CONSTRUCTING EFFICIENT SIMULATED MOMENTS USING TEMPORAL CONVOLUTIONAL NETWORKS

JONATHAN CHASSOT AND MICHAEL CREEL

DEEP LEARNING FOR SOLVING AND ESTIMATING DYNAMIC MODELS (DSE2023CH)

August 24, 2023

OUR CONTRIBUTION

- Propose a method to estimate model parameters leveraging **deep learning**
- Allow for inference using **well-established econometric methods**
- Provide an **exactly identifying** and **informative** set of statistics for simulation-based inference
- Achieve performance equal to or better than maximum likelihood for small and moderate sample sizes for several DGPs

MOTIVATING EXAMPLE

- Parameter estimation for a complex data-generating process (DGP)
- Failure of traditional estimation methods (ML, GMM)

MOTIVATING EXAMPLE

- **Parameter estimation for a complex data-generating process (DGP)**
  - Presence of latent variables
  - High-dimensional integrals
  - ...
- **Failure of traditional estimation methods (ML, GMM)**
  - Intractable likelihood
  - Computational constraints
  - No closed-form theoretical moments

→ Use simulation-based inference methods

METHODOLOGY

SIMULATION-BASED INFERENCE

- Method of Simulated Moments (McFadden, 1989)
- Approximate Bayesian Computation (Rubin, 1984)
- Indirect Inference (Gouriéroux, Monfort & Renault, 1993; Smith, 1993)
- Bayesian Limited Information Estimation (Kwan, 1999; Kim, 2002; Chernozhukov & Hong, 2003)

Match sample statistics with statistics obtained through simulation

METHOD OF SIMULATED MOMENTS

- Use **simulated moment conditions** instead of theoretical ones
- Match data moments and simulated moments

$$\hat{\theta}_{\mathrm{MSM}} = \underset{\theta \in \Theta}{\arg\min} \left(T^{-1} \sum_{t=1}^{T} m(x_t, \theta)^{\top} \right) W_T \left(T^{-1} \sum_{t=1}^{T} m(x_t, \theta) \right),$$

where W_T is a weighting matrix and

$$m(x_t, \theta) = f(x_t) - \frac{1}{S} \sum_{s=1}^{S} f(\tilde{x}_t(\theta))$$

with S the number of simulations and $\tilde{x}_t(\theta)$ the data simulated at parameter θ.

METHOD OF SIMULATED MOMENTS

- …but how do we choose $f(\cdot)$ in

$$m(x_t, \theta) = f(x_t) - \frac{1}{S}\sum_{s=1}^{S} f(\tilde{x}_t(\theta))$$

OPTIMAL MOMENT CONDITIONS

- Choosing optimal moment conditions is difficult
- Overidentification leads to high asymptotic efficiency but also to **high bias and/or variance** in finite samples (Donald, Imbens & Newey, 2009)

OPTIMAL MOMENT CONDITIONS

- Choosing optimal moment conditions is difficult
- Overidentification leads to high asymptotic efficiency but also to **high bias and/or variance** in finite samples (Donald, Imbens & Newey, 2009)

Theory:

- Gallant & Tauchen (1996) show that the optimal choice of moment conditions corresponds to the **score function** (equivalent to MLE)

OPTIMAL MOMENT CONDITIONS

- Choosing optimal moment conditions is difficult
- Overidentification leads to high asymptotic efficiency but also to **high bias and/or variance** in finite samples (Donald, Imbens & Newey, 2009)

Theory:

- Gallant & Tauchen (1996) show that the optimal choice of moment conditions corresponds to the **score function** (equivalent to MLE)

Practice:

- Moment selection criteria and algorithms (e.g., Donald, Imbens & Newey, 2009; Cheng & Liao, 2015; DiTraglia, 2016)
- Our work is related to this category

OPTIMAL MOMENT CONDITIONS

Goal:

- Given a data set $\{x_t \mid x_t \in \mathbb{R}^k\}_{t=1}^{T}$, generated by our DGP under true parameter value θ_0, we would like to find $f(x_t) \approx \theta_0$

OPTIMAL MOMENT CONDITIONS

Goal:

- Given a data set $\{x_t \mid x_t \in \mathbb{R}^k\}_{t=1}^{T}$, generated by our DGP under true parameter value θ_0, we would like to find $f(x_t) \approx \theta_0$

Idea:

- Generate samples $\{\tilde{x}_t(\theta) \mid \tilde{x}_t(\theta) \in \mathbb{R}^k\}$ with $\theta \in \Theta$ and use deep learning to infer $f(\cdot)$, a **mapping from data to parameters**

NEURAL NETWORKS

- Long Short-Term Memory Networks (LSTM)
- Temporal Convolutional Networks (TCN)

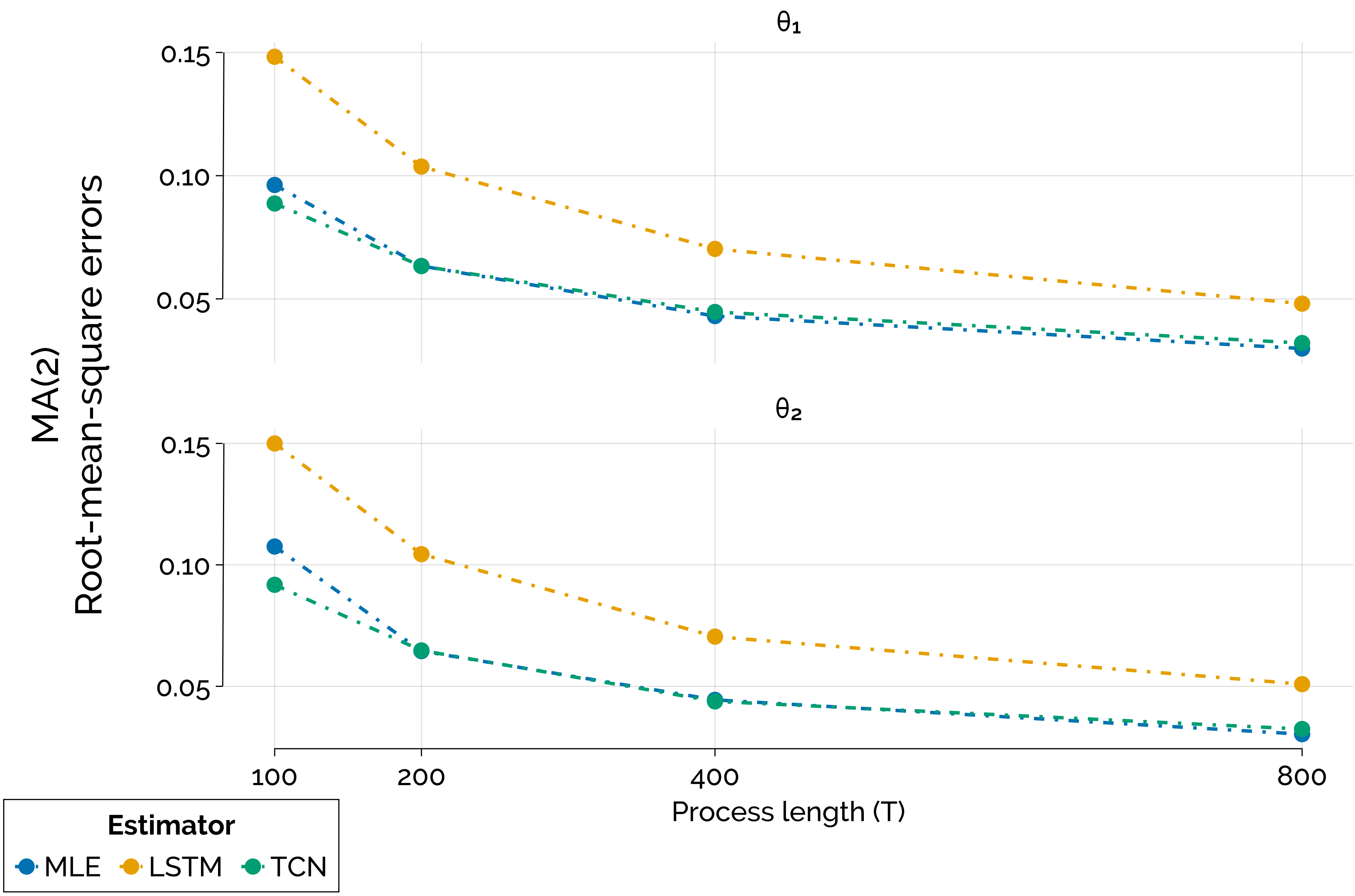
NEURAL NETWORKS

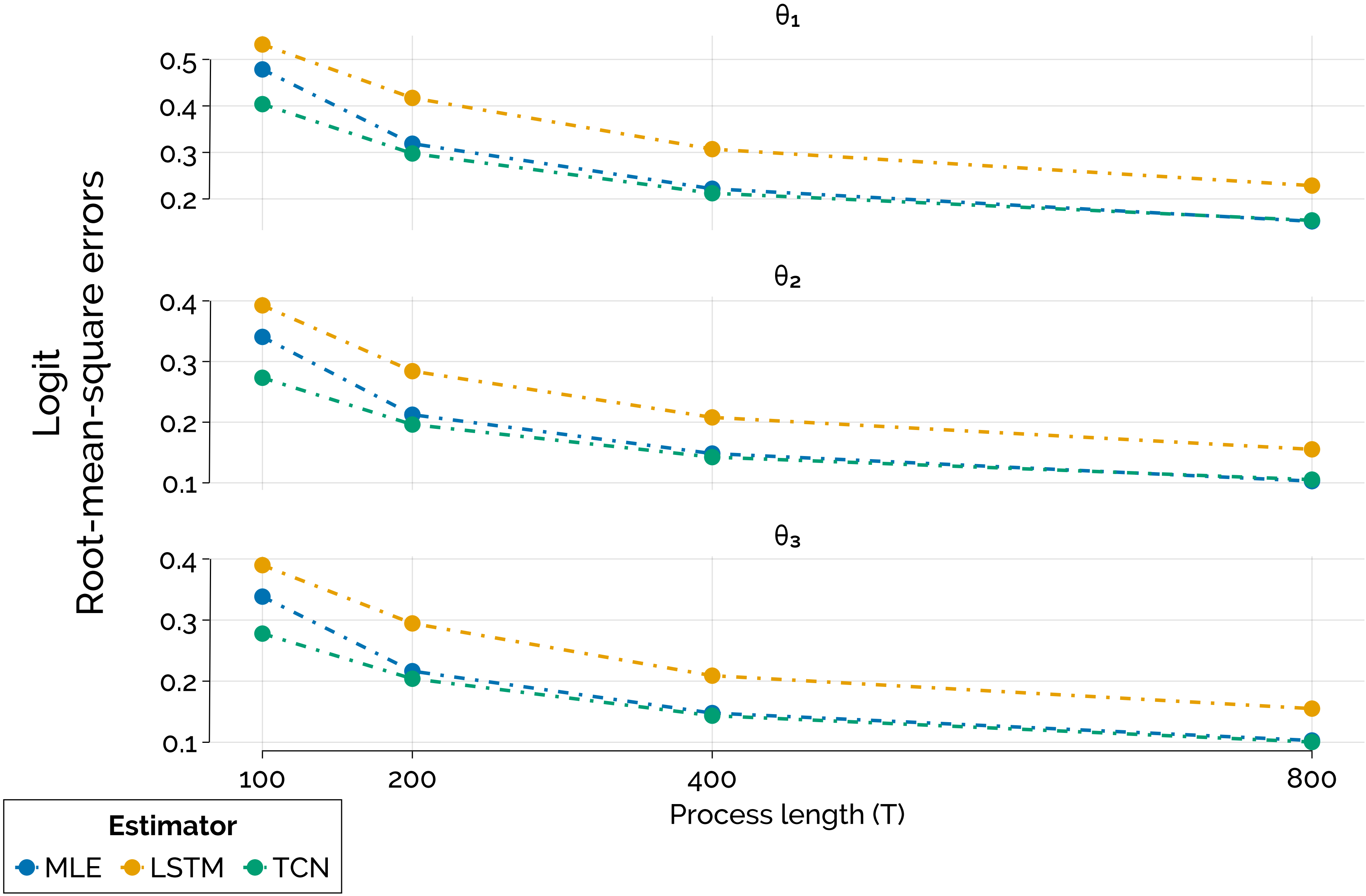
- **Long Short-Term Memory Networks (LSTM)**
  - Hochreiter & Schmidhuber (1997)
  - Dominated sequence modelling pre-Transformers (Vaswani et al., 2017)
  - Difficulties in modeling long-term dependencies
  - Serve as a **baseline** deep learning model in this work
- **Temporal Convolutional Networks (TCN)**
  - Introduced as *WaveNet* (van den Oord et al., 2016)
  - Fully parallelizable
  - Flexible receptive field size
  - Serve as the **main model** in this work

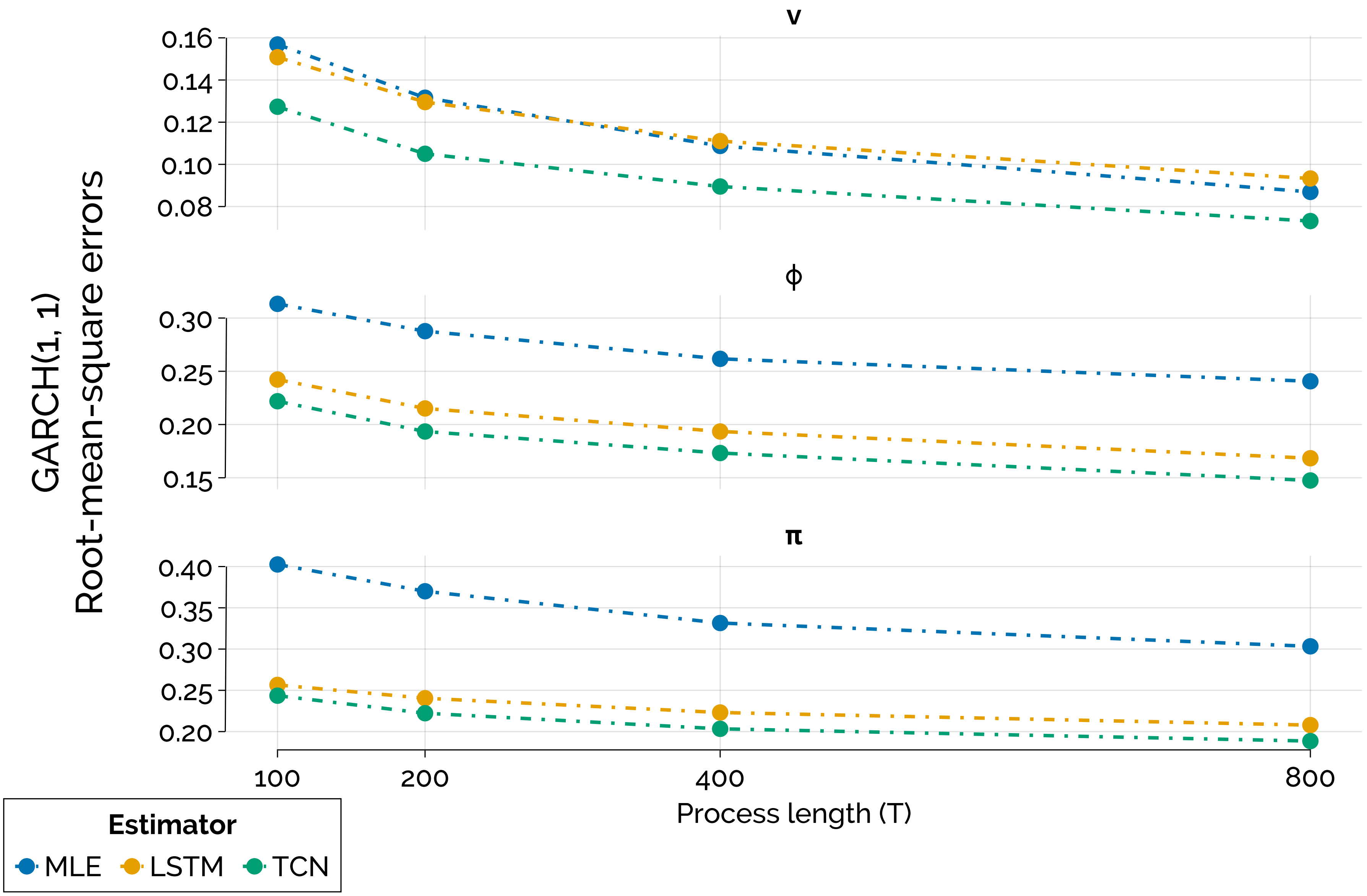
BENCHMARK RESULTS

RESULTS FOR SIMPLE DGPS

- Benchmark our TCNs and LSTMs against MLE for 3 data-generating processes (MA(2), Logit, GARCH(1,1)) where the **likelihood is tractable**
- Sample sizes 100, 200, 400, and 800
- Comparison across 5'000 test samples for each setting
- **Conjecture:** if the neural networks do well, they will also perform well when the MLE is not available


θ₁
θ₂
MA(2)
Root-mean-square errors
0.15
0.10
0.05
100
200
400
800
Process length (T)
Estimator
MLE
LSTM
TCN


θ1
θ2
θ3
Logit
Root-mean-square errors
0.5
0.4
0.3
0.2
0.1
100
200
400
800
Process length (T)
Estimator
MLE
LSTM
TCN


v
0.16
0.14
0.12
0.10
0.08
ϕ
0.30
0.25
0.20
0.15
π
0.40
0.35
0.30
0.25
0.20
GARCH(1, 1)
Root-mean-square errors
100
200
400
800
Process length (T)
Estimator
MLE
LSTM
TCN

JUMP-DIFFUSION PROCESS

JUMP-DIFFUSION STOCHASTIC VOLATILITY

$$
\begin{aligned}
dp_t &= \mu\, dt + \sqrt{\exp h_t}\, dW_{1t} + J_t\, dN_t \\
dh_t &= \kappa(\alpha - h_t) + \sigma\, dW_{2t}
\end{aligned}
$$

JUMP-DIFFUSION STOCHASTIC VOLATILITY

$$
\begin{aligned}
dp_t &= \mu\, dt + \sqrt{\exp h_t}\, dW_{1t} + J_t\, dN_t \\
dh_t &= \kappa(\alpha - h_t) + \sigma\, dW_{2t}
\end{aligned}
$$

- p_t: logarithmic price
  - μ: average drift in price
  - J_t: jump size ($J_t = a\lambda_1\sqrt{\exp h_t}$) with $\mathbb{P}[a=1] = \mathbb{P}[a=-1] = \frac{1}{2}$
  - N_t: Poisson process with jump intensity λ_0
- h_t: logarithmic volatility
  - κ: speed of mean-reversion
  - α: long-term mean volatility
  - σ: volatility of the volatility
- W_{1t}, W_{2t}: correlated Brownian motions with correlation ρ

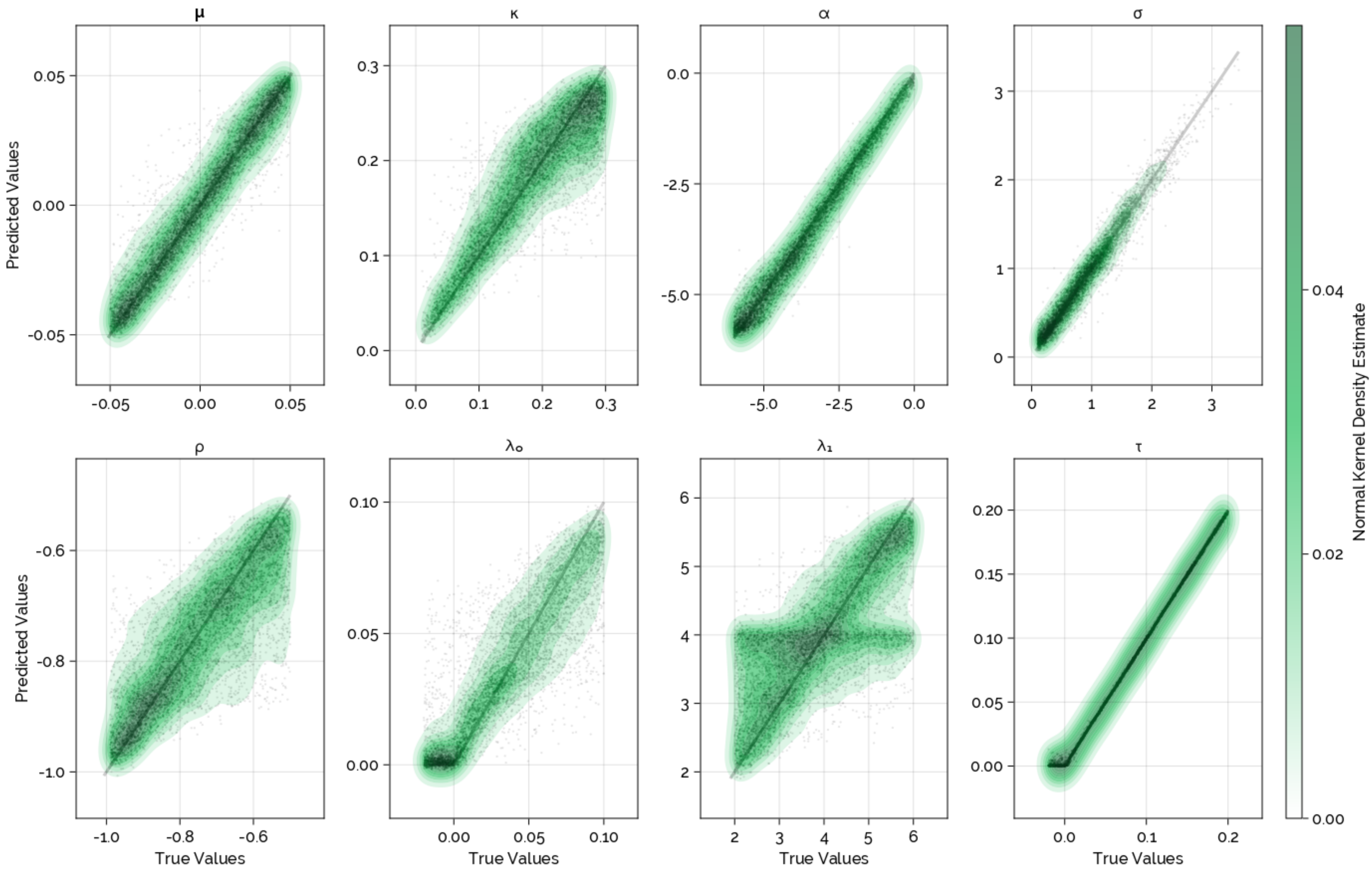
JUMP-DIFFUSION STOCHASTIC VOLATILITY

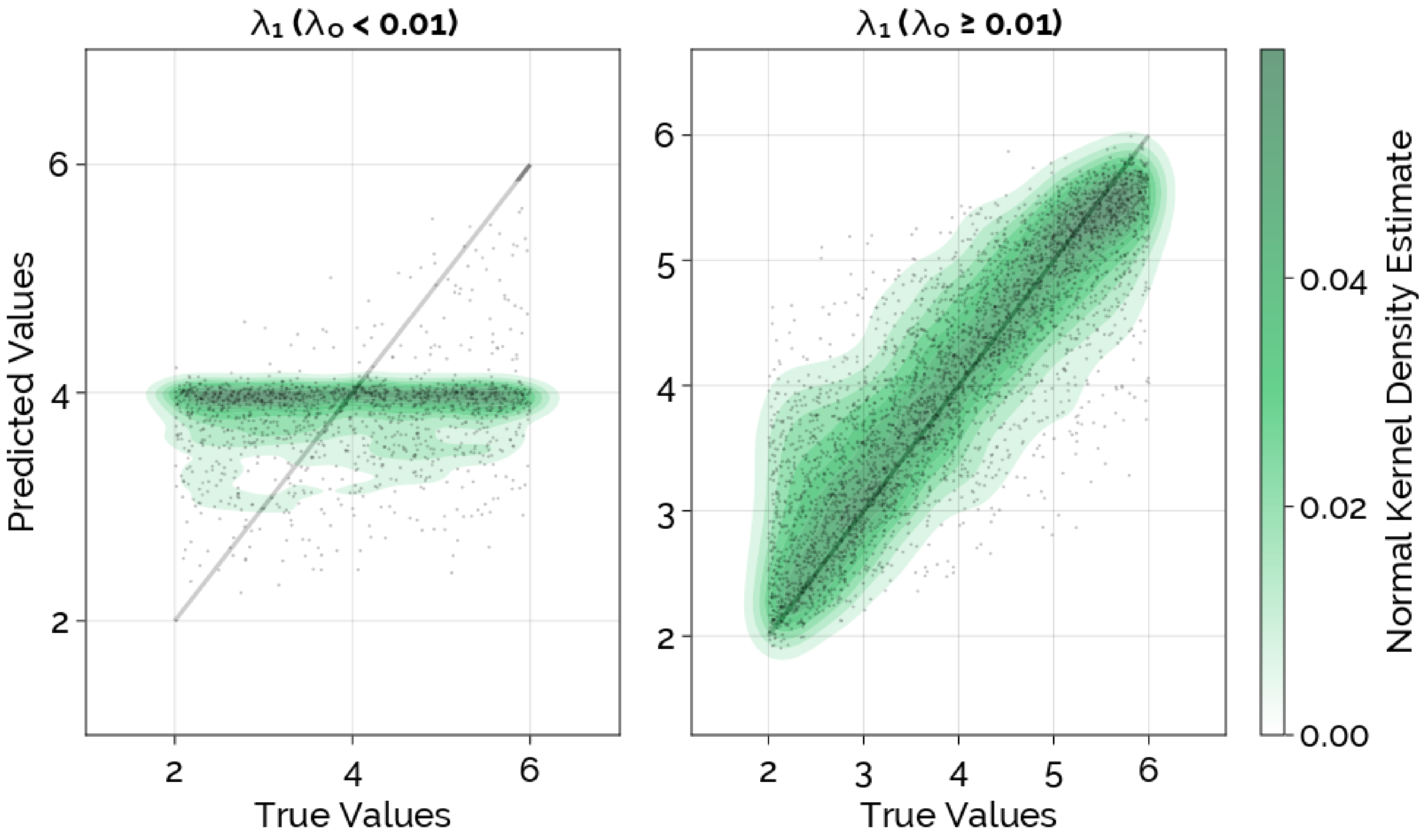
Parameters:

1. μ: average drift in price
2. κ: speed of mean-reversion
3. α: long-term mean volatility
4. σ: volatility of the volatility
5. ρ: correlation between W_{1t} and W_{2t}
6. λ_0: jump intensity
7. λ_1: jump magnitude
8. τ: the volatility of a measurement error $N(0, \tau^2)$ added to the observed price

Observables:

1. logarithmic returns
2. realized volatility
3. bipower variation


μ
κ
α
σ
ρ
λ₀
λ₁
τ
Predicted Values
True Values
Normal Kernel Density Estimate
0.00
0.02
0.04


λ_1 (λ_0 < 0.01)
λ_1 ($\lambda_0 \geq$ 0.01)
Predicted Values
True Values
True Values
Normal Kernel Density Estimate
0.00
0.02
0.04

CONCLUSION

CONCLUSION

- Best case scenario for deep learning
- Once the network is trained, inference is as fast as matrix multiplication
- Limited *only* in the cost of simulation
- Promising results on three simple DGPs and one moderately complex DGP
- Easy to implement
  - Full Julia package under development (already functional): https://github.com/JLDC/DeepSimulatedMoments.jl