Delphi によるコンピュータシミュレーション入門

高木英至

1 コンピュータシミュレーション

この章では Delphi を用いた計算モデルの作成を解説する。計算モデルの作動を実験することが計算機実験、つまりコンピュータシミュレーションである。

シンボリックな体系で記述したモデルには大別して３つの種類がある。第１は自然言語で記述した言語モデル、第２は数理モデル、第３が計算機の言語で対象をモデル化した計算モデルである。その何れのモデルでも、モデルの帰結（予測）を導くこと、つまり思考実験としてのシミュレーションを行うことができる。しかし言語モデルは厳密さに難があり、数理モデルはモデル構築の柔軟性に欠ける面がある。計算モデルは数理モデルのような一般性のある結論は導けないものの、モデル構築の柔軟性によって効率的に帰結を導出することができる。計算モデルが様々な研究分野で導入され普及したのはそのためである。社会心理学においても、認知領域はもちろん、社会的影響、対人関係、集団過程、群集行動、社会的ジレンマなどの領域で計算モデルの適用例を多く見出すことができる。

計算モデルによるシミュレーション（以下、シミュレーションと略する）のためには古くからシミュレーション専用の言語ないしソフトが開発されていた。現在の Windows ないし MacOS では、Stella、EX･TD(Extend)、VisSim などが選択対象といえる。これらのソフトは一様にヴィジュアルなモデル構築を支援し、積分などの多様な計算モジュールや便利な結果表示のモジュールを備えている。その利便性や、自分でプログラムして誤りを犯すリスクを考えると、こうしたシミュレーションソフトを用いるのは有力な選択肢である。

しかし、シミュレーション専用のソフトは、一定のパタンにあったモデル、例えば力学系のモデルや待ち行列などを構築する利便性を達成している反面、他のパタンのモデルを構築するには不便なことがある。例えば、マルチエージェント型のシミュレーションを行うのは別仕様のソフトが必要になってくる。また、ヴィジュアルなモデル構築はモデルが複雑になると分かりやすいとはいえない。

Delphi などの汎用の開発言語で計算モデルを構築する利点と欠点は専用のソフトを使う場合の逆である。プログラマは白紙の状態から計算手順を考え、ソースコードを書かなければならない。しかし他方で、モデル構築の自由度と柔軟性を得ることができるし、原則としてどのようなタイプのモデルにも対応できる。ある程度定まった計算の手法については既存のライブラリを利用することで対処できるだろう。

この章では社会心理学における計算モデルの実例を示しながら、Delphi でどのように計算モデルを作れるかを解説してみる。何を取り上げるかについては次の基準を設定した。第１は実際の研究論文で使われた、したがってある程度は社会心理学的に意味のあるモデルの再現プログラムを用いる、という点である。第２は、プログラムがこの本で説明できる程度に単純なことである。以下で取り上げる２つの例は、全体のソースコードは長く見えても本質的な計算

の部分はきわめて短く、解説を要する計算手法を使う訳でもない。

2　内集団の多様性認知： Linville らの計算モデル

2．1　モデルの背景

はじめに「内集団の多様性認知」を説明する Linville, Fischer & Salovey (1989)のモデルを取り上げる。このモデルは単純でありながら、記憶依存型の認知の要点をよく押さえている。

人が内集団と外集団に異なった認知を持つことはよく知られている。まず人は、外集団成員より内集団成員を好意的に認知する（ingroup favoritism）。また、人は外集団成員より内集団成員を多様であると認知する傾向がしばしば報告されている。ここで取り上げる現象は後者、つまり内集団の多様性認知という効果である。

この効果の成立には様々なルートが想定できる。第１に、社会的アイデンティティ理論のように、一定の動機づけによって人は内集団をより多様と認知するかも知れない。「多様性」は望ましい特性と解釈される可能性もある。黒い羊効果のように内集団の望ましくない成員をより悪く（もしくは望ましくない成員の比率をより多く）認知するなら、内集団は結果としてより多様に認知されるかも知れない（逆に黒い羊効果自体が内集団成員の多様性認知の結果であるかも知れない）。第２に、人は内集団成員との相互作用を予期しやすいために内集団成員を外集団成員より綿密に情報処理し、その結果、内集団成員をより多様と認知するかも知れない。第３に、外集団には予めステレオタイプが定着している場合が多く、外集団成員の認知が stereotype-driven である可能性もあるだろう。

しかし Linville らは上記のような要因によらずとも内集団成員が外集団成員より多様と認知される、と考えた。内集団は外集団より熟知度(familiarity)が高い、つまり接触するイグゼンプラ（成員事例）の数において内集団が外集団より高い。この熟知度の差が多様性認知の差となると考える。

内集団のイグゼンプラには多く接するので内集団をより多様に認知する、とは、言葉でいえばその通りのような気もする。しかしこの結論は論理的にどのように導けるのか？

2．2　Linville らのモデル

Linville らがそのモデルで仮定するのは単純化したイグゼンプラ(exemplar)の記憶依存型の認知である。観察する行動事例（イグゼンプラ）は集団所属（行為者が内集団成員か外集団成員か）と１次元の属性の値を持っていると仮定する。属性には１から７までの値（水準）がある。属性の値は集団所属にかかわらず同じ確率分布で生じる。イグゼンプラの集団所属と属性は誤りなく認知されるとしよう。

観察者は１期間で一定数のイグゼンプラに順次接し、確率的にそのイグゼンプラを長期記憶に移すと考える。接したイグゼンプラが長期記憶に貯蔵される確率(*PLearn*)は 0.5 であるが、極端な属性値（１か７）の場合は記憶の確率が上がる（*PLearn* = 0.9）。ただしイグゼンプラの記憶には確率的に忘却が生じる（*PForget* = 0.1）。観察者が内集団や外集団に対する判断を形成するときには、集団所属の値（C_i）をプローブ(probe)として長期記憶内を検索する。プローブに合うイグゼンプラは確率的に（*PRetrieve*=0.75）再生されるが、極端な属性値（１か７）のイグゼンプラでは再生の確率は高まる（*PRetrieve*=0.95）。プローブに合わないイグゼンプ

ラの再生確率はゼロである。再生されたイグゼンプラは活性化の強度を持つ。

観察者が認識する対象集団の属性値の分布は、再生したイグゼンプラの強度の和に比例すると考える。A_j をある属性値とし、$SAS(C_i, A_j)$を集団カテゴリC_iで再生した属性値がA_jのイグゼンプラの活性化強度の総和とすれば、C_iでA_jの属性値を持つ成員の比率は次の$P(A_j | C_i)$で決まる。

$$P(A_j \mid C_i) = \frac{SAS(C_i, A_j)}{\sum_{k=1}^{m} SAS(C_i, A_k)}.$$

ただし活性化の強度はすべてのイグゼンプラで 1.0 と仮定するので、$P(A_j | C_i)$は再生したイグゼンプラの中でA_jの値を持つものの比率と等しくなる。

２．３　フォームのデザイン

上記のモデルを Delphi でコード化しプログラムしたのが p_group1 である。付録のＣＤ－ＲＯＭに収録した p_group1.exe という実行ファイルをクリックして起動すると、デスクトップに図１のようなウィンドウが現れる。図１はこのプログラムのフォームを表す。

シミュレーションのプログラムで重要なのはその結果ないし経過をいかに視覚的に表示するかである。計算だけをして後で解析するという方法もある。しかしシミュレーションの経過を表示することでコードの誤りを見つけられることもあるし、現象に関する新たな発見もある。

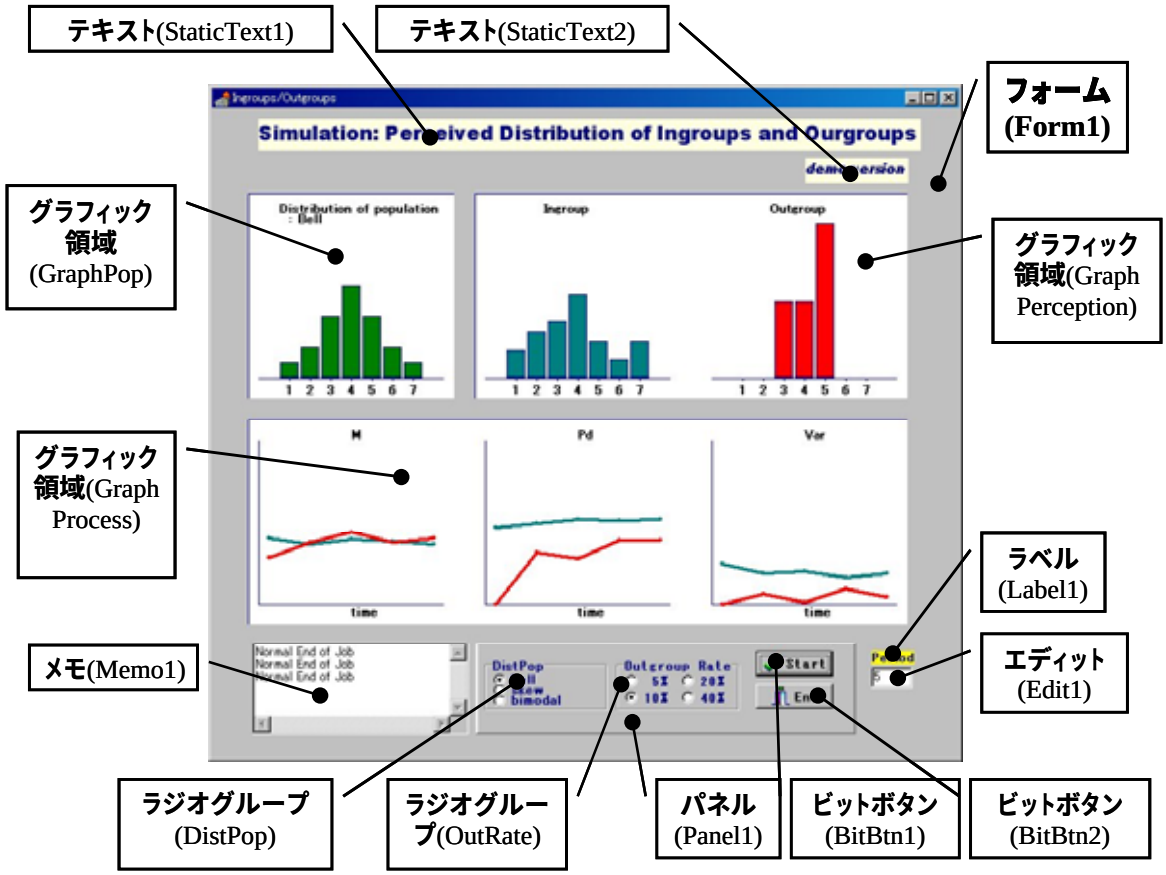

図1：p_group1 **のフォーム**

p_group1 ではフォームに３つの描画（グラフィック）領域のコンポーネント（クラスは TPaintBox）を使っている。GraphPop というグラフィック領域は属性の母集団分布を表示する。ラジオグループ DistPop で別の分布を選択すれば GraphPop の表示も変化する。GraphPerception は, 内集団と外集団のそれぞれに対する分布知覚を、GraphProcess はシミュレートした多様性指標の時間的経過を表示する。

２．４　ユニットファイルのデザイン

p_group1 のユニットファイルの中身は付録のＣＤ－ＲＯＭに収録したファイルの中の p_group1a.pas である。ここで全体を出すには長いので、リスト１にこのユニットファイルの中身を、フォームのクラス TForm1 の宣言部を中心に載せている。リスト１からこのユニットファイルのデザインが分かる。まず行７～１９にこのフォームに付けたコンポーネントが現れる。このフォーム内の手続きは行２０～２７の８つ、そのうち作業の流れを制御するのは FormCreate、BitBtn1Click、BitBtn2Click の３つのイベントハンドラである。まずプログラムが実行されると同時に FormCreate が実行される。この手続きは図１の画面を設定するだけである。Start ボタン（BitBtn1）がクリックされると BitBtn1Click が呼び出される。この手続きは RunSimulation を呼び出し、１回のシミュレーションを実行する。End ボタン（BitBtn2）がクリックされると BitBtn2Click が呼び出され、プログラムを終了する。その他の DrawGraphPop、ClearGraphPerception、ClearGraphProcess、DistPopClick は描画の手続きであり、計算の本質にはかかわらない。

リスト１

```
unit p_group1a;

interface
・・・途中省略・・・
type
  TForm1 = class(TForm)
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    GraphPop: TPaintBox;
    GraphPerception: TPaintBox;
    GraphProcess: TPaintBox;
    Memo1: TMemo;
    Panel1: TPanel;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    DistPop: TRadioGroup;
    OutRate: TRadioGroup;
    Label1: TLabel;
    Edit1: TEdit;
```

```
    procedure DrawGraphPop(id: byte);//GraphPop を描く
    procedure ClearGraphPerception;//枠だけの GraphPerception を描く
    procedure ClearGraphProcess;//枠だけの GraphProcess を描く
    procedure FormCreate(Sender: TObject);//起動時の作業指定:起動時画面の描画
    procedure RunSimulation;//シミュレーションプログラム
    procedure BitBtn1Click(Sender: TObject);//'Start' ボタン（シミュレーション１試行実行）
    procedure BitBtn2Click(Sender: TObject);//'End' ボタン（プログラムの終了）
    procedure DistPopClick(Sender: TObject);//RadioButton の DispPop クリック時の作業
・・・途中省略・・・
  end;

  const
      PopSize      : array[1..2] of integer = (200, 200);//GraphPop の縦横幅
・・・途中省略・・・
  var
      Form1        : TForm1;

implementation
・・・実現部を省略・・・
end.
```

２．５　シミュレーション手続き

このモデルの本質的な部分は手続き RunSimulation である。そこで RunSimulation がどのようにできているかを見てみよう。

（１）変数・定数・型の宣言

リスト２は RunSimulation 冒頭の変数、定数、型の宣言部分のコードである。まず const 以下の３行で、シミュレートする期間(Nperiods)と１期で観察者が出会うイグゼンプラの数(Nencounters)を宣言している。つまりこのプログラムは、５期間、計１００個のイグゼンプラに出会う状況をシミュレートすることになる。

リスト２

```
procedure TForm1.RunSimulation;
  const
      Nperiods       =  5;
      Nencounters    = 20;
  type
    exemplar_type = record
                        group : byte;
```

```
                    attribute : array[1..5] of byte;
                    strength : real;
                end;
    memory_set    = set of 1..Nencounters;
    index_type    = array[1..2,1..Nperiods] of real;
    null_type     = array[1..2,1..Nperiods] of boolean;
var
    exemplar          : array[1..Nencounters] of exemplar_type;
    WorkingMemory     : memory_set;
    LongTermMemory : array[1..500] of exemplar_type;
    OutgroupRate      : real;
    NOutGroup         : byte;
    LTMsize           : byte; { size of long term memory }
    iperiod           : integer;
    Pdistribution     : array[1..2,1..7] of real;
    index_M, index_Pd, index_Var : index_type;
    ex_null           : null_type;
```

次の type 以下の行では変数の型を定義している。イグゼンプラの型をレコード型として定義する。イグゼンプラは集団所属(group)、属性(attribute)、活性化強度(strength)という要素を持つ。group と attribute は byte 型(０から２５５までの整数)であり、strength は実数型(real)である。これらの要素を exemplar_type というレコードとしてまとめている（attribute は配列で定義され、５つの要素を持つが、このシミュレーションでは１番目の要素しか使わない)。var 以下の変数の宣言部では、exemplar という変数（一期で出会うイグゼンプラ）を exemplar_type を持つデータの配列として宣言している。つまり、Nencounters（＝２０）個の exemplar がすべて、group、attribute、strength の要素を持つのである。また、長期記憶を表す変数 LongTermMemory もこの exemplar_type の配列として定義している。

memory_set はプログラムの途中で登場する WorkingMemory の型であり、１から Nencounters までの整数を要素とする集合（集合型）として定義する。変数宣言部(var 以下)で WorkingMemory という変数を memory_set で定義している。

index_type はシミュレーションの結果を表す指標（変数の index_M、Index_Pd、index_Var）の型として定義しており、実数の２次元配列となっている。なお、index_M は属性値の集団ごとの平均、index_Pd と index_Var は属性値のバラツキの指標である。index_Pd[i,j] は i 番目の集団（i=1 なら内集団、i=2 なら外集団）の j 期の Pd 指標の値である。指標変数の型宣言をするのは、指標の変化を GraphProcess に描画する際に指標変数をパラメータ（引数）とするのに便利なためである。

null_type は論理型(boolean)の配列として定義している。null_type で定義する変数は ex_null と手続き draw_index の変数 null であり、同様に ex_null をパラメータとして使う便宜のためにここで型宣言をしている。

（２）計算手順

手続き RunSimulation の末尾の begin から end; に至るまでのメインプログラム部分がリスト３である。リスト３の中で呼び出される手続きはこのメインプログラムの前で記述されている。

リスト３

```
begin
    initialize;
    repeat
        initialize_period;
        generate_exemplar;
        Learning;
        forgetting;
        add_to_LongTermMemory;
        retrieval_perception;
        indices;
        graphs;
        slowdown;
    until iperiod = Nperiods;
    finale;
end; {of RunSimulation }
```

まず、この RunSilulation は手続き initialize で計算の初期設定を行う。乱数の宣言である randomize も initialize に入っている。initialize の次の repeat から until iperiod = Nperiods;までがループをなし、このループは期間の数だけ、つまり現在の期間を表す変数 iperiod の値が Nperiods になるまで、繰り返される。

２．２のモデルをコード化しているのがこのループ内の諸手続きである。はじめに手続き initialize_period でその期間における初期設定を行う。次の手続き generate_exempLar は、その期間に観察者が接するイグゼンプラを発生させる。手続き Learning では長期記憶への貯蔵が、手続き forgetting で忘却が生じる。手続き add_to_LongTermMemory は忘却分を勘案した長期記憶の調整を行う。手続き retrieval_perception ではその期間での再生、および再生に基づく属性値分布の認知を行う。次に手続き indices で結果指標の計算をし、手続き graphs で結果指標をグラフに描画する。手続き slowdown は時間稼ぎの手続きであり、シミュレーションの過程を可視的にするためにプログラムの実行を遅らせる働きをしている（時間稼ぎが必要なのはデモ用のプログラムだけである）。

以下ではこの repeat … until のループ内の主な手続きを解説しておこう。

（３）イグゼンプラの発生

手続き generate_exemplar の作業はイグゼンプラを作って group、attribute、strength の値を代入するだけである。この手続きのソースについては付録ＣＤ－ＲＯＭにある p_group1a.pas をご覧頂きたい。１期に生じるイグゼンプラ数は Nemcounters（=20）と決まっている。そのうちの外集団成員の比率(OutGroupRate)、および２０のうちの外集団事例の

数(Noutgroup)が initialize で決まる。さらにイグゼンプラの何番目を外集団事例にするかも乱数で決める。しかる後、Nencounters の数だけイグゼンプラを発生させ、exemplar 変数に値を代入する。イグゼンプラの attribute の値はユニットファイル全域で定義された確率分布の定数 PLevel から一様乱数を用いて求める。group の値はここまでで決まっており、strength にはすべて 1.0 を代入する。

（4）記憶

Learning、forgetting、および add_to_LongTermMemory の３つの手続き（リスト４）によってイグゼンプラは長期記憶に格納される。まず、Linville らのアイディアには登場しない WorkingMemory という集合型の変数を使う。WorkingMemor は RunSimulation の変数宣言部で宣言された変数であり、RunSimulation 内のどの手続きでも使うことができる。

リスト４

```
{*****    Learning    *****}
  procedure Learning;
    var
        i          : byte;
        PLearn     : real;
    begin
        WorkingMemory := [];
        for i := 1 to Nencounters do begin
            if (exemplar[i].attribute[1] = 1) or (exemplar[i].attribute[1] = 7)
            then PLearn := 0.9
            else PLearn := 0.5;
            if random <= PLearn then
                WorkingMemory := WorkingMemory + [i];
        end; { of i loop }
    end; { of Learning }

{*****    forgetting *****}
  procedure forgetting;
    const
        PForget   = 0.1;
    var
        i     : byte;
    begin
        for i := 1 to Nencounters do
            if (i in WorkingMemory) and (random <= PForget) then
                WorkingMemory := WorkingMemory - [i];
    end; { of forgetting }
```

```
{*****     add exemplars to LongTermMemory     *****}
  procedure add_to_LongTermMemory;
    var
        i       : byte;
    begin
        for i := 1 to Nencounters do begin
            if (i in WorkingMemory) then begin
                LTMsize := LTMsize + 1;
                LongTermMemory[LTMsize] := exemplar[i];
            end; { of if }
        end; { of i loop }
    end; { of add_to_LongTermMemory }
```

手続き Learning の冒頭でまず、WorkingMemory を空にする。そして次に for i のループで、確率 PLearn で接触したエグゼンプラをいったん WorkingMemory に格納する。

手続き Forgetting ではその WorkingMemory に入っているイグゼンプラを、確率 PForget で WorkingMemory から抜くのである。

手続き add_to_LongTermMemory では、WorkingMemory にいったん格納され、かつ忘却を免れたイグゼンプラを LongTermMemory[LTMsize]として、つまり最初からの通し番号 LTMsize 番目の長期記憶要素として記憶する。LTMsize とはそのときの長期記憶内のイグゼンプラ数である。注意すべきは、変数 exemplar と LongTermMemory とは同じ型(exemplar_type)で定義されていることである。そこで LongTermMemory[LTMsize] := exempLar[i];とだけ指定すれば、exemplar[i] の持つ group、attribute、strength の値はそのまま LongTermMemory[LTMsize]に代入される。

（５）再生と属性分布知覚

イグゼンプラの再生と集団の属性分布認知を手続き retrieval_perception で行う（リスト５）。イグゼンプラの属性値が極端であるか否かによって再生確率(PRetrieve)を変え、長期記憶から再生されるイグゼンプラを決める。そして集団 i、属性値 j の活性化強度を SAS[i,j]に加算してゆく。この SAS[i,j]の値に比例して、集団 i における属性値 j のイグゼンプラの推定比率を Pdistribution[i,j]に代入する。

リスト５

```
{*****     retrieval & distribution perception     *****}
  procedure retrieval_perception;
    var
        i, j        : byte;
        PRetrieve : real;
        SAS           : array[1..2,1..7] of real;
        id1, id2    : byte;
        ss            : real;
```

```
begin
    for i := 1 to 2 do
        for j := 1 to 7 do
            SAS[i,j] := 0.0;
    for i := 1 to LTMsize do begin
        if (LongTermMemory[i].attribute[1] = 1)
            or (LongTermMemory[i].attribute[1] = 7)
        then PRetrieve := 0.95
        else PRetrieve := 0.75;
        if random < PRetrieve then begin
            id1 := LongTermMemory[i].group;
            id2 := LongTermMemory[i].attribute[1];
            SAS[id1,id2] := SAS[id1,id2] + LongTermMemory[i].strength;
        end; { of if random < }
    end; { of i loop }
    for i := 1 to 2 do begin
        ss := 0.0;
        for j := 1 to 7 do ss := ss + SAS[i,j];
        if ss > 0.0 then
        begin
            for j := 1 to 7 do
                Pdistribution[i,j] := SAS[i,j]/ss;
        end
        else
        begin
            for j := 1 to 7 do Pdistribution[i,j] := 0.0;
            ex_null[i,iperiod] := true;
        end; { of if ss > 0.0 }
    end; { of i loop }
end; { of retrieval_perception }
```

問題は、SAS[i,j]を合計である変数 ss の値がゼロであることが生じ得ることである。特にもともとのイグゼンプラ数の少ない外集団について ss の値はゼロになりやすい。発生されるイグゼンプラ数が少ない上に、そのすべてが記憶されるとは限らず、記憶されたイグゼンプラが再生されるとも限らないからである。そこで if ss > 0.0 の else の begin … end;において ss がゼロのときの処理を施す。すなわち、論理型変数である ex_null[i,iperiod]に、集団 i の ss が iperiod 期においてゼロであるときに真(true)を代入する。この ex_null の値はその後の計算結果の処理に用いる。このプログラムでは、手続き indices で計算した指標を手続き graphs で表示する際に利用する。また、計算試行を多数繰り返してその合計の結果を表示するプログラム（後述の p_group2）では、ss=0.0 となったケースを指標の平均値の算出から除外するため

に用いる。

（6）指標の計算

手続き indices では Linville らが用いた３つの指標を期間ごとに計算する。index_M[i,j] は集団 i の j 期における attiribute の値の平均値である。index_Pd と index_Var は観察者が知覚した属性値の分布の、集団・期間ごとのバラツキの指標である。index_Var は知覚した分布の分散にあたる。index_Pd が表す Pd は次の式で計算される。

$$P_d = 1 - \sum_{i=1}^{m} P_i^2$$

ただしP_iは知覚した分布における属性値の水準iの比率である。指標P_dはエントロピと同様に、属性が名義尺度と考えた場合のバラツキの程度を表している。

２．６　シミュレーションの結果

p_group1 を実行するとこのシミュレーションの１試行が画面に表示される。仮想の観察者が知覚した内集団・外集団の分布は GraphPerception に描かれ、３つの指標の期間間の変化は線グラフとして GraphProcess に描かれる。母集団の分布型を変化させたければ左側の DistPop のラジオグループで選択を行い、イグゼンプラにおける外集団成員の比率を変化させたければ右側の OutRate のラジオグループで選択を行えばよい。

Linviulle らの論文が紹介するシミュレーション結果では、index_M の平均値は内集団／外集団で差がなく、index_Pd と index_Var の平均値は内集団で高い、つまりイグゼンプラの多い内集団で分布はバラついて認知される。p_group1 はこのパタンの結果を出すこともあるけれど、試行によっては逸脱するパタンも示す。そこで読者は、このプログラムが本当に Linville らのモデルを再現しているのか、と疑問に思うかも知れない。

そのために用意したのが付録のＣＤ－ＲＯＭに収録した p_group2 というプログラムである。このプログラムは p_group1 に若干の手を加え、１０００試行を行ってその指標の平均値を GraphProcess に表示するようにしてある。Linville らが描く通りの結果であることを確認できるはずである。

３．組織における男性支配： Martell らの計算モデル

前節では内集団の多様性認知という、個人心理のメカニズムのモデルを扱った。次により「社会的」な計算モデルの例を取り上げよう。

３．１　モデルの背景

世の中にはジェンダーステレオタイプがある。sexism といってもよい。このステレオタイプは多くの場合、女性にとって不利に働く。例えば、同じ成功を収めたとしても行為者が男性であればその成功の原因が本人の能力にあると認知されるのに対し、女性であれば運や課題の容易さに成功が帰属されるやすい、といった結果が報告されている。世の中での成功が能力本位の評価に基づくとしても、こうしたステレオタイプの存在は女性の社会進出を阻む効果を持つと考えられる。

しかしこのステレオタイプの効果をどう評価するかについては１つの議論があり得るだろう。

確かに女性へのバイアスは有意な効果として析出されるとしても、過去の研究で評価に男女差が及ぼす効果は評価の分散の１～５%を説明するに過ぎない。つまり女性の不利益は意外と小さい。そこで、女性の不利な立場は認めるとしてもその是正のために社会的費用の大きな措置、例えば女性を優遇するアファーマティヴアクションの採用をすべきかどうかには、疑問がある、といった議論である。

こうした議論に反論し、Martell らはある単純なシミュレーションによって「小さなバイアスが女性を大きく傷つける」と結論づけるのである。

３．２　Martell らのモデル

Martell らは次のようなモデルを考えた。まず８つの階層レベルのある組織を仮定する。階層レベルは下から上に行くにしたがって成員規模が大きくなるピラミッド構造になる。最下層レベルは５００人、最上層レベルは１０人からなっている。この組織に成員が配置される。成員は性別と業績を示すスコアを持つ。スコアは平均５０、標準偏差１０の正規乱数で決まる。ただし男性成員はバイアスによって業績を過大に評価され、スコアの分散の一定比率を説明する分のボーナススコアを加算される。

この組織で次の過程が生じると仮定する。時間は離散的な期間(period)として進行する。まず初期状態では各階層レベルに、スコアとは関係なしに男女が同数ずつ配置されている。１期間のうちに１５%の成員が、階層とはかかわりなくランダムに退職する。その結果生じた空席に、１つ下のレベルの成員から、スコアの高い順に着任させる。一番下のレベルでは外部から新規に成員を雇用する（新たな成員を発生させる）。このモデルでは、昇進はレベルを追って生じ、中間レベルを通り越して新人がトップに抜擢されることはない。しかし最下層レベルで「入社」した非常に優秀な新人は、最短で８期間後にはトップに着任することができる。

以上の前提の下でシミュレーションを行えば、組織は初期状態からどのように変化するであろうか？　昇進はスコアに基づくので、上位レベルでスコアが高くなることは明らかである。また、男性はボーナススコア分が有利であるので、組織の上位で男性が多くなるもの確実である。問題はこの「小さなバイアス」で男性支配がどの程度、どのような様態で生じるか、という点である。

３．３　フォームのデザイン

付録のＣＤ－ＲＯＭに収録した feminist.exe というプログラムをクリックすると図２のようにフォームが画面に現れる。配置したコンポーネントはこの画面から視覚的に確認できる。テキスト、グラフィック領域、メモ、パネル、ラジオグループ、ビットボタン、エディット、ラベルを使った点は p_group1 と同じである。異なるのは、新たにボタン（TButton 型）、スクロールバー（TScrollBar 型）、そしてこの画面には現れないタイマー(TTimer 型)のコンポーネントを配置したことである。

グラフィック領域では、まず左上の OrgMap で組織構造を表示する。組織の各レベルでの男女比率は右上の MFMap で横バーで示す。右下の ScoreMap において各レベル成員の平均スコアを表示している。

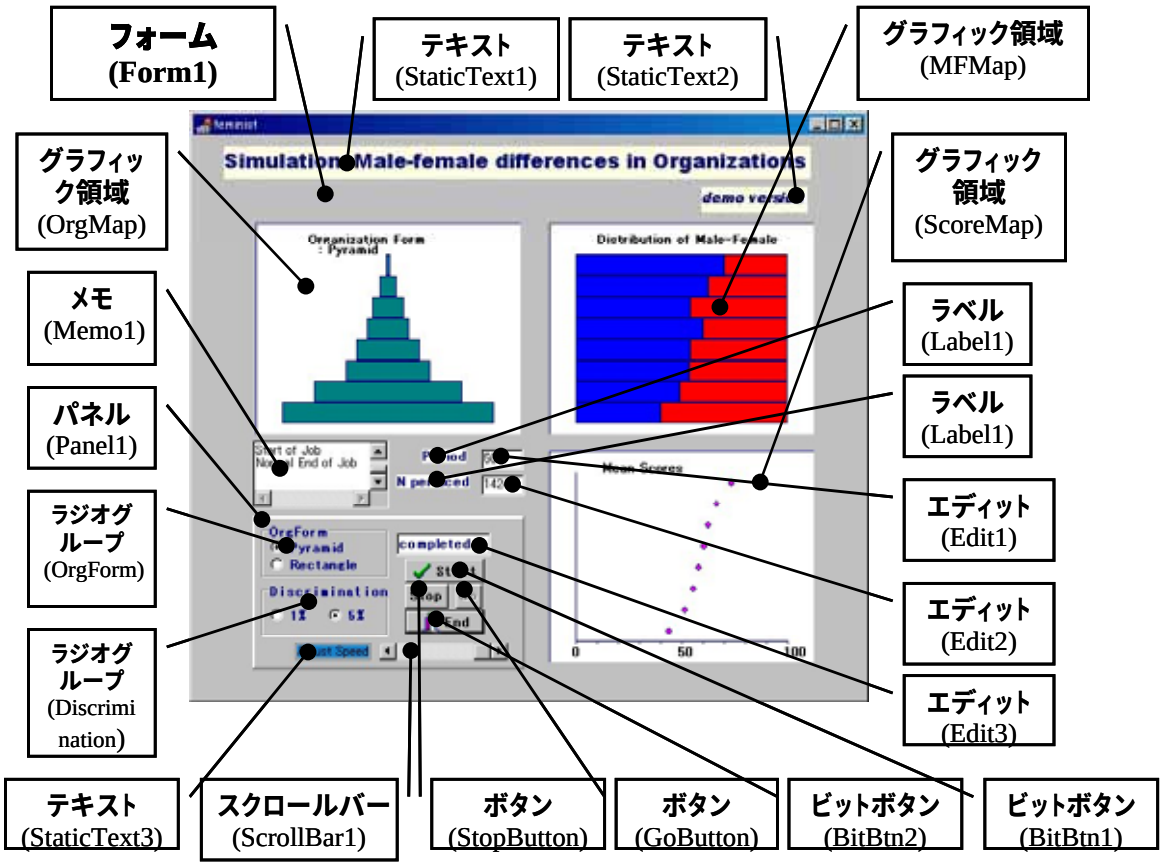

図2：feminist のフォーム

feminist が p_group1 と異なる点の１つは、シミュレーションの途中で実行を停止できるようにした点である。シミュレーションのデモを提示しながら途中で解説を入れるような場合を考慮している。Stop ボタンは、Start ボタンをクリックして開始したプログラムの実行を中断させるのに用いる。Stop ボタンで中断させた実行は Go ボタンをクリックすると再開する。

ラジオグループ OrgForm では組織構造を Martell らと同じ Pyramid にするか、あるいはどのレベルも同じ規模の矩形型（Rectangle）にするかを選ぶことができる。この選択はその場で OrgMap の表示に反映される。ラジオグループの Discrimination では、スコア分散における男女差の説明分（つまりボーナススコアの大きさ）を５%にするか１%にするかを選択できる。またスクロールバーによりシミュレーションの進行速度を変えることができる。

３．４　ユニットファイルのデザイン

feminist のユニットファイルの中身は付録のＣＤ－ＲＯＭに収録したファイルの中の feminista.pas である。リスト６にその全体の構造だけを取り出して表示している。行７～２７にこのフォームに付けた Timer 以下の２１個のコンポーネントを記している。行２８～４２に、このフォーム内で使う１つの関数と１４の手続きのヘッダが書いてある。

リスト６

```
unit feminist4a;
```

```
interface
・・・途中省略・・・
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    OrgMap       : TPaintBox;
    MFMap        : TPaintBox;
    ScoreMap     : TPaintBox;
    Memo1        : TMemo;
    Panel1       : TPanel;
    BitBtn1      : TBitBtn;
    BitBtn2      : TBitBtn;
    StopButton : TButton;
    GoButton     : TButton;
    Discrimination: TRadioGroup;
    OrgForm      : TRadioGroup;
    Edit1        : TEdit;
    Label1       : TLabel;
    Edit2        : TEdit;
    Label2       : TLabel;
    Edit3        : TEdit;
    ScrollBar1 : TScrollBar;
    function    normal(m, s: real) : real;//正規乱数の発生
    procedure reordering(tLevel: byte);//組織成員の並べ換え
    procedure DrawOrgMap;//OrgMap の描画
    procedure DrawMFMap(initial: boolean);//MFMap の描画
    procedure ClearScoreMap;//枠だけの ScoreMap の描画
    procedure InitialScreen(Sender: TObject);//初期画面の設定
    procedure FormCreate(Sender: TObject);//起動時の処理
    procedure initialize;//シミュレーション条件の初期化
    procedure SimOnePeriod;//１期間のシミュレーション
    procedure BitBtn1Click(Sender: TObject);//Start ボタンのクリック時処理
    procedure BitBtn2Click(Sender: TObject);//End ボタンのクリック時処理
    procedure StopButtonClick(Sender: TObject);//Stop ボタンのクリック時処理
    procedure GoButtonClick(Sender: TObject);//Go ボタンのクリック時処理
    procedure OrgFormClick(Sender: TObject);//ラジオグループ OrgForm クリック時の処理
```

```
    procedure timer1timer(Sender: TObject);//タイマーイベント呼び出し
・・・途中省略・・・
  end;

  const
        OrgMapSize       : array[1..2] of integer = (250, 200);
        MFMapSize        : array[1..2] of integer = (250, 200);
        ScoreMapSize     : array[1..2] of integer = (250, 200);
        Nincumbents      = 1424;
        Nlevels          = 8;
  type
      incumbent_type   = record
                              id          : integer;
                              employed    : boolean;
                              female      : boolean;
                              score       : real;
                              Level       : byte;
                         end;
        organization_type   = array[1..Nlevels] of integer;
  const
        SizeOfLevel      : array[0..1] of organization_type
                         =((500,350,200,150,100,74,40,10),
                           (178,178,178,178,178,178,178,178));
  var
        Form1            : TForm1;
        LevelSize        : organization_type;
        incumbent        : array[1..Nincumbents] of incumbent_type;
        Nfemale          : array[1..Nlevels] of integer;
        MeanScore        : array[1..Nlevels] of real;
        replaced         : boolean;
        iperiod          : integer;
        MFdif            : real;
        global_id        : integer;
        ScoreX           : array[1..2,1..Nlevels] of integer;

implementation
・・・実現部を省略・・・
end.
```

関数 normal は正規乱数（期待値 m、標準偏差 s の正規分布にしたがう乱数）を与える関数

である。予め組み込まれている乱数 random は０から１までの値をとる一様乱数を与える関数であり、その random から正規乱数を作る。特定の確率分布にしたがう乱数を一様乱数から作るアルゴリズムはいろんなアルゴリズムの解説書で記述されている。

feminist を実行させるとフォームが作られ、手続き FormCreate が実行される。FormCreate は Go ボタンを無効にし、タイマーの呼び出し間隔を５０ミリ秒に設定しつつタイマーをオフにする。そして手続き InitialScreen を呼び出して初期画面を作成する。ここで「Start ボタンをクリックする」というイベントをユーザがおこすと手続き BitBtn1Click が呼び出される。このとき、タイマーをはじめて有効にし、逆に Stop ボタン以外のボタンは無効にする。さらにシミュレーション条件の初期化（例えば組織人員の初期配置の決定）を手続き initialize で実行する。BitBtn1Click がするのはこれだけであるが、タイマーを有効にしたので、指定した間隔の時間が経過すると手続き timer1timer が実行される。timer1timer は手続き SimOnePeriod を呼び出す。この SimOnePeriod は１期間だけのシミュレーションを行う。さらに時間間隔が経過すると SimOnePeriod で次の１期間のシミュレーションを実行する。このようにして複数期間にわたるシミュレーションを実行してゆくのである。

Start ボタンをクリックして複数期間の１試行のシミュレーションが終了したら、もう一度 Start ボタンを押せば二度目のシミュレーションが実行され、End ボタンを押せばプログラムが終了する。しかし開始したシミュレーションを途中で止めたければ Stop ボタンを押せばよい。プログラムは「Stop ボタンをクリックした」というイベントを受け取り、現在実行中の SimOnePeriod が終了した時点で手続き StopButtonClick に移る。StopButtonClick ではまずタイマーを無効にし、有効にすべきボタンを有効にする。タイマーは無効になっているから、SimOnePeriod は呼び出されず、シミュレーションは止まったままである。この状態で Go ボタンを押すと再びタイマーが有効に設定されるから、中断した状態から StopButtonClick を再開することになる。

feminist ではシミュレーションの作業は SimOnePeriod 以外に、手続き initialize などに分散している。しかもある時点で実行して得た SimOnePeriod 内部の変数の値を次に実行する SimOnePeriod に受け継がせる必要がある。そのため、シミュレーションで使う変数・定数の一部をユニットファイルの冒頭で定義してユニットファイル全域で有効にしている（行４６～７５）。配列の定数の OrgMapSize、MFMapSize、ScoreMapSize はそれぞれのグラフィック領域の縦横の大きさであり、Nincumbents は組織の成員数、Nlevels は組織階層のレベル数である。ピラミッド型／矩形型での各レベルの人員数は定数 SizeOfLevel が与える。

変数の型を定義する type において、組織成員をレコード型の incumbent_type で定義している。組織成員は５つの変数の値のセットである。id には成員の発生順の一連番号が割り当てられる。emplyed は、組織に在職していれば真(true)である。女性であれば female が真となる。score には正規乱数で決めた実数値のスコア値（男性の場合はボーナススコアをプラスする）が入る。level はその成員が属する階層レベルである。type の下の変数定義部(var)では、成員を表す変数 incumbent を incumbent_type の配列として定義している。incumbent の中に格納されているデータは、組織に現在属している成員のデータである。

３．５　シミュレーションの手続き

シミュレーションの主要な計算は手続き SimOnePeriod の中にある。この手続きのメインプ

ログラムが SimOnePeriod の末尾の begin ～ end;である（リスト７）。

リスト７

```
begin
        timer1.enabled := false;
        initialize_period;//期間の初期化
        attrition;//組織成員の退職
        promotion;//昇進
        employment;//新規雇用
        indices;//指標の計算
        graphs;//期間データの描画
        slowdown;//時間稼ぎ
        timer1.enabled := true;
        if replaced then finale;//シミュレーション終了処理
end; {of SimOnePeriod}
```

まず作業のため timer1.enabled := false;でタイマーを無効にし、手続き initialize_period で現在の期間番号を決める。次の手続き attrition では退職者を決める。作業としては確率 ProbQuit で現成員の employed に偽(false)を代入するだけである。

promotion の手続き（リスト８）において組織内の昇進を処理する。最上位から順にレベル２（下から２番目のレベル）までのレベルを取り出し、そのレベルの成員のデータ番号の範囲(istart から iend まで)を求める。次にその範囲のそれぞれの成員につき、もし空席(employed が偽)であれば下のレベルから昇進させる成員を検索する。成員は手続き reordering によって各レベル内でスコアの降順に並べ換えされているから、上から探して最初にいた空席でない成員をあてればよい。昇進させる成員のデータを空席の成員の変数に代入し、昇進した成員のもとの変数には手続き null_incumbent によって空席データを入れておく。もし組織内に選べる成員がいなくなれば（番号の Nincumbents を過ぎれば）検索を止め、空席はそのままにしておく。

リスト８

```
  procedure promotion;//昇進
    var
          iLevel          : integer;
          istart, iend : integer;
          ip              : integer;
          icandidate      : integer;
          go_ahead        : boolean;

    procedure null_incumbent(itarget: integer);
       begin
```

```
            with incumbent[itarget] do begin
                id := 0;
                employed := false;
                score := 0.0;
                level := 0;
            end;
        end; { of null_incumbent }

    begin
        istart := 1;
        for iLevel := Nlevels downto 2 do begin
            if ilevel < Nlevels then
                istart := istart + LevelSize[iLevel+1];
            iend     := istart + LevelSize[iLevel] - 1;
            icandidate := iend;
            for ip := istart to iend do
            if not incumbent[ip].employed then begin
                repeat
                        go_ahead := false;
                        icandidate := icandidate + 1;
                        if icandidate > Nincumbents then go_ahead := true
                        else
                            if incumbent[icandidate].employed then go_ahead := true;
                until go_ahead;
                if icandidate <= Nincumbents then begin
                    incumbent[icandidate].Level := iLevel;
                    incumbent[ip] := incumbent[icandidate];
                    null_incumbent(icandidate);
                end; { of if icandidate <= Nincumbents }
            end; { of if not incumbent[ip].employed }
        end; { of iLevel }
    end; { of promotion }
```

組織の欠員を埋めるのが手続き employment である。新規の成員は無限母集団から無作為抽出する人員であるから、初期条件として成員を決めたのと同じ手続きで成員を発生させ、その値を決める。性別は男女確率半々で無作為に決めている。ピラミッド構造の組織の場合、この新規採用が生じるのは最下レベルにおいてだけである。ラジオグループ OrgForm で組織構造を矩形にすれば下の２レベルで新規採用が生じることになる。

本質的な計算はここまでである。後は手続き indices でレベルごとの女性数(Nfemale)と平均スコア(MeanScore)を求め、手続き graphs で画面に表示する。１期間のシミュレーション

結果は次に SimOnePeriod を呼び出したときの出発点となり、順次シミュレーションが進行する。シミュレーションの終了条件は Martell らの原論文にしたがい、最初からいた成員がすべてその後に採用した成員に入れ替わることである。終了条件が充たされると論理型の終了判定変数 relpaced に手続き indices において真が代入され、手続き finale の実行となる。finale ではタイマーを無効にするので、SimOnePeriod は呼び出されなくなる。

３．６　シミュレーションの結果

feminist はシミュレーションの１試行をデモンストレイトするプログラムである。したがって多数回の試行を行った集計結果のパタンから逸脱する結果を出すこともある。多数回の結果を出したければ feminist を手直しし、例えば条件ごとに１００試行を繰り返すようにプログラムを書き換えればよい。そのように書き換えて計算結果を集計すれば次のようなパタンを見出すことができる。

組織がピラミッド構造の場合、まず第１に、レベルが上がるほど平均スコアは単調に高くなる。第２に、最下層レベルでは女性の比率が高く、レベルが上がるほど女性比率が低下する。Martell らにとって重要なのはむろん２番目の結果である。つまりわずかなバイアスが男性中心の権力関係を作るというのである。

しかし組織構造を矩形に指定したとき、確かに上位レベルでは平均スコアと男性比率は上昇するけれど、その他の点ではピラミッド構造の場合とは違ったパタンが生じる。つまり中間レベルで平均スコアが落ち込み、女性比率が高まる傾向が生じる。なぜこのような傾向が生じるかは練習問題として考えてみてほしい。

４　結び

この章では Delphi による２つのシミュレーションプログラムを解説した。これらのプログラムは研究用としては単純なプログラムであり、解説を要するような計算手法も使っていない。例えば認知系のシミュレーションであれば、今日の主流はニューラルネット風の考えを導入したモデルである。また、　社会のシミュレーションであればマルチエージェント型のシミュレーションが多い。そして課題に応じて進化的計算手法を導入することもある。計算モデルの適用についてはそれぞれの分野での適用を解説した書籍・論文を参照してほしい。

しかしここでいいたいのは、耳障りの良い計算手法を導入するかどうかは本質的には重要でないことである。最も重要なのは、この章において配慮したように、背景となる問題が何であり、その問題の認識からいかなるアイディアやモデルを考えるか、という点である。それゆえ、シミュレーションを学ぶ上で最も重要なのは、第１に関心のある研究領域について高い理解を達成すること、第２に対象をモデル化する感覚を養うことであるといわねばならない。

参考文献

Linville, P.W., Fischer, G.W., & Salovey, P. (1989) Perceived distributions of characteristics of in-group and out-group members: Empirical evidence and a computer simulation, *Journal of Personality and Social Psychology, 57*, 165-188.

Martell, R.F., Lane, D.M., & Emrich, C. (1996) Male-female differences: A computer simulation. *American Psychologist, 51*, 157-158.