

# Quantitative Investment: Research and Implementation in MATLAB

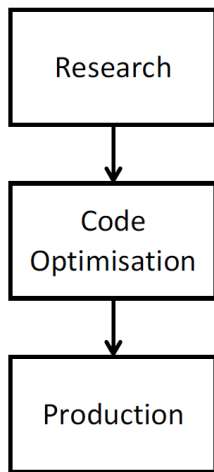
Edward Hoyle

Fulcrum Asset Management  
6 Chesterfield Gardens  
London, W1J 5BQ

`ed.hoyle@fulcrumasset.com`

24 June 2014  
MATLAB Computational Finance Conference 2014  
Etc Venues, St Paul's, London

# Trading Strategy Workflow



## 1 Research

- ▶ Identification of returns source and robust modelling
- ▶ Robust testing of trading signals
- ▶ Possibly multiple data sources
- ▶ MATLAB: Scripting, charting, reporting

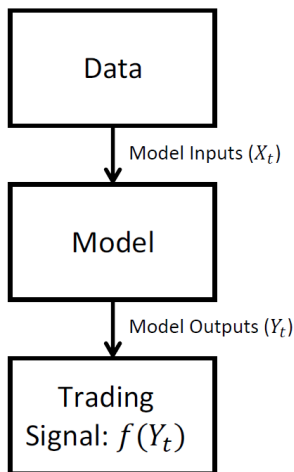
## 2 Code Optimisation

- ▶ Speed improvements
- ▶ Identification and removal of unnecessary processing or data querying
- ▶ MATLAB: Documentation, practice/experience

## 3 Production

- ▶ Set of guidelines for production code
- ▶ Central data source
- ▶ MATLAB: Functions, OO

# Generic Trading Strategy



## 1 Data

- ▶ Historical and real-time
- ▶ Financial, economic, sentiment, etc.

## 2 Model

- ▶ Simplification of reality which captures certain important features
- ▶ E.g. use inflation and growth to forecast bond yields
- ▶ E.g. high-yielding currencies appreciate against low-yielding currencies
- ▶ E.g. asset-price trends are persistent

## 3 Trading signal

- ▶ Convert model outputs into trading positions
- ▶ Which assets to buy and sell, and in what quantities
- ▶ Position sizing may be dependent on risk or regulatory constraints

# Research

## Starting a Project

### 1 Idea generation

- ▶ 'In house'
- ▶ From literature (broker or academic research)
- ▶ Anecdotal evidence
  - ★ 'Buy/sell Bund futures following declines/rises in the flash manufacturing PMI'
  - ★ Some technical analysis (e.g. support/resistance levels)

### 2 Model specification

- ▶ What are the required outputs?
- ▶ What should be input?
- ▶ Can the model be simplified without compromising on its most important features?

### 3 Data gathering

- ▶ Central banks
- ▶ Financial data providers
- ▶ Websites
- ▶ Brokers

# Personal Modelling Philosophy

## A Digression

I learnt this the hard way:

*If a simple trading model can be shown to work, a complicated model **may** improve results.*

*If no simple trading model can be shown to work, no complicated model will improve results.*

This does not mean that building successful trading models is easy. As is typical in research, 90% of the work is finding the right questions to ask; the other 10% is finding answers.

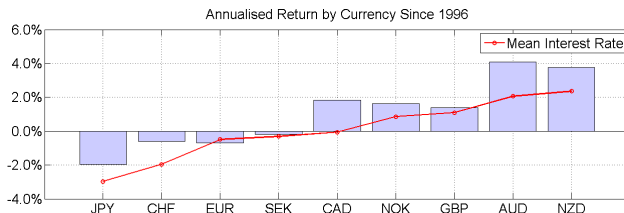
### Robustness checks

- Sensitivity of model output to parameters
- Stability of parameters
  - ▶ Do different fitting methods produce similar results?
  - ▶ How sensitive are parameters to the data sample?
- Are we sure there is no peek-ahead bias?
  - ▶ This is particular concern for models using macroeconomic variables
  - ▶ Have historical data been adjusted retrospectively?
  - ▶ Have data release delays been accounted for?
  - ▶ Have differing market trading hours (S&P versus Nikkei) been accounted for?

# Research

## Identifying Trading Signals

- Demonstrate that a trading signal has predictive power
  - ▶ This is often quite hard!
  - ▶ A successful backtest alone does not usually convince me
  - ▶ Charts can be very useful here



- Robustness checks
  - ▶ Do the signals have a directional bias? Should they?
  - ▶ Are signals effective after accounting for market trends?
  - ▶ Are both the long and short signals profitable?
  - ▶ Does a stronger signal have stronger forecasting power? Should it?

# Research

## Trading Strategy

- Backtesting
  - ▶ Drawdowns
  - ▶ Performance metrics
  - ▶ Persistence (across markets, across subsamples)
- Dependencies
  - ▶ Correlations with other strategies
- Robustness
  - ▶ Sensitivity to parameters
  - ▶ Regime analysis
    - ★ rising/falling interest rates
    - ★ economic growth/recession
    - ★ bull/bear market
- Adding to a broader portfolio
  - ▶ Effect on the portfolio's leverage?
  - ▶ Effect on the portfolio's risk allocation to assets and asset classes?
  - ▶ Effect on the portfolio's volatility/VaR/expected shortfall?



# Code Optimisation

## Code optimisation by example

- A real-world code-optimisation problem
- The code solved a constrained minimisation problem
- I shall show some of the steps taken to improve the efficiency and reliability of the solver
- A particular concern was to ensure that the solution was not sensitive to the starting point of the minimisation
- Speed was an important consideration here, but was secondary to robustness
- Note: The Parallel Computing Toolbox was not used when producing the following results (although it would probably have sped things up)

# Code Optimisation

## An Example

This is an optimisation problem that I encountered recently:

Define  $\|x\|_2^2 = \sum_i x_i^2$  and  $\|x\|_1 = \sum_i |x_i|$ , for  $x \in \mathbb{R}^n$ .

### The Problem

$$\begin{aligned} & \underset{x}{\text{minimize}} && \|Ax - b\|_2^2 + \lambda \|Cx\|_1, \\ & \text{subject to} && \|Dx - e\|_1 \leq f, \end{aligned}$$

where  $x \in \mathbb{R}^n$ ,  $A \in \mathbb{R}^{m_a \times n}$ ,  $b \in \mathbb{R}^{m_a}$ ,  $C \in \mathbb{R}^{m_c \times n}$ ,  $D \in \mathbb{R}^{m_d \times n}$ ,  $e \in \mathbb{R}^{m_d}$ ,  $f > 0$ ,  $\lambda > 0$ .

The objective function contains a quadratic term and a non-linear term.  
The constraint is non-linear.

# Code Optimisation

## First Attempt

- Don't work harder than you need to!
- Use `fmincon` (Optimization Toolbox)
- Very quick to set up the problem

# Code Optimisation

Performance: Naive `fmincon`

- 20 dimensional problem
- 100 different random starting points (not necessarily satisfying constraints)
- Timing = 1,144.92 seconds
- Objective: mean 2.1152, min 2.0951, max 2.1532

Table: Values for  $x_1, \dots, x_5$ .

$i$	Mean	Min	Max	SD
1	0.2932	0.2559	0.3120	0.0125
2	1.9193	1.7683	2.2189	0.1282
3	-0.7701	-1.5355	-0.2690	0.3528
4	4.5681	4.2161	4.8536	0.1180
5	0.5654	0.4741	0.7047	0.0464

# Code Optimisation

## Second Attempt

- We like the results from the first attempt
- Can we improve performance or robustness?
- Supply gradients for the objective and constraint using

```
options = optimoptions('fmincon'...  
    , 'GradObj', 'on', 'GradConstr', 'on')
```

$$\nabla \{ \|Dx - e\|_1 \} = D' \text{sign}(Dx - e)$$

$$\nabla \left\{ \|Ax - b\|_2^2 + \lambda \|Cx\|_1 \right\} = 2A'Ax - 2A'b + \lambda C' \text{sign}(Cx)$$

- Note: There are many other `fmincon` options that can be changed in the bid to improve performance

# Code Optimisation

Performance: `fmincon` with gradients

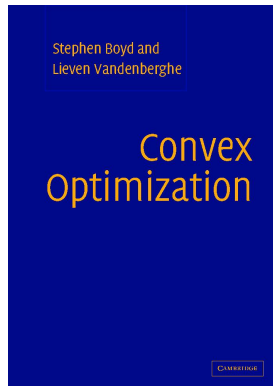
- 20 dimensional problem
- 100 different random starting points (not necessarily satisfying constraints)
- Timing = 96.04 seconds (92.1% speed up)
- Objective: mean 2.1153, min 2.0948, max 2.1865

Table: Values for  $x_1, \dots, x_5$ .

$i$	Mean	Min	Max	SD
1	0.2939	0.2559	0.3128	0.0126
2	1.9119	1.7768	2.2191	0.1255
3	-0.7636	-1.7491	-0.2658	0.3436
4	4.5720	4.2160	4.8536	0.1174
5	0.5659	0.4745	0.7047	0.0447

# Code Optimisation

Q. How Can We Improve Further? A. Do Some Research!



- Boyd & Vandenberghe
- Cambridge University Press
- Also available at <http://www.stanford.edu/~boyd/cvxbook/>

# Code Optimisation

## The Problem Revisited

Find an optimisation problem with the same optimal solution, but with a 'nicer' form:

### Equivalent Problem

$$\underset{x,u,t}{\text{minimize}} \quad x' A' A x - 2 A' b x + \lambda \mathbf{1}' t$$

$$\begin{aligned} \text{subject to} \quad & -t \preceq Cx \preceq t, \\ & -u \preceq Dx - e \preceq u, \\ & \mathbf{1}' u \leq f, \end{aligned}$$

where  $t \in \mathbb{R}_+^{m_c}$ ,  $u \in \mathbb{R}_+^{m_d}$ .

- The objective function is quadratic
- The constraints are linear
- We can use `quadprog` (Optimization Toolbox)



# Code Optimisation

Performance: quadprog

- 20 dimensional problem
- 100 different random starting points (not necessarily satisfying constraints)
- Timing = 1.34 seconds (99.9% speed up)
- Objective: mean 2.0041, min 2.0041, max 2.0041

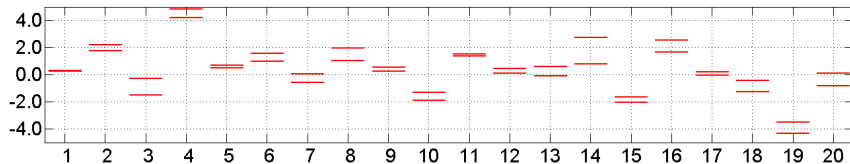
Table: Values for  $x_1, \dots, x_5$ .

$i$	Mean	Min	Max	SD
1	0.2917	0.2917	0.2917	0.0000
2	1.7458	1.7458	1.7458	0.0000
3	0.0141	0.0141	0.0141	0.0000
4	4.5358	4.5358	4.5358	0.0000
5	0.5095	0.5095	0.5095	0.0000

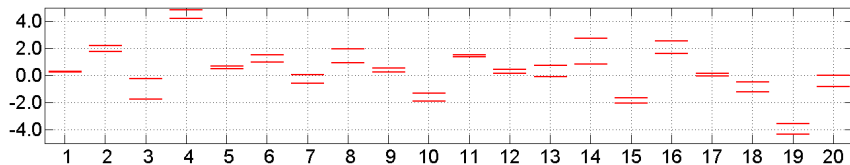
# Code Optimisation

Results: Range of values output for  $x_1, \dots, x_{20}$  after 100 optimisations.

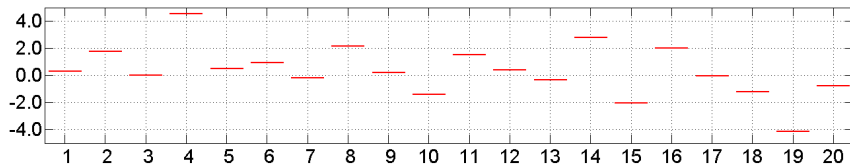
Naive FMINCON: max and min solutions



FMINCON with gradient: max and min solutions



QUADPROG: max and min solutions



# Implementation

## Case Study: Challenges Faced

- Not always a distinction between research and production code
  - ▶ Code difficult read and poorly commented
  - ▶ Scripts calling scripts (variables with unclear scope)
  - ▶ 'If' conditions in code which are never met
  - ▶ No centralised code library or version control
- Multiple data sources
  - ▶ Various databases, spread sheets, text files and MAT-files
  - ▶ Various update processes
- Strategies running on desktop computers
  - ▶ Dependencies on user profiles
  - ▶ Potential hardware risk (failing hard drives etc.)
- No API between trading strategies and trade-execution systems

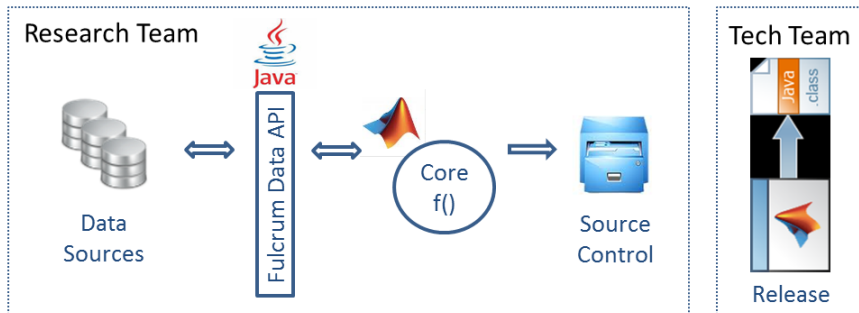
# Implementation

## Case Study: Writing Production Code

- Write production MATLAB code. Guidelines:
  - ▶ Strip down research code to bare bones
  - ▶ Remove commented-out code (unless there is a good reason otherwise)
  - ▶ Include **useful** comments
  - ▶ Use sensible variable names
  - ▶ Do not sacrifice readability for performance unless necessary
  - ▶ Standardise output
  - ▶ Use centralised data source
  - ▶ Include links to research (code and documentation)
  - ▶ Add to source control
- Export to Java (MATLAB Builder JA)

# Implementation

## Case Study: Production Code



# Implementation

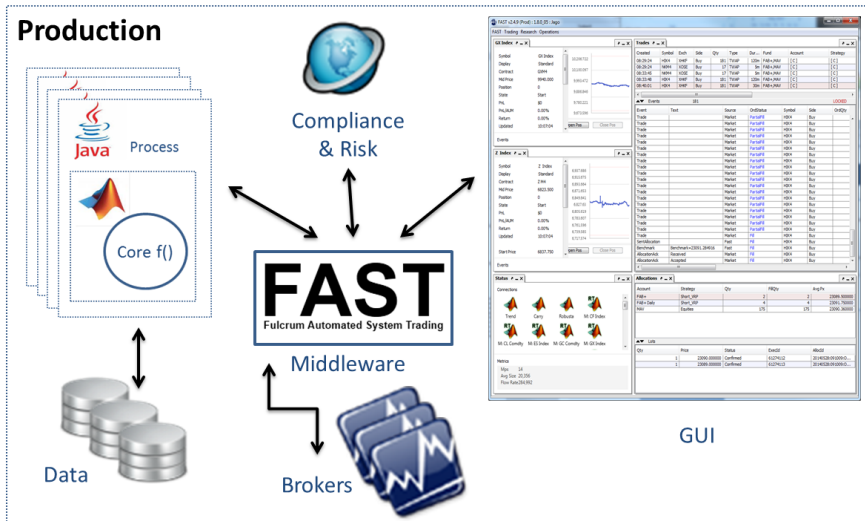
## Case Study: Production Environment

Java platform for the execution of medium-frequency trades:

- 1 Update central database
- 2 Run MATLAB strategies in parallel
- 3 Send strategy outputs to MATLAB trade aggregation and sizing routine
- 4 Display trades in GUI
- 5 Trades approved by PM using GUI
- 6 'High touch' trades sent to the traders' order manager
- 7 'Low touch' trades sent to the execution order manager
  - ▶ Orders are validated using the pre-trade compliance engine
  - ▶ Algorithmic execution parameters are determined
  - ▶ Orders are routed to the best broker

# Implementation

## Case Study: Production Environment



# MATLAB

## Why Use MATLAB in Production?

- Research is done using MATLAB, so we avoid having to rewrite code
  - ▶ Facilitates rapid deployment
  - ▶ Avoids errors in translation
- Retain access to high-level functions
- Able to run the trading strategies on a desktop using MATLAB for rapid debugging

Note that we are medium-frequency traders

- Low latency is not a requirement for our systems
- We measure run-times in seconds; we are indifferent to the milli, let alone the nano or pico!



# MATLAB

## Likes

What I like about MATLAB:

- Excellent user interface
- Thoroughly-tested high-level functionality
  - ▶ Statistics Toolbox
  - ▶ Optimization Toolbox
  - ▶ Database Toolbox
  - ▶ Datafeed Toolbox
  - ▶ Parallel Computing Toolbox
- Multi-paradigm environment (supports scripting, functions, objects, etc.)
- Good documentation
- Technical support