Hybrid techniques combining simulation and multi-criteria analysis :

An application of Surrogate Modelling to the calibration of Macroeconomic Models.

Siméon Michel

Submitted under the supervision of Prof. Hugues Bersini,
in part fulfilment of the requirements
for the Master in Computer Science and Engineering at
Université Libre de Bruxelles

Academic year 2019-2020

Abstract

Agent-Based Models (ABM) appear as promising tools to analyse macroeconomic problems. However, their use in large-scale situations raises two main challenges : their lack of empirical grounding and the lack of comparability between different ABMs. To bring the Agent-Based paradigm to policy-makers and macroeconomic researchers' attention, formal and well understood calibration methods should be developed. In this thesis, a calibration pipeline combining sequential sampling and supervised machine learning is presented. The simulated data produced by the ABMs are scored against empirical data using time-series comparison. After that, a Machine Learning algorithm is trained on the resulting response surface. Three algorithms are tested : XGBoost, Kriging method and Deep Neural Networks (DNN). We discover that the intelligent parameter space exploration enabled by the surrogate modelling approach can reduce significantly the computation time. The method is applied on two medium-scale ABMs : The Brock and Hommes model, and the Structural Stochastic Volatility model. Among the three ML algorithms, the best accuracy is obtained by the Deep Neural Networks. Despite the high computational complexity of this method compared to the two other methods, the analysis of the speed up power of the surrogate modelling method suggests that the best performing algorithm should always be chosen.

Keywords: Agent based model; calibration; surrogate; design of experiment; sequential sampling.

Résumé

Les modèles multiagents (MMA) semblent être des outils prometteurs pour l'analyse de problèmes macroéconomique. Malgré cela, leur usage à grande échelle soulève deux problèmes importants: leur manque de fondement empirique ainsi que le manque de compatibilité entre différents MMAs. Pour attirer l'attention des décideurs politiques et des chercheurs en macroéconomies, des méthodes formelles de calibration doivent être mises en place. Dans ce mémoire, une procédure de calibration qui combine échantillonnage séquentiel et méthode d'apprentissage supervisé est présentée. Les donnés simulés par les MMAs sont comparées à des données empiriques et classées en utilisant la comparaison de séries temporelles. Après cela, un algorithme d'apprentissage est entrainé sur la surface de réponse correspondante. Trois algorithmes sont testés : XGBoost, la méthode Kriging, et les réseaux de neuronne profonds. Nous découvrons que l'exploration intelligente de l'espace des paramètres permise par la modélisation par métamodèles permet de réduire significativement le temps d'exécution. La méthode est appliquée à deux MMAs d'échelle moyenne : le modèle Brock and Hommes et le modèle de volatilité stochastique structurelle. Parmi les trois algorithme d'apprentissage, la meilleure présicion est obtenue par les réseaux de neurones profonds. Malgré la plus grande complexité de cet algorithme, l'analyse du pouvoir d'accélération de la modélisation par métamodèle suggère que l'algorithme le plus performant devrait toujours être choisie.

Mots-clés: Modèles multi agents; calibration; métamodèle; conception d'expériences; echantillonage séquentiel.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, Agent-based modelling (ABM) has received a growing interest from different scientific communities, leading to progress in various research domains including Ecology, Biology, Social science[1]. The Agent-Based paradigm offers a natural and straightforward approach to modelling systems in which the relation between micro and macro properties is non isomorphic. This applies to a numerous number of systems. For example, in statistical physics, microscopic properties of individual atoms and molecules in a thermodynamic system are studied to derive its macroscopic bulk properties, in sociology, recognising the difference between micro and macro level phenomena (the micro level referring to individual behaviour and the macro level referring to global societal phenomenons) is at the epistemological core of the discipline.

Agent based modelling can be traced down to investigations into complex systems [47]. Complexity in systems has been largely discussed in the scientific literature [28]. We refer to the loose definition given in [48] : ≪*a complex system is one whose evolution is very sensitive to initial conditions or to small perturbations, one in which the number of independent interacting components is large, or one in which there are multiple pathways by which the system can evolve.*≫ Such systems are typically highly non-linear and non eligible to analytical description.

Such complex systems modelling have been achieved traditionally through two different paradigms : Agent-Based modelling and System Dynamics. As opposed to System Dynamics models, Agent-Based modelling can offer a greater freedom in modelling since models of the system are built in a bottom up fashion[40]. Therefore, there is no central or top-down control over their behaviour and more generally over the dynamic of the system[23]. However, this comes at a cost. Indeed, the ability to derive analytically some macro-level properties from the micro-level properties is lost. Different types of Agents

[1]See [32] for a scientometric overview and survey of the diversity and spread of the publications relating to Agent-Based Modelling and its various sub-domains.

are endowed with different behavioural and expectation rules. They interact through explicit protocols so that macro-level variables are only calculable through aggregation of the individual output of the agents. This brings a computational burden that has long discouraged the broad use of ABM. It has long been considered as a good tool to build simple theoretical models, calibrated to reproduce stylised behaviour and to validate theoretical assumptions, but not for running large-scale models.

However, as for the recent surge of Machine Learning techniques, the dramatic increase of computational power that occurred in the beginning of the 21^{th} century enabled techniques that were though to be only applicable on toy models to be operable on large scale problems, bringing Agent-Based paradigm to light again.

Since the first contribution to the foundation of agent based modelling, researchers in economics have used a similar approach to analyse macroeconomic dynamics and policy making, with some preliminary results from Barbara Bergmann's micro-simulation of the US economy [6] and Gunnar Eliasson micro-simulation of the Swedish economy.

However, this streamline of research stayed dissenting because of the difficulties to develop full scale applications and because of the success of methods derived from system dynamics. Indeed, for a long time and still nowadays, the mainstream models have mainly revolved around the Dynamic Stochastic Equilibrium Model (DGSE). Those models present several advantages. The biggest advantage is that, as these models are studied around equilibrium, the aggregation of micro-level variables can often be solved analytically through linearisation around the steady state. Additionally, The predominant place that DGSE has occupied in research and policy making has led to formal and well documented estimation and validation techniques. This is naturally a major requirement for using a modelling tool for real policy assessment.

However, the 2008 subprime crisis has shown that such models, which had long been thought to be trustworthy were in fact badly equipped to deal with crisis. Indeed, they failed to predict the possibility of a financial crisis, but also to provide concrete solutions to resolve it [35]. This inadequacy lies in the very restrictive basic assumptions of DSGE models. The assumptions of rational expectations, representative agents and perfect market are very strong and have been subject to an increasing number of critics in the macroeconomic research community. Such assumptions are indeed rarely verified and rely on little empirical evidence[10]. Additionally, such models rely on local approximations of the model dynamics around the equilibrium. As a consequence, DGSE are not able to model the large fluctuation and regime changes that form crisis. Crisis can then only be generated through external shock brought to the models and cannot be modelled endogenously.

As a response to that failure, several large scale agent-based economic research programs have been launched. Among them, the Eurace project[17] founded by the European Union and carried out by a consortium of 7 universities, the ≪Keynes meet Schumpeter≫(KS) framework[18] developed by Dosi, Fagiolo and Roventini, the java framework JAMEL[42] developed by Salle and Seppecher, to mention just a few[2].

The modelling of economic systems looks like a perfect playground for Agent-Based modelling. Firstly, their constituent agents are highly heterogeneous. They are heterogeneous in their nature since a macroeconomic market is occupied by different agent types : banks, households, firms, central banks,... but also heterogeneous among each type, since different agents can have different behavioural rules. Secondly, each agent interacts with a great number of other agents and such interactions include complex behaviour (including if else loop). This leads to non-linearity of their emergent dynamics and makes them extremely difficult to represent using traditional methods[36].

The enhanced flexibility of Agent-Based modelling gives birth to looser assumptions, allowing to better represent the complex evolving system nature of the economy. Agent behaviour is no longer required to obey rational expectation since bounded rationality is the natural behaviour implemented in Agent-Based modelling.

Still, Agent-Based Modelling is faced with several legitimate critics. Beside the classical critic that focus on the computational difficulties brought by the curse of dimensionality, one of the predominant critics addressed to ABM is the lack of sound empirical grounding. Indeed, when present, the calibration is often achieved through the replication of stylised facts observed in empirical situations. Input parameters of the model are manually adjusted until finding a combination that produces properties resembling to empirically observed properties. For example, one could try to reproduce the non stationary nature of the GDP, or the observed firm size distribution during a specific period of time. Though it can appear sufficient for small-scale theoretical models, a very large number of models are able to reproduce similar properties. These models can have a wide variety of starting assumptions and behavioural rules. This leads to the problem of ≪*wilderness of bounded rationality*≫[16]. Another methodological problem that arises is the comparison of models. Informal calibration does not provide any tools to compare and rate model accuracy.

To encourage the use of the ABM paradigm in macroeconomic research and to bring it to policy maker attention, formal calibration techniques must therefore be developed. The main difficulty in trying to bring models closer to empirical data is the high complexity hidden in AB Models. This complexity brings a computational burden when running simulations. Calibration and validation require an extensive number of runs to produce usable results. This makes Agent-Based model calibration highly sensitive to the *curse of*

[2]See [34] for a succinct comparison of different medium and large scale macroeconomic Agent Based frameworks.

dimensionality.

While a naive way would be to run an high number of Monte-Carlo runs of the model, this method immediately appears as unfeasible. Several calibration methods have been designed in the last few years to propose innovative solutions to this problem. Two families of formal calibration have received the major part of the scientific community's attention : The method of Simulated Minimum Distance, and Bayesian calibration [36]. These two families are both direct calibration methods in the way that calibration is achieved by running a great amount of Monte Carlo simulations on the model itself. Beside these methods, a new approach has appeared on the impulsion of Salle and Yildizoglu on the one hand and Lamperti Roventini and Sani on the other hand. This approach uses an indirect method to derive calibration by using surrogate models or meta-models.

The basis approach is to build a model that approaches well the multivariate input/output relationship of an AB Model and to use it to guide the calibration process. The main advantage is that it allows to dramatically reduce the execution time of the calibration. Salle and Yildizoglu introduced a method using Kriging interpolation as meta-model in [41], while Lamperti Roventini and Sani developed a method using the XGBoost learning algorithm as meta-model in [30]. These two methods have a common methodological approach since they both use indirect calibration. However, a lot of research can still be achieved in that field, since a great freedom is possible in the choice of the meta-model.

In this thesis, I try to apply these two surrogate modelling techniques on standard and relatively simple macroeconomic models : The Brook and Holmes asset pricing model [8] and the Structural Structural Stochastic Volatility model of Franke and Westerhoff [22]. Beside, I develop a third surrogate model using deep neural networks to learn the input/output mapping of the AB Model. This method is based on the work achieved by Tripathy and Bilionis [44] and Yan and Zhou in [50] in which they use Deep Neural Networks (DNN) to build surrogate models for numerical simulators.

The section 2 of this thesis is dedicated to a presentation of the agent-based paradigm applied to macroeconomics. A simple model is first presented to introduce the different concepts related to modelling macroeconomic situation with the Agent-Based paradigm. Then, the Brock and Hommes and the Structural Stochastic Volatility models are presented. Section 2 ends with the presentation of the concept of model calibration and validation as well as the concept of model selection.

The section 3 introduces the different building blocks of a surrogate pipeline. Different functional forms of the surrogates are exposed, as well as the sampling procedure, and the performance measure. Finally, a hyperparameters optimisation and a training procedure are introduced at the end of the section.

The section 4 presents the final surrogate pipeline for model calibration and selection and the section 5 showcases the obtained results.

Finally, concluding remarks are listed in section 6.

Chapter 2

Agent-Based Macroeconomics

In this section, we first review three macroeconomic Agent-Based Models. The first one is a simple model, but yet able to reproduce emergent properties analogous to real-world observed phenomena. The simplicity of the model (taking at most three parameters as input) enables a direct understanding of its underlying principles. For this reason this model will be used throughout this thesis to illustrate different aspects of the proposed surrogate modelling pipeline. The two other methods are medium-scale macroeconomic models built to analyse the dynamic of asset pricing. The fact that both models are derived with the same goal will allow us to compare them in a simple model comparison scheme using the proposed surrogate pipeline. The principles of model calibration and model comparison are exposed in the second part of this chapter.

2.1 Models

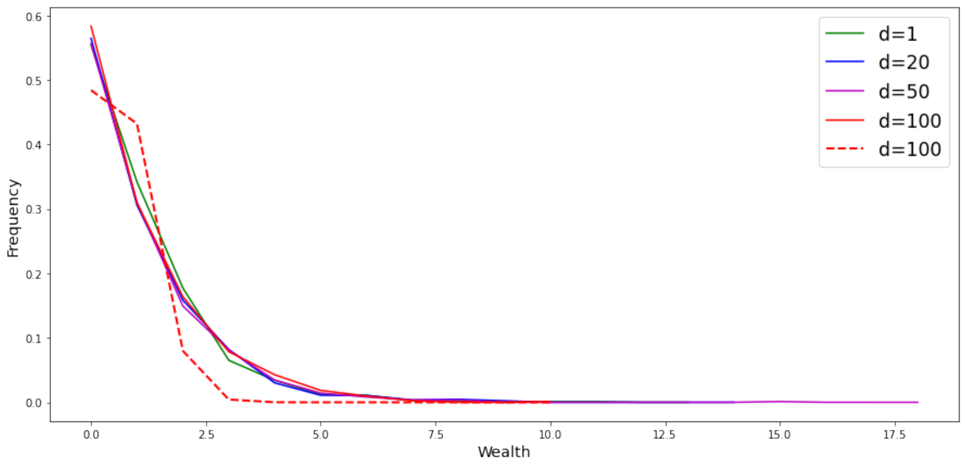
2.1.1 Simple model : The Boltzmann wealth model

The Boltzmann wealth is a simple macroeconomic model developed by Dragulescu and Yakovenko in [19]. Despite its simplicity, it produces an emergent wealth distribution that resembles actual wealth distributions observed in modern society. Indeed, they observed that for any money exchange policy between two agents, conserving the total wealth in the closed economy, the wealth distribution goes at equilibrium to a Boltzmann distribution. The Boltzmann distribution defines the distribution of probability that a system is in a certain state, as a function of molecule speeds for a gas at a certain temperature. It has been observed in numerous societies and economies from ancient Egypt to today that the wealth distribution follows a similar pattern than the Boltzmann distribution [49]. A simple analogy, from statistical physic to economy is that people exchange money when they meet, similarly to gas molecule sharing kinetic energy when they collide.

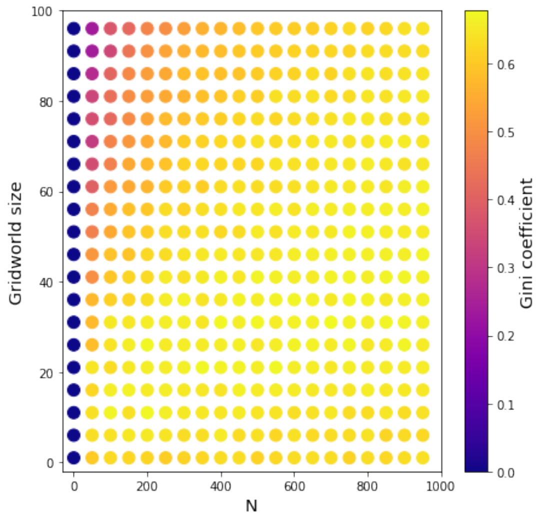
The model in its most simplified version is as follows : a population of N individuals interacts in a small grid world. At initialisation, each agent has a wealth of one unity and is placed randomly in a cell of the grid world. At each time step, each agent performs two actions. First he can move to an adjacent cell of the grid world. Secondly, if the agent is in the same cell as other agents, he randomly picks one of these agents and gives him one unity of wealth. Simulations are performed on a given number of time steps and the distribution of wealth is then observed. It is easy to see that this model is quite similar to the perfect gas models. Indeed, this closed toy economy can be compared to a closed volume of gas that reaches equilibrium starting from a given configuration.

In figure 2.1c, we observe the Gini coefficient of the economy at each time iteration (the Gini coefficient measures the degree of inequality of the system). Larger systems (with bigger grid worlds) take more time to achieve equilibrium. Again, this is in line with the gas analogy, where gas in a bigger volume will reach equilibrium after a longer time since collisions are less frequent. In figure 2.1b, we observe the resulting Gini coefficient for different population and grid size combinations.

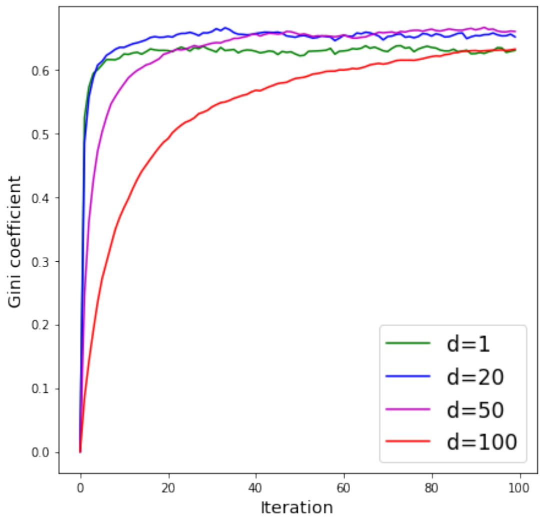
In figure 2.1a we observe the wealth distribution associated to a population of 1000 individuals for different grid world sizes (The grid world is chosen to be square so that d is the edge length of the square). The red dashed line corresponds to a Boltzmann distribution. We observe that the resulting distribution has a fatter tail than the associated Boltzmann distribution. This is a well known phenomenon, known as the Pareto distribution of top incomes. Empirical wealth distributions often observe a parametric distribution where the wealth distribution of the lower class follows a Boltzmann law, and the one of the upper class, which is made up of people who saved the most, follows a Pareto distribution.



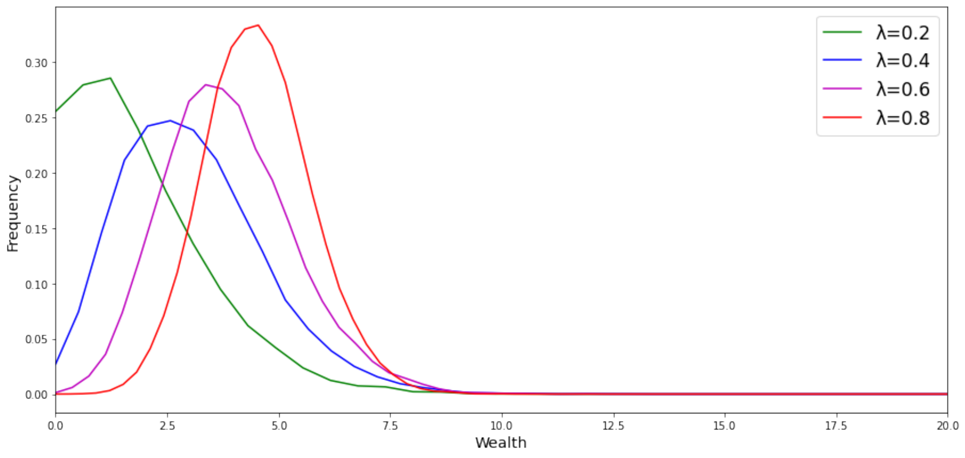
(a) Wealth distribution under different grid world sizes



(b) Gini coefficient at final round for different grid world sizes and population size combinations



(c) Gini coefficient at each iteration for different grid world sizes



(d) Wealth distribution under different saving factors

Figure 2.1: The Boltzmann wealth model

2.1.2 Simple model : The Boltzmann wealth model with savings

In figure 2.1a, we see that the most probable wealth for an individual to end up with is zero. This is not in line with wealth distribution observed in real society, where ultra-poor people do not form a majority. To obtain more accurate results, the propensity to save of individuals must be taken into account. Indeed, not in any society will individuals trade their entire wealth every time they exchange on the market. In line with the approached proposed in [11], we introduce a saving factor λ accounting for the fraction of wealth that each individual keeps out of his trading budget. While this fraction is obviously not homogeneous in a real population, we choose here to keep it as a constant of the model for the sake of simplicity. We observe in figure 2.1d, that this leads to a non zero, most probable wealth and that, as the saving factor goes up, the bump of the distribution slide to the right. This is in line with the results of [11] where the introduction of saving leads to an asymmetric Gaussian-like curve showing some kind of self-organisation in the money market. The different parameters of the experiment with their value ranges is shown in tabular 2.1.

Table 2.1: Domain of experiment for the Boltzmann model

Parameter	description	possible value	chosen interval
N	population size	$\mathbb{R}^+$	[10,1000]
λ	saving factor	[0,1]	[0,1]
d	grid world edge size	$\mathbb{R}$	[1,100]

2.1.3 The Brock and Hommes model

The Brock and Holmes asset pricing model was first described in [8]. The model studies the dynamic of asset pricing in a the market where expectation is formed heterogeneously between agents. The model presents an evolutionary dynamic between the different competing prediction strategies.

Each agent $a = 1, ..., n$ has a finite set of possible strategies and updates its belief over time following a fitness measure. The performance of each strategy is publicly available. To keep the model simple, the number of strategies is chosen to be two $h_a \in \mathbf{H} \equiv \{h_1, h_2\}$. Each trader has two assets at its disposal, a risky one, which pays an uncertain dividend y and has price p, and a risk-free one with gross return $R = (1 + r)$. At each time iteration, the trader can adjust the proportion of each asset in his portfolio. The belief of a trader agent about the price x_t of the risky asset is formed according to its prediction strategy (here h_1 or h_2). The evolutionary dynamics leads up to more selections of the strategy with higher fitness at each iteration. The formulation presented here-above is the one of

[26]. The wealth dynamic is given by :

$$W_{t+1} = RW_t + (p_{t+1} + y_{t+1} - Rp_t)z_t \tag{2.1}$$

With z_t being the number of shares bought a time t. For each strategy, the number of share bought a iteration t is obtained by a mean variance maximiser with α representing the risk aversion:

$$max_{z_{h,t}}\{E_{h,t}(W_{t+1}) - \frac{\alpha}{2}V_{h,t}(W_{t+1})\} \tag{2.2}$$

The associated solution is given by :

$$z_{h,t} = \frac{E_{h,t}(p_{t+1} + y_{t+1} - Rp_t)}{\alpha\sigma^2} \tag{2.3}$$

With σ^2 being the conditional variance of excess return, considered as constant among each trader and over time. We consider here the case of a closed system where there is no supply of outside share, in this case the market equilibrium is given by :

$$Rp_t = \sum n_{h,t}E_{h,t}(p_{t+1} + y_{t+1}) \tag{2.4}$$

Where $n_{h,t}$ is the share of risky asset that trader h holds a time t. The fundamental pricing of the asset under the no-arbitrage hypothesis is given by :

$$Rp^*_t = E_t(p^*_{t+1} + y_{t+1}) \tag{2.5}$$

In the original formulation the price is formulated as a deviation from the fundamental price : $x_t = p_t - p^*_t$. The important hypothesis of the model is to consider that the formation of expectation of the different strategies is of the form :

$$E_{h,t}(p_{t+1} + y_{t+1}) = E_t(p^*_{t+1}) + f_h(x_{t-1}, ...x_{t-L}) \tag{2.6}$$

That is, price are expected to deviate from the fundamental price and the deviation depends upon all the past deviations. Brock and Hommes postulate a simple form for the agent belief :

$$f_{h,t} = g_h x_{t-1} + b_h \tag{2.7}$$

With x_{t-1} being the price at the previous iteration, g_h the trend component and b_h the bias. In tabular 2.2, the agent behaviours corresponding to different bias/trend combinations are described. An unbiased agent with positive trend is called a pure trend chaser since it follows directly the past evolution of the asset price (strong trend chaser if $g > R$) . The opposite behaviour is called contrarian (strong contrarian if $g > R$). If the trend component is zero, the agent is said to be either purely upward biased (if $b > 0$) or purely downward biased (if $b < 0$). The special case $g = b = 0$ is referred as fundamentalism. Agents with this particular behaviour believe that the prices will return to their fundamental value.

Table 2.2: Agent behaviour types

Trend component	Bias component	Agent type
$g_h > 0$	$b_h = 0$	Pure trend chaser
$g_h < 0$	$b_h = 0$	Contrarian
$g_h = 0$	$b_h > 0$	Upward biased
$g_h = 0$	$b_h < 0$	Downward biased
$g_h = 0$	$b_h = 0$	Fundamentalist

At each time iteration, a trader chooses its strategy according to a fitness function measure :

$$\pi_{h,t-1} = \frac{1}{\alpha\sigma^2}(x - Rx_{t-1})(g_h x_{t-2} + b_h - Rx_{t-1}) \tag{2.8}$$

$\pi_{h,t-1}$ is the past realised profits. It is possible to add a memory by using the utility as fitness function :

$$U_{h,t-1} = \pi_{h,t-1} + \eta U_{h,t-2} \tag{2.9}$$

with η being the memory strength. At each iteration the equilibrium price is :

$$Rx_t = \sum_{h \in \mathbf{H}} n_{h,t-1} f_{h,t} \tag{2.10}$$

Where R is the gross return of the risk-free asset and $n_{h,t-1}$ is the fraction of agents with behaviour h at time $t-1$. At each time iteration, agents have perfect foresight about past prices and fractions of all other belief types so that they can compute equation 2.10 perfectly. Combining equation 2.9 and 2.10, we obtain the fraction of each strategy :

$$n_{h,t} = exp\left(\frac{\beta U_{h,t-1}}{\sum_{h\in\mathbf{H}} exp\beta(U_{h,t-1})}\right) \tag{2.11}$$

A run of the Brock and Hommes model goes as follows : Starting from an initial strategy distribution $n_1 + n_2 = 1$, the equilibrium price is calculated using equation 2.10. After that, the utility of each strategy is computed using equation 2.8 and 2.9. Given the fitness measure of each strategy, the new fraction of each strategy is derived from equation 2.11. The domain of experiment is shown in tabular 2.3. The theoretical support for a given parameter is given in the third column. In the fourth column, the intervals that will be used for the model calibration are displayed. They follow mainly the recommendations of Brock and Hommes in their original paper, as well as the calibration exercise performed in [30].

Table 2.3: Domain of experiment for the Brock and Hommes model

Parameter	Description	Support	Chosen interval
β	intensity of choice	$\mathbb{R}^+$	[0,10]
b_1	type 1 bias	$\mathbb{R}$	[-2,2]
b_2	type 2 bias	$\mathbb{R}$	[-2,2]
g_1	type 1 trend component	$\mathbb{R}$	[-2,2]
g_2	type 2 trend component	$\mathbb{R}$	[-2,2]
η	memory strength	[0,1]	[0,1]
σ	asset volatility	[0,1]	[0,1]
α	risk aversion	$\mathbb{R}^+$	[0,100]
R	gross return	$\mathbb{R}^+$	[1,1.2]

2.1.4 The Structural Stochastic Volatility model

The structural stochastic volatility model (SSV) is an asset pricing model developed by Franke and Westerhoff in [21]. It has been developed on the same basis than the Brock and Hommes model. Two trading strategies are competing on a market and the interplay of their counteracting forces influences the price dynamics. More specifically, it is the time-varying population shares of the two different groups that acts mainly on the pricing dynamic. In the Brock and Hommes model, these shares are computed functionally from a fitness measure. The opposite approach is to base the shares dynamic on transition probabilities with which each agent switches between the strategies with a given probability. This is the approach of Franke and Westerhoff. It is known as TPA (Transition Probabilities Approach). At the macroscopic level, the aggregation of the agent switches leads to an adjustment equation for the relative fractions of each group of traders.

The two trading strategies are chartism and fundamentalism[1]. Chartists (or trend chaser) chooses their buy/sell strategy by analysing past price movements. On the other hand, fundamentalists think that the current mispricing of an asset will eventually vanish to reach some fundamental value of the asset. Their mathematical expressions are shown in equation 2.12 and 2.13 where d_t is the demand for an asset at time t for a given strategy, p^* is the fundamental price of the asset and p_t is the price of the asset at time t.

$$d_{f,t} = \phi(p^* - p_t) + \epsilon_f \qquad \phi \in \mathbb{R}_0^+ \tag{2.12}$$

$$d_{c,t} = \chi(p_t - p_{t-1}) + \epsilon_c \qquad \chi \in \mathbb{R} \tag{2.13}$$

ϕ and χ quantifies the aggressiveness of fundamentalists and chartists on the market. The demand of the fundamentalist traders is proportional to the gap between the current

[1]We observe here a similarity with the Brock and Hommes model. Indeed, both model use two strategies and each strategy depends on two parameters: the bias and trend for the BH model and the aggressiveness and noise for SSV. Both models explore the same behaviour space with a different formulation.

price and the fundamental. On the other hand, the demand of the chartist traders is proportional to the last price fluctuation. We observe that a negative value of χ amounts to a contrarian behaviour (see tabular 2.2). The noise component ϵ_f and ϵ_c are both normally distributed random variables with zero mean and variance σ_c and σ_f. They can be seen as a digression from the core demand of the group, represented by the first term of both equations. Indeed, it is convenient to assume that a wide variety of specifications of both trading strategies exist in the population, not to mention that each agent can use a mix of both strategies. Because implementing these specifications in the demand equations would be tricky, those error terms are added to capture the within-group heterogeneity.

It is handy to express the fractions in terms of a majority index x_t of the fundamentalist[2]. This index ranges between -1 (if all the traders are chartist) and 1 (if all the traders are fundamentalist).

$$n_{f,t} = \frac{1+x_t}{2} \tag{2.14}$$

$$n_{c,t} = \frac{1-x_t}{2} \tag{2.15}$$

Total excess demand is then given by $\frac{1+x_t}{2}d_{f,t} + \frac{1-x_t}{2}d_{c,t}$. In all generality, this excess demand will not always balance. In contrast with the Brock and Hommes model where the market is considered as closed, here, a global market agent absorbs the excess of supply and serves the excess of demand. The price is adjusted at each time iteration in the direction of the excess of demand (supply) with a given factor μ. This factor represents the market liquidity, a higher μ value corresponds to a market with low liquidity. The price dynamic equation can be written :

$$p_{t+1} = p_t + \mu\Big(n_{f,t}\phi(p^* - p_t) + n_{c,t}\chi(p_t - p_{t-1}) + \epsilon_t\Big) \tag{2.16}$$

$$\epsilon_t \equiv \mathbf{N}(0, \sigma_t^2) \qquad \sigma_t^2 = \frac{(1+x_t)^2\sigma_f^2 + (1x_t)^2\sigma_c^2}{2} \tag{2.17}$$

The error term ϵ_t comes from the fact that the sum of two normal distributions with zero mean is again a normal distribution with zero mean and variance equal to the sum of the two single variances. We observe that the global variance is a function of the fraction of traders by means of the majority index. Indeed, given the fact that the two noise terms have different variances, a variation of the population has a double effect on the price equation : a direct effect with the presence of $n_{f,t}$ and $n_{c,t}$ and an effect on the variance of ϵ_t. The time variation of σ_t is called the *Structural Stochastic Volatility* and gives its name to the model.

To complete the model, a definition of the time variation of the majority index is

[2]A majority index of the chartist would be an equally good choice.

needed. Each fundamentalist has, at each time iteration, a probability $\pi_{t,f\to c}$ to switch to chartism and each chartist has a probability $\pi_{t,c\to f}$ to switch to fundamentalism. The number of agents is considered as sufficiently large to ignore the intrinsic noise from different realisations. The variation of the majority index is then given by :

$$x_{t+1} = x_t + (1 - x_t)\pi_{t,c\to f} - (1 + x_t)\pi_{t,f\to c} \tag{2.18}$$

Obviously, these probabilities are not constants over the experiment. To characterise the evolution of these probabilities, a switching index s_t is defined. An increase in s_t favours the chartist to switch to fundamentalism and vice versa. This switching index is constructed to take into account three factors that have proven to influence the behaviour of traders in the market. These three factors are quantified by the three α parameters described here-above.

$$s_t = \alpha_o + \alpha_x x_t + \alpha_d(p_t - p_t^*)^2 \tag{2.19}$$

- α_o is a predisposition parameter. It represents the natural predisposition of agents to switch to fundamentalism. A negative value of α_O favours the inverse conversion (from fundamentalist to chartist).
- α_x is the herding parameters[3]. The second term of equation 2.19 favours the transition to the most popular strategy. Indeed, researchers in behavioural finance identified that traders tend to sometimes follow the trend instead of acting on their own analysis.
- α_d measures the influence of price misalignment. Indeed, when the price moves largely away from its fundamental value, traders tend to go back to fundamentalist behaviour.

The probabilities of transition are specified as :

$$\pi_{t,c\to f} = \nu exp(s_t) \tag{2.20}$$

$$\pi_{t,f\to c} = \nu exp(-s_t) \tag{2.21}$$

With ν being a flexibility parameter that measures the ease of strategy change of the traders. We observe that if ν is close to zero and s_t is bound, then we have the probabilities of transition that stay in the [0,1] interval.

A run of the structural stochastic volatility model goes as follows : first the new price is computed using equation 2.16, then the switching index is calculated with the new

[3]Investopedia defines herding as the phenomenon by which investors follow what they perceive other investors are doing, rather than acting on their own analysis.

price using equation 2.19. The corresponding values of the probabilities of transition are then updated with equation 2.20 and 2.21. Finally, the majority index is computed using equation 2.18. The model parameters, as well as their considered ranges, are listed in tabular 2.4. The intervals chosen for calibration are mainly taken from [20].

Table 2.4: Domain of experiment for the Structural Stochastic Volatility model

Parameter	Description	Support	Chosen interval
ϕ	aggressiveness of fundamentalists in the market	$\mathbb{R}_0^+$	[0,5]
χ	aggressiveness of chartists	$\mathbb{R}$	[-5,5]
σ_f	noise in fundamentalist demand	$\mathbb{R}^+$	[0,2]
σ_c	noise in chartist demand	$\mathbb{R}^+$	[0,2]
μ	market impact factor of demand	$\mathbb{R}$	[-2,2]
p^*	log of fundamental value	$\mathbb{R}$	0
ν	flexibility parameter in the population dynamics	$\mathbb{R}^+$	[0,1]
α_o	predisposition parameter in the switching index	$\mathbb{R}$	[-5,5]
α_x	herding parameter in the switching index	$\mathbb{R}$	[-5,5]
α_d	dispersion parameter	$\mathbb{R}^+$	[0,100]

2.2 Model calibration and validation

There is still no consensual definition of calibration and validation taking preference in the field of Agent-Based modelling. Additionally, these two terms are often used to refer to similar approaches and techniques. Here, we differentiate both terms by their time of apparition in a modelling process. Calibration refers to the process of adjusting model parameters to obtain the best fit between the model calculation and an empirical set of observations. Validation is performed afterwards on a different dataset to verify that the learned parameters are still meaningful when applied to qualitatively equivalent data. Note that this definition is quite general since no assumptions are made on the method used to perform such tasks.

A critic that has been addressed to agent based modelling is the lack of rigorous calibration and validation. Bringing ABM closer to data is a key challenge to make them attractive for policy-makers. The simple replication of stylised facts does not appear as a good enough criterion to select a given model for policy analysis. Indeed, it appears that a broad range of models with highly different internal dynamics and statistical properties are able to reproduce the same set of stylised facts.

Such critique has pushed scientist to offer new rigorous methods to calibrate models, hence bringing to light a broad set of tools with different theoretical fundamentals. Among them, the Method of Simulated Minimum Distance (SMD), Bayesian inference and Markov Chain Monte Carlo (MCMC)[4]. However, all those methods present the inconvenience of requiring a significant number of model runs to provide a good estimation. This fact makes it difficult to calibrate large-scale ABM with these methods.

The most straightforward approach to calibration would be to directly estimate a model by the proximity of its underlying causal structure to actual causal structures observed in real-world economies. Since this approach would give certain guarantees about the relevance of the causal chain of the model, it guarantees good policy-analysis properties. Indeed, a model able to produce high quality data is not automatically well suited for policy analysis. This is analogous to the well-known "black box" problem that face Deep Neural Networks. No matter how good the performances are, the fact that the inner dynamic of a model is partially or totally hidden causes a problem for model analysis and comparison. However, no tools are available to effectively and quantitatively compare the causal mechanisms of ABM with empirically observed mechanisms[5].

Another approach is to ignore the underlying causal structure and to focus on the conditional probability structure of the produced data. The methodological approach is then to match two distributions : The distribution produced by the ABM during run-time on a specific macroeconomics variable and the actual empirical distribution of that variable. This is then a black-box approach since the internal mechanisms of the ABM are not taken into account. The ability to characterise the relevance of the internal formalism of the model is lost for the benefit of a more general approach. Indeed, in this context, the implementation of the ABM is irrelevant as long as it produces sequential data. The method can then be theoretically applied to almost any computational system[6]. This is the approach taken in this thesis.

2.2.1 Mathematical framework

Generally speaking, an ABM in a mapping from an input space containing all the possible parameters combination $D \subset \mathbb{R}^k$, with k the number of parameters of the ABM, to an output space $O \in \mathbb{R}^m$.

$$ABM : \mathbf{x} \in D \subset \mathbb{R}^k \to ABM(\mathbf{x}) \in O \subset \mathbb{R}^m \tag{2.22}$$

[4]see [36, 31] for a review of the different methods in the scientific literature

[5]Recent years have seen the emergence of attempts to compare the causal mechanisms underlying real and simulated data. For example [24] using SVAR regression or [3] using the Markov information criterion (MIC). However, this is not equivalent to directly compare the causal mechanisms themselves.

[6]Or at least to any system that can be represented by a finite-order Markov process.

Each parameter can have a different value range based on the user defined preference or based on the specific nature of that parameter. For example, in a macroeconomic scenario, a parameter could be the risk-free interest rate. In this configuration, the range to consider could be $[1, 1.3]$. The input space can have a dimension between one or two parameters to several tens of parameters for large models.

O is the output space. It is typically larger ($m > k$) since it contains the time series realisations of a large number of micro and macro level variables. However, to provide insightful results, the output space is typically synthesised into a set of aggregates variables[7]. Mathematically speaking, we define a projection of the output space onto a smaller space of variables of interest : $p : O \subset \mathbb{R}^m \to O_{synth} \subset \mathbb{R}^s$. This subset can contain for example the GDP growth rate, the inflation rate, an asset price,... In the rest of this thesis, the output set of the ABM will always refer to this synthesised output.

To provide a quantitative calibration, the model is estimated numerically at different points of the input space. Then, statistical properties of a certain output quantity are compared to the statistical properties observed in empirical data, in order to assess the quality of the model. Two different ways of comparing ABM realisation to empirical data are possible. The first one is to define a threshold starting from which the experimental and the empirical distribution are considered to be similar enough. This is referred as discrete calibration since the result of a calibration will be a Boolean. The other way is to assess the similarity of both distributions by calculating their distance by means of a specific metric. This is referred as continuous calibration since the outcome will be real-valued. Generally speaking, the calibration is a search process in which we aim to identify input parameter configurations $\mathbf{x} \in D$ leading to an output close enough to empirical observations.

$$Search \quad all \quad \mathbf{x} \quad s.t \quad dist(ABM(\mathbf{x}), y) \leq c \tag{2.23}$$

2.3 Model comparison

Beside the lack of rigorous calibration techniques, a second issue traditionally pinned to the ABM paradigm is the lack of comparability between models. Indeed, we have seen that different models with different structures and theoretical fundamentals are often able to reproduce a same set of stylised fact. When it is the case, an in-depth analysis to compare their performance and explanatory power should be undertaken. As for calibration, a direct comparison of the causal structure of various models is the most natural approach.

[7]Given the current state of knowledge, calibration is only achieved at the scale of macro level variables. The use of micro level variable realisations at calibration is a promising path for further research (see [12]). Nevertheless, it raises the challenge of empirical micro level data availability

However, no tools are yet available to perform this comparison quantitatively. The other approach is to rank the models by their performance against a given empirical dataset. We see that model calibration and model selection are tightly related since the calibration allows to derive a certain performance index that can further be used for model comparison. The difficulty here is that to be comparable, two models must primarily have the same scope (that is, both models must produce an output comparable to the considered empirical Dataset) but they should also share the same scale (The initial situation contained in their input parameter constellation should describe comparable situations).

Chapter 3

Surrogate Modelling

3.1 Overview

Surrogate models, or meta-models are built to approximate the complex multivariate input/output relationship of a system with a limited set of computational expensive simulations. Such models are built on top of complex systems and allow to derive an approximation of the inner distribution of the system. The benefit of the approach is that the surrogate model, after training, runs way faster than the system it tries to mimic. Building an appropriate surrogate allows to perform complex tasks on the system approximation in a restricted time.

A typical example of such a complex task is the Sensitivity Analysis. Indeed, when building a complex model, containing a lot of input variables, relationships between inputs and outputs are often poorly understood. Agent-Based models have well-defined microscopic properties, however, their interaction brings to light complex and hardly predictable macroscopic properties. Being able to quantify how much each input variable contributes to the global uncertainty of the model output is then an essential feature for quality assessment of the model. This is the task of Sensitivity Analysis. It immediately appears that such a task requires a high number of model runs. If the model shows stochastic properties, the number of runs required is even larger. We notice here the usefulness of having a good meta-model.

A typical example of a good candidate for surrogate modelling is a chemical reactor in which input properties of the reagent are observed as well as final properties of the system at the end of the reaction. In case of complex reactions, such systems are often not eligible to an analytical resolution. Furthermore, the implementation and the completion of the reaction can be an expensive process to implement. To be able to derive results under different input configurations without having to actually conduct the experiment each time, one could use a surrogate modelling of the reaction. This consists in studying the reaction under a broad range of input configurations and deriving a table of input/output properties of the reaction. Then, a model is trained to be able to approximate the input/output relationship of the experiment.

For a macroeconomic model, the principle is the same. The results of several computation of the ABM are recorded and the surrogate model is then trained on the input/output combination. Such an approach comes with a high modelling freedom. We identify three modelling choices that will highly influence the nature and the performance of the surrogate model :

1. The choice of the model : The first important choice is obviously the topology as well as the nature of the chosen surrogate. For this choice, the fields of machine learning and statistics provide a lot of well-documented tools (Gaussian Process, Regression trees, Neural Network, Polynomial Chaos Extension ...). A good understanding of the properties of the system is important to make a good choice, as certain assumptions can guide the choice of the model (smoothness, linearity,...). Many different surrogate types have been used in the literature, including Kriging regression [41, 5], Gradient Boosted Trees [30, 51] and Neural Network [44]. The choice is pretty much problem oriented. As for all machine learning problems, there is no free lunch, good model selection and validation techniques are needed to get appropriate results.

2. The sampling procedure : To train a chosen surrogate to approximate the original model, one needs to possess a training dataset of input/output combinations of the original model. This dataset is generated by choosing a certain number of inputs and by running them through the original model to get the corresponding output. The method used to choose the set of input to use is referred as the sampling. The choice of the sampling procedure will deeply influence the performance of the surrogate. We observe here the classical dilemma of exploitation versus exploration. Indeed, the modeller has to choose if he wants his model to approximate well the system in a narrow manifold representing the area of interest or if he wants his model to approximate well the system on a broader manifold representing the feasible space of the system. In the first case, the modelling is precise for a small subset of the possible input (considered as more important) but trying to predict the output of a sample that does not pertain to this region will potentially provide meaningless results. In the second case, the modelling will be more versatile, as each part of the domain is predicted with the same precision. However, if the parameter combinations used in practice form a small subset of the input domain, there is a loss of efficiency in predicting each part of the domain equally well. Obviously, the best sampling will always results in a trade-off between those two approaches. Another important

fact to take into account is that the number of runs of the original computationally demanding model is restraining. This implies that the sampling should represent at best the extent of the input domain while keeping the number of samples at the lowest possible level. We will see in section 3.4 that the space-filling properties of a sampling procedure can help lower the amount of samples needed for a meaningful modelling.

3. The performance measure : The performance measure refers to the quantity optimised by the surrogates to appropriately fit the original model. The ABM output set can contain a considerable number of variables. The choice of the output variable (or variables) used to calculate the error of a given prediction as well as the choice of the functional form of the error can have a huge impact. Furthermore, in the case of calibration, a fitness surface calculated with respect to an empirical distribution can be used. In that case, the surrogate is no longer trained to predict the value of an observable output variable of the ABM but rather to predict the distance between the data generated by the ABM and equivalent empirical data observed in a real world situation.

In contrast with the example of a chemical reactor, Agent Based models have the particularity of being fully computational simulation tools. This has the important consequence that the production of input/output data can be pipelined into the process of generating a surrogate model. In the context of calibration, the production of labelled data can be pipelined as well. By labelled data, we mean data whose divergence with a given empirical data has been ranked. This allows the learning process to be fully adaptable since new data can be generated at run-time. It means that the focus on exploration versus exploitation can be decided at run time.

3.2 Mathematical framework

We aim to approximate an original ABM $\mathcal{A}$ with a surrogate model $\mathcal{S}$. Conceptually, the ABM is no more than a mapping from an input set $D \subset \mathbb{R}^k$ to an output set $O \subset \mathbb{R}^m$. The input set contains a positive number of initialisation parameters k, $k > 1$. We will refer to D as the parameter space of $\mathcal{A}$. Each point of the parameter space $\mathbf{x} = (x_1, ...x_k)$, $k > 1$ is mapped to the output space by the ABM :

$$\mathcal{A} : \mathbf{x} = (x_1, ..., x_k) \in D \subset \mathbb{R}^k \to \mathcal{A}(x) \in O \tag{3.1}$$

The output set contains an arbitrary number of macro level properties of the ABM at the end of a run. The surrogate model generation comes down to a supervised learning task in which we aim to learn a given model $\mathcal{S}$ to approximate $\mathcal{A}$.

$$\mathcal{S} : \mathbf{x} = (x_1, ..., x_k) \in D \subset \mathbb{R}^k \rightarrow \mathcal{S}(x) \tag{3.2}$$

This supervised learning task amounts to a minimisation problem in which we aim to minimise a certain distance metric between $\mathcal{S}(\mathbf{x})$ and $\mathcal{A}(\mathbf{x})$.

$$\min_{\mathcal{S}} \sum_{\mathbf{x} \in \mathbf{DOE}} dist(\mathcal{A}(\mathbf{x}), \mathcal{S}(\mathbf{x})) \tag{3.3}$$

The Domain of Experiment (DOE) is the $n \times k$ matrix of the n chosen experimental points : $\mathbf{DOE} \equiv \{\mathbf{x}_1, ...\mathbf{x}_n\} \in M_{n,k}(D)$ with $M_{n,k}$ being a certain sampling function.

We can observe in equation 3.3 the three important aforementioned choices :

1. The Surrogates model $\mathcal{S}$:
2. The sampling function $M_{n,k}$
3. The performance metric $dist(\mathcal{A}(\mathbf{x}), \mathcal{S}(\mathbf{x}))$.

In the following sections, we study each of these choices. The problem of hyperparameters optimisation and the training procedure are discussed as well.

3.3 Choice of the model

The objective here is not to review all available methods for surrogates modelling. This would indeed not be possible considering the broad range of models proposed in the literature. As previously noted, Surrogates modelling comes down to function approximation. In some cases, a good understanding of the inner dynamics of models, or some prior information can guide the choice of an appropriate model. However, in complex cases, little or no information is available about the shape of the function. A broad set of different methods have been used in the surrogate modelling literature in different fields, including Statistic, Machine Learning, Design of Experiment,.. Three models frequently used in surrogates modelling are described here. They have been chosen for the great diversity they present in terms of structural properties and methodological approach. The three models considered are the Gradient Boosting Ensemble Trees Learner (XGBoost), the Kriging Method (also referred as Gaussian process regression in the machine learning community), and the Deep Neural Network (DNN).

3.3.1 XGBoost

XGBoost has gained a growing interest in the past years. Introduced by Tianqi Chen and Carlos Guestrin in [13], it has since then been credited with a numerous number of Kaggle competition successes and been used in several new cutting edge technologies. XGBoost is a decision-tree based ensemble Machine Learning algorithm using gradient boosting. It can be seen as an ensemble of *weak* approximators forming together a *strong* approximator. Besides its versatility and its good performance, the ensemble tree method allows to derive directly a feature importance ranking according to the relative number of times a parameter is split-on on the trees [2].

XGBoost is based on the Gradient Boosting Trees algorithm and provides several systems and algorithmic optimisations. We describe below the mathematical principles underlying the Gradient Boosting Trees algorithm.

Let us take our surrogate formulation where we aim to learn a surrogate model $\mathcal{S}$ to approximate the ABM $\mathcal{A}$ on a certain Domain of Experiment $\mathbf{DOE} \subset \mathbb{R}^k$. An ensemble trees learner ϕ of K trees is defined as :

$$\mathcal{S}(\mathbf{x}) = \phi(\mathbf{x}) = \sum_{k=1}^{K} f_k(\mathbf{x}), \quad f_k \in \mathbf{F} \tag{3.4}$$

With $\mathbf{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ being the space of regression. The function q represents the structure of the tree (mapping the input to the T leaves) and w corresponds to the weight attributed to each leaf. The final prediction for a given input is given by summing up the weights obtained from the corresponding leaves. There are no global weights assigned to the k trees so that we learn the structures of the trees instead of learning weights in $\mathbb{R}^k$. The objective function is the following :

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{3.5}$$

where $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$ is the regularisation term. It is used to penalise complex trees and prevent overfitting. The function l is a convex and differentiable function measuring the error. For example l could be the square loss $l = (\hat{y}_i - y_i)^2$. Since the objective function contains functions as parameters, it cannot be optimised using regular techniques. Instead, it is minimised iteratively by adding a tree at each iteration. At iteration t, the new tree f_t is added and eq 3.6 is optimised to get the f_t that minimises the model loss.

$$L^{(t)} = \sum_i l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x_i})) + \Omega(f_t) \tag{3.6}$$

In the general setting, the second order Taylor approximation is used to speed up the optimisation :

$$\tilde{L}^{(t)} = \sum_i \left(l(y_i, \hat{y}_i{}^{(t-1)}) + g_i f_t(\mathbf{x_i}) + \frac{1}{2} h_i f_t(\mathbf{x_i})^2 \right) + \Omega(f_t) \tag{3.7}$$

with $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$. The first term does not depend on f_t so it can be removed from the optimisation. We then get the final objective function with Ω expanded :

$$\tilde{L}^{(t)} = \sum_i \left(g_i f_t(\mathbf{x_i}) + \frac{1}{2} h_i f_t^2(\mathbf{x_i}) \right) + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \tag{3.8}$$

Instead of deriving the objective, given the discrete nature of the tree structure, a simpler approach is to greedily grow trees and choose the structure that minimise 3.8. To do so, the objective function is reorganised in term of leaves : Let's define $I_j = \{i | q(\mathbf{x_i}) = j\}$ as the instance set of leaf j. We can rewrite equation 3.8 as :

$$\tilde{L}^{(t)} = \sum_{j=1}^{T} \left((\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right) + \gamma T \tag{3.9}$$

Since equation 3.9 is a sum of T independent quadratic functions, we can compute the optimal weight of leaf j with :

$$w_{j^*} = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{3.10}$$

and the corresponding optimal value by :

$$\tilde{L}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{3.11}$$

This value can be used to evaluate the quality of the tree structure q. However, there are potentially an infinite number of tree configurations so that in practice, a greedy algorithm, starting from a zero depth tree and iteratively testing new branches is used. At each leaf node different splits are tried and ranked given the loss reduction that they provide. Let I_L and I_R be the instance set of the left and right leaf after the split and $I = I_R \cup I_L$, the loss reduction is given by :

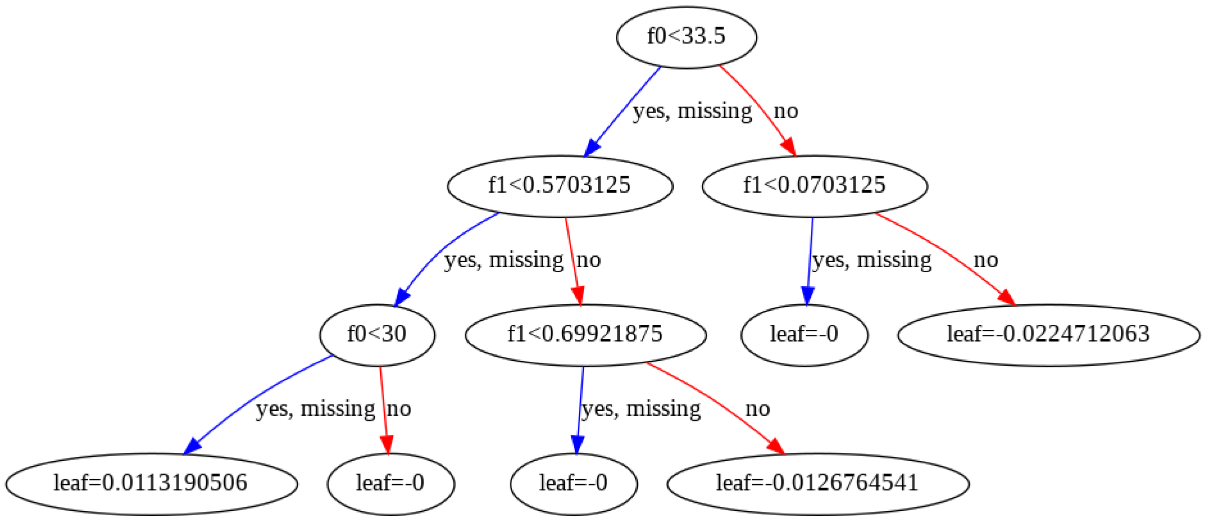
$$L_{split} = \frac{1}{2}\left(\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} + \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right) - \gamma \tag{3.12}$$

The split finding procedure is summarised in algorithm 1.

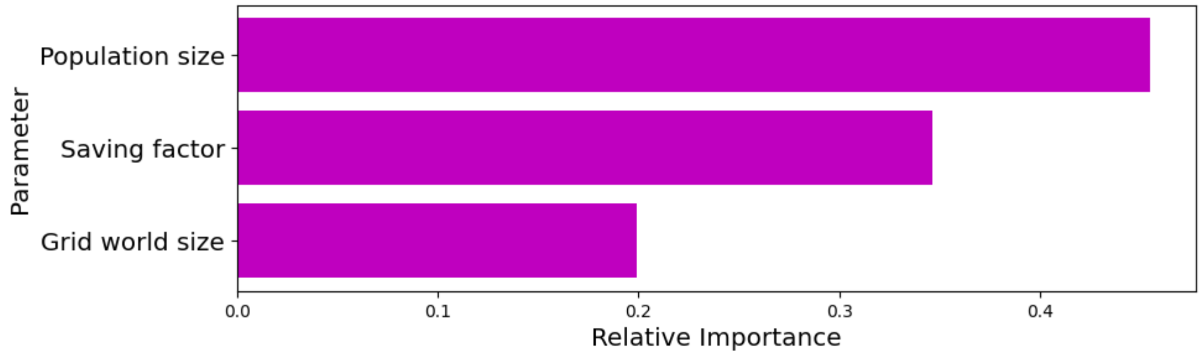
The final algorithm goes as follows : At each iteration, a tree is added. When a tree is added, g_i and h_i are first calculated. Then the split finding algorithm described here-above is used to greedily construct a tree $f_t(\mathbf{x})$ with optimal splitting. The tree is then added to the ensemble so that at the end, we get a predictor of the form 3.4. Usually, instead of simply adding the tree to the ensemble, the three is multiplied by a shrinkage factor before summation so that $\phi^{(t)}(\mathbf{x_i}) = \phi^{(t-1)}(\mathbf{x_i}) + \epsilon f_t(\mathbf{x_1})$. This means that no full optimisation is performed at each step and chance for future rounds is preserved. It helps prevent overfitting. At the end of the training procedure, the XGboost is composed of an ensemble of CART trees and the predictions are achieved using equation 3.4 by summing up the results of each CART tree. An example of a CART tree obtained by training an XGBoost surrogate on the Boltzmann wealth model with saving is shown in figure 3.1a. The relative importance of each parameter calculated by the XGboost algorithm is shown in figure 3.1b.

Algorithm 1 Split finding algorithm

```
for each leaf node do
    for each features do
        Sort all the instances by their feature value;
        Perform linear scan to calculate the best split along the feature using equation 3.12;
    end for
    Keep the split of the feature giving the best loss reduction
end for
Split the leaf having the best loss reduction on the corresponding feature
```



(a) A cart tree example for the Boltzmann wealth model with saving, f0 represent the population size, f1 the saving factor and f2 the grid world size.



(b) Relative importance of each feature for the Boltzmann wealth model with saving calculated by the XGBoost algorithm.

Figure 3.1: XGBoost Cart tree and parameter importance

3.3.2 The Kriging method

Kriging (also named Gaussian Process regression in the Machine Learning community) is a mathematical interpolation method introduced by the South African mining-engineer Krige and later formalised by the French mathematician Matheron. Kriging method is widely used in the field of spatial statistics. The approach is to formulate the surrogate function as the sum a of a polynomial term expression $\mu(\mathbf{x})$ and a deviation from that polynomial $Z(\mathbf{x})$. The deviation has the form of a stochastic process.

Let's denote the Kriging model Y:

$$\mathcal{S}(\mathbf{x}) = Y(\mathbf{x}) = \mu(x) + Z(\mathbf{x}) \tag{3.13}$$

$\mu(\mathbf{x})$ is the regression component. In the most general case μ is a low-order polynomial with l defining the degree of that polynomial.

$$\mu : \mathbf{x} = (x_1, ...x_k) \in D \subset \mathbb{R}^k \to \mu(\mathbf{x}) \equiv \sum_{j=1}^{l} \beta_j f_j(\mathbf{x}), \quad l > 0 \tag{3.14}$$

In this case we speak of Universal Kriging (UK). However, because $Z(\mathbf{x})$ is often able to approach well the distribution of the function considered, the degree of the polynomial $\mu(\mathbf{x})$ does not significantly affect the fit quality [37]. A common simplification, that gives comparable results, is to consider μ as a constantv$\mu(\mathbf{x}) = \mu$. In this case, the method is often referred as Ordinary Kriging (OK).

$Z(\mathbf{x})$ is a stochastic Gaussian process. It is often assumed to be second-order stationary so that it has zero mean, constant variance and correlation function R. The correlation is given by $C : (u, v) \in D^2 \to \sigma^2 R(u, v)$, with R being a $n \times n$ matrix and σ^2 a scale parameter. We observe here the difference with linear regression where the error is assumed as constant. In contrast the Gaussian process approach of Kriging suggests that the errors are codependent. More especially, Kriging considers that the closer the experimental points are, the higher the correlation between their output values will be. We observe here the analogy with spacial statistics, where two points situated close to each other on a map are likely to have resembling properties (think about the elevation for example).

Let $\mathbf{x}$ and $\mathbf{x}^{'}$ denote two points of the input space. The distance between the points for the parameter j $|x_j - x_j^{'}|$ is denoted h_j. As previously noted, in Kriging method, the correlation between the realisation of two input vectors is proportional to their distance. Different functions with such properties can be chosen (For Gaussian Process Regression we speak about the kernel of the regression[1]):

- $R(x, x^{'}) = exp(-\sum_{g=1}^{k} \theta_g h_g)$: Exponential correlation
- $R(x, x^{'}) = exp(-\sum_{g=1}^{k} \theta_g h_g^2)$: Gaussian correlation
- $R(x, x^{'}) = max(1 - \theta_j)$: Linear correlation

θ is a $1 \times k$ vector $\theta = \{\theta_1, ...\theta_k\}$ ranking the importance of each input dimension g=1, ...k in the correlation function calculation. Indeed, the higher the θ, the less influence dimension g has in the correlation. We also observe that the higher the distance in g, the smaller the correlation. This is coherent with the spacial statistic approach where two close points are more likely to be correlated.

The estimation of the model described in equation 3.13 requires to learn different parameters. In the case of Universal Kriging, the first parameters to estimate are the l

[1]see [39] for a review of the different kernels.

coefficients β of the polynomial term (equation 3.14). Then, the vector $\theta = \{\theta_1, ...\theta_k\}$ and the scale parameter σ^2 are estimated.

For ordinary Kriging parameters μ,θ, and σ^2 can be estimated directly by maximum log-likelihood of the observed data $\mathbf{y} = (y_1, ...y_n)$. The likelihood function for a Gaussian correlation is expressed as :

$$L(\mathbf{x}, \sigma, \mu) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}|R|^{\frac{1}{2}}} exp\left(-\frac{(\mathbf{y}-\mathbf{1}\mu)^T R^{-1}(\mathbf{y}-\mathbf{1}\mu)}{2\sigma^2}\right) \tag{3.15}$$

with $\mathbf{y}$ being the $1 \times n$ vector of the observed data, $\mathbf{1}$ is the $1 \times n$ all-ones vector and n is the number of observations. The optimal values for μ and σ^2 are obtained by differentiating the log of equation 3.15 with respect to μ and σ^2 and equating the result to zero. We obtain the expression of the optimal value $\hat{\mu}$ and $\hat{\sigma}^2$.

$$\hat{\mu} = \frac{\mathbf{1}^T R^1 y}{1^T R^{-1}\mathbf{1}} \tag{3.16}$$

$$\hat{\sigma}^2 = \frac{(\mathbf{y}-\mathbf{1}\mu)^T R^{-1}(\mathbf{y}-\mathbf{1}\mu)}{n} \tag{3.17}$$

By replacing the value obtained from equation 3.16 and 3.17 in equation 3.15 we obtain a concentrated likelihood function which depends only on R :

$$L(\mathbf{x}) = -\frac{n}{2}log(\hat{\sigma}^2) - \frac{1}{2}log(|R|) \tag{3.18}$$

To get the numerical estimation of μ, σ^2 and θ, we begin by maximising the equation 3.18 to get the optimal weight values $\hat{\theta}$. Then the matrix R is computed and used to get the optimal value of μ and σ^2 using equation 3.16 and 3.17.

Once the parameters are estimated, the procedure to predict an output value $\hat{y}_{pred}$ for a new input point $\mathbf{x}_{pred}$ is as follows : first the point $(\mathbf{x}_{pred}, y_{pred})$ is added to the expression of the likelihood (equation 3.15). As each parameter has already been estimated, the expression depends only on y_{pred}. We obtain the estimation of y_{pred} by maximising the log-likelihood. After some algebra, the expression of the estimation of y_{pred} can be expressed as:

$$\hat{y}_{pred}(\mathbf{x_{pred}}) = \hat{\mu} + \mathbf{r}^T R^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{3.19}$$

With $\mathbf{r}$ being the vector of correlations between the new point and every other points of the Dataset : $R(\mathbf{x}_{pred}, \mathbf{x}_i)$ $\forall i = 1, ..n$.

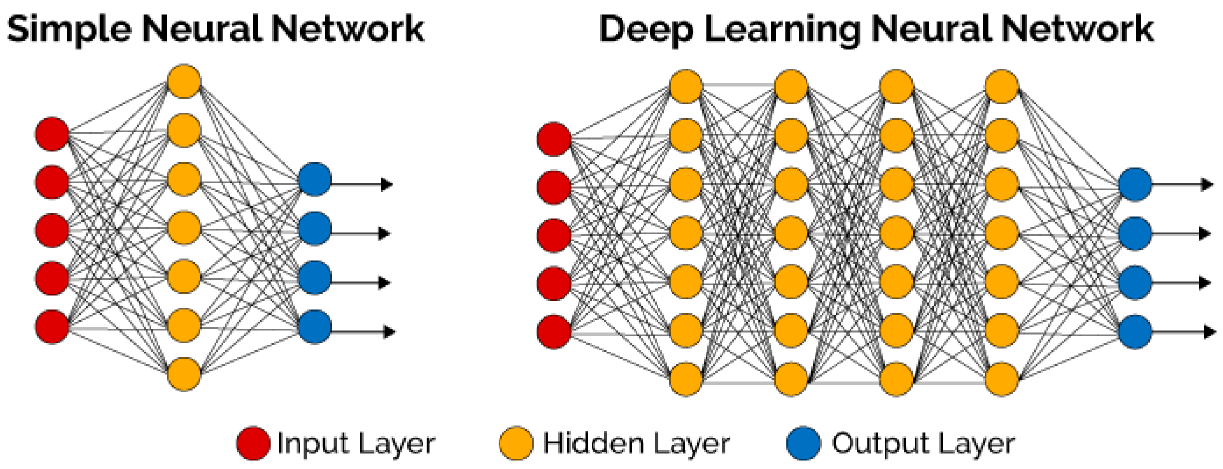


Figure 3.2: Neural Network and Deep Neural Network, source : https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351

3.3.3 Neural Networks

Neural networks are a set of different algorithms, named after the fact that they are modelled loosely after the human brain. They are designed primarily for pattern recognition. They have been around for a long time and they are still one of the most popular machine learning methods, especially for image processing. Their versatility and the high modelling freedom that they provide have led them to be used in several surrogate problems. [44, 25] uses a DNN surrogate for uncertainty quantification in stochastic elliptic partial differential equations. [50] uses a DNN surrogate to speed up the Bayesian inverse problem. Finally, [34] uses a DNN surrogate to optimise the geological CO_2 storage operation. However, to our knowledge, no surrogate modelling exercise using a DNN model has been performed on Agent Based Models.

Neural Networks (NN) approximate an input/output mapping $f : \mathbb{R}^k \to \mathbb{R}^m$ through a hierarchical ensemble of abstract layers containing latent variables. The most typical example is the MLP (MultiLayer Perceptron) which is a feed-forward neural network consisting in an input layer, an output layer and a fixed number of hidden layers. Using Neural Networks for surrogates modelling presents two advantages. First, Neural Networks, but more generally, MLP, are universal approximators. Indeed, they can approximate arbitrary well any real-valued continuous function on a compact subset of $\mathbb{R}^n$. It means that under certain assumptions regarding the function being approximated, an MLP with simple structure (containing at least one hidden layer) can approximate arbitrary well complex functions. The second advantage is that Neural Networks provide a natural dimension reduction from a high dimensional input space to a smaller exploitable latent space through a succession of non-linear projections (at each layer). Neural Networks are hierarchical, each step in the hierarchy is represented by a layer. The first layer is known as the input layer and the last layer as the output layer. The intermediate layers are known as hidden layers. While the size of the input an output layers are generally determined by the dimensionality of the function to approximate, the number and the size of the hidden layers is left to the choice of the modeller. Selecting the right network structure is a challenging task and requires a great amount of hyper-parameters optimisation and cross-validation. Deep Neural Networks are loosely defined as a generalisation of Neural Networks containing a high number of hidden layers. In figure 3.2, the architecture of a DNN with four hidden layers is presented. It projects a k-dimensional input vector $\mathbf{x} \in D \subset \mathbb{R}^k$ to a m-dimensional vector $\mathbf{y} \in O \subset \mathbb{R}^m$. The output of a layer is defined as the activation and is used as input in the next layer of the network. Each neuron calculates the weighted sum of the input vector it receives plus the bias. It then applies a non-linear function σ to produce the output. The output of the j^{th} layer is shown here-above :

$$\mathbf{x}^j = \sigma(W^(j)\mathbf{x}^{(j-1)} + b^{(j)}, \forall j = 1, ..L \tag{3.20}$$

Where $W^{(j)} \in \mathbb{R}^{d_j \times d_{j-1}}$ is the weights matrix, d_j is the number of neurons in layer j and $b^j \in \mathbb{R}^d$ is the bias vector. Note that $\curvearrowleft_0$ is the input vector of the network so that d_0 is the dimensionality of the input of the ABM. A lot of different activation functions have been proposed in the literature. Among them we find hyperbolic tangent, rectified linear unit (ReLu), sigmoid, logistic function[2]. Following [44] and [50] we will use the Swish activation function [38] defined as :

$$\sigma(x) = \frac{x}{1 + exp(-\gamma x)} \tag{3.21}$$

Where gamma can either be learned as hyperparameter through hyperparameters optimisation or a constant.

Once the hyperparameters of the network are fixed, the weights and bias are the only parameters of the network $\theta \equiv \{W^{(j)}, b^{(j)}\}$. They can be found using some kind of optimisation based on the training data $D \equiv \{\mathbf{x_i}, y_i\}$, $i = 1, ...N$. More especially, a certain error metric representing the discrepancy between the true value y_i and the value predicted by the DNN, $\hat{y_i} = DNN(\mathbf{x_i})$ is minimised. As DNNs are prone to overfitting, some regularisation is used to constrain the value of the weights and the sparsity of the estimator. Here a combination of L_1 and L_2 regularisation known as elastic net regularisation is used [52]. The final error measure is defined by :

$$L(\theta; \lambda) = \frac{1}{N}\sum_{i=1}^{N} \|y_i - DNN(\mathbf{x_i}; \theta)\|^2 + \lambda \sum_{i=1}^{L+1} \left(\|W^{(i)}\|_2^2 + \|W^{(i)}\|_1\right) \tag{3.22}$$

[2]See [33] for a comparison of the trend in practice and research for different activation functions

In practice, the optimisation is not performed on the all training Dataset at once but is iteratively achieved over a small random subset $D_M \subset D$ called batch.

$$\theta^* = argmin_\theta \frac{1}{N} \sum_{(\mathbf{x_i}, y_i) \in D_M} L(\theta; \mathbf{x_i}, y_i) \tag{3.23}$$

Solving such an optimisation problem is usually achieved using the stochastic gradient descent algorithm (SGD). The estimation of the gradient $\nabla_\theta L(\theta; D_M)$ at step k is calculated through back-propagation. Then, a small negative step is taken in the opposite direction giving :

$$\theta_{k+1} \leftarrow \theta_k - \epsilon \nabla_\theta L(\theta; D_M) \tag{3.24}$$

Where ϵ is the learning rate. Practically, several solvers are available to perform this kind of task, here we will use the famous ADAM solver [27]. For our network topology, we follow the idea developed in [44] to design the network as the concatenation of an encoding network performing dimensionality reduction with an expanding layer preceding the output layer. The encoding part starts from the input layer of dimension k and projects it to a final layer of dimension d (latent dimension), with $d < k$. The projection is achieved through an arbitrary number of hidden layers. This number could be found using hyperparameters optimisation but here we choose to fix it to three. The number of nodes in each hidden layer is fixed in order to have a linear decline from the input layer to the latent layer. The expanding layer has a width much larger than d, typically it contains a certain multiple γ of d for example $L = 100d$ nodes. The factor γ and the number of nodes in the latent space are fixed by hyperparameters optimisation such as expressed in table 3.1. A visualisation of the network topology is provided in figure 3.3.

Table 3.1: Hyperparameters selection

Parameter	description	chosen interval
α	Multiplication factor for the expanding layer	$[50 - 150]$
d	Number of nodes in the last layer (latent space)	$[1 - 5]$
λ_1	L1 regularisation term	$[0, 0.1]$
λ_2	L2 regularisation term	$[0, 0.1]$

3.4 Sampling methods

Surrogate modelling performance will highly depend on the chosen sampling procedure. An obvious solution would be a random sampling, generated by a pseudo-random number generator such as those provided in every scientific programming language. Random samples x are drawn by generating a value for each variable $j = 1, ...k$ from a given

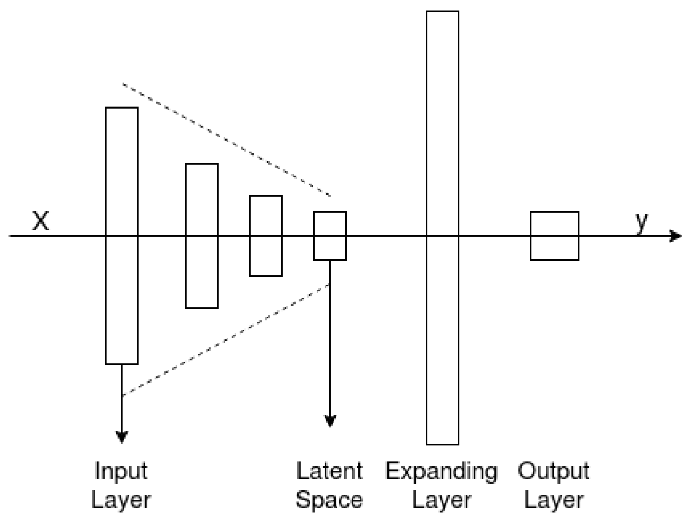


Figure 3.3: Topology of the Deep Neural Network

distribution, for example a uniform distribution U(0,1). Each value is then rescaled into the interval $[min_j, max_j]$ given by the modeller. A disadvantage of this method is that it can introduce clusters in certain regions, while leaving other regions empty. If the number of samples used is high, random sampling can achieve good performances (remember the convergence of Monte-Carlo method with infinite number of samples). On the other hand, if the evaluation of a sample requires high computational cost, the amount of samples used for model generation becomes a modelling constraint. In this case, a sampling method with better space filling properties can enhance the modelling performance while keeping the computational budget low.

Two good candidates possessing such properties are Latin Hypercube Sampling (LHS) and sampling based on Sobol sequences[3].

3.4.1 Latin Hypercube sampling

Latin Hypercube is a particular case of stratified sampling. The principle is to stratify each dimension of the parameter space into N intervals of same probability $\frac{1}{N}$. If there is no prior distribution attached to the variables, this amounts to dividing the dimensions into N intervals of same sizes. Then, N combinations are drawn randomly from the $N \times N$ hypercubes . These combinations are generated such that each sample is the only one in each axis-aligned hyperplane containing it. An example in two dimensions is shown in figure 3.4 where three samples are drawn. We see that there are three combinations of Latin squares : A,B and C. Each of these combinations has only one sample in each row and in each column. One of these combinations is then randomly chosen and a sample

[3]See [9] for a comparison of LHS and Sobol sampling with random sampling

situated inside the square is chosen for each of the squares (for example the centre of the square). In [41, 5], the use of Nearly Orthogonal Latin Sampling formulation of Cioppa [14] is suggested. Indeed, they argue that beside of the good space filling properties guaranteed by the Latin squares, the little or absent correlation among the columns guaranteed by the near-orthogonality provides the model with a sampled Dataset representing better the variety of the original Dataset.

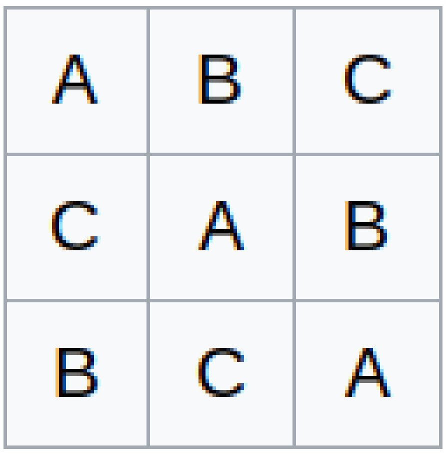


Figure 3.4: Latin squares

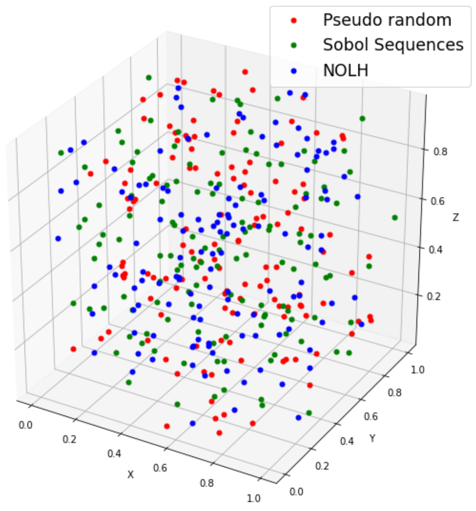


Figure 3.5: Comparison of the different sampling techniques with N=128 and k=3. In red, the points generated with a pseudo random generator, in blue, the points generated with the LHS method, in green, the points generated with the Sobol method.

3.4.2 Sobol Sequence sampling

This sampling method is based on the Sobol sequences introduced by the Russian mathematician Ilya M. Sobol. Sobol quasi-random sequences are designed to minimise the discrepancy. The discrepancy can be defined loosely as a measurement of the difference between the highest and the lowest density of points in a sequence. A high discrepancy means that either there is a small portion of the sequence presenting a high density or a large portion presenting a low density (or both). The discrepancy tends to zero when the sequence is equidistibuted. Sobol sequence properties require a sample size that is a power of two (2^k with $k \in \mathbb{N}_0^+$). Both Sobol and LHS sampling alongside with pseudo-random sampling are illustrated in figure 3.5. For clarity sake, the projection onto each 2D plan is presented in figure 3.6. We observe rather clearly that the Sobol and NOLH method present a lower discrepancy than the random method. Random sampling has been generated with the Numpy pseudo-random generator[4], Sobol sequence with the sobol-seq python package[5] and NOLH sampling with the pynolh package[6].

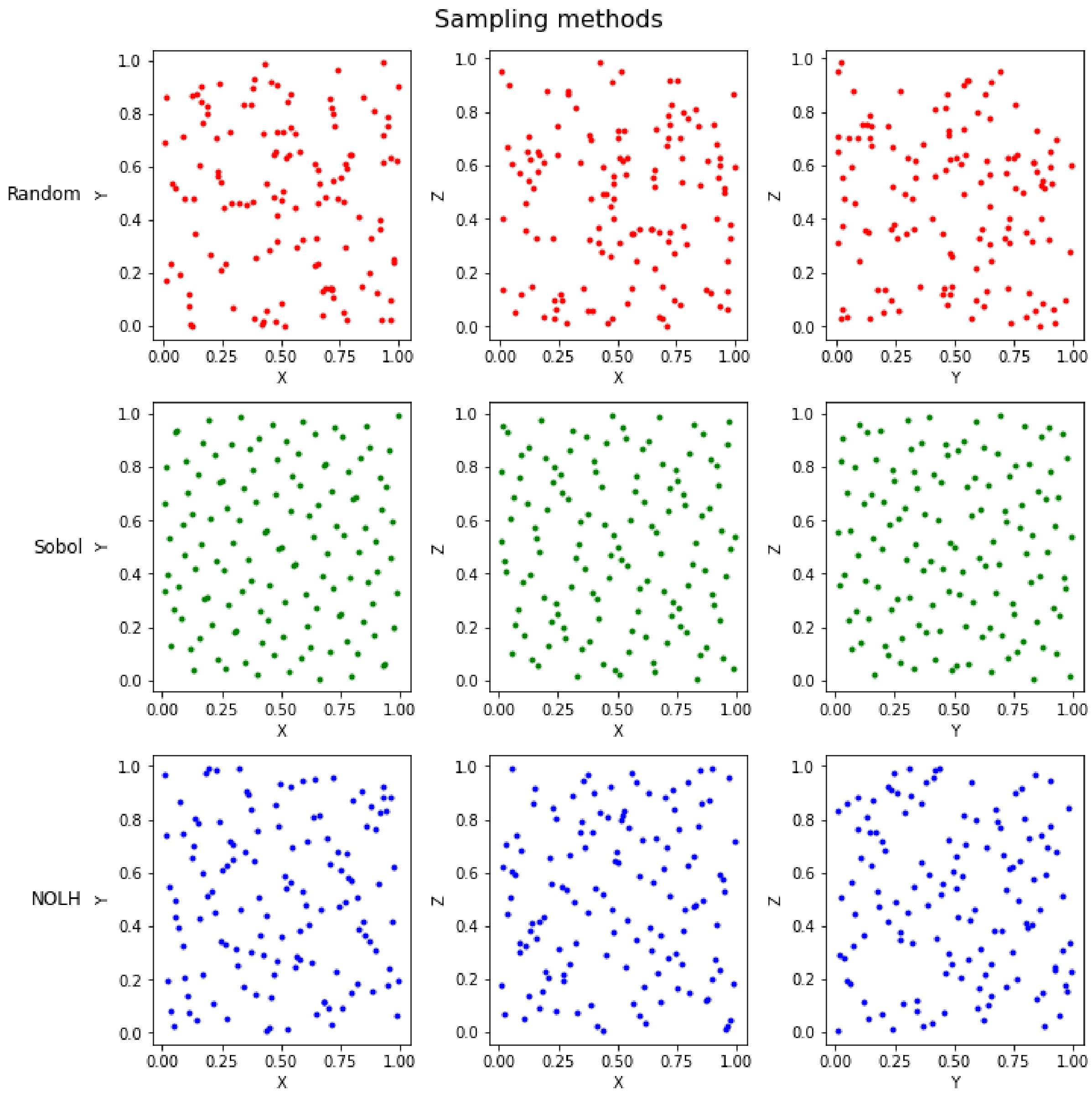


Figure 3.6: Comparison of the different sampling techniques with N=128 and k=3. In red, the points generated with a pseudo random generator, in blue, the points generated with the LHS method, in green, the points generated with the Sobol method.

[4]https://docs.scipy.org/doc/numpy-1.14.0/reference/routines.random.html
[5]https://github.com/naught101/sobol$_{s}eq$
[6]https://github.com/fmder/pynolh

3.5 Performance metrics

To evaluate the performance of a given surrogate on a set of training data, one should choose an error measurement metric. This error function will be used as the optimisation target when training a given method. Two choices are to be made: The form of the function and the target variable on which the function is evaluated. The choice of the target variable (or variables) has important consequences on the modelling. Indeed, as highlighted before, generally speaking an ABM has a broad set of observable output values. Even if micro-level variables are aggregated into macro-level variables, it is difficult and often not desirable to use every variables of the output set in the error calculation. Two different modelling choices are available here. The first one is a direct approach where the final value of one or several aggregate variables is used as the target variable for the surrogate optimisation. The second approach is what we call a calibrated error measure. In this approach, the surrogate takes into account the mapping from parameters into observable variables in terms of a fitness response that measures the distance of the data produced by the model to empirical data [5]. The distance is calculated between the temporal evolution of a given variable estimated by the system and its empirical counterpart.

For example, in an asset pricing estimation, the model tries to predict the evolution of the price of an asset. An empirical time series that can be used is the return (or log-return) of a given trading index (S&P 500 for example). The chosen term *Calibrated error* comes from the fact that, in this setting, the surrogate is trained to predict the distance of the model from empirical data. Minimising the error comes down to ≪bring the model closer to data≫, which is the typical task of calibration. The metric that will be used in the rest of this thesis is the mean-square error (MSE) in regression setting and the F_1 score in classification setting.

3.5.1 Direct error measure

For real-valued estimation with scalar output, the mean-square error (MSE) is defined as :

$$MSE(\hat{s}, D) = \frac{1}{N}\sum_{x_i \in D}(y_i - \hat{S}(\mathbf{x_i}))^2 \tag{3.25}$$

Where y_i is the value of a given output variable of the ABM and $\hat{S}(x_i)$ is the corresponding prediction of the surrogate. If we want to observe more than one variable in the output (the output is a vector of dimension $1 \times M$) we can use an averaged error :

$$MSE(\hat{s}, D) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} \mu_j (y_{ij} - \hat{S}_j(\mathbf{x_i}))^2 \tag{3.26}$$

where μ_j is the weight assigned to the j^{th} component of the vectorial output.

3.5.2 Calibrated error measure

The direct method remains at the surface in the sense that it trains a model to approximate the value of the considered macro variable at the end of the ABM simulation. The dynamic of the variable during run-time is totally hidden from the surrogate. Using a calibrated approach allows to compare the dynamic of the data generation process with the actual dynamic observed in empirical data.

Numerous tools to compare time-series are available from simple statistical tests to more elaborated methods. In [30], a two samples Kolmogorov-Smirnov (KS) test is run between the simulated log-returns and the empirical log-return on the SP 500 index. The KS is used to decide whether two samples are likely to come from the same distribution. Another simple approach is to use the sum of weighted squared difference between both time series values as proposed in [43]. These two approaches have the advantage to be well documented and easy to implement. However, they do not take into account the structure of the time-series and the conditional probabilities associated with each transition from one time frame to another. More sophisticated methods have appeared in the literature to tackle this problem. [4] develops an information criterion called Markov Information Criterion (MIC), based on the measurement of the cross entropy between a model data and its empirical counterpart. Another direction is proposed in [24] where autoregression (VAR) is used to compare the causal mechanisms driving the time-series evolution of both simulated and empirical data. Finally,[29] introduce a general theoretic criterion using a simple function of the L-divergence computed at different block lengths (GSL-div). These different criteria are relatively new and rely on different principles. Their lack of extensive benchmarking and mathematical development make them difficult to compare and make the choice of one criterion rather arbitrary. Here, we will use both the KS-test and the GSL-div method, the choice of the GSL-div method is mainly driven by the availability of a full python implementation[7].

In the section above the principles of the GSL-div distance metric are briefly introduced. In figure 3.7, the surrogate response surface of the Boltzmann wealth model is shown for the three surrogate models introduced. The surfaces are calculated using the direct error

[7]https://github.com/topics/gsl-div

approach (first row), the two sample KS test (second row) and the GSL divergence (third row).

The GSL-div

As the MIC approach, the GSL-div relies on information theory techniques rather than purely statistic techniques. The time-series that best match their empirical equivalent are those reproducing the highest possible portion of the empirical series in the most compact way. To measure this, one can resort to the Kullback-Leibner divergence which measures the inefficiency of using distribution q when real distribution is p :

$$D_{KL}(p||q) = \sum_{s \in S} p(s) log \left(\frac{p(s)}{q(s)} \right) \tag{3.27}$$

To make the distance symetric, the L-divergence is used in practise :

$$D_L(p||q) = D_{KL}(p||m) + D_{KL}(q||m) \tag{3.28}$$

with $m = \frac{p+q}{2}$ is the mean probability function. The L-divergence is used as the building block of the method. In practise, both series are symbolised and subdivided in time-windows of different lengths. Symbolisation means that the value of a time series in a given block can take only a finite set of value. In concrete terms, the real interval formed by the limiting value of the time-series $[x_{min}, x_{max}]$ is divided into b intervals with b being the resolution. The time series is subdivided into blocks of equal lengths l iteratively for $l = 1, ..., L$ with L being the maximal frame length considered. The alphabet S defines the set of all possible combinations of the b natural numbers taken l times.

$$S_{l,b} = \binom{A_b}{l} \qquad A_b = \{1, 2, ..., b\} \tag{3.29}$$

After subdivision, the frequencies of each symbol in the time-series are calculated. Let $x_s(t)$ and $y_s(t)$ be two symbolised time-series and $\mathbf{f}(s)$ and $\mathbf{f}^{'}(s)$ the frequency of occurrence of symbols s in the time-series. The GSL-div is the sum of subtracted L-divergence multiplied by weight w_i for each value of l :

$$D_{GSL}(f||f^{'}) = \sum_{i}^{L} w_i \left(-2 \sum_{s \in S_i} m_i(s) log_{a_i} m_i(s) + \sum_{s \in S_i} f_i^{'}(s) log_{a_i} f_i^{'}(s) \right) \tag{3.30}$$

with $m_i = \frac{f_i(s)+f_i^{'}(s)}{2}$.

The GSL divergence is a parametric measure with the three parameters being the vector of weights $\mathbf{w} = (w_i, ...w_L)$, the resolution b, and the maximum frame length L. However Lamperti found out that the measure is rather robust with respect to these parameters when large amounts of data are available[8].

[8]See [29] for an extensive exploration of the parameters of the GSL-Div

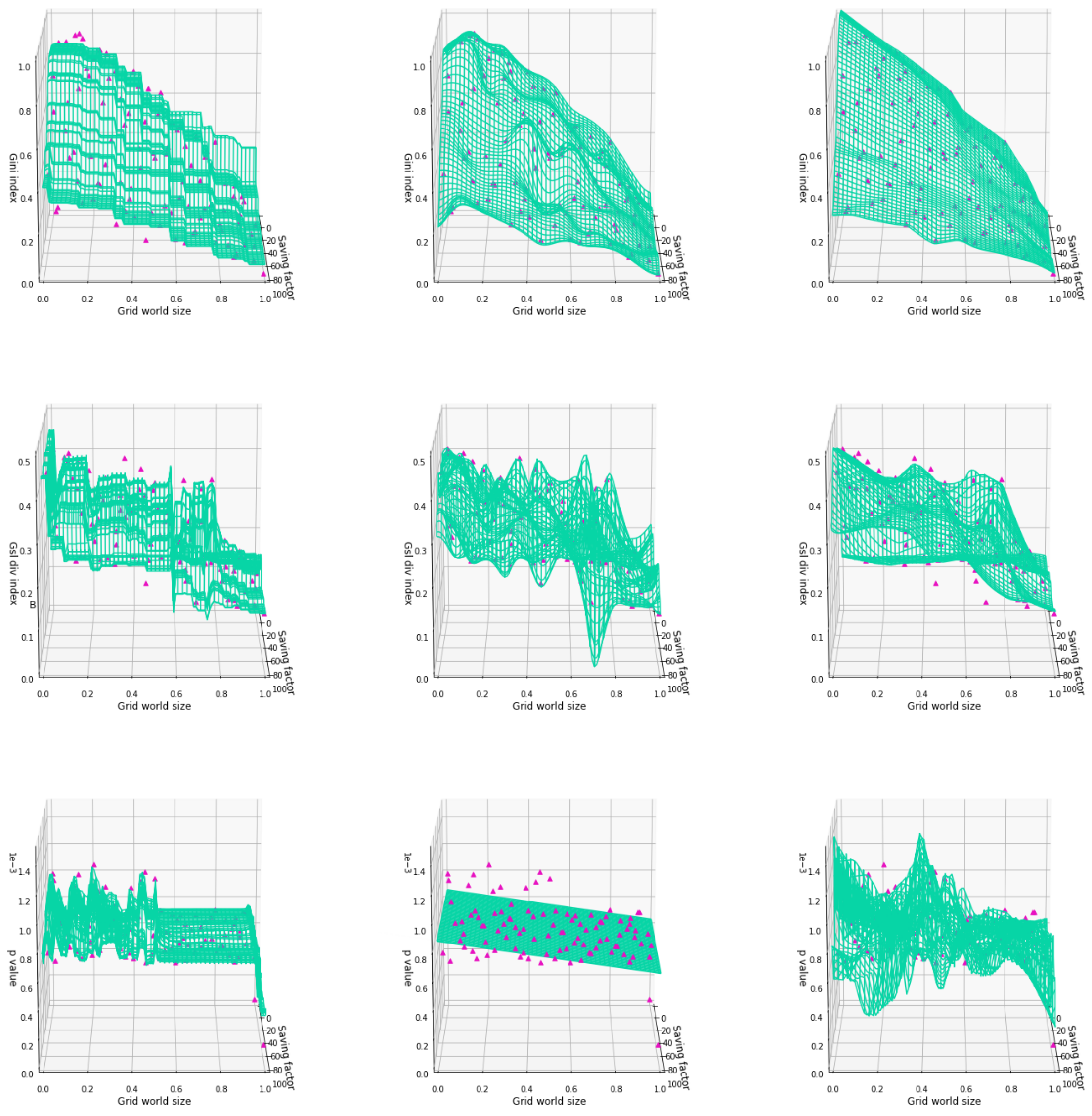


Figure 3.7: The three surrogate model fitness surfaces trained on 100 data points stemming from the Boltzmann wealth model with saving. Each data point is averaged over 5 Monte Carlo runs. The first row is trained with a direct error measure on the Gini coefficient. The second row is trained with the GSL-Div distance calculated between the Gini coefficient obtained during the last 20 time iterations of the model and the Gini coefficient of Belgium from 1998 to 2017. The last row is trained with the p-value of the two sample KS test between the two aforementioned time-series. The first column represents the fitness surfaces obtained with the XGboost model, the second with Kriging method and the last with Deep Neural Network. Each model for each plot is obtained by training on the Data and optimising the hyperparameters with cross validation so that in a same column, the model topology can differ. We observe, for example, that the Kriging method chose a Matern kernel for predicting both the Gini and the GSL-Div but chose a Dot Product kernel for predicting the p-value. We observe similarities between the p-value and the GSL-Div measure. Indeed, the points presenting the highest fitness are located in the top left corner with small values of the saving factor for both measures.

3.6 Hyperparameters optimisation

Each of the three models described above contains a certain amount of hyperparameters. These hyperparameters can be tuned manually by exploiting a certain prior knowledge about the model or by using certain heuristics or best-practises exposed in the literature. However, in the context of ABM meta-modelling, the complexity of the underlying distribution makes is difficult to use prior information to select hyperparameters. One should use more formal hyperparameters optimisation techniques (HPO).

We denote $h \in \mathbf{H}$, a given combination of hyperparameters. The aim of the HPO is to minimise the error :

$$R(h) = \sum_{i=1}^{N} (y_i - \hat{s}(x_i, \theta(h))^2 \tag{3.31}$$

Where $\theta(h)$ denotes the structure of the model learnt on the training Dataset (x_i, y_i), $i = 1, ..., N$ with hyperparameters combination h. $R(h)$ can be calculated on a specific validation Dataset D_{val}, or using cross-validation. Here the cross validation approach is chosen and we denote the obtained error $R_{cv}(h)$. A naive technique would be to sample the hyperparameters space $\mathbf{H}$ and to extensively try each combination and keep the one minimising $R_{cv}(h)$. However, calculating the cross-validation score is a computational expensive process so that running it for all $h \in \mathbf{H}$ can be infeasible in practise. An approach to speed up the process is to search a near-optimal parameter combination by iteratively using the performances of the last tested combination h to select a new candidate. This is the Bayesian global optimisation approach (BGO).

We are interested in solving the following stochastic global optimisation problem :

$$h^* = argmin_h \mathbb{E}[R_{cv}(h)] \tag{3.32}$$

Where the objective function R_{cv} is a black box : it has no closed form and its gradients are not available. Furthermore, its outputs are noisy. Bayesian Optimisation builds a Gaussian Process surrogate response surface of the objective function and uses it to select the next best candidate to evaluate in the true objective function. At each iteration, a new pair $(h, R_{cv}(h))$ is generated by maximising an acquisition function on the surrogate response surface. A popular choice for the acquisition function is the Expected-Improvement (EI).

$$EI(h) = \mathbb{E}[max(R_{cv}(h) - R_{cv}(h^*), 0)] \tag{3.33}$$

where h^* is the best hyperparameters combination found so far.

Hyper-parameters considered for optimisation for each of the three surrogate models are listed in tab 3.2. The parameter ranges for each parameter are indicated in the last column.

Table 3.2: Hyperparameters candidates for HPO of the three considered models

Model	Parameter	Description	Value Range
XGBoost	T	Max number of trees	[100,1000]
	ϵ	Learning rate	[0.01,1]
	depth	Max depth of the trees	[10,1000]
	λ	L2 regularization term on weights	[0,1]
	α	L1 regularization term on weights	[0,1]
	subsample	subsample ratio of the training instance	[0,0.25]
Kriging	Kernel	Kernel of the correlation function	[Matern,Dot Product, Gaussian, RBF]
Neural Network	ϵ	Learning rate	[1e-4,1e-2]
	α	Multiplication factor for the expanding layer	[30-100]
	d	Number of node in the last layer (latent space)	[10-30]
	λ_1	L1 regularization term	[0,0.1]
	$\lambda - 2$	L2 regularization term	[0,0.1]

3.7 Training procedure

When a fitness landscape measured against empirical data is used as error measure, the positive calibration, that is the ABM simulations producing a distribution close to its empirical counterpart, are likely to form a small subset of the parameter space. Given the desire to avoid a too high number of expensive ABM runs while also identifying positive calibration, it is imperative to have good performance of the surrogates on these calibrations. Different approaches to solve this problem are available. [46] reviews several methods to optimise the performance of surrogate models on the region of interest. Among them, the use of multiple surrogates looks promising. The idea is to train multiple surrogates on the domain and combining their prediction. The models can cover the whole domain so that their predictions are then averaged to produce the final prediction [1]. Another option is to train a cheap surrogate on the input domain and then construct more precise local surrogate models on the regions identified as promising.

Another way to improve the performance of a surrogate on the manifold of interest is to feed the model with more point stemming from that region comparatively to other less interesting regions. This is called sequential sampling. In a traditional sampling approach, the Dataset is generated before training the model. This is referred as a «one-shot»approach[15] where the parameter combination set is specified before simulations and the Dataset is used post-mortem to derive the surrogate. In contrast, the approach of sequential sampling makes use of a first trained surrogate to identify new promising candidates. Practically, after generation of a first surrogate, new candidates are selected and added to the dataset to evaluate a new surrogate, and so on, until a certain stopping criterion is met. This approach allows to iteratively strengthen the surrogates on the manifold of interest by feeding it with parameter combinations identified as good candidates to yield positive calibration. In the jargon of machine learning, this is referred to as Online Learning (OL) or Online Optimisation (OO). In Online Learning, the quality of new training data can be assessed with the surrogate model without the need to run the computationally heavy ABM for each candidate. The sequential training procedure is depicted in figure 3.8.

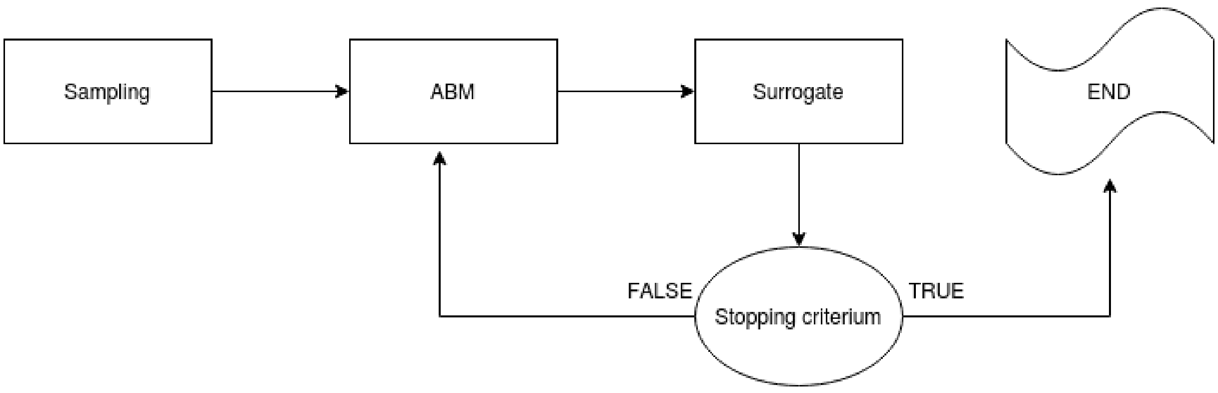


Figure 3.8: Sequential training procedure

Two main design parameters must be chosen : the stopping criterion and the expected performance measure of the samples. The stopping criterion is used to decide when the sequential sampling must be stopped. It can be performance based (The sampling stops when the surrogate achieves a given performance) or resource based (The sampling stops when a certain number of ABM computations has been made, this number is called the budget). We choose a resource based criterion with a given budget B of ABM computations. The expected performance measure of a sample refers to «how much a given sample will enhance the ability of surrogates to approximate well the ABM». Of course, this is a loose definition and many different criteria can be retained. The identification of the local minima of the response is an interesting approach but requires some kind of optimisation on the response surface. An example of such an approach can be found in [5] where they use a constrained optimisation algorithm to find all the local minima of a Kriging response surface. These minima are then identified as promising new sample candidates and are labelled with the original ABM.

However, as we want our training procedure to be used with different surrogate models, we cannot use such an optimisation algorithm. For our procedure to be easily used on each of the three models, we develop a method that combines both one-shot approach and in-loop data simulation, following the work of [30]. A Budget B is chosen according to the computational power available. This budget is then separated into two different buckets B_1 and B_2. At initialisation B_1 points are sampled with a good space filling sampling procedure (we use Sobol sampling). Then, the points are labelled using the ABM and the surrogate model is trained on the resulting dataset. After initialisation, a given amount of points is evaluated using the trained surrogate. The points that lead to a predicted positive calibration are then labelled using the ABM and added to the dataset. A new surrogate in then trained on the resulting dataset. The loop runs until the exhaustion of the budget. During the loop, a maximum of B_2 ABM calculations is run.

It is possible that, at a given step, no points resulting in positive calibration are found. In this case, a classifier is trained on the current Dataset and the points that maximise the entropy of the prediction are chosen. The justification fot this approach is that the points with max entropy are the one that the classifier is the most *ensure* about. Adding these points to the Dataset will therefore strengthen the surrogate on the regions of uncertainty.

If we analyse the training procedure through the lens of the exploitation versus exploration dilemma we see that both are encompassed into the procedure. Indeed, the exploration is guaranteed by the first round using a wide Dataset presenting good space-filling properties and by the entropy classifier. On the other hand, the exploitation is achieved by the active sampling feedback loop that exploits the manifold of interest. The training procedure is summarised in algorithm 2.

Algorithm 2 Sequential training algorithm

```
Require:
  Agent based model $ABM$
  Budget $B$
  Surrogate model $S$
  procedure SEQUENTIAL TRAINING(
      )Generate $X_{sobol}$ using Sobol Sampling
      Generate labels $Y_{sobol} = ABM(X_{sobol})$
      Train $S$ on $(X_{sobol}, Y_{sobol})$
      X = $X_{sobol}$
      Y = $Y_{sobol}$
      while $|Y| \leq B$ do
          Generate $X_{test}$ using Random Sampling
          Compute $Y_{test} = S(X_{test})$
          $X_{canditades} = \{x \in X_{test} | S(x) = 1\}$
          if $X_{canditades}$ does not contain enough sample then
              Fit Classifier $C$ on $(X, Y)$
              $X_{candidates} = X_{candidates} \cup \{x \in X_{test} | C(x)$ has max entropy$\}$
          end if
          $Y_{candidates} = ABM(X_{candidates})$
          $X = X \cup X_{candidates}$
          $Y = Y \cup Y_{candidates}$
          Train $S$ on $(X, Y)$
      end while
      Return $S$
  end procedure
```

Chapter 4

Final calibration protocol

In the previous section, we have introduced all the building blocks of a calibration pipeline using surrogate models. We have seen that the computational nature of the Agent-based models allows them to be pipelined in an Online learning procedure. The different building blocks are : the sampling procedure, the Agent-based Model itself, the distance to empirical data calculation, the surrogate model and the hyperparameters optimisation. In table 4.1, we review all the different methods covered in this thesis for each building block.

The final pipeline to derive a calibrated surrogate model is described in figure 4.1. The pipeline takes as input the domain of experiment (X in the figure) and a certain empirical time-series that we desire to match (y_{real}). In the first iteration, a Dataset of size N is generated with a given sampling function (LHS or Sobol). This Dataset is fed to the ABM and produces an output set Y consisting of the temporal evolution of a given macro property. The distance of each time- series $y \in Y$ of the output set to the empirical time-series is calculated so that a label for each y is produced. In the context of a discrete calibration, the label will be binary. For example, if a two samples KS test is performed, a 1 label will be assigned to an output whose p-value given by the KS test with the empirical series is superior to 5%. For a continuous calibration, the label will be real-valued. It can for example be the p-value of the KS test.

When the labels are produced, a first surrogate is generated and its hyper-parameters are optimised using the Bayesian global optimisation. The next step is the generation of new high quality samples for the next iteration. This is done according to the procedure described in section 3.7. Once the new samples are generated, they follow the same path through the ABM and the label maker to finally be added to the training set of the surrogate. The same procedure is repeated until the budget of maximum ABM runs is reached. At the end of this procedure the surrogate is trained to minimise the discrepancy between the ABM output and the surrogates predictions. We end up with a table containing all the input configuration $\mathbf{x}$ considered alongside with their corresponding label y. Good candidates of calibrated parameter combinations can be found by taking the combination $(\mathbf{x}, y)$ that has the highest label values (the highest label value corresponds to the *closest to empirical value* time-series). Additionally, since the surrogate model can be considered as a good estimation of the ABM, tasks requiring a lot of model evaluation can be performed on the surrogate instead of the ABM. Such tasks include sensitivity analysis, parameter importance quantification,...

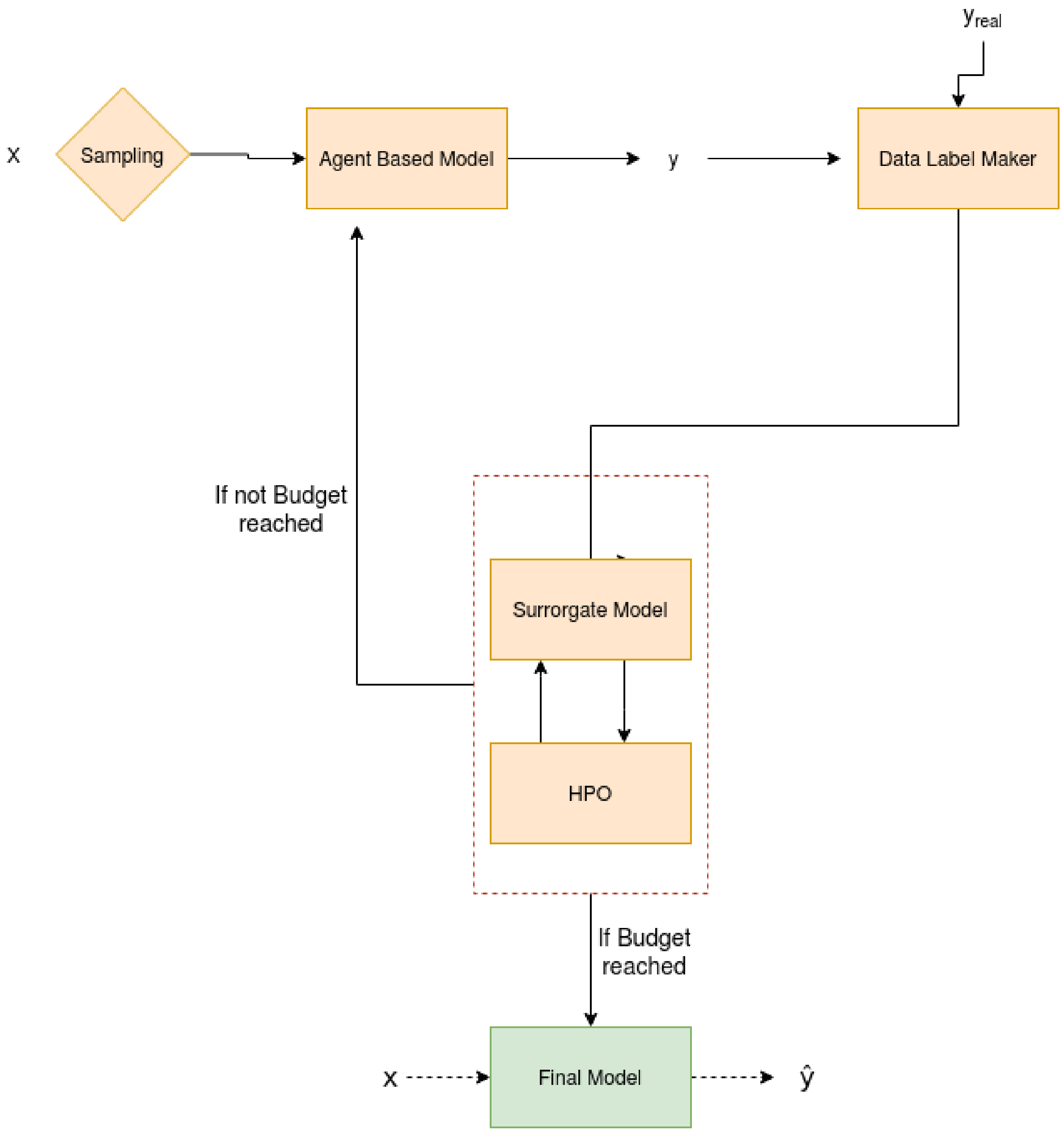


Figure 4.1: Calibration pipeline, the arrows indicate that a data transfer occurs between the different blocks.

Table 4.1: Building blocks of the surrogate calibration pipeline

Pipeline blocks	Variants	Remark
Sampling	Random	Bad performance for model exploration.
	Latin Hypercube	Good space filling properties and near orthogonality of sample points for NOLH.
	Sobol	Very good space filling properties, useful for sensitivity analysis
ABM	Brock and Hommes	
	Stochastic Structural Volatility	
Distance to empirical data calculation	KS test	Fast and easy to implement
	GSL div	More robust than KS test for series presenting time discrepancy.
Surrogate model	XGBoost	• Fast and very successful across a wide range of problems, provides a direct method for parameter importance ranking • Requires a lot of HPO and prone to overfitting
	Kriging	• Very fast and provides direct method to compute sensitivity analysis. • Does not perform well on highly non-linear response surfaces.
	Deep Neural Networks	• High modelling freedom allows to model highly non-linear systems. • Computationally heavy and require a lot of HPO, prone to overfitting.
Hyperparameter optimisation	Bayesian global optimisation	

Chapter 5

Results

5.1 Surrogate models performances

We introduced three different models for the surrogate approximation function: XGBoost, Kriging and DNN. The results on a dataset of max 1000 input/output combinations of the Brock and Hommes model are shown in figure 5.1 for a regression setting. The points are sampled with the Sobol procedure in order to have an optimal coverage of the domain of experiment. A snippet of the dataset can be found in Appendix A. The procedure to generate the performance measure is to present a large out-of-sample dataset to the surrogates and compare the predictions to the actual ABM outputs.

The two different approaches described in section 3.5 are tested here. Indeed, the first plot uses a direct error measure where the final n fraction of type 1 trader of the BH model is used as the target variable of the model. We observe that Kriging and XGBoost are faster to converge to their final MSE. It means that they need less samples to converge. On the other hand, the DNN approach needs more sample, but achieves better accuracy than the two other methods. The second and third plot use a calibrated error measure. In this case, the observed output variable of the BH model is the log return of the risky asset. More specifically, the model is run on T_{total} time steps and the log return distribution r_t is observed on the last T iterations.

$$r_t = log(\frac{p_t}{p_{t-1}}) \tag{5.1}$$

Focusing on the produced distribution after a certain number of iterations allows to keep the transitory period of the model out of the calibration. The produced log return is matched against the empirical daily log return of the S&P 500 index from the 1st of May 2018 to the 27th of April 2020[1] (See appendix B for a snippet of the table). In the second

[1]Data obtained from WSJ : https://www.wsj.com/market-data/quotes/index/SPX/historical-prices

plot, both series are compared using the p-value of the two samples KS test. In the third plot, both series are compared using the GSL-Div measure described in section 3.5.2.

The last plot shows the time of execution of each model training. As expected, the XGboost is by far the fastest algorithm. DNN is the slowest, but, for big datasets, the time of execution of the Kriging method seems to grow at the same pace as the one of the DNN. For the Kriging method, the time of execution is highly dependent on the Kernel. Here we use a Matern kernel as it is the one chosen by our hyperparameters optimisation in most cases. The performance gain of the DNN and Kriging approach does not seem to be significant enough to justify the use of a more computationally heavy algorithm for a regression setting.

In figure 5.2, the performances of the three models in a discrete setting are shown. The target variable used for model training is a binary label that equals to one is the p-value of the KS test is superior than 5% and to zero otherwise. Each model output is analysed in terms of accuracy, True Positive Rate (TPR) and F1 score. To recall, the accuracy is defined as the number of good predictions divided by the total number of predictions :

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{5.2}$$

The TPR (or recall) is defined as the number of true positive predictions divided by the total number of positive samples :

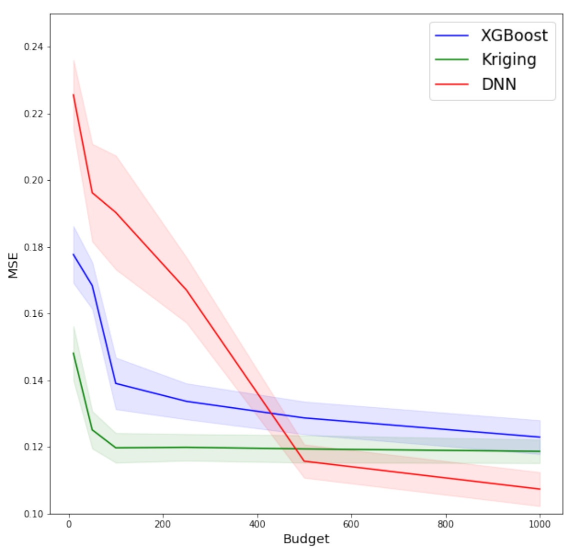
$$TPR = \frac{tp}{tp + fn} \tag{5.3}$$

Finally the F1 score is defined as :

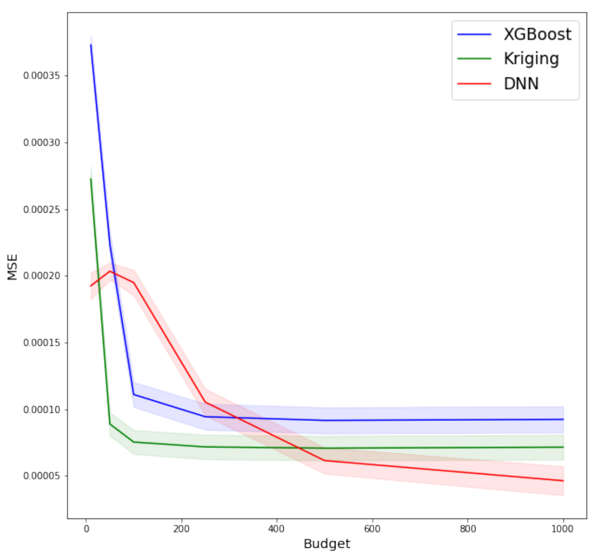
$$F1_{score} = \frac{tp}{tp + \frac{1}{2}(fp + fn)} \tag{5.4}$$

Where tp, tn, fp and fn represent respectively, the number of true positive, the number of true negative, the number of false positive, and the number of false negative calibration.

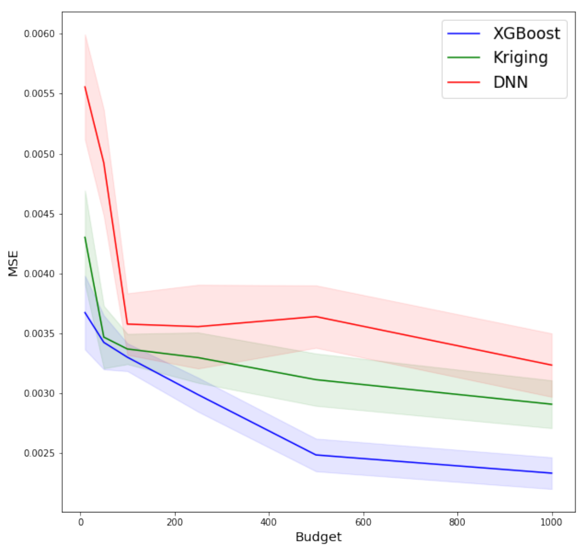
In Figure 5.2c, we observe that the accuracy is pretty good and constant across datasets of different sizes. This should not be interpreted as an indicator of good performance of the classification. Indeed, the present classification exercise is performed on a highly unbalanced dataset. The number of negative labels represents a very high fraction of the dataset. In this case, the accuracy will be high, even if the classifier does not capture well the inner dynamic of the ABM. Indeed, by classifying each point as *False*, the classifier achieves an accuracy equal to the fraction of negative labels in the dataset. To have a better insight on the prediction power of the classifier we can look at the True Positive Rate and the F1 score. In the context of a calibration, we are interested in identifying the



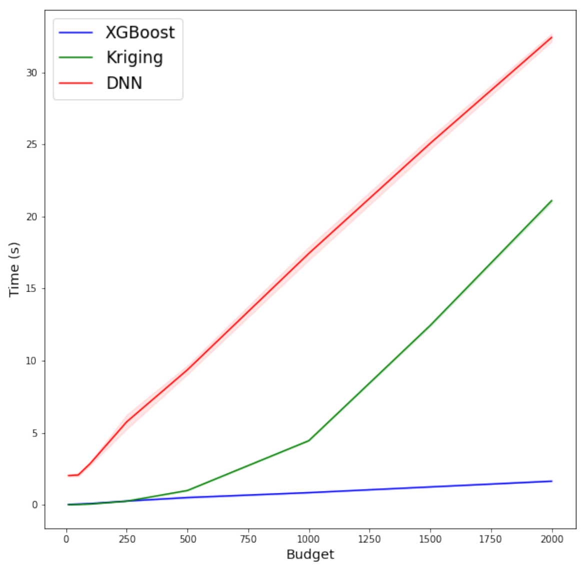
(a) MSE for n prediction



(b) MSE for p-value prediction



(c) MSE for Gsl-Div prediction



(d) Time of execution

Figure 5.1: Comparison of the Mean square error of the three models in a regression setting. The training set comprises a maximum of 1000 input combination of the Brock and Hommes model. The upper left plot depicts the MSE for models trained to predict the final fraction n of trader of type 1. The upper right plot depicts the MSE for models trained to predict the p-value of the KS test run with the predicted log-return of the risky asset an the SP 500 daily log-return. The bottom right plot depicts the MSE for models trained to predict the GSL-div measure of the two above-mentioned time-series. Results are averaged on 20 experiments and the 80% confidence interval is shown. The bottom right plot shows the running time of each model.

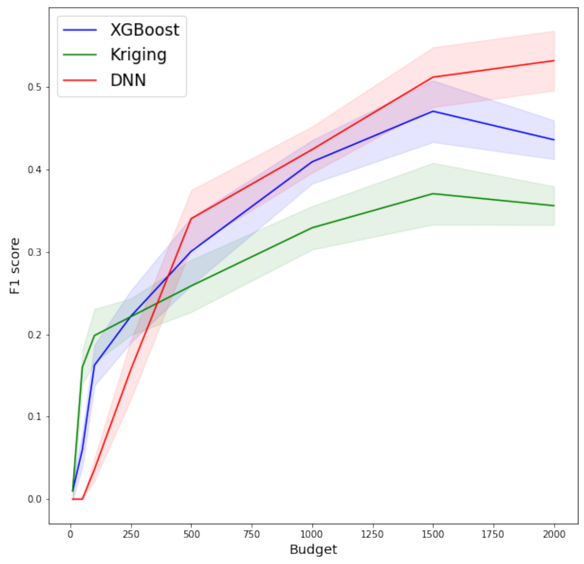
positive samples. That is, we are interested in finding all the input parameter combinations that lead to a log-return distribution close to the empirical one. This suggests that the false positive predictions should not be too strictly penalised and that the key metric to maximise is the TPR. Indeed, the predicted positive calibration can be easily verified by running the ABM (as long as they are not a majority). In 5.2b we observe the TPR of the classification performed by each model. As for the regression setting, the best performing model is the DNN. The best observed TPR is 0.6 when the model is trained with 2000 samples. These results are slightly below those observed in [30]. However, we will see in the next section that the custom training procedure described in section 3.7 can boost the TPR to higher values. In the discrete case, the values obtained from the three models suggest that the DNN approach performs significantly better. This observation is strengthened by the fact that, for Kriging and XGboost, the TPR and F1 score tends to stall and even decrease for bigger datasets while the predicted values of the DNN tends to increase even for large datasets.

5.2 Training procedure

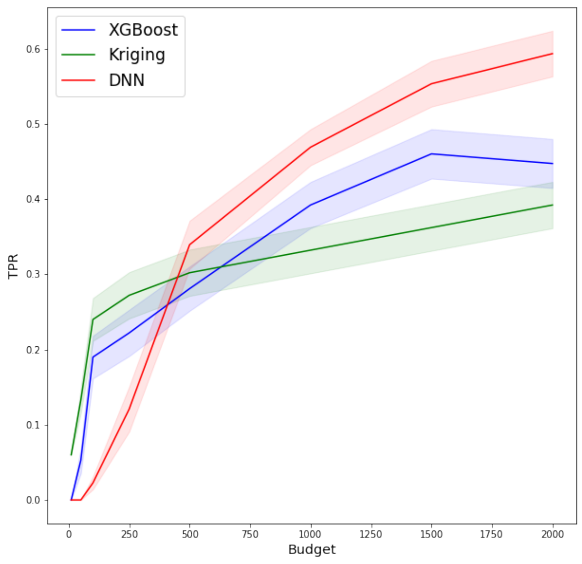
We want to assess the performance of the training procedure described in 3.7. To do so, we look at the F1 score and the True Positive Rate (TPR) of a discrete calibration exercise on the Brock and Holmes Model. A DNN surrogate model is trained using the calibration procedure described in section 4 and another DNN surrogate is trained with a simple one-shot approach.

In figure 5.3 we observe the resulting F1 score, TPR, and accuracy of both the surrogates described here above. While the F1 score is pretty equivalent for both methods, the TPR is substantially higher for the sequential calibration. This is in line with what we expected in section 3.7. Indeed, the training procedure is designed to add a maximum of sample points resulting in positive calibration to the dataset so that the learning algorithm can be more precise on those specific regions. The fact that the F1 score does not increase suggest that while the true positive predictions increase, the false positive predictions increase as well. This is confirmed by the fact that the accuracy slightly decreases. However, the fact that the accuracy remains at a high value confirms that the the increase of the false positive predictions is not dramatic. It implies that the positive predictions can easily be verified by running them through the ABM.

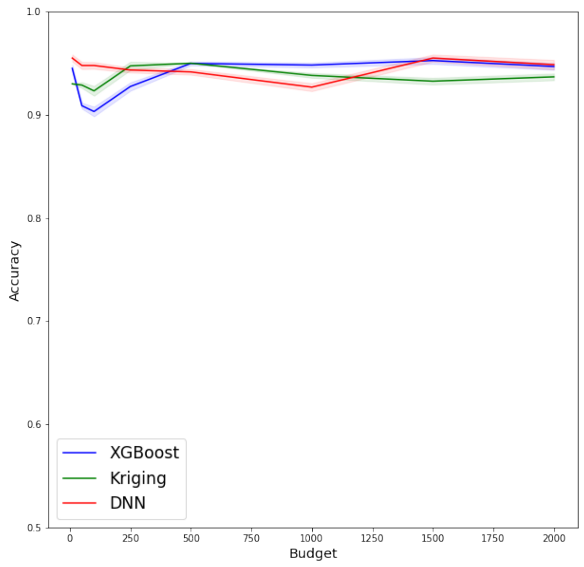
The speed up power of the surrogate modelling is shown in figure 5.4 for a XGboost surrogate. The processing time for a given number of model evaluations is shown for both the ABM and the surrogate pipelines corresponding to the three models trained with the sequential training procedure. For the surrogates, the training time (comprising a given number of ABM model evaluation as well) must be taken into account so that the curves begin with an offset equal to the training time. After that, the slopes are equals to the *one-sample evaluation time* of the surrogates. On the other hand, the time of execution of



(a) F1 Score



(b) True positive rate

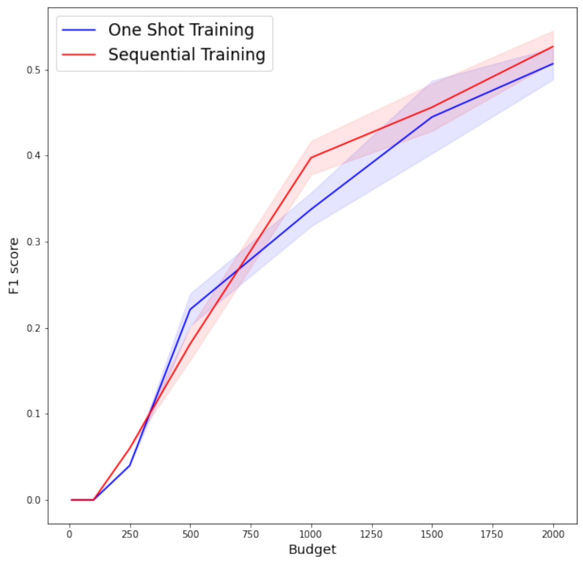


(c) Accuracy

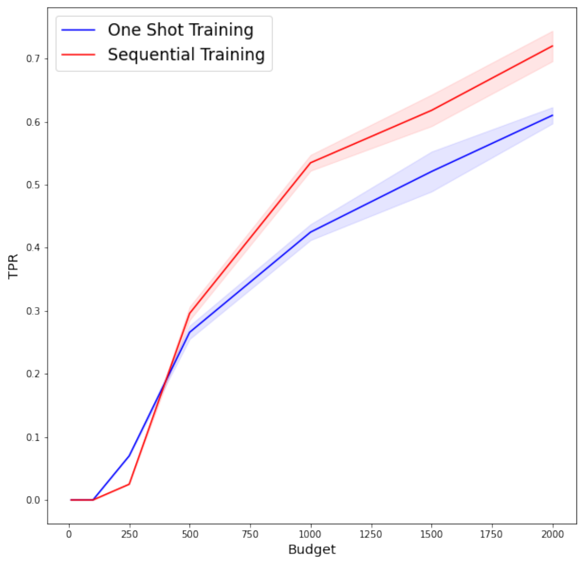
Figure 5.2: Performance of each model for a discrete calibrated error measure. The classification is achieved on a binary label equals to True when the p-value is higher than 5% and to False otherwise. The first plot depicts the f1 score, the second the true positive rate and the last the accuracy. Results are averaged on 20 experiments and the 80% confidence interval is shown.

the ABM is simply a linear curve with the slope equal to the *one-sample evaluation time* of the ABM.

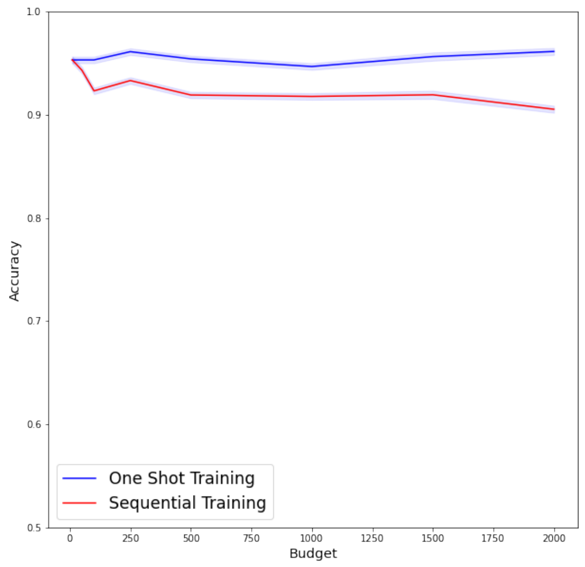
We observe that, for a surrogate trained with a budget of 2000, the critical number of model evaluations for the fastest pipeline (XGBoost) is around 2300. If more model evaluations are performed, then the surrogate approach is faster. For the slowest pipeline (DNN) a bit more than 10 000 model evaluations must be performed to have a performance



(a) F1 Score



(b) True positive rate



(c) Accuracy

Figure 5.3: Performance of a DNN surrogate for one-shot training and sequential training for a discrete calibrated error measure. The classification is achieved on a binary label equals to True when the p-value is higher than 5% and to False otherwise. The first plot depicts the f1 score, the second the true positive rate and the last the accuracy. Results are averaged on 20 experiments and the 80% confidence interval is shown.

gain. Note that it is the training time that is determining since the evaluation time of the three surrogates are approximately equivalents. It is important to note that when a surrogate is trained, it is valid as long as the ABM formulation or the domain of experiment is not modified. The model evaluation can then be performed on a long time period. We conclude that, for a high number of model evaluations, the surrogate model outperforms by far the original ABM in term of computational time. Additionally, the more model evaluations are achieved, the less the training time is significant compared to the performance gain guaranteed by the surrogate approach. This suggests that, if a high number of evaluations are going to be performed on the surrogate, the best performing algorithm should be chosen, no matter if it is the one requiring the largest training time.

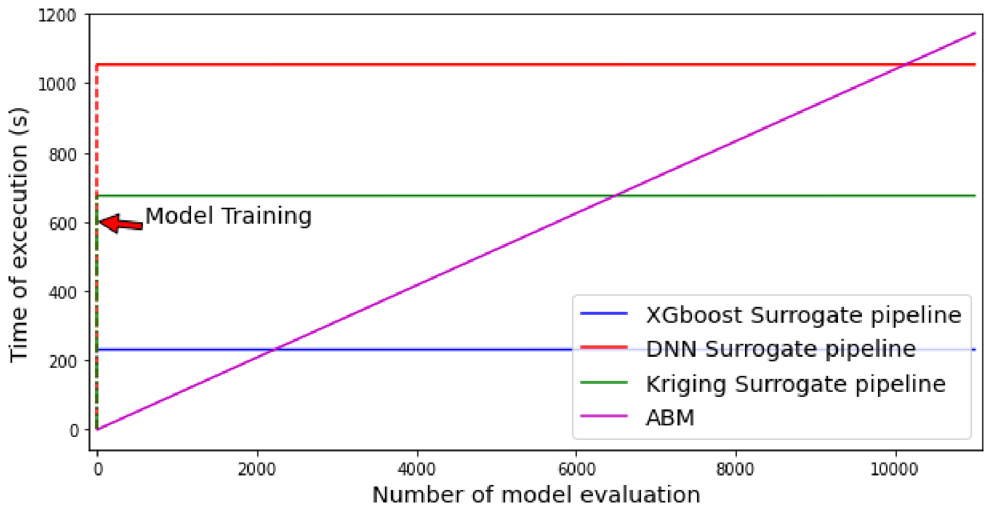


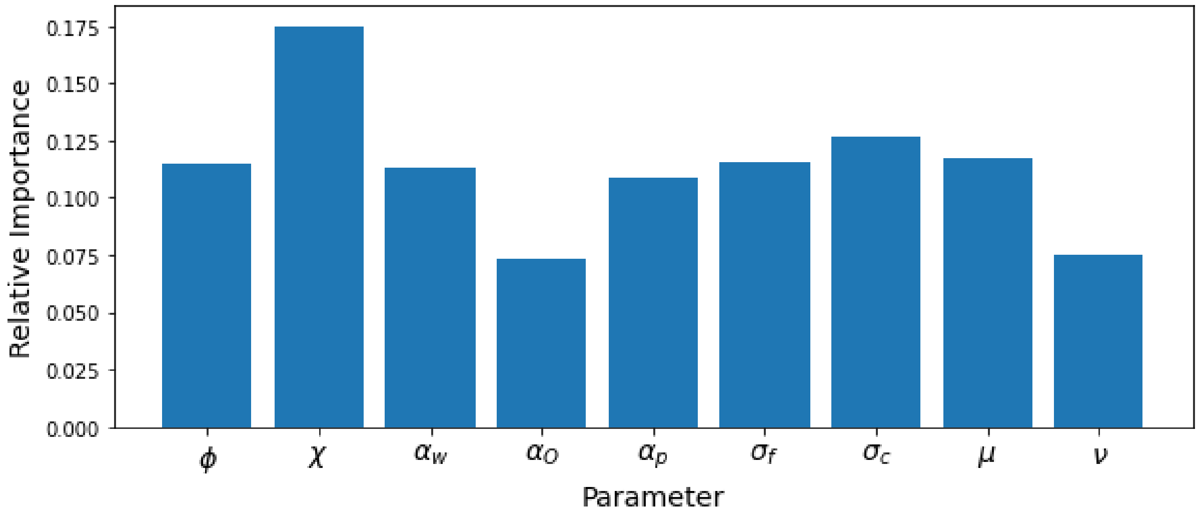
Figure 5.4: Execution time with respect to the number of model evaluations for both the Brock and Hommes model and the corresponding surrogate models trained with a 2000 samples budget. For the surrogates, the execution time includes the training time

5.3 Parameters sensitivity analysis

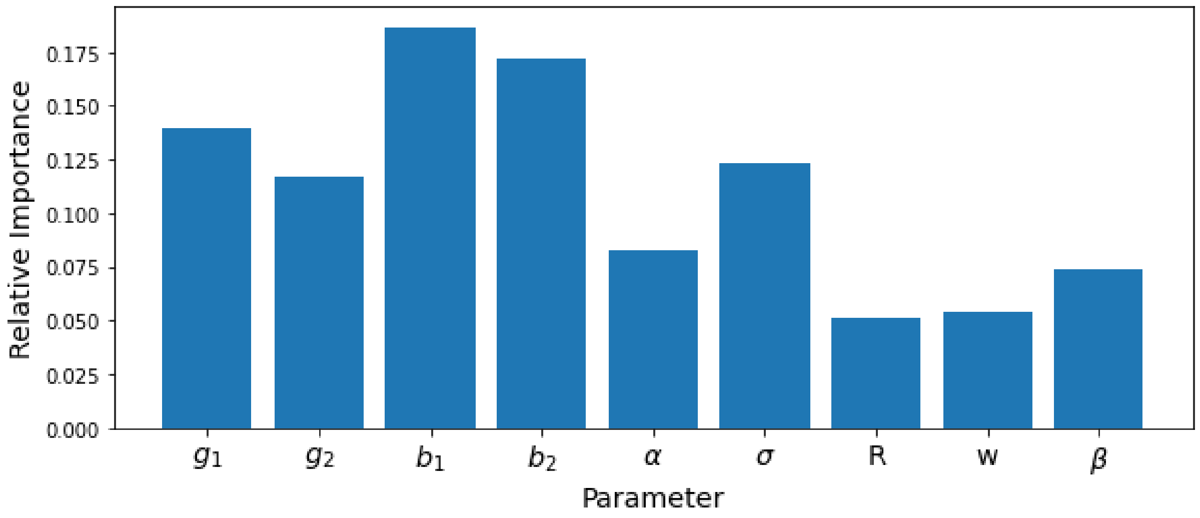
One of the tasks traditionally requiring a great number of model runs is the parameter sensitivity analysis. Different methods for parameter sensitivity analysis can possibly be coupled with the surrogates approach. For example, the structure of the Kriging method allows to easily derive the functional analysis of variance (ANOVA) of the model [41]. On the other hand, the XGboost present an even easier and computationally free approach to derive a feature importance ranking. This approach uses the number of splits over a given parameter as a quantification of the sensitivity of the model to this parameter.

The relative importance of each parameter for the Brock and Hommes model and the Structural Stochastic Volatility model are shown in figure 5.5a and 5.5b. We find that for both models, the most relevant parameters are those affecting the trading behaviour of the two groups of traders. In the BH model, these parameters are the two trend components g_1 and g_2 and the two bias terms b_1 and b_2. In the SSV model, these parameters are the aggressiveness of chartist, χ, and the one of fundamentalist ϕ and the variances of the noise component attached to each strategy : σ_f and σ_c. It is remarkable to observe that surrogates trained to achieve the best fit with empirical SP500 log return showcase a high importance for qualitatively equivalent parameters in both models. These results are in line with the calibration of the Brock and Hommes models achieved in [30] and [51].

For the SSV model, we see that α_w and α_p show higher parameter importance than α_o. The macroeconomic interpretation of that fact would suggest that herding and price misalignment influence the trader behaviours more than their natural predisposition toward a certain trading strategy.



(a) Feature importance of SSV model



(b) Feature importance of BH model

Figure 5.5: Feature importance

5.4 Model selection

In this section, we are interested in using the calibration procedure described in section 4 to perform a rapid model selection. The two models compared are the BH model and the SSV model. We compare the results of a calibration performed with a 2000 ABM computations budget. The values of the different metrics considered are listed in tabular 5.1. We observe that each metric considered is higher for the BH model. It means that the BH model is better suited to generate log-return distribution resembling to the S&P 500 index log return from the 1^{st} of May 2018 to the 27^{th} of April 2020.

Table 5.1: Model comparison. A DNN surrogate is trained using sequential training with a 2000 ABM computations budget. The p-value and the GSL-Div are of the produced table are then used to generate the above metrics

	Brook and Hommes model	Structural Stochastic volatility model
Average value of the Gsl-Div	0.55	0.48
Max value of the Gsl-Div	0.75	0.58
Max value of the p-value	0.172	0.161
Number of samples above 5% threshold	23	17

Chapter 6

Conclusion

In this thesis, we have investigated the use of surrogates modelling to help perform calibration of macroeconomic models. The proposed approach combines different known methods to perform calibration in a context of limited resources. We have seen that by combining sequential sampling, Machine Learning algorithms, hyperparameters optimisation and time-series comparison, we can achieve good calibration results while keeping the complexity relatively low by comparison with other methods. While surrogates modelling is already extensively applied in domains such as Global Optimisation, Electronic Design Automation or Ecological Modelling, their applications in an ABM framework remain scarce. Nevertheless, it has proven to be well suited for ABM approximations in a first time and ABM calibration and comparison in a second time.

Among the few contributions available to our knowledge we found the two papers on which the methods exposed in this thesis are mainly derived. Barde and Van der Hoog have proved in [5] that a surrogate calibration can be applied successfully to large-scale ABM models. Indeed, they apply a Kriging based surrogate modelling to the Eurace@Unibi macroeconomic simulation model[1]. One could hope that such a result will encourage researchers to look into surrogate modelling as a potential candidate for ABM calibrations. In [5], Lamperti, Roventini and Sani introduce a sequential training procedure to improve the True Positive Rate of the surrogate models. They claim a final TPR of 95% on the Brook and Hommes model for a regression setting. While we do not reach such performance (we only tested a classification setting), we observe the same qualitative behaviour where the use of a relatively small budget allows to quickly reach good performance on the entire domain.

The main contribution of this thesis is the application of a DNN as the function approximation block in the modelling procedure. We discovered that it seems to be better able to capture the inner dynamic of the ABMs, especially in classification setting.

[1]The calibration is achieved on a relatively low number of 513 input combinations.

One of the key identified properties of the surrogate approach is its ability to identify new promising input combinations. This feature allows a modeller to perform global optimisation in order to identify the positive calibration almost at free cost. This feature is special to surrogate modelling and cannot be found in any other calibration technique.

6.1 Perspectives

The extent of this thesis is pretty large since a great amount of methods and algorithms have been investigated. For this reason, the results obtained only scratch the surface of what is achievable for a surrogate modelling in a macroeconomic setting. The limited computational and time resources available made it difficult to extensively test time expensive models such as DNN, despite its promising ABM approximation properties. The use of more reliable time-series comparison tools such as the GSL-Div and the Markov Information Criterion has also been prevented by the limited computational power available. For full-scale applications of the protocol, such measures should be used. However, the use of the p-values of the KS test, despite its less reliable theoretical fundamentals, provide a good proof of concept for the online-training procedure that we developed.

In further research, the use of DNN as the function approximation block of the surrogate modelling could be more deeply investigated. Indeed, we have seen that under the assumption that the surrogate will be extensively used after training, the computational cost of the training procedure should not be taken into account as an critical parameter. Another element that could encourage the use of DNN is its current predominance in a great number of Machine Learning tasks. This led to the development of large, well-documented libraries such as PyTorch and TensorFlow. Additionally, different hardware optimisations have been put together to speed up the computation of DNN. A further improvement could then be to investigate the eligibility of the DNN model developed here to those hardware optimisations.

The use of the framework developed on more complex Agent Based Models could also be investigated. The advantage of the surrogate modelling paradigm is that it can be applied on any ABM regardless of its size or complexity as long as it produces input/output combinations. This makes the approach highly robust and flexible. The results presented here as well as in [30, 41, 5, 51] suggest that indirect calibration using surrogate models is a promising path of research to follow. It could, in the end, provide a standardised and robust set of tools for Agent-Based model calibration, helping to reduce the persisting lack of empirical grounding of the ABM macroeconomics models. The development of good, well-documented libraries following, for example, the work of [7] is key to the potential success of the surrogates paradigm[2].

Beside the calibration procedure described here, training surrogate approximations of ABM could be useful in other situations. Sander Van der Hoog exposes in [45] several paths of research for the surrogate modelling applied to ABM. Among the potential promising path we find the possibility to couple different ABMs by using their surrogate model. Indeed, having a meta-model for different models could allow to easily couple their input/output into a hierarchical modelling scheme. This would enable ≪ABMs of ABMs≫. Another path of research is to use the learnt model into a reinforcement learning scheme. Policy assessment could be achieved by setting up agents to choose the policy leading to maximum reward in a given ABM modelled economic situation.

[2]https://github.com/SMTorg/smt

Bibliography

[1] Acar, E., and Rais-Rohani, M. Ensemble of metamodels with optimized weight factors. *Structural and Multidisciplinary Optimization 37* (04 2008), 279–294.

[2] Archer, K. J., and Kimes, R. V. Empirical characterization of random forest variable importance measures. *Computational Statistics Data Analysis 52*, 4 (2008), 2249–2260.

[3] Barde, S. Direct comparison of agent-based models of herding in financial markets. *Journal of Economic Dynamics and Control 73* (2016), 329 – 353.

[4] Barde, S. A practical, accurate, information criterion for nth order markov processes. *Computational Economics* (09 2016).

[5] Barde, S., and van der Hoog, S. An empirical validation protocol for large-scale agent-based models, 06 2017.

[6] Bergmann, B. R. A microsimulation of the macroeconomy with explicitly represented money flows. In *Annals of Economic and Social Measurement, Volume 3, number 3*. National Bureau of Economic Research, Inc, 1974, pp. 475–489.

[7] Bouhlel, M. A., Hwang, J. T., Bartoli, N., Lafage, R., Morlier, J., and Martins, J. R. A python surrogate modeling framework with derivatives. *Advances in Engineering Software 135* (2019), 102662.

[8] Brock, W. A., and Hommes, C. H. Heterogeneous beliefs and routes to chaos in a simple asset pricing model. *Journal of Economic Dynamics and Control 22*, 8 (1998), 1235 – 1274.

[9] Burhenne, S., Jacob, D., and Henze, G. Sampling based on sobol' sequences for monte carlo techniques applied to building simulations. pp. 1816–1823.

[10] Carroll, C. Macroeconomic expectations of households and professional forecasters. *The Quarterly Journal of Economics 118* (02 2003), 269–298.

[11] Chakraborti, A., and Chakrabarti, B. Statistical mechanics of money: How saving propensity affects its distribution. *The European Physical Journal B - Condensed Matter and Complex Systems 17* (09 2000).

[12] Chen, S.-h., Chang, C.-L., and Du, Y.-R. Agent-based economic models and econometrics. *The Knowledge Engineering Review 000* (01 2009), 1–46.

[13] Chen, T., and Guestrin, C. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Aug 2016).

[14] Cioppa, T. Efficient nearly orthogonal and space-filling experimental designs for high-dimensional complex models /.

[15] Crombecq, K., Gorissen, D., Deschrijver, D., and Dhaene, T. A novel hybrid sequential design strategy for global surrogate modeling of computer experiments. *SIAM J. Scientific Computing 33* (01 2011), 1948–1974.

[16] De Vany, A. Bounded rationality in macroeconomics : Thomas J. Sargent, (Oxford University Press, New York, NY, 1993) pp. xii, 184; 28.00, *Paper*16.95. *Journal of Economic Dynamics and Control 20*, 5 (May 1996), 811–817.

[17] Deissenberg, C., [van der Hoog], S., and Dawid, H. Eurace: A massively parallel agent-based model of the european economy. *Applied Mathematics and Computation 204*, 2 (2008), 541 – 552. Special Issue on New Approaches in Dynamic Optimization to Assessment of Economic and Environmental Systems.

[18] Dosi, G., Fagiolo, G., and Roventini, A. Schumpeter meeting keynes: A policy-friendly model of endogenous growth and business cycles. *Journal of Economic Dynamics and Control 34*, 9 (2010), 1748 – 1767. Computational perspectives in economics and finance: Methods,dynamic analysis and policy modeling.

[19] Dragulescu, A., and Yakovenko, V. Statistical mechanics of money. *The European Physical Journal B 17*, 4 (Oct 2000), 723–729.

[20] Franke, R., and Westerhoff, F. Estimation of a structural stochastic volatility model of asset pricing. *Computational Economics 38* (06 2011), 53–83.

[21] Franke, R., and Westerhoff, F. Structural stochastic volatility in asset pricing dynamics: Estimation and model contest. *Journal of Economic Dynamics and Control 36* (03 2011).

[22] Franke, R., and Westerhoff, F. Structural stochastic volatility in asset pricing dynamics: Estimation and model contest. *Journal of Economic Dynamics and Control*

36, 8 (2012), 1193 – 1211. Quantifying and Understanding Dysfunctions in Financial Markets.

[23] Grazzini, J., Richiardi, M. G., and Tsionas, M. Bayesian estimation of agent-based models. *Journal of Economic Dynamics and Control 77* (2017), 26 – 47.

[24] Guerini, M., and Moneta, A. A method for agent-based models validation. *Journal of Economic Dynamics and Control 82*, C (2017), 125–141.

[25] Han, J., Jentzen, A., and E, W. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences 115*, 34 (Aug 2018), 8505–8510.

[26] Kampouridis, M., Chen, S.-h., and Tsang, E. simple asset pricing model " 1998 jedc paper.

[27] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization, 2014.

[28] Ladyman, J., Lambert, J., and Wiesner, K. What is a complex system? *European Journal for Philosophy of Science 3* (06 2013).

[29] Lamperti, F. An information theoretic criterion for empirical validation of simulation models. *Econometrics and Statistics 5* (2018), 83 – 106.

[30] Lamperti, F., Roventini, A., and Sani, A. Agent-based model calibration using machine learning surrogates, 2017.

[31] Lux, T., and Zwinkels, R. Empirical validation of agent-based models. *SSRN Electronic Journal* (01 2017).

[32] Niazi, M., and Hussain, A. Agent-based computing from multi-agent systems to agent-based models: a visual survey. *Scientometrics 89* (11 2011), 479–499.

[33] Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning, 2018.

[34] Pan, I., Babaei, M., Korre, A., and Durucan, S. Artificial neural network based surrogate modelling for multi- objective optimisation of geological co2 storage operations. *Energy Procedia 63* (2014), 3483 – 3491. 12th International Conference on Greenhouse Gas Control Technologies, GHGT-12.

[35] Platt, D. A comparison of economic agent-based model calibration methods. *Journal of Economic Dynamics and Control 113* (2020), 103859.

[36] Platt, D. A comparison of economic agent-based model calibration methods. *Journal of Economic Dynamics and Control 113* (2020), 103859.

[37] Quirante, N., Javaloyes, J., and Caballero, J. Rigorous design of distillation columns using surrogate models based on kriging interpolation. *AIChE Journal 61* (03 2015).

[38] Ramachandran, P., Zoph, B., and Le, Q. Swish: a self-gated activation function.

[39] Rasmussen, C. E., and Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[40] Rogers, A., and Tessin, P. Multi-objective calibration for agent-based models.

[41] Salle, I., and Yildizoglu, M. Efficient sampling and meta-modeling for computational economic models. *Computational Economics 44* (12 2014).

[42] Seppecher, P. Jamel, a Java Agent-based MacroEconomic Laboratory. Working Papers halshs-00697225, HAL, May 2012.

[43] Tedeschi, G., Recchioni, M., and Gallegati, M. A calibration procedure for analyzing stock price dynamics in an agent-based framework. *http://ssrn.com/abstract=2330739* (09 2013).

[44] Tripathy, R. K., and Bilionis, I. Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics 375* (dec 2018), 565–588.

[45] van der Hoog, S. Deep learning in (and of) agent-based models: A prospectus, 2017.

[46] Viana, F., Gogu, C., and Haftka, R. Making the most out of surrogate models: Tricks of the trade. *Proceedings of the ASME Design Engineering Technical Conference 1* (01 2010).

[47] Weisbuch, G., and Ryckebusch, S. *Complex systems dynamics: An introduction to automata networks.* 01 2018.

[48] Whitesides, G. M., and Ismagilov, R. F. Complexity in chemistry. *Science 284*, 5411 (1999), 89–92.

[49] Yakovenko, V. M., and Rosser, J. B. Colloquium: Statistical mechanics of money, wealth, and income. *Rev. Mod. Phys. 81* (Dec 2009), 1703–1725.

[50] Yan, L., and Zhou, T. An adaptive surrogate modeling based on deep neural networks for large-scale bayesian inverse problems, 2019.

[51] Zhang, Y., Li, Z., and Zhang, Y. Validation and calibration of an agent-based model: A surrogate approach. *Discrete Dynamics in Nature and Society 2020* (01 2020), 1–9.

[52] Zou, H., and Hastie, T. Regularization and variable selection via the elastic net (vol b 67, pg 301, 2005). *Journal of the Royal Statistical Society Series B 67* (02 2005), 768–768.

Appendix A

DOE of ABM models

Table A.1: 20 first points of the the Brock and Hommes DOE generated with Sobol Sampling

	g1	g2	b1	b2	alpha	sigma	R	w	beta	gsl	p_val	n
0	0.0	0.0	0.0	0.0	5.0	0.5	1.05	0.5	2.5	0.55000	4.95271e-104	0.5
1	1.0	-1.0	1.0	-1.0	7.5	0.25	1.075	0.25	1.25	0.55665	1.20105e-137	0.003797
2	-1.0	1.0	-1.0	1.0	2.5	0.75	1.025	0.75	3.75	0.53877	7.00027e-215	4.70660e-19
3	-0.5	-0.5	0.5	-1.5	8.75	0.875	1.0125	0.625	0.625	0.56027	7.39950e-300	0.50039
4	1.5	1.5	-1.5	0.5	3.75	0.375	1.0625	0.125	3.125	0.56027	7.39950e-300	0.5
5	0.5	-1.5	-0.5	-0.5	1.25	0.625	1.0875	0.875	1.875	0.56027	7.39950e-300	0.39436
6	-1.5	0.5	1.5	1.5	6.25	0.125	1.0375	0.375	4.375	0.55259	3.56866e-57	3.78288-e-163
7	-1.25	-0.75	-0.75	0.75	5.625	0.1875	1.00625	0.9375	0.9375	0.54709	2.64628e-134	0.35182
8	0.75	1.25	1.25	-1.25	0.625	0.6875	1.05625	0.4375	3.4375	0.54375	7.08396e-174	1.59391e-150
9	1.75	-1.75	0.25	1.75	3.125	0.4375	1.08125	0.6875	2.1875	0.55149	3.56866e-57	1.0
10	-0.25	0.25	-1.75	-0.25	8.125	0.9375	1.03125	0.1875	4.6875	0.56027	7.39950e-300	0.49728
11	-0.75	-1.25	1.75	0.25	4.375	0.8125	1.01875	0.3125	0.3125	0.48848	1.65616e-29	0.50018
12	1.25	0.75	-0.25	-1.75	9.375	0.3125	1.06875	0.8125	2.8125	0.56027	7.39950e-300	0.00052
13	0.25	-0.25	-1.25	1.25	6.875	0.5625	1.09375	0.0625	1.5625	0.55635	8.00889e-169	0.20364
14	-1.75	1.75	0.75	-0.75	1.875	0.0625	1.04375	0.5625	4.0625	0.56027	7.39950e-300	1.0
15	-1.625	-0.125	1.375	-0.375	2.8125	0.34375	1.053125	0.84375	3.90625	0.55933	1.79779e-242	0.00317
16	0.375	1.875	-0.625	1.625	7.8125	0.84375	1.003125	0.34375	1.40625	0.54213	3.06367e-119	1.27767e-17

Table A.2: 20 first points of the the Structural Stochastic Volatility DOE generated with Sobol Sampling

	phi	chi	alphaw	alphao	alphap	sigmaf	sigmac	mu	ni	n	p_val	gsl
0	2.5	0.0	0.0	0.0	10.0	1.0	1.0	0.5	0.5	0.5000	0.00299	0.36389
1	3.75	-2.5	2.5	-2.5	15.0	0.5	1.5	0.25	0.25	1.0	2.42373e-21	0.36403
2	1.25	2.5	-2.5	2.5	5.0	1.5	0.5	0.75	0.75	1.0	1.20105e-137	0.35361
3	1.875	-1.25	1.25	-3.75	17.5	1.75	0.25	0.625	0.125	0.0	7.84802e-13	0.29759
4	4.375	3.75	-3.75	1.25	7.5	0.75	1.25	0.125	0.625	0.0	9.34101e-36	0.38040
5	3.125	-3.75	-1.25	-1.25	2.5	1.25	1.75	0.875	0.375	0.0	7.39950e-300	0.35261
6	0.625	1.25	3.75	3.75	12.5	0.25	0.75	0.375	0.875	1.0	6.12985e-27	0.34452
7	0.9375	-1.875	-1.875	1.875	11.25	0.375	0.125	0.9375	0.1875	0.0	7.39950e-300	0.11046
8	3.4375	3.125	3.125	-3.125	1.25	1.375	1.125	0.4375	0.6875	1.0	1.71440e-07	0.31275
9	4.6875	-4.375	0.625	4.375	6.25	0.875	1.625	0.6875	0.4375	1.0	7.39950e-300	0.11045
10	2.1875	0.625	-4.375	-0.625	16.25	1.875	0.625	0.1875	0.9375	0.0	1.67154e-30	0.37393
11	1.5625	-3.125	4.375	0.625	8.75	1.625	0.375	0.3125	0.0625	0.0	1.00289e-06	0.32222
12	4.0625	1.875	-0.625	-4.375	18.75	0.625	1.375	0.8125	0.5625	0.0	7.39950e-300	0.11046
13	2.8125	-0.625	-3.125	3.125	13.75	1.125	1.875	0.0625	0.3125	0.0	3.54526e-31	0.37457
14	0.3125	4.375	1.875	-1.875	3.75	0.125	0.875	0.5625	0.8125	0.0	3.29709e-12	0.36661
15	0.46875	-0.3125	3.4375	-0.9375	5.625	0.6875	1.0625	0.84375	0.78125	1.0	0.02406	0.37470
16	2.96875	4.6875	-1.5625	4.0625	15.625	1.6875	0.0625	0.34375	0.28125	1.0	7.39950e-300	0.11046
17	4.21875	-2.8125	-4.0625	-3.4375	10.625	0.1875	0.5625	0.59375	0.53125	0.0	7.39950e-300	0.11046
18	1.71875	2.1875	0.9375	1.5625	0.625	1.1875	1.5625	0.09375	0.03125	0.0	1.5027e-32	0.37864
19	2.34375	-4.0625	-0.3125	-2.1875	13.125	1.4375	1.3125	0.46875	0.90625	1.0	7.3995e-300	0.11046

Appendix B

S&P 500 data

Table B.1: Log return of the S&P 500 index between 2^{nd} of May 2018 and 29^{th} of May 2018

	log_return
0	-0.007231951218717114
1	-0.0022562178107401465
2	0.012729810253516938
3	0.0034519805782133872
4	-0.00026567663510679296
5	0.009635647864919328
6	0.009327011495041226
7	0.001706139511771454
8	0.0008830994699682293
9	-0.006865655176056862
10	0.004052339933860338
11	-0.0008562387426369611
12	-0.00263566521405334
13	0.007359604524262764
14	-0.0031406888277452083
15	0.003243145333145847
16	-0.0020252628489343394
17	-0.0023600031860420145
18	-0.011631572622697206
19	0.01261587559480315
20	-0.006903333156201619

Table B.2: OHLC chart of the S&P 500 index between 1st of May 2018 and 29th of May 2018

Date	Open	High	Low	Close	Adj Close	Volume
2018-05-01	2642.959961	2655.270020	2625.409912	2654.800049	2654.800049	3559850000
2018-05-02	2654.239990	2660.870117	2631.699951	2635.669922	2635.669922	4010770000
2018-05-03	2628.080078	2637.139893	2594.620117	2629.729980	2629.729980	3851470000
2018-05-04	2621.449951	2670.929932	2615.320068	2663.419922	2663.419922	3327220000
2018-05-07	2680.340088	2683.350098	2664.699951	2672.629883	2672.629883	3237960000
2018-05-08	2670.260010	2676.340088	2655.199951	2671.919922	2671.919922	3717570000
2018-05-09	2678.120117	2701.270020	2674.139893	2697.790039	2697.790039	3909500000
2018-05-10	2705.020020	2726.110107	2704.540039	2723.070068	2723.070068	3333050000
2018-05-11	2722.699951	2732.860107	2717.449951	2727.719971	2727.719971	2862700000
2018-05-14	2738.469971	2742.100098	2725.469971	2730.129883	2730.129883	2972660000
2018-05-15	2718.590088	2718.590088	2701.909912	2711.449951	2711.449951	3290680000
2018-05-16	2712.620117	2727.760010	2712.169922	2722.459961	2722.459961	3202670000
2018-05-17	2719.709961	2731.959961	2711.360107	2720.129883	2720.129883	3475400000
2018-05-18	2717.350098	2719.500000	2709.179932	2712.969971	2712.969971	3368690000
2018-05-21	2735.389893	2739.189941	2725.699951	2733.010010	2733.010010	3019890000
2018-05-22	2738.340088	2742.239990	2721.879883	2724.439941	2724.439941	3366310000
2018-05-23	2713.979980	2733.330078	2709.540039	2733.290039	2733.290039	3326290000
2018-05-24	2730.939941	2731.969971	2707.379883	2727.760010	2727.760010	3256030000
2018-05-25	2723.600098	2727.360107	2714.989990	2721.330078	2721.330078	2995260000
2018-05-29	2705.110107	2710.669922	2676.810059	2689.860107	2689.860107	3736890000