Les Cahiers du GERAD ISSN: 0711–2440

A simulation-and-regression approach for dynamic programming, and its application to portfolio choice

E. Delage, M. Denault,
J.-G. Simonato

G–2014–42

June 2014

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A simulation-and-regression approach for dynamic programming, and its application to portfolio choice

Erick Delage[a]
Michel Denault[a]
Jean-Guy Simonato[b]

[a] *GERAD & Department of Decision Sciences, HEC Montréal, Montréal (Québec) Canada, H3T 2A7*

[b] *Department of Finance, HEC Montréal, Montréal (Québec) Canada, H3T 2A7*

erick.delage@hec.ca
michel.denault@hec.ca
jean-guy.simonato@hec.ca

June 2014

Les Cahiers du GERAD
G–2014–42

Abstract: Simulation-and-regression algorithms have become a standard tool for solving dynamic programs in many areas, in particular financial engineering and computational economics. In virtually all cases, the regression is performed on the state variables, for example on current market prices. However, it is possible to regress on decision variables as well, and this opens up new possibilities. We present numerical evidences of the performance of such an algorithm, in the context of dynamic portfolio choices in discrete-time (and thus incomplete) markets. The problem is fundamentally the one considered in some recent papers that also use simulations and/or regressions: discrete time, multiperiod reallocation, and maximization of terminal utility. In contrast to that literature, we regress on decisions variables and we do not rely on Taylor series expansions nor derivatives of the utility function. Only basic tools are used, bundled in a dynamic programming framework: simulations, —which can be black-boxed—, as a representation of exogenous state variables dynamics; regression surfaces, as non-anticipative representations of expected future utility; and nonlinear or quadratic optimization, to identify the best portfolio choice at each time step. The resulting approach is simple, highly flexible and offers good performances in time and precision.

Key Words: Simulation-and-regression methods, least-squares Monte Carlo methods, dynamic programming, portfolio choice, portfolio optimization.

Résumé : Les algorithmes de simulation et régression sont désormais un outil de base dans plusieurs domaines d'application de la programmation dynamique, notamment en ingénierie financière et en économie computationnelle. Presque toujours, la régression est effectuée sur les variables d'état, les prix du marché par exemple. Cependant, il est possible de faire la régression sur les variables de décision en plus, ce qui présente de nouvelles opportunités. Nous présentons dans cet article des résultats numériques illustrant la performance d'un tel algorithme, tel qu'appliqué à l'optimisation dynamique de portefeuilles en temps discret (marchés incomplets). Le problème est le même que celui qui a été considéré dans quelques articles récents qui utilisent aussi la simulation et la régression : temps discret, réallocation sur plusieurs périodes, maximisation de l'utilité terminale. Par opposition à la littérature en question, nos régressions sont aussi effectuées sur les variables de décision, et nous n'utilisons pas les séries de Taylor ni les dérivées de la fonction d'utilité. Seuls des outils de base sont utilisés, dans un cadre de programmation dynamique : des simulations, qui peuvent être en boîte noire, pour représenter la dynamique des variables d'état exogène; des surfaces de régression, pour représenter (de façon non-anticipative) l'utilité future espérée; et l'optimisation nonlinéaire ou quadratique, pour identifier le portefeuille idéal à chaque pas de temps. L'approche qui en résulte est simple, très flexible, et offre de bonnes performances tant en vitesse qu'en précision.

Acknowledgments: The authors acknowledge the financial support of HEC-Montréal and NSERC.

1 Introduction

The characterization of portfolio choices is a classical and important topic in finance. With few exceptions, the vast majority of dynamic portfolio choice models (multi-period setting with a possible portfolio reallocation at each time point), do not have closed form solutions and must rely on numerical methods. Examples of these can be found in Brennan et al. (1997) with the numerical solution of partial differential equations, Dammon et al. (2001) with binomial trees, or Balduzzi and Lynch (1999) with numerical integrations by quadratures. The main drawbacks of these lattice approaches is the difficulty to handle many state variables and realistic stock return processes.

More recently, some authors have proposed the use of Monte Carlo simulations to handle these portfolio problems. These methods offer a high level of flexibility in modeling the behaviour of the stochastic variables. Typically, they can even be black-boxed, meaning that no information is required about the underlying stochatic processes, as long as simulation samples can be generated. One example of the use of simulations is Detemple et al. (2003), who use Malliavin calculus and Monte Carlo simulations to compute optimal portfolios in a continuous-time (complete-market) dynamic setting. Their approach can handle a large number of state variables and is shown to converge to the optimal solutions. This method is however best suited to complete-market settings with diffusion processes. Incomplete market problems can be handled, but only if an analytical solution is available for the dual problem.

In a discrete-time incomplete market setting, Brandt et al. (2005) and Garlappi and Skoulakis (2010) propose simulation-and-regression algorithms to compute optimal portfolios for a variety of problems. These algorithms use Taylor series approximations of the value function at every time point, and a combination of least-squares regressions and Monte Carlo simulations or quadratures to obtain the required expected values entering these approximate value functions. The algorithm proposed here is closely related to these last two papers and falls within the category of simulation-and-regression approaches. However, unlike the above algorithms, we avoid the need of Taylor series approximation of the value function. This gives a simpler algorithm which does not require the work and complication associated with the computation of the derivatives of the utility function. These simplifications are achieved by introducing in the regressions the weights of the assets as independent variables. The resulting regression surfaces then become functions of the weights, which can be optimized to find the optimal portfolios.

As with other simulation-and-regression procedures, we use a recursive dynamic programming algorithm. The proposed approach also has the capacity to handle more realistic discrete-time processes than lattice methods. For example, GARCH processes with non-normal errors or discrete-time processes with jumps can be used in these procedures without much modifications. However, our algorithm can also handle utility functions that are not differentiable everywhere, such as piecewise linear utility functions. Although it is impossible to avoid the curse of dimensionality, our tests indicate that our method suffers much less from this curse than quadrature methods.

The approach developed in this paper shares many similarities with an approach independently developed in Nijman, Werker, Koijen (2007) and to which Koijen, Nijman, Werker (2010) refer.[1] The work presented here, however, differs in many ways. First, our focus is on exploring the possibilities of the method, and we provide results on a series of examples of portfolio choice problems that are benchmarked with numerical quadrature or analytical solutions. These examples show various difficulties coming from the size of the problem (number of risky assets, number of periods), the returns stochastic behavior (serial dependence) and utility function (dependence on wealth). We also illustrate the limits of Taylor series approaches with an example on conditional Value-at-Risk minimisation. Second, we find that with a careful implementation, the computation of the regression surfaces themselves (not an approximation) is not "computationally unattractive" as Koijen et al. suggest. See more details in Section 3.4. Third, we work directly on the optimization problems, not on a set of Karush-Kuhn-Tucker equations. This offers more insight and flexibility in choosing the bases and incorporating constraints, and is arguably simpler as well. Fourth, we introduce an inverse utility transformation similar to that proposed in Garlappi and Skoulakis (2010). This transformation allows

[1] We became aware that Nijman, Werker, and Koijen had proposed regressing on the decision variables, in the very last production stage of this paper, when we found their working paper of 2007.

for an important reduction in the number of discrete wealth points required to achieve precise results. Fifth, we compare two different approaches to evaluate a policy, dubbed "realized values" and "regression surface values". While the regression surface approach is believed to bear more bias in option pricing, our test show no such evidence in portfolio choice. Sixth, we give preliminary evidence that a two-step procedure with regression surfaces on smaller domains can provide enhanced precision, see example 2.

In the next section, we outline the problem of the investor. We describe our approach in Section 3 with a simple example first, to provide the intuition, and the general algorithm next. Numerical results are presented in Section 4 and the paper concludes with Section 5. An appendix section extends the simple example of Section 3.

2 Problem description

The algorithm could be used on a variety of problems of dynamic programming but in this paper we use it on a specific problem. In this section, we describe the problem, indicate the main hypotheses, and set the notation up.

We consider an investor endowed with wealth W_0 at time $t = 0$ who wants to maximize his expected utility of terminal wealth at date T . The investor can trade N risky assets and a risk-free security at times $t = 0, 1, \ldots, T-1$, and there are no frictions such as transaction costs or taxes. With these assumptions the investor's problem can be expressed as

$$V_0(W_0, S_0) = \max_{\mathbf{x}_{\min} \leq \{\mathbf{x}_t\}_{t=0}^{T-1} \leq \mathbf{x}_{\max}} E_0\left[u(W_T)\right]$$

with the following constraints for all t :

$$W_{t+1} = W_t \cdot \left(\mathbf{x}_t^\top \mathbf{r}_{t+1} + R_f\right)$$

where E_t denotes the expectation conditional on the information available at time t (here the initial wealth W_0 and initial state variable value S_0), $u(\cdot)$ is a concave utility function with the usual regularity conditions, R_f is the gross risk-free rate with simple compounding, $\mathbf{r}_{t+1}$ is a $N \times 1$ vector of simple returns in excess of the risk-free rate for the time period between t and $t+1$. Portfolio weights between times t and $t+1$ are represented by the $N \times 1$ vector $\mathbf{x}_t$, with lower and upper bounds $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$, respectively. Other constraints can be handled; linear ones, for example, have little impact on performance. We assume that the vector of excess stock return is generated from a known discrete-time stochastic process.

In the above problem, wealth is an endogenous variable that varies stochastically with time according to the portfolio weights $\mathbf{x}_t$ and the exogenous random returns $\mathbf{r}_{t+1}$. Depending on the utility function, the current wealth level may or may not influence the optimal portfolio weights. Similarly, depending on the return process, the current state of the process might influence the optimal weights.

Using the principle of optimality of dynamic programming, the above value function can be rewritten as the following recursion:

$$V_t(W_t, S_t) = \max_{\mathbf{x}_{\min} \leq \mathbf{X}_t \leq \mathbf{x}_{\max}} E_t\left[V_{t+1}\left(W_t \cdot \left(\mathbf{x}_t^\top \mathbf{r}_{t+1} + R_f\right), S_{t+1}\right)\right] \tag{1}$$

with the terminal condition

$$V_T(W_T, S_T) = u(W_T).$$

3 A simulation-and-regression algorithm that regresses on decision variables

In this section, we describe our solution approach to solve the above dynamic programming recursion. Subsection 3.1 discusses the three main tasks at each time step of the recursion. Subsection 3.2 introduces our algorithm with a short, intuitive numerical example. Subsection 3.3 provides the algorithm itself, and Subsection 3.4 discusses some implementation details.

3.1 Overview

Solving the dynamic program (1) with our approach involves three main tasks at each time step, as discussed below. It is important to first make a distinction between the state variable "wealth" and other state variables such as past returns (or dividends, economic indicators, etc.). The former is endogenous in that its value depends on portfolio decisions; the later are exogenous, in that they are independent of portfolio decisions. Endogenous state variables do not lend themselves naturally to simulation-and-regression schemes (though see Denault, Simonato, Stentoft (2013)) As such, wealth is discretized, as (all) state variables usually are in classical dynamic programming. Exogenous state variables as for them are treated by simulation-and-regression.

The three main tasks that build our approach are as follows.

Build a representation of the value function. In the absence of analytical solutions, the value function V_t needs to be represented compactly. We first build a parametric representation of future utilities, in terms of bases formed with both exogenous state variables *and* the decision variables. Using the exogenous state variables is rather natural, see the second task (expectations). Using the decisions variables as bases is not common, but the rationale is simple, see below. Note that this parametric representation is *not* an approximate value function, it is broader in scope: it approximates future utilities under any decision, optimal or suboptimal. See the third task.

Computation of expectations. Expectations are used to account for the unknown future. They enforce non-anticipativity, i.e. decisions must not rely on future information. This is where simulations and regressions do their work. Monte Carlo paths simulations serve to represent the dynamics of the exogenous state variables and returns. The parametric value function approximation is adjusted by regression to implement the expectation.

Optimization of the portfolio positions. The above parametric representation of future utilities, conditional on any simulated state variable value and wealth level, can be maximized with respect to the portfolio decision. This optimal portfolio decision can be used to compute the value function (for that specific state variable value, and wealth level). To conclude, our value function representation at any point in time is therefore a discrete set of numbers, each associated with a specific path of the exogenous state variable, and a specific wealth level.

3.2 Introductory example

An intuitive understanding of the algorithm is best conveyed by a simple example. Consider the two-period case of a portfolio consisting of one risky and one risk-free asset with a constant gross risk-free rate $R_f = 1.01$. The investor, endowed with current wealth $W_0 = 1$ at time $t = 0$, wants to maximize the utility of his wealth at time $t = 2$. Asset allocation takes place at time $t = 0$ and $t = 1$. No short sale nor borrowing is allowed.

The algorithm requires a preliminary step where grids of discrete portfolio weights and wealths are built, and excess returns are simulated. Let us use the grid of portfolio weights

$$x = \begin{bmatrix} 0.0 & 0.3 & 0.7 & 1.0 \end{bmatrix}^\top,$$

while the grid of wealth $w_{k,t}$ $t \in \{0, 1\}$ is:

$t = 0$	$t = 1$
	$w_{1,1} = 1.3$
$w_0 = 1$	
	$w_{2,1} = 0.8$

We keep this introduction as simple as possible and use only two sample paths of excess returns at each period (Period 1 from $t = 0$ to $t = 1$, Period 2 from $t = 1$ to $t = 2$)

Path	Period 1	Period 2
a	$r_1^{(a)} = 0.05$	$r_2^{(a)} = 0.02$
b	$r_1^{(b)} = -0.05$	$r_2^{(b)} = -0.02$

In a second step, the recursive dynamic programming procedure is implemented with the above quantities. Starting at $t = 1$, one period before maturity, we perform the following operations, for each state of wealth at this time period. We first compute a sample of wealth at $t = 2$. For example, from the wealth level $w_{1,1} = 1.3$, we compute a wealth at $t = 2$ using the first weight and path a of the simulated excess return :

$$\begin{aligned} W\left(x_1, r_2^{(a)}\right) &= w_{1,1} \cdot \left(x_1 r_2^{(a)} + R_f\right) \\ &= 1.3 \times (0.00 \times 0.02 + 1.01) = 1.3130. \end{aligned}$$

Using all the possible combinations of simulated returns at $t = 2$ and portfolio weights on the grid, the end-of-period sample of wealths at $t = 2$ is:[2]

$$\begin{aligned} \left[W(x_1, r_2^{(a)}),\, W(x_2, r_2^{(a)}),\, W(x_3, r_2^{(a)}),\, W(x_4, r_2^{(a)}),\, W(x_1, r_2^{(b)}),\, W(x_2, r_2^{(b)}),\, W(x_3, r_2^{(b)}),\, W(x_4, r_2^{(b)})\right] = \\ \left[\begin{array}{cccccccc} 1.3130 & 1.3208 & 1.3312 & 1.3390 & 1.3130 & 1.3052 & 1.2948 & 1.2870 \end{array}\right]. \quad (2) \end{aligned}$$

Computing the utility of these end-of-period wealths with the constant absolute risk aversion (CARA) utility function $u(W) = -\exp(-\gamma W)$ with $\gamma = 3$, the following linear regression system can be formed:

$$\begin{bmatrix} -0.0195 \\ -0.0190 \\ -0.0184 \\ -0.0180 \\ -0.0195 \\ -0.0199 \\ -0.0206 \\ -0.0210 \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0 & 0.0^2 \\ 1.0 & 0.3 & 0.3^2 \\ 1.0 & 0.7 & 0.7^2 \\ 1.0 & 1.0 & 1.0^2 \\ 1.0 & 0.0 & 0.0^2 \\ 1.0 & 0.3 & 0.3^2 \\ 1.0 & 0.7 & 0.7^2 \\ 1.0 & 1.0 & 1.0^2 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \end{bmatrix},$$

where the elements on the left end side are the utility of the computed wealths at $t = 2$, and the lines of the independent variable matrix are formed with a constant, and the powers up to 2 of the portfolio weights. In this system, ε_i are random errors with zero expected value, and β_1 to β_3 are unknown coefficients, whose estimated values obtained with an ordinary least square regression are:

$$\widehat{\beta}_1 = -0.0195, \quad \widehat{\beta}_2 = 0.0001, \quad \widehat{\beta}_3 = -0.0001.$$

Using these estimated coefficients, the approximate expected utility given a $w_{1,1} = 1.3$ wealth level at $t = 1$, can be written as a continuous function of the portfolio weights, which can be used to find the optimal proportion as :

$$\widehat{x}_{1,1} = \arg \max_{0 \le x \le 1} \; -0.0195 + 0.0001x - 0.0001x^2.$$

The optimal portfolio weight can easily be verified to be $\widehat{x}_{1,1} = 0.5$ with a function value at the optimum of $\widehat{v}_{1,1} = -0.0195$. This represents the highest achievable expected utility value, conditional on the information available at $t = 1$, and a wealth value of 1.3. It is thus a numerical estimate of a point on the value function for a wealth of 1.3.

Doing similar calculations for the other state of wealth $w_{2,1} = 0.8$ at $t = 1$, we obtain this table summarizing the computed quantities at $t = 1$ for each state of wealth.

In this example, that the *optimal portfolios* be identical for both wealth level is a numerical coincidence, largely due to the rounding that keeps the example simple. That the *optimal portfolios* be identical for different paths at the same wealth level is normal, and due to the fact that we do not regress on exogenous

[2] All computed values in this section have been performed with values rounded to the fourth digit.

Table 1: Values associated to two wealths at $t = 1$

$w_{k,1}$	$\widehat{x}_{k,1}$	$\widehat{v}_{k,1}^{(j)}$
1.3	$\widehat{x}_{1,1}^{(a)} = 0.5$	$\widehat{v}_{1,1}^{(a)} = -0.0195$
	$\widehat{x}_{1,1}^{(b)} = 0.5$	$\widehat{v}_{1,1}^{(b)} = -0.0195$
0.8	$\widehat{x}_{2,1}^{(a)} = 0.5$	$\widehat{v}_{2,1}^{(a)} = -0.0886$
	$\widehat{x}_{2,1}^{(b)} = 0.5$	$\widehat{v}_{2,1}^{(b)} = -0.0886$

state variables in this simple example; if we did introduce exogenous state variables, there would be one optimization problem per path (and wealth level), and the solutions would likely be different. That the *values* be the same for different paths at the same wealth level, is also normal; note however that in an alternate specification of our algorithm, illustrated at the end of this section, the values would likely be different, *even though we do not regress on exogenous state variables.* Compare Table 2 below.

Using the above values, the optimal weight at $t = 0$ can be computed using a similar procedure started at the wealth value $w_0 = 1$. We first compute a sample of wealths at $t = 1$ using the combinations of simulated returns and portfolio weights. This sample of wealths at $t = 1$ is

$$\left[W(x_1, r_1^{(a)}),\, W(x_2, r_1^{(a)}),\, W(x_3, r_1^{(a)}),\, W(x_4, r_1^{(a)}),\, W(x_1, r_1^{(b)}),\, W(x_2, r_1^{(b)}),\, W(x_3, r_1^{(b)}),\, W(x_4, r_1^{(b)})\right] = \left[\begin{array}{cccccccc} 1.0100 & 1.0250 & 1.0450 & 1.0600 & 1.0100 & 0.9950 & 0.9750 & 0.9600 \end{array}\right]. \quad (3)$$

Using these, we can generate a sample of points on the value function at $t = 1$ that will be used as dependant variables in a regression. This can be done by interpolation with the pairs of wealths and points on the value function, available in the above table summarizing the computed quantities at $t = 1$. For example, a first point on the value function at $t = 1$, associated to decision $x = 0$ and path a, can be obtained by a linear interpolation computed with the quantities in the table below:

Wealth	Value
1.3	-0.0195
1.0100	?
0.8	-0.0886

With these, the linearly interpolated value is -0.0596. A sample of points on the value function at $t = 1$ can be generated in a similar fashion to obtain the linear system

$$\left[\begin{array}{c} -0.0596 \\ -0.0575 \\ -0.0547 \\ -0.0527 \\ -0.0596 \\ -0.0617 \\ -0.0644 \\ -0.0665 \end{array}\right] = \left[\begin{array}{ccc} 1.0 & 0.0 & 0.0^2 \\ 1.0 & 0.3 & 0.3^2 \\ 1.0 & 0.7 & 0.7^2 \\ 1.0 & 1.0 & 1.0^2 \\ 1.0 & 0.0 & 0.0^2 \\ 1.0 & 0.3 & 0.3^2 \\ 1.0 & 0.7 & 0.7^2 \\ 1.0 & 1.0 & 1.0^2 \end{array}\right] \left[\begin{array}{c} \beta_1 \\ \beta_2 \\ \beta_3 \end{array}\right] + \left[\begin{array}{c} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \end{array}\right],$$

and the optimal portfolio allocation as the solution of :

$$\widehat{x}_0 = \arg \max_{0 \le x \le 1} \; -0.0596 + 0.0001x - 0.0001x^2$$

which yields $\widehat{x}_0 = 0.5$, the optimal portfolio at $t = 0$. This completes the dynamic program.

Alternate specification of the algorithm: "realized values". The algorithm illustrated above is straightforward; we later refer to it as the "regression surface value" specification (see Section 3.4).

However, the regression surface values $\widehat{v}_{k,1}$ from Table 1 used above in the interpolations are not the unique choices that could be made for such calculations. At $t = 1$, alternate values, which we call "realized

values", can be computed using the optimal portfolio weights $\widehat{x}_{k,1}$. These values also represent a maximum expected utility value, but with the additional information provided by the realized paths at $t = 2$. For path a and wealth at $t = 1$ $w_{1,1} = 1.3$, such a "realized value" is computed with

$$\begin{aligned}\overline{v}_{1,1}^{(a)} &= u\left(w_{1,1} \cdot \left(\widehat{x}_{1,1} r_2^{(a)} + R_f\right)\right) \\ &= u\left(1.3 \times (0.5 \times 0.02 + 1.01)\right) = -\exp\left(-3 \times 1.326\right) = -0.0187.\end{aligned}$$

while it is

$$\overline{v}_{1,1}^{(b)} = u\left(1.3 \times (0.5 \times (-0.02) + 1.01)\right) = -\exp\left(-3 \times 1.3\right) = -0.0202$$

for sample path b (and still $w_{1,1} = 1.3$).

The information that would be available at $t = 1$ could then be summarized as:

Table 2: Values associated to two wealths at $t = 1$

$w_{k,1}$	$\widehat{x}_{k,1}$	$\widehat{v}_{k,1}$	$W\left(\widehat{x}_{k,1}, r_2^{(j)}\right)$	$\overline{v}_{k,1}^{(j)}$
1.3	0.5	−0.0195	$W\left(\widehat{x}_{1,1}, r_2^{(a)}\right) = 1.326$	$\overline{v}_{1,1}^{(a)} = -0.0187$
			$W\left(\widehat{x}_{1,1}, r_2^{(b)}\right) = 1.300$	$\overline{v}_{1,1}^{(b)} = -0.0202$
0.8	0.5	−0.0886	$W\left(\widehat{x}_{2,1}, r_2^{(a)}\right) = 0.816$	$\overline{v}_{2,1}^{(a)} = -0.0865$
			$W\left(\widehat{x}_{2,1}, r_2^{(b)}\right) = 0.800$	$\overline{v}_{2,1}^{(b)} = -0.0907$

Then, setting up to solve the problem at $t = 0$, the required interpolations could be done with the values for $W\left(\widehat{x}_{k,1}, r_2^{(j)}\right)$ and $\overline{v}_{k,1}^{(j)}$ found in the above table. For example, for the first component of the vector of generated wealth (wealth = 1.01) used in the previous interpolation case, the linearly interpolated value would have been computed with

Wealth	Value
1.3	−0.0187
1.0100	?
0.81	−0.0865

and would have yielded a value of −0.0580.

However, for the fifth component of the vector of generated wealths, also of value 1.01 but associated to path b this time, the "realized value" approach would have given

Wealth	Value
1.3	−0.0202
1.0100	?
0.81	−0.0967

and would have yielded a value of −0.0611.

These two alternate specifications and their consequences will be discussed in more length in the next section.

Finally, it should be noticed that the above example does not include state variables. For problems with state variables, the regression should include the powers of these variables, since the optimal portfolio depends on their values on a given sample path. In this case, one optimization per sample path must be performed. We provide in appendix an example similar to the one above, but which includes a state variable in the regressions.

3.3 Complete algorithm

This section provides the complete algorithm, as applied to the portfolio choice problem. We refer to it as LSMC–S&D to emphasize that it is a Least-Squares Monte Carlo method with regression on State variables and Decision variables. As already outlined in the above subsection, two sets of operations are involved in the algorithm. A first set of operations performs the preliminary work which mainly consists of simulations. A second set of operations does the backward recursive work.

Preliminary work: Simulations and grid formation.

1. Compute and store a grid of $N \times 1$ portfolio weight vectors $\mathbf{x}_i$ for $i = 1$ to n_x with elements satisfying the bounds $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$.
2. Simulate and store n_r paths of $N \times 1$ excess return vectors $\left\{\mathbf{r}_t^{(j)}\right\}_{j=1}^{n_r}$ for $t = 1$ to T, and $M \times 1$ state variable vectors $\left\{\mathbf{s}_t^{(j)}\right\}_{j=1}^{n_r}$ for $t = 0$ to $T-1$. The number of state variables (M) may be different from the number of risky assets (N), but both are simulated n_r times. Note that the returns $\mathbf{r}_{t+1}^{(j)}$ associated to path j will in most cases depend on the information set at time t contained in vector $\mathbf{s}_t^{(j)}$.
   The state variable vectors contain the known information at a given time point with which an optimal decision can be taken. This set of variables is problem dependent and will vary with the time series process associated with the returns. For example, with a predictable return process, the state variable vector might contain up-to-time t returns, or an exogenous mean-reverting market price of risk which determines the return levels for the next period.
3. Compute and store a grid of representative future wealth values $\{w_{k,t}\}_{k=1}^{n_w(t)}$ for $t = 0$ to $T-1$. To do so, the weights grid and simulated returns can be used.
   The number of points in the grid is time varying to allow a uniform coverage of the range of possible wealth; it grows from a unique wealth at $t = 0$ to $n_w(T-1)$ grid points at $t = T-1$.
4. Choose the set of n_b bases for the regression surfaces. For example, it could be all linear and quadratic monomials of the decision variables and of the exogenous state variables, plus the cross-products of second order betweeen all variables.

Recursive work: Backward recursive computations.

Begin **Time loop, from** $t = T-1$ **to** 0 :

Begin **Wealth loop, from** $k = 1$ **to** $n_w(t)$:

1. **Generate a sample of** $n_x \times n_r$ **wealths** at $t+1$ using the simulated returns and the discretized portfolio weights :

$$W_{k,t+1}\left(\mathbf{x}_i, \mathbf{r}_{t+1}^{(j)}\right) := w_{k,t} \times \left(\mathbf{x}_i^\top \mathbf{r}_{t+1}^{(j)} + R_f\right) \quad \text{for } i = 1 \text{ to } n_x \text{ and } j = 1 \text{ to } n_r. \tag{4}$$

2. **Generate a sample of** $n_x \times n_r$ **values** $\mathbf{v}_{k,t+1}(\mathbf{x}_i, \mathbf{s}_t^{(j)})$.
   For all but the last time period, the value $v_{k,t+1}\left(\mathbf{x}_i, \mathbf{s}_t^{(j)}\right)$ is computed by interpolation through a set of $n_w(t+1)$ pairs

$$\left\{\overline{v}_{k,t+1}^{(j)},\ w_{k,t+1}\right\}_{k=1}^{n_w(t+1)}$$

   where each $\overline{v}_{k,t+1}^{(j)}$ is a previously computed point estimate of the value function, as discussed in step 6. The interpolation is done between the first components of the pairs at the values of the second component $W_{k,t+1}\left(\mathbf{x}_i, \mathbf{r}_{t+1}^{(j)}\right)$ defined in equation (4).
   For the single case of $t = T-1$, the estimates $v_{k,t+1}\left(\mathbf{x}_i, \mathbf{s}_t^{(j)}\right)$ are computed from the utility function applied to the wealth obtained from equation (4) i.e. $v_{k,T}\left(\mathbf{x}_i, \mathbf{s}_{T-1}^{(j)}\right) = u\left(W_{k,T}\left(\mathbf{x}_i, \mathbf{r}_T^{(j)}\right)\right)$.

3. **Construct $n_x \times n_r$ basis vectors $\mathbf{B}_{i,j}$**, each of length n_b. Each vector $\mathbf{B}_{i,j}$ is associated to a pair $\left(\mathbf{x}_i, \mathbf{s}_t^{(j)}\right)$ for $i = 1$ to n_x and $j = 1$ to n_r.
4. **Regress the dependent values $\mathbf{v}_{k,t+1}\left(\mathbf{x}_i, \mathbf{s}_t^{(j)}\right)$ on the independent basis vectors $\mathbf{B}_{i,j}$.** There are $n_x \times n_r$ such pairs. Note that we regress a value at $t+1$ on vectors of positions $\mathbf{x}_i$ that apply from t to $t+1$, and on information sets available at t, i.e. $\mathbf{s}_t^{(j)}$. The resulting regression surface is denoted as $L_{k,t}(\mathbf{x}, \mathbf{s})$ and is specified by the $n_b \times 1$ vector $\boldsymbol{\beta}_{k,t}$ of regression coefficients (one coefficient per basis). This regression surface is an approximate representation of the expected utility when taking position $\mathbf{x}$ at time t, conditional on the information available at that time.
5. **Optimize the regression surface $L_{k,t}(\mathbf{x}, \mathbf{s})$ with respect to the positions $\mathbf{x}$** for each of the $j = 1, \ldots, n_r$ information sets $\mathbf{s}_t^{(j)}$:

$$\widehat{\mathbf{x}}_{k,t}^{(j)} = \arg \max_{\mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{x}_{\max}} L\left(\mathbf{x}; \mathbf{s}_t^{(j)}, \boldsymbol{\beta}_{k,t}\right) \text{ for } j = 1 \text{ to } n_r.$$

   which gives n_r estimates of optimal portfolio weight vectors $\widehat{\mathbf{x}}_{k,t}^{(j)}$.
6. **Compute the n_r values $\overline{\mathbf{v}}_{k,t}^{(j)}$** associated with optimal positions $\widehat{\mathbf{x}}_{k,t}^{(j)}$ by:
   - Computing the realized wealth $W_{k,t+1}\left(\widehat{\mathbf{x}}_{k,t}^{(j)}, \mathbf{r}_{t+1}^{(j)}\right)$ for $j = 1$ to n_r.
   - Interpolating through the $\left\{\overline{v}_{k,t+1}^{(j)},\ w_{k,t+1}\right\}_{k=1}^{n_w(t+1)}$ pairs. This step is very similar to the precedent interpolation step, but on a much smaller scale (n_r interpolations instead of $n_x \times n_r$.

End **Wealth loop.**

End **Time loop.**

At the completion of the algorithm, the value at wealth level $W_{0,0}$ and time $t = 0$ can be approximated as the average over j of the values $\overline{v}_{0,0}^{(j)}$ assuming of course that all paths were started with the same state variable value. An optimal policy is available and defined by the set of regression parameters $\boldsymbol{\beta}_{k,t}$: they describe a surface which must simply be optimized over the decision variables. Notice that for the first period, the policy is unique as the initial wealth is. This policy is typically what one is interested in: after the first period, the investor would re-optimize, so that in practice, the policies associated to later periods hold little interest. If one nevertheless wishes to find those later policies, given the (stored) regression parameters an optimal policy can be computed at each of two neighbor wealth grid levels; then an approximate optimal policy at intermediate wealth levels can be derived by interpolation.

3.4 Implementation details

A few implementation details are worth mentioning, which motivate the approach or improve its performance.

Interpolation and inverse utility transformation. As noted in Garlappi and Skoulakis (2010), approximating an almost linear function is much easier than doing so for a nonlinear function. We introduce a very simple "inverse utility" transformation at the two interpolation steps above: simply apply the inverse utility function to the $\overline{v}_{k,t+1}^{(j)}$ values before interpolating, interpolate (linearly), and apply back the utility function to the solution of the interpolation. This small procedure is quite effective at reducing the number of wealth points that are required for a specified precision. Unless specifically noted, the inverse utility transformation was always used.

Realized values vs. regression surface values. In the last step of the algorithm, it is important to note that $\overline{v}_{k,t}^{(j)}$ is not the value $L\left(\widehat{\mathbf{x}}_{k,t}^{(j)}; \mathbf{s}_t^{(j)}, \boldsymbol{\beta}_{k,t}\right)$ that is an output of the maximization problem. Indeed, once the optimal solution is found, there are (at least) two ways that a value can be associated to that solution (and path). One can use the technique described in the algorithm, giving what we call "realized values". One could also simply use the objective value at the optimal solution, $L\left(\widehat{\mathbf{x}}_{k,t}^{(j)}; \mathbf{s}_t^{(j)}, \boldsymbol{\beta}_{k,t}\right)$, as the new values of $\overline{\mathbf{v}}_{k,t}^{(j)}$.

We call these "regression surface values". This is the same alternative discussed by Longstaff and Schwartz (2001) in the context of option pricing; there the authors argue in favor of "realized values", in opposition to Tsitsiklis and Van Roy (1999) who used "regression surface values". It is important to mention that this use of "future" information is not anticipative: the optimal choice does not rely on this information, only the value associated to the optimal choice, *after* it has been made.

We tested both approaches and found very little difference between the answers, and certainly no systematic bias, for our portfolio choice problems.

Regression procedure. The regression step must be coded with some care, but it is easy to do so. Essentially, the building of the matrix of the so-called "normal equation" must be performed as a sum of rank-one matrices (the number of such matrices being $n_x \times n_r$), and this operation is best done with a compiled language (we used "C"). It is also worth mentioning that it is possible to ensure that most of the regression work is done only once per time step. We differ from Nijman, Werker, Koijen (2007) on this topic of regression. Indeed, the authors do a two-step regression: one regression in the space of the state variables and then one in the space of the decision variables. They find the resulting "approximate regression surface" more computationaly attractive. In our experience, the optimizations, not the regressions, still claim the majority of the computing time, even with a comparatively large problem with six decision variables and dozens of bases. For smaller problems, the time spent on regressions is small even in absolute time, of the ordre of seconds, tens of seconds at the most.

Finally, when simulating the paths, note that it is practical to ensure that the state variable before $t = 0$, S_0, does not have the same value for all paths even if it would be coherent with a real-life situation; otherwise, the matrix composed of all the basis vector $B_{i,j}$ has (at least) two constant columns, and the regression will fail because of singularity.

Optimization. Note that a large number of optimizations must be performed: one per simulated path, per discrete wealth level, and time level. For this reason, for larger problems, we use linear and quadratic terms only for the bases in the decision variables. The resulting quadratic optimization problems can be solved very efficiently with dedicated software and warm starts. (Note that in Section 4 some problems in smaller dimensions were also tested with higher degree monomials). Bases in the state variables are also limited to quadratic, for the sake of computational simplicity. All cross-products of order two (also between decision variables and state variables) are included.

4 Numerical experiments

We present numerical results for a variety of problems. The first example is very simple but still displays some fundamental characteristics of our approach; we also use it to illustrate a potential improvement direction for the method. The following examples have various features to highlight the behaviour of the algorithm. Example 2 moves away from the one-risky-asset case and example 3 shows the effect of returns that depend on state variables. Example 4 is quite general: multiple periods, multiple assets, depending on multiple sources of uncertainty, with a CARA utility function. Finally, example 5 shows the impact of using a nondifferentiable utility function, comparing the derivative-based approaches of Brandt, Goyal, Santa-Clara and Stroud (2005) and Garlappi and Skoulakis (2010), hereafter "BGSS05" and "GS10", with our algorithm. In some cases, the so-called "certainty equivalent" values are provided: these are simply the result of applying the inverse utility function to the problem's expected utility (as obtained under an algorithm or another).

Our LSMC–S&D algorithm and the benchmark quadrature code were programmed mostly in Matlab, with C procedures helping out at a few critical places. For most tests, a desktop personal computer was sufficient. For the larger problems, we used a server with 12 cpu's to run the optimizations in parallel. To give an idea of the computing times, a problem of six assets (five risky assets), with VAR returns (see example 4), an horizon of four periods, 10000 simulations, a basis of order 2 in the state variables and the decision variables (including all cross-products, 66 terms in total), and a decision variable grid at 20% intervals, is solved in less than six minutes on a desktop computer with four cpu's. With this number of simulations, the

solution is stable only to more or less one or two percent on each asset, but the objective function is already very stable.

4.1 Example 1: Two assets, one period, independent returns

We examine first an example taken from Brandt et al. (2005). It is a one-period example with one risky asset and one risk-free asset to assess the behavior of the method in a simple setting. The stock excess return is distributed as a shifted log normal distribution with constant mean μ and standard deviation σ. We use a constant relative risk aversion (CRRA) utility function, defined as:

$$u\left(W_T\right) = \frac{W_T^{1-\gamma}}{1-\gamma} \tag{CRRA}$$

where γ is a risk aversion coefficient. The weights of the risky and risk-free assets are constrained to be non-negative.

As it is well known from the portfolio literature, there is no available analytical solution for this case, and the optimal portfolio is independent from the wealth level, which is set here to $W_0 = 1$. For this example, the wealth does not need to be discretized since it is a one period problem. The preliminary work can be described as follows:

- Simulate the returns in excess of the risk-free rate with

$$r_T^{(j)} = R_f \times \left(e^{\mu+\sigma\varepsilon_{j,T}} - 1\right)$$

  where $\varepsilon_{j,T}$ are standard normal random variates.[3]
- Set the uniform grid of portfolio weights as $x_i = x_1 + (i-1)\delta_x$ for $i = 1$ to n_x with $x_1 = 0$, $x_{n_x} = 1$ and $\delta_x = \left(x_{n_x} - x_1\right) / \left(n_x - 1\right)$.
- Define the basis vector as $\begin{bmatrix}1 & x_i & x_i^2\end{bmatrix}^\top$. Because the optimal portfolio does not depend on the return and thus nor on the index j, the basis simply consist of a constant and powers of the discretized portfolio weights for the risky asset. A basis with the portfolio weights raised at the fourth power is also examined i.e. $\begin{bmatrix}1 & x_i & x_i^2 & x_i^3 & x_i^4\end{bmatrix}^\top$.

There is no recursive work because this is a one period problem. The wealths at time T are computed with $W_{0,T}\left(x_i, r_T^{(j)}\right) = w_0 \times \left(x_i r_T^{(j)} + R_f\right)$ for $i = 1$ to n_x and $j = 1$ to n_r, while the sample of value function points is computed with $v_{0,T}\left(x_i, r_T^{(j)}\right) = u\left(W_{0,T}\left(x_i, r_T^{(j)}\right)\right)$ for $i = 1$ to n_x and $j = 1$ to n_r. The regression coefficients are estimated by solving the normal equation while the optimization is performed using a golden section search algorithm.

Table 3 presents the optimal weights computed for two scenarios of values for the mean and standard deviation of the excess return. The first case roughly correspond to typical values that could be found with monthly returns with a risk-free rate equal to 0.05/12, while the second is for annual values with a risk-free rate of 0.05. The weights have been obtained with three methods. The first method is a numerical quadrature approach which is used here as a benchmark. The second method is the BGSS05 simulation and regression approach. The third method is the LSMC-S&D approach proposed in this paper, for different grids of portfolio weights and bases (powers up to two and powers up to four). For the second and third methods, ten optimizations have been performed, each with different simulations of $n_r = 5,000$ sample paths. The table presents the average of these 10 optimizations, while the standard deviation of these ten weights are reported below in parenthesis. As shown by these standard deviations, the results are fairly stable and do not vary much.

As already noticed in their paper, Brandt et al. (2005) find that their method is more precise for small pairs of μ and σ than with larger values for these parameters. This result is also obtained here and the weights

[3] In order to reduce the sample variability, a moment matching simulation is used to generate the standard normal random variates. See, for examples, Barraquand (1995) or Boyle et al. (1997).

Table 3: Example 1, portfolio weights

	$\mu = 0.01$, $\sigma = 0.05$				$\mu = 0.10$, $\sigma = 0.15$			
	$\gamma = 5$	$\gamma = 10$	$\gamma = 15$	$\gamma = 20$	$\gamma = 5$	$\gamma = 10$	$\gamma = 15$	$\gamma = 20$
				Benchmark (quadrature)				
	0.9000	0.4499	0.2999	0.2248	0.9890	0.4944	0.3288	0.2462
				LSMC–S&D, one-step, $\delta_x = 0.10$				
$B2$	0.8974	0.4507	0.3026	0.2390	0.9363	0.4984	0.3390	0.3130
	(0.001)	(0.001)	(0.001)	(0.001)	(0.003)	(0.004)	(0.001)	(0.006)
$B4$	0.9015	0.4506	0.3001	0.2241	0.9909	0.4964	0.3157	0.1532
	(0.002)	(0.001)	(0.001)	(0.000)	(0.005)	(0.002)	(0.009)	(0.014)
				LSMC–S&D, two-step, $\delta_x = 0.10$				
$B2$	0.9013	0.4483	0.3009	0.2258	0.9911	0.4698	0.3329	0.2540
	(0.002)	(0.001)	(0.001)	(0.001)	(0.005)	(0.019)	(0.002)	(0.002)
$B4$	0.9015	0.4506	0.3003	0.2251	0.9946	0.4949	0.3306	0.2464
	(0.002)	(0.001)	(0.001)	(0.000)	(0.005)	(0.002)	(0.002)	(0.001)
				BGSS05				
	0.8966	0.4489	0.2993	0.2245	0.6799	0.3574	0.2425	0.1835
	(0.001)	(0.001)	(0.000)	(0.000)	(0.007)	(0.003)	(0.002)	(0.002)

Benchmark (quadrature) are the weights computed with an adaptative Simpson quadrature algorithm. *LSMC–S&D, one-step* and *LSMC–S&D, two-step* are the average of 10 weights computed with the LSMC–S&D approach with one and two steps, with each weight computed using 5,000 sample paths. Number in parenthesis are the standard errors of the 10 computed weights. δ_x is the increment used to build a grid of portfolio weights used as basis in the regressions. $B2$ are weights computed with basis up to power two; $B4$ are weights computed with basis up to power four. Common random numbers are used for the LSMC–S&D approaches. *BGSS05* are the average of 10 weights computed with the Brandt et al. (2005) least-square Monte-Carlo simulation approach, with each weight computed using 5,000 sample paths.

computed with this approach are slightly off from the benchmark values for $\mu = 0.1$ and $\sigma = 0.15$, especially for larger values of the risk aversion coefficient. From results not reported here for the sake of brievety, a first interesting result to notice about the LSMC-S&D approach is that a crude discretization with $\delta_x = 0.2$ work equally well as finer discretizations with $\delta_x = 0.1$ or 0.05. Hence, it is possible to have a reasonable precision without imposing a large computational burden. As expected, this approach implemented with basis up to power four performs better than the quadratic case with basis up to power two. However, the differences are not large and the quadratic case offers a reasonable performance. Finally, for the annual case, for a risk aversion coefficient of 20, the LSCM-S&D and BGSS05 approaches deteriorate and have difficulties to pin point the solution.

A detailed examination of the expected utility function shows that these difficulties are mainly caused by the large curvature of the function. For $\gamma = 20$, the function shows little curvature for $x = 0$ to 0.5, but a large curvature for $x = 0.5$ to 1.0. It is possible to capture with more precision these varying degrees of curvature by computing two separate regression functions, one for $x = 0$ to 0.5, and another one for $x = 0.5$ to 1.0. This does not increase the total number of observations with which the regressions are computed but rather breaks the problem into two regressions with half of the original sample in each. Two sets of optimal weights must then be computed, and the one obtaining the largest function is the optimal one. The third panel of Table 3 shows the results obtained with this procedure. For each case, the optimum can be obtained with a good precision, with basis up to two and four.

4.2 Example 2: Four assets, one period, independent returns

This example is taken from Garlappi and Skoulakis (2010). This is again a one period example, but with three risky assets. The three risky assets have the same distribution as the example above, with the following values for the mean vector and variance-covariance matrix:

$$\boldsymbol{\mu} = [0.0530 \quad 0.0620 \quad 0.0570]^\top,$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 0.0263 & 0.0219 & 0.0183 \\ 0.0219 & 0.0324 & 0.0282 \\ 0.0183 & 0.0282 & 0.0714 \end{bmatrix}.$$

The utility function is of the CRRA family. Neither short sales nor borrowing is allowed. Again for this example, there is no closed form solution and a quadrature procedure can be used to compute benchmark weights. The preliminary work can be described as follows:

- Simulate and store the returns in excess of the risk-free rate with a multivariate version of the process described in the previous subsection.
- Compute and store the set of discretized portfolio weight $\{\mathbf{x}_j\}_{j=1}^{n_x}$. The cartesian product of the set $\{0, .1, \ldots, 1.0\}$ with itself is used to build the set of discrete portfolio weights, with the elements summing to a value greater than one removed from the cartesian product set.
- Define the basis vectors as a multivariate version of those introduced in the previous subsection (monomials up to power 2 and up to power 4). Unlike the previous case, however, cross-product of the weights are introduced in the basis vectors. We choose to include the cross-products of order two only, i.e. of the type $x_i x_j$. For basis up to power two, this result in a basis vector of dimension 10

$$\begin{bmatrix} 1 & x_1 & x_2 & x_3 & x_1x_2 & x_1x_3 & x_2x_3 & x_1^2 & x_2^2 & x_3^2 \end{bmatrix}.$$

  For bases up to power four, the dimension is 16.

As in the previous example, there is no recursive work, and we set $W_0 = 1$. The optimization of the least square functions are performed using a quadratic programming approach with constraints for basis up to two. For basis up to four, a nonlinear programming algorithm with constraints is used.

Table 4 shows quadrature benchmark weights taken from Garlappi and Skoulakis (2010) and the weights computed with our algorithm. The results obtained with this example are qualitatively similar to those obtained with the one asset case. The method performs well for small values of the risk aversion coefficient, but has difficulties to identify the optimum for larger values such as $\gamma = 10$ and 15. To improve the precision of the solutions, a two-step approach can be implemented for this case by simply partitioning the weights into two sets. A first set contains all the points of the decision grid for which the weight of the risk-free asset is smaller than 0.5, the second set contains the other points; there are therefore two regression surfaces, optimized independently, and after optimization, only the best solution of the two is kept. The intuition of this partitioning is that, given the no short selling constraint, very large risk aversion coefficients such as $\gamma = 10$ and 15 gives rise to small weights for risky assets. Partitioning the weights this way is thus able to reduce the relevant domain of the weights for the risky assets. As show in the table, this two-step approach is able to obtain weights very close to those of the benchmark quadrature approach.

4.3 Example 3: Two assets, multiple periods, and mean-reverting returns

This example is inspired from Wachter (2002), and was also examined by Detemple, Garcia and Rindisbacher (2005). It involves a single risky stock and a risk-free asset, over many periods. However, the risky stock price evolves according to a geometric Brownian motion with a mean reverting price of risk. There is a single source of uncertainty entering both the stock price and the mean reverting price of risk. In such a case, the optimal portfolio depends on the current value of the price of risk. The utility is CRRA.

A quasi-analytical solution for the optimal portfolio weight is developed in Wachter (2002). It is important to mention that the problem treated by Wachter and Detemple et al. differs in two ways from the problem we consider: they reallocate continuously, and they put no bounds on the portfolios weights (no limits to shortselling and borrowing). In contrast, we allocate discretely, and we need constraints on the weights, even if they need not be tight ones. We therefore compute (quadrature) benchmark results for the problem we solve with our LSMC–S&D method.[4]

[4] We did check that with a fine discretization of time and wide bounds on weights, our solution converges to the quasi-analytic solution of Wachter.

Table 4: Example 2, portfolio weights

	$\gamma = 2$			$\gamma = 5$			$\gamma = 10$			$\gamma = 15$		
						Benchmark (quadrature)						
	0.2896	0.4749	0.2355	0.2391	0.2282	0.1070	0.1194	0.1134	0.0530	0.0795	0.0754	0.0352
						LSMC–S&D, one-step						
B2	0.2914	0.4698	0.2388	0.2376	0.2072	0.1326	0.1024	0.0064	0.2258	0.0150	0.0000	0.2835
	(0.001)	(0.001)	(0.000)	(0.003)	(0.002)	(0.001)	(0.019)	(0.009)	(0.010)	(0.030)	(0.000)	(0.007)
B4	0.3084	0.4544	0.2372	0.2290	0.2369	0.1059	0.2016	0.0000	0.1387	0.0270	0.0359	0.4620
	(0.000)	(0.000)	(0.000)	(0.004)	(0.008)	(0.003)	(0.022)	(0.000)	(0.015)	(0.081)	(0.108)	(0.185)
						LSMC–S&D, two-step						
B2	0.3067	0.4564	0.2368	0.2301	0.2213	0.1089	0.1215	0.1081	0.0618	0.0810	0.0487	0.0707
	(0.016)	(0.014)	(0.002)	(0.001)	(0.001)	(0.000)	(0.001)	(0.001)	(0.001)	(0.002)	(0.002)	(0.001)
B4	0.2956	0.4705	0.2339	0.2346	0.2291	0.1086	0.1175	0.1179	0.0530	0.0772	0.0753	0.0436
	(0.010)	(0.005)	(0.005)	(0.005)	(0.010)	(0.001)	(0.001)	(0.000)	(0.000)	(0.001)	(0.002)	(0.001)

Benchmark (quadrature) are the weights taken from Garlappi and Skoulakis (2010) and computed with a quadrature approach. *LSMC–S&D, one-step* and *two-step* are the average of 10 weights computed with our one-step and two-step LSMC–S&D approach, with each weight computed using 5,000 sample paths. Number in parenthesis are the standard errors of the 10 computed weights. $B2$ are weights computed with basis up to power two; $B4$ are weights computed with basis up to power four.

The stock price dynamics is given by

$$\frac{dP_t}{P_t} = \mu_t dt + \sigma dY_t$$

where μ_t is a time varying expected return, σ is the constant stock return volatility, and Y_t is a Brownian motion. The dynamics for the price of risk, which is a state variable in this system, is defined as

$$s_t = \frac{\mu_t - r_f}{\sigma}$$

with

$$ds_t = -\lambda\left(s_t - \overline{s}\right)dt - \sigma_s dY_t$$

where $r_{f,t}$ is the continuously compounded risk-free rate, λ is the speed of mean reversion, $\overline{s}$ is the long run mean, and σ_s is the volatility parameter. Since both process share the same Brownian motion, the stock price is perfectly correlated with the price of risk. For this problem, the preliminary work is as follows:

- Using the gross risk-free rate defined as $R_f = e^{r_f \Delta t}$ where Δt is the length of a discrete time interval, the excess stock returns are obtained by first simulating the log of stock prices and prices of risk at the discrete time points $t = \Delta t, 2\Delta t, \ldots, T\Delta t$. These are simulated, using a moment matching procedure, with the exact solution of the above stochastic differential equation system given in Appendix C of Wachter (2002). The excess stock returns can then be computed as $r_{j,t+\Delta t} = P_{j,t+\Delta t}/P_{j,t} - R_f$.
- Compute the discretized portfolio weights with $x_i = x_1 + (i-1)\delta_x$ for $i = 1$ to n_x with $x_1 = -0.5$, $x_{n_x} = 1$ and $\delta_x = \left(x_{n_x} - x_1\right)/\left(n_x - 1\right)$.
- Define the basis vector as $\begin{bmatrix}1 & x_i & x_i^2 & s_{j,t} & s_{j,t}^2 & x_i s_{j,t}\end{bmatrix}^\top$. We use here the price of risk to form the basis entering the regressions since, at each time point, the optimal portfolio depends on the value of this variable.
- Using the simulated returns and starting at a wealth level of $W_0 = 1$, build a grid of representative future wealths at each time point by simulating the wealth that would be achieved for a portfolio fully invested in the risky asset at each time pont. This will produce conservative estimates of the maximum and minimum wealth values attainable with the above system. Using these simulated values, a each t, a grid is built with

$$w_{i,t} = \text{percentile}\left(\{\widetilde{w}_{j,t}\}_{j=1}^{n_r},\ p_i\right) \text{ for } i = 1 \text{ to } n_w$$

where $\widetilde{w}_{j,t}$ denotes a simulated wealth of a portfolio fully invested in the risky asset, and p_i is a probability level computed as $p_i = p_1 + (i-1)\delta_p$ for $i = 1$ to n_w with $p_1 = p_{\min}$, $p_{n_w} = 1 - p_{\min}$ and $\delta_p = (p_{n_w} - p_1)\,/\,(n_w - 1)$.

The recursive work can then be done as indicated in Section 3, with the interpolations performed using regression surfaces values. Table 5 presents the results obtained with the parameter values taken from Detemple et al. (2005) where this model is also examined. The weights computed with a Gauss Hermite quadrature are reported and used as benchmarks. Our method is casted in discrete time and uses 20 time-step per year in these computations. As shown in the table, the averages of the ten computed portfolio weights computed with the LSMC–S&D approach are close to the benchmark proportions, with some discrepancies for the five year cases. The standard errors indicate the variability in the computed proportions, that were obtained with 10,000 sample paths and a grid of wealth with 5 points. The table also reports the weights obtained with the BGSS05 approach and taken from Detemple et al. (2005).

Table 5 also shows the computed prices for the LSMC-S&D approach proposed here with the same grid for the wealth (5 points), but without the inverse utility transformation outlined in the algorithm. The precision clearly deteriorates for most cases. Results not reported here shows that, without this transformation, a precision similar to the one obtained with the transformation can be achieved with grids of 100 points. The inverse utility transformation can thus save a large amount of work that would otherwise be required to achieve precise solutions.

Table 5: Example 3, portfolio weights

	$\gamma = 2$	$\gamma = 3$	$\gamma = 4$	$\gamma = 5$
		$T = 20$		
Benchmark (quadrature)	0.2271	0.1397	0.1005	0.0813
LSMC–S&D with InvUtil	0.2116	0.1308	0.1020	0.0790
	(0.025)	(0.019)	(0.013)	(0.009)
LSMC–S&D without InvUtil	0.0670	0.0569	0.0548	0.0540
	(0.006)	(0.004)	(0.003)	(0.004)
BGSS05	0.2406	0.1502	0.1090	0.0855
		$T = 40$		
Benchmark (quadrature)	0.1936	0.1127	0.0703	0.0428
LSMC–S&D with InvUtil	0.1634	0.0908	0.0653	0.0490
	(0.023)	(0.015)	(0.010)	(0.014)
LSMC–S&D without InvUtil	0.0575	0.0529	0.0577	0.0548
	(0.006)	(0.004)	(0.003)	(0.003)
BGSS05	0.1862	0.1040	0.0708	0.0533
		$T = 100$		
Benchmark (quadrature)	0.1513	0.0529	-0.0022	-0.0391
LSMC–S&D with InvUtil	0.0868	0.0056	-0.0163	-0.0190
	(0.024)	(0.019)	(0.016)	(0.010)
LSMC–S&D without InvUtil	0.0305	0.0312	0.0306	0.0313
	(0.007)	(0.006)	(0.007)	(0.004)
BGSS05	0.0931	0.0299	0.0116	0.0430

Benchmark (quadrature) are the weights computed with a Gauss-Hermite quadrature integration; *LSMC–S&D with InvUtil* and *LSMC–S&D without InvUtil* are the average of 10 weights computed with, respectively, our LSMC–S&D algorithm with and without inverse utility transformation, with each weight computed using 10,000 sample paths, 20 time step per year, $x_{\min} = -0.5$, $x_{\max} = 1.0$, $n_x = 16$ ($\delta_x = 0.1$). Number in parenthesis are the standard errors of the 10 computed weights. Parameter values: $r_f = 0.06$, $\sigma = 0.2$, $\lambda = 0.2712$, $\overline{s} = 0.9456$, $s_0 = 0.1$, $\sigma_s = 0.2268$. *BGSS05* are the weights computed by the least-square Monte-Carlo approach with Taylor series of Brandt et al. (2005), and taken from Table 1 of Detemple et al. (2005).

4.4 Example 4: Multiple assets and periods, autoregressive returns

The last two examples are of our making. There are multiple assets and multiple allocation periods. The returns are predictable: they follow a vector autoregressive, lag-1 process, so that all returns of a period depend on all the returns of the previous period. This process increases the difficulty of the problem since the past returns form a state variable which conditions the optimization problem. The utility function is from the constant absolute risk aversion (CARA) class defined as

$$u(W) = -\exp(-\gamma W) \tag{CARA}$$

In this framework, unlike the CRRA case, the optimal portfolios are wealth dependant. Hence, simplifications often made possible in the CRRA case are not available in this setting. We conduct our experiments with a no-short-sale constraint, but similar behaviour is observed with other bounds.[5]

The VAR(1) model for the dynamics of the excess return for security i in a N risky security portfolio is given by:

$$\begin{bmatrix} R_{1,t+1} \\ R_{2,t+1} \\ \vdots \\ R_{N,t+1} \end{bmatrix} = \begin{bmatrix} a_{0,1} \\ a_{0,2} \\ \vdots \\ a_{0,N} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \begin{bmatrix} R_{1,t} \\ R_{2,t} \\ \vdots \\ R_{N,t} \end{bmatrix} + \begin{bmatrix} \sigma_1 z_{1,t+1} \\ \sigma_2 z_{2,t+1} \\ \vdots \\ \sigma_N z_{N,t+1} \end{bmatrix}$$

where $R_{i,t}$ are the log excess returns (i.e. excess returns under continuous compounding). The parameters $a_{i,t}$ are constant, the σ_i are standard deviation parameters, and $z_{i,t}$ for $i = 1$ to N are zero-mean unit-variance correlated random noises. Written in matrix form, the system is:

$$\mathbf{R}_{t+1} = \mathbf{A}_0 + \mathbf{A}_1 \mathbf{R}_t + \boldsymbol{\sigma}_z \mathbf{z}_{t+1}$$

where $\boldsymbol{\sigma}_\mathbf{z}$ is a $n \times n$ diagonal matrix of standard error parameters It is assumed that the z's are normaly distributed. Hence the continuously compounded excess returns are normaly distributed. This simplifies the benchmark computation (with Gauss-Hermite quadrature).

For this problem, the preliminary work is as follows:

- Using the above process, the log excess returns are simulated with a moment matching procedure for the mean and variance of the error terms The excess stock returns are then computed as $r_{i,t} = e^{R_{i,t}} - 1$.
- Compute and store the set of discretized portfolio weights $\{\mathbf{x}_j\}_{j=1}^{n_x}$ including the no-short-sale constraint. In this example, the cartesian product of the set $\{0, 0.1, \ldots, 1.0\}$ with itself is used to build the set of discrete portfolio weights, with the elements summing to a value greater than one removed from the cartesian product set.
- Form the basis vector with the following: a constant, the weigths for each risky asset up to power two, the lagged returns of the risky asset up to power two, and all the possible combinations of products of order two between the asset weight and the lagged returns. With two risky assets for example, we have:

$$\left[\, 1 \;\; x_1 \;\; x_2 \;\; R_{1,t} \;\; R_{2,t} \;\; x_1^2 \;\; x_1 x_2 \;\; x_1 R_{1,t} \;\; x_1 R_{2,t} \;\; x_2^2 \;\; x_2 R_{1,t} \;\; x_2 R_{2,t} \;\; R_{1,t}^2 \;\; R_{1,t} R_{2,t} \;\; R_{2,t}^2 \,\right]$$

- Build grids of wealths for each time points as follows. Simulate a set of n_r "low wealths" (recall n_r is the number of simulated return paths) by investing (anticipatively) solely in the stock with the worst return, at each period and along each of n_r paths of returns. For each period, consider the quantile q of the n_r "low wealths" to be the bottom of the wealth grid for that period. Compute "high wealths", and the top of the wealth grids, along the same line, using the $1-q$ quantile this time. Finally, compute a "wealth grid step" by splitting the grid at time $t = 1$ in $n_w(1)$ steps. Then all further wealth grids are divided with the same constant "wealth grid step" (of course, the number of grid points increases with t, since the range between the top and bottom increases).

[5] In fact, for wider decision spaces, if extreme leverage is allowed for example, our algorithm is simply slower.

The recursive work can then be done as indicated in Section 3 (We report results with the "realized values" approach; results with the "regression surface" approach were very similar). The parameters required to implement the model are obtained by fitting the VAR(1) model by least squares with the monthly data series of book-to-market portfolios (quintiles) from Jan. 1990 to Dec. 2013, created with the CRSP database and obtained from Kenneth French's web site. The risk-free rate is the 30 days to maturity risk-free bond return from CRSP.

We report results with four and five assets (three and four risky assets), over ten time periods, see Tables 6 and 7. The benchmark results are obtained with a Gauss-Hermite quadrature approach with six quadrature nodes, six wealth points, and ten return points for each risky asset. Note that in the case of four risky assets, with the state variable "wealth", this is a five-dimensional quadrature with a computation time of more than a day. To comment on the results, the weights computed with our algorithm and with quadrature are reasonably close in the four-asset case, and somewhat further apart in the five-asset case. In a all cases and without surprise, the standard deviation decreases as the number of simulation increases; the difference between our solution and the benchmark is almost always less than one standard deviation. It should be noted that the difficulty of finding precise weights increases with the number of assets, for the simple reason that newly introduced assets have a risk-return profile that is similar to one or more of assets, so that they become rather interchangeable.

Table 6: Example 4, portfolio weights, four-assets case

	Benchmark (quadrature)			
	0.2873	0.1484	0.1097	0.4545
	LSMC–S&D			
$n_r = 1000$	0.3129	0.1869	0.1316	0.3686
	(0.063)	*(0.135)*	*(0.177)*	*(0.138)*
$n_r = 5000$	0.3044	0.1144	0.1702	0.4110
	(0.032)	*(0.069)*	*(0.080)*	*(0.054)*
$n_r = 10000$	0.3021	0.1371	0.1413	0.4196
	(0.026)	*(0.047)*	*(0.074)*	*(0.037)*
$n_r = 20000$	0.3070	0.1686	0.1102	0.4142
	(0.018)	*(0.023)*	*(0.035)*	*(0.030)*

Numbers in the second column are the weights of the risk-free security; numbers in columns 3 to 5 are the weights of the risky securities. *Benchmark (quadrature)* are the weights computed with a Gauss-Hermite quadrature; *LSMC–S&D* are the average of 8 weights computed with our LSMC–S&D algorithm. Parameters values are $T = 10$, $n_w = 16$, $q = 0.0001$ $x_{\min} = 0.0$, $x_{\max} = 1.0$, $r_f = 1.0028$ and $\gamma = 4$. Number in parenthesis are the standard errors of the eight computed weights.

Table 7: Example 4, portfolio weights, five-assets case

	Benchmark (quadrature)				
	0.1759	0.1616	0.2309	0.2340	0.1976
	LSMC–S&D				
$n_r = 1000$	0.1834	0.1539	0.2698	0.2833	0.1096
	(0.061)	(0.091)	(0.291)	(0.234)	(0.107)
$n_r = 5000$	0.1395	0.1924	0.3672	0.1800	0.1209
	(0.040)	(0.121)	(0.178)	(0.159)	(0.077)
$n_r = 10000$	0.1634	0.1224	0.4160	0.1670	0.1312
	(0.024)	(0.041)	(0.142)	(0.117)	(0.041)
$n_r = 20000$	0.1615	0.1190	0.4093	0.1403	0.1700
	(0.013)	(0.040)	(0.137)	(0.082)	(0.038)

Numbers in the second column are the weights of the risk-free security; numbers in columns 3 to 6 are the weights of the risky securities. *Benchmark (quadrature)* are the weights computed with a Gauss-Hermite quadrature; *LSMC–S&D* are the average of 8 weights computed with the least-square Monte-Carlo simulation approach proposed here. Parameter values are $T = 10$, $n_w = 16$, $q = 0.0001$ $x_{\min} = 0.0$, $x_{\max} = 1.0$, $r_f = 1.0028$ and $\gamma = 4$. Number in parenthesis are the standard errors of the eight computed weights.

4.5 Example 5: Conditional Value-at-Risk, a case of nondifferentiable utility

In this last example, we consider a problem of a somewhat different nature, in that the investor's risk preferences are expressed by the Conditional Value-at-Risk (CVaR) of his portfolio, as opposed to a more usual utility function. The investor wants to maximize expected terminal wealth while minimizing CVaR, a setting that is similar to Markowitz's mean-variance portfolio optimization. Conditional Value-at-Risk (CVaR) has gained significant popularity since its introduction as a coherent measure of risk in Artzner et al. (1999); using a result of Uryasev and Rockafellar (2000), we rewrite the original wealth-CVaR problem as a piecewise linear utility maximization problem. Our numerical experiments illustrate the difficulty that a nondifferentiable utility function can bring to methods that rely on Taylor expansions, such as that proposed by Brandt et al. (2005) and Garlappi and Skoulakis (2010).

Specifically, we are interested in solving a portfolio choice problem that balances the two objectives of minimizing risk as represented by CVaR, and minimizing expected lost wealth.[6] The balance between those two objectives is determined by a fixed risk aversion parameter ρ. The problem is :

$$\text{minimize}_{\{x_t\}_{t=0}^{T-1}} \quad \rho\,\text{CVaR}_\beta[W_0 - W_T] + (1-\rho)\,E[W_0 - W_T]\,, \tag{5}$$

with the following wealth evolution constraints for all t linking W_t and x_t

$$W_{t+1} = W_t\left(\mathbf{x}_t^\top \mathbf{r}_{t+1} + R_f\right) \tag{6}$$

where $W_0 - W_T$ captures the lost wealth. CVaR is defined as the expected value beyond a quantile

$$\text{CVaR}_\beta(Z) = E_0[Z|Z \geq Z^\beta]$$

with Z^β being the $(1-\beta)$-quantile of random variable Z, i.e. the smallest amount for which $P(Z \leq Z^\beta) \geq \beta$ (Z^β is also known as the Value-at-Risk of Z.) Rockafellar and Uryasev (2000) established the non-trivial result that Conditional Value-at-Risk can be computed as the optimal value of the problem:

$$\text{CVaR}_\beta[W_0 - W_T] = \min_s \; -s + \frac{1}{1-\beta}E\left[\max(\,0\,;\,W_0 - W_T + s\,)\right]\,. \tag{7}$$

Introducing equation (7) in equation (5), the original portfolio choice is equivalent to

$$\text{minimize}_{\{x_t\}_{t=0}^{T-1}} \quad \min_s \;\; \rho\left(-s + \frac{1}{1-\beta}E\left[\max(\,0\,;\,W_0 - W_T + s\,)\right]\right) + (1-\rho)E[W_0 - W_T]\,,$$

under the wealth evolution constraint (6). It is a well known fact in optimization that this last problem can also be solved as

$$\text{minimize}_s \quad \min_{\{x_t\}_{t=0}^{T-1}} \quad \rho\left(-s + \frac{1}{1-\beta}E\Big[\max(\,0\,;\,W_0 - W_T + s\,)\Big]\right) + (1-\rho)E\big[W_0 - W_T\big]\,,$$

under the wealth evolution constraint (6). This last formulation is also equivalent to

$$\text{minimize}_s \quad \max_{\{x_t\}_{t=0}^{T-1}} \quad E\big[u_s(W_T)\big] \tag{8}$$

under the wealth evolution constraint

$$W_{t+1} = W_t\left(\mathbf{x}_t^\top \mathbf{r}_{t+1} + R_f\right)$$

and for the utility function

$$u_s(y) \triangleq \min_{\{x_t\}_{t=0}^{T-1}} \left(\,\rho s + (1-\rho)(y - W_0)\,;\,\rho s + \frac{\rho}{1-\beta}(y - W_0 - s) + (1-\rho)(y - W_0)\right)\,.$$

[6] Our model is presented in terms of lost wealth for simplicity of exposure given that Conditional Value-at-Risk is typically defined in terms of a random variable we wish would be as small as possible. Minimizing lost wealth is equivalent to maximizing excess wealth.

This can be seen from the fact that minimizing a function is equivalent to maximizing the negative of the function, and

$$\begin{aligned} & \sup_{\{x_t\}_{t=0}^{T-1}} \ \rho\left(s + \frac{1}{1-\beta} E\left[\min(\,0\,;\, W_T - W_0 - s\,)\right]\right) + (1-\rho)E[W_T - W_0] \\ = & \sup_{\{x_t\}_{t=0}^{T-1}} \ E\left[\min\left(\,\rho s + (1-\rho)(W_T - W_0)\,;\, \rho s + \frac{\rho}{1-\beta}(W_T - W_0 - s) + (1-\rho)(W_T - W_0)\right)\right]. \end{aligned}$$

It is interesting to note that $u_s(y)$ describes a piecewise linear concave function with a slope of $1-\rho$ for values above $W_0 + s$ and a slope of $\frac{\rho}{1-\beta} + 1 - \rho$ for values below $W_0 + s$.

The final formulation (8) of our original wealth-CVaR problem is a one-dimensional minimization problem whose objective is itself a portfolio choice problem with a nondifferentiable utility function. Solving iteratively this one-dimensional minimization means that at each iteration we need to solve a problem of the type that was solved in the previous examples.

We revisit the example of Subsection 4.2 (four assets, one period) to compare the portfolios obtained with our LSMC–S&D approach, as well as the Brandt et al. (2005) and Garlappi and Skoulakis (2010) approaches. While Table 8 compares the portfolio weights obtained using each scheme, Figure 1 presents the performance in terms of expected return and 90% conditional value at risk for portfolios that are obtained for different levels of risk aversion ρ between 0 and 1. One can observe from this figure that while employing LSMC–S&D allows to identify portfolios that make different types of compromise between mean excess wealth and its conditional value at risk, the other two schemes are only able to identify the most risk averse and the least risk averse portfolios. This is due to the fact that the dynamic program that needs to be solve here implicates a utility function that is piecewise linear with a non-differentiable point. Indeed, both of these methods are misled by their use of the Taylor expansion of the value function, which, when it exists and for any choice of order since all derivatives beyond the first one is null, simply reduces to a linear approximation. Our method circumvent this issue by regressing on the global structure of the value function instead of a Taylor expansion which relies on local information. Let us mention that the objective of this experiment was to illustrate how the two schemes based on the Taylor series expansion of the value function might not be flexible enough in some cases. While there obviously exists ways to smoothen a piecewise linear utility function before applying these schemes, no prior work has described formally how this should be done. In fact, in the context of a multi-period problem it might be necessary to smoothen the value function at each period. Overall, in developing our experiment, we opted for an implementation of those two schemes that was closest to the authors' original proposal rather than improvising corrections.

Table 8: Example 5, Optimal portfolio weights under Mean return vs. CVaR tradeoff – three risky assets, one period

	$\rho = 0$			$\rho = 0.25$			$\rho = 1$		
Sample Average Approx.	0.0	0.0	1.0	0.3473	0.4404	0.2124	0.0	0.0	0.0
LSMC–S&D	0.0	0.0	1.0	0.3735	0.4363	0.1901	0.0	0.0	0.0
BGSS05	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
GS10	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0

Sample Average Approx. are the weights obtained when solving the problem exactly with the 100 sample paths. The simulation-and-regression approaches also employed 100 sample paths for simulation, a set of 5000 vectors of portfolio weight uniformly distributed over the probability simplex for regression on a basis vector that included all monomials of degree 2 or less. The two Taylor series based schemes can only exploit first order information while our schemes exploits monomials up to power four. Finally, the minimization with respect to s in problem (8) was performed using Matlab's "fminbnd", with one of the three above mentionned approach used to evaluate the inner maximization.

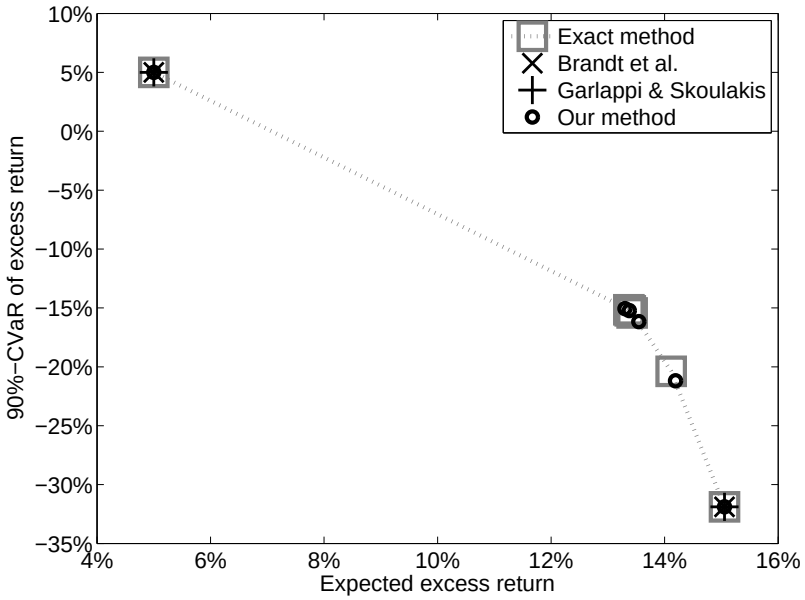

Figure 1: Example 5, Efficient set frontier of performances of portfolios obtained form solving the mean-CVaR model with different risk aversion levels when using four different resolution schemes for the inner dynamic programming step. Note that the "x" and "+" markers overlap in two locations that emphasize the fact that these methods are incapable of identifying portfolio that trade-off between expected excess wealth and its conditional value at risk through the procedure that is described

This experimental set-up also allows us to evaluate what is the added value of performing regression on a basis vector that involves monomials with degree higher than 2. In this example, since it involves three risky assets, a basis vector with monomials of a maximum degree of K should involve all the terms $x_1^{d_1} x_2^{d_2} x_3^{d_3}$ such that $d_1 + d_2 + d_3 \leq K$. Table 9 presents the average sub-optimality of the portfolios that are returned by a solution scheme that employs our simulation-and-regression approach as the degree of the largest monomial involves is increased. Note that the sub-optimality that is reported is an average over the sub-optimality achieved for a set of 11 different risk aversion levels between 0 and 1. While there are no guarantees in general that employing a basis with monomials of higher order will necessarily improve the quality of the solution that is obtained, on average we can observe that it is the case.

Table 9: Example 5, convergence of our approximation scheme for solving the mean-CVaR problem as regression is performed on monomials of higher order

Max degree	1	2	3	4	6	8
Solution's sub-optimality	0.186%	0.019%	0.008%	0.008%	0.006%	0.005%

5 Conclusion

We present a simulation-and-regression technique for dynamic programs that relies on regression with respect to both state variables and decision variables (LSMC–S&D). The tools that are involved are simple and well researched: Monte Carlo simulation, ordinary least-squares, and continuous optimization. The technique, however, is little known in the area of portfolio optimization, and in finance in general. To our knowledge, this paper is the first exploration of the usefulness of LSMC–S&D for a variety of portfolio choice problems. Implemented with care, the algorithm is very flexible, robust, and its performance degrades at a much slowler rate than a standard quadrature approach, when the number of dimension increases.

A Introductory example: Regression on a state variable

In this appendix, we extend the introductory example of Section 3.2 to the case where we regress on a state variable as well. To keep it compact and clear, the section is written with variables, not numbers.

Consider a two period example for a portfolio consisting of one risky and one risk-free asset with a constant gross risk-free rate R_f. The investor, endowed with current wealth W_0 at time $t = 0$, wants to maximize the utility of his wealth at time $t = 2$, with an optimal asset allocation at time $t = 0$ and $t = 1$, without any shortsales or borrowing.

Denote the grid of portfolio weights used here as

$$x = \left[\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \right]^\intercal,$$

while the grid of wealths is:

$t = 0$	$t = 1$
	$w_{1,1}$
w_0	
	$w_{2,1}$

.

In this example, the return process has an AR(1) structure. Hence, when forming a portfolio at $t = 0$, the known return at $t = 0$ is a state variable.[7] To keep the example simple, only two sample paths of returns are simulated at each period:

path	Period 0	Period 1	Period 2
a	$r_0^{(a)}$	$r_1^{(a)}$	$r_2^{(a)}$
b	$r_0^{(b)}$	$r_1^{(b)}$	$r_2^{(b)}$

In a second step, the recursive dynamic programming procedure is implemented with the above quantities. Starting at $t = 1$, one period before maturity, we perform the following operations, for each state of wealth at this time period. We first generate a sample of wealth at $t = 2$. For example, using the wealth level $w_{1,1}$, it is possible to generate a wealth at $t = 2$ using the first weight and path a of the simulated return with

$$W\left(x_1, r_2^{(a)}\right) = w_{1,1} \times \left(x_1 r_2^{(a)} + R_f\right).$$

Using all the possible combinations of simulated returns at $t = 2$ and portfolio weights on the grid, the end of period sample of wealths at $t = 2$ is:

$$\left[\begin{array}{cccccc} W\left(x_1, r_2^{(a)}\right) & \cdots & W\left(x_4, r_2^{(a)}\right) & W\left(x_1, r_2^{(b)}\right) & \cdots & W\left(x_4, r_2^{(b)}\right) \end{array} \right].$$

[7] The time 0 return is known at $t = 0$. This known return value is $r_0^{(a)}$ in the simulated sample of the table. The return at $t = 0$ for the second sample path appearing in the table has been simulated by starting the simulation at $t = -1$. As it will become clear later in the example, this is done in order to avoid singular matrices in the regressions at time $t = 0$.

Computing the utility of these end of period wealths with the utility function $u(W)$, the following linear regression system can be formed:

$$\begin{bmatrix} u\left(W\left(\cdot\right)\right) \\ u\left(W\left(\cdot\right)\right) \\ u\left(W\left(\cdot\right)\right) \\ u\left(W\left(\cdot\right)\right) \\ u\left(W\left(\cdot\right)\right) \\ u\left(W\left(\cdot\right)\right) \\ u\left(W\left(\cdot\right)\right) \\ u\left(W\left(\cdot\right)\right) \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & r_1^{(a)} & \left(r_1^{(a)}\right)^2 & x_1 r_1^{(a)} \\ 1 & x_2 & x_2^2 & r_1^{(a)} & \left(r_1^{(a)}\right)^2 & x_2 r_1^{(a)} \\ 1 & x_3 & x_3^2 & r_1^{(a)} & \left(r_1^{(a)}\right)^2 & x_3 r_1^{(a)} \\ 1 & x_4 & x_4^2 & r_1^{(a)} & \left(r_1^{(a)}\right)^2 & x_4 r_1^{(a)} \\ 1 & x_1 & x_1^2 & r_1^{(b)} & \left(r_1^{(b)}\right)^2 & x_1 r_1^{(b)} \\ 1 & x_2 & x_2^2 & r_1^{(b)} & \left(r_1^{(b)}\right)^2 & x_2 r_1^{(b)} \\ 1 & x_3 & x_3^2 & r_1^{(b)} & \left(r_1^{(b)}\right)^2 & x_3 r_1^{(b)} \\ 1 & x_4 & x_4^2 & r_1^{(b)} & \left(r_1^{(b)}\right)^2 & x_4 r_1^{(b)} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \end{bmatrix},$$

where the elements on the left end side are the utility of wealths at $t = 2$, and the lines of the independent variable matrix are formed with a constant, powers up to 2 of the portfolio weights, powers up to 2 of the returns at $t = 1$ (our state variable), and the product of the weights and the returns. In this system, ε_i are random errors with zero expected value, and β_1 to β_6 are unknown coefficients, whose value can be estimated with an ordinary least square regression. Using these estimated coefficients, the approximate expected utility, conditional on the first value of the state variable can be written as a continuous function of the portfolio weights, which can be used to find the optimal portfolio weight as:

$$\widehat{x}_{1,1}^{(a)} = \underset{0\leq x\leq 1}{\arg\max}\ \widehat{\beta}_1 + \widehat{\beta}_2 x + \widehat{\beta}_3 x^2 + \widehat{\beta}_4 r_1^{(a)} + \widehat{\beta}_5 \left(r_1^{(a)}\right)^2 + \widehat{\beta}_6 x r_1^{(a)}$$

which obtains a function value at the optimum of $\widehat{v}_{1,1}^{(a)}$ which correspond to the regression surface value. Using the optimal weight $\widehat{x}_{1,1}^{(a)}$ and the realized return $r_2^{(a)}$, it is possible to compute the realized value $\overline{v}_{1,1}^{(a)} = u\left(w_{1,1}\cdot\left(\widehat{x}_{1,1} r_2^{(a)} + R_f\right)\right)$.

Similarly, the approximate expected utility, conditional on the second value of our state variable can be written as a continuous function of the portfolio weights, which can be used to find the optimal portfolio weight as:

$$\widehat{x}_{1,1}^{(b)} = \underset{0\leq x\leq 1}{\arg\max}\ \widehat{\beta}_1 + \widehat{\beta}_2 x + \widehat{\beta}_3 x^2 + \widehat{\beta}_4 r_1^{(b)} + \widehat{\beta}_5 \left(r_1^{(b)}\right)^2 + \widehat{\beta}_6 x r_1^{(b)}$$

which obtains a function value at the optimum of $\widehat{v}_{1,1}^{(b)}$ which correspond to the regression surface value. It is possible to compute the realized value as $\overline{v}_{1,1}^{(b)} = u\left(w_{1,1}\cdot\left(\widehat{x}_{1,1} r_2^{(b)} + R_f\right)\right)$. Similar calculations for the other state of wealth leads to the table below which summarizes the computed quantities for each state of wealth:

$w_{k,1}$	$\widehat{x}_{k,1}^{(j)}$	$\widehat{v}_{k,1}^{(j)}$	$W\left(\widehat{x}_{k,1}, r_2^{(j)}\right)$	$\overline{v}_{k,1}^{(j)}$
$w_{1,1}$	$\widehat{x}_{1,1}^{(a)}$	$\widehat{v}_{1,1}^{(a)}$	$W\left(\widehat{x}_{1,1}, r_2^{(a)}\right)$	$\overline{v}_{1,1}^{(a)}$
	$\widehat{x}_{1,1}^{(b)}$	$\widehat{v}_{1,1}^{(b)}$	$W\left(\widehat{x}_{1,1}, r_2^{(a)}\right)$	$\overline{v}_{1,1}^{(b)}$
$w_{2,1}$	$\widehat{x}_{2,1}^{(a)}$	$\widehat{v}_{2,1}^{(a)}$	$W\left(\widehat{x}_{2,1}, r_2^{(a)}\right)$	$\overline{v}_{2,1}^{(a)}$
	$\widehat{x}_{2,1}^{(b)}$	$\widehat{v}_{2,1}^{(b)}$	$W\left(\widehat{x}_{2,1}, r_2^{(a)}\right)$	$\overline{v}_{2,1}^{(b)}$

Using the above values, the optimal weight at $t = 0$ can be computed using a similar procedure. We first compute a sample of wealth at $t = 1$ using all combinations of simulated returns and portfolio weights. This sample is

$$\left[\ W\left(x_1, r_1^{(a)}\right)\ \cdots\ W\left(x_4, r_1^{(a)}\right)\ W\left(x_1, r_1^{(b)}\right)\ \cdots\ W\left(x_4, r_1^{(b)}\right)\ \right].$$

Using these, we can generate a sample of points on the value function at $t = 1$ that will be used as dependant variables in a regression. This can be done by linear interpolation with the pairs of wealths and realized values (the quantities summarized in the above table). For example, the value corresponding to $W\left(x_1, r_1^{(a)}\right)$ is obtained by a linear interpolation computed with the quantities below:

wealth	value
$w_{1,1}$	$\overline{v}_{1,1}^{(a)}$
$W\left(x_1, r_1^{(a)}\right)$	$-$
$w_{2,1}$	$\overline{v}_{2,1}^{(a)}$

while a value corresponding to $W\left(x_4, r_1^{(b)}\right)$ would be computed with

wealth	value
$w_{1,1}$	$\overline{v}_{1,1}^{(b)}$
$W\left(x_4, r_1^{(b)}\right)$	$-$
$w_{2,1}$	$\overline{v}_{2,1}^{(b)}$

Denote these interpolated values by $v\left(x_1, r_1^{(1)}\right)$ and $v\left(x_4, r_1^{(b)}\right)$.With similar computations for the other values of the sample of simulated wealth at $t = 1$, we obtain the following linear system

$$\begin{bmatrix} v\left(x_1, r_1^{(a)}\right) \\ v\left(x_2, r_1^{(a)}\right) \\ v\left(x_3, r_1^{(a)}\right) \\ v\left(x_4, r_1^{(a)}\right) \\ v\left(x_1, r_1^{(b)}\right) \\ v\left(x_2, r_1^{(b)}\right) \\ v\left(x_3, r_1^{(b)}\right) \\ v\left(x_4, r_1^{(b)}\right) \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & r_0^{(a)} & \left(r_0^{(a)}\right)^2 & x_1 r_0^{(a)} \\ 1 & x_2 & x_2^2 & r_0^{(a)} & \left(r_0^{(a)}\right)^2 & x_2 r_0^{(a)} \\ 1 & x_3 & x_3^2 & r_0^{(a)} & \left(r_0^{(a)}\right)^2 & x_3 r_0^{(a)} \\ 1 & x_4 & x_4^2 & r_0^{(a)} & \left(r_0^{(a)}\right)^2 & x_4 r_0^{(a)} \\ 1 & x_1 & x_1^2 & r_0^{(b)} & \left(r_0^{(b)}\right)^2 & x_1 r_0^{(b)} \\ 1 & x_2 & x_2^2 & r_0^{(b)} & \left(r_0^{(b)}\right)^2 & x_2 r_0^{(b)} \\ 1 & x_3 & x_3^2 & r_0^{(b)} & \left(r_0^{(b)}\right)^2 & x_3 r_0^{(b)} \\ 1 & x_4 & x_4^2 & r_0^{(b)} & \left(r_0^{(b)}\right)^2 & x_4 r_0^{(b)} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \end{bmatrix},$$

and the optimal portfolio allocation as:

$$\widehat{x}_{1,0}^{(1)} = \underset{0 \le x \le 1}{\arg\max} \ \widehat{\beta}_1 + \widehat{\beta}_2 x + \widehat{\beta}_3 x^2 + \widehat{\beta}_4 r_0^{(a)} + \widehat{\beta}_5 \left(r_0^{(a)}\right)^2 + \widehat{\beta}_6 x r_0^{(a)}.$$

We note here that only one optimization is performed since only one value of the state variable (the known value) is relevant at this point. This value has been placed, by convention, in the first sample path when simulating the returns in the preliminary step.

References

[1] Artzner, P., Delbaen, F., Eber, J.-M., Heath, D., 1999, Coherent measures of risk, Mathematical Finance, 9, 203–228,

[2] Balduzzi, P., Lynch, A., 1999, Transaction costs and predictability: Some utility cost calculations, Journal of Financial Economics 52, 47–78.

[3] Barraquand, J., 1995, Numerical valuation of high dimensional multivariate european securities, Management Science 41, 1882–1891.

[4] Boyle, P., M. Broadie and P. Glasserman, 1997, Monte Carlo methods for security pricing, Journal of Economic Dynamics and Control 21, 1263–1321.

[5] Brandt, M., Goyal, A., Santa-Clara, P., Stroud, J., 2005, A simulation approach to dynamic portfolio choice with an application to learning about return predictability, Review of Financial Studies 18, 831–873.

[6] Brennan, M., Schwartz, E., Lagnado, R., 1997, Strategic asset allocation, Journal of Economic Dynamics and Control 21, 1377–1403.

[7] Dammon, R., Spatt, C., Zhang, H., 2000, Optimal consumption and investment with capital gains taxes, Review of Financial Studies 14, 583–616.

[8] Denault, M., Simonato, J.-G., Stentoft, L., 2013, A simulation-and-regression approach for stochastic dynamic programs with endogenous state variables, Computers and Operations Research 40, 2760–2769.

[9] Detemple, J. Garcia, R., Rindisbacher, M., 2003, A Monte Carlo method for optimal portfolios, Journal of Finance 58, 401–446.

[10] Detemple, J., Garcia, R., Rindisbacher, M., 2005, Intertemporal asset allocation: A comparison of methods, Journal of Banking and Finance 29, 2821–2848.

[11] Garlappi, L., Skoulakis, G., 2009, Numerical solutions to dynamic portfolio problems: the case for value function iteration using Taylor expansion, Computational Economics 33, 193–207.

[12] Garlappi, L., Skoulakis, G., 2010, Solving consumption and portfolio choice problems: The state variable decomposition method, Review of Financial Studies 23, 3346–3400.

[13] Koijen, R., Nijman, T., Werker, B., 2010, When can life cycle Investors benefit from time-varying bond risk premia ?, Review of Financial Studies 23, 741–780.

[14] Longstaff, F., Schwartz, E, 2001, Valuing American options by simulations: a simple least squares approach, Review of Financial Studies 14, 113–148.

[15] Nijman, T., Werker, B., Koijen, R., 2007, Appendix to: when can life-cycle investors benefit from time-varying bond risk premia?, Working paper, Network for Studies on Pensions, Aging and Retirement.

[16] Rockafellar, R. T., Uryasev, S., 2000, Optimization of conditional value-at-risk, Journal of Risk 2, 21–41.

[17] Tsitsiklis, J., Van Roy, B., 2001, Regression methods for pricing complex American-style options, IEEE Transactions on Neural Networks 12, 694–703.

[18] Wachter, J., 2002, Portfolio and consumption decisions under mean-reverting returns: An exact solution for complete markets, Journal of Financial and Quantitative Analysis 37, 63–91.