Forward-Backward Finite-Differencing for XVA in small netting sets

Peter Jäckel

16th May 2026

Abstract

We give details of a framework for the computation of *Expected Positive Exposure* profiles and *Credit Value Adjustments* based on finite-differencing methods. The methodology combines the backward and the forward Kolmogorov equation to generate future exposure distributions from which the various risk measures can be inferred. The framework bypasses the usual difficulties associated with Monte Carlo simulation based approaches and, for small to moderate portfolios, is fast and efficient.

1 Introduction

A common starting point for the calculation of *Credit Value Adjustment* (CVA) figures is the formula [ZP07, equation (15)]

$$\mathrm{CVA} \;=\; (1-R)\cdot\int_0^T \mathrm{EPE}(T')\,\mathrm{d}p_{\mathrm{D}}(T') \tag{1.1}$$

where R is an assumed recovery rate, $p_{\mathrm{D}}(T')$ is the T'-forward probability of default to time T', and the *Expected Positive Exposure* $\mathrm{EPE}(T')$ is

$$\mathrm{EPE}(T') \;=\; P(0,T')\cdot\mathsf{E}\left[\max\left(\tilde{v}(T'),0\right)\right] \tag{1.2}$$

with $\tilde{v}(T')$ being the net present value of (all yet to be exchanged cashflows of) the trade (or basket of trades that are netted) observed at (the future) time T', $P(0,T')$ representing the T'-maturity zero coupon bond price at time 0, i.e., today, and the expectation being set in the T'-forward probability measure.

As for a justification of equation (1.1), we point to the fact that it is part of various regulatory prescriptions, and the de-facto core logic in most, if not all, commercially available CVA systems. Also, for most counterparty credit risks, the assumption of negligible causal correlation between international financial markets and a counterparty's idiosyncratic default is perfectly reasonable. This does, however, mean that from hereon we are explicitly excluding any situation that would be classified as *wrong way risk* [Wro].

The integration over the (increments of the) default probability curve $p_{\mathrm{D}}(T')$ in the CVA definition formula (1.1) is in practice carried out by the aid of market-observable, or credit institution estimated, CDS quotes for the respective counterparty from which a default probability function can be computed by standard means. The details as to how we obtain the probability function are ultimately of secondary interest as long as we have easy access to the curve $p_{\mathrm{D}}(\cdot)$, so that we can carry out the integration in (1.1).

The key step of the CVA calculation then lies in the methodology to compute the Expected Positive Exposure function $\mathrm{EPE}(T')$.

2 CVA as an early exercise problem

Expression (1.2) for the expected positive exposure has the exact form of a contract on a sequence of (possibly contingent) cashflows occurring at or after time T' that the contract holder can choose to terminate exactly once, namely at time T'. In other words, it can be seen as a contract on all the cashflows occurring at or after time T' whose net present value (at time T') we denote as $\tilde{v}(T')$, with the addition of the *right to cancel* at T', i.e., with the added option of an early exercise to get out of the deal. This means that, for the correct calculation of $\mathrm{EPE}(T')$, we need to know at time T' if it is optimal to terminate there and then, or not. In general, intuitively, we think of this choice being simple once we have knowledge of $\tilde{v}(T')$, i.e., the actual value of those future (contingent) cashflows.

The only problem is that, depending on our chosen overall valuation framework, we may *or may not* have access to $\tilde{v}(T')$. In a generic Monte Carlo simulation framework, for instance, we only ever observe cashflows along the simulated path, each one divided by the contemporaneously observed numéraire value. This means that, for any one path, at its time horizon T', we can only know the sum of the numéraire-denominated cashflows of *this path*, but not of the net present value of all future cashflows as seen on time T' in the T'-filtration $\mathcal{F}_{T'}$ of this very path. This dilemma is what gives rise to a wealth of literature on *Bermudan* and *American* Monte Carlo methods, e.g., bundling methods [Til93], regression techniques [Car96, LS98], and lower and upper bound methods based on strategies or exercise boundary parametrization optimisations [Jäc02, BG97, And00, JA10]. For simple products in a model setting in which we can know analytically the value $\tilde{v}(T')$ given any filtration $\mathcal{F}_{T'}$, we can resort to the full evaluation of the product, incurring the respective significant calculation time overhead, and this is how various commercial software providers of CVA systems operate. In general, for non-linear products, or in a model setting in which we do not have analytical access to the value $\tilde{v}(T')$, when in a Monte Carlo framework, we are forced to use early exercise valuation methods. This works as far as $\mathrm{EPE}(T')$ is concerned, and we can repeat the calculation with a different exercise objective if we wish to have the expected negative exposure $\mathrm{ENE}(T')$. And then we have to repeat the early exercise optimisation (or regression, etc.) once for each time horizon T' of interest. When we wish to generate entire future distributions for subsequent quantile level curve extraction or for other, more detailed analysis, we need to use further, more sophisticated, techniques since the brute force approach of regressions (or early exercise boundary optimisations) for numerously many potential future value levels won't cut it anymore. Still, this works in practice, but unsurprisingly involves some considerable calculation effort. Consequently, many CVA system implementations employ sizeable teams of quantitative analysts, and even larger pools of IT resources.

In contrast, in a backward induction setting, i.e., in a finite-differencing framework discretised to a number of spatial nodes $\boldsymbol{x}_i$, $i = 1 \dots N$, for the market risk factor vector $\boldsymbol{x}$ (also known as *state vector*), we roll back the sum of expected values of numéraire-denominated future cashflows. This gives us access to the value $\tilde{v}(T')$ on each and every of the N discretisation nodes of our lattice, on which the filtration $\mathcal{F}_{T'}$ is fully characterised by the state vector $\boldsymbol{x}_i$. Denoting $n(t)$ as the value of the chosen numéraire, and defining the numéraire-scaled value $v(t)$ as

$$v(\boldsymbol{x}; t) \quad := \quad \tilde{v}(\boldsymbol{x}; t) \cdot \frac{n(0)}{n(\boldsymbol{x}; t)} \tag{2.1}$$

we have by virtue of the fundamental theorem of asset pricing

$$v(\boldsymbol{x}_i; T') \quad = \quad \mathsf{E}^{\mathcal{M}(n(\cdot))}\Big[v(T') \Big| \boldsymbol{x}(T') = \boldsymbol{x}_i \Big] \quad = \quad v_i(T') \tag{2.2}$$

where $v_i(T')$ is the value of v on node $\#i$ at time T', and the expectation is taken in the probability measure induced by the numéraire $n(\cdot)$.

The disadvantage of the finite-differencing approach is that it does not readily permit extension to an arbitrary number of market risk factors. For a book of netted cross-currency swaps, perhaps the most we can extend this to is one base currency interest rate, three foreign currency interest rates, and the associated three FX rates against the common base, making it seven factors and requiring a seven-dimensional finite-differencing engine[1]. Still, for an institution with a small or moderately sized derivatives book in which only a manageable number of trades with little netting require active CVA computations, finite-differencing can be the preferred method of choice. This is particularly pertinent when considering the, arguably, stupendous operational and financial expenses involved in the setting up and running of the sizeable XVA development and calculation teams when large scale Monte-Carlo engines are implemented or used in third party XVA calculation systems. For many small to medium size companies that hold derivatives books, a small-scale (netting set size) solution based on finite-differencing methods is perfectly viable!

3 Future value distributions via finite-differencing

In order to compute any exposure measure for a given time horizon T' in a finite-differencing framework, we require two components that have to be computed separately. First, in a setting of any permissible probability measure, we need the numéraire-denominated value of all (contingent) cashflows occurring on or after T' conditioned on any one of the spatial discretisation nodes $\boldsymbol{x}_i$, which we above denoted as $v_i(T')$. Second, we need the risk-neutral probability that node $\#i$ is attained in the chosen probability measure, which we shall denote as $p_i(T')$. In a continuous setting, in between cashflows, the equation governing the value function $v(\boldsymbol{x};t)$ is

$$\frac{\partial}{\partial t}v \;=\; -L\cdot v \tag{3.1}$$

and known as the *backward Kolmogorov equation*. The operator L is the infinitesimal generator of the dynamics and specific to the chosen model and contains drift terms adapted to the probability measure induced by the chosen numéraire. In a finite-differencing implementation, we populate the numéraire-scaled value function v on the lattice nodes on the last point in the time line T by the then occurring cashflows $c(\boldsymbol{x};T)$, weighted by the numéraire value ratio $n(0)/n(\boldsymbol{x};t)$ according to

$$v_i(T) \;:=\; c(\boldsymbol{x}_i\,;T)\cdot\frac{n(0)}{n(\boldsymbol{x}_i\,;T)} \tag{3.2}$$

and roll backwards in time according to (3.1). We skip the discussion of the underlying backward induction numerical analysis itself as we consider it standard. Let us just remember that in our enumeration of spatial nodes i, we are not, in principle, restricting ourselves to any dimensionality of the governing financial pricing model: i is an enumeration index across all the nodes in the multi-dimensional[2] lattice (or whatever spatial node geometry may be in use in the induction method [Jäc17, Jäc18]).

Since we later wish to compute XVA figures that depend on the value distribution for multiple future time horizons T_ℓ, the backward induction implementation needs to be configured to preserve in memory a full value lattice at the pre-selected distribution recording times $\{T_\ell\}$. One may fear that this poses a memory limitation issue, but we can assure the reader that with contemporary computer hardware, this is not even remotely a problem, though, we do of course want to make use of 64 bit computer architecture

[1] With modern CPUs and efficient multi-threading, perhaps we can even go to 5 interest rate and 4 FX dimensions.

[2] Realistic valuations are readily possible in up to 3 currencies against USD in a cross-currency standard extended Vasicek / Hull-White model with one interest rate in each currency, plus 3 FX rates against USD, making this 7-dimensional — and this even on standard desktop hardware originating from 2015!

since the 2Gb limit of 32 bit hardware is hardly sufficient for multiple currencies and multiple product legs (e.g., when each trade in a netting set is itself implemented as a virtual product leg for detailed analysis purposes).

Having started the backward induction at the final payment time, we iterate backwards in time as per usual, adding further cashflow values into each node on each payment date T_ℓ:

$$v_i(T_\ell) \;\; += \;\; c(\boldsymbol{x}_i\,;T_\ell)\cdot\frac{n(0)}{n(\boldsymbol{x}_i\,;T_\ell)}\;. \tag{3.3}$$

This works fine for any cashflow that is paid at time T_ℓ and whose amount is fully determined in the filtration $\mathcal{F}_{T_\ell}$ at time T_ℓ and does not depend on any information from earlier times (making it path-dependent), so, in practice, it is perfect for fixed cashflows and those that depend only on financial observables visible at time T_ℓ such as a spot FX rate for currency conversion etc. In reality, however, many financial cashflows have some sort of path dependency where the payment amount to be paid at some time T depends on one ore more *earlier* observed fixings, and we will discuss that aspect in section 3.2. Notwithstanding those intricacies, in principle, we continue the backward induction until we have arrived at $T_0 = 0$.

In contrast to the value distributions (across spatial nodes) that we obtain from the backward valuation sweep, we obtain the distribution of the probability function $p(t)$, by virtue of the *forward Kolmogorov equation*

$$\tfrac{\partial}{\partial t}p \;\; = \;\; L^*\cdot p\;, \tag{3.4}$$

also known as the *Fokker-Planck* equation, and L^* is the adjoint of L. In a numerical finite-differencing implementation, we simply deposit a unit amount of probability in the central node of the lattice at time $t = 0$, and roll forward in time to any desired time horizon T', and record the probability distribution at all desired recording times $\{T_\ell\}$ by preserving a copy of the probability value lattice in memory at each such time. The forward sweep is comparatively easy for the following reasons:-

- All coefficients and linear algebra logic required for the forward induction is already available since it is identical to that needed for the backward induction, with the sole difference being that the so-called *propagator matrix* for the forward induction is given by the *adjoint*, i.e., the *transpose* of the propagator matrix of the backward induction.
- The forward induction needs to be applied only to a scalar value whereas in the backward induction we usually need to roll back several product legs simultaneously to be able to compute the effective trade value from their combination.
- The forward induction is seeded only at $t = 0$ and requires no intermediate (cashflow) injections or payout evaluations[3].

For completeness, we mention that the generated probability distributions, stored as one probability value for each finite-differencing node $\#i$ for each recording time T_ℓ, i.e., $p_{i\ell} = p_i(T_\ell)$, depend on the chosen numéraire! They are not to be mistaken with Arrow-Debreu security prices, though, the relationship is simple: the value of an Arrow-Debreu security for model state $\#i$ at time T_ℓ, i.e., a contract that pays one currency unit if that state is attained at time T_ℓ, and nothing in any other state at that time, is

$$\mathrm{AD}_i(T_\ell) \;\; = \;\; p_{i\ell}\cdot\frac{n(0)}{n(\boldsymbol{x}_i;T_\ell)}\;. \tag{3.5}$$

[3] though some allowance for contingency considerations is required for derivatives with barrier features

For operational reasons explained later, it is important that we carry out the single forward induction sweep to generate and store the node-wise probabilities p_i for each of the recording times T_ℓ *before* the backward induction (valuation) stage. More on this in section 3.2.

It is worth noting that both the backward and the forward induction stage are to be performed, in principle, once as a single sweep throughout all time horizons, storing the values v_i and p_i on all time horizons as we come across them. The final evaluation of the expected positive exposure then becomes the simple calculation

$$\mathrm{EPE}(T') \quad = \quad \sum_{i=1}^{N} \max\left(v_i(T'), 0\right) \cdot p_i(T') \; . \tag{3.6}$$

For this purpose, and this needs to be absolutely clear in the following, our algorithm must store a complete set of values (i.e., net present values), one for each spatial node, and a complete set of probabilities, one for each spatial node, at each of the designated distribution recording times $\{T_\ell\}$. To be clear: that's two separate stored lists for each T_ℓ: one for probabilities, and one for the financial values, where each financial value itself may of course be comprised of multiple *leg values*.

For other exposure measures beyond EPE, we typically wish to have a full net-present-valued distribution for any time horizon T_ℓ. This is equally straightforward since for each time slice T_ℓ, we earlier obtained a pair of values (p_i, v_i) of probability and trade value for each node, the probability from the forward sweep with the forward Kolmogorov equation, and the trade value from the backward induction stage with the backward Kolmogorov equation. This set of value-probability pairs forms a discretised distribution, with a special case being $t = 0$ where the distribution collapses to the single pair $(p = 1, v = v_0)$ with v_0 being today's net present value of the contract. For all later time horizons, by standard ranking and cumulative probability interpolation techniques, we can form spatially continuous effective (numéraire-scaled) trade value probability functions $p(v)$ which in turn can be queried for quantile levels if Potential Future Exposure calculations or Value at Risk assessments are desired. Since we have a full distribution for each time slice T_ℓ, these quantities can also be represented as curves over time. The only practical limit of the possible analyses is the patience, or perhaps the endurance, of the human analyst. As a general Caveat we mention, however, that all risk measures that are not in the form of an expectation depend on the chosen probability measure and should therefore be used with a clear understanding of the meaning of the figures. Naturally, it is in principle possible to translate the discrete distribution given by the set $\{(p_i, v_i)\}$ to other probability measures, and generate quantile levels for those, if desired, by judicious multiplication and division with and by the original and the target numéraire value for each lattice node.

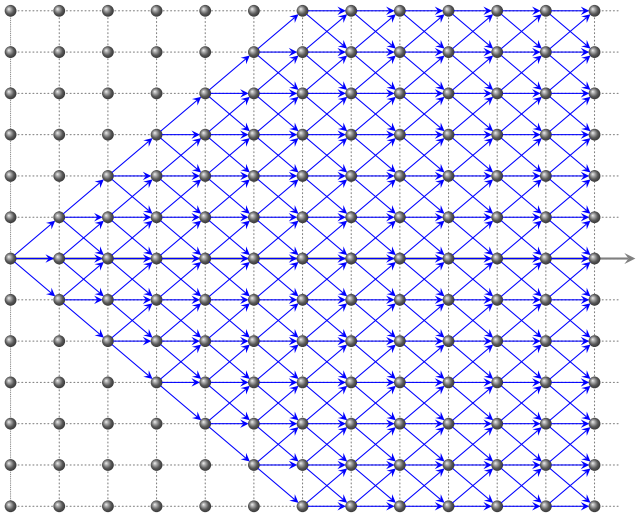
3.1 A comparison with Monte Carlo simulations

In order to compare this approach with the concept of *simulated paths* in a Monte Carlo framework, where each path typically is assigned the same weight of $1/m$ when m paths are simulated, consider that any finite-differencing time-stepping scheme represents a set of transition probabilities at time t_j from each of the nodes at that time slice to a set of attainable nodes at the neighbouring time slice $t_{j\pm1}$ (with ±1 for forward/backward induction). The matrix of these transition probabilities is the above mentioned *propagator* matrix. The number of nodes that any node at t_j connnects to at $t_{j\pm1}$ depends on the integration scheme, and the used so-called *finite-differencing stencils*. In a standard high-efficiency fully explicit (1st order) scheme with the "Linear Gaussian Markov" model[4] (assuming zero correlation between interest rates and

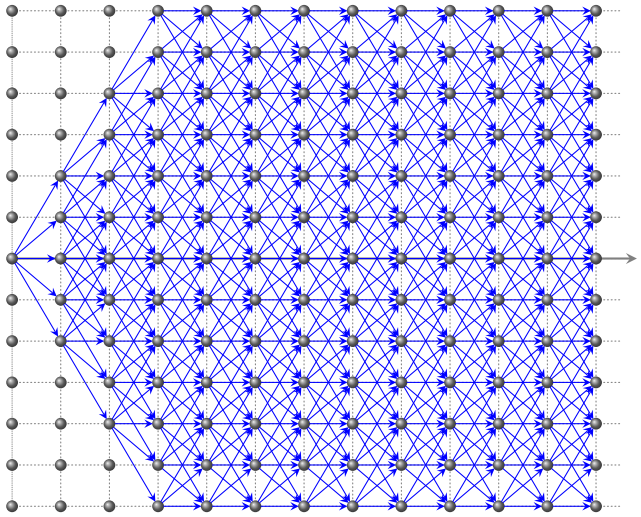
[4] also known as Hull-White model or extended Vasicek model, though we use 'LGM' to include its extension to multiple currencies with one FX spot factor for each currency against the base currency USD.

FX) for a cross-currency swap, for instance, this would mean each node at t_j connects to 7 nodes at $t_{j\pm1}$ (and more when correlation is added). In a fully implicit scheme, as we would usually use it for a one-dimensional local-volatility model, for instance, *each node connects to each node*, meaning a full N connections (ignoring boundary conditions for simplicity). Denoting the connection count as K, the forward-backward finite-differencing approach over n time steps is equivalent to the simulation of, approximately, as many as $\approx K^n$ simulation paths (though in reality somewhat fewer if we take boundary conditions into account). To put this into perspective we show below a few examples of path counts for a 3 year trade evaluated with weekly time steps, i.e., $n = 156$, together with a schematic of the path construction logic in some of the mentioned schemes where we assumed 11 nodes per dimension, except for the fully implicit local volatility case, and we ignored the effect of absorbing boundary conditions.

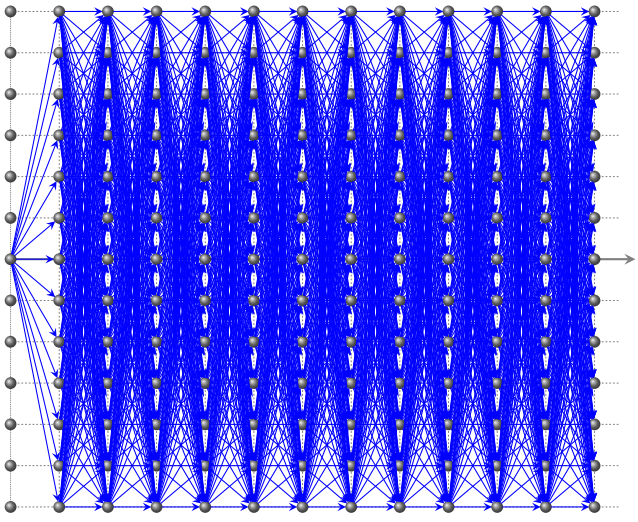
Product	Model	Integration scheme	K	K^n (n=156)	Actual path count
Vanilla IRS	1D LGM	Explicit (1st order)	3	$2.7 \cdot 10^{74}$	$9.5 \cdot 10^{72}$
Vanilla IRS	1D LGM	Explicit (2nd order)	5	$1.1 \cdot 10^{109}$	$7.0 \cdot 10^{104}$
XCCY IRS	3D LGM	Explicit (1st order)	7	$6.8 \cdot 10^{131}$	$1.4 \cdot 10^{130}$
XCCY IRS	3D LGM	Explicit (2nd order)	25	$1.2 \cdot 10^{218}$	$3.2 \cdot 10^{213}$
FX Barrier	1D Local Volatility	Implicit	101	$4.7 \cdot 10^{312}$	$4.7 \cdot 10^{312}$



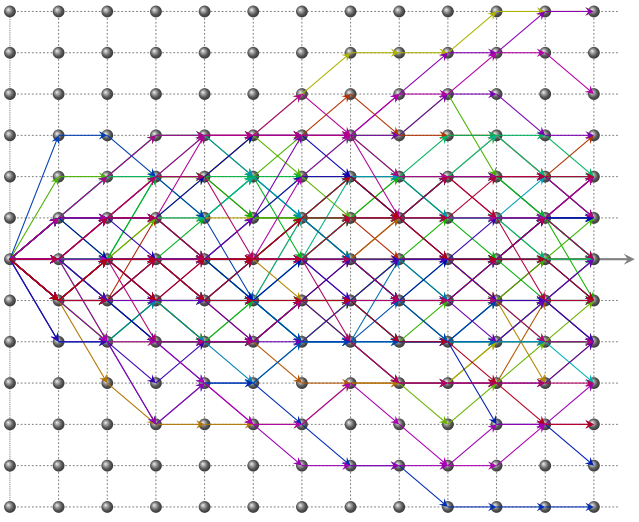
1D Explicit (1st order)



1D Explicit (2nd order)



1D Implicit



Monte Carlo simulation

This should be compared with the typical use of as few as 10,000 $(=10^5)$ $(=10^5)$ simulation paths as usually done with typical commercial software for XVA purposes. The difference in orders of magnitude of the effective path count is clearly staggering. This can perhaps be understood if we view the forward-backward induction methodology as a

recombining ensemble simulation

where many paths simultaneously inherit contributions from many paths at the previous time step, in contrast to the standard Monte Carlo simulation method where all paths are evaluated entirely separately. The important point is that, as a net result of the *ensemble* behaviour of the forward-backward induction method, the sole source of residual numerical accuracy is due to the spatial discretisation itself, and this is borne out in the accuracy of the valuation of the product as seen out of today ($t = 0$).

The main difference in the result of a Monte Carlo simulation is that the generated value distribution typically gives us one distinct value per path (since one usually does not use spatial discretisations within Monte Carlo frameworks), i.e., typically tens of thousands of different values[5], whereas the granularity of the distribution generated by a lattice method is limited to the total number of nodes. In practice, however, for XVA calculations, there is no issue if we have only a few hundred of value levels in our distribution in a one-dimensional case, and when we have more than one dimension, the number of spatial nodes will always be in the many thousands, or millions, or more, anyway.

3.2 Cashflows depending on past fixings

The direct roll-back-and-add-in-cashflows-along-the-way procedure we described above works perfectly as long as any payment at a given time t' depends in a Markovian sense only on the state vector $\boldsymbol{x}(t')$, and not on anything that happened *earlier*. This is the true nature of the *backward induction* logic: at any time t' it has easy access to expectations of future cashflows, and is perfectly conditioned on the precise state of the world at t', but has no recourse whatsoever to the past. In the context of CVA calculations of, for example, interest rate swaps, however, we often need to make a payment of an IBOR rate (aka "term rate") observed at time t at the end of its accrual period, say $t + \tau$ (or indeed of a compounded rate that depends on daily observations from t to $t + \tau$, though, to simplify the discussion we will limit ourselves to the observation of forward looking term rates such as IBORs for the time being). Hence, when we condition on one lattice node at the payment time $t' := t + \tau$, *we don't know what the IBOR rate was at the earlier time t that determines the coupon that is being paid in the filtration of this lattice node.* This issue is not unique to floating rate payments, though, and requires generic handling.

To further illustrate the problem, we show in figure 1 an example with 4 observation points out of all of which payments are *forward-scheduled* to be made at later points on the time line. Two of the observation points are in fact in the past, but give rise to payments in the future, such as would be the case with a recent IBOR fixing whose period end date and payment date is still in the future. In this example, as we start the initial backward induction stage from the very last finite-differencing time line point, we cannot pick up the value of the cashflow occurring then since we do not know what the payment amount should be. We have the same situation at all of the payment dates at the end of the five shown *forward scheduling arcs* in figures 1 and 2.

In order to cater for these past-observation-dependent payments, instead of a single, trivial, complete backward induction sweep, we use a sequence of nested backward-forward (sub-)iterations. As we will see,

[5] this is for *linear* products; obviously, for options that may pay out 0 there can be many paths leading to 0 value

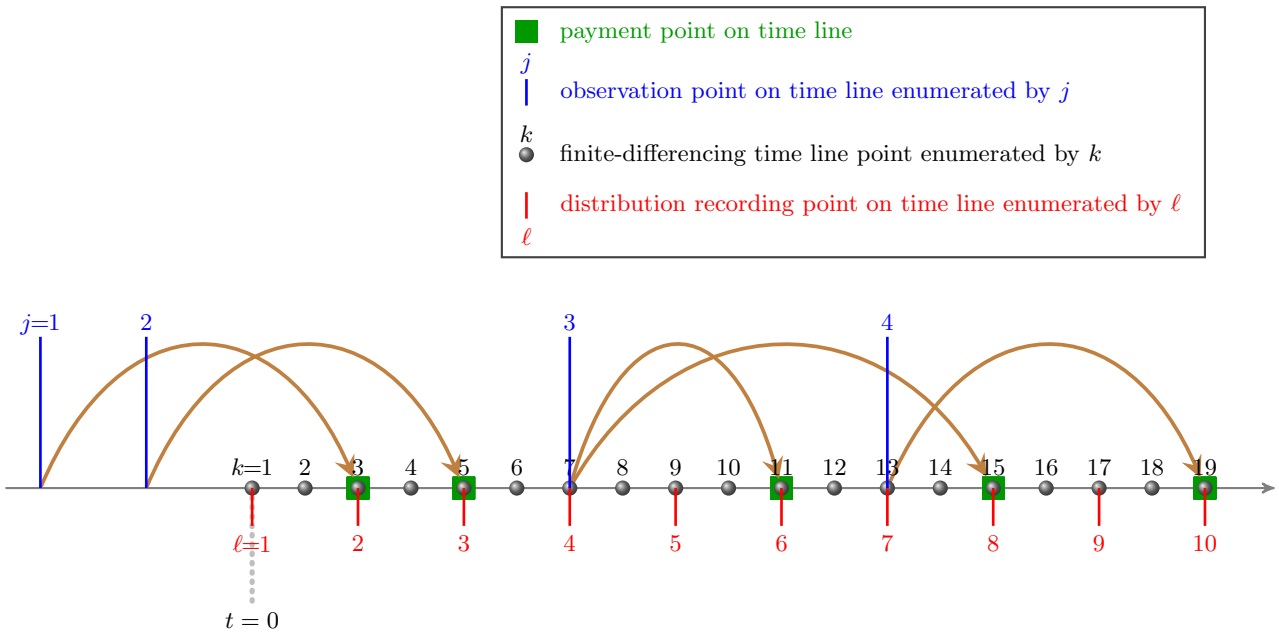


FIGURE 1: A time line with *forward-scheduled* payments.

the required forward induction sub-iterations will necessitate knowledge of the probability for each spatial node at the observation times, and this is why *we need the probabilities first.*

We assume in the following that the time line is discretised for the finite-differencing algorithm such that all payment dates, observation dates (such as IBOR fixing dates), and, of course, all desired distribution recording points in time are part of the finite-differencing time line. We also assume that all payment and observation dates are included in the set of distribution recording points. We will enumerate the observation points by j, the finite-differencing points on the time line by k, and the set of distribution recording points by ℓ. Note that observation points can be in the past (in which case they merely amount to past fixings of financial observables) but finite-differencing and distribution recording points start at $t = 0$ and extend only into the future.

We now start the description of the algorithm that accommodates the exposure to cashflows occurring on or after a specific distribution recording point which depend on an observation prior to the respective recording point. In an outer iteration, starting at the very last point on the finite-differencing time line, we roll back to the nearest earlier observation date, picking up any cashflows that are known within their respective node filtration such as fixed coupon payments, for instance. Every time we have arrived at any observation point j, we start an inner iteration which is to loop over all forward scheduled payment points that arise out of this observation point, i.e., essentially a loop over all of the scheduling arcs depicted in figure 1 for any given observation point j. In each inner iteration (distinct by the payment time targeted by the selected cashflow), we isolate the payments that are to be made out to the selected payment point, compute their numéraire-scaled net present value on each and every lattice node as seen at the current observation point j, and store this value in a new lattice value container that serves as work space for this inner iteration. *We then forward-propagate these values from observation point j all the way to its payment point by the aid of the forward Kolmogorov equation* (though more details are to follow on how we do that). As we roll forward and come across (or arrive at) any distribution recording points, we add the forward-rolled value into the respective distribution recording point's node values.

We show in figure 2 a schematic to illustrate this nested procedure. In the shown example, specifically, we have the following steps in order of execution:-

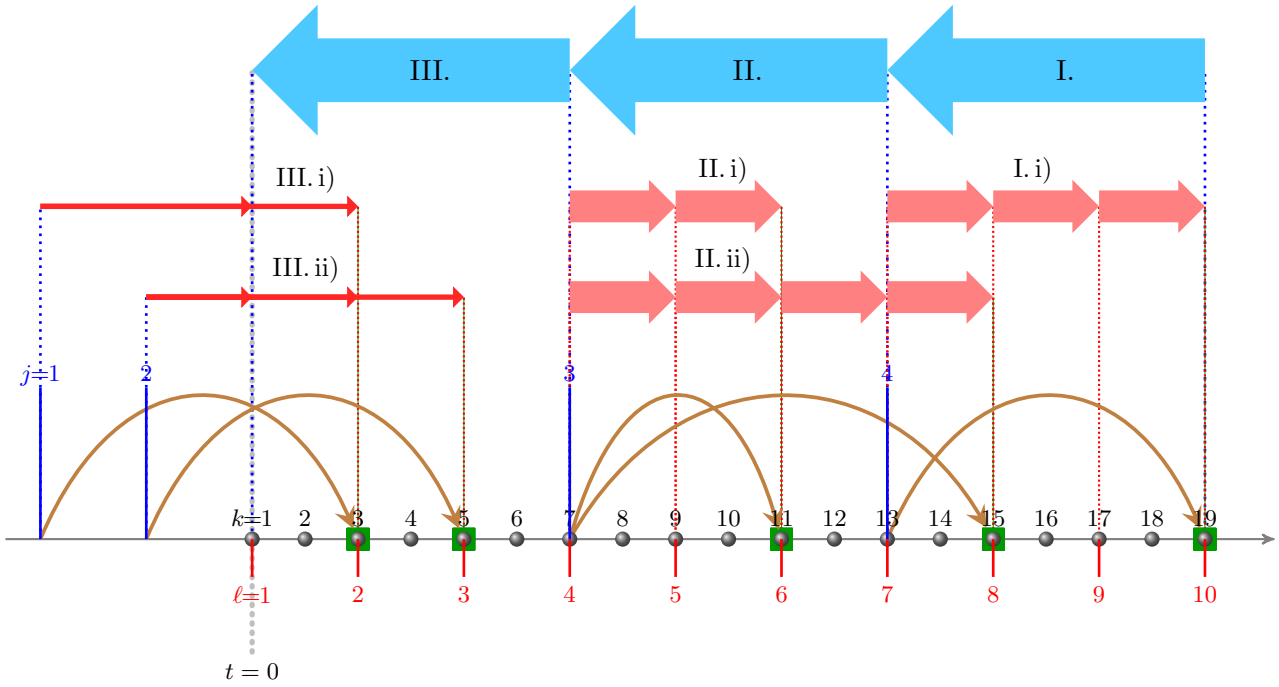


FIGURE 2: Schematic of the nested backward-forward iteration to incorporate *forward-scheduled* payments into later distribution recording points.

I. Roll back from $\ell = 10$ to $\ell = 7$, collecting all cashflow (net present) values that are known in the filtration of any one lattice node such as fixed coupons, if any. Note that *rolling back* refers to the box-standard, well-known numerical backward induction method that does of course depend on the chosen model and spatial discretisation and *representation* rules, iterating backwards over the time discretisation points (k=19 to 1 in figure 2) by the aid of some numerical time-integration scheme such as first order explicit or implicit, Crank-Nicolson, or whatever may have been chosen, see, e.g., [Jäc13]. Whenever we pass a recording point, take a snapshot copy of the lattice values, and store it in memory for that recording point. When we arrive at $\ell = 7$, add numéraire-scaled values of cashflows scheduled out of this time horizon (for payment at $\ell = 10$) to the lattice nodes since they had not been picked up yet.

   i) On each lattice node at $\ell_{\text{from}} = 7$, compute the numéraire-scaled value of *only* the cashflow that is from here scheduled for payment on $\ell_{\text{to}} = 10$. Populate a new lattice container from those numéraire-scaled net present cashflow values and node probabilities via equation (3.8) in section 3.2.1. Propagate those values with the forward Kolmogorov equation to $\ell_{\text{to}} = 10$, i.e., carry out the forward induction from $\ell_{\text{from}} = 7$ to $\ell_{\text{to}} = 10$. Every time we pass (or arrive at) a recording point, add the forward-propagated values after division by their node probabilities according to equation (3.10) into the lattice values stored at that recording point. This concludes the sub-iteration (forward induction) out of the observation point at $j = 4$ which is of course the same as the recording point $\ell = 7$

II. Pick up the main backward induction value lattice that had previously been rolled back to $\ell = 7$. Continue rolling it to $j = 3$ ($\ell = 4$), collecting all cashflow values that are known on their node such as fixed coupons, if any. Whenever we pass a recording point, take a snapshot copy of the lattice values, and store it. When we arrive at $\ell = 4$, add numéraire-scaled values of cashflows scheduled out of this time horizon for payment at $\ell = 6$ and $\ell = 8$ to the lattice nodes since they had not been picked up yet. From $j = 3$ ($\ell = 4$), we have two separate forward induction sub-iterations to carry out since two distinct *forward-scheduled* cashflows have their fixing (observation) at this point in time.

   i) On each node at $\ell = 4$, compute the numéraire-scaled value of *only* the cashflow that is from here scheduled for payment at $\ell = 6$, and populate a new lattice container with those values via equation (3.8). Propagate those values with the forward Kolmogorov equation to $\ell = 6$. Every time we pass (or arrive at) a recording point, add the forward-propagated values into the lattice values stored at that recording point, after division by their node probabilities according to equation (3.10).

ii) Restart at $\ell = 4$, computing on each node the numéraire-scaled value of *only* the cashflow that is from here scheduled for payment at $\ell = 8$, and populate a new lattice container with those values via equation (3.8). Propagate those values with the forward Kolmogorov equation to $\ell = 8$. Every time we pass (or arrive at) a recording point, add the forward-propagated values into the lattice values stored at that recording point, after division by their node probabilities according to equation (3.10).

III. Pick up the main backward induction value lattice that had previously been rolled back to $\ell = 4$. Continue rolling it to $\ell = 1$, collecting all cashflow values that are known on their node such as fixed coupons, if any. Whenever we pass a recording point, take a snapshot copy of the lattice values, and store it.

HANDLING OF PAST FIXINGS FOR FUTURE PAYMENTS

These are all to be treated as if they are cashflows scheduled from the valuation date at $t = 0$ ($\ell = 1$). We use all required historical fixing informations, and, in our example, conduct two forward induction sub-iterations in complete analogy to what we did in step II.

i) Using the historical fixing for past observation time point $j = 1$, compute the cashflow that is from there scheduled for payment in the future at $\ell = 2$, and add its numéraire-scaled net present value to the main backward induction lattice (for $t = 0$). Also populate a new lattice container from those cashflow values and forward-propagate in complete analogy to step II.i) from $\ell = 1$ and $\ell = 2$.

ii) Using the historical fixing for past observation time point $j = 2$, compute the cashflow that is from there scheduled for payment in the future at $\ell = 3$, and add its numéraire-scaled net present value to the main backward induction lattice (for $t = 0$). Also populate a new lattice container from those cashflow values and forward-propagate in complete analogy to step II.ii) from $\ell = 1$ and $\ell = 3$.

3.2.1 The forward propagation of values

The promotion of node values from an earlier time horizon to a later one is, as we use it here, in fact, an approximation. The precise calculation that we are doing is as follows. Consider that for each node $\#i$ at time horizon T_f we are able to compute the net present value of a contingent cashflow as seen out of that node's filtration. Denote the net present value of this cashflow conditioned on node $\#i$ as $\tilde{c}_i$. In general, $\tilde{c}_i$ will of course be a function of the value of the state vector at that node, for instance via a given model's mapping from the state vector to the observation of the IBOR rate on this node at time T_f. Further denote as T_p the later payment time of this cashflow to which we need to promote at least an approximation of its value for subsequent credit exposure calculations. Then, for each (distribution recording) time T_k that is between T_f and T_p, we aim to compute the T_k-expectation of the cashflow that fixes at time T_f conditional on any specific filtration node $\#i$ at time T_k:

$$\hat{c}_i(T_k) \quad := \quad \mathsf{E}\left[\tilde{c}(\boldsymbol{x}; T_f) \cdot \frac{n(0)}{n(\boldsymbol{x}; T_f)} \middle| \boldsymbol{x}(T_k) = \boldsymbol{x}_i \wedge \boldsymbol{x}(0) = \boldsymbol{x}_0 \right] \tag{3.7}$$

where by convention we typically consider the initial value of the state vector at time $t = 0$ to be zero, i.e. $\boldsymbol{x}_0 \equiv \mathbf{0}$. In essence, we are taking an expectation over all possible outcomes at time T_f over all paths in time that end at T_k in node $\#i$ and start at $t = 0$ in the central node where $\boldsymbol{x}(0) = \boldsymbol{x}_0$. We mention that this collection of paths that are all pinned into one specific state vector value to the left at $t = 0$ and to the right at $t = T_k$ are technically known as a *Brownian Bridge*. Note that the conditioning on the initial value of the state vector encompasses attainability requirements. For instance, any node that, at time T_f cannot be attained via any path from $t = 0$ [where $\boldsymbol{x}(0) = \boldsymbol{x}_0$], is clearly not admissible to contribute to the numéraire-scaled forward-propagated $\hat{c}_i(T_k)$. In practice, we compute the values (3.7) as follows. First,

populate a lattice for time horizon T_f with the probability-weighted values

$$c'_i(T_f) \;:=\; \tilde{c}_i(T_f) \cdot \frac{n(0)}{n_i(T_f)} \cdot p_i(T_f) \tag{3.8}$$

where we have used the much earlier computed node probabilities $p_i(T_f)$. Then, propagate c' via forward-rolling finite-differencing according to the forward Kolmogorov equation

$$\tfrac{\partial}{\partial t} c' \;=\; L^* \cdot c' \tag{3.9}$$

to time T_k. Finally, once more using the much earlier prepared node probabilities p_i, but now for time horizon T_k, we can compute the conditional value $\hat{c}_i(T_k)$ simply via the law of conditional expectations

$$\mathsf{E}[x|A] = \mathsf{E}\left[x \cdot \mathbf{1}_{\{A\}}\right] / P(A)$$

as

$$\hat{c}_i(T_k) \;=\; \frac{c'_i(T_k)}{p_i(T_k)}\,. \tag{3.10}$$

Once we have $\hat{c}_i(T_k)$, we simply add it to the node values of the recorded distribution at time T_k. Then, we continue rolling forward the values of c' using (3.9) until we arrive at T_p were we terminate this inner iteration.

To be clear:

We approximate the NPV at time T_k of a *path-dependent* IBOR rate payment that fixes at T_f and pays at time T_p with $T_f < T_k \leq T_p$ by its expectation conditional on the model state at time T_k.

It is self-evident that the methodology we describe here does not give the true distributional risk of any *path-dependent* contingent cashflow that fixes at T_f but is paid at a later time T_p for any time $T_k > T_f$. However, since the lag periods involved are typically of a much smaller scale than the time horizons of the entire derivatives contract (just think of a typical cross-currency swap), the approximation of the conditional expectation is in a commercial environment perfectly justifiable for the purpose of XVA calculations. In addition, we mention that the approach we have described *exactly* preserves, in expectation, the net present value as seen from $t = 0$ of any cashflow fixing at T_f at all of its propagated later time horizons T_k with $T_f < T_k \leq T_p$ as an immediate consequence of the definition of c' and the forward Kolmogorov propagation (3.9). This exact preservation of value in expectation is an important feature that avoids any bias that other, more simplistic methods (such as evaluating the IBOR rate at the later time T_k from the state vector $\boldsymbol{x}_i$ on node $\#i$) cannot easily match.

Remark 3.1. The incorporation of past cashflows via their Brownian Bridge expectation (3.7) by directly adding the resulting value $\hat{c}_i(T_k)$ into the node value $v_i(T_k)$ has one further implication. If we follow the evaluation logic precisely, and consider its real world financial representation, we notice that the term amounting to

$$\frac{\tilde{c}(\boldsymbol{x}; T_f)}{n(\boldsymbol{x}; T_f)} \tag{3.11}$$

on the right hand side of (3.7) is equivalent to an immediate net cash settlement of the future owed payment, followed by the net cash amount then being converted into a certain number of units of the numéraire asset which are then in some sense placed into a vault owned by the debtor, and to be handed over in lieu of the originally owed cash payment at T_p. This has no immediate effect on the value of the outstanding

payment, but it does make a difference as to how the value of this deposited security changes over time as we go from T_f towards T_p. For instance, whether the numéraire is a money market (savings account), or a long-dated zero coupon bond can make a difference to the performance of the numéraire asset in any lattice node filtration at T_p in which interest rates are significantly changed from what they were at $t = 0$. In practice, however, once again we invoke the argument that the time spans between T_f and T_p are usually not of any significant size. Plus, it is of course always possible in our implementation to choose a numéraire that minimizes any such synthetically generated extra variation of the exposure distribution. We would, for example, quite obviously avoid the use of a 30 year zero coupon bond as the numéraire in the valuation of a 3 year cross-currency swap. What's more, in practical tests we found that the here explained effect is below any measurable noise level in comparison to the numerical residual accuracy that we incur due to all other discretisation related approximation effects, especially if we choose an ill suited numéraire, as any quantitative analyst can confirm who ever did any measure dependency analyses of net drift errors in a Libor market model, for example. Short of a complete re-engineering of our overall valuation framework, there is very little we can do about the existence of this convert-cash-debt-to-numéraire-asset-units effect. However, given the elegance of the drift-free Kolmogorov forward propagation logic (3.7) to (3.10), and the fact that in practice the effect is entirely negligible, this small residual is primarily of academic interest and a compromise we are happily willing to make.

To conclude this section, we mention that the division by the probability in (3.10) obviously requires some care to avoid division by zero or near zero numbers. In practice, this is of no particular concern since, on any node on which p_i is zero or virtually zero, the later contribution of $\hat{c}_i(T_k)$ to any subsequently computed expectation will readily diminish, too. This is so reliable in practice that one can simply set $\hat{c}_i(T_k)$ to zero whenever $p_i(T_k)$ is not positive.

3.2.2 Compounded floating rates

Thus far, we expressly limited the discussion to forward looking rates, also known as *term* or, previously, InterBank Offered Rates. We will now explain why it is, in our assessment, admissible to use a forward looking term rate susbstitution for overnight compounded rates et. al. without any additional adjustments.

Recall that we explained in the previous section that on our finite-differencing lattice we use at any time T_k on any filtration node $\#i$ (i.e., on the lattice node that is associated with the model's driving factor variable $\boldsymbol{x}$ to be in the value state denoted by $\boldsymbol{x}_i$) the Brownian bridge expectation given in equation (3.7) to approximate the path-dependent unknown fixing of an IBOR rate that was fixed at an earlier time T_f. Denote this spatial node $\#i$ at observation time[6] T_k as the (i, k) filtration node. Now, carefully, let us appreciate that we are using the *single value* determined by equation (3.7) for all that we need to do with respect to the (i, k) filtration node, irrespective of the shape of the path that led us from $t = 0$ to this node. Meaning, instead of having a wide range of path-specific realisations of $c_i(T_k; T_f)$ which in reality involved the fixing at the earlier time T_f, and instead of essentially already having a distribution of possible stilled owed IBOR payment, we use *one single value*, namely $\hat{c}_i(T_k)$, to represent the distribution of possible outcomes for $c_i(T_k; T_f)$. We are effectively bundling all those paths into just being represented by that one value $\hat{c}_i(T_k)$. As we argued and promised, and as we shall show in some numerical results later, this projection of the distribution of $c_i(T_k; T_f)$ in the (i, k) filtration node into just one value $\hat{c}_i(T_k)$ is perfectly viable for commercial purposes. After all, we must not forget that all alternative solutions involving Monte Carlo simulations ultimately also

[6] the 'observation time' is a point in time for which we need a full value distribution to generate EPE, CVA, DVA, etc.

incur information-reducing simplifications, e.g., via involved regression function representations within the respectively used Monte Carlo early exercise algorithm [Jäc02, Car96, JA10, LS98], and the inaccuracies incurred there are never entirely negligible even if seemingly rarely mentioned in the literature these days. The message is this: we are happy to use one value $\hat{c}_i(T_k)$ instead of a full distribution of $c_i(T_k;T_f)$ single-fixing IBOR values that we should have just for that one (i,k) node. We are effectively ignoring a small amount of variance in the credit exposure on each filtration node, but we are all in all satisfied with this approximation.

And here comes the final argument: the distribution of $c_i(T_k;T_f)$ *term rate* (IBOR) values that fully fixed at an earlier time T_f, conditional on the (i,k) filtration node, *will necessarily be more widely dispersed than that of a compounding rate* that started fixing at T_f and is either ending at T_k or still ongoing when conditioning on the (i,k) filtration! This is obvious as the compounded rate at time T_k in filtration node (i,k) is completely pinned down in its instantaneous (overnight) contribution by the (i,k) filtration condition, meaning, by $\boldsymbol{x}(T_k) \equiv \boldsymbol{x}_i$ in (i,k). In a manner of speaking, for compounded rates, we'd require in (i,k) a distribution of *average rates* between T_f and T_k with a condition on the last contributing rate being known in (i,k). Our IBOR rates, in contrast, were single observation rates at the earlier time T_f, and averages have less variance than single (end-of-path) observations. As a consequence, we can be certain that, if our approximation (3.7) is good enough for IBOR rates, then it is even better for compounded rates!

4 Numerical analysis

The successful implementation of the presented methodology crucially depends on an efficient finite-differencing framework. In this section, we describe a set of considerations and techniques that are aimed at accuracy and speed of the backward and forward induction stages. This includes methods to be able to work with very sparse spatial lattices, low numbers of time steps, as well as high-efficiency low level linear operator evaluation algorithms.

4.1 Integration in time

The starting point of the finite-differencing framework is a discretisation of space into, ideally, a small number of lattice nodes. In practice this can be as few as 21 nodes per dimension, but when required for higher dimensionalities can even be pushed down to as little as 11 nodes or so per dimension. When we choose a scheme for integration in the time direction, we can take advantage of the fact that we will operate with such ultra-sparse spatial discretisations. The specific choices available are, as usual, first and higher order fully explicit schemes, first and higher order fully implicit schemes, and various mixed schemes. The main downside of explicit schemes are typically:-

1. Explicit schemes become unstable above a certain time step threshold which scales typically like $1/n_x^2$ where n_x is the number of spatial nodes per dimension.

2. For anything other than a simplistic model, an exact assessment of the stability threshold of the explicit scheme can only be obtained numerically by calculating an estimate of the largest eigenvalue of the spatially-discrete-continuous-time generator matrix.

3. In a forward propagation framework, when near the stability threshold, explicit schemes generate initially undesirably oscillations in the probability profile.

4. Explicit schemes spread out on the spatial lattice only by as many nodes (to each side) per time step as is the order of the scheme. For example, a first order explicit scheme in a single forward propagation step will only spread probability from the central node to the nearest neighbour nodes. In an ultra-sparse lattice with a total of 11 nodes, 5 time steps would be required for any probability to reach the boundary of the lattice (though in truth 5 time steps is not exactly an intimidatingly large number).

The disadvantages of (partially) implicit schemes are typically:-

I. Direct solution of the full linear systems becomes unworkable in multiple dimensions, even when lattices are ultra-sparse in each dimension.

II. Various operator-splitting schemes struggle even with moderate correlations between dimensions.

III. Higher order operator splitting schemes are typically rather involved.

IV. Some operator splitting schemes are of second order in time but become first order when correlation is introduced.

V. In more than two dimensions, operator splitting schemes become more and more involved, forfeiting the long time step advantage they may have over explicit schemes.

Whilst the above lists may not be exhaustive, they highlight the fact that for the purpose of regular three-dimensional use, and possibly even higher, in a context of few spatial nodes in each dimension, we may not have the need for implicit schemes. This is due to the fact that, as the number of spatial nodes is small, only a small number of time steps suffices to distribute probability in the forward Kolmogorov calculation to the wings of the lattice as we would want in a diligent risk management framework that should be able to show tail risk at least to some extent.

Regarding the need for short time steps for the mere sake of accuracy, as long as a certain step size preserves stability, we can crank up the convergence order using a higher order fully explicit scheme we obtain easily from the Taylor expansion of the exact propagator

$$\mathrm{e}^{\Delta t\cdot\tilde{L}^*} \;=\; 1+\Delta t\cdot\tilde{L}^*+\tfrac{1}{2}\Delta t^2\cdot{\tilde{L}^*}^2+\tfrac{1}{3!}\Delta t^3\cdot{\tilde{L}^*}^3+\cdots \tag{4.1}$$

where $\tilde{L}$ is the spatially-discrete-continuous-time generator matrix. As long as the first order explicit scheme is stable, all higher order explicit schemes are stable, too. In practice, for the purpose of CVA calculations or similar, we probably need to have time steps no longer than two weeks anyway in order to record the respective future value distributions. This suggests that, as long as an explicit scheme is stable, we will readily attain sufficient numerical integration accuracy with a second order fully explicit scheme, also known as the *predictor-corrector* scheme, but we could of course go higher in order if required.

As for the first order scheme's stability threshold, since n_x is small, stability is in our experience in practice always attained for monthly time steps, let alone two-weekly ones. Still, we need to have a methodology to assess the stability threshold, and we will come back to this very soon.

4.1.1 Linear operator evaluation

The spatially-discrete-continuous-time generator matrix $\tilde{L}$ is the result of spatial disretisation and the replacement of the spatial differential terms in L with suitably chosen finite-differencing stencils. The most popular stencil for $\partial^2 f/\partial x^2$, for instance, reads

$$\tfrac{1}{\Delta x^2}\left[f(x-\Delta x)-2\cdot f(x)+f(x+\Delta x)\right]\ . \tag{4.2}$$

The net result for the evaluation of $\tilde{L}$ on any one node is a weighted sum over the values of a set of nodes that are part of all the respectively involved derivative stencils. In a regular lattice that we can view as a linear sequence of nodes (row by row, column by column, etc.), this weighted sum is for all lattice locations over nodes of *exactly the same offsets*. The only minor exception are boundary nodes where, if we are using higher order inward-looking stencils, we'd need to add those special contributions. In practice, we prefer using zero diffusion (second derivative) boundary conditions and, at most, first order inward-looking advection (first derivative) terms, which preserves the homogeneity of the integer offset logic across all nodes. We show an example in figure 3 for the two-dimensional *seven-point* stencil for diffusion in two directions, cross-diffusion, and centre-differencing advection in both directions on a lattice of 11×6 nodes. The homogeneity of offsets

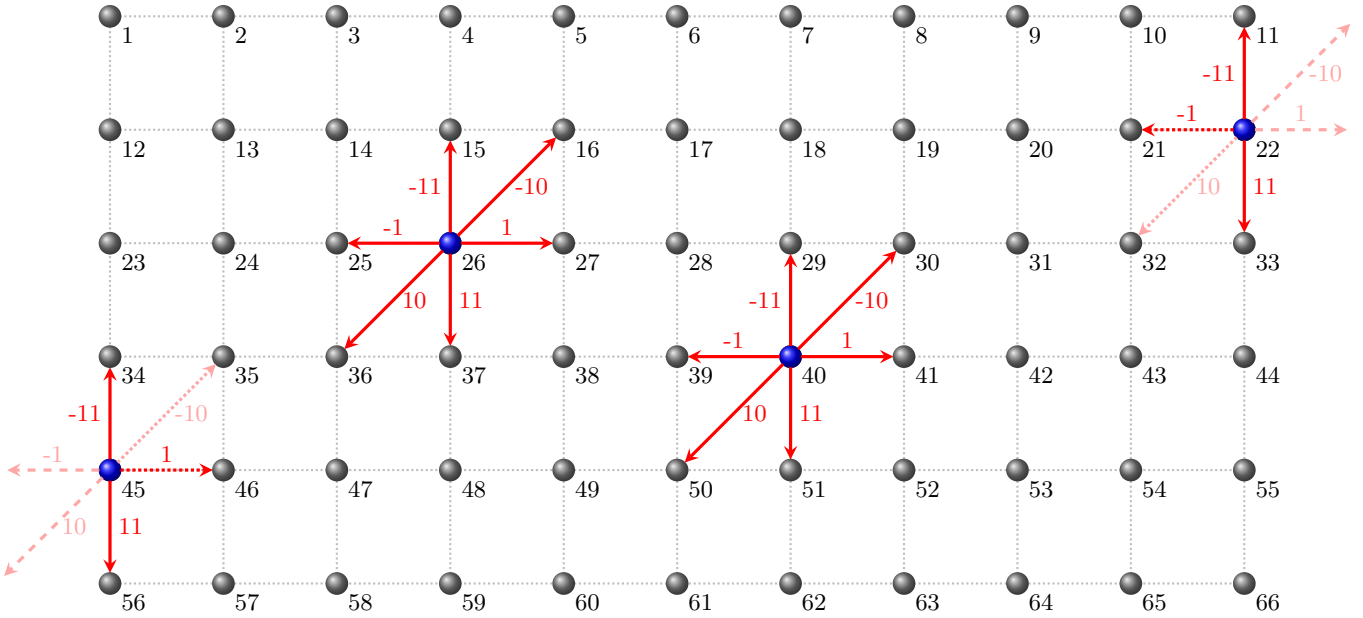


FIGURE 3: Homogeneous finite-differencing stencil node offsets on a two-dimensional lattice of sequentially enumerated nodes. Solid arrows stand for standard advection or diffusion transitions. Solid dotted arrows represent transitions that can be used for inward-differencing advection transitions. Faint dashed arrows are invalid transitions that are suppressed. Faint dotted arrows are given zero transition weight since on those nodes the cross-diffusive term is set to zero.

enables us to encode the matrix $\tilde{L}$ by a single vector of m integer offset values, where m is the total number of nodes involved in all the stencil terms that are applied, plus a coefficient block of $m \cdot N$ numbers, where N is the total number of nodes in the lattice. This form of sparse encoding makes for very fast evaluation of both $\tilde{L} \cdot v$ as well as $\tilde{L}^* \cdot v$, even when v needs to represent a multi-valued function as is usually the case with financial products that consist of multiple underlying legs. In the example of figure 3, the vector of node offsets of the generator matrix is

$$(-11 , -10 , -1 , 0 , 1 , 10 , 11)^\top \tag{4.3}$$

with the central entry 0 referring to the diagonal element on any one given row. It is obvious that this encoding implies some invalid transitions for all boundary nodes, though this can easily be handled by setting their coefficients to zero and the definition that all offsets that leave the enumerated set of nodes are simply not applicable. Row #22 of the generator's coefficient block, for instance, which is responsible for transitions out of node #22, would have the form

$$(c_{y-} , \text{N/A} , c_{x-} , -[c_{y-}+c_{x-}+c_{y+}] , \text{N/A} , 0 , c_{y+}) \tag{4.4}$$

with the diagonal element being the sum of all off-diagonal elements, with non-applicable transitions counting as zero, and all other transition coefficients being dependent on the respectively chosen dynamic model. With

```
//  Y := L̃ * X where X and Y have N rows (lattice nodes) and k columns (product legs).
//   int[]    offsets:       m   entries.
//   double[] coefficients:  N rows and m columns in row-major order.
//   double[] X:             N rows and k columns in row-major order.
//   double[] Y:             N rows and k columns in row-major order.
void LinearOperation(int[] offsets, int m, double[] coefficients,
                     int N, double[] Y, double[] X, int k) {
    for (int i = 0; i < N; ++i)
        for (int l = 0; l < k; ++l) {
            double tmp = 0;
            for (int j = 0; j < m; ++j) {
                int r = i + offsets[j];
                if (r >= 0 && r < N)
                    tmp += coefficients[i * m + j] * X[r * k + l];
            }
            Y[i * k + l] = tmp;
        }
}
```

Code sample 1: Efficient evaluation of action of sparsely encoded matrix represented by the integer vector `offsets[]` and the transition coefficient block `coefficients[]`.

these definitions, the computation of the linear operation

$$\boldsymbol{Y} \;:= \tilde{L} \cdot \boldsymbol{X} \tag{4.5}$$

where $\boldsymbol{X}$ contains $N \cdot k$ values representing the values of k legs of a financial product on N nodes, and $\boldsymbol{Y}$ is of course of equal size, can be written concisely as is shown in code excerpt 1. Note that this code handles *any dimensionality*: there is no need for any form of loop nesting over the dimensions since all considerations of dimensionality are subsumed into the definition of the offset and coefficient vector. Since this is a linear loop over all nodes, *it is* also *straightforward to split this into segments for multi-threaded evaluation*. What's more, with the change of *exactly two code characters*, namely one sign and one lookup integer, we can equally efficiently evaluate the transpose of the linear operator as shown in code excerpt 2. Obviously, the transpose linear operation can be simplified further for the case of the pure probability forward propagation in which case the loop in the variable `l` over the product's legs can be removed.

4.1.2 Conservative stability estimate using Gerschgorin's circle theorem

The use of any finite-differencing scheme requires the satisfaction of stability criteria. Let us recall that the backward induction of values in a discrete spatial and temporal setting, for any two-time-level method[7] is always in the form

$$\boldsymbol{v}(t - \Delta t) = A \cdot \boldsymbol{v}(t) \tag{4.6}$$

where A is the *backward induction propagator matrix*. Obviously, in a repeated iteration of (4.6) we must demand that none of the eigenvalues of A are outside the unit circle, else the associated eigenmode would incur geometric growth, which we would observe as what we call *numerical instability*. We mention the most basic formulation of the requirement for stability in this context in full generality because it appears to be overlooked in the literature that any integration method, explicit or implicit or mixing of some kind, and

[7] We acknowledge the existence and usefulness of multi-time-level methods such as BDF2 or Runge-Kutta methods, etc., but restrict the discussion to the simplest of all, namely two-time-level methods.

```
//  Y := L̃^⊤ * X where X and Y have N rows (lattice nodes) and k columns (product legs).
//   int[]    offsets:       m   entries.
//   double[] coefficients:  N rows and m columns in row-major order.
//   double[] X:             N rows and k columns in row-major order.
//   double[] Y:             N rows and k columns in row-major order.
void TransposeOperation(int[] offsets, int m, double[] coefficients,
                        int N, double[] Y, double[] X, int k) {
    for (int i = 0; i < N; ++i)
        for (int l = 0; l < k; ++l) {
            double tmp = 0;
            for (int j = 0; j < m; ++j) {
                int r = i - offsets[j];
                if (r >= 0 && r < N)
                    tmp += coefficients[r * m + j] * X[r * k + l];
            }
            Y[i * k + l] = tmp;
        }
}
```

Code sample 2: Efficient evaluation of action of transpose of sparsely encoded matrix represented by the integer vector `offsets[]` and the transition coefficient block `coefficients[]`. Note the two sole differences to code excerpt 1 framed in red.

any multi-time-level method, can in principle be stable or unstable, depending on its full structure, and the dependency can be non-trivial. Ultimately, whether a method is numerically stable or not is is only ever determined by the spectrum of the propagator A[8]. For the simple first order explicit integration method, the backward induction propagator for a time step of size $\Delta t > 0$ is given by

$$A = 1 + \Delta t \cdot \tilde{L} \,. \tag{4.7}$$

For there to be stability, A must have no eigenvalues outside the unit circle. Obviously, therefore, the spectrum of $\tilde{L}$ must not have any positive real parts. This, however, is not only a consequence of the first order integration scheme, but an inherent feature resulting from the fact that the continuous-time propagator

$$\mathrm{e}^{\Delta t \cdot \tilde{L}} \tag{4.8}$$

for any sensible model cannot have eigenvalues outside the unit circle. For this to be the case, it suffices that the spatially discretised generator $\tilde{L}$ is a *Metzler* matrix. This should always be the case if we consider that our financial model's spatial discretisation (we are still in continuous time), when viewed as a stochastic process (forward Kolmogorov view) should only ever allow transitions from the current state to others, meaning, conditional on being in any state (discretisation node) $\#i$, the probability of still being in the same state at a later time can only be ≤ 1, but *never greater than* 1. This latter view puts us in the framework of continuous-time Markov chains whose transition matrices satisfy

$$\tilde{L}_{ij} \;\geq\; 0 \quad \forall \quad i \neq j \tag{4.9}$$

$$\sum_j \tilde{L}_{ij} \;=\; 0 \,. \tag{4.10}$$

[8] We mention the little known fact that even in simple two-dimensional diffusions, it is possible for a fully implicit scheme with an inconsistent corner-specific (spatial finite-differencing) boundary condition specification to become unstable! The hallmark of such accidents is a growth of oscillations coming in from two of the four 2d lattice corners, which is, alas, easy to miss since the growing modes have little amplitude in the centre region of the lattice, but they still destroy the usability of the numerical result! An eigenvalue analysis of the 2d propagator A, however, will always tell!

In the absence of any advection, for instance, any non-zero diffusive terms in $\tilde{L}$ *with positive diffusivity* can only give rise to eigenvalues on the real axis with non-positive value. This is a simple consequence of the fact that real world diffusions only ever *diffuse*, they do not give rise to wave transport nor do they ever concentrate the distribution. Mathematically, we achieve this by the aid of symmetric mono-axial diffusion stencils as in (4.2), and the use of the seven-point stencil for cross-diffusion (correlation) terms as we will discuss in section 4.2.1.clear

In the presence of advection such as in a standard Ornstein-Uhlenbeck setting (aka extended Vasicek aka Hull-White model), or indeed as introduced for martingale conditions as later elaborated in section 4.3, we may have to make a pragmatic choice. Ideally, we prefer that the respective drift is numerically incorporated via full or at least partial upwind schemes to whichever extent required such that the Metzler condition (4.9) is satisfied. In practice, though, we can not always afford this for reasons of numerical accuracy since the nearest neighbour upwind scheme is nothing other than the use of a forward differencing term to represent advections terms proportional to $\partial/\partial x$, and that means going from second order in Δx to only first order. We necessarily do this on the boundaries anyway since we must adhere to the no-arbitrage rules for underlyings such as FX rates which have a spot drift given by $\mathrm{d}S = (r_d - r_f)S\mathrm{d}t$. Clearly, when both r_d and r_f are diffusions, and we are, say, in a (relatively sparse) three-dimensional lattice, then the low-S and high-S faces of the lattice cube contribute significantly to the overall valuation and we cannot simply set the spot drift for S to zero on these two outer layers, whence we use one-sided differencing there, and that's fine in practice.

Everywhere else, however, except of course on boundary nodes, for the sake of numerical accuracy, we keep the centre-differencing approach for advection terms, even if it violates our preferred Metzler property in some rows of the generator matrix. It turns out, though, that with our simple heuristic given in the following, we can operate very well in practice, so we shall for the sake of argument continue with our explanation as if the Metzler property was exactly preserved.

In plain terms, we use the following approach to determine the maximum time step that still gives us numerical viability for our purposes. First, we use Gerschgorin's circle theorem [Ger31] to estimate the largest absolute eigenvalue of the generator matrix $\tilde{L}$, say $|\lambda_{\max}|$. Since the first modes to become unstable in explicit schemes are typically associated with the Nyquist frequency whose modes are of oscillatory nature and thus associated with a negative (real part) of the respective eigenvalue of (4.7), we use the simple heuristic to assume that the eigenvalue associated with $|\lambda_{\max}|$ is on the negative real axis. The mere demand that $(1 - \Delta t \cdot |\lambda_{\max}|)$ is inside the unit circle would lead to

$$1 - \Delta t \cdot |\lambda_{\max}| \;>\; -1 \qquad \text{whence} \qquad \Delta t \;<\; \frac{2}{|\lambda_{\max}|}\,. \tag{4.11}$$

However, taking to heart our preference that we avoid negative transition probabilities in addition to having numerical stability, we go further and demand

$$1 - \Delta t \cdot |\lambda_{\max}| \;>\; 0 \tag{4.12}$$

since this is the self-transition[9] probability in a first order explicit scheme. from which we obtain

$$\Delta t \;<\; \frac{1}{|\lambda_{\max}|}\,. \tag{4.13}$$

The final piece in the puzzle is the estimation of $|\lambda_{\max}|$. For this, we use the conservative approximation that is simply the maximum over all rows of $\tilde{L}$ of the sum of the absolute values of all coefficients of each

[9] *self* in the sense of the probability of initially being in the mode associated with $\lambda_{\max}$ and remaining in the same mode

row:

$$|\lambda_{\max}| \quad :\approx \quad \max_{i=1\ldots N} \left(\sum_j \left| \tilde{L}_{ij} \right| \right) \quad . \tag{4.14}$$

Since we previously encoded the generator $\tilde{L}$ as a block of N rows of k coefficients each, this is simply the maximum over N rows of the sum of the absolute values of k coefficients each, and extremely fast to calculate. In reality, this estimate is of course conservative, and experiments show that it can be readily conservative by a factor two to three. Even if it were a factor 4 overly conservative, it would mean we are taking four time steps where we need only one. This, however, gives us in practice an extra safety buffer. It also helps with the smoothing out of distributions that otherwise could initially look somewhat jagged, meaning, we would typically want to have two to four times as many steps as strictly required in an explicit scheme. Even if the factor were as big as 6, we'd still benefit tremendously from not having to compute the eigenvalue by other means, which invariably costs as much as several tens of evaluations of the linear operator, and we'd still have significantly less effort in the actual finite-differencing propagation. For a first order one-dimensional scheme this may not be so clear cut since then a fully implicit scheme is only about $3N$ individual floating point operations per time step, with the explicit scheme being about N, but when we go to second order, this becomes abundantly clear, let alone when we go to higher dimensions. All in all, we find the explicit stepping scheme for (ultra-)sparse spatial discretisations with a Gerschgorin stability estimate to be a highly efficient, reliable, and robust method for practical applications.

We show in figure 4 an example for the actual spectrum of the spatially discretised generator in a real world example for a USDCHF cross-currency calculation in a standard one-factor-per-currency-and-FX LGM model where we used $5\times5\times5 = 125$ nodes for simplicity. As the generator matrix is symmetric, we

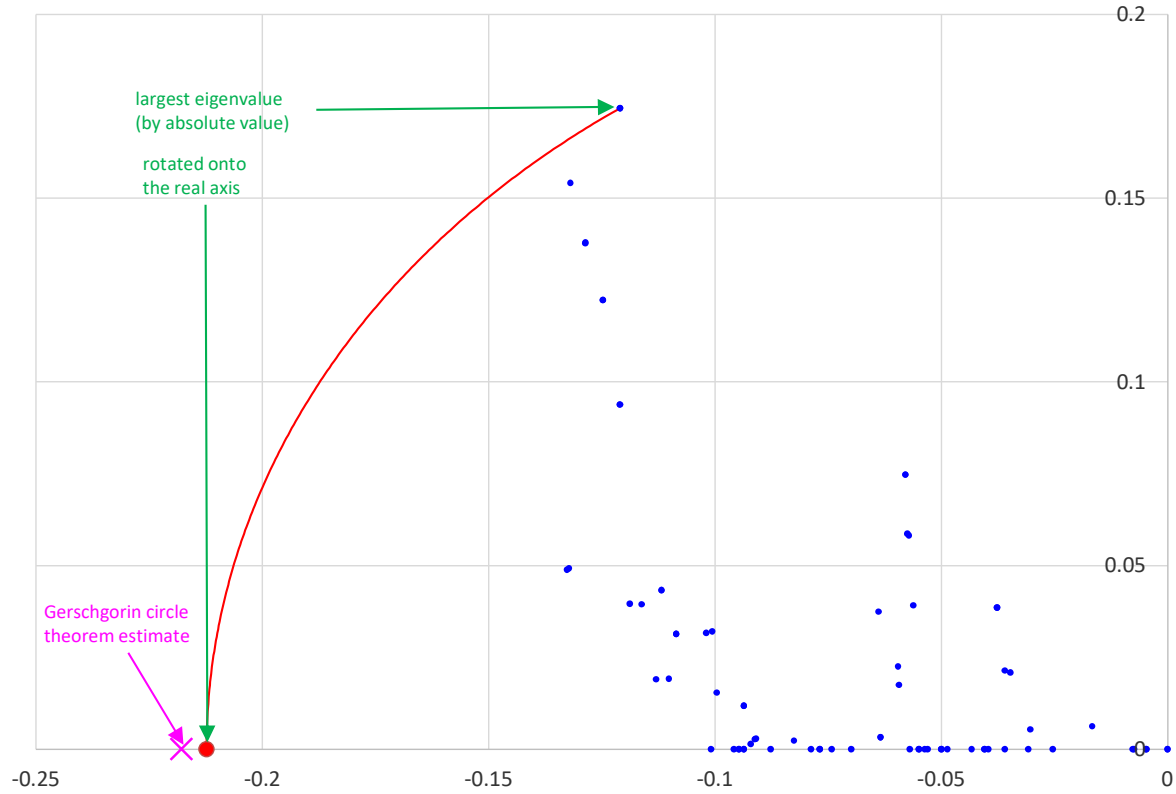


FIGURE 4: Eigenvalues of a real world calculation example for a generator discretised onto a lattice of $5\times5\times5$ nodes.

only show values with positive imaginary part. As we can see in this example, the assumption of the largest eigenvalue (by absolute value) being on or near the real axis does not really hold whence, our assumption,

effectively amounts to the rotation of that largest eigenvalue onto the real axis. We show next to that the result of our Gerschgorin circle theorem in that case. We mention that our heuristic still gives us a perfectly stable maximum explicit integration scheme size $\Delta t < 1/|\lambda_{\max}|$ in this case. We admit that further research should be conducted to fully justify our Gerschgorin heuristic and hope that this will be addressed at some point. For the purpose of this article, however, we are content with it having proved to be highly reliable in production applications (in up to a total of 4 currencies + 3 FX = 7 dimensions).

4.2 Cross-diffusion stencils

In multi-dimensional calculations, a common point of misunderstandings, and sometimes contention, is the incorporation of cross-diffusive terms. In two dimensions, assuming that volatility terms are subsumed into the definition of the x and y coordinates, a generic diffusion operator conditioned on evaluation in just the location (x, y), can be written in the form

$$L \;=\; \tfrac{1}{2}\partial_x^2 + \rho\partial_{xy} + \tfrac{1}{2}\partial_y^2 \,. \tag{4.15}$$

4.2.1 The seven-point stencil

The discrete diffusivity coefficients[10] of the seven-point stencil for L in (4.15) as depicted in figure 5 are defined as

$$c_x = \left(\frac{1}{\Delta x^2} - |c_{xy}|\right)_+ \qquad c_y = \left(\frac{1}{\Delta y^2} - |c_{xy}|\right)_+ \qquad c_{xy} = \frac{\rho}{\Delta x \Delta y} \tag{4.16}$$

where the $(\cdot)_+$ flooring operations ensure non-negative diffusivity and thus the Metzler property of the generator matrix. With these coefficients, the 5-th (i.e., middle) row of the generator matrix for a lattice of 3×3 nodes reads

$$\tilde{L}_{5\,\cdots} \;=\; \tfrac{1}{2} \cdot (\; [-c_{xy}]_+ \;,\; c_y \;,\; [c_{xy}]_+ \;,\; c_x \;,\; -2 \cdot (c_x + c_y + |c_{xy}|) \;,\; c_x \;,\; [c_{xy}]_+ \;,\; c_y \;,\; [-c_{xy}]_+ \;) \,. \tag{4.17}$$

When $\rho \geq 0$, this becomes of course

$$\tilde{L}_{5\,\cdots} \;=\; (\; 0 \;,\; \tfrac{1}{2}c_y \;,\; \tfrac{1}{2}c_{xy} \;,\; \tfrac{1}{2}c_x \;,\; -(c_x + c_y + c_{xy}) \;,\; \tfrac{1}{2}c_x \;,\; \tfrac{1}{2}c_{xy} \;,\; \tfrac{1}{2}c_y \;,\; 0 \;) \,. \tag{4.18}$$

The seven-point stencil is of second order in $(\Delta x, \Delta y)$.

4.2.1.1 Preservation of the diagonal transport line when $\rho \to \pm 1$

An interesting observation of the seven-point stencil is the transport behaviour it displays in the limit of perfect codependence. It is obvious that even in the simplistic homogeneous case of $\Delta x = \Delta y$, the conventional four-point stencil as shown in figure 6 cannot converge to the correct stratification of transport being restricted along diagonal lines since the four-point stencil always invokes transfer to and from *all* eight surrounding nodes, which is in contrast to the structure of the partial differential operator (4.15) when $\rho = \pm 1$. The seven-point stencil, however, preserves the stratification *exactly* when $\rho = \pm 1$ and $\Delta x = \Delta y$. We mention that it is of course always possible to transform the case of perfect or very high correlation to principal axes, but we are still interested in the situation when such transformations are undesirable for other reasons.

[10] Note that we differ with our definition by a factor 2 from the literature on diffusivity coefficients in material sciences. This is merely to bring our usage of *diffusivity* in scale to mean *instantaneous variance*.

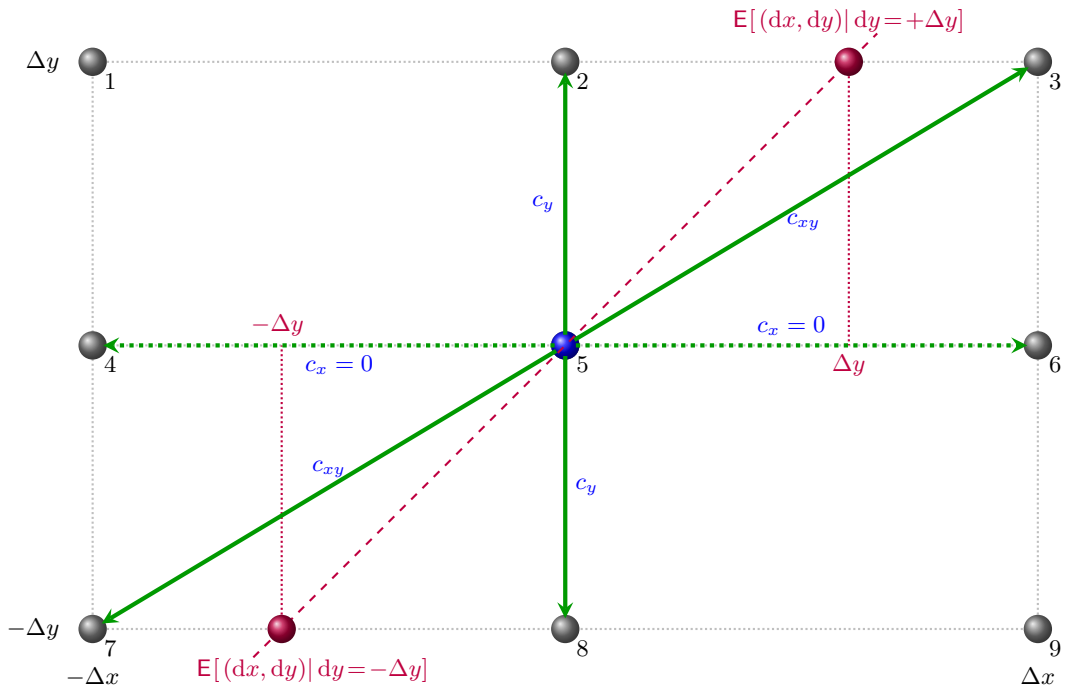


FIGURE 5: The seven-point stencil preserves diagonal transport *in expectation* in the limit of $\rho = 1$ when $\Delta x > \Delta y$.

In the case of $\rho = 1$ and $\Delta x > \Delta y$, the stratification along diagonal lines can no longer be preserved since there simply are no nodes along diagonal lines. It is of interest to note what then is the result of transport out of the central node. Considering, for example, that we have $c_x = 0$ in this situation, we compute the weighted average of transport in the top right quadrant, or, to put it differently, the conditional expectation value of the transition location. We do this by interpreting the off-diagonal coefficients of $\tilde{L}$ as the transition intensity from the central node to its neighbours, in line with the theoretical framework of Markov chains. Conditional on having an upward move, i.e., $\mathrm{d}y = \Delta y$, we obtain

$$\mathsf{E}\Big[\mathrm{d}x\Big|\,\mathrm{d}y=+\Delta y\Big] = \frac{\frac{1}{2}c_y\mathrm{d}t\cdot 0+\frac{1}{2}c_{xy}\mathrm{d}t\cdot\Delta x}{\frac{1}{2}c_y\mathrm{d}t+\frac{1}{2}c_{xy}\mathrm{d}t} \tag{4.19}$$

$$= \frac{\mathrm{d}t}{\Delta x\Delta y}\cdot\Delta x\Big/\left[\left(\frac{1}{\Delta y^2}-\frac{1}{\Delta x\Delta y}\right)\mathrm{d}t+\frac{\mathrm{d}t}{\Delta x\Delta y}\right] = \Delta y\,. \tag{4.20}$$

In other words, *at least* ***in expectation, transport is still diagonal***, as we demonstrate in figure 5. This is clearly a desirable feature one would intuitively hope for from any sensible cross-diffusion stencil in the limit of perfect codependence.

4.2.1.2 Marginal diffusivity

It is usually taken for granted that the mono-axial diffusivity along each dimension is preserved in a 2- (or higher-) dimensional setting when correlation is introduced. This means, given a central amount of probability placed at any specific location at time t, we expect its *net* spreading out (i.e., its *diffusion*) in both directions over time to adhere to the individually given mono-axial diffusion coefficients. This is, however, not necessarily given by any specific finite-differencing stencil (which represents *spatial* discretisation whilst we still work in continuous time at this point). To elaborate this effect for the seven-point stencil, we define the net marginal absolute transport in the x-direction, which we denote as the *marginal diffusivity*, as

$$\tfrac{1}{2}\cdot\frac{\mathsf{E}[|\mathrm{d}x|]}{\mathrm{d}t}\,. \tag{4.21}$$

For the seven-point stencil, when $\rho \geq 0$ and $\Delta x \geq \Delta y$, we obtain

$$\begin{aligned} \tfrac{1}{2}\frac{\mathsf{E}[|\mathrm{d}x|]}{\mathrm{d}t} &= \tfrac{1}{2}c_x \cdot \Delta x + \tfrac{1}{2}c_y \cdot 0 + \tfrac{1}{2}c_{xy} \cdot \Delta x \qquad (4.22) \\ &= \tfrac{1}{2}\left(\frac{1}{\Delta x^2} - \frac{\rho}{\Delta x \Delta y}\right)_+ \mathrm{d}t + \tfrac{1}{2}\frac{\rho}{\Delta x \Delta y} \\ &= \frac{1}{2\Delta x^2} \cdot \max\left(1, \rho \cdot \frac{\Delta x}{\Delta y}\right) \qquad (4.23) \end{aligned}$$

due to the flooring of c_x at zero in (4.16). This means the marginal diffusivity in the x direction increases beyond the originally intended value of $\frac{1}{2\Delta x^2}$ when

$$|\rho| \; > \; \frac{\Delta y}{\Delta x} \qquad (4.24)$$

where $\frac{\Delta y}{\Delta x} \leq 1$ in our setting of $\Delta x \geq \Delta y$. This increased marginal diffusivity is good and proper: it is exactly what ensures the diagonal-transport-in-expectation in the limit of $|\rho| \to 1$. It does, however, have a side effect which becomes important when we use ultra-sparse discretisations: whenever the underlying variable x is associated with a transformed variable, e.g., e^x, and this transformed variable is subject to certain drift constraints (such as being a martingale), the marginal diffusivity in one direction generates a bias that requires compensation. In a spatially continuous setting of the ubiquitous exponential e^x, with $\mathrm{d}x = \sigma \mathrm{d}W$ this shows up as the term $-\frac{1}{2}\sigma^2 \mathrm{d}t$ as we all know from Itō's lemma. In a spatially discrete setting, however, the compensation term can differ significantly as we will discuss in more detail in section 4.3, simply due to the sparsity of the discretisation. On top of the pure discretisation effect, however, we have in addition (potentially, depending on ρ and $\Delta y/\Delta x$) a change of the effective marginal diffusivity in each of the coordinate directions caused by the choice of cross-diffusion stencil! This effect is not necessarily subtle depending on the magnitude of ρ and the lattice discretisation sizes whence we must consider spot-drift corrections of exponential martingales not only when discretisations are sparse. The main message here is: beware of attempts to compensate Itô terms (like $\frac{1}{2}\sigma^2 \mathrm{d}t$) induced by log-spot transformations by merely adding such continuous-space-calculated terms to the numerical algorithm! We should always compensate directly on the numerical lattice in order to be able to capture additional effects that we cannot see in our continuous-space analytics. Ultimately, our aim is always to preserve any original martingale condition *exactly*, even if we had only 3 spatial node levels in the respective dimension!

4.2.1.3 Four versus Seven when $\Delta x \neq \Delta y$

The most commonly used stencil for a single bi-variate cross-diffusion stencil is probably the form

$$\begin{aligned} \partial_{xy} f \;\approx\; \tfrac{1}{4\Delta x \Delta y}\Big[& f(x+\Delta x, y+\Delta y) + f(x-\Delta x, y-\Delta y) \qquad (4.25) \\ & - f(x-\Delta x, y+\Delta y) - f(x+\Delta x, y-\Delta y)\Big] \end{aligned}$$

which we refer to as the *four-point stencil* for obvious reasons. Note that this is a stencil for just the cross-diffusion term itself. To form a spatial discretisation for the full generator (4.15), we'd add two more terms involving the mono-axial diffusion stencil (4.2) once in the x and once in the y direction, which gives us

$$\begin{aligned} \frac{1}{2}\partial_x^2 f + \rho \partial_{xy} f + \frac{1}{2}\partial_y^2 f \;\approx\; & \tfrac{1}{2\Delta x^2}\Big[+f(x-\Delta x, y) - 2\cdot f(x,y) + f(x+\Delta x, y)\Big] \qquad (4.26) \\ & + \tfrac{\rho}{4\Delta x \Delta y}\Big[+f(x+\Delta x, y+\Delta y) + f(x-\Delta x, y-\Delta y) \\ & \qquad - f(x-\Delta x, y+\Delta y) - f(x+\Delta x, y-\Delta y)\Big] \\ & + \tfrac{1}{2\Delta y^2}\Big[+f(x, y-\Delta y) - 2\cdot f(x,y) + f(x, y+\Delta y)\Big]\,. \end{aligned}$$

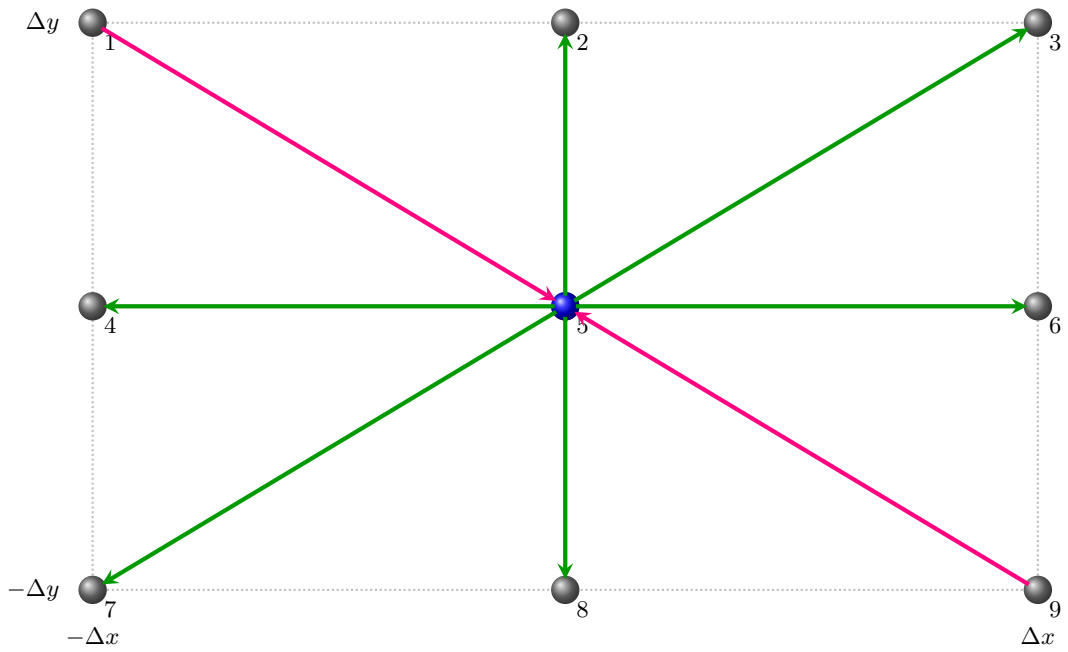


FIGURE 6: The transitions of the full diffusion with the four-point cross-diffusion stencil.

We show the transitions corresponding to the right hand side terms of the discrete generator (4.26) in figure 6 where we once again used $\Delta x > \Delta y$ and $\rho > 0$ in our example. As we can see, whenever $\rho \neq 0$, this stencil, interpreted as a spatially discrete continuous-time Markov chain, will have

negative transition probabilities.

We need to be clear about this point: negative transition probabilities render any model (or its implementation) inherent of financial arbitrage! It is remarkable that this is hardly ever noticed or commented on in the literature on financial mathematics. Considering the importance being given in financial mathematical research on avoiding *sources of arbitrage*, it is surprising that despite all those efforts at the point of mathematical model formulation, when it comes to numerical implementations, such obvious violations of the desired criteria are ignored simply because they belong in the domain of numerical analysis. What's more, this feature would violate the *the Metzler property* of our spatially discretised generator matrices, and that is something we wish to avoid if we can. Given that the seven-point stencil is just like the four-point stencil of second order in $(\Delta x, \Delta y)$, our preference for the seven-point stencil is therefore clear.

Having scolded the four-point stencil, we mention one of its more pleasant features: in contrast to the seven-point stencil, it always preserves the marginal diffusivity, irrespective of the magnitude of ρ. For instance, the net transport in the x direction alone in the right hand side of the nine point cell in figure 6 always amounts to the same for all levels of ρ due to the fact that the net outgoing transport intensity from $(0,0)$ to $(\Delta x, \Delta y)$ of the cross-diffusion stencil is exactly compensated by its counterpart from $(\Delta x, -\Delta y)$ to $(0,0)$. It is due to this preservation property that the four-point stencil is somewhat less vulnerable to log-spot drift compensation errors generated by analytical Itô-drift terms being incorporated as a simple advection term (whose coefficient is taken from the analytical continuous-space derivation, i.e., as the analytical Itô term). In our context, however, we definitely, overall, prefer the benefits of the seven-point stencil and, as we will discuss in section 4.3, have ways to mitigate any martingale drift errors by other, simple, means that will automatically compensate for any small marginal diffusivity discrepancy arising in high correlation situations.

4.2.2 The fifteen point stencil in three dimensions

Cross-currency swaps are a popular financial instrument to reduce or gain exposure to interest rates in a foreign currency whilst funding in a domestic currency. Despite being comparatively simple in their financial complexity, even the simplest model for the calculation of their exposure profiles requires three dimensions in the resulting dynamics. Focussing solely on the diffusive aspect but allowing for correlation, after local rescaling as we did before in two dimensions to subsume volatility coefficients into the length scale of the axis variables, the spatially continuous generator can be written as

$$L \;=\; \tfrac{1}{2}\partial_x^2 + \tfrac{1}{2}\partial_y^2 + \tfrac{1}{2}\partial_z^2 + \rho_{xy}\partial_{xy} + \rho_{yz}\partial_{yz} + \rho_{xz}\partial_{xz} \; . \tag{4.27}$$

We assume in the following (without loss of generality by allowing for individual axis reflections) that $\rho_{xy} \geq 0$, $\rho_{xz} \geq 0$, and $\rho_{yz} \geq 0$. As we replace L by a spatially discrete approximation $\tilde{L}$, we notice that in the limit of perfect codependence between all three underlyings we should end up with a spatially discrete generator that permits transport only in the cubic diagonal direction, at least in the homogeneous case $\Delta x = \Delta y = \Delta z$. Alas, this limiting case cannot be attained when we use three seven-point stencils in the three coordinate pair planes. This is simply because none of the stencils invoke a transition from the central node to the outer corner in the set of nearest neighbour cells formed by $3 \times 3 \times 3$ nodes as we show schematically in figure 7. In order to be able to cater for high correlations even in three dimensions, we therefore add the cubically diagonal corner points to the set of nodes we use to spatially discretise the three-dimensional diffusion operator L given in (4.27). Denoting the value v at node (i,j,k) as $v_{i,j,k}$, and using the following definitions based on the action at the central node at (2,2,2)

$$\begin{aligned}
(\tilde{L}_x \cdot v)_{2,2,2} &= \tfrac{1}{2}(v_{1,2,2} - 2v_{2,2,2} + v_{3,2,2}) & (\tilde{L}_{yz} \cdot v)_{2,2,2} &= \tfrac{1}{2}(v_{2,1,1} - 2v_{2,2,2} + v_{2,3,3}) \\
(\tilde{L}_y \cdot v)_{2,2,2} &= \tfrac{1}{2}(v_{2,1,2} - 2v_{2,2,2} + v_{2,3,2}) & (\tilde{L}_{xz} \cdot v)_{2,2,2} &= \tfrac{1}{2}(v_{1,2,1} - 2v_{2,2,2} + v_{3,2,3}) \\
(\tilde{L}_z \cdot v)_{2,2,2} &= \tfrac{1}{2}(v_{2,2,1} - 2v_{2,2,2} + v_{2,2,3}) & (\tilde{L}_{xy} \cdot v)_{2,2,2} &= \tfrac{1}{2}(v_{1,1,2} - 2v_{2,2,2} + v_{3,3,2}) \\
& & (\tilde{L}_{xyz} \cdot v)_{2,2,2} &= \tfrac{1}{2}(v_{1,1,1} - 2v_{2,2,2} + v_{3,3,3}) \; ,
\end{aligned} \tag{4.28}$$

we write the discretisation of L as the following sum of uni-directional spatially discrete diffusion operators

$$\tilde{L} = c_x\tilde{L}_x + c_y\tilde{L}_y + c_z\tilde{L}_z + c_{xy}\tilde{L}_{xy} + c_{yz}\tilde{L}_{yz} + c_{xz}\tilde{L}_{xz} + c_{xyz}\tilde{L}_{xyz} \; . \tag{4.29}$$

Due to the total number of nodes involved in (4.29), we refer to this as the *fifteen point stencil in three dimensions*. From a Taylor expansion for $v(x,y,z)$ around the central node (see appendix A), by matching the six individual derivatives in (4.27), we identify the following coefficients:-

$$\begin{aligned}
c_x &= \frac{1}{\Delta x^2} - c_{xy} - c_{xz} - c_{xyz} & c_{yz} &= \frac{\rho_{yz}}{\Delta y \Delta z} - c_{xyz} \\
c_y &= \frac{1}{\Delta y^2} - c_{xy} - c_{yz} - c_{xyz} & c_{xz} &= \frac{\rho_{xz}}{\Delta x \Delta z} - c_{xyz} \\
c_z &= \frac{1}{\Delta z^2} - c_{xz} - c_{yz} - c_{xyz} & c_{xy} &= \frac{\rho_{xy}}{\Delta x \Delta y} - c_{xyz}
\end{aligned} \tag{4.30}$$

Note that since we had six derivative terms to match, and seven free parameters, the coefficient c_{xyz} is in principle as yet free for us to choose. We want to bear in mind non-negativity of diffusivities, though, meaning *all* of the coefficients must satisfy

$$c_{\ldots} \;\geq\; 0 \; . \tag{4.31}$$

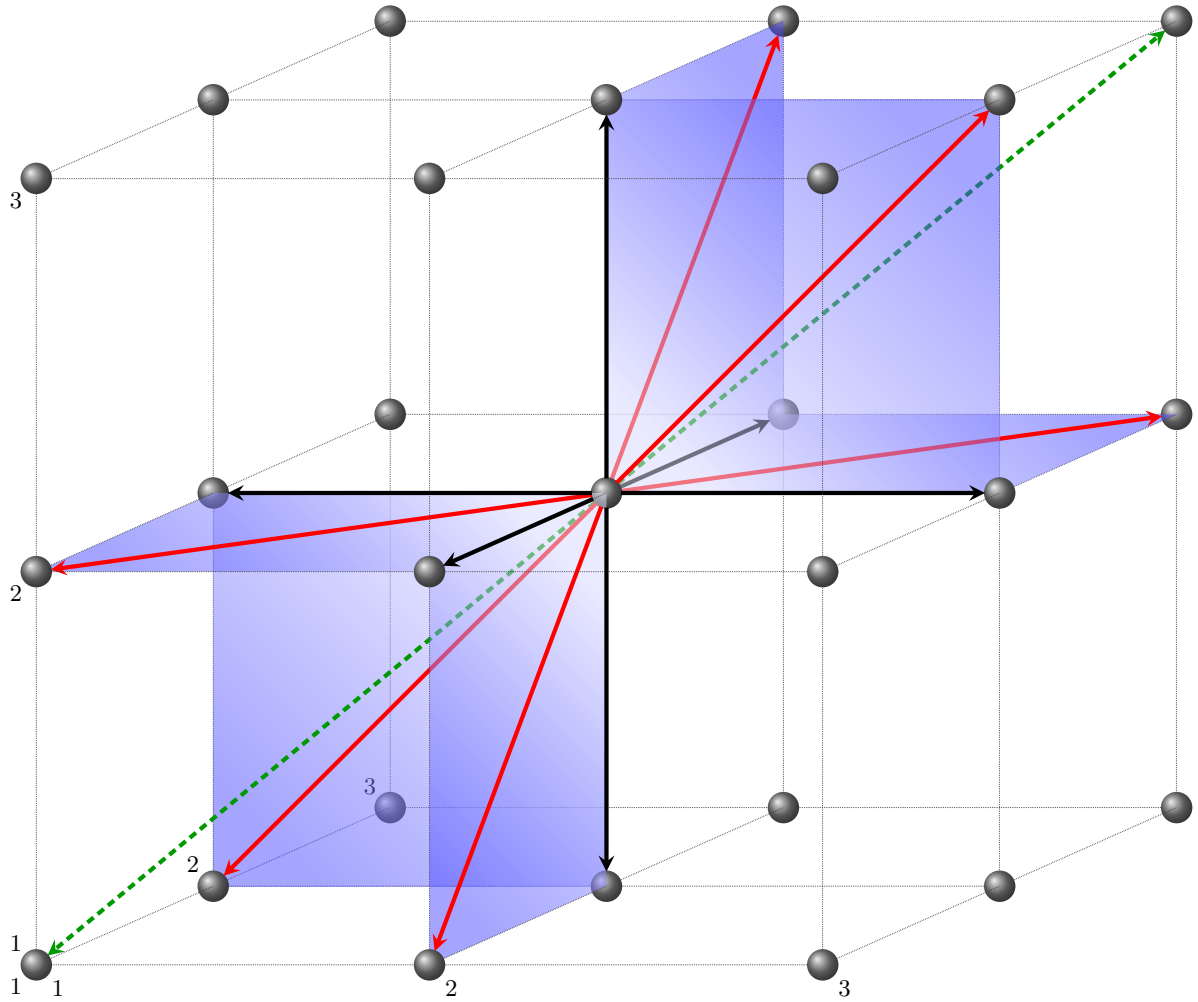


FIGURE 7: In three dimensions, the use of two-dimensional seven-point stencils alone for cross-diffusive terms cannot attain the case of purely cubically diagonal transport of perfect codependence between all three underlyings. Solid black arrows stand for transitions due to mono-axial diffusions, solid red arrows represent bi-axial cross-diffusion transitions as given by the three two-dimensional seven-point stencils. The dashed green arrows indicate the cubic diagonal transitions that cannot be attained by two-dimensional stencils alone.

We therefore need to have

$$c_{xyz}^{\min} \;\leq\; c_{xyz} \;\leq\; c_{xyz}^{\max} \tag{4.32}$$

with

$$c_{xyz}^{\max} \;:=\; \min\left(\tfrac{\rho_{yz}}{\Delta y\Delta z}, \tfrac{\rho_{xz}}{\Delta x\Delta z}, \tfrac{\rho_{xy}}{\Delta x\Delta y}\right) \tag{4.33}$$

$$c_{xyz}^{\min} \;:=\; \max\left(0, \tfrac{\rho_{xy}}{\Delta x\Delta y} + \tfrac{\rho_{xz}}{\Delta x\Delta z} - \tfrac{1}{\Delta x^2}, \tfrac{\rho_{yz}}{\Delta y\Delta z} + \tfrac{\rho_{xy}}{\Delta x\Delta y} - \tfrac{1}{\Delta y^2}, \tfrac{\rho_{xz}}{\Delta x\Delta z} + \tfrac{\rho_{yz}}{\Delta y\Delta z} - \tfrac{1}{\Delta z^2}\right) . \tag{4.34}$$

Recalling that we had restricted to the case of $\rho_{xy} \geq 0$, $\rho_{xz} \geq 0$, and $\rho_{yz} \geq 0$, we notice that the bracket for c_{xyz} diminishes to the single value 0 when the smallest of the correlations becomes zero, as we should expect. As a practical choice, we use the upper limit, i.e., we always set

$$c_{xyz} \;:=\; \min\left(\tfrac{\rho_{yz}}{\Delta y\Delta z}, \tfrac{\rho_{xz}}{\Delta x\Delta z}, \tfrac{\rho_{xy}}{\Delta x\Delta y}\right) . \tag{4.35}$$

This makes all of c_{xy}, c_{yz}, and c_{xz} in (4.30) non-negative. Just as it was the case with the seven-point stencil in two dimensions, however, it is possible for perfectly reasonable configurations to yield net negative diffusivities c_x, c_y, or c_y simply due to $\Delta x \neq \Delta y \neq \Delta z$, whence c_x, c_y, and c_y need to be floored at zero.

We had so far restricted ourselves to the case $\rho_{xy} \geq 0$, $\rho_{xz} \geq 0$, and $\rho_{yz} \geq 0$. When this is not given, we identify a principal diagonal axis, and simply associate the up-down transitions in the x, y, and z direction with the direction of the principal axis, meaning, we effectively swap the sign of some of the coordinates, or mirror-flip the respective axis to use a visual interpretation. A principal direction of codependence can be identified exactly if

$$\rho_{xy} \cdot \rho_{xz} \cdot \rho_{yz} \quad > \quad 0 \tag{4.36}$$

since only in that case is it possible to attain the limit of perfect codependence given by $|\rho_{xy}| = |\rho_{xz}| = |\rho_{yz}| = 1$ due to the requirement that the correlation matrix must be positive semi-definite. The directions in which the fifteen point stencil transitions are to be assigned are uniquely determined by the choice of the two principle outer corners, and those can be selected by the simple rule

$$\begin{aligned} (\mathrm{d}x, \mathrm{d}y, \mathrm{d}z)_{\mathrm{up}} \quad &= \quad \big(\quad \Delta x, \quad \operatorname{sign}(\rho_{xy}) \cdot \Delta y, \quad \operatorname{sign}(\rho_{xz}) \cdot \Delta z \,\big) \\ (\mathrm{d}x, \mathrm{d}y, \mathrm{d}z)_{\mathrm{down}} \quad &= \quad \big(-\Delta x, \; -\operatorname{sign}(\rho_{xy}) \cdot \Delta y, \; -\operatorname{sign}(\rho_{xz}) \cdot \Delta z \,\big) \; . \end{aligned} \tag{4.37}$$

With this choice of realignment of the fourteen transitions out of the central node, we can now write the effective diffusitivity coefficients of the three-dimensional fifteen point stencil as

$$\begin{aligned} c_x &= \left(\frac{1}{\Delta x^2} - c_{xy} - c_{xz} - c_{xyz} \right)_+ & c_{yz} &= \frac{|\rho_{yz}|}{\Delta y \Delta z} - c_{xyz} \\ c_y &= \left(\frac{1}{\Delta y^2} - c_{xy} - c_{yz} - c_{xyz} \right)_+ & c_{xz} &= \frac{|\rho_{xz}|}{\Delta x \Delta z} - c_{xyz} \\ c_z &= \left(\frac{1}{\Delta z^2} - c_{xz} - c_{yz} - c_{xyz} \right)_+ & c_{xy} &= \frac{|\rho_{xy}|}{\Delta x \Delta y} - c_{xyz} \end{aligned} \tag{4.38}$$

$$c_{xyz} \quad = \quad \min\left(\frac{|\rho_{yz}|}{\Delta y \Delta z}, \frac{|\rho_{xz}|}{\Delta x \Delta z}, \frac{|\rho_{xy}|}{\Delta x \Delta y} \right) \cdot \mathbf{1}_{\{\rho_{xy} \cdot \rho_{xz} \cdot \rho_{yz} > 0\}} \; .$$

We mention that in the homogenous case $\Delta x = \Delta y = \Delta z =: \Delta$ and $|\rho_{xy}| = |\rho_{xz}| = |\rho_{yz}| =: \rho$, all of the bi-axial transition coefficients vanish, i.e.,

$$c_{xy} \quad = \quad c_{xz} \quad = \quad c_{yz} \quad = \quad 0 \; , \tag{4.39}$$

and all of the mono-axial coefficients become

$$c_x \quad = \quad c_y \quad = \quad c_z \quad = \quad \hat{c} - c_{xyz} \quad \geq \quad 0 \tag{4.40}$$

with

$$\hat{c} \quad = \quad \frac{1}{\Delta^2} \qquad \text{and} \qquad c_{xyz} \quad = \quad \rho \cdot \hat{c} \; . \tag{4.41}$$

In a stochastic interpretation of the discrete dynamics as a Markov chain, this transition to the cube's principal corners can be seen as a *common move of all underlyings* with transition intensity

$$\lambda_{\mathrm{common}} := \rho \cdot \hat{c}/2 \; , \tag{4.42}$$

and the three mono-axial transition directions can be identified as idiosyncratic moves each with transition intensity

$$\lambda_{\mathrm{idio}} := \hat{c}/2 - \lambda_{\mathrm{common}} \; . \tag{4.43}$$

As a point of interest, we note that *this representation is identical with the definition of a Marshall-Olkin copula [ELM01] in a three-dimensional nearest-neighbour transition discretisation*.

Our final word on the fifteen point stencil is on its transition behaviour in the limiting case of perfect codependence when, say, $\Delta x > \Delta y > \Delta z$. In complete analogy to the discussion in section 4.2.1.1, we can show that in this situation we obtain

$$\mathsf{E}\Big[\mathrm{d}x\,\Big|\,\mathrm{d}z = +\Delta z\Big] \;=\; \mathsf{E}\Big[\mathrm{d}y\,\Big|\,\mathrm{d}z = +\Delta z\Big] \;=\; \Delta z \tag{4.44}$$

as we would hope. In other words, *in expectation* the fifteen point stencil preserves the three-dimensional transport diagonal in the limiting case of perfect correlation when the lattice discretisations in the three dimensions (each relative to their respective volatility) differ.

4.2.3 Higher-dimensional cross-diffusion stencils

In principle, it is possible to drive the inclusion of additional hypercube nodes to higher dimensions, leading to further free parameters that can be chosen to match the non-negativity condition (4.31). In appendix B, we give symbolic algebra code for the derivation of the relationships between the mono- and bi-axial coefficients c_i and c_{ij} to the remaining free parameters $c_{\ldots}$ in arbitrary dimensionalities. In four dimensions, for instance, we obtain for the coefficients of

$$\begin{aligned}\tilde{L} \;=\;& c_x\tilde{L}_x + c_y\tilde{L}_y + c_z\tilde{L}_z c_\omega\tilde{L}_\omega \\ &+ c_{xy}\tilde{L}_{xy} + c_{yz}\tilde{L}_{yz} + c_{xz}\tilde{L}_{xz} + c_{x\omega}\tilde{L}_{x\omega} + c_{y\omega}\tilde{L}_{y\omega} + c_{z\omega}\tilde{L}_{z\omega} \\ &+ c_{xyz}\tilde{L}_{xyz} + c_{xy\omega}\tilde{L}_{xy\omega} + c_{yz\omega}\tilde{L}_{yz\omega} + c_{xz\omega}\tilde{L}_{xz\omega} + c_{xyz\omega}\tilde{L}_{xyz\omega}\end{aligned} \tag{4.45}$$

the following generic formulae (before flooring)

$$\begin{aligned}
c_{xy} &= \frac{\rho_{xy}}{\Delta x\Delta y} - c_{xyz} - c_{xz\omega} - c_{xyz\omega} & c_{yz} &= \frac{\rho_{yz}}{\Delta y\Delta z} - c_{xyz} - c_{yz\omega} - c_{xyz\omega} \\
c_{xz} &= \frac{\rho_{xz}}{\Delta x\Delta z} - c_{xyz} - c_{xz\omega} - c_{xyz\omega} & c_{x\omega} &= \frac{\rho_{x\omega}}{\Delta x\Delta\omega} - c_{xy\omega} - c_{xz\omega} - c_{xyz\omega} \\
c_{y\omega} &= \frac{\rho_{y\omega}}{\Delta y\Delta\omega} - c_{xy\omega} - c_{yz\omega} - c_{xyz\omega} & c_{z\omega} &= \frac{\rho_{z\omega}}{\Delta z\Delta\omega} - c_{xz\omega} - c_{yz\omega} - c_{xyz\omega} \\
c_x &= \frac{1}{\Delta x^2} - c_{xy} - c_{xz} - c_{x\omega} - c_{xyz} - c_{xz\omega} - c_{xy\omega} - c_{xyz\omega} \\
c_y &= \frac{1}{\Delta y^2} - c_{xy} - c_{yz} - c_{y\omega} - c_{xyz} - c_{yz\omega} - c_{xy\omega} - c_{xyz\omega} \\
c_z &= \frac{1}{\Delta z^2} - c_{xz} - c_{yz} - c_{z\omega} - c_{xyz} - c_{yz\omega} - c_{xz\omega} - c_{xyz\omega} \\
c_\omega &= \frac{1}{\Delta\omega^2} - c_{x\omega} - c_{y\omega} - c_{z\omega} - c_{xy\omega} - c_{yz\omega} - c_{xz\omega} - c_{xyz\omega}
\end{aligned} \tag{4.46}$$

with all other coefficients with three or more subscript indices being free subject to (4.31). Obviously, a judicious choice of the free parameters to ensure minimal flooring is somewhat complicated since it essentially poses a linear programming problem. Also, the selection logic of the principal axis that was (4.37) in three dimensions requires more thought, though, we shall not go into this any further at this point.

4.3 Log-spot drift corrections

In most financial applications, and exposure calculations are no exception, the dominant criterion for accuracy of any calculation is the matching of par strikes of forward contracts. In more abstract terminology, it is the requirement that certain financial observables remain (deflated) martingales within the chosen measure.

In finite-differencing implementations, this translates into a requirement of zero bias as we propagate in time. In a diffusive model, when a transformation from a spot variable S to its logarithm,

$$x \;:=\; \ln(S)\,, \tag{4.47}$$

is involved for numerical efficiency, we incur an advection term that we all know and love in the form

$$-\tfrac{1}{2}\sigma^2 \tag{4.48}$$

due to *Itō's lemma*. To compensate for this analytically, the log-spot transformation is therefore also sometimes chosen as

$$S(t) \;=\; \mathrm{e}^{-\frac{1}{2}\sigma^2 t + x(t)} \quad\Longleftrightarrow\quad x(t) \;=\; \ln(S(t)) + \tfrac{1}{2}\sigma^2 t\,. \tag{4.49}$$

This, however, *is only exact in continuous space*. When we are dealing with a spatial discretisation $\tilde{L}$ of the generator, there is a discrepancy that is typically proportional to Δx^2. This discrepancy is usually ignored by practitioners of finite-differencing methods, but that's a luxury we cannot afford when we are operating with ultra-sparse discretisations. In the following sections, we will outline how we can compensate exactly to retain the desired martingale property of the spot S when we have a spatial discretisation of the log-spot.

4.3.1 Correction with internal drift compensation

We first consider the one-dimensional case when the generator is the spatial discretisation of

$$\mu\partial_x + \tfrac{\sigma^2}{2}\partial_x^2\,. \tag{4.50}$$

and the term $\mu\partial_x$ is to ensure that $S := \mathrm{e}^x$ is a martingale, which is why in a spatially continuous setting we would use $\mu = -\frac{\sigma^2}{2}$. We assume that the first derivative ∂_x is accommodated by the symmetric centre-differencing stencil

$$\partial_x f \;\approx\; \tfrac{1}{2\Delta x}\left[f(x+\Delta x) - f(x+\Delta x)\right] + \mathcal{O}(\Delta x^2)\,, \tag{4.51}$$

and that we use the standard stencil (4.2) for ∂^2/∂_x^2. Crucially, we assume that neither diffusion nor advection is applied on boundary nodes, though, we later make an exception for the term $(r_d - r_f)\partial_S$ in any FX direction. With equidistant spacing of lattice nodes in x, we compute the net drift of S on interior nodes as

$$\frac{\mathsf{E}[\mathrm{d}S]}{S\mathrm{d}t} \;=\; \tilde{\delta}\cdot\left(\mathrm{e}^{\Delta x} - 2 + \mathrm{e}^{-\Delta x}\right) + \tilde{\mu}\cdot\left(\mathrm{e}^{\Delta x} - \mathrm{e}^{-\Delta x}\right) \tag{4.52}$$

where we define

$$\delta \;:=\; \sigma^2\,, \qquad \tilde{\delta} \;:=\; \frac{\delta}{2\Delta x^2} \;=\; \frac{\sigma^2}{2\Delta x^2} \qquad \text{and} \qquad \tilde{\mu} \;:=\; \frac{\mu}{2\Delta x}\,. \tag{4.53}$$

Note that with these definitions, the discrete generator matrix for a one-dimensional calculation is tri-diagonal with each row in the form

$$\tilde{L}_{\cdots} \;=\; \left(\;\cdots\quad\cdots\quad(\tilde{\delta}-\tilde{\mu})\quad -2\tilde{\delta}\quad(\tilde{\delta}+\tilde{\mu})\quad\cdots\quad\cdots\;\right). \tag{4.54}$$

The spot drift (4.52) vanishes, meaning the spot S is a martingale, when we set (see also [ABR97])

$$\tilde{\mu} \;:=\; -\tilde{\delta}\cdot\frac{\mathrm{e}^{\Delta x} - 2 + \mathrm{e}^{-\Delta x}}{\mathrm{e}^{\Delta x} - \mathrm{e}^{-\Delta x}} \;=\; -\tilde{\delta}\cdot\frac{1-\mathrm{e}^{-\Delta x}}{1+\mathrm{e}^{-\Delta x}}\,. \tag{4.55}$$

Expanding this in Δx gives us

$$\mu \;=\; -\frac{\sigma^2}{2}\cdot\left[1 - \tfrac{1}{12}\Delta x^2 + \tfrac{1}{120}\Delta x^4 - \tfrac{17}{20160}\Delta x^6 + \mathcal{O}(\Delta x^8)\right] \tag{4.56}$$

which is, as expected, given by the continuous term in leading order, and corrections starting in order Δx^2.

To obtain a feeling for the magnitude of the error, consider a discretisation according to

$$\Delta x \;=\; \frac{2 \cdot m_{\text{std.dev.}} \cdot \sigma \cdot \sqrt{T}}{n_x - 1} \tag{4.57}$$

with $m_{\text{std.dev.}} = 4$ standard deviations in both directions, $\sigma = 35\%$, $T = 4$, and $n_x = 17$. In that case, we already have $\Delta x = 0.35$, and a relative error in the forward to maturity of about $\mathrm{e}^{-(\sigma^2/2)\cdot(\Delta x^2/12)\cdot T} - 1 \approx -0.25\%$, which is unacceptable, and this is only for a deal of 4 year maturity. In practice, we may need to go out much further, and may need to reduce the number of nodes down to 11 or even 9, rendering the spatially continuous drift compensation term $-\sigma^2/2$ by itself as somewhat academic for commercial finite-differencing implementations.

We mention that, on boundary nodes with absorbing boundary conditions, i.e., where we have no explicit diffusion term (effectively $\sigma = 0$), we also have no (compensatory) advection term ($\mu = 0$) and thus trivially obtain S being a martingale on boundary nodes in this setting.

4.3.2 Correction with external drift compensation

A pure compensation on the lattice as given in (4.55) works very well to preserve the martingale property of the spot variable. For significant volatility or time to maturity, however, it has the effect that a lattice that is laid out along constant x coordinates will incur most of the probability gradually drifting towards one side of the discretisation and possibly be absorbed at that boundary. This can be compensated by making the grid wider, re-boxing methods, and so on. An easy technique, however, to take away most of this drift on the lattice with its accumulation of probability at one end is to use a spot transformation formula based on the analytical compensator term we obtain from spatially and temporally continuous calculations, i.e., instead of $S := \mathrm{e}^x$ we use something like

$$S(t) \;:=\; \mathrm{e}^{-\frac{\sigma^2}{2}t + x(t)} \,. \tag{4.58}$$

As in the previous section, we include compensation coefficients $\tilde{\mu}$ that we add as centre-differencing advection terms. This gives us for the net drift of S on interior nodes

$$\frac{\mathsf{E}[\mathrm{d}S]}{S} \;=\; \mathrm{e}^{-\tilde{\delta}\Delta x^2 \mathrm{d}t}\left[1 + \left(\tilde{\delta}\cdot\left(\mathrm{e}^{\Delta x} - 2 + \mathrm{e}^{-\Delta x}\right) + \tilde{\mu}\cdot\left(\mathrm{e}^{\Delta x} - \mathrm{e}^{-\Delta x}\right)\right)\cdot \mathrm{d}t\right] - 1 \,. \tag{4.59}$$

To first order in $\mathrm{d}t$, we obtain

$$\frac{\mathsf{E}[\mathrm{d}S]}{S\mathrm{d}t} \;=\; \tilde{\delta}\cdot\left(\mathrm{e}^{\Delta x} - 2 + \mathrm{e}^{-\Delta x} - \Delta x^2\right) + \tilde{\mu}\cdot\left(\mathrm{e}^{\Delta x} - \mathrm{e}^{-\Delta x}\right) \,, \tag{4.60}$$

and thus we must set

$$\tilde{\mu} \;:=\; -\tilde{\delta}\cdot\frac{\mathrm{e}^{\Delta x} - 2 + \mathrm{e}^{-\Delta x} - \Delta x^2}{\mathrm{e}^{\Delta x} - \mathrm{e}^{-\Delta x}} \,. \tag{4.61}$$

Expanding this in Δx gives us

$$\mu \;=\; -\frac{\sigma^2}{2}\cdot\left[\tfrac{1}{12}\Delta x^2 - \tfrac{1}{90}\Delta x^4 + \tfrac{73}{60480}\Delta x^6 + \mathcal{O}(\Delta x^8)\right] \tag{4.62}$$

which is of leading order Δx^2 as we would have expected since the leading order $\mathcal{O}(1)$ is externally compensated by the exponential factor.

An interesting extension of this analysis is the case when the external (exponential) compensation term uses a diffusion coefficient δ that is not exactly matched by the effective diffusion δ' on the lattice, such as

can be the case when seven-point stencils are used for cross-diffusions as we mentioned in section 4.2.1.3. In that case, the analysis becomes

$$\frac{\mathsf{E}[\mathrm{d}S]}{S} = \mathrm{e}^{-\tilde{\delta}\Delta x^2 \mathrm{d}t}\left[1 + \left(\tilde{\delta}' \cdot \left(\mathrm{e}^{\Delta x} - 2 + \mathrm{e}^{-\Delta x}\right) + \tilde{\mu} \cdot \left(\mathrm{e}^{\Delta x} - \mathrm{e}^{-\Delta x}\right)\right) \cdot \mathrm{d}t\right] - 1 \tag{4.63}$$

$$\frac{\mathsf{E}[\mathrm{d}S]}{S\mathrm{d}t} = \tilde{\delta}' \cdot \left(\mathrm{e}^{\Delta x} - 2 + \mathrm{e}^{-\Delta x}\right) - \tilde{\delta}\Delta x^2 + \tilde{\mu} \cdot \left(\mathrm{e}^{\Delta x} - \mathrm{e}^{-\Delta x}\right) \ , \tag{4.64}$$

In order to be able to handle gracefully the case when either of Δx and $(\tilde{\delta}' - \tilde{\delta})$ are very small or even identically zero, we represent the solution for $\tilde{\mu}$ as

$$\tilde{\mu} := -(\tilde{\delta}' - \tilde{\delta}) \cdot \frac{\mathrm{e}^{\Delta x} - 2 + \mathrm{e}^{-\Delta x}}{\mathrm{e}^{\Delta x} - \mathrm{e}^{-\Delta x}} - \tilde{\delta} \cdot \frac{\mathrm{e}^{\Delta x} - 2 + \mathrm{e}^{-\Delta x} - \Delta x^2}{\mathrm{e}^{\Delta x} - \mathrm{e}^{-\Delta x}} \tag{4.65}$$

which can be implemented using the expansions already mentioned in (4.56) and (4.62) when Δx is very small. The advantage of the above segregated formulation is that it allows falling back exactly without any round-off error to the higher order case (4.61) when

$$\tilde{\delta}' \equiv \tilde{\delta} \tag{4.66}$$

which is why we advocate its use to ensure that the spot S is a martingale in this setting.

4.3.2.1 Boundary nodes

Now we consider the case of the externally applied exponential compensation on a boundary node on which we have no lattice diffusivity. Here, we compensate the externally generated drift with an inward-differencing advection term. On the lower boundary in x, we find

$$\frac{\mathsf{E}[\mathrm{d}S]}{S} = \mathrm{e}^{-\tilde{\delta}\Delta x^2 \mathrm{d}t}\left[1 + \tilde{\mu} \cdot \left(\mathrm{e}^{\Delta x} - 1\right) \cdot \mathrm{d}t\right] - 1 \tag{4.67}$$

$$\frac{\mathsf{E}[\mathrm{d}S]}{S\mathrm{d}t} = -\tilde{\delta}\Delta x^2 + \tilde{\mu} \cdot \left(\mathrm{e}^{\Delta x} - 1\right) \ , \tag{4.68}$$

and hence we must have

$$\tilde{\mu}_{\text{lower boundary}}(\Delta x) := \tilde{\delta} \cdot \frac{\Delta x^2}{\mathrm{e}^{\Delta x} - 1} \tag{4.69}$$

for the spot S to be a martingale on lower boundary nodes without explicit lattice diffusivity.

On the upper boundary, inward-differencing for the advection effectively means backward-differencing. We obtain

$$\frac{\mathsf{E}[\mathrm{d}S]}{S} = \mathrm{e}^{-\tilde{\delta}\Delta x^2 \mathrm{d}t}\left[1 + \tilde{\mu} \cdot \left(1 - \mathrm{e}^{-\Delta x}\right) \cdot \mathrm{d}t\right] - 1 \tag{4.70}$$

$$\frac{\mathsf{E}[\mathrm{d}S]}{S\mathrm{d}t} = -\tilde{\delta}\Delta x^2 + \tilde{\mu} \cdot \left(1 - \mathrm{e}^{\Delta - x}\right) \tag{4.71}$$

and

$$\tilde{\mu}_{\text{upper bondary}}(\Delta x) := \tilde{\delta} \cdot \frac{\Delta x^2}{1 - \mathrm{e}^{-\Delta x}} \tag{4.72}$$

$$= -\tilde{\mu}_{\text{lower boundary}}(-\Delta x) \tag{4.73}$$

as the upper boundary node martingale condition.

4.3.3 Effect on the convergence of temporal integration schemes

As it turns out, the drift compensation terms we computed in the previous sections have a different impact on first and second order explicit finite-differencing schemes as approximations to the exact continuous-time propagator

$$\tilde{A}(\Delta t) \;=\; \mathrm{e}^{\Delta t \cdot \tilde{L}} \tag{4.74}$$

given by

$$\begin{aligned} \tilde{A}_1(\Delta t) &= 1 + \Delta t \cdot \tilde{L} \\ \tilde{A}_2(\Delta t) &= 1 + \Delta t \cdot \tilde{L} + \tfrac{1}{2}\Delta t^2 \cdot \tilde{L}^2 \; . \end{aligned} \tag{4.75}$$

Note that the first and second order fully implicit scheme propagators can formally be expressed as

$$\begin{aligned} \tilde{A}_1^{\text{implicit}}(\Delta t) &= \tilde{A}_1(-\Delta t)^{-1} \\ \tilde{A}_2^{\text{implicit}}(\Delta t) &= \tilde{A}_2(-\Delta t)^{-1} \; . \end{aligned} \tag{4.76}$$

In the following, we will focus solely on fully explicit schemes, in particular with a view on higher dimensional applications, though we mention that the net drift behaviour for implicit schemes can be readily derived from the explicit schemes by the aid of the reciprocity relationships (4.76). As before, in all comparisons in this section we use a one-dimensional homogoneous discretisation for our analysis and assume that no diffusivity is applied on boundary nodes.

4.3.3.1 Case of internal drift compensation

Since we have no diffusivity on boundary nodes, the drift compensation term $\tilde{\mu}$ from (4.55) in section 4.3.1 is only to be part of the generator matrix $\tilde{L}$ on interior rows. With this, we find that, by construction,

$$\tilde{L} \cdot \boldsymbol{X} \;=\; 0 \tag{4.77}$$

where

$$X_i \;=\; \mathrm{e}^{i \cdot \Delta x} \; . \tag{4.78}$$

As a consequence, all of the first and second order schemes, both explicit and implicit, trivially satisfy the martingale condition

$$A \cdot \boldsymbol{X} \;=\; \boldsymbol{X} \; . \tag{4.79}$$

4.3.3.2 Case of external drift compensation

Due to its practical importance, we look in more detail at the effect of log-spot drift corrections when the translation from the state variable x to a spot variable S involves an analytical compensation term $\mathrm{e}^{-\frac{\sigma^2}{2}t}$ as given in (4.58) described in section 4.3.2. In particular, we compute the effective residual instantaneous drift

$$\eta_i \;:=\; \frac{\mathsf{E}\Big[\, S(t+\Delta t)|_{S(t)=S_i} \Big]}{S_i} - \; 1 \tag{4.80}$$

of the spot S on node #i out of n nodes over a time step Δt for six different scenarios[11].

[11] We mention that one can in principle extend the analysis to include an arbitrary additional drift term νt, e.g., to accom-

First order scheme $A_1(\Delta t)$ without compensation: $\tilde{\mu} \equiv 0$ on all nodes. We rely solely on the external drift compensation, i.e., on the multiplicative factor $\mathrm{e}^{-\frac{\sigma^2}{2}t}$ in (4.58). We obtain

$$\begin{aligned} \eta_1 \;=\; \eta_n \;&=\; -\tfrac{1}{2}\delta\Delta t && + \;\; \mathcal{O}(\Delta t^2) \\ \eta_i \;&=\; -\tfrac{1}{2}\left[\left(\delta - \delta'\right) - \tfrac{1}{12}\delta' \cdot \Delta x^2\right]\Delta t && + \;\; \mathcal{O}(\Delta t^2) \qquad \forall\, i \,:\, 1 < i < n \end{aligned} \tag{4.83}$$

with $\delta \equiv \sigma^2$ being the instantaneous diffusivity in the non-discretised setting, and δ' the net diffusivity on the lattice. Obviously, in a one-dimensional setting there is no reason why δ and δ' should differ since there is no cross-diffusion stencil to give rise to any discrepancy, but we keep the two terms separate to see the effect we obtain in higher dimensions. We can see that the boundary nodes' external compensation is effectively wrong and introduces a drift error of order $\mathcal{O}(\Delta t)$ on the boundary. We emphasize that, for ultra-sparse lattices with, say, only 11 levels per spatial dimension, this effect is significant and should not be ignored.

Interior nodes are correctly compensated to order $\Delta x^2 \Delta t$ if the generator's internal nodes' diffusivity δ' matches the external compensator's δ, but in general will appear to have a finite bias over n_T steps each of length $\Delta t := T/n_T$, which also has a non-negligible deterioration effect on the net outcome. In the more benign case when $\delta = \delta'$, on interior nodes over n_T steps this bias is to leading order

$$\left[1 + \tfrac{1}{24}\Delta x^2 \left(\delta \frac{T}{n_T}\right) - \tfrac{1}{8}\left(\delta \frac{T}{n_T}\right)^2\right]^{n_T} - 1\,. \tag{4.84}$$

First order scheme $A_1(\Delta t)$ with compensation on interior nodes: $\tilde{\mu}$ as in (4.65) *only* on interior nodes. This scenario gives us

$$\begin{aligned} \eta_1 \;=\; \eta_n \;&=\; -\tfrac{1}{2}\delta\Delta t && + \;\; \mathcal{O}(\Delta t^2) \\ \eta_i \;&=\; -\tfrac{1}{8}\delta\Delta t^2 && + \;\; \mathcal{O}(\Delta t^3) \qquad \forall\, i \,:\, 1 < i < n \end{aligned} \tag{4.85}$$

Boundary nodes still incur a drift term of order Δt but interior nodes are correct up to order $o(\Delta t^2)$. The integration scheme as a whole, over n_T steps each of length $\Delta t := T/n_T$, depends on the degree to which the boundary nodes contribute. A rule of thumb for the assessment of this contribution is to estimate the probability of the boundary being attained. As long as boundary nodes are essentially not attainable (for instance, due to the overall lattice width), the scheme will appear to converge like $\sim 1/n_T$.

First order scheme $A_1(\Delta t)$ with full compensation: $\tilde{\mu}$ as in (4.65) on interior nodes, and (4.69) and (4.72) on boundary nodes. Here, we have

$$\eta_i \;=\; -\tfrac{1}{8}\delta\Delta t^2 \quad + \;\; \mathcal{O}(\Delta t^3) \qquad \forall\, i \,:\, 1 \le i \le n \tag{4.86}$$

---

modate quanto effects or a deterministic FX spot drift, as in

$$S(t) \;:=\; \mathrm{e}^{-\frac{\sigma^2}{2}t + \nu t + x(t)} \tag{4.81}$$

in which case we'd define the residual drift as

$$\eta_i \;:=\; \frac{\mathsf{E}\left[\, S(t+\Delta t)|_{S(t)=S_i}\right]}{S_i} - \mathrm{e}^{\nu \Delta t} \tag{4.82}$$

but we will limit ourselves to the mere inclusion of the Itô term $-\frac{\sigma^2}{2}t$ here.

The scheme as a whole, over n_T steps each of length $\Delta t := T/n_T$, converges like $\sim 1/n_T$ with net bias to leading order

$$\left[1 - \tfrac{1}{8}\left(\delta\frac{T}{n_T}\right)^2\right]^{n_T} - 1\,. \tag{4.87}$$

Not surprisingly, we recommend this approach when a spot variable of the form $S(t) = \mathrm{e}^{-\frac{\sigma^2}{2}t + x(t)}$ is required to be a martingale.

Second order scheme $A_2(\Delta t)$ without compensation: $\tilde{\mu} \equiv 0$ on all nodes. For the second order explicit scheme, we need to show more than just the leading term to identify the respective point of interest. The analysis yields

$$\begin{aligned}
\eta_1 \;=\; \eta_n \;&=\; -\tfrac{1}{2}\delta\Delta t + \tfrac{1}{8}\delta^2\Delta t^2 && +\;\; \mathcal{O}(\Delta t^3)\\
\eta_2 \;=\; \eta_{n-1} \;&=\; -\tfrac{1}{2}\left[(\delta - \delta') - \tfrac{1}{12}\delta'\cdot\Delta x^2 + \mathcal{O}(\Delta x^4)\right]\Delta t\\
&\quad -\;\tfrac{1}{8}\delta'^2\frac{\left[1 - \Delta x + \mathcal{O}(\Delta x^2)\right]}{\Delta x^2}\Delta t^2 && +\;\; \mathcal{O}(\Delta t^3)\\
\eta_i \;&=\; -\tfrac{1}{2}\left[(\delta - \delta') - \tfrac{1}{12}\delta'\cdot\Delta x^2 + \mathcal{O}(\Delta x^4)\right]\Delta t\\
&\quad +\;\tfrac{1}{8}(\delta' - \delta)\left[(\delta' - \delta) + \tfrac{1}{6}\delta'\Delta x^2 + \mathcal{O}(\Delta x^4)\right]\Delta t^2 && +\;\; \mathcal{O}(\Delta t^3)
\end{aligned} \tag{4.88}$$

for $2 < i < n-1$. We obtain essentially the same behaviour as for the first order scheme, highlighting that a second order scheme is a waste of time unless we compensate the external drift application. Even more worryingly, we notice the occurrence of a term that is of second order in Δt but reciprocal in Δx^2 not on the boundary nodes, but on those adjacent to the boundary nodes! Depending on their probability of being attained, this can cause a noteworthy source of bias, further emphasizing that higher order schemes ought to be combined with log-spot drift compensation terms. When $\delta = \delta'$, on interior nodes away from the boundary over n_T steps the bias is to leading order

$$\left[1 + \tfrac{1}{24}\Delta x^2\left(\delta\frac{T}{n_T}\right) - \tfrac{1}{48}\left(1 + \frac{\Delta x^2}{2}\right)\left(\delta\frac{T}{n_T}\right)^3\right]^{n_T} - 1\,. \tag{4.89}$$

Second order scheme $A_2(\Delta t)$ with compensation on interior nodes: $\tilde{\mu}$ as in (4.65) *only* on interior nodes. We arrive at

$$\begin{aligned}
\eta_1 \;=\; \eta_n \;&=\; -\tfrac{1}{2}\delta\Delta t + \tfrac{1}{8}\delta^2\Delta t^2 && +\;\; \mathcal{O}(\Delta t^3)\\
\eta_2 \;=\; \eta_{n-1} \;&=\; -\tfrac{1}{8}\delta\frac{\left[\delta' - \tfrac{1}{2}(\delta + \delta')\Delta x + \mathcal{O}(\Delta x^2)\right]}{\Delta x^2}\Delta t^2 && +\;\; \mathcal{O}(\Delta t^3)\\
\eta_i \;&=\; -\tfrac{1}{48}\delta^3\Delta t^3 && +\;\; \mathcal{O}(\Delta t^4)
\end{aligned} \tag{4.90}$$

for $2 < i < n-1$. We still have a term that is of second order in Δt but reciprocal in Δx^2 on the nodes adjacent to the boundary, but now this is the leading term as we have no term of order $\mathcal{O}(\Delta t)$ on those nodes.

Second order scheme $A_2(\Delta t)$ with full compensation: $\tilde{\mu}$ as in (4.65) on interior nodes, and (4.69) and (4.72) on boundary nodes. Here, we have

$$\eta_i \;=\; -\tfrac{1}{48}\delta\Delta t^3 \quad +\;\; \mathcal{O}(\Delta t^4) \qquad \forall\, i\,:\, 1 \le i \le n \tag{4.91}$$

The scheme as a whole, over n_T steps each of length $\Delta t := T/n_T$, converges like $\sim 1/n_T^2$ with leading order bias

$$\left[1 - \tfrac{1}{48}\left(\delta\frac{T}{n_T}\right)^3\right]^{n_T} - 1\,. \tag{4.92}$$

Conclusion from the above six scenarios. Having seen the results of the error analysis for all six mentioned scenarios, we can clearly state that the correct handling of drift compensation terms has the expected benefit. More interesting is though that neglecting boundary terms leads to consequences for the second order explicit sheme that we did not anticipate, and remind ourselves of the importance of a formal error analysis.

Numerical examples. We show in figure 8 a numerical comparison of the relative bias of an FX forward for $T = 10$, $\sigma = 20\%$, and $n_T = 120$ as a function of n_x when rolled back with the first order fully explicit scheme using the three different levels of log-spot drift correction in conjunction with the external drift compensation (4.58). The spatial discretisation Δx was set according to (4.57) with $m_{\text{std.dev.}} = 5$.

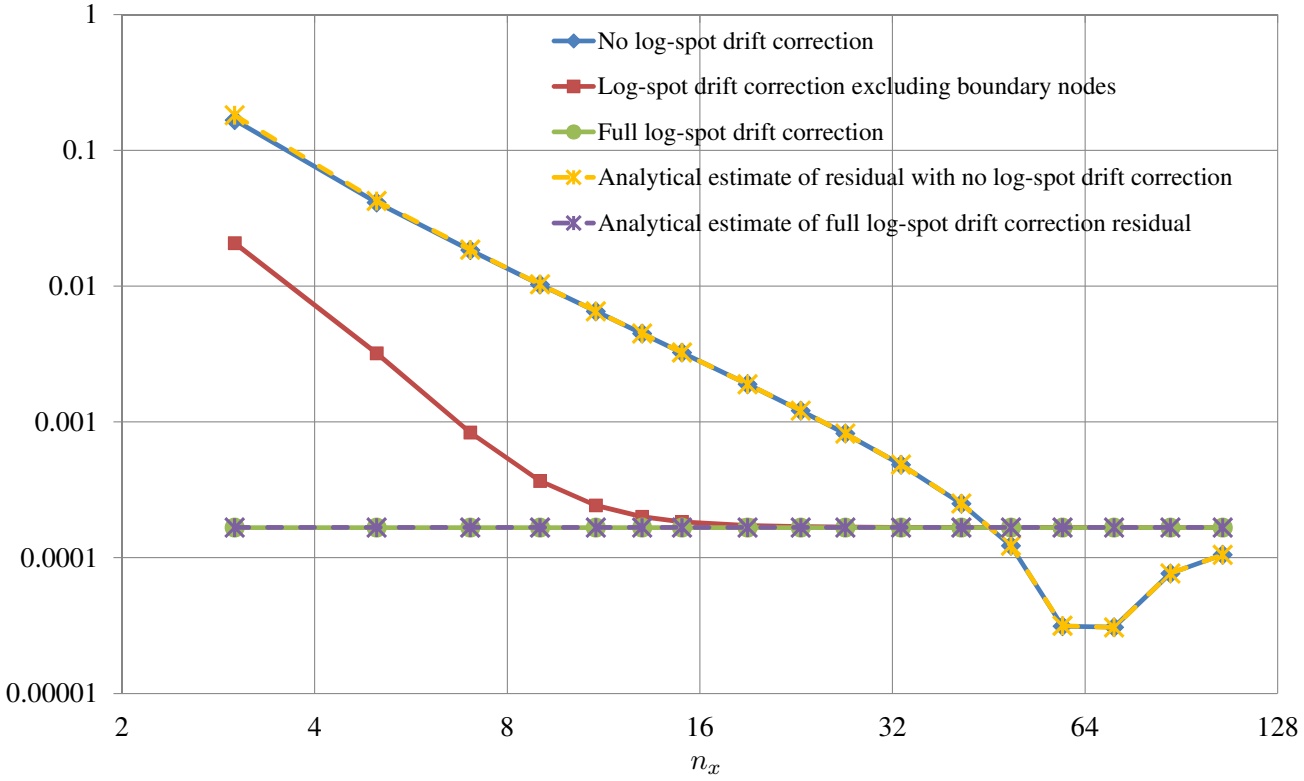


FIGURE 8: Net bias (in absolute value) to $T = 10$ of the first order fully explicit scheme when externally compensated according to (4.58) for the three different levels of log-spot drift correction. Note that the analytical estimates are of absolute magnitude and not a coefficient regression.

The analytical estimates are taken from formulae (4.84) and (4.87). The downward dip of the uncorrected curve (labelled "No log-spot drift correction") below the fully corrected flat line (labelled "Full log-spot drift correction") is caused by a crossing over of the positive term $\sim \Delta x^2 \Delta t$ with the negative term $\sim \Delta t^3$ in (4.84): the dipping curve returns to converge to the flat from below the line for large n_x. We show for comparison the much smaller net bias of the second order scheme for the same parameters in figure 9 with its analytical bias estimates given by (4.89) and (4.92). We can see in these figures that even with monthly steps to a maturity of 10 years we could effectively use as few spatial nodes as 3 with the second order explicit scheme if our sole concern were only the bias of the spot process. Nevertheless, it is probably clear

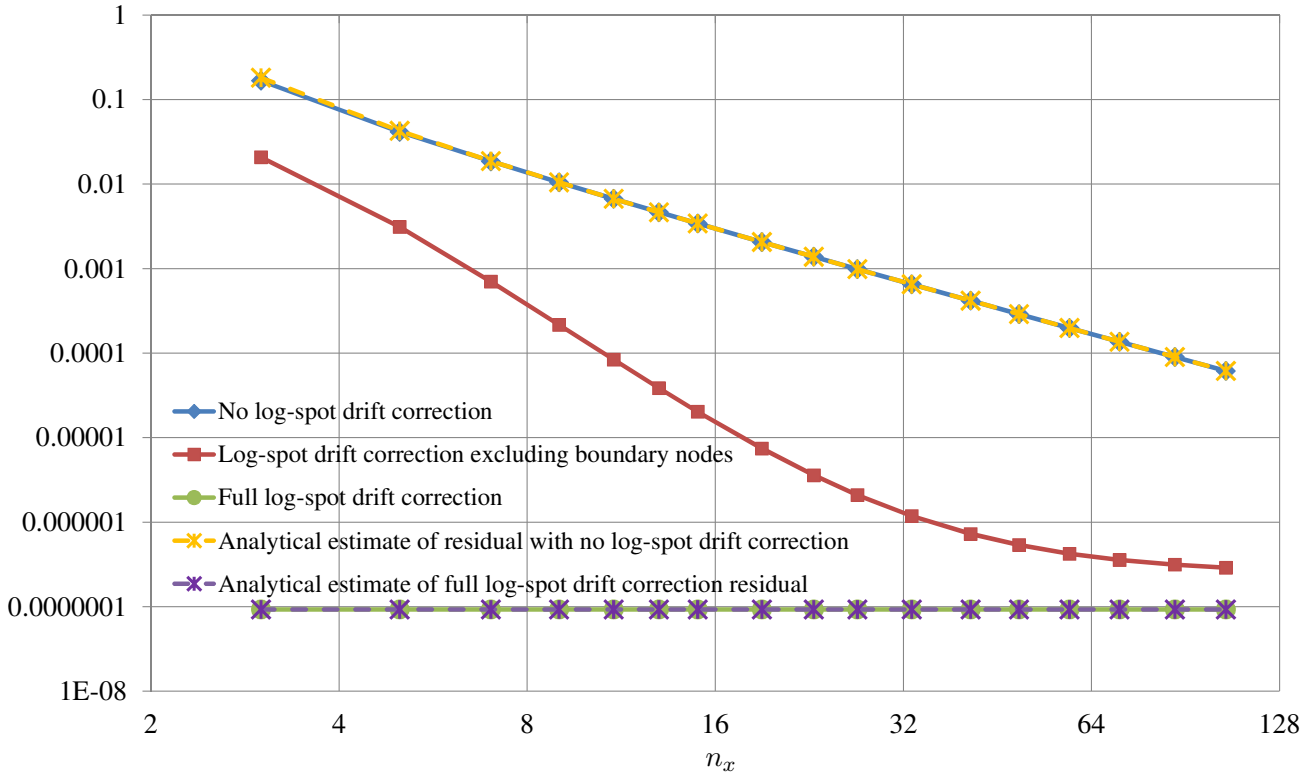


FIGURE 9: Net bias (in absolute value) to $T = 10$ of the second order fully explicit scheme when externally compensated according to (4.58) for the three different levels of log-spot drift correction. Note that the analytical estimates are of absolute magnitude and not a coefficient regression.

that the log-spot drift correction terms we derived enable calculations with finite-differencing methods on *ultra-sparse* lattice discretisations. For low dimensionalities, this makes the finite-differencing method a fast and accurate numerical tool for exposure calculations that is difficult to match with Monte Carlo techniques. This, in particular, when further analysis is desired that depends on a reasonble value distribution granularity with good accuracy on the value on each sample — something that is virtually impossible to achieve with regression methods inside a Monte Carlo simulation framework.

5 Practical examples for Expected Positive Exposure (EPE)

We show in figures 10 to 15 the expected positive exposure EPE(t) for 6 trade examples. Note that EPE(t) is in terms of today's net present value for all time horizons t which enables a like-for-like comparison irrespective of discounting differences in respective currencies. On the left hand side, we show the data from our forward-backward calculations, and on the right hand side diagrams generated by commercial vendor software. Our finite-differencing time lines are computed and shown over the union of monthly time steps and all fixing dates and payment dates. All vendor data are generated with 10,000 paths and monthly time steps. Further details of the calculations are given in tables 1 and 2.

6 Conclusion

The purpose of this article was to illustrate that finite-differencing methods can be used for a significant range of applications that most practitioners consider the exclusive realm of Monte Carlo simulation frameworks. The application in focus in this context was the computation of CVA, DVA, EPE and other XVA numbers.

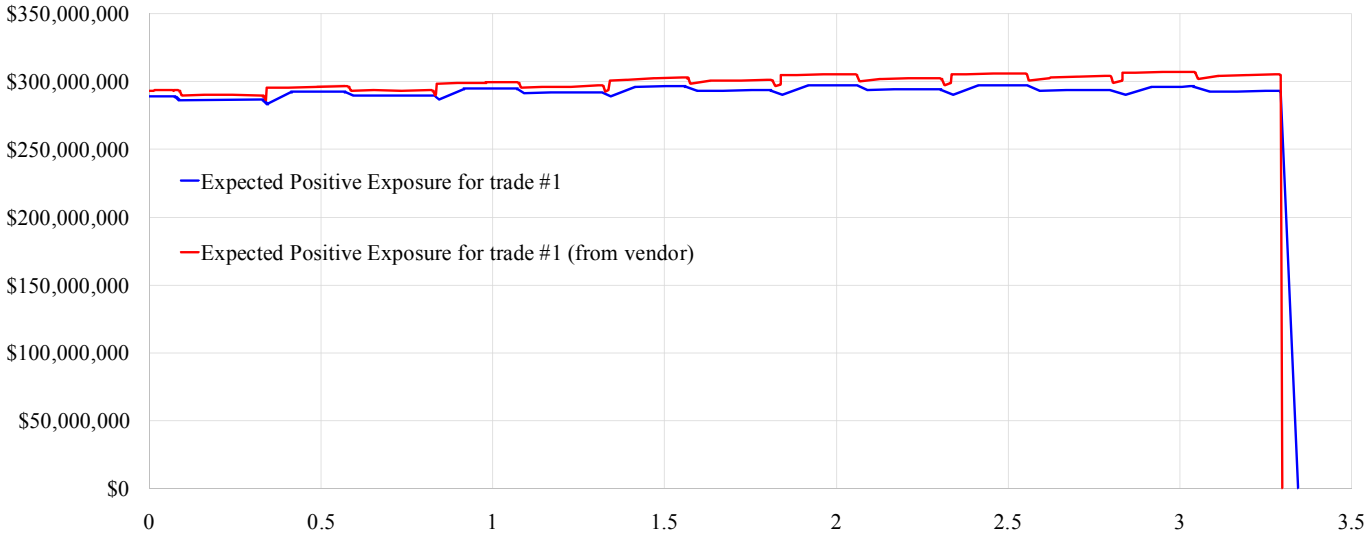


FIGURE 10: $EPE(t)$ for trade #1 over maturity in years.

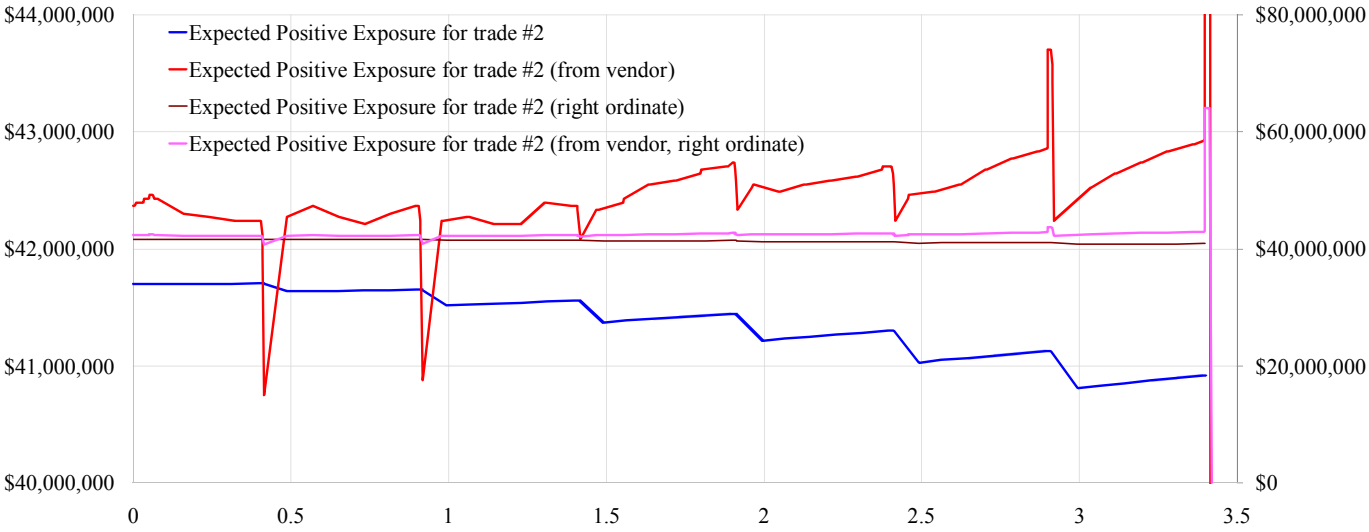


FIGURE 11: $EPE(t)$ for trade #2 over maturity in years. The spike towards the end is due to an error in the vendor's trade schedule. Note the (harmless) ragged vendor profile caused by Monte Carlo simulation "noise" (red line).

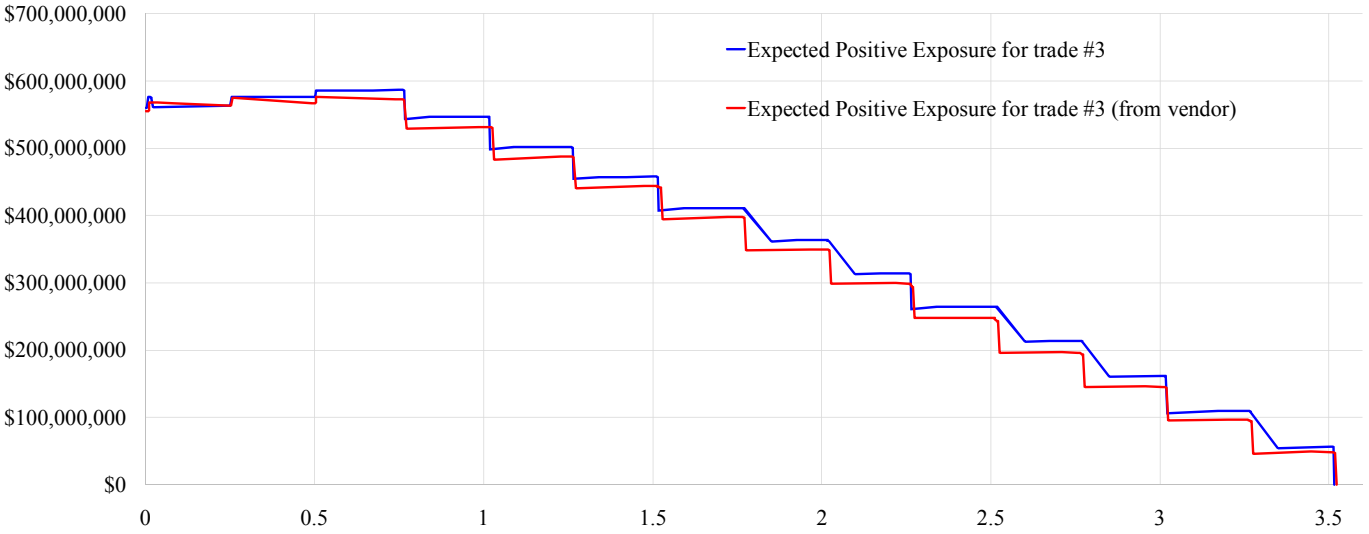


FIGURE 12: $EPE(t)$ for trade #3 over maturity in years.

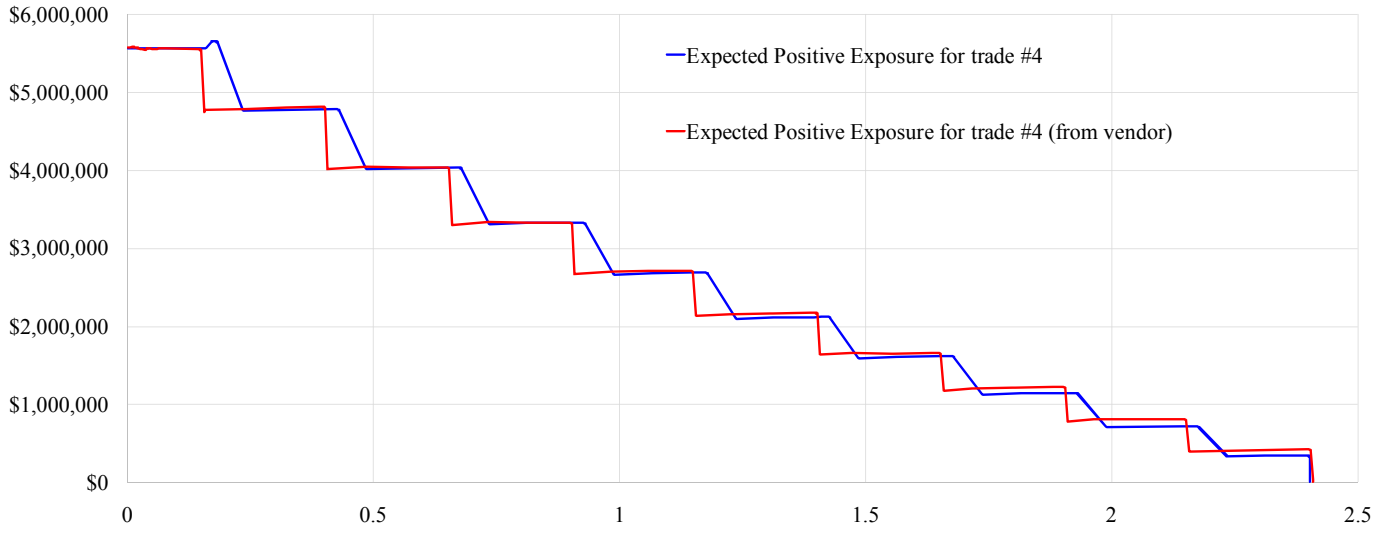


FIGURE 13: $EPE(t)$ for trade #4 over maturity in years.

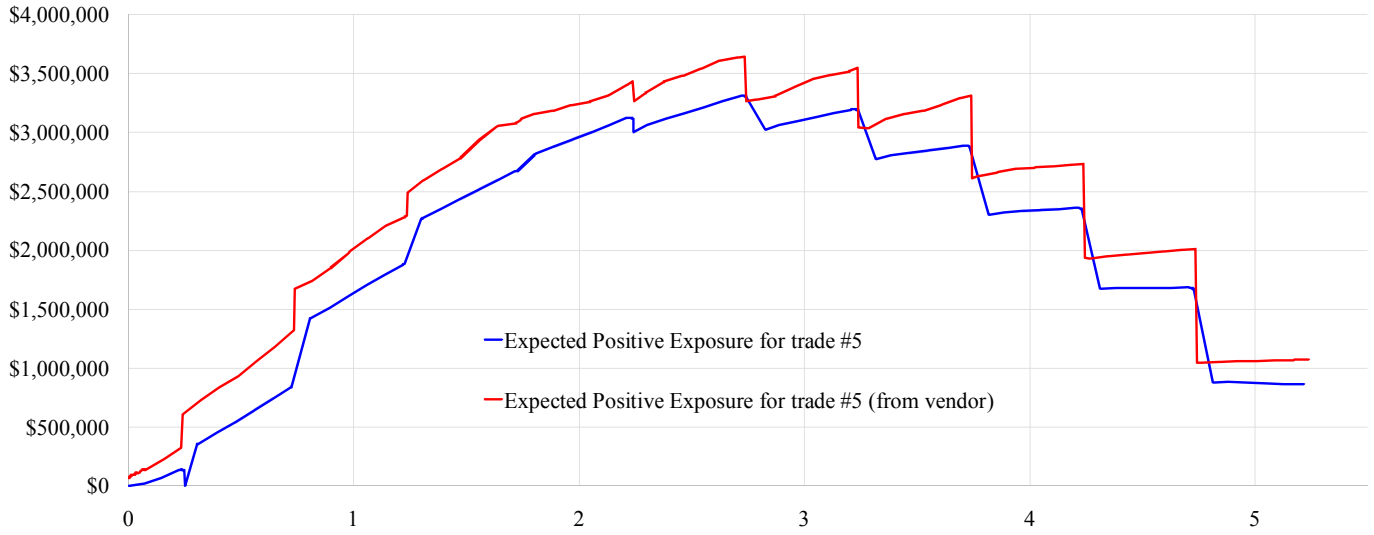


FIGURE 14: $EPE(t)$ for trade #5 over maturity in years. Here, too, we see some (harmless) Monte Carlo simulation "noise" in the vendor profile (red line).

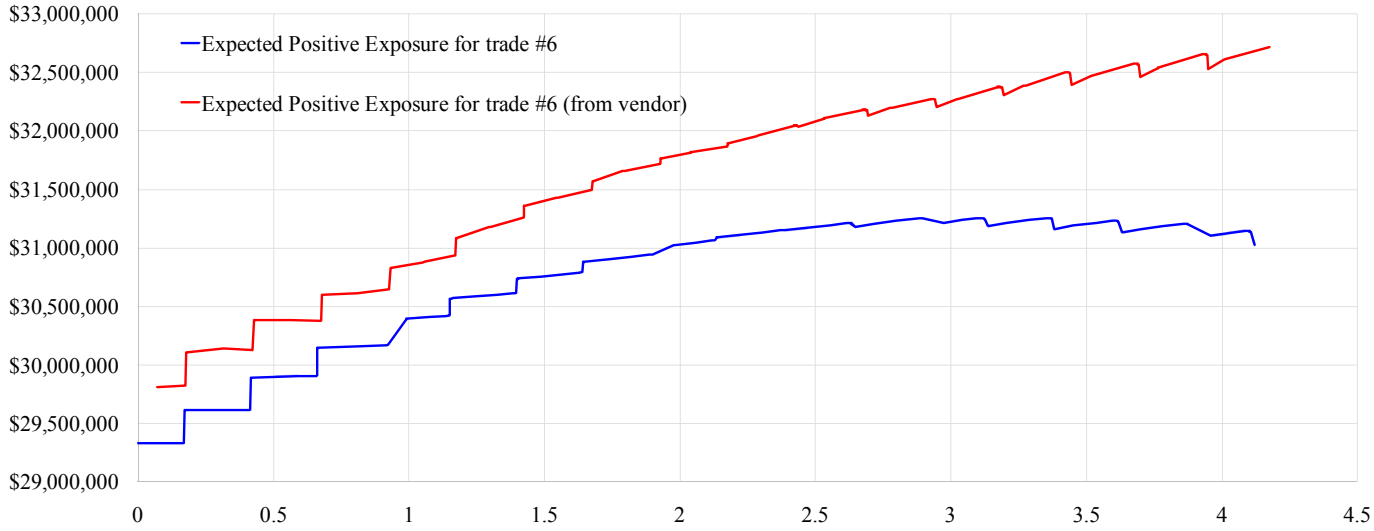


FIGURE 15: $EPE(t)$ for trade #6 over maturity in years.

		Forward/backward fin. diff.		Vendor	
Trade	Description	Trade value	Accuracy	Trade value	Accuracy
#1	(USD_3M_LIBOR+3.2%) - 8% 6M XXX XCCY swap with final notional exchange	287,742,477	0.001%	288,754,798	0.35%
#2	3.8% 6M USD - 8.3% 6M XXX XCCY swap with full notional exchanges	41,700,748	0.001%	41,861,437	0.39%
#3	4.8% 3M USD - (XXX_3M_IBOR+2.6%) XCCY, many amortization steps, final notional exchange	557,450,326	0.002%	552,458,841	-0.89%
#4	2.6% 3M USD - USD_3M_LIBOR	5,553,121	-0.002%	5,688,730	2.44%
#5	5.6% 6M USD - (USD_6M_LIBOR+3.6%)	-6,167,721	-0.003%	-7,069,249	14.61%
#6	USD_3M_LIBOR+2.5% - 10.4% 3M XXX XCCY swap with full notional exchanges	29,324,597	0.007%	29,444,366	0.42%
		RMS:	0.005%	RMS:	5.45%

Table 1: Comparison across 6 trades. All values are in USD. The accuracy is the relative difference of the finite-differencing trade value to the analytical trade value. All forward/backward finite-differencing calculations were done with 31 nodes per dimension. Market data are from an undisclosed date that is of no significance (other than that Libor was still available, of course). XXX represents an undisclosed secondary currency.

		Forward/backward fin. diff.			Vendor		
Trade	CDS	CVA	Time	Memory	CVA	Time	Memory
#1	7.5%	64,130,044	0.42	66	64,939,658	2.76	3,400
#2	8.2%	10,198,507	0.19	27	10,169,871	1.88	1,500
#3	7.8%	89,662,356	0.47	77	90,409,855	3.15	3,600
#4	9.8%	558,960	0.03	3	597,849	0.72	400
#5	8.0%	653,170	0.05	3	663,540	1.25	200
#6	8.0%	8,351,819	0.54	97	8,576,303	2.39	3,800
Total			2.02			19.17	

Table 2: Comparison across 6 trades. All values are in USD. The CDS is the (flat) par quote of the counterparty to maturity of the trade, ignoring any amortization weighting. The time is calculation time in seconds. All forward/backward finite-differencing calculations were done with 31 nodes per dimension. Memory is in MB.

We highlighted that a number of complementary approximations and techniques are required to bring such a finite-differencing-based framework successfully into production. First, we need to make some compromises on the precise representation of product payouts on their, strictly speaking, strong path dependence. We have argued that this is acceptable and given numerical evidence for it. Second, it is imperative that all registers are pulled to make the chosen model's implementation within a finite-differencing framework accurate within a sparse, i.e., low spatial node count, discretisation. We elaborated key considerations to preserve the 'forward' values within a sparse lattice evaluation framework, and discussed fundamental features such as the Metzler property, and we demonstrated the numerical convergence of the discussed drift correction techniques. Finally, we gave fully fledged Expected Positive Exposure calculation examples and compared these to the output of a commonly used software vendor for XVA calculation purposes.

We did not attempt to extend the presented methodology to more complicated derivative contracts such as barrier options or any in the realm of outright exotic derivatives, though such extensions are certainly feasible. The focus of this article was on portfolios of simple derivatives such as FX forward contracts, interest rate swaps of all variations, and vanilla options which make up the vast bulk of financial institutions derivatives holdings, and are often the sole concern for many smaller sized companies whose main business is not investment banking.

A The fifteen point stencil expansion

The coefficients of the fifteen point stencil given in (4.30) can be computed with the following Maxima code:

```
define(f(x,y,z),f0+fx*x+fy*y+fz*z
                  +fxx*x*x/2+fyy*y*y/2+fzz*z*z/2
                  +fxy*x*y+fxz*x*z+fyz*y*z);
Lx:(f(dx,0,0)-2*f0+f(-dx,0,0))/2; Lxy:(f(dx,dy,0)-2*f0+f(-dx,-dy,0))/2;
Ly:(f(0,dy,0)-2*f0+f(0,-dy,0))/2; Lyz:(f(0,dy,dz)-2*f0+f(0,-dy,-dz))/2;
Lz:(f(0,0,dz)-2*f0+f(0,0,-dz))/2; Lxz:(f(dx,0,dz)-2*f0+f(-dx,0,-dz))/2;
Lxyz:(f(dx,dy,dz)-2*f0+f(-dx,-dy,-dz))/2;
L:cx*Lx+cy*Ly+cz*Lz+cxy*Lxy+cxz*Lxz+cyz*Lyz+cxyz*Lxyz;
expand(solve([rxy=diff(L,fxy),rxz=diff(L,fxz),ryz=diff(L,fyz)],[cxy,cxz,cyz]));
expand(solve([1=2*diff(L,fxx),1=2*diff(L,fyy),1=2*diff(L,fzz)],[cx,cy,cz]));
```

The output is:

$$\text{(\%o10)}\quad [[cxy=\frac{rxy}{dxdy}-cxyz,\ cxz=\frac{rxz}{dxdz}-cxyz,\ cyz=\frac{ryz}{dydz}-cxyz]]$$

$$\text{(\%o11)}\quad [[cx=\frac{1}{dx^2}-cxz-cxyz-cxy,\ cy=\frac{1}{dy^2}-cyz-cxyz-cxy,\ cz=\frac{1}{dz^2}-cyz-cxz-cxyz]]$$

Note that `cxyz` remains a free parameter in the above calculation, as explained when we chose it according to (4.35) in section 4.2.2.

B Higher-dimensional cross-diffusion stencil rules

A generalisation of the coefficient calculation of the previous section to n dimensions is:-

```
load('pdiff)$
Lgen(lst,d):=block([up:makelist(0,d),down:makelist(0,d),i,k:length(lst)],
  for i:1 thru k do (up[lst[i]]:concat(x,lst[i]),down[lst[i]]:-concat(x,lst[i])),
    (apply(f,up)-2*apply(f,makelist(0,d))+apply(f,down))/2)$
Ltilde(d):=block([indices:setify(makelist(i,i,1,d)),j,s,e,lst,tmp:0],
  for j:1 thru d do (s:powerset(indices,j),
    for e in s do (lst:listify(e),tmp:tmp+arrayapply(c,lst)*Lgen(lst,d))), tmp)$
deriv(f,lst,d):=block([indcs:makelist(0,d)],
  for i:1 thru length(lst) do (indcs[lst[i]]:indcs[lst[i]]+1),
  apply(apply(pderivop,append([f],indcs)),makelist(0,d)))$
Dij(d):=full_listify(powerset(setify(makelist(i,i,1,d)),2))$
cij(d):=block([lst:[]], for ij in Dij(d) do (lst:append(lst,[arrayapply(c,ij)])),lst)$
Dijeqns(d,L):=block([derv,lst:[]], for ij in Dij(d) do (derv:deriv(f,ij,d),
  lst:append(lst,[arrayapply(r,ij)=ratsimp(coeff(L,derv))])), lst)$
Di(d):=full_listify(powerset(setify(makelist(i,i,1,d)),1))$
ci(d):=block([lst:[]], for i in Di(d) do (lst:append(lst,[arrayapply(c,i)])), lst)$
Dieqns(d,L):=block([derv,lst:[]], for i in Di(d) do (derv:deriv(f,append(i,i),d),
  lst:append(lst,[1=2*coeff(L,derv)])), lst)$
```

With $n = 4$, we obtain from

```
  n:4$
  /* Expansion of L̃ to 2nd order. */
  Lxpn:(taylor(Ltilde(n),makelist(concat(x,i),i,1,n),0,2))$
  /* Solve mixed derivative coefficients for bi-axial cij.    */
  expand(solve(Dijeqns(n,Lxpn),cij(n)));
  /* Solve second derivative coefficients for mono-axial ci. */
  expand(solve(Dieqns(n,Lxpn),ci(n)));
```

the output:

$$(\%o14)\quad \left[\left[c_{1,2}=\frac{r_{1,2}}{x_1x_2}-c_{1,2,4}-c_{1,2,3,4}-c_{1,2,3},\ c_{1,3}=\frac{r_{1,3}}{x_1x_3}-c_{1,3,4}-c_{1,2,3,4}-c_{1,2,3},\ c_{1,4}=\frac{r_{1,4}}{x_1x_4}-c_{1,3,4}-c_{1,2,4}-c_{1,2,3,4},\right.\right.$$
$$\left.\left.c_{2,3}=\frac{r_{2,3}}{x_2x_3}-c_{2,3,4}-c_{1,2,3,4}-c_{1,2,3},\ c_{2,4}=\frac{r_{2,4}}{x_2x_4}-c_{2,3,4}-c_{1,2,4}-c_{1,2,3,4},\ c_{3,4}=\frac{r_{3,4}}{x_3x_4}-c_{2,3,4}-c_{1,3,4}-c_{1,2,3,4}\right]\right]$$

$$(\%o15)\quad \left[\left[c_1=\frac{1}{x_1^2}-c_{1,4}-c_{1,3,4}-c_{1,3}-c_{1,2,4}-c_{1,2,3,4}-c_{1,2,3}-c_{1,2},\ c_2=\frac{1}{x_2^2}-c_{2,4}-c_{2,3,4}-c_{2,3}-c_{1,2,4}-c_{1,2,3,4}-c_{1,2,3}-c_{1,2},\right.\right.$$
$$\left.\left.c_3=\frac{1}{x_3^2}-c_{3,4}-c_{2,3,4}-c_{2,3}-c_{1,3,4}-c_{1,3}-c_{1,2,3,4}-c_{1,2,3},\ c_4=\frac{1}{x_4^2}-c_{3,4}-c_{2,4}-c_{2,3,4}-c_{1,4}-c_{1,3,4}-c_{1,2,4}-c_{1,2,3,4}\right]\right]$$

References

[ABR97] L. Andersen and R. Brotherton-Ratcliffe. The equity option volatility smile: An implicit finite-difference approach. *Journal of Computational Finance*, 1:5–38, 1997.

[And00] L. Andersen. A simple approach to the pricing of Bermudan swaptions in the multifactor LIBOR market model. *Journal of Computational Finance*, 3(2):5–32, Winter 1999/2000.

[BG97] M. Broadie and P. Glasserman. Pricing American-Style Securities Using Simulation. *Journal of Economic Dynamics and Control*, 21(8–9):1323–1352, 1997.

[Car96] J.F. Carriere. Valuation of the early-exercise price for options using simulations and nonparametric regressions. *Insurance: Mathematics and Economics*, 19:19–30, 1996.

[Con10] R. Cont, editor. *Encyclopedia of Quantitative Finance*. John Wiley and Sons, 2010. www.wiley.com/go/eqf.

[ELM01] P. Embrechts, F. Lindskog, and A. McNeil. Modelling Dependence with Copulas and Applications to Risk Management. Working paper, Department of Mathematics, ETHZ CH-8092, Zürich, Switzerland, April 6 2001. www.risklab.ch/ftp/papers/DependenceWithCopulas.pdf.

[Ger31] S. Gerschgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izv. Akad. Nauk. USSR Otd. Fiz.-Mat.*, 6:749–754, 1931.

[JA10] P. Jäckel and L. Andersen. Exercise boundary optimization methods. In Cont [Con10]. www.jaeckel.org/ExerciseBoundaryOptimization.pdf.

[Jäc02] P. Jäckel. *Monte Carlo methods in finance*. John Wiley and Sons, February 2002.

[Jäc13] P. Jäckel. Finite differencing schemes as Padé approximants, April 2013. www.jaeckel.org/FiniteDifferencingSchemesAsPad%C3%A9Approximants.pdf.

[Jäc17] P. Jäckel. Cluster Induction. WBS Fixed Income Conference Florence (September 2017) and QuantMinds Lisbon (May 2018), 2017. www.jaeckel.org/ClusterInduction(presentation_version).pdf and www.jaeckel.org/ClusterInduction(print_version).pdf.

[Jäc18] P. Jäckel. Cluster Induction. Quantminds Conference, Lisbon, May 2018.

[LS98] F. A. Longstaff and E. S. Schwartz. Valuing American options by simulation: A simple least squares approach. Working paper, The Anderson school, UCLA, 1998.

[Til93] J.A. Tilley. Valuing American options in a path simulation model. *Transactions of the Society of Actuaries*, 45:93–104, 1993.

[Wro] Wikipedia: Wrong way risk. https://en.wikipedia.org/wiki/Wrong_way_risk.

[ZP07] Steven H. Zhu and Michael Pykhtin. A Guide to Modeling Counterparty Credit Risk. *GARP Risk Review*, July/August 2007. ssrn.com/paper=1032522.