波形解析ソフトウェア imc FAMOS 頻度処理セミナー

Rev.C - 2024/01/10

株式会社 東陽テクニカ

機械計測部　〒103-8284 東京都中央区八重洲 1-1-6

機械技術課　TEL:03-3279-0771 (直通:03-3245-1104) FAX:03-3246-0645

https://www.toyo.co.jp/mecha/　e-mail:imc@toyo.co.jp

目次

改訂履歴		
改訂日	版数	改訂内容
2023/11/24	A	初版
2023/11/29	B	4.1.3 節：修正マイナー則を考慮する、を新規に追加
2024/01/10	C	多チャンネルの場合の解析具体例を追加

1.イントロダクション

imc FAMOS 頻度処理セミナーにご参加いただき誠にありがとうございます。

1989 年に windows 波形解析ソフトウェアとして開発された imc FAMOS は PC ベースでの波形データの観察、解析、レポート作成をサポートします。全世界では 30,000 ライセンス以上、50 以上のカンパニー&サイトライセンスの実績を持ち、計測エンジニアや CAE エンジニアの方々へ幅広くご使用いただいております。およそ 100 種類を超えるインポートフィルタやマウスで操作可能なカーブウィンドウ、数百の関数群を組み合わせることで、現場での解析業務を飛躍的に効率化することができるソフトウェア環境です。

本セミナーでは imc FAMOS における頻度解析の実施方法を学びます。
頻度解析の実行には、imc FAMOS Enterprise のライセンスが必要となります。

頻度解析は、振動等のデータを「どの程度の大きさが」「どれくらいの回数あるか」分類する手法です。どの程度の大きさが、という概念は「クラス」と呼ばれます。
この解析結果を利用する典型的な例としては疲労寿命推定があります。

本セミナーでは、大きく分けて以下のトピックをご説明します。

2 章：クラスカウント法
3 章：レインフロー法
4 章：疲労寿命推定

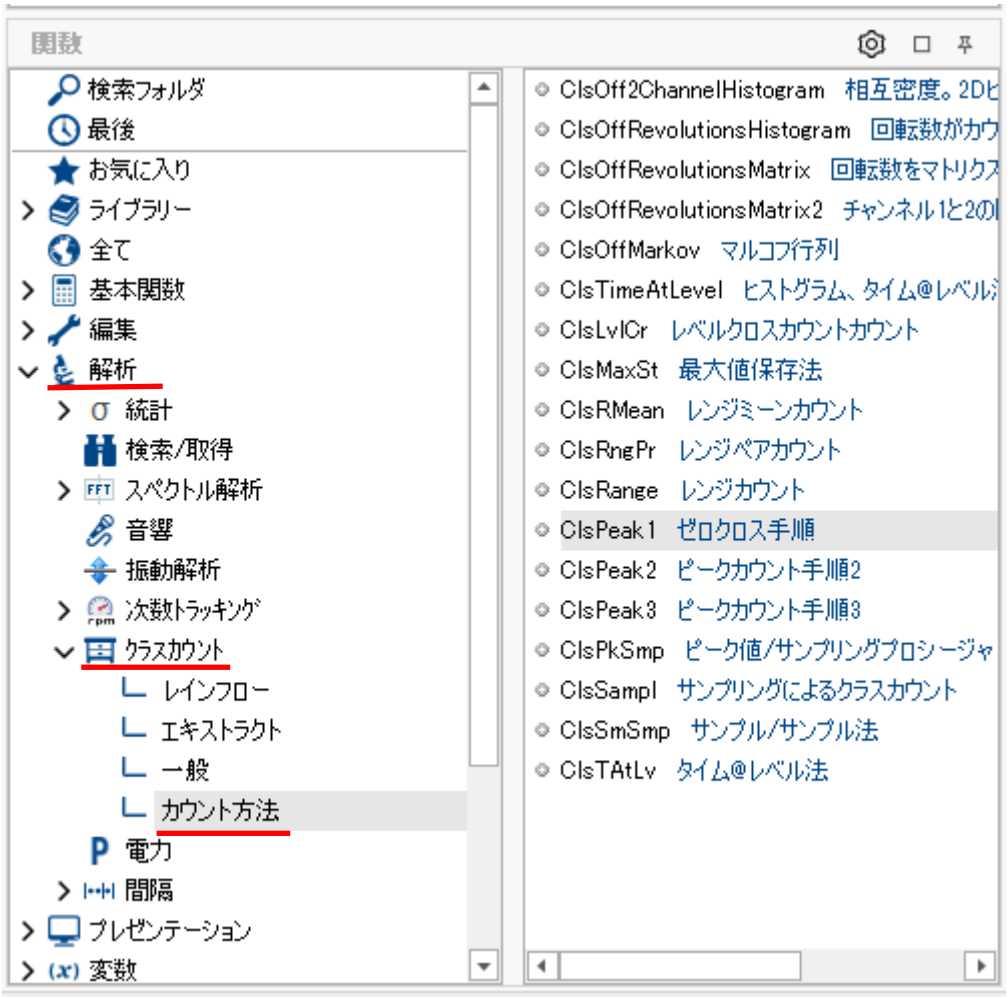
本ユーザートレーニングの内容についての不明点、ソフトウェアの不具合を発見された場合は、お手数ですが
”5 章 サポート”に示す当社までご連絡ください。ドイツ imc 社と株式会社東陽テクニカが継続的にサポート致します。

最後に実車計測システム/テストベンチシステム/データサーバー等について何かお困りのことがありましたら何なりとご相談ください。実践的なノウハウを持つエンジニアが貴社の業務をお手伝い致します。

2.クラスカウント法

クラスカウント法は、データの**極値**をカウントしていく手法です。このセミナーでは、クラスカウント法の代表的な関数として ClsPeak1、ClsPeak2、ClsPeak3 の 3 つを説明していきます。

クラスカウント法の関数は、imc FAMOS の関数ボックスの[解析 > クラスカウント > カウント方法]のカテゴリーにまとめられています。



3 つの関数の概要は以下の通りです。

ClsPeak1：ある参照ラインを設定し、そのラインを超えた 2 点間の最も大きい極値のみをカウントします。

ClsPeak2：ある参照ラインを設定し、そのラインを超えたすべての極値をカウントします。

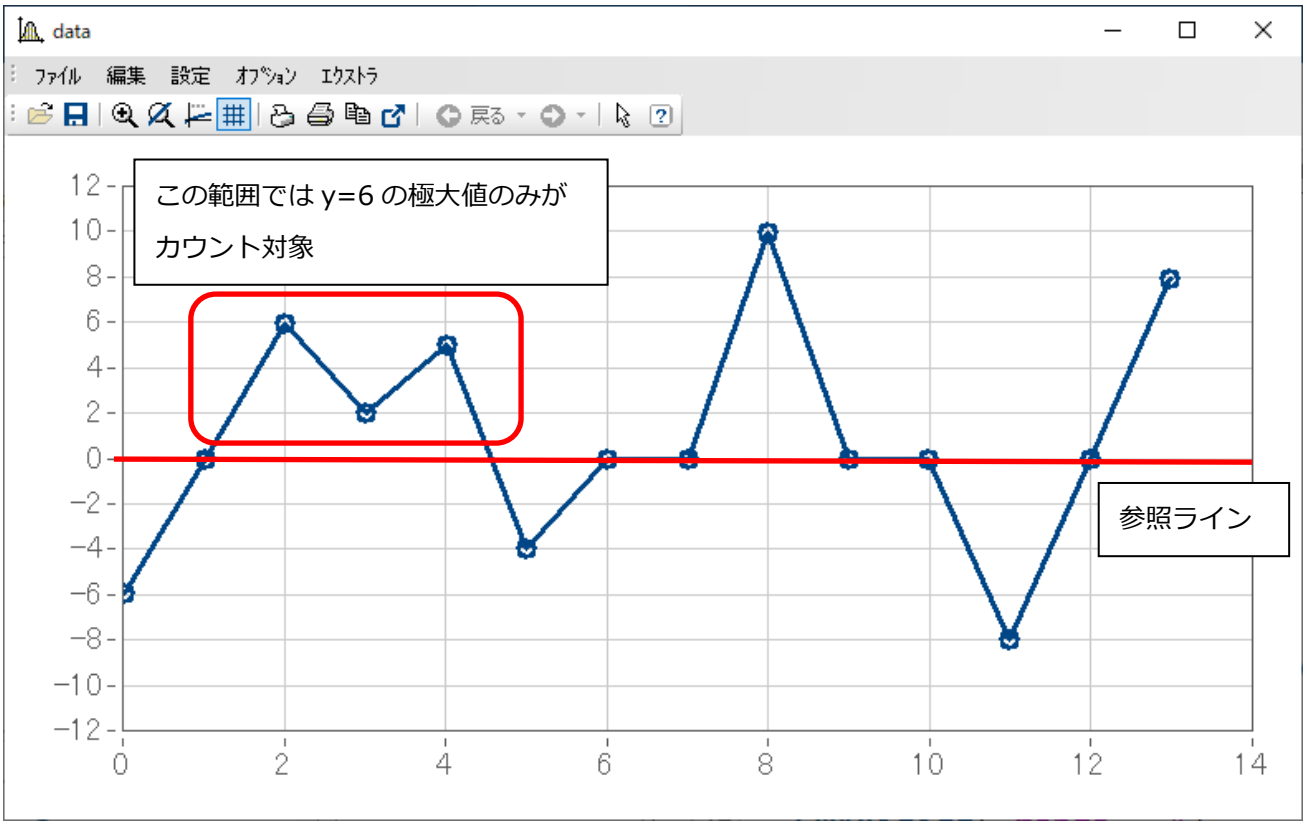
ClsPeak3：すべての極値を、極大値と極小値でそれぞれ別個にカウントします。

2.1.ClsPeak1 関数

2.1.1.カウント方法について

ClsPeak1 関数は、ある参照ラインを設定し、そのラインを超えた 2 点間の最も大きい極値のみをカウントする、という手法を取ります。

例として下図のデータで、参照ラインとして 0 の場合を考えます。
そうすると、最初の山は y=6 と y=5 の 2 つの極大値の両方が、「参照ラインを超えた 2 点間」に含まれています。そのため、ここではその中で最も大きい極値である **y=6 のみがカウント**されます。
なお、y=2 に極小値も存在していますが、極小値は逆に「参照ラインを下回った 2 点間」をカウントしていくため、この極小値はカウント対象となりません。

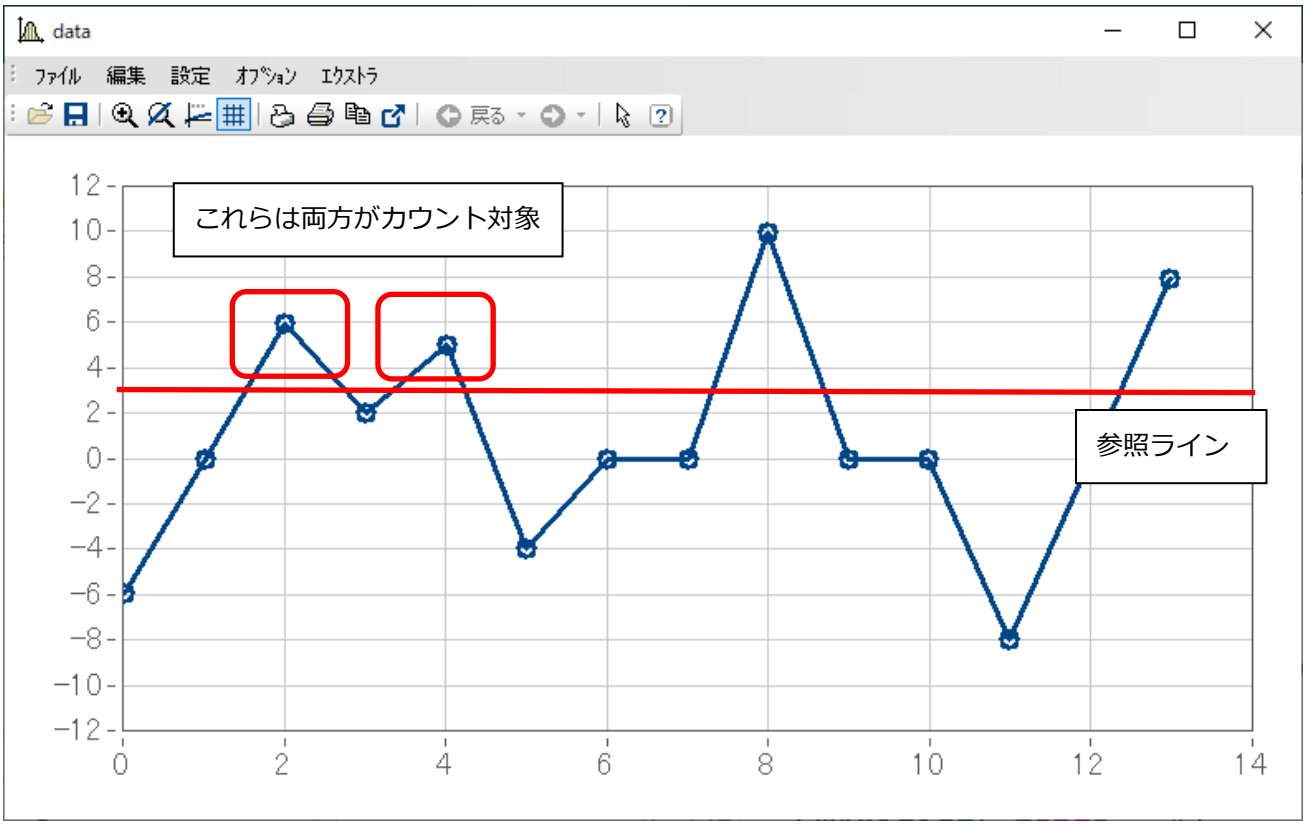


参照ラインを 0 とした場合、正の領域ではある振幅の最大値、負の領域ではある振幅の最小値を取得するような挙動となります。
一般的に**最大最小法**と呼ばれる計数方法です。

同じデータで、参照ラインとして 3 の場合としてみましょう。

そうすると、y=6 と y=5 の 2 つの極大値は、それぞれ別の「参照ラインを超えた 2 点間」に含まれています。そのため、この状態では **y=6 の極値も y=5 の極値も両方カウント**されます。

(先ほどの例とは逆に、y=2 の極小値もカウント対象となります)



同様の考え方が他の極大値、極小値すべてに対しても行われてカウントされていきます。

なお、データの始点/終点もカウントの対象となります。

2.1.2.関数のパラメータについて

ClsPeak1 関数のパラメータは以下のようになります。

Result = ClsPeak1(data, MaxValue, MinValue, Number of bins, Reference, Hysteresis, Options)

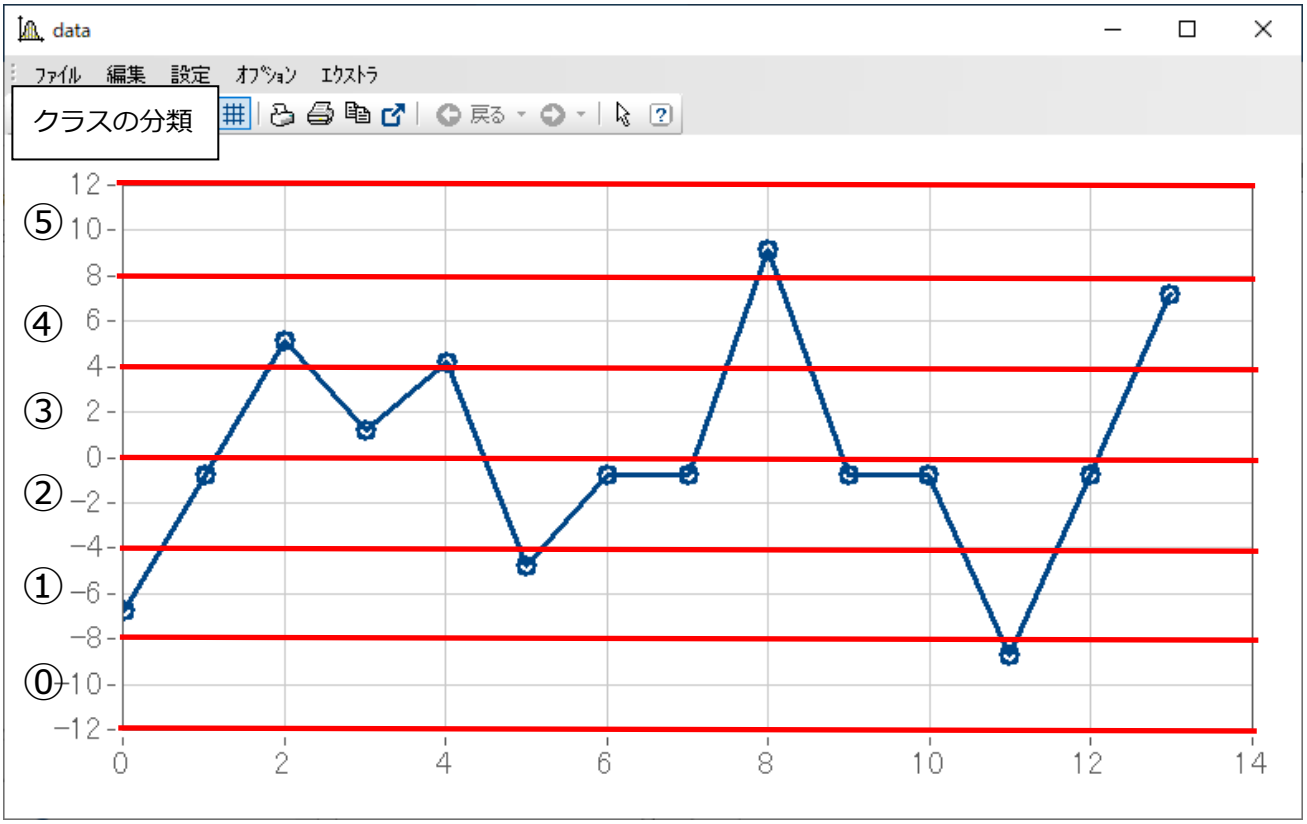
Result	カウント結果の変数名
data	カウント対象データの変数名
MaxValue	カウントするレンジの上限
MinValue	カウントするレンジの下限
Number of bins	分類するクラスの数
Reference	参照レベル
Hysteresis	小さな振動を無視するためのヒステリシスの大きさ
Options	カウントする際の挙動に関するオプション設定

MaxValue, MinValue, Number of bins は、クラスの設定に関するパラメータです。

例として、下図のように-12 から+12 までの範囲を 6 個のクラスにするとします。この時のパラメータは

MaxValue	12
MinValue	-12
Number of bins	6

となります。(imc FAMOS のクラスは標準では 0 から始まります)



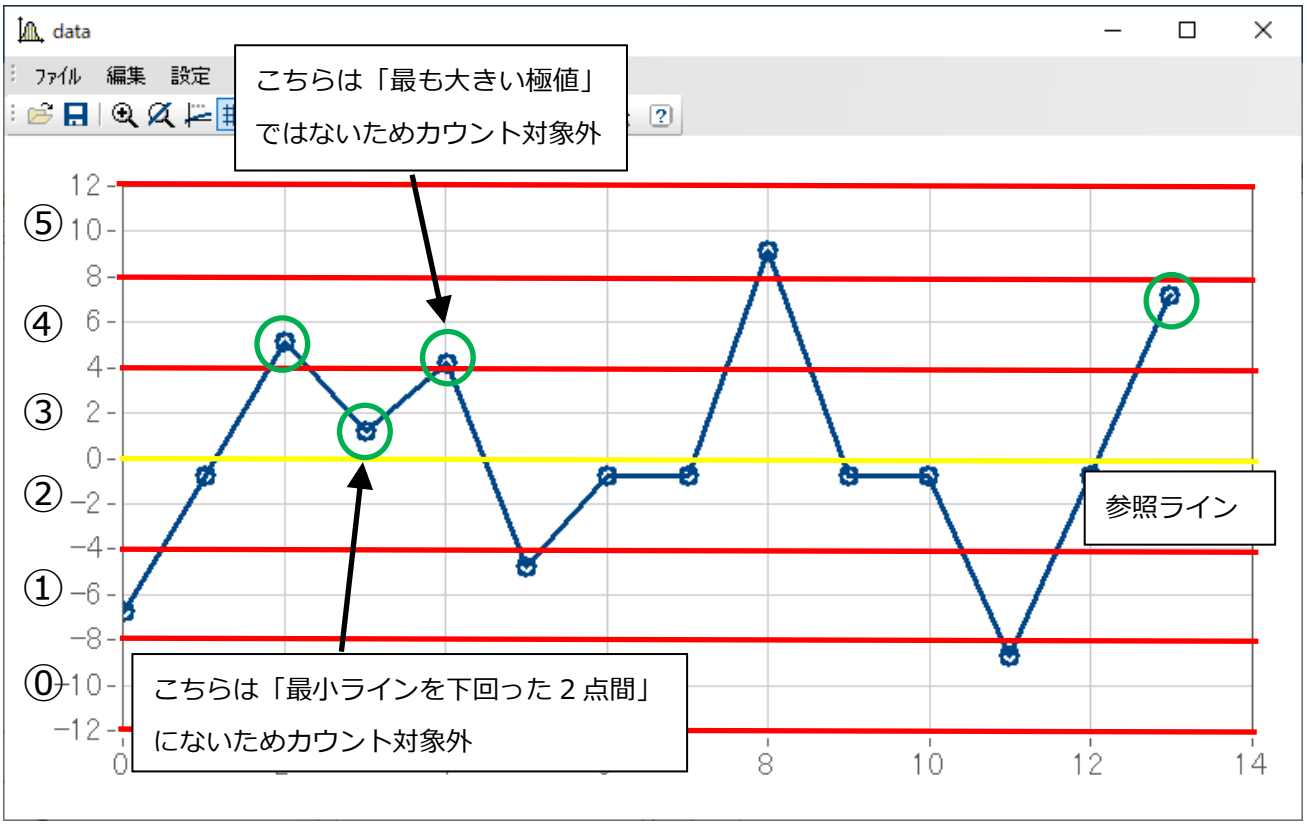
実際の解析例は、下図のようにどのクラスにいくつの極値がカウントされたか、という系列となります。この解析は Reference=0 で行っています。

なお、このデータは分類のわかりやすさのために、2.1.1 節のデータとは多少値を変更しています。(クラスの境界に値が乗らないよう)

[Classes]	result
0	1.0
1	2.0
2	0.0
3	0.0
4	2.0
5	1.0

ここで、波形としては③の領域に極小値が 1 つ、④の領域に極大値が 3 つありますが、カウント結果ではそれぞれ③が 0、④が 2 となっています。
これは 2.1.1 節の説明にあるように、参照ラインとカウント方法の兼ね合いによるものです。

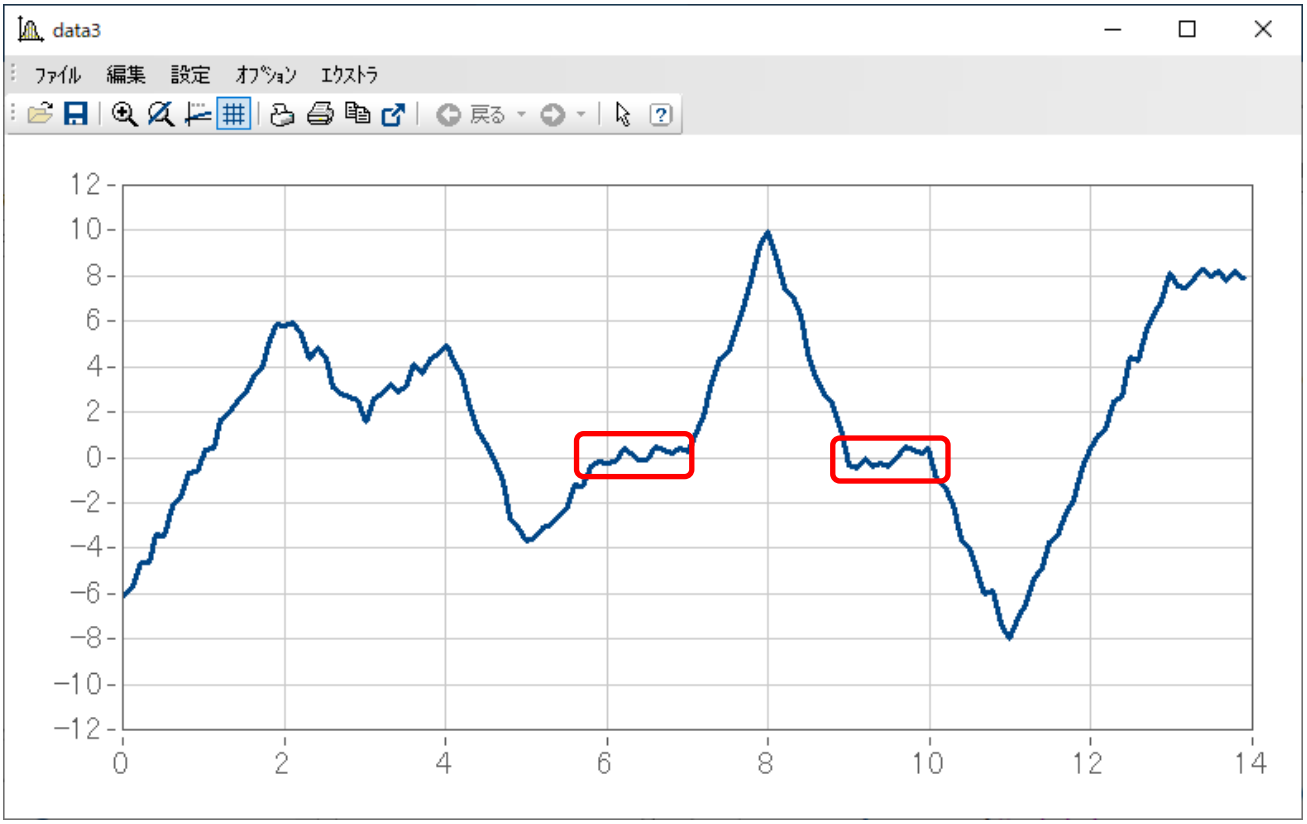
③の極小値については、「参照ラインを下回った 2 点間」に無いため(上回った領域にあるため)、カウントの対象外となっています。
④の最初の極大値 2 つは、どちらも同じ「参照ラインを上回った 2 点間」に属しているため、最も大きい極大値 1 つのみがカウント対象となっています。



[Classes]	result
0	1.0
1	2.0
2	0.0
3	0.0
4	2.0
5	1.0

パラメータの説明に戻って、Reference(参照レベル)の考え方は 2.1.1 節を参照してください。
0 を中心に正負に振れるようなデータであれば、0 としておくのがよいでしょう。また、中心が 0 で無いデータに対しては、data = data - Mean(data)のように、平均値を引いて 0 中心のデータに加工したほうが扱いやすいかもしれません。

Hysteresis は微小な振動(ノイズ等)を無視させるためのパラメータです。
実際の計測データは基本的にノイズ分が含まれており、例えば下図のようなデータを参照ライン=0 としてそのままカウントした場合、0 付近に存在するノイズ成分で余分なカウント結果が出現します。
ここで Hysteresis=1 のように設定すると、振幅が 1 に満たない微小な振動を無視するようになり、実際に欲しい解析結果に近い値が得られるようになります。
実用上は、波形を参照して適当な値を入力するか、1 クラスの幅そのものとしておくのがよいです。



最後に、options はオプション設定用の値で、以下の 4 つが存在します。

0：オプション設定なし。

1：クラスの上限/下限の外にあるデータもカウントの対象とします。
例えば、クラスの上限が 8 で y=10 の極値がある場合、この極値は上限のクラスに含まれるものとしてカウントします。逆に options=0 の場合はこの極値は無視されます。

2：参照レベルを関数が自動的に決定します。Reference で設定した値は無視されます。

3：1 と 2 のオプションを同時に適用します。

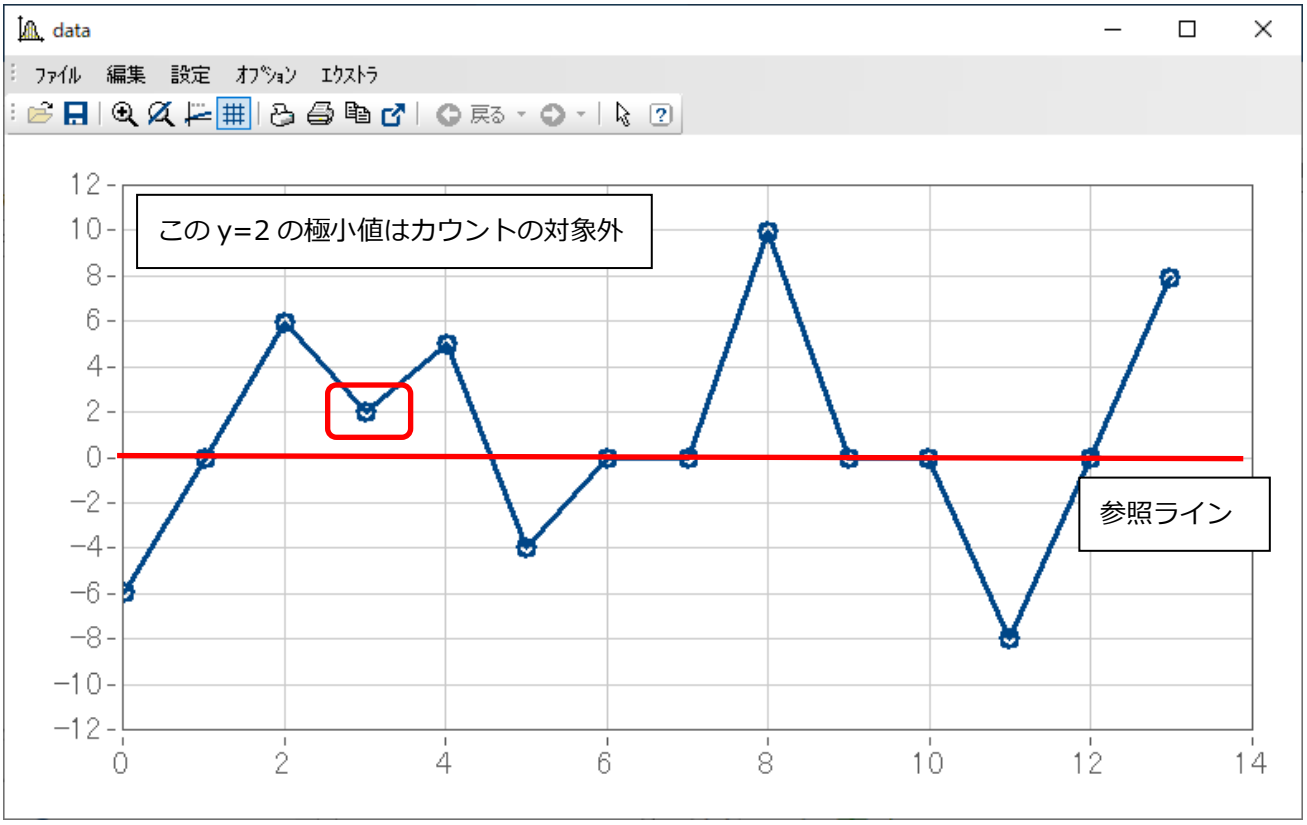
基本は 0 または 1 で良いと思われます。

2.2.ClsPeak2 関数

2.2.1.カウント方法について

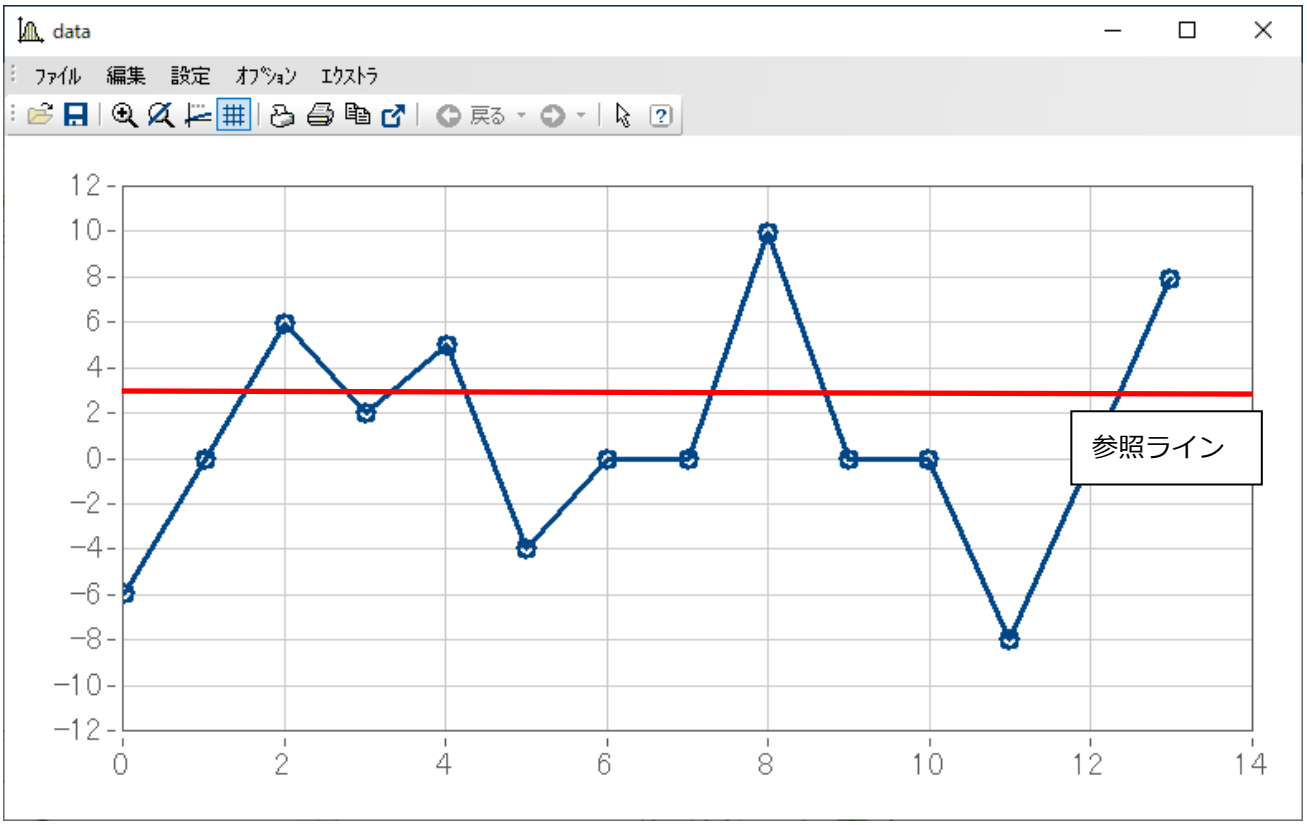
ClsPeak2 関数は、ある参照ラインを設定し、そのラインを超えたすべての極値をカウントする、という手法を取ります。

例として下図のデータで、参照ラインとして 0 の場合を考えます。
ここで「ある参照ラインを設定し、そのラインを超えた」という考え方ですが、極大値の場合は参照ラインよりも上にあること、極小値の場合は参照ラインよりも下にあることを意味します。
つまり、このデータにおいては **y=2 の極小値のみはカウントの対象外**となります。



ClsPeak1 関数とは異なり、最初の 2 つの極大値は両方ともカウントの対象となります。

例として参照ラインを 3 とするなら、このデータにおいてはすべての極大値・極小値がカウント対象となります。



データの始点/終点もカウントの対象となります。

2.2.2.関数のパラメータについて

ClsPeak2 関数のパラメータは以下のようになります。

Result = ClsPeak2(data, MaxValue, MinValue, Number of bins, Reference, Hysteresis, Options)

Result	カウント結果の変数名
data	カウント対象データの変数名
MaxValue	カウントするレンジの上限
MinValue	カウントするレンジの下限
Number of bins	分類するクラスの数
Reference	参照レベル
Hysteresis	小さな振動を無視するためのヒステリシスの大きさ
Options	カウントする際の挙動に関するオプション設定

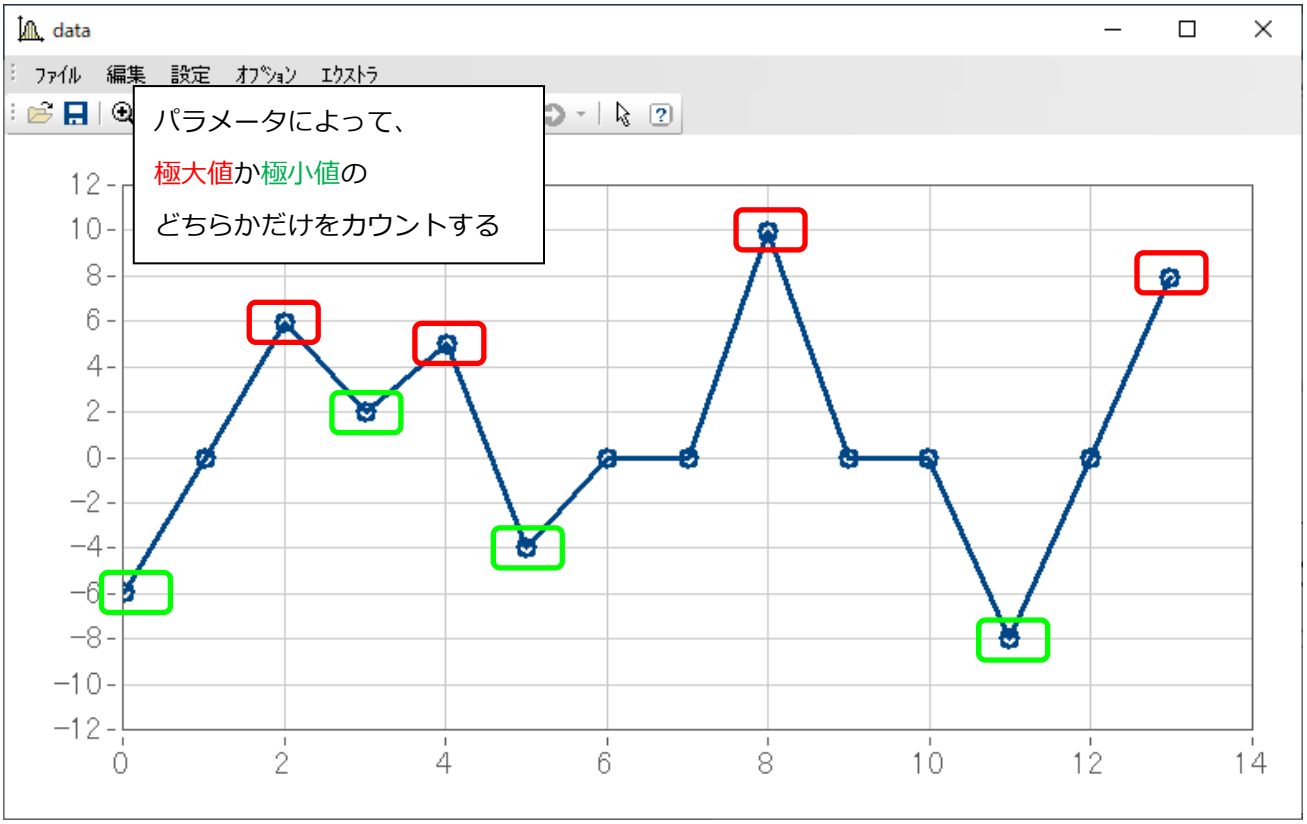
これらのパラメータの内容はすべて 2.1.2 節の ClsPeak1 関数と同様です。

2.3.ClsPeak3 関数

2.3.1.カウント方法について

ClsPeak3 関数は、すべての極値を、極大値と極小値でそれぞれ別個にカウントする、という手法を取ります。

それぞれ別個に、というのは後述するパラメータの設定に応じて**「極大値のみをカウントする」**解析モードと、**「極小値のみをカウントする」**解析モードが存在するということを意味します。
例として下図のデータの場合、パラメータ設定に応じて赤枠で囲った極大値のみをカウントする、または緑枠で囲った極小値のみをカウントする、というどちらかの挙動となります。



データの始点/終点もカウントの対象となります。始点/終点が極大値/極小値のどちらとしてカウントされるかはデータの形状に依存します。
上図のデータの例では、始点は極小値、終点は極大値として扱われます。

極大値(Peak)のカウントと極小値(Valley)のカウントを合算した場合、一般的に **PeakValley 法**と呼ばれる計数方法にあたります。

2.3.2.関数のパラメータについて

ClsPeak3 関数のパラメータは以下のようになります。

Result = ClsPeak3(data, MaxValue, MinValue, Number of bins, Hysteresis, Options)

Result	カウント結果の変数名
data	カウント対象データの変数名
MaxValue	カウントするレンジの上限
MinValue	カウントするレンジの下限
Number of bins	分類するクラスの数
Hysteresis	小さな振動を無視するためのヒステリシスの大きさ
Options	カウントする際の挙動に関するオプション設定

Options 以外のパラメータについては 2.1.2 節の ClsPeak1 関数と同様です。(Reference は存在しません)

Options はオプション設定用の値で、以下の 4 つが存在します。

0：極大値のみをカウントします。

1：クラスの上限/下限の外にあるデータもカウントの対象とし、極大値のみをカウントします。
例えば、クラスの上限が 8 で y=10 の極値がある場合、この極値は上限のクラスに含まれるものとしてカウントします。逆に options=0 の場合はこの極値は無視されます。

2：極小値のみをカウントします。

3：クラスの上限/下限の外にあるデータもカウントの対象とし、極小値のみをカウントします。

基本的にクラスカウントを行う場合は極大値も極小値もカウント対象とすることが多いので、ClsPeak3 関数で頻度処理を行う場合は、Options の 0 または 1 でカウントした結果と、Options の 2 または 3 でカウントした結果を合算します。

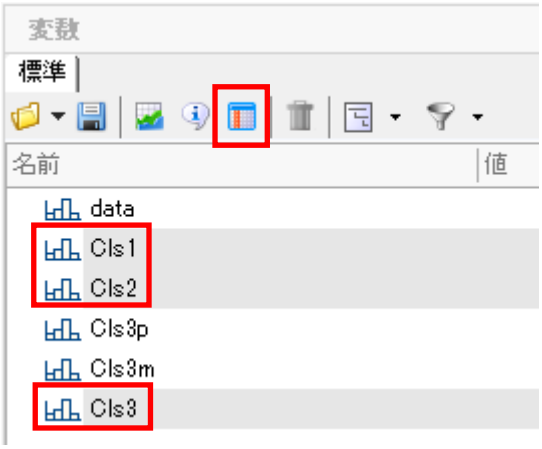
2.4.演習：各 ClsPeak 関数の結果比較

では各 ClsPeak 関数の違いを実際に確認するために、下記のシーケンスを実行してみましょう。

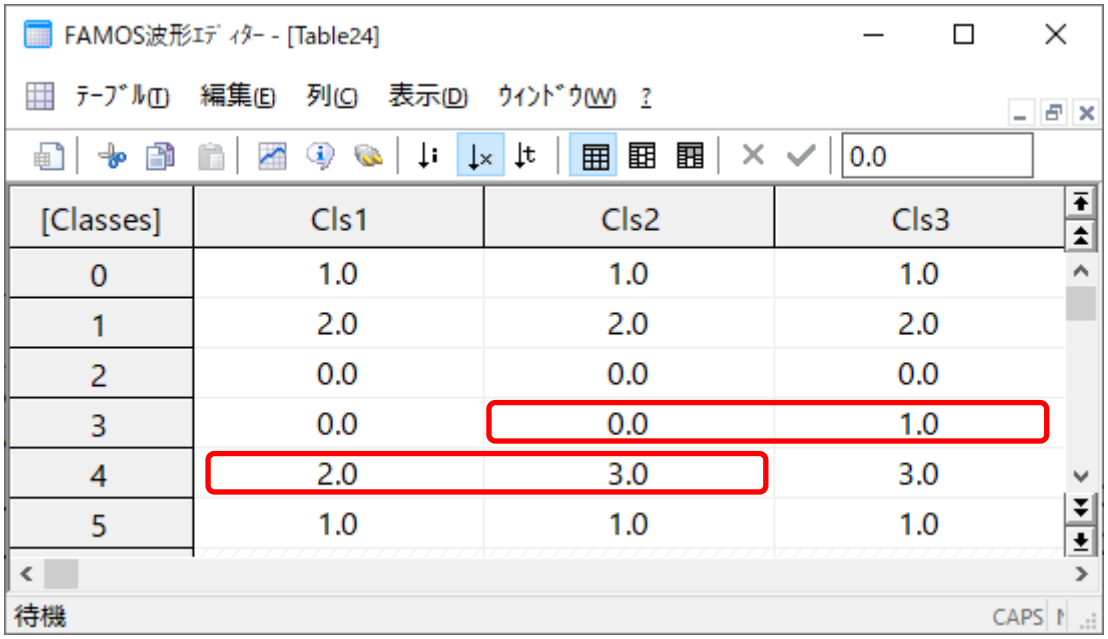
テストデータの波形形状自体は、今までの説明に用いてきたものと同じです。クラスの境界に乗るとどのクラスに分類されているかがわかりにくいので、少しだけ値をずらしています。

```
------------------------------------------------------------------------
; テストデータ定義
; クラスの境界そのものに乗らないように少しだけずらす
data = [-6, 0, 6, 2, 5, -4, 0, 0, 10, 0, 0, -8, 0, -8]
data = data - 0.5

; パラメータ
_MaxValue     = 12
_MinValue     = -12
_NumberOfBins = 6
_Reference    = 0
_Hysteresis   = 0
_Options      = 0

; 各 ClsPeak 関数でカウント
; ClsPeak3 だけは極大/極小を合算
Cls1   =   ClsPeak1(data,   _MaxValue,   _MinValue,   _NumberOfBins,   _Reference,
_Hysteresis, _Options)
Cls2   =   ClsPeak2(data,   _MaxValue,   _MinValue,   _NumberOfBins,   _Reference,
_Hysteresis, _Options)
Cls3p = ClsPeak3(data, _MaxValue, _MinValue, _NumberOfBins, _Hysteresis, 0)
Cls3m = ClsPeak3(data, _MaxValue, _MinValue, _NumberOfBins, _Hysteresis, 2)
Cls3 = Cls3p + Cls3m

Del _*
------------------------------------------------------------------------
```

実行出来たら、結果の Cls1、Cls2、Cls3 を選択してデータエディタで表示してみましょう。



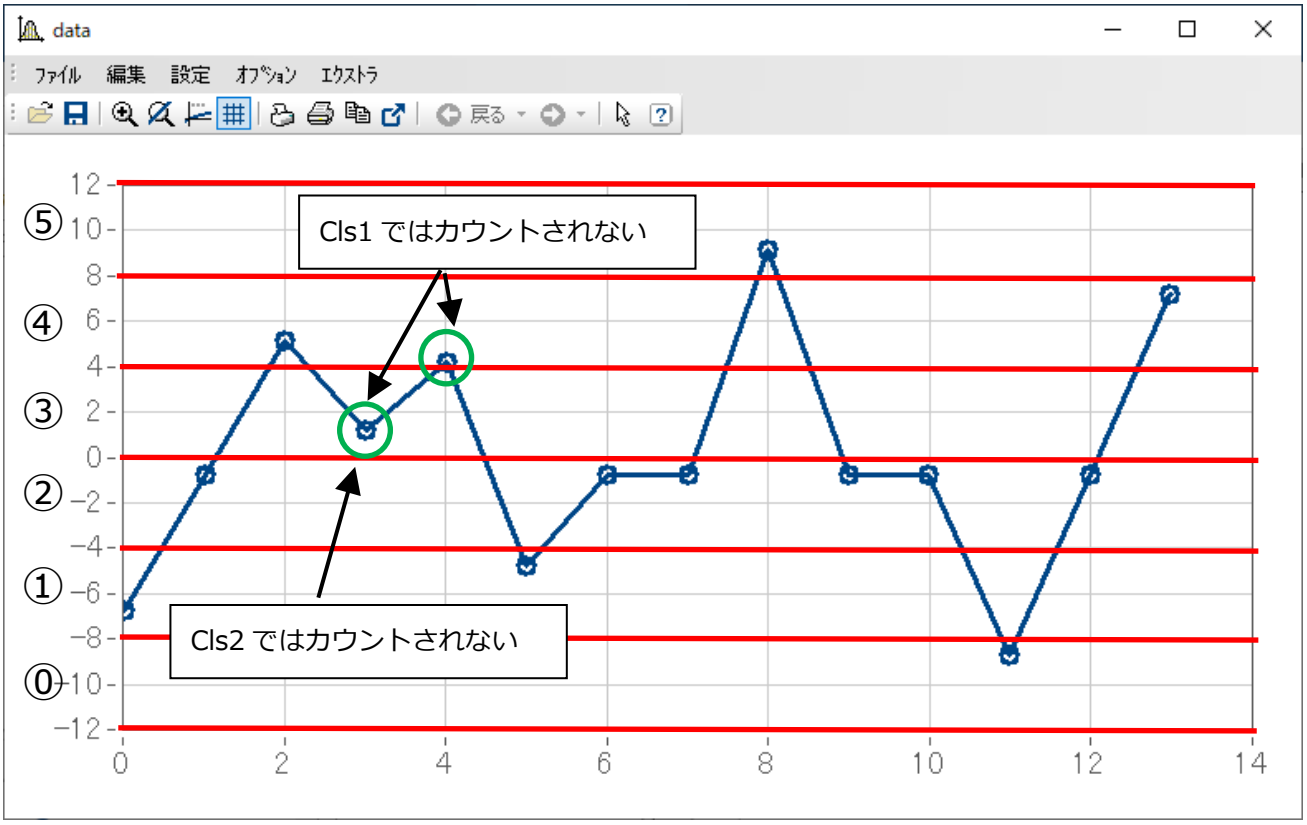
結果は下図のようになっているはずです。

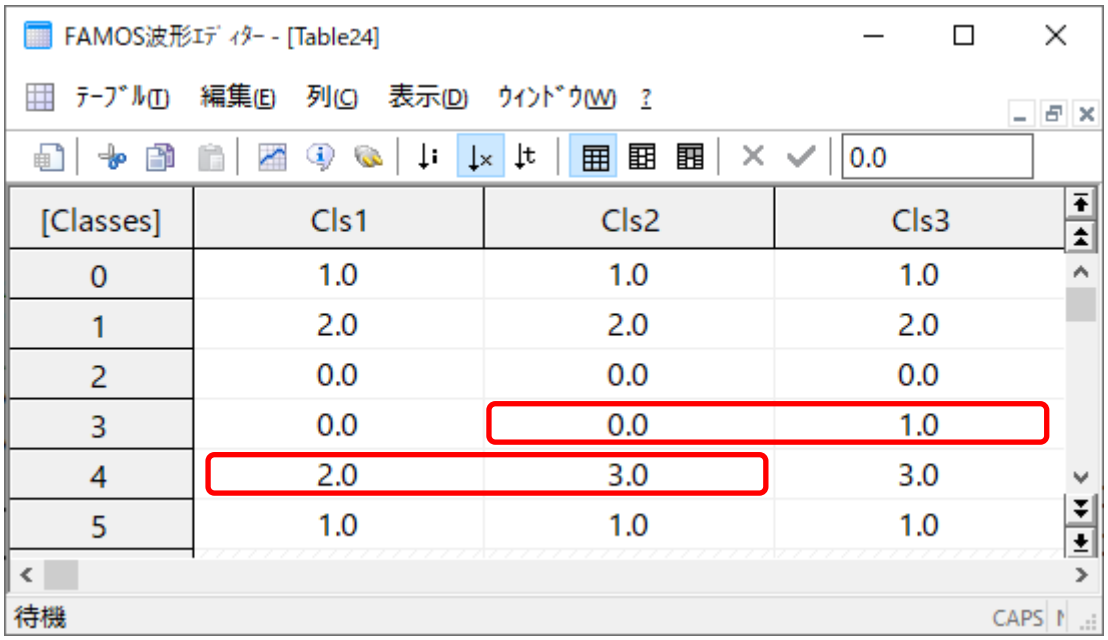


[Classes]	Cls1	Cls2	Cls3
0	1.0	1.0	1.0
1	2.0	2.0	2.0
2	0.0	0.0	0.0
3	0.0	0.0	1.0
4	2.0	3.0	3.0
5	1.0	1.0	1.0

それぞれの結果を比較すると、Cls2 が Cls1 よりも 4 クラス目のカウントが 1 多く、Cls3 が Cls2 よりも 3 クラス目のカウントが 1 多くなっています。

それぞれの結果の違いを、対象データとクラス分類の領域を並べて図示してみます。





[Classes]	Cls1	Cls2	Cls3
0	1.0	1.0	1.0
1	2.0	2.0	2.0
2	0.0	0.0	0.0
3	0.0	0.0	1.0
4	2.0	3.0	3.0
5	1.0	1.0	1.0

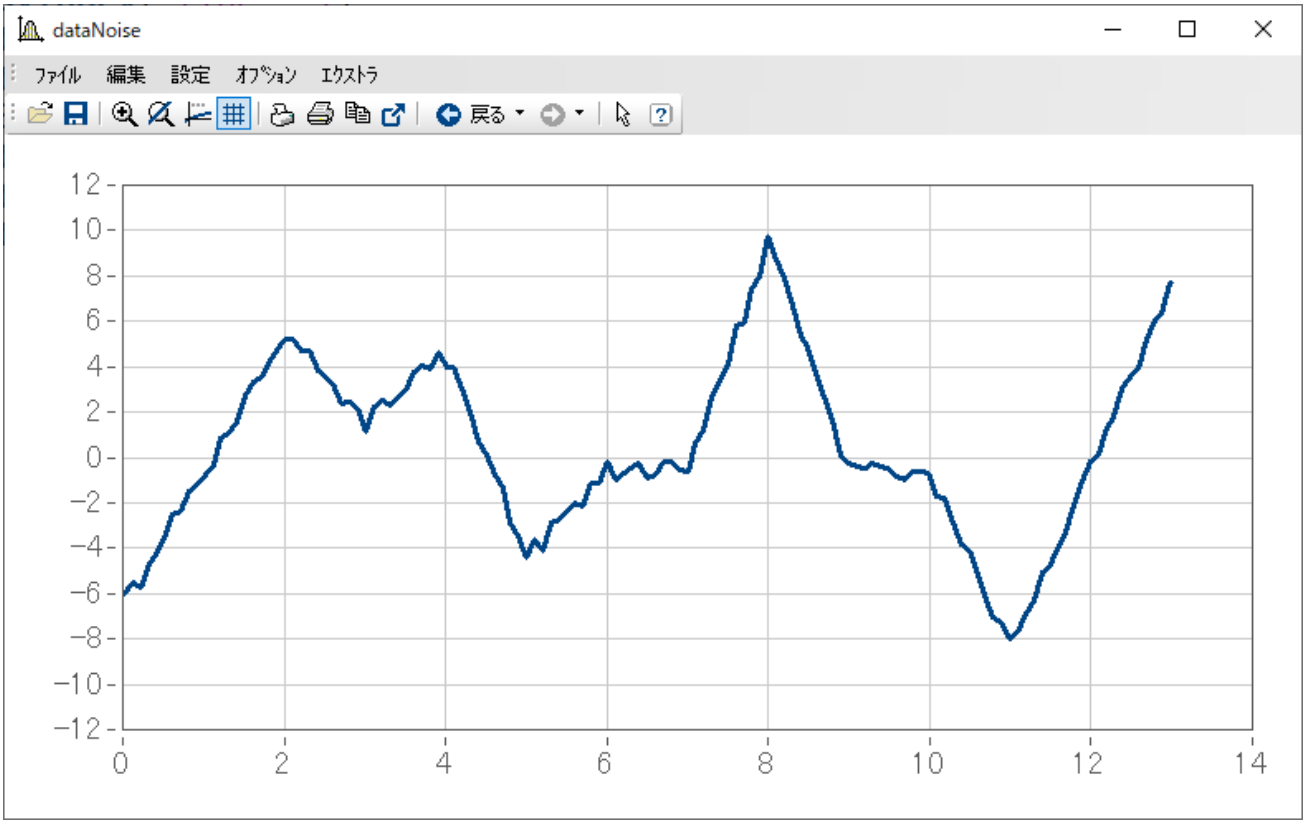
それぞれのカウント方式から、Cls1 と Cls2 ではそれぞれ図示した極値がカウントの対象外となっています。クラスカウントの方式は考え方の違いであり、どれが正しいという性質のものではありませんので、実行したい解析内容によって使い分けてください。

2.5.演習：ノイズ有りデータとヒステリシス

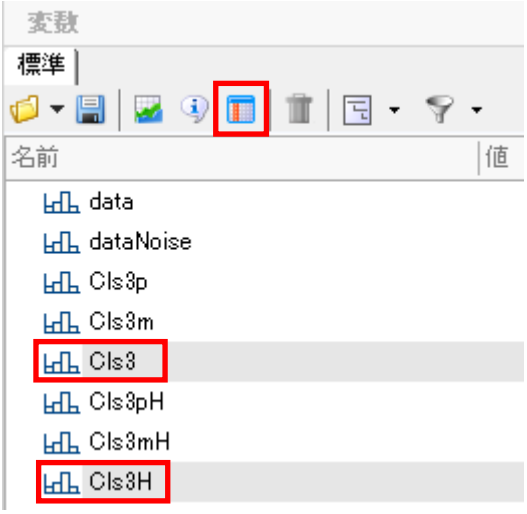
次に、実際の計測データと傾向が近いノイズ有りデータと、パラメータのヒステリシスによるカウント結果の様子を確認してみましょう。わかりやすさのため、すべての極値をカウントする ClsPeak3 関数を使用します。

```
----------------------------------------------------------------------
; テストデータ定義
data = [-6, 0, 6, 2, 5, -4, 0, 0, 10, 0, 0, -8, 0, 8]
data = data - 0.5

; 仮想的に振幅±0.5 程度のノイズ成分を加える
_noise = Random(131, 0, -0.5, 0.5, 1)
_noise = Xdel(_noise, 0.1)

; ノイズ有りデータ
dataNoise = RsampEx(data, _noise, 0, 0) + _noise

; パラメータ
_MaxValue     = 12
_MinValue     = -12
_NumberOfBins = 6

; 適当な大きさのヒステリシスを適用する
_Hysteresis   = 1

; ヒステリシスの有無で比較
Cls3p = ClsPeak3(dataNoise, _MaxValue, _MinValue, _NumberOfBins, 0, 0)
Cls3m = ClsPeak3(dataNoise, _MaxValue, _MinValue, _NumberOfBins, 0, 2)
Cls3 = Cls3p + Cls3m

Cls3pH = ClsPeak3(dataNoise, _MaxValue, _MinValue, _NumberOfBins, _Hysteresis,
0)
Cls3mH = ClsPeak3(dataNoise, _MaxValue, _MinValue, _NumberOfBins, _Hysteresis,
2)
Cls3H = Cls3pH + Cls3mH

Del _*
----------------------------------------------------------------------
```

このシーケンスで作成されるノイズ有りデータは下図のような波形となっています。



ヒステリシス無しの結果 Cls3 と、ヒステリシスを適用した Cls3H を選択してデータエディタで表示してみましょう



結果は下図のようになり、ヒステリシスを使用しない場合はカウント結果がかなり大きくなることがわかります。ヒステリシスを適用した場合は演習 2.4 のようなノイズ無しデータをカウントした場合と同様の結果となっています。

[Classes]	Cls3	Cls3H
0	1.0	1.0
1	5.0	2.0
2	13.0	0.0
3	6.0	1.0
4	6.0	3.0
5	1.0	1.0

待機

なお、ノイズの形状やヒステリシスの大きさによっては、ノイズ無しの理想的データ(演習 2.4 のような)をカウントした結果と比較して、カウント結果が減ることもあります。
例えば今回のデータで、ヒステリシスの大きさを 1 クラスの幅そのものとした例が下図です。

[Classes]	Cls3	Cls3H
0	1.0	1.0
1	5.0	2.0
2	13.0	0.0
3	6.0	0.0
4	6.0	2.0
5	1.0	1.0

待機

実際の計測データではカウント対象となる極値自体が非常に多く、上記のような極値の見逃しが、解析上大きく影響してくることは基本的に無いものと考えられます。

ヒステリシスの大きさは特別に理由が無ければ 1 クラスの幅そのものが推奨されます。
_Hysteresis = (_MaxValue - _MinValue) / NumberOfBins のように計算することで、1 クラスの幅とすることができます。

2.6.演習：実用例-多チャンネルのクラスカウント

多チャンネルの頻度処理を実行する場合、グループ変数を使用すると簡単に実行できます。

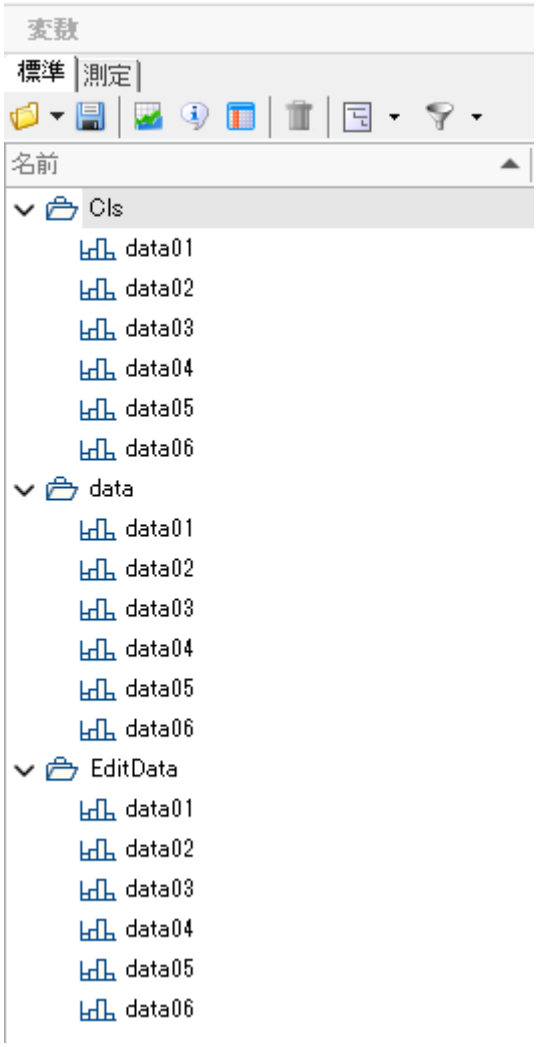
例として、セミナー資料に含まれる”Sample.dat”を読み込み、すべてのチャンネルをグループ変数 data としてまとめてみましょう。

```
--------------------------------------------------------------------------
; 対象データはグループ変数 data にまとまっているものとする

; パラメータ
_MaxValue     = 3
_MinValue     = -3
_NumberOfBins = 24
_Reference    = 0
_Hysteresis   = 0
_Options      = 0

; 0 が中心となるように編集する
EditData = data - Mean(data)

; ClsPeak 関数でカウント
Cls   =   ClsPeak1(EditData,   _MaxValue,   _MinValue,   _NumberOfBins,   _Reference,
_Hysteresis, _Options)

Del _*
--------------------------------------------------------------------------
```

上記のシーケンスはあくまで一例です。

0 が中心となるように編集する、の処理は実際の解析で不要であるなら飛ばして構いません。

実行後は下図のように、対象データ data と同じチャンネル名称で、Cls というグループ変数にカウント結果が収納されます。

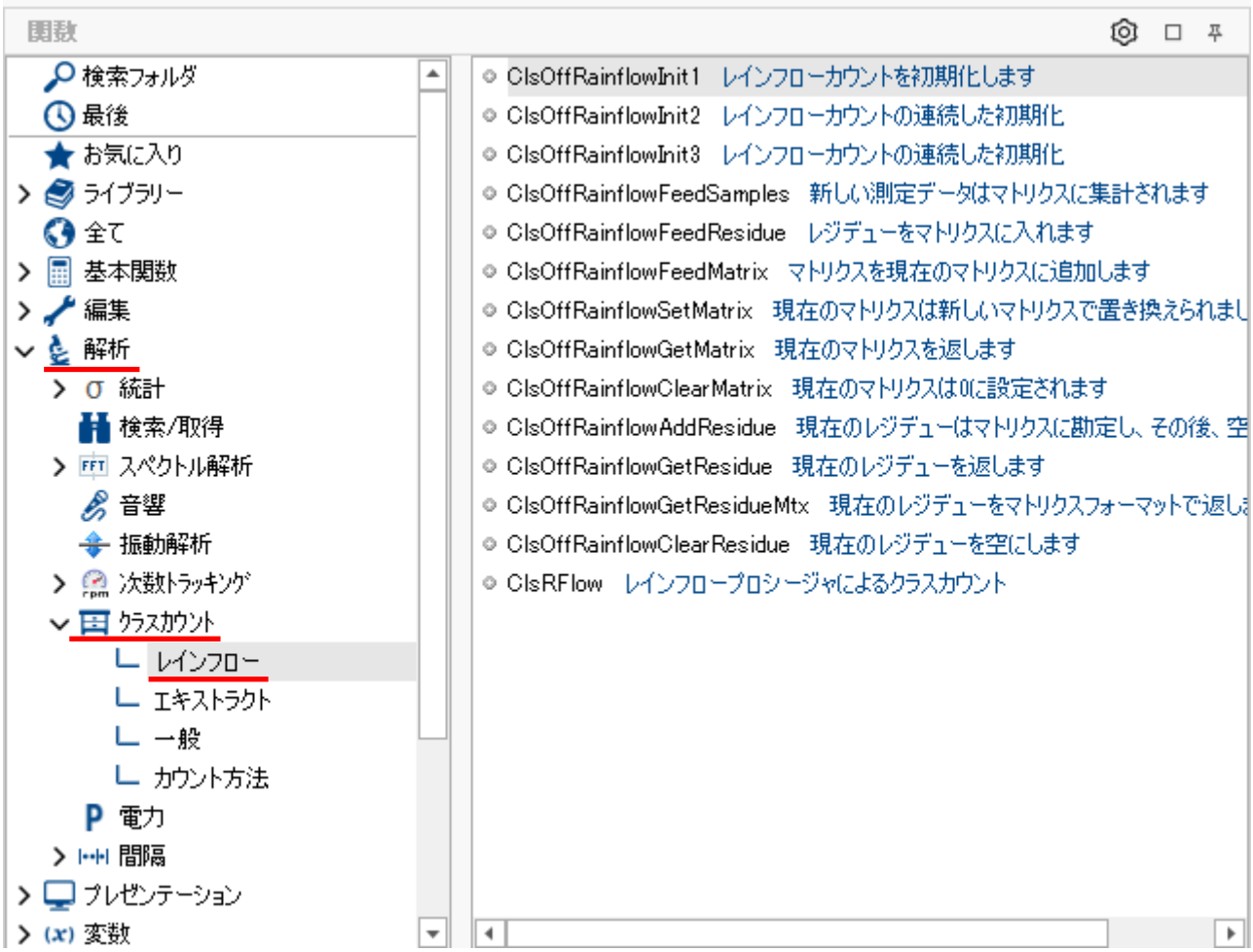


このシーケンスをさらに発展させて、ファイルから直接データを読み込みたい、あるフォルダに含まれるファイルを一括で解析したい、といった方法は、imc FAMOS 中級セミナーの章「データの読み込み」をご参照ください。

3.レインフロー法

レインフロー法は、**振幅の大きさ**を基準にクラスをカウントしていく手法です。レインフロー法の解析は、クラスカウント法とは異なり、あるデータを解析しようとしたときに常に複数の関数を使用することになります。

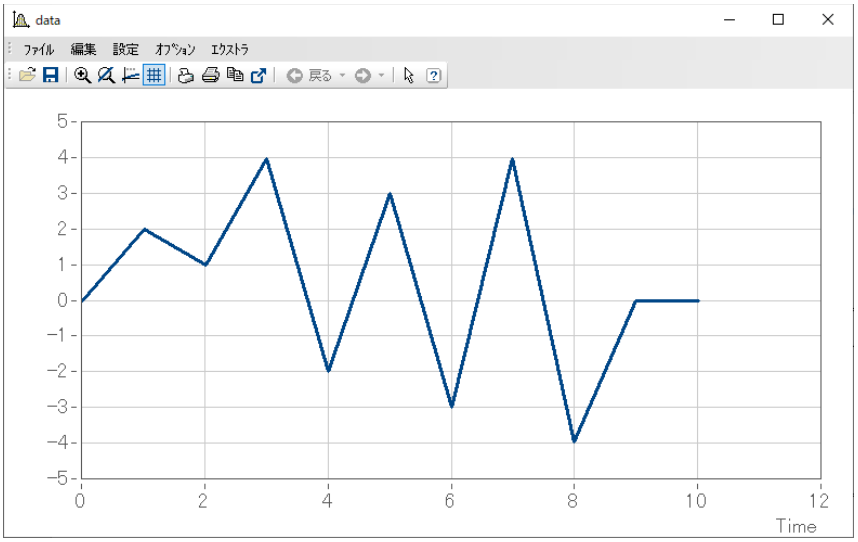
レインフロー法の関数は、imc FAMOS の関数ボックスの[解析 > クラスカウント > レインフロー]のカテゴリーにまとめられています。



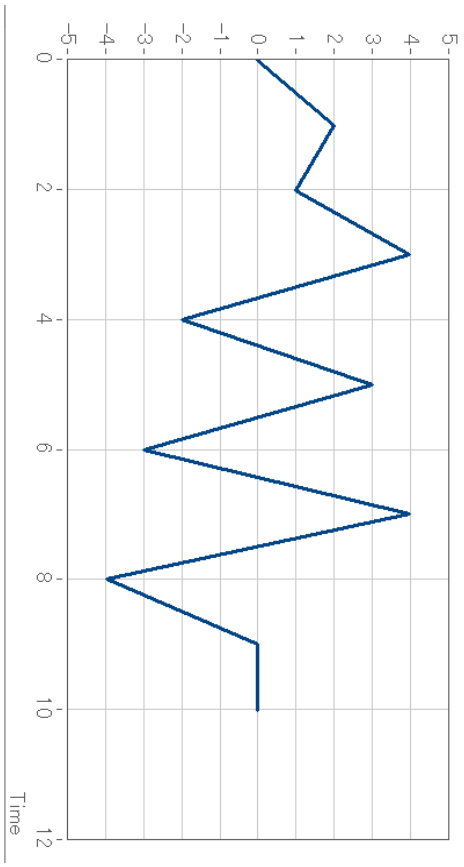
3.1.レインフロー法概要

レインフロー法は、振幅の大きさを基準にクラスをカウントしていく手法です。

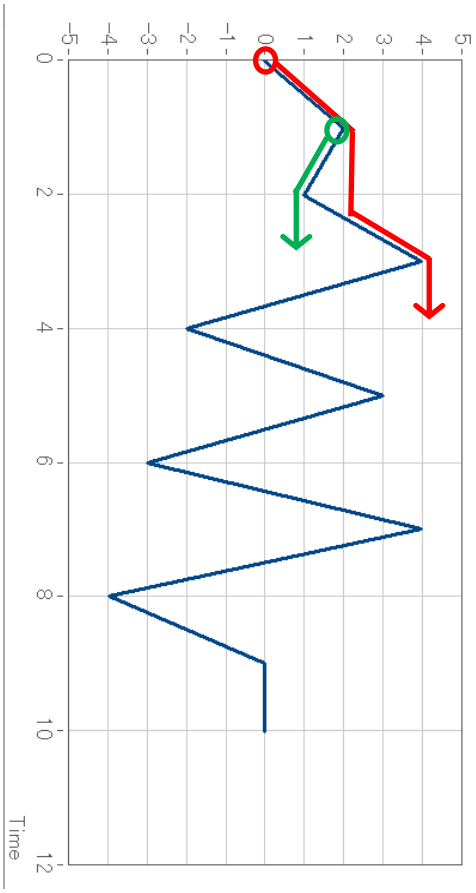
この振幅の大きさを考える手法は、屋根を流れる雨水の経路をイメージするとわかりやすいです。例として、下図のようなデータがあるものとします。



そして、時間軸を下方向にとった下図のような形で考えます。



例えば、下図の赤〇、緑〇からのポイントから雨水が流れるとすると、図のような経路を取ることがイメージできます。雨水は、その他すべての極値(尖った点)からも同様に流れていきます。



レインフロー法で重要となるのは、一度極値から流れ始めた雨水が、どの点で停止するかです。**流れの始点**と**流れの終点**を使うことで、その経路での振幅を考えることができます。

以後、この流れの始点を**スタートクラス**、流れの終点を**ターゲットクラス**と表現します。これは imc FAMOS のヘルプ上で使用されている表現と同一です。

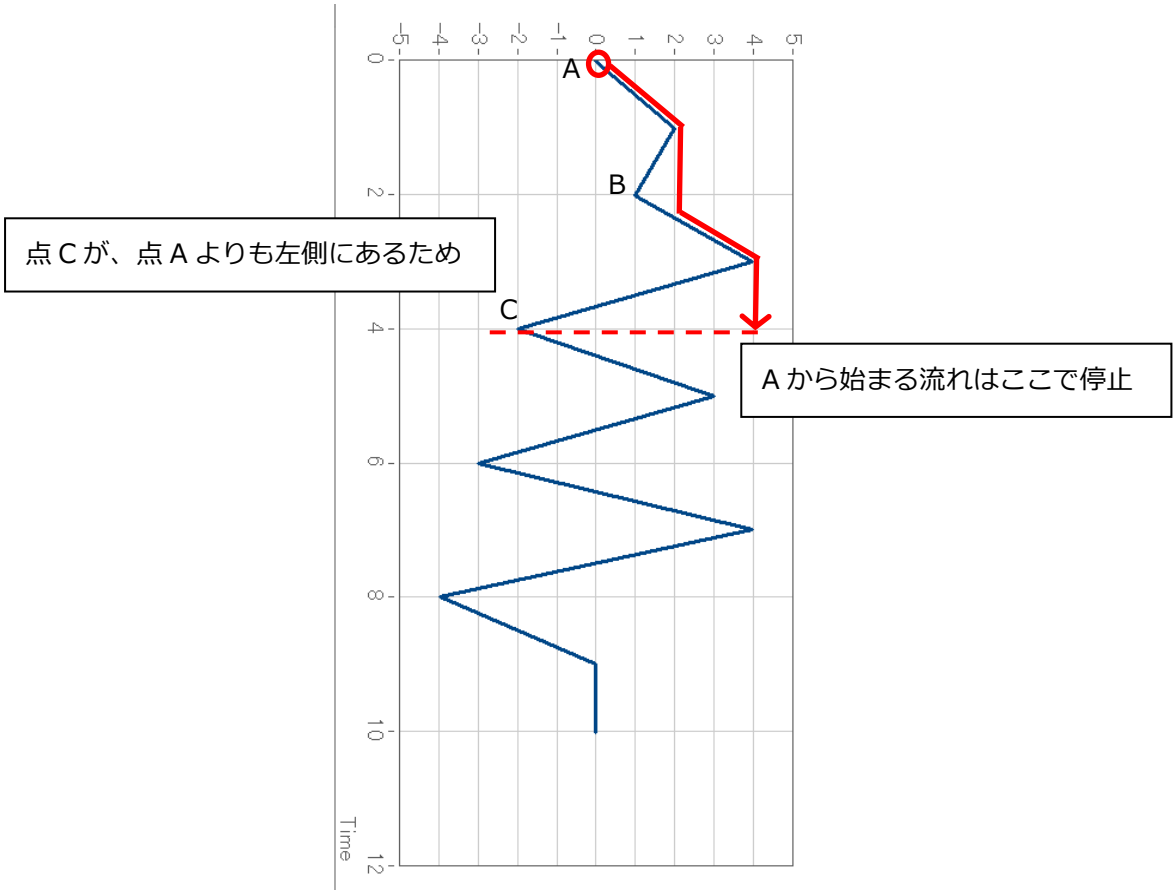
雨水は、以下の 2 条件で停止します。

1)流れが右向きである場合、その出発点よりも、もっと左側に次の右向きの出発点がある場合。
　(左向きの場合も同様、左右を読み替えて考える)

2)屋根に既に雨水が流れていた場合。

1)流れが右向きである場合、その出発点よりも、もっと左側に次の右向きの出発点がある場合。
の具体例を考えてみましょう。

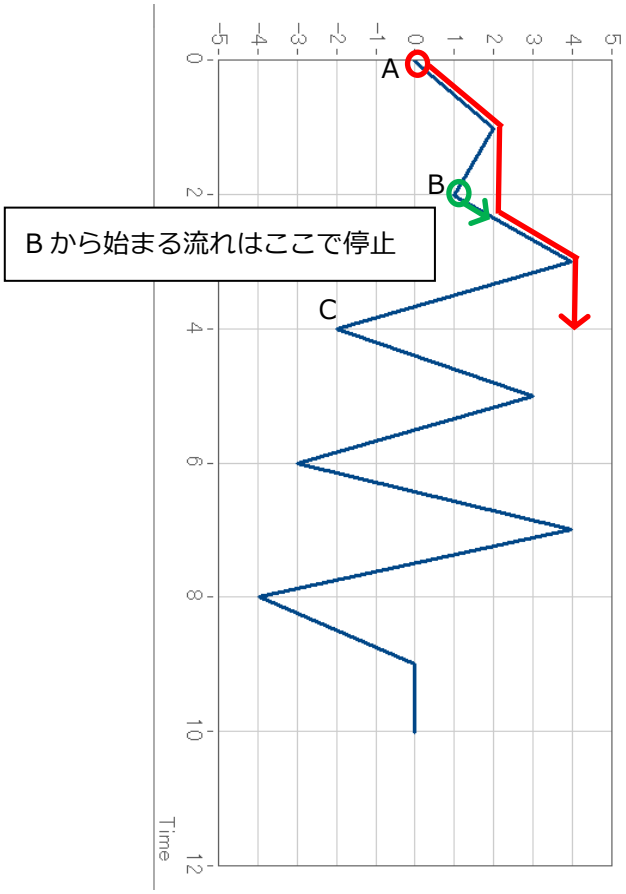
下図の点 A から流れた雨水は、このような経路を通ったうえで t=4 である点で停止します。
これは、点 C が、点 A よりも左側にある、同じく右向きの出発点であるためです。
点 B も右向きの出発点ではありますが、「出発点よりも左側にある」という条件を満たさないため、ここでは停止しません。



この流れの振幅は、スタートクラスが y=0、ターゲットクラスが y=4、という情報が使われます。
(実際の解析においてはこれらの値から振幅や平均値を求めます)

2)屋根に既に雨水が流れていた場合。
の具体例を考えてみましょう。

先ほどと同じく、下図の点 A から流れた雨水は、このような経路を通ったうえで t=4 である点で停止します。
次に点 B から右向きに流れる雨水を考えると、これはすぐに点 A から流れた雨水とぶつかることになり、
「屋根に既に雨水が流れていた場合」の条件を満たして停止します。



この流れの振幅は、スタートクラスが y=1、ターゲットクラスが y=2、という情報が使われます。

レインフロー法では、これらの出発点、停止点から「**サイクル**」という考え方を使います。ある振動が例えば y=0 から y=4 まで変化したとして、逆向きの y=4 から y=0 の変化があればそのセットが「1 サイクル」です。片方のみの変化があればそれは「ハーフ(1/2)サイクル」として考えられます。
そのため、最終的な結果ではカウントとして 0.5 という数値が現れることがあります。これは、該当するクラスにおいてハーフサイクルがカウントされたことを意味します。

3.2.レインフロー法の関数記述例

レインフロー法は関係する関数が多く、結果の出力方法も何通りか存在します。まずは関数単体ではなく、全体を通した基本的な関数の記述例を見てみましょう。

パラメータについては実際の値ではなく_(アンダーバー)付きの名前としています。

```
-----------------------------------------------------------------------------------------------------
; レインフロー法の解析用パラメータの設定
ClsHandle = ClsOffRainflowInit1(_Classes, _UnitUse, _UnitColumn, _UnitRow,
                                _UnitCounter, _Unit_Y_Residue, _SV_0)
ClsOffRainflowInit2(ClsHandle, _Min, _Max, _Hysteresis, _Axes, _Type, _OuterBins,
                    _Calculation)
ClsOffRainflowInit3(ClsHandle, _IgnoreSmallSpans, _Precise, _CountStartEnd,
                    _SV_Null1, _SV_Null2, _SV_Null3)

; 実際にカウント対象となるデータを与える
ClsOffRainflowFeedSamples(ClsHandle, Samples)

; レジデューを結果に加える
ClsOffRainflowAddResidue(ClsHandle, _Weight)

; 結果を返す
RainflowMatrix = ClsOffRainflowGetMatrix(ClsHandle)

; 結果を見やすい形に変換する
ResultRainflow = MatrixSumLines(RainflowMatrix, 0)
-----------------------------------------------------------------------------------------------------
```

関数やパラメータ、変数が多いですが、要点である入力データと解析結果の変数はそれぞれ以下の通りです。

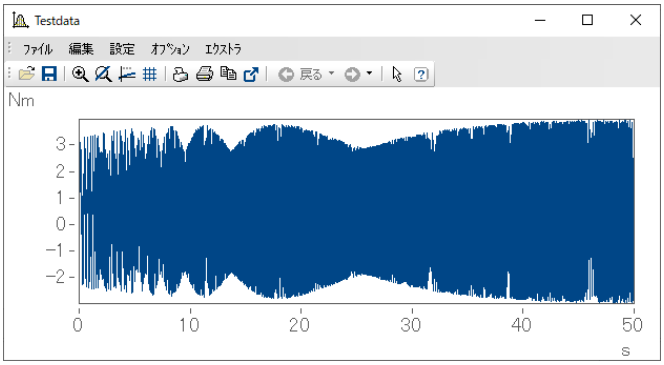
Samples	カウント対象となる入力データ
RainflowMatrix	カウント結果、3 次元の配列
ResultRainflow	カウント結果を 2 次元配列に変換したもの

次ページではレインフロー法のカウント結果について触れます。

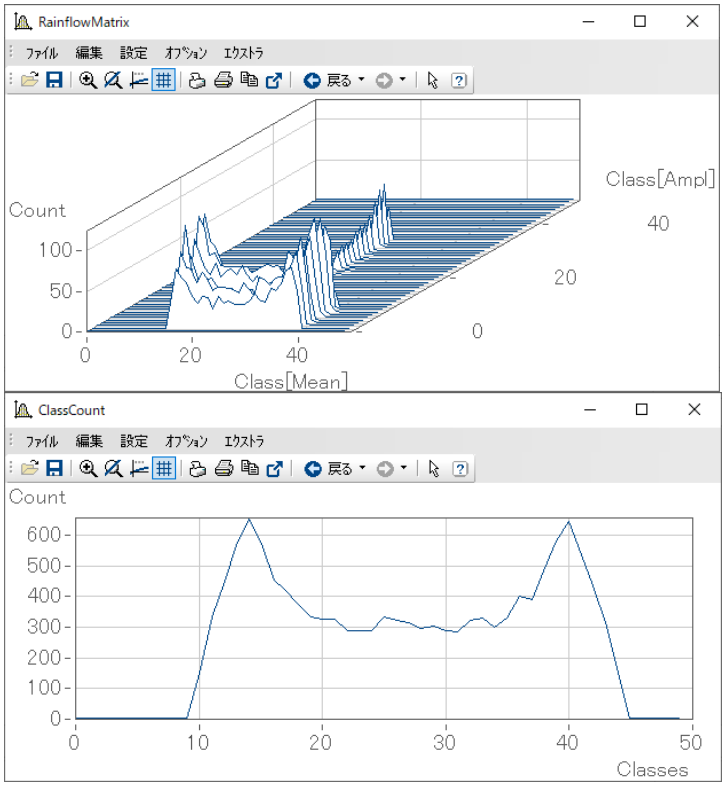
次節からは具体的な関数のパラメータを説明します。

カウントの方式それ自体も異なりますが、2 章のクラスカウント法と比較すると、関数の結果の形も大きく異なります。関数記述例で「結果を返す」「結果を見やすい形に変換する」という箇所が存在する理由を説明します。

例として、下図のようなデータをレインフロー法とクラスカウント法(ClsPeak2 関数)で解析した結果を並べてみます。クラスの分類等、パラメータはなるべく同じになるように設定しています。

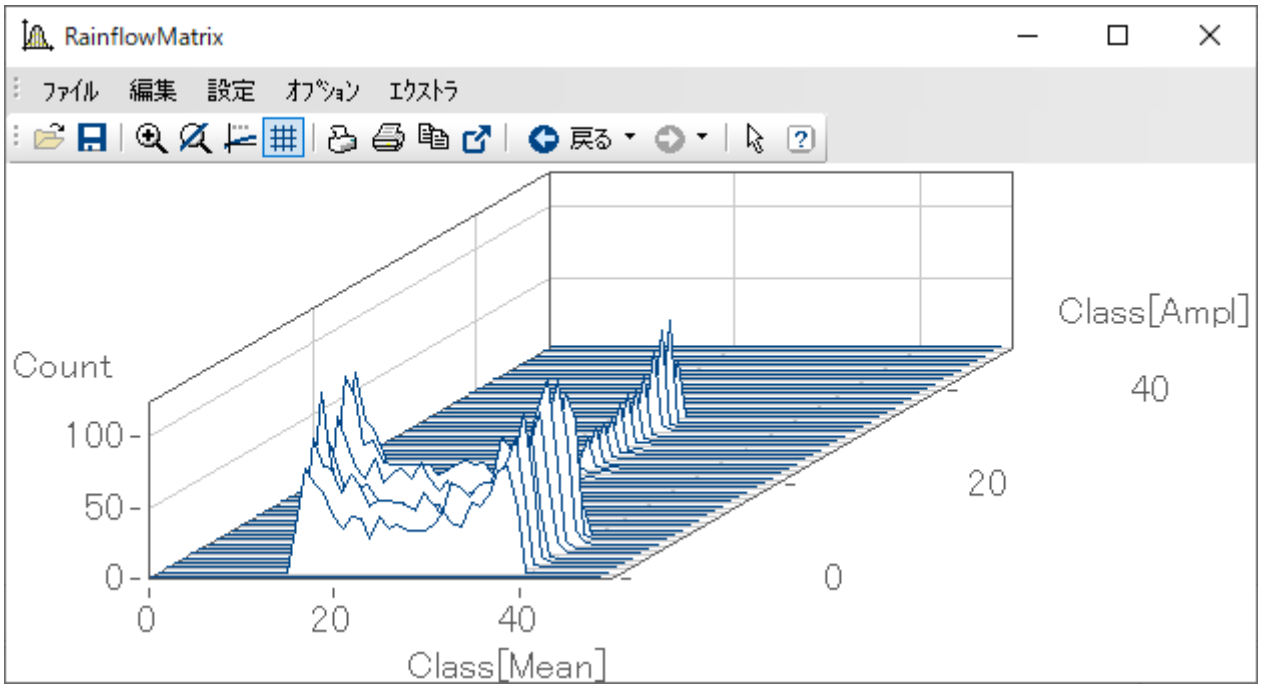


上がレインフロー法の結果(3.2 節の関数記述例では RainflowMatrix に当たるもの)、下がクラスカウント法の結果です。



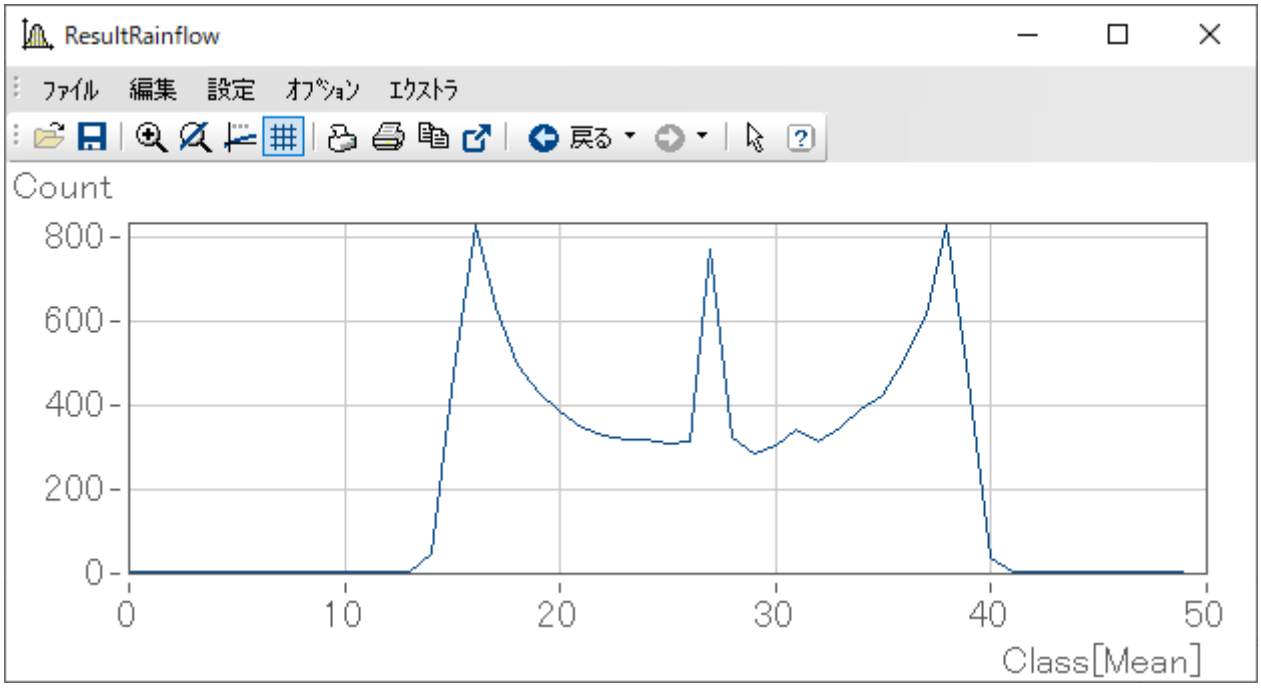
このように、クラスカウント法ではクラスとカウント結果の関係が 2 次元になるのに対して、レインフロー法では奥行き方向の Z 軸も増えて 3 次元となります。通常、カウントされた後の結果としては 2 次元である方が出力時などに扱いやすいため、関数記述例ではこの 3 次元から 2 次元への変換を「結果を見やすい形に変換する」としています。

結果が 3 次元になる理由としては、レインフロー法では**スタートクラス・ターゲットクラス**の 2 つを情報として扱うためです。例えば下図の結果では、X 軸(横方向)は平均値、Z 方向(奥行き方向)は振幅という分類を行っています。



平均値は、(スタートクラス + ターゲットクラス) / 2、
振幅は、(スタートクラス – ターゲットクラス) / 2 の絶対値、で計算されます。

例えばこの結果から振幅の情報を無くして平均値の情報のみにまとめると、下図のような結果となります。



3.3.レインフロー法の関数詳細

3.3.1.ClsOffRainflowInit1 関数

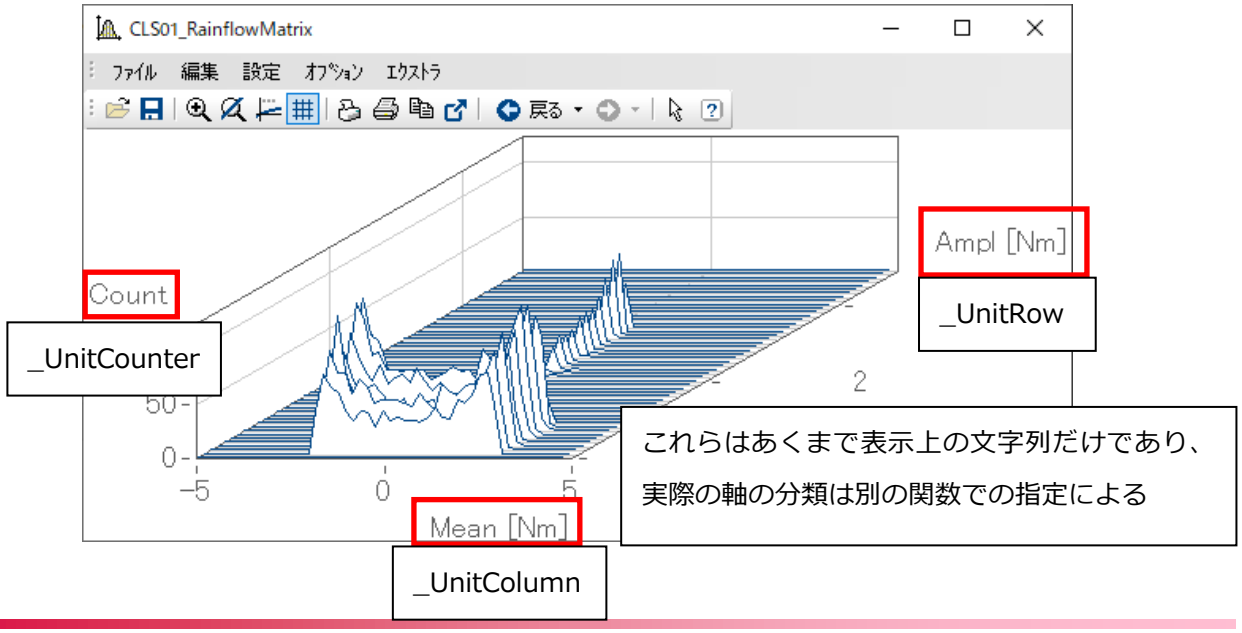
ClsOffRainflowInit1 関数は、レインフロー法の解析用パラメータを設定するための関数の 1 つです。パラメータは以下のようになります。

```
ClsHandle = ClsOffRainflowInit1(_Classes, _UnitUse, _UnitColumn, _UnitRow, _UnitCounter,
                                _Unit_Y_Residue, _SV_0)
```

ClsHandle	以後の関数群が参照するための変数 (名称は任意)
_Classes	クラス数をいくつにするかの設定、4 以上 1000 以下
_UnitUse	クラス幅をどのようにスケーリングするかの設定 0：クラス番号でスケーリング(0, 1, 2...) 1：入力データ自体の物理単位でスケーリング
_UnitColumn	結果の X 軸の単位 (X 軸の意味は後述)
_UnitRow	結果の Z 軸の単位 (Z 軸の意味は後述)
_UnitCounter	結果の Y 軸の単位 (Y 軸はカウント回数)
_Unit_Y_Residue	レジデューの Y 軸の単位 (基本的にカウント回数)
_SV_0	常に 0 に設定しておく現在は使われていないパラメータ

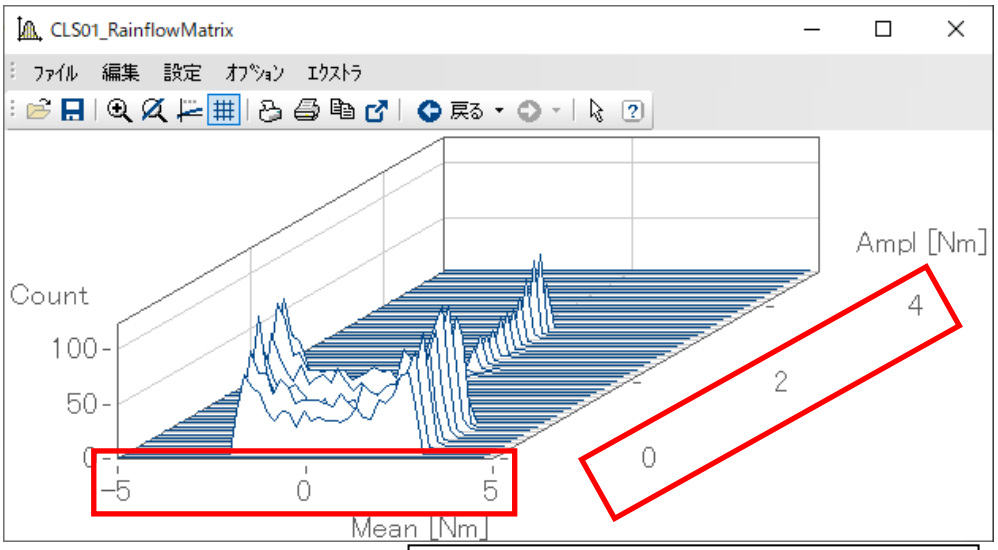
〇軸の単位、というパラメータは表記上の文字列の設定です。

例えば下図の結果の場合、X 軸の単位は Mean [Nm]、Z 軸の単位は Ampl [Nm]、Y 軸の単位は Count という文字列がそれぞれ設定されています。ここでの X 軸、Z 軸の設定はあくまで表記上の文字列のみであり、実際の軸の意味自体は ClsOffRainflowInit2 関数で設定します。



クラス軸のスケーリング、_UnitUse は下図の例では 1、つまり入力データ自体の物理単位によるスケーリングが使用されています。

_UnitUse を 0 とした場合は、両方の軸とも 0, 1, 2…という何番目のクラスに属するか、という番号表示の軸となります。



_UnitUse = 1、物理単位スケーリングの例

これ単体では解析用パラメータの設定は完結せず、ClsOffRainflowInit2、ClsOffRainflowInit3 関数へ続きます。続く関数では、この関数で作成される変数である ClsHandle(変数名自体は任意)を参照します。

3.3.2.ClsOffRainflowInit2 関数

ClsOffRainflowInit2 関数も ClsOffRainflowInit1 関数と同様、レインフロー法の解析用パラメータを設定します。パラメータは以下のようになります。

ClsOffRainflowInit2(ClsHandle, _Min, _Max, _Hysteresis, _Axes, _Type, _OuterBins, _Calculation)

ClsHandle	ClsOffRainflowInit1 関数で作成された変数
_Min	クラスの下限
_Max	クラスの上限
_Hysteresis	小さな振動を無視するためのヒステリシスの大きさ、1 つのクラス幅が推奨値 ヒステリシスの考え方については 2.1.2 節を参照
_Axes	X 軸にどのような意味を割り当てるかの設定、詳細後述
_Type	X 軸、Z 軸にどのような意味を割り当てるかの設定、詳細後述
_OuterBins	クラスの上限/下限の外にある値をどのように扱うかの設定、詳細後述
_Calculation	解析用のアルゴリズムの設定、詳細後述

まずは軸の意味を設定するパラメータ、_Axes、_Type を説明します。まずはパラメータの設定可能な値と内容だけ並べてみます。

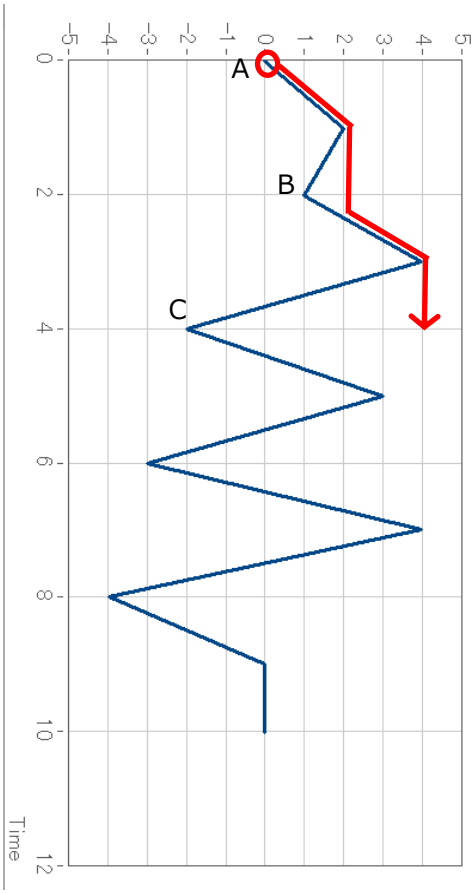
[_Axes の設定値]　X 軸の意味
　0：ターゲットクラスまたは振幅
　1：平均またはスタートクラス
[_Type の設定値]　X 軸と Z 軸の意味
　0：スタートクラスとターゲットクラス
　1：振幅と平均
　2：平均とスパン

これらのパラメータは組み合わせで内容が決定されます。
例えば_Type の値を 1 とした場合、これは「X 軸と Z 軸はどちらかが振幅、どちらかが平均を意味する値となる」という設定です。
次に_Axes を考えましょう。ここで_Axes を 1 とすると、「X 軸の意味は振幅/平均のうちの平均である」と決まります。自動的に Z 軸が振幅となります。

このように_Axes、_Type は組み合わせることで X 軸は何を意味する軸か、Z 軸は何を意味する軸か、がそれぞれ決定されます。

次はここで出てくる単語、スタートクラス、ターゲットクラス、振幅、平均、スパン、の意味をそれぞれ見ていきましょう。

3.1 節の繰り返しになりますが、レインフロー法には「流れの始点：**スタートクラス**」「流れの終点：**ターゲットクラス**」の概念が存在します。下図の例では点 A から始まる流れは y=0 が始点、y=4 が終点です。これらの y=〇ポイントは、それぞれいずれかのクラスに分類されているはずです。



軸の意味で出てくる単語は、それぞれ下記を意味します。

スタートクラス	流れの始点が分類されるクラス
ターゲットクラス	流れの終点が分類されるクラス
振幅	(スタートクラス – ターゲットクラス) / 2 の絶対値
平均	(スタートクラス + ターゲットクラス) / 2
スパン	(スタートクラス – ターゲットクラス) の絶対値

物理的な意味で考えるなら、以下のようになります。

振幅	ある波形のピーク to ピークの半分(片振幅)
平均	ある波形の中間点
スパン	ある波形のピーク to ピーク(両振幅)

残る_OuterBins と_Calculation の設定値を見ていきましょう。

_OuterBins はクラスの上限/下限の外にある値の扱いに関する設定です。

0：クラスの上限/下限の外にある極値は無視されます。

1：クラスの上限/下限の外にある極値は、クラス内に割り当てられます。(こちらが推奨設定)

_Calculation は解析用のアルゴリズムの設定です。これらは規定のアルゴリズムであるためその中身は詳しく説明せず、アルゴリズムの名称等のみ述べます。

0：標準的な 4 点アルゴリズム (standard-algorithm 4 points)

1：コールマンシーガ補正 (Chlormann Seeger correction)

2：1990 年に再承認された ASTM E1049 に基づいた計算

3：RAINFLOW-HCM メソッド (U.H. Clormann, T. Seeger; Stahlbau 3/1986)

_Calculation として 2 を選択した場合、後述する ClsOffRainflowAddResidue 関数を呼び出す必要があります。また、ClsOffRainflowInit3 関数における_Precise のパラメータは無視されます。
_Calculation として 2 または 3 を選択した場合、ClsOffRainflowInit3 関数における_Precise のパラメータは 5 以上とする必要があります。

3.3.3.ClsOffRainflowInit3 関数

ClsOffRainflowInit3 関数も ClsOffRainflowInit1 関数と同様、レインフロー法の解析用パラメータを設定します。パラメータは以下のようになります。

ClsOffRainflowInit3(ClsHandle, _IgnoreSmallspans, _Precise, _CountStartEnd,
_SV_Null1, _SV_Null2, _SV_Null3)

ClsHandle	ClsOffRainflowInit1 関数で作成された変数
_IgnoreSmallSpans	1 つのクラスに収まる小さなスパンを無視するかどうか
_Precise	スパンを計算する精度の設定、より高い値を推奨
_CountStartEnd	境界にある値をカウントするかどうか、カウントする設定を推奨
_SV_Null1	常に 0 に設定しておく現在は使われていないパラメータ
_SV_Null2	常に 0 に設定しておく現在は使われていないパラメータ
_SV_Null3	常に 0 に設定しておく現在は使われていないパラメータ

_IgnoreSmallSpans、_Precise、_CountStartEnd は特定の値を選択するパラメータです。

_IgnoreSmallSpans は、1 つのクラス内に収まってしまう小さなスパンをどのように扱うかの設定です。

0：小さなスパンもカウントします。

1：小さなスパンを無視します。

_Precise は、スパンをどれほど精度良く計算するかの手法の設定です。

0：極値は最初にクラスに割り当てられ、次に振幅/平均の計算において振幅/スパンは引き算により求められますが、これはあまり正確ではありません。

1：スパンと平均値を正確に計算し、それに基づいてクラスに関連させます。

2：1 と同様の処理に加えて、より正確にレジデューから削除を行います。

3：2 と同様の処理に加えて、レジデューが消える場合にも正確に削除を行います。

4：3 の処理に近いですが、ClsOffRainflowFeedSamples 関数が複数回呼ばれた場合であっても、丸めずに正確なレジデューを追加します。

5：4 の処理に近いですが、より正確です。

6：5 の処理に近いですが、クラスの境界にある値をより正確に計算します。

特別な理由が無い限りは_Precise の設定は最も高い値を推奨します。imc FAMOS のバージョンによっては、ここで説明しているパラメータが実装されていないこともあります。
ClsOffRainflowInit3 関数の_Calculation において 2 または 3(ASTM E1049、HCM)を選択している場合はこの_Precise のパラメータを 5 以上に設定しなければなりません。

_CountStartEnd は、境界にある値をカウントするかどうかの設定です。

0：境界にある値はカウントしません。

1：境界にある値もカウントします、こちらの設定を推奨します。

3.3.4.ClsOffRainflowFeedSamples 関数

ClsOffRainflowFeedSamples 関数は、実際にレインフロー法でカウントを行う対象データを与えるための関数です。パラメータは以下のようになります。

ClsOffRainflowFeedSamples(ClsHandle, Samples)

ClsHandle	ClsOffRainflowInit1 関数で作成された変数
Samples	解析対象とするデータの変数

この関数は複数回実行することができます。例えば試験を複数回行っていてそれらすべてのデータをまとめて 1 つのカウント結果としたい場合、この関数を繰り返してそれぞれのデータを与えることで実行できます。

3.3.5.ClsOffRainflowAddResidue 関数

レインフロー法におけるカウントでは、1 回分のカウントに満たない成分がレジデュー(Residue)として扱われます。この関数ではこのレジデューを最終的なカウント結果にどのように反映させるかの設定を行います。パラメータは以下のようになります。

ClsOffRainflowAddResidue(ClsHandle, _Weight)

ClsHandle	ClsOffRainflowInit1 関数で作成された変数
_Weight	カウント結果に反映させるための重み

_Weight としては 0～1 の範囲の値が設定可能ですが、実際に入力するべき値としては 3 通りです。

0：レジデューの値はカウント結果には反映されません。

0.5：レジデューの値はハーフサイクルと同じものとしてカウント結果に反映されます。

1：レジデューの値は 1 サイクルと同じものとしてカウント結果に反映されます。

つまり、_Weight が 1 に近いほどカウントの結果が大きくなります。

なお、ClsOffRainflowInit3 関数の_Calculation において 2 (ASTM E1049)を選択している場合は、このアルゴリズム自体でレジデューの扱いが定義されているため、この_Weight のパラメータ設定は無視されます。

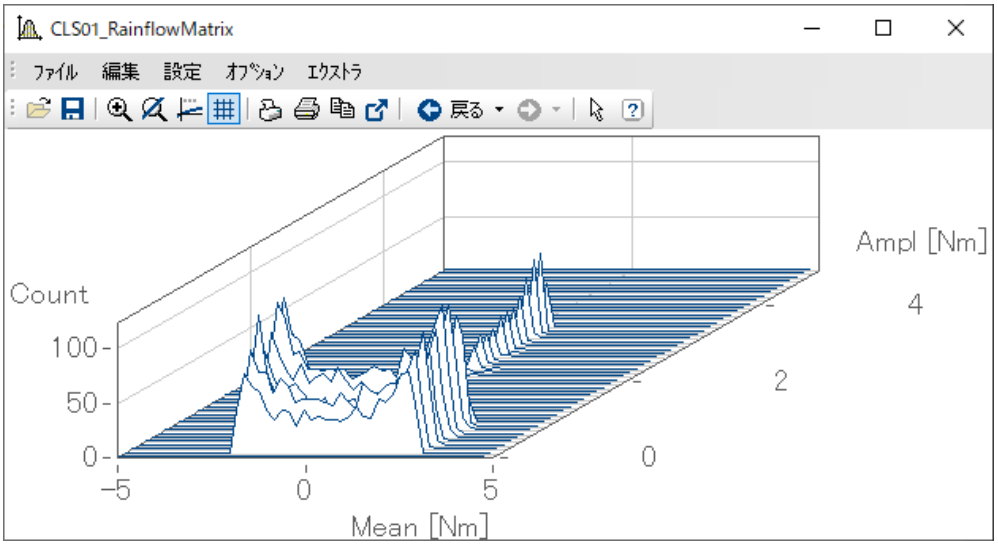
3.3.6.ClsOffRainflowGetMatrix 関数

ClsOffRainflowGetMatrix 関数は、ここまでの関数により設定された解析方法、与えられたデータによるカウント結果を返すための関数です。記述は下記のようになります。

```
RainflowMatrix = ClsOffRainflowGetMatrix(ClsHandle)
```

ここで、式の左辺は結果の変数の名前、ClsHandle は ClsOffRainflowInit1 関数で作成された変数です。

この関数で得られる結果は、下図のような奥行き方向の存在する 3 次元データです。
X 軸、Z 軸にどのような意味が割り振られるかは、ここまでに説明した関数で決定されます。



3.3.7.MatrixSumLines 関数

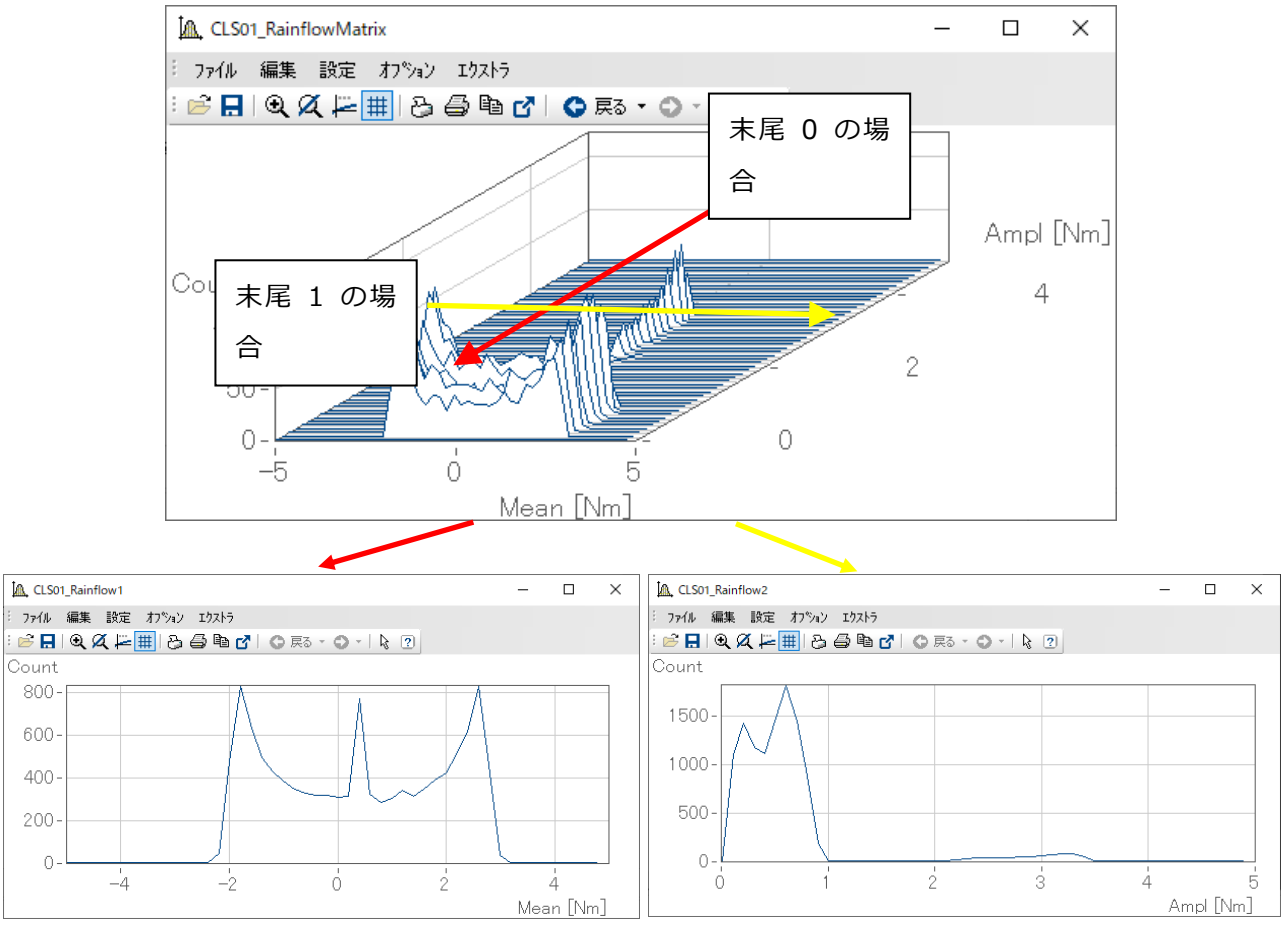
MatrixSumLines 関数はレインフロー法とは直接関係の無いもので、3 次元データの編集を行うための関数です。この関数を利用することで、本来 3 次元であるレインフロー法の解析結果を、クラスカウント法のような 2 次元の結果に変換することができます。なお、疲労寿命推定は 3 次元データのままで計算可能です。結果の表示や出力の際に見やすい形とするために利用できます。

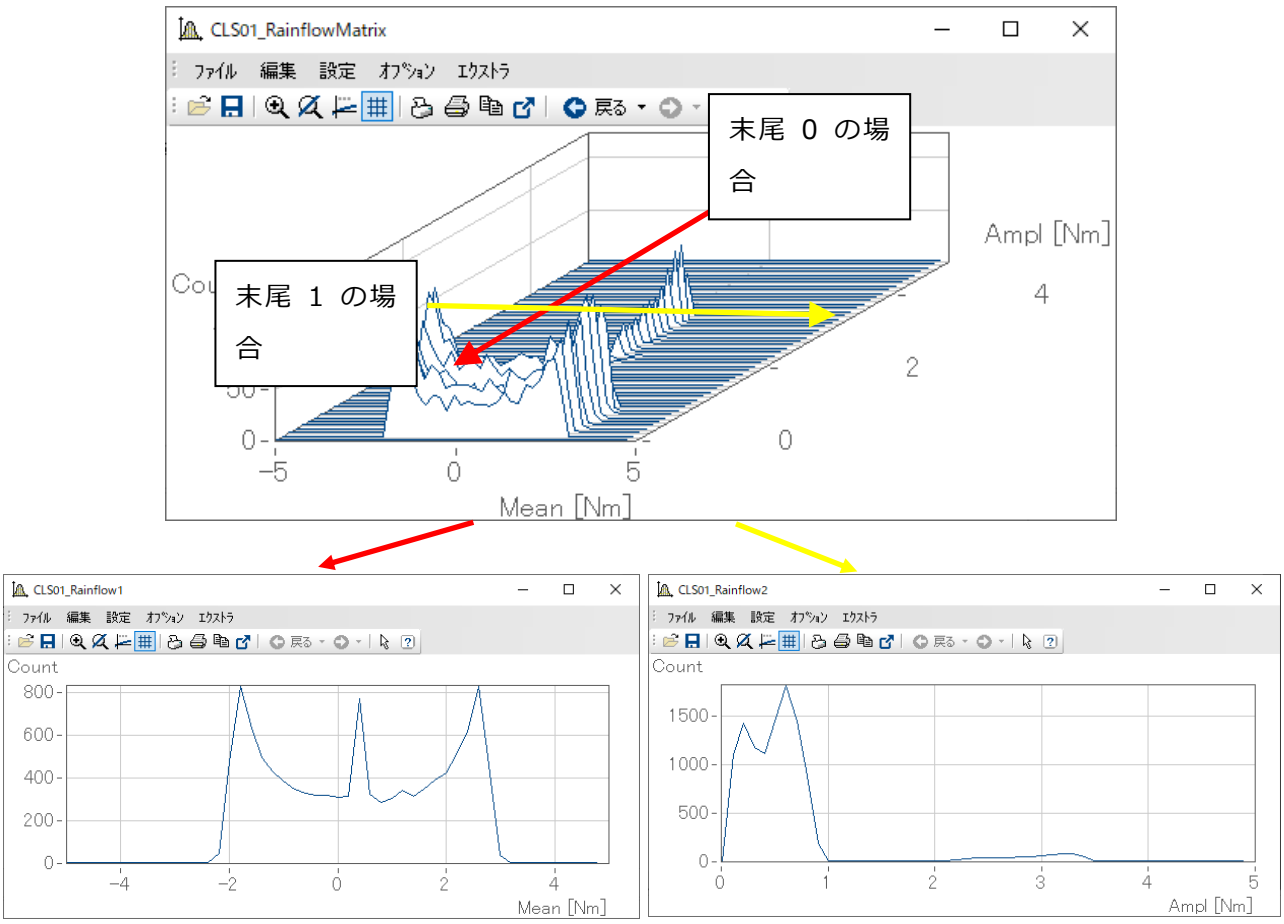
記述は下記のようになります。

```
ResultRainflow = MatrixSumLines(RainflowMatrix, 0)
```

ここで、式の左辺は結果の変数の名前、RainflowMatrix は ClsOffRainflowGetMatrix 関数で作成された 3 次元のカウント結果の変数です。末尾の数値は 0 または 1 が設定できます。

末尾の 0 または 1 に応じて、3 次元の結果を X 軸、または Z 軸方向にまとめて 2 次元の結果に変換します。例として下図のような Mean, Ampl を X 軸、Z 軸として持つ 3 次元データにこの関数を適用してみます。





末尾が 0 の場合、X 軸方向にすべての結果が合算され、左側のように Mean-Count の 2 次元のカウント結果となります。同様に末尾が 1 の場合、Z 軸方向にすべての結果が合算され、右側のように Ampl-Count の 2 次元のカウント結果となります。ここまでの関数を利用することで、X 軸、Z 軸をどのように設定するか、最終的な結果をどちらにまとめるか、を任意に設定可能です。

3.3.8.ClsOffRainflowClearMatrix、ClsOffRainflowClearResidue 関数

3.2 節の関数記述例では使用していませんが、追加で 2 つの関数を説明します。これらの関数の記述は以下のようになります。

ClsOffRainflowClearMatrix(ClsHandle)
ClsOffRainflowClearResidue(ClsHandle)

ここで、ClsHandle は ClsOffRainflowInit1 関数で作成された変数です。

これらの関数は、既に存在する結果を 0 の状態にリセットする役割があります。Matrix がカウント結果それ自体、Residue がレジデューを対象としています。

別々のデータを何度もレインフロー法で解析する場合に使用する関数です。処理の流れとしては以下のようになります。

ClsOffRainflowInit1, 2, 3 関数でレインフロー法のパラメータ指定
↓
ClsOffRainflowFeedSamples 関数でデータを与えてカウント
↓
ClsOffRainflowGetMatrix 関数等でカウント結果を取得
↓
ClsOffRainflowClearMatrix、ClsOffRainflowClearResidue でリセット
↓
ClsOffRainflowFeedSamples 関数でまた別のデータを与えてカウント
↓
…

3.4.演習：レインフロー解析の実行例

では、具体的なパラメータとサンプルデータを使用してレインフロー解析を実行してみましょう。

サンプルデータ：C:¥Users¥Public¥Documents¥imc¥imc FAMOS¥_Demo Projects¥Order tracking¥otr01.dat

---

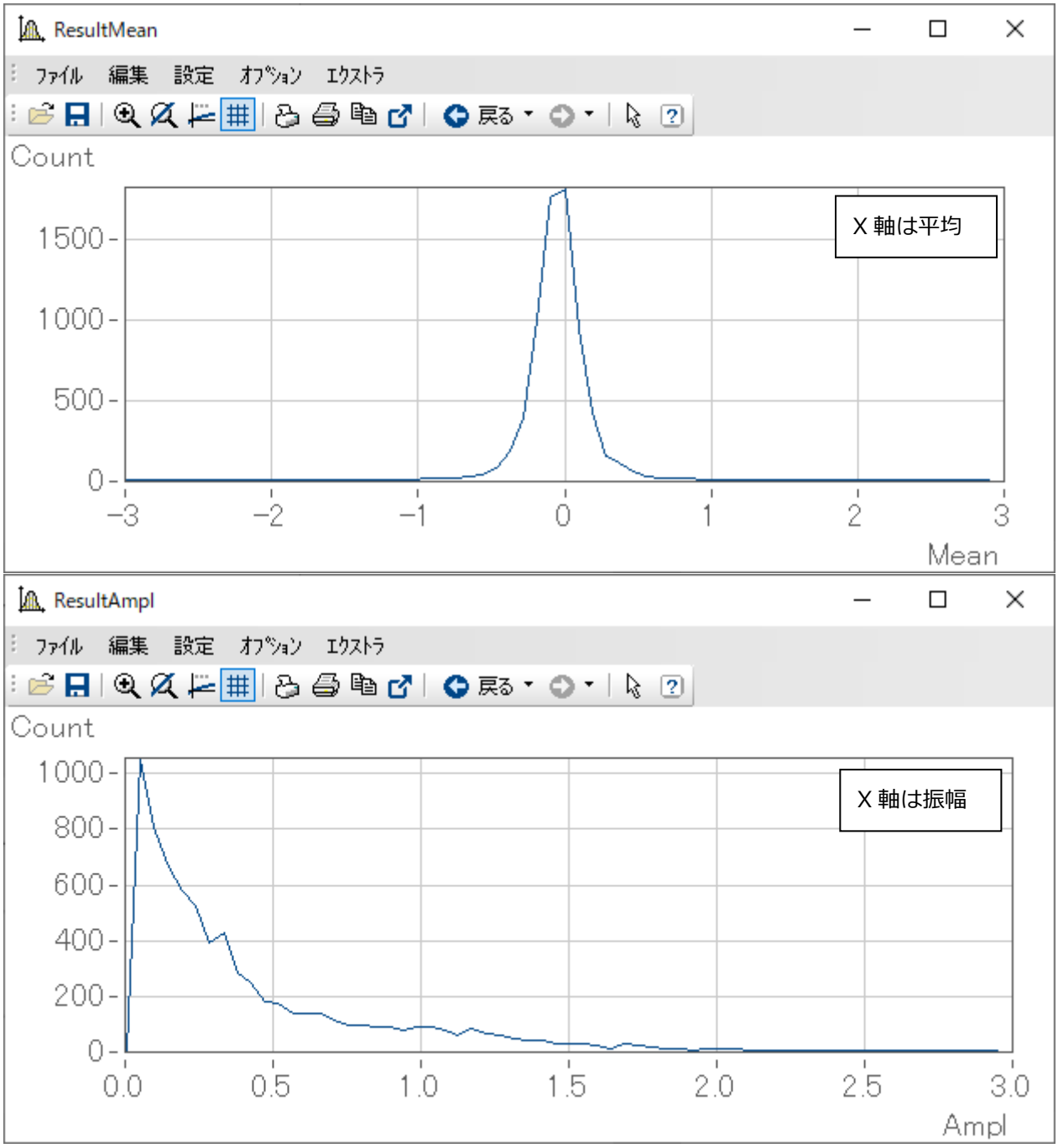
```
; レインフロー解析のパラメータ定義
_Classes              = 64    ; クラス数
_UnitUse              = 1     ; 1:物理単位でスケーリング
_UnitColumn           = "Mean"
_UnitRow              = "Ampl"
_UnitCounter          = "Count"
_Unit_Y_Residue       = "Count"
_Min                  = -3    ; クラス最小
_Max                  = 3     ; クラス最大
_Hysteresis           = (_Max - _Min) / _Classes  ; ヒステリシスは 1 クラスの幅
_Axes                 = 1     ; 1: X 軸は平均
_Type                 = 1     ; 1:X/Z 軸は振幅と平均
_OuterBins            = 1     ; 1:クラスの外側もカウント
_Calculation          = 0     ; 0:標準アルゴリズム
_IgnoreSmallSpans     = 0     ; 0:小さなスパンをカウントする
_Precise              = 6     ; 精度 6
_CountStartEnd        = 1     ; 1:境界をカウントする
_Weight               = 0.5   ; レジデューの重み

; レインフローカウントの初期化
ClsHandle = ClsOffRainflowInit1(_Classes, _UnitUse, _UnitColumn, _UnitRow,
                                _UnitCounter, _Unit_Y_Residue, 0)
ClsOffRainflowInit2(ClsHandle, _Min, _Max, _Hysteresis, _Axes, _Type, _OuterBins,
                    _Calculation)
ClsOffRainflowInit3(ClsHandle, _IgnoreSmallSpans, _Precise, _CountStartEnd,
                    0, 0, 0)
```

(次ページに続く)

```
; カウント対象となるデータを与える
ClsOffRainflowFeedSamples(ClsHandle, Vibration)

; レジデューを結果に加える
ClsOffRainflowAddResidue(ClsHandle, _Weight)

; 結果を返す
RainflowMatrix = ClsOffRainflowGetMatrix(ClsHandle)

; 平均/振幅のクラスでそれぞれまとめる
ResuleMean = MatrixSumLines(RainflowMatrix, 0)
ResultAmpl = MatrixSumLines(RainflowMatrix, 1)

Del _*
-----------------------------------------------------------------------------------------------
```

レインフロー解析ではパラメータの種類が多岐に渡り、それぞれのパラメータの指定も少々複雑です。
そのため実際にシーケンスを作って管理する際には、上記の実行例のように
・変更する可能性のあるパラメータはすべて 1 箇所にまとめておく
・それ単体では意味の分からないパラメータ(_UnitUse など)は適宜コメントを入れておく
としておくと、以後使う際の見直しや改造がやりやすくなります。

最終的に平均/振幅それぞれのクラスでまとめて 2 次元にした様子が下図です。



おそらく実用上は、下側の「振幅のクラスでまとめた結果」を使用するケースが多いかと思われます。

3.5.演習：実用例-多チャンネルのレインフロー解析

多チャンネルのレインフロー解析を実行する場合、2.6 節のクラスカウント法の場合とは異なり、解析にはループ処理が必要になります。いずれにしろグループ変数を使用するとシーケンス記述は簡易化できます。例として、セミナー資料に含まれる"Sample.dat"を読み込み、すべてのチャンネルをグループ変数 data としてまとめておきましょう。

```
------------------------------------------------------------------------
; 対象データはグループ変数 data にまとまっているものとする

; レインフロー解析のパラメータ定義
_Classes            = 64    ; クラス数
_UnitUse            = 1     ; 1:物理単位でスケーリング
_UnitColumn         = "Mean"
_UnitRow            = "Ampl"
_UnitCounter        = "Count"
_Unit_Y_Residue     = "Count"
_Min                = -3    ; クラス最小
_Max                = 3     ; クラス最大
_Hysteresis         = (_Max - _Min) / _Classes  ; ヒステリシスは 1 クラスの幅
_Axes               = 1     ; 1:X 軸は平均
_Type               = 1     ; 1:X/Z 軸は振幅と平均
_OuterBins          = 1     ; 1:クラスの外側もカウント
_Calculation        = 0     ; 0:標準アルゴリズム
_IgnoreSmallSpans   = 0     ; 0:小さなスパンをカウントする
_Precise            = 6     ; 精度 6
_CountStartEnd      = 1     ; 1:境界をカウントする
_Weight             = 0.5   ; レジデューの重み

; 0 が中心となるように編集する
EditData = data - Mean(data)
```

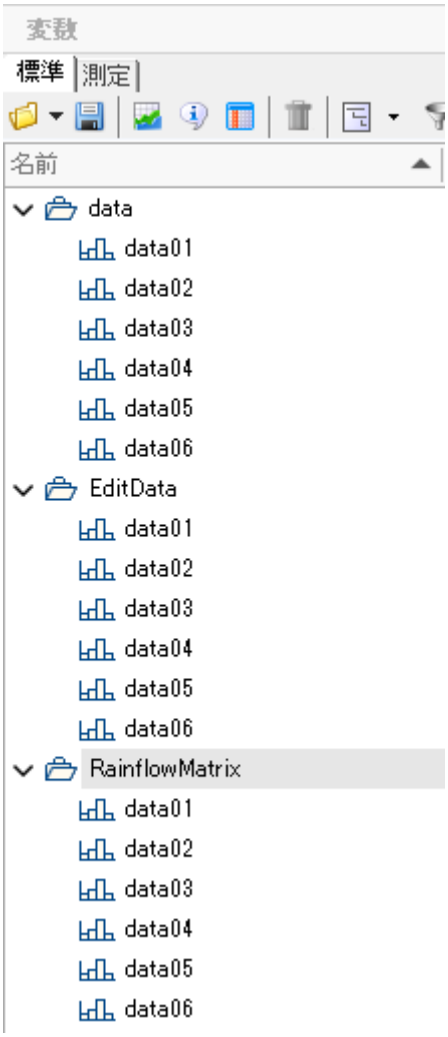
(次ページに続く)

```
; レインフローカウントの初期化
ClsHandle = ClsOffRainflowInit1(_Classes, _UnitUse, _UnitColumn, _UnitRow,
              _UnitCounter, _Unit_Y_Residue, 0)
ClsOffRainflowInit2(ClsHandle, _Min, _Max, _Hysteresis, _Axes, _Type, _OuterBins,
                    _Calculation)
ClsOffRainflowInit3(ClsHandle, _IgnoreSmallSpans, _Precise, _CountStartEnd,
                    0, 0, 0)

; 結果となるグループ変数を作成
RainflowMatrix = data * 0

; ループ処理で全チャンネルを解析
_i = 1
While _i <= GrChanNum?(EditData)

        ; カウント対象となるデータを与える
        ClsOffRainflowFeedSamples(ClsHandle, EditData:[_i])

        ; レジデューを結果に加える
        ClsOffRainflowAddResidue(ClsHandle, _Weight)

        ; 結果を返す
        RainflowMatrix:[_i] = ClsOffRainflowGetMatrix(ClsHandle)

        ; リセット
        ClsOffRainflowClearMatrix(ClsHandle)
        ClsOffRainflowClearResidue(ClsHandle)

        _i = _i + 1
End

Del _*
Del ClsHandle
------------------------------------------------------------------------
```

レインフロー法の場合、カウントのためのデータを与え ClsOffRainflowFeedSamples 関数が、リセットを行うまでカウント結果を合算し続ける挙動のため、上記のようなループを用いたシーケンスが必要となります。

こちらも 2.6 節のクラスカウント法の例と同様、0 が中心となるように編集する、の処理は実際の解析で不要であるなら飛ばして構いません。

結果は下図のように、対象データ data と同じチャンネル名称で、RainflowMatrix というグループ変数に収納されます。
平均/振幅それぞれのクラスにまとめる処理は省略しています。



3.6.他の解析ソフトウェアと imc FAMOS の結果合わせこみ

他の解析ソフトウェアでレインフロー解析を行った結果と、imc FAMOS の結果を合わせこみたい場合は以下のケースをご参照ください。

3.6.1.カウント結果が倍程度異なる

他の解析ソフトウェアの結果と imc FAMOS の結果を揃えたい場合において、「imc FAMOS 側のカウント結果が他のソフトウェアの半分程度しかない」というケースが存在します。

これは、imc FAMOS のカウント結果がレインフロー法の「**サイクル**」で計数されていることによります。
例えば、ある振動が y=0 から y=4 まで変化したとして、逆向きの y=4 から y=0 の変化があればそのセットが「1 サイクル」と考えられます。

他の解析ソフトウェアの場合、サイクルではなく一方向の変化(つまりは 1/2 サイクル)を 1 回とカウントするものがあります。これらのソフトウェアと結果を揃える場合、**imc FAMOS 側のカウント結果を 2 倍**してください。

3.6.2.カウント結果の一致率があまりよくない

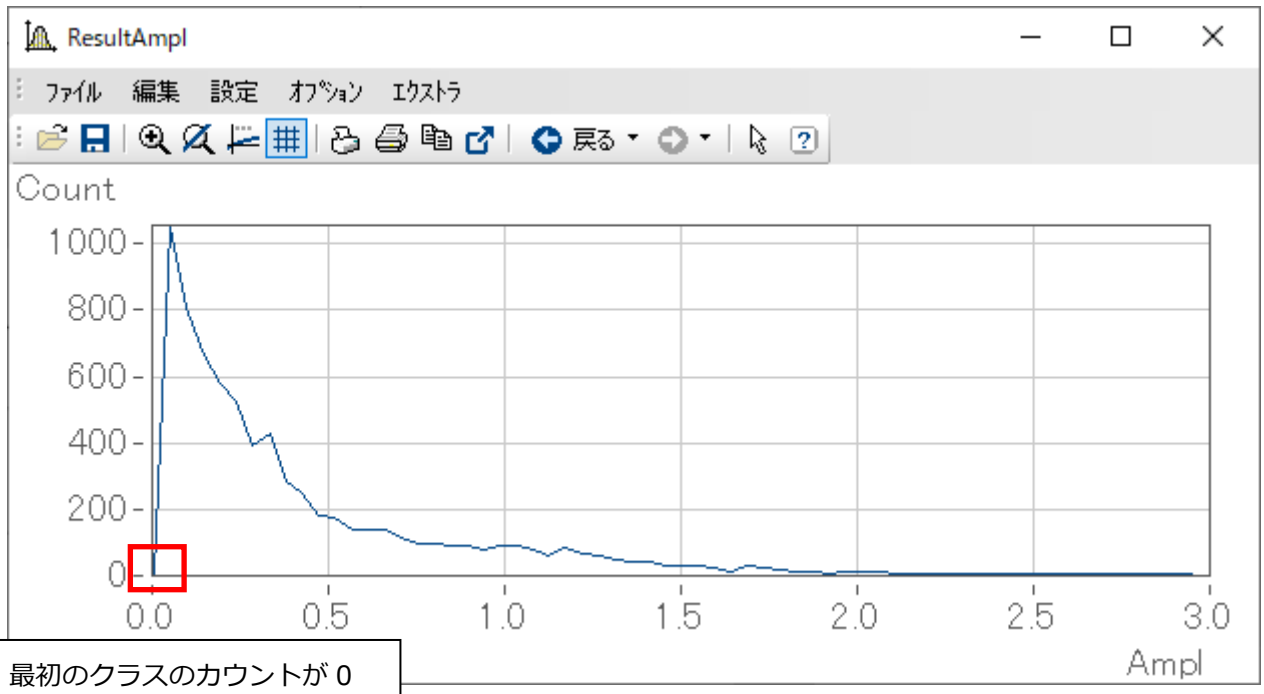
まずは、クラス数やクラス最小/最大等の基本的なパラメータをすべて同一となるように設定しましょう。
クラスの最小/最大ではなく、1 クラスの幅、等で設定するソフトウェアもあるので注意してください。

基本的なパラメータが同じであれば、imc FAMOS のカウント結果に影響が大きいものは解析アルゴリズム(_Calculation)と精度(_Precise)です。
これらの組み合わせを各種試してみることで、近い結果が得られることが多いです。

imc FAMOS のバージョンによっては、精度(_Precise)の選択肢が本セミナーで紹介したものよりも少ない可能性があることに注意してください。

3.6.3.最初のクラスのカウント結果が合わない

ここまでの例の場合、振幅クラスでまとめた結果として、1 クラス目のカウント結果が 0 となっています。



最初のクラスのカウントが 0

これはヒステリシスの大きさを 1 クラスの幅そのものとしている影響によります。

1 クラス目のカウント結果も出てくることが望ましい場合、基本的にヒステリシスは 0 を使用します。

ヒステリシスはソフトウェアによっては「無効振幅」と表現されていることもあります。

ヒステリシスが 0 にも関わらず 1 クラス目のカウント結果が 0 である/他の解析ソフトウェアの設定と合わないという場合は、

・精度(_Precise)の設定が小さすぎる

・小さなスパンをカウントしない設定になっている(_IgnoreSmallSpans = 1)

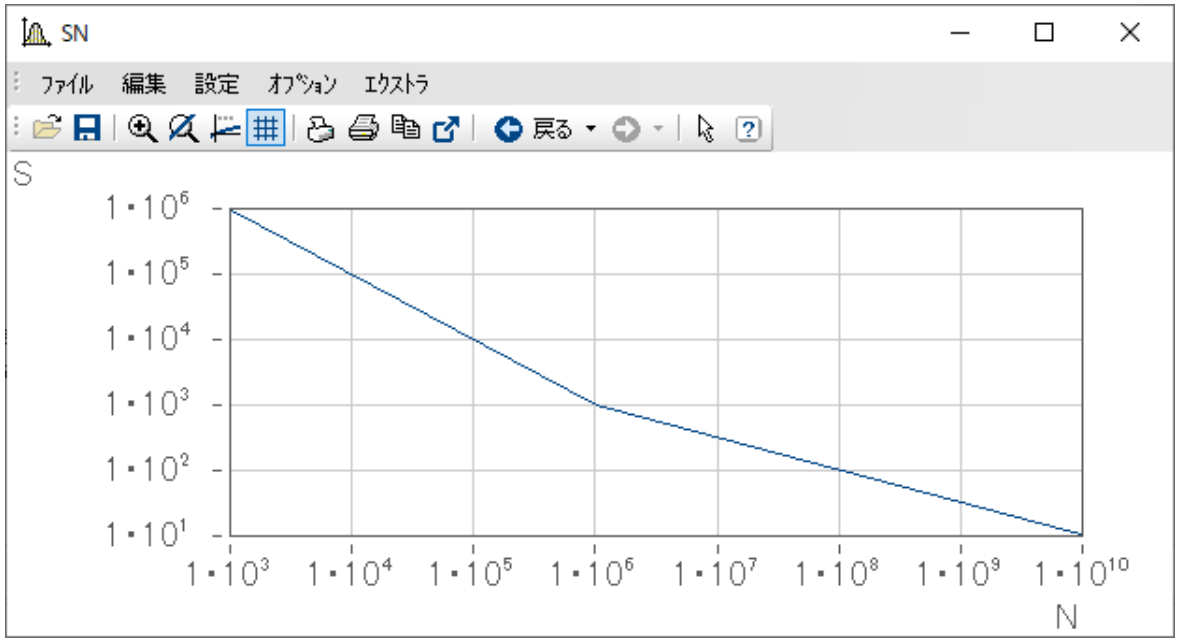
ということが考えられます。

4.疲労寿命推定

頻度解析の結果を利用する典型的な例としては疲労寿命推定があります。

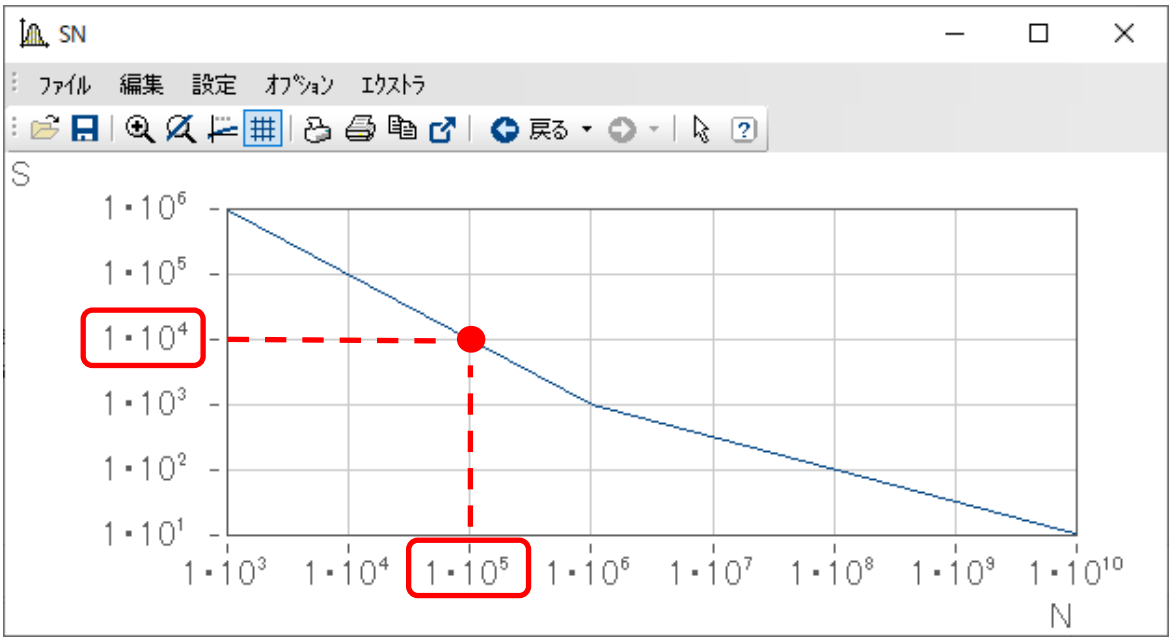
ここでは、S-N 線図を用いて材料・製品等の疲労寿命を推定する方法について説明します。

S-N 線図とは下図のような曲線で、応力(S)と繰り返し回数(N)の関係を表しており、材料・製品自体の特性を示すものです。

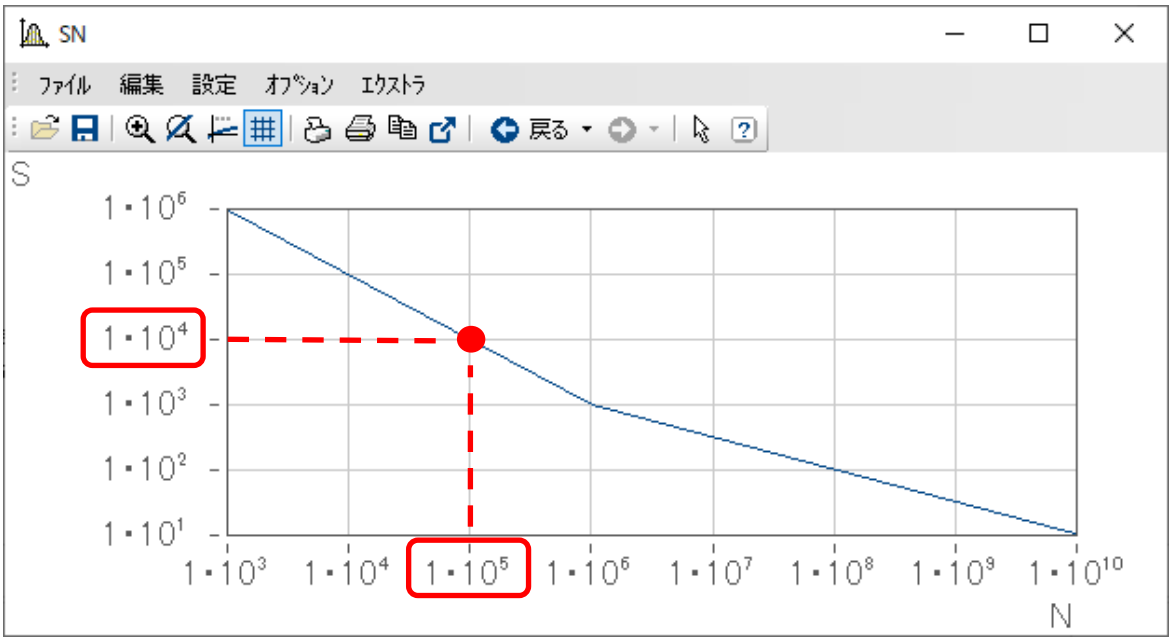


例えば下図に示す 1 点に着目してみましょう。

これは、この S-N 線図が示す対象は、10^4 程度の応力(S)を加えられた場合、10^5 程度の繰り返し回数で破断・破壊されるということを意味します。



疲労寿命推定では、この S-N 線図を元に、まず「ダメージ」を見積もります。
先ほど説明した 1 点、10^4 程度の応力(S)を加えられた場合、10^5 程度の繰り返し回数で破断・破壊される、ということは、10^4 程度の応力が「1 回」加えられた時のダメージは 1 / 10^5、と言い換えることができます(=ダメージの累計が 1 で破断・破壊)。



同じことがすべての応力の値に対して考えられるので、試験データを頻度解析した結果からは、その試験での累計のダメージ、を求めることができます。

そこから、対象の疲労寿命を推定できます。例えば以下のような考え方です。

試験データの累計ダメージ	0.02
試験データが示す時間(実際の動作環境に換算する)	10h
推定される疲労寿命	10h / 0.02 = 500h

「試験データが示す時間」は、例えば車両であれば距離[km]で表現してもよいでしょう。その場合は疲労寿命も距離[km]の単位で求められることになります。

次節からは、実際に imc FAMOS で疲労寿命推定を行う場合のシーケンス記述について説明していきます。

4.1.S-N 線図の作成

S-N 線図は、以下の条件を満たすのであればどのような方法で作成しても構いません。

・S > 0 かつ N > 0 であること。

・S が単調減少であること。

・N が単調増加であること。

ここでは一例として、シーケンス上で S-N 線図を定義する方法を紹介します。

4.1.1.各点の値を個別に指定する方法

以下のように、S と N の値をそれぞれ個別に指定することで S-N 線図を作成します。

```
----------------------------------------
; カンマ区切りで値を入力
_S = [1e6, 1e3, 1e1]
_N = [1e3, 1e6, 1e10]

; 結合、N(=横軸)が先であることに注意
SN = XYof(_N, _S)
----------------------------------------
```

4.1.2.S = a*N^b の式で定義する方法

S = a*N^b の式で与えられる S-N 線図を作成する場合は、以下のように記述します。

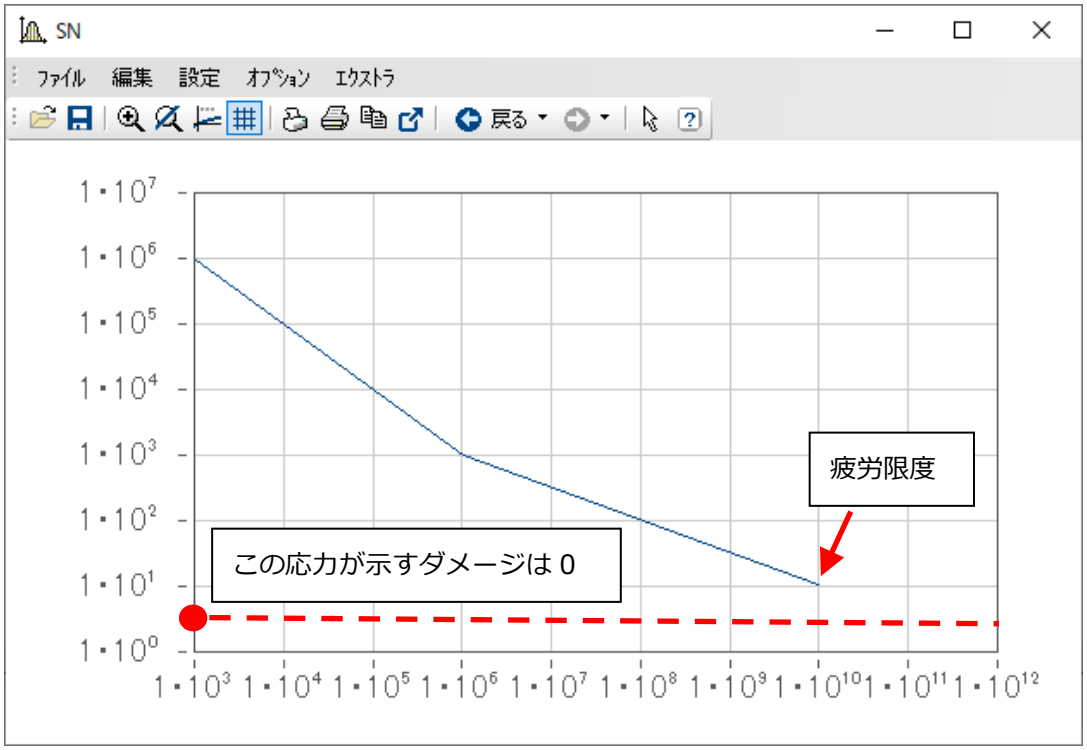
この S-N 線図は、両対数グラフ上では直線となります。

```
------------------------------------------------
; N の範囲を指定する、最小/最大の 2 点でよい
_N = [1, 1e10]

; 係数 a, b を指定する、特性上 b はマイナスの値
_a = 20
_b = -0.2

; S は式で定義
_S = _a * _N^_b

; 結合、N(=横軸)が先であることに注意
SN = XYof(_N, _S)
------------------------------------------------
```

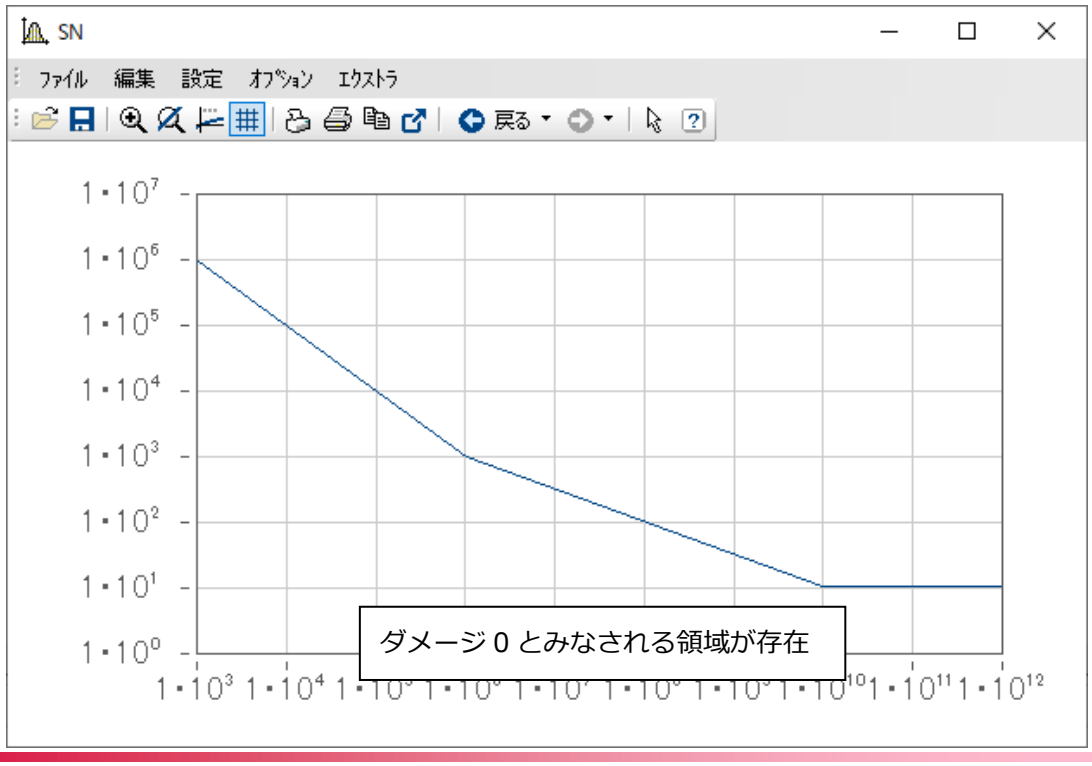
4.1.3.修正マイナー則を考慮する

以降に説明するシーケンスで累計ダメージを計算する場合、与えられた S-N 線図はマイナー則に従うものとして解釈されます。

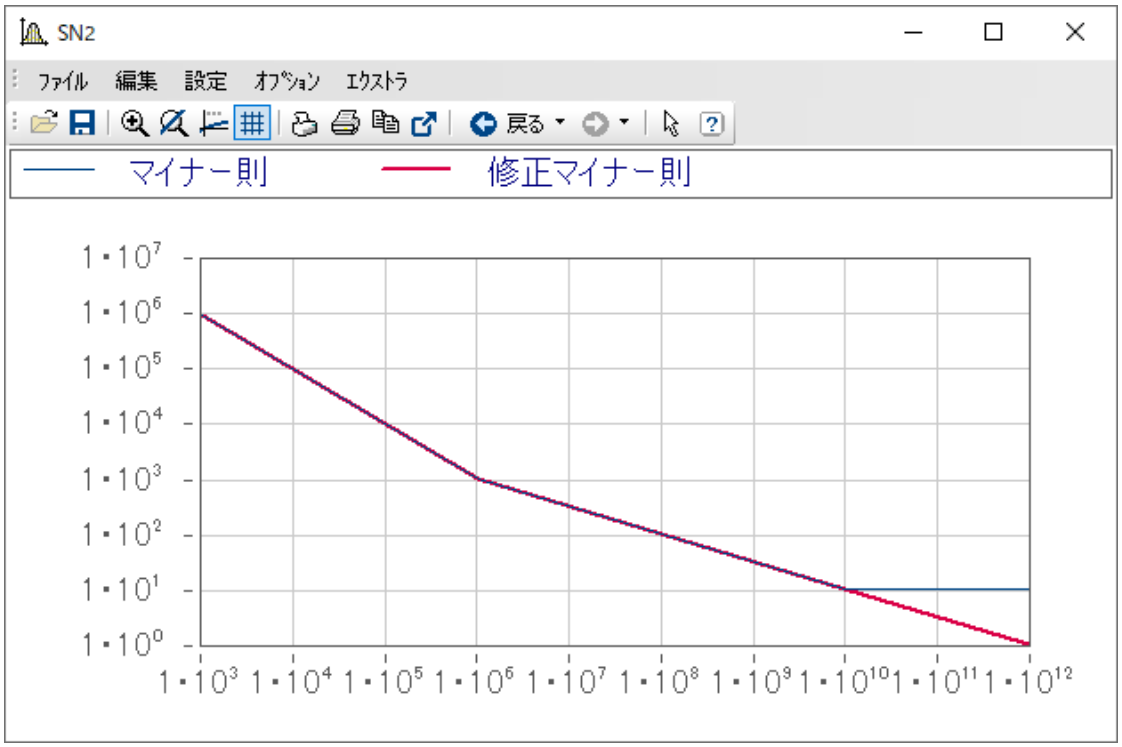
マイナー則では、疲労限度(最小応力)以下のダメージは 0 であるものとして扱われます。例えば下図の S-N 線図において、赤丸に示す点に対応するダメージは 0 です。



つまり、仮想的には S-N 線図は以下のように延長されているものとして扱われます。



しかし実際の材料は疲労限度以下の領域においてもダメージを受けることが知られており、疲労限度からさらに S-N 線図を延長した修正マイナー則がよく使われています。



明示的にマイナー則で解析する、あるいは S-N 線図が修正マイナー則適用済みである、解析上必要な範囲の応力はすべて定義済みである、という場合にはその S-N 線図をそのまま使って構いません。
そうでない場合は、次のようなシーケンスを用いることで、修正マイナー則が適用された S-N 線図を求めることが可能です。

4.1.3.1.修正マイナー則計算：両対数スケールの場合

以下は、応力 S、回数 N の軸ともに対数スケールで直線とする場合のシーケンス例です。

```
-----------------------------------------------------
; S-N 線図自体は SN という変数名で作成済みとする


; 求める範囲の最小応力の指定
_minS = 0.1


; S-N 線図の各要素
_S = SN.Y
_N = SN.X


; 最後の 2 点間を延長する
_index1 = Leng?(_S) - 1
_index2 = Leng?(_S)


; 最後の 2 点間の傾き
_a = (log(_S[_index1]) - log(_S[_index2])) / (log(_N[_index1]) -
log(_N[_index2]))


; 最小応力に対応する N
_logAddN = log(_N[_index2]) - ((log(_S[_index2]) - log(_minS)) / _a)
_addN = 10^logAddN


; 既存の S-N 線図に追加
_S = Join(_S, _minS)
_N = Join(_N, _addN)
SN = XYof(_N, _S)

Del _*
-----------------------------------------------------
```

考え方としては、ある 2 点間(S1, N1), (S2, N2)を通る直線 y = ax + b を考え、その直線が指定の最小応力 S を通るときの N を求める、ということを行っています。

随所で log を使用しているのは、両対数スケール上で直線となる系列を考えているためです。

4.1.3.2.修正マイナー則計算：片対数スケールの場合

同様に、応力 S がリニア、回数 N が対数スケールである場合に直線とするシーケンス例です。

```
----------------------------------------------------
; S-N 線図自体は SN という変数名で作成済みとする

; 求める範囲の最小応力の指定
_minS = 0.1

; S-N 線図の各要素
_S = SN.Y
_N = SN.X

; 最後の 2 点間を延長する
_index1 = Leng?(_S) - 1
_index2 = Leng?(_S)

; 最後の 2 点間の傾き
_a = (_S[_index1] - _S[_index2]) / (log(_N[_index1]) - log(_N[_index2]))

; 最小応力に対応する N
_logAddN = log(_N[_index2]) - ((_S[_index2] - _minS) / _a)
_addN = 10^logAddN

; 既存の S-N 線図に追加
_S = Join(_S, _minS)
_N = Join(_N, _addN)
SN = XYof(_N, _S)

Del _*
----------------------------------------------------
```

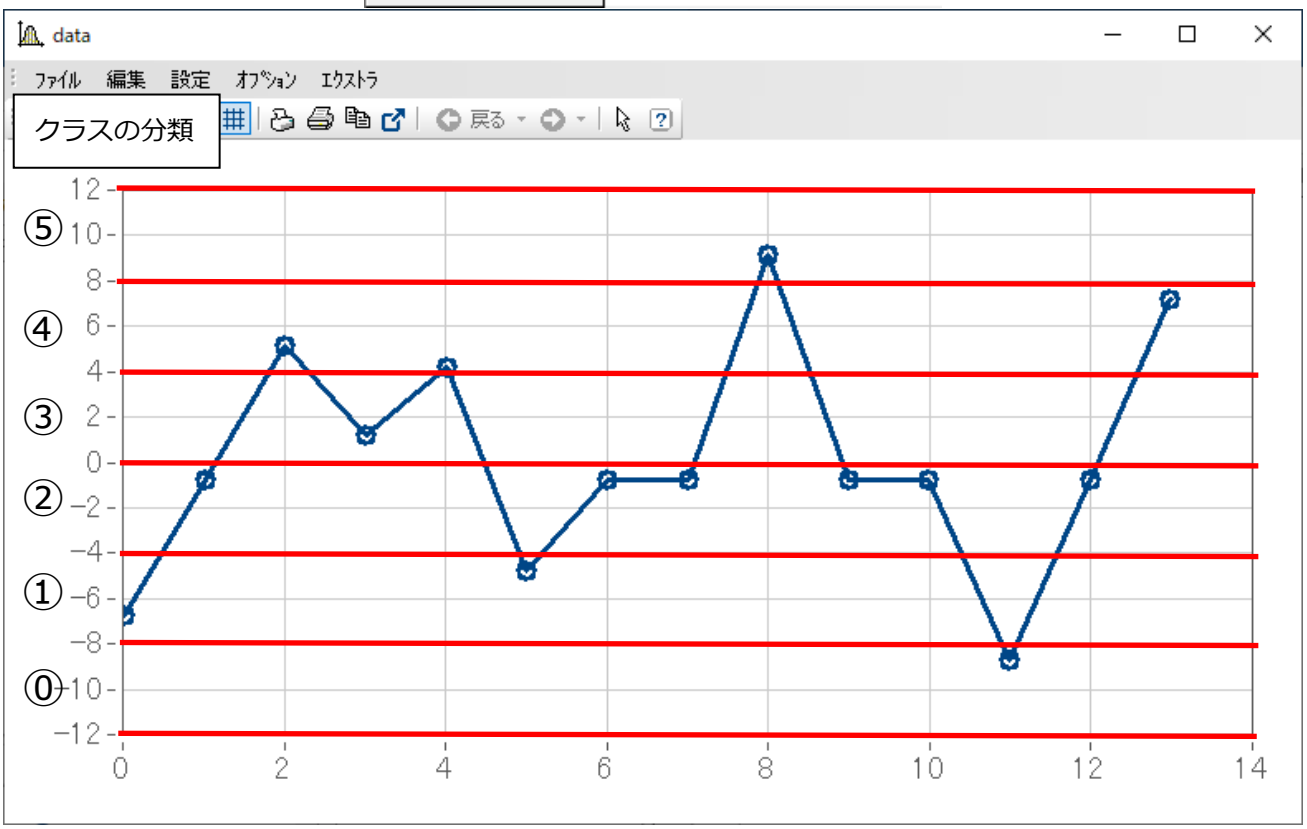
4.1.3.1 節のシーケンスと比較すると、応力 S の軸だけをリニアスケールで(=log を使用せずに)計算しています。

4.2.ダメージ計算/疲労寿命推定

4.2.1.クラスカウント法向け

クラスカウント法の場合、カウント結果は 0 から始まるインデックスでクラスに分類されています。
これは実際のデータでは、関数で設定したレンジ下限/上限と、分類するクラスの数から、どの値に相当するデータなのか区別されます。例えば下図の例では、Class の 0 は値としては-12～-8 の範囲です。

[Classes]	result
0	1.0
1	2.0
2	0.0
3	0.0
4	2.0
5	1.0



S-N 線図では応力 S として物理単位が使われるため、シーケンスではクラスのインデックスから物理単位に換算して計算を行う必要があります。
また、例えば上の例では Class の 0 は値としては-12～-8 の範囲ですが、この範囲で最大(-12)の値を取るか中間(-10)の値をとるか等は考え方次第ですが、以下の例では中間の値を使用します。

ここでは、それぞれ以下の名前を変数として使います。S-N 線図やカウント結果などは、このシーケンスの外で既に定義されているものとして扱っています。

SN	S-N 線図、応力 S の単位系はクラスカウント法の対象としたデータと同一 S-N 線図は両対数スケールとする
Result	クラスカウント法のカウント結果
_MaxValue	カウントしたレンジの上限(ClsPeak〇関数に使ったパラメータ)
_MinValue	カウントしたレンジの下限(ClsPeak〇関数に使ったパラメータ)
_NumberOfBins	分類されたクラス数(ClsPeak〇関数に使ったパラメータ)
Damage	S-N 線図とカウント結果から求められたダメージ
TrialTime	対象データが示す時間、距離など
LifeTime	推定された疲労寿命

シーケンス記述例は以下のようになります。

```
---------------------------------------------------------------------
; S-N を両対数スケールで検索するために換算
_xlog = Log(SN.X)
_ylog = Log(SN.Y)

; 1 クラスのレンジ幅と、最初のクラスの中間点を計算
_range = (_MaxValue - _MinValue) / _NumberOfBins
_class0 = _MinValue + _range / 2

; すべてのクラスでループ計算
Damage = 0
_i = 1
While _i <= _NumberOfBins
        ; そのクラスの振幅、応力 S に相当する
        _class = Abs(_class0 + _range * (_i - 1))

        ; 対応する回数 N を S-N 線図から検索する
        _slog = PosiEx(_ylog, Log(_class))

        ; 対応する回数 N がない(=S-N 線図の範囲外)の場合はダメージ 0
        If Leng?(_slog) <> 0
                _nlog = Value2(_xlog, _slog, 0)
                _N = 10^_nlog

                ; ダメージに換算する
                Damage = Damage + Result[_i] * (1 / _N)
        End

        _i = _i + 1
End

; 疲労寿命の推定
TrialTime = 10          ; 対象データ相当の時間/距離などの指定
LifeTime = TrialTime / Damage

Del _*
---------------------------------------------------------------------
```

S-N 線図が両対数スケールのため、応力 S から回数 N を検索する手順が少々特殊ですが、要点としては「クラスに対応する応力 S を求める」「応力 S に対応する回数 N を求める」「回数 N をダメージに換算して累計を取る」の 3 点のみです。

4.2.2.クラウカウント法向け：多チャンネルの場合

4.2.1 節相当で、カウント結果の Result が 1ch ではなく多チャンネルの結果が含まれるグループ変数である場合には、シーケンス記述例は以下のようになります。

```
; S-N を両対数スケールで検索するために換算
_xlog = Log(SN.X)
_ylog = Log(SN.Y)


; 1 クラスのレンジ幅と、最初のクラスの中間点を計算
_range = (_MaxValue - _MinValue) / _NumberOfBins
_class0 = _MinValue + _range / 2


; 結果のグループ変数
Damage = Result * 0
_i = 1
While _i <= GrChanNum?(Damage)
       Damage:[_i] = 0
       _i = _i + 1
End
```

(次ページに続く)

```
; すべてのチャンネル/クラスでループ計算
_i = 1
While _i <= GrChanNum?(Result)

        ; そのチャンネルのカウント結果
        _count = Result:[_i]

        _j = 1
        While _j <= _NumberOfBins

                ; そのクラスの振幅、応力 S に相当する
                _class = Abs(_class0 + _range * (_j - 1))

                ; 対応する回数 N を S-N 線図から検索する
                _slog = PosiEx(_ylog, Log(_class))

                ; 対応する回数 N がない(=S-N 線図の範囲外)の場合はダメージ 0
                If Leng?(_slog) <> 0
                        _nlog = Value2(_xlog, _slog, 0)
                        _N = 10^_nlog

                        ; ダメージに換算する
                        Damage:[_i] = Damage:[_i] + _count[_j] * (1 / _N)
                End

                _j = _j + 1
        End

        _i = _i + 1
End

; 疲労寿命の推定
TrialTime = 10          ; 対象データ相当の時間/距離などの指定
LifeTime = TrialTime / Damage

Del _*
-----------------------------------------------------------------------
```

チャンネルとクラスでループが二重となります。

4.2.3.レインフロー法向け

レインフロー法の場合、専用の関数が存在しており簡単な記述で疲労寿命推定を行うことができます。
ここでは、それぞれ以下の名前を変数として使います。

SN	S-N 線図、応力 S の単位系はレインフロー法の対象としたデータと同じであり、レンジ(振動の最大値-最小値)でスケールが記述されている必要があります。
ClsHandle	ClsOffRainflowInit1 関数で作成された変数(カウントまで完了していること)
Damage	S-N 線図とカウント結果から求められたダメージ
TrialTime	対象データが示す時間、距離など
LifeTime	推定された疲労寿命

レインフロー法の場合、上記の ClsHandle に情報が含まれているため、軸がどのように設定されているか等で疲労寿命推定のシーケンスを変える必要はありません。

シーケンスの記述は以下のようになります。

```
---------------------------------------------------------------------------------------------
; ダメージ計算、後ろ 2 つのパラメータは任意に設定する
Damage = ClsOffWoehlerSN(ClsHandle, SN, _ClassRelation, _Interpolation)

; 疲労寿命の推定
TrialTime = 10        ; 対象データ相当の時間/距離などの指定
LifeTime = TrialTime / Damage
---------------------------------------------------------------------------------------------
```

ClsOffWoehlerSN のパラメータが示す意味は以下の通りです。

ClsOffWoehlerSN(ClsHandle, SN, _ClassRelation, _Interpolation)

ClsHandle	ClsOffRainflowInit1 関数で作成された変数
SN	S-N 線図
_ClassRelation	カウントのクラスは実際には幅があるが、その幅の中でどの程度の応力と見積もるかの設定 0：クラス全体の範囲で等しく分布していると考えます 1：クラスの最大値であると考えます (=最もダメージが大きくなる) 2：クラスの中間点であると考えます
_Interpolation	S-N 線図の補間の仕方 0：S-N 線図は両対数スケールとして補間する 1：S-N 線図は応力 S がリニア、回数 N が対数スケールとして補間する

4.2.4.レインフロー法向け：多チャンネルの場合

多チャンネルのレインフロー解析で疲労寿命推定も行う場合は、3.5 節のシーケンスの途中に ClsOffWoehlerSN 関数の処理を挟み込むだけです。

```
--------------------------------------------------------------------------------
; ループ処理で全チャンネルを解析
_i = 1
While _i <= GrChanNum?(EditData)

        ; カウント対象となるデータを与える
        ClsOffRainflowFeedSamples(ClsHandle, EditData:[_i])

        ; レジデューを結果に加える
        ClsOffRainflowAddResidue(ClsHandle, _Weight)

        ; 結果を返す
        RainflowMatrix:[_i] = ClsOffRainflowGetMatrix(ClsHandle)

        (ここに ClsOffWoehlerSN 関数の処理を挿入)

        ; リセット
        ClsOffRainflowClearMatrix(ClsHandle)
        ClsOffRainflowClearResidue(ClsHandle)

        _i = _i + 1
End
--------------------------------------------------------------------------------
```

5.サポート

本テキストで解説した imc FAMOS の基本的な操作を今後の解析ならびにデータ整理、ドキュメント作成にご活用下さい。詳細な解析やドキュメント作成、シーケンスに関するサンプルや FAQ は弊社ホームページにてご利用いただけます。

imc FAMOS ホームページ
https://www.toyo.co.jp/mecha/products/detail/imc-famos.html

imc FAMOS FAQ ページ
https://www.toyo.co.jp/mecha/faq/

imc ダウンロードサイト(評価版、マニュアル、インポートフィルタ等がダウンロード可能です)
https://www.toyo.co.jp/mecha/contents/detail/imc_download_site.html

また、弊社スタッフまで直接ご質問・ご相談がある場合は下記連絡先までご連絡下さい。

株式会社東陽テクニカ
機械技術部　FAMOS サポート
機械計測部　FAMOS 担当
TEL :　03-3279-0771(代表)
FAX :　03-3246-0645
E-mail :　imc@toyo.co.jp