Calibration of Derivative Pricing Models: a Multi-Agent Reinforcement Learning Perspective

Nelson Vadori
nelson.n.vadori@jpmorgan.com
J.P. Morgan AI Research
Paris, France

ABSTRACT
One of the most fundamental questions in quantitative finance is the existence of continuous-time diffusion models that fit market prices of a given set of options. Traditionally, one employs a mix of intuition, theoretical and empirical analysis to find models that achieve exact or approximate fits. Our contribution is to show how a suitable game theoretical formulation of this problem can help solve this question by leveraging existing developments in modern deep multi-agent reinforcement learning to search in the space of stochastic processes. Our experiments show that we are able to learn local volatility, as well as path-dependence required in the volatility process to minimize the price of a Bermudan option. Our algorithm can be seen as a particle method à la Guyon et Henry-Labordere where particles, instead of being designed to ensure 
\( \sigma \) volatility process to minimize the price of a Bermudan option. We consider continuous-time diffusion models of the form 
\[ dS_t = S_t \sigma_t dW_t, \]
where \( W \) is a Brownian motion, and we let the stochastic volatility process \( \sigma \) be very general for now, for example it could be path-dependent, function of extra sources of randomness, or rough [7, 10]. Such models are uniquely determined by \( \sigma \), so we will call \( \sigma \) a model, and denote \( \mathbb{E}[X] \) the price of option \( X \) under model \( \sigma \). As it is usual to do so, we consider, without loss of generality, zero interest rate and dividends, so that \( S \) coincides with the forward price process.

One of the most fundamental questions arising in quantitative finance is the following:

**Problem 1.** Given a set of options \( O \) with market prices \( \mathcal{P}_{\text{mkt}} \), find a continuous-time diffusion \( \sigma \) such that model prices \( \mathcal{P}_{\sigma} \) of \( O \) are equal to \( \mathcal{P}_{\text{mkt}} \).

In the case where \( O \) consists of a single call option, the well-known Black & Scholes model is a solution to problem 1. When \( O \) consists of many call options, the Dupire formula and associated local volatility model is a solution. Actually, it is now well-known that one can take any \( \sigma \) and localize it using the Gyöngy Markovian projection formula \( \sigma_{\text{loc}}(t, S_t) = \mathbb{E}[\sigma^2_t | S_t] \), resulting in the localized volatility \( \mathbb{E}[\sigma^2_t | S_t]^{-1/2} \sigma_{\text{loc}}(t, S_t) \sigma_t \), and the associated class of stochastic local volatility models (SLV) [11]. For more general \( O \), though, solving problem 1 is hard.

As an example, let us consider the case where \( O \) consists of two types of options: european call options for all strikes (smile) and stochastic local volatility models (SLV) [11]. For more general \( O \), though, solving problem 1 is hard.

\[ \text{as \"vanillas\"}, \text{for which } \mathcal{X} := (S_t - k)^+ := \max(S_t - k, 0), \text{where } \text{the option expiry } t \text{ and strike } k \text{ are fixed. We consider continuous-time diffusion models of the form } dS_t = S_t \sigma_t dW_t, \text{where } W \text{ is a Brownian motion, and we let the stochastic volatility process } \sigma \text{ be very general for now, for example it could be path-dependent, function of extra sources of randomness, or rough [7, 10]. Such models are uniquely determined by } \sigma, \text{so we will call } \sigma \text{ a model, and denote } \mathbb{E}[X] \text{ the price of option } X \text{ under model } \sigma. \text{As it is usual to do so, we consider, without loss of generality, zero interest rate and dividends, so that } S \text{ coincides with the forward price process.} \]

One of the most fundamental questions arising in quantitative finance is the following:

**Problem 1.** Given a set of options \( O \) with market prices \( \mathcal{P}_{\text{mkt}} \), find a continuous-time diffusion \( \sigma \) such that model prices \( \mathcal{P}_{\sigma} \) of \( O \) are equal to \( \mathcal{P}_{\text{mkt}} \).

In the case where \( O \) consists of a single call option, the well-known Black & Scholes model is a solution to problem 1. When \( O \) consists of many call options, the Dupire formula and associated local volatility model is a solution. Actually, it is now well-known that one can take any \( \sigma \) and localize it using the Gyöngy Markovian projection formula \( \sigma_{\text{loc}}(t, S_t) = \mathbb{E}[\sigma^2_t | S_t] \), resulting in the localized volatility \( \mathbb{E}[\sigma^2_t | S_t]^{-1/2} \sigma_{\text{loc}}(t, S_t) \sigma_t \), and the associated class of stochastic local volatility models (SLV) [11]. For more general \( O \), though, solving problem 1 is hard.

As an example, let us consider the case where \( O \) consists of two types of options: european call options for all strikes (smile) and all dates \( t \in [t_1, t_2] \), and a Bermudan option with exercise dates \( t_1, t_2 \), for which we have \( \mathcal{X} \equiv \max((S_{t_1} - k_1)^+, \mathbb{E}[S_{t_1} - k_1]^+]) \), where \( k_1, k_2 \) are fixed and \( \mathbb{E}[S_{t_1}] \) denotes the expectation conditional on \( \mathcal{F}_{t_1} \), the information available up to \( t_1 \). Due to \( \mathbb{E}[S_{t_1}] \) and the \("max\" non-linearity, the Bermudan option price depends on the forward smile \( t_1 \to t_2 \) and cannot be replicated with vanillas. Such option price is lower bounded by the maximum of the two european call options \( (t_1, k_1), (t_2, k_2) \), which we call maximum european: \( \mathbb{E}[S_{t_1} - k_1]^+ \), \( \mathbb{E}[S_{t_2} - k_2]^+ \). Assume that \( \mathcal{P}_{\text{mkt}} \) is close to its lower bound, so that our goal is to find \( \sigma \) so as to minimize \( \mathcal{P}_{\sigma} \). First, notice that \( \mathcal{P}_{\sigma} \) equals the maximum european means that if the maximum european is on \( t_1 \), we should never exercise at \( t_2 \) for any path \( \omega \), and vice-versa if the maximum european is on \( t_2 \). One can make the following heuristic reasoning to get the intuition that \( \sigma_t \) for \( t \in [t_1, t_2] \) must remember the value of \( S_{t_1} \), and hence be path-dependent, if we hope to minimize \( \mathcal{P}_{\sigma} \). If the maximum european is on \( t_2 \), you need on paths \( \omega \) for which \( S_{t_1} \) is high, a high volatility \( \sigma_t \) on \( [t_1, t_2] \) so that you do not exercise at \( t_1 \). If the maximum european is on \( t_1 \), the conclusion is

\[ \text{remember that } \mathbb{E}[\mathbb{E}[S_{t_1} | \cdot]] = \mathbb{E}[\cdot]. \]
in making our method work, where policy exploration across all players is obtained via interpolation of a few basis players (\(10^5\)).

2 GAME THEORETICAL FORMULATION OF THE CALIBRATION PROBLEM

As stated in introduction, we let \(S := (S_t)_{t \in [0, T]}\) be the stochastic price path process over some finite time horizon \([0, T]\) on a complete probability space \((\Omega, \mathcal{F}, \mathbb{P})\) and consider continuous-time diffusion models of the form:

\[
dS_t = \sigma_t dW_t, \tag{1}
\]

where \(W\) is a Brownian motion and \(\sigma\) is allowed to be a very general stochastic process.

We start by giving a concise exposition of the key concepts in Markov decision processes (single-agent case) and stochastic games (multi-agent case), that we will use to develop our approach. Reinforcement learning (RL) and its multi-agent counterpart (MARL) are used as numerical methods to solve the latter, which we will focus on in section 3. We refer to [22] for a presentation of the core concepts in RL, and [12, 14] illustrate some of its applications to finance.

Under a Markov decision process framework, an agent evolves over a series of discrete time steps \(t \in [0, T]\). At each \(t\), it observes some information called its 'state' \(x_t\), takes an action \(a_t\), and as a consequence, transitions to a new state \(x_{t+1}\) as well as gets a reward \(r(x_t, a_t, x_{t+1})\), where the transition dynamics \(T(x_{t+1}|x_t, a_t)\) is called the 'environment' and denotes the probability to reach \(x_{t+1}\) conditional on \(x_t, a_t\). The design of the state \(x_t\) is quite flexible, in particular one can include time \(t\) as well as path-dependent information. This is even more true since the recent developments around neural networks which can be fed a large amount of information, and aforementioned path-dependent data can be encoded by means of the hidden state of a LSTM (long short-term memory). The agent samples its actions according to the stochastic policy \(\pi(a_t|x_t)\) representing the probability density function of \(a_t\) conditionally on \(x_t\), and its goal is simply to maximize its expected cumulative reward:

\[
\sup_{\pi} v(\pi) := \mathbb{E}_\pi \sum_{t=0}^{T} r(x_t, a_t, x_{t+1}). \tag{2}
\]

where \(\mathbb{E}_\pi\) indicates that Law(\(a_t|x_t = \pi\), and \(v(\pi)\) is usually called the value function. We focus here on the finite time horizon case \(T < \infty\) and omit the classical discounting of rewards by \(\gamma^t\) since we do not need it in the present work. Due to the finite time horizon, we assume that \(x_t\) includes time \(t\), since it is well-known that solutions of (2) must be time-dependent. A stochastic game is simply the extension of a Markov decision process to \(n > 1\) players: each agent \(i \in [1, n]\) is now allowed to have its own policy \(\pi_i \equiv \pi_i(a_t|x_t)\) and reward function \(r_i\), but the latter now depends, in the most general case, on all agents’ actions and states \(r_i \equiv r_i(x_t, a_t, y_{t+1})\), where we denote vectors in bold \(x_t := \{x_{t}^i\}_{i \in [1,n]}, a_t := \{a_{t}^i\}_{i \in [1,n]}\).

We consider the fully observable case where each player observes information \(x_t\) and actions \(a_t\) from all players at each time \(t\). We focus our attention on symmetric cooperative games where:

- (cooperation) all agents rewards are the same \(r_{i} = r\).
• (symmetry) agents are interchangeable: \( r(x_t, a_t, x_{t+1}) = r_{\rho(t)}(x_t^{\rho}, a_t^{\rho}, x_{t+1}^{\rho}) \) for all permutations \( \rho \), where the "super-script \( \rho \)" denotes the permuted vector. In simple terms, this condition means that what matters is "what" is done and "where", but not "who" did it. In mathematical terms, the "what", "where" and "who" are respectively \( d_t^i \), \( x_t^i \) and \( i \).

One should think about such games as a set of interacting particles, or ants, acting locally towards a common goal at the aggregate level.

Solving cooperative games in the context of MARL has for example been studied in [6, 9]. Similar to the single-agent case (2), we denote the value function of the symmetric cooperative game as:

\[
v(\pi) := \mathbb{E}_{\pi \sim i \in [1, n]} \sum_{t=0}^{T} r(x_t, a_t, x_{t+1}).
\]

**Definition 1. (Nash equilibrium)** The vector of players' policies \( \pi^*: (\pi^*_i)_{i \in [1, n]} \) of the symmetric cooperative game associated to value function \( v \) in (3) is a Nash equilibrium if:

\[
v(\pi_i; \pi^*_i) \leq v(\pi^*_i) \quad \forall \pi_i, \forall i \in [1, n],
\]

where \( v(\pi_i; \pi^*_i) \) denotes the value of the game associated to \( \pi^*_i \), but with \( \pi^*_i \) replaced by \( \pi_i \).

In simple terms, no player has an incentive to deviate unilaterally in a Nash equilibrium.

Back to our calibration problem. Let each Monte Carlo trajectory be a player, so that \( n \) is the number of Monte Carlo trajectories, typically very large \((10^5 - 10^7)\). Let player \( i \)'s action \( a_t^i \) be \( \sigma_t^i \), the volatility to use for trajectory \( i \) over the timestep \([t, t+1]\). We can more generally allow \( \sigma_t^i \sim \phi(\cdot; a_t^i, x_t^i) \) for some distribution \( \phi \), but we discuss this later. The discretization of the dynamics (1) yields:

\[
\ln S^i_t = \ln S^i_t - \frac{1}{2} \sigma_t^i \delta + \sqrt{\delta} \sigma_t^i Z^i_t,
\]

where \((Z^i_t)_{t \in [0,T], i \in [1,n]}\) is a collection of i.i.d. standard normal random variables and \( \delta \) is the time step value, for example \( \delta = \frac{1}{252} \) for a time step of one day. The state \( x_t^i \) typically contains stock and volatility for trajectory \( i \) over a past historical window of size \( h \), \((S_{t-h}^i, \sigma_{t-h}^i)_{t \in [t-h,t]}\), as well as time \( t \). It could also contain information relative to other trajectories, to include the McKean-Vlasov case where the diffusion coefficients depend on Law\((S_t)\). As previously mentioned, the now standard technique to encode history of observations, required for example in imperfect information games, is via the hidden state of an LSTM [9], but other approaches are possible, for example path signatures [1].

We assume that we have \( n^2 \) option cashflows \( \mathcal{X}^m \) \( m \in [1, n^2], t \in [0, T] \) associated to market target prices \( c^m \in \mathbb{R} \). Option cashflows are random variables, and each \( \mathcal{X}^m \) is assumed to be a function of \((x_t, a_t, x_{t+1})\), remembering from our previous discussion that states \( x_t^i \) can potentially encapsulate a large amount of information, including path-dependent one. We then define the game reward obtained at \( t \) as the calibration loss:

\[
r(x_t, a_t, x_{t+1}) := - \sum_{m=1}^{n^2} \omega^m \left( \phi^m_t(x_{t+1}^m) - c^m_t \right)^2,
\]

where \( \omega^m \) are weights chosen by the user (for example related to the Vega of the option), \( x_{t+1}^m \) is the Monte Carlo price and \( \phi^m_t \) a suitable transformation, for instance the Black & Scholes implied volatility mapping in the case of a call option, in which case \( c^m_t \) would be the market implied volatility. In the case of a call option, we have \( X^m_t := (S_t - k_m)^+ \). In the case of the Bermudan \((t_1, t_2)\), we have \( X_t = 0 \) for \( t < t_2 \) and \( X_t := \max((S_t - k_2)^+, \mathbb{E}_t[(S_{t_2} - k_2)^+]) \), where the conditional expectation is calculated empirically using all trajectories over \([t_1, t_2]\) using, for instance, American Monte Carlo regression methods [17].

The deep MARL methods we employ in section 3 allow to have sparse reward signals as it is the case for the Bermudan option, the key being to use a neural network to approximate the value function, i.e. the expected sum of future rewards, given a particular state \( x_t^i \). Such approximation is called a "critic" in RL jargon, and was one of the keys to the success of AlphaGo, Deepmind’s pioneering work on the game of Go [21].

Even though the true rewards can be sparse like in the Bermudan example, it is actually possible in many cases to construct per-step approximations of true rewards. Reward design, or "shaping" is an active area of research in RL. It consists in suitably "shaping" the reward function \( r(\cdot) \) to improve learning. In our case, it is perfectly reasonable to define "fake" cashflows \( \bar{X}_t \) and rewards \( \bar{r} \) such that cumulative rewards agree:

\[
\sum_{t=0}^{T} r(x_t, a_t, x_{t+1}) = \sum_{t=0}^{T} \bar{r}(x_t, a_t, x_{t+1}).
\]

In the Bermudan \((t_1, t_2)\) example, a reasonable choice would be \( \bar{X}_t := \max((S_t - k_1)^+, \mathbb{E}_t[(S_{t_2} - k_2)^+]) \), where \( \bar{X}_t \) can be expressed either in percentage of the forward at time \( t \), or in delta space \( \bar{K}_{2,t} = S_t \left( \frac{k_2}{S_t} \right)^{\frac{T-t}{T-t_2}} \). We then define the fake rewards as:

\[
\bar{r}(x_t, a_t, x_{t+1}) := r_\bar{X}(x_t, a_t, x_{t+1}) - r_\bar{X}(x_{t-1}, a_{t-1}, x_t),
\]

where \( r_\bar{X} \) is the reward in (6) but associated to cashflows \( \bar{X}_t \) and the implied volatility mapping associated to \((k_2, t_2)\). One can observe that (7) is true due to the telescopic property (8) and that both \( r \) and \( \bar{r} \) coincide at time \( t_2 \).

In order to get a more meaningful reward signal, one should subtract to the reward in (6) the calibration error obtained with a reference model, for example the Black & Scholes model or the local volatility model.

We have the following game theoretical interpretation of the calibration problem, which follows by definition of the game reward (3), (6).

**Proposition 1. (Calibration as a Nash equilibrium)** Let the number of Monte Carlo paths \( n \) and time discretization value \( \delta \) be
fixed. Let \( \pi^* := (\pi^*_i)_{i \in [1,n]} \) be a minimizer, over all models \( \sigma \) simulated with Monte Carlo parameters \((n, \delta)\), of the calibration error associated to a given set of options. Then \( \pi^* \) is a Nash equilibrium of the calibration game.

Our calibration game is symmetric, so it is common to look for symmetric equilibria, i.e. where all players play the same strategy \( \pi^*_i = \pi^* \forall i \in [1,n] \) [13, 15, 24]. Asymmetric equilibria of symmetric games exist but they represent a degenerate case [5], i.e. they are not natural and one has to work hard to identify them. We discuss in section 3 MARL driven methods to find such equilibria.

Players' actions \( a_i \) and volatility dynamics \( \sigma \). For simplicity, we previously defined the volatility to be players' actions: \( a_i^t := \sigma_i^t \).

Note that since actions are sampled from the distribution \( \pi_t \), they are by construction stochastic. It is however straightforward to consider a distribution \( \varphi \equiv \varphi(z) \) parametrized by a vector of parameters \( z \), and simply define players’ actions to be those parameters, i.e. \( \sigma^t_i \sim \varphi(a_i^t|x^t_i) \).

Specific examples of such include the case where \( x \) are parameters of a Gaussian mixture, possibly correlated with \( \Delta \). In that case the overall correlation matrix is of size \( n_{mix} + 1 \) (volatility and spot), where \( n_{mix} \) is the number of components in the mixture. The rationale for adding more components is to make volatility increments deviate from Gaussianity, conditional on \( \Delta \).

\( \sigma^t_i \) are parameters driving drift of a Gaussian mixture, possibly correlated with \( \Delta \). In that case the overall correlation matrix is of size \( n_{mix} + 1 \) (volatility and spot), where \( n_{mix} \) is the number of components in the mixture. The rationale for adding more components is to make volatility increments deviate from Gaussianity, conditional on \( \Delta \).

\[ \text{In } \sigma^t_i = \ln \sigma_i^t + \mu_i^t \delta + \eta_i^t \rho_i^t \sqrt{\Delta^t} + \eta_i^t \sqrt{1 - \rho_i^t} \sqrt{\Delta^t}. \quad (9) \]

It is important to keep in mind that since actions depend on the state \( x^t_i \), which as discussed can encapsulate complex information such as path-dependence, \( (\mu_i^t, \eta_i^t, \rho_i^t) \) can be complex functions of the state.

3 MULTI-AGENT REINFORCEMENT LEARNING OF THE CALIBRATION GAME

We discuss in this section a MARL-based method to solve empirically the game presented in section 2. Our main innovation is the concept of basis players needed to handle large number of players, which turned out to be crucial in making our algorithm work. We postpone to future work the connection of our algorithm to mean-field RL based methods. We will consider a policy gradient approach, i.e. where \( \pi \equiv \pi_\theta \) is a neural network with parameters \( \theta \) that will be iteratively improved.

Our action and state spaces are continuous as subsets of \( \mathbb{R}^d \), and we aim at finding symmetric equilibria of the symmetric game, as discussed in section 2, i.e. one where all players play the same strategy. Denoting \( v(\pi'; \pi) \) the expected utility of a player when it plays \( \pi' \) while all others play \( \pi \), one can find symmetric equilibria by solving:

\[ \min_{\pi} \max_{\pi'} v(\pi'; \pi) = v(\pi; \pi). \quad (10) \]

This was done in [15] and takes its origins in [13] under the name SOFA (symmetric opponents form approach). In our context, the \( \min \) and \( \max \) operations each involve solving a RL problem, so they are costly. The alternative that we will adopt is to approximate \( \max_{\pi'} \) by a gradient step. This is a form of self-play, but in the context of \( n > 2 \) players, where \( n - 1 \) players, the "rest of the world", are tied together by a policy \( \pi \). This was discussed in [23, 24] under the name shared equilibrium, where it was shown that policy sharing among players amounts to picking one player at random and making a step towards improving its reward, while freezing other players. In general, it is important to understand that convergence of self-play is linked to transitivity in games and is different, for example, from maximizing the sum of players' rewards. As a simple example, the game where 2 players receive respectively \( f(\pi') - f(\pi) \) and \( f(\pi) - f(\pi') \) has by construction a sum of players' utilities equal to zero, but optimizing at each training iteration the utility of a randomly selected player will eventually lead to finding a (local) maximizer of \( f \).

Recent successes of policy gradient methods are based on two pillars: the policy network \( \pi_\theta \), and the value function network \( \psi \), parametrized by \( \psi \). \( V_\theta(x^t_i) \) estimates the expected sum of future rewards of an agent currently in state \( x^t_i \) and serves as a variance reduction method. The advantage \( A_\theta \) represents the benefit of action \( a \) in state \( x \), relative to some average level in state \( x \):

\[ A_\theta(x, a) := -V_\theta(x) + E \sum_{t'=t}^T r(x^t, a^t, x^{t+1}) | x^t = x, a^t = a. \quad (11) \]

The conditional expectation appearing in \( A_\theta \) is estimated by a sample of the simulated trajectory, but it could be estimated using truncated trajectories instead (Generalized Advantage Estimation) [19].

The gradient \( g_\theta \) according to which the shared policy \( \pi_\theta \) is updated is computed by collecting all players’ experiences simultaneously and treating them as distinct sequences of local states, actions and rewards experienced by the shared policy [9, 24], yielding the following expression under A2C (advantage actor critic) [18], where we allow a set of parallel runs \( B \):

\[ g_\theta = \frac{1}{n} \sum_{i=1}^n \frac{1}{B} \sum_{b=1}^B \sum_{t=0}^T V_\theta \ln \pi_\theta(x^{t+b}_i | x^{t}_i) \cdot A_\theta(x^{t+b}_i, a^{t+b}_i). \quad (12) \]

The gradient in equation (12) is presented in the context of A2C, but other policy gradient updates based on \( A_\theta \) and \( \pi_\theta \) can be used instead, for example we use proximal policy optimization (PPO) in our experiments [20].

The value network \( V_\theta \) is estimated classically by performing a few steps of mini-batch gradient descent on the following objective, at each policy update:

\[ \frac{1}{nB} \sum_{i=1}^n \sum_{b=1}^B \sum_{t=0}^T (V_\theta(x^{t+b}_i) + \sum_{t'=t}^T r(x^{t'}_i, a^{t'}_i, x^{t'+1}_i))^2. \quad (13) \]

Basis players. The gradient update (12) scales with the time horizon \( T \) and the number of players \( n \). It is clearly not efficient when \( n \) is large. In the case of local volatility or particle methods [11], one does not compute the leverage function at every timestep of every Monte Carlo path, as it would be highly inefficient. Instead, one computes it on a much smaller grid of size \( n_g \sim 50 \) in space and interpolate. We take inspiration from this idea to interpolate...
all players from a smaller set of basis players. In practice, a common choice for the action space exploration strategy in policy gradient methods is to let \( \pi_0 = (\pi^H_0, \pi^N_0) \) output the mean and standard deviation of actions:

\[
d_i^t = \pi_0(x_i^t) = \pi_0^H(x_i^t) + Z_{i,t}^N \cdot \pi_0^N(x_i^t),
\]

\[
Z_{i,t}^N \sim \mathcal{N}(0,1), \quad i \in [1, n_p].
\]

\( \pi_0^H \) and \( \pi_0^N \) are natural interpolators, so \( \pi_0^H(x_i^t) \) and \( \pi_0^N(x_i^t) \) can simply be computed by the forward pass of a neural network and as such can be evaluated cheaply for all trajectories \( j \). The exploration variable \( Z_{i,t}^N \) represents how players explore the space of actions and find "good" and "bad" actions, by using (12) to update their knowledge. Without \( Z_{i,t}^N \), i.e. \( Z_{i,t}^N \equiv 0 \), equation (12) would not lead to any knowledge improvement. It is the analogous of the exploration policy used in Q-learning, typically \( \epsilon \)-greedy. Our innovation is to compute action \( a_i^t \) of each trajectory \( j \) as:

\[
a_i^t := \pi_0^H(x_i^t) + \bar{Z}_{i,j}^N \pi_0^N(x_i^t),
\]

\[
\bar{Z}_{i,j}^N = \text{interp}[x_i^t, \pi_0^N]_{i \in [1,n_p]}, \quad j \in [1, n],
\]

where \( \bar{Z}_{i,j}^N \) is interpolated from the \( n_p \) basis players’ exploration \( Z_{i,t}^N \). This way, exploration performed by basis players is propagated to the whole Monte Carlo computation, as each basis player impacts other players in its neighborhood via interpolation. As an interpolation method, we tried linear interpolation and k-nearest neighbors (k-NN), and both yielded similar results. Basis players are picked at random among the \( n \) trajectories at every time \( t \). We summarize our approach in algorithm 1.

Algorithm 1 MARLVol

**Input:** learning rate \( \alpha \), initial policy and value network parameters \( \theta, \psi \), number of basis players \( n_p \).

1. **while** game reward not converged **do**
2. **for** each parallel run \( b \in [1, B] \), time \( t \in [0, T] \) **do**
3. pick \( n_p \) basis players at random among \([1, n] \).
4. sample actions \( a_i^{t,b} \sim \pi_0 \) for basis players \( i \in [1, n_p] \) as in (14).
5. interpolate actions \( a_i^{t,b} \) for all trajectories \( j \in [1, n] \) as in (15).
6. compute \( a_j^{t,b} \sim \varphi(\cdot; a_i^{t,b}, x_i^{t,b}) \) for all trajectories \( j \in [1, n] \) as discussed in section 2.
7. diffuse \( S_{j,t}^L \) for all trajectories \( j \in [1, n] \) as in (5).
8. compute reward \( r_{j,t}^{b} \) for all trajectories \( j \in [1, n] \) as in (6).
9. compute \( g_0 \) as in (12) with trajectories from the \( n_p \) basis players.
10. update policy network \( \theta = \theta + \alpha \cdot g_0 \) and update value network \( V_\psi \) by performing a few steps of gradient descent on the objective (13).

**return** \( \theta \)

---

4 EXPERIMENTS

We consider S&P 500 (SPX) smiles as of August 1st 2018 for \( t_1 = 21 \) days and \( t_2 = 51 \) days. Smiles for dates in between are interpolated.

We first learn local volatility: our target instruments are all call options for all maturities up to \( t_2 \). We see in figure 1 that we are able to fit the continuum of market smiles. In this case \( x_i^t \) consists of \((t, S_i^t)\), and actions \( a_i^t \) consist of the mean and variance of a normal distribution, which we sample \( \ln a_i^t \) from. We trained using a time step \( \delta = 1 \) day (\( \delta = \frac{1}{365} \)), so that \( T = 51 \), \( n = 1.2 \cdot 10^5 \) trajectories, \( B = 10 \) parallel runs and \( n_p = 100 \) basis players. We checked empirically that the results are not too sensitive to these parameters, for example we considered \( T = 102 \) (\( \delta = 0.5 \) day), \( n = 2.4 \cdot 10^5 \), and \( n_p \) from 20 to 100. Our approach doesn’t suffer too much from increasing these parameters except from the incompressible cost of the Monte Carlo simulation, since we do not need to store the computational graph as discussed in section 1.

We then turn to the Bermudan \((t_1, t_2)\) case for which we have \( X := \max\{(S_{t_1} - k_1)^+, E_{t_1}\left[(S_{t_2} - k_2)^+ight]\} \). We choose strikes \( k_1 = S_0 = 1, k_2 = 1.012 \), so that the vanilla calls \((t_1, k_1), (t_2, k_2)\) have the same value: this is the interesting case, since if one call is worth much more than the other, the Bermudan price will already be close to its maximum european lower bound. We define the switch value as the difference between the Bermudan and the maximum european prices, both expressed in implied volatility points associated to \((t_2, k_2)\). By construction the switch value is nonnegative. We consider similar numerical parameters as in the local volatility case, and considered different ways to define actions as discussed in section 2 and in (9), but our conclusions remained similar.

The option price is calculated using American Monte Carlo [17] with polynomial regression of degree 8. We compare two cases regarding the construction of \( x_i^t \): (i) the path-dependent case considers \( x_i^t = (t, S_i^t, \sigma_i^{t-1}, S_i^{t-1}, \sigma_i^{t-1}, \sigma_i^{t-2}, \sigma_i^{t-2}, \sigma_i^{t-3}, \sigma_i^{t-3}, \sigma_i^{t-4}, \sigma_i^{t-4}) \), and (ii) the non path-dependent case considers \( x_i^t = (t, S_i^t, \sigma_i^{t-1}) \). We see in figure 2 that path-dependence is required to minimize the price, while remaining calibrated to vanillas. Note that in this case, our calibration reward only consists of the Bermudan objective, while we ensure calibration to vanillas for all dates \( t \in [t_1, t_2] \) using localization. We check in figures 3–4 the nature of the learnt path-dependence, and we see that as intuited in section 1, we learn a volatility increasing in \( S_{t_1} \), decreasing in \( S_{t_1} \). In figure 4, we see that the cutoff strike \( k_1 \) is perfectly learnt by our RL policy, as it is obviously never optimal to exercise at \( t_1 \) if \( S_{t_1} < k_1 \).

Due to the flexibility in defining the state \( x_i^t \), our approach can be extended straightforwardly to more Bermudan dates, where path-dependent information corresponding to past exercise dates is fed into the state process.

The RL policy \( \pi_0 \) was trained in the RLlib framework [16], and run on AWS using an EC2 C5 xlarge instance with \( B = 10 \) CPUs, using Proximal Policy Optimization [20]. We used configuration parameters in line with the latter paper, that is a clip parameter of 0.3, an adaptive KL penalty with a KL target of 0.01 and a learning rate of \( 0.1 \). Each policy update was performed with 30 iterations of stochastic gradient descent with SGD minibatch size of 0.1 \( \cdot n_p \cdot T \cdot B \). We used a fully connected neural net with 3 hidden layers, 50 nodes per layer, and tanh activation. Although tanh is much more
expensive to evaluate than ReLU, we noticed that it lead to smoother and more efficient learning.

5 CONCLUSION
In this paper we provided a novel view on the derivative pricing model calibration problem using game theory and multi-agent reinforcement learning. We illustrated its ability to learn local volatility, as well as path-dependence in the volatility process required to minimize Bermudan option prices.

DISCLAIMER
This paper was prepared for information purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co and its
affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

REFERENCES

[1] I. P. Arribas, C. Salvi, and L. Sepruch. 2020. Sigmoid-SDEs model for quantitative finance. Proceedings of the First ACM International Conference on AI in Finance (2020).

[2] Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. 2021. Learning to Optimize: A Primer and A Benchmark. https://arxiv.org/abs/2103.12828 (2021).

[3] Tianlong Chen, Wenyi Zhang, Jingyang Zhou, Shiyu Chang, Sijia Liu, Lisa Amini, and Zhangyang Wang. 2020. Training Stronger Baselines for Learning to Optimize. NeurIPS (2020).

[4] C. Cuchiero, W. Khouwari, and J. Teichmann. 2020. A generative adversarial network approach to calibration of local stochastic volatility models. Risks (2020).

[5] Mark Fey. 2012. Symmetric games with only asymmetric equilibria. Games and Economic Behavior 75, 1 (2012), 424–427. https://doi.org/10.1016/j.geb.2011.09.008

[6] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS’16). 2145–2153.

[7] Jun Gathera, Thibault Jaisson, and Mathieu Rosenbaum. 2018. Volatility is rough. Quantitative Finance (2018).

[8] Patryk Gierzatowicz, Mark Sabate-Vidales, David Siska, Lukasz Spruch, and Zan Zuric. 2020. Robust pricing and hedging via neural SDEs. https://arxiv.org/abs/2007.04154 (2020).

[9] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In Autonomous Agents and Multiagent Systems. Springer International Publishing, 66–83.

[10] Julien Guyon. 2014. Multi-agent reinforcement learning in finance. Risk Magazine (2014).

[11] Julien Guyon and Pierre Henry-Labordere. 2012. Being particular about calibration. Risk Magazine (2012).

[12] Ben Hambly, Renyuan Xu, and Huining Yang. 2021. Recent Advances in Reinforcement Learning in Finance. https://arxiv.org/abs/2112.04553 (2021).

[13] Andreas Hefti. 2017. Equilibria in symmetric games: Theory and applications. Theoretical Economics (2017).

[14] Sebastian Jaimungal. 2022. Reinforcement learning and stochastic optimisation. Finance and Stochastics 26 (01 2022), 103–129. https://doi.org/10.1007/s00780-021-00467-2

[15] Zun Li and Michael P. Wellman. 2021. Evolution Strategies for Approximate Solution of Bayesian Games. Proceedings of the AAAI Conference on Artificial Intelligence (2021).
Figure 4: Bermudan option with exercise dates $t_1 = 21$ days, $t_2 = 51$ days. (Left) average volatility $\sigma_t$ and (right) average localized volatility $E[\sigma_t^2|S_t]^{-\frac{1}{2}} \sigma_t$ as a function of $\ln S_{t_1}$. Strike $\ln k_1 = 0$. We learn to allocate higher volatility on the region $S_{t_1} > k_1$ where the $t_1$ call option has non zero payoff. Learnt volatility is an increasing function of $S_{t_1}$, matching our intuition. SPX, August 1st 2018.

[16] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. RLLib: Abstractions for Distributed Reinforcement Learning. In Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80). 3053–3062.

[17] Francis A. Longstaff and Eduardo S. Schwartz. 2001. Valuing American Options by Simulation: A Simple Least-Squares Approach. The Review of Financial Studies. 14 (2001), 113–147.

[18] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (New York, NY, USA) (ICML ’16). JMLR.org, 1928–1937.

[19] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. ICLR (2016).

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. https://arxiv.org/abs/1707.06347 (2017).

[21] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature 529, 7587 (jan 2016), 484–489. https://doi.org/10.1038/nature16961

[22] Richard S. Sutton and Andrew G. Barto. 2018. Reinforcement Learning: An Introduction (second ed.). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

[23] Nelson Vadori, Leo Ardon, Sumitra Ganesh, Thomas Spooner, Selim Amrouni, Jared Vann, Menqda Xu, Zeyu Zheng, Tucker Balch, and Manuela Veloso. 2020. Calibration of Shared Equilibria in General Sum Partially Observable Markov Games. Advances in Neural Information Processing Systems (NeurIPS) (2020).