reposit Version 1.8

and

The Future of Spreadsheet Addins
# Table of Contents

1. The reposiT Project
   1.1 object repository
   1.2 SWIG module
   1.3 An example - QuantLib::Index
   1.4 Autogeneration of Documentation
   1.5 Development status
   1.6 TODO list

2. The Future of Spreadsheet Addins
   2.1 Defining characteristics of QuantLibXL
   2.2 Office 365
   2.3 Office Addins
   2.4 SharePoint
   2.5 Power BI
   2.6 Google Drive
   2.7 LibreOffice Online (LOOL)

3. Conclusions
namespace QuantLib {

    //! Abstract instrument class
    /*! This class is purely abstract and defines the interface of concrete
     * instruments which will be derived from this one.
     */
    class Instrument {
    public:
        /*! returns the net present value of the instrument.
         * Real NPV() const;
        */
    
    
        //! Vanilla option (no discrete dividends, no barriers) on a single asset
    class VanillaOption : public Instrument {
            public:
                    VanillaOption(const boost::shared_ptr<StrikedTypePayoff>&,  
                                    const boost::shared_ptr<Exercise>&);
    
    
    
    };

    }

    How would you export that functionality to Excel?
    How do you call a C++ constructor from Excel?
    After you create the object, where do you store it?
reposit Object Repository

**QuantLib::Instrument**

Real NPV() const;

**QuantLib::VanillaOption**

VanillaOption(  
    const shared_ptr<StrikedTypePayoff>&,  
    const shared_ptr<Exercise>&);

**Excel worksheet functions:**

- `qlVanillaOption()`: construct an object of type QuantLib::VanillaOption, wrap it in a QuantLibAddin::VanillaOption, and store it in the object cache.

- `qlInstrumentNPV()`: retrieve an object from the repository, downcast it to a QuantLibAddin::Instrument, invoke member function NPV on the contained QuantLib::Instrument, and return the result to Excel.
reposit Object Repository

Function Arguments

qVanillaOption

ObjectId: " european_option"
Payoff: B23
Exercise: B18
Permanent: 
Trigger: 

Construct an object of class VanillaOption and return its id.

Object Id: id of object to be created.

Function Arguments

qInstrumentNPV

ObjectId: " european_option#0000"
Trigger: B23

Returns the NPV for the given Instrument object.

Object Id: id of existing QuantLib::Instrument object.

Formula result =
namespace QuantLib {

    class Instrument {
        public:
            /*! returns the net present value of the instrument. */
            Real NPV() const;
    };

class VanillaOption : public Instrument {
    public:
        VanillaOption(const boost::shared_ptr<StrikedTypePayoff>&,
                     const boost::shared_ptr<Exercise>&);
    };
}

reposit SWIG module

XLL_DEC double *qlInstrumentNPV(char *ObjectId, OPER *Trigger);

XLL_DEC char *qlVanillaOption(char *ObjectId, char *Payoff, char *Exercise, OPER *Permanent, OPER *Trigger, bool *Overwrite);

The customized reposit SWIG module parses QuantLib header files and autogenerates QuantLibXL source code.
Supported Features

Object Repository

The repository supports a variety of features including:

- **Addins**: Bindings to Excel and C++ are provided, allowing you to implement a C++ program which replicates the behavior of your spreadsheet.
- **Inheritance**: Inheritance relationships in your C++ library are preserved in the object oriented interface that is exported to Excel.
- **Serialization**: Objects in the repository may be serialized, enabling you to save and load the state of your spreadsheet.
- **Conversions**: The library supports a wide variety of conversions between C++ and Excel data types.
- **Coercions**: You can configure coercions which allow the user to enter data of different types which are automatically converted into the target data type.
- **Enumerations**: Enumerations allow the user to enter strings which are mapped to C++ enumerated types or template classes.

SWIG Module

Autogeneration of addin source code is provided by a customized SWIG module. The code generation utility supports the following features:

- **Functions**: Simple functions in your C++ library are easily exported to target platforms.
- **Objects**: Classes in your C++ library are exported to the C++ and Excel addins. An autogenerated interface to these classes allows end users to construct objects, store them in the repository, and query and update them.
- **Inheritance**: Object hierarchies in your C++ library are recognized by the code generation utility and preserved in the functions in the Excel/C++ addins.
- **Serialization**: All of the code required to serialize your objects is generated automatically.
- ** Overrides**: A facility is provided to override the autogenerated code in case you want to customize it.
In some cases we want the native QuantLib functions, in some cases we want to override behavior in the QuantLibAddin namespace.
namespace QuantLib {

    class Index {
        public:
            //! Returns the fixing for the given Index object. The fixing is retrieved from the ...
            double fixing(
                        const Date& fixingDate,                          //!< fixing date(s).
                        bool forecastTodaysFixing                        //!< If set to TRUE it forces the forecasting ...
            );

            //! Returns the calendar (e.g. TARGET) for the given Index object.
            Calendar fixingCalendar();
    };

    class InterestRateIndex : public Index {
        public:
            //! Returns the fixing days (e.g. 2) for the given InterestRateIndex object.
            Natural fixingDays();

            %generate(c#, dayCounter);

            //! Returns the DayCounter (e.g. Actual/360) for the given InterestRateIndex object.
            const DayCounter& dayCounter();

            //! Returns the value date for the given fixing date for the given ...
            Date valueDate(const Date& fixingDate);

            //! Returns the tenor (i.e. length, e.g. 6M, 10Y) for the given ...
            Period tenor();
    };
}

Grab some functionality from the QuantLib namespace...
namespace QuantLibAddin {

class Index {
    public:
        // ! Adds fixings for the given Index object.
        void addFixings(
            const std::vector<QuantLib::Date>& dates,  // !< fixing dates.
            const std::vector<QuantLib::Real>& values, // !< fixing values.
            bool forceOverwrite                         // !< Set to TRUE to force overwriting...
        );
    }
};

class InterestRateIndex : public Index {};

class IborIndex : public InterestRateIndex {};

class SwapIndex : public InterestRateIndex {
    public:
        SwapIndex(
            const std::string& familyName,            // !< index name.
            const QuantLib::Period& p,               // !< index tenor...
            QuantLib::Natural fixingDays,            // !< swap rate fixing...
            QuantLib::Currency& crr,                 // !< Index Currency.
            const QuantLib::Calendar& calendar,      // !< holiday calendar...
            const QuantLib::Period& fixedLegTenor,   // !< tenor of the...
            QuantLib::BusinessDayConvention fixedLegBDC, // !< business day...
            const QuantLib::DayCounter& fixedLegDayCounter, // !< day counter of the...
            const boost::shared_ptr<QuantLib::IborIndex>& index, // !< swap's floating ibor..
            const QuantLib::Handle<QuantLib::YieldTermStructure>& disc // !< discounting...
        );
    }
};
namespace QuantLib {
  class Index {
    public:
      //! Returns the fixing for the given Index object. The fixing is retrieved from the ...
      double fixing(
        const Date& fixingDate,       //!< fixing date(s).
        bool forecastTodaysFixing    //!< If set to TRUE it forces the forecasting ...
      );

      //! Returns the calendar (e.g. TARGET) for the given Index object.
      Calendar fixingCalendar();
  };

  class InterestRateIndex : public Index {
    public:
      //! Returns the fixing days (e.g. 2) for the given InterestRateIndex object.
      Natural fixingDays();

      //! Returns the DayCounter (e.g. Actual/360) for the given InterestRateIndex object.
      const DayCounter& dayCounter();

      //! Returns the value date for the given fixing date for the given ...
      Date valueDate(const Date& fixingDate);

      //! Returns the tenor (i.e. length, e.g. 6M, 10Y) for the given ...
      Period tenor();
  };

  namespace QuantLibAddin {
    class Index {
      public:
        //! Adds fixings for the given Index object.
        void addFixings(
          const std::vector<QuantLib::Date>& dates,   //!< fixing dates.
          const std::vector<QuantLib::Real>& values,  //!< fixing values.
          bool forceOverwrite                        //!< Set to TRUE to force overwriting...
        );
    };

    class InterestRateIndex : public Index {
    };

    class IborIndex : public InterestRateIndex {
    };

    class SwapIndex : public InterestRateIndex {
      public:
        SwapIndex(  
          const std::string& familyName,        //!< index name.
          const QuantLib::Period& p,            //!< index tenor...
          QuantLib::Natural fixingDays,         //!< swap rate fixing...
          QuantLib::Currency& crr,              //!< Index Currency...
          const QuantLib::Calendar& calendar,   //!< holiday calendar...
          const QuantLib::Period fixedLegTenor, //!< tenor of the...
          QuantLib::BusinessDayConvention fixedLegBDC,  //!< business day...
          const QuantLib::DayCounters fixedLegDayCounter, //!< day counter of the...
          const boost::shared_ptr<QuantLib::IborIndex>& index,  //!< swap's floating ibor...
          const QuantLib::Handle<QuantLib::YieldTermStructure>& disc //!< discounting...
        );
    };
  }
};
QuantLibAddin 1.8

Related Pages

Here is a list of all related documentation pages:

Version History
Functions - All
Functions - Bonds
Functions - Calendars
Functions - Coupon Vectors
Functions - Date
Functions - Day Counters
Functions - Exercise
Functions - Handles
Functions - Indexes
Functions - Instruments
Functions - Interpolation
Functions - Leg
Functions - Math
Functions - Models
Functions - Payoffs
Functions - Piecewise Yield Curve
Functions - Prices
Functions - Pricing Engines
Functions - Processes
Functions - Quote
Functions - Quotes
Functions - Rate Helpers
Functions - Schedule
Functions - Settings
Functions - Term Structures
Functions - Utilities
Functions - Vanilla Swaps
Functions - Volatilities

qIndexAddFixings

```c
void returnValue
qIndexAddFixings(
    string objectID,
    vector<long> dates,
    vector<double> values,
    bool forceOverwrite,
    any Trigger
)
```

**Type:**
Member

**Description:**
Adds fixings for the given Index object.

**Supported Platforms:**
Excel

**Parameters**
- `objectID`: ID of Enumeration of class QuantLib::DayCounter
- `dates`: fixing dates.
- `values`: fixing values.
- `forceOverwrite`: Set to TRUE to force overwriting of existing fixings, if any.
- `Trigger`: Dependency tracking trigger
namespace QuantLib {

    //! purely virtual base class for indexes
    /*! \warning this class performs no check that the
        provided/requested fixings are for dates in the past,
        i.e. for dates less than or equal to the evaluation
        date. It is up to the client code to take care of
        possible inconsistencies due to "seeing in the
        future"
    */
    class Index : public Observable {
        public:
            virtual ~Index() {}  
            virtual std::string name() const = 0;
            ...
Development Status

reposit is a work in progress and is intended to replace an older build of QuantLibXL in which addin source code was autogenerated by a Python script.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Old Build</th>
<th>New Build</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Addin Functions Supported</td>
<td>1,080</td>
<td>250</td>
</tr>
<tr>
<td>Support for Rate Curve Framework</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Code Autogeneration:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object Wrappers</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Addin Functions</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Looping Functions</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Enumerations</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Documentation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Platforms Supported:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Excel</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LibreOffice Calc</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>C#</td>
<td>X</td>
<td>✓</td>
</tr>
</tbody>
</table>
## TODO List

<table>
<thead>
<tr>
<th>Task</th>
<th>Status</th>
<th>Required to Supercede Old Build</th>
</tr>
</thead>
<tbody>
<tr>
<td>remaining 750 functions</td>
<td>In Progress</td>
<td>Yes</td>
</tr>
<tr>
<td>enable support for the Excel Wizard</td>
<td>Pending</td>
<td>Yes</td>
</tr>
<tr>
<td>dynamic XLLs</td>
<td>Pending</td>
<td>Yes</td>
</tr>
<tr>
<td>add support for VC10-14</td>
<td>Pending</td>
<td>Yes</td>
</tr>
<tr>
<td>autogeneration of source code for enumerations</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>refactor conversion code and typemaps</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>re-enable the LibreOffice Spreadsheet addin</td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>
Table of Contents

1. The reposit Project
   1.1 object repository
   1.2 SWIG module
   1.3 An example - QuantLib::Index
   1.4 Autogeneration of Documentation
   1.5 Development status
   1.6 TODO list

2. The Future of Spreadsheet Addins
   2.1 Defining characteristics of QuantLibXL
   2.2 Office 365
   2.3 Office Addins
   2.4 SharePoint
   2.5 Power BI
   2.6 Google Drive
   2.7 LibreOffice Online (LOOL)

3. Conclusions
## History of Excel Versions

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Version Name</th>
<th>Release Year</th>
<th>Major Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Excel 2.0 (1987)</td>
<td>1987</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Excel 3.0 (1990)</td>
<td>1990</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Excel 4.0 (1992)</td>
<td>1992</td>
<td>C API (XLOPERs)</td>
</tr>
<tr>
<td>5</td>
<td>Excel 5.0 (1993)</td>
<td>1993</td>
<td>Excel Basic</td>
</tr>
<tr>
<td>7</td>
<td>Excel 95 (v7.0)</td>
<td>1995</td>
<td>32-bit</td>
</tr>
<tr>
<td>8</td>
<td>Excel 97 (v8.0)</td>
<td>1997</td>
<td>Excel VBA</td>
</tr>
<tr>
<td>9</td>
<td>Excel 2000 (v9.0)</td>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Excel 2002 (v10.0)</td>
<td>2002</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Excel 2003 (v11.0)</td>
<td>2003</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Excel 2007 (v12.0)</td>
<td>2007</td>
<td>big grid, multithreading, ribbon</td>
</tr>
<tr>
<td>14</td>
<td>Excel 2010 (v14.0)</td>
<td>2010</td>
<td>64-bit</td>
</tr>
<tr>
<td>15</td>
<td>Excel 2013 (v15.0)</td>
<td>2013</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Excel 2016 (v16.0)</td>
<td>2016</td>
<td></td>
</tr>
</tbody>
</table>
# History of QuantLibXL Releases

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Release Month &amp; Year</th>
<th>Excel Feature Support</th>
<th>Calc Addin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3.10</td>
<td>Jul 2005</td>
<td></td>
<td>Eric Ehlers</td>
</tr>
<tr>
<td>0.3.11</td>
<td>Oct 2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3.12</td>
<td>Mar 2006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3.13</td>
<td>Aug 2006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3.14</td>
<td>Dec 2006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4.0</td>
<td>Feb 2007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8.0</td>
<td>May 2007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9.0</td>
<td>Jan 2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9.6</td>
<td>Sep 2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9.7</td>
<td>Nov 2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0.1</td>
<td>Oct 2010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1.0</td>
<td>May 2011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2.0</td>
<td>Jul 2012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4.0</td>
<td>Jun 2014</td>
<td>64-bit (XLL only)</td>
<td></td>
</tr>
<tr>
<td>1.5.0</td>
<td>Apr 2015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6.0</td>
<td>Aug 2015</td>
<td></td>
<td>Lars Callenbach</td>
</tr>
<tr>
<td>1.7.0</td>
<td>Dec 2015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.8.0</td>
<td>??? 2016</td>
<td>64-bit also for VBA</td>
<td></td>
</tr>
</tbody>
</table>
The Future of QuantLibXL

QuantLibXL celebrates its 11\textsuperscript{th} birthday this month.
Where will the project be 11 years from now?

- Will spreadsheets still exist in 2027?
- Will the QuantLibXL project last that long?
- If the QuantLibXL does last another 11 years, will it still be recognizable then, or will it have to transform into something else?
Defining Features of QuantLibXL

As we take a tour of the next generation of spreadsheets, keep in mind the chief components of QuantLibXL:

- **UDFs**: User Defined Functions. `qlSwap()`, `qlInstrumentNPV()`, etc.

  *At present, the new generation of cloud-enabled spreadsheets do not support UDFs. This will probably have to change?*

- **Repository**: A cache of objects. Currently required in order to use an object-oriented library (such as QuantLib) in a functional environment (such as Excel).

  *In the new world of cloud-enabled spreadsheets, it won't make sense to cache objects in the memory of a local XLL.*

- **Macros**: The Rate Curve Framework is an application layer written around QuantLibXL, in Excel VBA.

  *Next generation spreadsheets offer very limited support for macros and instead provide other means for extending the UI e.g. apps and services.*
Office 365

Subscription based software-as-a-service providing rolling releases of an Office implementation that is cloud-enabled:

- OneDrive (cloud storage)
- Skype (VoIP)
- Exchange Server (calendar and mail)
- Office Online (see below)
- SharePoint (see below)
Office Online

Browser-based, lightweight implementation of Office applications. Supports real-time co-authoring of Office documents.

Excel Online does not support UDFs / macros.
Office Add-Ins

Customize Office using JavaScript, HTML, and CSS - add new functionality and content

- Targets both desktop Excel (2013 and 2016) and Excel Online
- Extend Excel UI
- Implement interactive content via HTML and JavaScript (jQuery, Angular, etc...)
- Connect to REST endpoints and consume web services
- Invoke Excel's JavaScript API

Office addins are hosted web services. They do not currently support Excel UDFs.
SharePoint

- Enterprise Content and Document Management
- Personal Cloud
- Intranet & Corporate Social Network
- Software Framework

SharePoint 2007 introduced Excel Services which facilitates multiuser access to spreadsheets in SharePoint Server.

SharePoint 2010 introduced the possibility to implement Excel UDFs using managed code in Excel Services.

SharePoint 2013 introduced the possibility to implement Excel UDFs in JavaScript for Excel Online.

SharePoint is one of the few platforms currently supporting UDFs for cloud-enabled spreadsheets. As a proprietary enterprise solution, SharePoint is not an ideal target platform for an open source project such as QuantLibXL.
For the sake of completeness we mention Power BI:

Microsoft proprietary business analytics service, including a series of addins for Excel capable of processing huge datasets (millions of rows)

- **Power Query**: retrieve data from selected sources & transform it
- **Power Pivot**: merge data from multiple sources for analysis in pivot tables
- **Power View**: provides interactive visualization of PowerPivot data models
- **Power Map**: geographic representation of data

This is more about mining big data than it is about quantitative finance, and if QuantLibXL were to fit in to the above puzzle it would be at a lower level.
How would you open a QuantLibXL spreadsheet in Google Drive?

Office documents can be opened on Google Drive:

- Using Office Compatibility Mode
- By converting Office documents to Google Drive format
- By installing the Google Drive plugin for Office

Supports Google Apps Script but not Excel macros/addins.

It would take a lot of development effort to get a QuantLibXL spreadsheet working on Google Drive.
Libre Office Online (LOOL)

A web server implementing an HTML5 UI which replicates the behavior of LibreOffice

- Promises complete fidelity between LibreOffice desktop and LibreOffice Online
- All Writer, Calc, and Impress supported file-types supported
- Open source project which can be hosted on private hardware

Offers another possibility for implementing the QuantLibAddin interface on the cloud.
Conclusions

- As of today, the desktop version of Excel (2016) continues to support the features required by QuantLibXL (UDFs, cache, macros).

- QuantLibXL will continue to target desktop Excel for as long as that platform remains viable.

- Cloud-enabled spreadsheets do not yet support a general mechanism for implementing UDFs.

- If one day Excel Online supports UDFs, QuantLibXL could be implemented as a web service targeting Excel Online and allowing existing QuantLibXL workbooks to migrate from desktop Excel to Excel Online.

- We would need to find a new solution for object cache and macros.