**変化表の最小二乗法での最適化について**

変化表とは、光学系を構成する各面の曲率、面間隔を微小変化させたときに、主要収差や焦点距離がどの程度変化するかを一覧にしたものです。コンピュータによる自動設計が一般的になるまで良く使われていたといいます。専門の光学設計者の方がどのような形式のものを使っていたのか見たことはないのですが、中川治平先生の『レンズ設計工学』の５章での説明を参考に作ってみました。現在は曲率のみの変化表となっています。

例をあげて使い方の説明をしてみます。貼り合わせ３枚の望遠鏡対物レンズを想定しました。色消しの条件式から各レンズのパワーを計算して設計を始めるのが通法ですが、最小二乗法の最適化による方法でやってみました。

貼り合わせ３枚のレンズが完成した後で、それに合わせて簡単なレデューサーを設計してみます。

**1、データ入力**

サンプルデータの中にある『D100-BK7-CaF-K10-Scaling』というファイルを読み込んでいただければ、「3、最適化」から行えます。データ入力の練習を兼ねて、出来れば手動でデータを入れていただければと思います。

popsを起動します。

光学面数は『４面』、波長数『5』とします。

光学名は『D100-Bk7-CAF2-K10-StratData』

とします。

曲率半径は、

１面『500』

２面『300』

３面『－300』

４面『－2000』

面間隔は

１面間隔『8』

２面間隔『12』

３面間隔『8』

各面間隔のガラス名のカラムをダブルクリックしてガラス表からガラス材を選択します。

１面間隔Oharaのガラスから『S-BSL7』を、

２面間隔OldSchottのガラス表の最下部にある『CaF-C』を、

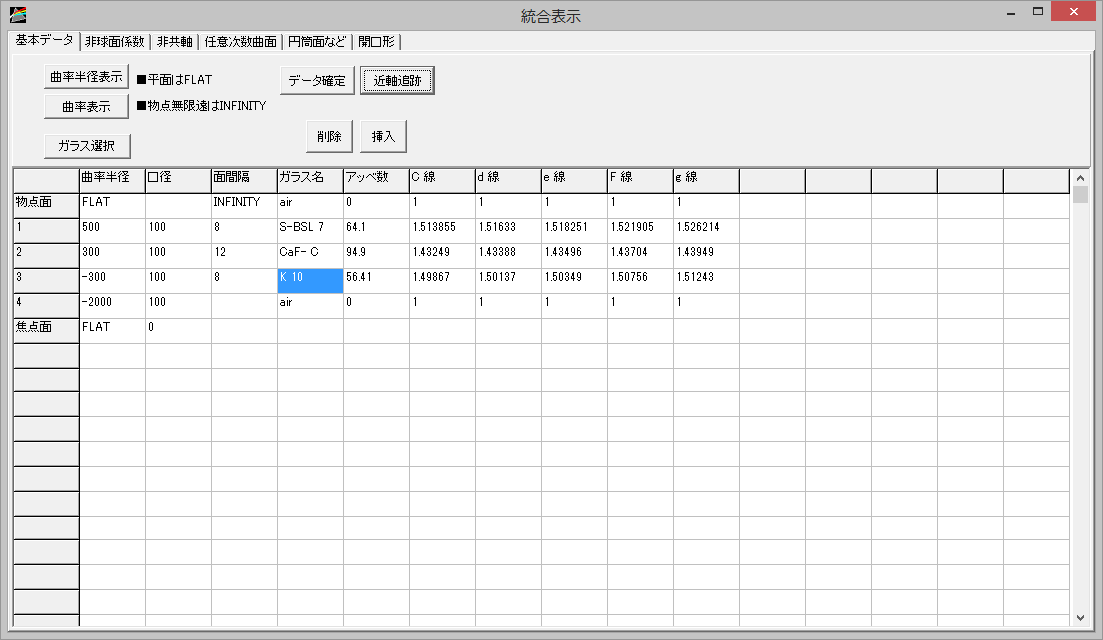
３面間隔OldSchottから『Ｋ10』

を選択します。

曲率半径は多くの対物レンズのデータを見て、大体これくらいだろうという感じで設定したものです。自動設計のあるプログラムでは、曲率０の平面から最適化のスタートが出来ますが、拙プログラムでは出来ません。

ガラス材は吉田正太郎先生の『新版屈折望遠鏡光学入門』のP236にある例からとりました。

下にデータ画面と近軸追跡画面のキャプチャーを示します。

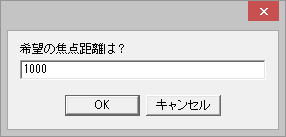
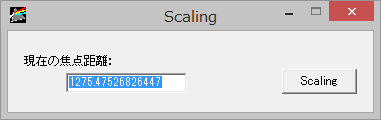




**2、スケーリング**

メインメニューの『その他』から『スケーリング』を選択します。

スケーリングのウィンドが出ます。『Scaling』のボタンを押すと『希望の焦点距離は』と聞いてきますので、『1000』を入力します。自動的に焦点距離1000に各データがスケーリングされます。スケーリングウィンドは閉じておきます。

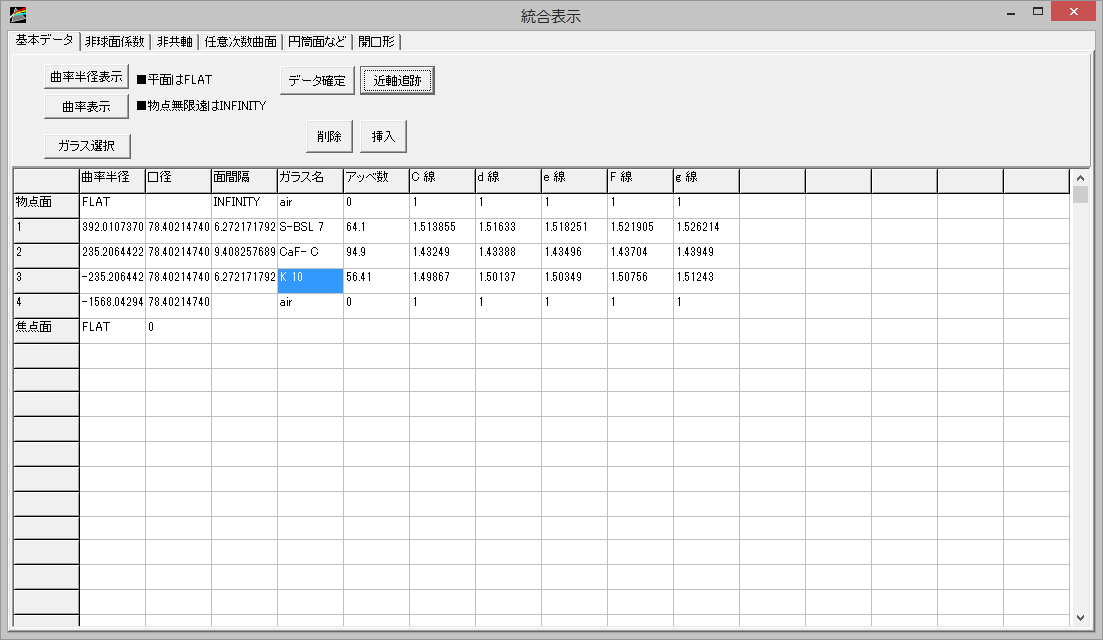


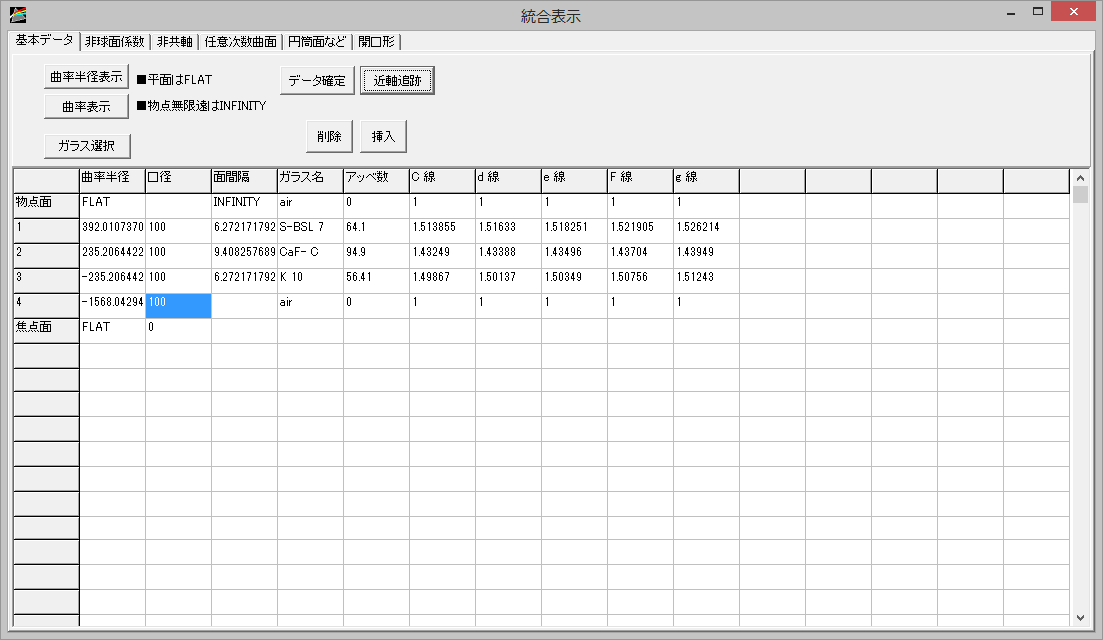


近軸追跡をすると上のようになっているはずです。

口径と面間隔も比例で変更されています。口径は『78.402147・・』になっているので、これを元の『100』にして『近軸追跡』をします。下に変更前と変更後のデータ表を載せておきます。

ここでデータを別名で保存しておきます。データは随時保存しておくとよいと思います。



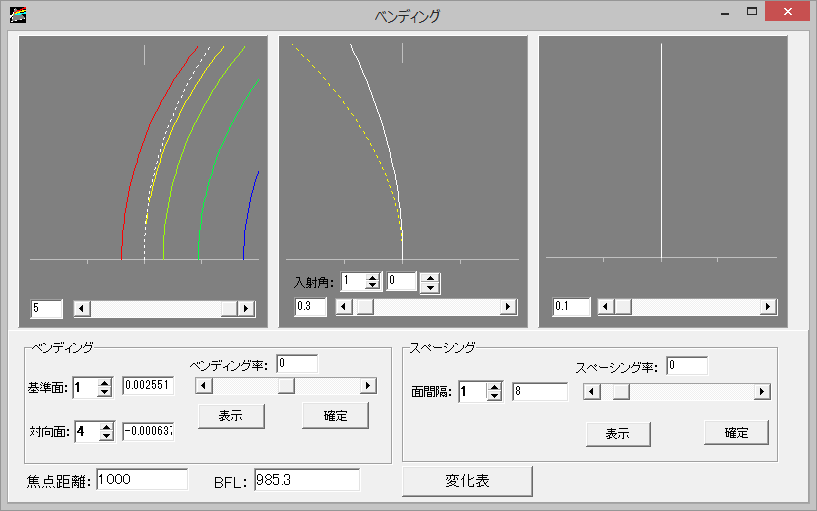


**3、最適化**

メインフォームの『設計』から『ベンディング』の『縦収差ベンディング』を選択します。

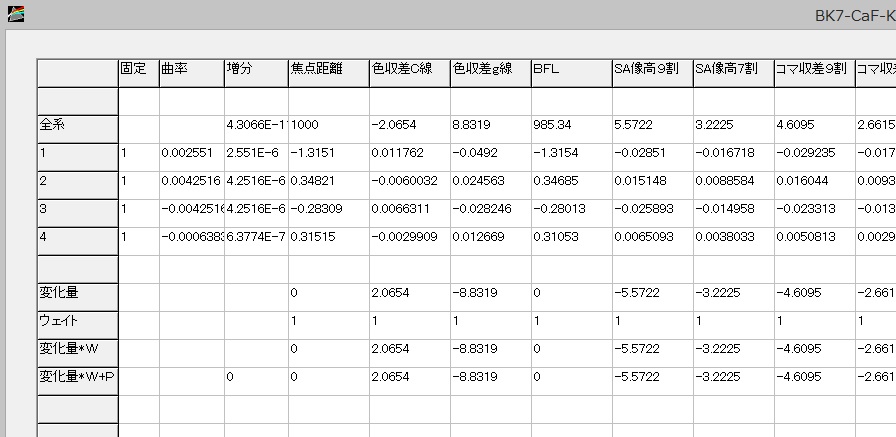
下にキャプチャを載せます。

縦収差のスケールは『5』、入射角『1』、非点収差グラフのスケールは『0.3』にしてあります。大きな色収差と球面収差があります。



次に『設計』から『ベンディング』の『変化表\*最小二乗法』を選択し変化表を表示します。（あるいはベンディングフォームの『変化表』ボタンを押します。）

変化表の『変化表作成』ボタンをクリックします。



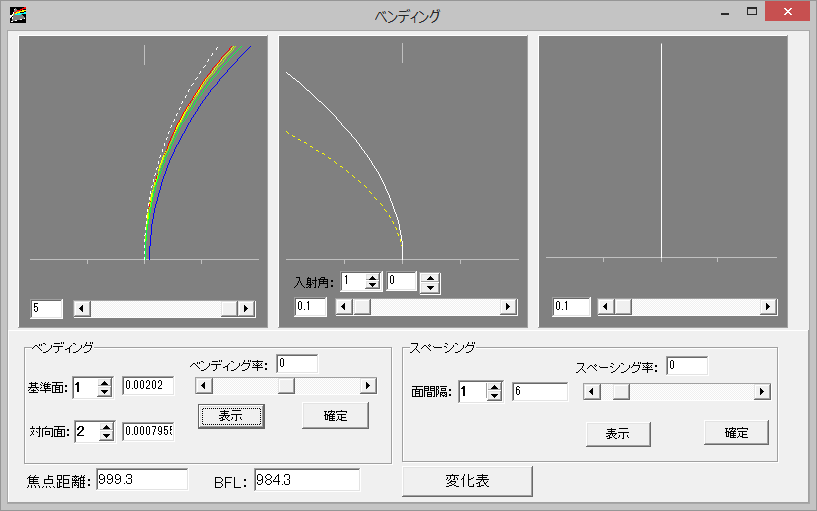
表の最上段に各収差量が、そしてこれを『0』にするように変化量が設定されます。各曲率を微小変化させたときの収差変化量が各欄に表示され、これらを連立方程式として収差最小となる各面の曲率を計算します。

『最小二乗法』のボタンを押すと表右に方程式の解と新しい曲率データが表示されます。

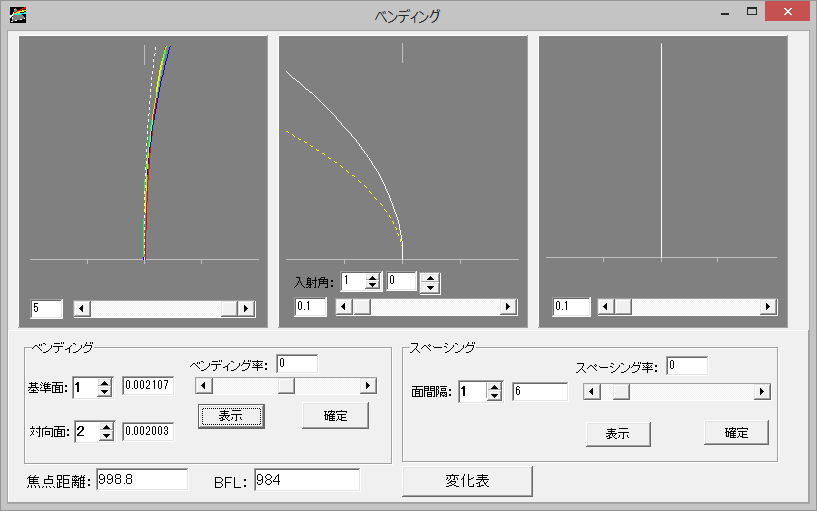


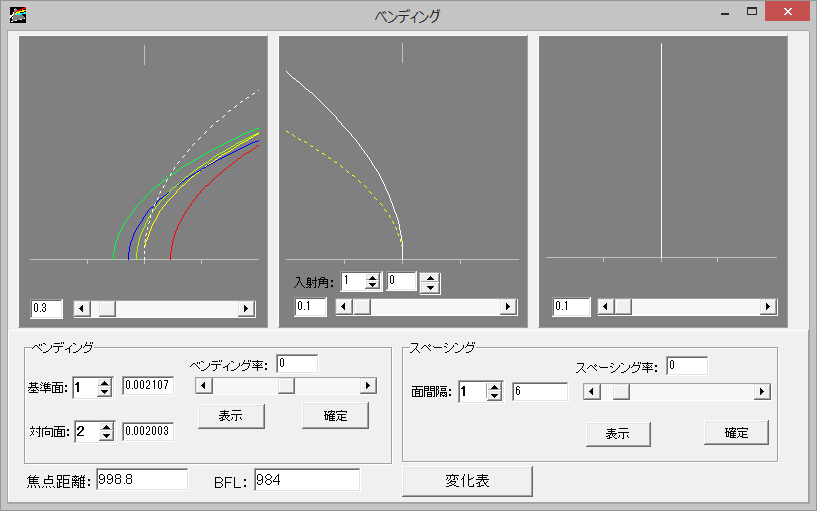
『書き換え』ボタンで新しいデータに書き換えられます。

縦収差ベンディングフォームの『表示』で収差の状況を確かめます。（Ver.1.0.8からは自動で縦収差ベンディングフォームが書き換えられます。）色収差は大きく補正されています。



変化表に戻り再度『変化表作成』で新しいデーターでの変化表を作ります。もう一度『最小二乗法』『書き換え』をして同様に確かめてみると相当少なくなっています。

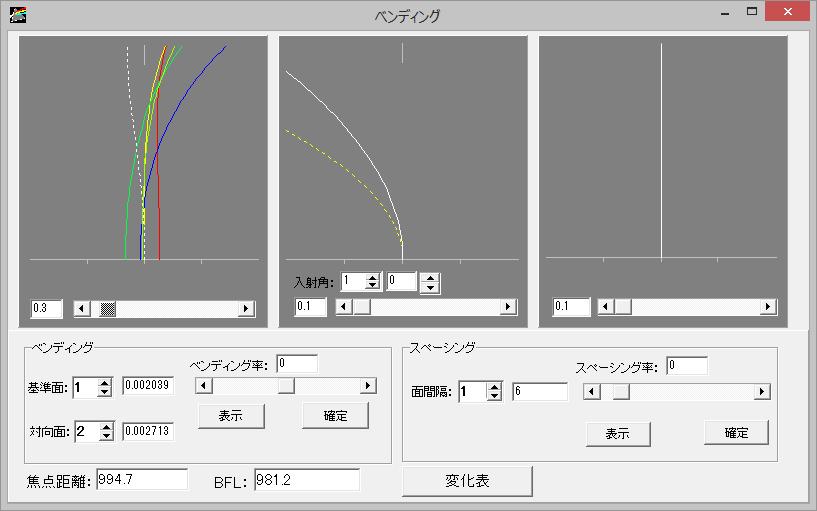




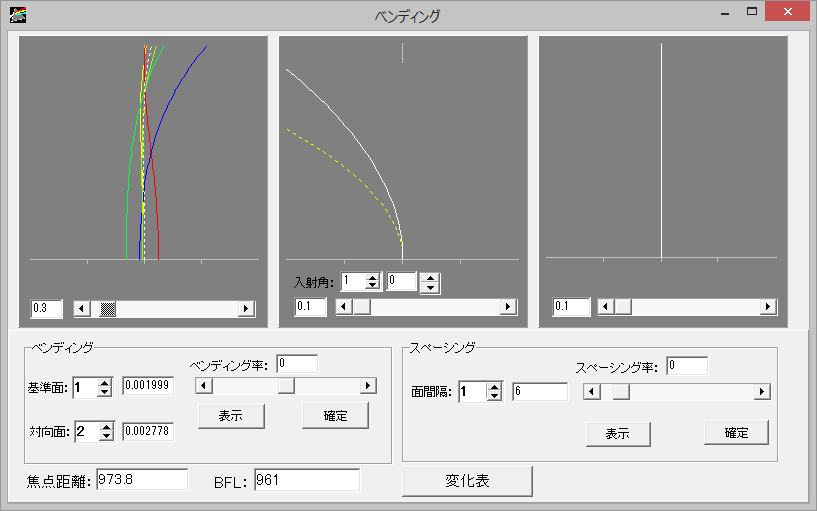
グラフスケールを『0.3』にしたのが上図です。

『繰返し』ボタンは、『変化表作成』『最小二乗法』『書き換え』の一連の作業を、スピンエディットで設定した回数繰り返します。

デフォルトの１０回繰返しを行うと以下のようになります。

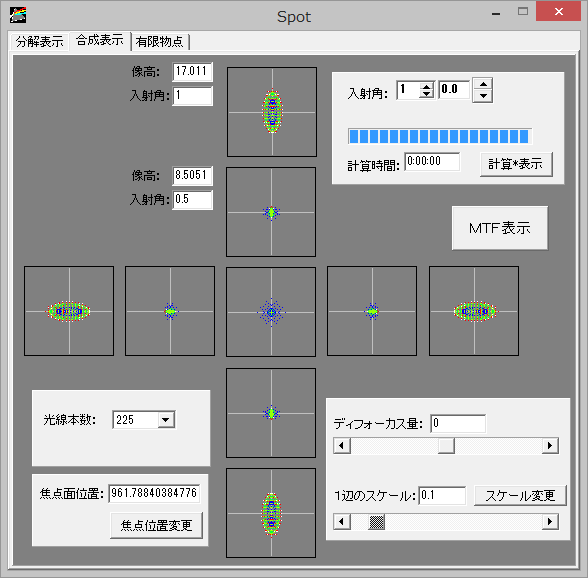


１０回繰返しを5回くらいすると収差が動かなくなってきます。

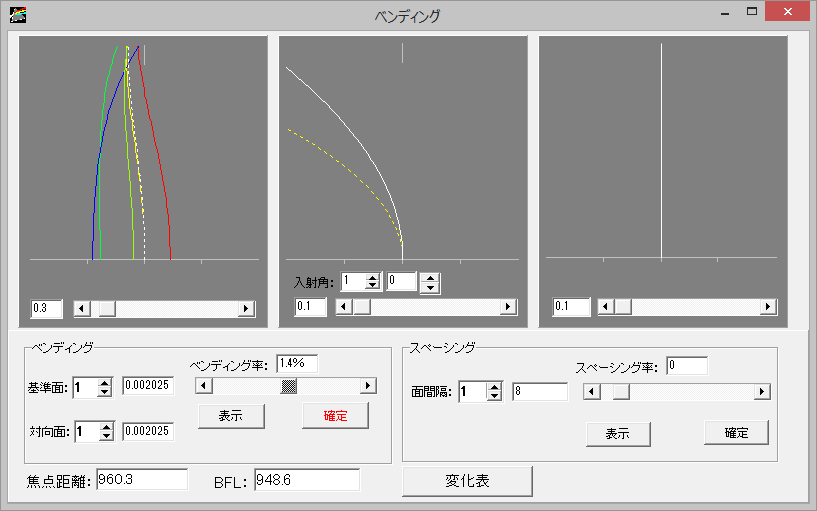


密着系レンズで線形に収差が変化するようで、特に何も考えずとも『繰返し』をするだけでほぼ収差最小のところに落ち着くようです。

軸上色収差を最小にする条件となるのでｇ線の上部がはみ出ています。スポットを見るとｇ線のハローが出ています。

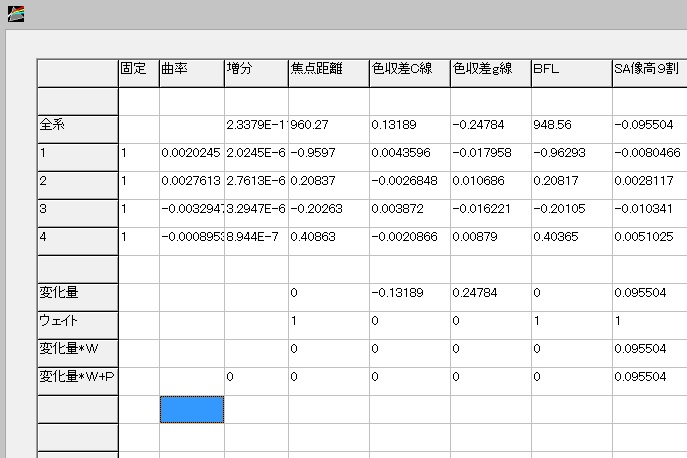


そこで縦収差ベンディングフォームで基準面を『１』対抗面を『１』にして1.4%程度ベンディングし以下のような状態にします。

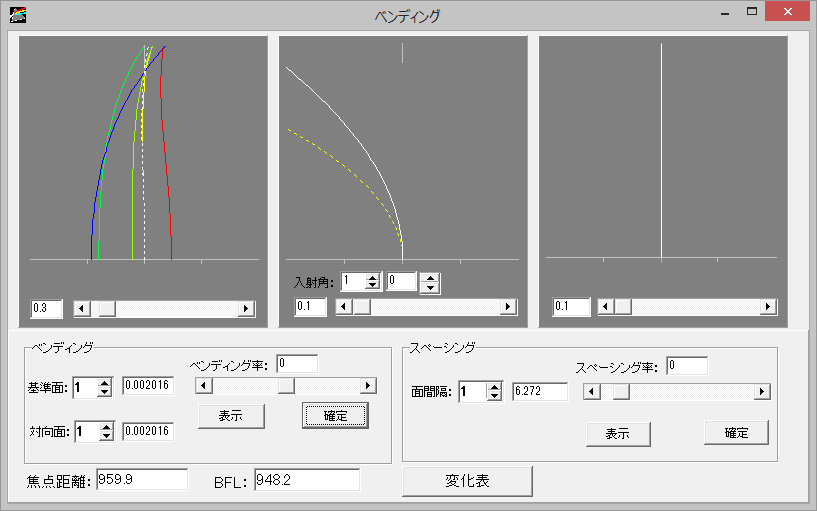


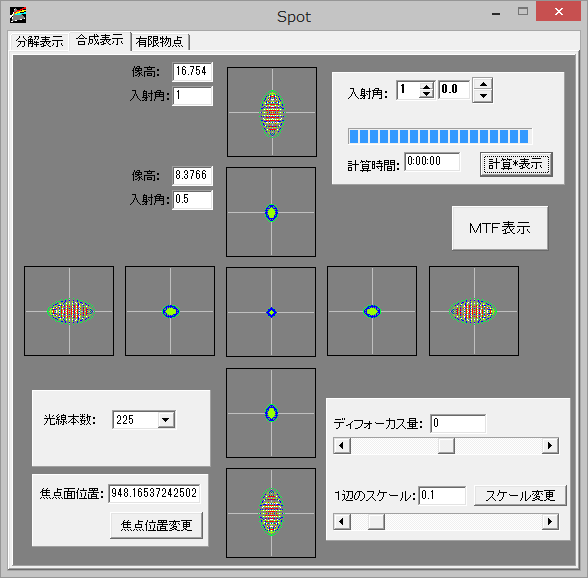
この状態で『確定』をします。忘れないでやってください。

変化表に戻り、『変化表作成』をします。色収差Ｃ線、色収差ｇ線のウェイトを『０』にして、『ウェイト確定』ボタンで確定をしたのち、『最小二乗法』をかけます。これで軸上色収差はこのままで球面収差のみ動きます。



『書き換え』でデータを新しくして収差を確認します。





軸上色収差は少し増えますが、ハローはなくなります。補正状況は使用目的でさまざまですが、変化表の使い方を説明する意味で、補正する収差を選択する方法の説明を行ってみました。収差補正はもう少し良くなるところがあるように思います。

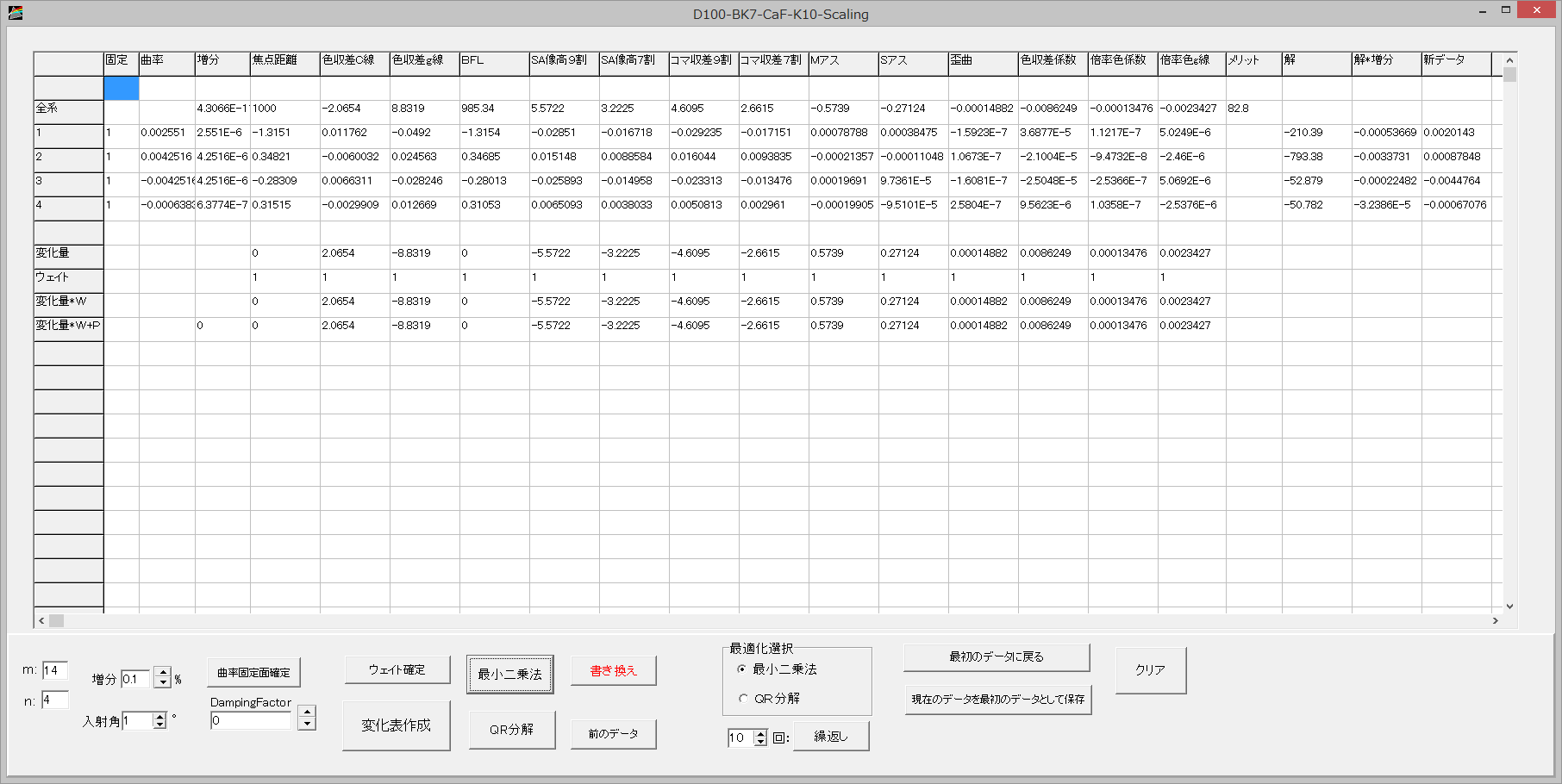
ここで適当な名前を付けてデータを保存しておきます。

通常の連立方程式を正方行列を求める方法で解くことが出来ない場合に、QR分解で解を求めます。『繰返し』をする場合は、ラジオボタンで最小二乗法とQR分解を選択できるようにしてあります。

変化させたくない面は『固定』欄に『0』をいれて、『曲率固定面確定』で確定すると、変化させないように出来ます。例えばレデューサーを設計するときに、対物部を変化させないようにする時などに使えます。

『増分』は変化表作成時の曲率変化量を変えます。光学系によって、また設計の段階によって変えます。『入射角』は評価する光学系により変えます。

**4、最小二乗法の説明**



最小二乗法での最適化について今回の実例で説明します。

上のキャプチャーは、今回の例でスケーリングが終わった時の変化表です。スケールが小さくなり見にくいと思いますが、表示を拡大してみてください。２列目曲率の列には上から１面２面・・の曲率が表示されています。ちなみに平面を0と扱えるため、光線追跡の計算は内部では曲率で処理されます。次の列、増分にはこのケースでは0.1%分の曲率変化量が表示されています。次の列からは、曲率を増加させた場合に変化する焦点距離、球面収差などの変化量が計算されています。

１面でいえば2.551Ｅ－6曲率を増加させると、焦点距離は－1.3151短くなり、Ｃ線の近軸色収差は0.01176増加し、球面収差は―0.017281少なくなる・・・という具合です。

さて最上部の行に焦点距離1000、色収差Ｃ線－2.0654、色収差ｇ線8.8319、ＢＦＬ985、像高９割の球面収差3.3331・・・とあります。最下部の行、変化量には、焦点距離は変化させたくないので0、色収差Ｃ線は０にしたいので符号を逆にした2.0654を、色収差ｇ線は－8.8319（現在より8.8319減らす）・・・と数値が入るようになっています。

これらの数値を収差ごとにまとめ、

焦点距離・・・－1.3151Ｘ1+0.34821Ｘ2－0.28309Ｘ3+0.31515Ｘ4＝0・・変化させない

色収差Ｃ線・・0.01176Ｘ1－0.0060032Ｘ2+0.0066311Ｘ3－0.0029909Ｘ4＝2.0654

色収差ｇ線・・－0.0492Ｘ1+0.024563Ｘ2－0.028246Ｘ3+0.012669Ｘ4＝－8.8319

ＢＦＬ・・・・－1.3154Ｘ1+0.34685Ｘ2－0.28013Ｘ3+0.3105Ｘ4＝0・・変化させない

ＳＡ像高9割・－0.017281Ｘ1+0.0091577Ｘ2－0.015472Ｘ3+0.0039319Ｘ4＝－5.5722

・

・

このように14本の連立方程式を立て、この方程式を最小二乗法で解き、Ｘｎを求めると、（Ver.0.1.8からは最適化に使う収差数を14に増やしています。）

Ｘ1＝－210.39

Ｘ2＝－793.38

Ｘ3＝－52.879

Ｘ4＝－50.782

となります。各面の曲率増加量は、0.1%増分に解を掛けた、

2.551Ｅ－6×－210.39＝－0.00053669

4.252Ｅ－6×－793.38＝－0.0033731

4.252Ｅ－6×－52.879＝－0.00022482

6.379Ｅ－7×－50.782＝－3.2386Ｅ-5

となり、これを現在の曲率に加えれば、焦点距離やＢＦＬを変化させないで、各収差を最小にする近似解となるわけです。

以上が最小二乗法による最適化の実際です。

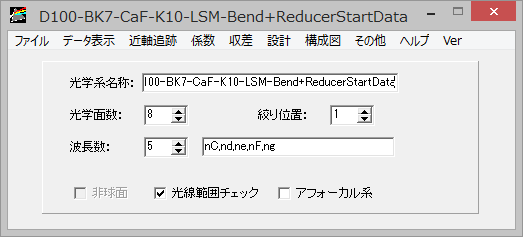
曲率の変化と収差の変化が線形という前提ですので、実際は非線形の変化が多い光学系では成り立たないことの方が多く、とんでもない収差補正状態になってしまうことが多々あります。

１４本の方程式ですので、最適化できる光学面数は１４面となります。

最適化に関してだけ考えれば、微係数を作る収差をスポットダイヤグラムの数値を使っても良いのですが、縦収差を使った変化表は、光学系の構成要素と収差の関係を把握しやすいといわれています。

**5、レデューサーの設計**

さて一応の完成を見たこのレンズに、倍率0.75倍くらいのレデューサーを設計してみます。メインフォームの光学面数を『8』にします。



第５面の曲率半径を『４００』

第６面の曲率半径を『－９００』

第７面の曲率半径を『－９００』

第８面の曲率半径を『－１０００』

口径は５,６,７,８面とも『５０』

第４面間隔は『８００』（対物レンズからレデューサーまでの距離）

第５面間隔は『５』

第６面間隔は『０．１』

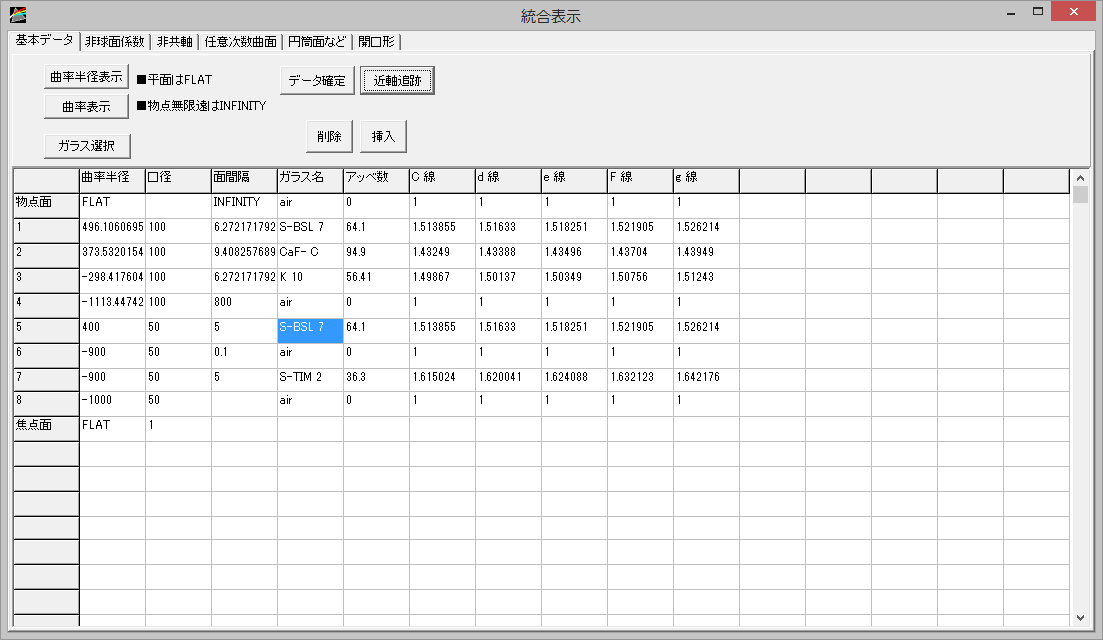
第７面間隔は『５』

各面間隔のガラス名のカラムをダブルクリックしてガラス表からガラス材を選択します。

第5面間隔のガラスはBK7なのでOharaのガラスから『S-BSL7』を、

第６面間隔は空気なのでそのまま、

第７面間隔はOharaのガラスから『S-TIM2』を選択します。

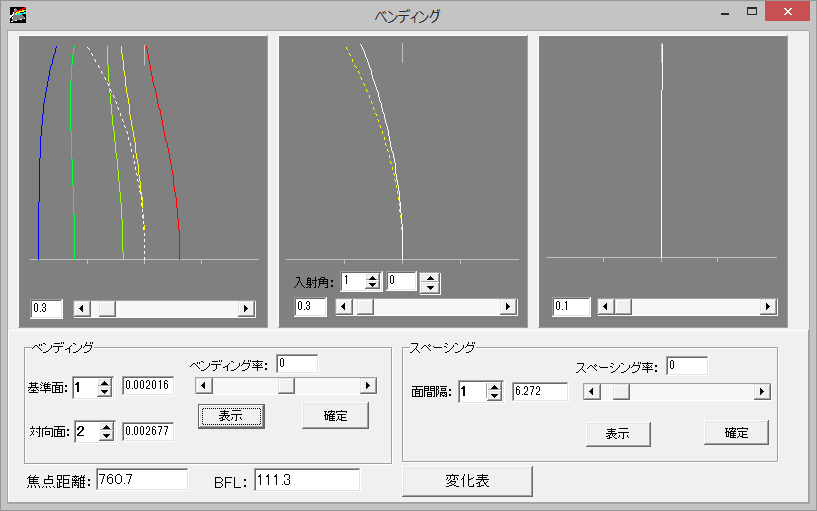


『近軸追跡』フォームを開き近軸追跡をします。

焦点距離が大体760ｍｍくらいになるようなパワーをレデューサー系が持つように、各面のＲを調整します。形状も普通の色消しレンズ状でいいと思います。

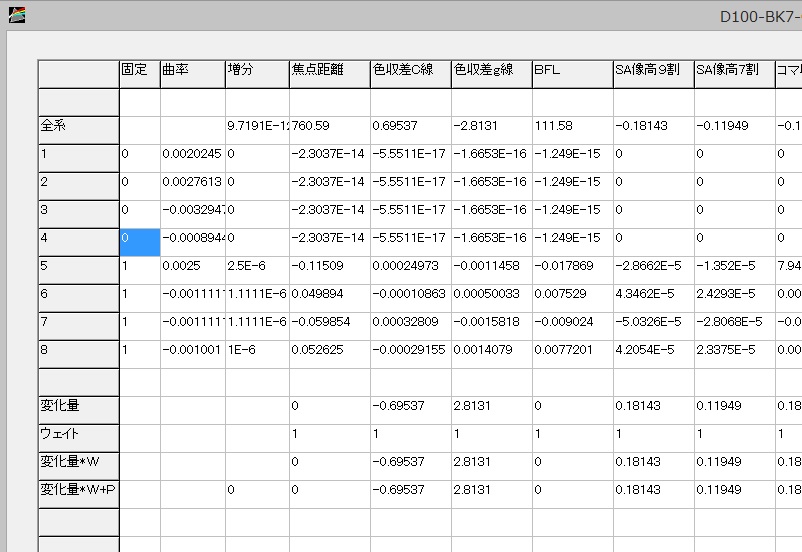


一度縦収差ベンディングフォームを閉じ再度開きます。続いて変化表を開きます。一度クリアをして、再度『変化表作成』をします。



対物レンズ部分は変化させたくないので、『固定』列を４面まで『0』にして、『曲率固定面確定』ボタンで確定をします。

先に対物レンズのベンディングの時に、色収差Ｃ線とｇ線のウェイトを『０』にしたのを『1』に戻してください。

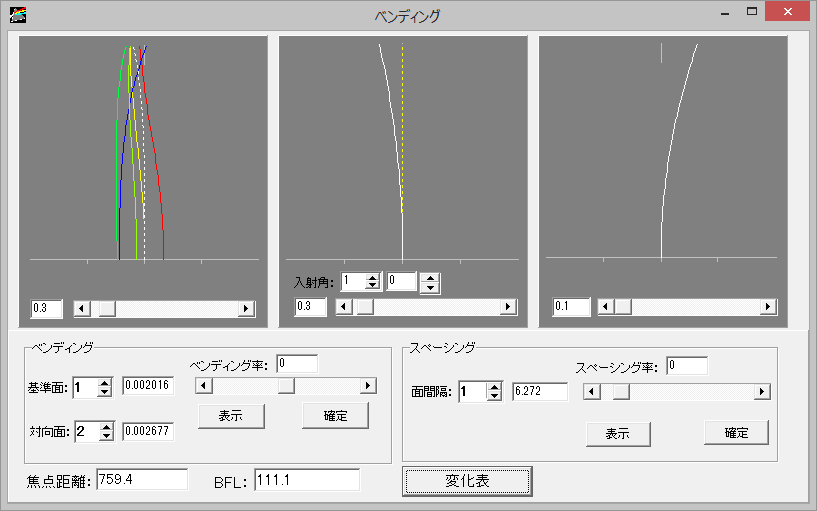


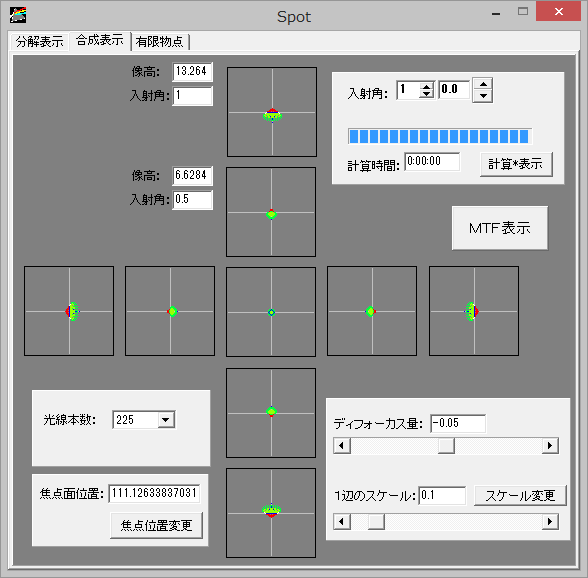
Ｃ線、ｇ線のウェイトを『1』に戻してあります。1、2、3、4面は固定の『０』に設定してあります。

２回程度最小二乗法を掛けるとすぐに収束します。

スポット図を見ると、多少ディフォーカスが必要ですが周辺像は確かに小さくなっていて、短焦点化、像面平坦化がされています。

初期値の設定は、焦点距離750～760ｍｍくらいを目安に色々な数値に出来ます。初期値の違いにより自動補正の結果は随分違ったものになります。今回の初期値はあくまで一つの例とお考え下さい。





以上実例で大まかな変化表の最適化について説明しました。

以下に最小二乗法（過剰条件の連立方程式解法）をプログラムコードを示して説明をします。

**最小二乗法による過剰条件の連立方程式の解法**

下記のように変数と式の数が等しい連立方程式の場合

Ｘ1＋　2Ｘ2＋　8Ｘ3 ＝　6.1

2Ｘ1＋　3Ｘ2＋　4Ｘ3 ＝ 4.5

7Ｘ1＋　2Ｘ2＋　2Ｘ3 ＝ 4.3

行列式では下記のようになります。

1 2 8 　Ｘ1  　6.1

2 3 4 　 Ｘ2  ＝ 4.5

7 2 2　　　Ｘ3　　　　　　4.3

Ｎ×Ｎの正方行列なので逆行列と積で解くことが可能で、エクセルの数式にも逆行列と積があります。解法プログラムも、掃き出し法、ガウス・ザイデル法など色々なアルゴリズムがプログラミングの参考書に載っています。

しかし下記のように条件式が変数より多い場合（Ｎ×ＭでＮ＜Ｍ）は特別な場合を除き解を持たないとされ、最小二乗法により近似解が得られると説明されています。

Ｘ1＋　2Ｘ2＋　8Ｘ3 ＝　6.1

2Ｘ1＋　3Ｘ2＋　4Ｘ3 ＝ 4.5

7Ｘ1＋　2Ｘ2＋　2Ｘ3 ＝ 4.3

6Ｘ1＋　Ｘ2＋　5Ｘ3 ＝　5.3

2Ｘ1＋　9Ｘ2＋　2Ｘ3 ＝ 6.3

3Ｘ1＋　4Ｘ2＋　5Ｘ3 ＝ 5.9

解法プログラムを探したのですが、探し方が下手だったのかウェブ上には参考になるプログラムコードが見当たりませんでした。漸く株式会社タイコの牛山善太先生の「最小二乗法について」で正規方程式の解説を見つけました。それを参考に何とかプログラムを組むことが出来ました。以下にDelphiのコードのコア部分をそのまま載せ、若干の解説してみました。実際の実行にはプロジェクトファイルに組込み、コンパイルが必要です。またプログラムはアルゴリズムを検証するためのもので、ソースコードに上記の連立方程式のデータをそのまま書き込んでいます。

//////////////////////////////////////////////////////////////////////////////

unit LSM;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls;

type

TForm1 = class(TForm)

Button1: TButton;

Memo1: TMemo;

procedure Button1Click(Sender: TObject);

private

{ Private 宣言 }

a:array[1..100,1..100] of double;

b:array[1..100] of double;

X:array[1..100,1..100] of double;

S:array[1..100,1..100] of double;

T:array[1..100] of double;

M,N,N1,NA1:integer;

public

{ Public 宣言 }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

procedure TForm1.Button1Click(Sender: TObject);

var i,j,k,na,kp:integer;

ss,tt,p,temp:double;

str:string;

begin

M:=6;

N:=3;

a[1,1]:=1;a[1,2]:=2;a[1,3]:=8; //方程式係数の部分のデータセット

a[2,1]:=2;a[2,2]:=3;a[2,3]:=4;

a[3,1]:=7;a[3,2]:=2;a[3,3]:=2;

a[4,1]:=6;a[4,2]:=1;a[4,3]:=5;

a[5,1]:=2;a[5,2]:=9;a[5,3]:=2;

a[6,1]:=3;a[6,2]:=4;a[6,3]:=5;

b[1]:=6.1; //方程式右辺の部分

b[2]:=4.5;

b[3]:=4.3;

b[4]:=5.3;

b[5]:=6.3;

b[6]:=5.9;

for k:=1 to N do begin //正規方程式の正方行列を作る、重要な部分。

for j:=1 to N do begin

ss:=0;

for i:=1 to M do begin

ss:=ss+a[i,j]\*a[i,k];

end;

S[j,k]:=ss;　　　　　　　//Ｎ×Ｎの正方行列

end;

end;

for j:=1 to N do begin

tt:=0;

for i:=1to M do begin

tt:=tt+a[i,j]\*b[i];

end;

T[j]:=tt;

end;

for j:=1 to N do begin //掃き出し法への準備

for i:=1 to N do begin

X[j,i]:=S[j,i];

end;

end;

for i:=1 to N do begin

X[i,N+1]:=T[i];

end;

//掃き出し法で連立方程式を解く

N1:=N+1;

for na:=1 to N do begin

p:=0;

na1:=na+1 ;

for i:=na to N do begin

if p>abs( X[i,na]) then begin

end else begin

p:=abs( X[i,na] );

kp:=i;

end;

end;

if p<1e-8 then begin

ShowMessage('不定です。');

exit;

end;

for j:=na to N1 do begin

temp:=X[na,j];

X[na,j]:=X[kp,j];

X[kp,j]:=temp;

end;

for j:=na1 to N1 do begin

X[na,j]:=X[na,j]/X[na,na];

end;

for i:=1 to N1 do begin

if i=na then begin

end else begin

for j:=na1 to N1 do begin

X[i,j]:=X[i,j]-x[i,na]\*X[na,j];

end;

end;

end;

end;

Memo1.Lines.Add(' '); //表示

Memo1.Lines.Add('a1\*X1+a2\*X2+a3\*X3=Y ');

Memo1.Lines.Add(' ');

for j:=1 to M do begin

str:=' ';

for i:=1 to N do begin

str:=str+FloatToStr(a[j,i])+' ' ;

end;

str:=str+' '+floatToStr(b[j]);

Memo1.Lines.Add(str);

end;

Memo1.Lines.Add(' ');

for i:=1 to N do begin

Memo1.Lines.Add('X'+IntToStr(i)+'='+FloatToStr(X[i,N+1])) ;

end;

end;

end.

////////////////////////////////////////////////////////////////////////////////

一般的な式で説明し、ソースコードの対応する部分を照らし合わせてみます。

重要なのは正方行列を作る部分です。

一般に以下のような連立方程式があるとき、

a11X1 + a12X2 + ・・ + a1nXn = b1

a21X1 + a22X2 + ・・ + a2nXn = b2

* 式1

・　　

am1X1 + am2X2 + ・・ + aｍnXn = bm

行列式で表せば、

a11 a12 ・・ a1n X1 b1

a21 a22 ・・ a2n X2 b2

　 ・　　　　　　・　　＝　　・

am1 am2 ・・ amn Xn bm

上記のようになります。

先ほどのソースコードでは、配列にデータを書き込んでいる以下の部分が行列aに当たります。

a[1,1]:=1;a[1,2]:=2;a[1,3]:=8;

a[2,1]:=2;a[2,2]:=3;a[2,3]:=4;

a[3,1]:=7;a[3,2]:=2;a[3,3]:=2;

a[4,1]:=6;a[4,2]:=1;a[4,3]:=5;

a[5,1]:=2;a[5,2]:=9;a[5,3]:=2;

a[6,1]:=3;a[6,2]:=4;a[6,3]:=5;

上記の部分です。

さて最小二乗法で近似解を見つけるということは式1を、

φ＝Σim=1 **(**ai1X1 + ai2X2 +・・・+ ainXn - bi**)**2　　　式3

としてφが最小になるＸを見つけるということになります。

式3をＸｎで偏微分すると

δφ/δＸｎφ＝2Σim=1 **(**ai1X1 + ai2X2 +・・・+ ainXn - bi**)**aiｎ

＝2 [X1Σim=1aiｎai1 + X2Σim=1aiｎai2 +・・+ XnΣim=1aiｎain -Σim=1aiｎbi ]

となり連立方程式の行列表記では下記のようになります。

　Σim=1ai1ai1 　Σim=1ai1ai2 ・・Σim=1ai1aiｎ　 X1　　 Σim=1ai1bi

　Σim=1ai2ai1 　Σim=1ai2ai2 ・・Σim=1ai2aiｎ　 X2　　 　Σim=1ai2bi

・＝・式4

　　　　　　　 ・　　　　　　　　　 　　　　　　 ・

　Σim=1aiｎai1 　Σim=1aiｎai2 ・・Σim=1aiｎaiｎ　 Xn　　 　Σim=1ainbi

これでＮ×Ｍだった行列がＮ×Ｎとなり一般的解法が使えることになります。これを正規方程式と呼ぶようです。

ソースコードでは S[k,j]という配列がこれに当たり、

S[1,1] = Σim=1ai1ai1S[1,2] = Σim=1ai1ai2

S[2,1] = Σim=1ai2ai1

であり、Σを求めているコードが下記の部分です。

ss:=0;

for i:=1 to M do begin

ss:=ss+a[i,j]\*a[i,k];

end;

S[j,k]:=ss;

またソースコードの配列Ｔは、式4の右辺

T[1] =Σim=1ai1bi

に当たります。

後は通常の掃き出し法で方程式を解いています。

過剰条件の連立方程式を解法でした。出来てしまえば簡単なことなのですが、概念だけでなく実際のコードも合わせた情報がなかったのでまとめてみました。添え字がややこしく、間違いがあるかもしれません。ご容赦ください。