1982年発行の「**パソコンによる数値計算**」朝倉書店発行と言う本を偶然見つけ購入した。
以前から、実験式作成に興味があったのでこの本を購入したのです。

この中に、「最適な実験式を求める手順」と言うのがあり非常に興味がそそられた。
そこで、この本にあるフローチャートをプログラムに書こうと思い立ったのが約30年前のことです

最初は、BASIC言語、C言語でDosベースのプログラム、C++言語でWindowsベース。
ただ、思ったほどすばらしい結果にはなりませんでした。多分誤植と思われる部分が
あったので、修正してプログラムしました。

この本に従って基本動作をここにまとめます。

基本動作の順序

1 CMfiDlgクラスで補完するデータを RecordData 関数で **m_ArrayXY** アレーに保存
2 CMfiDlg::OnButtonMfi() 補間開始

3 与えられた実験データがランダムに並んでいる場合を想定し、それらの並び換えを行う
CMfiDlg::sorting(void)関数で行っている

4 最小二乗法と平均法に区別
ラジオボタンでチェックされた m_cbIndex == TRUE の時は、最小二乗法とする
CMfiCalc : : mfi.Step_Main 関数を呼ぶ

5 平均法（MfiCalc5.cpp）
CMfiCalc : : mfi.Cal_2 関数を呼ぶ

6 最適な多項式の選定
ニュートンの補間公式にもとづいて階差表を作成する準備が必要です。
n組のデータは、xに関して等間隔に並んでいるものに適用されます。
これは、隣接するyの値の差をとりその差の値についてもまた次々と差を取る階差表を
作る必要があるためです。

① Δx が一定 (P1=1) の場合と$\Delta \log x$ が一定 (P1=2) の場合の振り分け
CMfiCalc::Step_1関数が行っている

② 与えられたデータが等間隔ではなく、不規則に配列している場合を想定して、
直線補間法によりデータの等間隔化を図る。
試行錯誤の結果、ラグランジュの補間法ではなく、直線法で行っている。
CMfiCalc::Step_21 Step 21 (Δx一定)
CMfiCalc::Step_22 Step 22 ($\Delta \log x$一定)
CMfiCalc::Step_2y Step 2y (Get YA)
直線補間法によるデータ等間隔 Y作成し temp->m_YA[i]に代入

③ P1=1 及び P1=2 別に、第1階差から指定された第m階差まで順番に階差をとってゆく
CMfiCalc::Step_3to9 自動計算式セレクション
Step_3 階差パターン別に振り分けるループ
Step_4 階差パターン別に振り分ける
Step_5 階差計算

④ 第m 階差値、$\Delta^m Y^i$ ($=\Delta_{m-1} Y^{i+1} - \Delta_{m-1} Y^i$)の中で絶対値の最も
大きいもの ($=E_{max}$) を抽出し、あらかじめ設定した判定条件(E) と大小関係を
比較する。
Step_6 階差が収束 （0.001以下の差）

7 最小二乗法は、次のようにプログラムしています。

```
for(k=0; m>=k; k++){
        for( int i=0; m>=i; i++){
            a[k][i] = 0.0;
            for( int j=1; n>=j; j++){
                a[k][i] += temp->m_V[i][j] * temp->m_V[k][j];
            }
    }
    b[k] = 0.0;
    for( j=1; n>=j; j++){
            b[k] += temp->m_YC[j-1] * temp->m_V[k][j];
    }
}
for(k=0; m-1>=k; k++){
    for(i=k+1; m>=i; i++){
            w = a[i][k]/a[k][k];
            for(j=k; m>=j; j++){
                a[i][j] -= w * a[k][j];
            }
            b[i] -= w * b[k];
    }
}

temp->m_AA[m] = b[m]/a[m][m];
for(k=m-1; 0<=k; k--){
    temp->m_AA[k] = b[k];
            for(i=k+1; m>=i; i++){
                temp->m_AA[k] -= (a[k][i] * temp->m_AA[i]);
            }
    temp->m_AA[k] = temp->m_AA[k]/a[k][k];
}
```

カスタムクラスの作成

● MfiDlg.h の冒頭で、以下のクラスを定義している

CMfiData クラスを定義　元のデータを保持する文字定数

```
CString m_strX;
CString m_strY;
```

CMfiDlgクラスで補完するデータを RecordData 関数で m_ArrayXY アレーに保存

CMfiResult クラス　補間したの式の係数などを保持するクラス

```
int m_nFomular;      数式のタイプ
int m_nm;            n次の数式
double m_aa[7];      定数を保持する変数
```

● MfiCalc.h の冒頭で、以下のクラスを定義している

CMfiValue クラスを定義

```
double m_X;
double m_Y;
```

CMfiTemp　仮変数クラス

```
CArray <CMfiValue, CMfiValue> m_XY; // 基のデータ
CArray <double, double> m_XA; // 等間隔化のデータX
CArray <double, double> m_YA; // 等間隔化のデータY
CArray <double, double> m_YB; // ニュートン補間公式でとるべき階差データ
CArray <double, double> m_YC; // 最小二乗法に代入されるデータY
CArray <double, double> m_AA; // Calc_3(最小二乗法で生成された常数)
double m_V[30][30];          // Select_g で生成される常数
int m_ma;
int m_pattern;           選定された式のパターン
double m_emvalue[13];  // 各パターンの最小階差
int m_emma[13];        // 各パターンの多項式次数
int m_max;
double m_XCC; // Step4(階差パターン式)で生成された常数
double m_YCC; // Step4(階差パターン式)で生成された常数
double m_CN;  // Step4(階差パターン式)で生成された常数
CEmData m_Em; //3種類の最適式
```

CEmData　階差クラスで以下の変数を持っています

```
double m_Em0;
double m_Em1;
double m_Em2;
int    m_nEm0;
int    m_nEm1;
int    m_nEm2;                    CMfiCalc::min_em
```

● 予備作業(MfiCalc2.cpp)

CMfiDlg::CombertDataToValue 文字データを数値データに変更

CMfiDlg::CombertDataToString 数値データを文字データに変更

● 補間開始 (MfiDlg.cpp)

CMfiDlgクラスで補間するデータを RecordData 関数で m_ArrayXY アレーに保存

CMfiDlg::OnButtonMfi()　補間開始

CMfiDlg::sorting(void)　Xデータを小さい順にソーティングする

● 補間ルーチン (MfiCalc.cpp)

CMfiCalc::Step_Main

Step 関数を呼び、補間動作のStepを定義する

補間式選定は、ニュートン法による階差表を作成し最も小さい階差を持つ
次数の多項式を選定する。

その為に、Xデータ間が等間隔(⊿xまたは⊿log x)のデータを作成しなければ
ならない。

CMfiCalc::Step　Step 1～9までの関数を呼び出す

① CMfiCalc::Step_1　データ X の間隔 を見極める
PI = 1 .... Δx一定
PI = 2 .... Δlog x 一定

a データが2つしかない場合は、その差が20倍以上の時、PI=2 とし
それ以外の時は、PI=1 としている。

b 標準偏差(Standard Diviation)を計算し、関係の強さを判定する
標準偏差が小さいほうが関係が深い。
Δx(S)< Δlog x(S) の時 PI=1
Δx(S)> Δlog x(S) の時 PI=2

CMfiCalc::Sd_Normal ⊿x の標準偏差
CMfiCalc::Sd_Log ⊿log x の標準偏差

② CMfiCalc::Step_2　x(n) と log x(n) の偏差値を取り等間隔にあるかを検証
オリジナルデータの第一差(⊿x1) の1%以下を等間隔とする
戻り値 ret = TRUE は、等間隔にある
ret = FALSE は、等間隔でない

③ CMfiCalc::Step_21　Step 21 (⊿x一定)
⊿x一定の等間隔 X を作成し、temp->m_XA に代入

⊿X＝データの最大と最小幅をを個数で割る
xn = X(i) + ⊿X * i

④ CMfiCalc::Step_22　Step 22 (⊿log x一定)
⊿logx 一定の等間隔 X 作成し、temp->m_XA に代入

⊿X＝log(データの最大/最小幅)をを個数で割る
xn = X(i) + log(⊿X)* i

⑤ CMfiCalc::Step_2y　Step 2y (Get YA)
線形補間によるデータ等間隔 Y作成し temp->m_YA[i]に代入

$y' = yi + (y^{i+1} - y^{i})/(x^{i+1} - x^{i}) * (x-x^{i})$

⑥ CMfiCalc::Step_3to9　自動計算式セレクション

a PI=1の時、多項式次数を増やしながらStep_3 ～Step_6 を8回繰り返す
b PI=2の時、多項式次数を増やしながらStep_3 ～Step_6 を5回繰り返す
Step_3　階差計算式をパターン選定
Step_4　階差計算式のパターンと次数を選定
Step_5　階差計算 (最大の階差を取得)
Step_6　階差の収束判定 (0.0001以下の差)
多項式の最大次数は式のタイプによるり、各式の次数は
CMfiCalc::Select_g で定義。

c 収束した式のパターンと多項式の次数を temp クラスに代入する

d　収束しない場合は、PI を変更してループ計算をする。

e　それでも収束しない場合は、temp クラスに「0」を代入し手動計算とする。

```
temp->m_ma      = 0;
temp->m_pattern = 0;
```

⑦　CMfiCalc::Final_Step

⑧　CMfiCalc::Final_StepB　最小二乗法で補間する式を決める
下の関数（CMfiCalc::Select_g）を呼び出す。
更に、関数（MfiCalc3）、多項式の定数を計算し　temp に代入する。

⑨　CMfiCalc::Select_g　　最終式の選定
補間式の次数や定数を計算する
取るべき階差 $\Delta^m y^i$ を算定

MfiCalc2	CMfiCalc::f_1	F1 (Fomular 1)
	CMfiCalc::f_2	F2 (Fomular 2)
	CMfiCalc::f_3	F3 (Fomular 3)
	CMfiCalc::f_4	F4 (Fomular 4)
	CMfiCalc::f_5	F5 (Fomular 5)
	CMfiCalc::f_6	F6 (Fomular 6)
	CMfiCalc::f_7	F7 (Fomular 7)
	CMfiCalc::f_8	F8 (Fomular 8)

V の算出（多項式の各定数を計算

MfiCalc3	CMfiCalc::g_1	G_1 Fomular
	CMfiCalc::g_2	G_2 Fomular
	CMfiCalc::g_3	G_3 Fomilar
	CMfiCalc::g_4	G_4 Fomular
	CMfiCalc::g_5	G_5 Fomular
	CMfiCalc::g_6	G_6 Fomular
	CMfiCalc::g_7	G_7 Fomular
	CMfiCalc::g_8	G_8 Fomular
	CMfiCalc::g_9	G_9 Fomular
	CMfiCalc::g_10	G_10 Fomular
	CMfiCalc::g_11	G_11 Fomular
	CMfiCalc::g_12	G_12 Fomular
	CMfiCalc::g_13	G_13 Fomular

⑩　CMfiCalc::min_em
Step_Main 関数内でこの関数が呼ばれ、最小の階差式が代入される

階差の最も小さい多項式と、それに続く小さい多項式を計3つ選定する
emm[0]・emm[1]・emm[3]
この値は、ダイアログに表示され手動で選定する時の参考値とする。

⑪ CMfiCalc::Cal_3　　MfiCalc4.cpp
階差で選定した式と次数を使い、最小二乗法による補間
CMfiCalc::Sg_1
CMfiCalc::Sg_2
CMfiCalc::Sg_3
CMfiCalc::Sg_4
CMfiCalc::Sg_5
CMfiCalc::Sg_6
CMfiCalc::Sg_7
CMfiCalc::Sg_8
CMfiCalc::Sg_9
CMfiCalc::Sg_10
CMfiCalc::Sg_11
CMfiCalc::Sg_12
CMfiCalc::Sg_13

⑫ CMfiCalc::get_y　　MfiCalc4.cpp
補間した式と定数に連続した値を代入しシミュレーショングラフを描画

⑬ CMfiCalc::Cal_2　平均法法による補間　　MfiCalc5.cpp
補間するデータを二組のグループに分け、その積和を作る。
第一グループ
```
for(int i=1; k>=i; i++){
        xk += temp->m_XY[i-1].m_X;
        yk += temp->m_XY[i-1].m_Y;
}
```
第二グループ
```
for(int i=k+1; n>=i; i++){
        xn += temp->m_XY[i-1].m_X;
        yn += temp->m_XY[i-1].m_Y;
}
```
補間式は、Y = aX + b とし、
```
b = -(yn - (n-k)/k*yk) / ((n-k)/k*xk-xn);
a = (yk - bb * xk)/k;
```
で計算される。

式の形が簡単で、実験の途中段階である程度測定結果の解析や
評価が必要とされるような場合に使われる。

MFI プログラムの分析

データの入力

MfiダイアログにMsFlexGrid を貼り付け、グリッド上でEditする。

CMfiDlg::OnInitDialog()	Grid の初期設定
CMfiDlg::SetTitle(CMSFlexGrid& fg)	グリッドのタイトルの表示
CMfiDlg::PreTranslateMessage(MSG* pMsg)	キーボード
CMfiDlg::OnOK()	入力データの確定
CMfiDlg::OnMouseDownGrid	マウスボタンダウンした時
CMfiDlg::OnScrollGrid()	グリッドのスクロール
CMfiDlg::OnSetfocusInput()	
CMfiDlg::OnUpdateInput()	データの入力
CMfiDlg::SetMfiData()	データの設定
CMfiDlg::SetPt(int NewPt)	グリッドのNo.の設定
CMfiDlg::RecordData(int pt)	データをメモリーに記録する
CMfiDlg::OnDblClickGrid()	データの削除
CMfiDlg::MoveCurRight()	右矢印キー
CMfiDlg::MoveCurLeft()	左矢印キー
CMfiDlg::MoveCurDown()	下矢印キー
CMfiDlg::MoveCurUp()	上矢印キー
CMfiDlg::OnButtonMfi()	補間開始

補間内の動作

- CMfiDlg::sorting(void)　　データを小さい順にソーティングする
- CMfiDlg::GetMfiData(int fixedrows, int fixedcols)　　グリッドにデータを書き込む
  ソーティング後のデータをGetMfiData関数を使いグリッドに書き込む
- CMfiDlg::CombertDataToValue(void)

- m_cbIndex == TRUEの時　　最小二乗法補間ステップ

```
mfi.Step_Main(&m_tempDlg);
m_ctrGraph.m_TpG.m_XY.Copy(m_tempDlg.m_XY);
```

グラフを描画するデータの書き込み

CMfiDlg::ApproxGraph(int n)

それ以外は、　　平均法による補間

```
mfi.Cal_2(&m_tempDlg);
ApproxGraph2(nXY);
```

グラフの描画

3つの基本クラス

StaticGraph	グラフが描画されるPicture Control に関連付けされしたクラス
MfiGraph	StaticGraphクラスに連動する一般クラス
MfiFunction	グラフ描画に使われる計算を集めた一般クラス

StaticGraph は、Picture Control に関連付けたクラス

CArray <GraphXY, GraphXY> m_GXY;　補間データ保持用にアレー

CMfiTemp m_TpG;　（仮変数クラス CMfiTemp)の中間変数のクラス

m_status = FALSE;　補間完了でTRUEとなる

m_dev = 1000;

- CStaticGraph::OnPaint()

元データを作業用データアレーにコピー

MfiGraph は、描画関数をコーディングしている。

● グラフに描画する補間データのアレーを定義

CArray <GraphXY, GraphXY> m_GXY; // 補間されたデータ
CArray <GraphXY, GraphXY> m_TpG; // 元のデータ

● CMfiGraph::MainGraph 初期設定

1 スクリーンサイズを(1000,1000)に初期設定
2 描画モードの設定 pDC->SetMapMode(MM_ANISOTROPIC);
3 元データアレーを作業用アレー(m_XY MfiFunc で定義)にコピー

```
fn.m_XY[i].m_X = m_TpG[i].m_X;
fn.m_XY[i].m_Y = m_TpG[i].m_Y;
```

4 座標グラフの最大・最小値を取得し、グラフのスケールを設定

スクーリンのサイズ倍率を計算

```
double mx = base_mm.xmax / 1000.0;
double my = base_mm.ymax / 1000.0;
```

グラフ描画の倍率を計算して m_Gmg に保存
mx 及びmyが 1以下の時は、倍率が 1
それ以上のとき、大きいほうが倍率

5 マージを含んだスクリーンサイズ再設定

6 pDC->SetWindowExt 描画エリアウィンドウの x 範囲と y 範囲を設定します。
7 pDC->SetViewportExt 描画エリアビューポートの x 範囲と y 範囲を設定します。
8 pDC->SetViewportOrg 描画エリアビューポートの原点を設定します。

9 グラフエリアを白色で塗りグラフペーパとする
10 Mag mg = fn.GetMagnification Mag 構造体に倍率を考慮した座標を取得

11 fn.SetZero 座標グラフの基点を設定
12 Graph_Paper グラフの座標を描く
13 DrawAppGraph グラフを描画
補間後のデータがグラフ用紙からはみ出る部分を描画しないようにしている

● CMfiGraph::Graph_Paper グラフ用紙の作成

● CMfiGraph::DrawAppGraph
元のデータのプロット
補間データのプロット

● CMfiGraph::GetPoint Plot する座標を得る
● CMfiGraph::P_Set 点をプロットする
● CMfiGraph::DrawLine ラインを描く
● CMfiGraph::Base_ValueX グラフの値を描く
● CMfiGraph::Base_ValueY グラフの値を描く
● CMfiGraph::BackColor グラフのバックを塗る

MfiFunctionは、グラフの目盛りや描画する為の基点などを計算するクラス

● MfiFunction で定義されている構造体

MaxMin 構造体

```
double xmax;  補完されたX最大値
double xmin;  Xの最小値
double ymax;  補完されたY最大値
double ymin;  Yの最小値
BOOL logX;    対数表現が必要か否か
BOOL logY;
```

GraphXY 構造体
　　double m_X;　補間する元データ
　　double m_Y;
Mag 構造体
　　int magX;　　グラフ描画の倍率
　　int magY;

CArray <GraphXY, GraphXY> m_XY;　　構造体GraphXYを持ったアレー

● 各種の関数

各関数は、作業用アレーm_XYのデータを使う為、事前にデータのコピーが必要

① CMfiFunc::SetZero　　0点座標のセット

引数	MaxMin *base_mm	最大最小値
	Mag *mg	倍率
	long off_x	描画の点
	long off_y	
返値	CPoint	グラフのゼロ点

② CMfiFunc::GetMagnification　倍率を考慮したグラフ座標を計算

③ CMfiFunc::GetMaxMin　　補間する元データから最大・最小値の取得
　　最大値/最小値）が20以上だと対数表示とする

④ CMfiFunc::SetGraphScale　　表示グラフの基点スケールをセット
　　対数表示の時、基点は1の倍数
　　戻り値は、 MaxMin構造体 base_mm

⑤ CMfiFunc::GetGraphScale　　グラフの目盛りを計算する

⑥ CMfiFunc::GetBaseX_min　　グラフの最小基点　Xを計算
　　X が10以上の時　　対数表示または通常
　　X が 0の時
　　X が10以下の時　　対数表示または通常
　　X が負の値の時

⑦ CMfiFunc::GetBaseX_max　　グラフの最大基点　Xを計算
　　X が1以上の時　　対数表示または通常
　　X が 0の時
　　X が少数以下の時　対数表示または通常
　　X が負の値の時

⑧ CMfiFunc::GetBaseY_min　　グラフの最小基点　Yを計算
　　Y が10以上の時　　対数表示または通常
　　Y が 0の時
　　Y が10以下の時　　対数表示または通常
　　Y が負の値の時

⑨ CMfiFunc::GetBaseY_max　　グラフの最大基点　Yを計算
　　Y が1以上の時　　対数表示または通常
　　Y が 0の時
　　Y が少数以下の時　対数表示または通常
　　Y が負の値の時