

Dual bootstrap in Quantlab

< Guide for use of the dual bootstrap functionality >

Version: 1.22

Last update: 2015-03-10

© 2002-2015 Algorithmica Research AB. All rights reserved.

Table of contents

1. Introduction	4
2. Creating a discount function	6
2.1. Wrapper functions	6
2.2. The disc_z_model object	8
2.3. The swap_curve_disc_ext object	12
2.4. Code examples.....	13
2.4.1. example1 - basic.....	13
2.4.2. example2	14
3. Creating a forward function	16
3.1. FRA/IRS wrapper function	16
3.2. The fwd_z_model object	18
3.3. The swap_curve_fwd_ext object	20
3.4. Creating synthetic short instruments (before the first FRA)	21
3.5. Forward index rate interpolation.....	22
3.6. The forward function <fwd_func>	22
3.7. FRA/IRS/BasisSwap(2Swap) wrapper function	23
3.8. The swap_curve_b2_ext object	24
3.9. FRA/IRS/BasisSwap(1Swap) wrapper function	25
3.10. The swap_curve_b1_ext object	27
3.11. Code examples.....	28
3.11.1. example1 - basic.....	28
3.11.2. example 2	29

4. Creating an fx implied discount function	31
4.1. Wrapper functions for currency basis swaps	31
4.2. The swap_curve_bc_ext object	34
4.3. Wrapper functions for currency fix-float swaps	35
4.4. The swap_curve_cff_ext object	39
5. Creating a swap surface	40
6. Curves.....	42
References.....	45

1. Introduction

This document describes how to use the dual curve bootstrap functionality in Quantlab 3.1. By dual curve bootstrap, we refer to the new paradigm in the swap market for curve construction where the forward index rates (projecting rates) and discount rates are separate curves. The “old style” curve construction used the same curve for the forward index rates and for discounting. The market standard discount curve now consists of OIS rates.

Curve construction in the swap market, in the new paradigm, means paying attention to the tenor of the forward index rates (i.e. tenor basis spreads) and carefully select the instruments from which the curves are created and the blending principles applied. Normally it is necessary to also create synthetic instruments (e.g. FRA and/or Deposits) in specific, basis curve consistent, ways to fill up some segments of the forward curve or discount curve. In addition, it is desirable to obtain smooth (forward) curves. Quantlab offers several interpolation models with excellent smoothness properties (e.g. Adams-Deventer Maximum Smoothness-model, Hagan-West forward monotone convex spline, tension splines, etc). It is also possible to apply Hyman filters to models where it is relevant. Furthermore, when creating the forward index rates, in Quantlab, the interpolation models operate directly on the tenor rate that is calculated, ensuring a “smooth” behavior of the forward rates and more direct control over the rates you are ultimately interested in. The swap curve in each currency is, in the new paradigm, best described as a swap-“surface” where the extra dimension is the tenor of the forward index rate. Quantlab also provides a swap-surface object from which interpolated rates for non-standard tenors can be obtained.

For bootstrapping the discounting curve (single curve bootstrap), Quantlab includes, in addition to all standard single curve bootstrap features, modeling with a step function between Central bank meeting dates while, if desired, simultaneously using a different interpolation model for the longer end of the curve.

Quantlab 3.1 encapsulates all the complexities involved and provides ways to create discount functions and forward curves in a very succinct way.

Typically the instruments used in dual bootstrap are (all supported):

1. OIS swaps
2. Deposits (normally synthetic)
3. Forward rate agreements (FRA) (real and synthetic)
4. Money market futures
5. IRS (fixed/floating interest rate swap)

6. Basis swaps - as 2 swaps

7. Basis swaps - as 1 swap

There are principally two ways to use the dual bootstrap functionality. One is to use a set of core Quantlab functions and classes. This will give the user maximum flexibility and control of every aspect of the curve building process. The other way is to use a set of wrapper functions that tries to provide “best-practice” curve building for a number of specific situations and therefore facilitates the creation of discount functions and forward functions. This document describes the latter. The code for the wrapper functions are available for viewing in Qlang, making it very easy for the user to create different versions as well as understanding the details involved.

A selection of the provided wrapper functions are explained in this document, as follows:

1. **disc_curve_create (base)**: builds a discount function, for example from OIS instruments,
2. **disc_curve_create (extended)**: builds a discount function with the possibility to extend the curve if it is shorter than the swap curve.
3. **fwd_curve_create**: builds a forward function from FRA/IRS (and synthetic instruments)
4. **fwd_b2_curve_create**: builds a forward function from FRA/IRS/Basis swap(2swaps) (and synthetic instruments)
5. **fwd_b1_curve_create**: builds a forward function from FRA/IRS/Basis swap(1swap) (and synthetic instruments)
6. **cb_curve_base_disc, cb_curve_flat_disc**: creates a currency curve and calculates fx-implied discount functions.
7. **swap_surface_b2_create**: creates a swap curve surface with all tenors from FRA/IRS/Basis swap(2swap). (A swap_surface_b1_create is available but not discussed).

The wrapper functions not discussed in this document are principally the same as the ones explained but varies in terms of input instruments and other details.

Additionally, cross currency curve building is also supported but only discussed briefly in this document.

2. Creating a discount function

2.1. *Wrapper functions*

A discount function can be created with a single wrapper function call. There are two main versions, a base version and an extended version. The extended version allows for extending the discounting curve based on a second curve by extrapolating the basis spread between the discounting curve and the extrapolation curve.

Base version:

```
disc_func      disc_curve_create(  date          trade_date,
                                   curve option(nullable) short_curve,
                                   curve option(nullable) middle_curve,
                                   curve option(nullable) long_curve,
                                   curve_prio             prio1,
                                   curve_prio             prio2,
                                   logical                 no_overlap_middle,
                                   logical                 merge_middle,
                                   logical                 merge_crv,
                                   integer                 blend_buf_days,
                                   disc_z_model            disc_model,
                                   out swap_curve_disc_ext disc)
```

Arguments and return values explained:

Return value	Comment
disc_func	The disc_func object represents a discount function. This object has many member functions (see the Quantlab function browser).

Arguments	Data type	Comment
trade_date	date	Trade date.
short_curve	curve	The curve for the short segment of the discounting curve. Typically, this is the segment for instruments with time-to-maturity of less than 1 year.
middle_curve	curve	The curve for the middle segment of the discounting curve. This segment of the curve is intended to be used

		for forward starting instruments such as FRA or Futures.
long_curve	curve	The curve for the long segment of the discounting curve.
prio1	curve_prio	Indicates which segment has first priority. Short segment = CP_SHORT, middle segment = CP_MIDDLE and long segment = CP_LONG.
prio2	curve_prio	Indicates which segment has second priority. Short segment = CP_SHORT, middle segment = CP_MIDDLE and long segment = CP_LONG.
no_overlap_middle	logical	Determines how the blending between the short and middle segments of the curve is done in terms of the last included short instrument. If this flag is true (false) then the blending assures that maturity of the last short instrument is before settlement (maturity) of the first middle instrument. Ex. If false then a 3M Deposit is allowed together with a 2X5 FRA.
merge_middle	logical	The middle segment is merged into the curve.
merge_crv	logical	All curve segments are merged into the curve.
blend_buf_days	integer	The minimum number of days between “corner” instruments for the curve segments in the curve blending.
disc_model	disc_z_model	the model object (see below)

out Arguments	Data type	Comment
disc	swap_curve_disc_ext	This object represents the created discounting curve object from which several results and information can be retrieved e.g. the blended curve (see below).

Extended version:

disc_func	disc_curve_create(date	trade_date,
		curve option(nullable)	short_curve,
		curve option(nullable)	middle_curve,
		curve option(nullable)	long_curve,
		curve	extrap_curve,
		string	class_name,
		curve_prio	prio1,
		curve_prio	prio2,
		logical	no_overlap_middle,

logical	merge_middle,
logical	merge_crv,
integer	blend_buf_days,
disc_z_model	disc_model,
disc_z_model	extrap_model,
out swap_curve_disc_ext	disc)

Most arguments are the same as for the base version above and, hence, are not explained here. The additional arguments are:

Arguments	Data type	Comment
extrap_curve	extrap_curve	Base curve used to extend the discounting curve. The synthetic instruments appended to the discounting curve are Deposits or FRA contracts and their rates are calculated by extrapolating the basis spread.
class_name	string	Class name for the instruments to be added (Deposit or FRA).
extrap_model	disc_z_model	the extrapolation model object (see below)

2.2. The disc_z_model object

The disc_z_model object is created from a disc_model_parm object.

```
disc_model_parm    disc_p    = disc_model_parm();
```

The disc_model_parm is an object that represents all settings relevant for a model. It is first created as a default object where all settings are set to default values. The default values are all configurable by the user in the file fiws_user_config.ql. All defaults can be overridden by using an appropriate set function. It is possible to set a specific model with default values and as well as setting all model specific parameters individually.

When a disc_model_parm object is created with the desired properties we can create the disc_z_model object.

```
disc_z_model    disc_model    = init_disc_model(disc_p);
```

The model set functions exist in two versions where one is exposing all model specific arguments and one will use the model specific default settings.

model set functions	Model enum	Comment
set_model(disc_z_model_type) set_model(string)		general set function for a model
set_model_boot(...)	DZ_BOOT	standard bootstrap
set_model_step(...)	DZ_ON_STEP	bootstrap with a piecewise flat O/N rate.
set_model_max_smooth(...)	DZ_MAX_SMOOTH	Adams/Deventer maximum smoothness model.
set_model_max_smooth_spread(...)	DZ_MAX_SMOOTH_SPREAD	Adams/Deventer maximum smoothness model where exact fit is defined as being within a certain spread (per instrument)
set_model_step_boot(...)	DZ_ON_STEP_BOOT	standard bootstrap combined with a piecewise flat O/N rate, up to the step_end date (see below).
set_model_tanggaard(...)	DZ_TANGGAARD	maximum smoothness model with residuals.
set_model_tension_w_exact(...)	DZ_TENSION_W_EXACT	exact fit tension spline model on w. We refer to the reference for the tension spline for explanation of w.
set_model_tension_w_spread(...)	DZ_TENSION_W_SPREAD	tension spline model on w where exact fit is defined as being within a certain spread (per instrument).
set_model_tension_y_exact(...)	DZ_TENSION_Y_EXACT	exact fit tension spline model on spot rate.
set_model_tension_y_spread(...)	DZ_TENSION_Y_SPREAD	tension spline model on spot rate where exact fit is defined as being within a certain spread (per instrument).
set_model_tension_z_exact(...)	DZ_TENSION_Z_EXACT	exact fit tension spline model on instantaneous forward rate.
set_model_tension_z_spread(...)	DZ_TENSION_Z_SPREAD	tension spline model on instantaneous forward rate where exact fit is defined as being within a certain spread (per instrument).

For details on the maximum smoothness model, see Adams-Van Deventer and Forsgren references in the reference section.

For details on the tension spline models, see the Andersen, L. reference in the reference section.

For details on the Tanggaard model, see the Tanggaard reference in the reference section.

Other set functions are:

set functions	Model	Comment
set_cal(calendar)	all	sets the calendar
set_cb_dates(vector(date)	all except DZ_ON_STEP and DZ_ON_STEP_BOOT	sets the central bank meeting dates
set_mc_force_pos(logical)	DZ_BOOT, DZ_ON_STEP_BOOT and only when disc_ip_type = DIPT_HAGAN_WEST	sets a flag in the Hagan West model to control enforcement of positive rates.
set_apply_hyman(logical)	DZ_BOOT, DZ_ON_STEP_BOOT	sets a flag if the hyman filter will be applied.
set_ip(disc_ip_type)	DZ_BOOT, DZ_ON_STEP_BOOT	sets the interpolation scheme. Available models includes: 1. DIPT_HAGAN_WEST: Hagan/West monotone convex spline. 2. DIPT_HERMITE_AKIMA: 3. DIPT_HERMITE_CR: 4. DIPT_HERMITE_FINDIFF: 5. DIPT_HERMITE_FB: 6. DIPT_HERMITE_KRUGER: 7. DIPT_HERMITE_MONO: 8. DIPT_HERMITE_P: 9. DIPT_LINEAR: linear interpolation 10. DIPT_SPLINE_NAT : natural cubic spline interpolation 11. DIPT_SPLINE_FIN : financial cubic spline interpolation 12. DIPT_STEP: step function interpolation.
set_rt(rate_type)	DZ_BOOT,	sets the interpolation rate type. Available

	DZ_ON_STEP_BOOT	<p>choices are:</p> <p>RT_CONT: Continuous compounding.</p> <p>RT_ON: Daily compounding.</p> <p>RT_SIMPLE: Simple compounding.</p> <p>RT_EFFECTIVE: Annual effective compounding.</p> <p>RT_SEMI_ANNUAL: Semi-annual compounding.</p> <p>RT_QUARTERLY: Quarterly compounding.</p> <p>RT_MONTHLY: Monthly compounding.</p> <p>RT_BANKDISC: Bank discount rate.</p> <p>RT_LOGDF: Log of the discount function</p> <p>Note: the Hagan/West model is only defined for RT_CONT.</p>
set_left_der(number)	DZ_BOOT,	sets the left derivative in the interpolation scheme.
set_hermite_right_der(number)	DZ_BOOT, DZ_ON_STEP_BOOT	sets the right derivative in the interpolation scheme for the hermite models.
set_step_end(number)	DZ_ON_STEP_BOOT	sets the step end date.
set_spreads(vector(number))	DZ_MAX_SMOOTH_SPREAD, DZ_TENSION_Y_SPREAD, DZ_TENSION_W_SPREAD, DZ_TENSION_Z_SPREAD	<p>sets the instrument spreads for spread models. If the vector has one element it will be used for all instruments.</p> <p>Otherwise the vector should have the same size as the number of instruments in the blended curve.</p>
set_spread_type(z_mod_spread_type)	DZ_MAX_SMOOTH_SPREAD, DZ_TENSION_Y_SPREAD, DZ_TENSION_W_SPREAD, DZ_TENSION_Z_SPREAD	<p>sets the instrument spread type for spread models. Available choices are:</p> <p>ZMS_BP: spread in basis points</p> <p>ZMS_P: spread in price</p>
set_sigma(vector(number), vector(number))	DZ_TENSION_Y_EXACT, DZ_TENSION_W_EXACT, DZ_TENSION_Z_EXACT, DZ_TENSION_Y_SPREAD, DZ_TENSION_W_SPREAD, DZ_TENSION_Z_SPREAD	sets the sigma. We refer to the reference for the tension spline for explanation of sigma.
set_epsilon(number)	DZ_TENSION_W_EXACT, DZ_TENSION_W_SPREAD	sets the epsilon. We refer to the reference for the tension spline for

		explanation of epsilon.
set_w_factor(number)	DZ_TANGGAARD	sets the w factor. We refer to the reference for the tanggaard model for explanation of w.
set_std_w(std_weights)	DZ_TANGGAARD	sets the standard weights.
set_smooth_m(smoothness_measure)	DZ_MAX_SMOOTH, DZ_MAX_SMOOTH_SPREAD, DZ_TANGGAARD	sets the smoothness measure. We refer to the reference for the maximum smoothness model for explanation of this measure.
set_xpol_type(disc_extrap_type) set_xpol_type(string);	all	sets the extrapolation type. Available choices are: DXP_MODEL: extrapolation according to selected model. DXP_MODEL_FLAT: flat extrapolation of the rate type for the model. DXP_ON_FLAT: extrapolation with flat O/N rate. The corresponding string arguments are: "MODEL", "MODEL_FLAT", "ON_FLAT".
set_xpol_cut(number option(nullable));	all	sets the extrapolation cutoff period. The period is defined as the number calendar days between the “cutoff” date and trade date divided by 365. Two values have a specific meaning: -1: no cutoff 0 or null: cutoff at curve max date

2.3. *The swap_curve_disc_ext object*

The swap_curve_disc_ext object (inherited from swap_curve_ext) represents the created discounting curve and contains the input data and several intermediate results as well as provides access to and calculates the general disc_func object.

The member functions are:

Function	Arguments	Comment
curve blended_curve()	none	Returns the blended curve. The

		blended curve is the actual instruments used in the curve building procedure. Depending on the blending parameters several instruments is likely to be excluded and if an extrapolation curve is used several instruments may be added to the curve.
string name()	none	Returns the name of the curve.
disc_func disc_df()	none	Returns the discount function.
disc_func disc_df(...)	disc_z_model disc_model, interpolator option(nullable) disc_ip, rate_type option(nullable) disc_ip_rt, vector(dates) option(nullable) cb_dates, number option(nullable) step_end	Returns the discount function but with the possibility to override the model arguments. The arguments are described above.
date curve_end()	none	Returns the end date of the curve.

2.4. Code examples

The examples below uses a slightly different `disc_curve_create (...)` function than was explained above. The only difference is that here the creation of the curve object and calculation of the `disc_func` is done in two steps.

2.4.1. example1 - basic

Assume we would like to use the default values for the Hagan-West model.

```
//--Step 1-- create the discount model
```

```
// create a disc_model_parm object with default values
```

```
disc_model_parm disc_p = disc_model_parm();
```

/*if we would like to override any default value in the disc_model_parm object this is where all the set functions should be applied. Here we override the default model to set the model to Hagan-West using default values for this model. Since the Hagan-West model is implemented as an interpolation scheme in the bootstrap model we use the function set_model_boot(...). There is one model function, update_spreads(...), which may need to be used after the curve object is created, see below. */

```
disc_p.set_model_boot(DIPT_HAGAN_WEST);
```

// create the model object

```
disc_z_model      disc_model      = init_disc_model(disc_p);
```

//--Step 2-- create the discount curve with blending parameters

```
number           blend_buf_days      = 5;
logical          no_overlap_middle    = false;
logical          merge_middle        = false;
logical          merge_crv           = false;
curve_prio       prio1                = CP_LONG;
curve_prio       prio2                = CP_MIDDLE;
```

```
swap_curve_disc_ext disc      = disc_curve_create( today, null<curve>,null<curve>, long_crv, prio1,
prio2, no_overlap_middle, merge_middle, merge_crv, blend_buf_days);
```

/* if a model with spreads is used (eg. DZ_MAX_SMOOTH_SPREAD) and we wish to set a spread per instrument this is where we would use the update_spreads() function (because it is only after the swap_curve_disc_ext object is created that we know the blended curve)*/

// create the disc_func

```
disc_func      df_ois      = disc.disc_df (disc_model);
```

2.4.2. example2

//--Step 1-- create the discount model

// create a disc_model_parm object with default values

```
disc_model_parm      disc_p      = disc_model_parm();
```

// set model parameters one by one

```
disc_p.set_model("BOOT");           //bootstrap
disc_p.set_rt("CONT");              //continuously compounded spot rates
```

```

disc_p.set_ip("SPLINE_FIN");           //financial spline
disc_p.set_left_der(0);                //flat extrapolation to the left
disc_p.set_xpol_type("ON_FLAT");       //flat extrapolation of O/N rate to the right
disc_p.set_xpol_cut(-1);               //-1->no cutoff
disc_p.set_apply_hyman(true);          // apply hyman filter

// create the model object
disc_z_model      disc_model           = init_disc_model(disc_p);

//create the extrapolation model from defaults
disc_model_parm   extrap_p             = disc_model_parm();
disc_z_model      extrap_model         = init_disc_model(extrap_p);

//--Step 2-- create the extrapolated discount curve with blending parameters

number           blend_buf_days       = 5;
logical          no_overlap_middle    = false;
logical          merge_middle         = false;
logical          merge_crv            = false;
curve_prio       prio1                =CP_LONG;
curve_prio       prio2                =CP_MIDDLE;

swap_curve_disc_ext  disc_base = disc_curve_create( today, null<curve>,null<curve>, long_crv,
                                                    prio1,prio2, no_overlap_middle,
                                                    merge_middle, merge_crv, blend_buf_days);

swap_curve_disc_ext  disc      = disc_curve_create( today, disc_base, extrap_curve,
                                                    depo_class_name, extrap_model );

// create the disc_func
disc_func          df_ois             = disc.disc_df (disc_model);

```

3. Creating a forward function

3.1. *FRA/IRS wrapper function*

A forward function based on a short curve, FRA curve and an IRS curve can be created with a single wrapper function call.

```
fwd_func      fwd_curve_create(  date          trade_date,
                                disc_func            df_disc,
                                disc_func            df_synt_base,
                                date                 synt_end,
                                curve option(nullable) short_curve,
                                curve option(nullable) middle_curve,
                                curve option(nullable) long_curve,
                                curve_prio           prio1,
                                curve_prio           prio2,
                                string option(nullable) depo_class_name,
                                string option(nullable) fra_class_name,
                                synt_short_style      synt,
                                fwd_z_model           fwd_model,
                                logical                merge_middle,
                                logical                merge_crv,
                                integer                blend_buf_days,
                                vector(number) option(nullable) edfut_cvx_adj,
                                out swap_curve_fwd_ext fwd_crv,
                                number option(nullable) long_first_fix)
```

Arguments and return values explained:

Return value	Comment
fwd_func	The fwd_func object represents a forward rate function (see the Quantlab function browser for available member functions).

Arguments	Data type	Comment
trade_date	date	Trade date.
df_disc	disc_func	The discount function e.g. OIS.
df_synt_base	disc_func	Discount function used as base discount function when creating the synthetic instruments.

synt_end	date	The maturity of the longest instrument for the curve used to calculate df_synt_base. This date will be used to specify the maximum length of the basis curve used for creating synthetic instruments (in order to avoid extrapolation).
short_curve	curve	The curve for the short segment of the forward curve, typically just the relevant tenor deposit rate. If this curve contains a “full” deposit curve, all non-relevant tenor rates will be removed.
middle_curve	curve	The curve for the FRA/Futures segment of the forward curve. All FRAs must have the same tenor and the same tenor as the IRS floating leg.
long_curve	curve	The curve for the IRS segment of the forward curve. All swaps must have the same tenor of the floating leg and this tenor must match all the FRA/Futures contract tenors.
prio1	curve_prio	Indicates which segment has first priority (SHORT, MIDDLE or LONG)
prio2	curve_prio	Indicates which segment has second priority (SHORT, MIDDLE or LONG)
depo_class_name	string	Class name for relevant Deposits (for synthetic instruments).
fra_class_name	string	Class name for relevant FRA contracts (for synthetic instruments). Used only when synt = SY_FRA.
synt	synt_short_style	<p>Possible choices are SY_FRA, SY_DEPO and SY_NONE. There are two built-in ways to create synthetic instruments for the short end of the curve up to the first FRA/ Futures contract. (As explained above, if the user wish to use a different principle it is possible to access the core functions directly.)</p> <p>SY_FRA: In this case, both synthetic FRAs and a single synthetic Deposit are created. The FRAs are all of the relevant tenor and starting 1w, 2w and every month until the first FRA starts. The added Deposit has a maturity equal to the tenor.</p> <p>SY_DEPO: Only a synthetic Deposit is created with the relevant tenor i.e. with maturity equal to the tenor.</p> <p>SY_NONE: No synthetic instruments are created.</p> <p>For details on the creation of synthetic short instruments,</p>

		see below.
fwd_model	fwd_z_model	The forward model object (see below)
merge_middle	logical	The FRA contracts are merged into the curve.
merge_crv	logical	All curve segments are merged into the curve.
blend_buf_days	integer	The minimum number of days between curve segments in the curve blending.
vector(number)	edfut_cvx_adj	A vector of spreads that adjusts the FRA/Futures segment of the curve. This parameter can, for example, be used to provide a convexity adjustment for Futures.
number	long_first_fix	The first floating rate fixing for the IRS curve. If null, no fixing is used.

out Arguments	Data type	Comment
fwd_crv	swap_curve_fwd_ext	This object represents the created forward curve (without basis swaps) object from which several results and information can be retrieved e.g. the blended curve.

3.2 The fwd_z_model object

The fwd_z_model object is created from a fwd_model_parm object.

```
fwd_model_parm    fwd_p    = fwd_model_parm();
```

The disc_model_parm is an object that represents all settings relevant for a model. It is first created as a default object where all settings are set to default values. The default values are all configurable by the user in the file fiws_user_config.ql. All defaults can be overridden by using an appropriate set function. It is possible to set a specific model with default values and as well as setting all model specific parameters individually.

When a fwd_model_parm object is created with the desired properties we can create the fwd_z_model object.

```
fwd_z_model    fwd_model    = init_fwd_model(fwd_p);
```

The model set functions exist in two versions where one is exposing all model specific arguments and one will use the model specific default settings.

set function	Model enum	Comment
set_model(fwd_z_model_type) set_model(string)		general set function for a model
set_model_linear(...)	FZM_LINEAR	linear interpolation
set_model_spline_nat(...)	FZM_SPLINE_NAT	natural cubic spline interpolation
set_model_spline_fin(...)	FZM_SPLINE_FIN	financial cubic spline interpolation
set_model_step(...)	FZM_STEP	step interpolation
set_model_step_linear(...)	FZM_STEP_LINEAR	step interpolation with linear interpolation between knot points.
set_model_hermite_akima(...)	FZM_HERMITE_AKIMA	
set_model_hermite_cr(...)	FZM_HERMITE_CR	
set_model_hermite_findiff(...)	FZM_HERMITE_FINDIFF	
set_model_hermite_fb(...)	FZM_HERMITE_FB	
set_model_hermite_kruger(...)	FZM_HERMITE_KRUGER	
set_model_hermite_mono(...)	FZM_HERMITE_MONO	
set_model_hermite_p(...)	FZM_HERMITE_P	
set_model_tension_exact(...)	FZM_TENSION_EXACT	exact fit tension spline
set_model_tension_spread(...)	FZM_TENSION_SPREAD	tension spline model where exact fit is defined as being within a certain spread (per instrument).

For details on the tension spline models, see the Andersen, L. reference in the reference section.

Other set functions are:

set functions	Model	Comment
set_apply_hyman(logical)		sets a flag if the hyman filter will be applied
set_left_der(number)		sets the left derivative in the interpolation scheme
set_hermite_right_der(number)		sets the right derivative in the interpolation scheme for the hermite models
set_spreads(vector(number))		sets the instrument spreads for spread models. If the vector has one

		element it will be used for all instruments. Otherwise the vector should have the same size as the number of instruments in the blended curve.
set_spread_type(z_mod_spread_type)	FZM_TENSION_SPREAD	sets the instrument spread type for spread models. Available choices are: ZMS_BP: spread in basis points ZMS_P: spread in price
set_sigma(vector(number), vector(number))	FZM_TENSION_EXACT, FZM_TENSION_SPREAD	sets the sigma. We refer to the reference for the tension spline for explanation of sigma.
set_xpol_type(disc_extrap_type) set_xpol_type(string);	all	sets the extrapolation type. Available choices are: FXP_MODEL: extrapolation according to selected model. FXP_TENOR_FLAT: flat extrapolation of the rate type for the model. The corresponding string arguments are: "MODEL", "TENOR_FLAT".
set_xpol_cut(number option(nullable));	all	sets the extrapolation cutoff period . Two values have a specific meaning: -1: no cutoff 0 or null: cutoff at curve max date

3.3 The swap_curve_fwd_ext object

The swap_curve_fwd_ext object (inherited from swap_curve_ext) represents a forward curve (created without basis swaps) and contains the input data and several intermediate results as well as provides access to and calculates the general fwd_func object.

The member functions are:

Function	Arguments	Comment
curve blended_curve()	none	Returns the blended curve. The blended curve is the actual instruments used in the curve building procedure. Depending on the blending parameters several instruments is likely to be excluded and any added synthetic instruments

			will also be present.
string	name()	none	Returns the name of the curve.
number	fix_freq()	none	Returns the fixed leg frequency of the IRS curve.
number	flt_freq()	none	Returns the forward rate tenor.
disc_func	disc_df()	none	Returns the discount function.
cashflows_fwd	cf_fwd()	none	Returns the cashflows_fwd object. This object is the most basic representation of the instruments used to create the forward curve and is the input to the bootstrap_fwd(...) function (see the Quantlab function browser)
fwd_func	fwd()	none	Returns the forward function.
fwd_func	fwd(...)	interpolator fwd_ip	Returns the forward function but with the possibility to override the interpolation argument. The argument is described above.
fwd_func	fwd_comb(...)	fwd_func date fwd_pre, cut_off	Returns a forward function. This version allows to use a previously calculated fwd_func for the beginning of the curve up to a date specified by the cut_off argument.
fwd_func	fwd_comb(...)	fwd_func date fwd_pre, cut_off, interpolator fwd_ip	As the previous function but with the possibility to override the interpolation argument. The argument is described above.
curve	imp_swap_curve(...)	fwd_func fflt_idx	Returns the implied IRS curve from a given fwd_func.
fwd_func	fwd_risk(...)	interpolator fwd_ip, number r_shift, logical const_basis, out disc_func df_disc	Returns a shifted fwd_func calculated from a parallel shift of all instruments in the underlying curve. The parallel shift is specified by r_shift. If const_basis is true then the discount function will also be shifted in a way that preserves the basis spread.
date	curve_end()	none	Returns the end date of the curve.

3.4. *Creating synthetic short instruments (before the first FRA)*

The synthetic instruments, if included, are created from a base discount function and its basis spread vs. the FRA contracts.

Step 1. Create a basis spread function using all the FRAs and the provided base discount function. The base discount function could be an OIS discount function, a discount function calculated from a different tenor forward function or any other discount function.

Step 2. Calculate the implied rates for the synthetic instruments from the base discount function and add the interpolated basis spread from step 1.

The created instruments now have a basis consistent with the FRA contracts and they will provide added information for the short end of the curve. Their curvature will depend on the base discount function and the interpolated basis spread.

3.5. *Forward index rate interpolation*

The interpolation models used when creating a forward function operates directly on the tenor rate we wish to calculate. This ensures that we get forward rates without the oscillating behavior that sometimes can be seen in the standard bootstrapping models for a discount function (when interpolating different types of spot rates). See below for some typical forward rate examples.

3.6. *The forward function <fwd_func>*

The forward function contains all the forward rates for any date for a certain tenor. The member function `fwd()` which takes an Act365 period (from trade date) as argument returns the rate. Of course, if the period corresponds to a point used in the original curve we get this point exactly otherwise, it is interpolated with the model used in the bootstrap-function.

For example, assume the forward function is calculated for a 6M tenor and we wish to calculate the forward rate in one month i.e. we would like to know the 1x7 forward rate. Assume today is 9-Sep-2011 and the 1M date is 13-Oct-2011.

```
fwd_func f_fwd = fwd_curve_create(...);  
fwd_rate_1x7 = f_fwd.fwd((#2011-10-13 - #2011-09-09)/365);
```

3.7. *FRA/IRS/BasisSwap(2Swap) wrapper function*

This version uses FRA/Futures, IRS and Basis swaps (2 Swaps) and the short end of the curve can be created synthetically.

```
fwd_func      fwd_b2_curve_create( date           trade_date,
                                   disc_func       df_disc,
                                   disc_func       df_synt_base,
                                   date            synt_end,
                                   curve option(nullable) short_out_curve,
                                   curve option(nullable) middle_out_curve,
                                   curve option(nullable) long_in_curve,
                                   curve option(nullable) basis_t1t2_curve,
                                   curve_prio      prio1,
                                   curve_prio      prio2,
                                   string option(nullable) depo_class_name,
                                   string option(nullable) fra_out_class_name,
                                   synt_short_style synt,
                                   fwd_z_model     fwd_model,
                                   logical          merge_middle,
                                   logical          merge_crv,
                                   integer          blend_buf_days,
                                   vector(number) option(nullable) edfut_cvx_adj,
                                   logical          x_pol_basis,
                                   out swap_curve_b2_ext fwd_crv,
                                   number option(nullable) first_fix_out)
```

Since most arguments are the same as for the previous wrapper, only the unique arguments will be explained. Note, “in” and “out” stands for the different tenors involved and the “out curve” is the tenor we are calculating. If there are more or less tenors available in the basis curve compared to the underlying swap curve (i.e. long_in_curve) the missing tenor instruments will be created synthetically based on linear interpolation. If there are fewer basis swaps than underlying swaps the “missing” basis swaps will be created. If there are more basis swaps than underlying swaps the “missing” underlying swap will be created.

Arguments	Data type	Comment
basis_t1t2_curve	curve	The curve for the Basis swap of the forward curve. All Basis

		swaps must have one tenor of the floating leg that matches the IRS curve. The tenor of the other floating leg must match the FRA contracts. The spreads must be a 2 swaps quotation.
x_pol_basis	logical	If true the basis curve will be extrapolated, with a constant basis spread, up to the length of the swap curve.

out Arguments	Data type	Comment
fwd_crv	swap_curve_b2_ext	This object represents the created forward curve object (created via a basis swap quoted as 2 swaps) from which several results and information can be retrieved e.g. the blended curve.

Note that the FRA contracts correspond to the tenor we wish to calculate. The IRS combined with the Basis swap will convert the IRS to the FRA tenor. For example, the FRA curve could be a [EUR 3M FRA]-curve, the IRS curve could be a [EUR IRS 6M EURIBOR]-curve and the Basis swap could be a [EUR 3M/6M EURIBOR]-curve. In this case “out” = 3M and “in” = 6M. The [EUR IRS 6M EURIBOR]-curve will be converted to a [EUR IRS 3M EURIBOR]-curve and we proceed with calculations for the 3M tenor. We could also, with the same principle, calculate the 6M tenor. A difference in daycount conventions between the Basis swap and the IRS is handled by converting the basis spread.

3.8. *The swap_curve_b2_ext object*

The swap_curve_b2_ext object (inherited from swap_curve_ext) represents a forward curve (created from basis swaps quoted as 2 swaps) and contains the input data and several intermediate results as well as provides access to and calculates the general fwd_func object.

The member functions are:

Function	Arguments	Comment
curve blended_curve()	none	Returns the blended curve. The blended curve is the actual instruments used in the curve building procedure. Depending on the blending parameters several instruments is likely to be excluded and any added synthetic instruments will also be present.
string name()	none	Returns the name of the curve.

number	fix_freq()	none	Returns the fixed leg frequency of the IRS curve.
number	flt_freq()	none	Returns the forward rate tenor.
disc_func	disc_df()	none	Returns the discount function.
cashflows_fwd	cf_fwd()	none	Returns the cashflows_fwd object. This object is the most basic representation of the instruments used to create the forward curve and is the input to the bootstrap_fwd(...) function (see the Quantlab function browser)
fwd_func	fwd()	none	Returns the forward function.
fwd_func	fwd(...)	interpolator fwd_ip	Returns the forward function but with the possibility to override the interpolation argument. The argument is described above.
fwd_func	fwd_comb(...)	fwd_func date fwd_pre, cut_off	Returns a forward function. This version allows to use a previously calculated fwd_func for the beginning of the curve up to a date specified by cut_off.
fwd_func	fwd_comb(...)	fwd_func date interpolator fwd_pre, cut_off, fwd_ip	As the previous function but with the possibility to override the interpolation argument. The argument is described above.
curve	imp_swap_curve(...)	fwd_func f_flt_idx	Returns the implied IRS curve from a given fwd_func.
fwd_func	fwd_risk(...)	interpolator number logical out disc_func fwd_ip, r_shift, const_basis, df_disc	Returns a shifted fwd_func calculated from a parallel shift of all instruments in the underlying curve. The parallel shift is specified by r_shift. If const_basis is true then the discount function will also be shifted in a way that preserves the basis spread.
date	curve_end()	none	Returns the end date of the curve.

3.9. FRA/IRS/BasisSwap(1Swap) wrapper function

This version uses FRA/Futures, IRS and Basis swaps (1 Swap) and the short end of the curve can be created synthetically.

```

fwd_func      fwd_b1_curve_create( date          trade_date,
                                   disc_func      df_disc,
                                   disc_func      df_synt_base,
                                   date            synt_end,
                                   curve option(nullable) short_in_curve,
                                   curve option(nullable) middle_in_curve,
                                   number          in_freq,
                                   curve option(nullable) long_in_curve,
                                   curve option(nullable) short_out_curve,
                                   curve option(nullable) middle_out_curve,
                                   curve option(nullable) basis_t1t2_curve,
                                   curve_prio      prio1,
                                   curve_prio      prio2,
                                   string option(nullable) depo_class_name,
                                   string option(nullable) fra_in_class_name,
                                   string option(nullable) fra_out_class_name,
                                   synt_short_style synt,
                                   fwd_z_model     fwd_model,
                                   logical          merge_middle,
                                   logical          merge_crv,
                                   integer          blend_buf_days,
                                   vector(number) option(nullable) edfut_cvx_adj_in,
                                   vector(number) option(nullable) edfut_cvx_adj_out,
                                   out swap_curve_b1_ext fwd_crv,
                                   number option(nullable) first_fix_base,
                                   number option(nullable) first_fix_flat)

```

This function is similar to the previous case with the difference that we have a complete FRA/IRS curve for the tenor in the basis swap we take as input. If there are less tenors available in the basis curve compared to the underlying swap curve (i.e. long_out_curve) the missing tenor basis swaps will be created synthetically based on linear interpolation. There is one more wrapper available (not shown here) that instead of the “in curves” takes a forward function (fwd_func) as input.

out Arguments	Data type	Comment
fwd_crv	swap_curve_b1_ext	This object represents the created forward curve object (created via a basis swap quoted as 1 swap) from

		which several results and information can be retrieved e.g. the blended curve.
--	--	--

The “out curves” represent the tenor we wish to calculate. The “in curves” will create a forward function according to section 3.1. The “in” forward function combined with the Basis swap will create instruments with the same tenor as the “out curves”. For example, the FRA in curve could be a [EUR 6M FRA]-curve, the IRS in curve could be a [EUR IRS 6M EURIBOR]-curve, the FRA out curve could be [EUR 3M FRA]-curve and the Basis swap could be a [EUR 3M/6M EURIBOR]-curve. In this case, we calculate the 3M tenor. We could also, with the same principle, calculate the 6M tenor.

3.10. The swap_curve_b1_ext object

The swap_curve_b1_ext object (inherited from swap_curve_ext) represents a forward curve (created from basis swaps quoted as 1 swap) and contains the input data and several intermediate results as well as provides access to and calculates the general fwd_func object.

The member functions are:

Function	Arguments	Comment
curve blended_curve()	none	Returns the blended curve. The blended curve is the actual instruments used in the curve building procedure. Depending on the blending parameters several instruments is likely to be excluded and any added synthetic instruments will also be present.
string name()	none	Returns the name of the curve.
number fix_freq()	none	Returns the fixed leg frequency of the IRS curve.
number flt_freq()	none	Returns the forward rate tenor.
disc_func disc_df()	none	Returns the discount function.
cashflows_fwd cf_fwd()	none	Returns the cashflows_fwd object. This object is the most basic representation of the instruments used to create the forward curve and is the input to the bootstrap_fwd(...) function (see the Quantlab function browser)
fwd_func fwd()	none	Returns the forward function.
fwd_func fwd(...)	interpolator fwd_ip	Returns the forward function but

				with the possibility to override the interpolation argument. The argument is described above.
fwd_func	fwd_comb(...)	fwd_func date	fwd_pre, cut_off	Returns a forward function. This version allows to use a previously calculated fwd_func for the beginning of the curve up to a date specified by cut_off.
fwd_func	fwd_comb(...)	fwd_func date interpolator	fwd_pre, cut_off, fwd_ip	As the previous function but with the possibility to override the interpolation argument. The argument is described above.
curve	imp_swap_curve(...)	fwd_func	fflt_idx	Returns the implied IRS curve from a given fwd_func.
fwd_func	fwd_risk(...)	interpolator number logical out disc_func	fwd_ip, r_shift, const_basis, df_disc	Returns a shifted fwd_func calculated from a parallel shift of all instruments in the underlying curve. The parallel shift is specified by r_shift. If const_basis is true then the discount function will also be shifted in a way that preserves the basis spread.
date	curve_end()	none		Returns the end date of the curve.

3.11. Code examples

The examples below uses a slightly different fwd_curve_create (...) function than was explained above. The only difference is that here the creation of the curve object and calculation of the fwd_func is done in two steps.

3.11.1. example1 - basic

Assume we would like to use default values for all settings including the model itself.

```
//at this point we have created a disc_func
//--Step 1—create the forwarding model

//first create a fwd_model_parm object with default values
fwd_model_parm    fwd_p    = fwd_model_parm();
```

/*if we would like to override any default value in the fwd_model_parm object this is where all the set functions should be applied. There is one model function, update_spreads(...), which may need to be used after the curve object is created, see below. */

// create the model object

```
fwd_z_model      fwd_model      = init_fwd_model(fwd_p);
```

//--Step 2-- create the forward curve with blending and synt parameters

```
number           blend_buf_days      = 5;
logical          merge_middle        = false;
logical          merge_crv           = false;
curve_prio       prio1               = CP_LONG;
curve_prio       prio2               = CP_MIDDLE;
synt_short_style synt               = SY_FRA;
date             ois_end              = disc.curve_end();

swap_curve_fwd_ext fwd_crv           = fwd_curve_create(today,df_ois,df_ois,ois_end,short_curve,
                                                         middle_crv,long_crv,prio1,prio2,
                                                         depo_class_name,fra_class_name,synt,
                                                         merge_middle,merge_crv,blend_buf_days,
                                                         null<vector(number)>,null<number>);
```

// if a model with spreads is used (eg. FZM_TENSION_SPREAD) and we wish to set a spread per instrument this is where we would use the update_spreads() function (because it is only after the swap_curve_fwd_ext object is created that we know the blended curve).*/

//and create the fwd_func

```
fwd_func         f_fwd           = fwd_crv.fwd(fwd_model);
```

3.11.2. example 2

//at this point we have created a disc_func

//--Step 1—create the forwarding model

//first create a fwd_model_parm object with default values

```
fwd_model_parm   fwd_p           = fwd_model_parm();
```

// set model parameters one by one

```
fwd_p.set_model("TENSION_SPREAD");           //tension spline model with spreads
```

```

fwd_p.set_sigma([1,10,30], [3,3,3]);           //tension sigma x and y vector
fwd_p.set_xpol_type("TENOR_FLAT");             //flat extrapolation of tenor rate to the right
fwd_p.set_xpol_cut(-1);                        //-1->no cutoff
fwd_p.set_spread_type(ZMS_BP);                 // spread type is set to basis points
fwd_p.set_spreads([0.5]);                      // spread is 0.5 basis points for all instruments

// create the model object
fwd_z_model      fwd_model      = init_fwd_model(fwd_p);

//--Step 2-- create the forward curve with blending and synt parameters

number           blend_buf_days      = 5;
logical          merge_middle        = false;
logical          merge_crv           = false;
curve_prio       prio1               =CP_LONG;
curve_prio       prio2               =CP_MIDDLE;
synt_short_style synt                = SY_FRA;
date             ois_end              = disc.curve_end();

swap_curve_fwd_ext fwd_crv           = fwd_curve_create(today,df_ois,df_ois, ois_end, short_curve,
                                                         middle_crv, long_crv, prio1, prio2,
                                                         depo_class_name, fra_class_name, synt,
                                                         merge_middle, merge_crv, blend_buf_days,
                                                         null< vector(number)>, null<number>);

//and create the fwd_func
fwd_func         f_fwd               = fwd_crv.fwd(fwd_model);

```

4. Creating an fx implied discount function

4.1. *Wrapper functions for currency basis swaps*

An fx-implied discount function can be created with a single wrapper function call. There are two versions, depending on which currency discount function should be calculated.

Discount function for the flat leg currency in the currency basis swap:

```
disc_func      cb_curve_flat_disc (    date                trade_date,
                                      instrument option(nullable) fx_spot,
                                      curve option(nullable)      fxswap_crv,
                                      instr_class_name option(nullable) depo_class_fxbase,
                                      instr_class_name option(nullable) depo_class_fxprice,
                                      curve                        basis_crv,
                                      number option(nullable)      basis_first_fix_flat,
                                      number option(nullable)      basis_first_fix_sprd,
                                      logical                        merge_crv,
                                      logical                        prio_fxswap,
                                      integer                       blend_buf_days,
                                      disc_func option(nullable)    df_disc_flat,
                                      disc_func                     df_disc_sprd,
                                      fwd_func                      f_fwd_flat,
                                      fwd_func                      f_fwd_sprd,
                                      string option(nullable)       name,
                                      interpolator option(nullable) disc_ip,
                                      rate_type option(nullable)    disc_ip_rt,
                                      out swap_curve_bc_ext         currb)
```

Arguments and return values explained:

Return value	Comment
disc_func	The disc_func object represents a discount function. This object has many member functions (see the Quantlab function browser).

Arguments	Data type	Comment
trade_date	date	Trade date.
fx_spot	instrument	The fx spot currency. Required only if an fx-swap curve is used in the short end.

fxswap_crv	curve	The fx-swap curve. This curve is not required.
depo_class_fxbase	instr_class_name	Class name for Deposits in the base currency. Note that base currency is related with the fx-swap curve (it does not refer to the currency basis swap)
depo_class_fxprice	instr_class_name	Class name for Deposits in the price currency.
basis_crv	curve	The currency basis swap curve.
basis_first_fix_flat	number	The first floating rate fixing for the flat leg in the currency basis curve. If null, no fixing is used.
basis_first_fix_sprd	number	The first floating rate fixing for the base leg in the currency basis curve. If null, no fixing is used.
merge_crv	logical	All curve segments are merged into the curve.
prio_fxswap	logical	Indicates if the fx-swap instruments have first priority when blending the curve.
blend_buf_days	integer	The minimum number of days between “corner” instruments for the curve segments in the curve blending.
df_disc_flat	disc_func	The discount function for the flat leg in the currency basis curve. In the case where the flat leg discount function is the output this argument is optional. If it is not null then the output will be a spread-adjusted discount function. This may be useful if one would like to preserve the curvature of an input discount function.
df_disc_sprd	disc_func	The discount function for the base leg in the currency basis curve.
f_fwd_flat	fwd_func	The forward rates of appropriate tenor for the flat leg in the currency basis curve.
f_fwd_sprd	fwd_func	The forward rates of appropriate tenor for the base leg in the currency basis curve.
name	string	An optional name of the blended currency curve.
disc_ip	interpolator	Interpolation model object. Available models includes: 1. ip_hagan_west(logical force_pos): Hagan/West monotone convex spline, if force_pos = true, the forward rates are enforced to be positive. 2. ip_hermite_4th_order() : 3. ip_hermite_akima(): 4. ip_hermite_catmull_rom() : 5. ip_hermite_finite_diff(): 6. ip_hermite_fritsch_butland() : 7. ip_hermite_kruger():

instrument option(nullable)	fx_spot,
curve option(nullable)	fxswap_crv,
instr_class_name option(nullable)	depo_class_fxbase,
instr_class_name option(nullable)	depo_class_fxprice,
curve	basis_crv,
number option(nullable)	basis_first_fix_flat,
number option(nullable)	basis_first_fix_sprd,
logical	merge_crv,
logical	prio_fxswap,
integer	blend_buf_days,
disc_func	df_disc_flat,
disc_func option(nullable)	df_disc_sprd,
fwd_func	f_fwd_flat,
fwd_func	f_fwd_sprd,
string option(nullable)	name,
interpolator option(nullable)	disc_ip,
rate_type option(nullable)	disc_ip_rt,
out swap_curve_bc_ext	currb)

The only difference between `cb_curve_sprd_disc(...)` and `cb_curve_flat_disc(...)` is that the `df_disc_sprd` is optional while `df_disc_flat` is required.

4.2. *The swap_curve_bc_ext object*

The `swap_curve_bc_ext` object represents a currency swap curve created from a currency basis swap and contains the input data and several intermediate results as well as provides access to and calculates the general `disc_func` object.

The member functions are:

Function	Arguments	Comment
string <code>name()</code>	none	Returns the name of the curve.
curve <code>basis_curve()</code>	none	Returns the currency basis curve.
number <code>flt_freq_sprd()</code>	none	Returns the forward rate tenor for the spread leg in the currency basis swap.
number <code>flt_freq_flat()</code>	none	Returns the forward rate tenor for the flat leg in the currency basis

		swap.
day_count_method flt_dc_sprd()	none	Returns the daycount method for the spread leg in the currency basis swap.
day_count_method flt_dc_flat()	none	Returns the daycount method for the flat leg in the currency basis swap.
string ccy_sprd()	none	Returns the currency for the spread leg in the currency basis swap.
string ccy_flat()	none	Returns the currency for the flat leg in the currency basis swap.
fwd_func f_fwd_sprd ()	none	Returns the fwd_func for the spread leg in the currency basis swap.
fwd_func f_fwd_flat()	none	Returns the fwd_func for the flat leg in the currency basis swap.
disc_func disc_df_sprd ()		Returns the disc_func for the spread leg in the currency basis swap. Note that this is the input disc_func and not the fx-implied disc_func.
disc_func disc_df_flat ()	none	Returns the disc_func for the flat leg in the currency basis swap. Note that this is the input disc_func and not the fx-implied disc_func.
disc_func adj_disc_df_sprd	interpolator disc_ip, rate_type disc_ip_rt	Returns the fx-implied disc_func for the spread leg in the currency basis swap.
disc_func adj_disc_df_flat	interpolator disc_ip, rate_type disc_ip_rt	Returns the fx-implied disc_func for the flat leg in the currency basis swap.

4.3. *Wrapper functions for currency fix-float swaps*

An fx-implied discount function can be created with a single wrapper function call. There are two versions, depending on which currency discount function should be calculated.

Discount function for the fixed leg currency in the currency fix-float swap:

disc_func cff_curve_fix_disc (date trade_date,

instrument option(nullable)	fx_spot,
curve option(nullable)	fxswap_crv,
instr_class_name option(nullable)	depo_class_fxbase,
instr_class_name option(nullable)	depo_class_fxprice,
curve	fixfloat_crv,
number option(nullable)	flt_first_fix,
logical	merge_crv,
logical	prio_fxswap,
integer	blend_buf_days,
disc_func option(nullable)	fix_df_disc,
disc_func	flt_df_disc,
fwd_func	flt_fwd,
string option(nullable)	name,
interpolator option(nullable)	boot_ip,
rate_type option(nullable)	boot_ip_rt,
out swap_curve_cff_ext	currff)

Arguments and return values explained:

Return value	Comment
disc_func	The disc_func object represents a discount function. This object has many member functions (see the Quantlab function browser).

Arguments	Data type	Comment
trade_date	date	Trade date.
fx_spot	instrument	The fx spot currency. Required only if an fx-swap curve is used in the short end.
fxswap_crv	curve	The fx-swap curve. This curve is not required.
depo_class_fxbase	instr_class_name	Class name for Deposits in the base currency. Note that base currency is related with the fx-swap curve.
depo_class_fxprice	instr_class_name	Class name for Deposits in the price currency.
fixflt_crv	curve	The currency fix-float swap curve.
flt_first_fix	number	The first floating rate fixing for the float leg in the currency swap curve. If null, no fixing is used.
merge_crv	logical	All curve segments are merged into the curve.
prio_fxswap	logical	Indicates if the fx-swap instruments have first priority when blending the curve.
blend_buf_days	integer	The minimum number of days between “corner” instruments for the curve segments in the curve

		blending.
fix_df_disc	disc_func	The discount function for the fixed leg in the currency swap curve. In the case where the fixed leg discount function is the output this argument is optional. If it is not null then the output will be a spread-adjusted discount function. This may be useful if one would like to preserve the curvature of an input discount function.
flt_df_disc	disc_func	The discount function for the float leg in the currency swap curve.
flt_fwd	fwd_func	The forward rates of appropriate tenor for the float leg in the currency swap curve.
name	string	An optional name of the blended currency curve.
boot_ip	interpolator	<p>Interpolation model object. Available models includes:</p> <ol style="list-style-type: none"> 1. ip_hagan_west(logical force_pos): Hagan/West monotone convex spline, if force_pos = true, the forward rates are enforced to be positive. 2. ip_hermite_4th_order() : 3. ip_hermite_akima(): 4. ip_hermite_catmull_rom() : 5. ip_hermite_finite_diff (): 6. ip_hermite_fritsch_butland() : 7. ip_hermite_kruger(): 8. ip_hermite_monotone(): 9. ip_hermite_parabolic(): 10. ip_linear(): linear interpolation 11. ip_spline(): cubic spline interpolation 12. ip_step(): step function interpolation. <p>The interpolator object is created from the above constructors. Each constructor has several more arguments (not shown) related to extrapolation and Hyman filters. It is possible to define extrapolation properties both in the front and end of the curve. It is also possible to apply different types of Hyman filters(eg. monotonicity) to all interpolators except linear and the Hagan/West model. All interpolation models are local except spline. A detailed discussion on these models is beyond the scope of this document.</p>
boot_ip_rt	rate_type	Interpolation rate types. Available choices are: RT_CONT: Continuous compounding.

		RT_ON: Daily compounding. RT_SIMPLE: Simple compounding. RT_EFFECTIVE: Annual effective compounding. RT_SEMI_ANNUAL: Semi-annual compounding. RT_QUARTERLY: Quarterly compounding. RT_MONTHLY: Monthly compounding. RT_BANKDISC: Bank discount rate. RT_LOGDF: Log of the discount function Note: the Hagan/West model is only defined for RT_CONT.
--	--	--

out Arguments	Data type	Comment
currff	swap_curve_cff_ext	This object represents the created currency curve object from which several results and information can be retrieved e.g. the blended curve (see below).

Discount function for the float leg currency in the currency fix-float swap:

```

disc_func      cff_curve_flt_disc (   date           trade_date,
                                     instrument option(nullable)  fx_spot,
                                     curve option(nullable)       fxswap_crv,
                                     instr_class_name option(nullable)  depo_class_fxbase,
                                     instr_class_name option(nullable)  depo_class_fxprice,
                                     curve                           fixflt_crv,
                                     number option(nullable)       flt_first_fix,
                                     logical                         merge_crv,
                                     logical                         prio_fxswap,
                                     integer                         blend_buf_days,
                                     disc_func                       fix_df_disc,
                                     disc_func option(nullable)    flt_df_disc,
                                     fwd_func                       flt_fwd,
                                     string option(nullable)       name,
                                     interpolator option(nullable)  disc_ip,
                                     rate_type option(nullable)    disc_ip_rt,
                                     out swap_curve_cff_ext        currff)

```

The only difference between `cff_curve_flt_disc(...)` and `cff_curve_fix_disc(...)` is that the `flt_df_disc` is optional while `fix_df_disc` is required.

4.4. *The swap_curve_cff_ext object*

The `swap_curve_cff_ext` object represents a currency swap curve created from a currency fix-float swap and contains the input data and several intermediate results as well as provides access to and calculates the general `disc_func` object.

The member functions are:

Function	Arguments	Comment
string <code>name()</code>	none	Returns the name of the curve.
curve <code>fixflt_curve()</code>	none	Returns the currency fix-float swap curve.
number <code>fix_freq ()</code>	none	Returns the coupon frequency for the fixed leg.
number <code>flt_freq ()</code>	none	Returns the forward rate tenor for the float leg.
day_count_method <code>fix_dc()</code>	none	Returns the daycount method for the fixed leg.
day_count_method <code>flt_dc ()</code>	none	Returns the daycount method for the float leg.
string <code>fix_ccy ()</code>	none	Returns the currency for the fixed leg.
string <code>flt_ccy()</code>	none	Returns the currency for the float leg.
fwd_func <code>flt_fwd ()</code>	none	Returns the <code>fwd_func</code> for the float leg.
disc_func <code>flt_disc_df ()</code>		Returns the <code>disc_func</code> for the float leg. Note that this is the input <code>disc_func</code> and not the fx-implied <code>disc_func</code> .
disc_func <code>fix_disc_df ()</code>	none	Returns the <code>disc_func</code> for the fixed leg. Note that this is the input <code>disc_func</code> and not the fx-implied <code>disc_func</code> .
disc_func <code>fix_adj_disc_df</code>	interpolator <code>disc_ip</code> , rate_type <code>disc_ip_rt</code>	Returns the fx-implied <code>disc_func</code> for the base leg in the currency basis swap.
disc_func <code>flt_adj_disc_df</code>	interpolator <code>disc_ip</code> , rate_type <code>disc_ip_rt</code>	Returns the fx-implied <code>disc_func</code> for the flat leg in the currency basis swap.

5. Creating a swap surface

The single currency swap curve can be represented as a surface where the extra dimension is the tenor of the forward index rate. The discount function is assumed to be the lowest tenor and will anchor the rate interpolation below the first forward tenor. Extrapolation is currently not done beyond the longest tenor.

There are several wrapper functions for creating the swap surface depending on what instruments are used and other assumptions. Here, the function which uses the FRA/IRS/Basis swap(2 swaps) is shown and will represent the principles.

```
swap_curve_surface    swap_surface_b2_create(
                        date                trade_date,
                        disc_func           df_disc,
                        curve option(nullable) short_main_crv,
                        curve option(nullable) fra_main_crv,
                        curve option(nullable) swap_main_crv,
                        vector(curve))      short_t_crv,
                        vector(curve)       fra_t_crv,
                        vector(curve)       basis_t_crv,
                        curve_prio          prio1,
                        curve_prio          prio2,
                        string               depo_class_name,
                        string              fra_main_class_name,
                        vector(string)       fra_t_class_name,
                        synt_short_style     synt,
                        interpolator option(nullable) bs_ip,
                        logical              merge_fra,
                        number               blend_buf_days,
                        vector(number) option(nullable) edfut_cvx_adj,
                        logical              x_pol_basis,
                        number option(nullable) swap_first_fix )
```

The arguments that have been explained previously will not be discussed.

Return value	Comment
swap_curve_surface	The swap_curve_surface object is an “interpolation”-object and contains the discount function and all tenor forward functions.

Arguments	Data type	Comment
short_main_curve	curve	The main tenor curve for the short segment of the forward curve.
fra_main_curve	curve	The main tenor curve for the FRA/Futures segment of the forward curve. All FRAs must have the same tenor and the same tenor as the main IRS floating leg.
swap_main_curve	curve	The main tenor curve for the IRS segment of the forward curve. All swaps must have the same tenor of the floating leg and this tenor must match all the FRA/Futures contract main tenors.
short_t_curve	curve	The non-main tenor short curves.
fra_t_curve	vector(curve)	The non-main tenor FRA/Futures curve.
basis_t_curve	vector(curve)	The Basis swap curves. All Basis swaps must have one tenor of the floating leg that matches the main tenor IRS curve. The tenor of the other floating leg must match the corresponding non-main tenor FRA/Futures curve. The spreads must be a 2 swaps quotation.

6. Curves

All curves need to be setup to include only one type of instrument. The blending of different types of instruments will be done at runtime and blending of instruments with different tenors will typically generate an error (for forward curves).

A complete set of curves for a currency thus comprises of:

- Discounting curve e.g. OIS.
- IRS curve for the main floating leg tenor.
- FRA's or Futures for the main tenor.
- Basis swap curves (or IRS curves) for other tenors.
- FRA's or Futures for other tenors.

As an example, for EUR this would typically mean:

- Eonia is used for discounting.

6M tenor (main):

- IRS: Annual fixed vs. 6M EURIBOR; 1yr – 60yrs.
- FRA: 6M EURIBOR; 0x6, 6x12, 12x18, 18x24 and 1x7, 2x8, ...

3M tenor:

- Basis swaps (quoted as either 1 swap or 2 swaps): 3M/6M; 1yr – 50 yrs.
- Money market futures: 3M EURIBOR (convexity adjusted)
- FRA: 3M EURIBOR; 0x3, 3x6, 6x9,... and 1x4, 2x5, ...

1M tenor:

- Money market monthly IRS: 2M-12M.
- Basis swaps (quoted as either 1 swap or 2 swaps): 1M/6M; 1yr – 50 yrs.

- FRA: 1M EURIBOR; 0x1

12M tenor:

- Basis swaps (quoted as either 1 swap or 2 swaps): 6M/12M; 1yr – 50 yrs.
- FRA: 12M EURIBOR; 12x24

As an example of typical a result, let us look at the EUR curves for 14-Mar-2012 (using ICAP prices provided by Reuter).

In the graph below the OIS, 1M, 3M, 6M and 12M tenors are shown. The curves are created from:
OIS: EONIA swaps

1M: Money Market IRS and 1M/6M Basis swaps,

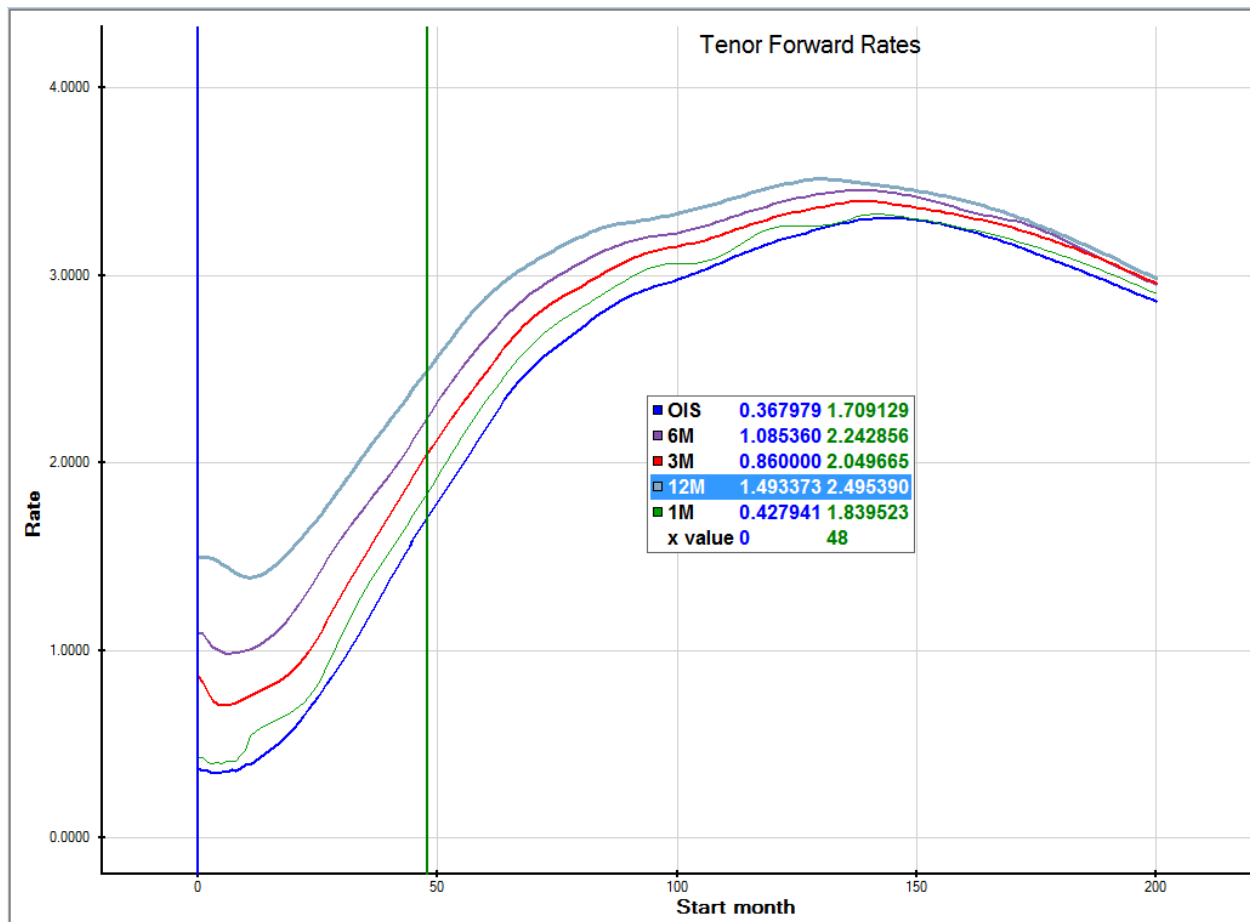
3M: 3M Futures and 3M/6M Basis swaps,

6M: 6M FRA and 6M IRS,

12M: 12M FRA and 6M/12M Basis swaps

In addition, for each forward tenor, synthetic Deposits and FRAs are created to fill up the short end before the first instrument. The rates for the synthetic instruments are calculated from an interpolated basis vs. the OIS curve.

In the graph below, only “exact fit”-models are used. The interpolation model for the OIS curve is the Hagan/West forward monotone convex spline. The forward rate interpolation model is a hermite cubic spline.



References

- Adams, K. J., & Van Deventer, D. R. (1994). Fitting yield curves and forward rate curves with maximum smoothness. *The Journal of Fixed Income*.
<http://www.ijournals.com/doi/abs/10.3905/jfi.1994.408102>
- Andersen, L. (2007). Discount curve construction with tension splines. *Review of Derivatives Research* 10, Issue 3, 2007. <http://www.springerlink.com/index/F7675413855KLM87.pdf>
- Forsgren, A. (1998). A note on maximum smoothness approximation of forward interest rates - corrections to Adams-Deventer. Report TRITA-MAT-1998-OS3, Department of Mathematics, Royal Institute of Technology. <http://www.math.kth.se/~andersf/doc/smoothadd.ps>
- Hagan, P. S., & West, G. (2006). Interpolation methods for curve construction. *Applied Mathematical Finance*, 13(2), 89–129.
<http://www.tandfonline.com/doi/full/10.1080/13504860500396032#.Uadprp38LmE>
- Tinggaard C. (1997). Nonparametric smoothing of yield curves. *Review of Quantitative Finance and Accounting*, 9, 251-267, 1997. <http://www.math.ku.dk/~rolf/teaching/tinggaardRQFA.pdf>