Real-time trading in MATLAB

Yair Altman

Undocumented Matlab.com

altmany@gmail.com

A common algo-trading challenge

- Trading platforms are relatively closed
  - Difficult to develop automated trading platforms
  - Vendor lock-in – algos are often un-portable
  - Internal algo customizability usually limited
- Common solutions:
  - Use Excel with trading-platform plugin
  - Use limited internal programmability (MT4, TS)
  - Develop custom C++/Java applications

Why use MATLAB?

- Numerous out-of-the-box analysis functionality
    - Much more functionality than Excel or C++/Java
- Dedicated toolboxes for specific uses
    - Financial, Data-feed, Statistics, Econometrics, Optimization, Trading, …
- Tried-and-tested
    - Prevents risk of losses due to computational bugs
    - Most functions have editable source code – no secrets
    - Reduces total cost of ownership (develop/test/maintain)
- Easy learning curve – engineering power without needing to be a software developer
- Excellent at exactly the task taking most time/cost to develop: the algo strategy/model
    - All other components are usually far easier to develop
- mathworks.com/discovery/algorithmic-trading.html

However…

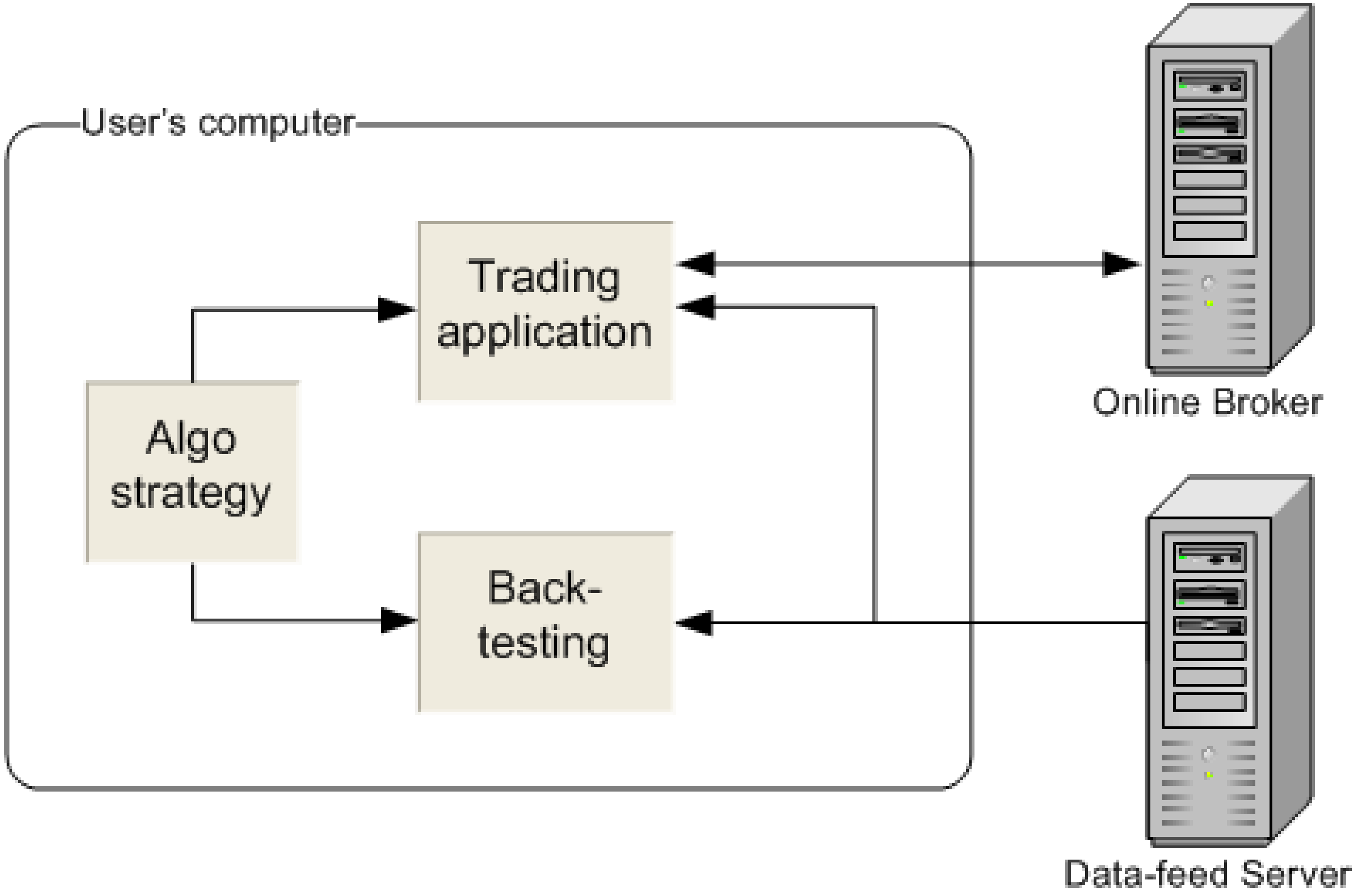
- MATLAB could not until recently complete the trading loop –

- Send automated trade orders to broker
- Modify/cancel open orders
- Track trade executions
- Receive portfolio/account info

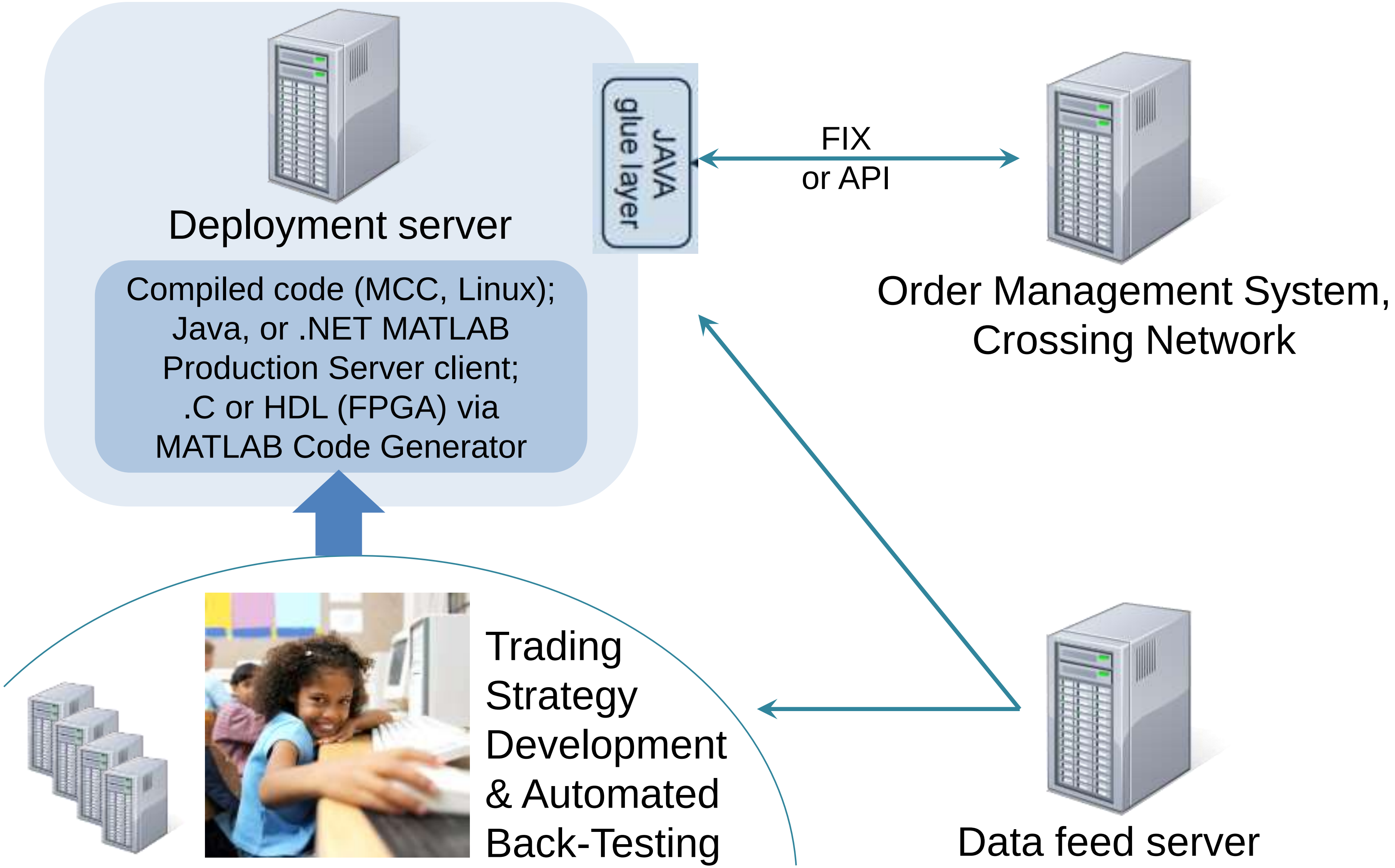
Solutions

- MATLAB 8.1 (R2013a): new Trading Toolbox
  - Windows only
  - Bloomberg EMSX
  - Trading Technologies X_TRADER
  - R2013b: Added CQG + IB interfaces
  - mathworks.com/products/trading
- MATLAB 7.1 (R14 SP3) onward: IB-MATLAB
  - Windows, Mac, Linux
  - Interactive Brokers only
  - UndocumentedMatlab.com/ib-matlab

General trading application design



Deployment in large institutions



Deployment server

Compiled code (MCC, Linux); Java, or .NET MATLAB Production Server client; .C or HDL (FPGA) via MATLAB Code Generator

JAVA glue layer

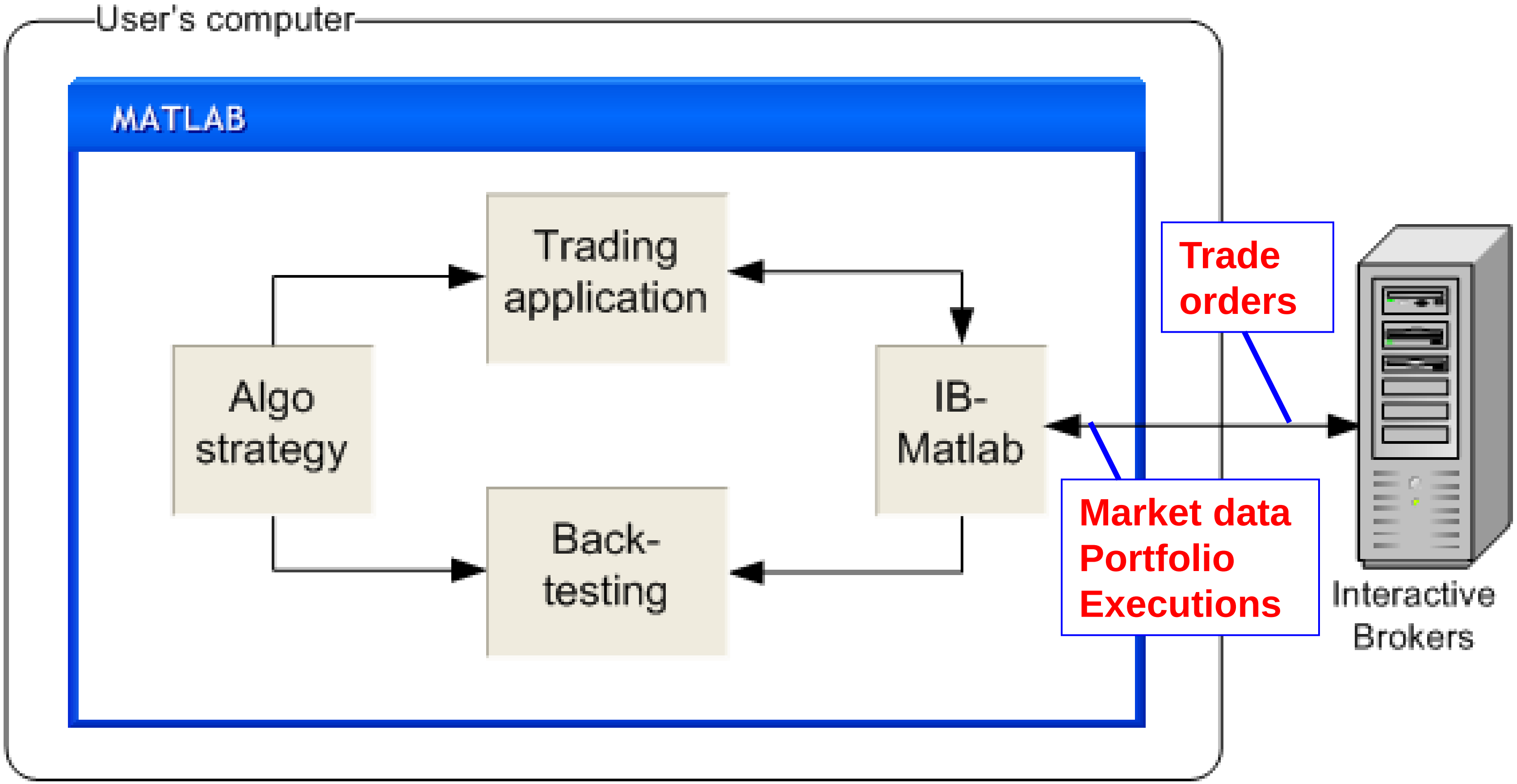
FIX
or API

Order Management System, Crossing Network

Trading Strategy Development & Automated Back-Testing

Data feed server

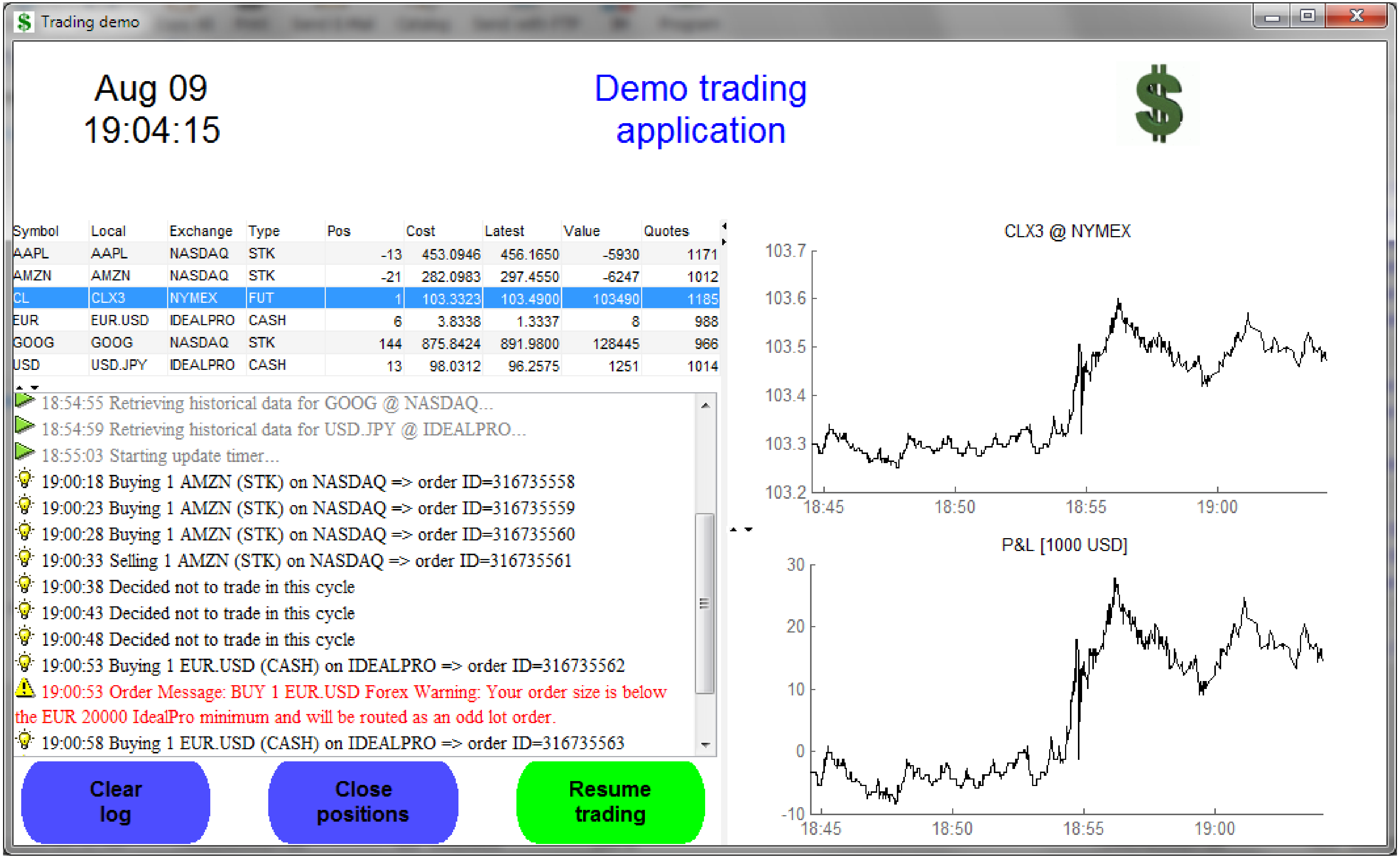
Today's demo



Trade orders

Market data
Portfolio
Executions

Live demo



Symbol	Local	Exchange	Type	Pos	Cost	Latest	Value	Quotes
AAPL	AAPL	NASDAQ	STK	-13	453.0946	456.1650	-5930	1171
AMZN	AMZN	NASDAQ	STK	-21	282.0983	297.4550	-6247	1012
CL	CLX3	NYMEX	FUT	1	103.3323	103.4900	103490	1185
EUR	EUR.USD	IDEALPRO	CASH	6	3.8338	1.3337	8	988
GOOG	GOOG	NASDAQ	STK	144	875.8424	891.9800	128445	966
USD	USD.JPY	IDEALPRO	CASH	13	98.0312	96.2575	1251	1014

```
18:54:55 Retrieving historical data for GOOG @ NASDAQ...
18:54:59 Retrieving historical data for USD.JPY @ IDEALPRO...
18:55:03 Starting update timer...
19:00:18 Buying 1 AMZN (STK) on NASDAQ => order ID=316735558
19:00:23 Buying 1 AMZN (STK) on NASDAQ => order ID=316735559
19:00:28 Buying 1 AMZN (STK) on NASDAQ => order ID=316735560
19:00:33 Selling 1 AMZN (STK) on NASDAQ => order ID=316735561
19:00:38 Decided not to trade in this cycle
19:00:43 Decided not to trade in this cycle
19:00:48 Decided not to trade in this cycle
19:00:53 Buying 1 EUR.USD (CASH) on IDEALPRO => order ID=316735562
19:00:53 Order Message: BUY 1 EUR.USD Forex Warning: Your order size is below
the EUR 20000 IdealPro minimum and will be routed as an odd lot order.
19:00:58 Buying 1 EUR.USD (CASH) on IDEALPRO => order ID=316735563
```

Interactive Brokers (IB)

- Low-cost online broker
- Consistently ranked Best Online Broker by *Barron's*
  - commissions
  - execution prices
  - features
  - exchanges
  - reports
- Widely used worldwide
- Fully documented API
  interactivebrokers.com/en/software/ibapi.php

IB-MATLAB

- Connects MATLAB to IB
  - Receive account data (portfolio, cash, limits)
  - Receive market data feed (historic, snapshot, streaming quotes)
  - Send trade orders to market
  - Modify/cancel open orders
  - Track executions (MATLAB callback functions)
  - Synchronous + asynchronous modes
  - Fully supports IB's API
  - 5-10 mS latency for IB events
- Works on all MATLAB platforms, Java-based API
- Hundreds of installations, trades $100M/day

IB-MATLAB: getting portfolio data

```
>> data = IBMatlab('action','PORTFOLIO')
data =
     1x12 struct array with fields:
        symbol
        localSymbol
        ...

>> data(1)
ans =
        symbol: 'AMZN'
   localSymbol: 'AMZN'
      exchange: 'NASDAQ'
       secType: 'STK'
      currency: 'USD'
         right: '0'
        expiry: ''
        strike: 0
      position: 9200
   marketValue: 1715800
   marketPrice: 186.5
   averageCost: 169.03183335
      contract: [1x1 struct]
```

IB-MATLAB: getting market data

```
>> data = IBMatlab('action','QUERY', 'symbol','GOOG')
data =
           reqId: 22209874
         reqTime: '02-Dec-2010 00:47:23'
        dataTime: '02-Dec-2010 00:47:23'
   dataTimestamp: 734474.032914491
          ticker: 'GOOG'
        bidPrice: 563.68
        askPrice: 564.47
            open: 562.82
           close: 555.71
             low: 562.4
            high: 571.57
       lastPrice: -1
          volume: 36891
            tick: 0.01
         bidSize: 3
         askSize: 3
        lastSize: 0
 contractDetails: [1x1 struct]
```

IB-MATLAB: getting historical data

```
>> data = IBMatlab('action','HISTORY', 'symbol','IBM', ...
                   'barSize','1 hour', 'useRTH',1)
data =
    dateNum: [1x7 double]
   dateTime: {1x7 cell}
       open: [161.08 160.95 161.66 161.17 161.57 161.75 162.07]
       high: [161.35 161.65 161.70 161.60 161.98 162.09 162.34]
        low: [160.86 160.89 161.00 161.13 161.53 161.61 161.89]
      close: [160.93 161.65 161.18 161.60 161.74 162.07 162.29]
     volume: [5384 6332 4580 2963 4728 4465 10173]
      count: [2776 4387 2990 1921 2949 2981 6187]
        WAP: [161.07 161.25 161.35 161.31 161.79 161.92 162.14]
    hasGaps: [0 0 0 0 0 0 0]


>> data.dateTime
ans =
 '20110225 16:30:00' '20110225 17:00:00' '20110225 18:00:00'
 '20110225 19:00:00' '20110225 20:00:00' '20110225 21:00:00'
 '20110225 22:00:00'
```

IB-MATLAB: sending orders to market

```
% Alternative #1: using a MATLAB struct
paramsStruct = [];
paramsStruct.action = 'BUY';
paramsStruct.symbol = 'GOOG';
paramsStruct.quantity = 100;
paramsStruct.limitPrice = 850;
orderId = IBMatlab(paramsStruct);

% Alternative #2: using name/value pairs
orderId = IBMatlab('action','BUY', 'symbol','GOOG', ...
                   'quantity',100, 'limitPrice',850);
```

IB-MATLAB: processing execution events

```
% Set the callback function for IB trade execution events
orderId = IBMatlab('action','BUY', 'symbol','GOOG', ...
                   'quantity',1, 'limitPrice',850,  ...
                   'CallbackExecDetails',@myExecDetailsFcn);



% Sample event callback function
function myExecDetailsFcn(hObject, eventData)

   % Extract the basic event data components
   contractData  = eventData.contract;
   executionData = eventData.execution;

   % Now do something useful with this information...

end
```

Some design considerations
(in no particular order)

- Build or buy
- Data feed provider (IB / IQFeed / eSignal / …)
- Synchronous (periodic) or asynchronous (reactive)
- Latency/frequency
  - → streaming quotes or periodic historical data requests
  - → perhaps we need to use C / FPGA code (for µS latency)
- Level of robustness, failsafe mechanisms
- GUI or GUI-less
- Semi or fully automated

Example for a very simple application design

```
% Main application entry point
function tradingApplication()
    tradeSymbol('CLX3', 15*60, @timerCallbackFunction);  %@15 mins
    tradeSymbol('GOOG', 10*60, @timerCallbackFunction);  %@10 mins
    tradeSymbol('DAX',   5*60, @timerCallbackFunction);  % @5 mins
    tradeSymbol('FTSE',  1*60, @timerCallbackFunction);  % @1 min
end


% Start an endless timer at the specified frequency that will
% run the specified callbackFunc upon each invocation
function hTimer = tradeSymbol(symbolName, period, callbackFunc)
    % Create the timer object
    hTimer = timer('ExecutionMode','fixedRate', ...
                   'Period',period, ...
                   'TimerFcn',{callbackFunc,symbolName});
    % Start the timer
    start(hTimer);
end  % tradeSymbol
```

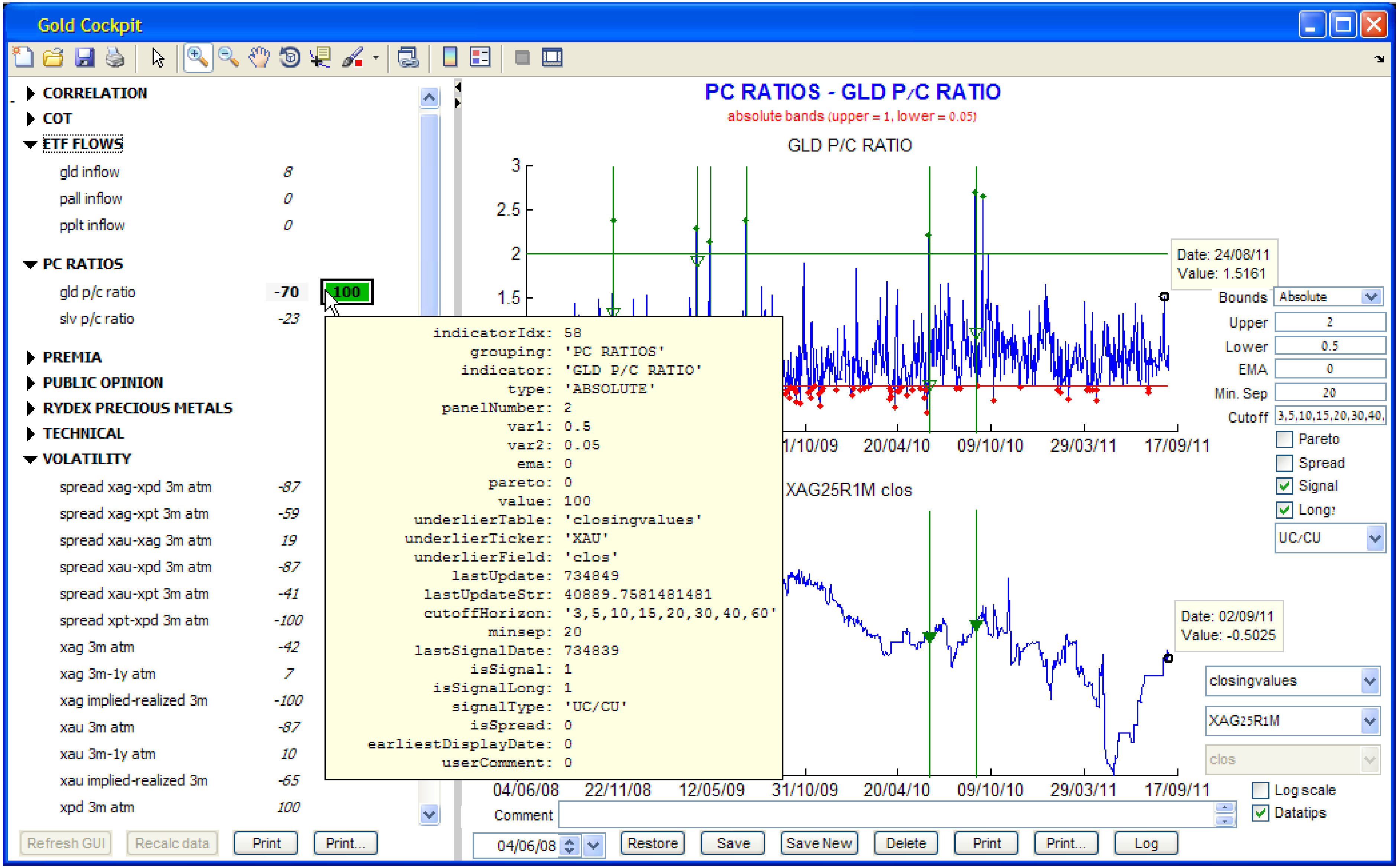
Example for a very simple application design

```
function timerCallbackFunction(hTimer, eventData, symbolName)
   try
      % Load previously-stored data for the specified contract
      persistentData = load(symbolName);

      % Get the latest data for this contract
      latestData = getLatestData(symbolName);

      % Process the data (secret sauce - algo strategy)
      [tradeParams, persistentData] = processData(latestData, ...
                                                  persistentData);

      % Process trade signals (send orders to IB)
      IBMatlab(tradeParams{:});

      % Save decision-making data for next timer invocation
      save(symbolName,'persistentData');
   catch
      processError(lasterror);
   end
end
```

Some additional bells & whistles

- Main engine (non-GUI)
  - Risk (open positions) management
  - Asynchronous trade execution tracking
  - FIX connection (rather than API)
  - Alerts via email/SMS (text-message)
- Graphical User Interface (GUI)
  - Open positions
  - Real-time market graph
    - TA indicators/bands, OHLC bars/candles, trade signals
  - Trades/executions log
  - P&L updates
  - Manual overrides (“panic button”)

Sample advanced MATLAB GUI



▸ CORRELATION

▸ COT

▾ ETF FLOWS

gld inflow	8
pall inflow	0
pplt inflow	0

▾ PC RATIOS

gld p/c ratio	-70	100
slv p/c ratio	-23	

▸ PREMIA

▸ PUBLIC OPINION

▸ RYDEX PRECIOUS METALS

▸ TECHNICAL

▾ VOLATILITY

spread xag-xpd 3m atm	-87
spread xag-xpt 3m atm	-59
spread xau-xag 3m atm	19
spread xau-xpd 3m atm	-87
spread xau-xpt 3m atm	-41
spread xpt-xpd 3m atm	-100
xag 3m atm	-42
xag 3m-1y atm	7
xag implied-realized 3m	-100
xau 3m atm	-87
xau 3m-1y atm	10
xau implied-realized 3m	-65
xpd 3m atm	100

```
     indicatorIdx: 58
         grouping: 'PC RATIOS'
        indicator: 'GLD P/C RATIO'
             type: 'ABSOLUTE'
      panelNumber: 2
             var1: 0.5
             var2: 0.05
              ema: 0
           pareto: 0
            value: 100
    underlierTable: 'closingvalues'
   underlierTicker: 'XAU'
    underlierField: 'clos'
        lastUpdate: 734849
     lastUpdateStr: 40889.7581481481
     cutoffHorizon: '3,5,10,15,20,30,40,60'
            minsep: 20
    lastSignalDate: 734839
          isSignal: 1
      isSignalLong: 1
        signalType: 'UC/CU'
          isSpread: 0
earliestDisplayDate: 0
       userComment: 0
```

Bounds	Absolute
Upper	2
Lower	0.5
EMA	0
Min. Sep	20
Cutoff	3,5,10,15,20,30,40,

☐ Pareto
☐ Spread
☑ Signal
☑ Long?

UC/CU

closingvalues

XAG25R1M

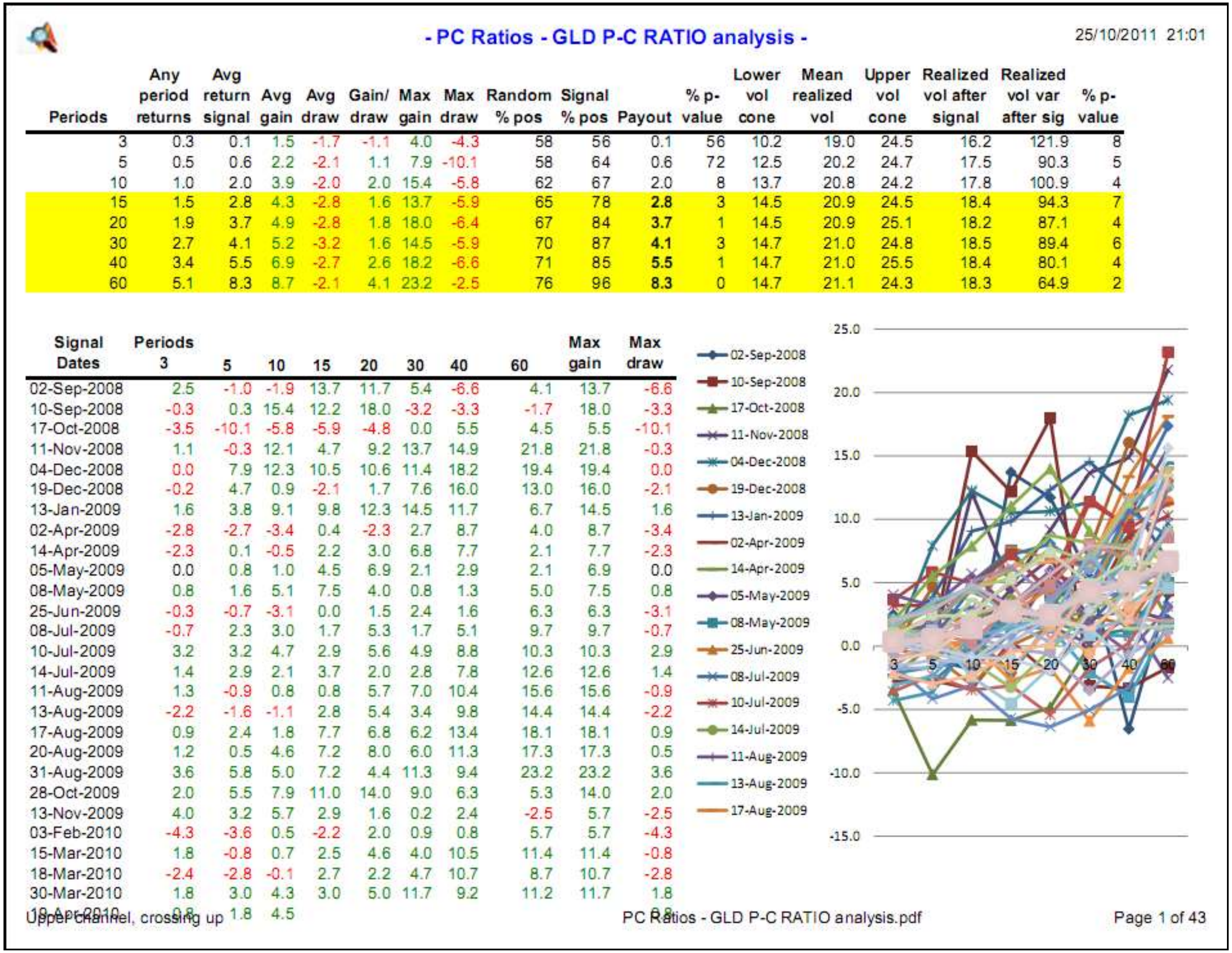
clos

☐ Log scale
☑ Datatips

Comment

04/06/08

Sample PDF report



- PC Ratios - GLD P-C RATIO analysis -

25/10/2011 21:01

Periods	Any period returns	Avg return signal	Avg gain	Avg draw	Gain/ draw	Max gain	Max draw	Random % pos	Signal % pos	Payout	% p-value	Lower vol cone	Mean realized vol	Upper vol cone	Realized vol after signal	Realized vol var after sig	% p-value
3	0.3	0.1	1.5	-1.7	-1.1	4.0	-4.3	58	56	0.1	56	10.2	19.0	24.5	16.2	121.9	8
5	0.5	0.6	2.2	-2.1	1.1	7.9	-10.1	58	64	0.6	72	12.5	20.2	24.7	17.5	90.3	5
10	1.0	2.0	3.9	-2.0	2.0	15.4	-5.8	62	67	2.0	8	13.7	20.8	24.2	17.8	100.9	4
15	1.5	2.8	4.3	-2.8	1.6	13.7	-5.9	65	78	**2.8**	3	14.5	20.9	24.5	18.4	94.3	7
20	1.9	3.7	4.9	-2.8	1.8	18.0	-6.4	67	84	**3.7**	1	14.5	20.9	25.1	18.2	87.1	4
30	2.7	4.1	5.2	-3.2	1.6	14.5	-5.9	70	87	**4.1**	3	14.7	21.0	24.8	18.5	89.4	6
40	3.4	5.5	6.9	-2.7	2.6	18.2	-6.6	71	85	**5.5**	1	14.7	21.0	25.5	18.4	80.1	4
60	5.1	8.3	8.7	-2.1	4.1	23.2	-2.5	76	96	**8.3**	0	14.7	21.1	24.3	18.3	64.9	2

Signal	Periods								Max	Max
Dates	3	5	10	15	20	30	40	60	gain	draw
02-Sep-2008	2.5	-1.0	-1.9	13.7	11.7	5.4	-6.6	4.1	13.7	-6.6
10-Sep-2008	-0.3	0.3	15.4	12.2	18.0	-3.2	-3.3	-1.7	18.0	-3.3
17-Oct-2008	-3.5	-10.1	-5.8	-5.9	-4.8	0.0	5.5	4.5	5.5	-10.1
11-Nov-2008	1.1	-0.3	12.1	4.7	9.2	13.7	14.9	21.8	21.8	-0.3
04-Dec-2008	0.0	7.9	12.3	10.5	10.6	11.4	18.2	19.4	19.4	0.0
19-Dec-2008	-0.2	4.7	0.9	-2.1	1.7	7.6	16.0	13.0	16.0	-2.1
13-Jan-2009	1.6	3.8	9.1	9.8	12.3	14.5	11.7	6.7	14.5	1.6
02-Apr-2009	-2.8	-2.7	-3.4	0.4	-2.3	2.7	8.7	4.0	8.7	-3.4
14-Apr-2009	-2.3	0.1	-0.5	2.2	3.0	6.8	7.7	2.1	7.7	-2.3
05-May-2009	0.0	0.8	1.0	4.5	6.9	2.1	2.9	2.1	6.9	0.0
08-May-2009	0.8	1.6	5.1	7.5	4.0	0.8	1.3	5.0	7.5	0.8
25-Jun-2009	-0.3	-0.7	-3.1	0.0	1.5	2.4	1.6	6.3	6.3	-3.1
08-Jul-2009	-0.7	2.3	3.0	1.7	5.3	1.7	5.1	9.7	9.7	-0.7
10-Jul-2009	3.2	3.2	4.7	2.9	5.6	4.9	8.8	10.3	10.3	2.9
14-Jul-2009	1.4	2.9	2.1	3.7	2.0	2.8	7.8	12.6	12.6	1.4
11-Aug-2009	1.3	-0.9	0.8	0.8	5.7	7.0	10.4	15.6	15.6	-0.9
13-Aug-2009	-2.2	-1.6	-1.1	2.8	5.4	3.4	9.8	14.4	14.4	-2.2
17-Aug-2009	0.9	2.4	1.8	7.7	6.8	6.2	13.4	18.1	18.1	0.9
20-Aug-2009	1.2	0.5	4.6	7.2	8.0	6.0	11.3	17.3	17.3	0.5
31-Aug-2009	3.6	5.8	5.0	7.2	4.4	11.3	9.4	23.2	23.2	3.6
28-Oct-2009	2.0	5.5	7.9	11.0	14.0	9.0	6.3	5.3	14.0	2.0
13-Nov-2009	4.0	3.2	5.7	2.9	1.6	0.2	2.4	-2.5	5.7	-2.5
03-Feb-2010	-4.3	-3.6	0.5	-2.2	2.0	0.9	0.8	5.7	5.7	-4.3
15-Mar-2010	1.8	-0.8	0.7	2.5	4.6	4.0	10.5	11.4	11.4	-0.8
18-Mar-2010	-2.4	-2.8	-0.1	2.7	2.2	4.7	10.7	8.7	10.7	-2.8
30-Mar-2010	1.8	3.0	4.3	3.0	5.0	11.7	9.2	11.2	11.7	1.8
19-Apr-2010	0.8	1.8	4.5							

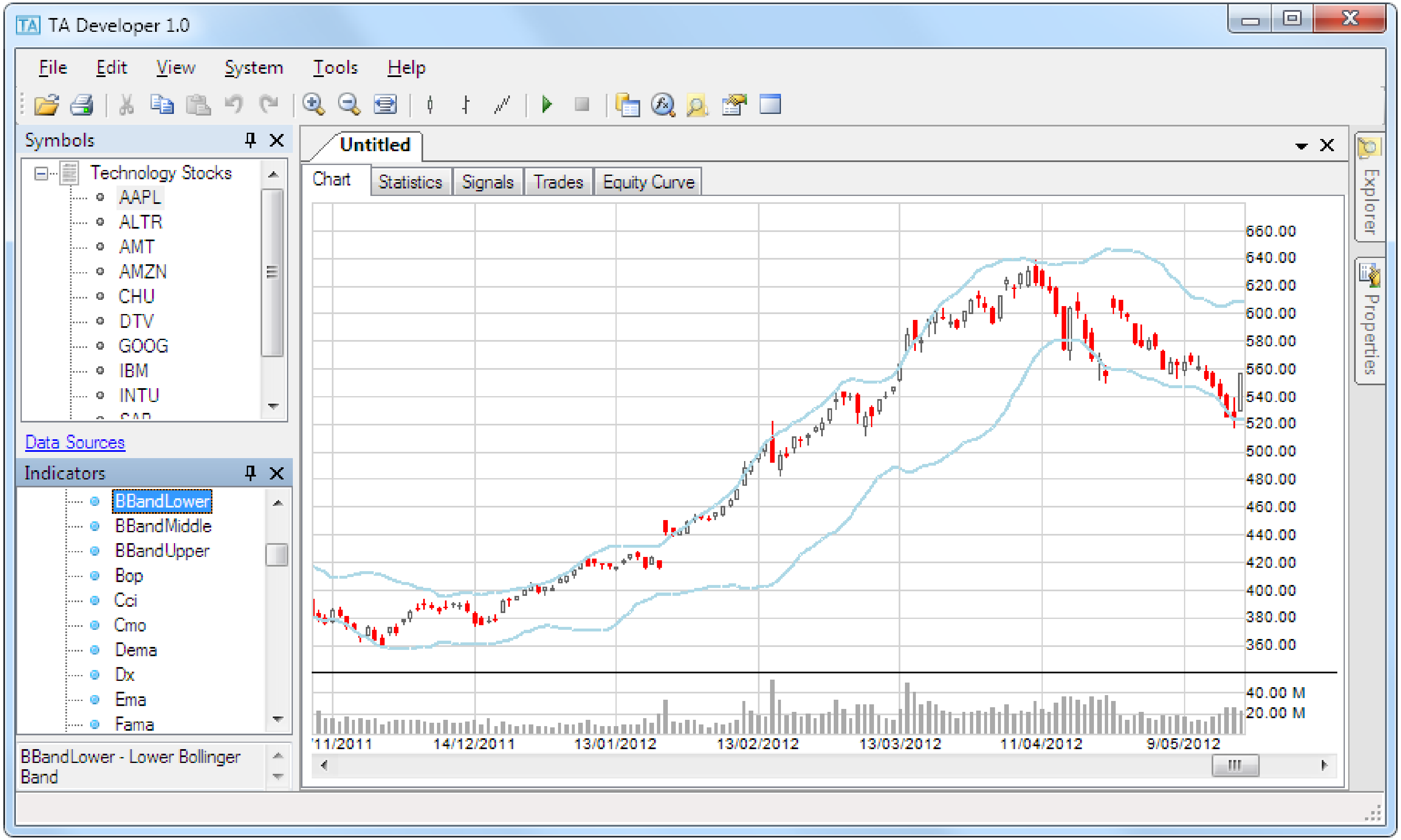
Upper channel, crossing up

PC Ratios - GLD P-C RATIO analysis.pdf

Page 1 of 43

Backtesting in MATLAB

tadeveloper.com



Conclusion

- Technology is no longer a barrier to developing a relatively low-cost algorithmic trading engine
- We no longer need to handle connectivity plumbing
- We no longer need to prototype in MATLAB, deploy in C
  - MATLAB can handle entire investment management lifecycle
  - Simpler development, reduced cost & time-to-market
- The only major thing left to do is "just" to devise a winning strategy
  - With the analysis tools available in MATLAB this should be easier than ever

Resources

mathworks.com/products/trading

undocumentedmatlab.com/ib-matlab

interactivebrokers.com/en/software/ibapi.php

altmany@gmail.com