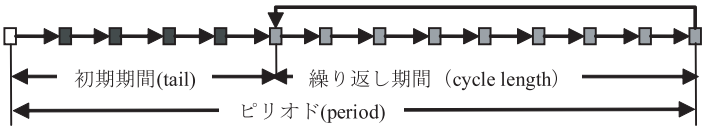

図 6･4　擬似乱数

擬似乱数の生成にあたっては，以下の三つの特徴を持つことが好ましい．

（1）コンピュータ上で効率的に生成しやすいこと
（2）繰り返し期間（period）が長いこと
（3）連続した生成値が独立で一様分布に従っていること

広く利用されているのが，線形合同法（linier-congruential generator: LCG）である．LCG のうち，式 (6･6) に示す漸化式が良く利用され，これを用いて，非負整数列 $\{x_n\}$ を生成する．

$$x_n = ax_{n-1} + b \quad \mod \quad m \tag{6･6}$$

ここで，x_n は，0 から $m-1$ の非負整数値になる．定数 a, b は非負の整数である．a, b, m の選択は，生成される乱数の繰り返し期間や自己相関性に影響を及ぼす．一般に，繰り返し期間を長くするために，m はできる限り大きな値を，またコンピュータ上で効率的に計算するために，2 のべき乗（2^k）の値が望ましい．b が 0 でない場合，$m = 2^k$ であり，b が奇数

であること，また，$a-1$ が 4 の倍数であることなどが望ましいことがわかっている[1)]．

0 または 1 からなるランダムシーケンスを必要とする暗号アプリケーションに適応した乱数発生法として，Tausworthe generator も良く利用されている．一般に，Tausworthe generator は，式 (6･7) に示す漸化式を持つ．

$$b_n = c_{q-1}b_{n-1} \oplus c_{q-2}b_{n-2} \oplus \cdots \oplus c_0 b_{n-q} \tag{6･7}$$

$$x^q + c_{q-1}x^{q-1} + c_{q-2}x^{q-2} + \cdots + c_0 \tag{6･8}$$

ここで，b_i, c_i は，0 または 1 の値をとる．作用素 $\oplus$ は排他的論理和（2 進数で桁上がりのない和）を意味する．また，式 (6･8) を特性多項式（characteristic polynomial）という．式 (6･7) で生成される乱数のピリオドは，特性多項式に依存する．式 (6･7) のように次数 q の特性多項式によって生成される乱数が最大取りえるピリオドは 2^q-1 であり，このピリオドを得るよう $\{c_i\}$ が設定された特性多項式を，原始多項式（primitive polynomial）という．式 (6･7) で得られた 0 または 1 からなる数列 $\{b_i\}$ を連続した s ビットずつのグループに分割し，その各グループの最初の l ビットを用いて，式 (6･9) により擬似一様乱数を生成する．これにより生成された乱数は，連続した二つの乱数の相関性がない．

$$x_n = \sum_{j=1}^{l} 2^{-j} b_{sn+j-1} \tag{6･9}$$

擬似乱数の生成には，そのほかにも様々な方法がある．Extended Fibonacci generator，Combined generator[1)]，Mersenne Twister generator[2)]などの方法がある．

シミュレーションでは，実測された値をもとに分析した経験分布の中から，最も適合する理論分布を用いて，ランダムな変動を模擬することが多い．よく利用される理論分布には，独立で同一分布に従う確率変数の和の分布を示す正規分布，待ち行列や故障頻度の解析に利用される指数分布，ポアソン分布，信頼性分析に利用される Weibull 分布などがある．これら理論分布に従う非一様乱数（non-uniform random numbers）は，一様乱数を生成し，これから，逆関数法（inverse transformation），採択棄却法（rejection method），合成法（composition method），などの手段を用いて生成する．ここでは，逆関数法と選択棄却法について説明し，主要な分布について具体的な生成方法を以下に述べる．

逆関数法は，図 6･5 左側に示すように，ランダム変数 x の分布 $F(x)$ が与えられたとき，$u = F(x)$ は $[0,1]$ 上に値を持つ変数になることを基本とする．$[0,1]$ 上の一様分布に従う乱数 u を生成し，逆関数 $x = F^{-1}(u)$ を計算することで，ランダム変数 x を生成する．

逆関数法による指数分布に従うランダム変数の生成方法について示す．指数分布の密度関数，及び，分布関数とその逆関数は以下の式で示される．

$$\text{密度関数：}\ f(x) = \lambda e^{-\lambda x} \tag{6･10}$$

$$\text{分布関数：}\ F(x) = 1 - e^{-\lambda x},\ x = \frac{-1}{\lambda} In(1-u) \tag{6･11}$$

一様乱数 $\{u_i\}$ を生成し，式 (6･11) を用いて平均 (λ^{-1}) の指数分布に従うランダム変数 $\{x_i\}$

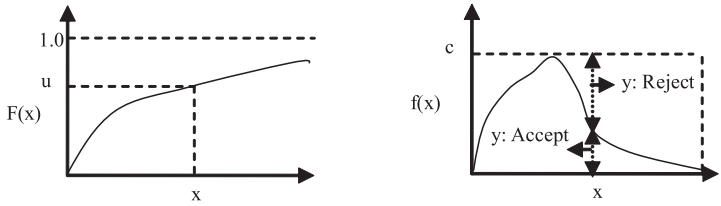

図 6･5　逆関数法と選択棄却法による非一様乱数の生成

表 6･1　主な理論分布の逆関数 [1, 3)

理論分布	分布関数	逆関数
幾何分布（Geometric）	$1-(1-p)^x$	$\left[\frac{In(u)}{In(1-p)}\right]$
ロジスティック分布（Logistic）	$1-\frac{1}{1+e^{(x-\mu)/b}}$	$\mu - bIn\left(\frac{1}{u}-1\right)$
パレート分布（Pareto）	$1-x^{-a}$	$\frac{1}{u^{1/a}}$
ワイブル分布（Weibull）	$1-e^{(x/a)^b}$	$a(In(u))^{1/b}$

を生成する．表 6･1 にその他の主な理論分布の分布関数とその逆関数について整理する．

選択棄却法は，図 6･5 右側に示すように，ランダム変数を生成しようとする分布の密度関数 $f(x)$ に対し，ランダム変数を生成できる別の密度関数 $g(x)$ が存在し，すべての変数 x について，$cg(x) \geq f(x)$ を必ず満足する場合に，以下の手順でランダム変数を生成する．

まず，（1）密度関数 $g(x)$ に従って，ランダム変数 x を生成する．（2）次に，$[0, cg(x)]$ 上の一様分布に従う変数 y を生成する．（3）このとき，条件式 $y \leq f(x)$ を満足する場合に，x をランダム変数として出力する．そうでない場合はこれを棄却し，（1）のステップに戻って条件式が満たされるまで処理を繰り返す．（3）のステップが選択棄却処理になる．

正規分布に従うランダム変数を生成する方法としては，Box-Muller の方法が良く知られている[4]．以下のステップにより標準正規分布（平均 0，標準偏差 1）のランダム変数を生成する．

（1）$[0, 1]$ 上の一様乱数 U_1, U_2 を生成する．（2）変数 V_1, V_2, S を $V_1 = 2U_1 - 1, V_2 = 2U_2 - 1, S = V_1^2 + V_2^2$ と定義する．（3）$S > 1$ の場合，ステップ（1）に戻る．そうでない場合，（4）それぞれ独立した二つのランダム変数を式 (6･12) に従って計算し，二つの独立した

ランダム変数 x, y を生成する.

$$x = V_1 \sqrt{\frac{-2In(S)}{S}}, y = V_2 \sqrt{\frac{-2In(S)}{S}} \quad (6\cdot12)$$

■参考文献

1) R. Jain, “The art of computer systems performance analysis – Techniques for Experimental Design, measurement, Simulation, and Modeling,” John Wiley & Sons, Inc., New York, 1991.
2) M. Matsumoto and T. Nishimura, “Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator,” ACM Trans. On Modeling and Computer Simulation, vol.8, no.1, pp.3-30, 1998.
3) A.M. Law and W. David Kelton, “Simulation Modeling and Analysis, 3rd ed.,” McGraw-Hill, New York, 2000.
4) S.M. Ross, “Simulation, 2nd ed.,” Academic Press, New York, 1997.

■5 群 - 1 編 - 6 章

6-4 モデルの検証と正当性の評価

（執筆者：山田博司）[2008 年 9 月 受領]

シミュレーションモデルは，様々な仮定を設けて作成される．これらの仮定が合理的であること，設定した仮定がモデルに正確にインプリメントされていることの 2 点が重要である．

まずモデルの検証（verification）について述べる．シミュレーションモデルは，大きなコンピュータプログラムであり，プログラム開発やデバッグに関する様々なテクニックが有効になる．モジュール化と階層構造によるモデル構築がひとつの有力な方法である．モデル全体に階層構造を持たせ，機能ごとの小さなモジュールで全体モデルを構成する．OPNET の例で言えば，ネットワーク，ノード，プロセスによる階層化を実現し，プロトコルごとにプロセスモデルを構築することでモジュール化を図る．独立したモジュール構成は，モデルを並行して開発可能になり，デバッグ作業も容易になる．プログラミングでは，処理ごとにチェック機能を記述し，その都度パラメータの値を表示させる方法，シミュレーションイベントのトレース情報を出力させる方法，オンラインでのアニメーション表示などが，検証には有効である．市販ツールには，これら機能を持つデバッグツールが完備されている．開発にかかわっていない第 3 者に対して作成したコード内容を説明することは，バグ発見に役に立つ．簡易モデルや確定的なモデルを用いてその結果の妥当性を解析することも必要である．

次にモデルの正当性評価（validation）について述べる．以下の三つの観点から正当性を評価する．

（1）モデリングで用いた仮定の妥当性
（2）入力パラメータ値の特性
（3）出力結果

（1）については，対象システムの専門家による確認が最も実践的な方法である．通信プロトコルの評価を行う場合，RFC の記述と比較することもひとつの手段である．RFC には，プロトコルの状態遷移，メッセージフォーマット，処理に関する説明があり，必要な部分がモデルに組み込まれているか，専門家と確認することが確実な手段となる．

（2）では，実システムの測定値との比較は最も望ましい方法である．ただし，シミュレーション時に，システムが存在しない場合や測定コストの問題から測定ができない場合もある．そのため，同じようなシステムから入力仮定を類推する場合や仮定を設けて理論分布で対処する場合もある．

（3）では，仮定を簡略化した解析モデルの理論結果と比較する手段がよく用いられる．ただし，解析モデルも今評価しているシミュレーションモデルも，精密に実システムを表現できていないかもしれないということに注意しなければいけない．以上，シミュレーションモデルの正当性評価について述べたが，実システムに対して十分に正当化されたモデルは存在し得ないという点は，評価上留意しなければいけない．

多くのシミュレーション実験では，定常状態における結果を扱う．そのため，シミュレーションが定常状態に至るまでの過渡的な状態での情報を削除することが必要になる．また，適切なシミュレーション時間を取ることも重要になる．これらの問題について述べる．

はじめに過渡状態の扱いについて述べる．一般に過渡状態を正確に定義することは難しい．

そのため，以下に述べる発見的な方法がある．長時間シミュレーションをすることで，過渡状態の影響が小さくなることが予想される．しかし，どの程度時間を取ればよいかを決定することが難しいこと，また，長時間コンピュータリソースを占有することから，この方法はあまり利用されない．初期値として定常状態に近い状態を設定する方法として，簡略化した待ち行列モデルから平均値などを導き，これを初期値として使う方法もある．また，「定常状態における状態変数の変化は，初期変動のときの変化より小さい」という仮定に基づいて，初期時間部分を削除するなど，様々な方法がある．例えば，観測値 $\{x_1, x_2, x_3, \cdots, x_{n-1}, x_n\}$ のうち，最初の l 個を取り除いた残りの観測値 $\{x_{l+1}, \cdots, x_{n-1}, x_n\}$ の最大値 $x_{l\_\max}$，最小値 $x_{l\_\min}$ を算出する．$l = 1, 2, \cdots, n-1$ とし，$l+1$ 番目の観測値，x_{l+1} が $x_{l\_\max}$，$x_{l\_\min}$ と一致しない場合，l を過渡状態の長さとして，データ $\{x_{l+1}, \cdots, x_{n-1}, x_n\}$ を用いる方法がある．また，種（seeds）を変更して，サイズ n の m 回のシミュレーション実験を実行する．このとき，$j = 1, \cdots, n; i = 1, \cdots, m$ とする．以下のステップで，削除する初期状態の長さ l を決める方法がある．

（1）各時点における平均値 $\bar{x}_j = (1/m)\sum_{i=1}^{m} x_{ij}$ を求める．

（2）次に，全体平均 $\bar{\bar{x}} = (1/n)\sum_{j=1}^{n} \bar{x}_j$ を求める．

（3）初期値として，$l = 1$ とし，l 以降の $n - l$ 個からなる全体平均 $\bar{\bar{x}}_l = (1/n-l)\sum_{j=l+1}^{n} \bar{x}_j$ を算出する．

（4）全体平均の相対的変化 $(\bar{\bar{x}}_l - \bar{\bar{x}})/\bar{\bar{x}}$ を算出する．

（5）（3）と（4）の処理を繰り返して，$l = 1, 2, \cdots, n-1$ について（4）で定義した相対的変化を計算する．

（6）順番に相対的変化の値をプロットし，相対的変化が安定する時点 l を見つけ出す．このとき，最初の l 個のデータを，初期状態として削除する方法である[1)]．

最後にシミュレーションの信頼区間について述べる．シミュレーション時間が短いと得られた結果の変動は大きくなる．一方，シミュレーション時間を長くすることは，コンピュータや人的リソースを浪費することになる．シミュレーションで得られるデータは，それぞれ独立ではなく，相関性を持つことが多い．ここでは信頼性区間を算出する方法として，バッチミーン法（batch means），再生法（regeneration）の二つの方法について説明する[1, 2, 3)]．

図 6･6 左側のように，バッチミーン法では，シミュレーションを長時間走らせ，初期状態を削除したのち，残りの観測値をいくつかのグループ，バッチ（batch）に分割する．例えば，$N + n_0$ 個の観測値から，初期状態の観測値 n_0 を削除し，残りの N 個の観測値を，n 個の観測値からなる m 個のバッチに分割する．x_{ij} を i バッチの j 番目の観測値とする．それぞれ分割したバッチの平均 $\bar{x}_i$，観測値全体の平均 $\bar{\bar{x}}$ は，式 (6･13)，式 (6･14) で表せされる．このとき，バッチ平均の分散 $Var(\bar{x})$ は，式 (6･15) で表現され，信頼区間は，式 (6･16) で示さ

れる．ここで，$z_{1-\alpha/2}$ は，標準正規分布 $N(0,1)$ の $1-\alpha/2$ パーセンタイル値である．

$$\overline{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij} \qquad i = 1, 2, \cdots, m \tag{6･13}$$

$$\overline{\overline{x}} = \frac{1}{m}\sum_{i=1}^{m} \overline{x}_i \tag{6･14}$$

$$Var(\overline{x}) = \frac{1}{m-1}\sum_{i=1}^{m} (\overline{x}_i - \overline{\overline{x}})^2 \tag{6･15}$$

$$[\overline{\overline{x}} \mp z_{1-\alpha/2} Var(\overline{x})] \tag{6･16}$$

次に再生法について述べる．説明のための例として，図 6･6 右側に，単一待ち行列の行列長の推移の例を示している．空の状態からスタートし，再び行列が空になる時点があり，その後，また行列が伸びるという事象を繰り返す．再生法では，システム状態が初期値にもどり，また新規に状態が変化する現象を再生（regeneration）といい，初期値に戻る時点を再生点（regeneration point）という．再生点で区切られた区間（再生サイクル: regeneration cycle）での変化は，それぞれ独立になると考えられ，この特性をもとに信頼区間を求めるのが，再生法である．今，m 回の再生サイクルを考え，それぞれの区間サイズを $\{n_1, n_2, \cdots, n_m\}$ とする．以下のステップで，信頼区間を算出する．

まず，それぞれのサイクルにおける平均 $\overline{x}_i$ は，式 (6･17) で表現される．全体平均 $\overline{\overline{x}}$ は，各サイクルでのサイズが異なることから，式 (6･18) を用いて，式 (6･19) で表現される．

$$\overline{x}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} x_{ij} \tag{6･17}$$

$$y_i = \sum_{j=1}^{n_i} x_{ij} \tag{6･18}$$

$$\overline{\overline{x}} = \frac{\sum_{i=1}^{m} y_i}{\sum_{i=1}^{m} n_i} \tag{6･19}$$

サイクル長の期待値と実際のサイクル長の差分の分散 $Var(w)$ は式 (6･20) で表現される．

$$Var(w) = \frac{1}{m-1}\sum_{i=1}^{m} w_i^2 \qquad \text{ただし，} \quad w_i = y_i - n_i\overline{\overline{x}},\ i = 1, 2, \cdots, m \tag{6･20}$$

このとき，信頼区間は，式 (6･21) で示される．

$$[\overline{\overline{x}} \mp z_{1-\alpha/2}\frac{\sqrt{Var(w)}}{\overline{n}\sqrt{m}}] \qquad \text{ただし，} \quad \overline{n} = \frac{1}{m}\sum_{i=1}^{m} n_i \tag{6･21}$$

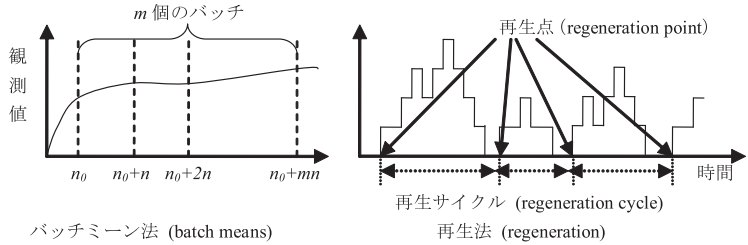

図 6·6　バッチミーン法と再生法

以上のような統計処理をして，シミュレーションで得られる値の信頼区間を算出する．

■参考文献

1) R. Jain, “The art of computer systems performance analysis – Techniques for Experimental Design, measurement, Simulation, and Modeling,” John Wiley & Sons, Inc., New York, 1991.
2) A.M. Law and W. David Kelton, “Simulation Modeling and Analysis, 3rd ed.,” McGraw-Hill, New York, 2000.
3) S.M. Ross, “Simulation, 2nd ed.,” Academic Press, New York, 1997.