Algorithmic Trading with MATLAB®

Martin Demel, Application Engineer
Agenda

- Introducing MathWorks
- Introducing MATLAB (Portfolio Optimization Example)
- Introducing Algorithmic Trading with MATLAB

Break

- Credit Risk Modeling with MATLAB
- Risk Management using various VaR computation methods
- Overview of derivatives pricing capabilities and further financial computing products
- Q&A
RWE Develops and Deploys an Automated System for Natural Gas and Power Trading and Risk Management

Challenge
Automate business processes for quoting gas contracts and hedging against price fluctuations

Solution
Engage MathWorks Consulting to develop and deploy to a production environment an automated pricing and risk management system that fits within the company’s existing IT infrastructure

Results
- Models created in minutes, not weeks
- 100% accurate results delivered
- Technical expertise applied to core business goals

“MathWorks consultants were well-qualified, professional, and fast. They understood not only the technical issues but also the business goals, which is essential when working on a core business system. We got more than we expected from MathWorks Consulting.”

Dr. Norbert Tönder
RWE
Challenges when building trading strategies

- Increasing complexity
  - More data
  - More complicated models
- Increasing computational speed
  - Push to higher frequency
- Long deployment cycle
  - (Re)coding is costly and error-prone
Agenda

- Introduction: Algorithmic trading
- Developing an automated trading decision engine
  - Identify a successful trading rule
  - Extend trading rule set
  - Automate trading rule selection
- Implementing MATLAB into your production trading environment
- Wrap up and Q&A
The problem at hand: Identifying profitable trading strategies

- Commodities analyst
- Developing a trading strategy
  - Multiple trading rules
  - High frequency
- Management requirements:
  - Tested on historical data
  - Uses sophisticated analytics to identify optimal trading rule combination
  - Integrates with existing data and execution APIs
Trading decision engine

Development and testing

Historical Data
- End of Day / Intraday
- Files
- Databases

Strategy Modeling
- Research / Algorithms
- Model Development
- Calibration

Back Testing
- Profit / Loss
- Risk Exposure

Implementation

Live Data
- Real-Time Feeds
- Event-Based

Decision Engine
- Models
- Trading Rules

Execution
- Broker API
- Order Routing
Requirements for the trading engine

- Sophisticated analytics
  - Custom rules & indicators
  - Non-traditional techniques

- Scalable speed
  - Higher frequency data
  - More trading rules

- Quick to develop and deploy
  - Try different strategies
  - Embed in trading engine
Trading decision engine

Goal:
Task 1: Build a back testing environment around historical data and a preliminary trading rule
Task 2: Move to a higher frequency (minute-by-minute) and re-calibrate the model
Task 3: Develop a rule selection system for instruments using evolutionary learning

Development and testing

Historical
- End of Day / Intraday Files
- Databases

Strategy
- Research / Algorithms
- Model Development
- Calibration

Back Testing
- Profit / Loss
- Risk Exposure

Implementation

Live Data
- Real-Time Feeds
- Event-Based

Decision Engine
- Models
- Trading Rules

Execution
- Broker API
- Order Routing
Task 1: Develop a back testing environment

Goal: Build a back testing environment around historical data and a preliminary trading rule

Development and testing

- **Historical Data**
  - End of Day
  - Intraday
  - Files
  - Databases

- **Strategy Modeling**
  - Research / Algorithms
  - Model Development
  - Calibration

- **Back Testing**
  - Profit / Loss
  - Risk Exposure

 Implementation

- **Live Data**
  - Real-Time Feeds
  - Event-Based

- **Decision Engine**
  - Models
  - Trading Rules

- **Execution**
  - Broker API
  - Order Routing
Task 1: Develop a back testing environment

Key tasks
- Import data from files
- Create a preliminary rule
- Test the rule’s performance

Solutions
- MATLAB data tools
- High-level programming and pre-built functions
- Powerful graphics environment
Task 2: Expand the scale of the engine

Goal: Move to a higher frequency (minute-by-minute) and re-calibrate the model

Development and testing

Implementation

Historical Data
- End of Day
- Intraday
- Files
- Databases

Strategy Modeling
- Research / Algorithms
- Model Development
- Calibration

Decision Engine
- Models
- Trading Rules

Back Testing
- Profit / Loss
- Risk Exposure

Execution
- Broker API
- Order Routing

Live Data
- Real-Time Feeds
- Event-Based
Task 2: Expand the scale of the engine

Key tasks
- Importing data from databases
- Increase computational speed

Solutions
- MATLAB data tools: Database Toolbox
- High-performance computing: Parallel Computing Toolbox, MATLAB Distributed Computing Server
Task 3: Rule selection engine

Goal: Develop a rule selection system for instruments using evolutionary learning

Development and testing

Historical Data
- End of Day
- Intraday
- Files
- Databases

Strategy Modeling
- Research / Algorithms
- Model Development
- Calibration

Back Testing
- Profit / Loss
- Risk Exposure

Implementation

Live Data
- Real-Time Feeds
- Event-Based

Decision Engine
- Models
- Trading Rules

Execution
- Broker API
- Order Routing
Task 3: Rule selection engine

Key tasks

- Increase number of rules
- Incorporate advanced analytics to select best combination
Working with multiple strategies

Working with multiple strategies

- Represent different combinations as *bit strings*

```
Signal 1  AND  Signal 2  OR  Signal 3
1  0  0  1  0  1  1  1  1  1
```

Signals Active?
Building Custom Evolution Algorithms

- **Selection**
  - *Retain* the best performing bit strings from one generation to the next. *Favor these for reproduction*

- **Crossover**
  - parent1 = [1 0 1 0 0 1 1 0 0 0]
  - parent2 = [1 0 0 1 0 0 1 0 1 0]
  - child = [1 0 0 0 0 1 1 0 1 0]

- **Mutation**
  - parent = [1 0 1 0 0 1 1 0 0 0]
  - child = [0 1 0 1 0 1 0 0 0 1]
Task 3: Rule selection engine

Key tasks
- Increase number of rules
- Incorporate advanced analytics to select best combination

Solutions
- High-level programming
- MATLAB Toolboxes: Global Optimization, …
Agenda

- Introduction: Algorithmic trading
- Developing an automated trading decision engine
  - Identify a successful trading rule
  - Extend trading rule set
  - Automate trading rule selection
- Implementing MATLAB into your production trading environment
- Wrap up and Q&A
Implementing the Decision Engine

Goal: Evaluate and test the decision engine with real-time feeds and execution through a messaging bus

Development and testing

- **Historical Data**
  - End of Day / Intraday
  - Files
  - Databases

- **Strategy Modeling**
  - Research / Algorithms
  - Model Development
  - Calibration

- **Back Testing**
  - Profit / Loss
  - Risk Exposure

Implementation

- **Live Data**
  - Real-Time Feeds
  - Event-Based

- **Decision Engine**
  - Models
  - Trading Rules

- **Execution**
  - Broker API
  - Order Routing
Key Tasks

Key tasks

- Read live market data from data feed
- Connect to trading “engine”

Solutions

- Datafeed Toolbox
- Many external APIs
  - .NET, Java, C/C++, etc.
  - 3rd party APIs
Deploying Applications with MATLAB

- Give MATLAB code to other users

- Share applications with end users who do not need MATLAB
  - Stand-alone executables
  - Shared libraries
  - Software components
Review: Requirements for the trading engine

- Sophisticated analytics
  - Custom rules & indicators
  - Non-traditional techniques

- Scalable speed
  - Higher frequency data
  - More trading rules

- Quick to develop and deploy
  - Try different strategies
  - Embed in trading engine
MATLAB’s solutions

- **Sophisticated analytics**
  - Advanced graphics environment
  - Toolboxes give access to hundreds of new techniques
  - Flexible and customizable

- **Scalable speed**
  - Parallel computing solution

- **Quick to develop and deploy**
  - High-level programming
  - Automated deployment