Model Calibration in MATLAB

Sam Bailey, PRUDENTIAL
Calibrating the Risk Scenarios

Need to calibrate statistical models for all the market risks we are exposed to — for example

- equity level
- equity volatility
- interest rate level
- interest rate volatility
- credit spreads
- defaults
- property
- FX

Collect market data
Transform
Statistical Fit
Current Calibration Process

Previously all done in spreadsheets
Worst case example – credit spread has total of 24 spreadsheets

Data collection – 6 different benchmarks
Data transforms – 6 steps
Calibration – 12 sheets (pre and post transition, separate simulation, 4 different benchmarks)

Calibration Report

Table Reordering…

Final output

Approximately 50 dependencies, several hundred links and copy/pastes
Key Issues

- Very error prone
  - Enormous number of manual steps
  - Large number of copy pastes. Large spreadsheets prone to crashing.

- Very time consuming
  - Approx 2 man months to run process end to end
Solution - Create a unified calibration system in MATLAB and .NET

- Data Presentation Layer
- Common Interface
- Data Cleaning
- Configure Calibration
- Apply Expert Judgement
- Calibrate
- Calibration Modelling Tool
- Report Generation Tool
- REPORT
- REPORT
- .NET Control GUI
- Calls MATLAB Library via .NET Builder
- Write Layer
- Database Controls Calibration Data and Versions
- HMDS
- IMS
- RSG
Data Presentation Layer

- Queries data from multiple sources (Market data database, Other e.g. spreadsheet)
- Creates a standard data object in MATLAB that the Calibration Tool can access
Data Upload to Input Management System and Report Generator

- Calibration Tool Outputs a Standard Calibration Object
  - Calibration Parameters
  - Distribution Percentiles
  - Fit Statistics (K-S test, Stationarity Test)

CALIBRATION TOOL

```matlab
[params, fit, emp, ks, percentiles] = calibrate(xp, distribution, method)
% [params, fit, emp] = calibrate(xp, distribution, method)
% fits a model of type 'distribution' to array of data 'xp' using 'method'
where

    [fx,x] = hist(xp(~isnan(xp)));
    for ii = 1:length(fx)
        Fx(ii) = sum(fx(1:ii));
    end
```

IMS WRITE

WRITES PARAMETERS AND PERCENTILES DIRECT TO IMS
BASELINES THE CAL SETUP SO CAN REPEAT PREVIOUS CALIBRATIONS

REPORT GENERATOR
Calibration Interface

- Configures Risk (GBP -> FTSE)
- Version Controls the Calibration
- Stores for Posterity – can reproduce any calibration
Generates calls to MATLAB via .NET builder

% CUR_equitytri = cal.equityCalibrate(RISK DRIVER, METHOD, TRANSFROM, PMGRISKPREMIUM, FIT, SKEWNESS AND KURTOSIS SOURCE (OPTIONAL))

GBP_equitytri = cal.equityCal('GBP_equitytri', 'Average', 'log', 3, 'Normal', PMGGroup1);

USD_equitytri = cal.equityCal('USD_equitytri', 'Concatanation', 'log', 3.25, 'BestFit', AllEconomies);

CNY_equitytri = cal.equityCal('TWD_equitytri', 'Concatanation', 'log', 3, 'BestFit', AllEconomies);

VND_equitytri = cal.equityCal(averageReturns({'TWD_equitytri', 'MYR Index'}), 'Individual', 'log', 3, 'BestFit');

THB_equitytri = forceFit(cal.equityCal('THB_equitytri', 'Individual', 'log', 3, 'BestFit'), 0.995, 0.45);

MYR_equitytri = scaleFit(cal.equityCal('MYR_equitytri', 'Individual', 'log', 3, 'BestFit'), 1.5);

% writeToIms(CAL OBJECT, CAL NAME, CAL SET)

writeToIms(CNY_equitytri, 'FY14', '1.1');
writeToIms(VND_equitytri, 'FY14', '1.1');
writeToIms(THB_equitytri, 'FY14', '1.1');
writeToIms(MYR_equitytri, 'FY14', '1.1');

% writeReport(CAL OBJECT, CAL NAME, CAL SET)

writeReport(GBP_equitytri, 'FY14', '1.1');
writeReport(USD_equitytri, 'FY14', '1.1');
writeReport(CNY_equitytri, 'FY14', '1.1');
writeReport(VND_equitytri, 'FY14', '1.1');
writeTable({VND_equitytri.percentiles, CNY_equitytri.percentiles, USD_equitytri.percentiles, GBP_equitytri.percentiles}, 'FY14', '1.1');

Can call from .NET in production, or make the same calls from MATLAB IDE during development
Automated Calibration Report Generation

CDF
QQ plot
Percentiles
Goodness of Fit Tests
Parameters

Source Data: 'ASX Index FX Last', 'ASX Index EGY DVD 12M'

Percentiles

<table>
<thead>
<tr>
<th>Individual Test</th>
<th>0.025</th>
<th>0.05</th>
<th>0.25</th>
<th>0.50</th>
<th>0.75</th>
<th>0.95</th>
<th>0.975</th>
</tr>
</thead>
<tbody>
<tr>
<td>P005_g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P005_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NC_g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NC_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP_c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP005_g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP005_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP005_c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP005_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP5_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP10_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP50_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Goodness of Fit Tests

<table>
<thead>
<tr>
<th>Individual Tests</th>
<th>P005_g</th>
<th>P005_r</th>
<th>S0</th>
<th>NC_g</th>
<th>NC_r</th>
<th>PP_c</th>
<th>PP_r</th>
<th>NP005_g</th>
<th>NP005_r</th>
<th>PP005_c</th>
<th>PP005_r</th>
<th>NP5</th>
<th>NP5_r</th>
<th>NP10</th>
<th>NP10_r</th>
<th>NP50</th>
<th>NP50_r</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
</tr>
</tbody>
</table>

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mu</td>
<td>-0.244</td>
</tr>
<tr>
<td>SIGMA</td>
<td>0.08573</td>
</tr>
<tr>
<td>ALPHA</td>
<td>0.24874</td>
</tr>
<tr>
<td>BETA</td>
<td>0.39168</td>
</tr>
</tbody>
</table>
Fit Tool Suite – optimisation toolbox

- We use several distributions not in the Statistics Toolbox (EGB2, Tilted Laplace, Log Gamma, Negative Log Gamma)
- Full set of fitting tools and options
- Can be run in Object Oriented mode or as simple functions

```matlab
classdef fitObj<handle

    properties
        params=[];
        distribution='';
        success=[];
    end

    methods

        function fitDistribution(obj, distMoments, varargin) ...

        function mymoments = analyticMoments(obj) ...

        function result = mypdf(obj, x) ...

        function result = myicdf(obj, x) ...

    end

end
```
Fit Tool Suite

- Using fmincon to minimise the errors in the Skewness and Kurtosis
- Can optimise over 4 parameters – previous tool (in Excel) could only do 1.
Solving cashflow equations – optimisation toolbox

- Determining the ‘Cost of Downgrade’ – cannot model analytically.

- Hold a bond portfolio for N years, with constraint that it must be investment grade (BBB or above).

- Some % of portfolio will end up below investment grade due to downgrades.

- Need to calculate the cost incurred to rebalance as an additional spread.

- Can project forward in time, problem becomes a form of the cashflow equation

\[
DCF = \frac{CF_1}{(1 + r)^1} + \frac{CF_2}{(1 + r)^2} + \cdots + \frac{CF_n}{(1 + r)^n}
\]

- Cashflow equations cannot be solved analytically for interest rate, \( r \). Need to call fmincon 100,000 times for our 100,000 stochastic simulations.

- Takes 45 minutes to run.
Using fminunc to solve cashflow equations

Solutions from fminunc for the interest rate are noisy (blue dots)

- Matches the Excel goal seek though.

Rather than solving for \( r \) for every simulation

- pre solve for fixed percentiles of the distribution.

- then interpolate.

Interpolation can be vectorised, whole model runs in 1.5 secs.
Conclusions

• We have migrated a time-consuming, error prone manual process in Excel into an automated tool in MATLAB and .NET

• Runs in a fraction of the time (press one button and wait 3 mins versus 2 months of copy/paste/refresh)

• Separates code from data
  – once we’ve tested the code base, we’re confident it will work with the next version of the data
  – all vectorised – we never need to worry about whether there is 12 months of data or 90 years of data

• Use of the optimisation toolbox made the process quicker and easier

• Auditable, traceable, repeatable

• Easy to change settings and re-run