LCH SwapAgent Ltd

Swaption Volatility Cube Generation and Pricing Model Review

February 2018

Version 1.1

Contents

1 Executive Summary

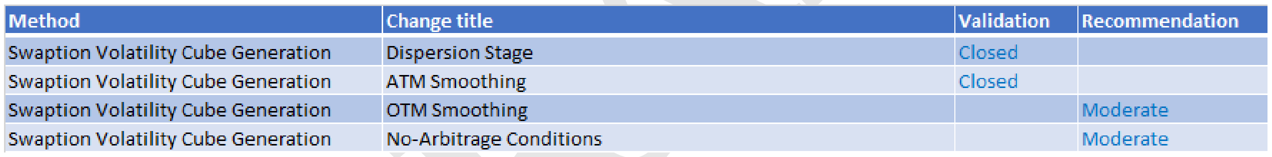
1.1 Context and Scope of the Review

The scope of this document is the review of the swaption volatility cube generation and pricing model. Our main points of focus are listed below:

- Review of the swaption volatility cube generation;
- Validation of the swaption pricing and risk model, including life cycle events;

The analysis was based on methodology documents provided by the LCH SwapAgent team, and supported by meetings and conference calls to discuss key points and findings. The LCH SwapAgent team also provided samples of market data for swaption volatility and interest rate curves, which allowed to benchmark and challenge some aspects of the modelling framework and provide concrete numerical results.
The quantitative analysis has been performed using BMAnalytics, which is BMA proprietary C++ analytics library.

1.2 Validations and Recommendations



Method	Change title	Validation	Recommendation
Swaption Volatility Cube Generation	Dispersion Stage	Closed	
Swaption Volatility Cube Generation	ATM Smoothing	Closed	
Swaption Volatility Cube Generation	OTM Smoothing		Moderate
Swaption Volatility Cube Generation	No-Arbitrage Conditions		Moderate

1.2.1 Swaption Volatility Cube Generation

1.2.1.1 Swaption Volatility Cube Generation: Dispersion Stage (moderate)(closed)

The Dispersion stage is an iterative filtering method which aims at extracting a valid average volatility from the set of inputs provided by market contributors for a given structure point (Expiry, Tenor, Strike).
As the numerical analysis and results are conclusive, we validate the Dispersion Stage of the swaption volatility cube generation.

1.2.1.2 Swaption Volatility Cube Generation: ATM Smoothing (moderate)(closed)

A smoothing algorithm, based on a local, vega-weighted, linear interpolation method, is implemented to smooth the ATM surface and attempt to detect and fix any potential outlier.
As the numerical analysis and results are conclusive, we validate the ATM Smoothing stage of the swaption volatility cube generation.

1.2.1.3 Swaption Volatility Cube Generation: OTM Smoothing (moderate)

A smoothing algorithm, based on a local, vega-weighted, linear interpolation method, is implemented to smooth the OTM slice and attempt to detect and fix any potential outlier. This algorithm tends to work well under normal market conditions, but numerical analysis shows that it is less efficient under stressed market conditions, with a manual review process likely to take place in this case.

We recommend to enhance the manual review process by defining a fall-back method to adjust the OTM volatility slice when the OTM Smoothing process fails to ensure a well-behaved volatility slice, which could occur under stressed market conditions.

1.2.1.4 Swaption Volatility Cube Generation: No-Arbitrage Conditions (moderate)

The definition of a valid swaption volatility cube should include the standard no-arbitrage conditions, such as the butterfly arbitrage condition and the in-plane triangular arbitrage condition. However, these conditions are not part of the smoothing stage, which does not guarantee an arbitrage-free cube, as shown with several numerical examples in the case of stressed market conditions.

We recommend to enhance the manual review process by defining a fall-back method to adjust the OTM volatility slice when the OTM Smoothing process fails to ensure a well-behaved volatility slice, which could occur under stressed market conditions. In a second step, it should be considered to add specific no-arbitrage conditions in the smoothing algorithms for both ATM and OTM.

2 Swaption Volatility Cube Generation

2.1 Executive summary

2.1.1 Context and scope of our review

In this section, we review the swaption volatility cube generation.

2.1.2 Validation

2.1.2.1 Swaption Volatility Cube Generation: Dispersion Stage (moderate)(closed)

The dispersion stage is the first step of the volatility cube generation. It consists in an iterative filtering of the market participants submissions for each data point defined on the cube.
We validate the proposed dispersion method for the filtering of market participants submissions of volatility quotes.

2.1.2.2 Swaption Volatility Cube Generation: ATM Smoothing (moderate)(closed)

A smoothing algorithm, based on a local, vega-weighted, linear interpolation method, is implemented to smooth the ATM surface and attempt to detect and fix any potential outlier.
As the numerical analysis and results are conclusive, we validate the ATM Smoothing stage of the swaption volatility cube generation.

2.1.3 Recommendation

2.1.3.1 Swaption Volatility Cube Generation: OTM Smoothing (moderate)

A smoothing algorithm, based on a local, vega-weighted, linear interpolation method, is implemented to smooth the OTM slice and attempt to detect and fix any potential outlier. This algorithm tends to work well under normal market conditions, but numerical analysis shows that it is less efficient under stressed market conditions, with a manual review process likely to take place in this case.
We recommend to enhance the manual review process by defining a fall-back method to adjust the OTM volatility slice when the OTM Smoothing process fails to to ensure a well-behaved volatility slice, which could occur under stressed market conditions.

2.1.3.2 Swaption Volatility Cube Generation: No-Arbitrage Conditions (moderate)

The definition of a valid swaption volatility cube should include the standard no-arbitrage conditions, such as the butterfly arbitrage and the in-plane triangular arbitrage. However, these conditions are not part of the smoothing stage, which does not guarantee an arbitrage-free cube, as shown with several numerical examples in the case of stressed market conditions.

We recommend to enhance the manual review process by defining a fall-back method to adjust the OTM volatility slice when the OTM Smoothing process fails to to ensure a well-behaved volatility slice, which could occur under stressed market conditions. In a second step, it should be considered to add specific no-arbitrage conditions in the smoothing algorithms for both ATM and OTM.

2.2 Swaption Volatility Cube Generation: Dispersion Stage

2.2.1 Current Method

The objective of the dispersion step is to filter the input volatility data provided by market participants on a daily basis. The dispersion is run for each point or structure of the volatility cube individually. A structure is defined by the expiry, underlying swap tenor or tail, and strike of the swaption. The resulting data set is then passed to the next step, the smoothing process.

Initial Data Set
The data set considered in this stage is a vector of volatility points for the same structure, each point corresponding to a given market contributor.

Data Cleaning
The first step consists in removing any volatility which is not strictly positive. This is a standard and basic data cleaning step.

Order of Magnitude Check
The Order of Magnitude Check is the second step, which is a first filter designed to remove potential outliers. It compares the ratio of each pair of data points against predefined and configurable thresholds. If no breach is detected, the data set is moved to the next step. Otherwise, a Group Check process is run to identify and remove the minimum number of points in order to obtain a data set that would pass the Order of Magnitude Check. For a data set of size N, the Order of Magnitude Check is successful if the following condition is met:

$$\forall i, j \in [1, .., N] \, v_i/v_j \leq h_r \; and \; v_j/v_i \leq h_r$$

When this condition is not valid, the Group Check process is run as follows:

- Select a number of points k to be removed, starting with $k = 1$;
- Apply the Order of Magnitude Check on all the sub-groups of $N - k$ points;
- If a valid group of size $N - k$ is found, then the corresponding group is stored and the process continues for the other groups of size $N - k$ in order to identify all groups of size $N - k$ passing

the Order of Magnitude Check;

- If at least one group of size $N - k$ is valid, then the process stops. The group of size $N - k$ with the closest average to the initial (full) group average is retained. The corresponding k points removed are flagged and dropped.
- If no group of size $N - k$ is valid, then the process is run again with $k + 1$.

First Dispersion Check
The third step consists in running the First Dispersion Step. The modified mean is calculated for each point as

$$g_i = \frac{1}{N-1} \sum_{j=1, j \neq i}^{N} v_j$$

The absolute outage is defined as the distance between the volatility point and its modified mean:

$$d_i^a = vi - g_i$$

The percentage outage is defined as the relative distance between the volatility point and its modified mean:

$$d_i^o = \frac{d_i^a}{g_i}$$

Another indicator is introduced to assess if the volatility point is close enough to quotes from other contributors. A point v_i is deemed to have peer support if $\frac{v_{i+1} - v_i}{\max(v_i, v_{i+1})} < h_p$ or $\frac{v_i - v_{i-1}}{\max(v_i, v_{i-1})} < h_p$ where v_{i+1} and v_{i-1} are the two closest quotes to v_i, and h_p is a configurable threshold level.
A volatility point is then deemed valid if and only if either $|d_i^a| \leq h^a$ or $|d_i^o| \leq h^o$ or v_i has peer support.

Note that an additional test, the maximum tolerance test, is run. However, as it is only used for flagging and diagnostic purposes, and does not affect the filtering of volatility points, it has not been included in the review.

Dealer Ranking Filter
The fourth step is a Dealer Ranking Filter. As this is not implemented and scheduled to go live as yet, this step has not formally been reviewed. Its principle seems however quite valid, as it would favour volatility inputs from dealers with larger volumes.

Second Dispersion Check
The fifth step consists in running a Second Dispersion Step. This is in practice very similar to the

First Dispersion Check, but applied to a potentially smaller set.

Min and Max Check

The sixth step is reasonably simple and consists in including back volatility points that might have been removed at a previous stage, if they are being between the minimum and maximum values of the current data set.

Minimum Set of Points

As the different steps of the Dispersion process are run, the number of remaining points after each step is checked to ensure that a minimum number of values is retained.

Top and Tail

The final Top and Tail stage consists in taking the average of the data set, after removing the highest and lowest values using the following rule set, based on the final data set size n:

- if $n \geq 7$, drop the 2 highest and the two lowest values before taking the average;
- if $5 \leq n \leq 6$, drop the highest and the lowest values before taking the average;
- if $n \leq 4$ keep all values and take the average.

The final value is then assigned to the given structure and will be used in the Smoothing stage detailed in the next section.

2.2.2 Review and Discussion

The Dispersion method uses reasonably standard methods to filter iteratively a set of data points in order to extract a valid average.

The Dispersion algorithm has been implemented in BMAnalytics, which is BMA proprietary C++ analytics library. Tests have then been run using contributor values and final outputs provided by LCH. This allowed for an independent benchmarking of the method.

Tests have been run on the ATM volatility, and on various moneyness in the following set: {-250,-100,-20,20,100,250}. The values obtained with our independent implementation match the final values provided by LCH.

Numerical analysis leads to two minor points that might need further analysis:

- **Group Check**: The Group Check process has a bias towards keeping large values as the group with the closest average to the full population will be selected in case several groups of identical sizes have been identified as valid. This could potentially favour large outliers - and result in a significant bias for the average. It should be considered to use an alternative metrics as a tiebreaker - a candidate could be to keep the group with the lower standard deviation.

- **Identical Values**: The treatment of strictly identical values in the set of point submitted by contributors is not very clear (see structure point Expiry = 180, Tail = 2555 of the Strike = -250 surface). From numerical testing, it looks as if the value is used with only one occurrence in the Dispersion method even if it has been submitted several times, which could lead to a bias for the final group average.

As the numerical analysis and results are conclusive, **we validate the Dispersion Stage of the swaption volatility cube generation.**

2.3 Swaption Volatility Cube Generation: Smoothing Stage

2.3.1 Current Method

The smoothing stage objective is to build a well-shaped volatility cube from the outputs of the dispersion stage. The shaping process will identify and attempt to correct any outage by comparing the level of each point of the cube to linear interpolated values based on neighbouring points.
The smoothing process proceeds sequentially with a re-shaping of the ATM volatility surface, followed by a re-shaping of the OTM volatility slices. While both steps uses similar methods is spirit, the ATM step works on the full grid with horizontal and vertical interpolations while the OTM step focuses only on a slice, with interpolation in the strike diection only.

2.3.1.1 ATM Smoothing Process

The ATM smoothing process starts with the calculation of the horizontal and vertical interpolated values for each grid point, using a vega-weighted linear interpolation method. More precisely, for a given point $v(i,j)$ on the grid, corresponding to the i-th expiry and j-th tenor, the interpolated levels are calculated as follows:

- In the interior of the grid, i.e. for $0 < i < I$ and $0 < j < J$:

$$\begin{aligned} v^h(i,j) &= \frac{y(i+1,j)-y(i,j)}{y(i+1,j)-y(i-1,j)}v(i-1,j)+\frac{y(i,j)-y(i-1,j)}{y(i+1,j)-y(i-1,j)}v(i+1,j) \\ v^v(i,j) &= \frac{y(i,j+1)-y(i,j)}{y(i,j+1)-y(i,j-1)}v(i,j-1)+\frac{y(i,j)-y(i,j-1)}{y(i,j+1)-y(i,j-1)}v(i,j+1) \end{aligned}$$

- At the boundaries of the grid, i.e. for $i = 0$ or $i = I$ or $j = 0$ or $j = J$:

$$v^h(i,j) = \frac{y(i,j) - y(i+1,j)}{y(i+1,j) - y(i+2,j)}(v(i+2,j) - v(i+1,j)) + v(i+1,j)$$

Note that I represents the number of expiries and J the number of tenors defining the ATM volatility surface. The vega for the point (i,j) of the grid is given by $y(i,j)$.
From the interpolated values, it is possible to define the absolute and percentage outages, in the horizontal and vertical directions, as follows:

$$\begin{aligned} a^h(i,j) &= v^h(i,j) - v(i,j) \\ a^v(i,j) &= v^v(i,j) - v(i,j) \\ d^h(i,j) &= \frac{v^h(i,j) - v(i,j)}{v^h(i,j)} \\ d^v(i,j) &= \frac{v^v(i,j) - v(i,j)}{v^v(i,j)} \end{aligned}$$

The maximum percentage outage, in absolute value, will determine which direction to use to determine if there is a breach. If $d^v(i,j) = max(d^v(i,j), d^h(i,j))$ then $d(i,j) = d^v(i,j)$ and $a(i,j) = a^v(i,j)$, otherwise $d(i,j) = d^h(i,j)$ and $a(i,j) = a^h(i,j)$.
A point on the ATM grid is deemed to be in breach if and only if $|a(i,j)| > h_a$ and $|d(i,j)| > h_d$. This is tested for each point on the grid.
Once the breaches have been identified, the next step consists in assessing if they can be fixed, i.e. if they can be brought closer to their interpolated value. A simple requirement is that moving point (i,j) should not increase the percentage outage of its neighbouring points, in the horizontal and vertical directions. Note that the diagonals are not used in this approach. This condition is verified if the percentage outage at (i,j) is of opposite sign to percentage outages of its surrounding points. This translates into the simple set of conditions below:

$$\begin{gathered} d(i,j)d^h(i+1,j) < 0 \\ d(i,j)d^h(i-1,j) < 0 \\ d(i,j)d^v(i,j+1) < 0 \\ d(i,j)d^v(i,j-1) < 0 \end{gathered}$$

The fixing procedure for each point (i,j) in breach and satisfying the conditions above consists in an iterative process by which the point is moved towards its interpolated values in small steps.

After each adjustment, the conditions are checked again to determine if the process can continue or if it should stop to prevent a deterrent impact on the percentage outage of surrounding points. The process stops ultimately as point reaches its interpolated value. The increments are set to 0.01. Note that the vegas are not updated as the volatility point is moved.
In the current version of the algorithm, the AMT Smoothing process is not an iterative process which implies that the volatility levels and vegas used are always the initial values, and do not get updated with any fixed values as the algorithm progresses on a the grid.

2.3.1.2 OTM Smoothing Process

The OTM process is fairly similar to the ATM process. Two specific steps are run at the beginning of the procedure.
The first one consists in using the delta to flatten the volatility on the wings of the OTM slice, to avoid unreliable quotes for deep out-of-the-money strikes. The delta is calculated for each of the point on the slice using the Black Normal model:

$$\delta_k = \Phi\left(\frac{m_k}{v_k\sqrt{T}}\right)$$

where m_k is the moneyness and T is the expiry of the option. The deltas are compared to a configurable delta threshold h_d. If $\delta_k < h_d$ then the volatility point is deemed unreliable and v_k is set to the value of the closest point with a delta above the threshold. Deltas are calculated for payer swaptions above the ATM and for receiver swaptions below the ATM. This results in a flat extrapolation on the left and right side of the slice, starting from the first and last points with a significant delta value, which we not k_0 and k_N respectively.

The second specific step consists in checking the overall shape of the vega as a function of strike. The vega is calculated in the Black Normal model:

$$y_k = \sqrt{T}\phi\left(\frac{m_k}{\sigma_k\sqrt{T}}\right)A(i,j)$$

The vega function is expected to have a bell shape, with its maximum at the ATM point. This translates into the following condition, which is applied on both sides of the ATM:

$$\forall m_{k-1} > 0, m_k > 0, m_{k+1} > 0 \; or \; \forall m_{k-1} < 0, m_k < 0, m_{k+1} < 0$$
$$(y_k - y_{k-1}) \times (y_{k+1} - y_k) \geq 0$$

If this condition is not valid, then the point is flagged for review. The first level of correction consists in setting the volatility to the ATM value. If this is not sufficient, then the vega will be

manually reviewed.
Once this two preliminary checks have been completed, the interpolated values for each slice point are calculated using a vega-weighted linear interpolation method. More precisely, for a given point v_k on the slice, corresponding to the k-th strike the interpolated levels is calculated as follows:

- In the interior of the slice, i.e. for $k_0 < k < k_N$:

$$v_k^* = \frac{y_{k+1} - y_k}{y_{k+1} - y_{k-1}} v_{k-1} + \frac{y_k - y_{k-1}}{y_{k+1} - y_{k-1}} v_{k+1}$$

- At the boundaries of the slice, i.e. for $k = k_0$ or $k = k_N$:

$$v_k^* = \frac{y_k - y_{k+2}}{y_{k+1} - y_{k+2}} (v_{k+1} - v_{k+2}) + v_{k+1}$$

  and

$$v_k^* = \frac{y_k - y_{k-2}}{y_{k-1} - y_{k-2}} (v_{k-1} - v_{k-2}) + v_{k-1}$$

  respectively.

From the interpolated values, it is possible to define the absolute and percentage outages as follows:

$$\begin{aligned} d_k^a &= v_k^* - v_k \\ d_k^o &= \frac{v_k^* - v_k}{v_k^*} \end{aligned}$$

A point is then deemed to be in breach if $d_k^a > h_a$ and $d_k^o > h_o$.
Once the breaches have been identified, the same type of conditions as in the ATM Smoothing process is used to assess if a fix is possible. This is again based on the requirement that the percentage outage of the surrounding points should not increase in absolute terms. This translates into:

$$\begin{aligned} d_k^o \times d_{k+1}^o &< 0 \\ d_k^o \times d_{k-1}^o &< 0 \end{aligned}$$

Note that and the low and high strike boundaries, as well as around the ATM point, only one surrounding point is used.
The same type of iterative process as for the ATM Smoothing is then run to fix any breach deemed fixable.
In the current version of the algorithm, the OTM Smoothing process is not an iterative process which implies that the volatility levels and vegas used are always the initial values, and do not get

updated with any fixed values as the algorithm progresses on a given slice.

2.3.2 Review and Discussion

2.3.2.1 ATM Smoothing Process

The ATM Smoothing algorithm has been implemented in BMAnalytics, which is BMA proprietary C++ analytics library. Tests have then been run using output from the Dispersion stage and final outputs provided by LCH. This allowed for an independent benchmarking of the method.

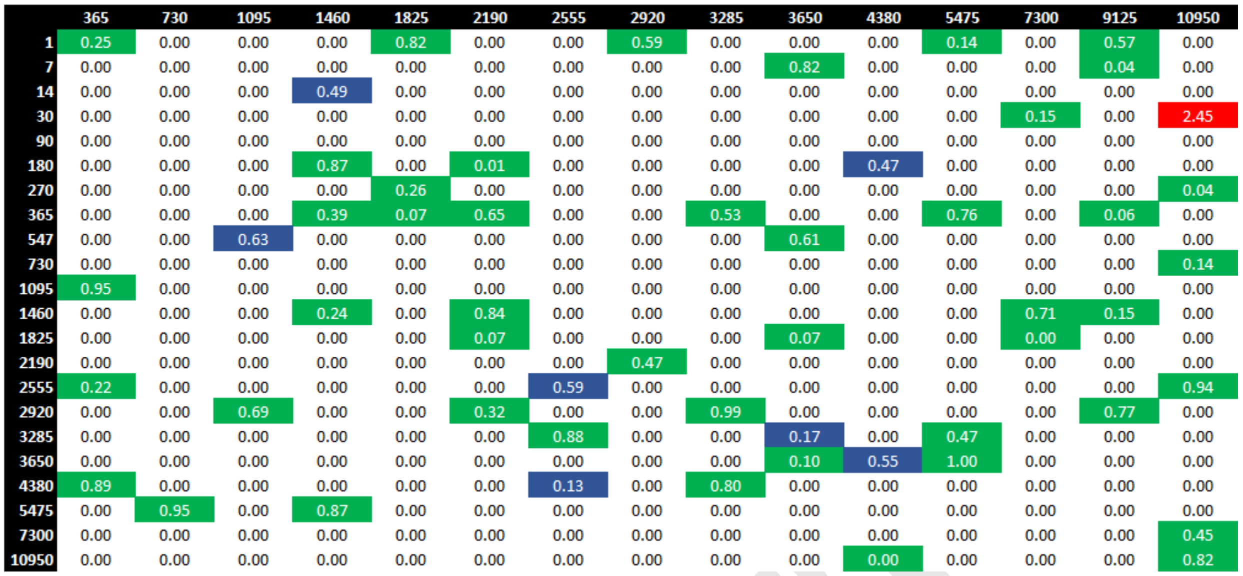
BMA's implementation is slightly different from the one defined in LCH documentation as it is using a partially iterative algorithm:

- Breaches are identified based only on the initial volatility levels. Consequently, the set of breaches is not updated after volatility points are fixed.
- Updated volatilities are used to evaluate the conditions to determine if a breach can be fixed.
- Updated volatilities are used in the iterative process run to fix breaches.

The algorithm starts at the upper left corner of the surface, i.e. the lower expiry and shorter tenor. Any difference between the output value of our implementation and the final value provided by LCH is analysed to assess if this difference is only due to the partially iterative algorithm used. The ATM volatility surface resulting from the partially iterative algorithm is very close from the output provided by LCH, as seen with the figures below. Most of the differences are identified as coming from the difference between the algorithms, namely that the use of a partially iterative algorithm.
The following color scheme is used on the results to simplify their interpretation:

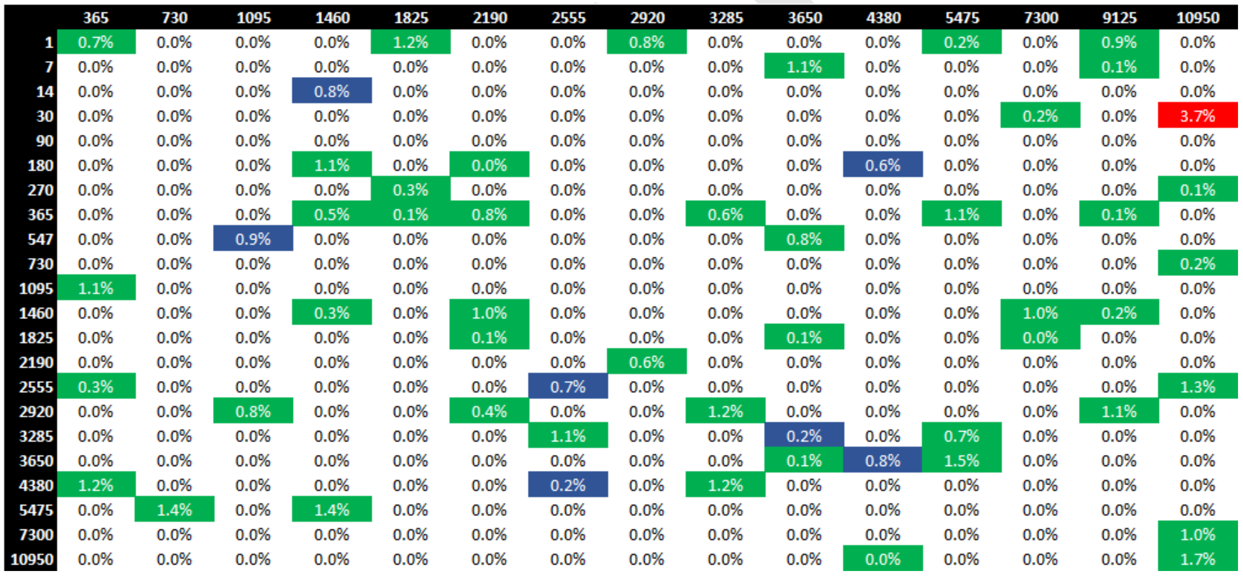
- Points in green in Figure 1 and Figure 2 represents points that have been modified by the LCH method.
- Points in blue represents points that have been modified only by the iterative algorithm.
- Points in red indicate significant differences.

As the numerical analysis and results are conclusive, **we validate the ATM Smoothing Stage of the swaption volatility cube generation.**
The ATM Smoothing stage should however be complemented by the introduction of no-arbitrage conditions. This point is discussed in details in a dedicated section below.



	365	730	1095	1460	1825	2190	2555	2920	3285	3650	4380	5475	7300	9125	10950
1	0.25	0.00	0.00	0.00	0.82	0.00	0.00	0.59	0.00	0.00	0.00	0.14	0.00	0.57	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.82	0.00	0.00	0.00	0.04	0.00
14	0.00	0.00	0.00	0.49	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.00	2.45
90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
180	0.00	0.00	0.00	0.87	0.00	0.01	0.00	0.00	0.00	0.00	0.47	0.00	0.00	0.00	0.00
270	0.00	0.00	0.00	0.00	0.26	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
365	0.00	0.00	0.00	0.39	0.07	0.65	0.00	0.00	0.53	0.00	0.00	0.76	0.00	0.06	0.00
547	0.00	0.00	0.63	0.00	0.00	0.00	0.00	0.00	0.00	0.61	0.00	0.00	0.00	0.00	0.00
730	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14
1095	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1460	0.00	0.00	0.00	0.24	0.00	0.84	0.00	0.00	0.00	0.00	0.00	0.00	0.71	0.15	0.00
1825	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00
2190	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.47	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2555	0.22	0.00	0.00	0.00	0.00	0.00	0.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.94
2920	0.00	0.00	0.69	0.00	0.00	0.32	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.77	0.00
3285	0.00	0.00	0.00	0.00	0.00	0.00	0.88	0.00	0.00	0.17	0.00	0.47	0.00	0.00	0.00
3650	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.55	1.00	0.00	0.00	0.00
4380	0.89	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00
5475	0.00	0.95	0.00	0.87	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7300	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.45
10950	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.82

Figure 1: Absolute Volatility Differences: Partially Iterative Algorithm vs. LCH SwapAgent Method



	365	730	1095	1460	1825	2190	2555	2920	3285	3650	4380	5475	7300	9125	10950
1	0.7%	0.0%	0.0%	0.0%	1.2%	0.0%	0.0%	0.8%	0.0%	0.0%	0.0%	0.2%	0.0%	0.9%	0.0%
7	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.1%	0.0%	0.0%	0.0%	0.1%	0.0%
14	0.0%	0.0%	0.0%	0.8%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
30	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.2%	0.0%	3.7%
90	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
180	0.0%	0.0%	0.0%	1.1%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.6%	0.0%	0.0%	0.0%	0.0%
270	0.0%	0.0%	0.0%	0.0%	0.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.1%
365	0.0%	0.0%	0.0%	0.5%	0.1%	0.8%	0.0%	0.0%	0.6%	0.0%	0.0%	1.1%	0.0%	0.1%	0.0%
547	0.0%	0.0%	0.9%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.8%	0.0%	0.0%	0.0%	0.0%	0.0%
730	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.2%
1095	1.1%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
1460	0.0%	0.0%	0.0%	0.3%	0.0%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.0%	0.2%	0.0%
1825	0.0%	0.0%	0.0%	0.0%	0.0%	0.1%	0.0%	0.0%	0.0%	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%
2190	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.6%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
2555	0.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.7%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.3%
2920	0.0%	0.0%	0.8%	0.0%	0.0%	0.4%	0.0%	0.0%	1.2%	0.0%	0.0%	0.0%	0.0%	1.1%	0.0%
3285	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.1%	0.0%	0.0%	0.2%	0.0%	0.7%	0.0%	0.0%	0.0%
3650	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.1%	0.8%	1.5%	0.0%	0.0%	0.0%
4380	1.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.2%	0.0%	1.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
5475	0.0%	1.4%	0.0%	1.4%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
7300	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.0%
10950	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.7%

Figure 2: Relative Volatility Differences: Partially Iterative Algorithm vs. LCH SwapAgent Method

2.3.2.2 OTM Smoothing Process

The OTM Smoothing algorithm has been implemented in BMAnalytics, which is BMA proprietary C++ analytics library. Tests have then been run using output from the Dispersion stage and final outputs provided by LCH. This allowed for an independent benchmarking of the method.

BMA's implementation is slightly different from the one defined in LCH documentation as it is using a partially iterative algorithm:

- Breaches are identified based only on the initial volatility levels. Consequently, the set of breaches is not updated after volatility points are fixed.
- Updated volatilities are used to evaluate the conditions to determine if a breach can be fixed.
- Updated volatilities are used in the iterative process run to fix breaches.
- Vega is not updated even after a volatility point is fixed as it is considered a second order factor in the algorithm.

The algorithm is applied between the first and last strikes for which the delta is higher than the delta threshold. The OTM Smoothing algorithm is tested under normal market conditions and stressed market conditions.
Under normal market conditions, the OTM Smoothing method performs well, as the volatility inputs are already reasonably smooth. The use of a delta threshold helps with the volatility smile wings and removes any potentially unreliable data for deep out-of-the money strikes. Figure 3 below shows the volatility slice after smoothing for a few (Expiry, Tenor) pairs. It is also interesting to note that the algorithm is fairly stable under the perturbation of a small number of points.

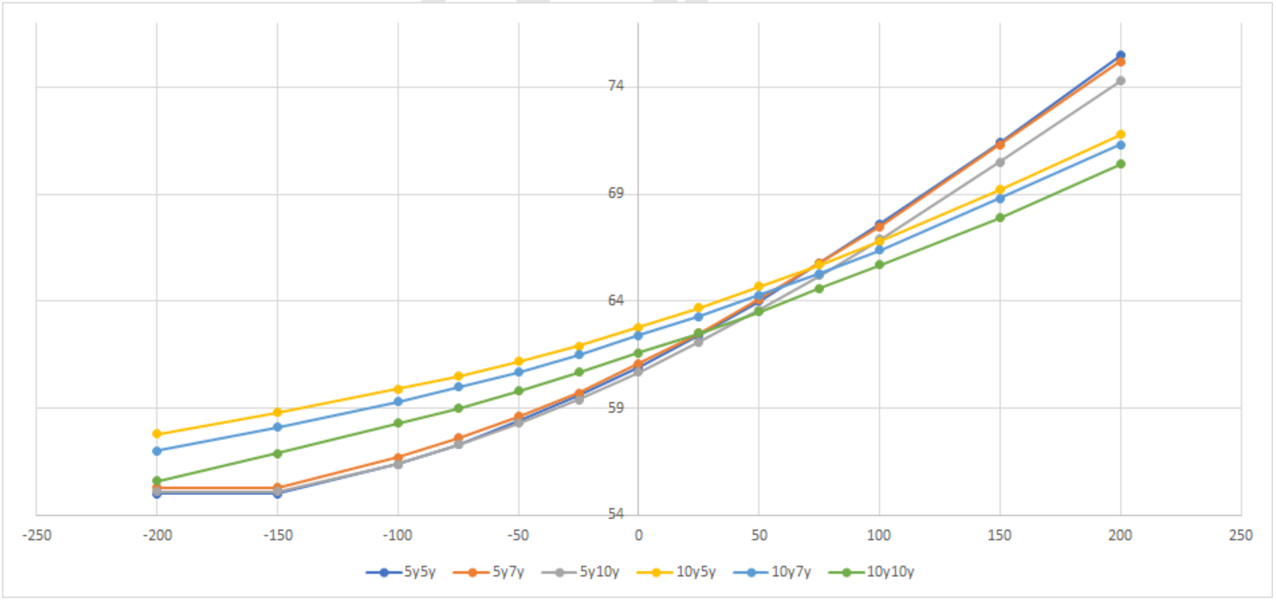


Figure 3: OTM Volatility Slices under Normal Market Conditions

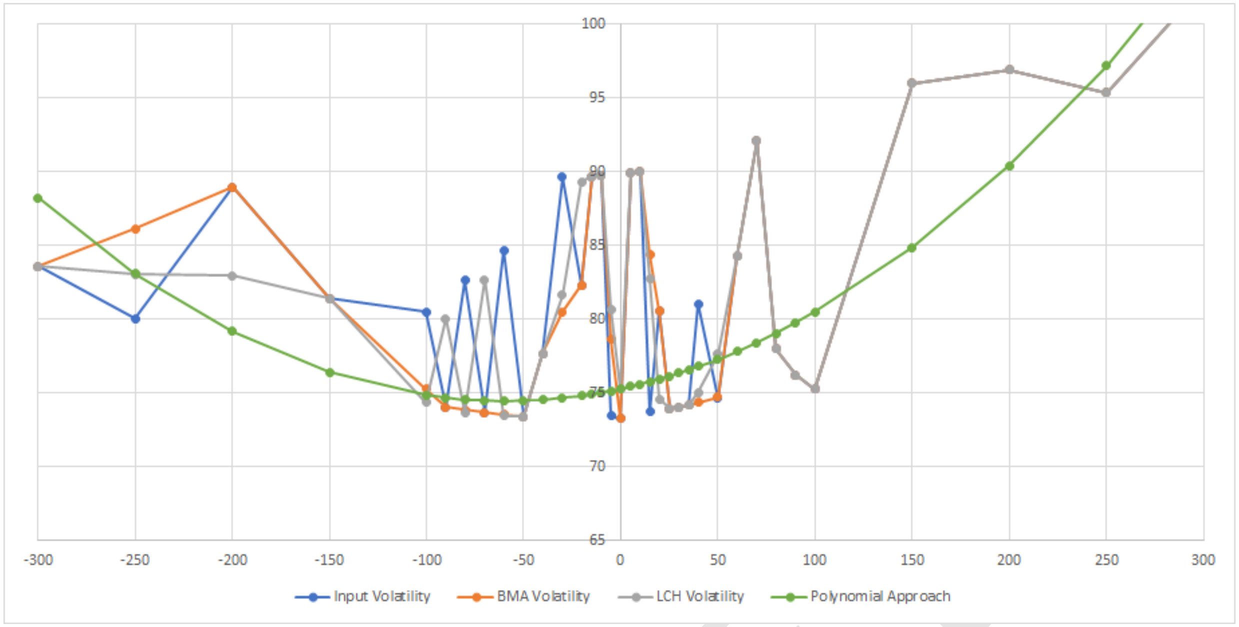


Figure 4: OTM Volatility Slice for Structure (1825, 3650) Under Stressed Market Conditions

Under stressed market conditions, the algorithm is less efficient as it struggles when several contiguous points have to be fixed. As shown with Figure 4, both the LCH and BMA implementations fail to deliver a smooth volatility curve. The same results can be obtained on many other pairs of (Expiry, Tenor) on stressed market inputs.
The frequency and size of the oscillations will make it difficult to use the corresponding slice in practice - typically for non-quoted moneyness values.
As will be detailed in the next section, this translates also into clear pricing arbitrages in the strike dimension, on quoted moneyness points.

It is very likely in this case, that a manual review process will take place to adjust the volatility surface before it is actually used for pricing and risk purposes. However the manual review process does not use any precise fall-back method which could make it difficult to find a valid adjustment.

Several parametric methods exist in the literature to construct implied volatility surfaces (see [3] for example). For illustration purposes, we take a simple approach with a quadratic polynomial form defined as:

$$v(m) = b_1 + b_2 \times m + b_3 \times m^2$$

where m is the moneyness. Parameter $b1$ is given by the ATM volatility for the particular (Expiry,Tenor) for OTM slice considered. The other parameters, b_2 and b_3 can be calibrated using a

simple least mean square method. The results for structure (1825, 3650) is displayed on Figure 4. The resulting quadratic form could be used as a fall-back method to support the manual review and adjustment process.
We recommend to enhance the manual review process by defining a fall-back method to adjust the OTM volatility slice when the OTM Smoothing process fails to ensure a well-behaved volatility slice, which could occur under stressed market conditions.

2.4 Swaption Volatility Cube Generation: No-Arbitrage Conditions

The volatility cube process does not include any no-arbitrage conditions. The smoothing process in particular attempts to detect and eventually fix outages by using regularity conditions which do not take into account the specific nature of the underlying data.
In this section, we use the final results of the smoothing stage and test if the volatility cube is arbitrage-free. We note $Sw(T, \theta, K)$ the price at time t of a payer swaption of strike K, with exercise date T and tenor θ. Two standard arbitrage conditions are checked:

- Butterfly arbitrage: for a given structure (Expiry, Tenor), the following relationship in the strike dimension should hold:

$$\forall \Delta K > 0, \;\; Sw(T, \theta, K + \Delta K) - 2 \times Sw(T, \theta, K) + Sw(T, \theta, K - \Delta K) \geq 0$$

- In-plane triangular arbitrage condition: for a given strike K, the following relationship in the (Expiry, Tenor) dimension should hold:

$$\forall T, \theta_1, \theta_2, \;\; Sw(T, \theta_1, K) + Sw(T + \theta_1, \theta_1 + \theta_2, K) \geq Sw(T, \theta_1 + \theta_2, K)$$

The tests for the butterfly arbitrage are run on OTM slices, while the in-plane triangular arbitrage condition is tested on the ATM surface.
The first step consists in converting the volatility points into price points, using a Black Normal model:

$$Sw(T, \theta, K) = A_{T,\theta} \times DF^{Option}_{disc} \times \Big[(F - K)\Phi(d_1) + \sigma\sqrt{T}\phi(d_1) \Big]$$

where

$$d_1 = \frac{F - K}{\sigma\sqrt{T}}$$

and $A_{T,\theta}$ is the annuity of the underlying swap.
All pricing inputs are provided by LCH and/or taken directly from the surface or slice considered, including the annuity, the volatility and the moneyness.

In plane triangular arbitrage can be found on several occasions on the tested ATM volatility surface. Examples include the following structure points:

- $(3650, 1825)$, $(5475, 3650)$ and $(3650, 5475)$
- $(2555, 365)$, $(2920, 730)$ and $(2555, 1095)$
- $(1825, 365)$, $(2190, 730)$ and $(1825, 1095)$

The arbitrages are however reasonably small, and it should be possible to adjust the corresponding volatilities to ensure no-arbitrage without affecting dramatically the overall shape of the surface. This would require adding this specific no-arbitrage condition to the ATM Smoothing process defined in the previous section.

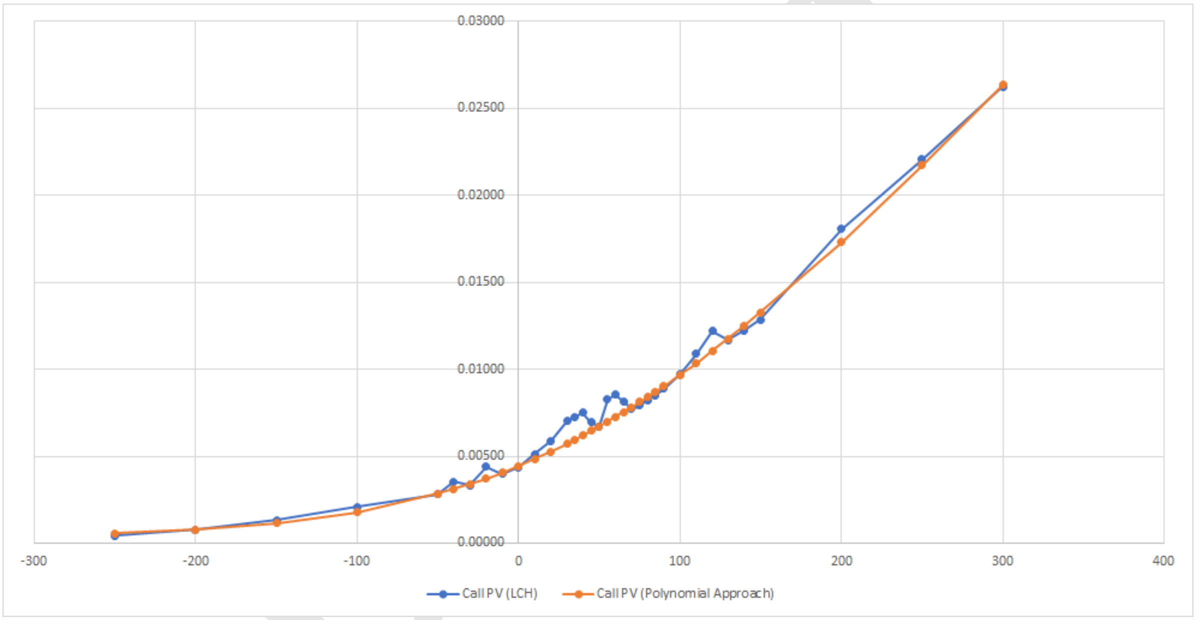


Figure 5: Swaption Price as a Function of Moneyness for Structure (1825, 3650) under Stressed Maket Conditions

Under stressed market conditions, butterfly arbitrage can be found frequently on a number of OTM slices. Taking structure (1825, 3650) again as an example, the swaption price as a function of moneyness is given in Figure 5. The butterfly condition is breached in this case as the price function is not a convex function of the moneyness. In this particular case, the swaption price is not an increasing function of moneyness either.

The arbitrages are reasonably severe in this case and it is unlikely that they could be fixed by applying regularity only conditions as the OTM Smoothing process attempts to. It is likely in these market conditions that a manual review process will be required. As mentioned in a previous

section, a dedicated volatility interpolation model could be used in this case, after the filtering done on the wings using the delta threshold.
A similar quadratic polynomial function as introduced in the previous section is calibrated and used for illustration purposes. The corresponding price function is displayed on Figure 5. As expected, this provides a much more stable price function, even if this simple method is not guaranteed to be arbitrage-free. A number of arbitrage-free implied volatility construction models are documented in the literature and could potentially be used as a fall-back method (see [3] for example).

We recommend to enhance the manual review process by defining a fall-back method to adjust the OTM volatility slice when the OTM Smoothing process fails to ensure a well-behaved volatility slice, which could occur under stressed market conditions. In a second step, it should be considered to add specific no-arbitrage conditions in the smoothing algorithms for both ATM and OTM.

References

[1] CDSClear Annual Review, Version 1.2, November 2016, BMA Risk Management Solutions.

[2] SwapAgent: Swaptions Murex Proof of Concept, V0, January 2018, LCH SwapClear.

[3] Implied volatility surface: construction methodologies and characteristics. July 2011. Cristian Homescu. Working Paper.