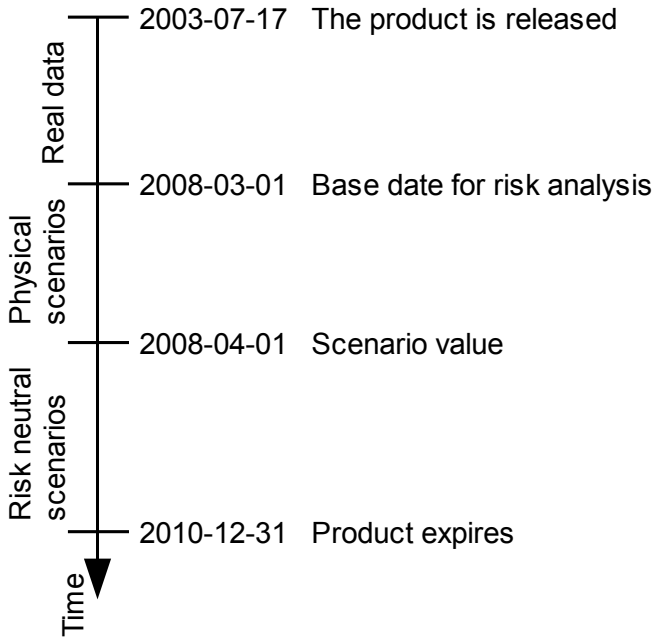
THETARIS
Engineering in Finance

Case Study: Risk Management with Thetaris Theta Suite and Algorithmics OpenMtF

This document describes the process of modelling a financial product and how the product is efficiently deployed for daily risk analysis in a risk management system such as Algo Suite. All steps are explained from a user perspective and performed within the Theta Suite, the graphical front end for Thetaris technology. Hence, the purpose of this document is a quick introduction to all steps from product modelling to final computation. This demonstrates the simplicity and elegance of the Theta Suite approach.

Case study

In this document we will look at a convertible bond and prepare it for the Algorithmics risk management system Algo Suite. For the purpose of this document we pick March 1st, 2008 as the time of risk analysis and evaluate a product that was released on July 17th, 2003. The horizon for our analysis will be one month.

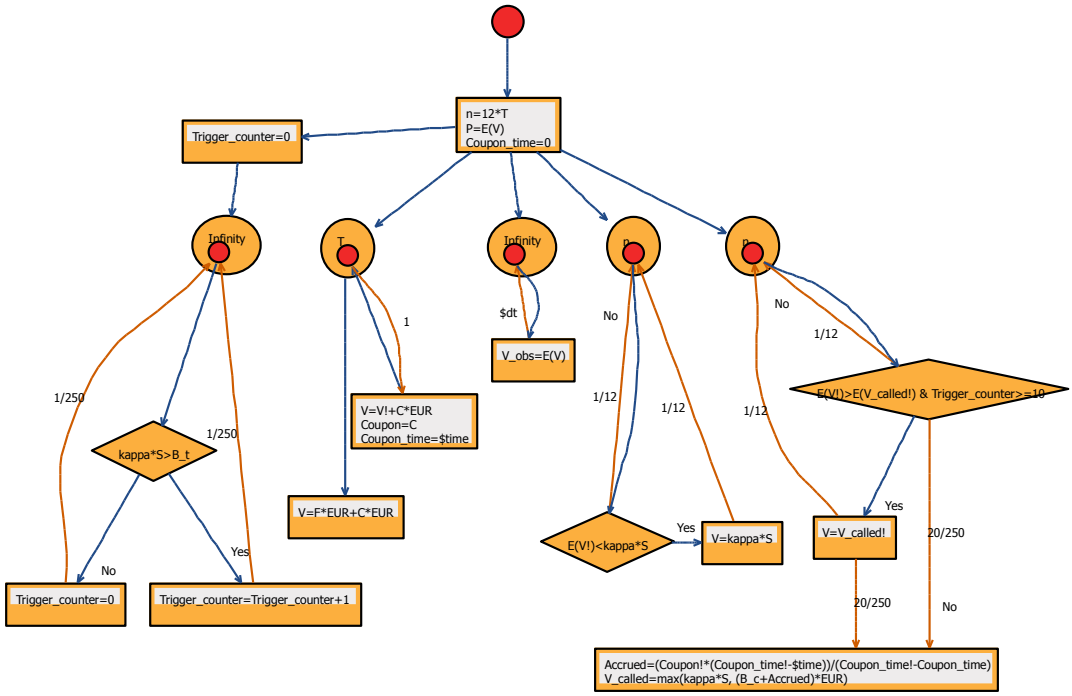


The tasks demonstrated in this document include the creation of a model for the financial product, the maintenance of product fixings, the mapping of risk factors to the scenario cube and the computation of a result cube. This example is demonstrated with a convertible bond as a product and the Muromatchi model for risk neutral pricing with a defaultable underlying. Physical scenarios are taken from the example scenario cube provided by the OpenMtF package.

Defining the product

The product is a typical convertible bond. Convertible bonds are complex instruments due to the conversion and call constraints. In this example, we look at a convertible bond from EPCOS AG (Germany), which is traded under ISIN XS0170329337. The details of this convertible can be found in the Appendix.

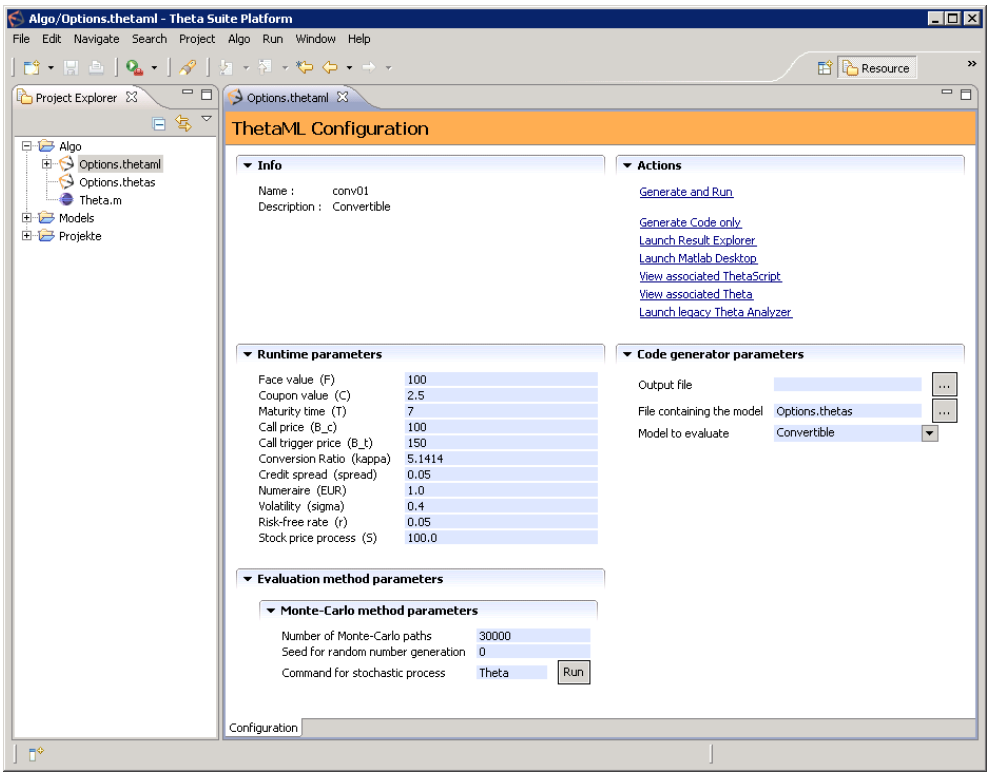
In general convertible bonds incorporate a variety of features. In this case, the instrument is convertible into shares of the issuing company and the convertibles may be converted by the holder at any time. In addition, the issuer has the right to redeem or call back the convertible at a specified call price. If the issuer does so, the investor can choose between receiving the call price or converting into shares. The ability of the issuer to call back the issue is restricted by "soft" call constraints. A soft call constraint requires that the issuer's stock price remains above a discretely observed trigger level before the issue can be called, in this case the convertible cannot be called until the underlying has been above the trigger level for 10 out of 20 days. In addition, the issuer has to give notice 20 days in advance of calling the issue.



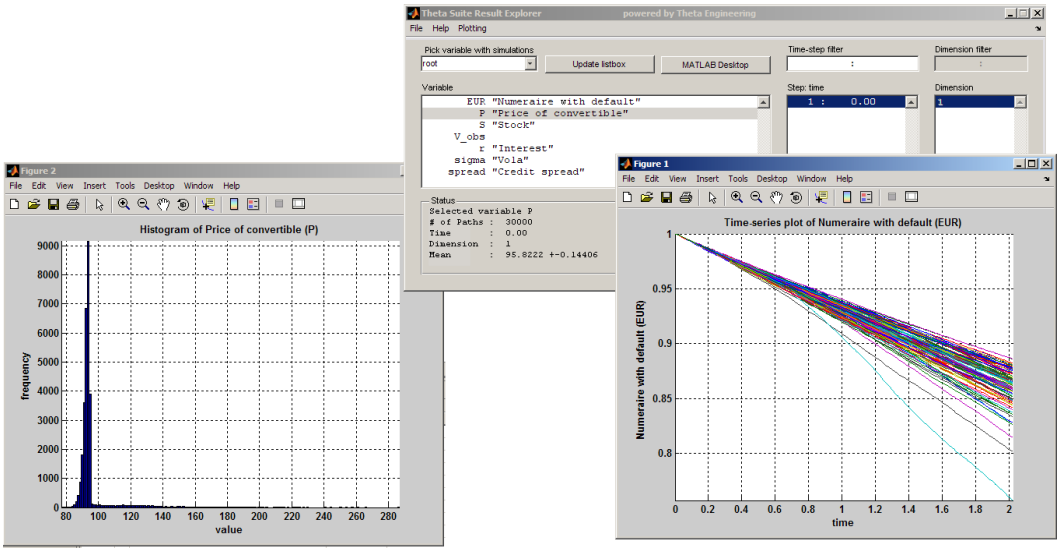
Thetagram representation of the financial product (see Appendix for term sheet and ThetaScript)

After the product has been modelled in ThetaScript (see Appendix), we can start to provide it with market parameters. The picture above shows the same product represented graphically as Thetagram.

Right clicking on the model file allows you to initialize a runtime configuration as shown below. All imported variables are queried in an input mask. A click on “Generate and Run” runs the numerical evaluation and displays the Theta Result Explorer, which features result visualizations.



Run configuration for the product.
All required input parameters are queried in this form.

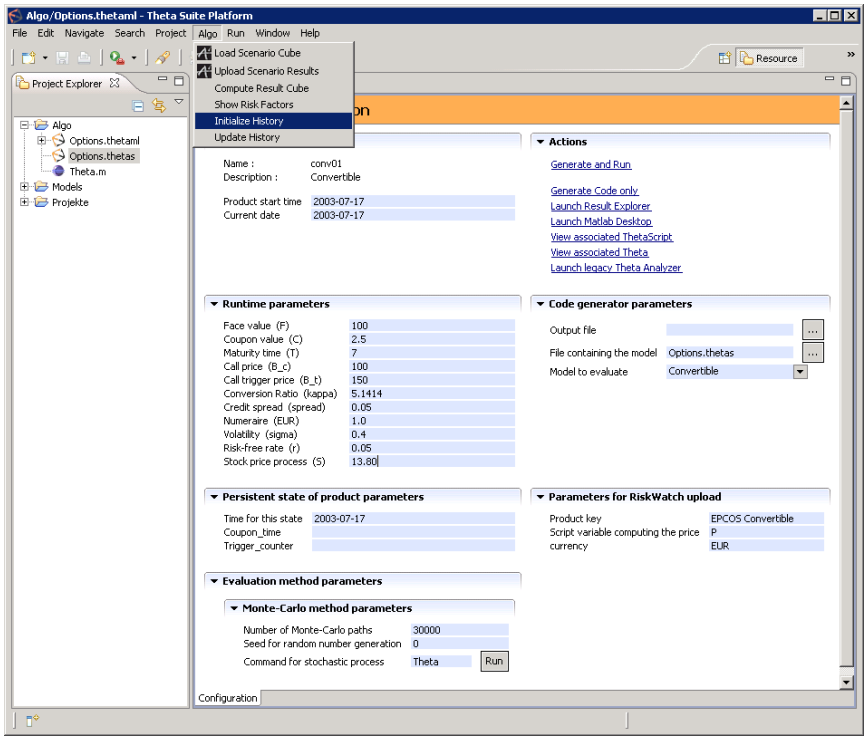


The Theta Result Explorer shows the product price to be 95.82.
The two plots show the histogram of outcomes and the time series for the discount factor.

Updating product parameters from real data

Now that we have a working ThetaML model for our product it is easy to bring it to life. We again assume that we are at July 1st, 2003 and just started to introduce this product to our risk management system and re-evaluate its price and remaining risks on a daily basis. Two things have to be considered when tracking the product's risk contribution during its lifetime. First, the ThetaML must be evaluated with a new start time, thus simulating only the remaining life time. Second, the evaluation must take into account possible path dependencies. Previous fixings and the history of relevant state parameters must be maintained correctly. The Theta Suite allows you to do both in an easily accessible way.

After we have verified the model's correctness we can prepare it for productive use. For this we open the product's configuration file "Options.thetaml" in the Theta Suite. Then we choose the "Initialize History" action from the "Algo" group in the menu bar. The form now displays two new sections. One of it represents the state of previous fixings and path dependent parameters. In our case these are called "Coupon_time" and "Trigger_counter". Both variables are defined in ThetaML and detected automatically to belong to this state. The second prompts the user for the corresponding identifiers needed to update the data in the Algorithmics Mark-to-Future Cube.

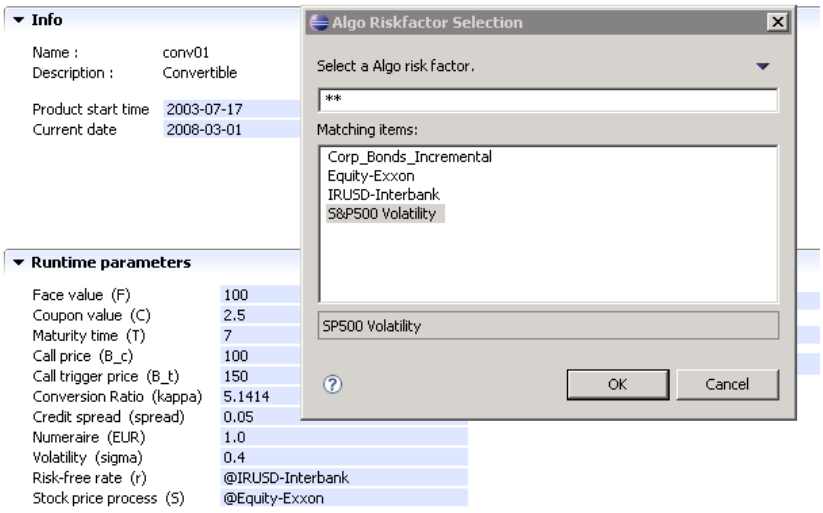


Since our modelled product is dependent on the price path updates for market data parameters must be provided regularly. We and enter the new date and all current parameters for interest rate and stock price. Then we select the menu entry "Update History" from the "Algo" group in the menu bar. The state parameters are now updated and the product can be re-evaluated with the real history of parameters. The whole process can be repeated to always reflect the current state in product evaluation. In this document we will proceed until March 1st, 2008.

Pricing scenarios

At this point our model is prepared for daily evaluation, making sure that path dependencies are correctly maintained. Until now we did not yet upload any results to the Algorithmics risk management system. This is our next task. Again the Theta Suite makes all relevant steps easily accessible.

First, we have to match the input parameters of our model to the risk parameters generated by the Algorithmic Scenario Engine. For the scenarios of selected risk factors, the product's price will be evaluated. The Theta Suite can read text output generated by the OpenMtF example program SampleASE. In our example we match the stock price to "@Equity-Exxon" and the interest rate to the curve "@IRUSD-Interbank".



Now we are ready to start the evaluation of all scenarios and update the Algorithmics Mark-to-Future Cube. First we select "Compute Result Cube" from the menu. After the computation is complete we can continue to evaluate scenarios for additional products, or commit our results with a final click on "Write Scenario Output". At the time of writing the result is stored into a text file and processed with the OpenMtF example program GenerateCube.

Final words

This case study presents a step-by-step walk through to the current integration of Thetaris' Theta Suite and Algorithmics' OpenMtF. The prototype demonstrates that a simple and easy integration can be done. The next steps towards deployment would be the creation of more examples and the setup of a demonstration workflow starting at Algorithmics ASE over OpenMTF and Thetaris Theta Suite to Algorithmics ARE.

Appendix

The detailed ThetaScript for the product

The term sheet for this product can be downloaded from this link:

http://www.epcos.de/web/generator/Web/Sections/InvestorRelations/FinancialReporting/Wandelanleihe2003/PDF/convertible__03.pdf,property=Data__nn.pdf;/convertible_03.pdf

```
% Model of a convertible bond with soft call conditions
Model Convertible
	import S "Stock"
	import F "Face value"
	import C "Coupon value"
	import EUR "Numeraire with default"
	import T "Maturity time"
	import B_c "Call price"
	import B_t "Call trigger price"
	import sigma "Vola"
	import r "Interest"
	import kappa "Conversion Ratio"
	import spread "Credit spread"
	export P "Price of convertible"
	export V_obs

	% monthly conversion
	n = 12 * T

	P = E(V)
	Coupon_time = 0

	fork
		% Compute trigger (10 days above trigger ~ trigger >=10)
		Trigger_counter = 0
		loop inf
			if kappa*S > B_t
				Trigger_counter = Trigger_counter + 1
			else
				Trigger_counter = 0
			end
			Theta 1/250
		end
	end

	fork
		% implement conversion constraint
		loop n
			if E(V!)<kappa * S
				V = kappa * S
			end
			Theta 1/12
		end
	end
```

```
        fork
             % implement call constraint
             loop n

                  % Call if trigger constraint satisfied and calling is
                  % optimal for the issuer
                  if (E(V!)> E(V_called!)) & Trigger_counter >= 10
                       V = V_called!
                  end

                  % Compute price after call notice
                  fork
                       Theta 20/250
                       % compute accrued interest
                       Accrued =  Coupon! * (Coupon_time!-@time)/ ...
                                 (Coupon_time! - Coupon_time)
                       V_called = max(kappa*S, (B_c + Accrued) * EUR )
                  end

                  Theta 1/12
             end
        end

        % implement payment of coupons
        loop T
             Theta 1
             V = V! + C * EUR
             % remember coupon value and time for computation of
             % accrued interest
             Coupon = C
             Coupon_time = @time
        end
        % pay face value at maturity time
        V = F*EUR + C * EUR
End
```