

1-28-2012_docx_Trading Automated Strategy Development polynomial Regression Channel.docx
1-28-2012_mhtml_Trading Automated Strategy Development polynomial Regression Channel.mht

Trading Automated Strategy Development
Automated Trading System (ATS) development platform
Trading Automated Strategy Development Regression polynomial linear

Trading Automated Strategy Development polynomial Regression Channel Vs linear
Trading Automated Strategy Development polynomial Regression Channel

[DOC]

Review of

www.onafree.com/.../9-6-10_Mon_doc_...File Format: Microsoft Word - Quick View

In an automated portfolio trading system strategies of mixed securities such as linear and polynomial, with trading rules that deal with the Regression Channel, Analysis of Variance” using “Internet Software Development” to automate the ...

https://docs.google.com/viewer?a=v&q=cache:s5WdnPcqnsmJ:www.onafree.com/PortfolioManager/9-6-10_Mon_doc_COST%2520FUNCTION%2520MODEL%2520ASSUMPTIONS%2520NARRATIVE%2520AND%2520VALIDITY.doc+Trading+Automated+Strategy+Development+polynomial+Regression+Channel&hl=en&gl=us&pid=bl&srcid=ADGEESjSUsBMcXgjR3AP5lwNIoVs23HOctog9dts2a8E_QiHkTJxVqOEGlKH-cOPidjVL4em8i7_q9ecDUVMTme8HSPttZ4ugQ5JE88jvan8_WPGPvEPPmBToJSuV12KEMBQTe4hA8&sig=AHIEtbTk3SQLeTaLeLMxsaISY3qdOe8JmQ

Polynomial Regression Channel Source Code for NinjaTrader

Well then if we make one that works properly look at all the new users Sierra will have.

<https://www.sierrachart.com/supportboard/archive/index.php/t-27303.html>

here is ninja code written by a russian if you have a programming background you will see that it recalculates, redraws up to 10 bar back.

```
#region Using declarations
using System;
using System.ComponentModel;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Xml.Serialization;
using NinjaTrader.Cbi;
using NinjaTrader.Data;
using NinjaTrader.Gui.Chart;
#endregion
```

```
// This namespace holds all indicators and is required. Do not change it.
```

```
namespace NinjaTrader.Indicator
```

```
{
```

```
/// <summary>
```

```
/// Polynomial Regression Channel
```

```
/// </summary>
```

```
[Description("Polynomial Regression Channel")]
```

```
//port of indicator from MQL4 to NinjaTrader. Original author: fxcanada
```

```
// http://codebase.mql4.com/4332
```

```

[Gui.Design.DisplayName(" PRC")]
public class PRC : Indicator
{
#region Variables
// Wizard generated variables
private int degree = 3; // Default setting for Degree
private int period = 250; // Default setting for Period 250
private double strdDev = 1.62; // Default setting for StrdDev 2
private double strdDev2 = 2.000; // Default setting for StrdDev 2
// User defined variables (add any user defined variables below)
private double[,] ai = new double[10,10];
private double[] b = new double[10];
private double[] x = new double[10];
private double[] sx = new double[10];
private double sum;
private int ip;
private int p;
private int n;
private int f;
private double qq;
private double mm;
private double tt;
private int ii;
private int jj;
private int kk;
private int ll;
private int nn;
private double sq;
private double sq2;
private int i0 = 0;
private int mi;
private int sounds = 0;
private bool Standard = true;
private bool fillZones = false;
private bool showLine = false;
private Color fillColor = Color.Yellow;
#endregion

/// <summary>
/// This method is used to configure the indicator and is called once before any bar data is loaded.
/// </summary>
protected override void Initialize()
{
Add(new Plot(new Pen(Color.DimGray, 2), PlotStyle.Line, "Fx")); //Plot0
Add(new Plot(Color.FromKnownColor(KnownColor.Red), PlotStyle.Line, "Sqh")); //Plot1
Add(new Plot(Color.FromKnownColor(KnownColor.Blue), PlotStyle.Line, "Sql")); //Plot2
Add(new Plot(Color.FromKnownColor(KnownColor.Red), PlotStyle.Line, "Sqh2")); //Plot3
Add(new Plot(Color.FromKnownColor(KnownColor.Blue), PlotStyle.Line, "Sql2")); //Plot4
Plots[0].Pen.DashStyle = DashStyle.DashDot;

Add(new Plot(new Pen(Color.Orange, 2), PlotStyle.Line, "TEMA")); //Plot5
Add(new Plot(new Pen(Color.Peru, 2), PlotStyle.Line, "HA TEMA")); //Plot6

CalculateOnBarClose = false;
Overlay = true;
PriceTypeSupported = true;

```

```

}

/// <summary>
/// Called on each bar update event (incoming tick)
/// </summary>
protected override void OnBarUpdate()
{
    if( CurrentBar < period) return;

    ip = period;
    p = ip;
    sx[1] = p + 1;
    nn = degree + 1;
    //-----sx-----
    for(mi=1;mi<=nn*2-2;mi++) // математическое выражение - для всех mi от 1 до nn*2-2
    {
        sum=0;
        for(n=i0;n<=i0+p;n++)
        {
            sum+=Math.Pow(n,mi);
        }
        sx[mi+1]=sum;
    }
    //-----syx-----
    for(mi=1;mi<=nn;mi++)
    {
        sum=0.00000;
        for(n=i0;n<=i0+p;n++)
        {
            if(mi==1) sum+=Close[n];
            else sum+=Close[n]*Math.Pow(n,mi-1);
        }
        b[mi]=sum;
    }
    //=====Matrix=====
    =====
    for(jj=1;jj<=nn;jj++)
    {
        for(ii=1; ii<=nn; ii++)
        {
            kk=ii+jj-1;
            ai[ii,jj]=sx[kk];
        }
    }
    //=====Gauss=====
    =====
    for(kk=1; kk<=nn-1; kk++)
    {
        ll=0;
        mm=0;
        for(ii=kk; ii<=nn; ii++)
        {
            if(Math.Abs(ai[ii,kk])>mm)
            {
                mm=Math.Abs(ai[ii,kk]);
                ll=ii;
            }
        }
    }
}

```

```

}
}
if(ll==0) return;
if (ll!=kk)
{
for(jj=1; jj<=nn; jj++)
{
tt=ai[kk,jj];
ai[kk,jj]=ai[ll,jj];
ai[ll,jj]=tt;
}
tt=b[kk];
b[kk]=b[ll];
b[ll]=tt;
}
for(ii=kk+1;ii<=nn;ii++)
{
qq=ai[ii,kk]/ai[kk,kk];
for(jj=1;jj<=nn;jj++)
{
if(jj==kk) ai[ii,jj]=0;
else ai[ii,jj]=ai[ii,jj]-qq*ai[kk,jj];
}
b[ii]=b[ii]-qq*b[kk];
}
}
x[nn]=b[nn]/ai[nn,nn];
for(ii=nn-1;ii>=1;ii--)
{
tt=0;
for(jj=1;jj<=nn-ii;jj++)
{
tt=tt+ai[ii,ii+jj]*x[ii+jj];
x[ii]=(1/ai[ii,ii])*(b[ii]-tt);
}
}
//=====
=====
for(n=i0;n<=i0+p;n++)
{
sum=0;
for(kk=1;kk<=degree;kk++)
{
sum+=x[kk+1]*Math.Pow(n,kk);
}
Fx.Set(n,x[1]+sum);
}
//-----Std-----
sq=0.0;
for(n=i0;n<=i0+p;n++)
{
sq+=Math.Pow(Close[n]-Fx[n],2);
}
sq=Math.Sqrt(sq/(p+1))*strdDev;

for(n=i0;n<=i0+p;n++)

```

```

{
Sqh.Set(n,Fx[n]+sq);
Sql.Set(n,Fx[n]-sq);
}

if(showLine)
{
sq2=0.0;
for(n=i0;n<=i0+p;n++)
{
sq2+=Math.Pow(Close[n]-Fx[n],2);
}
sq2=Math.Sqrt(sq2/(p+1))*strdDev2;

for(n=i0;n<=i0+p;n++)
{
Sqh2.Set(n,Fx[n]+sq2);
Sql2.Set(n,Fx[n]-sq2);
}
}

//-----Fill Zones
if(fillZones && showLine)
{
DrawRegion("HiZone", CurrentBar, 0, Sqh, Sqh2, Color.Empty, fillColor, 2);
DrawRegion("LoZone", CurrentBar, 0, Sql, Sql2, Color.Empty, fillColor, 2);
}

//-----Alerts
if(CrossAbove(Close, Sqh, 1) && sounds > 0 || CrossBelow(Close, Sql, 1) && sounds > 0)
{if(Standard)
{if(sounds == 1)
{Alert(" Ready",NinjaTrader.Cbi.Priority.High,"Ready","c:\\windows\\media\\chimes.wav",4,Color.Black,Color.Yellow);}
if(sounds == 2)
{Alert(" Ready",NinjaTrader.Cbi.Priority.High,"Ready","c:\\windows\\media\\ding.wav",4,Color.Black,Color.Yellow);}
if(sounds == 3)
{Alert(" Ready",NinjaTrader.Cbi.Priority.High,"Ready","alert4.wav",4,Color.Black,Color.Yellow);}}
else
{if(sounds == 1)
{PlaySound(@"c:\\windows\\media\\chimes.wav");}
if(sounds == 2)
{PlaySound(@"c:\\windows\\media\\ding.wav");}
if(sounds == 3)
{PlaySound(@"\\alert4.wav");}}
}

}

#region Properties
[Browsable(false)] // this line prevents the data series from being displayed in the indicator
properties dialog, do not remove
[XmlIgnore()] // this line ensures that the indicator can be saved/recovered as part of a chart template,
do not remove

```

```
public DataSeries Fx
{
get { return Values[0]; }
}
```

```
[Browsable(false)] // this line prevents the data series from being displayed in the indicator
properties dialog, do not remove
[XmlIgnore()] // this line ensures that the indicator can be saved/recovered as part of a chart template,
do not remove
public DataSeries Sqh
{
get { return Values[1]; }
}
```

```
[Browsable(false)] // this line prevents the data series from being displayed in the indicator
properties dialog, do not remove
[XmlIgnore()] // this line ensures that the indicator can be saved/recovered as part of a chart template,
do not remove
public DataSeries Sqj
{
get { return Values[2]; }
}
```

```
[Browsable(false)] // this line prevents the data series from being displayed in the indicator
properties dialog, do not remove
[XmlIgnore()] // this line ensures that the indicator can be saved/recovered as part of a chart template,
do not remove
public DataSeries Sqh2
{
get { return Values[3]; }
}
```

```
[Browsable(false)] // this line prevents the data series from being displayed in the indicator
properties dialog, do not remove
[XmlIgnore()] // this line ensures that the indicator can be saved/recovered as part of a chart template,
do not remove
public DataSeries Sqj2
{
get { return Values[4]; }
}
```

```
[Description("type of polynomial 1, 2, 3 or 4")]
[Category("Parameters")]
public int Degree
{
get { return degree; }
set { degree = Math.Min(4,Math.Max(value,1)); }
}
```

```
[Description("nmbr of bars to use in calculation")]
[Category("Parameters")]
public int Period
{
get { return period; }
set { period = Math.Max(1, value); }
}
```

```
[Description("Standard Deviations")]
[Category("Parameters")]
public double StrdDev
{
    get { return strdDev; }
    set { strdDev = Math.Max(0.5, value); }
}
```

```
[Description("Standard Deviations")]
[Category("Parameters")]
public double StrdDev2
{
    get { return strdDev2; }
    set { strdDev2 = Math.Max(0.5, value); }
}
```

```
[Description("0 = no sound, 1 = Chimes, 2 = Ding, 3 = Alert4(NT)")]
[Gui.Design.DisplayName ("\\t\\tSound Selection")]
[Category("Options")]
public int Sounds
{
    get { return sounds; }
    set { sounds = Math.Min(3,Math.Max(value,0)); }
}
```

```
[Description("Standard alert (true) or constant sound barrage (false)")]
[Gui.Design.DisplayName ("\\t\\tStandard Alert?")]
[Category("Options")]
public bool standard
{
    get { return Standard; }
    set { Standard = value; }
}
```

```
[Description("Fill the zones created by the two Sqh and Sql lines")]
[Gui.Design.DisplayName ("Fill OB/OS zones?")]
[Category("Options")]
public bool FillZones
{
    get { return fillZones; }
    set { fillZones = value; }
}
```

```
[Description("This will plot the second set of lines which then gives the option of filling the zone")]
[Gui.Design.DisplayName ("\\t\\tPlot Sqh2 and Sql2 lines?")]
[Category("Options")]
public bool ShowLine
{
    get { return showLine; }
    set { showLine = value; }
}
```

```
[Browsable(false)]
```

```

public string fillColorSerialize
{
get { return NinjaTrader.Gui.Design.SerializableColor.ToString(fillColor); }
set { fillColor = NinjaTrader.Gui.Design.SerializableColor.FromString(value); }
}

```

```

[XmlIgnore()]
[Description("Fill zone color.")]
[Category("Options")]
[Gui.Design.DisplayNameAttribute("Fill zone color")]
public Color FillColor
{
get { return fillColor; }
set { fillColor = value; }
}
#endregion
}
}

```

#region NinjaScript generated code. Neither change nor remove.

// This namespace holds all indicators and is required. Do not change it.

namespace NinjaTrader.Indicator

```

{
public partial class Indicator : IndicatorBase
{
private PRC[] cachePRC = null;

```

```

private static PRC checkPRC = new PRC();

```

```

/// <summary>

```

```

/// Polynomial Regression Channel

```

```

/// </summary>

```

```

/// <returns></returns>

```

```

public PRC PRC(int degree, int period, double strdDev, double strdDev2)

```

```

{
return PRC(Input, degree, period, strdDev, strdDev2);
}

```

```

/// <summary>

```

```

/// Polynomial Regression Channel

```

```

/// </summary>

```

```

/// <returns></returns>

```

```

public PRC PRC(Data.IDataSeries input, int degree, int period, double strdDev, double strdDev2)

```

```

{
checkPRC.Degree = degree;
degree = checkPRC.Degree;
checkPRC.Period = period;
period = checkPRC.Period;
checkPRC.StrdDev = strdDev;
strdDev = checkPRC.StrdDev;
checkPRC.StrdDev2 = strdDev2;
strdDev2 = checkPRC.StrdDev2;

```

```

if (cachePRC != null)

```

```

for (int idx = 0; idx < cachePRC.Length; idx++)

```

```

if (cachePRC[idx].Degree == degree && cachePRC[idx].Period == period &&
Math.Abs(cachePRC[idx].StrdDev - strdDev) <= double.Epsilon && Math.Abs(cachePRC[idx].StrdDev2 -
strdDev2) <= double.Epsilon && cachePRC[idx].EqualsInput(input))
return cachePRC[idx];

```

```

PRC indicator = new PRC();
indicator.BarsRequired = BarsRequired;
indicator.CalculateOnBarClose = CalculateOnBarClose;
indicator.Input = input;
indicator.Degree = degree;
indicator.Period = period;
indicator.StrdDev = strdDev;
indicator.StrdDev2 = strdDev2;
indicator.SetUp();

```

```

PRC[] tmp = new PRC[cachePRC == null ? 1 : cachePRC.Length + 1];
if (cachePRC != null)
cachePRC.CopyTo(tmp, 0);
tmp[tmp.Length - 1] = indicator;
cachePRC = tmp;
Indicators.Add(indicator);

```

```

return indicator;
}

```

```

}
}

```

// This namespace holds all market analyzer column definitions and is required. Do not change it.

```

namespace NinjaTrader.MarketAnalyzer

```

```

{
public partial class Column : ColumnBase
{
/// <summary>
/// Polynomial Regression Channel
/// </summary>
/// <returns></returns>
[Gui.Design.WizardCondition("Indicator")]
public Indicator.PRC PRC(int degree, int period, double strdDev, double strdDev2)
{
return _indicator.PRC(Input, degree, period, strdDev, strdDev2);
}
}

```

```

/// <summary>
/// Polynomial Regression Channel
/// </summary>
/// <returns></returns>
public Indicator.PRC PRC(Data.IDataSeries input, int degree, int period, double strdDev, double
strdDev2)
{
return _indicator.PRC(input, degree, period, strdDev, strdDev2);
}
}

```

```

}
}

```

```

// This namespace holds all strategies and is required. Do not change it.
namespace NinjaTrader.Strategy
{
public partial class Strategy : StrategyBase
{
/// <summary>
/// Polynomial Regression Channel
/// </summary>
/// <returns></returns>
[Gui.Design.WizardCondition("Indicator")]
public Indicator.PRC PRC(int degree, int period, double strdDev, double strdDev2)
{
return _indicator.PRC(Input, degree, period, strdDev, strdDev2);
}

/// <summary>
/// Polynomial Regression Channel
/// </summary>
/// <returns></returns>
public Indicator.PRC PRC(Data.IDataSeries input, int degree, int period, double strdDev, double
strdDev2)
{
if (!Initialize && input == null)
throw new ArgumentException("You only can access an indicator with the default input/bar series from
within the 'Initialize()' method");

return _indicator.PRC(input, degree, period, strdDev, strdDev2);
}

}
}
#endregion

```

<http://www.mypivots.com/board/topic/6450/1/polynomial-regression-channel-bands>

http://blog.bigmiketrading.com/2009_04_01_archive.html

<http://www.youtube.com/watch?v=peYbHALhyX8>
Tutorial 49 Part 1 - Linear Regression Channel

Polynomial Regression Channel

<http://www.ninjatrader.com/support/forum/showthread.php?t=16318>

PolynomialRegressionChannel.zip (2.9 KB, 416 views)

This is the EA of the polynomial in Metatrader language, the image are the four operations yesterday afternoon (I think I have had much luck!)

Time frame 1 minute

Attached Images

polynomial_EA.png (21.8 KB, 351 views)

Attached Files

e-Regr.zip (2.4 KB, 126 views)

<http://www.codeproject.com/Articles/28948/Curve-Fitting-using-Lagrange-Interpolation>

http://www.esignal.com/solutions/studies_addon.aspx

<http://www.codeproject.com/Articles/4512/A-VB-NET-class-useful-for-basic-linear-algebra>

http://www.janarps.com/Videos/Radar1/radar_1/radar_1.html

Now, for a Limited Time, Hurst Polynomial Regression Bands Included at No Extra Charge.

In addition to the Crown Jewels Tool Kit, Jan Arps' Traders' Toolbox now also offers the revolutionary new Hurst Polynomial Regression Bands Tool Kit, consisting of the following tools:

The Hurst Bands-Historic indicator plots a set of standard deviation channel lines around a polynomial regression centerline to help the user identify levels of optimum probability for entering and exiting trades. The Hurst Bands-Actual indicator plots the progress of the end point of the Hurst Bands-Historic calculation for each bar on the chart, thus providing a continuous evaluation of how the Hurst Bands' relationships have changed with time.

The Hurst Bands-Oscillator provides an indication of normalized deviations of price from the regression mean, thus revealing when prices are overbought or oversold in varying volatility environments.

The Hurst Bands-Multi-Order Centerlines indicator provides an excellent means for identifying trend direction and strength through the interaction between two polynomial regression centerlines of different orders.

The Hurst Polynomial Regression Bands indicators are available, for a limited time only, as part of the Crown Jewels service at no additional cost.

Develop your strategy concept, construct it, historically test it, optimize it, simulation test it and then live deploy it through your live brokerage account with NinjaTrader's strategy development tools. No programming knowledge required! This session is designed to provide a high-level overview of the following areas:

Create a strategy in the Strategy Wizard

Create entry and exit conditions and invoke actions

Backtest using the Strategy Analyzer

Understand the backtest results

Plan to attend this session if you want to maximize your efficiency in utilizing NinjaTrader for automated strategy development. After completing the Automated Strategy Development Level I training session, you will:

Know the process for constructing a strategy using the Strategy Wizard

Know how to create entry and exit conditions

Know how to backtest a strategy

Know how to view the performance results of a backtested strategy