FRAME CONSULTING
Financial Risk & Applications

DiffFusion.jl

A New Exposure Simulation Framework for the Post-Libor World

FRAME Consulting GmbH, Berlin[1]

In this document, we introduce the new exposure simulation framework DiffFusion.jl. Key take-aways are

- DiffFusion.jl covers products and market nuances that emerged in the course of the Libor cessation.
- DiffFusion.jl's design is based on a clear modular architecture that allows for flexible extension in scope and depth.
- DiffFusion.jl is transparent and available as open-source project.

The high-level architecture of the framework is shown in Figure 1.

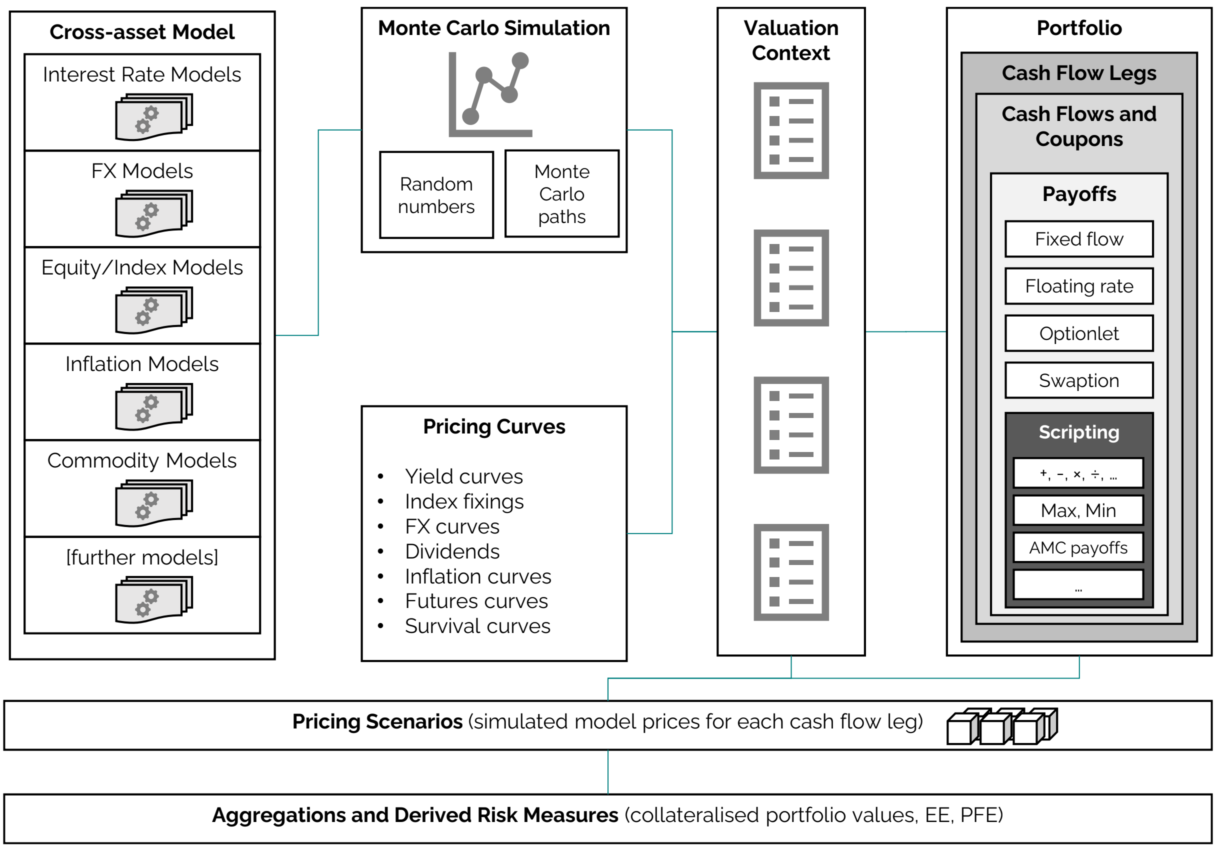

Figure 1: Architecture of the DiffFusion.jl framework.

What Is the Purpose of the Framework?

Scenario-based financial instrument pricing is at the core of most risk management processes and methods. The DiffFusion.jl modelling framework provides a flexible and computationally efficient simulation and pricing engine. It contains state-of-the art implementations of single-factor and multi-factor models.

The framework is designed for regular large-scale portfolio simulations as well as ad-hoc and interactive pricing and risk calculation analysis. As

[1] Contact: Sebastian Schlenkrich. Mobile: +49-1522-715-0780. Mail: sebastian.schlenkrich@frame-consult.de.

such, the framework can be used in production processes as well as for benchmarking and model validation purposes.

What Products are Covered by the Framework?

The DiffFusion.jl framework already includes standard fixed income and interest rate cash flows for principal payments, fixed rates and floating rates. Floating rate cash flows can reference classical forward-looking Libor-like rates and new backward looking compounded risk-free rates.

Linear FX and cross currency instruments are covered by the interest rate cash flows and exchange rate conversion. Additional principal payments in mark-to-market cross currency swaps are handled by a specific cash flow leg type.

Upcoming releases will allow decorating floating rate types with caps and floors. Furthermore, OIS-based and Libor-based cash-settled and physically settled European swaptions will also be available.

Further cash flow types for inflation and commodities will be added going forward.

Exotic products typically require bespoke cash flow and payoff specifications via payoff scripting. Payoffs and payoff-scripting are integral functionalities of the DiffFusion.jl framework by design. Consequently, exotic products can easily be incorporated.

What is the Design of the Framework?

The architecture of the DiffFusion.jl framework follows a clear modular design. The framework components are illustrated in Figure 1.

Simulation models in the DiffFusion.jl framework represent market standard risk neutral valuation models. The DiffFusion.jl framework covers models for interest rates, exchange rates, equities/indices, inflation, and commodity futures. All models can be combined into hybrid models for joint simulation.

A key objective of the framework is flexibility regarding the choice of model. For example, users can combine multi-factor rates models for major currencies with single-factor rates models for other currencies.

Scenario generation is based on **Monte Carlo simulation** of risk factors. Risk factors are represented by generic model state variables. We note that the generic model state variables are independent of input curves for rates and indices. This is a key design feature of the framework and proves particularly advantageous for sensitivity calculation.

Financial instruments in the DiffFusion.jl framework are represented as a layered composition of payoffs, cash flows and cash flow legs. Each layer adds a level of abstraction and common functionality. Cash flow legs are finally combined into products and portfolios. This design is illustrated in the upper right part of Figure 1.

Portfolios and products are decoupled from the simulation model and Monte Carlo method. The link between model, simulation and market data is established via a ***Valuation Context*** object. This decoupling also contributes to the flexibility and extensibility of the framework.

Scenario prices for portfolios and products are stored in a **scenario cube**. The axes of the cube are simulated scenarios, future observation times and individual product legs. With this data we calculate **risk measures** like expected exposure and potential future exposure. New risk measures can easily be added to the framework. Alternatively, scenario cubes can also be processed by client applications or directly by the user.

Sensitivities of future prices with respect to valuation inputs are calculated efficiently using Automatic Differentiation (AD) methods. AD methods are directly available in the chosen Julia programming language.

Portfolio risk measures need to take into account **collateralization**. Upcoming releases will include a collateral model covering relevant CSA parameters like threshold, minimum transfer amount and independent amount. Collateralized portfolios are modelled using a margin period of risk. The methodology allows for an accurate portfolio modelling.

How can the framework be accessed and used?

We develop and maintain the DiffFusion.jl framework as an open-source project. Open-source access yields full transparency on models and methodologies. The extensive test suite provides guidance on model usage and can act as blue print for user tests and model validation activities. This aims at creating trust and acceptance by the users.

The DiffFusion.jl framework is implemented in the Julia programming language. Julia is a modern high-level programming language designed for high-performance computing.

The framework can be accessed from widely-used high-level languages like, e.g., Python or R. Users who are familiar with Julia or feel comfortable with high-level programming languages can also directly incorporate the framework as a package in Julia user code.

How We Can Help You Integrate the DiffFusion.jl Framework

FRAME Consulting is a consultancy specialised in financial risk management and applications. Our clients benefit from our extensive expertise in risk management models and methodologies. A particular focus of our expertise is on valuation models for all types of financial products and across all asset classes.

We support our clients throughout all steps of system integration. In particular, we provide services related to the following steps:

- technical setup of the DiffFusion.jl framework in client's IT infrastructure,
- workshops and trainings on available models, methodologies, products and risk calculations,
- interfacing client market data and instrument data for ad-hoc analysis and regular production calculation runs,
- embedding the DiffFusion.jl framework's risk measure calculations in the clients' risk management processes,
- developing bespoke methodologies and features tailored to client requirements.

Please reach out to info@frame-consult.de for more information and how to get started.

Technical Appendix

In the following sections we provide more technical details on the DiffFusion.jl framework.

DiffFusion.jl covers the following process steps:

- model-based scenario generation,
- scenario-based financial instrument pricing and sensitivity calculation, as well as
- scenario-based risk measure calculation.

The framework develops its full potential when the process steps are combined. However, the processes can also be used independently.

What Models are Covered?

We make extensive use of the Heath-Jarrow-Morton (HJM) framework for the component models. In its basic variant, the model state variables follow Ornstein–Uhlenbeck processes. This approach yields most analytical tractability for simulation, pricing and model calibration. More complex model variants can be easily added to the framework.

Interest rates are modelled as multi-factor Gaussian HJM models. Such models allow for a rich set of simulated yield curves involving curve shifts, slopes and curvatures. The models can be calibrated to the full surface of at-the-money swaptions. In its single-factor form, the model reduces to the classical Hull-White interest rate model.

Exchange rates are modelled in a classical Black-Scholes-type model. Exchange rate models are linked to the corresponding domestic and foreign interest rate models. Calibration of the model takes into account the joint evolution of exchange rates as well as corresponding interest rates.

Equities and indices are modelled analogous to exchange rates. In that context, the foreign interest rate model is replaced by a dividend yield term structure. The framework allows for a modelling of discrete dividend and dividend yields.

Inflation models are designed following the foreign currency analogy. As a consequence, inflation models are also analogous to exchange rate models. This approach covers the classical Jarrow-Yildirim three-factor model as well as the two-factor Dodgson-Kainth model. Initial inflation forward curves are direct inputs to the model. This allows for modelling seasonality.

Commodity futures are modelled following the HJM framework applied for interest rates. As a consequence, we can cover single-factor models and multi-factor models. This allows for a modelling of futures volatility term structures. Initial futures curves are direct input to the model and can incorporate commodity-specific features like seasonality patterns.

Credit spreads will be modelled via CIR models. CIR model state variables cannot be represented as Ornstein–Uhlenbeck processes. However, we can apply a (local) log-normal approximation. This approach keeps the CIR model implementation within the design framework of the other component models.

How are Monte Carlo Scenarios Generated?

Component models for e.g. interest rates, exchange rates and equities are combined into a cross-asset hybrid model. The joint evolution of the hybrid model state variables is then calculated by Monte Carlo simulation.

The Monte Carlo method in the DiffFusion.jl framework utilizes the Ornstein–Uhlenbeck properties and uses bias-free simulation of model state variables whenever the model allows for that. This approach provides flexibility on the chosen time grid and allows simulating long time-horizons without sacrificing simulation accuracy.

Random numbers are drawn from efficient pseudo-random number generators or low-discrepancy sequences.

Why Do We Use Julia Language?

Risk factor simulation and scenario-based financial instrument pricing are computationally expensive calculations. Efficient implementation of such calculations requires fast compiled machine code, ability to parallelize and distribute calculations as well as support for high-performance computing hardware like GPU. The Julia language natively fulfills all these requirements.

At the same time, development in Julia language is lightweight and similar to languages like Python. As a result, new functionalities can easily be added within the framework or attached by client applications or user interaction.

Sensitivity calculation is critical for risk management processes. For exposure simulations, sensitivity calculations can be particularly challenging from a computational perspective. These challenges are addressed by Automatic Differentiation (AD) methods. Julia language supports forward mode and reverse mode AD via operator overloading and source transformation. We leverage these language features and provide efficient and accurate Delta and Vega calculations.

Documentation and Source Code

Documentation and source code for the DiffFusion.jl framework are hosted on GitHub at https://github.com/frame-consulting/DiffFusion.jl.

Related Literature and References

- L. Andersen, V. Piterbarg. Interest Rate Modeling. 2010.
- R. Jarrow, Y. Yildirim. Pricing Treasury Inflation Protected Securities and Related Derivatives Using an Hjm Model. 2003.
- L. Andersen. Markov Models for Commodity Futures: Theory and Practice. 2008.
- A. Green. XVA. 2016.
- L. Andersen, M. Pykhtin, A. Sokol. Rethinking Margin Period of Risk. 2016.
- The Julia Programming Language
- PyJulia - a Python interface to the Julia language
- JuliaCall - an R interface to the Julia language