Multi-Asset Option Market Simulation

Magnus Wiese

Joint work with: Ben Wood, Alexandre Pachoud, Ralf Korn, Hans Buehler, Phillip Murray, Lianjun Bai

J.P. Morgan, London
University of Kaiserslautern

March 8, 2022

[0] Opinions expressed in this presentation are those of the authors, and do not necessarily reflect the view of J.P. Morgan.

Overview

Deep Hedging

Problem setting

- Deep Hedging (DH): reinforcement learning-based automation of hedging a portfolio of derivatives under market frictions; e.g. tx costs, liquidity constraints.

Deep Hedging

Problem setting

- Deep Hedging (DH): reinforcement learning-based automation of hedging a portfolio of derivatives under market frictions; e.g. tx costs, liquidity constraints.
- **Problem**: reinforcement learning needs lots of data. Lack of historical sample: on a daily scale past 10 years account to ~ 2500 samples.

Deep Hedging

Problem setting

- Deep Hedging (DH): reinforcement learning-based automation of hedging a portfolio of derivatives under market frictions; e.g. tx costs, liquidity constraints.
- **Problem**: reinforcement learning needs lots of data. Lack of historical sample: on a daily scale past 10 years account to ~ 2500 samples.
- Where is all that data going to come from?

Deep Hedging

Problem setting

- Deep Hedging (DH): reinforcement learning-based automation of hedging a portfolio of derivatives under market frictions; e.g. tx costs, liquidity constraints.
- **Problem**: reinforcement learning needs lots of data. Lack of historical sample: on a daily scale past 10 years account to ~ 2500 samples.
- Where is all that data going to come from?

Approach

- Calibrate a spot and equity option market simulator to historical market data to approximate the conditional dynamics of the market.
- Train DH on simulated data.

How do we represent the market?

$(\Omega, \mathcal{F} = (\mathcal{F}_t)_{t \in \mathbb{N}}, \mathbb{P})$

Equity spot and option market process

$$X = (S, C) : \Omega \times \mathbb{N}_0 \to \mathbb{R}_{>0} \times \mathbb{R}^{mn}_{>0}$$

$S = (S_t)_t$: spot price process taking values in $\mathbb{R}_{>0}$
$C = (C_t)_t$: call price process defined on a grid

$$C_t = \begin{pmatrix} C_t(\tau_1, k_1) & \cdots & C_t(\tau_1, k_n) \\ \vdots & \ddots & \vdots \\ C_t(\tau_m, k_1) & \cdots & C_t(\tau_m, k_n) \end{pmatrix}$$

where $C_t(\tau_i, k_j)$ has payoff $(S_{t+\tau_i}/S_t - k_j)^+$.

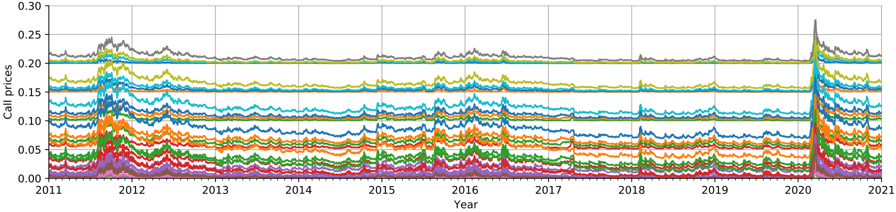

Figure: Eurostoxx 50 call grid prices; $\mathcal{K} = (0.8, 0.85, \ldots, 1.2)$, $\mathcal{T} = (20, 40, 60, 120)$.

Static arbitrage

Call prices need to fulfil ordering constraints; otherwise will exhibit **static arbitrage** (riskless profits).

$$C(\tau_{i+1}, \cdot) - C(\tau_j, \cdot) \geq 0; \;\; i = 1, \dots, m-1$$
$$C(\cdot, k_{j+1}) - 2C(\cdot, k_j) + C(\cdot, k_{j-1}) \geq 0; \;\; j = 1, \dots, n-1$$

Guaranteeing no static arbitrage

Map grid prices to the L_1-closest arb-free grid of prices. Use an equivalent arb-free representation **discrete local volatilities (DLVs)** [3] and fulfil simpler constraints (positivity) to ensure no static arb.

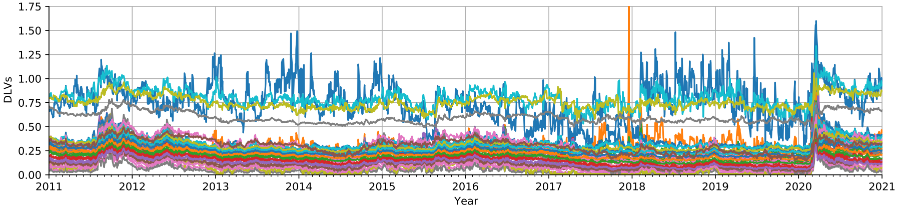

Figure: Eurostoxx 50 DLVs; $\mathcal{K} = (0.8, 0.85, \dots, 1.2)$, $\mathcal{T} = (20, 40, 60, 120)$.

Statistical arbitrage

Statistical arbitrage: abitrage in expectation, may still be present in the current simulator as we do not require X to be a martingale. For example,

$$C_t(\tau, k) \neq \mathbb{E}_{\mathbb{P}}\left((S_{t+\tau}/S_t - k)^+ | \mathcal{F}_t\right)$$

expected realized prices do not have to coincide with simulated market prices.

Current work on stat-arb removal:

- "Deep Hedging: Learning to Remove the Drift under Trading Frictions with Minimal Equivalent Near-Martingale Measures" [2] published in *Risk - Cutting Edge*
- "Risk-Neutral Market Simulation" [17] presented at *AAAI 2022 WFS*

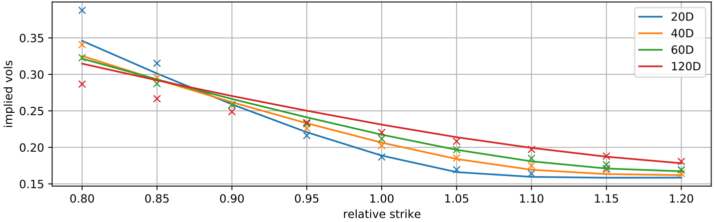

Figure: IV grid (line) at $t = 0$ and *realized* IV grid (x).

Problem formulation

$(x_t)_{t=1}^{T} \sim p$: finite realization, p real-world density associated to $\mathbb{P}$.

[1]Notation: $x_{t-q+1:t} = (x_{t-q+1}, \dots, x_t)$

Problem formulation

$(x_t)_{t=1}^T \sim p$: finite realization, p real-world density associated to $\mathbb{P}$.

- **Proximity to real-data**: fit model density p_θ such that the conditional densities are *close*[1]

$$p(x_{t+1}|x_{t-q+1:t}) \approx p_\theta(x_{t+1}|x_{t-q+1:t})$$

- **Easy to sample**:

$$X_{t+1} = F_\theta^{-1}(U_{t+1}; X_{t-q+1:t})$$

where U_{t+1} is iid adapted noise, e.g. multivariate uniform.

[1]Notation: $x_{t-q+1:t} = (x_{t-q+1}, \ldots, x_t)$

Problem formulation

$(x_t)_{t=1}^T \sim p$: finite realization, p real-world density associated to $\mathbb{P}$.

- **Proximity to real-data**: fit model density p_θ such that the conditional densities are *close*[1]

$$p(x_{t+1}|x_{t-q+1:t}) \approx p_\theta(x_{t+1}|x_{t-q+1:t})$$

- **Easy to sample**:

$$X_{t+1} = F_\theta^{-1}(U_{t+1}; X_{t-q+1:t})$$

  where U_{t+1} is iid adapted noise, e.g. multivariate uniform.

How do we calibrate F_θ?

1. **Compression**: reduce DLV grid to a small set of factors
2. **Density calibration**: calibrating the conditional density on the compressed space

[1] Notation: $x_{t-q+1:t} = (x_{t-q+1}, \dots, x_t)$

Calibration phase 1: DLV compression

Call prices / DLVs and their returns are highly correlated:

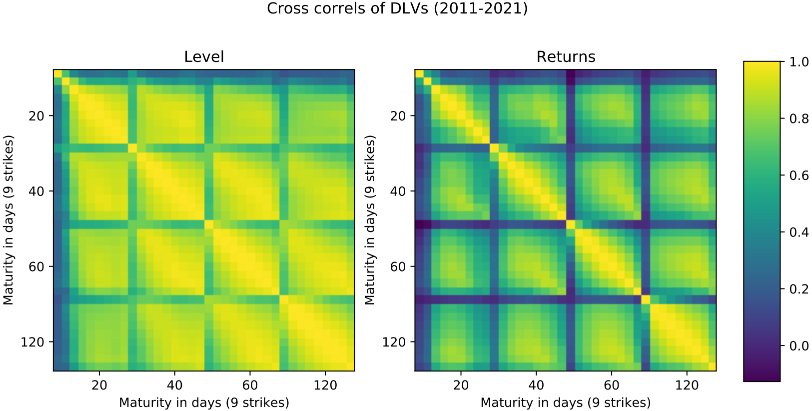

Figure: Cross correlation matrix of DLV levels (left) and DLV returns (right). Strike-maturity pairs show higher correlation for proximate strikes and maturities.

Autoencoders

DLVs are highly compressible: $\Rightarrow$ learn a low-dimensional *efficient* representation of the factors.

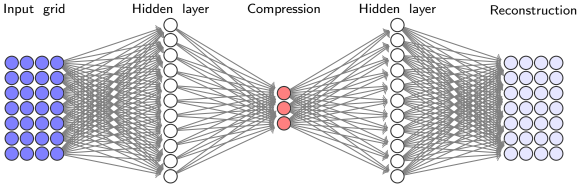

Figure: Illustration of a shallow DLV autoencoder $(28 - 11 - 3 - 11 - 28)$.

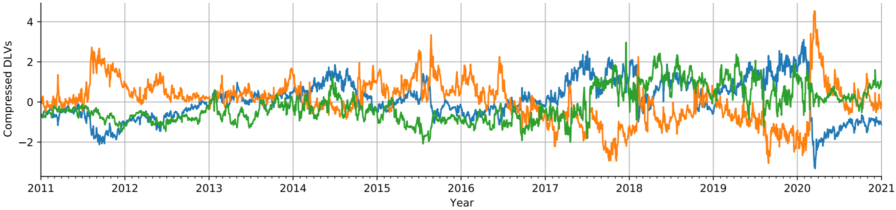

Figure: Three-dimensional endo representation.

Comparison: autoencoders vs. principal component analysis for compressing DLVs

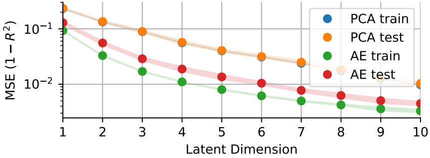

Figure: Comparison of reconstruction error (MSE) obtained from calibrated auteoncoders and PCA for different latent dimensions.

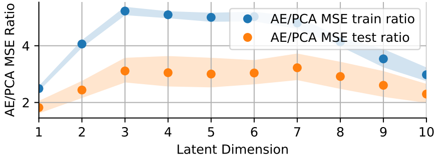

Figure: Ratio (AE / PCA) of reconstruction errors on train and test set. AE is approximately twice as efficient in compressing DLVs in terms of reconstruction erros.

Calibration phase 2: conditional density calibration

Literature overview

- Parametric; VAR / GARCH / SV blends
- Generative adversarial networks [10]

$$\min_{G_\theta} \max_{D} \mathbb{E}_p(\ln(D(X_{t+1}|X_{t-q+1:t}))) + \mathbb{E}_{p_\theta}(\ln(1 - D(X_{t+1}|X_{t-q+1:t})))$$

- Signature MMD [4, 14]

$$\text{Sig-MMD}(p, p_\theta) = \|\mathbb{E}_p(S(X_{t+1}|X_{t-q+1:t})) - \mathbb{E}_{p_\theta}(S(X_{t+1}|X_{t-q+1:t}))\|_2$$

- Neural SDEs [5, 6]
- Normalizing flows [15]; KL divergence

$$\text{KL}(p, p_\theta) = \mathbb{E}_p\left(\mathbb{E}_p\left(\ln \frac{p(X_{t+1}|X_{t-q+1:t})}{p_\theta(X_{t+1}|X_{t-q+1:t})} \Big| X_{t-q+1:t}\right)\right)$$

Calibration phase 2: conditional density calibration

Literature overview

- Parametric; VAR / GARCH / SV blends
- Generative adversarial networks [10]

$$\min_{G_\theta} \max_{D} \mathbb{E}_p(\ln(D(X_{t+1}|X_{t-q+1:t}))) + \mathbb{E}_{p_\theta}(\ln(1 - D(X_{t+1}|X_{t-q+1:t})))$$

- Signature MMD [4, 14]

$$\text{Sig-MMD}(p, p_\theta) = \|\mathbb{E}_p(S(X_{t+1}|X_{t-q+1:t})) - \mathbb{E}_{p_\theta}(S(X_{t+1}|X_{t-q+1:t}))\|_2$$

- Neural SDEs [5, 6]
- Normalizing flows [15]; KL divergence

$$\text{KL}(p, p_\theta) = \mathbb{E}_p\left(\mathbb{E}_p\left(\ln \frac{p(X_{t+1}|X_{t-q+1:t})}{p_\theta(X_{t+1}|X_{t-q+1:t})} \middle| X_{t-q+1:t}\right)\right)$$

How do we calibrate a conditional density p_θ with NNs?

Inverse sampling theorem in multi-dimensions

$X = (X_1, X_2, \ldots, X_d)$: wlog $[0,1]^d$-valued random variable with *conditional* CDFs $F = (F_1, \ldots, F_d)$:

$$\begin{aligned} F_1(x_1) &= \mathbb{P}(X_1 \leq x_1) \\ F_2(x_2; x_1) &= \mathbb{P}(X_2 \leq x_2 | X_1 = x_1) \\ &\vdots \\ F_d(x_d; x_1, \ldots, x_{d-1}) &= \mathbb{P}(X_d \leq x_d | X_1 = x_1, \ldots, X_{d-1} = x_{d-1}) \end{aligned}$$

Inverse sampling theorem in multi-dimensions

$X = (X_1, X_2, \dots, X_d)$: wlog $[0,1]^d$-valued random variable with *conditional* CDFs $F = (F_1, \dots, F_d)$:

$$
\begin{aligned}
F_1(x_1) &= \mathbb{P}(X_1 \leq x_1) \\
F_2(x_2; x_1) &= \mathbb{P}(X_2 \leq x_2 | X_1 = x_1) \\
&\vdots \\
F_d(x_d; x_1, \dots, x_{d-1}) &= \mathbb{P}(X_d \leq x_d | X_1 = x_1, \dots, X_{d-1} = x_{d-1})
\end{aligned}
$$

Inverse sampling theorem:

$$
\begin{aligned}
X_1 &= F_1^{-1}(U_1) \\
X_2 &= F_2^{-1}(U_2; X_1) \\
&\vdots \\
X_d &= F_d^{-1}(U_d; X_1, \dots, X_{d-1})
\end{aligned}
$$

Inverse sampling theorem in multi-dimensions

$X = (X_1, X_2, \ldots, X_d)$: wlog $[0,1]^d$-valued random variable with *conditional* CDFs $F = (F_1, \ldots, F_d)$:

$$\begin{aligned}
F_1(x_1) &= \mathbb{P}(X_1 \leq x_1) \\
F_2(x_2; x_1) &= \mathbb{P}(X_2 \leq x_2 | X_1 = x_1) \\
&\vdots \\
F_d(x_d; x_1, \ldots, x_{d-1}) &= \mathbb{P}(X_d \leq x_d | X_1 = x_1, \ldots, X_{d-1} = x_{d-1})
\end{aligned}$$

Inverse sampling theorem:

$$\begin{aligned}
X_1 &= F_1^{-1}(U_1) \\
X_2 &= F_2^{-1}(U_2; X_1) \\
&\vdots \\
X_d &= F_d^{-1}(U_d; X_1, \ldots, X_{d-1})
\end{aligned}$$

Density:

$$p_\theta(x_k | x_1, \ldots, x_{k-1}) = \partial_{x_k} F_k(x_k; x_1, \ldots, x_{k-1})$$

Neural spline flow construction

How can we approximate the conditional CDFs with neural networks?

$$F_k(x_k; x_1, \ldots, x_{k-1}) = \mathbb{P}(X_k \leq x_k | X_1 = x_1, \ldots, X_{k-1} = x_{k-1}), k = 2, \ldots, d$$

Neural spline flow construction

How can we approximate the conditional CDFs with neural networks?

$$F_k(x_k; x_1, \dots, x_{k-1}) = \mathbb{P}(X_k \leq x_k | X_1 = x_1, \dots, X_{k-1} = x_{k-1}), k = 2, \dots, d$$

$0 = u_0 < u_1 < \cdots < u_B = 1$: partition of $[0, 1]$
$NN_k : \Theta \times \mathbb{R}^{k-1} \to \mathbb{R}^B$: neural network

Neural spline flow construction

How can we approximate the conditional CDFs with neural networks?

$$F_k(x_k; x_1, \dots, x_{k-1}) = \mathbb{P}(X_k \leq x_k | X_1 = x_1, \dots, X_{k-1} = x_{k-1}), k = 2, \dots, d$$

$0 = u_0 < u_1 < \cdots < u_B = 1$: partition of $[0, 1]$
$NN_k : \Theta \times \mathbb{R}^{k-1} \to \mathbb{R}^B$: neural network

$$\begin{aligned} (p_1, \dots, p_B) &= softmax(NN_{k,\theta}(x_1, \dots, x_{k-1})) \\ (\hat{x}_0 = 0, \hat{x}_1, \dots, \hat{x}_{B-1}, \hat{x}_B = 1) &= (\sum_{j=1}^{i} p_j)_{i=0,\dots,B} \end{aligned}$$

Approximated CDF evaluated at the knots $(\hat{x}_j)_{j=1}^{B}$:

$$\hat{F}_\theta(\hat{x}_j; x_1, \dots, x_{k-1}) = u_j, \quad j = 1, \dots, B$$

Neural spline flow construction

How can we approximate the conditional CDFs with neural networks?

$$F_k(x_k; x_1, \dots, x_{k-1}) = \mathbb{P}(X_k \leq x_k | X_1 = x_1, \dots, X_{k-1} = x_{k-1}), k = 2, \dots, d$$

$0 = u_0 < u_1 < \cdots < u_B = 1$: partition of $[0, 1]$
$NN_k : \Theta \times \mathbb{R}^{k-1} \to \mathbb{R}^B$: neural network

$$(p_1, \dots, p_B) = softmax(NN_{k,\theta}(x_1, \dots, x_{k-1}))$$

$$(\hat{x}_0 = 0, \hat{x}_1, \dots, \hat{x}_{B-1}, \hat{x}_B = 1) = (\sum_{j=1}^{i} p_j)_{i=0,\dots,B}$$

Approximated CDF evaluated at the knots $(\hat{x}_j)_{j=1}^B$:

$$\hat{F}_\theta(\hat{x}_j; x_1, \dots, x_{k-1}) = u_j, \quad j = 1, \dots, B$$

Different interpolation schemes:

- linear interpolation: [8]
- rational quadratic interpolation: [8]
- cubic interpolation [7]

Neural spline flow construction

How can we approximate the conditional CDFs with neural networks?

$$F_k(x_k; x_1, \dots, x_{k-1}) = \mathbb{P}(X_k \leq x_k | X_1 = x_1, \dots, X_{k-1} = x_{k-1}), k = 2, \dots, d$$

$0 = u_0 < u_1 < \cdots < u_B = 1$: partition of $[0, 1]$
$NN_k : \Theta \times \mathbb{R}^{k-1} \to \mathbb{R}^B$: neural network

$$\begin{aligned}(p_1, \dots, p_B) &= softmax(NN_{k,\theta}(x_1, \dots, x_{k-1})) \\ (\hat{x}_0 = 0, \hat{x}_1, \dots, \hat{x}_{B-1}, \hat{x}_B = 1) &= (\sum_{j=1}^{i} p_j)_{i=0,\dots,B}\end{aligned}$$

Approximated CDF evaluated at the knots $(\hat{x}_j)_{j=1}^{B}$:

$$\hat{F}_\theta(\hat{x}_j; x_1, \dots, x_{k-1}) = u_j, \quad j = 1, \dots, B$$

Different interpolation schemes:

- linear interpolation: [8]
- rational quadratic interpolation: [8]
- cubic interpolation [7]

Challenges in approximating neural splines in a limited data environment

- Curse of dimensionality [11]
- Extrapolation problems: how does the approximated function generalise to unseen market states / regions where historical market data is sparse?
- Smoothness of the approximated function: in the limit, neural spline flows can approximte the empricial dirac distribution. How should we regularize the function? (See e.g. [16] for a study on noise regularizatoin).
- Uncertainty quantification: Bayesian NNs [13]
- Variance in the function approximation $\rightarrow$ bagging [11] can help

Deriving the loss function for the compressed process

$p_\theta(x_t | x_{t-q+1:t})$: conditional density estimator; neural spline flow.
$Y = (S, \sigma) : \Omega \times \mathbb{N} \to \mathbb{R}_{>0} \times \mathbb{R}^d$: compressed process
$X = (S, \psi(\sigma))$: market process with call price decoder $\psi : \mathbb{R}^d \to \mathbb{R}^{mn}_{>0}$

Deriving the loss function for the compressed process

$p_\theta(x_t|x_{t-q+1:t})$: conditional density estimator; neural spline flow.
$Y = (S, \sigma) : \Omega \times \mathbb{N} \to \mathbb{R}_{>0} \times \mathbb{R}^d$: compressed process
$X = (S, \psi(\sigma))$: market process with call price decoder $\psi : \mathbb{R}^d \to \mathbb{R}^{mn}_{>0}$

If $\psi : \mathbb{R}^d \to \mathbb{R}^{mn}_{>0}$ is injective [1, 9], then minimizing the KL-divergence on X and Y is equivalent:

$$\begin{aligned}
\mathrm{KL}(p, p_\theta) &= \mathbb{E}_p\left(\mathbb{E}_p\left(\ln \frac{p(X_{t+1}|X_{t-q+1:t})}{p_\theta(X_{t+1}|X_{t-q+1:t})}|X_{t-q+1:t}\right)\right) \\
&= \mathbb{E}_p\left(\mathbb{E}_p\left(\ln \frac{p(Y_{t+1}|Y_{t-q+1:t})}{p_\theta(Y_{t+1}|Y_{t-q+1:t})}|Y_{t-q+1:t}\right)\right) \\
&= -\mathbb{E}_p\left(\mathbb{E}_p\left(\ln p_\theta(Y_{t+1}|Y_{t-q+1:t})|Y_{t-q+1:t}\right)\right) + const
\end{aligned}$$

where we used in the second equality

$$p(x_{t+1}|x_{t-q+1:t}) = p(y_{t+1}|y_{t-q+1:t})\left(\det J_\psi(\sigma_{t+1})^T J_\psi(\sigma_{t+1})\right)^{-\frac{1}{2}}$$

Single-asset Eurostoxx 50 numerical results (1)

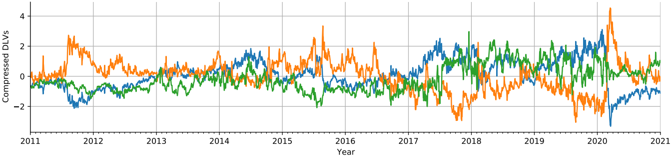

Figure: Encoded DLVs.

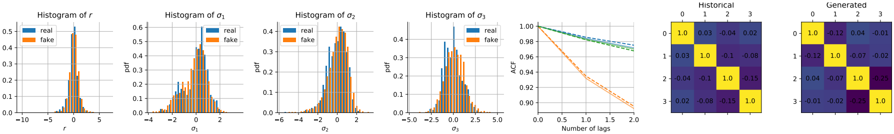

Figure: Eurostoxx 50 long-term performance of the level process $(r_t, \boldsymbol{\sigma}_t)$.

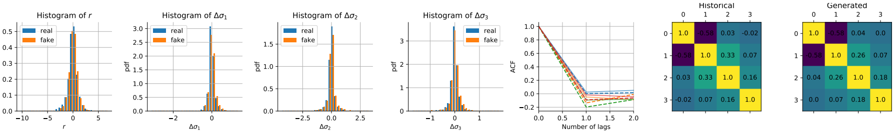

Figure: Eurostoxx 50 long-term performance of the return process $(r_t, \Delta\boldsymbol{\sigma}_t)$.

Single-asset Eurostoxx 50 numerical results (2)

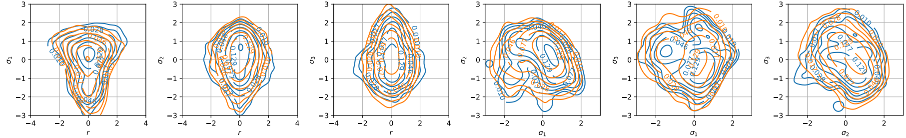

Figure: Eurostoxx 50 kernel density estimate of the joint (long-term) level distribution $(r_t, \boldsymbol{\sigma}_t)$ of the empirical (blue) and generated distribution (orange).

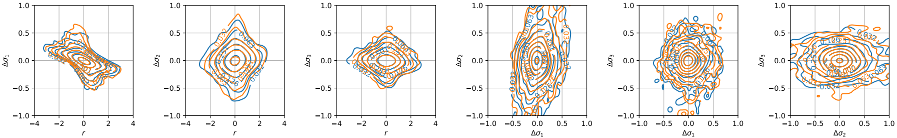

Figure: Eurostoxx 50 kernel density estimate of the joint (long-term) return distribution $(r_t, \Delta\boldsymbol{\sigma}_t)$ of the empirical (blue) and generated distribution (orange).

Single-asset Eurostoxx 50 numerical results (3)

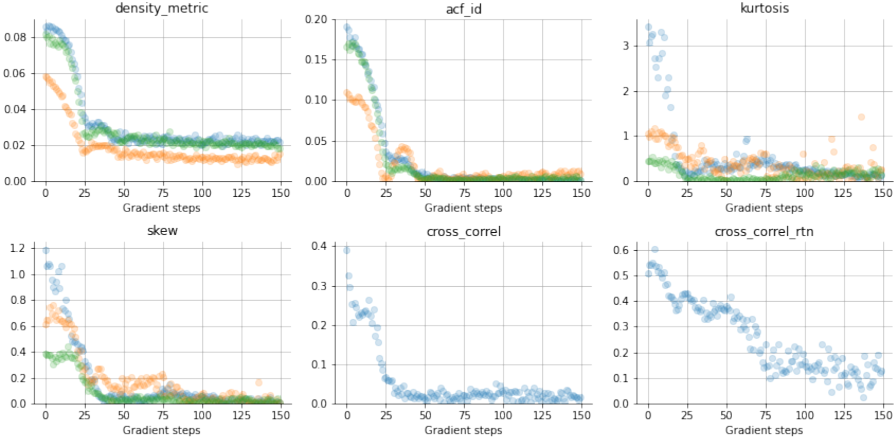

Figure: Development of standard test metrics during the course of optimisation.

Multi-asset market simulation

Multi-asset problem setting

- DH may be used to hedge a portfolio of derivatives or basket options.
- **Scalability**: An N-variate asset universe $X^i, i = 1, \dots, N$ has 2^N basket combinations. $\Rightarrow$ Naively, we would need 2^N market simulators to cover the full spectrum of combinations.
- **Problem**:
  - How can we reuse already calibrated single-asset simulators $p_\theta(X^i_{t+1} | X^i_{t-q+1:t})$ to derive a multi-asset simulator for a basket of assets?
  - How do we sample from the joint process $(X^i_{t+1})^N_{i=1} | (X^i_t)^N_{i=1}$ and ensure that single-asset dynamics $X^i_{t+1} | X^i_t \sim p_\theta$ remain the same?

Multi-asset market simulation

Approach used in the paper

- Compute the uniform noise process

$$U^i_{t+1} = F(X^i_{t+1}; X^i_{t-q+1:t}), i = 1, \dots, N$$

- Assume no dependence on the state, i.e.

$$(U^i_{t+1})_{i=1}^N \perp (X^i_{t-q+1:t})_{i=1}^N$$

- Calibrate a parametric / non-parametric copula for $(U^i_{t+1})_{i=1}^N$; here we compare a Gaussian copula and a flow-based approximation of the joint distribution.

Method 1: Gaussian copula

Under the Gaussian copula assumption we assume that

$$\Phi^{-1}(U_{t+1}^i)_{i=1}^N \sim \mathcal{N}(0, \Sigma)$$
$$X_{t+1}^i = F^{-1}(U_{t+1}^i; X_{t-q+1:t}^i), i = 1, \ldots, N$$

where Φ is the Normal CDF applied component-wise and the cross-correlation matrix Σ follows the block structure:

$$\Sigma = \begin{pmatrix} I_{d+1\times d+1} & \Sigma_{1,2} & \cdots & \Sigma_{1,N} \\ \Sigma_{1,2}^T & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \Sigma_{N-1,N} \\ \Sigma_{1,N}^T & \cdots & \Sigma_{N-1,N}^T & I_{d+1\times d+1} \end{pmatrix} \in \mathbb{R}^{N(d+1)\times N(d+1)}$$

to ensure that the single-asset dynamics are not changed. Σ can be estimated from the "Gaussianized" time series $\mathbf{z}^i = (z_t^i)_{t=1+q}^T$ for $i = 1, \ldots, N$

$$u_{t+1}^i = F(x_{t+1}^i, x_{t-q+1:t}^i) \quad z_{t+1}^i = \Phi^{-1}(u_{t+1}^i), t = 1 + q, \ldots, T$$

Method 2: Flow-based approximation

- Gaussian copula is limited to it's parametric form.
- Neural spline flow can approximate any joint density, but cannot ensure that the marginals are uniform (see [12] for ongoing work). It therefore can change the single-asset dynamics.
- In the flow case we approximate the joint density of the uniform process

$$U_{t+1}^i = F(X_{t+1}^i; X_t^i), i = 1, \ldots, N$$

Multi-asset Eurostoxx 50 and S&P 500 numerical results

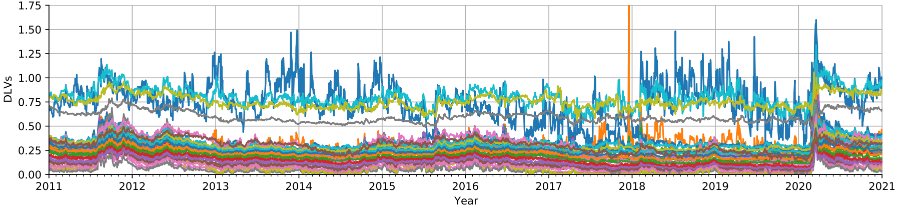

Figure: Eurostoxx 50 DLVs; ; $\mathcal{K} = (0.8, 0.85, \ldots, 1.2)$, $\mathcal{T} = (20, 40, 60, 120)$

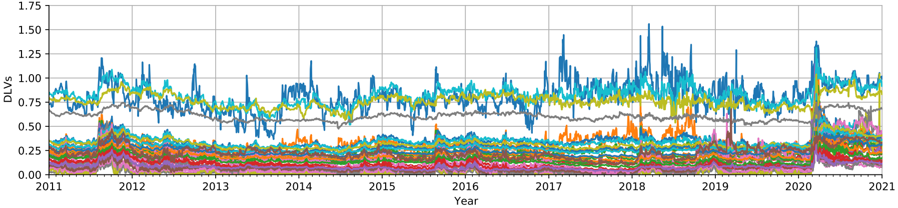

Figure: S&P 500 DLVs; $\mathcal{K} = (0.8, 0.85, \ldots, 1.2)$, $\mathcal{T} = (20, 40, 60, 120)$

Multi-asset Eurostoxx 50 and S&P 500 numerical results

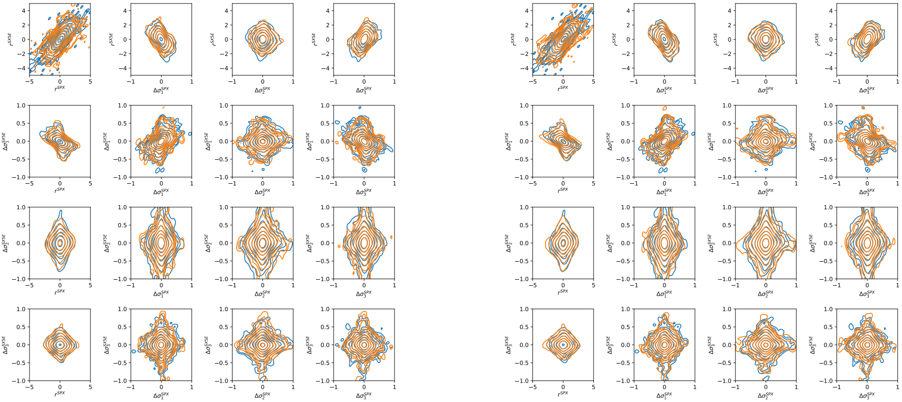

Figure: Eurostoxx 50 / S&P 500 kernel density estimate of the historical (blue) and generated (orange) joint return distribution $(r_t^{\mathrm{SX5E}}, \boldsymbol{\Delta\sigma}_t^{\mathrm{SX5E}}, r_t^{\mathrm{SPX}}, \boldsymbol{\Delta\sigma}_t^{\mathrm{SPX}})$ where the generated joint latent distribution was governed by a Gaussian copula (left) and a normalizing flow (right). We use the ticker symbols SX5E and SPX to abbreviate the Eurostoxx 50 and S&P 500 components in the plot.

Multi-asset Eurostoxx 50 and S&P 500 numerical results

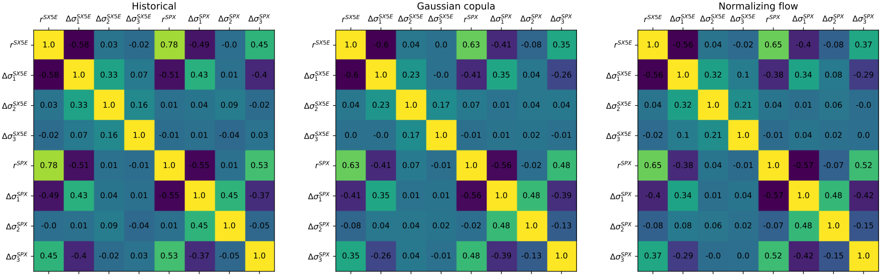

Figure: Eurostoxx 50 / S&P 500 cross-correlation matrix of the joint return process of the historical data (left) and generated data under the Gaussian copula (middle) and normalizing flow (right).

TL;DL

- Market simulator presented is a closed framework to simulate spot and call price dynamics while maintaining no static arbitrage. Statistical arbitrage removal through a change in measure is discussed in [2, 17].
- High-dimensional DLV grids were compressed to a low-dimensional representation. Nonlinear autoencoders for dimensionality reduction showed substantial performance improvements when compared to PCA.
- Dynamics of the compressed representation were calibrated through neural spline flows. Objective: minimize the NLL. Standard test metrics were used to validate the fitting performance.
- To scale single-asset market simulation to multi-asset markets we proposed using Gaussian copula and normalizing flows.

Future work

Single-asset:

- Invariant distributions: $p_\theta(x_t) \stackrel{!}{=} \hat{p}(x_t), t \in \mathbb{N}$
- Controlled extrapolations in low-density regions, e.g. COVID-19 crash period.
- Path explosions: $\lim_{t\to\infty} \mathbb{P}(X_t > k) > 0, \forall k > 0$.

Multi-asset:

- Stochastic and state-dependent cross-correls $(\Sigma_t)_t$ or copulas
- Joint representations / transfer learning

References I

[1] J. Brehmer and K. Cranmer.
Flows for simultaneous manifold learning and density estimation, 2020.

[2] H. Buehler, P. Murray, M. S. Pakkanen, and B. Wood.
Deep hedging: Learning to remove the drift under trading frictions with minimal equivalent near-martingale measures, 2022.

[3] H. Buehler and E. Ryskin.
Discrete local volatility for large time steps (short version).
2016.

[4] I. Chevyrev and H. Oberhauser.
Signature moments to characterize laws of stochastic processes.
arXiv preprint arXiv:1810.10971, 2018.

[5] S. N. Cohen, C. Reisinger, and S. Wang.
Arbitrage-free neural-sde market models, 2021.

References II

[6] S. N. Cohen, C. Reisinger, and S. Wang.
Estimating risks of option books using neural-sde market models.
arXiv preprint arXiv:2202.07148, 2022.

[7] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios.
Cubic-spline flows.
arXiv preprint arXiv:1906.02145, 2019.

[8] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios.
Neural spline flows.
Advances in Neural Information Processing Systems, 32:7511–7522, 2019.

[9] M. C. Gemici, D. Rezende, and S. Mohamed.
Normalizing flows on riemannian manifolds.
arXiv preprint arXiv:1611.02304, 2016.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio.
Generative adversarial nets.
Advances in neural information processing systems, 27, 2014.

References III

[11] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman.
The elements of statistical learning: data mining, inference, and prediction, volume 2.
Springer, 2009.

[12] C. K. Ling, F. Fang, and J. Z. Kolter.
Deep archimedean copulas.
CoRR, abs/2012.03137, 2020.

[13] R. M. Neal.
Bayesian learning for neural networks, volume 118.
Springer Science & Business Media, 2012.

[14] H. Ni, L. Szpruch, M. Wiese, S. Liao, and B. Xiao.
Conditional sig-wasserstein gans for time series generation, 2020.

[15] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan.
Normalizing flows for probabilistic modeling and inference.
arXiv preprint arXiv:1912.02762, 2019.

References IV

[16] J. Rothfuss, F. Ferreira, S. Boehm, S. Walther, M. Ulrich, T. Asfour, and A. Krause.
Noise regularization for conditional density estimation.
arXiv preprint arXiv:1907.08982, 2019.

[17] M. Wiese and P. Murray.
Risk-neutral market simulation.
arXiv preprint arXiv:2202.13996, 2022.

Disclosure

Opinions and estimates constitute our judgement as of the date of this Material, are for informational purposes only and are subject to change without notice. It is not a research report and is not intended as such. Past performance is not indicative of future results. This Material is not the product of J.P. Morgan's Research Department and therefore, has not been prepared in accordance with legal requirements to promote the independence of research, including but not limited to, the prohibition on the dealing ahead of the dissemination of investment research. This Material is not intended as research, a recommendation, advice, offer or solicitation for the purchase or sale of any financial product or service, or to be used in any way for evaluating the merits of participating in any transaction. Please consult your own advisors regarding legal, tax, accounting or any other aspects including suitability implications for your particular circumstances. J.P. Morgan disclaims any responsibility or liability whatsoever for the quality, accuracy or completeness of the information herein, and for any reliance on, or use of this material in any way. Important disclosures at: `www.jpmorgan.com/disclosures`.