

# Integrating QuantLib and *Mathematica*

J. Robert Buchanan

Department of Mathematics

November 22, 2008

- Develop tools for students to explore properties of options and derivatives.
- Create an open source, high-level programming environment for rapid prototyping and implementation of financial instruments.
- Provide opportunities for undergraduate research projects in financial mathematics.
- Similar projects are available, but are commercial (*e.g.* Derivatives Expert (URL: [www.ifs.dk/DerivativesExpert/](http://www.ifs.dk/DerivativesExpert/)) , UnRisk (URL: [www.unrisk.com](http://www.unrisk.com)), *etc.*).

# Software Components

- QuantLib, an open source, C++ library of code for financial instruments, pricing engines, and numerical routines (URL: [www.quantlib.org](http://www.quantlib.org)).
- *Mathematica*, a commercial computer algebra system and tool for doing mathematics (URL: [www.wolfram.com](http://www.wolfram.com)).
- *MathLink*, *Mathematica*'s communication protocol for dynamically loading externally compiled executable functions.

# Adding an External Function to *Mathematica*

- 1 Create source code for external function (C/C++/FORTRAN/Java).
- 2 Compile source code with an interface “stub” which specifies how *Mathematica* expressions will be communicated to the external function.
- 3 Load the external routine into a running *Mathematica* kernel.
- 4 Evaluate the function as if it were a native function.

- Conflicting data structures between QuantLib and *Mathematica*.
  - QuantLib has an extensive object-oriented library.
  - *Mathematica*'s basic data structure is the list, functions are evaluated by matching patterns and applying rules.
- Limited support for date and calendar calculations in *Mathematica*.

# Dates and Calendars

- QuantLib supports calendars for a collection of stock exchanges.
  - Dates can be tested for being business days.
  - Dates can be rolled to the nearest business day.
  - Date arithmetic can be performed.
- *Mathematica* supports a date list data structure and date arithmetic, but no international holiday calendars.

# Testing for Business Days

```
In[158]:= Options[BusinessDayQ]
Out[158]= {Calendar -> UnitedStates, JointCalendarConvention -> JoinHolidays}

In[159]:= BusinessDayQ[{2008, 5, 1}, "Calendar" -> "Mexico"]
Out[159]= False

In[160]:= BusinessDayQ[{2008, 8, 5},
    "Calendar" -> {"Canada", "UnitedStates", "Mexico"}]
Out[160]= True
```

# Rolling to Nearest Business Day

```
In[161]:= Options[RollDate]
```

```
Out[161]:= {Calendar -> UnitedStates, JointCalendarConvention -> JoinHolidays,  
           BusinessDayConvention -> Unadjusted}
```

```
In[162]:= RollDate[{2008, 5, 26}, "BusinessDayConvention" -> "Preceding"]
```

```
Out[162]:= {2008, 5, 23}
```

```
In[163]:= RollDate[{2008, 5, 26}, "BusinessDayConvention" -> "Following",  
           "Calendar" -> {"Mexico", "UnitedStates"},  
           "JointCalendarConvention" -> "JoinHolidays"]
```

```
Out[163]:= {2008, 5, 27}
```



# Shifting Dates

```
In[210]:= Options[ShiftDate]
Out[210]= {Calendar -> UnitedStates, JointCalendarConvention -> JoinHolidays}

In[211]:= ShiftDate[{2008, 5, 27}, {1, 1, 1}]
Out[211]= {2009, 6, 30}

In[212]:= ShiftDate["May 27, 2008", {-1, 1, -1}, "Calendar" -> "Finland"]
Out[212]= {2007, 6, 27}
```

# Counting Days/Years Between Dates

```
In[191]:= BusinessDaysBetween[{2008, 5, 27}, {2018, 5, 30}, "Calendar" → "Brazil"]
```

```
Out[191]= 2517
```

```
In[192]:= BusinessDaysBetween[{2008, 5, 27}, {2018, 5, 30}, "Calendar" → "Argentina"]
```

```
Out[192]= 2511
```

```
In[193]:= BusinessYearsBetween[{2007, 5, 27}, {2008, 5, 30},  
    "DayCounter" → "Business252"]
```

```
Out[193]= 0.996032
```

```
In[194]:= BusinessYearsBetween[{2007, 5, 27}, {2008, 5, 30},  
    "DayCounter" → "ActualActual"]
```

```
Out[194]= 1.00984
```

# Options and Derivatives

- Quantlib implements a variety of financial instruments and pricing engines for valuing them.
- Financial instruments possessing a closed form, analytic pricing formula can be programmed directly in the *Mathematica* language.
- Financial instruments requiring a numerical scheme to price could also be programmed in *Mathematica*, but this duplicates work already done in QuantLib.

# Multilevel Interfaces to QuantLib Routines

- Low level “command line interface” where all variables and parameters of an instrument must be specified.
- Medium level interface provides default values for some optional parameters.
- High level interface provides a form-based GUI to QuantLib routines.

# Low and Medium Level Interface Examples

```
In[20]= qlVanillaOption[3, 2, 81.0, {80.0, 0.0, 0.050, 0.20}, {35}, 1, 4,  
  {{2008, 11, 18}, {2008, 11, 20}, {2009, 11, 20}}]
```

```
Out[20]= {Value → 4.87931, Delta → -0.387344, Gamma → 0.0240286,  
  Theta → -1.28231, Vega → 30.5125, Rho → -35.5822, DividendRho → 30.7416}
```

```
In[21]= VanillaOption["European", "Put", 81, BlackScholesModel[80, 0.05, 0.20],  
  Table[DatePlus[DateList[], n], {n, {0, 2, 365}}]]
```

```
Out[21]= {Value → 4.88235, Delta → -0.387155, Gamma → 0.0239964,  
  Theta → -1.2788, Vega → 30.5447, Rho → -35.6556, DividendRho → 30.8004}
```

# High Level Example

In[22]:= **VanillaOptionGUI** []

The screenshot shows a software interface for configuring a Vanilla Option. It is divided into three main sections: Financial Instrument, Black-Scholes-Merton Model Parameters, and Variables. The Financial Instrument section includes dropdown menus for Exercise (American, Bermudan, European), Payoff (Call, Put), Joint Calendar Convention (JoinHolidays, JoinBusinessDays), and Day Counter (SimpleDayCounter). The Black-Scholes-Merton Model Parameters section has input fields for Underlying (100), Dividend Yield (0), Risk Free Rate (0.05), and Volatility (0.2). The Variables section has input fields for Strike (100), Today's Date ("18 Nov 2008"), Settlement Date ("20 Nov 2008"), and Expiry ("18 Nov 2009"). To the right of the input fields, a large text area displays the output of the calculation: {Value → 5.56429, Delta → -0.363535, Gamma → 0.0188208, Theta → -1.66826, Vega → 37.4324, Rho → -41.6849, DividendRho → 36.1515}.

**Financial Instrument**

Exercise: American, Bermudan, **European**

Payoff: Call, **Put**

Calendar: ▶

Joint Calendar Convention: **JoinHolidays**, JoinBusinessDays

Day Counter: **SimpleDayCounter**

**Black-Scholes-Merton Model Parameters**

Underlying: 100.

Dividend Yield: 0.

Risk Free Rate: 0.05

Volatility: 0.2

**Variables**

Strike: 100.

Today's Date: "18 Nov 2008"

Settlement Date: "20 Nov 2008"

Expiry: "18 Nov 2009"

{Value → 5.56429,  
Delta → -0.363535,  
Gamma → 0.0188208,  
Theta → -1.66826,  
Vega → 37.4324, Rho → -41.6849,  
DividendRho → 36.1515}

out[22]=

# Barrier Option Example

In[24]:= **BarrierOptionGUI** []

Out[24]=

**Barrier Option**

Barrier Type  DownIn  DownOut  UpIn  UpOut

Payoff  Call  Put

Calendar ▶

Joint Calendar Convention  JoinHolidays  JoinBusinessDays

Day Counter

---

**Black-Scholes-Merton Model Parameters**

Underlying

Dividend Yield

Risk Free Rate

Volatility

---

**Variables**

Strike

Barrier

Rebate

Today's Date

Settlement Date

Expiry

{Value → 0.920852}

# Design Questions

- Which interface level is the most useful?
- Would multiple return values be useful?