

Specifications for the Trading Algorithm "Nacer9"

1. Introduction

The Nacer9 trading strategy is an automated strategy on the NinjaTrader platform, using a combination of technical indicators to determine entry and exit points for positions. It relies on an analysis of the ATR (Average True Range), exponential moving averages (EMA) and MACD for dynamic position management.

2. Objectives

The main objective of this strategy is to take long or short positions based on the interaction of several technical indicators, with dynamic stop-loss management based on ATR. Positions are also closed at a set time.

3. Strategy settings

- **MACD parameters:**
 - MACDFastPeriod: Period of the MACD fast line.
 - MACDSlowPeriod: Period of the MACD slow line.
 - MACDSignalPeriod: Period of the MACD signal line.
- **Strategy settings:**
 - StartTime: Start time of operations (eg. 11:00).
 - EndTime: End time of operations (eg. 18:00).
 - CloseTime: Time at which positions should be closed (eg. 21:30).
 - ATRPeriode: Period used to calculate the ATR.
 - ATRMulti: Multiplier applied to the ATR value to adjust the stop level.
 - SL: Value in stop-loss points (in currency).

4. Conditions for entry into position

- **Long Position (Buy): The strategy enters a long position if all of the following conditions are met:**
 - $EMA1 < EMA2$ (short EMA below long EMA).
 - $EMA2 < EMA3$ (intermediate EMA below the longest).
 - The opening price is below the longest EMA (EMA3).
 - Price reaches a maximum above the longest EMA.
 - The time is between StartTime and EndTime.
 - The MACD is in negative territory ($MACD1.Avg[0] < 0$).
- **Short Position (Sell): The strategy enters a short position if all of the following conditions are met:**
 - $EMA1 > EMA2$ (short EMA above long EMA).
 - $EMA2 > EMA3$ (intermediate EMA above the longest).
 - The opening price is above the longest EMA (EMA3).
 - Price reaches a low below the longest EMA.
 - The time is between StartTime and EndTime.
 - The MACD is in positive territory ($MACD1.Avg[0] > 0$).

5. Dynamic stop-loss management

- **For long positions:**
 - The initial stop-loss level is set as $Position.AveragePrice - ATRValue$.
 - This level is dynamically adjusted based on price movement. If the price increases more than one ATR above the entry price, the stop-loss is moved to $Close[0] - ATRValue$.
- **For short positions:**
 - The initial stop-loss level is defined as $Position.AveragePrice + ATRValue$.
 - This level is dynamically adjusted based on price movement. If the price drops more than one ATR below the entry price, the stop-loss is moved to $Close[0] + ATRValue$.

6. Exit from positions

- **Exit Long Positions:** The long position is closed at the time specified by CloseTime.
- **Exit Short Positions:** The short position is closed at the time specified by CloseTime.

7. Risk management

The stop-loss is defined in terms of monetary value (in the market currency) and is calculated based on the ATR and the ATRMulti multiplier. The maximum number of bars needed to trade is 20, which ensures that the strategy only takes positions when enough data is available.

8. Market conditions

The strategy works with market data in the form of 15-minute time series and takes into account the market periods specified by StartTime and EndTime. It also makes sure to close all positions before the end of the trading session via CloseTime.

9. Summary of the rules of the strategy

Type of Trade	Entry requirements	Exit conditions
Position Longue	- EMA1 < EMA2 < EMA3- Open under EMA3- Negative MACD- Time between StartTime and EndTime	- Exit at Close Time
Short Position	- EMA1 > EMA2 > EMA3- Open above EMA3- Positive MACD- Time between StartTime and EndTime	- Exit at Close Time

10. Performance and monitoring

The strategy uses visual indicators (EMA and MACD) on the charts to facilitate real-time monitoring. Stops are adjusted dynamically, and orders are managed via NinjaTrader with real-time error handling.

This specification describes the strategy's features and parameters, explaining possible entries and exits as well as risk and stop-loss management.

11. Le code

```
namespace NinjaTrader.NinjaScript.Strategies
```

```
{  
  
    public class Nacer9 : Strategy  
    {  
  
        private double ATRValue;  
  
        private double LongStopLevel; // Niveau du stop pour position longue  
  
        private double ShortStopLevel; // Niveau du stop pour position courte  
  
  
        private EMA EMA1;  
  
        private EMA EMA2;  
  
        private EMA EMA3;  
  
        private MACD MACD1;
```

```
#region MACD Parameters
```

```
[NinjaScriptProperty]
```

```
[Range(1, int.MaxValue)]
```

```
[Display(Name = "MACD Période Rapide", Order = 7, GroupName = "Parameters")]
```

```
public int MACDFastPeriod { get; set; }
```

```
[NinjaScriptProperty]
```

```
[Range(1, int.MaxValue)]
```

```
[Display(Name = "MACD Période Lente", Order = 8, GroupName = "Parameters")]
```

```
public int MACDSlowPeriod { get; set; }
```

```
[NinjaScriptProperty]
```

```
[Range(1, int.MaxValue)]
```

```
[Display(Name = "MACD Période Signal", Order = 9, GroupName = "Parameters")]
```

```
public int MACDSignalPeriod { get; set; }
```

```
#endregion
```

```
#region Strategy Parameters
```

```
[NinjaScriptProperty]
```

```
[PropertyEditor("NinjaTrader.Gui.Tools.TimeEditorKey")]
```

```
[Display(Name = "HeureDepart", Order = 1, GroupName = "Parameters")]
```

```
public DateTime HeureDepart { get; set; }
```

```
[NinjaScriptProperty]
```

```
[PropertyEditor("NinjaTrader.Gui.Tools.TimeEditorKey")]
```

```
[Display(Name = "HeureFin", Order = 2, GroupName = "Parameters")]
```

```
public DateTime HeureFin { get; set; }
```

```
[NinjaScriptProperty]
```

```
[PropertyEditor("NinjaTrader.Gui.Tools.TimeEditorKey")]
```

```
[Display(Name = "HeureClose", Order = 3, GroupName = "Parameters")]
```

```
public DateTime HeureClose { get; set; }
```

```
[NinjaScriptProperty]
```

```
[Range(1, int.MaxValue)]
```

```
[Display(Name = "ATRPeriode", Order = 4, GroupName = "Parameters")]
```

```
public int ATRPeriode { get; set; }
```

```
[NinjaScriptProperty]
```

```
[Range(0.1, double.MaxValue)]
```

```
[Display(Name = "ATRMulti", Order = 5, GroupName = "Parameters")]
```

```
public double ATRMulti { get; set; }
```

```
[NinjaScriptProperty]
```

```
[Range(1, int.MaxValue)]
```

```
[Display(Name = "SL", Order = 6, GroupName = "Parameters")]
```

```
public int SL { get; set; }
```

```
#endregion
```

```
protected override void OnStateChange()
```

```
{
```

```
    if (State == State.SetDefaults)
```

```
    {
```

```
        Description = @"Entrer la description du nouveau Stratégie ici.";
```

```
        Name = "Nacer9";
```

```
        Calculate = Calculate.OnBarClose;
```

```
        EntriesPerDirection = 1;
```

```
        EntryHandling = EntryHandling.AllEntries;
```

```
        IsExitOnSessionCloseStrategy = true;
```

```
        ExitOnSessionCloseSeconds = 30;
```

```
        IsFillLimitOnTouch = false;
```

```
        MaximumBarsLookBack = MaximumBarsLookBack.TwoHundredFiftySix;
```

```

OrderFillResolution = OrderFillResolution.Standard;

Slippage = 0;

StartBehavior = StartBehavior.WaitUntilFlat;

TimeInForce = TimeInForce.Day;

TraceOrders = true;

RealtimeErrorHandling = RealtimeErrorHandling.StopCancelClose;

StopTargetHandling = StopTargetHandling.PerEntryExecution;

BarsRequiredToTrade = 20;

IsInstantiatedOnEachOptimizationIteration = true;

HeureDepart = DateTime.Parse("11:00", System.Globalization.CultureInfo.InvariantCulture);

HeureFin = DateTime.Parse("18:00", System.Globalization.CultureInfo.InvariantCulture);

HeureClose = DateTime.Parse("21:30", System.Globalization.CultureInfo.InvariantCulture);

ATRPeriode = 14;

ATRMulti = 2.5;

SL = 500;


// Définir les valeurs par défaut des paramètres MACD

MACDFastPeriod = 6;

MACDSlowPeriod = 13;

MACDSignalPeriod = 4;
}

else if (State == State.Configure)
{
    AddDataSeries(Data.BarsPeriodType.Minute, 15); // Série de données de 15 minutes
}

else if (State == State.DataLoaded)
{
    EMA1 = EMA(Close, 800);

    EMA2 = EMA(Close, 200);

    EMA3 = EMA(Close, 50);

```

```
MACD1 = MACD(Close, MACDFastPeriod, MACDSlowPeriod, MACDSignalPeriod); // Utilisation des paramètres du MACD
```

```
EMA1.Plots[0].Brush = Brushes.CornflowerBlue;
```

```
EMA2.Plots[0].Brush = Brushes.Aqua;
```

```
EMA3.Plots[0].Brush = Brushes.Crimson;
```

```
MACD1.Plots[0].Brush = Brushes.DarkCyan;
```

```
MACD1.Plots[1].Brush = Brushes.Crimson;
```

```
MACD1.Plots[2].Brush = Brushes.DodgerBlue;
```

```
AddChartIndicator(EMA1);
```

```
AddChartIndicator(EMA2);
```

```
AddChartIndicator(EMA3);
```

```
AddChartIndicator(MACD1);
```

```
SetStopLoss("", CalculationMode.Currency, SL, false);
```

```
}
```

```
}
```

```
protected override void OnBarUpdate()
```

```
{
```

```
if (BarsInProgress != 0)
```

```
return;
```

```
if (CurrentBars[0] < 1)
```

```
return;
```

```
// Calculer la valeur de l'ATR (sur le 15-minute)
```

```
ATRValue = ATR(ATRPeriode)[0] * ATRMulti;
```

```
// Ensemble 1 : Condition pour entrer en position longue
```

```
if ((EMA1[0] < EMA2[0])
```

```
&& (EMA2[0] < EMA3[0])
```

```
&& (Open[0] < EMA3[0])
```

```

    && (High[0] > EMA3[0])

    && (Times[0][0].TimeOfDay > HeureDepart.TimeOfDay)

    && (Times[0][0].TimeOfDay < HeureFin.TimeOfDay)

    && (MACD1.Avg[0] < 0))

{
    EnterLong(Convert.ToInt32(DefaultQuantity), "");

    LongStopLevel = Position.AveragePrice - ATRValue; // Initialisation du niveau de stop pour position
longue
}

// Ensemble 2 : Condition pour entrer en position courte
if ((EMA1[0] > EMA2[0])

    && (EMA2[0] > EMA3[0])

    && (Open[0] > EMA3[0])

    && (Low[0] < EMA3[0])

    && (Times[0][0].TimeOfDay > HeureDepart.TimeOfDay)

    && (Times[0][0].TimeOfDay < HeureFin.TimeOfDay)

    && (MACD1.Avg[0] > 0))

{
    EnterShort(Convert.ToInt32(DefaultQuantity), "");

    ShortStopLevel = Position.AveragePrice + ATRValue; // Initialisation du niveau de stop pour position
courte
}

// Mise à jour du stop dynamique basé sur l'ATR pour les positions longues
if (Position.MarketPosition == MarketPosition.Long)
{
    // Ajuster le stop uniquement si le prix a évolué favorablement
    if (Close[0] > Position.AveragePrice + ATRValue) // Si le prix a augmenté plus que l'ATR
    {
        LongStopLevel = Close[0] - ATRValue; // Déplacer le stop à la nouvelle position
    }
}

```

```

        SetStopLoss("LongStop", CalculationMode.Price, LongStopLevel, false);
    }
}

// Mise à jour du stop dynamique basé sur l'ATR pour les positions courtes
if (Position.MarketPosition == MarketPosition.Short)
{
    // Ajuster le stop uniquement si le prix a évolué favorablement
    if (Close[0] < Position.AveragePrice - ATRValue) // Si le prix a baissé plus que l'ATR
    {
        ShortStopLevel = Close[0] + ATRValue; // Déplacer le stop à la nouvelle position
        SetStopLoss("ShortStop", CalculationMode.Price, ShortStopLevel, false);
    }
}

// Ensemble 3 : Sortie des positions longues à l'heure de clôture
if (Times[0][0].TimeOfDay == HeureClose.TimeOfDay)
{
    ExitLong(Convert.ToInt32(DefaultQuantity), "", "");
}

// Ensemble 4 : Sortie des positions courtes à l'heure de clôture
if (Times[0][0].TimeOfDay == HeureClose.TimeOfDay)
{
    ExitShort(Convert.ToInt32(DefaultQuantity), "", "");
}
}
}
}

```


