Deep Joint Learning valuation of Bermudan Swaptions

Francisco Gómez Casanova, Álvaro Leitao, Fernando de Lope Contreras and Carlos Vázquez

Motivation and proposal

- In many areas of research requiring intensive scientific computing tools, there is an increasing use of deep learning and Artificial Neural Networks (ANN) techniques to overcome the drawbacks associated to the traditional numerical methods.
- This is also the case in computational finance, where a large variety of problems related to the pricing and risk management of (complex) financial products need to be efficiently solved.
- We consider early-exercise products, which are financial contracts that allow the holder to exercise a right prior to its expiration.
- Many solutions relying on “classical” methods are available, although multiple (computational) challenges remain posed.
- We propose a smart combination of several sophisticated ANN-based concepts: differential machine learning, Monte Carlo *sampled* training labels and joint learning.
- We also propose a novel design of interdependent ANNs to price early-exercise products, in this case, Bermudan swaptions.

Outline

1. Problem formulation

2. Deep Joint learning for Bermudan swaptions

3. Numerical experiments

4. Conclusions

Problem formulation

- Linear Gauss Markov (LGM) model:

$$dx_t = \alpha(t)dW_t, \quad x_0 = 0,$$

with $\zeta(t) := \int_0^t \alpha^2(\tau)d\tau$.

- The *numeraire* under LGM reads

$$N(t, x_t) = \frac{1}{D(t)} \exp\left(H(t)x_t + \frac{1}{2}H^2(t)\zeta(t)\right),$$

with $D(t)$ the discount factor of time t (observed in the market)

- $H(t)$ is a curve with a similar interpretation as the mean reversion in the Hull-White model:

$$H(t) = \frac{1 - \exp(-\kappa t)}{\kappa},$$

such as κ corresponds to the Hull-White mean reversion.

LGM: Analytic pricing formulas

- Zero coupon bond at t with maturity T:

$$Z(t, x_t; T) = \frac{D(T)}{D(t)} \exp\left(-(H(T) - H(t))x_t - \frac{1}{2}(H^2(T) - H^2(t))\zeta(t)\right)$$

- Interest Rate Swap (IRS) with payment tenor T_i, $i = 1, \ldots, M$:

$$V_S(t, x_t) = \phi\left(Z(t, x_t; T) - Z(t, x_t; T_M) - K\sum_{i=1}^{M} \Delta T_i Z(t, x_t; T_i)\right)$$

- European Swaption (on the previous IRS):

$$V_E(t, x_t) = \phi Z(t, x_t, T)\mathcal{N}\left(-\phi\frac{y_T^*}{\sqrt{\zeta(T) - \zeta(t)}}\right)$$
$$- \phi Z(t, x_t, T_M)\mathcal{N}\left(-\phi\frac{y_T^* + (H(T_M) - H(T))(\zeta(T) - \zeta(t))}{\sqrt{\zeta(T) - \zeta(t)}}\right)$$
$$- \phi K\sum_{i=1}^{M} \Delta T_i Z(t, x_t, T_i)\mathcal{N}\left(-\phi\frac{y_T^* + (H(T_i) - H(T))(\zeta(T) - \zeta(t))}{\sqrt{\zeta(T) - \zeta(t)}}\right)$$

LGM: Bermudan swaptions

- No closed-form solution.
- Valuation of a related product, i.e., Cancellable IRS (cIRS):

$$V_C^p = V_S^p - V_B^p,$$
$$V_C^r = V_S^r - V_B^r,$$

- The price of the Cancellable IRS is

$$\frac{V_C^p(t,x_t)}{N(t,x_t)} = \sup_{\tau\in\{T_i/T_i>t\}} \mathbb{E}\left[\max\left(\frac{V_S^p(\tau,x_\tau)}{N(\tau,x_\tau)},0\right)\right],$$
$$\frac{V_C^r(t,x_t)}{N(t,x_t)} = \sup_{\tau\in\{T_i/T_i>t\}} \mathbb{E}\left[\max\left(\frac{V_S^r(\tau,x_\tau)}{N(\tau,x_\tau)},0\right)\right].$$

- This formulation enables the use of dynamic programming and backward induction to determine the optimal cancellation policy and, then, solve the problem.

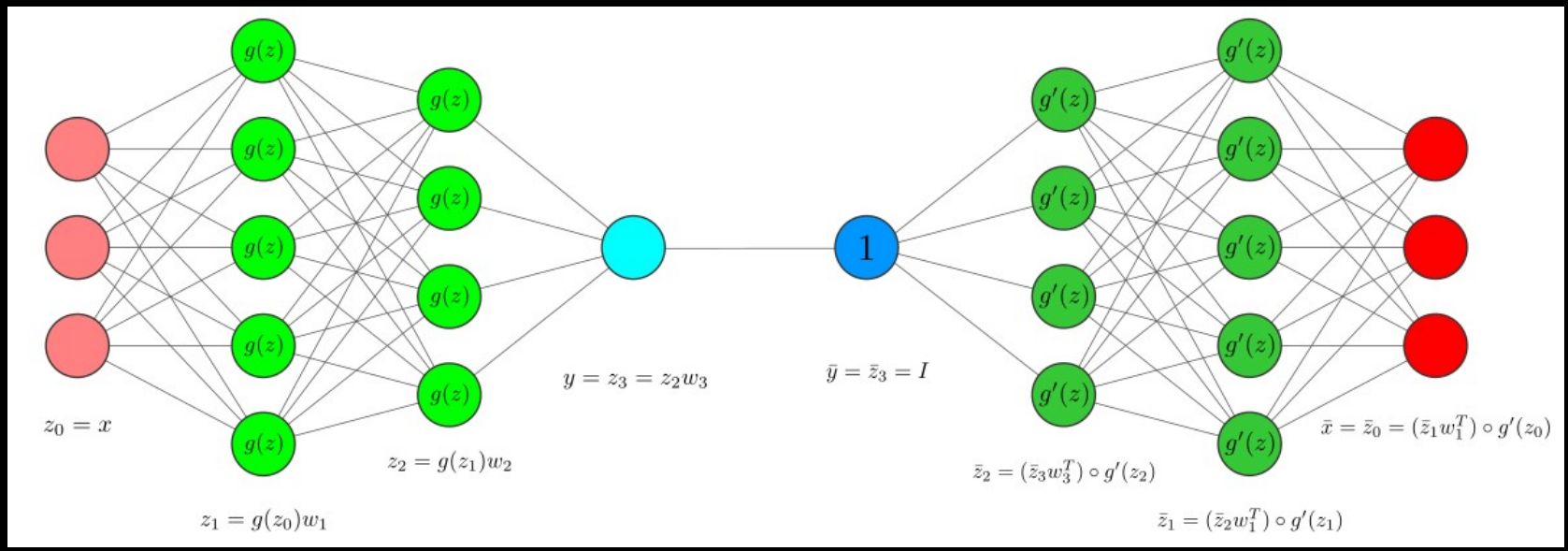
Deep learning approach for Cancellable IRS

citic

- We smartly combine three (individually) successful ANN components:
  - Differential Machine Learning $\rightarrow$ DANN
  - Training with *sampled* labels
  - Joint learning
- Cancellation policy: sequence of interconnected DANNs.
- Each DANN, associated with a cancellation opportunity, is used (once trained) to compute the labels of the next DANN.
- Then, the labels for the DANN of each cancellation opportunity depends on the estimations of all the “previous” DANNs.
- This specific design gets inspiration from the classical methods based on regression.
- An additional DANN approximates the final price given a newly generated samples, using joint learning feature.

Differential Machine Learning (DANN)

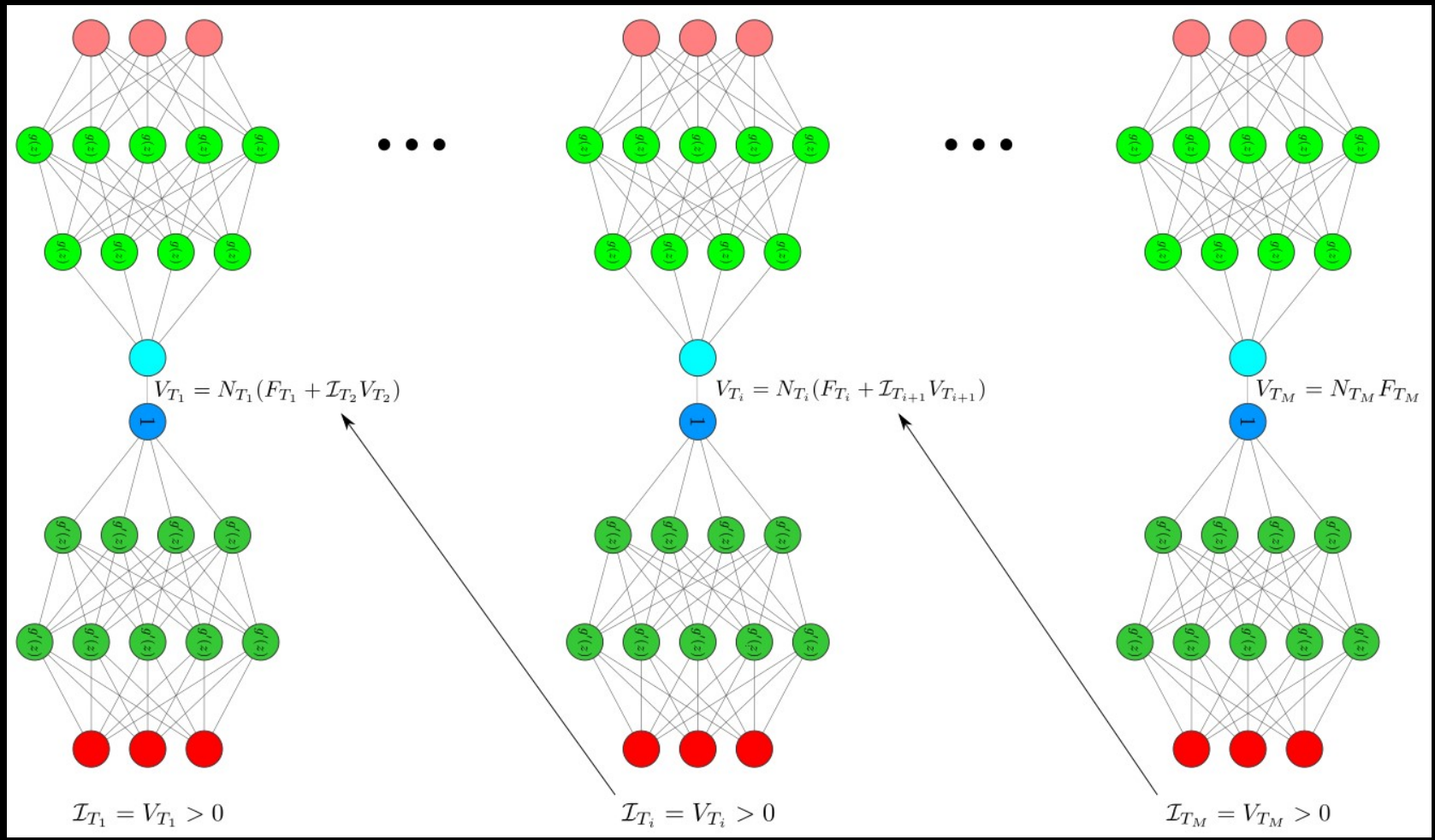
- Enlarge the network to consider (benefit from) the differentials of the output w.r.t the inputs.
- It requires the availability of those differentials.
- The loss function needs to incorporate both components.



Differential Machine Learning with ANNs.

Backward DANNs

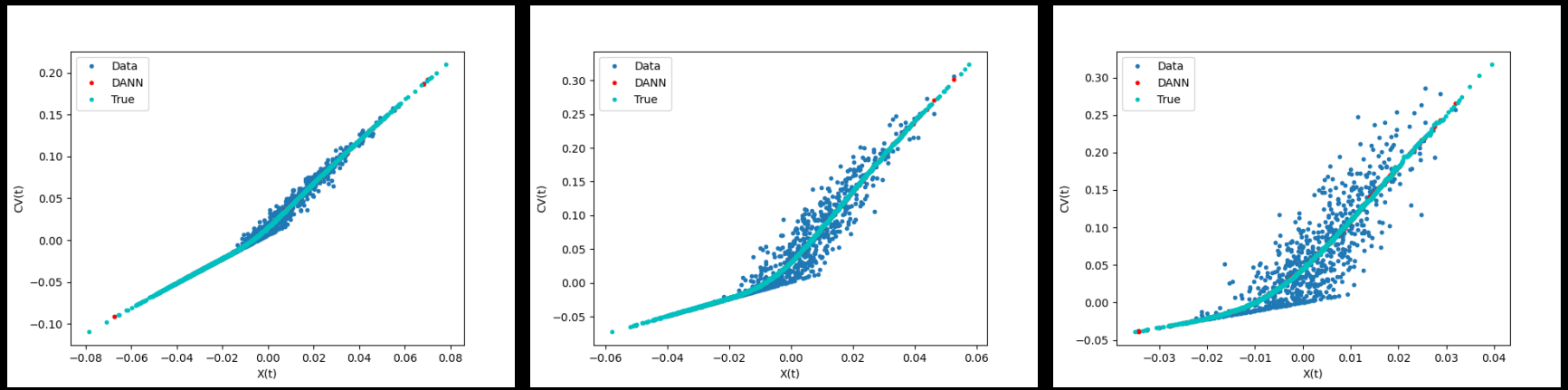
- Recursive DANN structured design.



Sequence of DANNs encapsulating the exercise policy.

Sampled (labels) payoffs

- Training the DANNs with highly noisy labels.
- The differential labels (also noisy) are obtained by AAD.



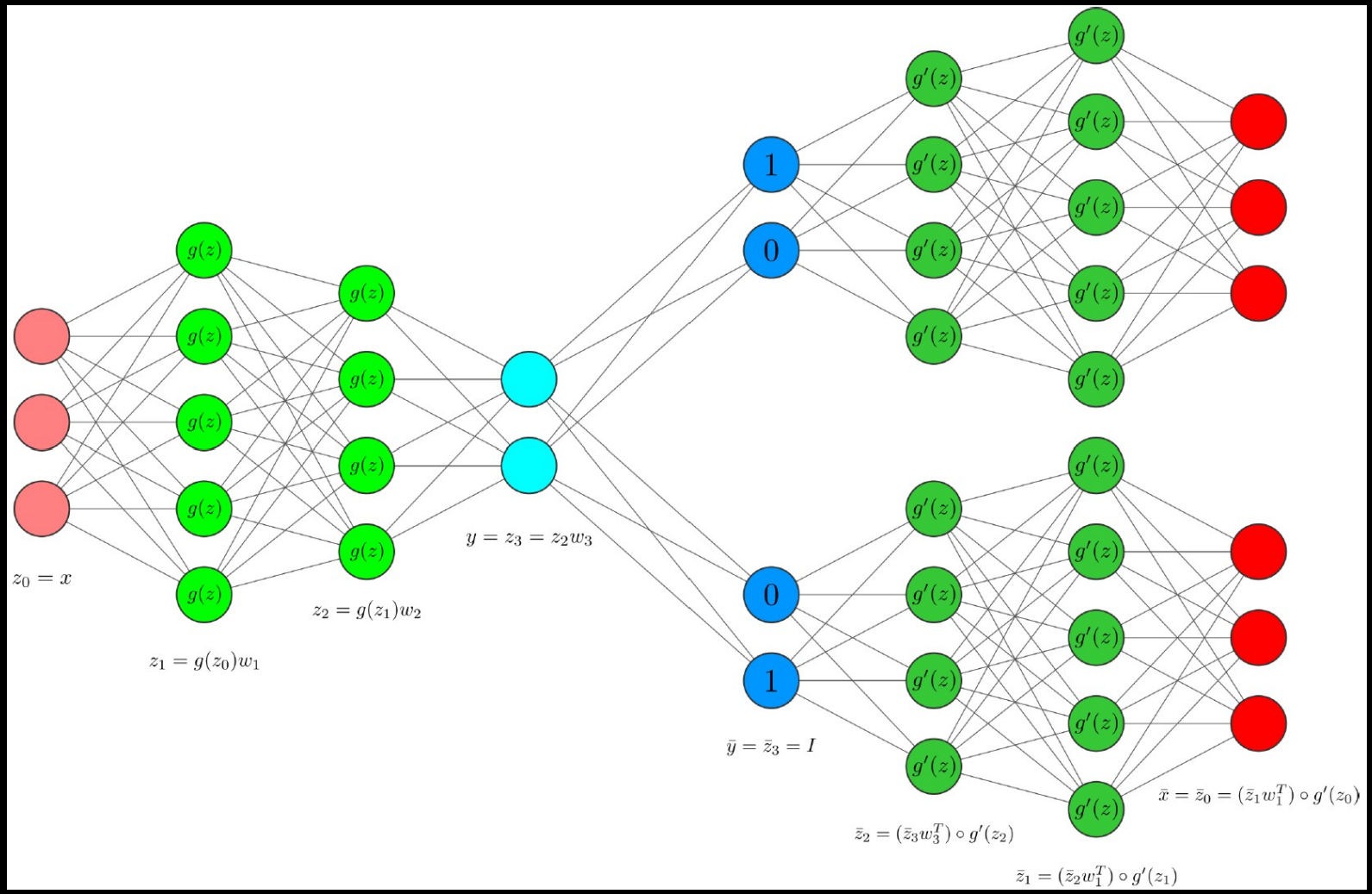
(a) $t = 7$ (b) $t = 4$ (c) $t = 1$

Sampled payoffs and Backward DANN approximation

Joint learning

- It is employed in the computation of the final price.
- Not only the cIRS is estimated, but also extra related products.
- Here, a set of European swaptions whose maturities coincide with each of the cancellation times (*coterminal* swaptions).
- Why? The Bermudan derivatives can be seen as a combination of a number of their European counterparts.
- Further, the values of the European coterminal swaptions are often available in closed-form for many models (LGM).
- The last aspect is of great importance: while the labels for the cIRS are noisy prices (sampled payoffs), the labels/prices for the coterminal European swaptions are the ground truth.
- Intuitively, adding labels with exact (non-noisy) values should help to improve the estimations provided by the whole DANN (at a prescribed training time budget), besides more quantities need to be predicted, boosted by the joint learning effect.

Joint learning (with DANNs)



DANN structure considering multiple outputs, i.e., integrating the joint learning approach.

Training set generation (I)

- Two aspects have a paramount importance: the domain of the input space and the sampling distribution within that domain.
- The first depends on the problem at hand, so the domain in derivative's valuation is often selected based on the observed market behaviour (past, present and future) or experience.
- The second aspect is trickier since, although a uniform distribution seems often appropriate, an smart sampling taking into account other factors like regions of more interest or with more error might provide a significant prediction improvement of the ANN model (given a training time budget).
- Also the relation between the model and market parameters is an important aspect to be exploited, which can be employed to avoid regions that represent unrealistic financial situations.
- Once the inputs are generated, one single Monte Carlo path is simulated to evolve the LGM model.

Training set generation (II)

- **Mean reversion,** $\kappa = \mathcal{U}(l_\kappa, u_\kappa)$.
- **Discount factors**: interest rate curves $\rightarrow$ discount curves

$$R(t) = \beta_0 G_0(t) + \beta_1 G_1(t) + \beta_2 G_2(t),$$

where

$$\beta_0 = \mathcal{U}(l_0, u_0), \quad \beta_1 = \mathcal{U}(l_1, u_1), \quad \beta_2 = \mathcal{U}(l_2, u_2), \quad \tau = \mathcal{U}(l_\tau, u_\tau).$$

The discount curve is constructed as $D(t) = \exp\left(-\int_0^t R(s)ds\right)$.

- **Fixed rate**. The fixed rate K, is perturbed from the ATM level:

$$K = ATM + \Delta K,$$

where

$$ATM = \frac{1 - D(T_M)}{\sum_{i=1}^{M} \Delta T_i D(T_i)},$$

with $\Delta K \in \mathcal{U}(l_K, u_K)$.

Training set generation (III)

- **LGM volatility**, α: assumed to be a piecewise constant function of the time with a dependency on κ (as observed in the market).
- How? First, Rebonato's parameterization

$$h(t) = (a + bt)\exp(-ct) + d$$

  Then, we choose implied volatilities as $\Sigma_j = h\left(\frac{t_{j-1}+t_j}{2}\right)$

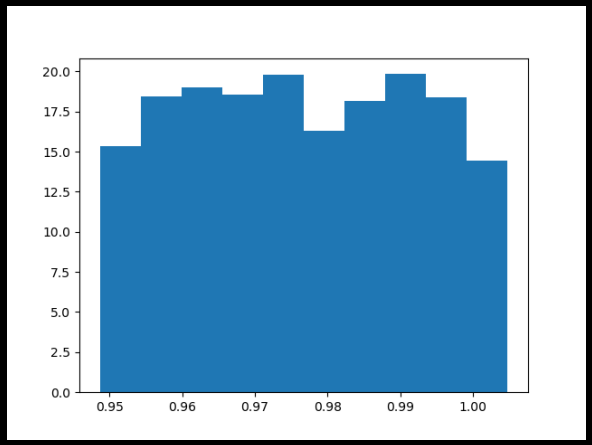
- As $\Sigma_j^2 \Delta t_j = \alpha_j^2 \int_{t_{j-1}}^{t_j} \exp\left(-2\kappa(t_j - s)\right) ds$, then $\alpha_j^2 = 2\kappa \frac{\Sigma_j^2 \Delta t_j}{1-\exp(-\kappa \Delta t_j)}$
- Finally,

$$\alpha(t) = \begin{cases} \alpha_1, & t \in (t_0, t_1], \\ \alpha_2, & t \in (t_1, t_2], \\ \vdots & \vdots \\ \alpha_J, & t \in (t_{J-1}, t_J]. \end{cases}$$

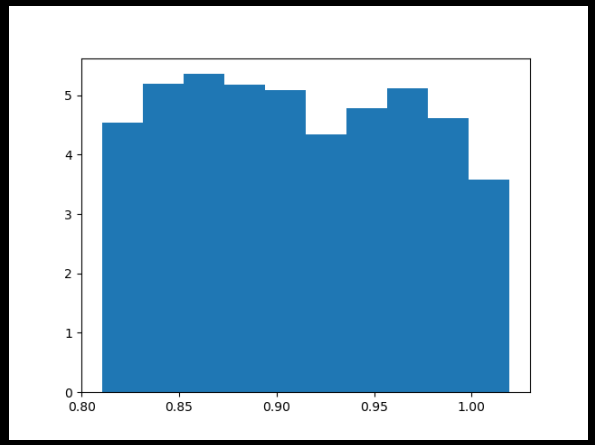
- The values of the Rebonato's model parameters are sampled by

$$a = \mathcal{U}(l_a, u_a), \quad b = \mathcal{U}(l_b, u_b), \quad c = \mathcal{U}(l_c, u_c), \quad d = \mathcal{U}(l_d, u_d).$$

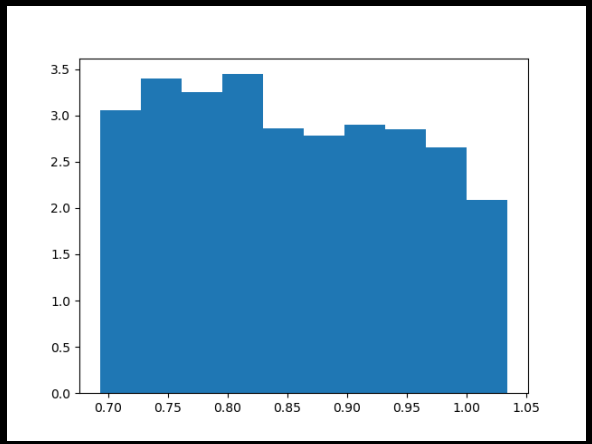
Training set generation (IV)



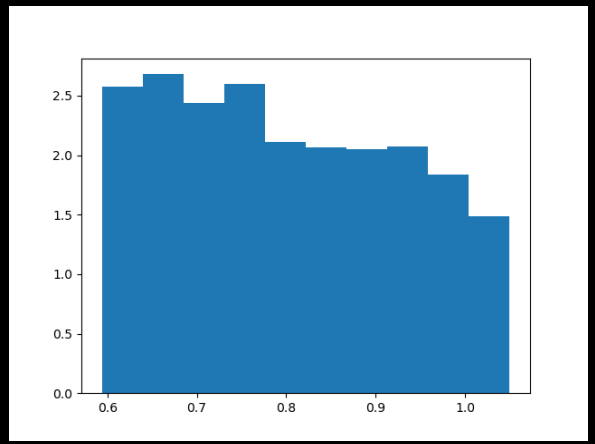
(a) $t = 1$.



(b) $t = 4$.



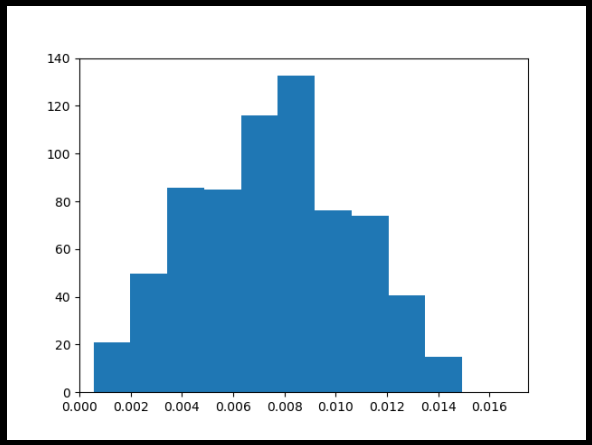
(c) $t = 7$.



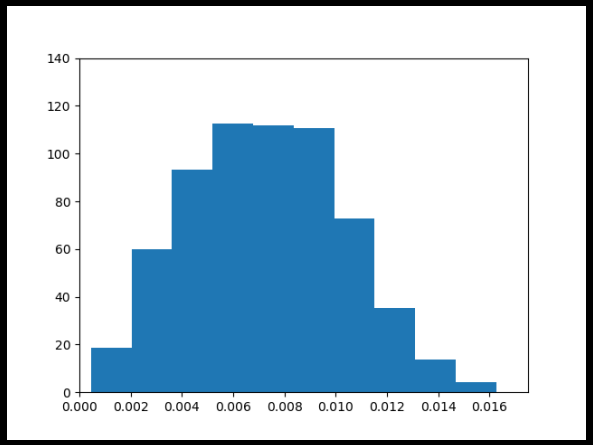
(d) $t = 10$.

Histograms of the discount factors at different time instants.

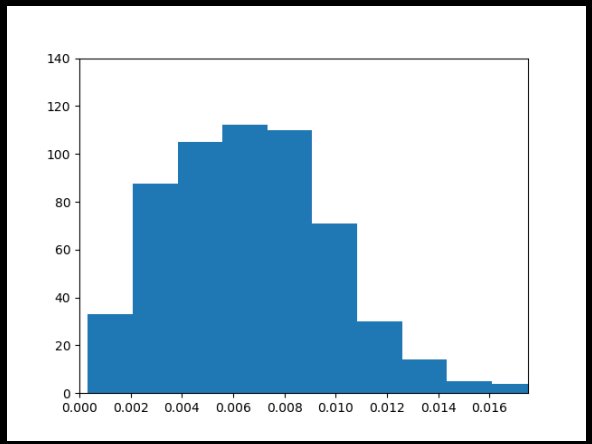
Training set generation (and V)



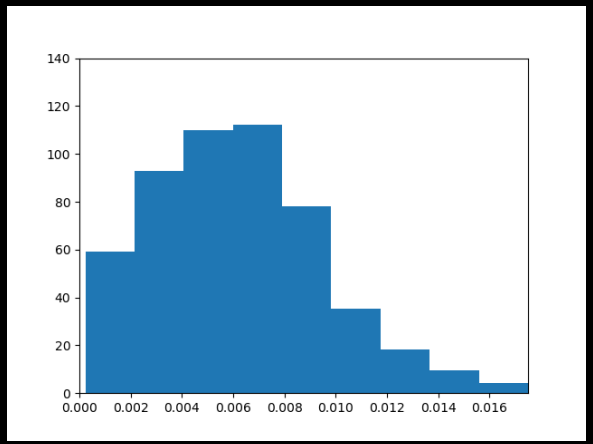
(a) α_1.



(b) α_2.



(c) α_3.



(d) α_4.

Histograms of the volatilites.

Numerical experiments

- Incremental testing in terms of the inputs' domain.
- The results are presented in the form of *differences*' histograms, including an *interquantile confidence interval*.
- The ground truth values (prices of cIRS) of the validation set are computed by a highly converged Monte Carlo pricer (acc: $\sim$ 1 bp)
- ANNs' hyperparameters configuration:

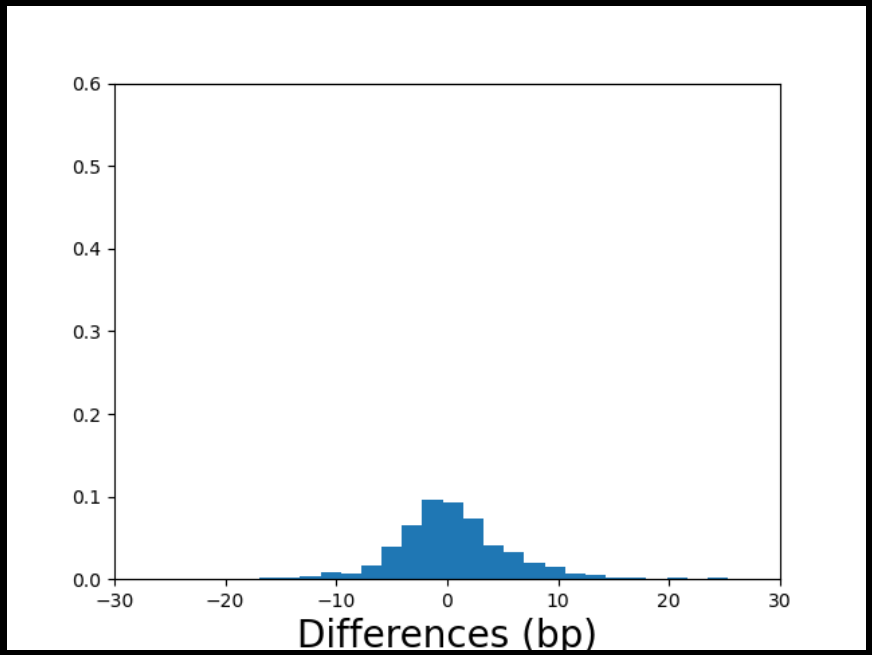
Hyperparameter	Value
Layers	4
Neurons	32
Epochs	128
Batch size	4096

Test base cases

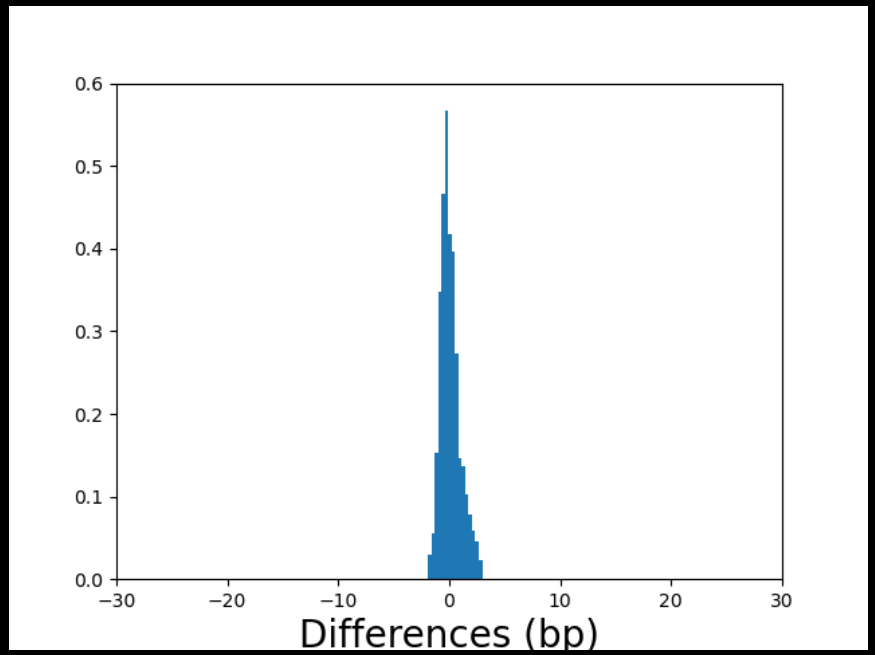
Test Case I	Test Case II
$l_\kappa = -0.05, \quad u_\kappa = 0.1$	$l_\kappa = -0.05, \quad u_\kappa = 0.1$
$l_a = -10^{-5}, \quad u_a = 10^{-5}$	$l_a = 10^{-5}, \quad u_a = 0.0075$
$l_b = -10^{-5}, \quad u_b = 10^{-5}$	$l_b = 0, \quad u_b = 0.0005$
$l_c = -10^{-5}, \quad u_c = 10^{-5}$	$l_c = 0, \quad u_c = 0.25$
$l_d = 0.0075 - 10^{-5}, \quad u_d = 0.0075 + 10^{-5}$	$l_d = 10^{-5}, \quad u_d = 0.0075$
$l_o = 0.02 - 10^{-5}, \quad u_o = 0.02 + 10^{-5}$	$l_o = 0.02 - 10^{-5}, \quad u_o = 0.02 + 10^{-5}$
$l_1 = -10^{-5}, \quad u_1 = 10^{-5}$	$l_1 = -10^{-5}, \quad u_1 = 10^{-5}$
$l_2 = -10^{-5}, \quad u_2 = 10^{-5}$	$l_2 = -10^{-5}, \quad u_2 = 10^{-5}$
$l_\tau = 1 - 10^{-5}, \quad u_\tau = 1 + 10^{-5}$	$l_\tau = 1 - 10^{-5}, \quad u_\tau = 1 + 10^{-5}$
$l_K = -10^{-5}, \quad u_K = 10^{-5}$	$l_K = -10^{-5}, \quad u_K = 10^{-5}$

Test Case III	Test Case IV
$l_\kappa = -0.05, \quad u_\kappa = 0.1$	$l_\kappa = -0.05, \quad u_\kappa = 0.1$
$l_a = 10^{-5}, \quad u_a = 0.0075$	$l_a = 10^{-5}, \quad u_a = 0.0075$
$l_b = 0, \quad u_b = 0.0005$	$l_b = 0, \quad u_b = 0.0005$
$l_c = 0, \quad u_c = 0.25$	$l_c = 0, \quad u_c = 0.25$
$l_d = 10^{-5}, \quad u_d = 0.0075$	$l_d = 10^{-5}, \quad u_d = 0.0075$
$l_o = -0.005, \quad u_o = 0.05$	$l_o = -0.005, \quad u_o = 0.05$
$l_1 = 0, \quad u_1 = 0.001$	$l_1 = 0, \quad u_1 = 0.001$
$l_2 = 0, \quad u_2 = 0.01$	$l_2 = 0, \quad u_2 = 0.01$
$l_\tau = 0.01, \quad u_\tau = 2$	$l_\tau = 0.01, \quad u_\tau = 2$
$l_K = -10^{-5}, \quad u_K = 10^{-5}$	$l_K = -0.01, \quad u_K = 0.01$

Impact of joint learning (I)



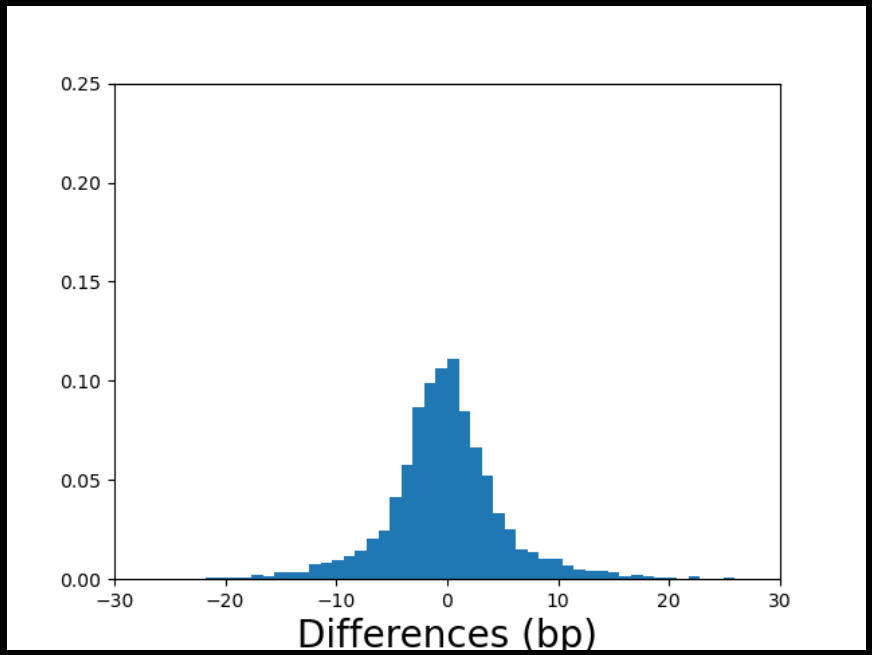
(a) $(Q_{10}Q_{90}) = (-5.7, 7.0)$.



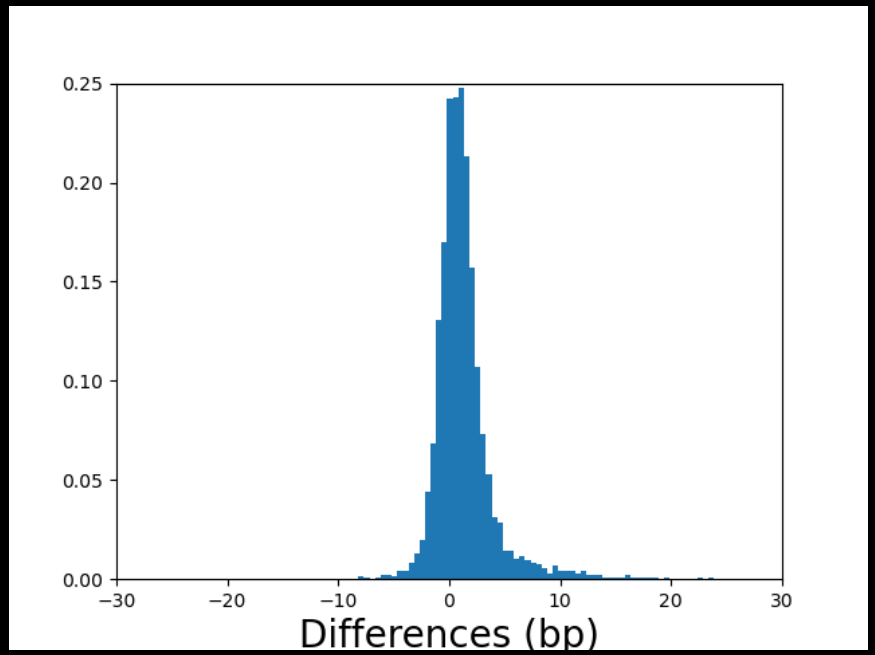
(b) $(Q_{10}, Q_{90}) = (-0.8, 1.4)$.

Pricing differences in basis points of Test Case I: Plain DANN (left) and DANN with joint learning (right).

Impact of joint learning (II)



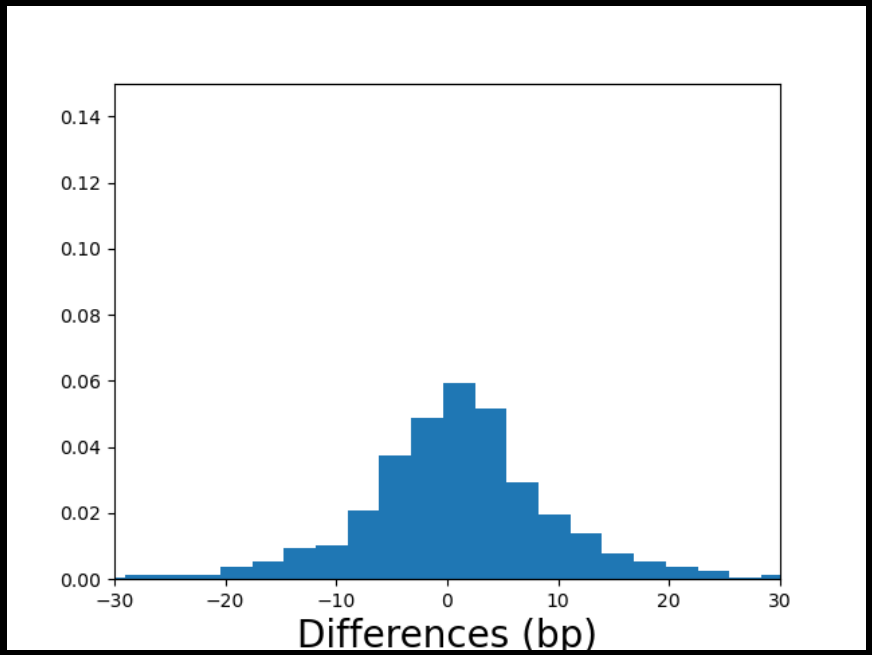
(a) $(Q_{10}, Q_{90}) = (-6.0, 5.7)$.



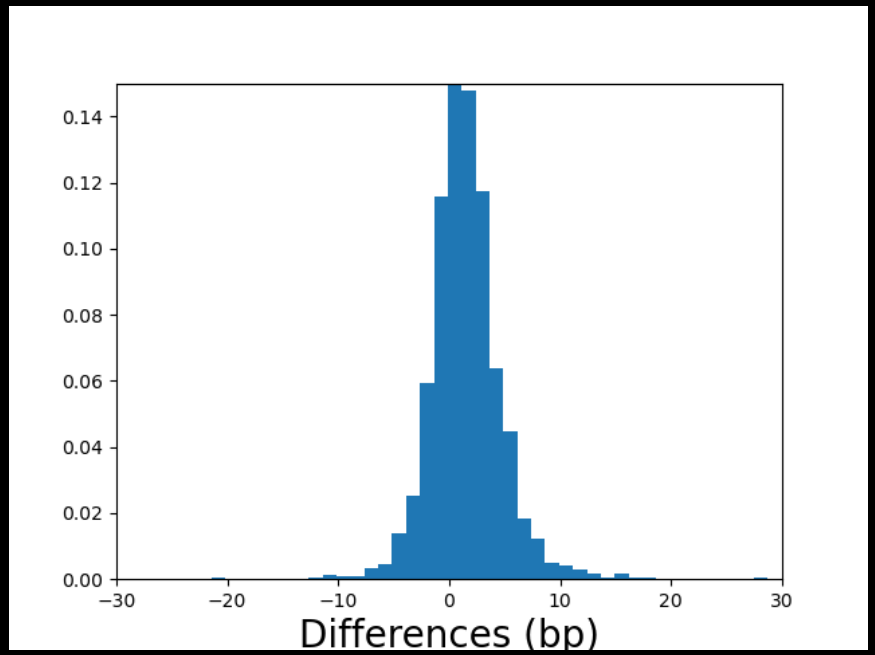
(b) $(Q_{10}, Q_{90}) = (-1.0, 3.7)$.

Pricing differences in basis points of Test Case II: Plain DANN (left) and DANN with joint learning (right).

Impact of joint learning (III)



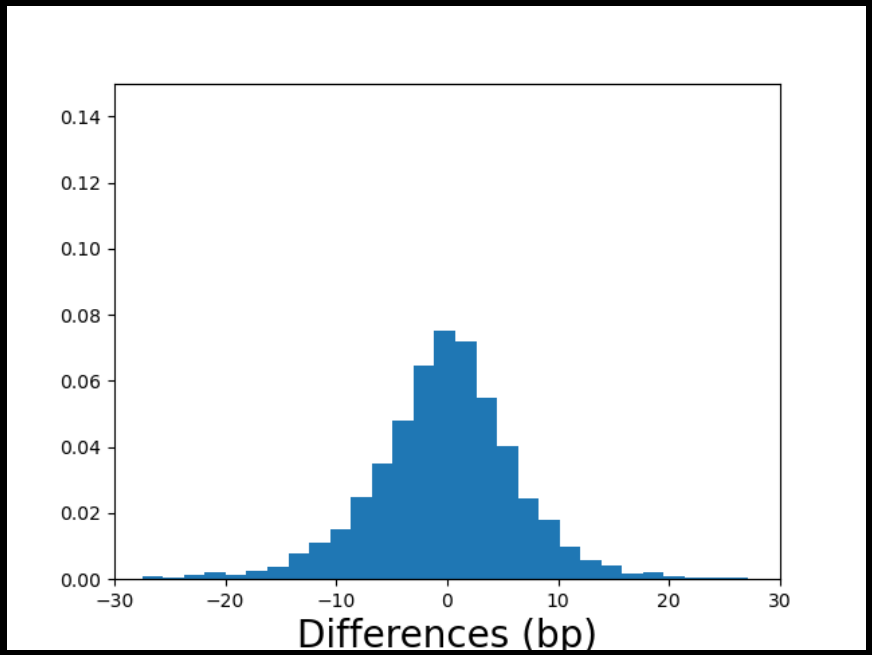
(a) $(Q_{10}, Q_{90}) = (-10.0, 12.4)$.



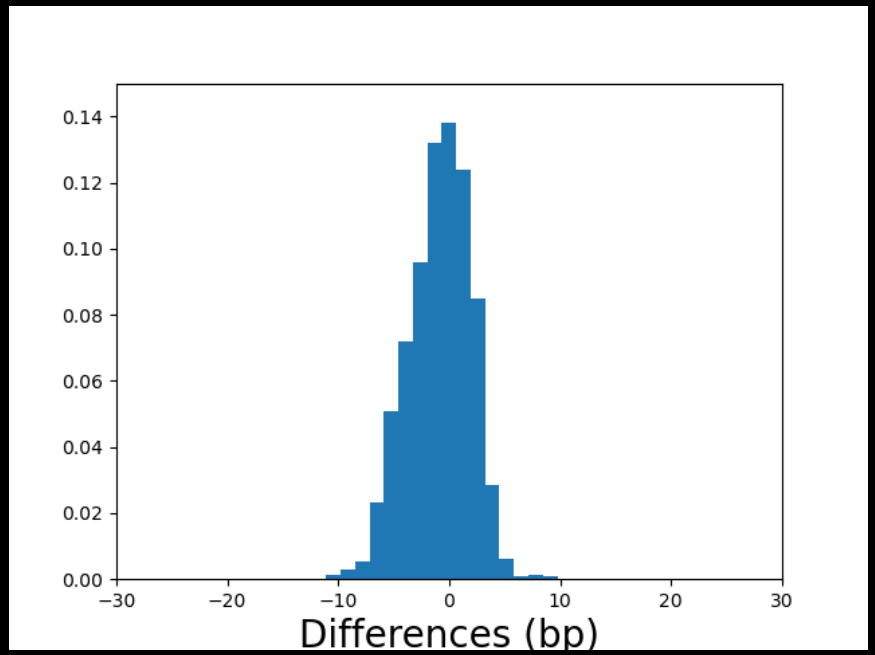
(b) $(Q_{10}, Q_{90}) = (-1.9, 5.1)$.

Pricing differences in basis points of Test Case III: Plain DANN (left) and DANN with joint learning (right).

Impact of joint learning (IV)



(a) $(Q_{10}, Q_{90}) = (-8.1, 7.4)$.



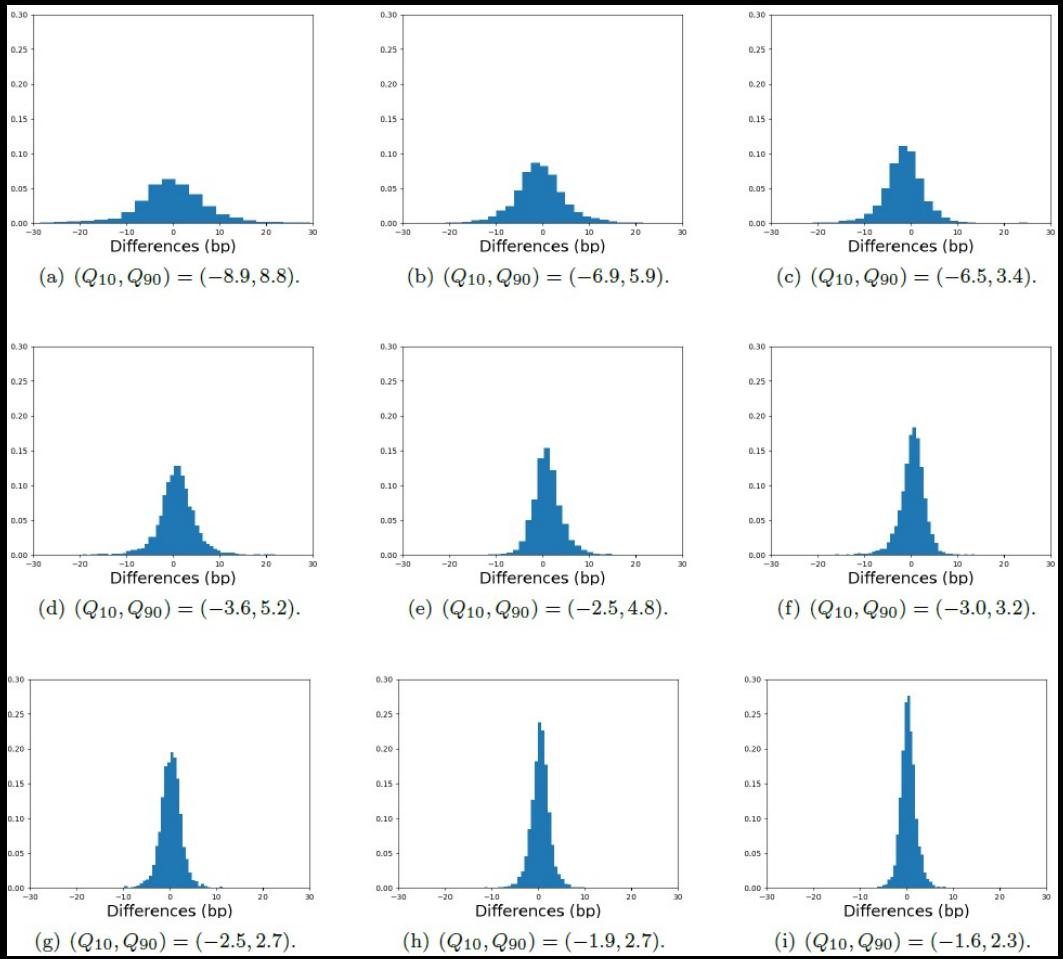
(b) $(Q_{10}, Q_{90}) = (-4.8, 2.5)$.

Pricing differences in basis points of Test Case IV: Plain DANN (left) and DANN with joint learning (right).

Impact of joint learning (IV)

- In all cases the differences' distributions are centered at cero, thus indicating that the DANN predictions do not present bias.
- The incorporation of the inputs related with the discount factors and the volatility seems to make the solution to be approximated more challenging.
- When the strike spread is included (test case IV) the DANN provides slightly better estimations. Although this might seem counterintuitive, that behavior appears due to the effect of the strikes far from ATM level (particularly those more in-the-money). This effect is implicitly present in the histograms, where a certain skewness is observed (more clearly visible in the case employing joint learning.
- An impressive (and general) reduction of the error thanks to the joint learning approach is achieved, where both the average error and the interquantile interval are, at least, halved.

Impact of number of samples and MC paths/sample (I)



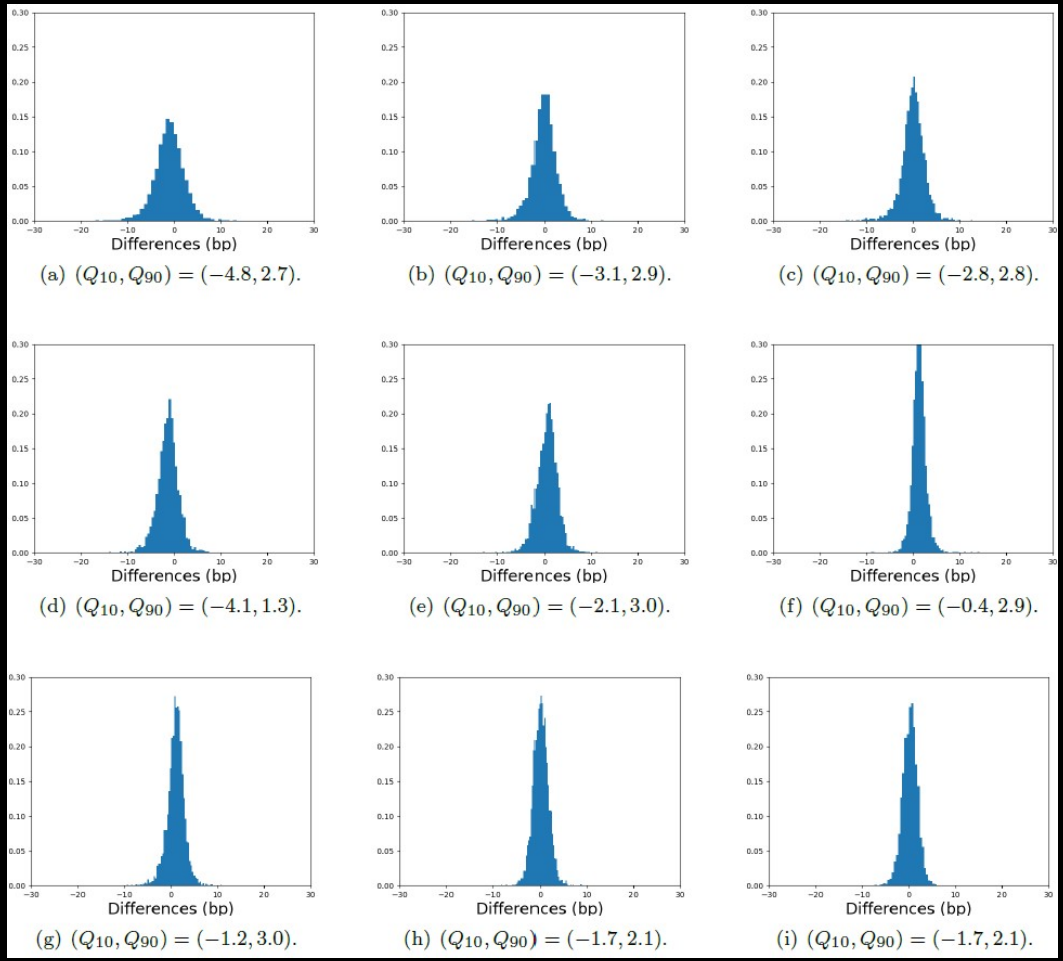
(a) $(Q_{10}, Q_{90}) = (-8.9, 8.8)$. (b) $(Q_{10}, Q_{90}) = (-6.9, 5.9)$. (c) $(Q_{10}, Q_{90}) = (-6.5, 3.4)$.

(d) $(Q_{10}, Q_{90}) = (-3.6, 5.2)$. (e) $(Q_{10}, Q_{90}) = (-2.5, 4.8)$. (f) $(Q_{10}, Q_{90}) = (-3.0, 3.2)$.

(g) $(Q_{10}, Q_{90}) = (-2.5, 2.7)$. (h) $(Q_{10}, Q_{90}) = (-1.9, 2.7)$. (i) $(Q_{10}, Q_{90}) = (-1.6, 2.3)$.

Plain DANN: Columns: $n_f = 2^{22}$ (left), $n_f = 2^{23}$ (central), $n_f = 2^{24}$ (right); Rows: $n_{MC} = 1$ (top), $n_{MC} = 4$ (middle), $n_{MC} = 16$ (bottom).

Impact of number of samples and MC paths/sample (II)



(a) $(Q_{10}, Q_{90}) = (-4.8, 2.7)$. (b) $(Q_{10}, Q_{90}) = (-3.1, 2.9)$. (c) $(Q_{10}, Q_{90}) = (-2.8, 2.8)$.

(d) $(Q_{10}, Q_{90}) = (-4.1, 1.3)$. (e) $(Q_{10}, Q_{90}) = (-2.1, 3.0)$. (f) $(Q_{10}, Q_{90}) = (-0.4, 2.9)$.

(g) $(Q_{10}, Q_{90}) = (-1.2, 3.0)$. (h) $(Q_{10}, Q_{90}) = (-1.7, 2.1)$. (i) $(Q_{10}, Q_{90}) = (-1.7, 2.1)$.

Joint DANN: Columns: $n_f = 2^{22}$ (left), $n_f = 2^{23}$ (central), $n_f = 2^{24}$ (right); Rows: $n_{MC} = 1$ (top), $n_{MC} = 4$ (middle), $n_{MC} = 16$ (bottom).

Impact of number of samples and MC paths/sample (and III)

- As expected, systematically increase the number of samples provided to the DANN improves the predictions, although the reduction of the interquantile intervals, i.e., in the deviations' variance, is rather limited.
- In contrast to the previous point, we again observe that the DANN trained relying on the joint learning approach provides more accurate estimations, significantly reducing the variance.
- The effect of including more Monte Carlo paths per sample presents the expected behavior, i.e., when the number of paths is multiplied by four the error is approximately halved (according to the theoretical convergence rate of Monte Carlo methods, $n^{-1/2}$).
- When most of the differences fall below ± 3 basis points, a certain level of saturation is observed meaning that considering either more samples or more Monte Carlo paths per sample no longer reduces the deviations in the predictions (or the reduction results to be negligible).

Conclusions

- An innovative solution for the Bermudan swaption valuation based on advanced deep learning techniques has been proposed.
- Additional very relevant components have been added on top of classical ANN approach.
- Some are appropriate adaptations of existing ideas: sampled (labels) payoffs and differential machine learning.
- Novel training strategy in quantitative finance: the joint learning.
- The idea behind our joint learning approach is to incorporate “similar” financial products as outputs, aiming that they help in the training process to reach more accurate solutions for complex derivatives at less/similar computational cost.
- Throughout several experiments, the advantages of using the proposed joint learning-based training have been highlighted.

References

[1] Patrick S. Hagan. *Evaluating and hedging exotic swap instruments via LGM*. Available at ResearchGate. 2002.

[2] Brian Huge and Antoine Savine. *Differential machine learning*. Available at ArXiv: 2005.02347. 2020.

[3] Francisco Gómez Casanova et al. *Deep Joint Learning valuation of Bermudan Swaptions*. Available at ArXiv: 2404.11257. 2024.

Acknowledgements & Questions

Thanks to the support of Centre for Information and Communications Technology Research (CITIC). CITIC is funded by the Xunta de Galicia through the collaboration agreement between the Consellería de Cultura, Educación, Formación Profesional e Universidades and the Galician universities for the reinforcement of the research centres of the Galician University System (CIGUS).

More: `alvaroleitao.github.io`

Eskerrik asko!