Convertible Bond Pricing

Alistair J. Chopping

Abstract

This report accompanies the code in the corresponding GitHub repository. We give an overview of the Longstaff-Schwartz algorithm for the pricing of American-style derivatives, before applying it to the pricing of a non-callable, non-puttable, zero coupon convertible bond. We show that our implementation produces a result in agreement with [4]. This project was inspired by [3].

Contents

1 Introduction

A convertible bond is a corporate bond that the investor has the right to convert into a predetermined number of shares of the underlying equity, if they so wish. Convertibles therefore provide both equity-like and bond-like features, giving an investor the limited downside of a bond with the theoretically unlimited upside of a stock.

For the issuer, convertible bonds can provide a cheap way of raising early-stage capital in that the product can be sold with a lower interest rate than a straight bond due its attractive (to the investor) conversion feature. In addition, if the investor of the bond converts then the issuer no longer has the responsibility of paying the investor a regular coupon, as would be the case for a straight bond. For the investor, a convertible bond provides limited downside risk in that if the equity does not perform as well as hoped, the investor can simply choose hold the bond without converting and the bond's value will not drop below the bond floor[1] (provided the equity does not perform *too* badly). Despite this, a convertible still exposes the investor to potential upside greater than that of a straight bond if the underlying does well. In this way, convertibles provide something of a "best of both worlds" for the investor, and as such are typically more expensive than a straight bond.

One factor affecting the likelihood of conversion is the *conversion price* of the bond, which is defined

$$\mathcal{P} := \frac{V_{\text{face}}}{N}, \tag{1.1}$$

with $\mathcal{P}$ the conversion price, V_{face} the face value of the bond and N the *conversion ratio*; the number of shares the investor receives upon conversion. If the share price $S > \mathcal{P} \iff NS > V_{\text{face}}$ (the convertible is "in-the-money"), then it is optimal for the investor to convert and receive NS rather than holding the bond and receiving the face value. In this case the price of the bond will be more sensitive to changes in the price of the underlying than to factors such as interest rates, which affect the fixed-income features of the bond. Conversely if the bond is out-of-the-money, $S < \mathcal{P}$, then the convertible behaves more like a bond - conversion is unlikely and the price of the bond is barely affected by changes in the underlying. If the share price is sufficiently low and the risk of default is high, then the bond is said to be *distressed* and its price will converge to the value of the shares one would hold on immediate conversion (known as the *parity*). If $S \approx \mathcal{P}$, then the price of the bond is sensitive to both interest rates *and* changes in the price of the underlying.

European-style convertible bonds, where the investor can convert only at the bond's maturity, are the theoretically simplest and the easiest to price, but are rarely found in the market. To price a European-style derivative, under risk-neutral pricing methodology one simply calculates the expected value of the cashflow at maturity, and discounts it to today. A far more realistic setup is that of an American-style convertible bond, whereby the user can convert at any point between the issue of the bond and the maturity date.

American-style derivatives present more of a challenge to price, since one cannot simply take the value of the derivative to be the discounted cashflow at maturity. Rather, at each point during the lifetime of the derivative one must compare the value of immediate exercise with the expected value of continuing to hold the derivative, with a view to potentially exercising at a later time. A number of methods have been proposed for the valuation of American-style derivatives, particularly in the context of American options. The Binomial Option Pricing Model (BOPM) for instance

[1] The bond floor is the fair price of the bond, calculated as if it were a straight bond with no conversion features.

is a tree-based model introduced by Cox, Ross and Rubinstein [1], which models the underlying as a binomial lattice. The method most pertinent to this report is the Longstaff-Schwartz (LS) algorithm [2], which uses a least-squares approach to estimate the conditional expected value of continuing to hold the derivative at each timestep. In this report, we use the algorithm to price a non-callable, non-puttable, zero-coupon convertible bond. We begin in chapter 3 with a review of the LS algorithm, before detailing our implementation & the numerical results in chapter 4.

This project was influenced by the review of LS American options pricing in [3], as well as by [5].

2 Preliminaries

2.1 Conditional Expectation & the Mean-Squared Error

Suppose we collect some data about a population; for example the height of some number of people N. Given this data $Y = (Y_1, ..., Y_N)$, a sensible "guess" of the weight of a random person should be a function of the data, $f(Y)$. The "best guess" turns out to be $f(Y) = \mathbb{E}[Y]$. By "best guess", we mean that $\mathbb{E}[Y]$ is the function that minimises the *mean-square-error*

$$\text{MSE} := \mathbb{E}[(Y - f(Y))^2], \tag{2.1}$$

namely,

$$\mathbb{E}[(Y - \mathbb{E}[Y])^2] \leq \mathbb{E}[(Y - f(Y))^2] \quad \forall f(Y) \neq \mathbb{E}[Y]. \tag{2.2}$$

Now suppose that we collect more data, the weights of the people, and store them in a random vector $X = (X_1, ..., X_N)$. Now, say we want to guess people's weight Y, given their height X. Our guess for the weight should be a function $f(X)$ of their height, and again we define the "best" guess to be the function that minimises the MSE; this time we have

$$\text{MSE} = \mathbb{E}[(Y - f(X))^2]. \tag{2.3}$$

It turns out that the best guess is $f(X) = \mathbb{E}[Y|X]$, namely

$$\mathbb{E}[(Y - \mathbb{E}[Y|X])^2] \leq \mathbb{E}[(Y - f(X))^2] \quad \forall f(X) \neq \mathbb{E}[Y|X]. \tag{2.4}$$

The proof of this statement is as follows.

$$\begin{aligned}
\mathbb{E}[(Y - f(X))^2] &= \mathbb{E}[(Y - f(X) + \mathbb{E}[Y|X] - \mathbb{E}[Y|X])^2] \\
&= \mathbb{E}[((Y - \mathbb{E}[Y|X]) - (f(X) - \mathbb{E}[Y|X]))^2] \\
&= \mathbb{E}\Big[(Y - \mathbb{E}[Y|X])^2 + (f(X) - \mathbb{E}[Y|X])^2 - 2(Y - \mathbb{E}[Y|X])(f(X) - \mathbb{E}[Y|X])\Big].
\end{aligned} \tag{2.5}$$

Focusing on the last term, we have

$$\mathbb{E}\Big[(Y - \mathbb{E}[Y|X])(f(X) - \mathbb{E}[Y|X])\Big] = \mathbb{E}\left[\mathbb{E}\Big[(Y - \mathbb{E}[Y|X])(f(X) - \mathbb{E}[Y|X])\Big] \,\middle|\, X\right], \tag{2.6}$$

by the law of iterated expectation. In the inner expectation, we can use the "taking out what is known" property of conditional expectation ($\mathbb{E}[g(X)Y|X] = g(X)\mathbb{E}[Y|X]$) to show that,

$$\begin{aligned}\mathbb{E}\Big[(Y-\mathbb{E}[Y|X])(f(X)-\mathbb{E}[Y|X])\Big] &= \mathbb{E}\left[\mathbb{E}\Big[(Y-\mathbb{E}[Y|X])(f(X)-\mathbb{E}[Y|X])\Big]\,\middle|\, X\right] \\ &= \mathbb{E}\left[\underbrace{(\mathbb{E}[Y|X]-\mathbb{E}[Y|X])}_{0}(f(X)-\mathbb{E}[Y|X])\right] \\ &= 0. \end{aligned} \tag{2.7}$$

Therefore, we see that

$$\begin{aligned}\mathbb{E}[(Y-f(X))^2] &= \mathbb{E}\Big[(Y-\mathbb{E}[Y|X])^2+(f(X)-\mathbb{E}[Y|X])^2\Big] \\ &= \mathbb{E}\Big[(Y-\mathbb{E}[Y|X])^2\Big]+\underbrace{\mathbb{E}\Big[(f(X)-\mathbb{E}[Y|X])^2\Big]}_{\geq 0} \\ &\geq \mathbb{E}\Big[(Y-\mathbb{E}[Y|X])^2\Big], \end{aligned} \tag{2.8}$$

where the bound is saturated iff $f(X) = \mathbb{E}(Y|X)$. Namely, $\mathbb{E}(Y|X)$ minimises the MSE. □

3 The Longstaff-Schwartz Algorithm

Throughout we technically consider the valuation of *Bermudan*-style derivatives, where the derivative can be exercised at a fixed and finite number of times up to and including the maturity date. Of course, a Bermudan derivative becomes an American derivative in the limit that the number of exercise opportunities approaches infinity, and we will consider the case that the derivative can be exercised at any of the time steps in the simulation. Thus, our Bermudan derivative converges to an American derivative as the number of time steps grows larger. We will remain as general as possible in detailing the Longstaff-Schwartz algorithm, without referring to a specific derivative. We begin by discretising the time interval $[0, T]$ into some chosen number of time steps, and simulating a large number M of underlying equity paths.

The Longstaff-Schwartz algorithm is a backward induction algorithm[2], in which at each time step the value of immediate exercise is compared to the value of continuing to hold the derivative. At any given time step t_n, we want to determine which paths are ITM, and which aren't. For paths that are ITM the investor will exercise the derivative if and only if exercise is more profitable than continuation. Namely, the investor exercises if exercise value > continuation value.

For paths that are ITM, but are not profitable (namely, those for which $S > \mathcal{P}$ but exercise value < continuation value) the investor will choose not to exercise and hold the derivative instead. Therefore, the value of the derivative at t_n in this instance is whatever it is at t_{n+1}, discounted back to t_n. Paths that are ATM/OTM are automatically not profitable to exercise, so the investor will also want to hold those. Again, for these paths the value of the derivative at t_n is whatever it is at the next time step t_{n+1}, discounted back to t_n.

[2]Namely, we begin at maturity $t = T$ and proceed backwards through each time step until $t = 0$.

For paths that are ITM, let's say that we (as the investor) do not exercise at time t_n. In this case, what's the value of the derivative at t_n? It's the discounted expectation value of the future value of the derivative at the next time step. Namely, the value of the derivative at t_n if we don't exercise is given by

$$V_{t_n} = e^{-r(t_{n+1}-t_n)} V_{t_{n+1}}. \tag{3.1}$$

Again for paths that are ITM, what if we do choose to exercise at time t_n? In this case, the value of the derivative is simply the immediate exercise value.

For paths that are OTM or ATM we assume that there is no chance that the investor will exercise because it is guaranteed that the continuation value > exercise value. Therefore, for these paths the value of the derivative at time t_n is again just the value of the derivative at t_{n+1} discounted back to t_n, (3.1).

We can implement the above in code simply by, at time t_n, setting the value of the derivative (3.1) for every path. Then, in the line below this, we set the value of the derivative to the exercise value only for the paths that are ITM.

However, we have not addressed how to actually calculate the continuation value; the value we use as a tool to determine whether or not the investor will exercise. This is not obvious - the investor doesn't know anything but the *current* price of the underlying, and so in particular they do not have a realisation of the (random variable representing the) discounted future cashflow. As such, we should not explicitly use future information to determine the continuation value, even though we have this information from our simulation of the underlying; we would not be calculating a fair value of the derivative if we simply used all of the realisations of the cash flow at t_{n+1} and took their mean to calculate the continuation value.

Rather than doing this, the Longstaff-Schwartz algorithm extracts an estimate (or "guess", using the terminology from chapter 2) of the *expected* discounted future cashflow at t_{n+1} given the current underlying price S_{t_n}, by fitting a Laguerre series to the pair $(S_{t_n}, e^{-r(t_{n+1}-t_n)}\mathcal{C}_{t_{n+1}})$, where $\mathcal{C}_{t_n}$ is the cash flow for the derivative at time t_n. This pair corresponds to the (X, Y) respectively in the explanation of conditional expectation values and least-squares minimisation in chapter 2. In this way, we are using the realisations of the future cashflow at t_{n+1} in order to minimise the mean-square error and ensure that our estimate of the continuation value is the "best guess", but we are only using *current* information about the underlying price at t_n to define the estimate itself.

How do we actually define the estimate of the continuation value? The continuation value is defined to be

$$\begin{aligned} F_{t_n} &:= \mathbb{E}[e^{-r(t_{n+1}-t_n)}\mathcal{C}_{t_{n+1}}|\mathcal{F}_{t_n}] \\ &= \mathbb{E}[e^{-r(t_{n+1}-t_n)}\mathcal{C}_{t_{n+1}}|S_{t_n}], \end{aligned} \tag{3.2}$$

where in the second equality we have assumed that the underlying follows a Markov process and as such the only relevant part of the history of the underlying is the price at the current time step. Note that we have discounted the cash flow in the definition of F_{t_n}.

Now, we assume that F_{t_n} can be represented as a linear combination of basis functions. Basis functions of what? A conditional expectation $\mathbb{E}[Y|X] \equiv Z$ is a random variable itself, and we want it to be square-integrable (so that we can talk about variances of Z, for instance). Namely, we want

that Z is an element of some L^2 space, and so the basis functions should form a basis of L^2. As a basis for L^2 one could choose Hermite polynomials, Chebyshev polynomials, Laguerre polynomials and a number of others. We will choose Laguerre polynomials, representing the continuation value as

$$F_{t_n} = \sum_{j=0}^{\infty} a_j L_j(S_{t_n}), \tag{3.3}$$

with S_{t_n} the underlying price at t_n and a_j some coefficients to be determined. Recall that when we wanted to compute $\mathbb{E}[Y|X]$, our "guess" was to be a function of X. So, when we want to compute $\mathbb{E}[e^{-r\Delta t}\mathcal{C}_{t_{n+1}}|S_{t_n}]$, our guess should be a function of S_{t_n}, as it is above. For implementation purposes we approximate the continuation value by a finite number of basis functions,

$$F_{t_n} \approx \sum_{j=0}^{\deg} a_j L_j(S_{t_n}), \tag{3.4}$$

with "deg" the maximum degree of the Laguerre polynomials we want to include in the approximation.

Recall that our guess is a "best guess" if it minimises the mean-square error, in our case given by

$$\mathbb{E}[(\underbrace{e^{-r(t_{n+1}-t_n)}\mathcal{C}_{t_{n+1}}}_{Y} - \sum_{j=0}^{\deg} a_j L_j(\underbrace{S_{t_n}}_{X}))^2], \tag{3.5}$$

where we have indicated the corresponding X and Y from our explanation in chatper 2. However, we proved in chapter 2 that the function that minimises the mean-square error is the conditional expectation $\mathbb{E}[Y|X]$. Therefore, in choosing the coefficients a_j such that the MSE is minimised, the sum of Laguerre polynomials does indeed approximate the conditional expectation - which by definition is the continuation value,

$$\sum_{j=0}^{\deg} a_j L_j(S_{t_n}) \approx \mathbb{E}[e^{-r\Delta t}\mathcal{C}_{t_{n+1}}|S_{t_n}] = F_{t_n}. \tag{3.6}$$

3.1 Definitions - LS for a Convertible Bond

The definition of convertible bond requires the specification of a number of parameters, which we will now detail.

Definition 3.1.1. *The **Face Value** V_{face} of a convertible bond is the amount that the investor will receive from the issuer upon maturity of the bond, if the investor has not converted the bond into shares.*

Definition 3.1.2. *The **Conversion Ratio** N is the number of shares the investor will receive upon conversion.*

Definition 3.1.3. *The **Conversion Price** $\mathcal{P}$ is the price-per-share at which the bond can be converted into shares. It is defined*

$$\mathcal{P} := \frac{V_{face}}{N}, \tag{3.7}$$

and the convertible is said to be in-the-money (ITM) at time t if

$$S_t > \mathcal{P}. \tag{3.8}$$

Definition 3.1.4. *The **Conversion Value** $V_{conv}(t)$ is the total value of the shares received by the investor upon conversion of the bond at time t,*

$$V_{conv}(t) := NS_t. \tag{3.9}$$

Definition 3.1.5. *Assuming that the investor has not yet converted, at the maturity date they can either choose to convert the bond into shares or receive the face value, and they will choose which they do according to which is more profitable. The **Cash Flow at maturity** $\mathcal{C}_T$ is the profitability of the bond at maturity, defined*

$$\mathcal{C}_T := max(V_{conv}(T), V_{face}). \tag{3.10}$$

Before maturity, the investor will convert only if the conversion value is greater than the value of the bond itself. The value of the bond itself is simply the discounted value of the bond at the next time step, and so the cash flow at $t_n < T$ is taken to be,

$$\mathcal{C}_{t_n} := max(V_{conv}(t_n), e^{-r(t_{n+1}-t_n)}V_{t_{n+1}}) \tag{3.11}$$

It is important to note that the continuation value and the value of the bond at the next time step are different. If at t_n the investor holds the bond instead of converting, they do not receive the continuation value, they simply keep the bond's value. The continuation value is simply a tool used to determine whether it is optimal to convert or hold at some given time step. Note that 3.11 is well-defined, since for our starting point we set the value of the bond at maturity to be $V_T = \mathcal{C}_T$.

4 Simulation

In this section we use the Longstaff-Schwartz algorithm to price a particular convertible bond. We begin with two definitions in order to clarify the instrument we are interested in pricing.

Definition 4.0.1. *A bond is called **callable** if the **issuer** has the right, but not the obligation, to buy the bond back from the investor at a predetermined price called the **call price**.*

Definition 4.0.2. *A bond is called **puttable** if the **investor** has the right, but not the obligation, to demand early repayment of the face value of the bond.*

In this report, we price a non-callable, non-puttable, zero coupon, American-style convertible bond.

4.1 Code Review & Numerical Results

We begin by simulating the underlying equity paths. Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space equipped with filtration $\mathbb{F} := \{\mathcal{F}_{t_n}\}_{n\in\mathcal{I}}$. We model the underlying equity process $\{S_t\}_{t\in[0,T]}$ as a Geometric Brownian Motion with respect to this probability space,

$$dS_t = (r-q)S_t dt + \sigma S_t dW_t, \tag{4.1}$$

where r is the risk-free rate, σ is the volatility and q is the dividend yield[3]. We discretise this SDE, resulting in

$$S_{t+\Delta t} = S_t + (r-q)S_t\Delta t + \sigma S_t \underbrace{(W^S_{t+\Delta t} - W^S_t)}_{\sim\mathcal{N}(0,\Delta t)}. \tag{4.3}$$

The code looks slightly different to this, with the stochastic term in multiplied by $\sqrt{\Delta t}$, since in the code we sample from the standard normal distribution rather than $\mathcal{N}(0,\Delta t)$. The function `GBM()` generates M underlying paths. We then define functions that compute the cashflow at maturity and the conversion value.

The function `LSMC()` implements the Least-Squares Monte Carlo method. We start by generating M underlying paths, and discretising the time frame. We compute the final price of the underlying for each path (stored in the array `ST`), define the conversion price and initialise the bond value to be the cash flow at maturity for each path.

We then have a `for` loop that starts at the maturity date T and progresses backwards through time until the first time step. Within this `for` loop, we define the discount factor - this is always the same for every pair of time steps (t_{n+1}, t_n), since we discretised the time interval uniformly. Next, we compute the price of each path at the current time step t_n, and store these prices in the array `curr_price`. We then produce a boolean index that is used to determine the current price of only the paths that are ITM at the current time step, since these are the only paths for which the probability of conversion is nonzero.

[3]The dividend yield is defined

$$q := \frac{\text{most recent full-year dividend}}{\text{current share price}} \tag{4.2}$$

.

We then fit a Laguerre series to the pair $(S_{t_n}, e^{-r(t_{n+1}-t_n)}\mathcal{C}_{t_{n+1}})$, with $\mathcal{C}_{t_{n+1}}$ the current value of the bond for the ITM paths, giving us an approximation to the continuation values for the ITM paths. `fit_laguerre` only produces the coefficients a_j of the series (3.4), and so we produce values of the series itself for each current price[4] using `contvalues`.

Towards the end of the `for` loop, we first define the immediate conversion value for all paths. Next, we define a "conversion index" `conv_idx`; a boolean index that indicates for which paths the investor will convert, and for which they will not. Finally, the value of the bond is set to the holding value (3.1) everywhere, before being set to the immediate conversion value for the paths for which the investor converts. The function returns the convertible bond's value for every path, stored in an array `CB`.

We then perform some plotting that we display in section 4.2, before taking the mean of `CB` to obtain a final price for the convertible bond.

4.2 Plots & Numerical Results

The final part of the code produces various plots to illustrate the main ideas of the algorithm, before finally computing the price of a particular convertible bond. The bond we are interested in pricing has a time to maturity $T = 2$ years, a face value of $V_{\text{face}} = \$100$, and a conversion ratio $N = 1$. We take the underlying to have volatility $\sigma = 0.4$, a dividend yield $q = 0.1$, an initial price at issue of $S_0 = \$100$, and we take the risk-free rate to be $r = 0.05$. In pricing the bond itself we use $M = 10,000$ underlying paths and $252 \times T$ time steps (taking the number of trading days in a year to be 252). In producing the plots however, we use fewer time steps and fewer underlying paths purely for visualisation purposes.

First, we plot a small number of underlying GBM paths by solving the discretised SDE (4.3). The plot is given in 1. Next, we are interested in knowing when the paths are ITM or OTM/ATM. To do this, we produce a colour-coded scatter plot where the points are coloured teal if the path is ITM at that particular time, or orange if otherwise. In figure 2 we show the effect of this for a single path to elucidate the idea, and also for all underlying paths. In figure 3 we show the fitting of the Laguerre series (3.4) to the (current price, discounted cashflow) pairs, for the final pair of time steps $(t_n = T, t_{n-1})$.

The final price calculated for the convertible bond with the above parameters, using $M = 10,000$ underlying paths, fluctuates[5] between approximately \$109.00 and 109.50. The value of $\sim$ \$109.13 calculated by [4] via tree-based methods falls within this range, and one could decrease the size of the range by increasing the number of simulated paths M, though of course the larger one makes M the more computing power is needed.

[4] We do not use only the ITM paths here, but *all* paths - otherwise we would not be able to obtain continuation values for the paths that are also ATM and OTM.

[5] Of course, we are dealing with random processes and so the price fluctuates each time the code is run.

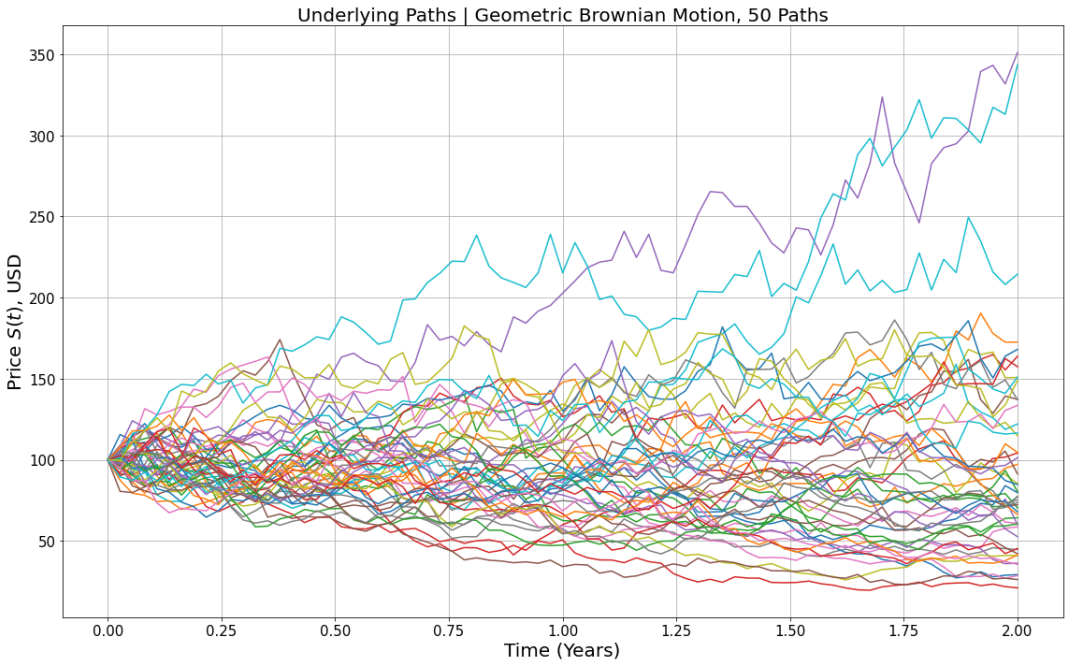

Figure 1: $M = 50$ realisations of the underlying price process. We choose the small number 50 purely for visualisation purposes; in computing the price of the convertible later we use $M = 10,000$.

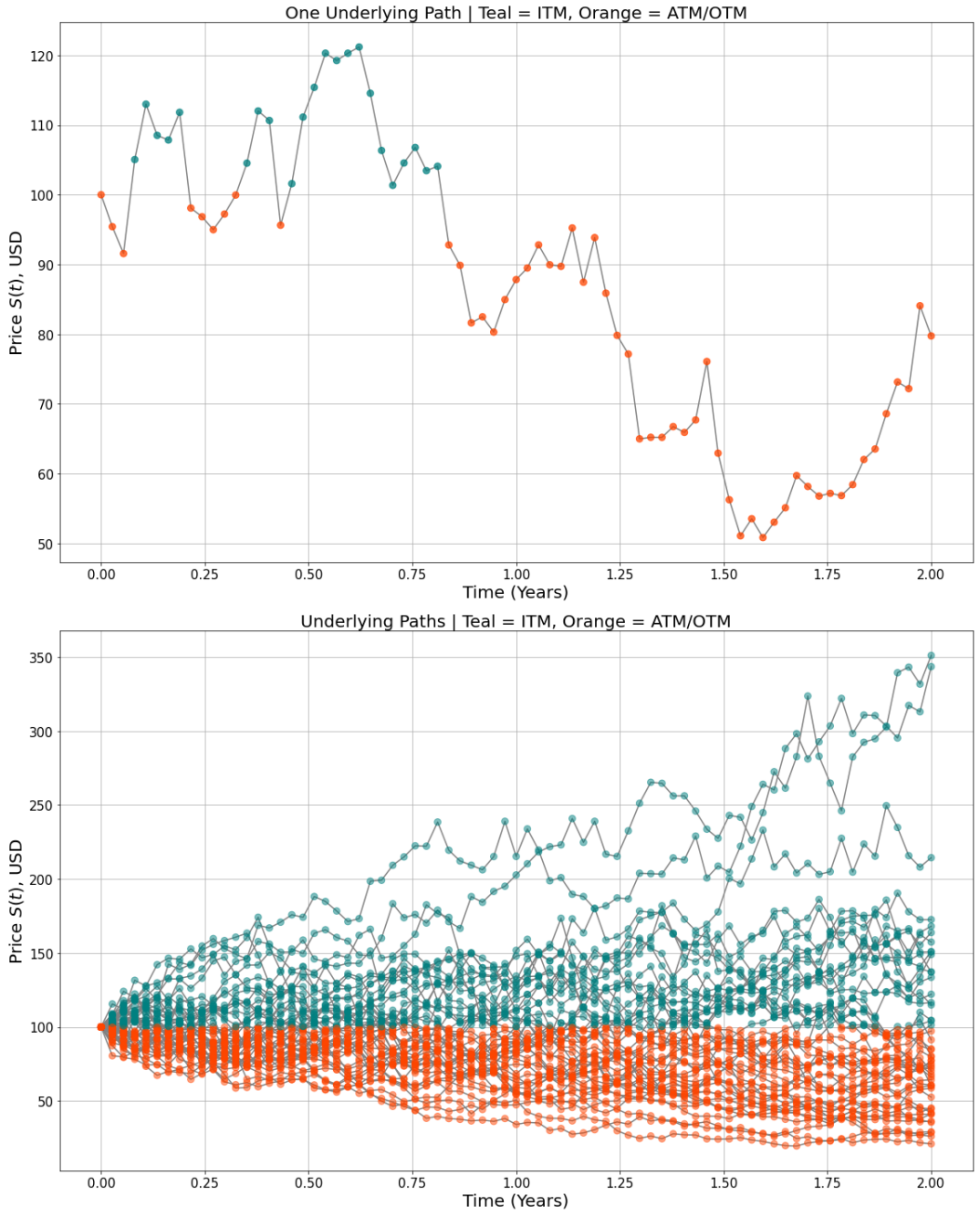

Figure 2: In the topmost plot, we show the effect of the colouring (teal=ITM, orange=otherwise) for a single price path. We see that once the path rises above the conversion price it becomes teal, and once it falls below the conversion price it becomes orange. In the bottom-most plot we show exactly the same procedure, but we plot all 50 paths.

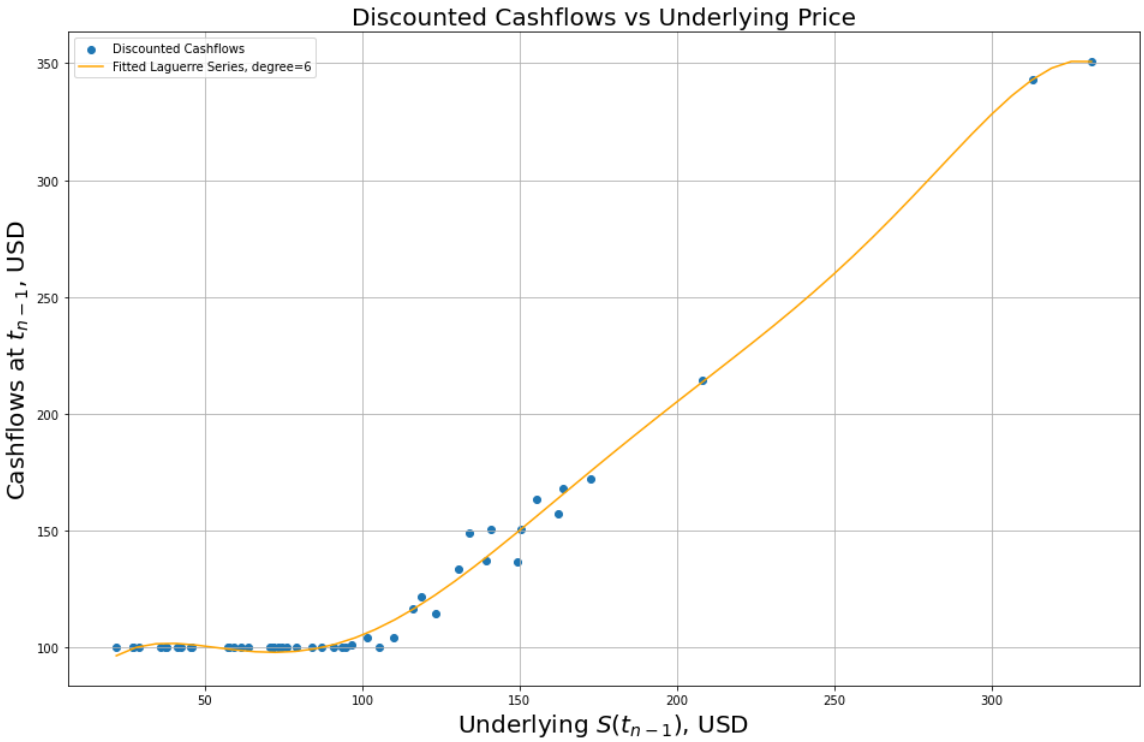

Figure 3: The scatterplot points in blue show how the discounted cashflow changes with the price of the underlying, and in orange we show the fitted Laguerre series.

5 Conclusion

In this report we detailed an implementation of the Longstaff-Schwartz algorithm applied to American convertible bond pricing. We began by proving an essential statistical ingredient of the algorithm, the fact that the conditional expectation minimises the mean-square error. We then gave an overview of the Longstaff-Schwartz algorithm for pricing American-style derivatives, before giving various definitions in order to elucidate the essential details associated with convertible bonds. We gave a walkthrough of the *Python* code in the GitHub repository, before showing various plots and giving our numerical result. We showed that our numerical result for the price of a particular non-callable, non-puttable, zero-coupon, American convertible bond was in agreement with [4]. An interesting area for further work could be to price more complex convertibles such as those with coupons, or those with call/put features.

References

[1] Cox, John C. and Ross, Stephen A. and Rubinstein, Mark, 1979. "Option pricing: A simplified approach," Journal of Financial Economics, Elsevier, vol. 7(3), pages 229-263, September.

[2] Longstaff, Francis A and Schwartz, Eduardo S, 2001. "Valuing American Options by Simulation: A Simple Least-Squares Approach," The Review of Financial Studies, Society for Financial Studies, vol. 14(1), pages 113-147.

[3] Santiago, Dialid, 2024. "American Options Pricing using the Longstaff-Schwartz Algorithm". Available at `https://quantgirluk.github.io/Understanding-Quantitative-Finance/AOS.html`.

[4] Loriato, Leandro Amato, 2014. "Convertible bond pricing: a Monte Carlo approach". Available at `https://api.semanticscholar.org/CorpusID:103138037`.

[5] Brinell, Gustav Markland, 2023. "Pricing Convertible Bonds Using Monte Carlo Simulations". Available at `https://umu.diva-portal.org/smash/get/diva2:1773590/FULLTEXT01.pdf`