Open Journal of Statistics, **2013, 3, 427-440**
Published Online December 2013 (http://www.scirp.org/journal/ojs)
http://dx.doi.org/10.4236/ojs.2013.36051

Scientific Research

High-Dimensional Regression on Sparse Grids Applied to Pricing Moving Window Asian Options

Stefan Dirnstorfer[1], Andreas J. Grau[1], Rudi Zagst[2]
[1]Thetaris GmbH, Munich, Germany
[2]Technical University Munich, Munich, Germany
Email: dirnstor@thetaris.com, grau@thetaris.com, zagst@tum.de

Received July 29, 2013; revised August 29, 2013; accepted September 6, 2013

ABSTRACT

The pricing of moving window Asian option with an early exercise feature is considered a challenging problem in option pricing. The computational challenge lies in the unknown optimal exercise strategy and in the high dimensionality required for approximating the early exercise boundary. We use sparse grid basis functions in the Least Squares Monte Carlo approach to solve this "curse of dimensionality" problem. The resulting algorithm provides a general and convergent method for pricing moving window Asian options. The sparse grid technique presented in this paper can be generalized to pricing other high-dimensional, early-exercisable derivatives.

Keywords: Sparse Grid; Regression; Least-Squares Monte Carlo; Moving Window Asian Option

1. Introduction

Methods for pricing a large variety of exotic options have been developed in the past decades. Still, the pricing of high dimensional American-style moving average options remains a challenging task. The price of this type of options depends on the full path of the underlying, not only at the final exercise date but also during the whole period of exercisable times. We consider in this paper the case of an early-exercisable floating-strike moving window Asian option (MWAO) with discrete observations for the computation of the exercise value. The exercise value of the MWAO depends on a moving average of the underlying stock over a period of time.

Carriere [1] first introduces the simulation-based method for solving American-type option valuation problems. A similar but simpler method is presented by Longstaff and Schwartz [2]. Their method is known as the Least Squares Monte Carlo (LSM) method. It uses the least-squares regression method to determine the optimal exercise strategy. Longstaff and Schwartz also use their LSM method to price an American-Bermuda-Asian option that can be exercised on a specific set of dates after an initial lockout period. Their American-Bermuda-Asian option has an arithmetic average of stock prices as the underlying. The pricing problem can be reduced to two dimensions after introducing another variable in the partial differential equation (PDE) to represent the arithmetic average.

The dimension reduction technique as in Longstaff-Schwartz [2] can not be applied for the pricing problem of MWAOs. Since moving averages shift up and down when the underlying prices shift up and down especially when the first observation in the moving window drops out and a new one comes in, the whole history of stock prices is important in determining the optimal exercise strategy of MWAOs. This leads to an arbitrary number of dimensions and presents a computational challenge. Pricing methods for MWAOs have been described by very few authors besides Broadie and Cao [3]. Broadie and Cao price a fixed strike MWAO, using polynomials of underlying asset price and arithmetic average as the regression basis function. Bilger [4] applies the LSM method to price MWAOs. He uses a different choice of basis functions (*i.e*. the underlying asset S and a set of averages) for evaluating the conditional expected option value. Kao and Lyuu [5] present results for moving average-type options traded in the Taiwan market. Their

method is based on the binomial tree model and they include up to 6 discrete observations in the averaging period for their numerical examples. Bernhart *et al*. [6] use a truncated Laguerre series expansion to reduce the infinite dimensional dynamics of a moving average process to a finite dimensional approximation and then apply the LSM algorithm to price the finite-dimensioned moving average American-type options. Their numerical implementations can handle dimensions up to 8, beyond that their method becomes infeasible. Dai *et al*. [7] use a forward shooting grid method to price European and American-style moving average barrier options. The window lengths in their numerical examples range from three or four days to two or three months.

In this paper, we apply an alternative type of basis functions—the sparse grid basis functions—to the simulation-based LSM approach for pricing American-style MWAOs. The sparse grid technique overcomes the low-dimension limit associated with full grid discretizations and achieves reasonable accuracy for approximating high-dimensional problems. Instead of using a predetermined set of basis functions in the least squares regressions, the sparse grid basis functions are adaptive to the data—it is more general and considers as many information in the moving window as possible. Using numerical examples, we demonstrate the convergence of the pricing algorithms for MWAOs for different numbers of Monte-Carlo paths, different sparse grid levels and a fixed length of observation period of 10 days. Sparse grid is a discretization technique that is designed to circumvent the "curse of dimensionality" problem in standard grid-based methods for approximating a function. The idea of sparse grid was originally discovered by Smolyak [8] and was rediscovered by Zenger [9] for PDE solutions in 1990. Since then, it has been applied to many different topics, such as integration [10,11] or Fast Fourier Transform (FFT) [12]. Sparse grids have also been used for finite element PDE solutions by Bungartz [13], interpolation by Bathelmann *et al*. [14], clustering by Garcke *et al*. [15], and PDE option pricing by Reisinger [16].

The structure of this paper is as follows: first, we formulate the pricing problem of a moving window Asian option and explain why this problem is computationally challenging. This is followed by a brief description of the LSM approach and the sparse grid technique. Finally, we provide some numerical examples for pricing MWAOs with discretely sampled observations using LSM with sparse grid type basis functions.

Throughout this paper, we consider equity options on a single underlying stock in the Black Scholes [17] framework.

2. Moving Window Asian Option

MWAO is an American-style option that makes use of the moving average dynamics of stock prices. Similar to an American option which pays the difference between the current underlying price and a fixed strike, a MWAO pays the difference between current stock price and the floating moving average or the difference between a floating moving average and a fixed strike.

2.1. Continuous Time Version

Before going into the details of an MWAO, we set up the process for the underlying stock. The stock prices are assumed following a geometric Brownian motion (GBM) process

$$\mathrm{d}S_t = rS_t\,\mathrm{d}t + \sigma S_t \mathrm{d}W_t, \tag{1}$$

where r is the constant riskless interest rate, σ is the constant stock return volatility and $\mathrm{d}W$ is the increment of a standard Wiener process under the risk-neutral measure. The initial stock price is denoted as S_0 and at time t the stock price is S_t. The value of a MWAO option written on the stock is denoted in general as V, or V_t resp. $V(t,S_t)$ when we stress the dependence on t or S_t. The option value V satisfies the following Black Scholes PDE

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial S_t^2} + rS_t \frac{\partial V}{\partial S_t} - rV = 0. \tag{2}$$

The following American constraint sets a minimum value for the function V. The constraint has to be satisfied at each time $t > t_0 + t_w$, where t_0 denotes the set-up time of the option and t_w denotes a fixed window length

$$V(t,S_t) \geq P(A_t,S_t), \tag{3}$$

$$A_t = \frac{1}{\int_0^{t_w}\alpha(\tau)\,\mathrm{d}\tau}\int_{t-t_w}^{t}\alpha(t-\tau)S_\tau\,\mathrm{d}\tau, \tag{4}$$

where $P(A_t,S_t)$ is the payoff of a MWAO from exercise at time t. It depends on the stock price S_t and a weighted moving average A_t of the stock prices. The moving average A_t is computed using the weight function α on a window of stock prices ranging from time $t-t_w$ to time t. In this paper, we consider the payoff function

$$P(A_t,S_t) = \max\{A_t - S_t, 0\}. \tag{5}$$

when setting the weight $\alpha \equiv 1$ in (4), we have an equally-weighted arithmetic moving average

$$A_t = \frac{1}{t_w}\int_{t-t_w}^{t} S_\tau\,\mathrm{d}\tau.$$

Differentiating this expression with respect to time t leads to

$$\mathrm{d}A_t = \frac{1}{t_w}\left(S_t - S_{t-t_w}\right)\mathrm{d}t,$$

where the updating process of A_t depends not only on the stock price at time t, but also on the stock price at time $t-t_w$. This is clearly not Markovian. An optimal exercise strategy performed on the above moving average process A_t has to consider S_t, S_{t-t_w} and all stock prices $S_u, t > u > t-t_w$ in between. Since all the values S_u are used in computing the moving average A_t, and A_t in turn determines the MWAO payoff in (3), there are infinitely many prices involved in the computation of an optimal exercise strategy. This is a challenging infinite-dimensional problem in continuous time [6,7].

2.2. Discretizations

To implement the pricing problem for the MWAO, we consider the finite dimensional case with discretely sampled observations. Define a set of times

$$\mathcal{T} = \{t_0 = 0, t_1, \cdots, t_n = T\},$$

with $t_i \in \mathcal{T}, i = 0, \cdots, n$, and $t_i < t_j$ for $i < j$. At time t_i, the value of S and V is respectively denoted as S_{t_i} and V_{t_i}.

We consider constant weight $\alpha(t)$ in defining a discretely sampled moving average A_t

$$\alpha(t) = \begin{cases} 1, & \text{for } t < m \\ 0, & \text{for } t = m \end{cases} \tag{6}$$

where m denotes the number of samples used in computing the moving average. Using the weight function $\alpha(t)$ in defining the moving average A_t, the boundary condition (3) at times $\{t_m, \cdots, t_{n-1}\}$ has the following discretized form

$$V_{t_i}\left(S_{t_{i-m}}, \cdots, S_{t_i}\right) \geq P\left(\frac{1}{m}\sum_{j=i-m}^{i} \alpha(i-j) S_{t_j}, S_{t_i}\right), \tag{7}$$

where the weight function $\alpha(i-j)$ assigns a zero weight to an initial observation in the moving window and a weight of one to the rest of the observations. With this weight function we have effectively used the past m samples to form the moving window. The above condition holds for $i \geq m$, after an initial allowance for the window length. The dimension of this MWAO problem equals to the number of discrete samples m used in the averaging window.

Our method for valuing the MWAO uses the discretized process and a quadrature of α. The valuation proceeds backwards in time, starting at the option maturity T, where condition (7) holds with equality. Then we solve for the option value at current time $V = V(t_0, S_{t_0})$. For short window length m and thus low dimensionality, this procedure can be reformulated in a PDE setting and solved numerically. However, due to the "curse of dimensionality", the PDE method is ineffective for dimensions of more than three or four. For window length m larger than four, the high dimensional problem has to be solved using approximate representations or special numerical techniques.

3. Numerical Procedure

The previous sections provided the mathematical formulations and discussed the discretization issues related to the MWAO. This section details the numerical methods we use for pricing MWAOs. The algorithm proposed by this paper is effectively a combination of three techniques that are well established in their respective fields. The three techniques that we use as a practical tool for valuing a MWAO are Monte Carlo simulation, least squares regression and sparse grids. Especially in quantitative finance, sparse grids technique has not yet lived to its full potential. This paper contributes to use sparse grids in solving high-dimensional problems. Since all the three techniques have been documented in full detail by the cited sources, we summarize in the following the main aspects of each technique. Without explicitly mentioning it, all prices in our computations are discounted prices, meaning that prices are already normalized by the bank account numeraire. We use $\tilde{S}$, $\tilde{P}$ and $\tilde{V}$ to denote discounted stock prices, discounted payoffs and discounted option values.

3.1. Monte Carlo Simulation

A standard method that is used when dimensionality causes numerical difficulties is the Monte Carlo simulation method. This method alone does not resolve our issue, but provides the framework for our algorithm. We assume that the stock price underlying a MWAO follows the GBM process defined in (1). The discretized stock price process is sampled at the set of discrete times $t_i \in \mathcal{T}$ so that each of the realizations $\tilde{S}^j$, $j \in \{1, \cdots, s_1\}$ with s_1 denoting the number of Monte-Carlo paths, has the following normalized representation

$$\tilde{S}^j_{t_{i+1}} = \tilde{S}^j_{t_i}{}^{\left(-\frac{1}{2}\sigma^2(t_{i+1}-t_i)+\sigma\sqrt{t_{i+1}-t_i}\phi^j_{t_{i+1}}\right)}, \tag{8}$$

where $\phi^j_{t_{i+1}}$ is drawn from a standardized Normal distribution.

The price of the MWAO is the expected value of the (discounted) payoff at the optimal stopping time. The optimal stopping time provides a strategy that maximizes the option value without using any information about future stock prices.

3.2. Least Squares

At each exercise time t_i, the option holder decides whether to exercise the option and get the payoff $\tilde{P}(\tilde{S}, t_i)$ or to continue holding the option. In order to

maximize the option value $\tilde{V}_{t_i}$ at time t_i, the holder exercises if

$$\tilde{P}\left(\tilde{S},t_i\right) \geq \mathbb{E}\left[\tilde{V}_{t_{i+1}}\middle|\tilde{S},t_i\right], \tag{9}$$

where $\mathbb{E}$ denotes the expectation taken under the risk neutral measure. In the LSM approach, the value of $\mathbb{E}\left[\tilde{V}_{t_{i+1}}\middle|\tilde{S},t_i\right]$ is approximated by a function $\tilde{P}^e\left(\tilde{S},t_i\right)$

$$\tilde{P}^e\left(\tilde{S},t_i\right) \approx \mathbb{E}\left[\tilde{V}_{t_{i+1}}\middle|\tilde{S},t_i\right]. \tag{10}$$

The value of $\tilde{P}^e\left(\tilde{S},t_i\right)$ is computed using a least square regression on many path-realizations $\tilde{S}_{t_i}^j, j=1,\cdots,s_1$. The regressions start at one time step before the maturity T.

The function $\tilde{P}^e\left(\tilde{S},t_i\right)$ is a linear combination of the basis functions ϕ_k

$$\tilde{P}^e\left(\tilde{S},t_i\right) = \sum_k a_k^i \phi_k\left(\tilde{S}_{t_{i-m}},\cdots,\tilde{S}_{t_i}\right). \tag{11}$$

The coefficients a_k^i are found by minimizing the L_2-norm

$$\left\| \sum_k a_k^i \phi_k\left(\tilde{S}_{t_{i-m}}^j,\cdots,\tilde{S}_{t_i}^j\right) - \tilde{V}_{t_i}^j \right\|_2, \; j=1,\cdots,s_1, \tag{12}$$

where $\tilde{V}_{t_i}^j$ is the option value of a Monte Carlo path realization $\tilde{S}^j$ at time t_i. The option value $\tilde{V}_{t_i}^j$ is given as the maximum between the estimated continuation value and the intrinsic value. A numerically more stable algorithm is to set

$$\tilde{V}_{t_i}^j = \begin{cases} \tilde{V}_{t_{i+1}}^j & \text{if } \tilde{P}^e\left(\tilde{S}^j,t_i\right) > \tilde{P}\left(\tilde{S}^j,t_i\right) \\ \tilde{P}\left(\tilde{S}^j,t_i\right) & \text{else} \end{cases}, \tag{13}$$

where the computed $\tilde{P}^e\left(\tilde{S}^j,t_i\right)$ is used only in early-exercise decisions, this avoids the accumulation of approximation errors when stepping backwards in time. Given the option payoff $\tilde{V}_T^j = \tilde{P}\left(\tilde{S}_T^j,T\right)$ at maturity time, a backward induction dynamic programming method solves for all values $\tilde{V}_{t_i}^j$, starting at time T and iterating back to t_0. Based on the values $\tilde{V}_{t_0}^j, j=1,\cdots,s_1$, we compute an estimated option value, known as the in-sample price

$$V^{in} = \frac{1}{s_1}\cdot\sum_{j=1}^{s_1}\tilde{V}_{t_0}^j. \tag{14}$$

This approach has an obvious shortcoming. Each of the estimated option values $\tilde{V}_{t_0}^j$ contains information about its future stock path $\tilde{S}_{t_i}^j, t_i \in \mathcal{T}$. In order to avoid this perfect foresight bias, we compute an out-of-sample option price: we generate additional simulation paths $\tilde{S}^l$, $l \in \{s_1+1,\cdots,s_1+s_2\}$ but use the coefficients a_k^i fitted to the old set of simulation paths $\tilde{S}^j$, $j \in \{1,\cdots,s_1\}$. Consequently, the out-of-sample value does not depend on knowledge of the future paths. The out-of-sample option value is computed by

$$V^{out} = \frac{1}{s_2}\cdot\sum_{l=s_1+1}^{s_1+s_2}\tilde{V}_{t_0}^l, \tag{15}$$

with

$$\tilde{V}_{t_i}^l = \begin{cases} \tilde{V}_{t_{i+1}}^l & \text{if } \tilde{P}^e\left(\tilde{S}^l,t_i\right) > \tilde{P}\left(\tilde{S}^l,t_i\right) \\ \tilde{P}\left(\tilde{S}^l,t_i\right) & \text{else} \end{cases}. \tag{16}$$

In our implementations, we compute only the out-of-sample value since it is the value for which we can state the optimal exercise policy without information about the future. The expected value of the out-of-sample price V^{out} is always a lower bound for the option value, and the estimate $\tilde{P}^e$ is crucial for the convergence of the least squares Monte Carlo simulation method. We are confined to finitely many samples and to finite degrees of freedom in the regressions, thus are not able to perfectly represent the real shape of $\mathbb{E}\left[\tilde{V}_{t_{i+1}}\middle|\tilde{S},t_i\right]$ using the estimate $\tilde{P}^e$. A less than optimal exercise strategy is performed and provides a lower biased option value.

3.3. Basis Functions

An important issue in the LSM approach is a careful choice of the basis functions ϕ_k in (11). We will use in this paper a linear combination of the sparse grid type basis functions to approximate the conditional expected value $\mathbb{E}\left[\tilde{V}_{t_{i+1}}\middle|\tilde{S},t_i\right]$ involved in the optimal exercise rules. We need one dimension for each observation in the averaging window, this leads to a high dimensionality in the computational problem. Sparse grid [8] is a discretization technique designed to circumvent this "curse of dimensionality" problem. It gives a more efficient selection of basis functions. This technique has been successfully applied in the field of high-dimensional function approximations [15] and many others [10,13,16,18].

In the following we provide a brief description of the sparse grid approach. We start from constructing one-dimensional basis functions in a general case and show how to build multi-dimensional basis functions from the one-dimensional ones. Next we create a finite set of basis functions for numerical computations. Sparse grid then efficiently combines the sets of basis functions in a way such that the resulting function set is linearly independent. Following this, we detail on two specific types of sparse basis functions - a polynomial function and a piecewise linear function - to be used as the basis functions in the LSM regressions in this paper.

3.3.1. Constructing the Basis Functions

When approximating a function with simpler functions or numerically extracting the shape of a function, it is com-

mon to build such a function representation using some basis functions. We consider here a set of basis functions $\phi=\{\phi_1,\phi_2,\cdots,\phi_n\}$ and we call the set (sets) of basis functions "function basis (bases)". From the set ϕ, we construct an approximating function $\tilde{f}$ as a linear combination of the basis functions

$$\tilde{f}(x)=\sum_{k=1}^{n}a_k\phi_k(x), \tag{17}$$

with coefficients a_k for $\phi_k, k=1,\cdots,n$, and $x\in X$ for some set X. The basis functions $\phi_k(x)$ can be a one-dimensional or multi-dimensional mapping from X to R.

For the one-dimensional case, many function bases are well known and widely used. Examples include polynomials, splines, B-splines, Bessel functions, trigonometric functions, and so on. For the multi-dimensional case, the set of basis functions that are commonly used is more scarce. There are two common approaches to constructing multi-dimensional basis functions from the one-dimensional ones: the radial basis functions [19] and the tensor product functions [20]. In this paper we focus on the tensor product approach. To construct a tensor product basis function, we select one-dimensional functions ψ and multiply their respective function value evaluated at the corresponding component of x. Specifically, for $X\subseteq R^m$ and each *m*-dimensional basis function ϕ_k, we choose a set of one dimensional functions

$$\{\psi_{k,1},\psi_{k,2},\cdots,\psi_{k,m}\}$$

and

$$x=(x_1,x_2,\cdots,x_m),$$

then multiply the j-th function $\psi_{k,j}$ evaluated at the j-th element x_j, for $j=1,\cdots,m$, to have the following representation of an m-dimensional basis function

$$\phi_k(x)=\psi_{k,1}(x_1)\psi_{k,2}(x_2)\cdots\psi_{k,m}(x_m). \tag{18}$$

3.3.2. Creating a Function Basis

Having created multi-dimensional basis functions in a general case, we now select a finite set of these functions to be our basis for numerical analyses. Since we will do computations on different levels of accuracy, we will also need a set of basis functions on different levels. Naturally, the more basis functions we put in the set, the more accurate are our function approximations.

We decide that on a level L there are $\lambda(L)+1$ basis functions. The one-dimensional function basis Ψ_L with $\lambda(L)+1$ functions is the following set

$$\Psi_L=\{\psi_1,\psi_2,\cdots,\psi_{\lambda(L)+1}\}, \tag{19}$$

where $\psi_i, i\in\{1,\cdots,\lambda(L)+1\}$ are one-dimensional basis functions, and $\lambda(L)$ is a monotonously increasing function that determines the size of our basis at level L.

In order to create an m-dimensional function basis, we choose a level $L=(L_1,L_2,\cdots,L_m)$. For each dimension $j\in\{1,\cdots,m\}$, the level L_j implies a function basis $\Psi_{L_j}=\{\psi_1,\psi_2,\cdots,\psi_{\lambda(L_j)+1}\}$ via (19). Using the tensor product function, the m-dimensional function basis Φ_L is constructed as

$$\Phi_L=\Phi_{(L_1,L_2,\cdots,L_m)}=\left\{x\mapsto\breve{\psi}_1(x_1)\breve{\psi}_2(x_2)\cdots\breve{\psi}_m(x_m)\middle|\breve{\psi}_j\in\Psi_{L_j}, j=1,\cdots,m\right\}, \tag{20}$$

where $\breve{\psi}_j(x_j)$ denotes any of the one-dimensional basis functions from Ψ_{L_j}, evaluated at the j th element of $x=(x_1,x_2,\cdots,x_m)$.

It was our original goal to construct a function basis with increasing expressiveness for increasing levels. This multi-dimensional level L is not a very good starting point. Consider the case where all dimensions are equally important, we would have to use a level of the form $L=(\ell,\ell,\cdots,\ell)$. The size of the resulting function set would be extremely large, a phenomenon known as the "curse of dimensionality" problem:

$$|\Phi_L|=\left|\Phi_{(\ell,\ell,\cdots,\ell)}\right|=|\Psi_\ell|^m. \tag{21}$$

Choosing $m=10$ dimensions and $\lambda(\ell)=3$ results in 3^{10} functions with a corresponding number of coefficients to be solved.

3.3.3. Sparse Grids

Sparse grids were developed as an escape from the "curse of dimensionality" problem. They allow reasonably accurate approximations in high dimensions at low computational cost. The idea behind the sparse grid is to combine the tensor basis of different levels.

Let Ω_{L^*} be the set of basis functions on level L^*. We define it as the union of all tensor function bases Φ_L where the sum of m levels $(L_1+L_2+\cdots+L_m)$ is equal to L^*, with $L^*\in\mathbb{N}_0$ and $\mathbb{N}_0$ denoting the set of all non-negative numbers

$$\Omega_{L^*}=\bigcup_{\substack{L_1+L_2+\cdots+L_m=L^*\\ L=(L_1,\cdots,L_m)}}\Phi_L. \tag{22}$$

Considering a simple example with $m=2$ dimensions on level $L^*=2$, *i.e.* $L_1+L_2=L^*=2$. For $L_k\in\mathbb{N}_0$ with $k=1,2$, there are three possible combi- nations of the levels L_1 and L_2 that sum to 2: $(0,2)$, $(1,1)$ and $(2,0)$. From this, we could create a set $\Omega_{L^*=2}$ as the union of the tensor function bases: $\Phi_{(2,0)}\cup\Phi_{(1,1)}\cup\Phi_{(0,2)}$. The first set $\Phi_{(2,0)}$ has high resolution in the first dimension L_1, while the last set $\Phi_{(0,2)}$ mainly resolves

the second dimension L_2. By construction, basis functions with high resolution in both dimensions, such as $\Phi_{(2,2)}$, are left out in the sparse set, making the sparse basis ideal for approximating functions with bounded mixed derivatives.

The computational effort of sparse grids compared with conventional full grids decreases radically while the error rises only slightly: for the representation of a function $\tilde{f}$ over an *m*-dimensional domain with minimal mesh size $h_L = 2^{-L}$, a sparse grid employs $O\left(h_L^{-1}\left|\log h_L\right|^{m-1}\right)$ points and a full grid has $O\left(h_L^{-m}\right)$ grid points. At the same time, the L_2 interpolation error for smooth functions is $O\left(h_L^2 \cdot \left|\log h_L\right|^{m-1}\right)$ for sparse grids and $O\left(h_L^2\right)$ for full grids.

Having constructed a sparse basis Ω_{L^*} on a top level, we now create two specific types of sparse basis functions that will be used as the function sets in our LSM regression problem: a polynomial sparse function basis and a piecewise linear function basis. Compared to the piecewise linear basis functions, the polynomial sparse basis functions are easier to understand and easier to implement. As solutions to higher dimensional problems, the piecewise linear basis functions are more adaptive. They can be extended to effectively place the basis functions on the dimension that contributes more to the problem solution. As a result, piecewise linear basis functions have seen wide applicability to solving PDEs [18] and interpolating functions [21]. In our paper, we use these two types of sparse basis functions to cross validate the results of our high dimensional pricing problem.

3.3.4. A Polynomial Sparse Basis

A polynomial function basis with $\lambda(L)+1$ basis functions in the one-dimensional case has the following construction

$$\Psi_L = \left\{1, x, x^2, x^3, \cdots, x^{\lambda(L)}\right\}. \tag{23}$$

From the one-dimensional functions Ψ_L, we can build a multi-dimensional tensor basis according to (20).

As an example, for the two-dimensional case $(m=2)$, we have the first dimension in x with one-dimensional functions

$$\Psi_{L_1} = \left\{1, x, x^2, x^3, \cdots, x^{\lambda(L_1)}\right\},$$

and the second dimension in y with one-dimensional functions

$$\Psi_{L_2} = \left\{1, y, y^2, y^3, \cdots, y^{\lambda(L_2)}\right\}.$$

Equation (20) then gives the following list of tensor basis functions

$$\Phi_{(L_1,L_2)} = \begin{Bmatrix} 1 & x & x^2 & \cdots & x^{\lambda(L_1)} \\ y & xy & x^2y & \cdots & x^{\lambda(L_1)}y \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y^{\lambda(L_2)} & xy^{\lambda(L_2)} & x^2y^{\lambda(L_2)} & \cdots & x^{\lambda(L_1)}y^{\lambda(L_2)} \end{Bmatrix} \tag{24}$$

Building upon the two-dimensional function basis $\Phi_{(L_1,L_2)}$, we construct a sparse basis function set for the three sparse levels $L^*=0$, $L^*=1$, and $L^*=2$, where $L^* = L_1 + L_2$. We let the three levels correspond to $\lambda\left(L^*=0\right)=0$, $\lambda\left(L^*=1\right)=2$ and $\lambda\left(L^*=2\right)=6$ number of basis functions. Using (22) to unite the function sets, our sparse bases for the two dimensional function space are the following sets

$$\begin{aligned} \Omega_{L^*=0}^{poly} &= \Phi_{(0,0)} = \{1\} \\ \Omega_{L^*=1}^{poly} &= \Phi_{(1,0)} \cup \Phi_{(0,1)} = \{1, x, x^2\} \cup \{1, y, y^2\} \\ &= \{1, x, x^2, y, y^2\} \\ \Omega_{L^*=2}^{poly} &= \Phi_{(2,0)} \cup \Phi_{(1,1)} \cup \Phi_{(0,2)} \\ &= \{1, x, x^2, x^3, x^4, x^5, x^6, y, y^2, y^3, y^4, \\ &\quad y^5, y^6, xy, x^2y, xy^2, x^2y^2\}. \end{aligned} \tag{25}$$

The above sparse basis function sets $\Omega_{L^*}^{poly}$ are created by taking the unions of the tensor bases of $\Phi_{(L_1,L_2)}$ according to (22). As an example, the sparse set $\Omega_{L^*=1}^{poly}$ is constructed as a union of

$$\Phi_{(1,0)} = \Phi_{(L_1=1,L_2=0)} = \{1, x, x^2\}$$

and

$$\Phi_{(0,1)} = \Phi_{(L_1=0,L_2=1)} = \{1, y, y^2\}.$$

The set $\Phi_{(1,0)}$ is arrived at by having $\lambda\left(L_1=1\right)=2$ number of basis functions in the x direction and $\lambda\left(L_2=0\right)=0$ basis functions in the y direction from the two-dimensional set $\Phi_{(L_1,L_2)}$, ceteris paribus.

We have just created a polynomial sparse basis $\Omega_{L^*}^{poly}$ in two dimensions. Higher dimensional function bases can be similarly constructed using the tensor product approach, depending on the dimension of the problem we intend to solve. In our LSM regression problems, we have for example used $m=10$ dimensions and a polynomial sparse level of $L^* \in \{0,1,2\}$.

While the polynomial sparse basis functions aim for a global fit, the piecewise linear basis functions fit locally to the approximated functions and it is to the task of constructing the piecewise linear basis functions that we now turn.

3.3.5. Piecewise Linear Functions on [0,1]

The piecewise linear function is a type of basis function that is commonly used in sparse grid applications. To create a piecewise basis for various levels, we utilize a construction approach known from multi-resolution analysis. We define a mother function $\psi(x)$ and generate our basis by scaling and translating from $\psi(x)$

$$\psi(x)=\begin{cases} x+1 & \text{when } x\in(-1,0] \\ 1-x & \text{when } x\in(0,1] \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

It has been found computationally advisable to have only one basis function on the first level. Hence we start with the constant 1 on level $L^*=0$ and transform ψ on successive levels. This construction avoids the inclusion of costly boundary points by creating boundary functions that are less scaled than inner functions. Klimke and Wohlmuth [21] is a good reference on piecewise linear basis functions.

The one-dimensional function bases of levels $L_1=0$, $L_1=1$, $L_1=2$, $\cdots$, $L_1=k$ in x as generated from the one-dimensional basis functions ψ in (26) are the following sets

$$\begin{aligned} \Psi_{L_1=0} &= \{1\} \\ \Psi_{L_1=1} &= \Psi_{L_1=0}\cup\{\psi(2x),\psi(2x-2)\} \\ \Psi_{L_1=2} &= \Psi_{L_1=1}\cup\{\psi(4x),\psi(4x-4)\}\cup\{\psi(8x-3),\psi(8x-5)\} \\ &\vdots \end{aligned} \tag{27}$$

$$\begin{aligned} \Psi_{L_1=k} &= \Psi_{L_1=k-1}\cup\{\psi(2^k x),\psi(2^k x-2^k)\} \\ &\quad\cup\{\psi(2^{k+1}x-3),\psi(2^{k+1}x-5),\psi(2^{k+1}x-7),\cdots,\psi(2^{k+1}x-(2^{k+1}-3))\}, \end{aligned} \tag{28}$$

where the series $3,5,7,\cdots,(2^{k+1}-3)$ in (28) is a sequence of odd numbers. The one-dimensional function bases in y can be analogously generated from (26).

The construction of a two-dimensional sparse basis in the x and y directions respectively for levels $L^*=0$, $L^*=1$ and $L^*=2$, where $L^*=L_1+L_2$, follows from (22)

$$\begin{aligned} \Omega^{piece}_{L^*=0} &= \Phi_{(0,0)}=\{1\} \\ \Omega^{piece}_{L^*=1} &= \Phi_{(1,0)}\cup\Phi_{(0,1)}=\{1,\psi(2x),\psi(2x-2),\psi(2y),\psi(2y-2)\} \\ \Omega^{piece}_{L^*=2} &= \Phi_{(2,0)}\cup\Phi_{(1,1)}\cup\Phi_{(0,2)} \\ &= \{1,\psi(2x),\psi(2x-2),\psi(4x),\psi(4x-4),\psi(8x-3),\psi(8x-5),\psi(2y),\psi(2y-2),\psi(4y), \\ &\quad \psi(4y-4),\psi(8y-3),\psi(8y-5),\psi(2x)\psi(2y),\psi(2x)\psi(2y-2),\psi(2x-2)\psi(2y), \\ &\quad \psi(2x-2)\psi(2y-2)\}. \end{aligned} \tag{29}$$

As an example, the sparse basis function set $\Omega^{piece}_{L^*=2}$ is created by taking the unions of the function bases of $\Phi_{(2,0)}$, $\Phi_{(1,1)}$, and $\Phi_{(0,2)}$, where $\Phi_{(2,0)}=\Phi_{(L_1=2,L_2=0)}$ is the tensor product of $\Psi_{L_1=2}$ and $\Psi_{L_2=0}$, with $\Psi_{L_1=2}$ given in (27) and $\Psi_{L_2=0}=1$. The set $\Phi_{(L_1,L_2)}$ is constructed using the tensor product function (20).

The two-dimensional sparse bases can be extended to higher dimensions depending on the problem dimensions we intend to solve. The resulting higher-dimensional function sets can then be used in the LSM regressions by selecting an optimal sparse level for that problem dimension.

3.3.6. Implementations

We perform the regressions required by (12) on sparse polynomial basis functions $\Omega^{poly}_{L^*}$ and sparse piecewise linear basis functions $\Omega^{piece}_{L^*}$ as explained in Section 3.3.4 and 3.3.5, respectively. As an example, in the case of sparse polynomial basis functions of $m=2$ dimensions, the approximating function $\tilde{P}^e$ of (11) is a linear combination of the basis functions in the polynomial sparse basis set $\Omega^{poly}_{L^*}$ in (25). At sparse level $L^*=1$, the set $\Omega^{poly}_{L^*=1}=\{1,x_1,x_1^2,x_2,x_2^2\}$, and the approximating function

$$\begin{aligned} \tilde{P}^e(x^j,t_i) &= a_1^i + a_2^i * x_1^j + a_3^i * (x_1^j)^2 \\ &\quad + a_4^i * x_2^j + a_5^i * (x_2^j)^2. \end{aligned}$$

For the sparse piecewise linear basis functions of $m=2$ dimensions, the approximating function $\tilde{P}^e$ is a linear combination of the basis functions in the piecewise linear sparse basis set $\Omega^{piece}_{L^*}$ in (29). At sparse level

$L^*=1$, the set

$$\Omega_{L^*=1}^{piece}=\{1,\psi(2x_1),\psi(2x_1-2),\psi(2x_2),\psi(2x_2-2)\},$$

and the approximating function

$$\tilde{P}^e(x^j,t_i)=a_1^i+a_2^i\times\psi(2x_1^j)+a_3^i\times\psi(2x_1^j-2)+a_4^i\times\psi(2x_2^j)+a_5^i\times\psi(2x_2^j-2)$$

In our implementations, we use sparse levels L^* from 0 up to 3 and dimensions $m=10$ for computing the basis functions. This is sufficient for our purposes. But, we do not perform the regressions on $\tilde{S}$ directly. Instead, we use scaled values of $\tilde{S}$ such that for each path j, we compute

$$x^j=(x_1^j,\cdots,x_{m+1}^j)=(\gamma_1^j\tilde{S}_{t_i}^j,\cdots,\gamma_{m+1}^j\tilde{S}_{t_{i-m}}^j),$$

where $(\gamma_1^j,\cdots,\gamma_{m+1}^j)$ are defined such that $x^j\in[0,1]^{m+1}$. For discretely sampled observations, the dimension of the problem is effectively m dimensions because of the weight function α used in computing the moving averages. When extrapolating to the continuous case, the dimension of the problem approaches infinity as m goes to ∞.

The regression itself is performed solving the least squares problem of (12) via QR-decomposition. Furthermore, the regression is only performed on paths with a positive exercise value $\tilde{S}^j:\tilde{P}(\tilde{S}^j,t_i)>0$. This significantly decreases the computational effort.

4. Numerical Example

We provide in this section a numerical case study of using sparse grid basis functions in LSM pricing MWAOs. We will use a discretely sampled averaging window spanning ten observations with the weight function α in (6).

The properties of the MWAO are defined in **Table 1**. The underlying stock prices are sampled at a regular frequency, e.g. every trading day at a specific time.

To analyze the convergence of our pricing algorithm for MWAOs, we use the notation $\widetilde{V_\alpha}^i(n,L^*,m)$ to denote the out-of-sample result V^{out} as defined in (15). Each Monte Carlo value $\widetilde{V_\alpha}^i(n,L^*,m)$ depends on the number of sample paths n, the level of the sparse grid function basis L^*, the number of observations in the averaging window m, and the quadrature scheme α. We compute different $\widetilde{V_\alpha}^i(n,L^*,m)$ with n, L^*, m and α fixed in order to get an estimate for the mean

$$\overline{V_\alpha}(n,L^*,m)=\frac{1}{I}\sum_{i=1}^{I}\widetilde{V_\alpha}^i(n,L^*,m) \quad (30)$$

of I different Monte Carlo prices. Each $i\in I$ valuation $\widetilde{V_\alpha}^i(n,L^*,m)$ is based on a randomly generated Monte Carlo seed. The number of I ranges from 10 to 1000 depending on an estimate of the Monte Carlo error. For instance, if the error is acceptable based on our 95% confidence level at $I=10$, we stop the computation and obtain the mean price estimate using (30). Otherwise, we continue computing $\widetilde{V_\alpha}^i(n,L^*,m)$ by increasing the value of I until the error is acceptable. The number of samples n per Monte Carlo price results from combining the in-sample paths $S^1,\cdots,S^{s_1}$ and the out-of-sample paths $S^{s_1+1},\cdots,S^{s_1+s_2}$ such that $n=s_1+s_2$. We use 30% of the sample paths for regressions and 70% for option valuation out-of-sample.

Table 1. Specifications of a moving window Asian option with a floating strike. The moving averages are based on discretely sampled stock prices.

Option type	moving window Asian option
Maturity T	0.4 years
Initial stock price S_0	100
Risk free rate r	5%
Volatility σ	0.40
Dividend rate d	0
Daily observations $\Delta_{obs}t$	1/250 years
Length of observation period m	10 days
Exercise value	$\max\left(\frac{1}{m}\left(\sum_{j=i-m}^{i}\alpha(i-j)S_{t_j}\right)-S_{t_i},0\right)$

Figure 1 presents the mean $\overline{V_\alpha}(n,L^*,10)$ for different numbers of samples n and different levels L^*, but with the dimensions $m=10$ and the weights α fixed. Included in the figure are the results from the polynomial and the piecewise linear sparse basis functions. The option values at sparse level $L^*=0$ converge quickly to a value of about $\overline{V_\alpha}=7.17$ which does not change after 30,000 simulations. With $m=10$, the sparse level 0 consists of just one basis function (*i.e.* a constant) and the resulting exercising decision is almost trivial. Sparse level $L^*=1$ consists of 21 basis functions. This allows for a more sophisticated strategy with a better utilization of the option. After about 300,000 simulations, the option value saturates at 7.62 for both polynomial and piecewise linear basis functions. The sparse level $L^*=2$ with 241 basis functions results in similar values after 1,000,000 simulations. For the polynomial sparse basis functions, a third level $L^*=3$ with 2001 basis functions already exceeds our available computational resources, such that the saturation level could not be computed. The values for polynomial sparse level 3 are thus not presented. One

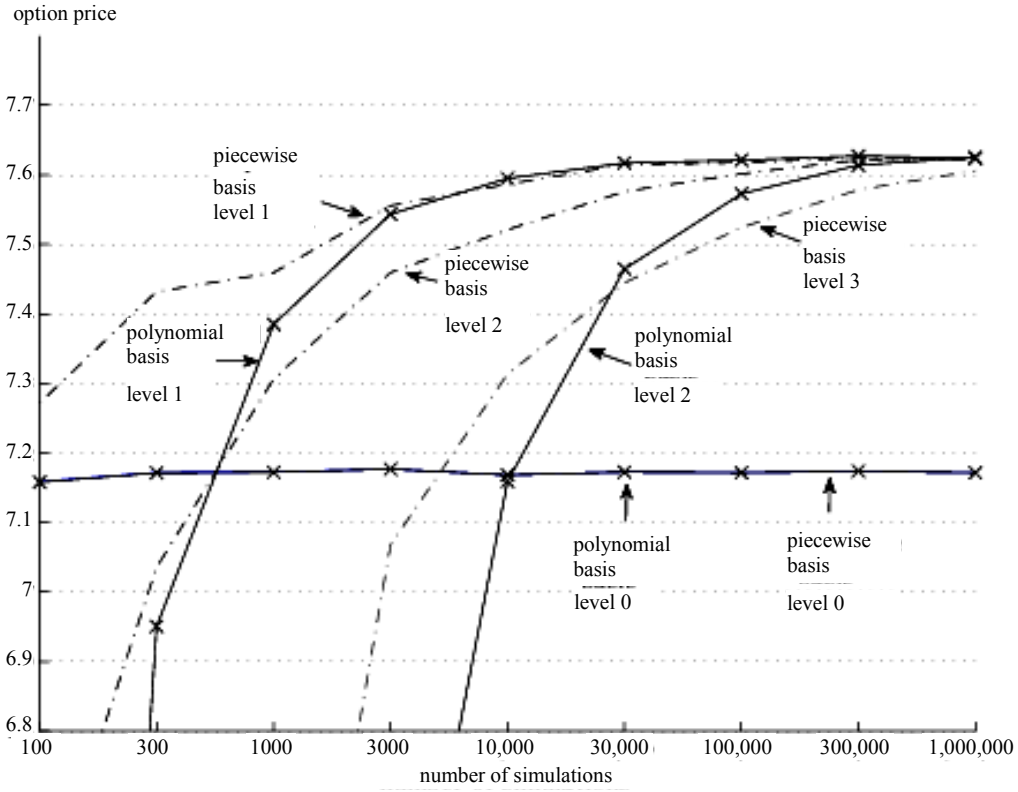

Figure 1. The option values for MWAO. The figure shows MWAO values estimated by least squares Monte Carlo combined respectively with polynomial sparse basis functions and with piecewise linear sparse basis functions. The sparse level ranges from 0 to 3 and the number of simulation paths runs from 1×10^2 to 1×10^6. In the figure, "polynomial basis level #" refers to the MWAO values estimated with the polynomial basis functions at a sparse level #, where # is the number for the sparse level. Similarly, "piecewise basis level #" points to MWAO values estimated with the piecewise linear basis functions at the sparse level #. The data for the MWAO are specified in Table 1.

thing worth mentioning is that higher sparse levels initially perform inferior to lower sparse levels due to the over-fitted regression functions.

The corresponding values in **Figure 1** are presented in **Table 2** for polynomial sparse basis functions and in **Table 3** for piecewise linear sparse basis functions. The mean values of a series of valuations $\widetilde{V_\alpha}^i\left(n,L^*,10\right)$, $i=1,\cdots,I$, is denoted by $\overline{V_\alpha}\left(n,L^*,10\right)$ (30), the standard deviation of the series across I valuations is denoted by $\hat{\sigma}$. For a single evaluation $\widetilde{V_\alpha}^i\left(n,L^*,10\right)$ with LSM, $\hat{\sigma}$ can be seen as a measure of how close the value is to the mean of I valuations. Thus $\hat{\sigma}$ does not measure the error relative to the true option value. The mean estimate $\overline{V_\alpha}\left(n,L^*,10\right)$ (30) will be biased lower than the true values due to the suboptimal estimate of the optimal exercise strategy $\mathbb{E}\left[\tilde{V}_{t_{i+1}}\middle|\tilde{S},t_i\right]$. At sparse level $L^*=1$ both polynomial and piecewise linear basis functions deliver similar MWAO prices up to two decimal points after 3×10^5 sample paths. At sparse level $L^*=2$ with 1×10^6 sample paths the MWAO prices are the same up to three decimal point in both cases. Based on the results, both types of sparse basis functions solve our high-dimensional least squares problem and the prices converge to a level of 7.62 at 1×10^6 sample paths for sparse levels $L^*=1$ and sparse level $L^*=2$. Since the polynomial sparse basis functions are easier to construct and easier to implement, we recommend them as the bases of choice for our MWAO pricing problem.

The price of our moving average window option has three main sources of error: the number of simulation paths n, the level of the function basis L^* and the number of integration samples m. The best possible approximation would have to reduce the errors originated from using these limiting parameters.

5. Conclusion

This paper presents a general and convergent algorithm for pricing early-exercisable moving window Asian options. The computational difficulty of the pricing task stems from what defines the option's underlying: an either discretely or continuously sampled average over a moving window. We have applied a generalized framework to solve this high-dimensional problem by combining the least-squares Monte Carlo method with the

Table 2. The option value for MWAO with data in Table 1 estimated by least squares Monte Carlo with polynomial sparse basis functions. The mean estimate $\overline{V}_\alpha(n,L^*,10)$ (30) is based on sample n, sparse level L^*, number of observations in the averaging window $m=10$, and the weight function α. The number of samples n ranges from 1×10^2 to 1×10^6, and the sparse level L^* goes from 0 to 2. The standard deviation of the series $\widetilde{V}_\alpha^{\,i}(n,L^*,10)$ is denoted by $\hat{\sigma}$.

\ level L^*	$L^*=0$		$L^*=1$		$L^*=2$	
# samples n	$\overline{V}_\alpha(n,0,10)$	$\hat{\sigma}$	$\overline{V}_\alpha(n,1,10)$	$\hat{\sigma}$	$\overline{V}_\alpha(n,2,10)$	$\hat{\sigma}$
1×10^2	7.158912	0.234	4.378564	0.470	4.446816	
3×10^2	7.171716	0.134	6.950492	0.264	3.399907	
1×10^3	7.172727	0.073	7.385057	0.114	3.325619	0.148
3×10^3	7.176859	0.043	7.543586	0.061	6.296143	0.083
1×10^4	7.168296	0.022	7.594902	0.034	7.159272	0.041
3×10^4	7.172753	0.014	7.617092	0.018	7.464719	0.018
1×10^5	7.171619	0.007	7.621127	0.010	7.572837	0.009
3×10^5	7.173722	0.005	7.627301	0.005	7.613978	0.006
1×10^6	7.171411	0.004	7.624000	0.003	7.625142	0.002

Table 3. The option value for MWAO with data in Table 1 estimated by least squares Monte Carlo with piecewise linear sparse basis functions. The mean estimate $\overline{V}_\alpha(n,L^*,10)$ (30) is based on sample n, sparse level L^*, number of observations in the averaging window $m=10$, and the weight function α. The number of samples n ranges from 1×10^2 to 1×10^6, and the sparse level L^* goes from 0 to 3. The standard deviation of the series $\widetilde{V}_\alpha^{\,i}(n,L^*,10)$ is denoted by $\hat{\sigma}$.

\ level L^*	$L^*=0$		$L^*=1$		$L^*=2$		$L^*=3$	
# samples n	$\overline{V}_\alpha(n,0,10)$	$\hat{\sigma}$	$\overline{V}_\alpha(n,1,10)$	$\hat{\sigma}$	$\overline{V}_\alpha(n,2,10)$	$\hat{\sigma}$	$\overline{V}_\alpha(n,3,10)$	$\hat{\sigma}$
1×10^2	7.157866	0.257	7.271832	0.293	6.527391	0.427	5.631500	0.468
3×10^2	7.171375	0.172	7.432182	0.146	7.035495	0.154	6.130585	0.240
1×10^3	7.172456	0.062	7.459452	0.094	7.306664	0.080	6.103082	0.354
3×10^3	7.176906	0.050	7.555757	0.052	7.459485	0.051	7.067918	0.077
1×10^4	7.168270	0.029	7.586647	0.046	7.521104	0.043	7.313780	0.036
3×10^4	7.172753	0.016	7.614570	0.020	7.576200	0.019	7.444699	0.021
1×10^5	7.171629	0.009	7.617702	0.012	7.601458	0.011	7.524500	0.010
3×10^5	7.173733	0.005	7.624306	0.007	7.621602	0.007	7.578699	0.006
1×10^6	7.171410	0.003	7.621104	0.004	7.625030	0.004	7.606029	0.004

sparse grid basis functions. The sparse grid technique has been specifically developed as a cure to the "curse of dimensionality" problem. It allows more efficient selection of basis functions and can successfully approximate high-dimensional functions with less computational effort. We have used both the polynomial and the piecewise linear sparse basis functions in the least-squares regressions and found that the results converge to values up to two decimal points, independent of the type of basis functions used. We recommend the polynomial sparse basis as the basis of choice for this type of pricing problems since they are easier to construct and easier to implement. The approach presented in this paper can be generalized to pricing other high-dimensional early-exercisable derivatives that use a moving average as the underlying.

REFERENCES

[1] S. Achatz, "Higher Order Sparse Grid Methods for El-

liptic Partial Differential Equations with Variable Coefficients," *Computing*, Vol. 71, No. 1, 2003, pp. 1-15. http://dx.doi.org/10.1007/s00607-003-0012-8

[2] V. Bathelmann, E. Novak and K. Ritter, "High Dimensional Polynomial Interpolation on Sparse Grids," *Advances in Compuational Mathematics*, Vol. 12, No. 4, 2000, pp. 273-288. http://dx.doi.org/10.1023/A:1018977404843

[3] M. Bernhart, P. Tankov and X. Warin, "A Finite Dimensional Approximation for Pricing Moving Average Options," *Journal on Financial Mathematics*, Vol. 2, No. 1, 2011, pp. 989-1013. http://dx.doi.org/10.1137/100815566

[4] R. Bilger, "Valuing American-Asian Options Using the Longstaff-Schwartz Algorithm," Msc Thesis in Computational Finance, Oxford University, Oxford, 2003.

[5] F. Black and M. Scholes, "The Pricing Of Options and Corporate Liabilities," *Journal on Political Economy*, Vol. 81, No. 3, 1973, pp. 637-659. http://dx.doi.org/10.1086/260062

[6] T. Bonk, "A New Algorithm for Multi-Dimensional Adaptic Numerical Quadrature," In: W. Hackbush, Ed., *Adaptive Methods—Algorithms, Theory and Applications*, *Notes on Numerical Fluid Mechanics*, Vieweg + Teubner, Braunschweig, 1994, pp. 54-68.

[7] M. Broadie and M. Cao, "Improved Lower and Upper Bound Algorithms for Pricing American Options by Simulation," *Quantitative Finance*, Vol. 8, No. 8, 2008, pp. 845-861. http://dx.doi.org/10.1080/14697680701763086

[8] H.-J. Bungartz, "A Multigrid Algorithm for Higher Order Finite Elements on Sparse Grid," *Electronic Transactions on Numerical Analysis*, Vol. 6, 1997, pp. 63-77.

[9] J. F. Carriere, "Valuation of the Early-Exercise Price for Options Using Simulations and Nonparametric Regression," *Insurance: Mathematics and Economics*, Vol. 19, No. 1, 1996, pp. 19-30. http://dx.doi.org/10.1016/S0167-6687(96)00004-2

[10] M. Dai, P. Li and J. E. Zhang, "A Lattice Algorithm for Pricing Moving Average Barrier Options," *Journal of Economic Dynamics and Control*, Vol. 34, No. 3, 2010, pp. 542-554. http://dx.doi.org/10.1016/j.jedc.2009.10.008

[11] S. Dirnstorfer, A. J. Grau and H. Li, "ThetaML Handbook," edition winterwork, 2012.

[12] S. Dirnstorfer and A. J. Grau, "Computer Aided Finance: Another Journey in the Quest for the Holy Grail of Financial Engineering," *WILMOTT Magazine*, 2008, pp. 68-73.

[13] J. Garcke, M. Griebel and M. Thess, "Data Mining with Sparse Grids," *Computing*, Vol. 67, No. 3, 2001, pp. 225-253. http://dx.doi.org/10.1007/s006070170007

[14] K. Hallatschek, "Fouriertransformation auf Dünnen Gittern mit Hierarchischen Basen," *Numerische Mathematik*, Vol. 63, No. 1, 1992, pp. 83-97. http://dx.doi.org/10.1007/BF01385849

[15] C.-H. Kao and Y.-D. Lyuu, "Pricing of Moving Average-Type Options with Applications," *Journal of Futures Markets*, Vol. 23, No. 5, 2003, pp. 415-440. http://dx.doi.org/10.1002/fut.10072

[16] A. Klimke and B. Wohlmuth, "Algorithm 847: Spinterp: Piecewise Multilinear Hierarchical Sparse Grid Interpolation in Matlab," *ACM Transactions on Mathematical Software*, Vol. 31, No. 4, 2005, pp. 561-579. http://dx.doi.org/10.1145/1114268.1114275

[17] F. A. Longstaff and E. S. Schwartz, "Valuing American Options by Simulation—A Simple Least-Squares Approach," *The Review of Financial Studies*, Vol. 14, No. 1, 2001, pp. 113-147. http://dx.doi.org/10.1093/rfs/14.1.113

[18] B. D. Martin, "Radial Basis Functions: Theory and Implementations," Cambridge University Press, Cambridge, 2003.

[19] M. Griebel, P. Oswald and T. Schiekoffer, "Sparse Grids for Boundary Integral Equations," *Numerische Mathematik*, Vol. 83, No. 2, 1999, pp. 279-312. http://dx.doi.org/10.1007/s002110050450

[20] B. Nicolas, "Elements of Mathematics, Algebra I," Springer-Verlag, Berlin, 1989.

[21] C. Reisinger, "Numerische Methoden für Hochdimensionale Parabolische Gleichungen am Beispiel von Optionspreisaufgaben," Dissertation, Ruprecht-Karls-Universität, Heidelberg, 2004.

[22] S. Schraufstetter, "A Pricing Framework for the Efficient Evaluation of Financial Derivatives Based on Theta Calculus and Adaptive Sparse Grids," Dr. Hut, 2012.

[23] S. A. Smolyak, "Quadrature and Interpolation Formulas for Tensor Products of Certain Classes of Functions," *Doklady Akademii Nauk SSSR*, Vol. 148, 1963, pp. 1042-1043. English Russian Translation: *Soviet Mathematics Doklady*, Vol. 4, 1963, pp. 240-243.

[24] C. Zenger, "Sparse Grids," In: W. Hackbusch, Ed., *Notes on Numerical Fluid Mechanics*, Vol. 31, Vieweg, Braunschweig, 1991, 1990, pp. 241-251.

Appendix

ThetaML Implementation of the MWAO

This appendix gives a ThetaML (Theta Modeling Language)[1] implementation of the MWAO pricing problem described in the text. To help understand the code, we first provide a brief description of the ThetaML language and some ThetaML specific commands and operators used in the models.

ThetaML supports standard control structures such as loops and if statements. It operates on a virtual timing model with the *theta* command. The *theta* command describes the time-determined behavior of financial derivatives. It allows time to pass. The *fork* command makes it possible to model simultaneous processes and enables cross-dependencies among the stochastic variables. The future operator "!" allows forward access to the future values of a variable, this operation is based on forward algorithms.

ThetaML modularizes the pricing task of MWAO into a simulation model for the stock prices and the numeraire, an exercise model for obtaining future early-exercise values, and a pricing model for computing the MWAO prices. ThetaML virtually parallels and synchronizes the processes—stock prices, numeraire, early exercise cash flows and MWAO prices—to the effect that it is as if these four processes step forward in time as they would have behaved in real life financial markets. ThetaML specific commands and operators are briefly explained in the models where they appear. Process variables in ThetaML models, such as "S", "CUR", "A", implicitly incorporates scenario- and time-indices.

```
1   model StateProcesses
2   % This model simulates stock prices that follow a Geometric Brownian motion process,
3   % and a numeraire process that are discounted at a constant interest rate;
4   % the arguments are in troduced into the model after the "import" keyword, results
5   % computed by the model are exported
6         import   S0       "Initial stock prices"
7         import   r        "Risk_free interest rate"
8         import   sigma    "Volatility of stock prices"
9         export   S        "Stock price process"
10        export   CUR      "Numeraire process in currency CUR"
11
12        % initialize the stock prices at    "S0"
13        S = S0
14        % initialize the numeraire at 1 CUR
15        CUR = 1
16        % "loop inf" runs till the expiry time of a pricing application
17        loop inf
18            % "@dt" extracts the smallest time interval from this model time to the next
19            theta @dt
20            % updates the GBM process of the stock prices for the time step "@dt"
21            S = S * exp((r - 0.5*sigma^2)*@dt + sigma*sqrt(@dt)*randn())
22            % update the numeraire for the time step "@dt"
23            CUR = CUR*exp(-r*@dt)
24        end
25  end
```

```
1   model MWAOExerciseValue
2   % This model computes the Early exercise values for MWAO, assuming daily exercises.
3   % Early exercises are possible after an initial allowance of a window length "m"
4         import   S                "Stock prices"
5         import   CUR              "Numeraire in currency CUR"
6         import   m                "Window length"
7         import   T                "Option maturity time"
8         export   ExerciseValue    "Exercise value"
9         export   TimeGrid         "A set of exercise times"
```

[*]ThetaML is a payoff description language that explicitly incorporates the passage of time. Product path dependencies, settlements, and early exercises are all appropriately addressed. ThetaML also offers the benefit to specify the complete structure of a structured product independent of the underlying stochastic processes. For details on ThetaML, please consult the references Dirnstorfer *et al*. [22], Dirnstorfer *et al*. [23] and Schraufstetter [24].

```
10
11        % initialize an array with length “m” to hold a past window of “m” stock prices
12      C = 0 * [1:m]
13      % initialize the moving average to “S/m”, where “S” are the time 0 stock prices
14      A = S/m
15      % initialize “C[m]” to the time 0 stock prices “S”
16      C[m] = S
17
18      ExerciseValue = 0
19      index = 1
20      % early exercise times range from “1/250” to “T”, equally spaced at “1/250”
21      TimeGrid = [1/250:1/250:T]
22
23      % “t” loops through the exercise time sand takes the value of the pointed element
24      loop t: Time Grid
25          % “theta” advances “t_@time” time units to the next time point
26          theta t_@time
27          % update the moving average by adding “S” evaluated at this time divided by “m”;
28          % at the end of “m” periods, subtract “S/m” evaluated at the start window time
29          A = A + S/m - C[index]/m
30          % record the stock price “S” at this time in “C”
31          C[index] = S
32          % increment the index by 1
33          index = index + 1
34          % reset “index” to 1 if “index” is bigger than “m”
35          if index > m
36              index = 1
37          end
38
39          % store the discounted exercise values for in-the-money paths
40          if @time >= 1/250 * (m-1)
41              ExerciseValue = max(A-S,0)*CUR
42          end
43
44          theta @dt
45          % reset Exercise Value to 0 at non-exercising times
46          ExerciseValue = 0
47
48      end
49  end
```

```
1   model MWAOPrice
2   % This model returns the MWAO prices across all the Monte-Carlo paths, using the
3   % early-exercise strategies obtained from the model MWAO Exercise Values
4         import Exercise Value “Exercise value”
5         import TimeGrid “A set of exercise times”
6         export Price “MWAO prices”
7
8         % at time 0, “Price” is assigned expected value of “value!”; the future operator “!” accesses the
9         % values of the variable “value” determine data later in stance
10        Price = E(value!)
11        % loop through the exercise times
12        loop t: TimeGrid
13            % early exercise evaluations, the “E” function computes the conditional expected
14            % discounted “value!”, using the Least Squares Monte Carlo regressions combined
15            % with the Sparse Grid type basis functions
```

```
16          if E(value!) < ExerciseValue
17              value = ExerciseValue
18          end
19          % time passing of "t-@time" with the "theta" command
20          theta t-@ time
21      end
22      % at the option maturity time, set the option pay off values
23      if ExerciseValue>0
24          value = ExerciseValue
25      else
26          value = 0
27      end
28  end
```

The stock prices and numeraire are first simulated in the external models "StateProcesses", then imported as processes into the exercise model "MWAOExercise Values" to compute future early-exercise values. The early-exercise cash flows are next imported as a process into the pricing model "MWAOPrice" to determine the MWAO price.

The ThetaML future operator "!" appears in the model "MWAOPrice". It allows the possibility to use the variable "value" at the model time when its values are not pre-assigned. Computationally, whenever the compiler encounters the future operator "!", it evaluates the codes backward in time. So, when computing the time 0 option price, the compiler goes from the option maturity back to time 0, and assigns the computed time 0 value to "Price".