# Tensoring Dynamic Sensitivities and Dynamic Initial Margin

Mariano Zeron*
Ignacio Ruiz†

April 22, 2021

This paper has been published in Risk Magazine, Cutting Edge section, April 2021.

**Abstract**

In this paper Chebyshev Tensors are used to compute dynamic sensitivities of financial instruments within a Monte Carlo simulation. Dynamic sensitivities are then used to compute Dynamic Initial Margin as defined by ISDA (SIMM). The technique is benchmarked against the computation of dynamic sensitivities obtained by using pricing functions as found in risk engines. Numerical tests were done on FX swaps and Spread Options, where the technique obtains high accuracy at different percentiles of the simulated distributions with substantial computational gains over the benchmark.

## 1 Introduction

Sensitivities of portfolios are typically computed by banks every day. These are used for P&L risk management, hedging purposed, VaR calculations and its associated regulatory capital, etc.

Banks typically compute forward portfolio valuations inside Monte Carlo (MC) simulations (e.g. XVA and IMM capital simulations). However, to our knowledge, none compute forward sensitivities. Doing so would bring many advantages. For example, better understanding of expected and tail-event hedging needs; future VaR and market-risk capital; better management of future IM funding costs and accurate MVA values. From all these, IM and MVA have gained popularity in recent years due to the introduction of mandatory margining between financial institutions.

*m.zeron@mocaxintelligence.com
†i.ruiz@mocaxintelligence.com

There has been a worldwide push for strong collateralisation of OTC derivatives since the 2008 financial crisis. Between Variation Margin (VM) and Initial Margin (IM), there was, up to 2017, more than $1,000$ billion dollars in Margin. Out of the two margins, IM should show the highest growth, potentially surpassing the trillion dollar mark.

As IM requirements translate into funding cost and liquidity risk, it is important to manage these today and in the future. This requires simulating Initial Margin inside MC simulations. We call simulated Initial Margin, *Dynamic Initial Margin* (DIM).

Specific uses of DIM include trade pricing (MVA), regulatory capital (IMM and CVA-FRTB), risk management (hedging and tail risk), stress testing and most likely, accounting MVA. Therefore, sound models for DIM will be central for financial institutions.

To simplify IM reconciliation between counterparties, the industry has adopted the Standard Initial Margin Model (SIMM) for inter-bank IM posting ([1]), based on portfolio sensitivities. However, using risk engine pricing functions to compute sensitivities carries substantial computational cost. Assuming an average of 10-50 sensitivities per trade, and a typical MC simulation with $1,000,000$ nodes, the cost of computing DIM has an order of $O(10^7)$. This is prohibitively high in practice.

As function approximators, Chebyshev Tensors enjoy strong convergence properties and are evaluated very efficiently. Chebyshev Tensors have already been shown to accelerate a wide range of risk calculations ([6], [7], [3], [4]). In this paper, we use them to compute dynamic sensitivities within a MC simulation and subsequently dynamic SIMM. We show computational reductions of up to 97.5%, compared to the benchmark, while keeping very high levels of accuracy both at an averaged and tail-event level.

# 2 Chebyshev Tensors

Chebyshev Tensors lie at the heart of the techniques presented in this paper. This Section briefly describes their main mathematical properties. For further details we refer the reader to [2] and [6].

## 2.1 Chebyshev points and tensors

Polynomial interpolation has enjoyed a bad reputation for a good part of the 20-th century. Even some textbooks on the subject of function approximation warn against them (Appendix in [2]). What is often missed is that using the right geometry of points applied to the correct class of function yields optimal approximation properties. The correct geometry of points is given by Chebyshev points

The Chebyshev points associated with the natural number $n$ are defined as follows

$$x_j = \mathrm{Re}(z_j) = \frac{1}{2}(z_j + z_j^{-1}), \qquad 0 \leq j \leq n,$$

where $z_j$ are the $n+1$ equidistant points on the upper half of the unitary circle, $z_j = e^{i\frac{\pi j}{n}}$.

The extension to higher dimensions is obtained by taking the Cartesian product of one-dimensional Chebyshev grids.

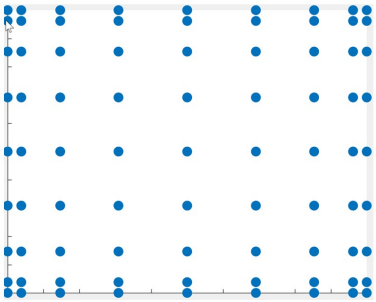

*Figure 1: Chebyshev points in dimension two.*

A *tensor* consists of a set of points $x_0, \ldots, x_n$ in Euclidean space along with a set of associated real values $v_0, \ldots, v_n$. When $x_0, \ldots, x_n$ are Chebyshev points, we have a *Chebyshev Tensor*. For the given points $x_0, \ldots, x_n$ and values $v_0, \ldots, v_n$, there is a unique polynomial $p_n$ that interpolates them. If $x_0, \ldots, x_n$ are Chebyshev points, we call $p_n$ the *Chebyshev Interpolant*.

## 2.2 Convergence properties

Chebyshev Tensors have unique convergence properties. When the function is Lipschitz continuous, convergence is guaranteed. If the function is differentiable, convergence is polynomial. The strongest form of convergence is obtained for analytic functions.[1] In this case, convergence is quasi-exponential. This means very few grid-points are needed to get high degrees of accuracy.

**Theorem 2.1.** *([3]) Let $f$ be a $d$-dimensional analytic function defined on $[-1,1]^d$. Consider its analytical continuation to a generalised Bernstein ellipse $E_p$, where it satisfies*

[1] We remind the reader that a function $f$ is analytic if for all $x$ in the domain of $f$, the Taylor expansion at $x$ converges to $f(x)$.

*$\|f\|_\infty \leq M$, for some $M$. Then, there exists a constant $C > 0$, such that*

$$\|f - p_n\|_\infty \leq C\rho^{-m}$$

*where $\rho = min_{(1\leq i\leq d)}\rho_i$, and $m = min_{(1\leq i\leq d)}m_i$. The collection of values $\rho_i$ define the radius of the generalised Bernstein ellipse $E_p$, and $m_i$ is the size of the Chebyshev grid for dimension $i$.*

We refer to [2] and [3] for a thorough treatment on the convergence Theorems of Chebyshev Tensors.

### 2.3 Pricing functions and Chebyshev Tensors

Pricing functions, outside isolated points, are often analytic. Not only is there growing evidence of this (see [3]), but practitioners regularly assume so, at least implicitly — the use of Taylor expansions to approximate them is an acknowledgement of this.

Non-differentiable points on pricing functions are usually payment dates, barriers, and strikes. These are easy to locate as they are defined by the trade itself. One deals with these points by splitting the domain of approximation along these points. One is left with a collection of sub-domains free of singularities over which Chebyshev Tensors enjoy the properties mentioned in Section 2.2.

Singular points can also be the result of structured payoffs. For example, taking the maximum between continuation and exercise value in American options introduces one. However, the continuation function is free of singularities and carries nearly all the computational cost. Hence, Chebyshev Tensors are built for this function.

Note that convergence rates of Chebyshev Tensors are determined by the smoothness of the function. How non-linear the trade is does not impact its smoothness. Linear products need less Chebyshev points than non-linear ones, but the type of convergence does not change.

Another case to consider is that of pricing functions that rely on simulations. What the authors have observed empirically is that Chebyshev Tensors approximate the function up to the level of accuracy provided by the latter. If prices are obtained through MC simulations with a noise of $1e^{-3}$, then the Chebyshev Tensor will reach this level of accuracy exponentially, remaining within this noise regardless of added Chebyshev points.

## 2.4 Tensor Extension Algorithms

Tensors suffer from the curse of dimensionality. One of the ways to side-step this problem, is through the use of the Tensor Extension Algorithm. Next we briefly describe it. Full details can be found in [5] and [8].

### 2.4.1 Tensors in TT format

The Tensor Extension Algorithm ([5]) works with tensors in TT format. These tensors admit a representation in terms of matrices which makes them memory efficient.

Say $\mathcal{X}$ is a $d$ dimensional tensor with $n$ grid points per dimension. The memory cost of storing the values on the grid is $\mathcal{O}(n^d)$. The TT Tensor, however, can be stored with $\mathcal{O}(dnr^2)$ — what was exponential growth in terms of dimension is now linear.[2]

Apart from the potentially huge memory cost reductions, tensors in TT format, when defined on Chebyshev points, can be evaluated very efficiently.

### 2.4.2 Approximation with Tensors in TT format

Say we want to build a tensor $\mathcal{T}$ for a function $f$. If the dimension $d$ is high, evaluating $f$ on the whole grid becomes impractical due to the curse of dimensionality. If we can approximate $\mathcal{T}$ with a TT tensor $\mathcal{X}$ (much cheaper to store), $\mathcal{X}$ may be used as proxy for $f$. This is what the Tensor Extension Algorithms do (see [5]).

There are three algorithms covered in [5]. The one we use is the *Sample Adaptive Algorithm*. This builds on the other two.

We start under the assumption that the dimension of $\mathcal{T}$ is too great to evaluate the whole grid with $f$. Therefore, we restrict our attention to a sub-grid $\mathcal{K}$. At each iteration of the algorithm, a candidate $\mathcal{X}$ in TT format is compared to the tensor $\mathcal{T}$ on $K$. If the error of approximation is low enough, the algorithm stops. In summary, the information of $\mathcal{T}$ on $\mathcal{K}$ is used to generate a Tensor $\mathcal{X}$ in TT format that is a proxy to $\mathcal{T}$.

If the fixed sub-grid $\mathcal{K}$ does not yield the needed accuracy, the algorithm increases the size of $\mathcal{K}$. The algorithm stops when either a suitable $\mathcal{X}$ is found or when a pre-established limit on the size of $\mathcal{K}$ is reached.

Note that the algorithm gives no guarantee that a suitable $\mathcal{X}$ is found. However, if the function $f$ is well behaved, one expects good results. The results presented in section 4

[2]The value $r$ is the size of the matrices.

can be taken as empirical evidence that for some pricing functions $f$, these algorithms can find suitable tensors $\mathcal{X}$ to work with. Further evidence of this is presented in [5].

# 3 Computing dynamic sensitivities with Chebyshev Tensors

Consider a Risk Factor Evolution Model (RFEM) that generates risk factors in a MC simulation. Take, for example, the Hull-White (HW) one-factor model

$$dr_t = a(b - r_t)dt + \sigma dW_t \tag{1}$$

It has parameters $\theta = (a, b, \sigma)$ and one stochastic variable $W_t$. Define the *model space* as the space spanned by the short rate $r$. Let the dimension of the model space be $k$. In this example $k = 1$. For a two-factor HW model, $k = 2$. For most models, the dimension is the same as the number of stochastic variables. In the context of MC simulations for XVA or IMM, $k$ tends to be small.

Once the parameters $\theta$ of the RFEM are calibrated, they remain fixed throughout the simulation. At every node of the simulation, the model space variables (e.g. short rate $r$) fully determine the market risk factors (e.g. a full swap rate curve). We call the space of market risk factors the *market space*, which includes things like interest rates curves, spreads, volatility surfaces, etc. The market space has high dimension — sometimes in the hundreds. Denote the dimension of the market space by $n$.

Let $g$ be the function that generates market risk factors from model space variables

$$\underset{\mathbb{R}^k}{\textbf{Model Space}} \xrightarrow{\quad g \quad} \underset{\mathbb{R}^n}{\textbf{Market Space}} \tag{2}$$

Functions like $g$ are often analytic — hence ideal for Chebyshev Tensors.

The following example shows how Chebyshev Tensors can be built to compute sensitivities within a MC simulation. A Foreign Exchange Swap is used as an example, which is also used to generate results presented in Section 5. The method is generic enough that it applies to any other trade type.

### Example

Let the pricing function of a FX Swap be $f$. We want sensitivities of $f$ with respect to swap rates (two different currencies) and the exchange rate. Say there are $n$ risk factors in

total.

Each sensitivity — as a function of market risk factors — has dimension $n$, where $n$ is typically large enough so that a single tensor cannot be built due to the curse of dimensionality. The dimension of the problem must be reduced. The following approach is the one we propose.

Consider a single time point within the MC simulation and the $i$-th swap rate be $s_i$. Define the following function $\varphi$

$$\mathbb{R}^k \xrightarrow{\widetilde{g}} \mathbb{R}^n \xrightarrow{S_i} \mathbb{R}, \qquad \mathbb{R}^k \xrightarrow{\varphi} \mathbb{R}$$

where $S_i$ denotes the partial derivative of $f$ with respect to $s_i$

$$S_i = \frac{\partial f}{\partial s_i}.$$

Note that $\widetilde{g}$ is the result of putting together the parametrisations $g$ (e.g. Equation 2) of the RFEMs used to diffuse the market risk factors that correspond to the FX Swap. In this example there are five models: a 2-factor model per yield curve (2 currencies, 2 curves per currency), and a one-factor model for the exchange rate. This gives a total of $k = 9$ dimensions.

The function $\varphi$ is the composition of analytic functions. Therefore, Chebyshev Tensors approximate them quasi-exponentially as the size of the grid increases in each dimension. By definition, $\varphi$ gives the value of the partial derivative of $f$ with respect to $s_i$ at each node of the simulation. This means that the Chebyshev Tensor directly approximates the sensitivity.

To build a Chebyshev Tensor for $\varphi$ do the following. Take the minimum and maximum value of each of the model space variables at the time point in question of the MC simulation. These values determine the hyper-rectangle to which $\varphi$ is restricted. Notice, the hyper-rectangle just mentioned is contained in $\mathbb{R}^k$. Next, build a Chebyshev grid on this hyper-rectangle.

Once the hyper-rectangle in $\mathbb{R}^k$ is ready, one must decide how to build the Chebyshev Tensor: either directly, by evaluating each grid point, or by using the Tensor Extension Algorithms (Section 2.4). If the dimension $k$ of $\varphi$ is between 1 and 4, one can build a Chebyshev Tensor directly. For example, if the trade is an Interest Rate Swap, where each yield curve can be modelled with 1 or 2 factor models, yielding tensors of 2 or 4 dimensions. If $k$ is greater than 4, one ought to consider the use of the Tensor Extension Algorithms

*Remark* 3.1. The description above corresponds to Chebyshev Tensors built for each market risk factor at each time point. However, one can also consider the time dimension in the construction of the Chebyshev Tensor. In this case, the domain of $\varphi$ increases in dimension by one. This may be a suitable thing to do in some cases. For example, if the trade has payments as it matures, one should consider the discontinuities that arise, as explained in Section 2.3. With many discontinuities there are many tensors to build and perhaps the time point approach is more direct. With no discontinuities, one can build a single tensor per risk factor, considerably increasing computational gains. This is the case, for example, of the Spread Option for which we show results in Section 4.

*Remark* 3.2. The focus of this article is the dynamic simulation of sensitives and subsequent DIM. However, the same technique can be applied to pricing functions $f$ instead of sensitivity functions $S_i$. This generates Chebyshev Tensors that efficiently compute portfolio price simulations for the standard XVA or IMM MC simulations.

# 4 Numerical Tests

Dynamic sensitivities were computed using Chebyshev Tensors — built using the Tensor Extension Algorithms presented in Section 2.4 — in two MC simulations each with $10,000$ paths and 11 time points into the future, covering the full lifespan of the trades. One for a FX Swap; the other for a European Spread Option. The objective was to use the sensitivities to compute IM at each scenario of the MC simulation and DIM profiles.[3]

The accuracy of the technique is measured with respect to the benchmark. The latter is obtained by computing dynamic sensitivities using the original pricing functions. In the case of the FX Swap through finite differences; in the case of the Spread Option, the pricing function is based on MC simulations, which returns prices and sensitivities at the same time. The time taken using Chebyshev Tensors is measured and compared to the time taken to perform the benchmark calculation.

DIM profiles should be computed at netting set level. Our tests only considered two trades. The assumption was that each constituted its own netting set. The technique employed naturally extends to a netting set with multiple trades by applying it trade by trade.

The tests were done in MATLAB, on a standard laptop with i7 cores. Calculations were parallelised whenever possible, in particular for the benchmark calculations, given their high computational cost.

[3]Note that the version of IM computed is SIMM, the one proposed by ISDA, and by now a standard in the industry for uncleared derivative transactions.

### 4.1 FX Swap

The FX Swap was between USD and EUR, at-the-money and had 5 years to maturity. The anayltic pricing routine used was the one implemented in the swapbyzero function in MATLAB.

The ISDA risk factors affecting the FX Swap consist of a collection of swap rates in two currencies and the exchange rate. Each currency has two yield curves; each curve diffused using a 2-factor Gaussian model. The exchange rate was diffused using Geometric Brownian Motion.

FX Swaps have payment dates. These create discontinuities in the pricing function along the time dimension. Therefore, Chebyshev Tensors were built per sensitivity and per time point in the simulation (see 3).

Given the RFEMs used, the tensors had dimension 9. The grid built had $262,144$ points; four points per dimension. As this takes a long time to build directly, we used the Sample Adaptive Algorithm (Section 2.4.2).

The maximum number of evaluations for the Sample Adaptive Algorithm was set at $1,000$. Each run started with 350 points. In most cases, 350 points were enough to reach an accuracy of $5e^{-3}$ and the algorithm never needed more than 500. The time taken for the algorithm to run varied, but never beyond 1 minute; in some cases just seconds. The average time taken for the Chebyshev Tensors to evaluate all scenarios on a time point of the simulation was of 15.5 seconds. The corresponding time for the benchmark method was $1,100$ seconds (see Section 5 for further details).

Figure 2 shows the errors for the first swap rate (USD forwarding curve) and the errors for the exchange rate. The errors are well under 1%. In fact, the maximum error was always below 1.5% across all risk factors considered (see Table 1).

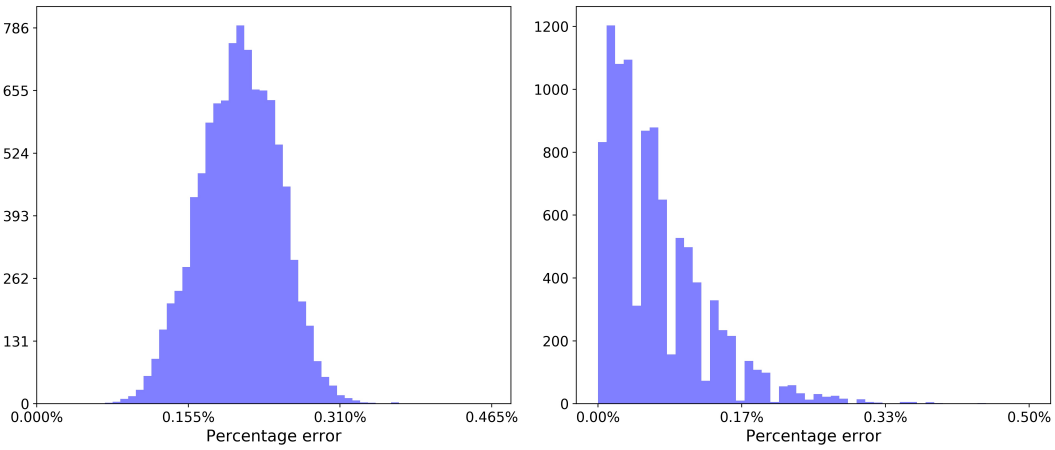


*Figure 2: Percentage relative errors of Chebyshev Tensors. Left histogram shows the first swap rate (USD forwarding curve). Right histogram shows the USD/EUR exchange rate. Both for the 9-th time point of the simulation.*

Figure 3 shows the Expected Initial Margin (EIM) and Potential Future Initial Margin (PFIM) profiles, obtained with the benchmark and with Chebyshev Tensors. The errors are all below 0.34% (see Figure 3 and Table 1). Similar results were obtained for other percentiles.

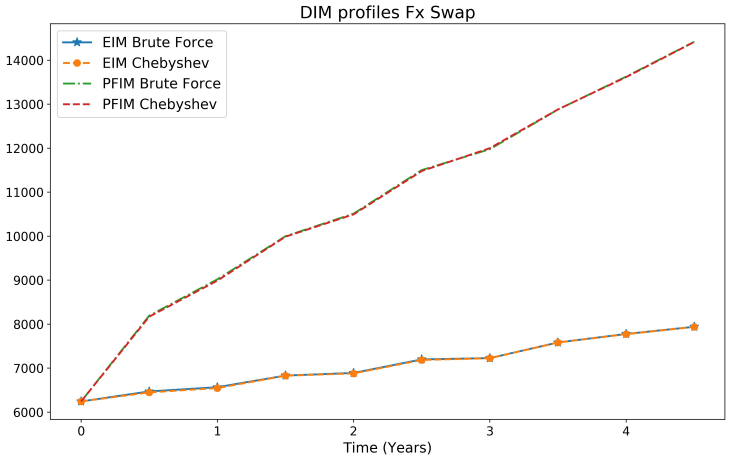


*Figure 3: DIM profiles — expectation and 95% quantiles — for FX Swap, with the benchmark and with Chebyshev Tensors.*

| FX Swap | ISDA sensitivities | EIM | PFIM |
|---|---|---|---|
| Maximum relative error | 1.5% | 0.34% | 0.32% |

*Table 1: Maximum relative percentage errors for market sensitivities, EIM and PFIM (95% quantile) profiles for the FX Swap.*

## 4.2 European Spread Option

The chosen European Spread option had one year to maturity. The pricing routine used was MC based with antithetic variates and $1,000$ paths. The average evaluation time was of 0.5 seconds per sensitivity. Notice this simulation of dynamic sensitivities is an instance of a nested MC simulation.

With $1,000$ paths, the 95% quantile of the noise of spot sensitivities (i.e. the delta) was measured at 8.6% (Figure 4). Reducing the noise by half requires (roughly) increasing the number of paths from $1,000$ to $10,000$. This takes the benchmark computation of dynamic sensitivities, for spot, from 10 hours to several days. The decision was made to stick to $1,000$ paths in the pricing function and work with the fact that Chebyshev Tensors would be accurate to this noise level.

The situation was worse for the sensitivities of the remaining risk factors. For Vega sensitivities, the 95% quantile of the noise was measured at 32.7%; for swap rates it was measured at 11.23%. Therefore, only delta dynamic sensitivities are presented.

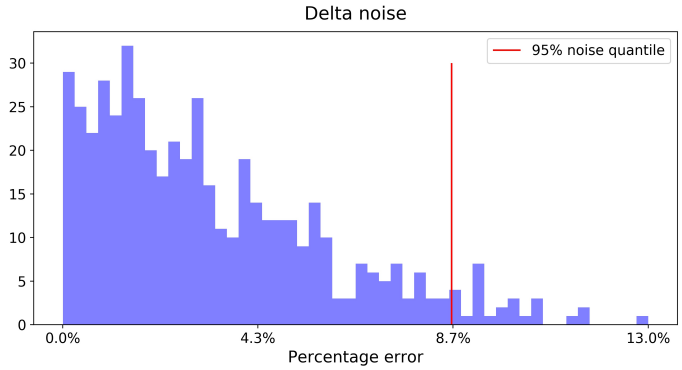


*Figure 4: Noise distribution for the Delta of the Spread Option obtained from the MC-based pricing function.*

If the trade were an American-style option, benchmark computation times would increase by another order of magnitude. Initially, we wanted to test this type of option. However, it was discarded due to the high computational cost for benchmark metrics. As highlighted in Section 2.3, the type of payoff does not hinder the properties of the Cheby-

shev method. With enough computational power, similar results should be obtained for the American version.

No discontinuities are present along the time dimension for this trade. Therefore, the Chebyshev Tensors built included time to maturity as a variable (see Remark 3.1) and hence only one tensor per market risk factor was built.

The Spread Option takes two spot underlyings with a volatility each and a yield curve. The underlyings were diffused using the Heston model, which diffuses both spot and volatility stochastically. The yield curve was diffused using a Hull-White 1-factor model. The models give a total of 5 dimensions. Adding time to maturity gives tensors of dimension 6.

A total of $46,656$ grid points were used (six points per dimension). Once again, the Sample Adaptive Algorithm was used to obtain a Chebyshev Tensor in TT format that approximates the sensitivities. Only $12,000$ random grid points were evaluated. After a 2-3 minutes, the algorithm typically reached an error of $5e^{-3}$; comparable to the noise level of the pricing function. Once built, the Chebyshev Tensors took an average of 11.1 seconds to evaluate the scenarios on each time point. The corresponding time for the benchmark approach was measured at $5,000$ seconds. Section 5 discusses computational savings.

Figure 5 shows relative error distributions of the Chebyshev Tensor built to compute dynamic sensitivities for one of the spot underlyings at a time point of the simulation. Note the error measured is within the noise of the Delta function (Figure 4). Similar results were obtained for other percentiles.

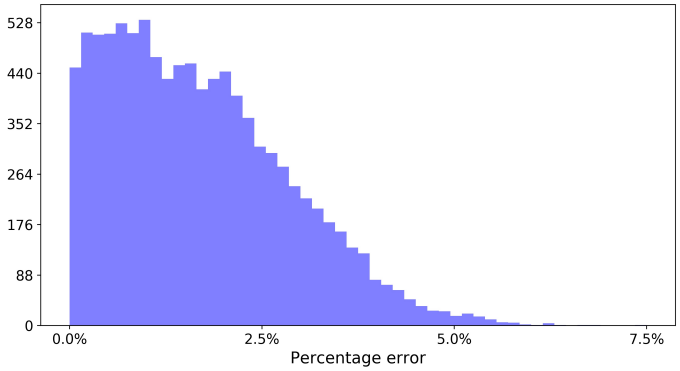


*Figure 5: Percentage relative errors of Chebyshev Tensors for the first spot. Histogram corresponds to the sixth time point in the simulation.*

Figure 6 shows equity Delta Margin profiles, as defined by SIMM, at expectation level (EIM) and 95% quantiles (PFIM) obtained with the benchmark and with Chebyshev Tensors.

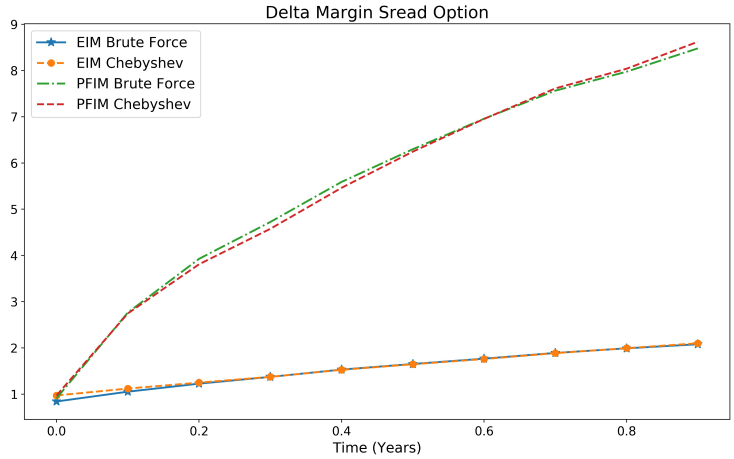


*Figure 6: Equity Delta Margin profiles — expectation and 95% quantiles — for European Spread Option, with the benchmark and with Chebyshev Tensors.*

The maximum error for market sensitivities was 10.7%. However, the vast majority of the errors were below 5%. The maximum errors at the level of Delta Margin profiles were 8.1% and 3.1%, for EIM and PFIM, respectively (Table 2). Given the noise presented in Figure 4, the main source of error in the EIM and PFIM seems to come from the noise of the pricing function.

| **Spread Option** | **Market sensitivities** | **EIM** | **PFIM** |
|---|---|---|---|
| Maximum relative error | 10.7% | 8.1% | 3.1% |

*Table 2: Maximum relative percentage error for Market sensitivities, EIM and PFIM Delta Margin profiles for the Spread Option. EIM profile corresponds to Expected IM, while PFE to 95% quantile.*

# 5 Discussion of Results

The small approximation errors in the case of the FX Swap (Table 1) were expected. The pricing function is analytic and the trade linear; few Chebyshev points are needed to obtain high degrees of accuracy. This is reflected in both the high accuracy values and the short training times of the Sample Adaptive Algorithm (see Table 3)

These levels of accuracy allow for hedging simulations, stable MVA pricing along with its hedging. Moreover, the fast calculations allow for fast scenario analysis such as portfolio optimisation routines that minimise future IM funding costs.

The computational savings for the FX Swap are summarised in Table 3. Computing benchmark sensitivities, for a single risk factor, on a single time point, requires $20,000$

calls to the pricing function. The Sample Adaptive Algorithm needed between 300 and 500 calls. The training of the algorithm took seconds in most cases. The evaluation of the Chebyshev Tensors on 10,000 scenarios took an average of 15.5 seconds. Compared to the cost of the benchmark calculation, for each risk factor and each time point — measured at 1,100 seconds — the training and evaluation times reported for Chebyshev Tensors are negligible. The computational savings are therefore around 97.5%.

| | **Benchmark evaluations** | **Train & Test evaluations** | **Avg training time** | **Cheb Tensor eval time** | **Comp savings** |
|---|---|---|---|---|---|
| FX Swap | 20,000 | $300 \sim 500$ | $<$ minute | 15.5 secs | 97.5% |
| Spread Option | 110,00 | 12,000 | $2 \sim 3$ minutes | 11.1 secs | $89\% \sim 98.9\%$ |

*Table 3: Computational savings obtained by using Chebyshev Tensors to compute dynamic sensitivities compared to benchmark.*

The Spread Option uses a MC based pricer. The accuracy achieved by the Chebyshev Tensor built to approximate Delta is within the noise reported in Figure 4 (see Figure 5 and Table 2). This translates into similar accuracy levels at the level of Delta Margin profiles (Figure 6, Table 2).

For the Spread Option, computing the benchmark dynamic sensitivities for a single risk factor, required 110,000 calls to the pricing function (10,000 paths, 11 time points). The Sample Adaptive Algorithm only needed 12,000. The training of Sample Adaptive Algorithm took between 2 and 3 minutes per risk factor. The evaluation of the Chebyshev Tensors took on average, at each time point, 11.1 seconds. Given the cost of the benchmark sensitivities calculation — 83.3 minutes per risk factor per time point — the training and evaluation of Chebyshev Tensors is negligible. Therefore, the computational savings are estimated — for 11 time points — at 89.1%. A typical MC simulation consists of 100 time points. Increasing the number of time points in the simulation does not change the building of Chebyshev Tensors — notice that time to maturity is included as a variable. Therefore, for a MC simulation with 100 time points, the computational savings would be of 98.9%.

## 5.1 Pre-trade analysis

Pre-trade analysis, which consists of measuring the impact on CCR metrics of possible incoming trades, has become a much-desired feature of XVA, IMM capital and PFE systems. Given the time constraints of the business and the time it takes to compute sensitivities (or PVs) within MC simulations, this is normally not possible with benchmark pricing functions.

Chebyshev Tensors help accelerate this type of analysis. With 90%+ computational

savings, in some cases it is possible to build them on the fly for each possible incoming trade. Otherwise, one should build a Chebyshev Tensor that includes, within its domain of approximation, all those parameters that differentiate instances of the same trade-type.

For example, say a Spread Option must be incorporated into the netting set. By including the strike as part of the Chebyshev Tensor, one builds an object capable of approximating a wide range of Spread Options. As long as the maturity and strike are within the domain of the tensor, this can be used to obtain the sensitivities of all possible Spread Options within the simulation. Given the speed of evaluation for Chebyshev Tensors, this can be done in a short period of time allowing for adequate pre-trade analysis.

# 6 Conclusion

This paper shows how to combine Chebyshev Tensors with the Tensor Extension Algorithms (Section 2.4), to compute dynamic sensitivities and with these, Dynamic Initial Margin, to a high degree of accuracy and low computational cost.

The technique was applied to an FX Swap and a European Spread Option. The benchmark computation for dynamic sensitivities was obtained by calling the pricing function at each node of the simulation. For the FX Swap the maximum relative error for dynamic sensitivities was 1.5%, while for DIM profiles was 0.34% (Section 4.1). For the Spread Option, maximum relative error for dynamic deltas was 10.7%, mostly due to the numerical noise of the underlying function being approximated. The maximum relative errors for the corresponding Delta Margin profiles were 8.1% and 3.1%.

Computational gains stand at 97.5% for the FX swap and between 89% and 98.9% for the Spread Option (Table 3).

The Chebyshev method presented in Section 3 can be applied to wide range of trade types. The key element is the dimension of the tensor to build. In CCR, most of the models used have a dimension that the Tensor Extension Algorithms can handle. Problems can appear in cases such as Basket Options with a large number of underlyings modelled independently. However, for a typical portfolio, we expect the technique to apply for the vast majority of live and newly incoming trades.

### Final note

We would like to thank the reviewers for their feedback and suggestions.

# References

[1] ISDA. Methodology, version R1.3. (Effective Date: April 1, 2017) //www2.isda.org/attachment/OTIzMQ==/ISDA%20SIMM%20vR1.3%20(PUBLIC).pdf.

[2] Trefethen, L. *Approximation Theory and Approximation Practice*. SIAM, 2013.

[3] Gaß, M., Glau, K., Mahlstedt, M., Mair, M. Chebyshev Interpolation for Parametric Option Pricing. *Finance Stoch* (2018) 22: 701. https://doi.org/10.1007/s00780-018-0361-y.

[4] Glau, K., Pachon, R., Pötz, C. Fast Calculation of Credit Exposures for Barrier and Bermudan options using Chebyshev (2019) interpolation. *https://arxiv.org/abs/1905.00238*

[5] Glau, K. Kressner, F. Statti, F. Low-rank tensor approximation for Chebyshev interpolation in parametric option pricing. (2019) *https://arxiv.org/abs/1902.04367*

[6] Zeron-Medina, M., Ruiz, I. Chebyshev Methods for Ultra-efficient Risk Calculations. (2018) https://arxiv.org/ftp/arxiv/papers/1805/1805.00898.pdf

[7] Zeron, M., Ruiz, I. Denting the FRTB IMA computational challenge via Orthogonal Chebyshev Sliding Technique.(2019) arXiv:1911.10948

[8] Steinlechner, M. Riemannian optimization for high-dimensional tensor completion. *SIAM J. Sci. Comput.* 38. (2016). S461–S484.