○「New MQL4（Build 600 以降）；基礎（その３）OnTester()を使う」　2014.08.17

・アメンボです、

New_MQL4 の特徴とはなんでしょう？

極論を言うと、MQL5 から受け継いだ「OOP」、つまり「On***()」関数群と「標準クラス」ライブラリではないかと、筆者は勝手に考えています。

・「On***()」関数群は、関連資料が比較的読みやすいです！

と、言うわけで簡単なものからコツコツと解析・実証結果を報告します。

（一方、標準クラス・ライブラリは解析が難題です。機能は結構凄いのですが！）

＜本稿で使用した MQL4 コード＞

※使用コードと「.set」ファイルを添付しました。

「new_mql4_2014_08_02.zip；New MQL4 基礎（その３）」(ZIP 形式圧縮)

※本稿は「MT4；version 4.00 Build670」「MetaEditor；version 5.00 Buid966」にて確認済み。

目次：

１．「On****()」ハンドリング関数（MQL5 との比較）

New_MQL4 で使用可能な「ハンドリング関数」を MQL5 の場合と比較しながら、使用方法を解説します

		機能ｻﾎﾟｰﾄ		New MQL4 のｻﾎﾟｰﾄ範囲			確認
				EA	Indicator	Script	
ﾊﾝﾄﾞﾘﾝｸﾞ関数	ｲﾍﾞﾝﾄ・ﾄﾘｶﾞと ﾓｰﾄﾞ別	MQL5	New MQL4	関数使用	ｲﾝﾃﾞｨｹｰﾀ表示	ｽｸﾘﾌﾟﾄ実行	
OnStart()	—	○	○	○	—	○	
OnInit()	開始	○	○	○	○	—	
OnDeinit()	終了	○	○	○	○	—	
OnTick()	ﾃｨｯｸ	○	○	○	—	—	
	ﾏﾙﾁｶﾚﾝｼｰ･ﾓｰﾄﾞ	○	?	?	—	—	
OnTimer()	ﾀｲﾏｰ	○	○	○	○	—	済
OnTrade()	order・deal・position	○	—	—	—	—	
OnTradeTransaction()		○	—	—	—	—	
OnTester()	ｽﾄﾗﾃｼﾞｰ・ﾃｽﾀｰ	○	○	○	—	—	本稿
OnBookEvent()	板(DOM)情報	○	—	—	—	—	
OnChartEvent()	未確認	○	?	?	?	—	
	ｶｽﾀﾑ･ｲﾍﾞﾝﾄ	○	○	○	○	—	
OnCalculate()	ｲﾝﾃﾞｨｹｰﾀ表示計算	○	○	—	○	—	済
	簡略ﾀｲﾌﾟ	○	?	—	?	—	

２．OnTester()使用法のイメージを掴む

まず、本節では「OnTester()」機能の全体イメージを掴むことを目指します。
（実際の活用法は、次節を参照ください）

本節は下記の資料を参考にしました、チョット読みでがあります。(筆者は飛ばし読み中)
出典；「The Fundamentals of Testing in MetaTrader5」一読を薦めます

（１）OnTester()使用上の要点

OnTester()の機能と動作；

使用場面	OnTester()内の記述処理 （例；任意[Custom]指標）	［エキスパート設定］ ［テスト中］タブの ［最適化パラメータ］
通常の EA 実行（RUN）	実行されない	―
バックテスト時	１回実行される	―
最適化（ｵﾌﾟﾃｨﾏｲｽﾞ）実行時	繰り返し実行され、「任意指標」の最適化に使える ・［最適化グラフ］は Balance 値を表示する	Balance
	繰り返し実行され、「任意指標」の最適化に使える ・［最適化グラフ］は Custom 値を表示する	Custom

※上記で、「Balance」「Custom」のどちらで実行しても、組合せ実行「原データ」は同一であるが、
［最適化グラフ］に表示される内容が異なってくる。

（２）イメージ把握用ＥＡ

OnTester()の動作を理解するための一例として下記を選択する。

OnTester()内の記述；

```
double OnTester()
  {
   double sink=MathSin(x*x+y*y);
   return(sink);
  }
```

※これは、数学の Sin(x*x+y*y)を計算して返す（return する）、内容です。
最適化を目的としているのではなく、「x」と「y」を少しづつ変化させて、
繰り返し計算していると言うのが、本当のところです。

※「x」と「y」を少しづつ変化させたいので、「総当たり」を使います。
（たぶん、遺伝的アルゴリズムでも、結果はそれほど変わらないとは思いますが）

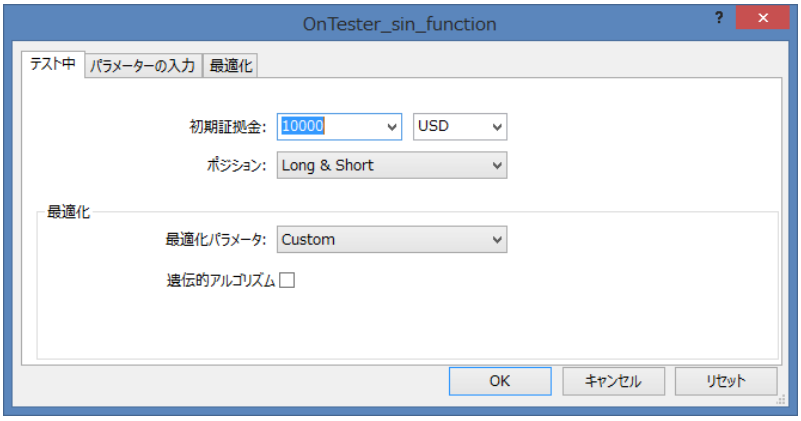
（3）使用するコード

```
//+------------------------------------------------------------------+
//|                                        OnTester_sin_function.mq4 |
//|                                                           amenbo |
//|                                                泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"
#property version   "1.00"
#property strict
//
//--- input parameters
input double   x=-3.0; // start=-3, step=0.05, stop=3
input double   y=-3.0; // start=-3, step=0.05, stop=3
//
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---

  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
   //---
   double sink=MathSin(x*x+y*y);
   //
   return(sink);
   //
  }
//+------------------------------------------------------------------+
```

※「青書」以外の部分は、『空っぽ』のＥＡです！

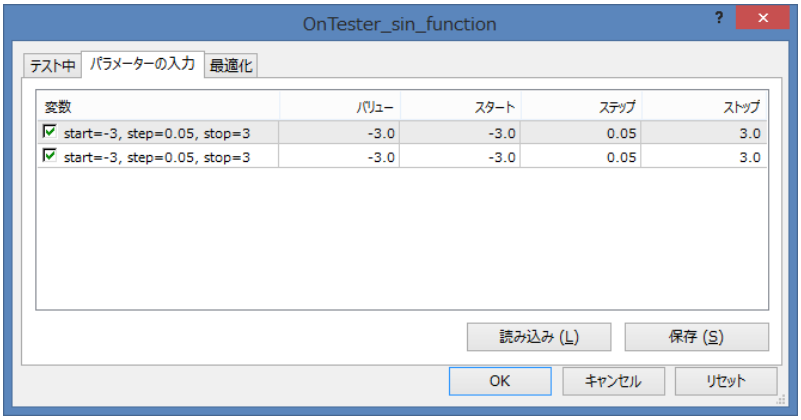
（4）設定内容

［エキスパート設定］の各タブ内容設定



・［最適化パラメータ］は、Custom に設定します。

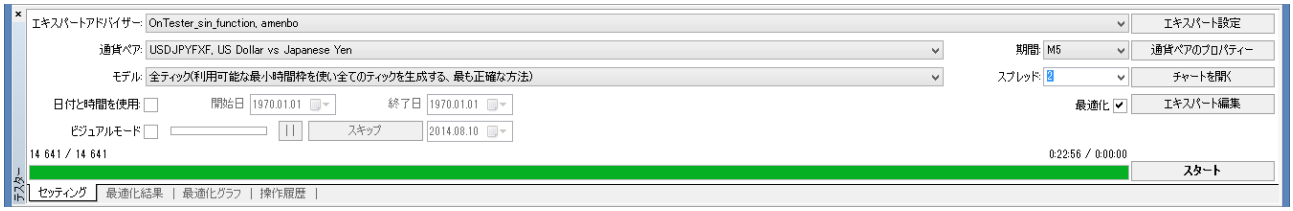
・［遺伝的アルゴリズム］は使いません。
（総当たりスキャニングのため）



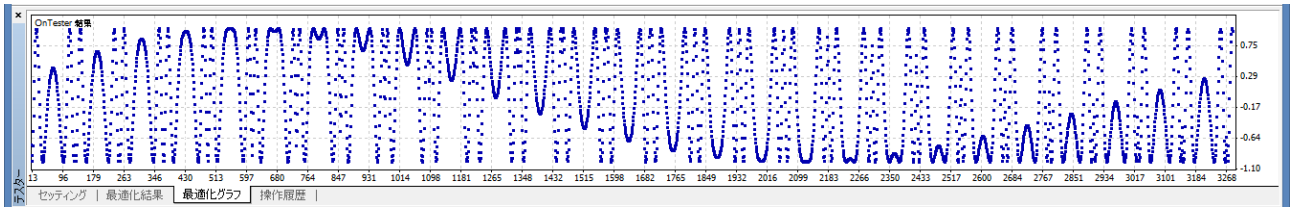
・X と Y は「-3.0～+3.0」の範囲で 0.05 ステップづつ変化させます。

（5）実行結果

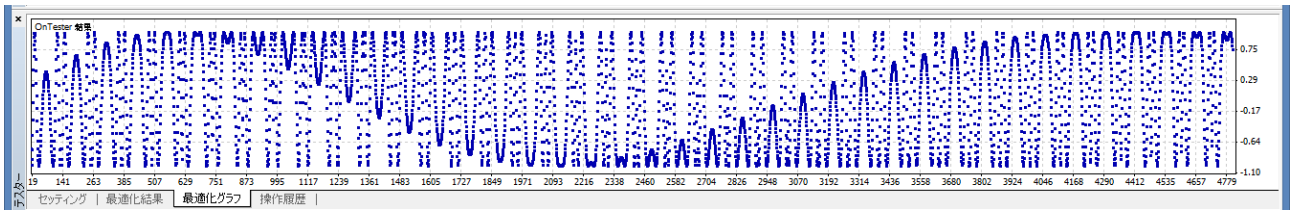
・時間の経過と従って続行結果を示します。



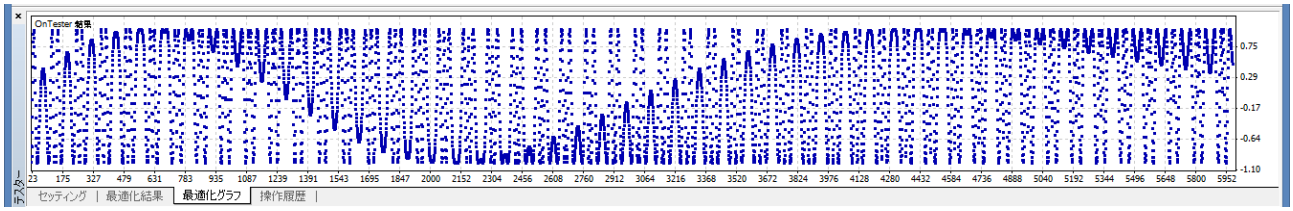
・最適化の途中で［最適化グラフ］の時間変化を覗いてみた。
　じっと見ていると、最適化の進行と共にグラフがどんどん変化していきます。



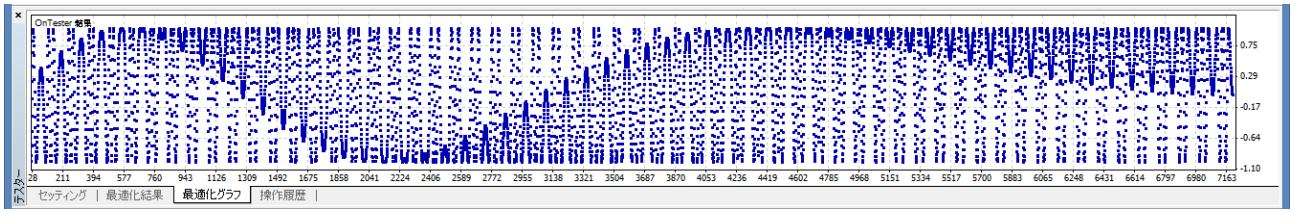
↓



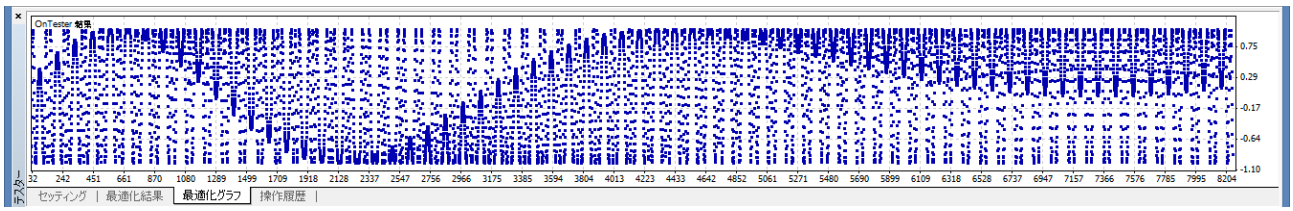
↓



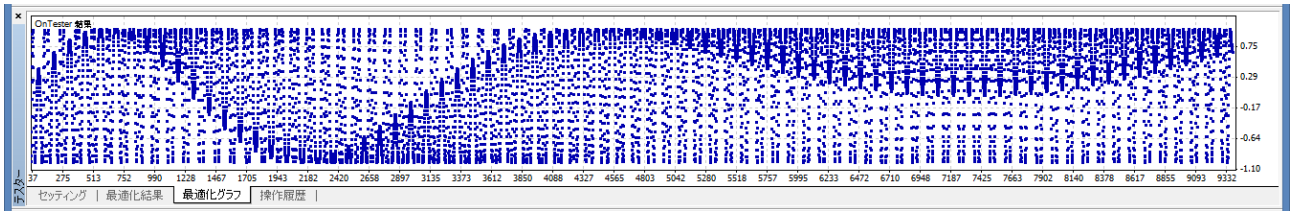
↓



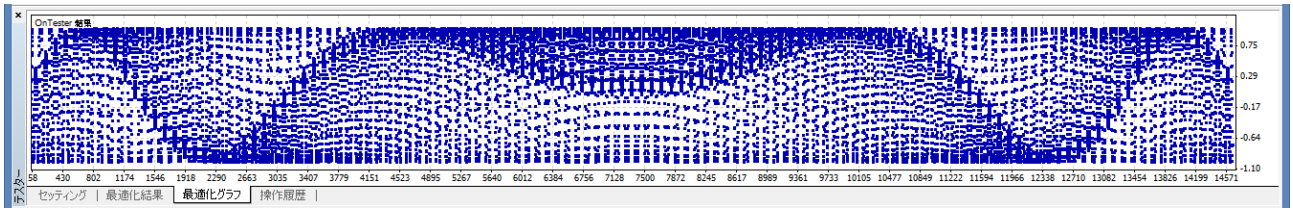
↓



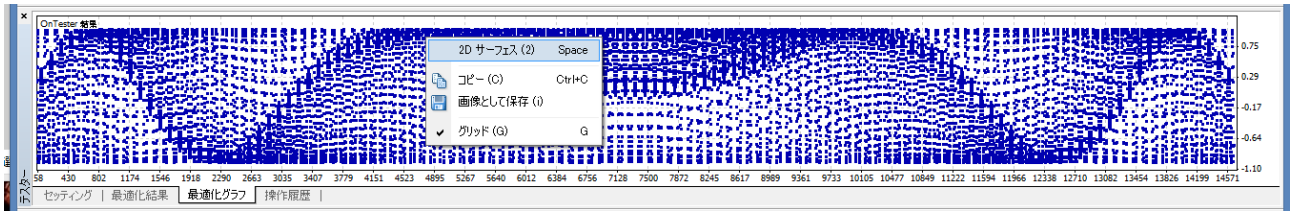
↓



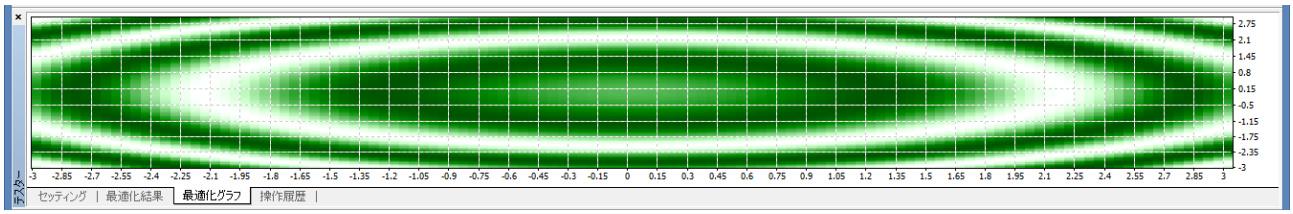
↓



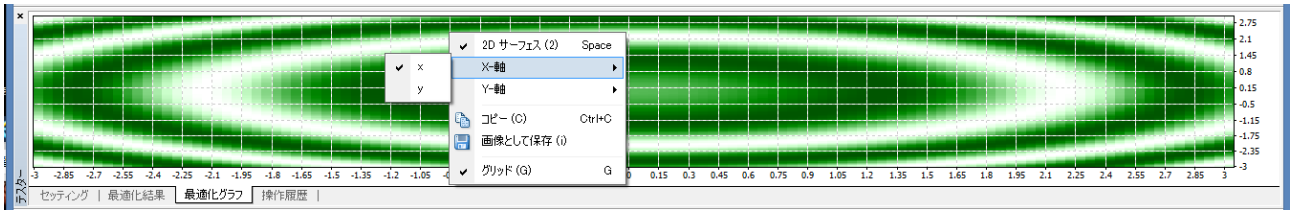
・最適化が終了したので、２次元表示（2D サーフェス）に切り替えます。



↓



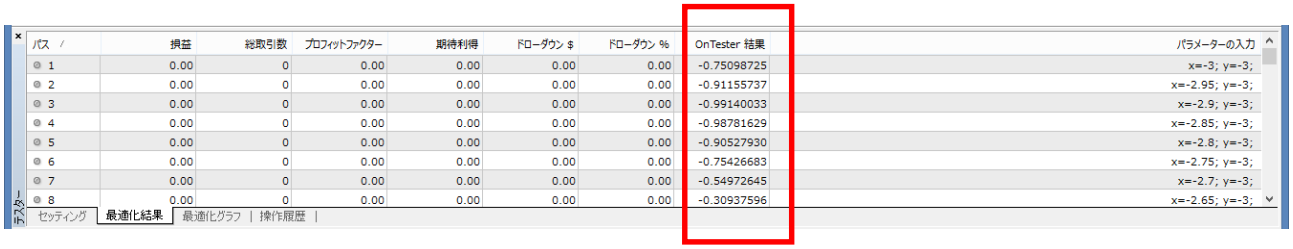
↓



※パラメータは「x」と「y」の２つです。

（6）ここに注目

※［OnTester 結果］の欄が追加されています。



パス	損益	総取引数	プロフィットファクター	期待利得	ドローダウン $	ドローダウン %	OnTester 結果	パラメーターの入力
1	0.00	0	0.00	0.00	0.00	0.00	-0.75098725	x=-3; y=-3;
2	0.00	0	0.00	0.00	0.00	0.00	-0.91155737	x=-2.95; y=-3;
3	0.00	0	0.00	0.00	0.00	0.00	-0.99140033	x=-2.9; y=-3;
4	0.00	0	0.00	0.00	0.00	0.00	-0.98781629	x=-2.85; y=-3;
5	0.00	0	0.00	0.00	0.00	0.00	-0.90527930	x=-2.8; y=-3;
6	0.00	0	0.00	0.00	0.00	0.00	-0.75426683	x=-2.75; y=-3;
7	0.00	0	0.00	0.00	0.00	0.00	-0.54972645	x=-2.7; y=-3;
8	0.00	0	0.00	0.00	0.00	0.00	-0.30937596	x=-2.65; y=-3;

※これは、[最適化パラメータ]として「Balance」「Custom」等、どれを選択した場合も同じです。
但し、

３．OnTester()の準備

※本節ではＥＡにCustom指標を組み込んで最適化を行った場合の検証を行います、
ただ、筆者にとって、いまいち動作が理解できなかった部分があるので、多少余分な検証作業を含んでいます。

出典；「Creating Custom Criteria of Optimization of Expert Advisors」一読を薦めます

（１）使用するＥＡ「Cross_Bollin_EA_OnTester.mq4」の作成法

・2012年5月に作成した「Bollin_EA_08.mq4」のコードをそのまま使用し、Onester()を単純に追加しました。　「Bollin_EA_08.mq4」　⇒　「Cross_Bollin_EA_OnTester.mq4」

・Cross_Bollin_EA_OnTester.mq4は、「Bollin_EA_08.mq4のコード」＋「OnTester()関数」と言う至って単純な合成で作成しました。

（２）組み込む「Custom指標」について

OnTester()内の記述；　「//」でコメント化している部分は省略

```
double OnTester()
  {
      double profit_On = TesterStatistics(STAT_PROFIT);
      double max_dd = TesterStatistics(STAT_BALANCE_DD);
      if(max_dd>0)
      {
          double rec_factor = profit_On/max_dd;
          return(rec_factor);
      }
      else return(0.0);
  }
```

最小必須知識；　上記の解説に必要な最小限の知識をまとめました

```
double TesterStatistics(
        ENUM_STATISTICS statistics_id
        );
```

・返し値＝［ENUM_STATISTICS id］で指定した統計値データ

・ENUM_STATISTICS id；(抜粋) 詳細はMT5のDocumentサイトを参照してください

ID	統計値の内容	TYPE
STAT_PROFIT	正味の損益	double
STAT_BALANCE_DD	最大ドローダウン	double
・・・・・・		
STAT_GROSS_PROFIT	トータル収益	double
STAT_GROSS_LOSS	トータル損失	double
・・・・・・		

※「STAT_PROFIT＝STAT_GROSS_PROFIT＋STAT_GROSS_LOSS」が成立します。

※「double TesterStatistics(」と定義されているが、返し値（TYPE）には「int」もある。

OnTester()内の解説；

```
double OnTester()
  {
  double profit_On = TesterStatistics(STAT_PROFIT);  ・・・profit_On＝利益（損益）
  double max_dd = TesterStatistics(STAT_BALANCE_DD); ・・・max_dd＝最大ドローダウン
  if(max_dd>0)   ・・・最大ドローダウンがプラス値なら
  {
      double rec_factor = profit_On/max_dd; ・・・［利益／最大ドローダウン］を計算する
      return(rec_factor);                   ・・・［利益／最大ドローダウン］値を返す
  }
  else return(0.0);   ・・・最大ドローダウンが「0；ゼロ」の場合は、「0.0」を返す
  }
```

※ここで使用する OnTester()は［損益／最大ドローダウン］値を返します
ただし、割り算で分母がゼロ（0）になる場合を回避しています。
※つまり、任意指標として［利益／最大ドローダウン］値を採用し、これを最適化することを検討します。

４．OnTester()内記述の Custom 値の最適化を行う

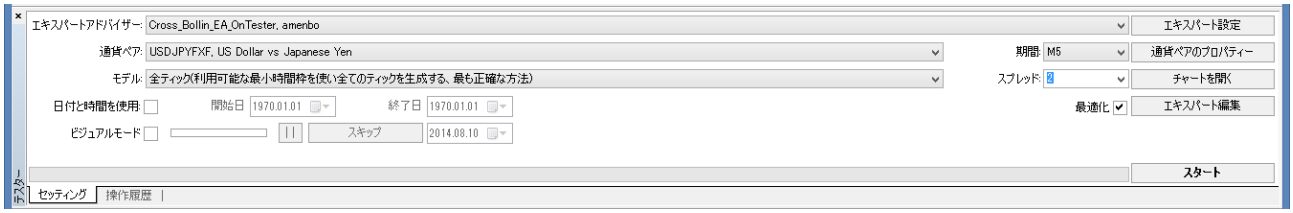
※結果比較のために、同じＥＡを使って、「最適化パラメータ＝Custom」と、通常利用する「最適化パラメータ＝Balance」での２通りの設定で最適化を行ってみます。
※最適化を実施した「スキャニング範囲」、および最適化完了後に得られた Custom と Balance が「最大値」になる各「extern　要素」も設定した「.set」ファイルを下記の名称で添付しています。

最適化［対象］	設定ファイル名称	［対象］最適化後の extern 値セット済	最適化スキャニング範囲データセット済
Custom 値	08_OnTrade_max_Custom.set	○	○
Balance	08_OnTrade_max_Balance.set	○	○

（１）「最適化パラメータ＝Custom」として実行する

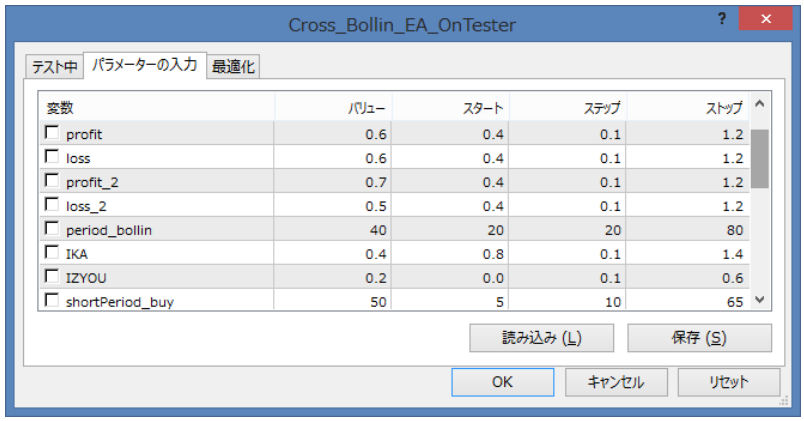
－１．ＥＡ、通貨ペア、モデルを設定する

「Cross_Bollin_EA_Ontester」、「USDJPYFXF」、「全ティック」



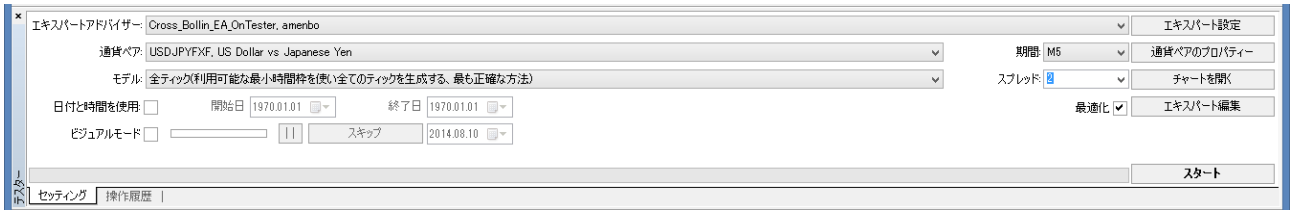
－２．［エキスパート設定］中の［最適化パラメータ］と［パラメータの入力］

※時間短縮のため、
［遺伝的アルゴリズム］を使います

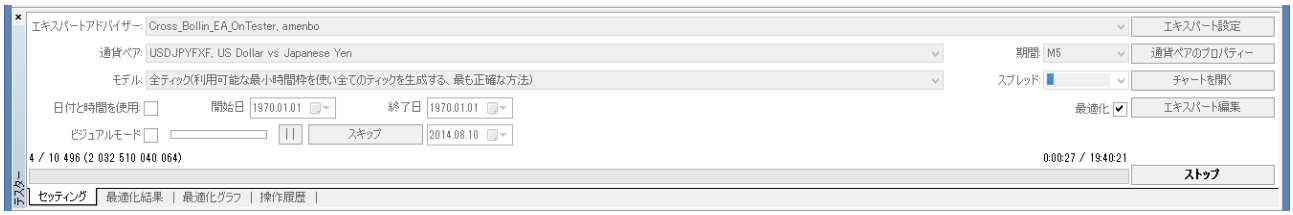


※［パラメータの入力］値は、
「Cross_Bollin_EA_OnTrade.mq4」
コード中に記載された値が
表示されます。

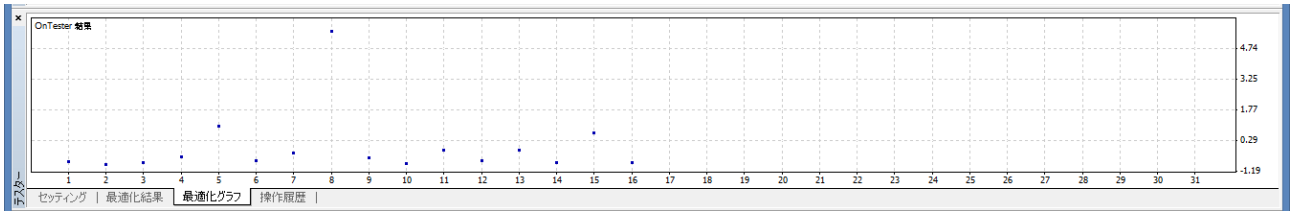
－３．ＥＡの最適化をスタートする



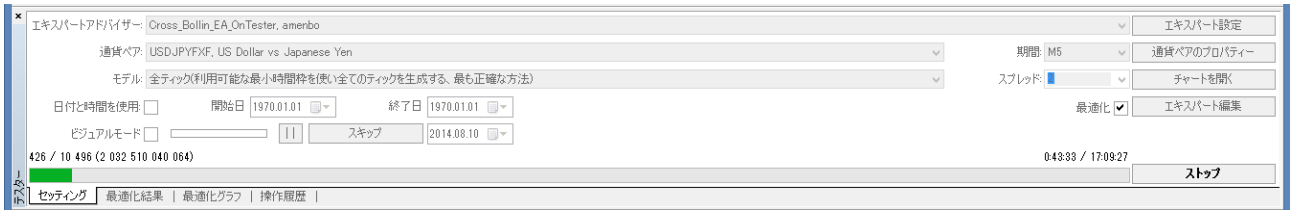
↓

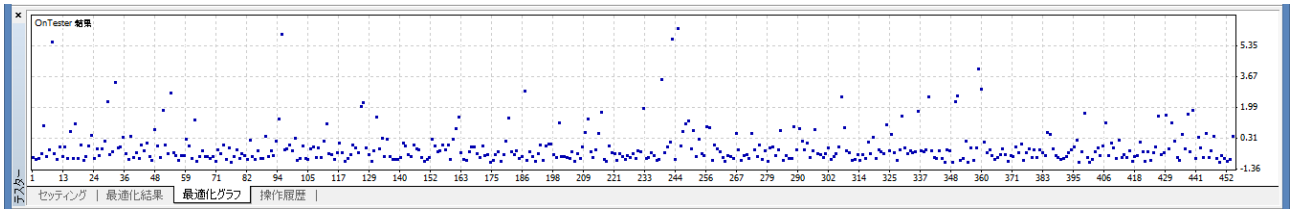


※19 時間もかかる！？、ともかく最適化の進行と、［最適化グラフ］を観察します。

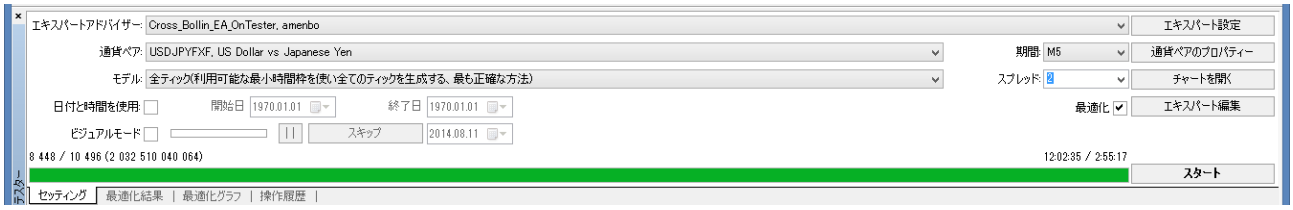


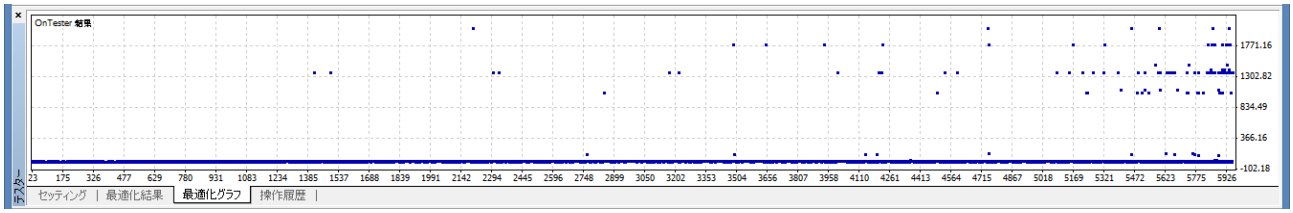
↓





↓

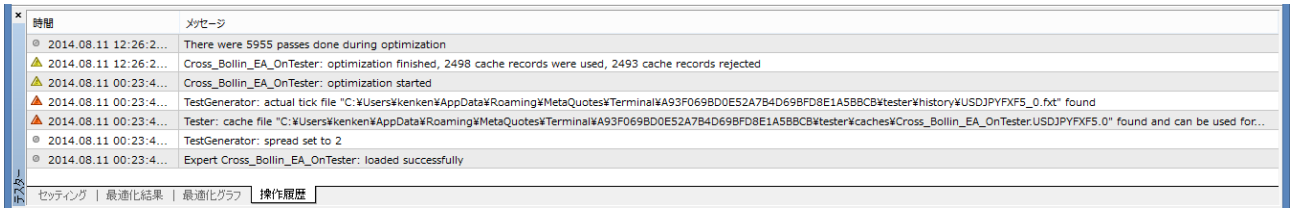




※ようやく終わったので、次は［最適化グラフ］以外にも詳細を調べます。
（でも何か［最適化グラフ］はスッキリしないし、面白くない）

－４．最適化結果を調べる

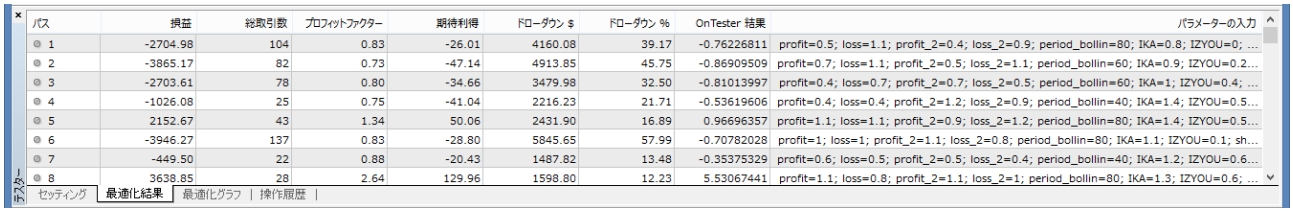
［動作履歴］



※「Tester:cache file ***」と「TestGenerator:actual tick file ***」部コメントの意味が良く理解できず。（今後の課題です）

［最適化結果］

A.「パス」順で表示、「OnTester 結果」部を拡大表示します



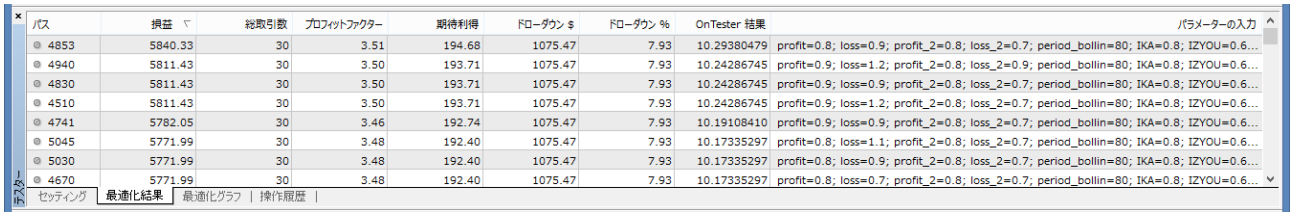
パス	損益	総取引数	プロフィットファクター	期待利得	ドローダウン $	ドローダウン %	OnTester 結果	パラメーターの入力
1	-2704.98	104	0.83	-26.01	4160.08	39.17	-0.76226811	profit=0.5; loss=1.1; profit_2=0.4; loss_2=0.9; period_bollin=80; IKA=0.8; IZYOU=0; ...
2	-3865.17	82	0.73	-47.14	4913.85	45.75	-0.86909509	profit=0.7; loss=1.1; profit_2=0.5; loss_2=1.1; period_bollin=60; IKA=0.9; IZYOU=0.2...
3	-2703.61	78	0.80	-34.66	3479.98	32.50	-0.81013997	profit=0.4; loss=0.7; profit_2=0.7; loss_2=0.5; period_bollin=60; IKA=1; IZYOU=0.4; ...
4	-1026.08	25	0.75	-41.04	2216.23	21.71	-0.53619606	profit=0.4; loss=0.4; profit_2=1.2; loss_2=0.9; period_bollin=40; IKA=1.4; IZYOU=0.5...
5	2152.67	43	1.34	50.06	2431.90	16.89	0.96696357	profit=1.1; loss=1.1; profit_2=0.9; loss_2=1.2; period_bollin=80; IKA=1.4; IZYOU=0.5...
6	-3946.27	137	0.83	-28.80	5845.65	57.99	-0.70782028	profit=1; loss=1; profit_2=1.1; loss_2=0.8; period_bollin=80; IKA=1.1; IZYOU=0.1; sh...
7	-449.50	22	0.88	-20.43	1487.82	13.48	-0.35375329	profit=0.6; loss=0.5; profit_2=0.5; loss_2=0.4; period_bollin=40; IKA=1.2; IZYOU=0.6...
8	3638.85	28	2.64	129.96	1598.80	12.23	5.53067441	profit=1.1; loss=0.8; profit_2=1.1; loss_2=1; period_bollin=80; IKA=1.3; IZYOU=0.6; ...

テスター　セッティング　最適化結果　最適化グラフ　操作履歴

ドローダウン %	OnTester 結果	
39.17	-0.76226811	profit=0.5; loss=1.1; profit
45.75	-0.86909509	profit=0.7; loss=1.1; profit_
32.50	-0.81013997	profit=0.4; loss=0.7; profit
21.71	-0.53619606	profit=0.4; loss=0.4; profit_
16.89	0.96696357	profit=1.1; loss=1.1; profit_
57.99	-0.70782028	profit=1; loss=1; profit_2=
13.48	-0.35375329	profit=0.6; loss=0.5; profit_
12.23	5.53067441	profit=1.1; loss=0.8; profit

※「OnTester 結果」が追加で表示されます。

B.「損益」順で収益の大きい方から表示；「損益」部をクリックし、昇順・降順を切替える

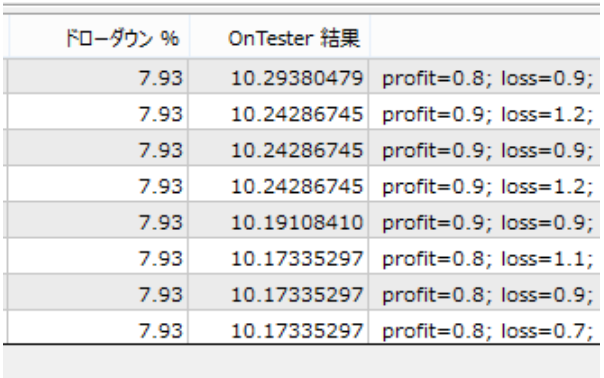


パス	損益	総取引数	プロフィットファクター	期待利得	ドローダウン $	ドローダウン %	OnTester 結果	パラメーターの入力
4853	5840.33	30	3.51	194.68	1075.47	7.93	10.29380479	profit=0.8; loss=0.9; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...
4940	5811.43	30	3.50	193.71	1075.47	7.93	10.24286745	profit=0.9; loss=1.2; profit_2=0.8; loss_2=0.9; period_bollin=80; IKA=0.8; IZYOU=0.6...
4830	5811.43	30	3.50	193.71	1075.47	7.93	10.24286745	profit=0.9; loss=0.9; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...
4510	5811.43	30	3.50	193.71	1075.47	7.93	10.24286745	profit=0.9; loss=1.2; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...
4741	5782.05	30	3.46	192.74	1075.47	7.93	10.19108410	profit=0.9; loss=0.9; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...
5045	5771.99	30	3.48	192.40	1075.47	7.93	10.17335297	profit=0.8; loss=1.1; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...
5030	5771.99	30	3.48	192.40	1075.47	7.93	10.17335297	profit=0.8; loss=0.9; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...
4670	5771.99	30	3.48	192.40	1075.47	7.93	10.17335297	profit=0.8; loss=0.7; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...

テスター　セッティング　最適化結果　最適化グラフ　操作履歴

パス	損益	総取引数	プロフィットファクター
4853	5840.33	30	3.51
4940	5811.43	30	3.50
4830	5811.43	30	3.50
4510	5811.43	30	3.50
4741	5782.05	30	3.46
5045	5771.99	30	3.48
5030	5771.99	30	3.48
4670	5771.99	30	3.48

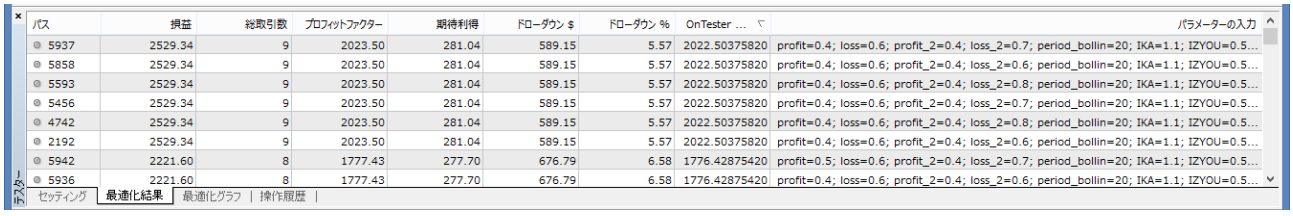
テスター　セッティング　最適化結果　最適化グラフ　操作履歴



ドローダウン %	OnTester 結果	
7.93	10.29380479	profit=0.8; loss=0.9;
7.93	10.24286745	profit=0.9; loss=1.2;
7.93	10.24286745	profit=0.9; loss=0.9;
7.93	10.24286745	profit=0.9; loss=1.2;
7.93	10.19108410	profit=0.9; loss=0.9;
7.93	10.17335297	profit=0.8; loss=1.1;
7.93	10.17335297	profit=0.8; loss=0.9;
7.93	10.17335297	profit=0.8; loss=0.7;

※「損益」最大値＝5840.33、その時の「OnTester」値＝10.29

C.「OnTester」で値の大きい方から表示；「OnTester」部をクリックし、昇順・降順を切替



パス	損益	総取引数	プロフィットファクター	期待利得	ドローダウン $	ドローダウン %	OnTester ...	パラメーターの入力
5937	2529.34	9	2023.50	281.04	589.15	5.57	2022.50375820	profit=0.4; loss=0.6; profit_2=0.4; loss_2=0.7; period_bollin=20; IKA=1.1; IZYOU=0.5...
5858	2529.34	9	2023.50	281.04	589.15	5.57	2022.50375820	profit=0.4; loss=0.6; profit_2=0.4; loss_2=0.6; period_bollin=20; IKA=1.1; IZYOU=0.5...
5593	2529.34	9	2023.50	281.04	589.15	5.57	2022.50375820	profit=0.4; loss=0.6; profit_2=0.4; loss_2=0.8; period_bollin=20; IKA=1.1; IZYOU=0.5...
5456	2529.34	9	2023.50	281.04	589.15	5.57	2022.50375820	profit=0.4; loss=0.6; profit_2=0.4; loss_2=0.7; period_bollin=20; IKA=1.1; IZYOU=0.5...
4742	2529.34	9	2023.50	281.04	589.15	5.57	2022.50375820	profit=0.4; loss=0.6; profit_2=0.4; loss_2=0.8; period_bollin=20; IKA=1.1; IZYOU=0.5...
2192	2529.34	9	2023.50	281.04	589.15	5.57	2022.50375820	profit=0.4; loss=0.6; profit_2=0.4; loss_2=0.6; period_bollin=20; IKA=1.1; IZYOU=0.5...
5942	2221.60	8	1777.43	277.70	676.79	6.58	1776.42875420	profit=0.5; loss=0.6; profit_2=0.4; loss_2=0.7; period_bollin=20; IKA=1.1; IZYOU=0.5...
5936	2221.60	8	1777.43	277.70	676.79	6.58	1776.42875420	profit=0.4; loss=0.6; profit_2=0.4; loss_2=0.6; period_bollin=20; IKA=1.1; IZYOU=0.5...

テスター　セッティング　最適化結果　最適化グラフ　操作履歴

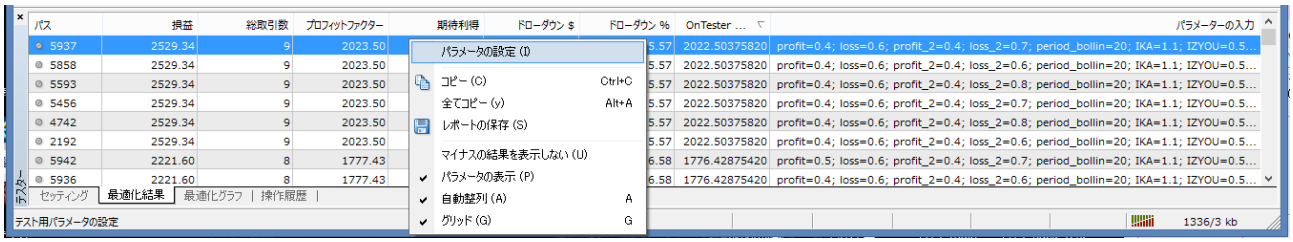
パス	損益	総取引数	プロフィットファクター	ドローダウン %	OnTester ...	
5937	2529.34	9	2023.50	5.57	2022.50375820	profit=0.4; loss=0.6; pro
5858	2529.34	9	2023.50	5.57	2022.50375820	profit=0.4; loss=0.6; pro
5593	2529.34	9	2023.50	5.57	2022.50375820	profit=0.4; loss=0.6; pro
5456	2529.34	9	2023.50	5.57	2022.50375820	profit=0.4; loss=0.6; pro
4742	2529.34	9	2023.50	5.57	2022.50375820	profit=0.4; loss=0.6; pro
2192	2529.34	9	2023.50	5.57	2022.50375820	profit=0.4; loss=0.6; pro
5942	2221.60	8	1777.43	6.58	1776.42875420	profit=0.5; loss=0.6; pro
5936	2221.60	8	1777.43	6.58	1776.42875420	profit=0.4; loss=0.6; pro

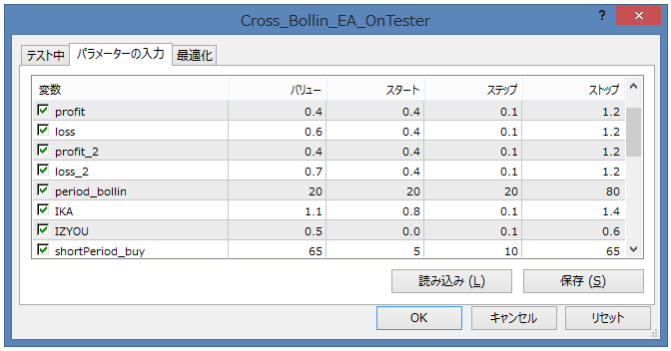
セッティング | 最適化結果 | 最適化グラフ | 操作履歴

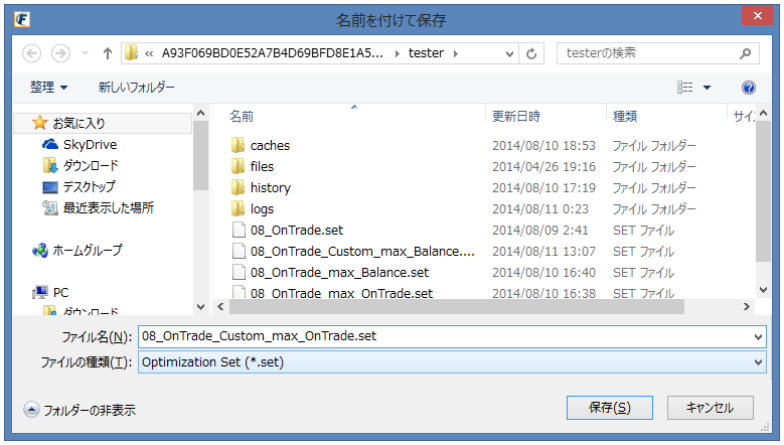
※「OnTester」最大値＝2022.50、その時の「損益」＝2529.34

－５．「Custom 値」の最適化結果を生み出す「exter 値」組合せを保存しておきます

・[OnTrade 結果（Custom 値）]が最大になるパラメータをセットし、[保存]







※ファイル名；
08_OnTrade_max_Custom.set
として「Tester フォルダ」に保存。

（2）「最適化パラメータ＝Balance」として実行した場合

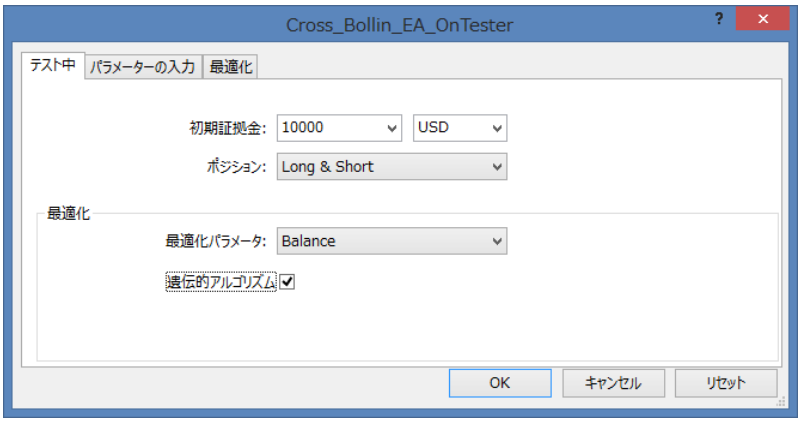
※「最適化パラメータ＝Custom」と「最適化パラメータ＝Balance」では、最適化後に得られるデータの範囲が異なります。

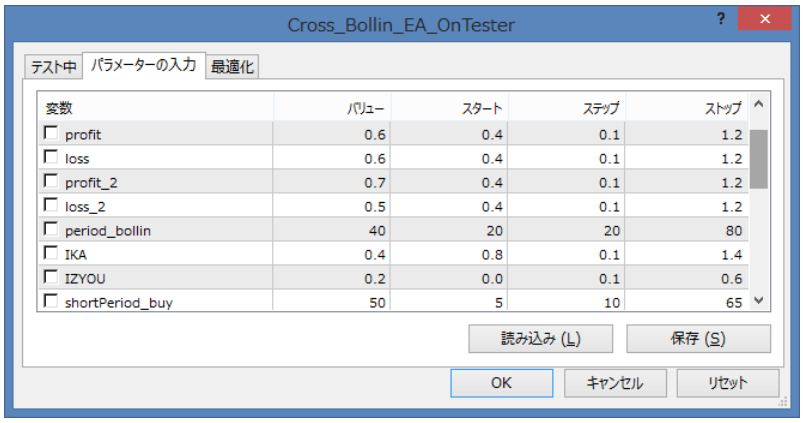
本稿では、「最適化パラメータ＝Custom」の場合との比較のために実行しました。

－１．ＥＡ、通貨ペア、モデルを設定する

「Cross_Bollin_EA_Ontester」、「USDJPYFXF」、「全ティック

－２．［エキスパート設定］中の［最適化パラメータ］と［パラメータの入力］

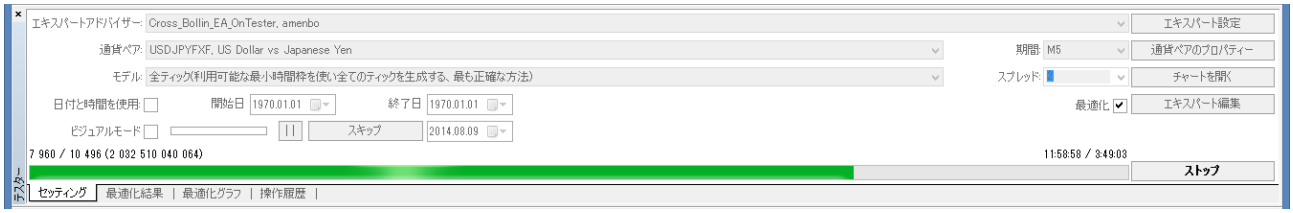




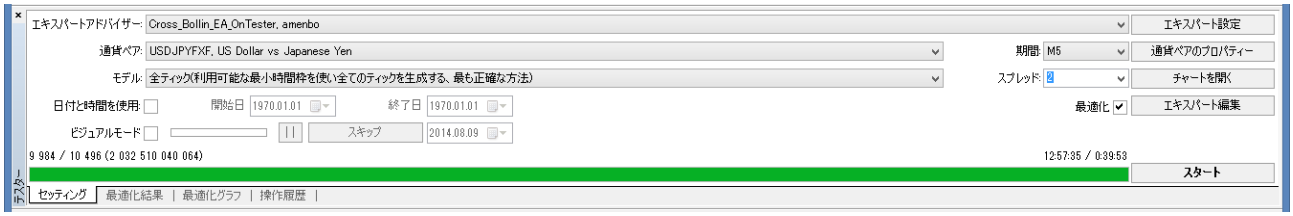
※時間短縮のため、
［遺伝滝アルゴリズム］を使います。

－３．ＥＡの最適化をスタートする

↓



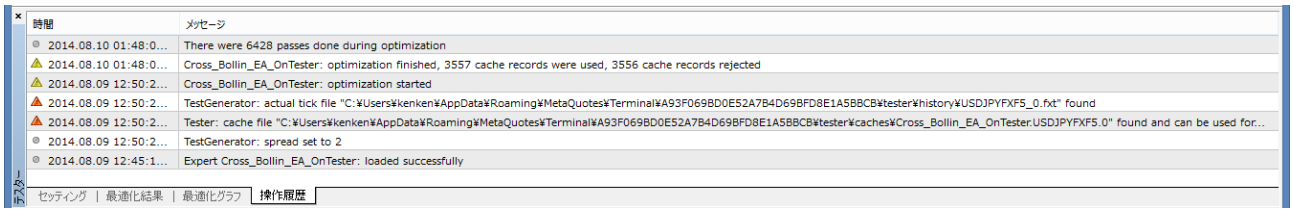
↓



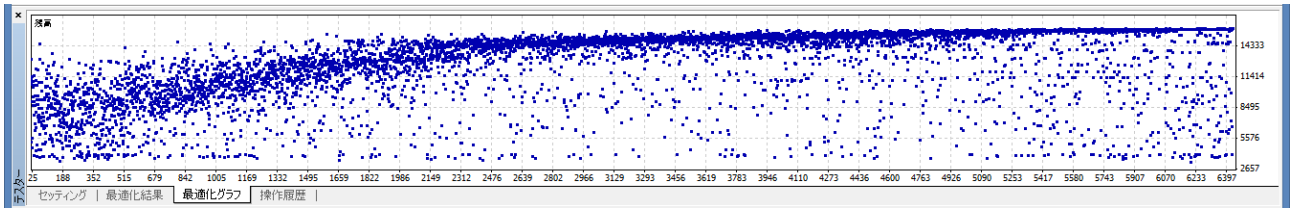
※終了後に、詳細を調べます。

－４．最適化結果を調べる

［動作履歴］

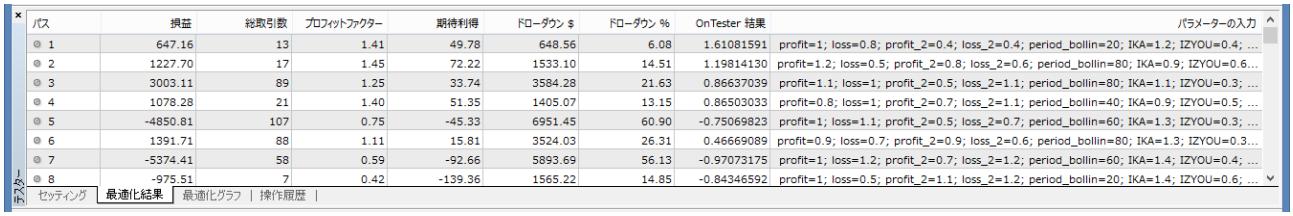


［最適化グラフ］



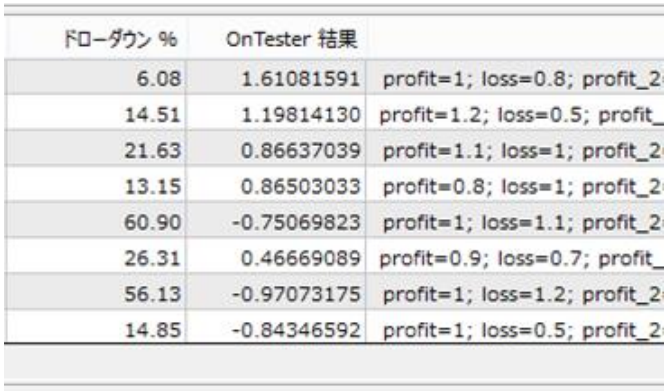
［最適化結果］

A．「パス」順で表示、「OnTester 結果」部を拡大表示します



パス	損益	総取引数	プロフィットファクター	期待利得	ドローダウン $	ドローダウン %	OnTester 結果	パラメーターの入力
1	647.16	13	1.41	49.78	648.56	6.08	1.61081591	profit=1; loss=0.8; profit_2=0.4; loss_2=0.4; period_bollin=20; IKA=1.2; IZYOU=0.4; ...
2	1227.70	17	1.45	72.22	1533.10	14.51	1.19814130	profit=1.2; loss=0.5; profit_2=0.8; loss_2=0.6; period_bollin=80; IKA=0.9; IZYOU=0.6...
3	3003.11	89	1.25	33.74	3584.28	21.63	0.86637039	profit=1.1; loss=1; profit_2=0.5; loss_2=1.1; period_bollin=80; IKA=1.1; IZYOU=0.3; ...
4	1078.28	21	1.40	51.35	1405.07	13.15	0.86503033	profit=0.8; loss=1; profit_2=0.7; loss_2=1.1; period_bollin=40; IKA=0.9; IZYOU=0.5; ...
5	-4850.81	107	0.75	-45.33	6951.45	60.90	-0.75069823	profit=1; loss=1.1; profit_2=0.5; loss_2=0.7; period_bollin=60; IKA=1.3; IZYOU=0.3; ...
6	1391.71	88	1.11	15.81	3524.03	26.31	0.46669089	profit=0.9; loss=0.7; profit_2=0.9; loss_2=0.6; period_bollin=80; IKA=1.3; IZYOU=0.3...
7	-5374.41	58	0.59	-92.66	5893.69	56.13	-0.97073175	profit=1; loss=1.2; profit_2=0.7; loss_2=1.2; period_bollin=60; IKA=1.4; IZYOU=0.4; ...
8	-975.51	7	0.42	-139.36	1565.22	14.85	-0.84346592	profit=1; loss=0.5; profit_2=1.1; loss_2=1.2; period_bollin=20; IKA=1.4; IZYOU=0.6; ...

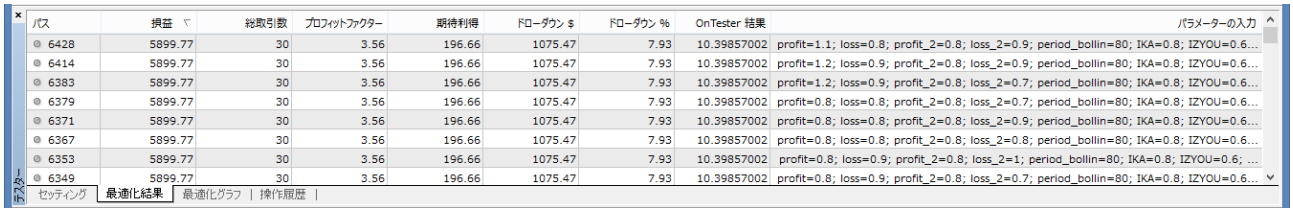
セッティング | 最適化結果 | 最適化グラフ | 操作履歴



ドローダウン %	OnTester 結果	
6.08	1.61081591	profit=1; loss=0.8; profit_2
14.51	1.19814130	profit=1.2; loss=0.5; profit_
21.63	0.86637039	profit=1.1; loss=1; profit_2
13.15	0.86503033	profit=0.8; loss=1; profit_2
60.90	-0.75069823	profit=1; loss=1.1; profit_2
26.31	0.46669089	profit=0.9; loss=0.7; profit_
56.13	-0.97073175	profit=1; loss=1.2; profit_2
14.85	-0.84346592	profit=1; loss=0.5; profit_2

※「OnTester 結果」が追加で表示されます。

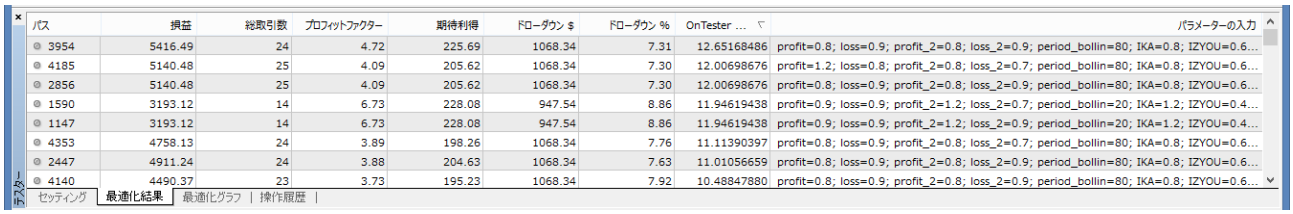
B．「損益」順で収益の大きい方から表示；「損益」部をクリックし、昇順・降順を切替える



パス	損益	総取引数	プロフィットファクター	期待利得	ドローダウン $	ドローダウン %	OnTester 結果	パラメーターの入力
6428	5899.77	30	3.56	196.66	1075.47	7.93	10.39857002	profit=1.1; loss=0.8; profit_2=0.8; loss_2=0.9; period_bollin=80; IKA=0.8; IZYOU=0.6...
6414	5899.77	30	3.56	196.66	1075.47	7.93	10.39857002	profit=1.2; loss=0.9; profit_2=0.8; loss_2=0.9; period_bollin=80; IKA=0.8; IZYOU=0.6...
6383	5899.77	30	3.56	196.66	1075.47	7.93	10.39857002	profit=1.2; loss=0.9; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...
6379	5899.77	30	3.56	196.66	1075.47	7.93	10.39857002	profit=0.8; loss=0.8; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...
6371	5899.77	30	3.56	196.66	1075.47	7.93	10.39857002	profit=0.8; loss=0.8; profit_2=0.8; loss_2=0.9; period_bollin=80; IKA=0.8; IZYOU=0.6...
6367	5899.77	30	3.56	196.66	1075.47	7.93	10.39857002	profit=0.8; loss=0.8; profit_2=0.8; loss_2=0.8; period_bollin=80; IKA=0.8; IZYOU=0.6...
6353	5899.77	30	3.56	196.66	1075.47	7.93	10.39857002	profit=0.8; loss=0.9; profit_2=0.8; loss_2=1; period_bollin=80; IKA=0.8; IZYOU=0.6; ...
6349	5899.77	30	3.56	196.66	1075.47	7.93	10.39857002	profit=0.8; loss=0.9; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...

テスター　セッティング　最適化結果　最適化グラフ　操作履歴

C．「OnTester」で値の大きい方から表示；「OnTester」部をクリックし、昇順・降順を切替

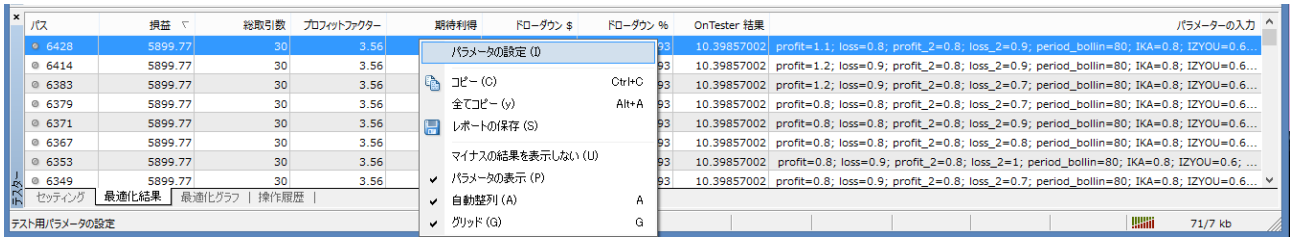


パス	損益	総取引数	プロフィットファクター	期待利得	ドローダウン $	ドローダウン %	OnTester ...	パラメーターの入力
3954	5416.49	24	4.72	225.69	1068.34	7.31	12.65168486	profit=0.8; loss=0.9; profit_2=0.8; loss_2=0.9; period_bollin=80; IKA=0.8; IZYOU=0.6...
4185	5140.48	25	4.09	205.62	1068.34	7.30	12.00698676	profit=1.2; loss=0.8; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...
2856	5140.48	25	4.09	205.62	1068.34	7.30	12.00698676	profit=0.8; loss=0.9; profit_2=0.8; loss_2=0.9; period_bollin=80; IKA=0.8; IZYOU=0.6...
1590	3193.12	14	6.73	228.08	947.54	8.86	11.94619438	profit=0.9; loss=0.9; profit_2=1.2; loss_2=0.7; period_bollin=20; IKA=1.2; IZYOU=0.4...
1147	3193.12	14	6.73	228.08	947.54	8.86	11.94619438	profit=0.9; loss=0.9; profit_2=1.2; loss_2=0.9; period_bollin=20; IKA=1.2; IZYOU=0.4...
4353	4758.13	24	3.89	198.26	1068.34	7.76	11.11390397	profit=0.8; loss=0.9; profit_2=0.8; loss_2=0.7; period_bollin=80; IKA=0.8; IZYOU=0.6...
2447	4911.24	24	3.88	204.63	1068.34	7.63	11.01056659	profit=0.8; loss=0.9; profit_2=0.8; loss_2=0.9; period_bollin=80; IKA=0.8; IZYOU=0.6...
4140	4490.37	23	3.73	195.23	1068.34	7.92	10.48847880	profit=0.8; loss=0.9; profit_2=0.8; loss_2=0.9; period_bollin=80; IKA=0.8; IZYOU=0.6...

テスター　セッティング　最適化結果　最適化グラフ　操作履歴

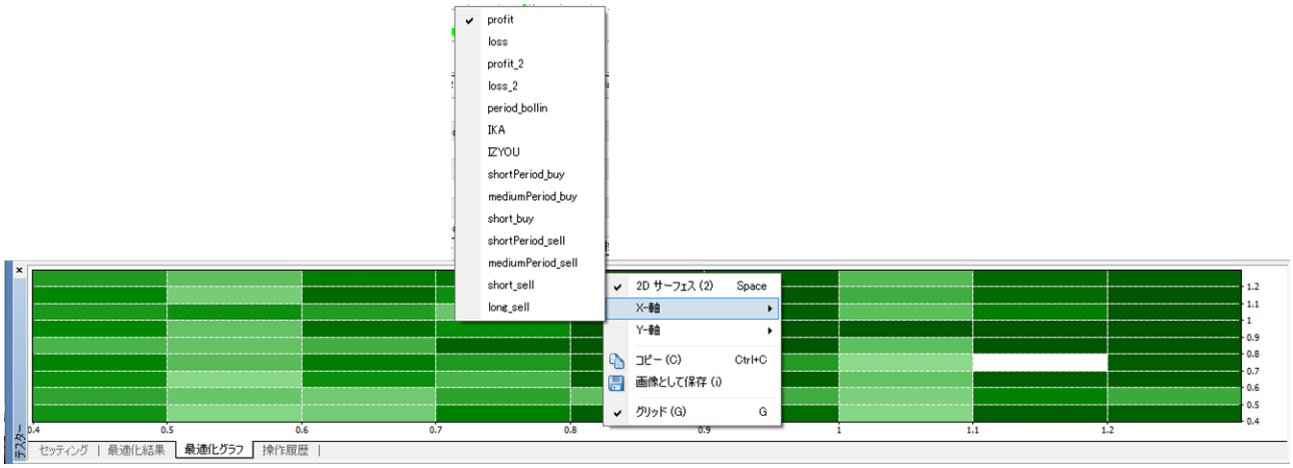
－５．「Balance 値」の最適化結果を生み出す「exter 値」組合せを保存しておきます

・Balance が最大になるパラメータをセットし、［保存］



※ファイル名；「08_OnTrade_max_Balance.set」として「Tester フォルダ」に保存。

－６．補足；　［最適化グラフ］の２次元表示例を示しておきます



（３）まとめ

※最適化パメータを、「Custom」と「Balance」にした場合の比較したものをまとめてみます

－１．結果を簡単にまとめておきます

最適化パラメータ	表示（昇順・降順）	Balance 値	Cutom 値
Custom	［損益］最大	5840.33	10.294
	［OnTester 結果］最大	2529.34	2022.504
Balance	［損益］最大	5899.77	10.399
	［OnTester 結果］最大	5416.49	12.652

※結果；

・Cutom 値＝［利益／最大ドローダウン］なので、
最適化パラメータが「Cutom」値でも、「Balance（残高）」値でも、
「［損益］最大」の Custom 値と Balance 値には大差が出なかった。

・一方、「［OnTester 結果］最大」の Custome 値には、大きな差が発生。
Custom 値が最大（2022.504）になる時の、Balance 値は意外なほど小さい値となった。
もし、実際にこれらを実トレードに適用する場合は、Custom 値と Balance 値のバランスをとる必要があると考える。
ただ、Custom 値が大きくなる「extern パラメータの組合せ」を採用すれば良い、と言うものではないのは当然です。

＜注意＞

本稿で実施した「最適化」は、2014.08.17 時点で MT4 に表示されていた
最新の USDJPYFXF 為替データに対して実施したものです。
（異なる時点でのヒストリカル・データを使用した場合は、異なる結果になります）

－２．バックテストをしてみる

※次に最適化パラメータを「Custom」と「Balance」した場合の最適化結果を、それぞれ「extern パラメータ」に設定し、バックテストを実施した時の内容を確認します。

※バックテスト時のコードには、次の修正を加えておきます；

```
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//---
   //Print("Ontester() が呼ばれた！");

  double profit_On=TesterStatistics(STAT_PROFIT);
  double max_dd    = TesterStatistics(STAT_BALANCE_DD);

  Print("TesterStatistics(STAT_PROFIT)=",DoubleToString(profit_On));

  if(max_dd>0)
  {
  double rec_factor       =profit_On/max_dd;
  Print("rec_factor=",DoubleToString(rec_factor));
        return(rec_factor);

  }
  else return(0.0);

  }
//+------------------------------------------------------------------+
```

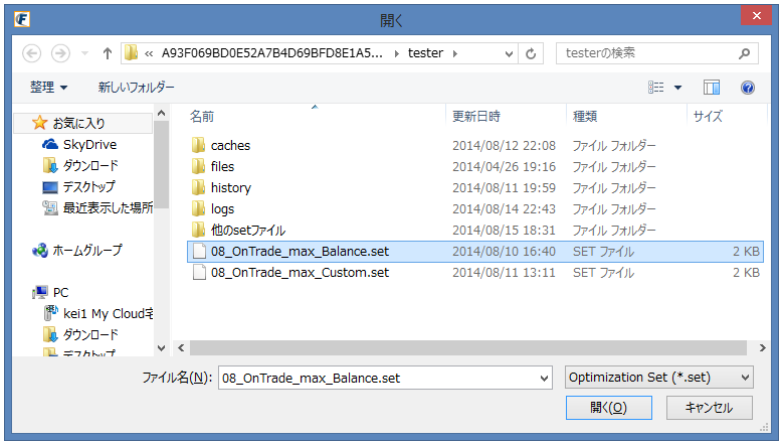
※ポイント；「青書」部は、バックテスト時に動作させる部分（最適化中はコメントアウト）
　バックテスト時のみ、下記のコメント「//」を外して結果をプリントする。

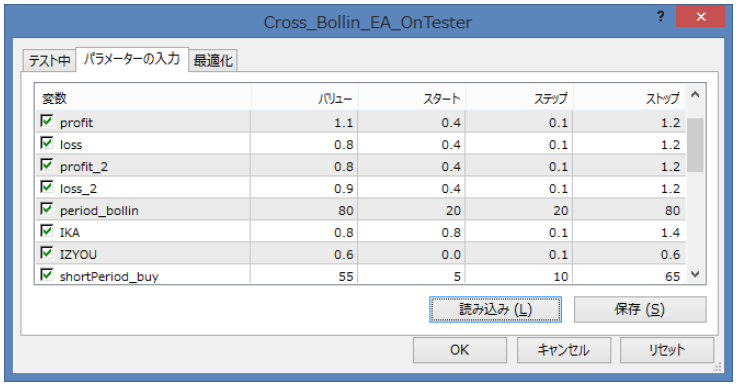
```
//Print("TesterStatistics(STAT_PROFIT)=",DoubleToString(profit_On));
//Print("rec_factor=",DoubleToString(rec_factor));
```

－２．最適化パラメータ対象が「Custom」と「Balance」の場合での収益曲線を比較する

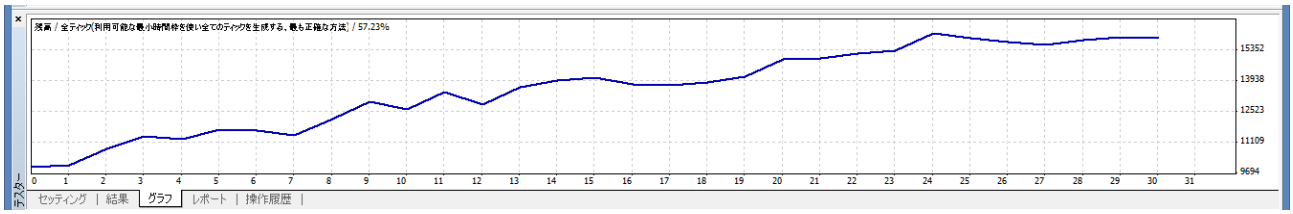
extern パラメータの「.set」ファイルを読み込んで、バックテストを行い、
それぞれの場合を比較する

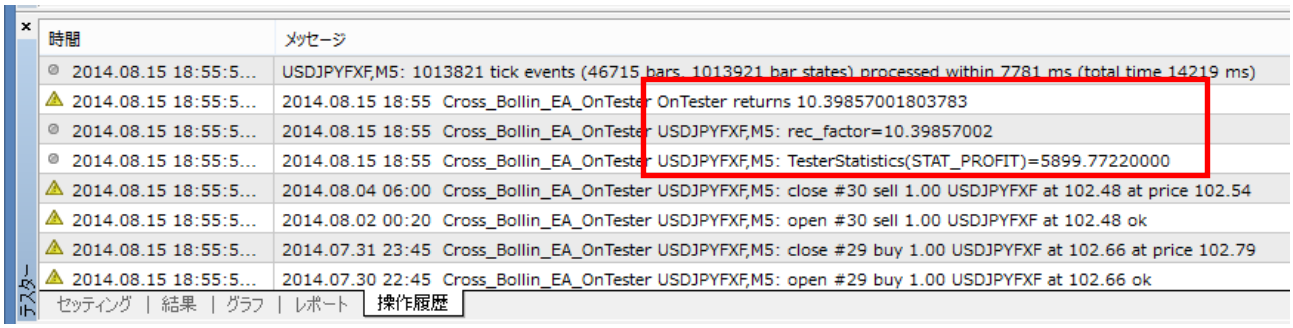
例；「08_OnTrade_max_Balance.set」の読み込み



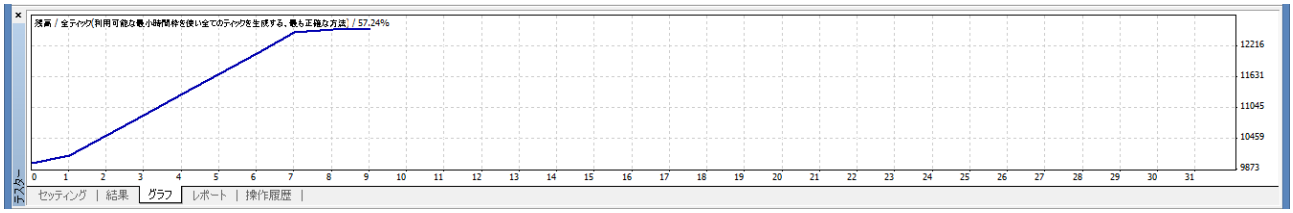


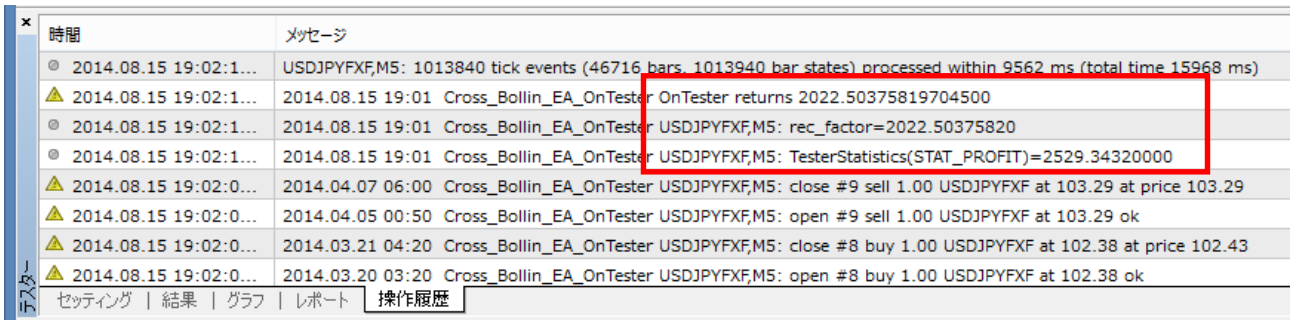
「08_OnTrade_max_Balance.set」をセットした場合；





「08_OnTrade_max_Custom.set」をセットした場合；





＜注意＞

本稿で実施した「バックテスト」は、2014.08.17 時点で MT4 に表示されていた
最新の USDJPYFXF 為替データに対して実施したものです。
（異なる時点でのヒストリカル・データを使用した場合は、異なる結果になります）

５．「Cross_Bollin_EA_OnTester.mq4」のコード

・本稿で試したＥＡのコードを掲載します、

本コードは 2012 年５月に作成・投稿したものに、「OnTester()」を継ぎ足して作成しています。

※ただし、ＥＡの本体コード中で「//Build_610 以降での warning を消す処理」と追記した部分のみ変更しています、また「#property strict」は記載しないことに注意。

詳細コード；

```
//+------------------------------------------------------------------+
//|                                    Cross_Bollin_EA_OnTester.mq4 |
//|                                                          amenbo |
//|                                                泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"
#property version   "1.00"
//+------------------------------------------------------------------+
//|                                                 Bollin_EA_08.mq4 |
//|                                                2012.05.25 amenbo |
//+------------------------------------------------------------------+
#define Magic_ID 1930

extern double Lots=1;
extern int max_position=1;//最大保有ポジション数
// 損益設定
//「①②」は最適化の実施後での設定値
extern double profit=0.60;
extern double loss=0.60;
extern double profit_2=0.70;
extern double loss_2=0.50;
// ①Bollin の幅
extern int period_bollin=40;
extern double IKA=0.4;
extern double IZYOU=0.2;
// ②フィルター設定
extern int shortPeriod_buy=50;
extern int mediumPeriod_buy=120;
extern double short_buy=-0.0044;
extern double long_buy=-0.0006;
//    ・・sell 専用
extern int shortPeriod_sell=35;
extern int mediumPeriod_sell=180;
extern double short_sell=0.0044;
extern double long_sell=0.0006;
//-----------------------------
//  ③トレンド判断
extern int trendPeriod=300;//70
extern double _up=-0.00024;//-0.0004
extern double _down=0.00024;
extern double div_=0.04;
//====================================
static datetime lastbar;
datetime in_time,out_time;
int barsTotal;
//売買シグナル判定用の配列データ
double EMA_[10];
double Bol_hi[10];
double Bol_lo[10];
double Price_01[10000];
double Trend_[10000];
//////////////////////////
int init()
{
```

```
   //
   lastbar=Time[1];
   barsTotal = Bars;
   in_time=Time[20];
   //
   return(0);
}
int deinit()
{
   return(0);
}
////////////////////////////////////
int start()
{
   if(Bars<100 || IsTradeAllowed()==false) return(0);
//  NewBar かチェック
   if(IsNewBar() && (Bars>barsTotal))
   {

      barsTotal=Bars;

      //＜ポジションが在る場合の処理＞
      if(OrdersTotal()>=1)
      {
         for(int i=0;i<OrdersTotal();i++)
         {
            if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES)==false) break;
            if(OrderMagicNumber()!=Magic_ID || OrderSymbol()!=Symbol()) continue;

            if(OrderMagicNumber()==Magic_ID && OrderSymbol()==Symbol())
            {
               bool ikisugi=(Time[0]-in_time)>=(300*Period()*60);

               if(OrderType()==OP_BUY)
               {
                  double kachi_buy_price=OrderOpenPrice()+profit;
                  double make_buy_price=OrderOpenPrice()-loss;
                  //
                  bool kachi_buy=(Bid>=kachi_buy_price);
                  bool make_buy=(Bid<=make_buy_price);
                  //
                  if(kachi_buy || make_buy || ikisugi)
                  {
                     //OrderClose(OrderTicket(),Lots,Bid,0,Blue);//Build_509 以前はこれでＯＫ
         bool c1=OrderClose(OrderTicket(),Lots,Bid,0,Blue);//Build_610 以降での warning を消す処理
                     continue;
                  }
               }

               if(OrderType()==OP_SELL)
               {
                  double kachi_sell_price=OrderOpenPrice()-profit;
                  double make_sell_price=OrderOpenPrice()+loss;
                  //
                  bool kachi_sell=(Ask<=kachi_sell_price);
                  bool make_sell=(Ask>=make_sell_price);
                  //
                  if(kachi_sell || make_sell || ikisugi)
                  {
                     //OrderClose(OrderTicket(),Lots,Ask,0,Blue);
                     bool c2=OrderClose(OrderTicket(),Lots,Ask,0,Blue);
                     continue;
                  }

               }
```

```
         }
      }
   }//end_of_if(OrdersTotal()>=1)

   //＜新規ポジションの設定と方針＞
   //アップトレンドでは⇒buyIN のみ実施（順張り）
   //ダウントレンドでは⇒sellIN のみ実施（順張り）
   //レンジ相場では　　⇒buy と sell（逆張り？―
   //if(OrdersTotal()<=max_position)
   if(OrdersTotal()==0)
   {

      //判断用の配列を準備
      ArraySetAsSeries(Bol_hi,true);
      ArraySetAsSeries(Bol_lo,true);
      //
      for(int j=0;j<=6;j++)
      {
         Bol_hi[j]=iBands(NULL,0,period_bollin,2,0,PRICE_OPEN,MODE_UPPER,j);  //2_σ
         Bol_lo[j]=iBands(NULL,0,period_bollin,2,0,PRICE_OPEN,MODE_LOWER,j);  //2_σ
      }

      //判断用の共通指標を準備
      bool Subtract=((IZYOU<(Bol_hi[0]-Bol_lo[0])) && ((Bol_hi[0]-Bol_lo[0])<IKA));
      bool hanareta=((Time[0]-in_time)>=(19*Period()*60));

    bool TREND_UP=((slope_kaiki(0,trendPeriod))<=(_up)) && (call_hensa(0,trendPeriod)<=div_);
  bool TREND_DOWN=((slope_kaiki(0,trendPeriod))>=(_down)) && (call_hensa(0,trendPeriod)<=div_);

      ////if(TREND_UP)
      ////{
         //USDJPY の上昇相場（円安）
         //買い条件は整ったか
         bool cross_up_1=((High[1]>Bol_hi[1]) && (Open[0]>Bol_hi[0]));
         bool cross_up_2=((Open[3]<Bol_hi[3]) && (Open[2]<Bol_hi[2]));
         bool cross_up_3=((Open[4]<Bol_hi[4]) && (Open[3]<Bol_hi[3]));
         bool cross_up_4=((Open[5]<Bol_hi[5]) && (Open[4]<Bol_hi[4]));
         bool cross_up_5=((Open[6]<Bol_hi[6]) && (Open[5]<Bol_hi[5]));
         bool cross_up=(cross_up_1 && (cross_up_2 || cross_up_3 || cross_up_4 || cross_up_5));
         //条件の補足
    bool cross_up_rapid=((High[2]>Bol_hi[2]) && (High[1]>Bol_hi[1]) && (Open[0]>Bol_hi[0]));
         //Filter；本当に買いか
         double slope_s_buy=slope_kaiki(0,shortPeriod_buy);
         double slope_m_buy=slope_kaiki(0,mediumPeriod_buy);
         bool SLOPE_BUY=((slope_s_buy<=short_buy) && (slope_m_buy<=long_buy));
         //
         if(((cross_up || cross_up_rapid) && SLOPE_BUY) && Subtract && hanareta)
         {
            //OrderSend(Symbol(),OP_BUY,Lots,Ask,0,0,0,"my_Buy_order",Magic_ID,Green);
        int ticket1=OrderSend(Symbol(),OP_BUY,Lots,Ask,0,0,0,"my_Buy_order",Magic_ID,Green);
            in_time=Time[0];
         }

      ////} else if(TREND_DOWN)
      ////{
      ///+
         //USDJPY の下降相場（円高）
         //売り条件は整ったか
         bool cross_down_1=((Low[1]<Bol_lo[1]) && (Open[0]<Bol_lo[0]));
         bool cross_down_2=((Open[3]>Bol_lo[3]) && (Open[2]>Bol_lo[2]));
         bool cross_down_3=((Open[4]>Bol_lo[4]) && (Open[3]>Bol_lo[3]));
         bool cross_down_4=((Open[5]>Bol_lo[5]) && (Open[4]>Bol_lo[4]));
         bool cross_down_5=((Open[6]>Bol_lo[6]) && (Open[5]>Bol_lo[5]));
bool cross_down=(cross_down_1 && (cross_down_2 || cross_down_3 || cross_down_4 || cross_down_5));
         //条件の補足
```

```
      bool cross_down_rapid=((Low[2]<Bol_lo[2]) && (Low[1]<Bol_lo[1]) && (Open[0]<Bol_lo[0]));
          //Filter；本当に売りか
          double slope_s_sell=slope_kaiki(0,shortPeriod_sell);
          double slope_m_sell=slope_kaiki(0,mediumPeriod_sell);
          bool SLOPE_SELL=(slope_s_sell>=short_sell) && (slope_m_sell>=long_sell);
          //
          if(((cross_down || cross_down_rapid) && SLOPE_SELL) && Subtract && hanareta)
          {
             //OrderSend(Symbol(),OP_SELL,Lots,Bid,0,0,0,"my_Sell_order",Magic_ID,Red);
            int ticket2=OrderSend(Symbol(),OP_SELL,Lots,Bid,0,0,0,"my_Sell_order",Magic_ID,Red);
             in_time=Time[0];
          }
       //*/
       ////} else if(TREND_UP==false && TREND_DOWN==false)
       ////{
          //レンジ相場
          //内容は検討中です
       ////}

   }//end_of_if(OrdersTotal()<=max_position)


} else
{
   //［５分足］内で起こる急変に対応する処理を記載する
   //  バックテストでは確認が困難な部分

   if(OrdersTotal()>=1)
   {
      for(int ii=0;ii<OrdersTotal();ii++)
      {
         if(OrderSelect(ii,SELECT_BY_POS,MODE_TRADES)==false) break;
         if(OrderMagicNumber()!=Magic_ID || OrderSymbol()!=Symbol()) continue;

         if(OrderMagicNumber()==Magic_ID && OrderSymbol()==Symbol())
         {

            if(OrderType()==OP_BUY)
            {
               double i_kachi_buy_price=OrderOpenPrice()+profit_2;
               double i_make_buy_price=OrderOpenPrice()-loss_2;
               //
               bool i_kachi_buy=(Bid>=i_kachi_buy_price);
               bool i_make_buy=(Bid<=i_make_buy_price);
               //
               if(i_kachi_buy || i_make_buy)
               {
                  //OrderClose(OrderTicket(),Lots,Bid,0,Blue);
                  bool c3=OrderClose(OrderTicket(),Lots,Bid,0,Blue);
                  continue;
               }
            }

            if(OrderType()==OP_SELL)
            {
               double i_kachi_sell_price=OrderOpenPrice()-profit_2;
               double i_make_sell_price=OrderOpenPrice()+loss_2;
               //
               bool i_kachi_sell=(Ask<=i_kachi_sell_price);
               bool i_make_sell=(Ask>=i_make_sell_price);
               //
               if(i_kachi_sell || i_make_sell)
               {
                  //OrderClose(OrderTicket(),Lots,Ask,0,Blue);
                  bool c4=OrderClose(OrderTicket(),Lots,Ask,0,Blue);
```

```
                     continue;
                  }

               }
            }
         }
      }//end_of_if(OrdersTotal()>=1)

   }

   return(0);
}
//////////////////////  以下は関数類  ////////////////////////////////////
bool IsNewBar()
{
datetime curbar = Time[0]; // Open time of current bar
if (lastbar!=curbar)
   {
    lastbar=curbar;
    return (true);
   }
  return(false);
}
//---------------------------------------
double slope_kaiki(int start_s,int period_s)
{
   ArraySetAsSeries(Price_01,true);
   Price_01[0]=Open[0];
   for(int j_=(start_s+1);j_<=((start_s+1)+(2*period_s));j_++)
   {
         Price_01[j_]=(1.0/4.0)*(Open[j_]+High[j_]+Low[j_]+Close[j_]);
   }
   double slope=kaiki_sen(Price_01,start_s,period_s,0);
   //
   return(slope);
}
//
double call_hensa(int start_h,int period_h)
{
   double koubai_M=kaiki_sen(Price_01,start_h,period_h,0);
   double seppen_M=kaiki_sen(Price_01,start_h,period_h,1);
   //
   double ooM=0;
   double sigma=0;
   for(int pM=start_h;pM<=(start_h+period_h);pM++)
   {
      Trend_[pM]=koubai_M*ooM+seppen_M;
      ooM=ooM+1;
      //
      div_=(Price_01[pM]-Trend_[pM])*(Price_01[pM]-Trend_[pM]);
      sigma=sigma+div_;
   }
   double div=sigma/100;//100 足あたりの値
   //Print 内容は、log ファイルにも記録されるので、バックテスト時に異常解析の邪魔になる
  // Print("勾配は＝   ",koubai_M);
  // Print("回帰線からの偏差＝  ",div);
  // Comment("勾配は＝ ",koubai_M,":回帰線からの偏差＝  ",div);
//---
   return(div);
}
//---------------------------------------
//回帰直線の傾き、切辺を返す
double kaiki_sen(double& price_[],int start_,int period_,int what_)
{
   double X_av=0.0,X_i=0.0;
```

```
   double Y_av=0.0,Y_i=0.0;
   double i_D;
   //
   ArraySetAsSeries(price_,true);
   //まず、平均値を求める
   for(int i=start_;i<=(start_+period_);i++)
   {
      i_D=i*1.0;
      X_i=X_i+i_D;
      Y_i=Y_i+price_[i];
   }
      X_av=(X_i/(period_+1));
      //X_av=(start_+(start_+period_))/period_;
      Y_av=(Y_i/(period_+1));
   //加算
   double bunshi=0.0,bunbo=0.0;
   double koubai_=0.0,seppen_=0.0;
   double j_D;
   //
   for(int j=start_;j<=(start_+period_);j++)
   {
      j_D=j*1.0;
      bunshi=bunshi+(j_D-X_av)*(price_[j]-Y_av);
      bunbo=bunbo+(j_D-X_av)*(j_D-X_av);
   }
      koubai_=bunshi/bunbo;
      //////seppen_=Y_av-(koubai_*X_av);//良い値が出ない
      seppen_=price_[start_];
   //
   if(what_==0) return(koubai_);
   if(what_==1) return(seppen_);
   //
   return (0);
}
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//---
   //Print("Ontester() が呼ばれた！");

         double profit_On = TesterStatistics(STAT_PROFIT);
         double max_dd    = TesterStatistics(STAT_BALANCE_DD);

         //Print("TesterStatistics(STAT_PROFIT)=",DoubleToString(profit_On));

         if(max_dd>0)
         {
         double rec_factor = profit_On/max_dd;
         //Print("rec_factor=",DoubleToString(rec_factor));
               return(rec_factor);

         }
         else return(0.0);

  }
//+------------------------------------------------------------------+
```

※「青書」部が、「OnTester()」として単純に追加した部分です。

※「//Print("******」部は、バックテストを行うときにのみ、コメントアウトを外す。

以　上