

# Learning Rough Volatility

**Blanka Horvath**

CIRM  
Luminy, 6th September 2018

**Innovative Research in Mathematical Finance**  
dedicated to the 70th birthday of  
**Yuri Kabanov**

- ▶ Rough volatility models have been around since October 2014  
(see the Rough Volatility website for a chronicle of developments)
- ▶ These models have repeatedly proven to be superior to standard models in many areas: in volatility forecasting, in option pricing, close fits to the implied vol surface, ...

- ▶ Rough volatility models have been around since October 2014 (see the Rough Volatility website for a chronicle of developments)
- ▶ These models have repeatedly proven to be superior to standard models in many areas: in volatility forecasting, in option pricing, close fits to the implied vol surface, ...
- ▶ Relaxing the assumption of independence of volatility increments was crucial for the superior performance of rough volatility models  $\Rightarrow$  but: several standard pricing methods no longer available & naive Monte Carlo methods slow
- ▶ Calibration time has been a bottleneck for rough volatility several advances have been made to speed up the calibration process [BLP '15, MP '17, HJM '17].

Today's talk:

Speedups for rough volatility models along two lines:

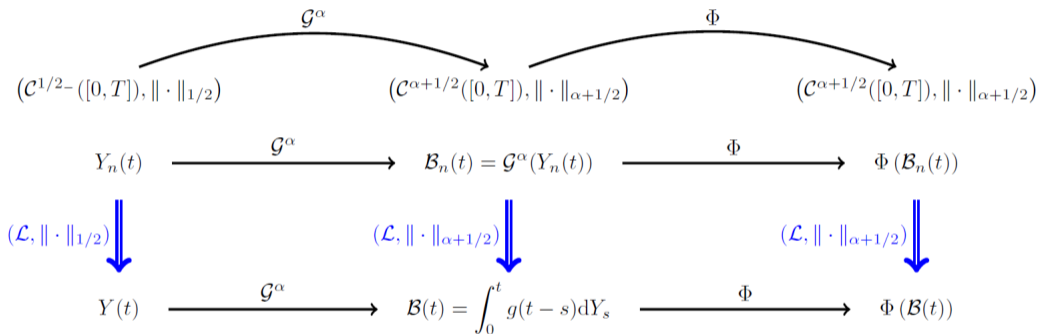
1. in **pricing** of vanilla options based on faster Monte Carlo approximations for a family of rough stochastic volatility models. [H-Jacquier-Muguruza '17])
2. in **calibration** by means of machine learning (ongoing with A. Sani, A. Muguruza and with M. Tomas).

## Framework

$$\begin{aligned}dX_t &= -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t, & X_0 &= 0, \\V_t &= \Phi\left(\int_0^t g(t-s)dY_s\right), & V_0 &> 0, \alpha \in (-1/2, 1/2), \\dY_t &= b(Y_t)dt + \sigma(Y_t)dZ_t, & dZ_t dW_t &= \rho dt.\end{aligned}$$

where  $\Phi \in \mathcal{C}^1$ ,  $g \in \mathcal{L}^\alpha := \{u^\alpha L(u) : L \in \mathcal{C}_b^1([0, T]), \alpha \in (-\frac{1}{2}, \frac{1}{2})\}$   
and  $Y$  satisfies Yamada-Watanabe conditions for path-wise uniqueness.

# FCLT for Hölder cont. processes:



Generalised Fractional Operators  $\mathcal{G}^\alpha$ 

## Definition

Let  $g \in \mathcal{L}^\alpha := \{u^\alpha L(u) : L \in \mathcal{C}_b^1([0, T]), \alpha \in (-\frac{1}{2}, \frac{1}{2})\}$  and fix  $\lambda \in (0, 1)$ . The GFO for  $f \in \mathcal{C}^\lambda([0, T])$  is

$$(\mathcal{G}^\alpha f)(t) := \begin{cases} \int_0^t f(s) \frac{d}{dt} g(t-s) ds, & \text{for } \alpha \in [0, 1), \\ \frac{d}{dt} \int_0^t f(s) g(t-s) ds, & \text{for } \alpha \in (-\lambda, 0). \end{cases}$$

**Remark:** If  $g(u) = u^\alpha$ , then GFO=Riemann-Liouville fractional operators

## FCLT for Hölder continuous processes

## Theorem (rough Donsker theorem)

Consider the sequence  $(W_n(t))_{n \geq 1}$  and  $W$  its weak limit in  $(\mathcal{C}^{1/2}([0, T]), \|\cdot\|_{1/2})$ .  
Then  $(\mathcal{G}^\alpha W_n)_{n \geq 1}$  converges weakly to  $\int_0^\cdot g(\cdot - s) dW_s$  in  $(\mathcal{C}^{\alpha+1/2}([0, T]), \|\cdot\|_{\alpha+1/2})$   
for  $\alpha \in (-\frac{1}{2}, \frac{1}{2})$ .



# FCLT for rough volatility models

## FCLT for rough volatility models

Define recursively in time, for any  $n \geq 1$ ,  $t \in [0, T]$ ,  $t_k = \frac{k}{N}$

$$X_n(t) := -\frac{1}{2} \frac{T}{n} \sum_{k=1}^{\lfloor nt \rfloor} \Phi((\mathcal{G}^\alpha Y_n)(t_k)) + \sqrt{\frac{T}{\sigma n}} \sum_{k=1}^{\lfloor nt \rfloor} \sqrt{\Phi((\mathcal{G}^\alpha Y_n)(t_k))} (W_n(t_{k+1}) - W_n(t_k))$$

## FCLT for rough volatility models

Define recursively in time, for any  $n \geq 1$ ,  $t \in [0, T]$ ,  $t_k = \frac{k}{N}$

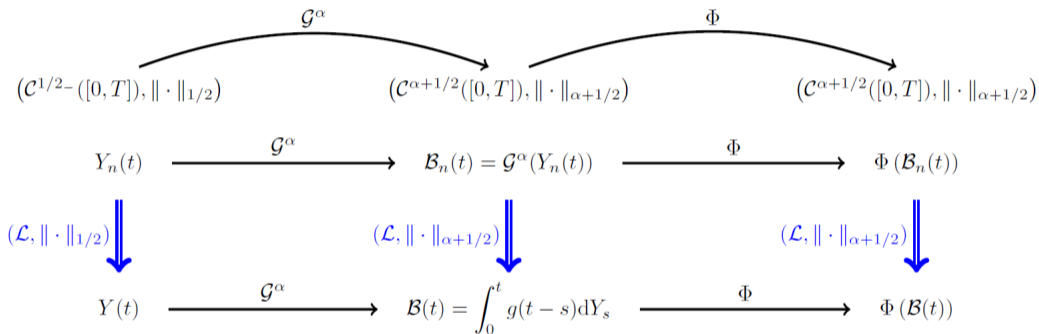
$$X_n(t) := -\frac{1}{2} \frac{T}{n} \sum_{k=1}^{\lfloor nt \rfloor} \Phi((\mathcal{G}^\alpha Y_n)(t_k)) + \sqrt{\frac{T}{\sigma n}} \sum_{k=1}^{\lfloor nt \rfloor} \sqrt{\Phi((\mathcal{G}^\alpha Y_n)(t_k))} (W_n(t_{k+1}) - W_n(t_k))$$

### Theorem (rDonsker for rough volatility models)

$(X_n)_{n \geq 1}$ , converges weakly to  $X$  in  $(\mathcal{C}^{1/2-}(\mathbb{T}), \|\cdot\|_{1/2-})$ ,

$$\begin{aligned} dX_t &= -\frac{1}{2} V_t dt + \sqrt{V_t} dW_t, & X_0 &= 0, \\ V_t &= \Phi \left( \int_0^t g(t-s) dY_s \right), & V_0 &> 0, \alpha \in (-1/2, 1/2), \\ dY_t &= b(Y_t) dt + \sigma(Y_t) dZ_t, & dZ_t dW_t &= \rho dt. \end{aligned}$$

# FCLT for Hölder cont. processes:




## Examples in this framework

$$\begin{aligned}dX_t &= -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t, & X_0 &= 0, \\V_t &= \Phi\left(\int_0^t g(t-s)dY_s\right), & V_0 &> 0, \alpha \in (-1/2, 1/2), \\dY_t &= b(Y_t)dt + \sigma(Y_t)dZ_t, & dZ_t dW_t &= \rho dt.\end{aligned}$$

↓  $g(u) = u^\alpha, \Phi(x) = \mathcal{E}(x), Y_t = Z_t$

## Examples in this framework

$$\begin{aligned}
 dX_t &= -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t, & X_0 &= 0, \\
 V_t &= \Phi\left(\int_0^t g(t-s)dY_s\right), & V_0 &> 0, \alpha \in (-1/2, 1/2), \\
 dY_t &= b(Y_t)dt + \sigma(Y_t)dZ_t, & dZ_t dW_t &= \rho dt.
 \end{aligned}$$



 $g(u) = u^\alpha, \Phi(x) = \mathcal{E}(x), Y_t = Z_t$

## Rough Bergomi:

$$\begin{aligned}
 dX_t &= -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t, & X_0 &= 0 \\
 V_t &= \xi_0(t)\mathcal{E}\left(2\nu C_H \int_0^t \frac{dZ_u}{(t-u)^{1/2-H}}\right), & \nu, \xi_0(\cdot) &> 0 \\
 dZ_t dW_t &= \rho dt, & \rho &\in (0, 1)
 \end{aligned}$$


## Examples in this framework

$$\begin{aligned}dX_t &= -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t, & X_0 &= 0, \\V_t &= \Phi\left(\int_0^t g(t-s)dY_s\right), & V_0 &> 0, \alpha \in (-1/2, 1/2), \\dY_t &= b(Y_t)dt + \sigma(Y_t)dZ_t, & dZ_t dW_t &= \rho dt.\end{aligned}$$

  $g(u) = u^\alpha, \Phi(x) = \eta + Id., dY_t = \kappa(\theta - Y_t)dt + \xi\sqrt{Y_t}dZ_t$

## Examples in this framework

$$\begin{aligned}
 dX_t &= -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t, & X_0 &= 0, \\
 V_t &= \Phi\left(\int_0^t g(t-s)dY_s\right), & V_0 &> 0, \alpha \in (-1/2, 1/2), \\
 dY_t &= b(Y_t)dt + \sigma(Y_t)dZ_t, & dZ_t dW_t &= \rho dt.
 \end{aligned}$$


 $g(u) = u^\alpha, \Phi(x) = \eta + Id., dY_t = \kappa(\theta - Y_t)dt + \xi\sqrt{Y_t}dZ_t$

### Rough Heston:

$$\begin{aligned}
 dX_t &= -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t, & X_0 &= 0, \\
 Y_t &= \int_0^t \kappa(\theta - Y_s)dt + \int_0^t \xi\sqrt{Y_s}dZ_s & V_0, \kappa, \xi, \theta &> 0, 2\kappa\theta > \xi^2 \\
 V_t &= \eta + \int_0^t (t-s)^\alpha dY_s, & \eta &> 0, \alpha \in (-1/2, 1/2).
 \end{aligned}$$



# Example: rough Bergomi smiles

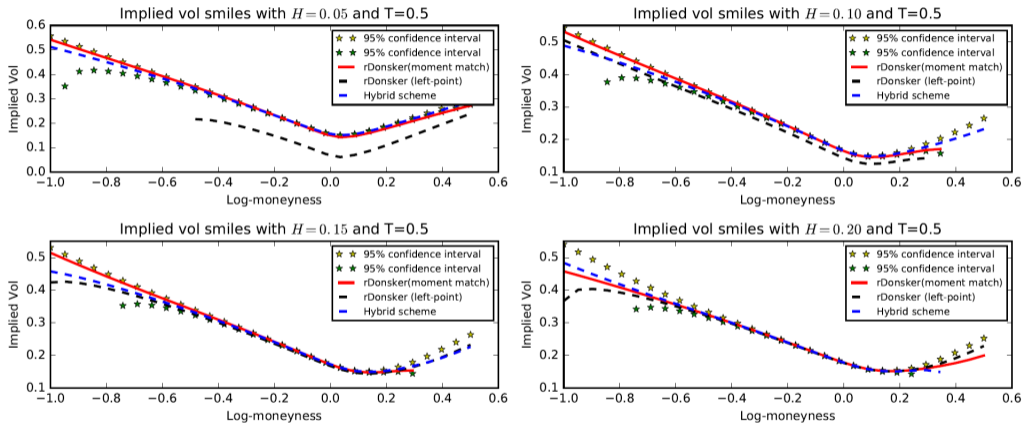


Figure 1: Implied volatilities of rDonsker with left-point and variance matching evaluation and the Hybrid scheme with  $5 \cdot 10^5$  simulations. Parameters:  $\nu = 1$ ,  $\rho = -0.7$ ,  $\xi_0 = 0.04$ ,  $n = 468$



# Example: rough Bergomi smiles

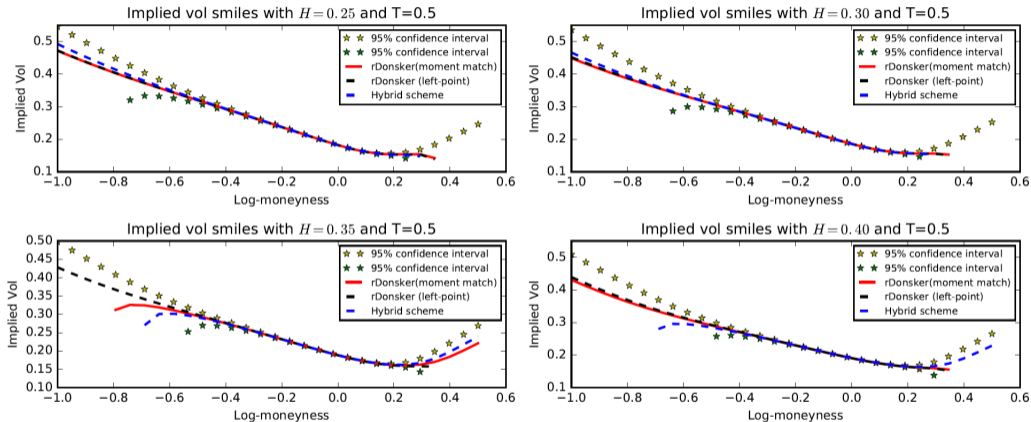
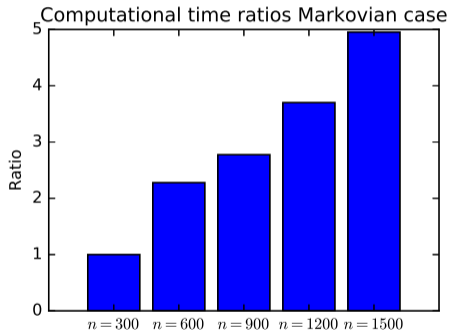
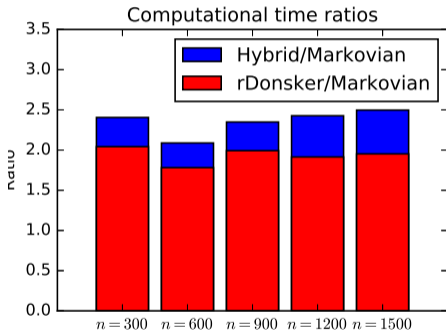


Figure 2: Parameters:  $\nu = 1, \rho = -0.7, \xi_0 = 0.04, n = 468$  steps

# Conclusion

- ▶ rDonsker is  $1.25\times$  faster than Hybrid scheme (because we omit the Cholesky bit)



Speedups for rough volatility models along two lines:

- Part 1: in **pricing** of vanilla options based on faster Monte Carlo approximations for a family of rough stochastic volatility models. [H-Jacquier-Muguruza '17])
- Part 2: in **calibration** by means of machine learning techniques (ongoing with A. Sani, A. Muguruza and with M. Tomas).

# Part 2: Speed-ups on calibration

## Part 2: Speed-ups on calibration

- ▶ one step away from of-the-shelf optimizers to explore the parameter space more efficiently, limiting the number of function evaluations for calibration. Tested for variance options in a "weighted rough Bergomi" framework (De Marco, Guyon)

$$dS_t = -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t \quad V_t = \xi_0(t)(\gamma\nu_t + (1-\gamma)\eta_t)$$

$$\nu_t = \mathcal{E} \left( \nu\sqrt{2H} \int_0^t (t-s)^{H-1/2} dZ_s \right) \quad \eta_t = \mathcal{E} \left( \eta\sqrt{2H} \int_0^t (t-s)^{H-1/2} dZ_s \right)$$

(w Amir Sani and Aitor Muguruza) and

- ▶ approximation by neural networks (w Mehdi Thomas and Aitor Muguruza)

# Optimizers to minimize nr. fn eval

- ▶ Limited-memory Bounded BroydenFletcherGoldfarbShanno (L-BFGS-B)
- ▶ Truncated Newton (TNC)
- ▶ Sequential Least-Squares Quadratic Programming (SLSQP)
- ▶ 2-stage Minimization
  - Stage 1: Classifier-Directed Global Minimizer (2-min Time Budget)
  - Stage 2: Local Minimizer initialized with Stage 1  $x_0$

# Approximation by neural networks I

(Ongoing work) General setup: two parts of the network:

1. Generator: Input (parameters) Output (implied volatilities)
2. Calibrator: Input (implied volatilities) Output (\*optimal\* parameters).

Both feed-forward neural networks for the generator three hidden layers (1000-800-600)-nodes. Calibrator 1 layer on top.



# Approximation by neural networks I

General setup: two parts of the network:

- 1 Generator (approximation of IV surfaces via NN)

In order to train the network we first need to build a training set (supervised learning).

# Approximation by neural networks I

General setup: two parts of the network:

- 1 Generator (approximation of IV surfaces via NN)

In order to train the network we first need to build a training set (supervised learning).

- ▶ For this we can use numerical valuation functions (Bergomi model, Rough Bergomi, Heston, ... Part 1): We generate 20,000 surfaces for each model, using a fixed grid of strikes and tenors.
- ▶ Though training time consuming, it can be done offline.
- ▶ We sample uniformly points in the parameter set  $\theta \in \Theta$ , then compute and save  $f(\theta)$ . Those samples will constitute our training set. We repeat this procedure until we reach enough samples for our surrogate function to be a good approximation.

# Approximation by neural networks I

General setup: two parts of the network:

## 2 Calibrator

In order to train the network we first need to build a training set. Conclusions:

# Approximation by neural networks I

General setup: two parts of the network:

## 2 Calibrator

In order to train the network we first need to build a training set. Conclusions:

- ▶ This can be done online, fast (within range of  $\sim 1$  second already unoptimized)
- ▶ Evaluation of parameters now more direct than via Monte Carlo. One minimizes now the distance between the (approximator) surrogate functions  $\hat{f}(\theta^*)$  and the volatility surface.

# Approximation by neural networks II

We see that after learning, calibrating **many** parameters is fast

# Approximation by neural networks II

We see that after learning, calibrating **many** parameters is fast  
⇒ approximate several models at the same time.

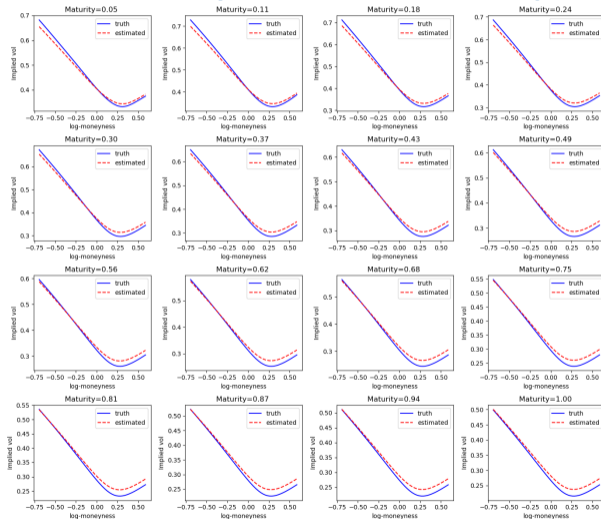
# Approximation by neural networks II

We see that after learning, calibrating **many** parameters is fast  
⇒ approximate several models at the same time.

New learning procedure:

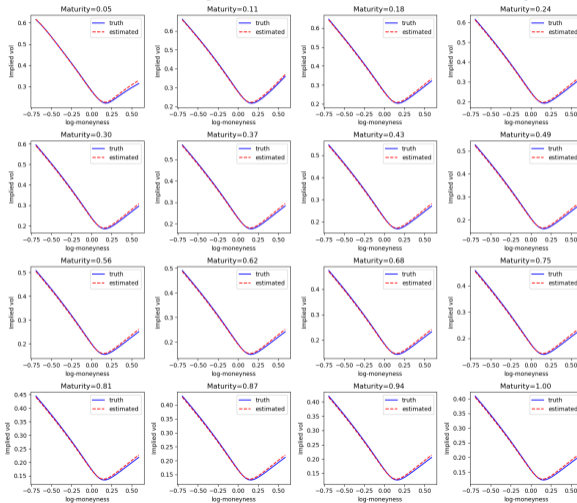
- ▶ Train the generator on several models at the same time (here Parameters from Heston and Bergomi parameters) in Monte Carlo experiments as before.
- ▶ Calibrate several models at the same time ⇒ determine the best-fit model to a given data (flag).
- ▶ Controlled experiments: train on both Bergomi and Heston ⇒ test on data generated by Heston.

# Approximation experiment via NN (Bergomi)





# Approximation experiment via NN (Bergomi)



# Conclusions and further steps

1. Adding further rough volatility models to the library to determine which model describes best the given set of data.
2. Calibration of Implied volatility gives an ideal objective function for learning (data  $\Leftrightarrow$  models).
3. Or: optimising the distribution of distributions directly (bypassing IV)

Thank you and  
Happy Birthday Yuri!

# FCLT for Hölder continuous processes

## FCLT for Hölder continuous processes

Define for any  $\omega \in \Omega$ ,  $n \geq 1$ ,  $t \in [0, T]$ , the approximating sequence

$$W_n(t, \omega) := \frac{1}{\sigma\sqrt{n}} \sum_{k=1}^j \xi_k(\omega) + \frac{nt - j}{\sigma\sqrt{n}} \xi_{j+1}(\omega), \quad \text{whenever } t \in \left[ \frac{j}{n}, \frac{j+1}{n} \right), \text{ for } j = 0, \dots, n-1.$$

where the family  $(\xi_i)_{i \geq 1}$  forms an iid sequence of centered random variables with finite moments of all orders and  $\mathbb{E}(\xi_1^2) = \sigma^2 > 0$ .

## FCLT for Hölder continuous processes

Define for any  $\omega \in \Omega$ ,  $n \geq 1$ ,  $t \in [0, T]$ , the approximating sequence

$$W_n(t, \omega) := \frac{1}{\sigma\sqrt{n}} \sum_{k=1}^j \xi_k(\omega) + \frac{nt-j}{\sigma\sqrt{n}} \xi_{j+1}(\omega), \quad \text{whenever } t \in \left[ \frac{j}{n}, \frac{j+1}{n} \right), \text{ for } j = 0, \dots, n-1.$$

where the family  $(\xi_i)_{i \geq 1}$  forms an iid sequence of centered random variables with finite moments of all orders and  $\mathbb{E}(\xi_1^2) = \sigma^2 > 0$ .

## Theorem (Donsker-Lamperti Theorem)

*The sequence  $(W_n)_{n \geq 1}$  converges weakly to a Brownian motion in  $(C^\alpha([0, T]), \|\cdot\|_\alpha)$  for all  $\alpha < \frac{1}{2}$ .*

# Monte-Carlo

# Monte-Carlo



# Monte-Carlo

The left-point approximation may be modified e.g.

$$\int_0^{\frac{T_i}{n}} g\left(\frac{T_i}{n} - s\right) dW_s \approx \frac{1}{\sqrt{n}\sigma} \sum_{k=1}^{j-1} g(t_k^*) \xi_k, \quad j = 0, \dots, n$$

where  $t_k^*$  is chosen optimally to match first and second moments

# Monte-Carlo

The left-point approximation may be modified e.g.

$$\int_0^{\frac{T_i}{n}} g\left(\frac{T_i}{n} - s\right) dW_s \approx \frac{1}{\sqrt{n}\sigma} \sum_{k=1}^{j-1} g(t_k^*) \xi_k, \quad j = 0, \dots, n$$

where  $t_k^*$  is chosen optimally to match first and second moments, i.e.,

$$g(t_k^*) = \sqrt{n \int_{\frac{T(k-1)}{n}}^{\frac{Tk}{n}} g(t-s)^2 ds}, \quad k = 1, \dots, n.$$

# Monte-Carlo

The left-point approximation may be modified e.g.

$$\int_0^{\frac{T_i}{n}} g\left(\frac{T_i}{n} - s\right) dW_s \approx \frac{1}{\sqrt{n}\sigma} \sum_{k=1}^{j-1} g(t_k^*) \xi_k, \quad j = 0, \dots, n$$

where  $t_k^*$  is chosen optimally to match first and second moments, i.e.,

$$g(t_k^*) = \sqrt{n \int_{\frac{T(k-1)}{n}}^{\frac{Tk}{n}} g(t-s)^2 ds}, \quad k = 1, \dots, n.$$

- ▶ This simple trick improves substantially the simulation (specially when  $\alpha$  is close to  $-1/2$ )

# Monte-Carlo

The left-point approximation may be modified e.g.

$$\int_0^{\frac{Tj}{n}} g\left(\frac{Tj}{n} - s\right) dW_s \approx \frac{1}{\sqrt{n}\sigma} \sum_{k=1}^{j-1} g(t_k^*) \xi_k, \quad j = 0, \dots, n$$

where  $t_k^*$  is chosen optimally to match first and second moments, i.e.,

$$g(t_k^*) = \sqrt{n \int_{\frac{T(k-1)}{n}}^{\frac{Tk}{n}} g(t-s)^2 ds}, \quad k = 1, \dots, n.$$

- ▶ This simple trick improves substantially the simulation (specially when  $\alpha$  is close to  $-1/2$ )
- ▶ The hybrid scheme also admits this trick