岡本安晴　2001.2.4;2004.7

極値探索－３

多変数関数：偏導関数を用いない場合

偏導関数を用いずに極小値の探索を行うものとして、Rosenbrock の方法と Brent の方法について説明する。これらの方法では、極小値を求めたい目的関数だけで極値探索が行えるので、目的関数が複雑なときでも簡単に極小値を求めることができる。

Rosenbrock の方法

極値探索の効率化を図るために、Rosenbrock 法では探索の主方向をそれまでの探索方向に適応させる方向に変えていく[(1)]。具体的には次のように探索を進める。

極小値を求める関数がｎ変数の関数 $y = f(x_1,\cdots,x_n)$ であるとき、ｎ個の探索方向をとり、それらを S_1、・・・、S_n とする。これらの探索方向は、最初はｎ個の座標軸の方向にとっておく（初期値）。ｎ個の探索方向に対して、順番に直線上の 1 次元極値探索を行う。Rosenbrock の方法では、この１次元極値探索は予め設定されたステップサイズを基にしておおまかに行われるが、ここではより効率的な探索が期待できる方法である２次関数の近似を用いたより精密な探索を行う[(1)、(2)]。

S_1 から S_n までの方向での１次元極値探索が終了したら、これらの探索方向ベクトルの更新を行う。この更新後の探索方向が、更新前の探索の移動方向に対応するようにする。すなわち、S_i 方向での移動量が λ_i であったとき、S_1 の新しい探索方向が探索における変数値の全変化 $\sum_{i=1}^{n}\lambda_i S_i$ に一致するようにとる。更新後、S_1 の方向は、S_1 から S_n までの探索を全て合わせた方向になる。２変数（ｎ＝２）の場合なら、$\lambda_1 S_1 + \lambda_2 S_2$ の方向が更新後の新しい S_1 の方向となる（図１）。

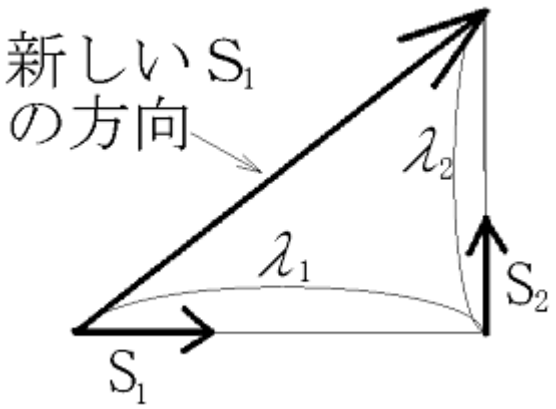

図１　S_1 方向への移動と S_2 方向への移動の和が次の S_1 方向となる（２変数の場合）。

S_2 の方向は、S_1 の方向での探索による移動を除いたもの $\sum_{i=2}^{n}\lambda_i S_i$ と一致するようにする。以下同様に、S_j の方向は S_j から S_n までの探索による移動の和 $\sum_{i=j}^{n}\lambda_i S_i$ の方向と一致するようにする。各探索方向、S_1、・・・、S_n、を新しく $S_j=\sum_{i=j}^{n}\lambda_i S_i$ で与えた後、正規直交化を行う。

探索方向の更新後、新しい探索方向で探索を行う。この探索方向の更新と探索を極小値に達したと判断されるまで繰り返す。

上の説明を手順としてまとめると、以下のようになる。

（１）変数の初期値を $\mathbf{x}_0$、探索方向 S_i の初期値を i 軸方向 $\mathbf{e}_i$ とする。$\mathbf{e}_i$ は i 番目の要素が１、他の要素はすべて０のベクトルである。

（２）$f_0 \leftarrow f(\mathbf{x}_0)$、$\mathbf{x} \leftarrow \mathbf{x}_0$、$i \leftarrow 1$ とおく。矢印「←」は、右側の値を左側に設定することを表わす。

（３）$f(\mathbf{x}+\lambda_i S_i)$ を最小にする λ_i を求める（１次元探索）。この探索は２次関数による近似で行うが、２次関数による近似の方法がうまくいかないときは golden section 法に切り換える。

（４）$\mathbf{x}\leftarrow\mathbf{x}+\lambda_i S_i$、$i\leftarrow i+1$ とおく。

$i \le n$ ならば（３）に戻る。

$i > n$ ならば（５）に進む。

（５）各 λ_i の大きさが十分に小さい、あるいは関数値 $f(\mathbf{x}_0)$ から $f(\mathbf{x})$ への変化が十分に小さいときは、極小値に収束したとして探索を終了する。λ_i の大きさ、および関数値の変化量が十分に小さくないときは、（６）に進む。

（６）$S_i \leftarrow \lambda_i S_i + \lambda_{i+1} S_{i+1} + \cdots + \lambda_n S_n$、$i = 1, \cdots, n$ とおく。これらの S_i を、以下のように Gram-Schmidt の直交化法で正規直交系にする。

（６ａ）$S_1 \leftarrow S_1 / \|S_1\|$、$i \leftarrow 2$ とおく。ここで、記号 $\| \ \|$ はベクトルの長さを表わす。

（６ｂ）$S_i \leftarrow S_i - \{(S_i \cdot S_1)S_1 + \cdots + (S_i \cdot S_{i-1})S_{i-1}\}$

$S_i \leftarrow S_i / \|S_i\|$

$i \leftarrow i+1$

とおく。ここで、記号「・」は内積を表わす。

（６ｃ）$i \le n$ ならば（６ｂ）に戻る。

$i > n$ ならば（７）に進む。

（７）$\mathbf{x}_0 \leftarrow \mathbf{x}$、 $i \leftarrow 1$ とおいて（２）に戻る。

ステップ(６)におけるGram-Schmidtの直交化法とは、１次独立なｎ個のベクトル、S_1、･･･、S_n、に対して、順番に長さを１に揃えながら、直交成分をとりだすものである。すなわち、まず、S_1の長さを１にする（ステップ（６ａ））。次に、S_1と直交する成分をS_2から取り出すために、S_2のS_1への正射影の長さを求める（図２）。

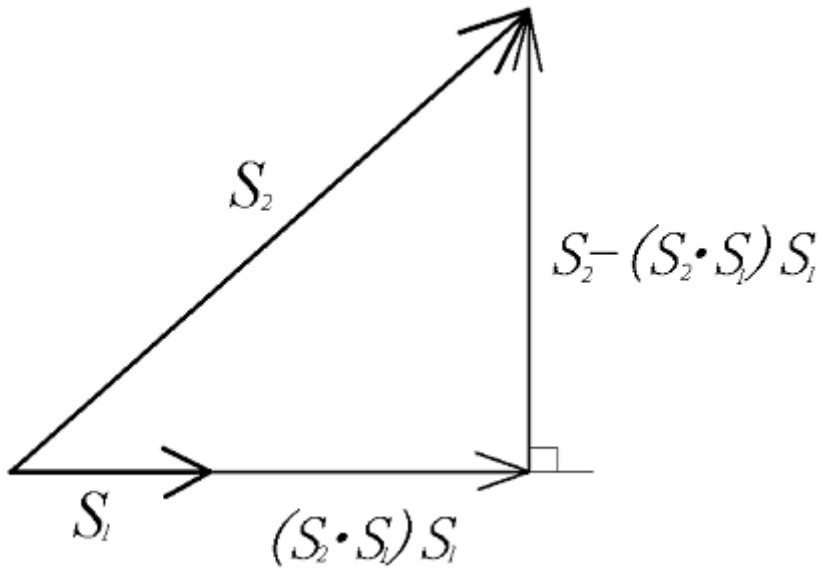

図２　S_1と直交するベクトル$S_2-(S_2\cdot S_1)S_1$

この正射影の長さは、S_1の長さが１なので、内積$S_2\cdot S_1$で与えられる。したがって、S_2のS_1への正射影のベクトルは、$(S_2\cdot S_1)S_1$で与えられる。このベクトルをS_2から引いたもの$S_2-(S_2\cdot S_1)S_1$は、S_2と直交する。この$S_2-(S_2\cdot S_1)S_1$の長さを１にしたものを新しいS_2とする。ステップ（６ｂ）では、この内積によって正射影を求め、正射影の和との差として直交ベクトルを求めるという手順を繰り返している。

上の手順で極小値を求める手続きMinByRosenbrockをユニットファイルUOptNoDiff.pasに用意した。MinByRosenbrockのヘッダーは、次のように宣言されている。

```
procedure  MinByRosenbrock( f                    //  極小値を求める関数
                              : TOptFunc;
                            n                    //  変数の数
                              : Longint;
                            var  x1              //  変数の配列
                              : TOptVector;
```

```
                crtrn_lambda,       //  変数の変化量の基準
                crtrn_f,            //  関数の変化量の基準
                max_lambda          //  変数の変化量の初期値
                   : Extended );
```

第１パラメータｆに極小値を探索する関数、第２パラメータｎに変数の総数、第３パラメータ x1 に変数を表わす配列を設定する。第１パラメータの型 TOptFunc、第３パラメータの型 TOptVector は、ユニットファイル UDefTypeOpt.pas において

```
TOptVector = array[1..NDimOpt] of Extended;
TOptFunc   = function( x : TOptVector; n : Longint ) : Extended;
```

と宣言されている。

第３パラメータ x1 の配列には、MinByRosenbrock を呼出す前に極値探索の初期値を設定しておく。MinByRosenbrock の実行終了時には、この第３パラメータに探索結果である極小値を与える変数値が返される。第４、第５パラメータには、変数の変化量および関数値の変化量に対する収束判定基準を設定する。変数値の変化量による収束判定は、探索方向S_iにおける変化量（移動量）λ_iの大きさが全て変数の変化量の収束判定基準より小さくなったとき収束したとみなされる。関数値の変化量による収束判定は、S_1からS_nまでの探索にわたる関数値の変化量が相対的に第５パラメータ crtrn_f で与えられた値以下であるか、あるいはS_1からS_nまでの探索前と探索後の関数値の絶対値の和が第５パラメータ crtrn_f 以下であれば収束したとみなされる。変数値の変化量の基準 crtrn_lambda は、１次元探索のホームページにおける解説にあるように、小さすぎる値を設定しても無意味である。MinByRosenbrock では Extended 型で計算が行われているので、Extended 型の有効桁数２０桁の約半分くらいが目安になる。関数値の変化量の基準 crtrn_f の方は Extended 型の有効桁数に合わせた精度でも有効であるが、関数値の計算における計算誤差を考慮して Extended 型の有効桁数より少し下げた値を設定しておけばよい。探索における計算が長時間か掛かる場合で Extended 型ほどの精度が必要でないときは、必要な有効桁数に合わせて Extended 型の有効桁数より少ない桁数に対応した値を設定すると探索が速く終了する。

第６パラメータ max_lambda には、最初の探索におけるステップサイズの初期値を設定する。探索の出発点（第３パラメータ x1）からこれくらい移動すれば関数値が減少すると思われる値を設定する。

手続き MinByRosenbrock]の使用例として、プロジェクト PCheck.dpr では、関数

$$y = f(x_1, x_2) = 100 \cdot (x_2 - x_1^3)^2 + (1 - x_1)^2 \qquad (1)$$

の極小値の探索を行っている。上の関数は、$x_1 = x_2 = 1$のとき最小値０をとる。

プロジェクトのユニットファイル UCheck.pas において、関数（１）を表わす関数 Cube が次のように宣言されている。

```
function Cube( x : TOptVector; n : Longint ) : Extended;
  begin
      Cube:=100*sqr(x[2]-x[1]*sqr(x[1])) + sqr(1-x[1]);
  end;
```

上の関数宣言のもとで、RGOButtonClick の実行部において配列ｘの初期値の設定と MinByRosenbrock の呼び出しがリスト１のように行われている。

リスト１ 手続き MinByRosenbrock の使用例

```
procedure TForm1.RGOButtonClick(Sender: TObject);
var x : TOptVector;
begin
      x[1]:=-1.2;
      x[2]:= 1.0;

      MinByRosenBrock( Cube,  2, x, 1.0e-11, 1.0e-15, 0.01 );

      Label1.Caption:='x1 = '+FloatToStrF(x[1],ffGeneral,9,4)+
                      '    x2 = '+FloatToStrF(x[2],ffGeneral,9,4);

      ExitButton.SetFocus;
end;
```

計算結果は Label1 に設定されるので、フォーム上に表示される（図 3)。

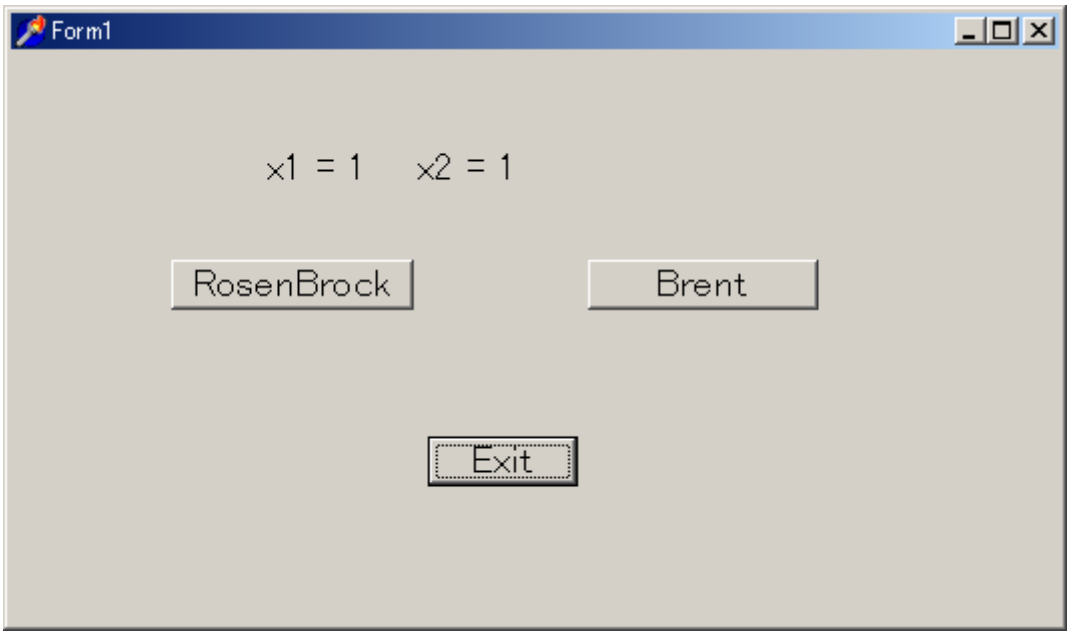

図３ 実行結果の表示

RGOButtonClick は、プロジェクトの実行開始時に表示されるフォーム（図４）上の Rosenbrock ボタンのクリックで呼出される。

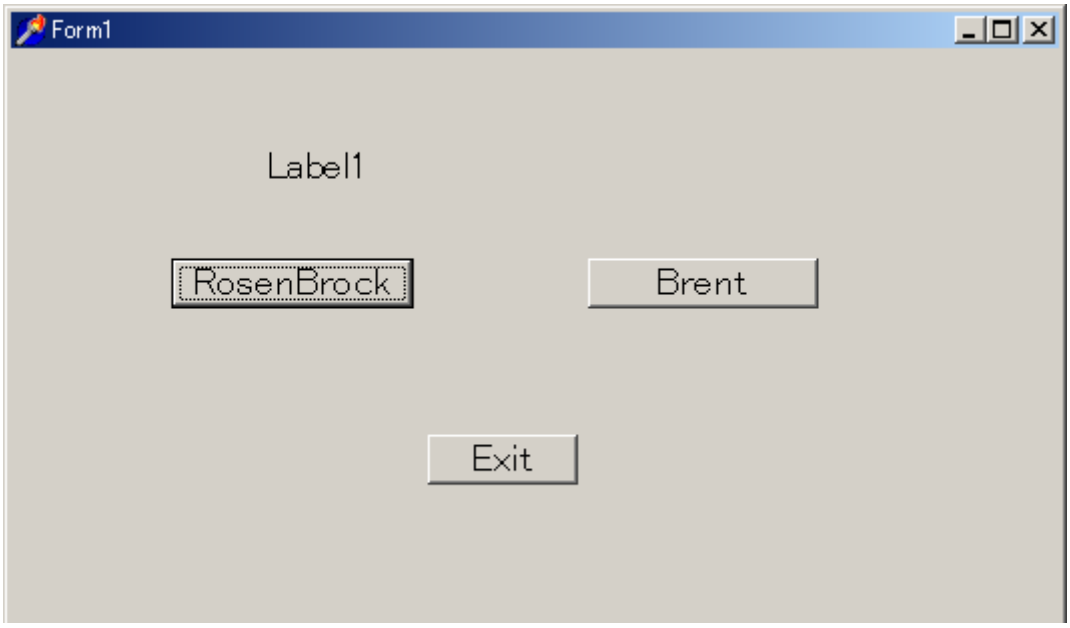

図４ PCheck.dpr の実行開始時のフォーム

手続き MinByRosenbrock を実行すると、探索の途中経過を表示するためのフォームが表示される。この表示を行わないときは、手続きの実行開始時におけるフォームの生成の部

分

```
FOptNoDiff:=TFOptNoDiff.Create(application);
FOptNoDiff.Visible:=true;
```

と実行終了時のフォームの廃棄の部分

```
FOptNoDiff.Close;
```

を、//などを用いてコメントとする。さらに、フォーム上の Memo コンポーネントに表示を行う手続き display の中も次のように

```
procedure display( s : string );
  begin
     ;{ with FOptNoDiff.Memo1.Lines do
         begin
           while count > 20 do Delete(0);
           Add( s );
         end;}
  end;
```

コメントにしておく。これらのコメントとした部分は消去してもよいものであるが、コメントにしておくと探索の途中経過の表示が必要となったときにコメント記号//などを取るだけで表示可能となる。

Brent の方法

Rosenbrock の方法では、探索方向のベクトル、S_1、・・・、S_n、はお互いに直交している。偏導関数を用いた極小値の探索では、探索方向を共役であるようにとることによって探索の効率を高めることが期待されている。偏導関数を用いない探索法において、探索方向を共役であるようにとるものとして Brent の方法[(3)]がある。共役な探索方向は Powell の方法によって作成されるが、Brent の方法では、（１）restart における方向ベクトルの再設定を特異値分解に基づいて行う、（２）谷や尾根における探索の停留から抜け出すためのランダム・ステップを導入する、（３）３点を通る２次曲線による極小点の探索を行う、などの工夫が加えられている。

ここでの Brent の方法に基づく極値探索法の概略は、以下のようである。

（1） $istep \leftarrow 1$、$NRandomStep \leftarrow 0$ とおく。座標値の初期値を $\mathbf{x}_0$ に設定する

（2） $NRandomStep > 0$ であれば、$\mathbf{x}_0$ にランダムな変動を加える。

$\mathbf{x}_s \leftarrow \mathbf{x}_0$ とおく。

$istep = 1$ のときは、探索方向、$\mathbf{S}_1$、・・・、$\mathbf{S}_n$、を座標軸の方向にとる。

$istep = 1$ でないときは、探索方向、$\mathbf{S}_1$、・・・、$\mathbf{S}_n$、を（2 a）で与える。

（2 a） $\mathbf{U} \leftarrow \left[\mathbf{S}_1 \quad \cdots \quad \mathbf{S}_n\right]$

$d_i \leftarrow f_i\left[\alpha_0, \alpha_1, \alpha_2\right]$；2 階の差分商

とおく。ここで、$f_i(\alpha) = f(\mathbf{x}_0 + \alpha \mathbf{S}_i)$ である。

$\mathbf{S}_i \leftarrow \mathbf{U} diag\left(d_1 \quad \cdots \quad d_n\right)^{-1/2}$ の i 番目の特異ベクトル

とおく。

（3） $\beta^* \leftarrow \arg\min_{\beta} f(\mathbf{x}_0 + \beta \mathbf{S}_n)$

$\mathbf{x}_0 \leftarrow \mathbf{x}_0 + \beta^* \mathbf{S}_n$

$k \leftarrow 1$

とおく。

（4） $\mathbf{x}_1 \leftarrow \mathbf{x}_0$

$i \leftarrow 1$

とおく。

（5） $\beta^* \leftarrow \arg\min_{\beta} f(\mathbf{x}_1 + \beta \mathbf{S}_i)$

$\mathbf{x}_1 \leftarrow \mathbf{x}_1 + \beta^* \mathbf{S}_i$

とおく。

$i < n$ のときは、$\mathbf{S}_i \leftarrow \mathbf{S}_{i+1}$ とおく。

$i = n$ のときは、$\mathbf{S}_n \leftarrow \mathbf{x}_1 - \mathbf{x}_0$ とおく。

$i \leftarrow i + 1$ とおく。

$i \le n$ ならば、（5）に戻る。

$\mathbf{x}_1 \neq \mathbf{x}_0$ かつ $NRandomStep = 0$ ならば（6）、そうでないならば（7）に進む。

（6） $\beta^* \leftarrow \arg\min_{\beta} f(\mathbf{x}_0 + \beta \mathbf{S}_n)$

$\mathbf{x}_1 \leftarrow \mathbf{x}_0 + \beta^* \mathbf{S}_n$

とおき、（8）に進む。

（7）$istep \leftarrow 1$

$NRandomStep \leftarrow 1$

とおき、（2）に戻る。

（8）$\mathbf{x}_0 \leftarrow \mathbf{x}_1$

$k \leftarrow k+1$

とおく。

$k \le n$ ならば、（4）に戻る。

（9）$istep \le 2$ ならば

$\mathbf{x}_{p,istep} \leftarrow \mathbf{x}_0$

$istep \leftarrow istep+1$

とおき、（１０）に進む。

$istep = 3$ならば、

$\mathbf{x}_{p,istep} \leftarrow \mathbf{x}_0$

とおく。

３点、$\mathbf{x}_{p,1}$、$\mathbf{x}_{p,2}$、$\mathbf{x}_{p,3}$、がお互いに異なる点であれば、これらの３点を通る

2 次曲線上で極小化を行う。

３点、$\mathbf{x}_{p,1}$、$\mathbf{x}_{p,2}$、$\mathbf{x}_{p,3}$、がお互いに異なる点でないときは

$istep \leftarrow 1$

とおく。

（１０）「$\mathbf{x}_s = \mathbf{x}_0$ または $f(\mathbf{x}_s) = f(\mathbf{x}_0)$」であれば、この極値探索を終了する。

「$\mathbf{x}_s = \mathbf{x}_0$ または $f(\mathbf{x}_s) = f(\mathbf{x}_0)$」でないときは、（2）に戻る。

上のような手順で極小値の探索を行う手続き MinByBrent をユニットファイル UOptNoDiff.pas に用意した。MinByBrent のヘッダーは、次のように宣言されている。

```
procedure MinByBrent( f              //  極小値を求める関数
                        : TOptFunc;
                      n              //  変数の数
                        : Longint;
                      var  x0        //  変数の配列
                        : TOptVector;
```

```
    criterionX,      //  変数の変化量の基準
    criterionF       //  関数の変化量の基準
      : Extended );
```

第１パラメータｆに極小値を求める関数、第２パラメータｎに変数の数、第３パラメータｘ０に初期値を格納した変数用の配列を設定する。極値探索で得られた極小値を与える変数値は、この第３パラメータの配列に返される。第４パラメータ criterionX と第５パラメータ criterionF には、極小値に達したかどうかの判定に用いる変数値と関数値の変化量の基準値を設定する。これらの基準値は、相対量および値の絶対値の変化量についてのものである。例えば、基準値をｃとし、探索によって値が v1 から v2 に変わったとするとき、

$$|v1 - v2| < c\left(|v1| + |v2|\right) \qquad \text{（相対値）}$$

または

$$|v1| + |v2| < c \qquad \text{（絶対値）}$$

が成り立てば、極小値に達したと判定している。相対値による基準は精度による判定である。しかし、値 v1 あるいは v2 が０に収束しているときは、この相対値による基準では収束したと判定されずに限りなく絶対値が小さくなっていき、アンダーフローエラーになる可能性がある。この場合は、絶対値による基準で判定を行う。

手続き MinByBrent によって極小値を求めるときも、MinByRosenbrock を用いる場合と同じように行う。（１）式の極小値を求めるときは、関数 Cube を MinByRosenbrock を用いたときのように宣言しておき、

```
x[1]:=-1.2;
x[2]:= 1.0;
MinByBrent( Cube, 2, x, 1.0e-11, 1.0e-15 );
```

というように MinByBrent を呼出す。

手続き MinByRosenbrock の使用例であるプロジェクト PCheck.dpr では、MinByBrent を用いた上の極小値を求める例も扱われている。PCheck.dpr の実行開始時のフォーム（図４）において、「Brent」ボタンをクリックするとリスト２の手続き BGOButtonClick が実行されて MinByBrent による極小値の探索が行われる。求められた極小値を与える変数値は、図３のように表示される。

リスト２　手続き MinByBrent の使用例

```
procedure TForm1.BGOButtonClick(Sender: TObject);
var x : TOptVector;
begin
      x[1]:=-1.2;
      x[2]:= 1.0;

      MinByBrent( Cube, 2, x, 1.0e-11, 1.0e-15 );

      Label1.Caption:='x1 = '+FloatToStrF(x[1],ffGeneral,9,4)+
                      '    x2 = '+FloatToStrF(x[2],ffGeneral,9,4);

      ExitButton.SetFocus;
end;
```

中心極限定理

一様分布をする乱数の独立なｎ個の和は、中心極限定理[4]により、ｎを大きくすると正規分布に近づくことがわかる。このことを、シミュレーションによって見てみることにする。

ｎ個の独立な一様乱数の和を多数生成して、その度数分布を最もよく表わす正規分布を極値探索法によって求め、その正規分布のグラフを生成した乱数の度数分布に重ねて描いてみる（図５）。ｎの値によって、正規分布のグラフと乱数の度数分布との重なり具合がどのように変わるのか調べてみる。

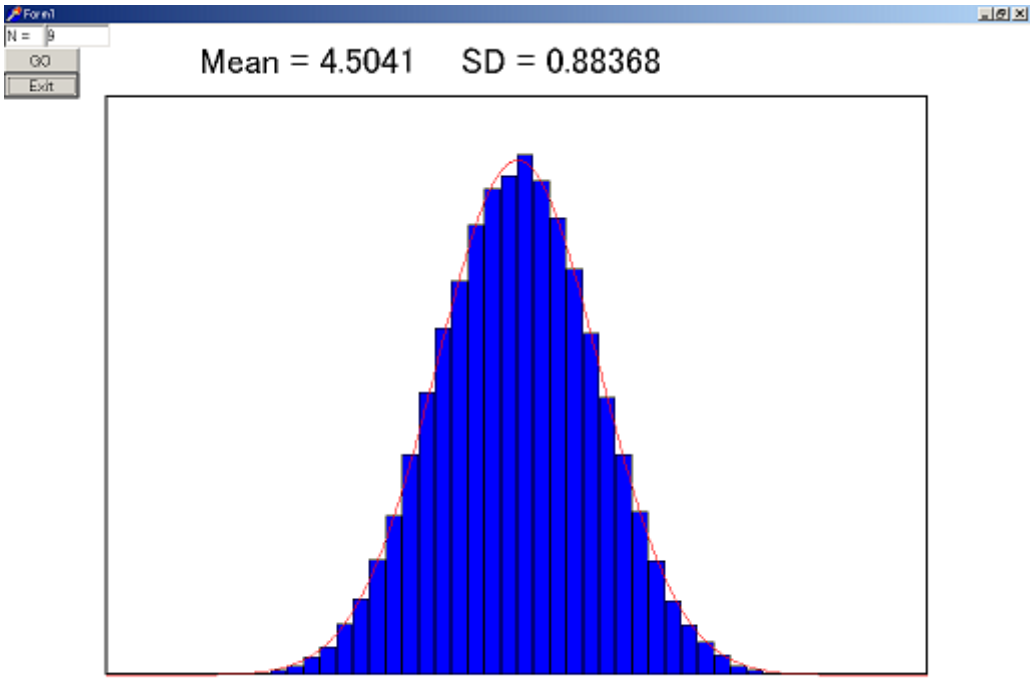

図５　一様乱数９個の和の度数分布と正規分布。棒グラフが度数分布を、釣鐘形の曲線が正規分布を表わす。

一様分布に従う独立なｎ個の確率変数を X_1、・・・、X_n で表わし、そのｎ個の和を Z_n で表わす。

$$0 \le X_i \le 1 \; ; \quad i = 1, \cdots, n$$

$$Z_n = X_1 + \cdots + X_n$$

なので、

$$0 \le Z_n \le n$$

となる。

いま、生成した Z_n の値を、区間 $[0, n]$ を NCat 個に等分割したカテゴリーに分けて数える。各カテゴリの幅 d は、

$$d = \frac{n}{NCat}$$

で与えられる。

生成した乱数 Z_n の総数を N、N 個の乱数のうち i 番目のカテゴリ $[(i-1)d, id]$ に分けられたものの数を f_i で表わすと、その相対頻度 p_i は

$$p_i = \frac{f_i}{N}$$

で与えられる。

平均μ、標準偏差σの正規分布を$N(x;\mu,\sigma)$で、その確率変数を$V_{\mu,\sigma}$で表わすと、$V_{\mu,\sigma}$がカテゴリ$[(i-1)d, id]$に入る確率$\Pr ob_i$は、dが十分に小さいとき次式で近似できる。

$$\Pr ob_i \approx d \cdot N((i-0.5)d, \mu, \sigma)$$

したがって、n個の一様乱数の和の分布を最もよく表わす正規分布は、p_iと$\Pr ob_i$ができるだけ近い値になるものとして求めることができる。p_iと$\Pr ob_i$の差を全体として表わすものとして$SSE(\mu,\sigma)$を次式で与える。

$$SSE(\mu,\sigma) = \sum_{i=1}^{n} \left(p_i - \Pr ob_i\right)^2$$

$$= \sum_{i=1}^{n} \left(\frac{f_i}{N} - d \cdot N((i-0.5)d, \mu, \sigma) \right)^2$$

ここで、$SSE(\mu,\sigma)$におけるμとσは、上式がμとσの関数であることを表す。

n個の一様乱数の和Z_nの分布を最もよく表わす正規分布を与える平均μと標準偏差σの値を、$SSE(\mu,\sigma)$の最小値を与えるものとして求める。

一様乱数の和Z_nの分布を調べるプロジェクトPCentral.dprでは、$SSE(\mu,\sigma)$の最小値を求めるとき、標準偏差σに正数という制約があるので次のように変数変換を行っている。

$$\sigma = 0.001 + x^2$$

すなわち、リスト3に示されているように、関数OptFuncを配列xの関数として与え、

$$\mu = x_1、 \quad \sigma = 0.001 + x_2^2$$

と変数変換を行ってから$SSE(\mu,\sigma)$を呼出している。

リスト３　$SSE(\mu,\sigma)$の最小値を与える$\sigma > 0$を求めるための関数OptFunc。

```
//    正規分布関数
function Normal( x, mean, sigma : Extended ) : Extended;
  begin
        Normal:=(1/(sigma*sqrt(2*pi)))*exp(-0.5*sqr((x-mean)/sigma))
  end;
```

```
//    度数分布と正規分布の差の２乗和
function SSE( mean, sigma : Extended ) : Extended;
  var d, v : Extended;
      i    : Longint;
  begin
      d:=N/NCat;
      v:=0.0;
      for i:=1 to NCat do
        v:=v+sqr((Freq[i]/NTrial)-(d*Normal((i-0.5)*N/NCat,mean,sigma)));
      SSE:=v;
  end;

//    SSE を関数値とする極値探索用の関数
function OptFunc( x : TOptVector; h : Longint ) : Extended;
  var mean, sigma : Extended;
  begin
        mean:=x[1];
        sigma:=0.001+sqr(x[2]);
        OptFunc:=SSE(mean,sigma);
  end;
```

極小値を求める手続きは、MinByRosenbrock、あるいは MinByBrent のいずれでも行えるようになっているが、用いない方の手続きはコメントとしておく。

極小値の探索手続きを呼出す前に、変数用配列に初期値を次のように与えている。

```
x[1]:=N/2.1;
x[2]:=1.0;
```

x[1]の値は理論的解 $\mu = n/2$ に近い値としている。これは、初期値が最小値を与える変数値から大きく離れていると探索がうまくいかないことがあるからである。極値探索によって最小値を求めるとき、最小値を与える変数値に近い値を初期値として与えると探索が成功し易くなる。最小値を与える変数値に近い初期値の見当を付けることが理論的に難しいときは、変数の変域をおおまかに格子で区切って、各格子点の値を比較することによって大体の見当を付けることができる。

また、MinByRosenbrock および MinByBrent では、極小値を与える変数値が絶対値で 0.01～1000 ぐらいの大きさであることを想定している。絶対値が非常に小さいときは変数の収束判定基準として crtrn_lambda あるいは criterionX に設定した値が有効に働かない可能性がある。求まった極小値を与える変数ｘの値が 0.000001 など非常に小さいときは、変数変換によって 1.0 ぐらいの値になるようにする。上の例の場合、x＝0.000001×ｔと変換すると、極小値を与える変数ｔの値は 1.0 になる。

PCenter.dpr を実行すると、図６のようなフォームが表示される。

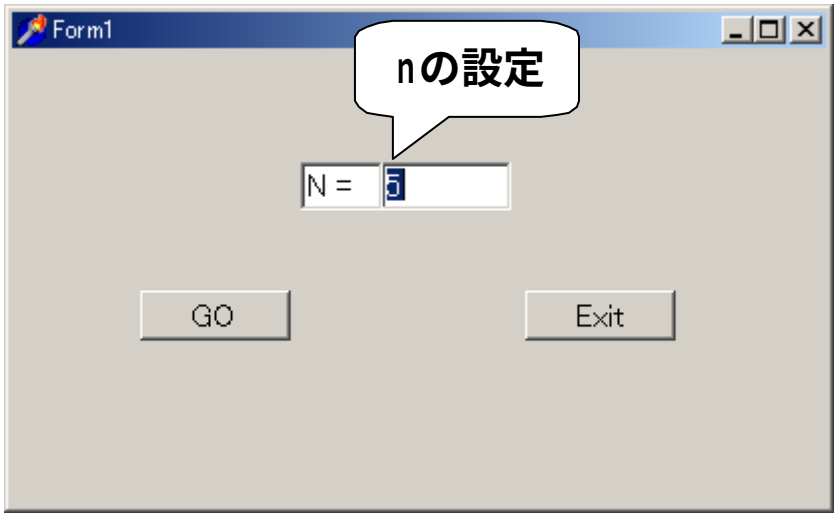

図６　和をとる乱数の個数ｎを設定してから
GO ボタンをクリック

「N=」の右側のエディットコンポーネントに和をとるときの一様乱数の個数ｎを設定する。「GO」ボタンのクリックで、設定した個数の和 Z_n が NTrial（= 100000）個生成され、その度数分布を最も良く近似する正規分布が求められて図７のように描画される。

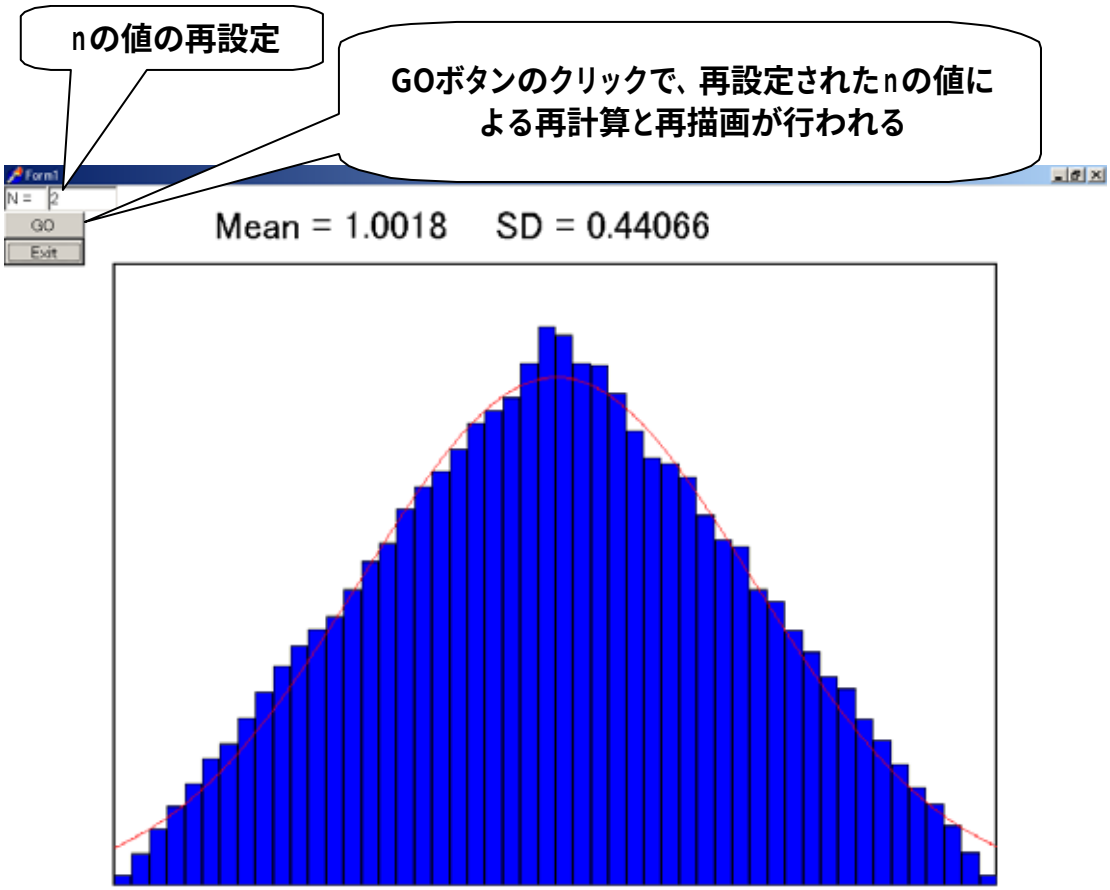

図７　ｎ＝２に設定して再描画

ただし、図７は、ｎの値を２に再設定した後、GO ボタンのクリックで再計算・描画したものである。ｎ＝２のときは、生成した乱数 Z_n は二等辺三角形の形に分布するが、正規分布との差異は顕著ではない。ｎ＝９のときの結果は図５に示されているが、この場合の乱数 Z_n の分布は釣鐘形になっていて、見た目には正規分布との区別がつかない。

プロジェクト PCenter.dpr のユニット UCenter で使用されているユニット URN は、乱数発生のためのユニットである。このユニットの一様乱数は、合同法によって生成されるものである[4]。

参 考 文 献

（1）S.S.Rao. Optimization: Theory and Applications (Second Edition). Pp.747, John Wiley & Sons, 1984.
（2）M.J.Box, D.Davies and W.H.Swann. Non-linear optimization techniques. Pp.60, Imperial Chemical Industries Limited, 1969.
（3）R.P.Brent. Algorithms for minimization without derivatives. Pp.195, Prentice-Hall,Inc, 1973.
（4）岡本安晴 「Delphi で学ぶデータ分析法」、Pp.275、ＣＱ出版社. 1998.