岡本安晴　2001.2.3;2001.8

方程式の根の計算[1]

方程式 $f(x)=c$ の根を求めるアルゴリズムについて説明する。まず、解の存在区間がわかっていて、その区間内で関数が単調関数である場合の方法として Bisection 法をとりあげる。導関数が与えられている場合には、Bisection 法より効率の良いニュートン法（Newton－Raphson 法）という方法が使える。導関数が与えられていないときは、微分係数の値を適当に推定することにより、ニュートン法を適用することができる。以上、３つの方法を順番に説明する。

Bisection 法

関数 $f(x)$ が、区間 $[a,b]$ において単調増加であって、$f(a)<c$ かつ $c<f(b)$ であるとする。

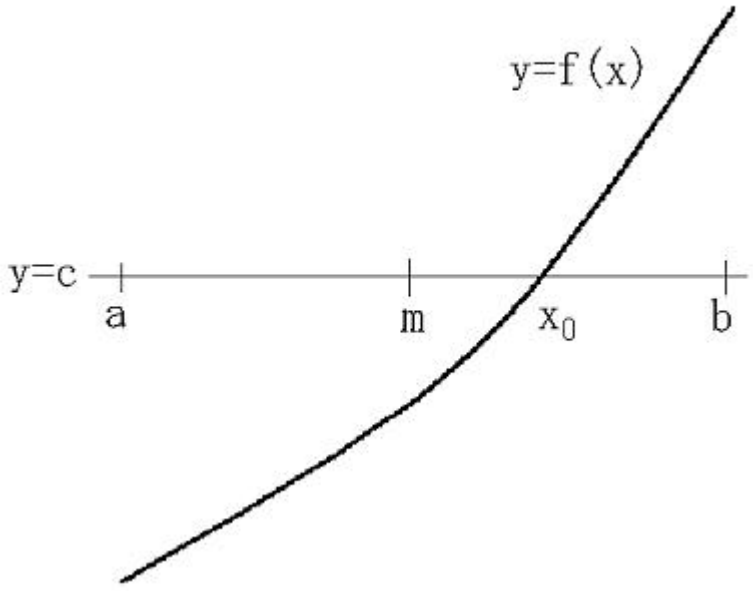

図 1.a　$f(m)<c$ のときは次の区間を $[m,b]$ とする、
すなわち m をあらためて a とおく。

[1] 本解説は、TRY！PC、2000 年７月号「Delphi による数値計算プログラミングのすすめ：第５回多重精度算術演算・方程式の根の計算」の原稿をもとにしたものである。

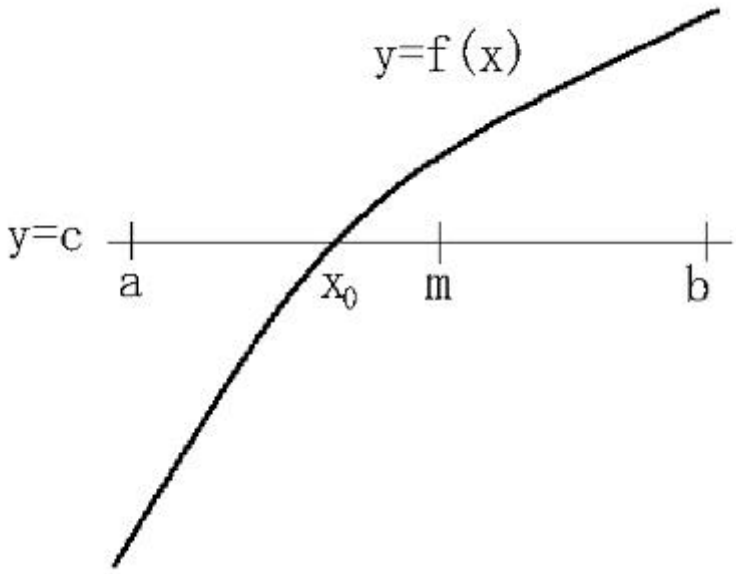

図1.b　$f(m)>c$ のときは次の区間を$[a,m]$とする、
すなわち m をあらためて b とおく。

このとき、$f(x_0)=c$ となる x の値 x_0 を、x_0 の存在区間の２分割の繰り返しによって求めることができる[1),2)]。すなわち、次のように分割を繰り返す。

（1）$m=\dfrac{a+b}{2}$ とおいて、区間$[a,b]$を$[a,m]$と$[m,b]$に分割する。

（2）（2．1）$f(m)<c$ なら根 x_0 は区間$[m,b]$にある（図1.a）。$a=m$ とおく。

（2．2）$f(m)=c$ なら $x_0=m$ とおいて終了。

（2．3）$f(m)>c$ なら根 x_0 は区間$[a,m]$にある（図1.b）。$b=m$ とおく。

（3）（3．1）a が b に十分近いときは、$a\approx b$ であり、$a\le x_0\le b$ なので、

$x_0=\dfrac{a+b}{2}$ とおいて終了。

（3．2）a が b に十分近くないときは、（1）に戻る。

上の手順を繰り返すと、１回の繰り返しで $b-a$ の大きさは $1/2$ になる。
いま、次式

$$b-a<10^{-m}\cdot|a+b| \qquad (1)$$

が成り立つまで、区間$[a,b]$の分割を繰り返したとする。このとき、根の推定値の精度は次のように評価できる。

上の（１）～（３）の手順の繰り返しで求めた根の推定値を$\hat{x}_0$で、根の真の値をx_0で表わすと、手順（２．２）において終了した場合を除いて、

$$a \le x_0 \le b \qquad (2)$$

かつ

$$\hat{x}_0 = \frac{a+b}{2} \qquad (3)$$

となる。

式（３）は、手順（2.2）によって根$\hat{x}_0$が求められた場合は成り立たない。この場合は、計算機での計算誤差の範囲内で、$\hat{x}_0$は真の値x_0に一致している。したがって、以下の$\hat{x}_0$とx_0の差を見る議論では除いて考える。

式（１）、（２）および（３）より次式

$$\left|x_0 - \hat{x}_0\right| < 10^{-m} \cdot \left|\hat{x}_0\right|$$

が成り立つ。

すなわち、算出した根の値$\hat{x}_0$のほぼm桁目までの精度で、根の真の値x_0が求められていることになる。

上の Bisection 法によって根を求める手続き Bisection をユニットファイル UCalcRoot.pas に用意した。手続きヘッダーは、次のようになっている。

```
procedure  Bisection( f : TFunc;
                      s,          //   f が増加関数のとき s > 0.0
                      c,          //   根に対する関数値
                      L_b, U_b,   //   根の存在区間の下限と上限
                      acc,        //   精度
                      zero        //   ゼロとみなす基準値
                                  : Extended;
                      var  Root   //   根  f(Root) = c
                                  : Extended );
```

第１パラメータ f に、根を求める関数を設定する。型 TFunc は

```
type  TFunc = function( x : Extended ) : Extended;
```

と宣言されている。

第２パラメータｓは、ｆが根を求める区間[a,b]で単調増加のときは正の値、例えば＋１、を、単調減少のときは負の値、例えば－１、を設定する。

第３パラメータｃは、根x_0に対応する関数値$f(x_0)=c$を設定する。

第４，５パラメータは、根x_0の存在区間[a,b]、$a<x_0<b$、の下限aと上限bの値を設定する。区間[a,b]において関数が単調関数であるようにaとbを選ぶ。

第６，７パラメータには、求める根x_0の値の精度と、計算上ゼロとみなす基準値を設定する。

最後のパラメータ Root には、求められた根$\hat{x}_0$の値が返される。

手続き Bisection を用いて８の３乗根を求めるプログラム例 PBisection.dpr を用意した。このプログラムの実行で表示されるフォーム上のＧＯボタンをクリックすると、手続き Bisection による３乗根の計算が始まる。

３乗根を求めるために、３乗を計算する関数が

```
function Power3( x : Extended ) : Extended;
  begin
          Power3 := x * sqr(x);
  end;
```

と宣言されている。この関数に対して、手続き Bisection を

```
Bisection( Power3, 1.0, 8.0, 1.0, 10.0, 1.0e-17, 1.0e-19, v );
```

と呼出して、８の３乗根を求めている。手続き Bisection を呼出すためには、手続き Bisection の宣言されているユニット UCalcRoot の使用を uses 節において宣言しておく。

求められた３乗根は、フォーム上のラベルコンポーネント Label1 の Caption コンポーネントに設定表示される(図 2)。

図２　８の３乗根の PBisection.dpr による計算結果

ユニット UCalcRoot のフォーム（フォームファイル UCalcRoot.dfm のフォーム）は、Bisection の実行時に表示することができる。このフォームには、Memo コンポーネントが貼り付けてある。Bisection の実行中における中間結果が、この Memo コンポーネントに表示される。Bisection による計算がなかなか終わらないときは、この Memo コンポーネントに計算の途中経過を表示させることによって、計算が終了しない原因を調べることができる。このフォームを表示すると、計算における処理時間が Memo コンポーネントの処理のため余計に掛かるので、普通はフォームが表示されないようにしておいて Bisection を呼出す。アップロードしてあるユニットファイル UCalcRoot.pas では、フォームの生成・廃棄および Memo コンポーネントへの表示に関わるところは//を付けたり{ }で囲んだりしてコメントとしてある。これらの//や{ }をはずすと、Bisection の呼び出しでフォームが表示され、Memo コンポーネントに計算の途中経過が表示される。

ニュートン法

Bisection 法においては、根の精度は１回の繰り返しで1/2になる、すなわち２進数表記で１桁精度が上がる。しかし、関数の導関数が与えられているときは、より速く精度を上げることが可能である。この導関数を利用する方法として、ニュートン法[1)、3)、2)]について説明する。

ニュートン法では、根 x_0 の近くの値 x_1 が与えられたとき、x_1 よりよい根の推定値 x_2 を、点 $(x_1, f(x_1))$ を通る接線を用いて求める（図３）。

点 $(x_1, f(x_1))$ を通る接線は

$$y = f(x_1) + (x - x_1) \cdot f'(x_1)$$

で与えられる。この接線と直線

$$y = c$$

との交点のｘ座標 x_2 を新しい根の推定値とする。接線の式に点 (x_2, c) を代入すると次

式を得る

$$c = f(x_1) + (x_2 - x_1) \cdot f'(x_1)$$

上式をx_2について解くと

$$x_2 = x_1 + \frac{c - f(x_1)}{f'(x_1)} \qquad (4)$$

となる。

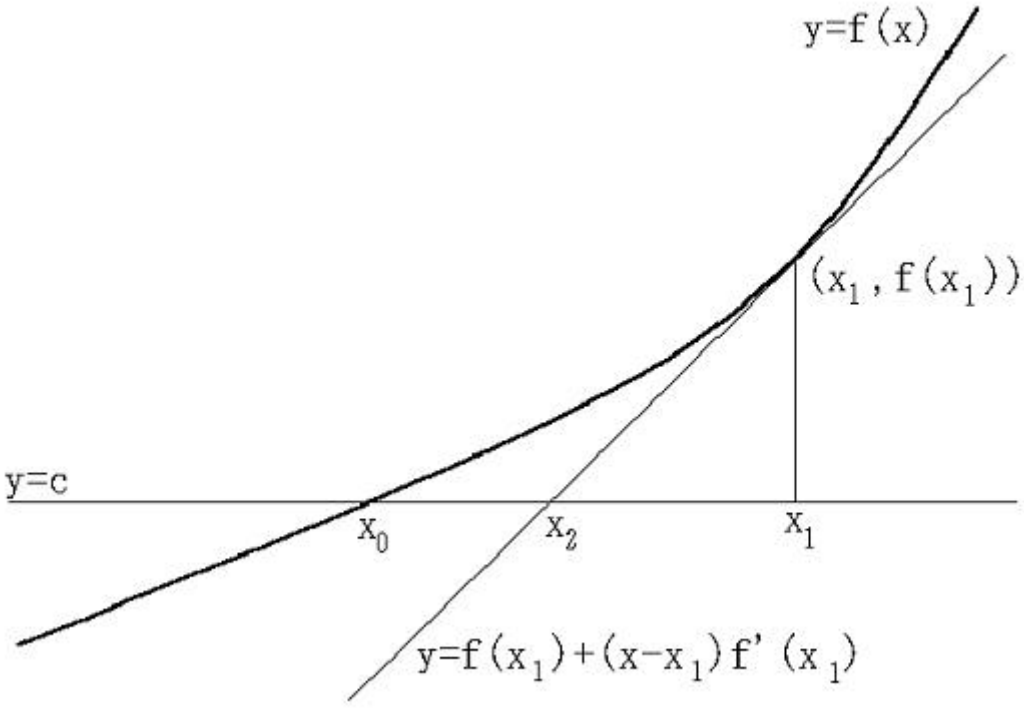

図３　接線$y = f(x_1) + (x - x_1) \cdot f'(x_1)$による根の推定値の更新

ニュートン法は、（４）式によって根の推定値の更新を行うもので、以下のような手順になる。

（１）現在の根の推定値x_1から次式

$$x_2 = x_1 + \frac{c - f(x_1)}{f'(x_1)}$$

により新しい推定値x_2をもとめる。

（２）（２．１）x_2がx_1に十分近ければ、x_2を根の値として終了する。

（２．２）x_2がx_1に十分近くないときは、x_2を現在の根の推定値x_1として（１）に戻る。

上の手順における１回の更新で、根の推定値の精度がどれぐらい上がるのか調べてみる。

関数 $f(x)$ のテイラー展開を次のようにおく[1)]。

$$f(x)=f(x_1)+(x-x_1)f'(x_1)+\frac{(x-x_1)^2}{2}f''(x(x))$$

ここで、$x(x)$ は x_1 と x の間にある数である。

上式において $x=x_0$ とおくと

$$c=f(x_0)=f(x_1)+(x_0-x_1)f'(x_1)+\frac{(x_0-x_1)^2}{2}f''(x(x_0)) \qquad (5)$$

となる。

また、点 (x_2,c) を接線の方程式に代入して、

$$c=f(x_1)+(x_2-x_1)f'(x_1) \qquad (6)$$

となる。

（５）式から（６）式を引くと、次式が得られる。

$$0=(x_0-x_2)f'(x_1)+\frac{(x_0-x_1)^2}{2}f''(x(x_0))$$

上式を変形して、

$$\frac{|x_0-x_2|}{|x_0|}=\left(\frac{|x_0-x_1|}{|x_0|}\right)^2\cdot\left|\frac{x_0\cdot f''(x(x_0))}{2\cdot f'(x_1)}\right| \qquad (7)$$

となる。上式は、誤差が２乗のオーダーで減少（２乗収束[3)]）することを表わしている。

いま、x_1 の精度が、次式の意味でm桁であるとする。

$$\frac{|x_0-x_1|}{|x_0|}\approx 10^{-m}$$

このとき、（７）式において $\left|x_0\cdot f''(x(x_0))/(2\cdot f'(x_1))\right|$ の項を無視すると、

$$\frac{|x_0-x_2|}{|x_0|}\approx 10^{-2m}$$

となる。ニュートン法では、１回の繰り返しで精度が２倍になることが期待される。

Bisection 法の場合は、根の存在区間 $[a_1,b_1]$ の長さが

$$b_1-a_1\approx 10^{-m}$$

のとき、次（更新後）の存在区間 $[a_2,b_2]$ の長さは

$$b_2 - a_2 \approx \frac{1}{2}\cdot 10^{-m} \approx 10^{-(m+0.3)}$$

となるので、１回の繰り返し（更新）による精度の向上は約 0.3 桁と考えられる。

以上のことより、ニュートン法の効率の良さがわかる。

手続き Newton はニュートン法によって根を求めるもので、ユニットファイル UCalcRoot.pas に宣言されている。関数ヘッダーは次のようになっている。

```
procedure Newton( var x : Extended;  //  根の初期値、戻り値は根 f(x)=c
                  c,                 //  関数値
                  L_b, U_b           //  根の存在範囲の下限と上限
                        : Extended;
                  f,                 //  関数
                  df                 //  導関数
                        : TFunc;
                  acc,               //  精度
                  zero               //  ゼロの基準値
                        : Extended );
```

第１パラメータには、根の推定値の初期値を設定した変数をおく。Newton の実行終了後、算出された根の値がこの変数に返される。

第２パラメータには、根 x_0 に対する関数値 $c = f(x_0)$ を設定する。

第３、４パラメータには、根の存在範囲の下限と上限を設定する。これらの値は、（４）式による解の更新値がこれらの下限と上限の範囲を超えないようにチェックするのに用いられている。

第５，６パラメータには、根を求める関数とその導関数を設定する。

第７，８パラメータには、求める根の精度とゼロとみなす基準値を設定します。

手続き Newton によって８の３乗根を求めるプログラム例が PNewton.dpr である。このプロジェクトのユニット UNewton.pas では、手続き Newton を用いるために、uses 節でユニット UCalcRoot の使用が宣言されている。また、３乗を与える関数とその導関数は、次のように宣言されている。

```
function Power3( x : Extended ) : Extended;
  begin
        Power3:=x*sqr(x);
  end;
```

```
function DPower3( x : Extended ) : Extended;
  begin
        DPower3:=3*sqr(x);
  end;
```

上の宣言に基づいて、８の３乗根を求めるために手続き Newton が次のように呼出されている。

```
x:=1.0;
Newton( x, 8.0, 0.0, 10.0, Power3, DPower3, 1.0e-17, 1.0e-19 );
```

求めた根の値は、Label コンポーネントの Caption プロパティに設定・表示される。

プロジェクトファイル PNewton.dpr の実行開始時に表示されるフォームのＧＯボタンをクリックすると計算が始まる。

ユニット UCalcRoot の手続き Newton では、手続き（１）～（２）の繰り返しが２０回を超えると Bisection 法を呼出すようになっている。これは、ニュートン法において同じ数値の組み合わせが交互に現れることがあるからである（図４）。

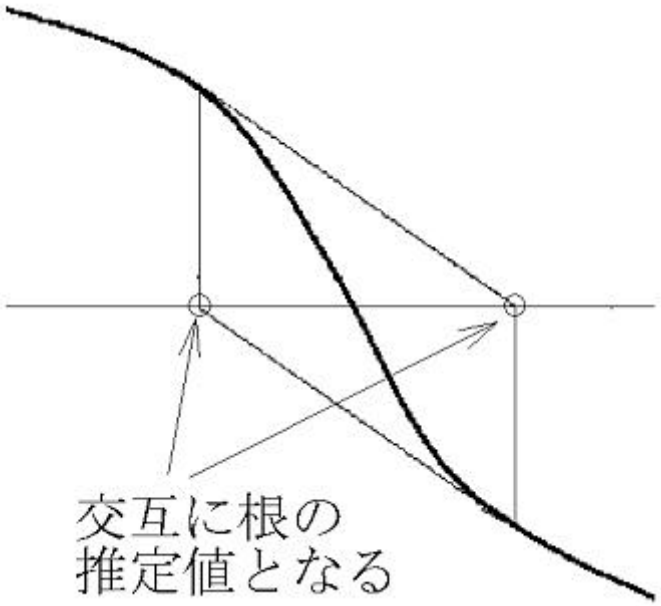

図４ ニュートン法において一組の数値が交互に根の推定値となる場合

例えば、関数

$$f(x)=\frac{1}{1+e^{x}}-\frac{1}{1+e^{-x}}$$

の場合、$f(x)=0$に対する根$x=0$を求めるとき、図４のような現象になる。図４のような場合でも、根を含む区間内で関数が単調であれば、手続き Bisection を用いて確実に根を求めることができる。

このことを示すプロジェクトを PNewtonE.dpr として用意した。このプロジェクトでは、ニュートン法で根を求める手続き Newton は UCkNewton.pas で宣言されているものを用いている。この Newton 法では、Memo コンポーネントをもつフォームが表示されて、計算の途中結果（根の推定値）が Memo コンポーネントに表示される。ニュートン法における１００回の繰り返しの後、ShowMessage が呼び出されてプログラムの実行が一時停止する。この一時停止中に、Memo コンポーネントに表示されている値を確認することができる。-33.3･･･と 33.3･･･が交互に現れていて、図４の状態になっていることがわかる。ShowMessage で表示されるダイアログボックスが邪魔でよく見えないときは、ダイアログボックスをドラッグして移動する。ダイアログボックスのＯＫボタンをクリックするとプログラムの実行が再び始まり、今度は Bisection が呼出され、計算の途中経過が Memo コンポーネントに表示される。Bisection 法によって根が求まると、Memo コンポーネントをもつ UCkNewton.dfm のフォームは閉じられて、メインの UNewtonE.dfm のフォームに求めた根の値が表示される。

Secant 法

ニュートン法は、成功したときは bisection 法より効率がいいが、導関数が必要である。Secant 法では、手順の繰り返しで得られる根の２つの推定値x_1とx_2、およびその関数値$f(x_1)$と$f(x_2)$を用いて導関数の値の推定を行う。すなわち、以下のような手順になる。

（１）根の 2 つの推定値をx_1とx_2とする。

（２）導関数の値$f'(x_2)$を

$$df = \frac{f(x_2)-f(x_1)}{x_2-x_1}$$

で推定する。

（３）$c=f(x)$の根の推定値を

$$x_1 \leftarrow x_2 \quad \text{および} \quad x_2 \leftarrow x_2 + \frac{c-f(x_2)}{df}$$

で更新する。記号 ← は、記号の右側の値を左側に設定することを表わす。

（４）(4.1) x_2がx_1に十分に近い値のときは、x_2を根の推定値として終了する。

(4.2) x_2がx_1に十分に近い値でないときは、（２）に戻る。

ユニットファイル UCalcRoot.pas に宣言されている手続き Secant は、上の secant 法

の手順で根を求めるものである。手続きヘッダーは

```
procedure Secant( var x : Extended;   //  根の初期値、戻り値は根 f(x)=c
                  c,                  //  関数値
                  L_b, U_b            //  根の存在範囲の下限と上限
                        : Extended;
                  f     : TFunc;      //  関数
                  acc,                //  精度
                  zero                //  ゼロの基準値
                        : Extended );
```

となっている。手続き Newton の場合と比べると、導関数を設定するためのパラメータ df がないことを除いて Newton と同じである。

プロジェクト PSecant.dpr は、手続き Secant を用いて８の３乗根を求めるものである。実行開始時のフォームのＧＯボタンのクリックで計算が始まり、求められた根はフォームに表示される。

偏差値（T得点）の計算

根の計算の応用例として、偏差値を求めるプログラムを作成した。

偏差値（T得点）とは、パーセンタイル値$100p$に対して、

$$p = \int_{-\infty}^{z} f(x)dx$$

となる値zから算出される値

$$T = 50 + 10z$$

のことである[4)]。ここで、$f(x)$は標準正規分布を表わす。また、パーセンタイル値は、得点を最下位から数えて何パーセントの位置にあるのかを表わすものである。最下位から 1/4 の位置にある得点は 25 パーセンタイルの位置にあり、3/4 の位置にある得点のパーセンタイル値は 75 である。

上式のpはzの関数になっていて、その導関数は次式で与えられる。

$$\frac{dp}{dz} = f(z)$$

導関数が与えられているので、偏差値を求めるプロジェクト PTscore.dpr では、手続き Newton を用いてpに対するzの値を求めている。

プロジェクト PTscore.dpr の実行で表示されるフォーム上の GO ボタンをクリックすると、図５のようなフォームになる。

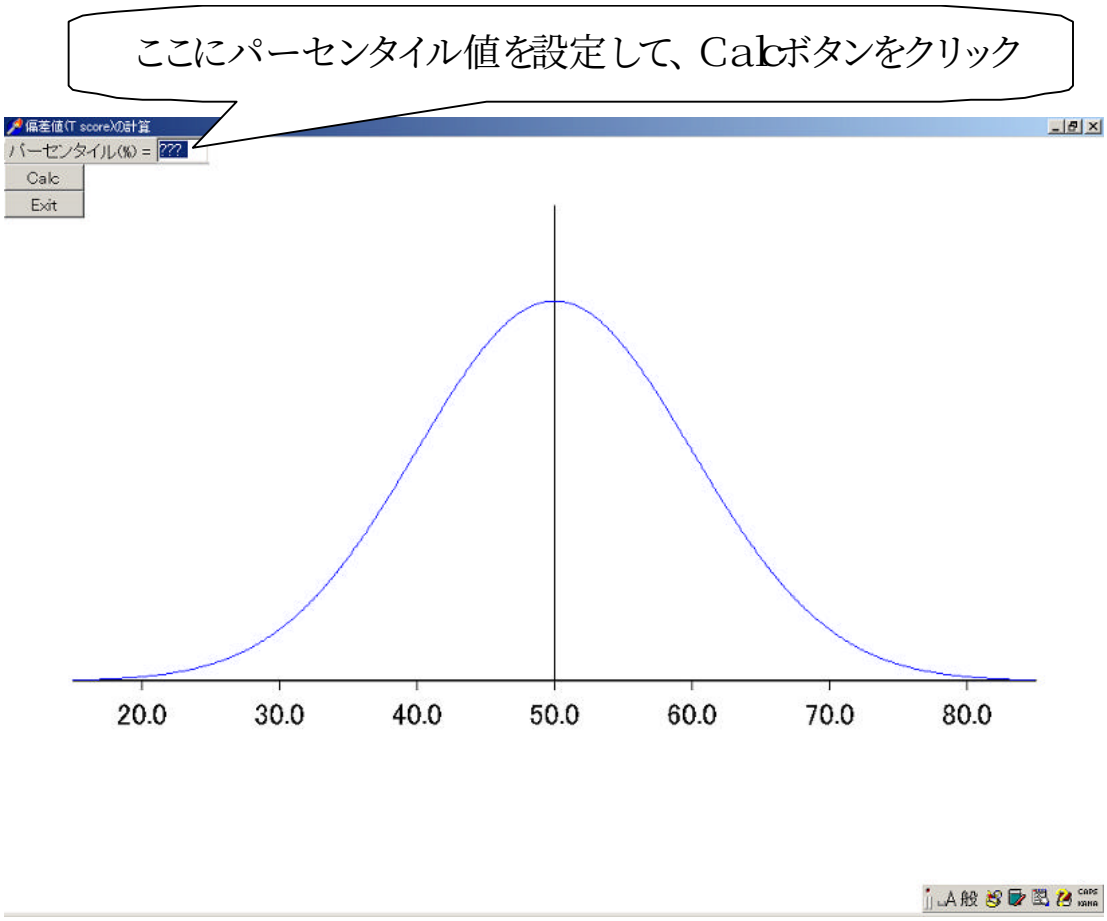

図５ GO ボタンのクリックで表示される画面

画面左上のエディットコンポーネントにパーセンタイル値100p**の値を設定してから** Calc **ボタンをクリックすると、図６の画面のようにパーセンタイル値**100p**に対する偏差値の値が表示され、対応する領域が緑色で塗り潰されて図示される。**

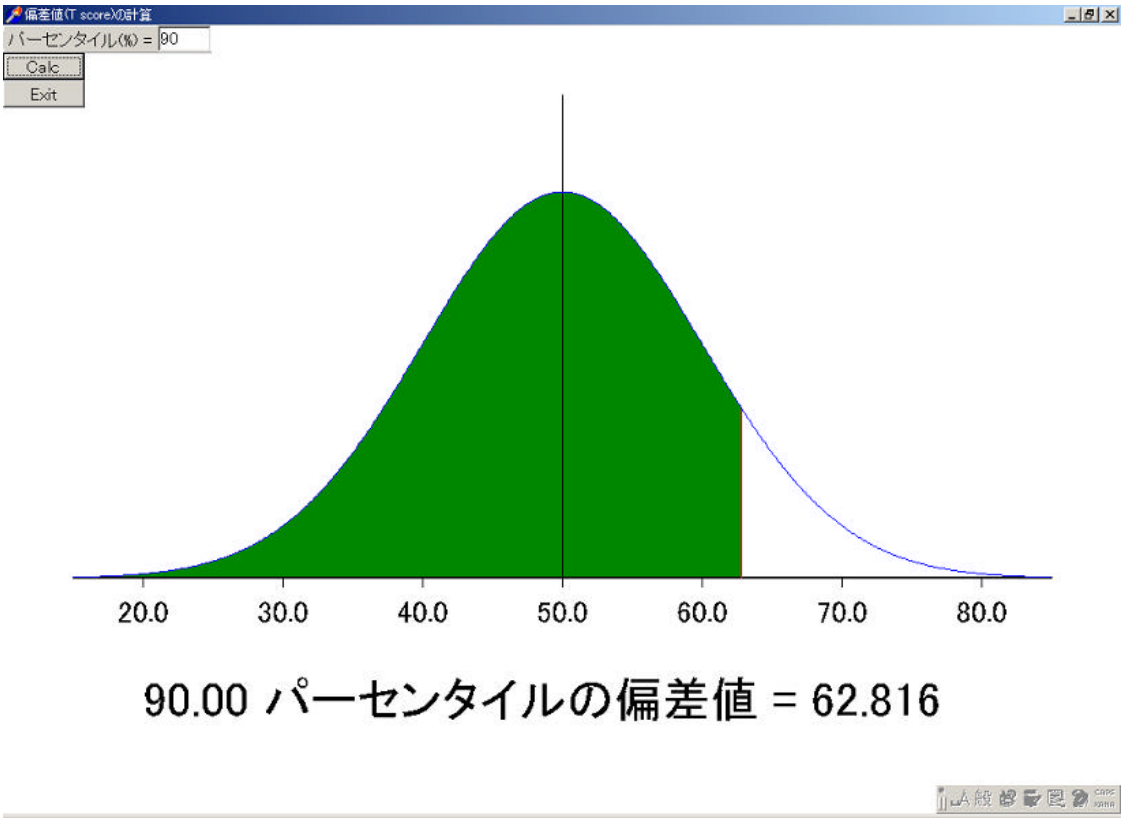

図６　計算結果の表示

再度、エディットコンポーネントにパーセンタイル値を設定し直して Calc ボタンをクリックすると、再設定された値で再計算が行われ、偏差値が表示・図示される。

プロジェクト PTscore.dpr で用いられているユニット UIntegral は積分を計算するためのユニットである。ホームページ「積分の計算」に詳しく説明した。

参 考 文 献

（１）R.L.Burden and J.D.Faires, Numerical Analysis, 3rd ed., Pp.676、Prindle, Weber & Schmidt, 1985.

（２）岡本安晴．Delphi で学ぶデータ分析法．Pp.274、CQ 出版株式会社、1998.

（３）川上一郎．数値計算．Pp.201、岩波書店、1989.

（４）芝祐順・南風原朝和．行動科学における統計解析法．Pp.293、東京大学出版会、1990.