Intelligent Systems with Applications 20 (2023) 200286

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Application of deep reinforcement learning in asset liability management

Takura Asael Wekwete [a,*,1], Rodwell Kufakunesu [b], Gusti van Zyl [b]

[a] Department of Computer Science, University of Pretoria, Pretoria, 0002, South Africa
[b] Department of Mathematics and Applied Mathematics, University of Pretoria, Pretoria, 0002, South Africa

ARTICLE INFO

Keywords:
Reinforcement learning
Deep learning
Deep reinforcement learning
Asset liability management
Duration matching
Redington immunisation
Deep hedging

ABSTRACT

Asset Liability Management (ALM) is an essential risk management technique in Quantitative Finance and Actuarial Science. It aims to maximise a risk-taker's ability to fulfil future liabilities. ALM is especially critical in environments of elevated interest rate changes, as has been experienced globally between 2021 and 2023. Traditional ALM implementation is still heavily dependent on the judgement of professionals such as Quants, Actuaries or Investment Managers. This over-reliance on human input critically limits ALM performance due to restricted automation, human irrationality and restricted scope for multi-objective optimisation. This paper addressed these limitations by applying Deep Reinforcement Learning (DRL), which optimises through trial, and error and continuous feedback from the environment. We defined the Reinforcement Learning (RL) components for the ALM application: the RL decision-making Agent, Environment, Actions, States and Reward Functions. The results demonstrated that DRL ALM can achieve duration-matching outcomes within 1% of the theoretical ALM at a 95% confidence level. Furthermore, compared to a benchmark weekly rebalancing traditional ALM regime, DRL ALM achieved superior outcomes of net portfolios which are, on average, 3 times less sensitive to interest rate changes. DRL also allows for increased automation, flexibility, and multi-objective optimisation in ALM, reducing the negative impact of human limitations and improving risk management outcomes. The findings and principles presented in this study apply to various institutional risk-takers, including insurers, banks, pension funds, and asset managers. Overall, DRL ALM provides a promising Artificial Intelligence (AI) avenue for improving risk management outcomes compared to the traditional approaches.

1. Introduction

A thriving and well-managed insurance, banking and risk management sector is vital for a country's sustainable economic growth. It encourages individuals and businesses to invest, spend and accumulate wealth with reduced uncertainty about the future (Ward & Zurbruegg, 2000). Asset Liability Management (ALM), also known as Asset Liability Modelling, is the Actuarial and Quantitative Finance risk management technique used by institutional risk-takers such as insurers, pension funds, banks and asset managers. A key aim of ALM is to derive an optimal investment asset allocation strategy for reducing interest rate risk exposure by taking into account the corresponding current and future liabilities. The inability to meet claim liabilities is one of the critical risks that institutional risk-takers address with ALM (Smink & van der Meer, 1997). Adequate implementation of ALM is especially critical in environments of rapidly changing interest rates as has been experienced globally between 2021 and 2023. Poor ALM was a key factor in the demise of several high profile banks in 2023 such as Sillicon Valley Bank (SVB), Credit Suisse and First Republic Bank (Geman, 2023, Daga, 2023, Barr, 2023).

A common ALM approach is Duration Matching, in which the asset allocation is chosen such that the timing of the future asset proceeds is aligned as much as possible with that of the expected liability outflows. The Duration of a set of cash flows is defined as the average timing of the cash flows weighted by the size of the respective discounted cash flows. The most common theoretical framework used for deriving Duration Matching is Redington Immunisation. One of the key objectives of Immunisation is for the optimal asset allocation weights to be chosen such that the Duration of the asset portfolio is close as possible to the Duration of liability cash flows (Fooladi & Roberts, 2000). Duration is a standard measure of the interest rate sensitivity of an asset or liability portfolio. Another secondary objective is to have the rate of change of the Duration (known as the Convexity) for the assets to be greater than that of the Liabilities where possible (Nieto et al., 2022).

* Corresponding author.
E-mail addresses: takurawekwete@gmail.com (T.A. Wekwete), rodwell.kufakunesu@up.ac.za (R. Kufakunesu), gusti.vanzyl@up.ac.za (G. van Zyl).
[1] Fellow Actuary - Actuarial Society of South Africa.

https://doi.org/10.1016/j.iswa.2023.200286
Received 23 May 2023; Received in revised form 22 July 2023; Accepted 7 October 2023
Available online 13 October 2023

In traditional ALM, the professional (Actuary, Quantitative Analyst or Investment Manager) typically initially estimates the Duration of the claim liabilities based on projected liability cash flows. For insurance contracts, such analysis and modelling usually incorporate risk factors such as sum insured, insurance term, policyholder risk factors, types of risks covered etc. Afterwards, the professionals typically select a portfolio of assets whose allocation into bonds, property, cash and other securities has a Duration within an acceptable threshold to that of liabilities. Assets which perfectly match the liabilities are not always available (Garrett, 2013, Kahlig, 2022).

Furthermore, this process must be repeated, for example, monthly or quarterly. The process usually involves multiple stakeholders, such as asset managers, treasury department, finance/reporting department, valuations/reserving department etc., and usually requires extensive data gathering across various data sources. This often presents a challenge in the speed of the process and, therefore, the frequency at which ALM re-balancing can be done. All the while, the composition of the underlying liability risks is constantly changing on a daily basis (sometimes hourly), especially in insurance, banking or asset management. Specific theoretical assumptions on the nature of interest rates underpin traditional Immunisation (Kahlig, 2022). However, the real-world conditions in which these models are applied often deviate from these assumptions, which requires the professionals to monitor these deviations and make appropriate in their application based on judgement. Deep Reinforcement Learning provides an avenue for the ALM to be less reliant on theoretical assumptions (Bühler et al., 2018).

ALM often needs to be achieved simultaneously with other objectives, such as maximising risk-adjusted returns or targeting a given differential between assets Duration and liability Duration. Consequently, in the traditional approach, some human judgement is commonly applied to the final asset allocation at each iteration in the quest to incorporate the additional objectives as much as possible. The applied human judgement is usually based on the professional's past experiences, corrections for model biases and balancing the importance of various additional objectives.

Unfortunately, this need for human intervention and judgement in the process introduces several limitations. The first limitation is that the traditional ALM process becomes difficult to automate. Consequently, there is a limit on the frequency of the updates and asset allocation re-balancing that can be done in a given period. In the time between the Duration Matching re-balancing iterations, there is also a risk of a rapid change in the interest rate hedge between assets and liabilities portfolios. Deep Reinforcement Learning introduces the possibility of minimising the requirement for human intervention (Hariom Tast & Lookabaugh, 2020).

Second, a significant drawback of the conventional ALM approach, Redington Immunisation, is its limited capacity for facilitating exploration, experimentation, and error correction of various asset allocations within a comprehensive feedback loop. This is because the number of potential allocations within and across various asset classes is numerous. In addition, the investment environment and the composition of the risk-takers liabilities are continuously changing. Hence, the traditional ALM Duration Matching approach is often not consistently and sufficiently exploratory of all the options within a learning cycle. Deep Reinforcement Learning provides a way for the Asset Liability Management to be executed in continuously improving feedback loop (Hariom Tast & Lookabaugh, 2020, Englisch et al., 2023, Krabichler & Teichmann, 2023).

Third, the high reliance on human intervention and judgement exposes the traditional Asset Liability Management investment processes to human behavioural irrationality and limitations. Humans are subject to emotions such as fear and greed, which can negatively lead to suboptimal asset allocation decisions. Humans are also vulnerable to a lack of consistency in applying the organisation's ALM Investment policy. Humans are also susceptible to other irrationalities such as confirmation bias, overconfidence, recency bias, availability bias, and many other biases (Syed & Bansal, 2018, Rabbani et al., 2021, Chiu et al., 2022, Bondt et al., 2013).

Fourth, in addition to the primary ALM, the need to weigh and prioritise current objectives in the ALM process can be challenging to implement. This is currently being done in the industry either by a rules-based software approach or human professional judgement. For example, suppose the ALM is carried out in a software tool such as spreadsheet software. In that case, it is often time-consuming and challenging to explicitly express all the other objectives and constraints. Deep Reinforcement Learning providers for the possibility to carry out optimisation of multiple objectives simultaneously (Englisch et al., 2023, Krabichler & Teichmann, 2023).

In this paper, we examined the feasibility, performance and advantages of using Deep Reinforcement Learning to implement ALM in relation to issues outline above of traditional ALM. Deep Reinforcement Learning incorporates Deep Learning into the Reinforcement Learning framework. A key benefit of this approach is that it creates an autonomous ALM decision-making agent which directly learns the objectives of ALM and which actions to take, or not take, with minimal supervision. Furthermore, this Deep Reinforcement Learning Asset Liability Modelling (DRL ALM) agent not only learns through a combination of historical data and outcomes, but also its own trial and error, and continuous feedback from the environment. DRL ALM presents the promise of introducing an autonomous decision-making agent that can be incorporated into company software systems, data pipelines and ALM processes for automated and faster implementation compared to traditional approaches. DRL ALM also holds the promise of rational, consistent and agile ALM implementation which is also able to learn from its own past actions and outcomes. Due to these strong advantages, there have been recent successes implementations by others to manage Asset Liability Management problems with Deep Learning and Reinforcement Learning (Englisch et al., 2023, Krabichler & Teichmann, 2023, Cheridito et al., 2020). Ultimately, the objective is to create a highly capable Artificial Intelligence (AI) Asset Liability Management assistant that can complement and increase the productivity of professionals in the Asset Liability Management field.

Section 2 formally outlines this paper's problem statement and research questions, that is, whether Reinforcement Learning can be applied to manage, fulfil and enhance ALM relative to the theoretical method of Redington Immunisation. Furthermore, set out to compare the performance of DRL ALM to a practical benchmark traditional ALM approach.

Section 3 discusses and synthesises existing papers and application of Reinforcement Learning in general and, more specifically, within the Insurance and Quantitative Finance domains. We also look at recent related ALM works and outline the existing gaps, which will be filled by the output of this paper.

Section 4 outlines and explains the theoretical framework of Redington Immunisation which underpins most traditional ALM.

Section 5 introduces and explains the Monte Carlo simulation processes used to generate data which was used to train and evaluate the Reinforcement Learning ALM Agent.

Section 6 introduces and explains the details Deep Reinforcement Learning framework and the training process.

In Section 7, we evaluate the performance of the Reinforcement Learning ALM and contrast it against the theoretical Immunisation and a practical benchmark traditional ALM approach. We conclude in Section 8 by summarising the main findings and future studies.

2. Problem statement and research questions

2.1. Problem statement

Implementing ALM using traditional ALM methods such as Redington Immunisation exposes an organisation to severe limitations and business risks such as limited automation of the ALM process, exposure

to risks of human irrationality and human error and constrained multiple multi-objectives optimisation. The problem statement of this paper was whether Reinforcement Learning can be used to successfully implement and automate ALM by Duration Matching to mitigate the problems and can effectively address the limitations of traditional methods?

Reinforcement Learning has been successfully applied in some other fields of Quantitative Finance. We explored whether this application can be extended to ALM primarily in the insurance context. Furthermore, we will explore whether Reinforcement Learning can sufficiently and practically improve the critical Risk Management objectives of a typical institutional risk-taker (such as a life insurer) relative to Immunisation.

2.2. Research questions

This paper presents investigations and findings guided by the following research questions:

(a) **Comparison of Deep Reinforcement Learning to theoretical Redington Immunisation**

The first research question was to investigate the following:

At a given point in time, do the Deep Reinforcement Learning ALM asset allocation results, and by extension, the Duration results, give similar outcomes to Redington Immunisation theory ALM? If they differ, to what extent do they differ?

For this research question, we focused on the universe of assets comprised primarily of Bonds, specifically Zero-Coupon Bonds, for simplicity. We also investigated the robustness of the Deep Reinforcement Learning solution under stress-testing.

(b) **Comparison of Deep Reinforcement Learning to a traditional benchmark ALM strategy**

The second research question was to investigate the following:

How do Deep Reinforcement Learning's asset allocations compare to those of a benchmark practical traditional ALM strategy in hedging a changing liability portfolio? Does it result in better interest rate hedging and to what extent?

For this research question, we tested and contrasted the daily asset allocation of Reinforcement Learning within a 30-day month.

2.3. Research limitations and focus

There are many Duration-based Asset Liability Management techniques. This paper focused on benchmarking Reinforcement Learning performance against the commonly used traditional Asset Liability Management approach of Duration Matching. We also limited our context to investments by long position investments within traditional hedging asset classes such as Bonds, Property and Cash or Cash-equivalents. In fact, for the experimentation, we focused only on Zero-Coupon Bonds for simplicity. However, this is not a very restrictive framework because Coupon-Bearing Bonds can be modelled as a series of Zero-Coupon Bonds with smaller face values and different maturities (Jarrow, 2004, Jarrow & Turnbull, 2000). Therefore, this paper's findings are also applicable when hedging is carried out with Coupon-Bearing Bonds.

We did not consider complicated investment strategies, such as those with short positions or derivatives, because the risk-taking institutions we considered in this paper are usually restricted from taking such risky investments. Although ALM is applied by many institutional risk takers such as insurers, banks, pension funds and asset managers, we mostly limited our scenario to that of an insurance company which sells everyday insurance products such as life, disability or health insurance.

3. Related work

3.1. Wider applications of deep learning and reinforcement learning

Reinforcement Learning has successfully been applied in a wide range of domains and applications. Deep Reinforcement Learning is often applied with the use of Deep Learning Models, such as Artificial

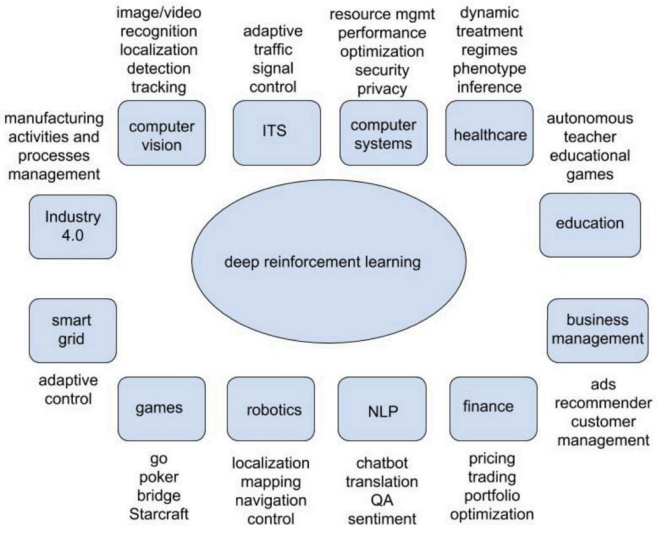

Fig. 1. Deep Reinforcement Learning Applications.

Neural Networks (ANNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) (Dong et al., 2021). When Deep Learning is used in Reinforcement Learning, this can be classified as Deep Reinforcement Learning (DLR) (Mousavi et al., 2018). Deep Learning models have also, in their own right, been applied successfully in a wide range of domains, such as Computer Vision, Speech Recognition, Natural Language Translation, Time Series Analysis, Autonomous Driving, Medical Diagnosis, among many other applications (Dong et al., 2021, He & Droppo, 2016, Hsu et al., 2016, Sak et al., 2014, Qu et al., 2017, Altché & de La Fortelle, 2017).

Fig. 1 illustrates a summary of some of the main application areas (Li, 2017, 2022). From Fig. 1, we can see the dynamic nature of Reinforcement Learning in very diverse fields.

3.2. Strengths and drawbacks of reinforcement learning (RL)

A key feature of model-free Reinforcement Learning (RL) is that one does not need to assume a model or impose underlying dynamics of financial relationships beforehand (Sato, 2019). This is incredibly advantageous in applications where there exist uncertainties on the truth of underlying dynamics or where the dynamics are consistently evolving. For example, this is the case for the investments of an insurer, and this model-free nature of RL gives flexibility and adaptability in such applications.

There has been extensive exploration of the various questions and challenges encountered in applying model-free RL in financial portfolio optimisation (Sato, 2019). In this paper, they explored popular RL approaches such as Q-Learning. Q-learning is a model-free RL approach aiming to maximise the total value of the expected reward or gain defined within the environment.

The performance of RL was noted to have several challenges, unfortunately. First, it is highly dependent on the appropriate specification of the objective value function, which is often challenging to specify while capturing the critical objectives. Second, the solutions are generally unstable if there is noise in the loss function. Third, RL is also susceptible to other issues such as over-fitting, the curse of high-dimensionality, dependence on many samples, interpretability and credit assignment problems. The credit assignment problem occurs when the consequences of a decision only materialise many iterations after the decision, which makes it difficult to attribute the decision to the outcome (Sato, 2019).

It has also been pointed out that Q-learning RL can sometimes be slow to converge, and alternative variations have been proposed (Devraj & Meyn, 2019).

This research has an advantage in that it prepares one for the theoretical and implementation challenges in the application. Unfortunately, the studies by Sato, Devraj and Meyn above did not focus on demonstrating detailed practical applications of RL, particularly in the insurance context.

3.3. Applications of reinforcement learning in quantitative finance

There has been some recent successful application of RL in other areas of Quantitative Finance. For example, there have been successful demonstrations of the application of RL to create an automated trading bot which made profitable actions on unseen test data (Hariom Tast & Lookabaugh, 2020). They also developed a Derivatives Option hedging strategy based on Deep Reinforcement Learning, which performed reasonably well. They also used a Reinforcement Learning-based portfolio allocation to maximise Risk-adjusted returns on Cryptocurrencies.

RL has also been applied in solving dynamic optimisation problems in Quantitative Finance, such as portfolio allocation, pricing and hedging contingent claims in model-free contexts (Kolm & Ritter, 2020, Dixon & Halperin, 2019). These applications show the flexibility of RL in various Quantitative Finance contexts. However, these applications were not applied in the context of ALM, which is where the gap in literature and application exists.

In Actuarial Science, there has been development of a Markov Decision Process model for a life insurer which was then successfully applied to RL algorithms to maximise the Risk-adjusted return on capital for the company (Abrate et al., 2021). This paper did incorporate the future liability cash flows of the insurer, but the objective was not to implement ALM based on Duration Matching. An advantage of this research is that the application was in the context of insurance companies, which is more aligned with this research. However, the objective was only to maximise Risk-adjusted returns without much regard for the liability profile of the insured liabilities. This also represents a gap in applying ALM to decide the strategy to match the liabilities' Duration.

Wuthrich and Merz introduced Deep learning as one of the important methodologies in the toolkit of actuarial practice and statistical modelling (Wutrich & Merz, 2023). However, there is no inclusion of Reinforcement Learning or application to Asset Liability Modelling.

3.4. Recent applications of deep learning and reinforcement learning to asset liability modelling

Cheridito et al. successfully applied a Deep Learning approach to estimate the tail risk measures of a portfolio of assets and liabilities (Cheridito et al., 2020). The tail risk measures used for assessing Asset Liability risk included Value-at-Risk and Expected Shortfall in order to ensure regulatory compliance with Solvency II (Cheridito et al., 2020). Although there was successful implementation of Deep Learning in Asset Liability modelling, the solution is static compared to a situation where Reinforcement Learning is used. The solution is not able to continuously learn over time and adapt to changing market and risk conditions as is enabled by Reinforcement Learning.

Krabichler and Teichmann recently successfully applied Deep Learning and Reinforcement Learning to develop a methodology for hedging interest rate risk of asset and liability portfolios for retail banks (Krabichler & Teichmann, 2023). They applied the method of *deep hedging* to a situation of a runoff (no going concern) portfolio of retail banking liabilities. The deep learning ALM application demonstrated an out-performance of 2% per annum on a return on equity basis. The Tensorflow library was applied in the Deep Learning implementation (Krabichler & Teichmann, 2023). Three main drawbacks of the Deep Learning approach were noted. The first was the need to overcome a common misunderstanding that the approach requires a large volume of historical data. The second was that the agent can be seen as black-blox which is difficult to decipher. The last was that there may be a need to convince regulators of the advantages, robustness and accuracy of this novel approach.

Englisch et al. used a similar approach of *deep hedging* also in a similar retail banking context but compare the application to more benchmark strategies in a comprehensive manner (Englisch et al., 2023). In addition, they also explicitly modelled yield curves in their scenario generation. They also introduced regulatory constraints such as Liquidity Coverage Ratio to the problem. The results showed that Deep Learning-based ALM outperformed all the benchmarks whilst staying within regulatory limits in almost all yield curve scenarios. However, several challenges were noted such as the difficulty of model explainability of the dynamic decision-making agent and the difficulty of appropriately defining loss functions that are preferences of an ALM strategy. There was also an extended application to unwinding Swap portfolios, however, with some mixed results (Englisch et al., 2023).

3.5. Research gaps addressed

From the papers discussed it is evident that Reinforcement Learning has been applied with some success in Quantitative Finance and Actuarial Science (Hariom Tast & Lookabaugh, 2020, Kolm & Ritter, 2020, Dixon & Halperin, 2019, Abrate et al., 2021). Although the recent application of Deep Learning and Reinforcement Learning for Asset Liability Modelling was successful in banking it was based on approaches customised specifically for retail banking (Krabichler & Teichmann, 2023, Englisch et al., 2023).

There is a gap in that there haven't been well-documented Deep Reinforcement Learning applications to Asset Liability Management in general. That is, an application that can replicate and improve on the well-established approach of Redington Immunisation theory applicable in all Actuarial and Quantitative Fiance spheres.

ALM implementation is more complicated compared to other Quantitative Finance applications, such as return maximisation, because the observed data for which optimisation is sought is always multi-dimensional. Furthermore, ALM requires one to consider patterns in both assets and liability data for a relatively complicated objective compared to the other problems. Duration matching assets to liabilities is a slightly more complex objective than maximising return, for example.

4. Traditional asset liability management (ALM)

4.1. Traditional immunisation theoretical framework

The traditional ALM approach is based on attempting to satisfy Redington's three conditions (Redington, 1952). The first condition is

$$A = L\,, \tag{1}$$

where $A = \int_0^\infty A_t e^{-rt} dt$ and $L = \int_0^\infty L_t e^{-rt} dt$, A_t and L_t are the asset cash flow and liability cash flow at time t, respectively, and r the constant continuously compounded discounted rate.

The second condition is

$$\frac{\partial A}{\partial r} = \frac{\partial L}{\partial r}\,, \tag{2}$$

which means that the asset sensitivity (Asset Duration) to interest rates must be the same as the Liability Value sensitivity (Liability Duration). This Duration is also known as the Macaulay Duration.

The third condition is

$$\frac{\partial^2 A}{\partial r^2} \geq \frac{\partial^2 L}{\partial r^2}\,, \tag{3}$$

and means that the asset convexity should exceed the liability convexity.

Redington immunisation theory assumes that the yield curve is flat and that when interest rates change, the changes are parallel shifts down or up of the yield curve (Kahlig, 2022).

In deriving the solutions which satisfy Redington Immunisation, one usually solves for the asset allocation that satisfies the first two conditions by solving for the arising simultaneous equations. From the set of possible solutions, one then verifies or selects (if there is more than one potential solution) the one which satisfies the third condition by plugging in the solution in the second derivative (Garrett, 2013).

If the second and third condition are both satisfied, then we would have a perfect matching, and we know that either the Present Value of assets (A) will increase by more than that of liabilities (L) if the interest rate decreases; or the Present Value of assets (A) will decrease by less than that of liabilities (L) if the interest rate increases. In practice, it is not always the case that one can always satisfy or prove Condition 3. This would be an imperfect interest rate hedge. However, even if only Condition 1 and 2 are met, the resulting asset value movements would move more or less in line with liabilities - this would still be much better interest rate protection than a situation where there is none at all. Therefore, we will focus on explicitly deriving for Conditions 1 and 2 in the next section (Section 4.2).

Due to the underlying assumptions of Redington's Immunisation outlined above, there are several challenges and limitations to implementing it in the real world (Kahlig, 2022). These include:

- Interest rates are not flat in most real-world cases. Since the underlying model assumes a constant interest rate, one must carefully choose an appropriate constant interest rate for all calculations.
- Immunisation only provides hedging against small changes in the interest rate. If changes are large, the hedge is imperfect.
- Immunisation requires frequent re-balancing of the asset weights to keep the Duration of the assets to be the same (or close enough) to that of the liabilities. This is especially challenging if the market conditions do not easily correspond to the underlying model assumptions and significant judgement is required in the re-balancing.
- It is not always possible to know the exact timing of asset and liability cash flows as is assumed by the Immunisation theory. In our problem, we assume these are known for simplification, but in reality, one may need to apply estimations or probability distribution assumptions.
- Assets of required maturity to achieve Immunisation may not exist in the market. A solution for this is not provided in the vanilla approach.

These limitations require that when implemented in practice, the user of the model should constantly monitor the results and conditions (e.g. the market yield curve, market volatility, etc.). If the requirements differ significantly from the underlying assumptions, then the user should make judgement adjustments. This is one of the significant limitations of the traditional approach in that it requires constant human supervision and application of adjustments to the raw results.

4.2. Implementation of traditional immunisation

The first two of Redington's conditions can be summarized as the following system of two equations:

(1) Present Value (PV) of Assets = Present Value of Liabilities;
(2) Duration of Assets = Duration of Liabilities.

In a scenario with two Zero-Coupon Bonds, Z_1 and Z_2, the above system of equations translates to the following:

$$(1)\ \ \omega_1 PV(Z_1) + \omega_2 PV(Z_2) = PV(Liability)$$
$$(2)\ \ \omega_1 Duration(Z_1) + \omega_2 Duration(Z_2) = Duration(Liability)$$

The Duration of the asset portfolio is the weighted Duration of the respective assets. ω_1 and ω_2 are the respective weights which add to 1.

In particular practical circumstances, is possible that the data inputs may not always meet the requirement that the coefficient matrix of this system be invertible, which presents challenges to using this traditional approach to ALM. Other challenges and limitations of the conventional approach were provided in Section 4.1.

In a Python program, the solution for $W^* = [\omega_1\ \omega_2]$ for the traditional approach was executed using the `numpy` library's Linear Algebra solvers with the `numpy.linalg.solve(M,E)` command.

5. Monte Carlo simulation of asset liability management environment

5.1. Monte Carlo data generation (scenario generation)

To generate some training data for RL, we simulated 10 000 Monte Carlo scenarios. We used 10 000 simulations because they are sufficient enough to be associated with acceptable changes in errors of resultant estimates (1% or less). Furthermore, this number of simulations is the level at which the RL agent would be sufficiently trained.

In each scenario, we assumed the data to be for five years or 60 months, that is, 60 time points. In each of the 10 000 scenarios, there were a liability Duration value and Bond term-to-maturity simulation values at all times. We simulated the data which can arise from the environment, such as that of an Insurance company or Bank. In these contexts, there is some randomness in the change of the Liability Duration from one-time step to the next.

One of the critical advantages of RL is that one can train the Agent in an environment based on simulated or synthetic data and still achieve successful real-world applications. This approach means that RL can be applied in a manner that saves cost and time by not requiring development and training on real-world data. Many successful applications of RL for even tricky tasks, such as in Autonomous Driving, are trained primarily from a simulated environment (Osiński et al., 2020).

5.2. Liability duration simulation

For a risk-taker such as an insurance company or pension fund, the Liability Duration increases when the institution covers a new risk which is expected to claim later than the current risk pool's weighted Liability Duration. The opposite is also true. Alternatively, the Liability Duration will reduce if shorter Duration insurance policies lapse from the insurance risk pool.

The movements in Liability Duration from one point to the next are generally random because there is randomness in the nature of the trends due to new business, lapses (terminations or expiry) and the claims that the risk taker experiences between time points. The process for generating the *ith* simulation's Liability Duration and reflecting its randomness is given in Equation (4):

$$D_{it} = D_{i,t-1} + \Gamma_{it} \times \delta_{it}\,, \tag{4}$$

where:

- D_{it} is the Liability Duration for the *ith* simulation at time $i \in \{1, 2, .., N\}$ and $t \in \{1, 2, .., T\}$; where the number of simulations (N) is 10 000 and the number of time steps is (T) is 60 months (5 years).
- $D_{i,t-1}$ is the Liability Duration for at the previous time point $t-1$.
- Γ_{it} is a Binomial distributed variable with two possible values +1 or -1, with probabilities P_t and $1-P_t$, respectively.
- P_t is randomly sampled from a Uniform Distribution with a range of [0,1], that is, $P_t \sim U(0,1)$. Therefore, P_t determines the probability by which the liability Duration increases or decreases.
- δ_{it} is the magnitude of the absolute movement in the Liability Duration, which is randomly sampled from a Uniform Distribution with a range of $[0\,;\,\Delta]$, that is, $\delta_{it} \sim U(0,\Delta)$

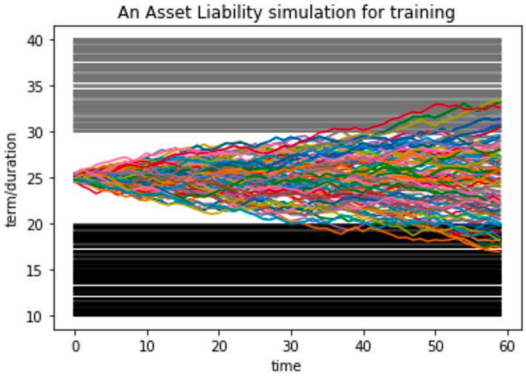

Fig. 2. An example of a set of Monte Carlo Simulations.

- Δ is the maximum possible change from one-time step to the next and is set at 0.5 years in the training data. Therefore, for this training data simulation, we assumed the Liability Duration could change by an absolute value of up to 0.5 years from one step to the next.

5.3. Bond term-to-maturity simulation

We assumed that the liabilities would be matched primarily by Zero-Coupon Bonds. In the simplified version of our problem, two Bonds are available in the market - (1) a short-dated Zero-Coupon Bond and; (2) a long-dated Zero-Coupon Bond. For each simulation i we randomly sampled the term of the short-dated Bond from a range of Uniform distributions with a range between 10 and 20 years. For the long-dated Bond from a range of Uniform distribution with a range between 30 to 40 years. The Bonds' terms are defined as:

- $T(Z_1)_i$ for first Zero-Coupon Bond term for *ith* simulation. Sampled from a uniform distribution of parameters (10 years, 20 years), that is, $T(Z_1)_i \sim U(10, 20)$
- $T(Z_2)_i$ for second Zero-Coupon Bond term for *ith* simulation. Sampled from a uniform distribution of parameters (30 years, 40 years), that is, $T(Z_2)_i \sim U(30, 40)$

For a given simulation, once sampled, the term to maturity of the short and long Bonds was assumed to remain the same at all time points. We assume that the risk-taker can find Zero-Coupon Bonds of simulations' given maturities in the market at all times throughout that given simulation. In this research question, we are primarily interested in checking the solution Bond allocations as liability Duration changes. We are not mainly interested in the Bonds' term-to-maturity variability but in the Liability Duration and its impact on the asset allocation for Duration Matching.

We assumed two Bonds because a multiple (P) Bond allocation task can be represented as a two-Bond allocation task by grouping all the other P - 1 as one notional combined Bond. Hence, this conceptual Bond would be a weighted term of these P - 1 Bonds. Therefore, to test the viability of Reinforcement Learning to ALM, it is sufficient to start with a two-Bond scenario to assess viability.

5.4. Visualisation of Monte Carlo simulations

A visualisation example showing a subset of typical simulation results from scenario generators in Sections 5.2 and 5.3 are shown in Fig. 2 above.

The lines in colour represent pathways of the evolution of liability over time generated from Section 5.2 with each pathway representing one scenario. There are many scenarios, each with a different pattern.

The horizontal lines in colour represent the Maturity terms of different Zero-Coupon bonds over time generated from Section 5.3 with each line representing one scenario. The grey lines represent the long-duration bonds and the black lines represent short-duration bonds. For a given scenario, a pair of lines is generated, one grey and one black.

6. Reinforcement learning (RL) asset liability management

6.1. Deep reinforcement learning framework components

In this section, we discuss the key components of the RL model, namely, the Environment, States, Actions, Agent and Reward Function. These components are necessary to define a RL model to be successful (Hariom Tast & Lookabaugh, 2020).

These components allow for the essence of RL, which is to learn through interactions (Arulkumaran et al., 2017). Hence, these components are interlinked and related.

6.1.1. Reinforcement learning environment

The first component to consider is the Environment. The Environment represents the operating environment where a risk-taking institution is exposed to both assets and liabilities, such as an insurance company, pension fund, bank, or asset manager. For simplicity, one can regard the standard environment to be that a life insurance company write contracts that payout a benefit on death.

6.1.2. Reinforcement learning states

The second component to consider are the States (S) of the process, which are the key metrics which can be observed within the environment by the Agent (Explained in Section 6.1.4) and used for decision making. At any given time, the State is the current term to maturities for the securities and also the Liability Duration. In the two Zero-Coupon Bond situations we have, this means we can observe the two asset terms ($T(Z_1)_t$ and $T(Z_2)_t$) and the liability Duration (D_t). This corresponds to a 3-Dimensional state problem, which requires additional considerations compared to a 1-Dimensional state problem. The history of Liability Duration is also observable and is an essential factor for solving the problem; hence, this is also an additional state.

6.1.3. Reinforcement learning actions

The third component to be considered for RL is the actions which can be taken. This represents the actions the Agent (Explained in Section 6.1.4) can take at time t, generally represented as a_t. In this context, the actions are the weight allocations to respective Zero-coupon Bonds for Duration matching. Possible actions are the weights of security i's allocation in the assets portfolio ω_i, for $i \in \{1, 2, .., P\}$ where there are P assets. For the two asset problem we have ω_1 and ω_2, that is, $P = 2$.

We are not considering short-selling; therefore, for a successful solution, we will require the weights to range between zero (0) and one (1). Therefore, these actions are continuous in nature and not discrete. We are only considering long positions because regulations generally bar the institutions such as insurance companies from taking short positions. Furthermore, for ALM, short positions are usually unsuitable for that risk-management objective.

6.1.4. Reinforcement learning agent

The Agent is the autonomous entity which performs the Actions and acts within the defined Environment. In addition, the Agent observes the outcomes of specific actions through feedback from the Environment - the input is in the form of a Reward Function (Explained in Section 6.1.5). The Agent's objective is to learn the behaviour or actions a_t which leads to desired outcomes for a given state S. This optimal behaviour is known as the policy π (Arulkumaran et al., 2017). The Agent learns the policy through a combination of trial and error and by observing the historical correlations and patterns between states, actions and rewards.

The policy π is a mapping from the states to actions. However, due to the inherent uncertainty, there is a probability estimation of the best

actions. Therefore, the policy π maps States (S) to probability distribution over all possible actions (A) (Arulkumaran et al., 2017). This is expressed in Equation (5):

$$\pi : S \longrightarrow p(A = a|S) . \tag{5}$$

The agent will take action with the highest probability at a given time (Arulkumaran et al., 2017). Often the Agent relies on Machine Learning, as is the case in this paper, to decipher the correlations and complex patterns between states, actions and rewards. This is the case in this paper, where the Agent uses Deep Learning to determine the policy. Hence, this problem can be called Deep Reinforcement Learning as we combine Reinforcement Learning and Deep Learning. More on the Deep Learning techniques used is given in Section 6.2.

6.1.5. Reinforcement learning reward function

A very important component of Reinforcement Learning is the Reward Function. The Reward Function is the feedback sent by the environment to evaluate the last action by the Agent. An appropriate reward might be the inverse of the absolute difference between the Asset Duration and the Liability Duration.

The objective of Reinforcement Learning optimisation is to determine the best Agent's policy to maximise the reward function. This is equivalent to minimising a penalty function or an error function. The mismatch error between the asset Duration and Liability Duration for a given scenario i and at a given time point t for this research question is given by Equation (6):

$$e_{it} = \omega_{1it} T(Z_1)_{it} + \omega_{2it} T(Z_2)_{it} - D_{it} . \tag{6}$$

For a given scenario i, we square and sum the Duration Mismatch Errors at all time points to get the scenario error as represented by Equation (7):

$$Simulation\ Sum\ Square\ Errors\ (SSE_i) = \sum_{t=1}^{T} e_{it}^2 . \tag{7}$$

We square the errors because, in the standard Duration Matching problem, we want to minimise the Duration absolute difference without regard to whether the asset Duration is smaller or not. The (SSE_i) function will be the feedback to the Agent on how well its current learnt policy, from a combination of past learning and trial and error, performed on the scenario. The Agent will implement more policy elements that reduce the error and fewer of those that increase the error. Hence, there is a reinforcement of appropriate policy elements.

In practice, the policy is not updated after every scenario but after a set of scenarios called a batch. This is because one scenario will not be sufficient grounds to update a policy because of limited data. If there are, N simulations and each batch is of size B scenarios, there will be N/B batches. The Batch SSE will be used to adjust the Agent's policy iteratively and this is given by Equation (8):

$$Batch\ Sum\ Square\ Errors\ (SSE) = \sum_{i \in Batch} \sum_{t=1}^{T} e_{it}^2 . \tag{8}$$

The range of scenarios in the first batch will be from 1 to B; for the second batch, they will be B+1 to 2B; for the third batch will be from 2B+1 to 3B, and so forth. The order in which the batches are shown to the Agent should not have any interdependence. If each scenario is independent of the next, as is the case in this data, then this condition is automatically met.

The expectation is that during the Agent's learning or training phase, the Batch SSE will reduce over time, signalling an improvement in the Agent's ability to implement the ALM. The Agent learning process is explained further in Section 6.2.3.

6.2. Reinforcement learning asset liability management implementation

This section explains how we practically implemented the Policy Search Reinforcement Learning.

In our problem, we apply the Policy Search approach to Reinforcement Learning as it is the most appropriate. In our situation, the actions are continuous as they are asset allocation weights. Furthermore, it is difficult to pre-determine the values (rewards) of specific actions for a given state because of the continuous nature of the activities, the vastness of the state space, and the nature of the problem. In such a problem, it is best that the autonomous Agent finds its optimal policy directly by exploring various behaviours and reinforcing those that perform well concerning the reward function (Sigaud & Stulp, 2019).

6.2.1. ALM agent specification in object orientated programming (OOP) in Python programming

As discussed in Section 6.1.4 the Agent represents the entity that performs the actions within the environment, that is, the entity which makes the asset allocations for ALM. We set up the Agent using Object-Oriented-Programming (OOP) or Class-based programming in Python. First, we define an agent class which is then used to create OOP objects which have certain attributes and methods.

The agent class has various capabilities arising from its methods. The agent class has the following methods:

- The constructor function which defines an `_init_` function, which in turn defines the Reinforcement Learning model parameters such as the time steps, batch size (B), number of environment features and some parameters for the Deep Learning model.The constructor defines a computational graph in TensorFlow, which captures the logic of the data flow, the reward function (as specified in Equation (8)) and also the Deep Learning model (as specified in Section 6.2.2).
- The batch trainer module defines a key function which trains the Agent object. This function goes through one pass of the data (one epoch) and trains the Deep Learning network based (explained in Section 6.2.2) on practical experience from trial and error and currently learnt policy. The training is, however, done in batches of the data
- The training module defines a function which calls the batch trainer module function over many passes (epochs) of the data in the training process.
- The predict module uses the trained model object to predict a given set of data or states. This function returns the action (asset allocation weights) for a given state (asset terms and liability Duration)
- The restore module restores the saved trained reinforcement learning models, which can be used for predictions or even additional training.

TensorFlow is a powerful Open-Source library developed by Google which makes the development and deployment of machine learning models more accessible, faster and scalable (Tensorflow.org, 2022). It is especially advantageous to use TensorFlow in Deep Learning because Deep Learning models are computationally intensive. TensorFlow's efficiencies derive from its graph database structure which is a network of interconnected computational nodes (Tensors). Tensors are multi-dimensional arrays which are used to represent the data, its subsequent transformations and computational results (Lang, 2022). How the data is represented differs from the standard relational format of data.

The computational graph structure is more efficient when the data and computations increase because they inherently store the relationships in the data together with the data itself (Pang et al., 2020). In using TensorFlow, one needs to not only learn the TensorFlow syntax but also clearly define all the computational relationships in the process. TensorFlow also requires one to specify the tensor dimensions accurately, data types and formats in a precise and consistent manner

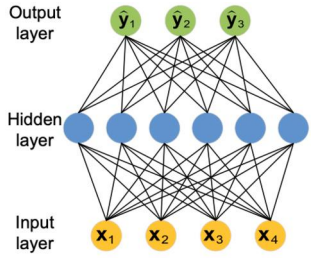

Fig. 3. An illustration of a Simple Artificial Neural Network (ANN).

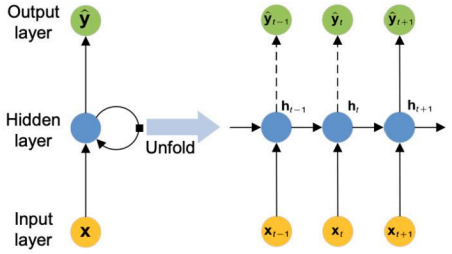

Fig. 4. An illustration of a Recurrent Neural Network (RNN).

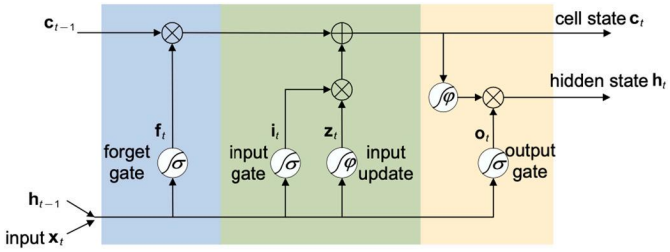

Fig. 5. An illustration of a Long-Short Term Memory (LSTM-RNN) cell.

when defining the graph relationships. If this is done correctly, TensorFlow achieves efficiency from its ability to execute operations in low-level and efficient C++ code on Central Processing Units (CPUs), more powerful Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs). When implemented in a Python programme, TensorFlow is used in conjunction with the `numpy` and `keras` libraries (Geron, 2019).

The hardware we used to run the software was the Apple MacBook Air 2020 model with 512 Gigabytes of Solid-State Drive (SSD) storage capacity and running on the *macOS* Operating System. The processing was implemented by this model's standard Apple M1 chip with 8 Gigabytes of Random Access Memory (RAM). This processor has an integrated 8-core Central Processing Unit (CPU) and an 8-core Graphics Processing Unit Unit (GPU) (Apple, 2023).

6.2.2. Deep learning with recurrent neural networks (RNNs)

In the direct Policy Search approach, the Agent uses an Artificial Neural Network (ANN) to map the state to action. A Neural Network provides Agent with a mechanism to pick out complex relationships within the mapping from state to action which improves the reward (Arulkumaran et al., 2017). A typical simple vanilla FeedForward Artificial Neural Network (ANN) is shown in Fig. 3 (Hua et al., 2019).

In the above, information and processing flow in one direction from inputs, through the layers and to output; hence, they are referred to as *FeedForward*. Such a traditional ANN assumes that the next information and outputs are independent of the previous. However, in this problem, the Liability Duration at time t has some correlation to the level and trend of the Liability Duration at times before t. There are also correlations in the actions over time because of the correlation in the states. Moreover, there is *path dependence* and time-series properties in the states and actions. Therefore, a traditional ANN is not best suited for this problem (Hariom Tast & Lookabaugh, 2020). In the experimentation we carried out in the paper we confirmed that use of a general Artificial Neural Network yielded poor results, which confirmed the need for an improved learning method.

A Recurrent Neural Network (RNN) is a special type of Artificial Neural Network that can capture time-dependent relationships in the states and actions. Therefore, an RNN can factor in the historical information and outputs for the following outputs and is more suited for this problem. An RNN has a time dimension to it where it uses its previous output as input in its current calculation. It also has a dynamic internal state which persists throughout time and uses that to determine the time-dependent predictions (Staudemeyer & Morris, 2019). A typical simple Recurrent Neural Network is shown in Fig. 4.

The right-hand side of Fig. 4 shows the RNN in expanded format. This shows the RNNs calculate the current output ($\hat{y}_t$) from the existing data (x_t) and the hidden state (output) of the previous iteration (h_{t-1}) (Hua et al., 2019).

Standard Recurrent Neural Networks (RNNs) have limitations in that they cannot detect relationships of more than ten time steps apart (Staudemeyer & Morris, 2019). Long-Short-Term-Memory Recurrent Neural Networks (LSTM-RNN) are a special type of Recurrent Neural Network (RNN) which can learn longer-term time-dependent relationships (Smagulova & James, 2019). LSTM-RNN can pick up time-dependent relationships of up to 1 000 time steps apart and hence are much more dynamic (Staudemeyer & Morris, 2019). LSTM-RNNs have been successful in a wide range of domains which have temporal or sequential data, such as Natural Language Processing (NLP) (He & Droppo, 2016, Hsu et al., 2016, Sak et al., 2014, Qu et al., 2017), Autonomous Driving (Altché & de La Fortelle, 2017) and time-series analysis techniques (Mallinar & Rosset, 2018). LSTM-RNN networks are enhanced by the incorporation of memory blocks (cells) with the ability to better persist relevant information for the process. There have been many versions of the LSTM-RNN since the first version which was introduced by Hochreiter and Schmidhuber (Hochreiter & Schmidhuber, 1997). However, a typical memory block (cell) of an LSTM-RNN in shown in Fig. 5.

The above memory is also recurrent (as it is still an RNN) - the outputs (c_t and h_t) are used in the next iteration at $t+1$ and so forth. The memory block achieves its temporal memory capabilities from the three main modules and gates within it (Hua et al., 2019): (1) The Forget Gate control how much of the information passed from the previous cell is irrelevant and discarded in the current cell, (2) The Input Gate controls how much new information which wasn't present in the previous cell is captured in the current cell, and (3) The Output Gate controls which of the information in the cell is used to determine the block's output, which is then passed to the following stages of the RNN.

We also use an LSTM-RNN with multiple cells and layers, and hence, we categorise this as a Deep Neural Network. In this situation, we used an LSTM-RNN with three hidden layers. The first hidden layer had 62 neurons. The second and third hidden layers have 46 neurons each. The combination of Deep Learning with Reinforcement Learning makes this application a Deep Reinforcement Learning (DRL) application. The training of the Agent is done in batches of the training data. Within each batch of simulations, the Agent applies a combination of exploration and past learnings to output weights and their respective rewards. At the end of each batch, the relationships between the states and rewards are persisted and fed through the Recurrent Neural Network, which finds the relationship between the state profiles and the rewards by optimising for its own weights and biases. Gradient Descent optimisation methods are used to train and determine the optimal weights and biases of the Recurrent Neural Network. In order to find the optimal weights for any neural network in general, the Gradient Descent Algorithm (or similar) is used to update the weights iteratively. This method updates the weights in the direction which results in the most significant reduction in the value of the error function (or surface) towards

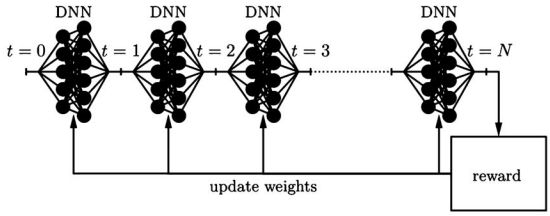

Fig. 6. A simple representation of Recurrent Neural Network and Reinforcement Learning architecture.

the global minimum (Staudemeyer & Morris, 2019). This involves determining partial derivatives of the weights (parameters) of individual neurons or cells concerning the respective weights (parameters). The impact of some of these is indirect in cases where there are multiple layers, and this requires the chain rule to determine the impact of weights which are more removed away from the output. This approach is commonly known as Back-propagation.

Optimisation of LSTM-RNN follows the same philosophy but tackles the time dimension challenge, which is not present in other Neural Networks. Here the method used is called Back Propagation Through Time (BPTT) (Sherstinsky, 2020). If R represents the reward function (or objective function more generally) and Θ represents all the parameters of the LSTM-RNN, then the approaches rely on the calculations of the partial derivatives $\frac{\partial R}{\partial \Theta}$. For many parameters, the chain rule will be repeatedly applied to determine the partial derivatives, as they will not have a direct link to the objective function. This approach requires that the Activation Functions (functions which compute outputs) used at each neuron or cell be differentiable or the derivative be estimable with numerical methods (Sherstinsky, 2020, Staudemeyer & Morris, 2019). After each update, the LSTM-RNN is used to predict the output. The differences between the predicted output from the actual are the errors which are used to carry out another round of updates to the parameters Θ. This process is repeated many times for a specified number of times or, alternatively, until an acceptable level of error is achieved. This iterative process and Deep Learning architecture is illustrated in a simple manner in Fig. 6 (Krabichler & Teichmann, 2023).

Within our Deep Reinforcement Learning application, the training data set is divided into batches of 1 000 simulations in each batch. So if there are 10 000 simulations in total, there would be ten batches, with each batch having 1 000 simulations. The reason is that we want to update the Agent's learnt policy π after having a sufficient level of information (the 1 000 simulations per batch). We also want to allow the Agent to apply the learnt policy to a new set of simulations and verify and improve its policy. Therefore, the LSTM-RNN weights and biases are updated, and this represents the updating of the policy π of the Reinforcement Learning Agent.

6.2.3. *The deep reinforcement learning agent training process*

The main steps at a high level are outlined in Algorithm 1.

The algorithm summarises many of the granular and intricate steps that are taken in implementing Deep Reinforcement Learning for ALM. Epochs refer to the number of times the entire data set of 10 000 simulations is cycled through during training.

We trained the Reinforcement Learning Agent on the training data simulations. The training process takes about one hour in total on a data set of 10 000 simulations, with batches of 1 000 and 100 epochs. The trained Agent object was saved in memory for retrieval in the testing stages using the restore function of the Agent class.

The training process progress was captured by the evolution of the Batch Sum Square Error in Fig. 7.

We can see the exponential reduction in the batch mean SSE over time as the Agent learns the optimal policy for the given data. After each iteration, the Agent reinforces its behaviour or policy π towards actions that are more correlated with minimising the Sum Square Error (SSE).

Algorithm 1 Main Steps for implementation of Deep Reinforcement Learning for Asset Liability Management.

1: Create the Agent class (see section 6.2.1), with its methods including the batch trainer module, training function, prediction function and the restore function
2: Use the Agent class to create an Agent object (which includes the creation of a computational graph for use in TensorFlow)
3: Launch a TensorFlow session using `tf.Session`
4: Specify the parameters such as batch size (100), number of features (3), number of training epochs (100) also the name of the model for records
5: Call the training function which triggers the batch trainer module
6: **repeat**
7: **for all** batches in each epoch **do**
8: Determine the indices of the data that corresponds to that batch
9: Run the TensorFlow computational graph (from Step 2) on the batch data simulations
10: Let the Agent apply existing policy and exploration on the batch
11: Record the States, Actions (Asset allocations) and Rewards for that batch
12: Update the weights of the LSTM-RNN based on Gradient Descent (as in Section 6.2.2)
13: Update the Agent's policy π in line with LSTM-RNN update
14: Update the policy which will be applied in the next batch
15: Print the time taken for the batch processing and print the Batch SSE
16: **end for**
17: **until** All epochs are complete
18: Save the trained Agent object

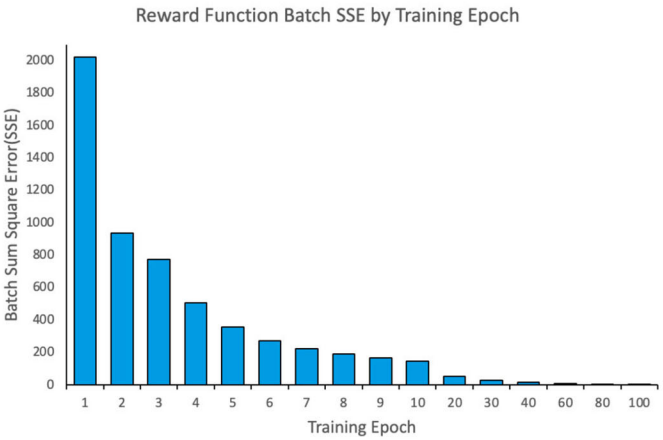

Fig. 7. Reward Function Batch SSE by Training Epoch.

By the 100th epoch, the agent's asset allocations manage to result in asset allocations that give an Asset Duration very close to the Liability Duration in the training data.

For proper testing, however, we will analyse the performance of the Deep Reinforcement Learning Agent on unseen test data in the next sections. In the next section, we will also compare the results of the Reinforcement Learning agent to that of the traditional ALM.

7. Results - evaluation and comparison

7.1. *Comparison of deep reinforcement learning ALM to theoretical Redington immunisation*

In this section, we compare the performance results of the Reinforcement Learning ALM and contrast it against theoretical Immunisation. In Section 7.1.1 and Section 7.1.2 we first evaluate the primary research question, which was: At a given point in time, do the Deep Reinforcement Learning ALM asset allocation results, and by extension, the Duration results, give similar outcomes to Redington Immunisation theory ALM? If they differ, to what extent do they differ?

In Section 7.1.3 we also evaluated the performance of the Reinforcement Learning approach under stress conditions with much higher volatility than trained on. In this section, we discuss some of the interesting key findings from the Stress Testing, which we implemented on the Reinforcement Learning ALM Agent. The purpose of the Stress

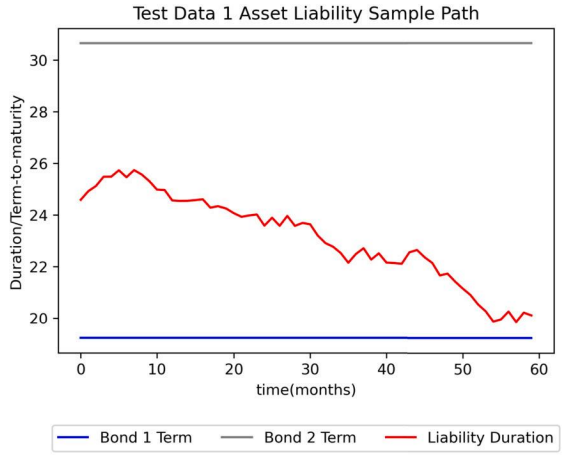

Fig. 8. Monte Carlo Simulation Testing Sample path.

Table 1
Test Sample Path Values at Annual Time-points (months).

Time-points	12	24	36	48	60
Liability Duration	24.97	24.02	22.15	21.65	20.10
Bond 1 Term	19.22	19.22	19.22	19.22	19.22
Bond 2 Term	30.66	30.66	30.66	30.66	30.66

Table 2
Test Sample Path Annual Time-points (months) Redington Immunisation Allocations.

Time-points	12	24	36	48	60
Liability Duration	24.97	24.02	22.15	21.65	20.10
Bond 1 Term	19.22	19.22	19.22	19.22	19.22
Bond 1 Allocation	49.7%	58.0%	74.4%	78.8%	92.3%
Bond 2 Term	30.66	30.66	30.66	30.66	30.66
Bond 2 Allocation	50.3%	42.0%	25.6%	21.2%	7.7%

Testing was to do further experimentation on the flexibility of Reinforcement Learning to different operating conditions.

7.1.1. Sample case evaluation

We generated 1 000 test scenarios using the same approach and parameters used to create the training data, that is, the simulation processes outlined in Section 5.1. We used this as our standard testing data (Test Data 1). An example of a path from the test data is shown in Fig. 8.

In the example above, we will be interested in whether Reinforcement Learning can give appropriate Duration Matching asset allocations at a given time point. We used five time points, that is, at 12 months, 24 months, 36 months, 48 months and 60 months as test time points. The Bond Maturities and Liability Duration at these time points are given in Table 1.

We first applied the Redington Immunisation ALM on the sample Test 1 Data, and we got the traditional asset allocations in Table 2.

We applied the Redington Immunisation approximation solution derived in Section 4.2. A function which executed this formula was defined in Python.

We then also applied the Deep Reinforcement Learning ALM outlined in Section 6.2.3 on the sample Test 1 Data. The saved trained Agent object was retrieved memory using the restore function of the Agent class and ran on the test data. We got the Reinforcement Learning asset allocations shown in Table 3.

One can already observe that the asset allocations are relatively close to each other for the sample scenario between the Redington Immunisation's allocations and the Reinforcement Learning allocations.

Based on its respective asset allocations to Zero-Coupon Bond 1 and Zero-Coupon Bond 2, we then calculated the traditional ALM method's

Table 3
Test Sample Path Annual Time-points (months) Deep Reinforcement Learning Allocations.

Timepoints	12	24	36	48	60
Liability Duration	24.97	24.02	22.15	21.65	20.10
Bond 1 Term	19.22	19.22	19.22	19.22	19.22
Bond 1 Allocation	49.2%	58.1%	72.8%	76.7%	93.7%
Bond 2 Term	30.66	30.66	30.66	30.66	30.66
Bond 2 Allocation	50.8%	41.9%	27.2%	23.3%	6.3%

outcomes of the Asset Duration outcomes using the formula shown by Equation (9):

$$D(Trad_{it}) = \omega_{1it} T(Z_1)_{it} + \omega_{2it} T(Z_2)_{it}\,, \tag{9}$$

where:

- $D(Trad_{it})$ is the Redington Immunisation asset Duration at time t for this *ith* test scenario.
- ω_{1it} is the derived Redington Immunisation asset allocation for Zero-Coupon Bond 1 time at time t for this *ith* test scenario.
- ω_{2it} is the derived Redington Immunisation asset allocation for Zero-Coupon Bond 2 time at time t for this *ith* test scenario.
- $T(Z_1)_{it}$ is the observed term-to-maturity for Zero-Coupon Bond 1 time at time t for this *ith* test scenario.
- $T(Z_2)_{it}$ is the observed term-to-maturity for Zero-Coupon Bond 2 time at time t for this *ith* test scenario.

Based on its respective asset allocations to Zero-Coupon Bond 1 and Zero-Coupon Bond 2, we also calculated the Reinforcement Learning model's asset Duration outcomes using the formula shown by Equation (10):

$$D(RL_{it}) = \omega_{it} T(Z_1)_{it} + \omega_{it} T(Z_2)_{it}\,, \tag{10}$$

where:

- $D(RL_{it})$ is the Reinforcement Learning asset Duration at time t for this *ith* test scenario.
- ω_{1it} is the derived Deep Reinforcement Learning asset allocation for Zero-Coupon Bond 1 time at time t for this *ith* test scenario.
- ω_{2it} is the derived Deep Reinforcement Learning asset allocation for Zero-Coupon Bond 2 time at time t for this *ith* test scenario.
- $T(Z_1)_{it}$ is the observed term-to-maturity for Zero-Coupon Bond 1 time at time t for this *ith* test scenario.
- $T(Z_2)_{it}$ is the observed term-to-maturity for Zero-Coupon Bond 2 time at time t for this *ith* test scenario.

The results for the above calculations are in the table below for both the Redington Immunisation ALM and Deep Reinforcement Learning ALM methods. We also show the differences between the calculated asset portfolio Duration of the Redington Immunisation ALM method and the Deep Reinforcement Learning ALM Asset Durations in Table 4.

We can see that, as expected, the Redington Immunisation approach's asset Duration gives shows an exact match to the Liability Duration. This is as expected as this is theoretical analytical approach, albeit with certain limiting underlying assumptions discussed in Section 4.1.

We can see that overall there is a slight difference. This shows that Reinforcement Learning ALM performs just as well as the Redington Immunisation ALM on matching Durations at a given time. The mean deviation in Asset Duration is 0.07 years, and the mean percentage difference is 0.28%, which are both negligible. The results above are only for one sampled test data scenario. In the next section, we aggregated the differences across all the 1 000 scenarios in Test Data 1 so that we could assess the Reinforcement Learning ALM performance across various scenarios.

Table 4
Test Sample Path Annual Time-points (months) Duration Differences.

Timepoints	12	24	36	48	60
Liability Duration	24.97	24.02	22.15	21.65	20.10
Immunisation ALM Duration	24.97	24.02	22.15	21.65	20.10
RL ALM Duration	25.04	24.01	22.33	21.89	19.95
Duration Difference (Years)	0.1	−0.0	0.2	0.2	−0.2
Duration Difference (%)	0.3%	0.0%	0.8%	1.1%	(0.8%)

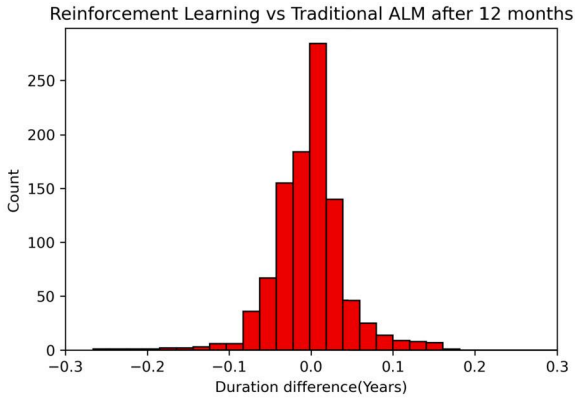

Fig. 9. Aggregate Duration Differential at 12 months.

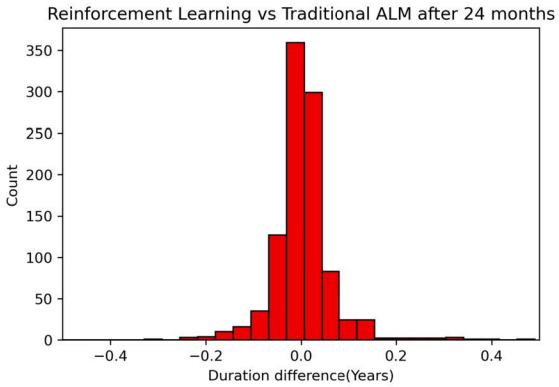

Fig. 10. Aggregate Duration Differential at 24 months.

7.1.2. *Aggregate evaluation on test data*

For each of the 1 000 test scenarios, we compared the Duration of the asset allocation from the Reinforcement Learning approach to that of the Redington Immunisation ALM. We did this at five time points, that is, after 1, 2, 3, 4 and 5 years. At each time point, we plotted the histogram of the differences between the RL and Redington Immunisation approaches' Duration differences in Figs. 9–13.

The 95% Confidence Interval(CI) of the difference can be estimated by the expression shown in Equation (11) (Guignard et al., 2021):

$$95\%CI(Lower; Upper) = (\hat{\mu} - 1.96 \times \hat{\sigma};\ \hat{\mu} + 1.96 \times \hat{\sigma})\,. \tag{11}$$

From Equation (11), we estimated that the 95% Confidence Interval of the Duration Difference at time 12 months to be (−0.09; 0.09), at 24 months is (−0.13; 0.13), at 36 months is (−0.16; 0.17), at 48 months is (−0.22; 0.21) and at time 60 months is (−0.25; 0.24). All of these values were mostly within 1% of the theoretical Redington Immunisation Duration.

We can see that the mean difference between the two approaches is close to zero (0) at all five (5) test time points. Furthermore, the distribution is symmetric, which shows that there is no bias in the performance of Reinforcement Learning compared to the traditional approach. All the 95% Confidence Intervals at the five (5) test time points

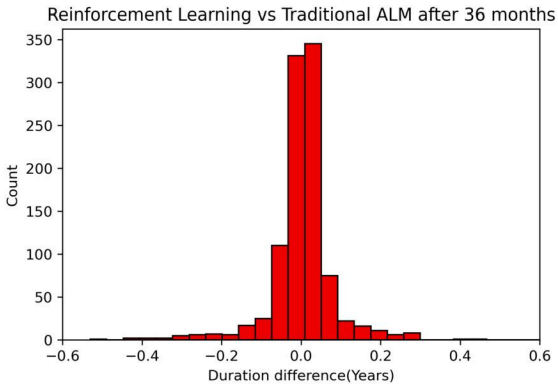

Fig. 11. Aggregate Duration Differential at 36 months.

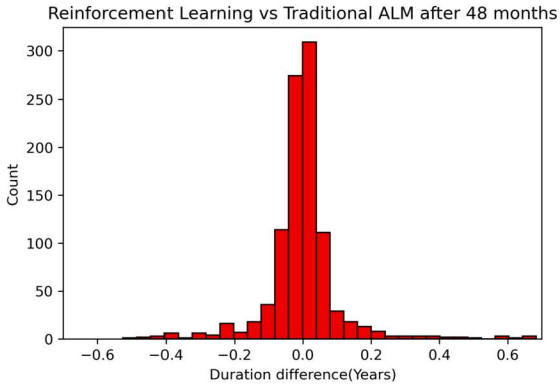

Fig. 12. Aggregate Duration Differential at 48 months.

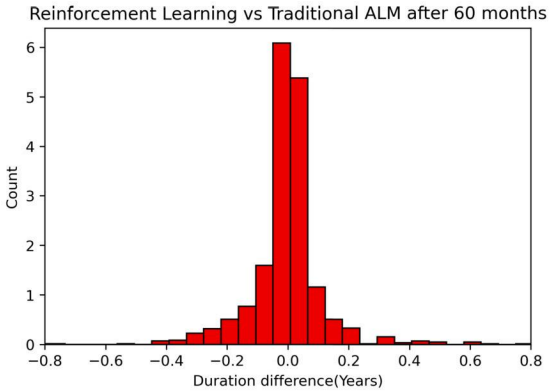

Fig. 13. Aggregate Duration Differential at 60 months.

are narrow around zero, and within 1% of the Redington Immunisation Duration results. This means that at a 95% Confidence Level, the allocations of the Reinforcement Learning ALM are statistically the same as those from the theoretical immunisation results.

7.1.3. *Aggregate evaluation on stress test data*

The Stress Test conditions explored correspond to a situation where either there are significant shifts in the level or shape of the yield curve in a short space of time. These periods would cause high volatility in the Liability Duration.

In this experiment, we repeated exactly the same experiment as in Section 7.1.2 with the exception of one difference. The only difference was that the test data used for evaluating the Reinforcement Learning ALM had a maximum possible Duration change (Δ) of one year between time points instead of half a year (0.5) as it was in Section 7.1.2.

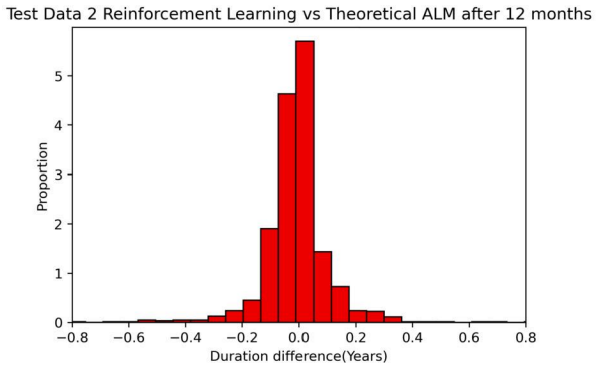

Fig. 14. Aggregate Duration Differential at 12 months.

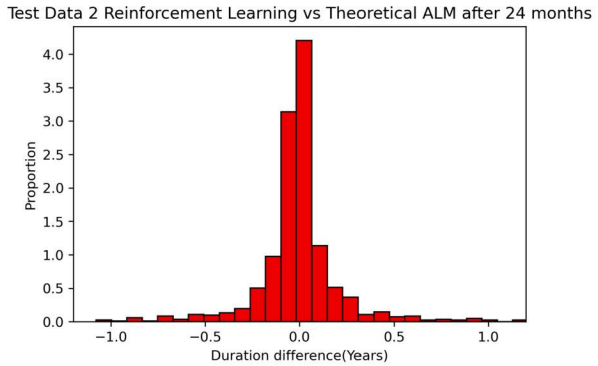

Fig. 15. Aggregate Duration Differential at 24 months.

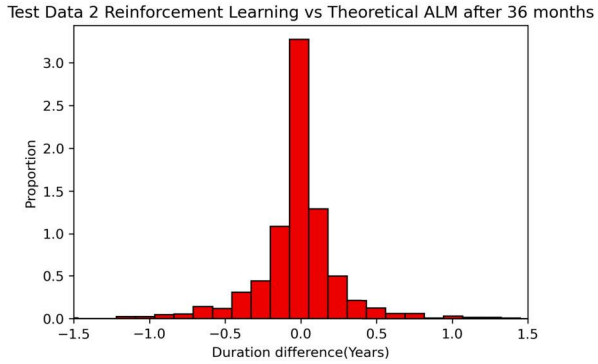

Fig. 16. Aggregate Duration Differential at 36 months.

We named this second test simulation data Test Data 2. This means that in this experiment we test how the same Reinforcement Learning ALM model would be able to perform on unseen data which is, in addition, much more volatile compared to what the data it had seen before. The aim was to test the Reinforcement Learning ALM on its adaptability to new conditions.

In Figs. 14–18 show histograms of the Duration differences at 12 months, 24 months, 36 months, 48 months and 60 months, respectively.

The 95% Confidence Interval of the Duration Difference at 12 months is (−0.71; 0.71), at 24 months is (−0.24; 0.24), at 36 months is (−0.44; 0.44), at 36 months is (−0.56; 0.56), at 48 months is (−0.62; 0.62) and at 60 months is (−0.71; 0.71).

Although the Confidence Intervals from the Stress Testing above are wider than what we saw in Section 7.1.2, it is remarkable that they are still very narrow (relative to the range of possible Duration values of up to 40 years or more). The mean difference between Reinforcement Learning Duration and the theoretical level is still close to 0 at all five test time points. Furthermore, the distribution is still symmetric, which

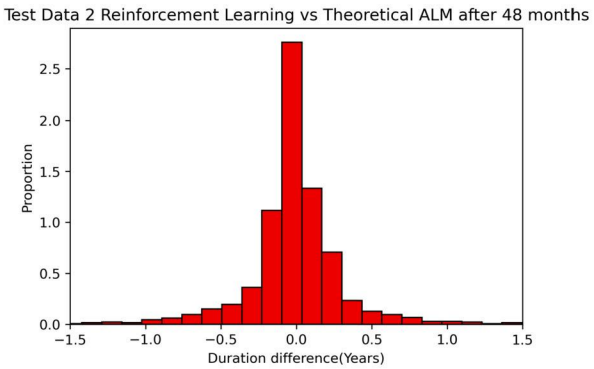

Fig. 17. Aggregate Duration Differential at 48 months.

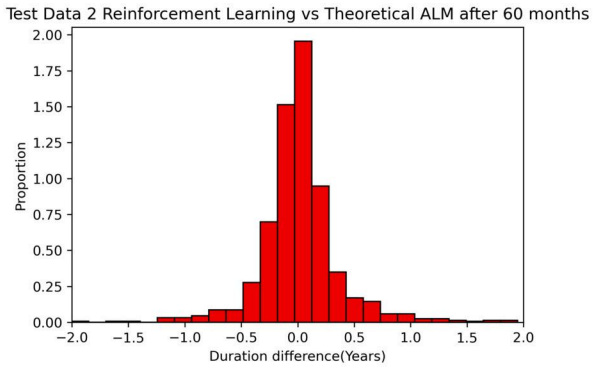

Fig. 18. Aggregate Duration Differential at 60 months.

shows that there is still no bias in the performance of Reinforcement Learning.

All the 95% Confidence Intervals at the five test time points are narrow around zero, and they all include zero within approximately 2-3% of the theoretical duration results. This is still an exceptional level of performance despite the test data being much more volatile compared to the simulations on which it was trained. In practical applications where the conditions differ from theoretical assumptions, this level of robustness would outperform the traditional approaches.

7.2. Deep reinforcement learning ALM compared to a benchmark traditional ALM approach

In this Section, we discuss the evaluation of the second research question, which was: How does Deep Reinforcement Learning's asset allocations compare to those of a benchmark real-world traditional ALM strategy in hedging a changing liability portfolio? Does it result in better interest rate hedging and to what extent?

In the first research question, we found that DRL gives similar results to Redington Immunisation at a given point in time. However, in order to fully assess the performance benefits of DRL ALM in practice it was important to compare against a typical ALM implementation that maybe in force in practice.

We contrasted the benefits of DRL ALM implementable on daily basis against a traditional ALM approach which is usually feasible on a monthly or on a weekly level for efficient organisation. In order to be conservative in our comparison, we assumed a weekly re-balancing for the traditional approach. We assessed the differences in outcomes at the end of the month between the daily Reinforcement Learning ALM approach and the weekly traditional ALM approach.

7.2.1. Benchmark sample case comparison evaluation

For testing purposes, we generated a new set of data not seen by the Reinforcement Learning Agent during the training phase. We generated

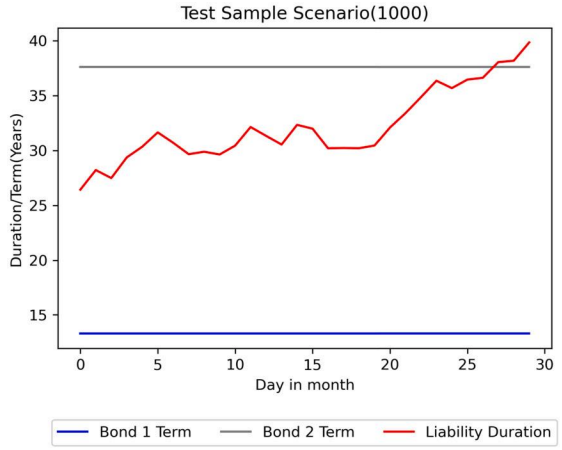

Fig. 19. Monthly Monte Carlo Simulation Test Sample Path.

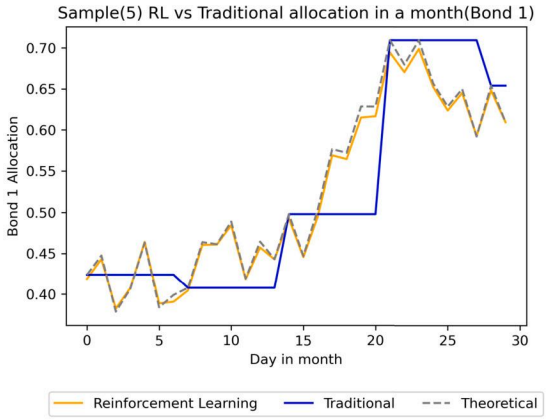

Fig. 20. Benchmark ALM Asset Allocation Comparisons (Bond 1).

1 000 test scenarios; hence, the data was also 3-Dimensional but with Dimensions of 30 × 1 000 × 3. In the next section, we selected one of the scenarios for illustration.

For illustrative purposes, we selected a scenario from the test data, which is graphically shown in Fig. 19:

In the above scenario, the Liability Duration gradually reduced from 23.4 years on Day 1 to 16.8 years on Day 23. From there onwards, the Liability Duration gradually increases to 19.1 years on day 30. The Term for Bond 1 is 10.2 years and that for Bond 2 is 33.1 years.

In the above scenario, we then applied the traditional ALM approach on weekly re-balancing frequency, followed by the Reinforcement Learning ALM on a daily re-balancing regime. We determined their respective asset allocations for Bond 1 and Bond 2. We also determined the daily theoretical asset allocations based on applying the traditional ALM approach on a daily basis. The theoretical asset allocation would represent the benchmark asset allocation if the analytical asset allocation had been possible daily.

The asset allocations under the theoretical, traditional and reinforcement learning regimes are shown in Fig. 20 and Fig. 21.

From the above, we can see that the Reinforcement Learning asset allocation is able to allocate in close proximity to the theoretical level at most time points. As expected, the allocation towards the shorter Duration Bond (Bond 1) increases until Day 23, which coincides with the decrease in Duration noted in this period in the sample graph. As expected, the allocation towards the longer Duration Bond (Bond 2) increases from day 23 until Day 30, which coincides with the increase in Liability Duration for this period.

A notable observation is that the traditional ALM allocation pattern is step-wise. It is constant except on the days on which there is weekly

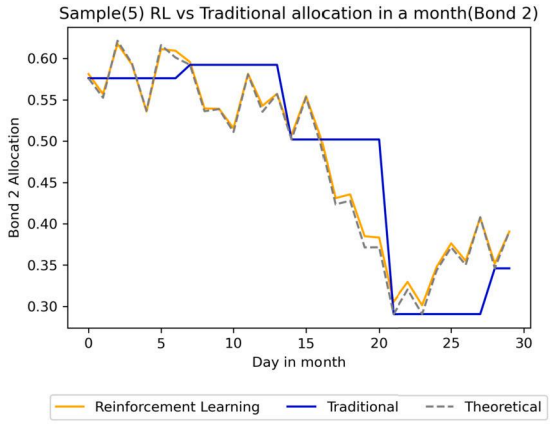

Fig. 21. Benchmark ALM Asset Allocation Comparisons (Bond 2).

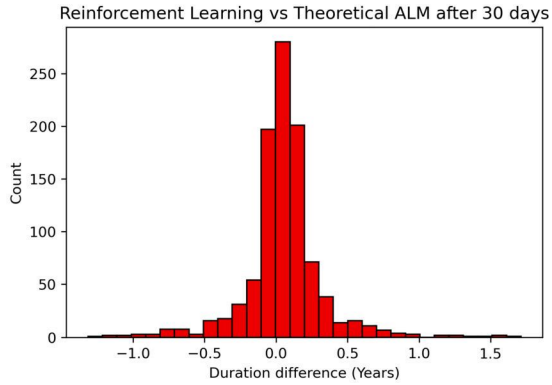

Fig. 22. Daily Reinforcement Learning Aggregate Duration Differential at Day 30.

rebalancing. On the rebalance days, the allocation is close to the allocations for both the Reinforcement Learning ALM and the Theoretical ALM. However, in between the rebalancing time points, there are deviations of the traditional allocations from the Reinforcement Learning and Theoretical allocations.

Although there is weekly rebalancing in the traditional regime, there is still a risk that there are sudden changes in interest rates at a time when this traditional weekly allocation deviates significantly from the Theoretical allocations. If this happens, there could be significant mismatches between the movements of the value of liabilities relative to that of the assets portfolio. This risk will be quantified in the next section.

The analysis done above was based on an illustration of one test scenario only. In the next section, we aggregate the relative performance of Reinforcement Learning against the traditional across all the 1 000 test data scenarios.

7.2.2. Benchmark aggregate comparison evaluation

For each of the 1 000 scenarios in the test data, we determine the asset allocations at day 30 for the Reinforcement Learning ALM, traditional ALM with weekly rebalancing and the Theoretical ALM. Based on these, we determined the respective Asset Duration for each approach for each of the 1 000 scenarios.

We used the Theoretical Asset Duration as the benchmark, which is the same as the actual Liability Duration on Day 30. The closer a result is to this level, the better the outcome of the respective method. Therefore, to assess the Reinforcement Learning ALM performance, we calculated the difference of its asset Duration from the Theoretical one on Day 30. We repeated this for each of the 1 000 scenarios and plotted a histogram of the Duration differences in Fig. 22.

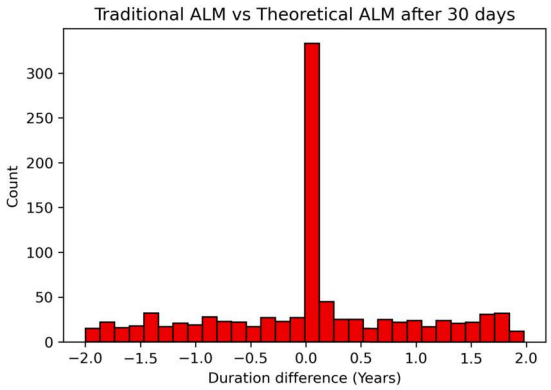

Fig. 23. Weekly traditional Aggregate Duration Differential at Day 30.

From the above, we can see that the Duration Differences are centred symmetrically and bell-shaped around 0. The mean difference is 0.055 years, which shows that, on average, the outcome of the Reinforcement Learning ALM is very close to what one would practically require for sufficient risk management. The standard deviation of these differences is 0.28. This means that 95% of the differences are not more than 0.56 years (approximately two standard deviations) years from the theoretical level.

Therefore, in order to assess the traditional ALM performance, we calculated the difference in its asset Duration from the Theoretical at Day 30. We repeated this for each of the one-thousand (1 000) scenarios and plotted a histogram of the Duration Differences in Fig. 23. From Fig. 23, we can see that the Duration Differences are still centred around 0 but are no longer bell-shaped. There is a higher proportion of differences which are significantly different from 0 at the end of the month.

The mean difference is around 0.03, but the standard deviation is much higher at 0.92. This means that 95% of the differences are not more than 1.84 years (approximately two standard deviations) from the theoretically correct level. This threshold level for the traditional ALM approach is over three times that of the Reinforcement Learning ALM (0.56 years). This means that with all else being the same, the traditional approach can differ from the theoretical Duration by three times as much as the Reinforcement Learning approach.

The Modified Duration of the Bond gives the impact of a 100-basis point (1%) on the price of a Bond. This is given by the formula shown in Equation (12) (Bierwag & Fooladi, 2006, Wu, 2022):

$$MD = \frac{D}{1 + \frac{YTM}{n}}, \tag{12}$$

where:

- MD is the Modified Duration. D is the Macaulay Duration, that is, the usual Duration we have been referring to for the rest of the paper (as Defined in Equation (2)) in years.
- YTM is the Yield-to-Maturity of the Bond, that is, the nominal annual interest rate.
- n is the frequency of interest compounding per year. In our case $n = 1$ because the Bonds are Zero-Coupon Bonds. In this case of $n = 1$, the YTM is also the effective annual interest rate.

From the above formulation, we can deduce that the estimated average net portfolio (Asset less liabilities) from traditional ALM with a weekly re-balancing at the end of 30 days has a 1.71% change in value from a 1% change in interest rates. On the other hand, the estimated average Net Portfolio (Assets less Liabilities) from Reinforcement Learning ALM with a daily re-balancing has a 0.53% change in value from a 1% change in interest rates.

From the above calculations, we can deduce that the Reinforcement Learning ALM's Net Portfolio will be much more securely hedged (approximately three times better) compared to the traditional ALM's Net Portfolio, with all else being the same. This particularly important consideration in a situation where interest rates are increasing. If the Asset Duration is much higher than the Liability Duration, then the asset portfolio value will fall by a bigger margin compared to the liability value, representing bigger exposure to interest rate risk.

8. Discussion, conclusion and future works

8.1. Results discussion

From Section 7.1, the first finding was that, in general, Deep Reinforcement Learning could successfully replicate the performance of Redington Immunisation theory. We calculated confidence intervals of the differences between Immunisation theory results and Deep Reinforcement Learning at test time points. The results first demonstrated that DRL ALM can achieve duration-matching outcomes within 1% of the theoretical immunisation theory at a 95% confidence level. The Reinforcement Learning ALM was able to achieve the same level of performance despite not relying on interest rate assumptions compared to the traditional method.

Due to the novelty of this work, there are no directly comparable works which applied Deep Reinforcement Learning to replicate Redington immunisation theory. As far we can tell this paper is the first. However, we can consider work of a similar magnitude where Deep Reinforcement Learning was applied to replicate the foundational approach of the Black-Scholes theory of hedging Derivatives (Hariom Tast & Lookabaugh, 2020). This work found that Deep Reinforcement Learning could replicate the same level of accuracy as the theoretical approach in implementing Delta Hedging. This Delta Hedging application was built on a *deep hedging* theoretical framework from a few years earlier (Bühler et al., 2018). This work's results are similar to this paper's in the sense that Deep Reinforcement Learning was successfully applied to replicate and improve upon the results of the foundational theory in their respective domains. In both cases this was achieved with much fewer theoretical assumptions. There were also many similarities in the methodologies such as the Deep Learning architectures, the model-free Reinforcement Learning approach and the use of Monte Carlo simulations of the underlying processes.

In Section 7.1, we further saw that as interest rate conditions deviated from precise theoretical assumptions (which are associated with more volatile liability Duration), Deep Reinforcement Learning was able to largely maintain its Duration Matching capabilities. This illustrated the increased robustness of Reinforcement Learning to market conditions compared to the traditional methods. Furthermore, Reinforcement Learning did not require external restrictions on many of the optimisation parameters, such as the range of values for the weights. We noticed that the Reinforcement Learning Agent was able to automatically learn, based on the Reward Function, that the appropriate weights needed to range between zero (0) and one (1) because the context did not allow short-selling. On the other hand, in immunisation theory, we needed to explicitly express this restriction or make a manual adjustment afterwards. Therefore, Reinforcement Learning demonstrated the ability to perform as well as Immunisation theory by just expressing to it the objective and with fewer assumptions and restrictions.

From Section 7.2, the investigations showed that Deep Reinforcement Learning ALM could be implemented more frequently with daily re-balancing of the asset portfolio allocation for Duration Matching as the liability continuously changes. This resulted in significantly better interest rate hedged portfolios than the traditional ALM approach with weekly re-balancing, which is the most frequent that can be feasibly achieved with traditional strategies in practice in most cases. In our case, we estimated that Reinforcement Learning net portfolio values are approximately three times less sensitive to interest rate movements

compared to the traditional ALM on a weekly regime. This is a significant out-performance, especially considering the large financial sums that large intuitions manage.

This fact that Deep Reinforcement Learning outperformed the benchmark could be expected. Two related works applied Deep Learning and Reinforcement Learning to Asset Liability modelling of retail banking assets and liabilities (Krabichler & Teichmann, 2023, Englisch et al., 2023). Although the applications were in retail banking with different outcome measurement metrics, their results also strongly demonstrated significant outperformance relative to several practical benchmark Asset Liability Management strategies.

This ability to be implemented on a more frequent basis is because Deep Reinforcement Learning requires less human intervention compared to the traditional approach (Hariom Tast & Lookabaugh, 2020, Englisch et al., 2023). This is partly because the Reinforcement Learning approach relies less on theoretical assumptions. In addition, Reinforcement Learning ALM was performed within scalable computational frameworks which unlocks faster and more automated processing by leveraging powerful open-source libraries such as TensorFlow. This enables one to practically apply Reinforcement Learning regularly at an enterprise level with high-velocity and voluminous data (Geron, 2019, Tensorflow.org, 2022, Lang, 2022). Reinforcement Learning also achieves more consistent hedging within the month since its allocations track better to the theoretical level. This means that with Deep Reinforcement Learning there are fewer times when there are significant mismatches of assets and liabilities.

8.2. Conclusion

In this paper we applied a combination of Reinforcement Learning and Deep Learning to Asset Liability Management (ALM). This combination is called Deep Reinforcement Learning (DRL). We termed this application Deep Reinforcement Learning Asset Liability Modelling (DRL ALM).

Before this paper, Reinforcement Learning had been applied in the Actuarial and Quantitative Finance domains to mostly carry out Trading, Portfolio Allocation, and Derivatives Hedging (Hariom Tast & Lookabaugh, 2020, Kolm & Ritter, 2020, Dixon & Halperin, 2019). Although the recent application of Deep Learning and Reinforcement Learning for Asset Liability Modelling was successful in banking it was based on approaches customised specifically for retail banking (Krabichler & Teichmann, 2023, Englisch et al., 2023). There had been no well-documented literature and applications of Reinforcement Learning, which consider both sides of a risk taker's balance sheet in all spheres of risk management theory. This paper fills this gap both in the literature and application, by providing an avenue for the improvement of Actuarial and Quantitative Finance practice by providing an avenue that replicates and potentially improves upon Redington Immunisation theory.

For testing purposes, we used Monte Carlo methods to simulate data similar to that encountered in real-world by risk-taking institutions. For each simulation, we had stochastic variation in the Liability Duration and the asset portfolio's maturity terms. We assumed that the assets were comprised of two zero-coupon bonds. The approach can, as we discussed, be validly extended to coupon-bearing bonds (Jarrow, 2004, Jarrow & Turnbull, 2000). We used 10 000 simulations for training the Reinforcement Learning Asset Liability Management agent. The data sets were generated from separate Monte Carlo simulation runs and are thus unseen by models during training, which provided a robust testing regime.

For the purpose of comparison, we set up a Redington Immunisation model, which is the foundation theory for most traditional approaches to ALM. It is based on solving simultaneous equations to determine the appropriate asset weights required to achieve matching of the Assets Duration and Liability Duration. We noted the restrictive theoretical assumptions as well as several practical problems including the resource-intensive requirements for careful monitoring and professional adjustments.

We then developed the Reinforcement Learning framework for implementation. The Reinforcement Learning approach required the development and implementation of Object Orientated Programming (OOP) or Class-based Programming in Python programming. First, we defined an autonomous Agent as a class that can create Reinforcement Learning objects with specific attributes, modules, and functions. We then defined the critical Reinforcement Learning components, such as the Agent, the Environment, the States of the environment (simulation data), the Actions of the Agent (Asset allocations), and the Reward Function (Sum Square Error). In addition, the core functionality of Reinforcement Learning in its data processing and computations was implemented using the TensorFlow library in Python. We used TensorFlow because of its enhanced abilities to manage large datasets and demanding computational jobs due its unique computation graph structure (Lang, 2022, Pang et al., 2020, Geron, 2019). Within this TensorFlow framework, we incorporated a Deep Learning model, the Long-Short-Term Memory Recurrent Neural Network (LSTM-RNN) (Smagulova & James, 2019, Staudemeyer & Morris, 2019). The LSTM-RNN enabled in-depth learning of the Reinforcement Learning Agent to be able to learn and discern time-dependent trends in the data in the formulation of its policy.

We compared the performance of the DRL ALM to both Redington Immunisation theory and a weekly traditional ALM as a benchmark of the current best possible implementation. The results demonstrated that DRL ALM can achieve duration-matching outcomes within 1% of the theoretical immunisation theory at a 95% confidence level. Furthermore, compared to a benchmark weekly rebalancing traditional ALM regime, high-frequency DRL ALM achieved superior outcomes which are, on average, 3 times less sensitive to interest rate changes. DRL ALM also demonstrated capacity for increased automation, speed, flexibility, and multi-objective optimisation in ALM, thereby, further potential for reducing the negative impact of human limitations and improving risk management outcomes. The findings and principles presented in this study apply to various institutional risk-takers, including insurers, banks, pension funds, and asset managers.

Overall, Deep Reinforcement Learning was not only able to perform similar tasks as traditional methods in general but was also improved on many aspects of ALM. Deep Reinforcement Learning demonstrated the potential to improve ALM by providing an avenue for more frequent and automated asset re-allocations for better interest rate hedging, less reliance on theoretical assumptions and restrictions, high-frequency, increased robustness in varying market conditions, and increased flexibility.

In summary, Deep Reinforcement Learning ALM has several gains relative to the traditional ALM approaches. These include:

1. The ability of DRL ALM to attain the same level of accuracy compared to theoretical immunisation expectations but with much fewer theoretical assumptions about market conditions such as interest rate behaviour. We saw this in Section 7.1.2, where the outcomes within 1% of the theoretical immunisation theory at a 95% confidence level.
2. Increased robustness of DRL ALM in hedging outcomes in stressed and/or volatile market conditions due to the reduced reliance on theoretical assumptions about the market. We saw this in Section 7.1.3 where the DRL ALM results were impressively accurate despite being applied to a more volatile environment relative to what it was trained on. This robustness emanates from the fact that the DRL ALM directly learns the objectives and can adjust its actions accordingly.
3. Better interest rate hedging results compared to a benchmark weekly rebalancing traditional ALM approach. In Section 7.2 we saw that DRL ALM achieved superior ALM outcomes which are 3

times less sensitive to interest rate changes on average for similar scenarios at the end of a 30-day period.

4. The better hedging results are partly attributable to the fact that once the DRL ALM agent is trained, it is relatively fast to get results from it for a given set of input scenario data. This makes it much easier to automate the ALM process to a much higher frequency basis such as daily or hourly. This is in contrast to a much slower traditional ALM process which requires some manual input such as extensive data gathering and professional judgement in its application.
5. DRL ALM, if successfully deployed, can reduce various business costs to institutional risk-takers. The better ALM outcomes seen from DRL ALM can enable more efficient capital allocation, minimise unnecessary asset reserving and increase return on capital for shareholders. In the long run, DRL ALM can save on costs by either reducing unnecessary employee headcount in ALM departments, especially for manual tasks, or improving the productivity of ALM employees. DRL ALM introduces a highly capable Artificial Intelligence (AI) Asset Liability Management assistant that can complement and increase the productivity of professionals in the Asset Liability Management field.

8.3. Future works

In this section we highlight important future work and why it is important. There are several areas of potential additional research that can be pursued which build on the research done in this paper.

First, an important future work is to test how the trained Deep Reinforcement Learning agent would perform on real-world asset and liability data of an institution. This would involve deploying the trained Reinforcement Agent in a real-world investment management or asset management floor with all real-world challenges. Some development would be needed to integrate the Reinforcement Learning agent object and its associated Python programme into the enterprise systems of the institution's deploying the agent. In this deployment, a feedback loop which feeds data to the Reinforcement Learning from the institution's asset and liability data systems, and simultaneously sends the Agents actions would need to be established. This would be important to further refine the DRL ALM approach to take into account practical considerations such data system integration and legacy system issues.

Second, another area of future research would be to test the limits of the Deep Reinforcement Agent. The first test is including multiple assets (P) instead of two assets. There would be valuable insights derived from investigating the feasibility of implementing Reinforcement Learning as P approaches a very large number, that is, as the Dimensionality of the problem increases exponentially. For such an assessment, one would need to do a significant amount of work in making the number of assets a dynamic part of the Reinforcement Learning workflow. This will requires additional development in the Python programme. This additional development will require improvement on the formulation of the Deep Learning model and Reinforcement Learning agent so that they can generate a dynamic number of actions (weights) in an automated manner, which would correspond to a large number of assets (P). There also would be valuable insights in assessing the choices allocation and whether there would be a significant reduction or improvement in performance.

Third, we would also recommend assessing the feasibility and impact on the performance of including different data types in the states such as asset returns, asset volatility, detailed risk profiles of liabilities (such as ages, health status, gender, etc.), and even non-numeric data. These non-numeric data could include some market sentiment data or some internal institutional data. Deep Learning Models are well known for their ability to cope well with non-numerical data, such as in Natural Language Processing, and it would be interesting to investigate whether this strength can be leveraged in Deep Reinforcement Learning. Fifth, in this paper, the asset portfolio was based on Zero-Coupon Bonds. This is not a very restrictive assumption, as the findings are applicable when hedging is carried out with Coupon-Bearing Bonds. However, it would be valuable to carry out research on the effect of incorporating more complicated asset types such as Equities, Convertible Bonds, Derivatives, etc. In addition, it would be interesting to investigate the impact on the choices of the Reinforcement Learning solution.

Fourth, another area of future research would be to incorporate additional restrictions or parameters such as regulatory capital regimes such as Solvency II and Solvency Assessment and Management (SAM), Liquidity requirements or other risk assessment measures such as Value-at-Risk. It will be valuable to assess how easily the current solution will extend to incorporate these additional objectives and how it performs in balancing the different objectives.

Last, we noticed some practical challenges that one would need to overcome in applying Deep Reinforcement Learning. The first challenge is that there are no off-the-shelve consolidated packages/libraries or pre-defined modelling approaches for implementing Reinforcement Learning. This means that one typically has to program many of the steps and details of the autonomous Reinforcement Agent classes and objects together with their attributes and functions. Furthermore, implementing Deep Learning in TensorFlow also requires a good understanding of unique and specific computational graph formulation and syntax. A good understanding of TensorFlow is also required in managing all the data in high dimensional matrices (Tensors), which are different from the usual handling of data in most libraries and programming languages. We, therefore, believe that there are also opportunities for research into simplifying the development of Reinforcement Learning solutions, especially when used with Tensorflow as is often the case.

There is also a gap in making deploying a Deep Reinforcement Learning Agent easier and quicker. Some work has been done in areas such as Stable Baselines (D'Eramo et al., 2020, Baselines, 2022). It would be valuable to assess how easy it is to apply these approaches to the Asset Liability Management problems.

CRediT authorship contribution statement

Takura Asael Wekwete: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Writing – original draft. **Rodwell Kufakunesu:** Supervision, Writing – review & editing. **Gusti van Zyl:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Abrate, C., Angius, A., De Francisci Morales, G., Cozzini, S., Iadanza, F., Puma, L. L., & Ronchiadin, S. (2021). Continuous-action reinforcement learning for portfolio allocation of a life insurance company. In Y. Dong, N. Kourtellis, B. Hammer, & J. A. Lozano (Eds.), *Machine learning and knowledge discovery in databases. Applied data science track*. Cham: Springer International Publishing (pp. 237–252).

Altché, F., & de La Fortelle, A. (2017). An LSTM network for highway trajectory prediction. In *2017 IEEE 20th international conference on intelligent transportation systems* (pp. 353–359).

Apple (2023). Macbook air (m1, 2020) - technical specifications (uk). Retrieved from https://support.apple.com/kb/SP825?locale=en_GB. (Accessed 10 July 2023).

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine, 34*(6), 26–38. https://doi.org/10.1109/MSP.2017.2743240.

Barr, C. (2023). *What's going on with first republic bank? - WSJ* (Accessed 2 April 2023).

Baselines, S. (2022). *Welcome to stable baselines docs! - rl baselines made easy — stable baselines 2.10.3a0 documentation* (Accessed 21 November 2022).

Bierwag, G., & Fooladi, I. (2006). Duration analysis: An historical perspective. *Journal of Applied Finance, 16*.

Bondt, W. D., Mayoral, R. M., & Vallelado, E. (2013). Behavioral decision-making in finance: An overview and assessment of selected research [La toma de decisión en las finanzas del comportamiento: Estado de la cuestión a partir de los trabajos seleccionados]. *Revista Española de Financiación Y Contabilidad*, *42*(157), 99–118. Retrieved November 22, 2022, from http://www.jstor.org/stable/42782820.

Bühler, H., Gonon, L., Teichmann, J., & Wood, B. (2018). Deep hedging. arXiv:1802.03042. Retrieved from https://arxiv.org/abs/1802.03042.

Cheridito, P., Ery, J., & Wüthrich, M. V. (2020). Assessing asset-liability risk with neural networks. *Risks*, *8*(1). https://doi.org/10.3390/risks8010016.

Chiu, C.-J., Ho, A. Y.-F., & Tsai, L.-F. (2022). Effects of financial constraints and managerial overconfidence on investment-cash flow sensitivity. *International Review of Economics & Finance*, *82*, 135–155. https://doi.org/10.1016/j.iref.2022.06.008.

Daga, A. (2023). *What happened at credit suisse and how did it reach crisis point? Reuters* (Accessed 2 April 2023).

D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., & Peters, J. (2020). Mushroomrl: Simplifying reinforcement learning research. https://doi.org/10.48550/ARXIV.2001.01102.

Devraj, A., & Meyn, S. (2019). *Zap q-learning*. University of Florida. https://proceedings.neurips.cc/paper/2017/file/4671aeaf49c792689533b00664a5c3ef-Paper.pdf. Retrieved from https://ieeexplore.ieee.org/document/8715554.

Dixon, M. F., & Halperin, I. (2019). The four horsemen of machine learning in finance. *SSRN Electronic Journal*, *26*, 18–28. Retrieved from https://ssrn.com/abstract=3453564.

Dong, S., Wang, P., & Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review*, *40*, Article 100379. https://doi.org/10.1016/j.cosrev.2021.100379.

Englisch, H., Krabichler, T., Müller, K. J., & Schwarz, M. (2023). Deep treasury management for banks. *Frontiers in Artificial Intelligence*, *6*. https://doi.org/10.3389/frai.2023.1120297.

Fooladi, I., & Roberts, G. (2000). Risk management with duration analysis. *Managerial Finance*, *26*, 18–28. https://doi.org/10.1108/03074350010766558.

Garrett, S. (2013). Chapter 9 - term structures and immunization. In S. Garrett (Ed.), *Introduction to the mathematics of finance* (second edition) (pp. 177–209).

Geman, H. (2023). *From Lehman to silicon valley bank and beyond: Why are mistakes repeated in the us banking system?* (Accessed 2 April 2023).

Geron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd edition). United States: O'Reilly Media, Inc. Retrieved from https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.

Guignard, F., Amato, F., & Kanevski, M. (2021). Uncertainty quantification in extreme learning machine: Analytical developments, variance estimates and confidence intervals. *Neurocomputing*, *456*, 436–449. https://doi.org/10.1016/j.neucom.2021.04.027.

Hariom Tast, S. P., & Lookabaugh, B. (2020). *Machine learning and data science blueprints for finance. (Chap. 9)*. O'Reilly Media, Inc. Retrieved from https://www.oreilly.com/library/view/machine-learning-and/9781492073048/.

He, T., & Droppo, J. (2016). Exploiting LSTM structure in deep neural networks for speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing* (pp. 5445–5449).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Hsu, W.-N., Zhang, Y., Lee, A., & Glass, J. (2016). Exploiting depth and highway connections in convolutional recurrent deep neural networks for speech recognition. In *Proc. interspeech 2016* (pp. 395–399).

Hua, Y., Zhao, Z., Li, R., Chen, X., Liu, Z., & Zhang, H. (2019). Deep learning with long short-term memory for time series prediction. *IEEE Communications Magazine*, *57*(6), 114–119. https://doi.org/10.1109/MCOM.2019.1800155.

Jarrow, R. A. (2004). Risky coupon bonds as a portfolio of zero-coupon bonds. *Finance Research Letters*, *1*(2), 100–105. https://doi.org/10.1016/j.frl.2004.03.003.

Jarrow, R. A., & Turnbull, S. M. (2000). The intersection of market and credit risk. *Journal of Banking & Finance*, *24*(1), 271–299. https://doi.org/10.1016/S0378-4266(99)00060-6.

Kahlig, J. (2022). Duration, convexity and immunisation. https://people.tamu.edu/~kahlig/notes/325/ch11-part-c.pdf. (Accessed 20 November 2022).

Kolm, P. N., & Ritter, G. (2020). Modern perspectives on reinforcement learning in finance. *SSRN Electronic Journal*, *1*, 18–28. Available at SSRN. Retrieved from https://ssrn.com/abstract=3449401.

Krabichler, T., & Teichmann, J. (2023). A case study for unlocking the potential of deep learning in asset-liability-management. *Frontiers in Artificial Intelligence*, *6*. https://doi.org/10.3389/frai.2023.1177702.

Lang, N. (2022). An introduction to TensorFlow. Get to know the machine learning, its architecture and the comparison to PyTorch. Retrieved from https://towardsdatascience.com/an-introduction-to-tensorflow-fa5b17051f6b. (Accessed 20 November 2023).

Li, Y. (2017). Deep reinforcement learning: An overview. https://doi.org/10.48550/ARXIV.1701.07274.

Li, Y. (2022). Introducing deep reinforcement learning, by Yuxi Li. Medium. https://medium.com/@yuxili/deeprl-6c8c48b6489b. (Accessed 2 November 2022).

Mallinar, N., & Rosset, C. (2018). Deep canonically correlated LSTMs, https://doi.org/10.48550/ARXIV.1801.05407.

Mousavi, S., Schukat, M., & Howley, E. (2018). *Deep reinforcement learning: An overview*. *Lecture notes in networks and systems* (pp. 426–440).

Nieto, A., Juan, A. A., & Kizys, R. (2022). Asset and liability risk management in financial markets. In J. Pilz, T. A. Oliveira, K. Moder, & C. P. Kitsos (Eds.), *Mindful topics on risk analysis and design of experiments* (pp. 3–17). Cham: Springer International Publishing.

Osiński, B., Jakubowski, A., Zięcina, P., Miłoś, P., Galias, C., Homoceanu, S., & Michalewski, H. (2020). Simulation-based reinforcement learning for real-world autonomous driving. In *2020 IEEE international conference on robotics and automation* (pp. 6411–6418).

Pang, B., Nijkamp, E., & Wu, Y. N. (2020). Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics*, *45*(2), 227–248. https://doi.org/10.3102/1076998619872761.

Qu, Z., Haghani, P., Weinstein, E., & Moreno, P. (2017). Syllable-based acoustic modeling with CTC-sMBR-LSTM. In *2017 IEEE automatic speech recognition and understanding workshop* (pp. 173–177).

Rabbani, A. G., Grable, J. E., O'Neill, B., Lawrence, F., & Yao, Z. (2021). Financial risk tolerance before and after a stock market shock: Testing the recency bias hypothesis. *Journal of Financial Counseling and Planning*, *32*(2). https://doi.org/10.1891/JFCP-19-00025. https://connect.springerpub.com/content/sgrjfcp/early/2020/12/22/JFCP-19-00025.full.pdf.

Redington, F. M. (1952). Review of the principles of life-office valuations. *Journal of the Institute of Actuaries (1886–1994)*, *78*(3), 286–340. Retrieved November 13, 2022, from http://www.jstor.org/stable/41139015.

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, https://doi.org/10.48550/ARXIV.1402.1128.

Sato, Y. (2019). Model-free reinforcement learning for financial portfolios: A brief survey. arXiv:1904.04973. Retrieved from https://arxiv.org/abs/1904.04973.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D*, *404*, Article 132306. https://doi.org/10.1016/j.physd.2019.132306.

Sigaud, O., & Stulp, F. (2019). Policy search in continuous action domains: An overview. *Neural Networks*, *113*, 28–40. https://doi.org/10.1016/j.neunet.2019.01.011.

Smagulova, K., & James, A. (2019). A survey on LSTM memristive neural network architectures and applications. *The European Physical Journal Special Topics*, *228*. https://doi.org/10.1140/epjst/e2019-900046-x.

Smink, M., & van der Meer, R. A. H. (1997). Life insurance asset-liability management: An international survey. *The Geneva Papers on Risk and Insurance. Issues and Practice*, *22*(82), 128–142. Retrieved from http://www.jstor.org/stable/41952310.

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM – a tutorial into long short-term memory recurrent neural networks. https://doi.org/10.48550/ARXIV.1909.09586.

Syed, Z., & Bansal, R. (2018). Do investors exhibit behavioral biases in investment decision making? A systematic review. *Qualitative Research in Financial Markets*, *10*, Article 00. https://doi.org/10.1108/QRFM-04-2017-0028.

Tensorflow.org (2022). Why tensorflow. Retrieved from https://www.tensorflow.org/about. (Accessed 20 November 2022).

Ward, D., & Zurbruegg, R. (2000). Does insurance promote economic growth? Evidence from OECD countries. Retrieved from https://www.jstor.org/stable/253847?seq=1#metadata_info_tab_contents. (Accessed 10 March 2023).

Wu, J. (2022). Financial market analysis for duration and modified duration. In *Proceedings of the 2022 7th international conference on financial innovation and economic development* (pp. 2637–2641).

Wutrich, M., & Merz, M. (2023). Statistical foundations of actuarial learning and its applications | springerlink. Retrieved from https://link.springer.com/book/10.1007/978-3-031-12409-9. (Accessed 1 July 2023).