Building Interactive Risk Dashboards with MATLAB

Timo Salminen, Model IT
The case

*Even today, when CROs need to see results, they ask for reports.*

goal

*Build an interactive risk dashboard that CROs and other business users want to use themselves*

challenges

*There are multiple visualization tools available for complex data but interacting with complex models requires heavy calculations and custom solutions*
The goal
Creating an interactive risk dashboard that answers all questions instantly
Design based on independent “micro apps”

- All apps are fully independent and communicate only with a common risk model
- Apps can visualize or alter current risk model state – or both

**APP EXAMPLES:**

- Shock yield curve
- Reverse stresses and sensitivities
- Visualize projections
- Risk Model
- Linear shocks
- Key rate durations
Underlying Risk Model

• Analysis backbone is high performance risk model with open API
• Can be called from any MATLAB program or script
• Optionally integrates with existing non-MATLAB software

CALL EXAMPLES:

Call for getting liability market value projection
proj = riskModel.GetMarketValue(...
    'PositionFilter', {'PortfolioType', 'liability'}, ...
    'DateGrouping', 'year');

Call for defining yield curve shock
riskModel.DefineScenarioComponent(...
    'ScenarioID', 'YieldCurveShock', ...
    'RiskFactor', 'EURSWAP', ...
    'Maturities', [8 16], ...
    'ShockType', 'absolute', ...
    'ShockAmount', [0 -0.002]);

Call for applying yield curve shock
riskModel.ApplyScenario(...
    'YieldCurveShock')
Case: Change in capital position

- App that visualizes capital requirement components and own capital in two risk model states
- Does not care how states differ from one another
- Can be applied to explaining all changes including new asset allocations and stress scenarios

LIVE DEMO
Case: Attributing change

- Design an app that attributes change in portfolio value between risk model states to risk factors
- Can be applied to explaining historical change, ad hoc stress or Value-at-Risk biting scenario
- Also supports navigation callbacks for interactive drill down with mouse or touch

<table>
<thead>
<tr>
<th>Change in Portfolio Value</th>
<th>Cash</th>
<th>Equity</th>
<th>Equity Derivatives</th>
<th>Fixed Income</th>
<th>Fixed Income Derivatives</th>
<th>Property Investments</th>
<th>RiskLife</th>
<th>Savings</th>
<th>UnitLink</th>
<th>UnitLink Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.04</td>
<td>0.01</td>
<td>0.00</td>
<td>-0.00</td>
<td>0.07</td>
<td>0.00</td>
<td>-0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.41</td>
</tr>
<tr>
<td>-0.23</td>
<td>-3.96</td>
<td>0.06</td>
<td>0.07</td>
<td>0.05</td>
<td>-0.00</td>
<td>0.00</td>
<td>-0.00</td>
<td>-0.00</td>
<td>-0.36</td>
<td>-0.41</td>
</tr>
<tr>
<td>-0.36</td>
<td>-0.58</td>
<td>-0.23</td>
<td>0.11</td>
<td>0.03</td>
<td>-0.41</td>
<td>-0.41</td>
<td>-0.41</td>
<td>-0.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td>-0.07</td>
<td>0.02</td>
<td>0.02</td>
<td>-0.07</td>
<td>-0.07</td>
<td>-0.07</td>
<td>-0.07</td>
<td>-0.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>-2.45</td>
<td>-0.07</td>
<td>-2.45</td>
<td>-2.45</td>
<td>-2.45</td>
<td>-2.45</td>
<td>-2.45</td>
<td>-2.45</td>
<td>-2.45</td>
<td>-2.45</td>
</tr>
</tbody>
</table>
Full Source Code of an app

- App for visualizing key rate durations of selected portfolio in two risk model states
- Uses BarTable component for presenting results both in interactive bar chart and numeric table
- Uses Risk model methods for setting up key rate shocks (up and down) and getting market value in shocked states
- Very little programming required to combine industry independent UI libraries and UI independent model into a useful app

```matlab
classdef KeyRateComponent < cFrame.UI.GUIComponent

methods

function Initialize(this)
% Initialize graphic component
this.AddBarTable(...
    'ID',       'KeyRateBars', ...
    'Title',    'Key rate durations', ...
    'XLabel',   'Key rate', ...
    'YLabel',   'Duration', ...
    'LegendNames', ('Current', 'Shock'), ...
    'YData',    zeros(1,60))
end

function Update(this)
% reserve data of right size
bardata = zeros(2, 60);
% Loop through before and after components
componentIDs = {'GUI_BEFORE', 'GUI_AFTER'};
for iC = 1:length(componentIDs)
    % Apply component state to risk model
    this.RiskModel.ApplyComponent(componentIDs{iC})
    % Set up key rate shocks
    this.RiskModel.SetKeyRateCalculation(1:60)
    % Get market value in all shocks
    mv = this.RiskModel.GetMarketValue;
    % Calculate key rate effects
    bardata(iC,:) = -(mv(1:60)-mv(61:end)) / 2;
end
% Update graphic component
this.SetObjectProperties('KeyRateBars', ...
    'YData', bardata')
end
end
```
Calling Risk Model from MATLAB Live Editor

- Ad Hoc Risk Dashboards can be created in minutes with MATLAB live editor
  - For example, visualize risk analysis before and after trades

LIVE DEMO
Risk Model internals

<table>
<thead>
<tr>
<th>Risk Model</th>
<th>Company Model</th>
<th>Scenario Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>main function</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Balance sheet valuation as a function of scenario model</td>
<td>• Risk factor calibration and simulation</td>
<td></td>
</tr>
<tr>
<td>• Modeling effect of investment strategies and management actions</td>
<td>• Stress test generation</td>
<td></td>
</tr>
<tr>
<td><strong>features</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Customizable group hierarchies</td>
<td>• Sensitivity calculations (e.g. grids and key rate shocks)</td>
<td></td>
</tr>
<tr>
<td>• Market value breakdown to instruments and maturities</td>
<td>• Fully customizable shock scenarios</td>
<td></td>
</tr>
<tr>
<td>• Fully customizable trade scenarios</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Vectorized valuations and parallel computing support</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**examples of underlying models**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Valuation formulas for derivatives</td>
<td>• Fat tail and copula models for risk factors</td>
</tr>
<tr>
<td>• Proxy models for complex liabilities and exotics</td>
<td>• Regulatory discount curve calculation</td>
</tr>
<tr>
<td>• Regulatory capital calculation models</td>
<td></td>
</tr>
<tr>
<td>• Dynamic investment strategies</td>
<td></td>
</tr>
</tbody>
</table>
The layout of dashboard

- Individual apps need to be combined to one dashboard
- Layout objects handle
  - positioning of apps within dashboard – using available screen space efficiently
  - Updating only visible apps when risk model changes ➔ avoid unnecessary calculations
Case: Stochastic Portfolio Optimization

• Optimizing 1-step buy-and-hold investment strategy with
  • Non-normal real world risk factor distributions with drift included
  • Simultaneous asset allocations and derivative overlays
  • Complex life insurance liabilities
  • Custom optimization criteria

• Solution
  1. Use risk model to calculate distributions for all decision variables
  2. Sample millions of investment strategies ➔ millions of stochastic distributions
  3. Calculate optimization criteria for all strategies
  4. Visualize results
Multiple frontiers are calculated:

- Optimal portfolios with full balance sheet and selected risk and return criteria
- Frontier with mean-variance targets – effect of selected criteria
- Frontier without derivatives – restricts ability to hedge unnecessary risks
- Frontier calculated omitting liabilities -

LIVE DEMO
Summary: full anatomy of risk dashboards

Based on independent layers – 100% written in MATLAB

**GUI**
- **Visualization components**
  - General purpose objects for presenting and interacting with data. Know nothing about layouts or use case (finance).
- **GUI Layout Components**
  - General purpose objects handling layouts of any GUI components. Know nothing about use case (finance).
- **Risk Dashboard Layout**
  - Very thin layer containing positioning of apps in risk dashboard
- **Risk “Micro” Apps**
  - Light-weight controllers that call risk model and present results in interactive graphic objects. Don’t know what UI technology is used for implementing the UI.
  - Creates a unified API for underlying scenario and company models. Doesn’t know anything about graphics or UIs.

**Model**
- **Company**
  - Scenario model provides API for simulating and stressing risk factors. Doesn’t care about which risk factor models are used. Doesn’t know that a company exists.
- **Scenarios**
  - Valuation model provides API for valuating balance sheet as a function of risk scenarios and for modifying investment strategy. Doesn’t care about how scenarios are created or positions are valued.
- **Valuation models**
  - Open Source – customizable and extendable

Valuation and risk factor models and visual presentation can differ between companies.
Thank you!

Questions?