Service Disclaimer

Overview

The Modulus APR library is the world's first template-driven, fully dynamic, pattern recognition engine for identifying patterns in financial data. Our patent-pending technology uses fuzzy logic to match dynamic patterns in stock, futures and forex data with extreme speed and accuracy.

Fuzzy Logic Processing

Human traders use implicit learning to identify patterns in price data. This occurs when traders are repeatedly exposed to certain complex price patterns, and eventually develop a personal understanding for the pattern even though the pattern and predictive feeling cannot be verbally explained in human language.

It is proven that fuzzy logic systems can be effectively used for decision-making support of trading processes by reproducing the intuitiveness of human traders.

The Modulus APR fuzzy logic processor identifies patterns as human traders would. APR features several optimization parameters, which allow you to fine tune the fuzzy logic pattern identification process based on your own preferences.

Pattern-Based Alerts and Back Testing

APR can be used as a basis for creating real-time pattern alerts or back testing strategies based on pre-defined or custom pattern definitions. APR ships with several pre-defined patterns such as Channels, Double Bottoms, Double Tops, Flags, Head & Shoulders, Pennants, Trend, Triangles, Triple Bottoms, Triple Tops, Wedges and other patterns. Custom patterns can be created using the supplied pattern designer utility, which you may deploy to your end users.

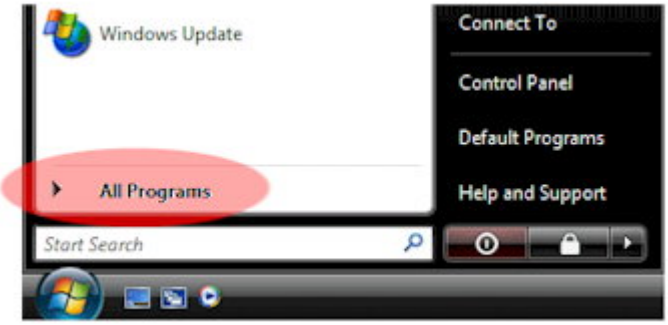
Prerequisites

APR is compatible with all versions of Windows and requires only 4MB of free RAM. The APR designer utility (optional) requires the Microsoft .NET 3.5 framework. A working knowledge of C# 2008 or later and WPR is required for working with the designer utility source code. A working knowledge of C programming is required to work with the source code to the APR library.

Introduction

APR may be used with any Windows programming language, including VB6, VB.Net, C#, VC++, Delphi, etc. ModulusAPR.dll is a standard Win32 dynamic link library (not a COM library). No COM registration is required for installation.

After installing APR, example projects may be found on the Windows Start menu in the All Programs \ “Modulus Financial Engineering\Advanced Pattern Recognition\” folder.



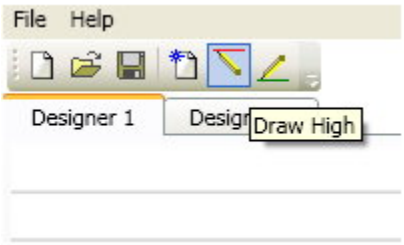
APR Pattern Designer Utility

The APR Pattern Designer is a utility application that allows you to draw pattern templates and save them in XML format. You may then pass the XML template files to the APR library in order to scan for the patterns that you have designed.

The APR Pattern Designer requires the Microsoft .NET 3.5 framework to run. Windows Vista provides this framework by default. If you do not have the .NET 3.5 framework on your computer, you must install it prior to running the APR Pattern Designer application.

The APR Pattern Designer may be distributed with your software application if you have purchased a distributable license of the APR library. You may not, however, distribute the APR DLL source code or APR designer source code.

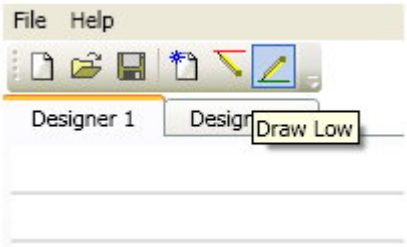
Creating a Pattern

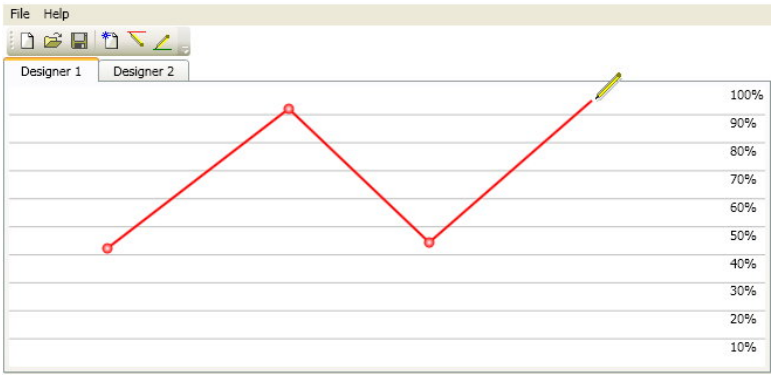


To being, click the Draw High button on the toolbar. This will enable the drawing tool. You may then create the upper boundary of your pattern by left clicking on the chart to create points.

You may create as many points as necessary to generate the pattern. You may draw anywhere within the drawing area.

Right click to finish the upper line drawing then select the Draw Low button to begin drawing the lower boundary.

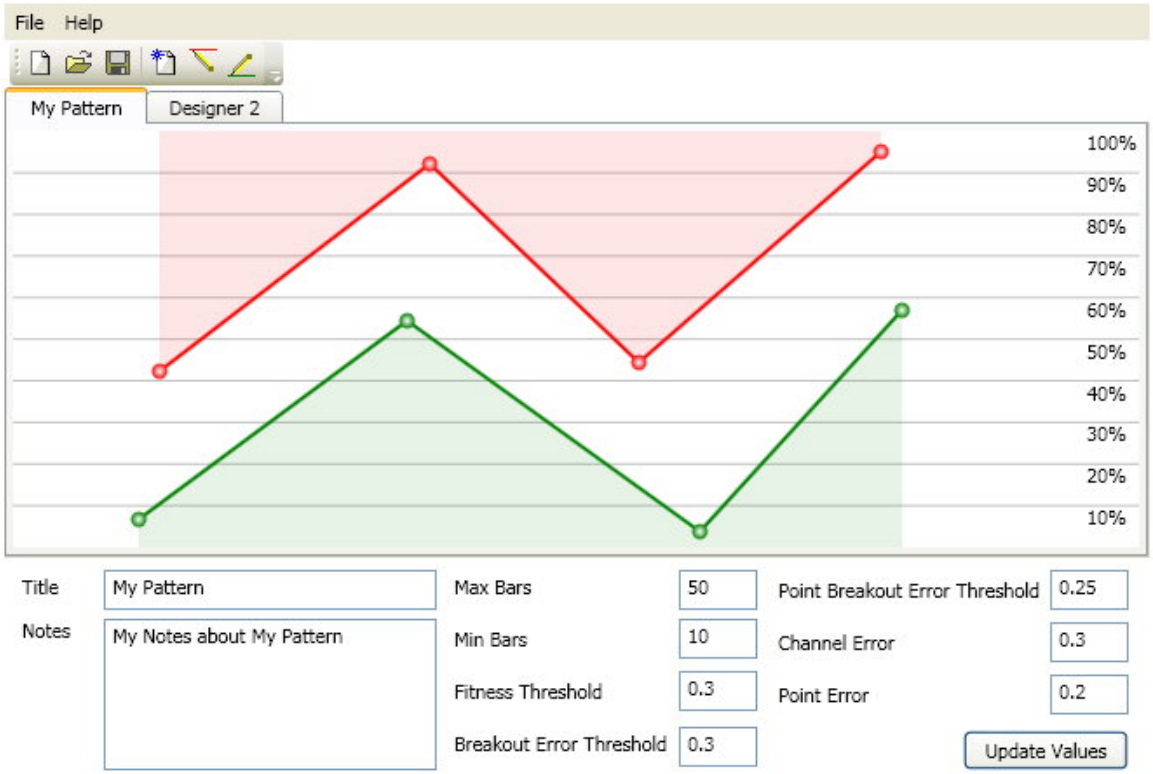




If you make a mistake while drawing the pattern, you may click the “Clear” button on the toolbar to clear the line drawings and start over.

Modifying Pattern Properties

You may modify the fuzzy logic parameters as needed after the pattern has been drawn. The default parameters may be used for most patterns, and should only be changed when absolutely necessary. These are considered “advanced” settings.



APR Fuzzy Logic Parameters

Before describing the fuzzy logic parameters and their meanings, it is important to understand how APR works.

APR scans through data by paging through incremental “windows”. A window is defined by the Max Bars and Min Bars parameters. For example, if you specify 50 for the maximum number of bars and 10 for the minimum number of bars, APR will search for patterns consisting of a number of records up to 40 bars in length. More weight is given to patterns containing the most number of bars. The minimum value for Min Bars is 5.

Each window of data is passed through a fuzzy logic filter. It is beyond the scope of this document to explain how the filter works. The filter is based on our patent-pending fuzzy logic algorithm, and may be reviewed in the DLL source code under a separate license.

Ranking Values

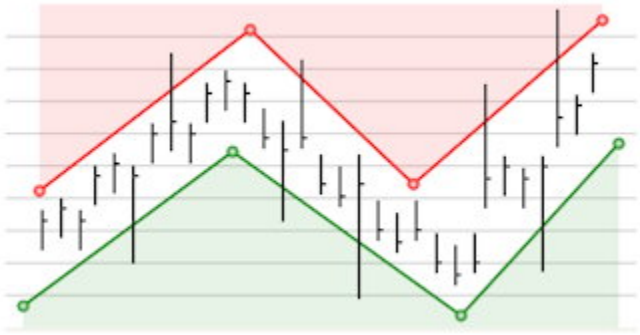
APR returns a ranking value for each pattern that is identified. This value ranges between 0 and 1, with 1 being a perfect pattern and zero being no match at all. Values above 0.6 seldom occur. Most ranking values are between 0.35 and 0.6.

Max Bars (10 to 1000, default 50)
Maximum number of bars in the pattern window. Processing time increases with the window size (Max Bars – Min Bars). Higher rankings are given to patterns containing the largest number of records. This value cannot be less than Min Bars.

Min Bars (5 to 990, default 1)
Minimum number of bars in the pattern window. Processing time increases with the window size (Max Bars – Min Bars). Higher rankings are given to patterns containing the largest number of records. This value cannot exceed Max Bars - 10.

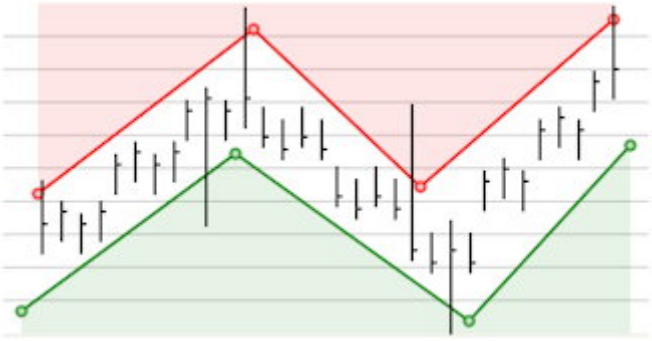
Fitness Threshold (0 to 0.95, default 0.3)
Filters patterns that have a fitness ranking below the specified value. For example, if a rank is 0.3 and Fitness Threshold is 0.31, the pattern will not be added to the results list.

Breakout Error Threshold (0 to 1, default 0.3)
This threshold value filters patterns if any bar is more than n-% outside the pattern channel. For example, if Breakout Error Threshold is set to 0.5 and any one bar in the pattern window is over 50% outside the channel, the pattern will not be added to the results list.



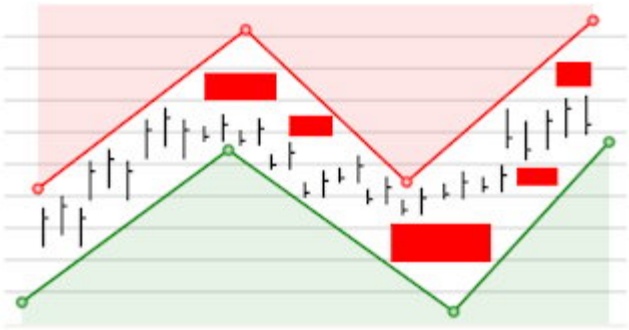
Breakout Error Threshold

Point Breakout Error Threshold (0 to 1, default 0.25)
This threshold value filters patterns if any bar that is within 1 record of a pattern data point is more than n-% outside the pattern channel. For example, if Point Breakout Error Threshold is set to 0.5 and any single bar in the pattern window that is within 1 record of a pattern point is over 50% outside the channel, the pattern will not be added to the results list.



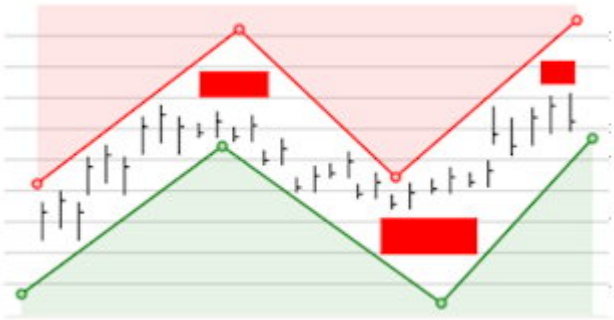
Point Breakout Error Threshold

Channel Error (0 to 1, default 0.3)
Filters patterns that have an accumulative error of abs(high | low - trend) that is more than n-% outside this tolerance value.



Channel Error

Point Error
Filters patterns that have an accumulative error of abs(high | low - trend) that is more than n-% outside this tolerance value near pattern data points.



Point Error

ModulusAPR.dll – Win32 DLL Documentation

The APR library is extremely simple to use and consists of only ten Win32 API methods. Please note that the following documentation is considered complete. All functions listed below are shown in the example projects. Refer to the example projects for clarification.

SetRecordsCount (RecordCount As Long) As Long

This function must be called prior to calling any other function. Returns -1 it the number of records are > 1000000 (limit value), otherwise returns the number passed to function. Also erases all values that may have been previously set by AppendRecord.

AppendRecord(Date As String, DateLength As Integer, Open As Double, High As Double, Low As Double, Close As Double, Volume As Long) As Long

Appends records that will be scanned for patterns. Returns -1 if called more than the RecordCount value set by SetRecordsCount, otherwise returns the number of records added.

ScanForPatterns(XmlDefinition As String, LicenseKey As String) As Long

Returns -100 if the license string is incorrect, returns -1 if the xml file can't be loaded, otherwise returns the number of patterns that have been identified. Please note, the license string can be found on the "Downloads & Updates" page of our web site, to the right of the APR download link. This is the same license key used for the installation.

StartPatternsIteration()

This function readies the DLL for pattern iteration and must be called prior to calling the ForEachPattern function. This function returns no value.

ForEachPattern() As Long

Iterates through each pattern that has been identified after calling StartPatternsIteration. Returns1 if there are patterns to iterate, otherwise 0 if there are no patterns to iterate.

StartPatternValuesIteration(PatternIndex As Long) As Long

This pattern readies the DLL for record iteration through a specified pattern and must be called prior to calling the ForEachPatternValue function. Returns -1 if an incorrect pattern index is passed, otherwise returns 0. Please note that PatternIndex is 0-based.

ForEachPatternValue(Upper as Double, Lower As Double) as Long

Iterates through each record in the specified pattern after calling StartPatternValuesIteration. Returns 1 if there are records to iterate, otherwise 0 if there are no more records to iterate.

GetXmlStringValue(ValueName As String, ByRef Value As String, ValueBufferLength As Long)

Returns a string value by reference pointer based on a key value name. ValueBufferLength is the buffer length of the Value string and must be > 0. Supported key value names for ValueName are "Title" and "Notes". These values are set by the pattern designer utility.

GetXmlIntegerValue(ValueName As String) As Long
Returns a long value based on a key value name. Returns -1 if an invalid value name is passed. Supported key value names for ValueName are “MaxBars” and “MinBars”. These values are set by the pattern designer utility.

GetXmlDoubleValue(ValueName As String) As Double
Returns a double value based on a key value name. Returns -1 if an invalid value name is passed. Supported key value names for ValueName are “FitnessThreshold”, “BreakoutErrorThreshold”, “PointBreakoutErrorThreshold”, “ChannelError” and “PointError”. These values are set by the pattern designer utility.

Technical Support

Developer support is provided for API calls, example projects, questions regarding the Designer utility and the Designer utility source code (if a source code license has been purchased). Support is not provided for individual patterns, template settings, etc. (end-user technical support). For developer technical support, email support@modulusfe.com