QuIC Mechanics FD User Guide

(HSBC Edition)

October 2006

CONFIDENTIAL INFORMATION

Last Updated for Product Version: 1.0
Document Revision: 1.0

TERMS OF USE

Trademark Notices

QUIC FINANCIAL TECHNOLOGIES INC.

Head Office
1095 W. Pender Street, Suite 1105
Vancouver, BC V6E 2M6
CANADA

EMEA
1 Cornhill
London, EC3V 3ND
UNITED KINGDOM

Research & Development
3553 – 31st Street NW, Suite 225
Calgary, AB T2L 2K7
CANADA

Americas
39th Floor, 245 Park Avenue
New York, NY 10167
USA

Asia Pacific
One Marina Boulevard #28-00
SINGAPORE 018989

www.quic.com

Within North America: 1.877.689.1888
Outside North America: 1.306.337.1446

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Part 1: Understanding QuIC Mechanics FD

1 INTRODUCTION

This chapter introduces the QuIC Mechanics FD product and describes the purpose and structure of this manual. It also identifies related publications and provides contact information for support.

1.1 About the QuIC Mechanics FD Product

QuIC Mechanics FD[1] provides a flexible framework for defining financial instruments for the QuIC Simulation Framework (QSF) that use the finite difference (FD) method for pricing. These instruments can represent complex structured transactions by combining multiple primitive features (events). The QSF supports the calculation of potential future exposure (PFE) for whole portfolios of such trades.

1.2 Scope and Purpose of This Manual

The scope and purposes of this manual are as follows:

- To describe the financial models and methods that QuIC Mechanics FD uses.
- To describe how to prepare the inputs for QuIC Mechanics FD transactions.
- To provide detailed reference information for the required data structures.

The present version of this manual does not cover preparation of the QuIC Script solution file that drives a pricing run. In the QuIC Mechanics FD solution, the solution script is relatively static; most of the setup task involves the data files.

1.3 Organization of This Manual

This manual consists of two parts that comprise the following chapters (in addition to the present chapter):

- Part 1, Understanding QuIC Mechanics FD, includes conceptual information to help you understand what the product is and how it works:
  - Chapter 2, Financial Models and Methods, defines potential exposure, explains how the finite difference approach is applied, and describes the G2++ short rate model.
  - Chapter 3, QuIC Mechanics FD Design, provides a high-level description of the software architecture of QuIC Mechanics FD.

[1] This manual uses the abbreviation QM-FD for QuIC Mechanics FD in tables.

- Part 2, Using QuIC Mechanics FD, provides the information that you need to set up trades:
  - Chapter 4, Setting Up a Trade, illustrates how to prepare the necessary data files with an example.
  - Chapter 5, Reference: Input Files, gives details about the structure of the required input files.
  - Chapter 6, Reference: Classes, identifies the creation parameters of the QuIC Mechanics FD classes (observables and events) that you specify in input files.
  - Chapter 7, Reference: Solver Preferences, identifies the solver preference parameters that you can specify to control the solution process.

1.4 Related QuIC Publications

Refer to the publications in Table 1 for further information about the QuIC environment in which you analyze trades with the QuIC Mechanics FD solution.

Table 1: Related publications

Title	Description
QuIC Workbench 2.5 – Orientation Guide	Provides an introductory tutorial for the QuIC Workbench application.
QuIC Workbench 2.5 – User Guide	Provides the installation procedure for QuIC Workbench and describes the user interface of the QuIC Workbench development environment.
QuIC Script 2.5 – Programmer's Guide	Describes all the constructs and the syntax of the QuIC Script language.
QuIC Script 2.5 Reference	An HTML document that you access from the QuIC Workbench Help menu. It documents the details of the built-in operators, library functions, data structures, enumerations, and variables that are available to the QuIC Script programmer.
GSF User Guide and Data Reference	Describes the QuIC Simulation Framework (QSF) and details how to prepare QSF inputs.

1.5 Where to Go for Help

For additional help with information covered in this manual, contact your designated account manager or QuIC customer service at 1.877.689.1888 within North America, or 1.306.337.1446 outside of North America.

Technical support is also available by email at techsupport@quic.com.

2 FINANCIAL MODELS AND METHODS

The QuIC Mechanics FD product is a flexible framework for using finite difference methods to price a portfolio of instruments and calculate the potential exposure of the portfolio. This chapter describes finite difference pricing and the interest rate model used for pricing the instruments, and provides details of calculating the instrument prices and portfolio potential exposure. The chapter starts with some definitions.

2.1 Some Definitions

This section explains the concepts of potential exposure, expected potential exposure, and mark to market.

2.1.1 Potential Exposure

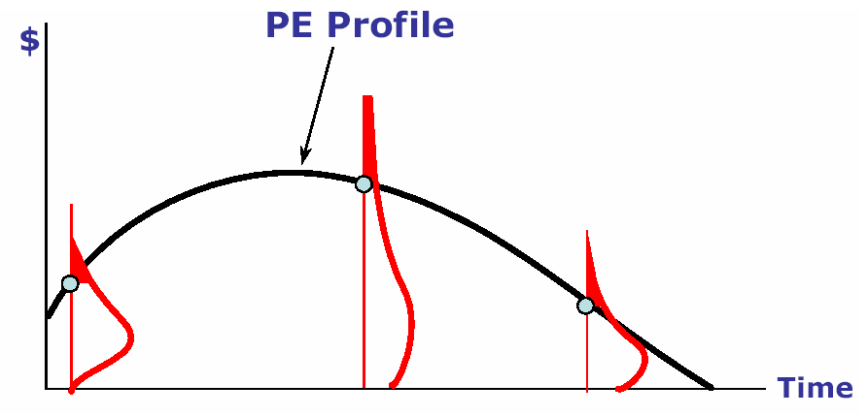
Potential exposure (PE) is conventionally defined as follows:

> The PE of a portfolio at time t is a specified percentile of the distribution of $max(V_t,0)$, where V_t is the value of the portfolio at time t. The distribution is obtained by pricing the portfolio in N_{out} different simulated market scenarios at each desired time t.

Although potential exposure is a general term that may apply to various metrics such as maximum likely exposure and expected exposure, this manual uses the term primarily to mean the maximum likely exposure at the 95th percentile. This manual also uses the term potential future exposure (PFE) to mean the same thing.

Figure 1 shows the PE profile (in the sense of maximum likely exposure) that results from plotting the PE ($) versus time.

Figure 1: PE profile



A typical PE profile has a hump shape that reflects two opposing forces:

- As time progresses, potential changes in market risk factors have an increasing impact on exposure, causing PE to increase.
- As time progresses and instruments move closer to maturity, cash-flows received by the holder reduce the portfolio's notional value (and hence the potential exposure to a counterparty default)

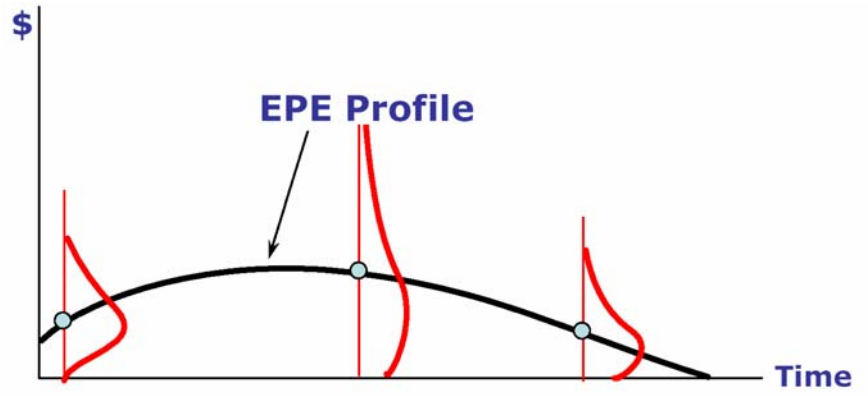
While PE is always greater than zero, positive and negative values of individual transactions can be extracted to provide a detailed breakdown of counterparty PE calculations.

2.1.2 Expected Potential Exposure

Referring to the previous definition of PE, the *expected potential exposure* (EPE) of a portfolio is the average of $max(V_t,0)$ over the N_{out} simulated market scenarios.

Figure 2 shows the EPE profile that results from plotting the EPE ($) versus time.

Figure 2: EPE profile



In order to calculate the PE and EPE, the portfolio must be revalued at every time step t, in every one of the N_{out} simulated market scenarios.

2.1.3 Mark to Market

Marking to market is the daily repricing of instruments using prevailing market data. Unlike PE and EPE, mark to market (MTM) prices can be positive or negative, reflecting a net inflow or outflow, respectively. Mark to market prices can be calculated on the basis of simulated risk factors at any time step.

2.2 Finite Difference Pricing

To calculate the PFE on a portfolio of instruments one needs to determine the MTM value of each instrument on future dates (the PFE dates). The finite difference (FD) approach lends itself well to this problem because the FD solution

generates a state space of instrument values for all possible interest rate scenarios over the simulation time horizon.

A separate Monte Carlo simulation is performed to construct simulated paths for the underlying risk factors. This produces a simulated yield curve for the relevant interest rate on all exposure dates and for all Monte Carlo scenarios. The Monte Carlo simulation of the risk factors does not need to use the same interest rate model as that used to value the instrument.

The value of the instrument along each simulation path is obtained by interpolating into the state space to produce a distribution of instrument prices on each exposure date.

A mapping between the simulated yield curves and the state space is defined to facilitate the interpolation. Given the distributions of MTM instrument prices for all instruments in the portfolio on all exposure dates, the potential exposure can be calculated.

The remainder of this section describes the finite difference pricing and PFE calculation in more detail. The first subsection provides details of finite difference pricing and the construction of the state space. The second subsection describes interpolation from this state space to determine the distribution of instrument prices.

2.2.1 State Space Generation

Assume we are using an interest rate model that emits a partial difference equation (PDE) for the value of an instrument as a function of time t and interest rate (short rate) r:

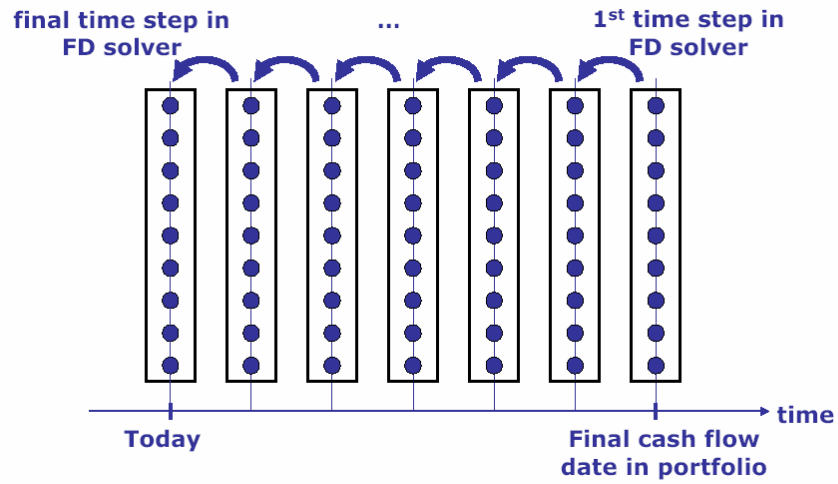
$$\frac{dV(r,t)}{dt} + L\left[V(r,t)\right] = 0$$

where L is a linear operator on $V(r,t)$.

For the two-factor short rate model that we consider below, r is replaced by the two factors x and y, and we have $V(x,y,t)$.

This PDE is solved by discretizing the equation using an implicit method (Crank-Nicholson) which results in a linear system of equations which is solved with iterative methods. Figure 3 illustrates that the PDE is solved by stepping backwards in time starting from the final cash flow date of the instrument, at which time the terminal condition of $V(r)=0$ is applied.

Figure 3: Finite difference solver



While $V(r)=0$ at terminal time, V undergoes discrete steps as cash flows of the instrument occur. On a relevant add cash flow date (`dtAddCashflow`) the state space V is updated as follows:

$$V(r,t^{+}) = V(r,t^{-}) + \mathrm{DCF}(r,t)$$

where $\mathrm{DCF}(r,t)$ is the cash flow value discounted from `dtAddCashflow` to the date on which the cash flow physically occurs (`dtPayment`). The reason cash flows are not added to the state space on the date they physically occur is that market fixings upon which the cash flow depends are typically fixed in the past (dates not yet reached by the FD solver). Because these fixings are not known on the payment date `dtPayment`, the value of the cash flow is not known and therefore cannot be added to the state space. This is an artifact of the backwards stepping of the FD solver. The next section, which describes how callable swaps are handled in the FD solution, indicates the importance of not adding cash flows to the state space until they become part of the forward-starting swap, even if the value of the cash flow is already known.

2.2.1.1 Callable Swaps

To handle call options in the instrument, we assure that at every time step the state space V contains only the value of all forward-starting coupons (cash flows). On any call date, V therefore contains the continuation value of staying in the swap. Hence the value of a callable swap can be tracked through the state space by updating the state space on each call date as follows:

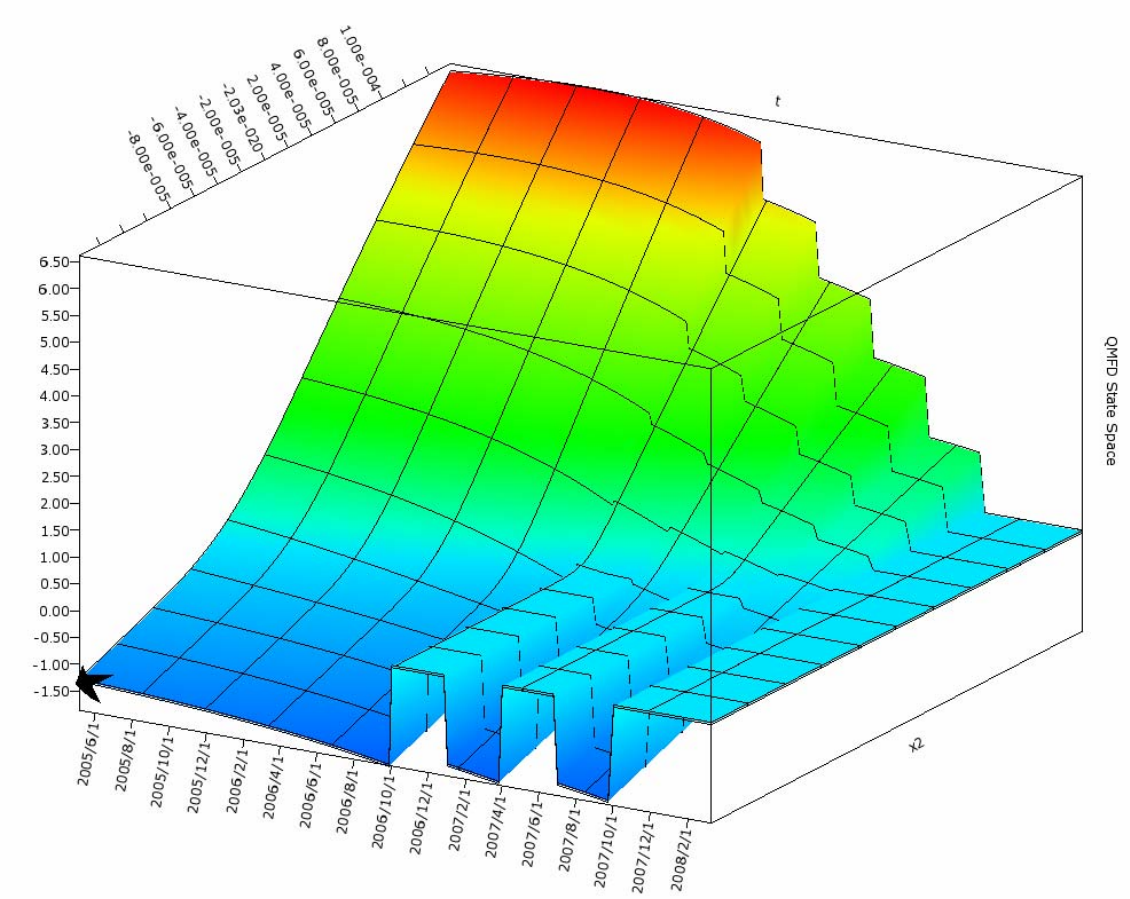
$$V(r,t) = \max\left(V(r,t), R(r,t)\right)$$

where, for a callable swap, the redemption value $R(r,t)=0$ (the swap is cancelled and no future cash flows occur).

Following this algorithm of adding to and resetting the state space, solving the PDE populates a state space for the value of the forward-starting callable swap.

Figure 4 plots the state space for a Bermudan callable swap—swap value as a function of time and short rate r. The terminal condition of $V=0$ is evident, as well as the discrete jumps in the swap value as cash flows are added to the state space. Also evident are two resets to $V=0$ that occurs on call dates for swap values for which the call option would be exercised.

Figure 4: FD example: Bermudan callable swap state space



2.2.1.2 Bermudan Swaptions

Bermudan swaptions are valued within the FD approach by using two state spaces. The first state space F holds the value of the forward-starting swap (the swap that is entered if the swaption is exercised). The second state space V holds the value of the swaption. The PDEs for F and V are valued simultaneously by stepping backwards in time starting from the terminal condition $F=V=0$ at the final cash flow of the underlying swap. The algorithm for updating these state spaces is as follows:

$$F(r,t^{+}) = F(r,t^{-}) + \mathrm{DCF}(r,t)$$

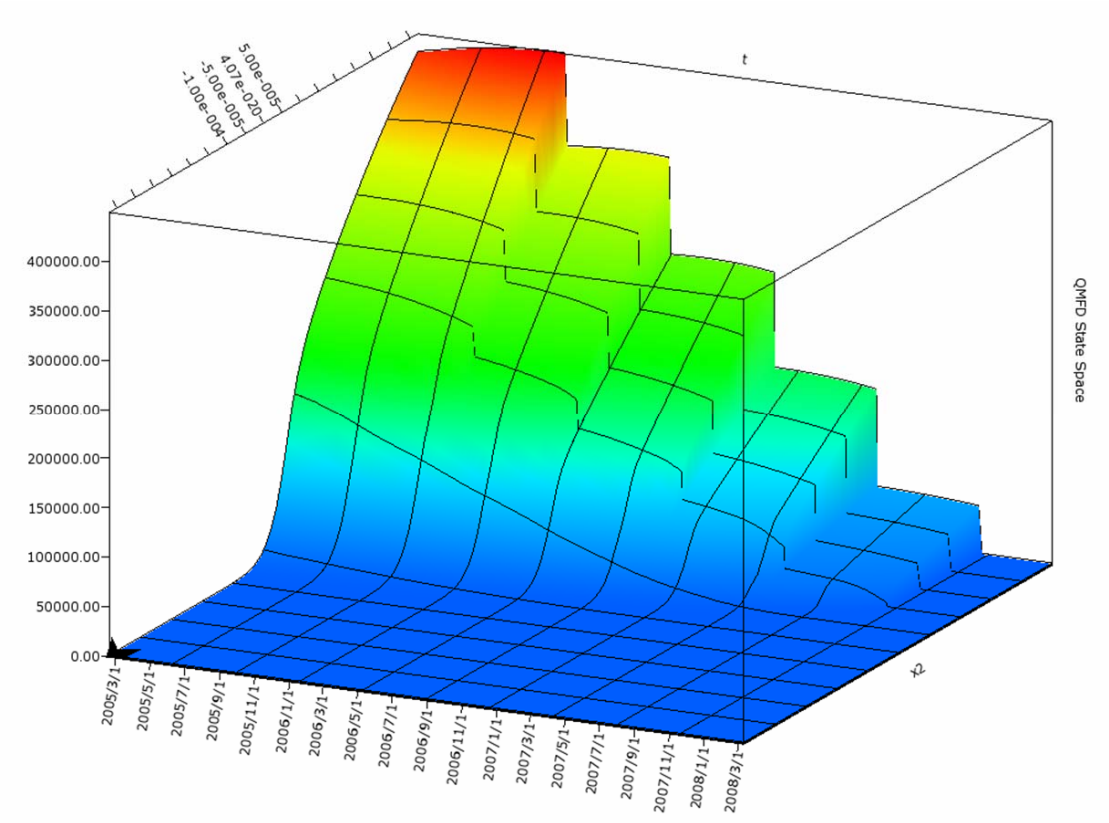
at each `dtAddCashflow` (the date at which the cash flow becomes part of the forward-starting swap), and

$$V(r,t) = \max\left(V(r,t), F(r,t)\right)$$

on the exercise dates of the swaption. State space $V(r,t)$ remains zero for values of r for which the option will not be exercised and becomes the value of the remaining swap if the option is exercised.

Figure 5 shows $V(r,t)$ for a Bermudan swaption with semiannual exercise dates. Discrete jumps in V are evident when the option is exercised, increasing the value of the option.

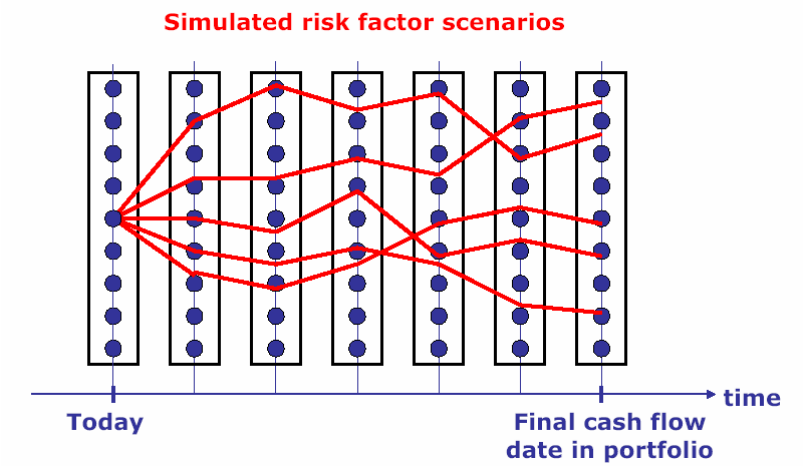
Figure 5: FD example: Bermudan swaption state space



2.2.2 Interpolating from the State Space and Calculating PE

After the state space for the value of the instrument $V(r,t)$ is generated, the next step in determining the PFE of the portfolio is to determine the mark to market value of the instrument along each path of the simulated risk factors on each exposure date. Figure 6 illustrates the Monte Carlo paths superimposed on the state space.

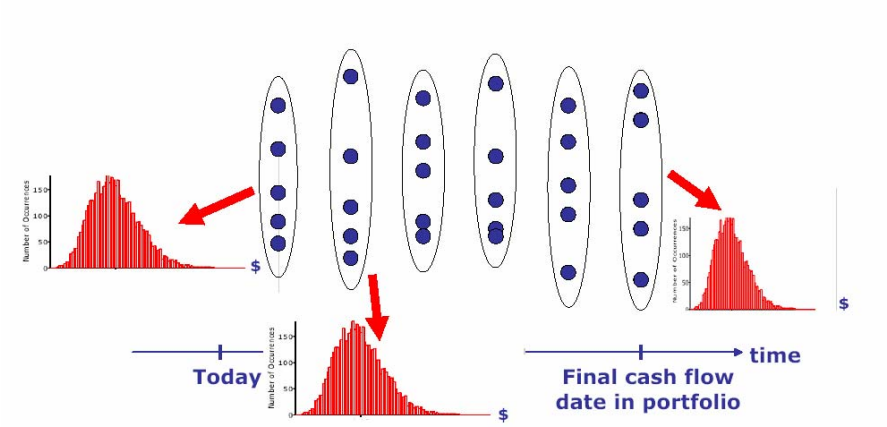
Figure 6: Risk factor simulation



For each simulation path, we calculate the MTM value of the instrument. This produces a distribution of prices on each exposure date. Figure 7 illustrates the formation of the MTM distribution by interpolation into the state space to extract an MTM value for each path on each date.

The potential exposure (95th percentile of the distribution) and expected exposure (mean of the distribution) are easily determined from the distribution of instrument values on each date.

Figure 7: Price distributions



If the portfolio contains multiple instruments, the price distributions for all instruments are stored and a PFE for the entire portfolio is calculated. This portfolio-wide aggregation is done by the QSF; for more details, see *GSF User Guide and Data Reference*.

The remainder of this section describes the following two issues that occur in calculating the MTM values on exposure dates:

- Because the state space at any time contains only the value of forward-starting cash flows, we must track what occurs during a coupon period within

which a PFE exposure date falls (for both the structured coupon and the funding coupon).

- We must define how to interpolate into the state space for each simulated path.

2.2.2.1 Calculating Upcoming Cash Flows Not Included in the State Space

On a given PE date, the state space on that date contains only the value of forward-starting cash flows. This occurs for two reasons. First, if the cash flow depends on an observable that is set at the beginning of the coupon period, and because the state space is populated by stepping backwards in time, then on any given date throughout the coupon period, the value of the upcoming cash flow is not yet known (the fixing date has not yet been reached by the FD solver). The second reason is to allow exercise conditions to be correctly handled. Even if the value of the upcoming cash flow is known (because the observable is set in-arrears or there is no market observable needed), QuIC Mechanics FD does not add the cash flow to the state space until it becomes part of a forward-starting coupon.

The MTM value of the instrument on a given cash flow date is therefore given by

$$\text{Price}(\omega,t) = \text{DCF}(\omega,t) + V(\omega,t)$$

where $\text{DCF}(\omega,t)$ is the value of the upcoming cash flow discounted to the valuation date.

The present value of the upcoming cash flow depends on the recent history of the simulation path ω, that is, what the last reset rate $r(\omega)$ was for the floating payment.

The PFE profile is generated by stepping forward in time. The calculated reset rates $r(\omega)$ are stored for each path so that on each valuation date the present value of the upcoming cash flow can be calculated and added to the value of the remaining cash flows (interpolated from the state space).

In some situations, not only the upcoming cash flow but also the next cash flow need to be added to the value interpolated from the state space. This can occur when the PE date falls in a fixing lag – between the fixing date and accrual start date of a coupon. QuIC Mechanics FD automatically determines when this is required.

QuIC Mechanics FD also keeps track of early exercise along each simulation path. On each exercise date, it compares the continuation value $V(\omega)$ to the exercise value (for example, zero for a swap, value of the forward swap for a swaption). For a given path, if exercise is optimal, the exercise decision is stored for that path. This information is then used to determine MTM values on future dates.

Consider the previous example of a callable swap that has two call dates. Depending on the path, some trades may exercise on the first call date and have zero value thereafter, others may exercise on the second call date, and still others may never exercise. This path dependency is tracked through the

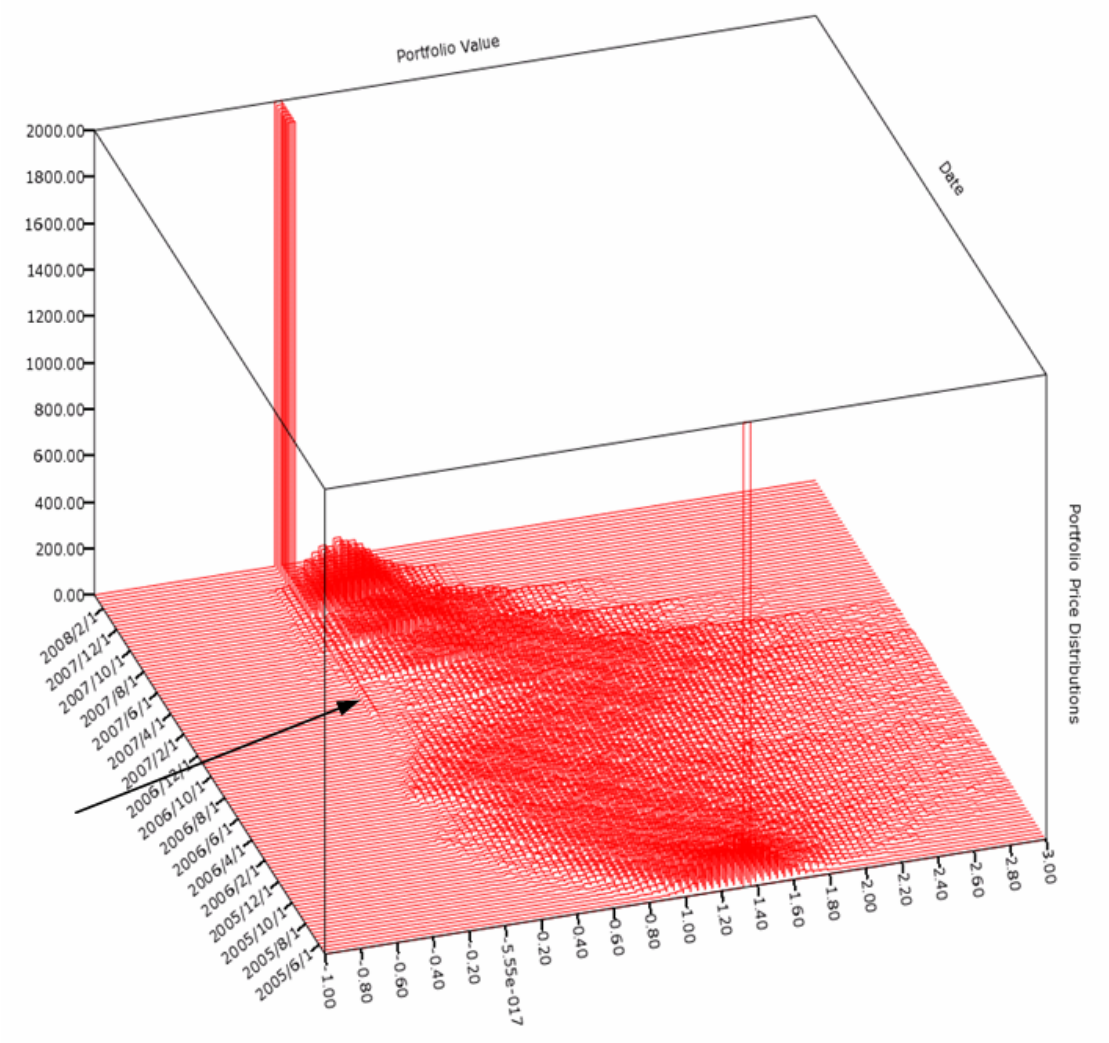
simulation and is efficiently handled in a vectorized fashion within the QuIC Platform. In particular, the grid-based conditional functionality of the QuIC Script language allows for efficient calculation of this path dependency. The following line of code illustrates how the grid of swap values for each path is updated for calls:

```
gPrice_ := gbPrevExercise_ ? gExerciseValue : gContinuationValue;
```

All grids in this equation have a scenario axis. For, scenarios for which `gbPrevExercise_` is true, `gPrice_` becomes the exercise value (=0[$] for a swap). For scenarios for which `gbPrevExercise_` is still false, `gPrice_` is the continuation value of the swap.

Figure 8 is a three-dimensional histogram plot of the instrument price distribution in time. Along the time axis one can see the redistribution of scenarios after call dates and cash flow dates. The arrow points to simulation paths that have been binned to zero after a call date—that is, paths for which gbPrevExercise = true.

Figure 8: FD example: Price distributions



2.2.2.2 Interpolating from the State Space

The second issue in determining the MTM values on PE dates is interpolating into the state space. For each path ω, what is the value $V(r)$? This is determined by defining a mapping between the simulated yield curve for path ω and the state space. This mapping depends on the model. For the G2++ model, we pick two tenor points of the simulated yield curve and find the x and y values for which the calibrated G2++ model replicates the simulated discount factors for these tenor points. See Section 2.3.3 on page 23 for details of the mapping for the G2++ model.

2.3 Pricing Model: G2++

QuIC Mechanics FD uses the two-additive-factor Gaussian model (G2++) to simulate short rates. This section describes the basis of the model, how it is calibrated, how it is mapped to the FD state space, and the analytic solution for range accrual swaps.

2.3.1 Description

In G2++, the dynamics of the instantaneous short rate process under a risk-adjusted measure Q is given by the following equation.

$$r(t) = x(t) + y(t) + \varphi(t), \quad r(0) = r_0$$

The processes $x(t)$ and $y(t)$ satisfy (for $t \geq 0$)

$$dx(t) = -ax(t)dt + \sigma dW_1(t), \quad x(0) = 0$$

$$dy(t) = -by(t)dt + \eta dW_2(t), \quad y(0) = 0$$

where (W_1, W_2) is a two-dimensional Brownian motion with instantaneous correlation ρ in

$$dW_1(t)dW_2(t) = \rho dt$$

The parameters a, b, σ, η and r_0 are positive constants and ρ satisfies $-1 \leq \rho \leq 1$. The function φ is deterministic and well defined over the time interval $[0, T^*]$, where T^* is a given time horizon.

To price derivatives using the G2++ model, a PDE formulation can be obtained by applying the Feynman-Kac theorem which results in the following PDE for the value of the instrument P:

$$P_t = \big(x + y + \varphi(t)\big)P + axP_x + byP_y - \tfrac{1}{2}\sigma^2 P_{xx} - \sigma\eta P_{xy} - \tfrac{1}{2}\eta^2 P_{yy} = 0$$

After the parameters a, b, σ, η and ρ are obtained from calibration, the function $\varphi(t)$ can be calculated from the discount factor curve.

2.3.2 Calibration

The parameters a, b, σ, η and ρ are derived by calibrating the G2++ model to caplets or European swaptions. This is done by performing a least squares optimization to minimize the sum of errors squared between analytical and model prices. In addition, we restrict the parameters a, b, σ, $\eta > 0$ and $-1 \le \rho \le 1$ by using the following mappings:

$$x \to x^2 \text{ and}$$
$$x \to \tfrac{2}{\pi}\arctan(x) \text{ for } \rho$$

The optimization methods available are Levenberg-Marquardt and the `minmd_f` optimization routine that is built in to QuIC Script. The latter uses gradient and Hessian approximations. These routines are only local optimization procedures that require an initial guess for the parameters to be fitted. The initial values of the parameters are provided by the user, and are used as model parameters if the optimization fails. Because local optimization routines are being used, we are not guaranteed to find a global minimum.

2.3.2.1 Calibration to European Swaptions

To calibrate to European swaptions, we minimize the squared sum of differences between the market price and the model price of these swaptions. The former is calculated using the Black formula for European options and the latter is calculated using the following formula[2]:

$$\mathrm{ES}(0,T,\mathcal{T},N,X,\omega) =$$
$$N\omega Z(0,T)\int_{-\infty}^{+\infty}\frac{\exp\left(-\frac{1}{2}\left(\frac{x-\mu_x}{\sigma_x}\right)^2\right)}{\sigma_x\sqrt{2\pi}}\left[\phi\left(-\omega h_1(x)\right)-\sum_{i=1}^{n}\lambda_i x e^{\kappa_i(x)}\phi\left(-\omega h_2(x)\right)\right] \quad [1]$$

This equation prices a European swaption with strike X, maturity T and notional N, which gives the holder the option to enter at time $t_0 = T$ into an interest rate swap with payment times $\mathcal{T} = \{t_1,\ldots,t_n\}$, $t_1 > T$, where the holder pays (receives) at the fixed rate X and receives (pays) LIBOR set in advance (at times $t_0,t_1,\ldots,t_{n-1}$). We denote by τ_i the year fraction from t_{i-1} to t_i, $i = 1,\ldots,n$, and set $c_i = X\tau_i$ for $i = 1,\ldots,n-1$ and $c_n = 1 + X\tau_n$. Also, ϕ denotes the Gaussian cumulative density function (CDF).

In equation [1], ω = 1 and ω = −1 would denote a payer and receiver swaption respectively. We also have:

[2] D. Brigo and F. Mercurio, *Interest Rate Models: Theory and Practice*, Springer-Verlag (2001).

$$h_1(x) = \frac{\bar{y} - \mu_y}{\sigma_y\sqrt{1-\rho_{xy}^2}} - \frac{\rho_{xy}(x-\mu_x)}{\sigma_x\sqrt{1-\rho_{xy}^2}}$$
$$h_2(x) = h_1(x) + B(b,T,t_i)\rho_y\sqrt{1-\rho_{xy}^2}$$
$$\lambda_i = c_i A(T,t_i)e^{-B(a,T,t_i)x}$$
$$\kappa_i(x) = -B(b,T,t_i)\left[\mu_y - \tfrac{1}{2}(1-\rho_{xy}^2)\sigma_y^2 B(b,T,t_i) + \rho_{xy}\sigma_y\frac{x-\mu_x}{\sigma_x}\right]$$

$\bar{y} = \bar{y}(x)$ is the unique solution of

$$\sum_{i=1}^{n} c_i A(T,t_i)e^{-B(a,T,t_i)x - B(b,T,t_i)\bar{y}} = 1$$

and we have

$$\mu_x = -M_x^T(0,t)$$
$$\mu_y = -M_y^T(0,t)$$
$$\sigma_x = \sigma\sqrt{\frac{1-e^{-2aT}}{2a}} \qquad [2]$$
$$\sigma_y = \eta\sqrt{\frac{1-e^{-2bT}}{2b}}$$
$$\rho_{xy} = \frac{\rho\sigma\eta}{(a+b)\sigma_x\sigma_y}\left[1-e^{-(a+b)T}\right]$$

$$M_x^T(s,t) = \left(\frac{\sigma^2}{a^2} + \rho\frac{\sigma\eta}{ab}\right)\left[1-e^{-a(t-s)}\right] - \frac{\sigma^2}{2a^2}\left[e^{-a(T-t)} - e^{-a(T+t-2s)}\right] - \frac{\rho\sigma\eta}{b(a+b)}\left[e^{-b(T-t)} - e^{-bT-at+(a+b)s)}\right]$$
$$M_y^T(s,t) = \left(\frac{\eta^2}{b^2} + \rho\frac{\sigma\eta}{ab}\right)\left[1-e^{-b(t-s)}\right] - \frac{\eta^2}{2b^2}\left[e^{-b(T-t)} - e^{-b(T+t-2s)}\right] - \frac{\rho\sigma\eta}{a(a+b)}\left[e^{-a(T-t)} - e^{-aT-bt+(a+b)s)}\right]$$

For more information, see Chapter 4 of Brigo and Mercurio.[2]

2.3.2.2 Calibration to Caplets

Instead of calibrating to European swaptions, you can calibrate the model to caplets. These caplets can be treated as one-period swaptions, and therefore the European swaption analytical formula can be used to price them.

2.3.3 Mapping to State Space

The PE scenarios are mapped to the G2++ model using two points on the simulated yield curve. The current implementation requires the use of the 3-month point and the 5-year point.

The mapping is performed as follows:

1. At time t for each PE scenario, find the 3-month zero coupon bond (ZCB) price, $P^{SIM}(t; t+3m)$, and the 5-year ZCB price, $P^{SIM}(t; t+5y)$, from the simulated yield curve.

2. Use the equation for ZCB prices in the G2++ model to solve for the values of factors x and y in the G2++ model that reproduce the 3-month and 5-year ZCB prices.

2.3.4 Analytic Formula for Range Accrual Swaps

QuIC Mechanics FD provides the option to price range accrual coupons by using an analytic solution for the range accrual factor. This analytic solution applies only when the monitored rate is a LIBOR. This section describes the derivation of this solution.

The calculation of a range accrual coupon at time T_{N-1} involves finding the present value of the expected cash flow at time T_N:

$$PV_{T_{N-1}}[CF(T_N)] =$$
$$Z(x_{T_{N-1}}, y_{T_{N-1}}, T_{N-1}, T_N)\hat{E}_{T_{N-1}}\left[\alpha_{N-1}\mathcal{N}\left(\frac{1}{|D_{N-1}|}\sum_{d\in D_{N-1}} 1_{\{LB\le R_d(x_d,y_d)\le UB\}}\right)\right]$$

where α_{N-1} is the accrual factor between T_{N-1} and T_N, $\mathcal{N}$ is the notional, D_{N-1} is the set of monitoring dates between T_{N-1} and T_N, $R_d(x_d, y_d)$ is the LIBOR measured at monitoring date d, and LB and UB are the lower and upper bounds of the range accrual region.

The expectation that the LIBOR is within the range accrual region can be calculated using the conditional distribution

$$\begin{aligned}\hat{E}_{T_{N-1}}&\left[1_{\{LB\le R(x_d,y_d,d)\le UB\}}\right]\\ &= \hat{P}_{T_{N-1}}\left[LB \le R_d(x_d, y_d) \le UB\right]\\ &= \iint_{x,y\in G} f_{T_{N-1},d}(x,y)dxdy\end{aligned}$$

where G is the region in x,y space where the LIBOR falls between the upper and lower bounds. The region G can be calculated as follows.

First, note that the integral can be broken into 2 separate integrals:

$$\iint\limits_{x,y\in G} f_{\tau_{N-1,d}}(x,y)dxdy = \iint\limits_{x,y\in G_1} f_{\tau_{N-1,d}}(x,y)dxdy - \iint\limits_{x,y\in G_2} f_{\tau_{N-1,d}}(x,y)dxdy$$

where G_1 is the region where R_d is below the upper bound UB and G_2 is the region where R_d is below the lower bound LB. We now calculate the region G_k where R_d is below some bound K.

The LIBOR observed at monitoring date T_d is given by

$$R_d = \frac{1}{\alpha_d}\left[\frac{1}{P(T_d,T_d^*)} - 1\right]$$

where T_d^* is the date when the monitoring rate expires (for example, $T_d^* = T_d + 1$ year for a 12-month LIBOR monitoring rate) and α_d is the accrual factor between T_d and T_d^*.

The zero coupon bond price $P(t,T)$ for time T observed at time t is given by

$$P(t,T) = A(t,T)\exp\{-B(a,t,T)x(t) - B(b,t,T)y(t)\}$$

where

$$A(t,T) = \frac{P^M(0,T)}{P^M(0,t)}\exp\left\{\tfrac{1}{2}\left[V(t,T) - V(0,T) + V(0,t)\right]\right\}$$

$$B(z,t,T) = \frac{1-e^{-z(T-t)}}{z}$$

and

$$\begin{aligned} V(t,T) = {} & \tfrac{\sigma^2}{a^2}\left[T - t + \tfrac{2}{a}e^{-a(T-t)} - \tfrac{1}{2a}e^{-2a(T-t)} - \tfrac{3}{2a}\right] \\ & + \tfrac{\eta^2}{b^2}\left[T - t + \tfrac{2}{b}e^{-b(T-t)} - \tfrac{1}{2b}e^{-2b(T-t)} - \tfrac{3}{2b}\right] \\ & + 2\rho\tfrac{\sigma\eta}{ab}\left[T - t + \frac{e^{-a(T-t)}-1}{a} + \frac{e^{-b(T-t)}-1}{b} - \frac{e^{-(a+b)(T-t)}-1}{a+b}\right] \end{aligned}$$

where a, b, σ, η, and ρ are parameters of the G2++ model, and $P^M(0,T)$ and $P^M(0,t)$ are zero time discount factors.

Setting $R_d = K$ and using the equation for $P(t,T)$, we can solve for the values of x and y where $R_d = K$. The solution is the following line:

$$y = C_1 x + C_2$$
$$C_1 = -\frac{B(a, T_d, T_d^*)}{B(b, T_d, T_d^*)}$$
$$C_2 = \frac{\ln\left\{A(T_d, T_d^*)\left[1+\alpha_d K\right]\right\}}{B(b, T_d, T_d^*)}$$

The integral for the probability that $R_d < K$ is now given by

$$\iint_{x,y\in G_K} f_{\tau_{N-1,d}}(x,y)dxdy = \int_{-\infty}^{\infty}\left[\int_{-\infty}^{C_1 x+C_2} f_{\tau_{N-1,d}}(x,y)dy\right]dx$$

Using the integral

$$\int_{-\infty}^{\infty} \phi(L_1 + L_2 z)\frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz \quad = \quad \phi\left(\frac{L_1}{\sqrt{1+L_2^2}}\right)$$

where $\phi(z)$ is the cumulative normal distribution function, the closed form expression for the probability that the LIBOR $R_d < K$ is as follows:

$$\hat{P}_{T_{N-1}}\left[R_d(x_d, y_d) \le K\right]$$
$$= \iint_{x,y\in G_K} f_{T_{N-1,d}}(x,y)dxdy$$
$$= \phi\left(\frac{H_2}{\sqrt{H_1^2 - 2H_1\rho_{xy} + 1}}\right)$$

where

$$H_1 = \frac{\sigma_x}{\sigma_y} C_1$$
$$H_2 = \frac{1}{\sigma_y}\left(C_2 + \mu_x C_1 - \mu_y\right)$$

and μ_x, μ_y, σ_x, σ_y, and ρ_{xy} are defined in Equations [2].

We can now calculate the range accrual coupon using this formula.

3 QUIC MECHANICS FD DESIGN

The QuIC Mechanics FD product is a tool for pricing instruments using finite difference (FD) methods and evaluating the potential future exposure (PFE) of the portfolio. While the previous chapter described FD pricing, PFE calculations, and details of the pricing model, this chapter describes the design of the code framework.

The goal of the QuIC Mechanics FD software design is to achieve a framework within which a wide variety of instruments can be priced with maximal reuse of code, as well as a framework within which new instruments can be added with minimal effort. Such a framework allows for efficient expansion as new instruments enter the portfolio.

QuIC Mechanics FD works in conjunction with the QuIC Simulation Framework (QSF), another component of the QuIC Analytics product set. QSF controls the breakup of the portfolio into counterparty and netting groups as well as the aggregation of mark to market (MTM) distributions across these groups and exposure dates. A QSF instrument is a plug-in to the QSF that defines the financial instrument, the pricing model, FD state space generation, and MTM valuation.

The design of the QuIC Mechanics FD product results in a framework within which a QSF instrument can be written in an efficient manner. It uses object-oriented programming to represent aspects of the QSF instrument as classes which interact through generic interfaces. A class encapsulates both attributes (data) and behaviour (functions). A class can act as a base or parent from which a more specialized subclass or child class inherits its basic attributes and behaviour. Hence, as new subclasses of these base classes are designed, they fit into the framework with little or no change to the rest of the framework.

The main tasks of instrument definition, pricing model definition and calibration, FD state space generation, and mark to market valuation are implemented by four main classes: QFDEvent, QModelFD, QFDStateSpace, and QFDValuator. A QSF instrument is designed by constructing objects of these classes.

3.1 QFDEvent

To attain a framework that handles a large number of instrument types, we divide the aspects of a financial instrument into its basic components which we call *events*. The QFDEvent base class has three subclasses: QFDEventCoupon, QFDEventExercise, and QFDEventNotionalCashFlow. An instrument is defined by combining arrays of these events as needed to build up the components of the instrument such as coupon payments and exercise decisions.

QFDEventCoupon is a coupon type event. The accrual of interest on a given notional for a given period of time in accordance with a given rate definition (such as fixed rate, leveraged floating rate, or range accrual). A collection of these individual coupon events can be combined together to describe a leg (structured or funding) of the swap instrument.

A large class of complex instruments can be described by grouping together different QFDEventCoupon components as needed to describe the instrument. Development of a new kind of instrument is often nothing more than designing a new subclass of QFDEventCoupon that implements the particulars of the new coupon payment.

The QFDObservable class supports event classes by representing the features and calculation of a market observable such as a LIBOR. Each coupon event can depend on one or more QFDObservable objects. By abstracting market-observable information from the coupon information, we have coupon events that are general for any kind of market observable. Expanding QuIC Mechanics FD to include new kinds of observables (for example, CMS rate or CMS spread) is then reduced to designing new subclasses of QFDObservable, which can be seamlessly incorporated into existing coupon events without adjustments to the QFDEventCoupon code.

QFDEventExercise is an exercise type event—the occurrence of an exercise decision on a given date (for example, to cancel a swap or enter a swaption). To build exercise decisions into our instrument, we create QFDEventExercise objects that describe how an exercise should be made.

QFDEventNotionalCashFlow is an event that describes the payment of a fixed amount on a given date (for example, a notional payment at the end of a bond).

This modular object-oriented approach to instrument definition allows for maximal code reuse as code in a subclass can be inherited from the base class. Furthermore, these classes represent only specific aspects of the financial instrument, and they are independent of the pricing model, details of state space generation, and MTM valuation. Hence, when a new event class is designed to handle a new event type (for a new instrument) it can be plugged into the framework with minimal revalidation of the framework because the other code in the framework is not modified during the course of the expansion.

3.2 QModelFD

QModelFD is a class that describes the interest rate models available for pricing the instrument. A model object is created within the QSF instrument, and then the market data is passed to its calibration function which carries out the calibration of the model.

When new models are desired, they can be added to the framework by designing a subclass of QModelFD that describes the features of the new model. As long as the new subclass has the correct interface (all context functions of the class correctly defined) then the new model can be plugged into the framework seamlessly. Validation and testing of the new model can be done independently of event and state space generation code.

3.3 QFDStateSpace

QFDStateSpace is a class that handles the setup of the FD solver. This object interfaces with a QModelFD object and arrays of QFDEvent objects to retrieve

from these objects the information that it needs to set up the FD solver and generate the state space for the instrument value. From the model it retrieves the PDE, while from the events is retrieves functions that describe how to update and reset the state space on cash flow dates and exercise dates. This object facilitates interpolation into the state space through a function called `GetInterpolatedValue_`.

3.4 QFDValuator

QFDValuator is a class whose functions are designed to be called within the context of the QSF instrument. The valuator functions are called for each PFE date and their job is to determine the distribution of MTM values on the given date. The valuator facilitates the interpolation into the state space to determine the value of future cash flows (through interaction with the QFDStateSpace object). It also tracks fixings and exercises for each scenario and uses this information to determine the value of upcoming cash flows (through interaction with the QFDEvent objects).

Subclasses of QFDValuator handle different combinations of interpolation into state spaces and summing of additional cash flows. For example a callable swap needs a different valuator than a swaption because the latter has two state spaces to interpolate from and the former has only one.

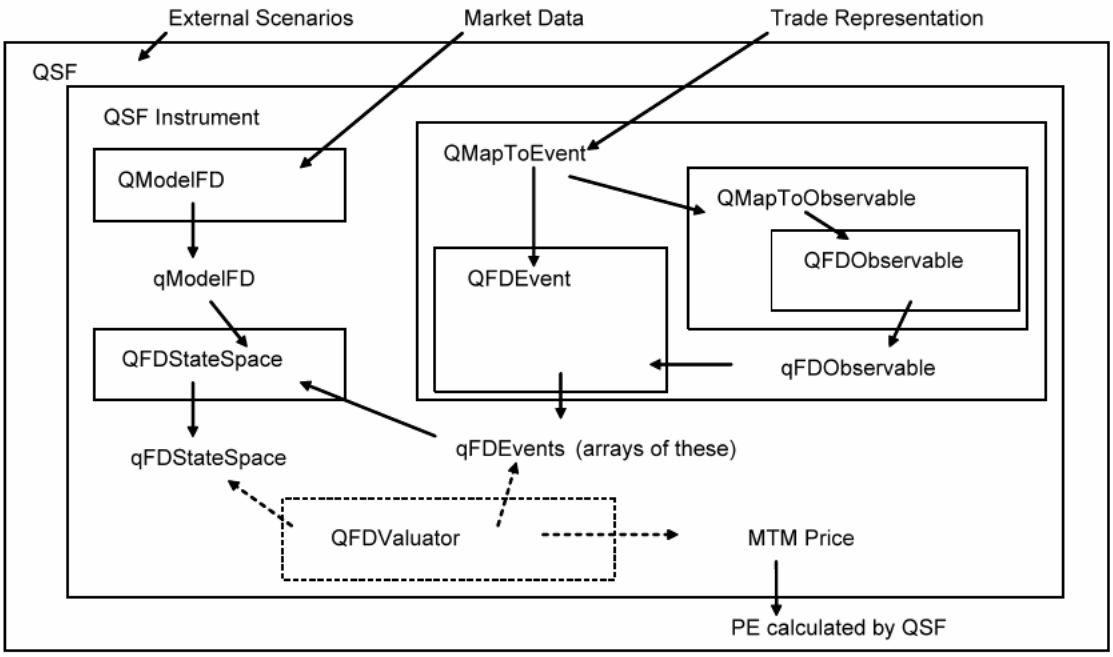
3.5 Generic QSF Instrument

Given these classes, designing a QSF instrument is relatively straightforward. One simply creates the array of event objects to describe the instrument, a model object to define the pricing model, and then passes these to the state space object to generate the state space. Finally, given the type of instrument to value, the proper valuator is defined and executed by the QSF on each exposure date, returning the distribution of MTMs for that date.

While this approach to designing instruments is efficient, the QuIC Mechanics FD framework goes one step further to define a generic QSF instrument. What "generic" means here is that the instrument itself does not define the type of model, type of valuator, or types of events. Instead, these elements are dictated by text-based inputs from the user. The single QuIC Mechanics FD QSF instrument can value and calculate PFE on any instrument that can be described by events within the event library, and it can use any model of the QModelFD base class and any valuator of the QFDValuator base class.

Figure 9 is a schematic diagram that shows how these objects are created and interact with each other to achieve a QSF instrument. The diagram uses the convention that class names begin with "Q" (the upper case letter) and object names are the same as the class of which they are an instance except that they begin with "q" (the lower case letter).

Figure 9: QuIC Mechanics FD software architecture



The class QMapToEvent has not yet been described. This class is a wrapper around the QFDEvent class. QMapToEvent accomplishes a mapping between a given text-based trade representation and the inputs to the QFDEvent objects. This wrapper class isolates the events from a given trade representation. If a new trade representations is desired, a wrapper class analogous to QMapToEvent can be designed to accommodate the trade representation, and no changes to the underlying events are required.

Referring to Figure 9, the trade representation is a text file that contains data structures to specify the events of the trade and the required inputs to those events. The wrapper class QMapToEvent facilitates the creation of the QFDObservable objects and the QFDEvent objects as dictated by the data in the trade representation. This class creates arrays of QFDEvent objects which are available to interact with other objects.

Market data curves are input to the QSF through an HDF5 database file. These curves are passed to a QModelFD object to calibrate the model. A QFDStateSpace object is created and the PDE is solved and the state space of the value of the instrument is generated through interaction with the QModelFD object and arrays of QFDEvent objects.

In the final step a valuator is registered into the QSF instrument context. The `GetMTMValue_` function of the valuator is executed at each PFE date to calculate the distribution of MTM values for each instrument of the portfolio. Finally, aggregation and reporting of results is done by the QSF.

Part 2: Using QuIC Mechanics FD

4 SETTING UP A TRADE

This chapter describes how to prepare the data files to analyze a portfolio of trades with the QuIC Mechanics FD solution. It illustrates the workflow with a complete example.

4.1 Overview

The QuIC Mechanics FD solution uses the capabilities of the QuIC Simulation Framework (QSF) with some extensions. To analyze a trade or a portfolio of trades, you enter data into the following two files:

- Transaction file—Each record (line) is a sequence of fields that specify one transaction. Some fields are references to data structures in the auxiliary transaction data file.

- Auxiliary transaction data file—Contains named data structures such as lists and maps that are more complex than a single field.

To be accessible to your driver script, these files must be in a directory that is in the set of data search paths of the QuIC Workbench application. For the details of setting the search paths, see *QuIC Workbench 2.5 – User Guide*.

Both files are text files in the comma-separated value (CSV) format. The next section provides some general advice for editing CSV files. The rest of the chapter describes the example and how you organize the data in these two files.

4.2 Editing CSV Files

You can edit the transaction file and the auxiliary transaction data file with any plain text editor, and that is the safest approach. However, in the Windows environment, the default application that opens CSV files is likely to be Microsoft Excel.

If you choose to edit CSV files with Excel, you need to be aware of the following potential problems that can occur and, where possible, take steps to avoid them:

- Excel does not include commas for empty fields at the end of a record.

- Excel generates extra commas at the end of a record if there is an unintended non-empty cell to the right of the actual data fields. For example, a single space in a cell to the right has this effect, but is not visually evident.

- Excel may change the format of date fields.

You can best avoid the comma problems by using a plain text editor rather than Excel. If you do not have empty trailing fields, however, you can use Excel if you have the correct configuration.

Note: If you use Excel to edit CSV files, be sure to save the files in CSV format, not in Excel format.

To use Excel successfully for QuIC CSV files, ensure that it uses the proper date format. All dates in the QuIC environment must be in the YYYY/MM/DD format (for example, `2006/05/08` for May 8, 2006). If Excel has the wrong date format, it may change the format of your existing dates when you save the file and render the file inoperable.

On some platforms, you can set the correct date format as the default for your computer, which Excel will recognize and use. The following procedure applies to the Windows XP operating system.

To set the default date format for your computer:

1. Navigate to the system Control Panel and open **Regional and Language Options**.
2. Click the **Regional Options** tab, click **Customize...**, and then click the **Date** tab.
3. In the **Short date format** list, select `yyyy/MM/dd`.
4. In the **Date separator** list, select the slash (`/`).

   Observe that the **Short date sample** text box shows the current date in the correct format.
5. Click **OK**, and then **OK** again.

4.3 Terms of the Example

The example is a swap in which you are the counterparty. The swap has the following main terms:

- The structured leg has an inverse floating coupon rate of 5.25% – LIBOR, with a cap of 6.0% and a floor of 3.0%.
- The funding leg coupon rate is LIBOR with no spread. The notional is 1,000,000 and is the same in every period.
- The day count convention for both legs is ACT365.
- All input dates are already adjusted for weekends and holidays.
- All amounts are in U.S. dollars.

For coupon start and end dates and fixing dates, see the maps in Figure 11.

4.4 Preparing the Transaction File

The transaction file is a CSV file whose general structure is defined by the QSF. Each line of the file defines one trade with a sequence of fields which are either simple data types (integers, real numbers, Booleans, or strings), or references to more complex data types. The complex data types include market data curves and auxiliary data (lists and maps). The references to these complex data types are simply strings that must match a key in the actual data structure in another file.

Each QSF instrument has its own specification for the required fields. For a general description of transaction files and the rules that apply to them, see *GSF User Guide and Data Reference*.

The present example consists of a single trade which is specified by the instrument named IRGenericInstrument. For the definitions of all the transaction input fields for IRGenericInstrument, see Section 5.1.1 on page 39.

Figure 10 shows the contents of a transaction file that implements this example (`GenericInstrument_CFIF_ML1234.csv`). This represents a single line in the file (because the portfolio consists of a single trade), but the contents wrap in this figure to make the whole line visible.

Figure 10: Example: Transaction file

```
CFIF_ML1234, IRGenericInstrument, CPY1, GROUP1, PAY,
  CFIF_ML1234_StructuredLegInfo, CFIF_ML1234_FundingLegInfo,
  CFIF_ML1234_ExerciseInfo, , , , Swaption, USDG2PPCurveList,
  USD.Exchange.USD
```

Table 2 explains each of these values.

Table 2: Explanation of values in sample trade

Fld #	Sample value	Description
1	`CFIF_ML1234`	Transaction ID
2	`IRGenericInstrument`	Instrument name
3	`CPY1`	Counterparty ID
4	`GROUP1`	Netting group ID
5	`PAY`	Whether you pay or receive the structured leg
6	`CFIF_ML1234_StructuredLegInfo`	Name of the structured leg map in the auxiliary data file
7	`CFIF_ML1234_FundingLegInfo`	Name of the funding leg map in the auxiliary data file
8	`CFIF_ML1234_ExerciseInfo`	Name of the exercise map in the auxiliary data file

Fld #	Sample value	Description
9	None	(Optional) Name of the autoredeem map in the auxiliary data file
10	None	(Optional) Name of the notional payments map in the auxiliary data file
11	None	(Optional) Name of the solver preference map in the auxiliary data file
12	`Swaption`	Type of option
13	`USDG2PPCurveList`	Name of a market data curve list in the auxiliary data file
14	`USD.Exchange.USD`	(Optional) QSF-style name of an exchange rate curve

4.5 Preparing the Auxiliary Transaction Data File

The transaction input line for this trade (see Figure 10) includes the following items that are references to lists or maps rather than simple data types or other types of reference:

- `CFIF_ML1234_StructuredLegInfo`
- `CFIF_ML1234_FundingLegInfo`
- `CFIF_ML1234_ExerciseInfo`
- `USDG2PPCurveList`

Each of these maps and lists must be present in the auxiliary data file, which is also known as the schedules file. For a comprehensive description of auxiliary data files and the rules that apply to them, see *GSF User Guide and Data Reference*. In summary, the auxiliary data file is a CSV file in which one record (line) defines one list, or a sequence of lines defines a map. Lines that begin with two slashes (`//`) are comment lines, which QuIC applications ignore.

In the case of lists, the first field is the list name, the second is the list type, and the remaining fields are the list data. The structure of the list data depends on the list type. The QSF has several predefined list types, and applications can define additional types.

In the case of maps, the first and last lines of the sequence delimit the map structure and the intermediate lines provide the data for named parameters that are relevant to the type of map and the specific objects that it references. The order of the named parameters (keys) within a map is not important. For details about the general map format and the specific map types, see Section 5.2 on page 40.

These map types (observable, leg, exercise, autoredemption, and notional payment) are only structures for organizing the inputs to the FD solution; they do

not necessarily correspond directly to single entities in the solution. They are a convenient way to input the data for multiple instances of the underlying observable and event entities. In the QuIC Mechanics FD solution (in contrast to the QuIC Mechanics Monte Carlo solution), an observable or event object represents an observable or event that occurs at a single time, not a whole series of observables or events. The inputs for the FD solution represent all events in the life of the trade explicitly. In this sequence of events, each event can be of a different type.

In these input maps, each parameter of a map may consist of a list of values that occur over time. The position of each item in the list represents an event time, and the values for the different parameters of an event (or observable) must be in corresponding positions. For example, all parameter values for the fifth event must be in the fifth data position. Even if a parameter has the same value in every event, you must repeat the value for each event.

Within one input map, you must include all the relevant parameters for all the observables (in the case of an observables map) or for all the events (in the case of a leg map) that occur in the trade. If a parameter is not relevant in a particular position (that is, for a particular event), you can omit the value but you must include the comma that separates the position from the next one.

Figure 11 shows the contents of the auxiliary data file for this example (`Schedules_GenericInstrument_ML1234.csv`). The parameters in the maps in this example are the ones that specify adjusted dates. A different set of parameters is required to specify unadjusted dates; for the details, see Section 6.1 on page 45 and Section 6.2 on page 47. In the actual file, each list and each parameter within a map occupies a single line, but in this figure the lines wrap to make them visible. This figure includes annotations which follow the lines that they explain.

Figure 11: Example: Auxiliary transaction data file

```
// Market Data Curves
USDG2PPCurveList, CurveList, YieldCrv, SwaptionVol, USD.Yield.USD,
USDSwapVol.SwaptionVolMtx.USD
```

This is a list of type `CurveList`, which specifies a set of internal names of curves followed by a corresponding set of QSF-style database curve names (the keys to retrieve the curves from the market and historical database).

```
/// CFIF_ML1234 trade

CFIF_ML1234_StructObs, {
strName, AS, CFIF_ML1234_StructObs_2005/12/13,
  CFIF_ML1234_StructObs_2006/03/13, CFIF_ML1234_StructObs_2006/09/11,
  CFIF_ML1234_StructObs_2007/03/12, CFIF_ML1234_StructObs_2007/09/10,
  CFIF_ML1234_StructObs_2008/03/10, CFIF_ML1234_StructObs_2008/09/08
strType, AS, Libor, Libor, Libor,
  Libor, Libor, Libor,
  Libor
strAdjusted, AS, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted,
  Adjusted
dtFixing, AD, 2005/12/13, 2006/03/13, 2006/09/11, 2007/03/12, 2007/09/10,
  2008/03/10, 2008/09/08
dtLiborStart, AD, 2005/12/15, 2006/03/15, 2006/09/13, 2007/03/14,
  2007/09/12, 2008/03/12, 2008/09/10
```

```
dtLiborEnd, AD, 2006/03/15, 2006/09/13, 2007/03/14, 2007/09/12, 2008/03/12,
  2008/09/10, 2009/03/11
strDaycountLibor, AS, act365, act365, act365, act365, act365, act365, act365
pHistCurve, AC, 3mLibor.HistoricalRates.USD, 3mLibor.HistoricalRates.USD,
  3mLibor.HistoricalRates.USD, 3mLibor.HistoricalRates.USD,
  3mLibor.HistoricalRates.USD, 3mLibor.HistoricalRates.USD
}
```

The `CFIF_ML1234_StructObs` map defines the LIBOR for the structured leg. For details, see Section 6.1.1 on page 45. Each parameter provides 7 values that correspond to the 7 events in the structured leg of the trade.

```
CFIF_ML1234_StructuredLegInfo, {
strName, AS, CFIF_ML1234_Struct_2006/03/15, CFIF_ML1234_Struct_2006/09/13,
  CFIF_ML1234_Struct_2007/03/14, CFIF_ML1234_Struct_2007/09/12,
  CFIF_ML1234_Struct_2008/03/12, CFIF_ML1234_Struct_2008/09/10,
  CFIF_ML1234_Struct_2009/03/11
strType, AS, CouponFloat, CouponFloat,
  CouponFloat, CouponFloat,
  CouponFloat, CouponFloat,
  CouponFloat
strAdjusted, AS, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted,
  Adjusted
rNotional, AR, 1000000.0, 1000000.0, 1000000.0, 1000000.0, 1000000.0,
  1000000.0, 1000000.0
dtAccrualStart, AD, 2005/12/15, 2006/03/15, 2006/09/13, 2007/03/14,
  2007/09/12, 2008/03/12, 2008/09/10
dtAccrualEnd, AD, 2006/03/15, 2006/09/13, 2007/03/14, 2007/09/12,
  2008/03/12, 2008/09/10, 2009/03/11
dtPayment, AD, 2006/03/15, 2006/09/13, 2007/03/14, 2007/09/12, 2008/03/12,
  2008/09/10, 2009/03/11
mpFDObservable, AL, CFIF_ML1234_StructObs, CFIF_ML1234_StructObs,
  CFIF_ML1234_StructObs, CFIF_ML1234_StructObs, CFIF_ML1234_StructObs,
  CFIF_ML1234_StructObs, CFIF_ML1234_StructObs
strCpnDaycount, AS, act365, act365, act365, act365, act365, act365, act365
rFixedRate, AR, 0.0525, 0.0525, 0.0525, 0.0525, 0.0525, 0.0525, 0.0525
rLeverageFactor, AR, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0
rCap, AR, 0.06, 0.06, 0.06, 0.06, 0.06, 0.06 0.06
rFloor, AR, 0.03, 0.03, 0.03, 0.03, 0.03, 0.03, 0.03
}
```

The `CFIF_ML1234_StructuredLegInfo` map defines the structured leg of the trade. All events in this leg are of type QFDEventCouponFloat, so all the parameters in this map are for this event class. For details, see Section 6.2.4 on page 51. The structured leg consists of 7 floating coupon events.

```
CFIF_ML1234_FundObs, {
strName, AS, CFIF_ML1234_FundObs_2005/12/13, CFIF_ML1234_FundObs_2006/03/13,
  CFIF_ML1234_FundObs_2006/06/12, CFIF_ML1234_FundObs_2006/09/11,
  CFIF_ML1234_FundObs_2006/12/11, CFIF_ML1234_FundObs_2007/03/12,
  CFIF_ML1234_FundObs_2007/06/11, CFIF_ML1234_FundObs_2007/09/10,
  CFIF_ML1234_FundObs_2007/12/10, CFIF_ML1234_FundObs_2008/03/10,
  CFIF_ML1234_FundObs_2008/06/09, CFIF_ML1234_FundObs_2008/09/08,
  CFIF_ML1234_FundObs_2008/12/08
strType, AS, Libor, Libor, Libor,
  Libor, Libor, Libor,
  Libor, Libor, Libor,
  Libor, Libor, Libor,
  Libor
strAdjusted, AS, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted,
  Adjusted, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted
dtFixing, AD, 2005/12/13, 2006/03/13, 2006/06/12, 2006/09/11, 2006/12/11,
  2007/03/12, 2007/06/11, 2007/09/10, 2007/12/10, 2008/03/10, 2008/06/09,
  2008/09/08,2008/12/08
dtLiborStart, AD, 2005/12/15, 2006/03/15, 2006/06/14, 2006/09/13,
  2006/12/13, 2007/03/14, 2007/06/13, 2007/09/12, 2007/12/12, 2008/03/12,
```

```
  2008/06/11,
  2008/09/10, 2008/12/10
dtLiborEnd, AD, 2006/03/15, 2006/06/14, 2006/09/13, 2006/12/13, 2007/03/14,
  2007/06/13, 2007/09/12, 2007/12/12, 2008/03/12, 2008/06/11,  2008/09/10,
  2008/12/10, 2009/03/11
strDaycountLibor, AS, act365, act365, act365, act365, act365, act365,
  act365, act365, act365, act365, act365, act365, act365
pHistCurve, AC, 3mLibor.HistoricalRates.USD,,,,,,,,,,,
}
```

The `CFIF_ML1234_FundObs` map defines the LIBOR for the funding leg. For details, see Section 6.1.1 on page 45. Each parameter provides 13 values that correspond to the 13 events in the funding leg of the trade.

```
CFIF_ML1234_FundingLegInfo, {
strName, AS, CFIF_ML1234_Fund_2006/03/15, CFIF_ML1234_Fund_2006/06/14,
  CFIF_ML1234_Fund_2006/09/13, CFIF_ML1234_Fund_2006/12/13,
  CFIF_ML1234_Fund_2007/03/14, CFIF_ML1234_Fund_2007/06/13,
  CFIF_ML1234_Fund_2007/09/12, CFIF_ML1234_Fund_2007/12/12,
  CFIF_ML1234_Fund_2008/03/12, CFIF_ML1234_Fund_2008/06/11,
  CFIF_ML1234_Fund_2008/09/10, CFIF_ML1234_Fund_2008/12/10,
  CFIF_ML1234_Fund_2009/03/11
strType, AS, CouponFloat, CouponFloat,
  CouponFloat, CouponFloat,
  CouponFloat, CouponFloat,
  CouponFloat, CouponFloat,
  CouponFloat, CouponFloat,
  CouponFloat, CouponFloat,
  CouponFloat
strAdjusted, AS, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted,
  Adjusted, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted
rNotional, AR, 1000000.0, 1000000.0, 1000000.0, 1000000.0, 1000000.0,
  1000000.0, 1000000.0, 1000000.0, 1000000.0, 1000000.0, 1000000.0,
  1000000.0, 1000000.0
dtAccrualStart, AD, 2005/12/15, 2006/03/15, 2006/06/14, 2006/09/13,
  2006/12/13, 2007/03/14, 2007/06/13, 2007/09/12, 2007/12/12, 2008/03/12,
  2008/06/11, 2008/09/10, 2008/12/10
dtAccrualEnd, AD, 2006/03/15, 2006/06/14, 2006/09/13, 2006/12/13,
  2007/03/14, 2007/06/13, 2007/09/12, 2007/12/12, 2008/03/12, 2008/06/11,
  2008/09/10, 2008/12/10, 2009/03/11
dtPayment, AD, 2006/03/15, 2006/06/14, 2006/09/13, 2006/12/13, 2007/03/14,
  2007/06/13, 2007/09/12, 2007/12/12, 2008/03/12, 2008/06/11, 2008/09/10,
  2008/12/10, 2009/03/11
mpFDObservable, AL, CFIF_ML1234_FundObs,  CFIF_ML1234_FundObs,
  CFIF_ML1234_FundObs, CFIF_ML1234_FundObs, CFIF_ML1234_FundObs,
  CFIF_ML1234_FundObs, CFIF_ML1234_FundObs, CFIF_ML1234_FundObs,
  CFIF_ML1234_FundObs, CFIF_ML1234_FundObs, CFIF_ML1234_FundObs,
  CFIF_ML1234_FundObs, CFIF_ML1234_FundObs
strCpnDaycount, AS, act365, act365, act365, act365, act365, act365, act365,
  act365, act365, act365, act365, act365, act365
}
```

The `CFIF_ML1234_FundingLegInfo` map defines the funding leg of the trade. All events in this leg are of type QFDEventCouponFloat, so all the parameters in this map are for this event class. For details, see Section 6.2.4 on page 51. The funding leg consists of 13 floating coupon events.

```
CFIF_ML1234_ExerciseInfo, {
strName, AS, CFIF_ML1234_Exer_2006/03/05, CFIF_ML1234_Exer_2006/09/03,
  CFIF_ML1234_Exer_2007/03/04, CFIF_ML1234_Exer_2007/09/02,
  CFIF_ML1234_Exer_2008/03/02, CFIF_ML1234_Exer_2008/08/31
strType, AS, Exercise, Exercise, Exercise,
  Exercise, Exercise, Exercise
strAdjusted, AS, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted, Adjusted
dtExercise, AD, 2006/03/05, 2006/09/03, 2007/03/04, 2007/09/02, 2008/03/02,
  2008/08/31
```

```
dtLastCoupon, AD, 2006/03/15, 2006/09/13, 2007/03/14, 2007/09/12,
  2008/03/12, 2008/09/10
}
```

The `CFIF_ML1234_ExerciseInfo` map defines the early exercise events of the trade. All events in this map are of type QFDEventExercise, so all the parameters in this map are for this event class. For details, see Section 6.2.7 on page 61.

4.6 Updating the Solution Driver Script

After you prepare the information for all trades in the two input data files and store them in a directory that QuIC Workbench searches, be sure to update your solution driver script to identify the correct file names. You are then ready to run the script to evaluate the portfolio.

5 REFERENCE: INPUT FILES

This chapter provides reference information about the contents of the input data files of the QuIC Mechanics FD solution.

5.1 Transaction File

The transaction file of the QuIC Simulation Framework (QSF) is a comma-separated value (CSV) file in which each non-comment line represents one trade. The instrument of which a trade is an instance determines the input fields for the trade.

This section describes the inputs for each available QuIC Mechanics FD instrument. In this release, the only instrument is IRGenericInstrument.

5.1.1 IRGenericInstrument

The IRGenericInstrument instrument is a generic QSF instrument that can represent a variety of interest rate swaps. It uses the finite difference methods of QuIC Mechanics FD for pricing.

Table 3 provides the details of the QSF transaction file input fields according to the definition of IRGenericInstrument. The data types are described in detail in *GSF User Guide and Data Reference*.

Table 3: IRGenericInstrument transaction inputs

Fld #	Input Field (internal name)	Description	Type	Constraints
1	`strID_`	Transaction ID	String	
2	`StrInstrument_`	Instrument name	String	`IRGenericInstrument`
3	`strCounterparty_`	Counterparty ID. For details, see *GSF User Guide and Data Reference*.	String	
4	`strNettingGroupID_`	Netting group ID. For details, see *GSF User Guide and Data Reference*.	String	
5	`strPayReceive_`	Whether you pay or receive the structured leg	String	One of: • `PAY` • `RECEIVE`
6	`mpStructLegInfo_`	Reference to the relevant structured leg map.	List/Map	Name of a leg map in the auxiliary data file.
7	`mpFundingLegInfo_`	Reference to the relevant funding leg map.	List/Map	Name of a leg map in the auxiliary data file.

Fld #	Input Field (internal name)	Description	Type	Constraints
8	`mpExerciseInfo_`	(Optional) Reference to the relevant exercise map.	List/Map	Name of an exercise map in the auxiliary data file.
9	`mpAutoRedeemInfo_`	(Optional) Reference to the relevant autoredemption map.	List/Map	Name of an autoredemption map in the auxiliary data file.
10	`mpNotionalCF_`	(Optional) Reference to the relevant notional payments map.	List/Map	Name of a notional payments map in the auxiliary data file.
11	`mpModelInfo_`	(Optional) Reference to the relevant solver preference map.	List/Map	Name of a solver preference map in the auxiliary data file.
12	`strOptionType_`	(Optional) Type of option. The default is `CallableSwap`.	String	One of: • `Swaption` • `CashSwaption` • `CallableSwap`
13	`pMktDataInfo_`	Reference to the relevant market data curve list.	List/Map	Name of a curve list in the auxiliary data file.
14	`pFXInfo_`	(Optional) Reference to the relevant exchange rate curve.	Curve	QSF-style name of a curve in the market and historical database.

5.2 Auxiliary Transaction Data File

In the QSF, the auxiliary data file is a text file in CSV format that consists of named lists of data of various types. Some of the list types are schedules, and the auxiliary data file is sometimes called the schedules file.

For details of the standard QSF list types, see *GSF User Guide and Data Reference*. QuIC Mechanics FD requires only the `CurveList` list type. This list identifies the data curves to retrieve from the market and historical database. This database is a standard database whose structure is defined by the QSF; see the same QSF manual for information about the required structure of market data CSV files. The QuIC Mechanics FD software distribution includes a script that converts the CSV file or files into an HFD5 database file, which is the required format for input to a pricing run.

In addition to a standard QSF list, the auxiliary data file for QuIC Mechanics FD also contains structures called *maps*. The following types of maps are supported in the auxiliary data file:

- Observable maps
- Leg maps
- Exercise maps

- Autoredemption maps
- Notional payment maps
- Solver preference maps

The following subsections specify the general format of an auxiliary data map and describe the usage of the various map types.

5.2.1 General Structure of Auxiliary Data Maps

Figure 12 shows the general format of an auxiliary data map.

Figure 12: General format of an auxiliary data map

```
Name,{
Key1, DataType1, Data1
Key2, DataType2, Data2
...
}
```

The elements of a map are as follows:

Name is an arbitrary string to identify the map. You refer to the map from the transaction file by this name. Typically, map names relate to the trade and to the type of map.

KeyN is a specific string that is the name of a QuIC parameter. The reference information in the remaining chapters identifies the valid keys in each case. Keys (and their associated type and data) can appear in any order within a map. You can omit a key line if it has a default value and you want to use the default.

DataTypeN is a string that identifies the type of data to follow. Table 4 shows the valid data type codes.

Table 4: Data types in auxiliary data maps

Base data type code	Meaning	Code for array of values of this type
B	Boolean	AB
C	QSF-style curve name	AC
D	Date	AD
H	Name of holiday set in the holiday database	AH
L	Name of list in the auxiliary data file	AL
N	Integer	AN
R	Real number	AR

Base data type code	Meaning	Code for array of values of this type
S	String	AS

DataN is the data value or values (separated by commas) for that key. You must provide a data value for each event that occurs during the life of the trade, even if the values are all the same.

5.2.2 Observable Maps

An observable map is a structure in the auxiliary transaction data file in which you specify the parameters of multiple observable objects.

Each coupon event that depends on a market-observable quantity refers to an observable map that defines the specific observable. The parameters in an observable map must contain data entries (possibly empty) that correspond to all events in the leg map that references the observable map. The QuIC Mechanics FD application unpacks these to create the individual observable objects.

An observable map in the auxiliary transaction data file must follow the general format specified in Section 5.2.1 on page 41.

For the parameters to include in an observable map, see the relevant observable class in Section 6.1 on page 45.

5.2.3 Leg Maps

A leg map is a structure in the auxiliary transaction data file in which you specify the parameters of all coupon events in one leg of a trade. If the leg consists of more than one type of coupon event, the leg map must contain the union of all relevant parameters from all the relevant coupon event types.

A leg map in the auxiliary transaction data file must follow the general format specified in Section 5.2.1 on page 41.

For the parameters to include in a leg map, see the relevant coupon event classes in Section 6.2 on page 47.

5.2.4 Exercise Maps

An exercise map is a structure in the auxiliary transaction data file in which you specify the parameters of all exercise events in a trade. If the trade has more than one type of exercise event, the exercise map must contain the union of all relevant parameters from all the relevant exercise event types.

An exercise map in the auxiliary transaction data file must follow the general format specified in Section 5.2.1 on page 41.

For the parameters to include in an exercise map, see the relevant exercise event classes in Section 6.2 on page 47.

5.2.5 Autoredemption Maps

Autoredemption is not available in the present release.

5.2.6 Notional Payment Maps

A notional payment map is a structure in the auxiliary transaction data file in which you specify the parameters of all notional payment events in a trade.

A notional payment map in the auxiliary transaction data file must follow the general format specified in Section 5.2.1 on page 41.

For the parameters to include in a notional payment map, see the relevant notional payment event class in Section 6.2 on page 47.

5.2.7 Solver Preferences Maps

A solver preferences map is a structure in the auxiliary transaction data file in which you can specify parameters to control the solution process.

Some solver preferences group together naturally into subsets. In these cases, QuIC Mechanics FD defines an additional associated map for the subset. To include such parameters, simply add the associated map with a unique name to the auxiliary transaction data file and enter the name of the associated map as the value of the relevant parameter in the main solver preferences map. The associated map must precede the main solver preferences map.

A solver preferences map and its associated maps in the auxiliary transaction data file must follow the general format specified in Section 5.2.1 on page 41.

For the parameters to include in a solver preferences map and its associated maps, see Chapter 7 on page 65.

Figure 13 shows an example of a solver preferences map that specifies both a calibration option and grid concentration information (in addition to other parameters). These two types of information both require associated maps.

Figure 13: Example: Solver preferences with associated maps

```
CalibOptions, {
MXITER, AR, 250
}
```

This is a calibration options map. The solver preferences map below refers to it.

```
Factor1Point, AR, 0.0
Factor2Point, AR, 0.0
alpha1, AR, 0.1
alpha2, AR, 0.1
```

These are ordinary auxiliary data lists. The grid concentration map below refers to them.

```
GridConcInfo, {
aaPts_, AL, Factor1Point, Factor2Point
aaAlpha_, AL, alpha1, alpha2
}
```

This is a grid concentration map. The solver preferences map below refers to it.

```
SolverPrefsRAcomp, {
anFactorGridPoints_, AR, 60, 60
mpCalibOptions_, L, CalibOptions
bUseGridConc, B, TRUE
mpGridConcInfo_, L, GridConcInfo
nStdDevs, N, 8
strSolver, S, _xfnModExpCN
rMaxTimeStep, R, 182
}
```

This is the main solver preferences map. It refers to both the calibration options map and grid concentration map above.

6 REFERENCE: CLASSES

This chapter provides reference information about the classes of the QuIC Mechanics FD solution. The primary purpose is to identify the parameters that you specify in input files to create instances of these classes for the IRGenericInstrument instrument.

6.1 Observables

An observable object represents both historical values and future simulated values of a market-observable quantity. Other classes can seamlessly request the value of an observable on any date without regard to whether the date is in the past or future.

This section describes the inputs for each available observable class. In this release, the only observable class is QFDObservableLibor.

6.1.1 QFDObservableLibor

The QFDObservableLibor class represents a LIBOR.

Table 5 identifies all the possible object creation parameters of the QFDObservableLibor class in alphabetical order by their map keys.

Table 5: QFDObservableLibor parameters (all)

Key	Type in input map	Description	Constraints
`dtEnd`	AD	Unadjusted end date of the LIBOR period. Include this if `strFixingType` is `in-arrears`.	
`dtFixing`	AD	Adjusted fixing date of the LIBOR.	
`dtLiborEnd`	AD	Adjusted end date of the LIBOR period.	
`dtLiborStart`	AD	Adjusted start date of the LIBOR period.	
`dtStart`	AD	Unadjusted start date of the LIBOR period.	
`nLagFixing`	AR	Number of days before `dtStart` that the LIBOR fixes. The default is 0.	
`nTenorRefRate`	AR	Tenor of the LIBOR in months.	

Key	Type in input map	Description	Constraints
`pg1dtFixingHoliday`	AH	Reference to the set of holidays for the fixing date. Include this if `nLagFixing` > 0.	Key in the holidays database.
`pg1dtRefRateHoliday`	AH	Reference to the set of holidays for the LIBOR.	Key in the holidays database.
`pHistCurve`	AC	Reference to the relevant historical fixings curve.	QSF-style name of a curve in the market and historical database.
`rHistRate`	AR	Historical LIBOR. QM-FD uses this only if `pHistCurve` is void.	
`strAdjusted`	AS	Specifies whether the input parameters are the subset for explicit adjusted dates or the subset for unadjusted dates (see Table 6 below).	One of: • `Adjusted` • `Unadjusted`
`strDaycountLibor`	AS	Day count convention for the LIBOR.	A value of the EDaycount enumeration in QuIC Script.
`strFixingType`	AS	Whether coupon rates are fixed at the beginning of the period (`in-advance`) or at the end of the period (`in-arrears`).	One of: • `in-advance` (default) • `in-arrears`
`strName`	AS	Arbitrary name to identify the observable in error messages.	
`strRefRateBDC`	AS	Business day convention for the LIBOR.	A value in the EBusDayConv enumeration in QuIC Script.
`strType`	AS	Type of observable. The value is the class-specific part of the name of the QM-FD observable class.	`Libor`

You can specify a QFDObservableLibor object in input files in either of two ways (according to the value of the `strAdjusted` parameter):

- With explicit dates already adjusted for holidays and lags
- With implicit dates from association with a coupon

Each case requires a distinct set of input parameters and also may include some common parameters. Table 6 identifies the relevant parameters for each case. The "Automatic" notation means that the instrument automatically supplies the

relevant dates from the coupon; do not include this parameter in the observables map.

Table 6: QFDObservableLibor parameter selection

Key	To create with explicit adjusted dates	To create with implicit coupon dates
`dtEnd`		Automatic (if `strFixingType` is `in-arrears`)
`dtFixing`	✓	
`dtLiborEnd`	✓	
`dtLiborStart`	✓	
`dtStart`		Automatic
`nLagFixing`		Optional
`nTenorRefRate`		✓
`pg1dtFixingHoliday`		✓ (if `nLagFixing` > 0)
`pg1dtRefRateHoliday`		✓
`pHistCurve`	Optional	Optional
`rHistRate`	Optional	Optional
`strDaycountLibor`	✓	✓
`strFixingType`		Optional
`strName`	Optional	Optional
`strRefRateBDC`		✓
`strType`	✓	✓

6.2 Events

An event object represents a unit of behaviour in the life of a trade—it is something that happens at a particular time. Some examples of events include:

- A fixed cash flow
- A coupon payment
- An early exercise decision

Table 7 identifies the event classes in the present version of the QuIC Mechanics FD solution.

Table 7: Event classes

Event class name	Description	Section Reference
`QFDEventCouponAnalyticRangeAccrualFixed`	Represents payment of a coupon with a fixed interest rate subject to a range condition on a reference LIBOR. Uses an analytic solution to evaluate the range factor.	6.2.1
`QFDEventCouponAnalyticRangeAccrualFloat`	Represents payment of a coupon with a floating interest rate subject to a range condition on a reference LIBOR. Uses an analytic solution to evaluate the range factor.	6.2.2
`QFDEventCouponFixed`	Represents payment of a coupon with a fixed interest rate.	6.2.3
`QFDEventCouponFloat`	Represents payment of a coupon with a floating interest rate. Depending on the parameter values, the event can be either a conventional or inverse floating coupon.	6.2.4
`QFDEventCouponRangeAccrualFixed`	Represents payment of a coupon with a fixed interest rate subject to a range condition on a reference LIBOR. The FD solver lands on every day of the period to evaluate the range factor.	6.2.5
`QFDEventCouponRangeAccrualFloat`	Represents payment of a coupon with a floating interest rate subject to a range condition on a reference LIBOR. The FD solver lands on every day of the period to evaluate the range factor.	6.2.6
`QFDEventExercise`	Represents a call or put.	6.2.7
`QFDEventNotionalCashFlow`	Represents a fixed payment on a specific date	6.2.8

This section describes the inputs for each available event class. The classes appear in alphabetical order. Table 8 defines the notation that appears in the formulas in this section.

Table 8: Notation in event descriptions

Symbol	Meaning
C	Coupon amount
N	Notional

Symbol	Meaning
τ	Year fraction
R	Fixed interest rate
r	Observable rate (for example, a LIBOR)
w	Weighting factor
s	Spread or margin
n	Number of days in a coupon period on which a condition was satisfied
m	Number of days in a coupon period

6.2.1 QFDEventCouponAnalyticRangeAccrualFixed

The QFDEventCouponAnalyticRangeAccrualFixed class is identical to the QFDEventCouponRangeAccrualFixed class except that it calculates n from an analytic formula.

The inputs for this class are identical to those for QFDEventCouponRangeAccrualFixed (see Section 6.2.5 on page 54) except that the value of `strType` must be `CouponAnalyticRangeAccrualFixed`.

6.2.2 QFDEventCouponAnalyticRangeAccrualFloat

The QFDEventCouponAnalyticRangeAccrualFloat class is identical to the QFDEventCouponRangeAccrualFloat class except that it calculates n from an analytic formula.

The inputs for this class are identical to those for QFDEventCouponRangeAccrualFloat (see Section 6.2.6 on page 57) except that the value of `strType` must be `CouponAnalyticRangeAccrualFloat`.

6.2.3 QFDEventCouponFixed

The QFDEventCouponFixed class represents the payment of a coupon with a fixed interest rate according to the following formula:

$$C = N\tau R$$

where the notation is as defined in Table 8 on page 48.

Table 9 identifies all the possible input parameters of the QFDEventCouponFixed class in alphabetical order by their map keys.

Table 9: QFDEventCouponFixed parameters (all)

Key	Type in input map	Description	Constraints
`bPay`	—	Whether the coupon is paid (`true`) or received (`false`).	
`dtAccrualEnd`	`AD`	Adjusted end date of the coupon period.	
`dtAccrualStart`	`AD`	Adjusted start date of the coupon period.	
`dtCouponEnd`	`AD`	Unadjusted end date of the coupon period.	
`dtCouponStart`	`AD`	Unadjusted start date of the coupon period.	
`dtPayment`	`AD`	Adjusted coupon payment date.	
`pg1dtCpnHoliday`	`AH`	Reference to the set of holidays for the coupon.	Key in the holidays database.
`rFixedRate`	`AR`	Coupon interest rate (R in the formula).	
`rNotional`	`AR`	Notional amount.	
`strAdjusted`	`AS`	Specifies whether the input parameters are the subset for explicit adjusted dates or the subset for unadjusted dates (see Table 10 below).	One of: • `Adjusted` • `Unadjusted`
`strCpnBDC`	`AS`	Business day convention for the coupon.	A value in the EBusDayConv enumeration in QuIC Script.
`strCpnDaycount`	`AS`	Day count convention.	A value of the EDaycount enumeration in QuIC Script.
`strName`	`AS`	Arbitrary name to identify the event in error messages.	
`strType`	`AS`	Type of event. The value is the class-specific part of the name of the QM-FD event class.	`CouponFixed`

You can specify a QFDEventCouponFixed object in input files in either of two ways (according to the value of the `strAdjusted` parameter):

- With explicit dates already adjusted for holidays and lags

- With nominal (unadjusted) dates

Each case requires a distinct set of input parameters and also may include some common parameters. Table 10 identifies the relevant parameters for each case. The "Automatic" notation means that the instrument automatically supplies the relevant information; do not include this parameter in the leg map.

Table 10: QFDEventCouponFixed parameter selection

Key	To create with explicit adjusted dates	To create with unadjusted dates
bPay	Automatic	Automatic
dtAccrualEnd	✓	
dtAccrualStart	✓	
dtCouponEnd		✓
dtCouponStart		✓
dtPayment	✓	✓
pg1dtCpnHoliday		✓
rFixedRate	✓	✓
rNotional	✓	✓
strAdjusted	✓	✓
strCpnBDC		✓
strCpnDaycount	✓	✓
strName	Optional	Optional
strType	✓	✓

6.2.4 QFDEventCouponFloat

The QFDEventCouponFloat class represents the payment of a coupon with a floating interest rate according to the following formula:

$$C = N\tau\left(wr + s\right)$$

where the notation is as defined in Table 8 on page 48.

This event can represent either a conventional or inverse floating coupon. To specify an inverse floater, enter a negative value of w. Optionally, the coupon rate is subject to a floor (minimum rate) and a cap (maximum rate).

Table 11 identifies all the possible input parameters of the QFDEventCouponFloat class in alphabetical order by their map keys.

Table 11: QFDEventCouponFloat parameters (all)

Key	Type in input map	Description	Constraints
`bPay`	—	Whether the coupon is paid (`true`) or received (`false`).	
`dtAccrualEnd`	AD	Adjusted end date of the coupon period.	
`dtAccrualStart`	AD	Adjusted start date of the coupon period.	
`dtCouponEnd`	AD	Unadjusted end date of the coupon period.	
`dtCouponStart`	AD	Unadjusted start date of the coupon period.	
`dtPayment`	AD	Adjusted coupon payment date.	
`mpFDObservable`	AL	Reference to the observable map that specifies the parameters of the observable rate (r in the formula).	Name of an observable map in the auxiliary data file.
`pg1dtCpnHoliday`	AH	Reference to the set of holidays for the coupon.	Key in the holidays database.
`pHistCurve`	AC	Reference to the relevant historical fixings curve.	QSF-style name of a curve in the market and historical database.
`rCap`	AR	Coupon rate cap. The default is `infinity`.	
`rFixedRate`	AR	Spread to apply to the observable rate (s in the formula). The default is 0.	
`rFloor`	AR	Coupon rate floor. The default is `-infinity`.	
`rLeverageFactor`	AR	Weighting factor to apply to the observable rate (w in the formula). The default is 1.	
`rNotional`	AR	Notional amount.	

Key	Type in input map	Description	Constraints
`strAdjusted`	AS	Specifies whether the input parameters are the subset for explicit adjusted dates or the subset for unadjusted dates (see Table 12 below).	One of: • `Adjusted` • `Unadjusted`
`strCpnBDC`	AS	Business day convention for the coupon.	A value in the EBusDayConv enumeration in QuIC Script.
`strCpnDaycount`	AS	Day count convention for the coupon.	A value of the EDaycount enumeration in QuIC Script.
`strName`	AS	Arbitrary name to identify the event in error messages.	
`strType`	AS	Type of event. The value is the class-specific part of the name of the QM-FD event class.	`CouponFloat`

You can specify a QFDEventCouponFloat object in either of two ways (according to the value of the `strAdjusted` parameter):

- With explicit dates already adjusted for holidays and lags
- With nominal (unadjusted) dates

Each case requires a distinct set of input parameters and also may include some common parameters. Table 12 identifies the relevant parameters for each case. The "Automatic" notation means that the instrument automatically supplies the relevant information; do not include this parameter in the leg map.

Table 12: QFDEventCouponFloat parameter selection

Key	To create with explicit adjusted dates	To create with unadjusted dates
`bPay`	Automatic	Automatic
`dtAccrualEnd`	✓	
`dtAccrualStart`	✓	
`dtCouponEnd`		✓
`dtCouponStart`		✓
`dtPayment`	✓	✓

Key	To create with explicit adjusted dates	To create with unadjusted dates
mpFDObservable	✓	✓
pg1dtCpnHoliday		✓
pHistCurve	Optional	Optional
rCap	Optional	Optional
rFixedRate	✓	✓
rFloor	Optional	Optional
rLeverageFactor	Optional	Optional
rNotional	✓	✓
strAdjusted	✓	✓
strCpnBDC		✓
strCpnDaycount	✓	✓
strName	Optional	Optional
strType	✓	✓

6.2.5 QFDEventCouponRangeAccrualFixed

The QFDEventCouponRangeAccrualFixed class represents the payment of a coupon with a fixed interest rate and which depends on a range condition on an observable reference rate (for example, a LIBOR) according to the following formula:

$$C = N\tau R\left(\frac{n}{m}\right)$$

where the notation is as defined in Table 8 on page 48. The value of n is determined by checking the following range condition on a specified observable on every day t in the coupon period:

$$RangeLower \le r(t) \le RangeUpper$$

Table 13 identifies all the possible input parameters of the QFDEventCouponRangeAccrualFixed class in alphabetical order by their map keys.

Table 13: QFDEventCouponRangeAccrualFixed parameters (all)

Key	Type in input map	Description	Constraints
`bPay`	—	Whether the coupon is paid (`true`) or received (`false`).	
`dtAccrualEnd`	`AD`	Adjusted end date of the coupon period.	
`dtAccrualStart`	`AD`	Adjusted start date of the coupon period.	
`dtCouponEnd`	`AD`	Unadjusted end date of the coupon period.	
`dtCouponStart`	`AD`	Unadjusted start date of the coupon period.	
`dtPayment`	`AD`	Adjusted coupon payment date.	
`mpFDObservableRA`	`AL`	Reference to the observable map that specifies the parameters of the observable rate for the range (r in the range formula).	Name of an observable map in the auxiliary data file.
`nMonitorEnd`	`AR`	Number of business days before the end of the period at which to stop monitoring. QM-FD uses the final fixing for the remaining days. The default is 0.	
`pg1dtCpnHoliday`	`AH`	Reference to the set of holidays for the coupon.	Key in the holidays database.
`pHistCurve`	`AC`	Reference to the relevant historical fixings curve.	QSF-style name of a curve in the market and historical database.
`rFixedRate`	`AR`	Coupon interest rate (R in the coupon formula).	
`rLowerBound`	`AR`	Lower bound of the range.	
`rNotional`	`AR`	Notional amount.	
`rUpperBound`	`AR`	Upper bound of the range.	
`strAdjusted`	`AS`	Specifies whether the input parameters are the subset for explicit adjusted dates or the subset for unadjusted dates (see Table 14 below).	One of: • `Adjusted` • `Unadjusted`

Key	Type in input map	Description	Constraints
strCpnBDC	AS	Business day convention for the coupon.	A value in the EBusDayConv enumeration in QuIC Script.
strCpnDaycount	AS	Day count convention.	A value of the EDaycount enumeration in QuIC Script.
strMonitorDayType	AS	Specifies which days of the range to monitor. The default is ALLDAYS.	One of: • ALLDAYS (every calendar day) • BUSDAYS (only business days) • PREVCAL (all days, but holidays use the fixing from the previous business day)
strName	AS	Arbitrary name to identify the event in error messages.	
strType	AS	Type of event. The value is the class-specific part of the name of the QM-FD event class.	CouponRangeAccrualFixed

You can specify a QFDEventCouponRangeAccrualFixed object in either of two ways (according to the value of the strAdjusted parameter):

- With explicit dates already adjusted for holidays and lags
- With nominal (unadjusted) dates

Each case requires a distinct set of input parameters and also may include some common parameters. Table 14 identifies the relevant parameters for each case. The "Automatic" notation means that the instrument automatically supplies the relevant information; do not include this parameter in the leg map.

Table 14: QFDEventCouponRangeAccrualFixed parameter selection

Key	To create with explicit adjusted dates	To create with unadjusted dates
bPay	Automatic	Automatic
dtAccrualEnd	✓	
dtAccrualStart	✓	
dtCouponEnd		✓
dtCouponStart		✓

Key	To create with explicit adjusted dates	To create with unadjusted dates
dtPayment	✓	✓
mpFDObservableRA	✓	✓
nMonitorEnd	Optional	Optional
pg1dtCpnHoliday		✓
pHistCurve	Optional	Optional
rFixedRate	✓	✓
rLowerBound	Optional	Optional
rNotional	✓	✓
rUpperBound	Optional	Optional
strAdjusted	✓	✓
strCpnBDC		✓
strCpnDaycount	✓	✓
strMonitorDayType	Optional	Optional
strName	Optional	Optional
strType	✓	✓

6.2.6 QFDEventCouponRangeAccrualFloat

The QFDEventCouponRangeAccrualFloat class represents the payment of a coupon with a floating interest rate and which depends on a range condition on an observable reference rate (for example, a LIBOR) according to the following formula:

$$C = N\tau\left(wr_1 + s\right)\left(\frac{n}{m}\right)$$

where the notation is as defined in Table 8 on page 48. The value of n is determined by checking the following range condition on a specified observable on every day t in the coupon period:

$$RangeLower \le r_2(t) \le RangeUpper$$

This event can represent either a conventional or inverse floating coupon. To specify an inverse floater, enter a negative value of w.

Table 15 identifies all the possible input parameters of the QFDEventCouponRangeAccrualFloat class in alphabetical order by their map keys.

Table 15: QFDEventCouponRangeAccrualFloat parameters (all)

Key	Type in input map	Description	Constraints
`bPay`	—	Whether the coupon is paid (`true`) or received (`false`).	
`dtAccrualEnd`	AD	Adjusted end date of the coupon period.	
`dtAccrualStart`	AD	Adjusted start date of the coupon period.	
`dtCouponEnd`	AD	Unadjusted end date of the coupon period.	
`dtCouponStart`	AD	Unadjusted start date of the coupon period.	
`dtPayment`	AD	Adjusted coupon payment date.	
`mpFDObservable`	AL	Reference to the observable map that specifies the parameters of the observable rate (r_1 in the coupon formula).	Name of an observable map in the auxiliary data file.
`mpFDObservableRA`	AL	Reference to the observable map that specifies the parameters of the observable rate for the range (r_2 in the range formula).	Name of an observable map in the auxiliary data file.
`nMonitorEnd`	AR	Number of business days before the end of the period at which to stop monitoring. QM-FD uses the final fixing for the remaining days. The default is 0.	
`pg1dtCpnHoliday`	AH	Reference to the set of holidays for the coupon.	Key in the holidays database.
`pHistCurve`	AC	Reference to the relevant historical fixings curve.	QSF-style name of a curve in the market and historical database.

Key	Type in input map	Description	Constraints
`rFixedRate`	AR	Spread to apply to the observable rate (s in the coupon formula). The default is 0.	
`rLeverageFactor`	AR	Weighting factor to apply to the observable rate (w in the coupon formula). The default is 1.	
`rLowerBound`	AR	Lower bound of the range.	
`rNotional`	AR	Notional amount.	
`rUpperBound`	AR	Upper bound of the range.	
`strAdjusted`	AS	Specifies whether the input parameters are the subset for explicit adjusted dates or the subset for unadjusted dates (see Table 16 below).	
`strCpnBDC`	AS	Business day convention for the coupon.	One of: • `Adjusted` • `Unadjusted`
`strCpnDaycount`	AS	Day count convention.	A value in the EBusDayConv enumeration in QuIC Script.
`strMonitorDayType`	AS	Specifies which days of the range to monitor. The default is `ALLDAYS`.	A value of the EDaycount enumeration in QuIC Script.
`strName`	AS	Arbitrary name to identify the event in error messages.	One of: • `ALLDAYS` (every calendar day) • `BUSDAYS` (only business days) • `PREVCAL` (all days, but holidays use the fixing from the previous business day)
`strType`	AS	Type of event. The value is the class-specific part of the name of the QM-FD event class.	`CouponRangeAccrualFloat`

You can specify a QFDEventCouponRangeAccrualFloat object in either of two ways (according to the value of the `strAdjusted` parameter):

- With explicit dates already adjusted for holidays and lags
- With nominal (unadjusted) dates

Each case requires a distinct set of input parameters and also may include some common parameters. Table 16 identifies the relevant parameters for each case. The "Automatic" notation means that the instrument automatically supplies the relevant information; do not include this parameter in the leg map.

Table 16: QFDEventCouponRangeAccrualFloat parameter selection

Key	To create with explicit adjusted dates	To create with unadjusted dates
bPay	Automatic	Automatic
dtAccrualEnd	✓	
dtAccrualStart	✓	
dtCouponEnd		✓
dtCouponStart		✓
dtPayment	✓	✓
mpFDObservable	✓	✓
mpFDObservableRA	✓	✓
nMonitorEnd	Optional	Optional
pg1dtCpnHoliday		✓
pHistCurve	Optional	Optional
rCap	Optional	Optional
rFixedRate	✓	✓
rFloor	Optional	Optional
rLowerBound	Optional	Optional
rNotional	✓	✓
rUpperBound	Optional	Optional
strAdjusted	✓	✓
strCpnBDC		✓
strCpnDaycount	✓	✓
strMonitorDayType	Optional	Optional

Key	To create with explicit adjusted dates	To create with unadjusted dates
`strName`	Optional	Optional
`strType`	✓	✓

6.2.7 QFDEventExercise

The QFDEventExercise class represents a call or a put.

Table 17 identifies the input parameters of the QFDEventExercise class in alphabetical order by their map keys. The "Automatic" notation means that the instrument automatically supplies the relevant information; do not include this parameter in the exercise map.

Table 17: QFDEventExercise parameters

Key	Type in input map	Description	Constraints
`dtExercise`	AD	Adjusted exercise date.	
`dtLastCoupon`	AD	For a callable swap: date of last cash flow upon call. For a swaption: start date of the swap. These must be adjusted dates.	
`rAmount`	AR	(Optional) Notional amount to exchange when the option is exercised. The default is 0.	
`strMaxMin`	AS	(Optional) Whether to take the maximum or minimum of the continuation value and exercise value of the trade. The default is `max`.	One of: • `max` • `min`
`strName`	AS	(Optional) Arbitrary name to identify the event in error messages.	
`strOptionType`	—	(Automatic) Type of option.	Do not include this parameter in an exercise map.
`strType`	AS	Type of event. The value is the name of the QM-FD event class.	`Exercise`

6.2.8 QFDEventNotionalCashFlow

The QFDEventNotionalCashFlow class represents a fixed payment on a specific date.

Table 18 identifies the input parameters of the QFDEventNotionalCashFlow class in alphabetical order by their map keys.

Table 18: QFDEventNotionalCashFlow parameters

Key	Type in input map	Description	Constraints
`dtPayment`	AD	Adjusted payment date.	
`rNotional`	AR	Amount of the payment.	
`strName`	AS	(Optional) Arbitrary name to identify the event in error messages.	
`strType`	AS	Type of event. The value is the name of the QM-FD event class.	`NotionalCashFlow`

7 REFERENCE: SOLVER PREFERENCES

A variety of parameters is available to control the solution process in QuIC Mechanics FD. You specify these in a solver preferences map (and optional associated maps) in the auxiliary transaction data file. The software uses default values for any solver preferences that you do not specify.

This chapter describes all the available parameters that you can specify in the main solver preferences map and in the following maps to which the main solver preferences map may refer:

- Grid concentration map
- Calibration options map

7.1 Parameters in the Main Solver Preferences Map

Table 19 defines the parameters that you can use in the main solver preferences map, in alphabetical order by their map keys. In this release, the default solver preferences are appropriate for the 2-factor G2++ model.

Table 19: Parameters in main solver preferences map

Key	Type in input map	Description	Constraints
`aInitialGuessG2PP`	`AR`	Initial guess for the calibrated values of the G2++ model parameters. Array of 5 parameter values as follows: • A • B • sigma • eta • rho The default is as follows: 1.0, 0.1, 0.02, 0.01, -0.5	Annualized values with no units. The value of rho is between -1 and 1.
`aMappingInfo_`	`AR`	Ordinates of the yield curve to use for mapping between external scenarios and the state space, in days. The default is 92 and 1825 (that is, 3 months, and 5 years, respectively).	
`anFactorGridPoints_`	`AN`	Number of grid points. The default is 50 for each factor.	
`bUseGridConc`	`B`	Whether to use grid concentration. The default is `false`.	

Key	Type in input map	Description	Constraints
mpCalibOptions_	L	Reference to a calibration options map; see Section 7.3 on page 65.	Name of a map in the auxiliary data file.
mpGridConcInfo_	L	Reference to a grid concentration map; see Section 7.2 on page 64.	Name of a map in the auxiliary data file.
nGridPointsAuxVar	AN	Number of grid points for the auxiliary variable axis (when the auxiliary variable is required). The default is 20.	
nStdDevs	AN	Number of standard deviations of the model factors to determine the minimum and maximum values of the axes. The default is 3 for each factor.	
rMaxTimeStep	AR	The maximum time step in days between two grid points. The default is 10.	
strModelType_	AS	Short rate model to use. The default is G2PP.	In this release, G2PP is the only model available.
strSolver	AS	Solver type. The default is _xfnModExpKrylov.	One of: • _xfnModExpKrylov (Krylov method) • _xfnModExpCN (Crank-Nicolson method) • _xfnModExpImp (Fully implicit method) • _xfnModExpRannacher (Rannacher method)

7.2 Parameters in a Grid Concentration Map

Table 20 defines the parameters that you can use in a grid concentration map, in alphabetical order by their map keys.

Table 20: Parameters in a grid concentration map

Key	Type in input map	Description	Constraints
aaAlpha_	AL	Parameters for each concentration point that control the degree of concentration. The default is 0.4.	

Key	Type in input map	Description	Constraints
`aaOnGrid_`	AL	Whether each concentration point is on a grid point (`true`) or is mid-way between grid points (`false`).	One of: • `true` • `false`
`aaPts_`	AL	Array of points for each model factor around which to concentrate the grid. The default for G2++ is the central point (x=0, y=0).	Any point in the factor space.

7.3 Parameters in a Calibration Options Map

Table 21 defines the parameters that you can use in a calibration options map, in alphabetical order by their map keys.

Table 21: Parameters in a calibration options map

Key	Type in input map	Description	Constraints
`AFCTOL`	R	Tolerance for absolute function convergence. The default is 1e-20.	
`MXITER`	N	Maximum number of iterations. The default is 200.	
`OPTMETH`	S	Optimization method. The default is `minmd_f`, the built-in QuIC Script optimizer.	One of: • `LevenbergMarquardt` • Any other value invokes the default
`RFCTOL`	R	Tolerance for relative function convergence. The default is 1e-6.	

APPENDIX A. ABBREVIATIONS

CSV	Comma-separated value (file format)
EPE	Expected potential exposure
FD	Finite difference (solver)
GSF	Generalized Simulation Framework (obsolete name for QSF)
LIBOR	London Interbank Offer Rate
PDE	Partial differential equation
PE	Potential exposure
PFE	Potential future exposure
QM-FD	QuIC Mechanics FD
QSF	QuIC Simulation Framework

INDEX

A

C

D

E

F