Turbocharging Monte Carlo pricing under rough volatility

Mikko Pakkanen

Department of Mathematics, Imperial College London, UK

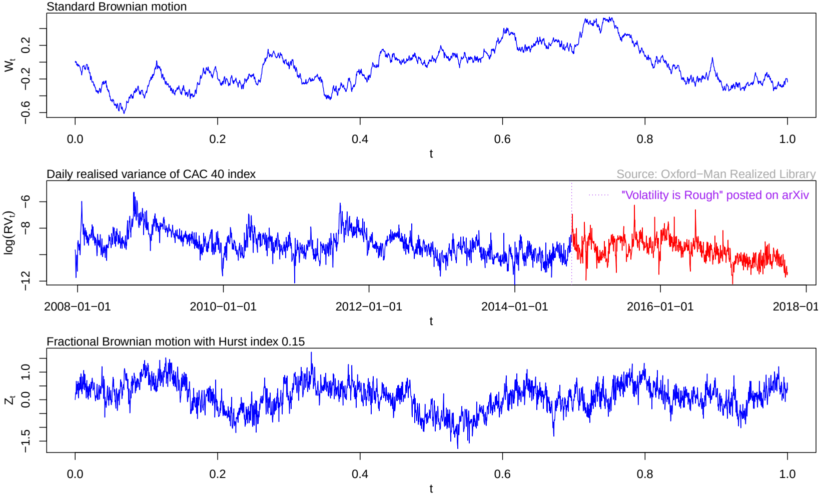
Jim Gatheral's 60th Birthday Conference
Courant Institute, New York, 14 October 2017

Joint work with Ryan McCrickerd

Imperial College London

Imperial Network of Excellence in Probabilistic Methods and Modelling

Volatility is (still) rough — 3rd anniversary!



The rough Bergomi model

The rough Bergomi model (Bayer, Friz, and Gatheral, 2016) is a non-Markovian extension of the variance curve model of Bergomi (2005).

The rough Bergomi model

The rough Bergomi model (Bayer, Friz, and Gatheral, 2016) is a non-Markovian extension of the variance curve model of Bergomi (2005).

This model, under a pricing measure, is given by

$$\frac{dS_t}{S_t} = \sqrt{V_t}\,\underbrace{\left(\rho dW_s + \sqrt{1-\rho^2}dW_s^{\perp}\right)}_{=:B_s},$$

The rough Bergomi model

The rough Bergomi model (Bayer, Friz, and Gatheral, 2016) is a non-Markovian extension of the variance curve model of Bergomi (2005).

This model, under a pricing measure, is given by

$$\frac{dS_t}{S_t} = \sqrt{V_t}\,\underbrace{\left(\rho dW_s + \sqrt{1-\rho^2}dW_s^{\perp}\right)}_{=:B_s},$$

where W and $W^{\perp}$ are independent Brownians and $\rho \in [-1, 1]$.

The rough Bergomi model

The rough Bergomi model (Bayer, Friz, and Gatheral, 2016) is a non-Markovian extension of the variance curve model of Bergomi (2005).

This model, under a pricing measure, is given by

$$\frac{dS_t}{S_t} = \sqrt{V_t}\,\underbrace{\left(\rho dW_s + \sqrt{1-\rho^2}dW_s^{\perp}\right)}_{=:B_s},$$

where W and $W^{\perp}$ are independent Brownians and $\rho \in [-1, 1]$. The spot variance V_t is a product $V_t = \xi_0(t)\mathcal{E}(\eta W^{\alpha})_t$ of

- the forward variance curve $t \mapsto \xi_0(t)$, known at time 0,
- the Wick exponential $\mathcal{E}(\eta W^{\alpha})_t = \exp\left(\eta W_t^{\alpha} - \frac{1}{2}\mathbf{Var}[\eta W_t^{\alpha}]\right)$ of a parameter $\eta > 0$ times a Gaussian random variable W_t^{α}.

The rough Bergomi model (cont.)

The random variable W_t^α follows the Gaussian Riemann–Liouville process

$$W_t^\alpha = \sqrt{2\alpha + 1} \int_0^t (t-s)^\alpha dW_s, \quad t \geq 0,$$

where the parameter $\alpha \in (-\frac{1}{2}, 0)$ controls the roughness of paths.

The rough Bergomi model (cont.)

The random variable W_t^α follows the Gaussian Riemann–Liouville process

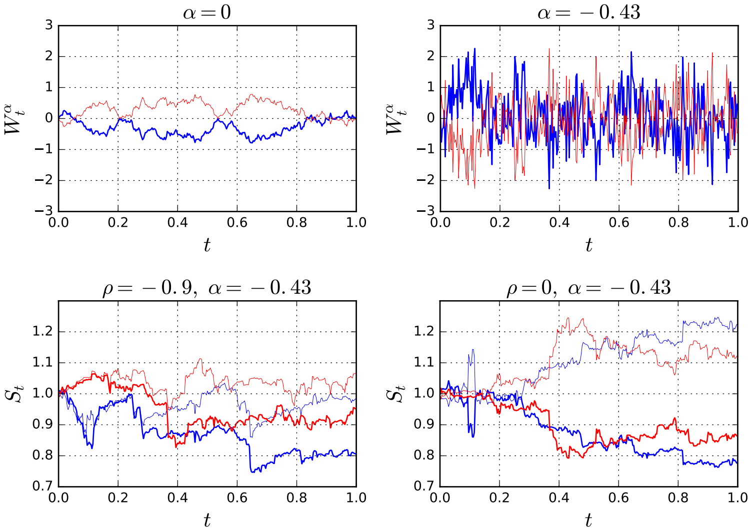
$$W_t^\alpha = \sqrt{2\alpha + 1} \int_0^t (t-s)^\alpha dW_s, \quad t \geq 0,$$

where the parameter $\alpha \in (-\frac{1}{2}, 0)$ controls the roughness of paths.

The paths of W^α have Hölder regularity $\alpha + \frac{1}{2}$ and locally look like the paths of a fractional Brownian motion with

$$H = \alpha + \frac{1}{2}.$$

Example: rough Bergomi paths

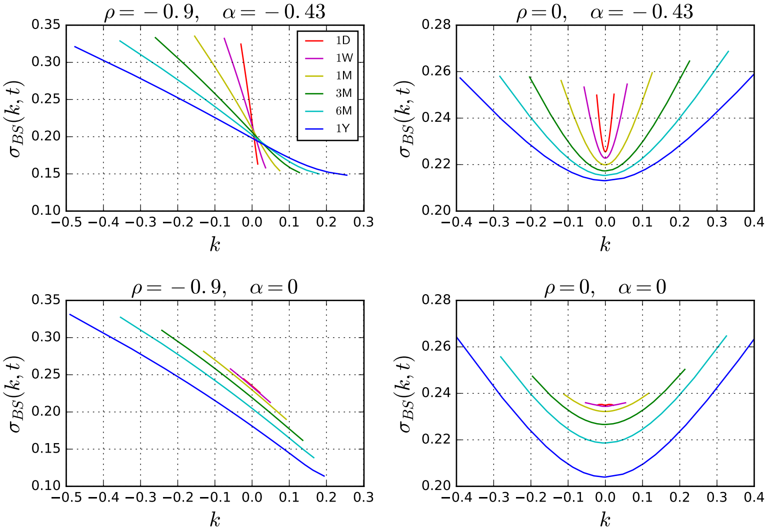


Intuition on the parameters

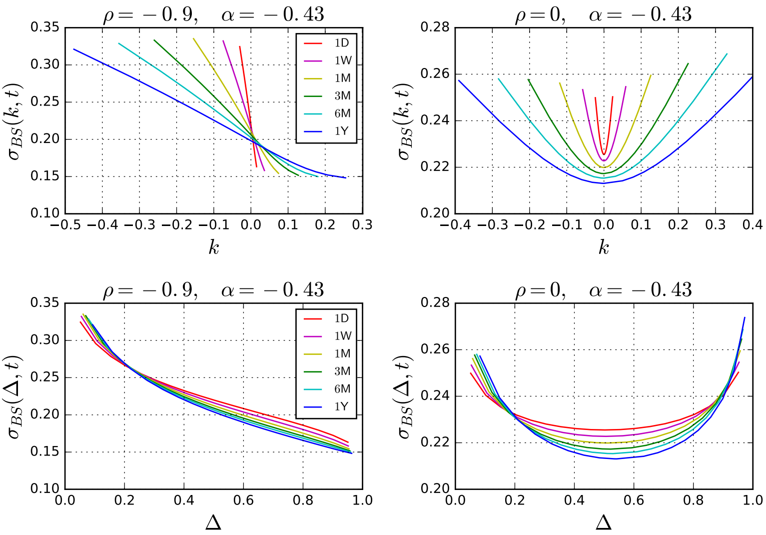
The rough Bergomi model has three time-homogeneous parameters, α, η, and ρ, with the following interpretations in terms of the implied volatility surface:

- η — smile,
- ρ — skew,
- α — near-maturity explosion (of smile and skew).

Example: rough Bergomi smiles



Example: rough Bergomi smiles



Pricing under rough volatility

Simulating rough volatility

Variance reduction methods

Pricing by Monte Carlo

The introduction of the rough Bergomi model has launched a quest for efficient pricing methods for the model.

Pricing by Monte Carlo

The introduction of the rough Bergomi model has launched a quest for efficient pricing methods for the model.

The model is non-affine and non-Markovian so standard methods (PDEs, characteristic functions) seem inapplicable.

Pricing by Monte Carlo

The introduction of the rough Bergomi model has launched a quest for efficient pricing methods for the model.

The model is non-affine and non-Markovian so standard methods (PDEs, characteristic functions) seem inapplicable.

Currently, the only operational pricing method for mere vanilla options is Monte Carlo.

Pricing by Monte Carlo

The introduction of the rough Bergomi model has launched a quest for efficient pricing methods for the model.

The model is non-affine and non-Markovian so standard methods (PDEs, characteristic functions) seem inapplicable.

Currently, the only operational pricing method for mere vanilla options is Monte Carlo.

Thus it is worthwhile to try to optimise, “turbocharge”, Monte Carlo pricing as much as possible.

Pricing by Monte Carlo

The introduction of the rough Bergomi model has launched a quest for efficient pricing methods for the model.

The model is non-affine and non-Markovian so standard methods (PDEs, characteristic functions) seem inapplicable.

Currently, the only operational pricing method for mere vanilla options is Monte Carlo.

Thus it is worthwhile to try to optimise, “turbocharge”, Monte Carlo pricing as much as possible.

In general, a good Monte Carlo pricer should have:

- low bias, to avoid systematic error,
- low variance, such that good accuracy can be achieved in reasonable runtime.

Simulating the rough Bergomi model

The rough Bergomi model has a simple stochastic structure — all randomness comes from a bivariate Gaussian process (B, W^{α}), while S can be approximated by Riemann sums.

Simulating the rough Bergomi model

The rough Bergomi model has a simple stochastic structure — all randomness comes from a bivariate Gaussian process (B, W^α), while S can be approximated by Riemann sums.

The covariance structure of (B, W^α) is not difficult to work out, so we could simulate exactly

$$\boldsymbol{X} := ((B_0, W^\alpha_0), (B_{1/n}, W^\alpha_{1/n}), (B_{2/n}, W^\alpha_{2/n}), \ldots, (B_{\lfloor nt \rfloor/n}, W^\alpha_{\lfloor nt \rfloor/n}))$$

by sampling from a $2\lfloor nt \rfloor$-dimensional Gaussian distribution.

Simulating the rough Bergomi model

The rough Bergomi model has a simple stochastic structure — all randomness comes from a bivariate Gaussian process (B, W^α), while S can be approximated by Riemann sums.

The covariance structure of (B, W^α) is not difficult to work out, so we could simulate exactly

$$\boldsymbol{X} := \big((B_0, W^\alpha_0), (B_{1/n}, W^\alpha_{1/n}), (B_{2/n}, W^\alpha_{2/n}), \ldots, (B_{\lfloor nt \rfloor/n}, W^\alpha_{\lfloor nt \rfloor/n})\big)$$

by sampling from a $2\lfloor nt \rfloor$-dimensional Gaussian distribution.

The simulation is based on the Cholesky factorisation of the covariance matrix of $\boldsymbol{X}$, which requires $\mathcal{O}(n^3)$ flops.

Simulating the rough Bergomi model

The rough Bergomi model has a simple stochastic structure — all randomness comes from a bivariate Gaussian process (B, W^α), while S can be approximated by Riemann sums.

The covariance structure of (B, W^α) is not difficult to work out, so we could simulate exactly

$$\boldsymbol{X} := \left((B_0, W^\alpha_0), (B_{1/n}, W^\alpha_{1/n}), (B_{2/n}, W^\alpha_{2/n}), \ldots, (B_{\lfloor nt \rfloor/n}, W^\alpha_{\lfloor nt \rfloor/n})\right)$$

by sampling from a $2\lfloor nt \rfloor$-dimensional Gaussian distribution.

The simulation is based on the Cholesky factorisation of the covariance matrix of $\boldsymbol{X}$, which requires $\mathcal{O}(n^3)$ flops.

The Cholesky factor needs to be computed only once, but subsequent realisations of $\boldsymbol{X}$ still take $\mathcal{O}(n^2)$ flops.

Approximating the process W^α

Exact simulation being too expensive, we seek to approximate the process W^α.

Approximating the process W^α

Exact simulation being too expensive, we seek to approximate the process W^α.

A naive approach would be to use forward Riemann sums

$$W^\alpha_{i/n} = \sum_{k=1}^{i} \int_{\frac{i}{n}-\frac{k}{n}}^{\frac{i}{n}-\frac{k-1}{n}} \left(\tfrac{i}{n}-s\right)^\alpha dW_s \approx \sum_{k=1}^{i} \left(\tfrac{k}{n}\right)^\alpha \left(W_{\frac{i}{n}-\frac{k-1}{n}} - W_{\frac{i}{n}-\frac{k}{n}}\right) =: \widehat{W}^{\alpha,n}_{i/n}.$$

Since $\widehat{W}^{\alpha,n}_{i/n}$ is a discrete convolution, $\widehat{W}^{\alpha,n}_{0}, \widehat{W}^{\alpha,n}_{1/n}, \ldots, \widehat{W}^{\alpha,n}_{\lfloor nt \rfloor/n}$ can be generated (using FFT) in $\mathcal{O}(n \log n)$ flops.

Approximating the process W^α

Exact simulation being too expensive, we seek to approximate the process W^α.

A naive approach would be to use forward Riemann sums

$$W^\alpha_{i/n} = \sum_{k=1}^{i} \int_{\frac{i}{n}-\frac{k}{n}}^{\frac{i}{n}-\frac{k-1}{n}} \left(\tfrac{i}{n}-s\right)^\alpha dW_s \approx \sum_{k=1}^{i} \left(\tfrac{k}{n}\right)^\alpha \left(W_{\frac{i}{n}-\frac{k-1}{n}} - W_{\frac{i}{n}-\frac{k}{n}}\right) =: \widehat{W}^{\alpha,n}_{i/n}.$$

Since $\widehat{W}^{\alpha,n}_{i/n}$ is a discrete convolution, $\widehat{W}^{\alpha,n}_{0}, \widehat{W}^{\alpha,n}_{1/n}, \ldots, \widehat{W}^{\alpha,n}_{\lfloor nt \rfloor/n}$ can be generated (using FFT) in $\mathcal{O}(n \log n)$ flops.

However:

- Forward Riemann sums are inaccurate since the integrand $s \mapsto \left(\frac{i}{n} - s\right)^\alpha$ has a singularity.
- This leads to biased estimates of implied volatility.

The hybrid scheme

The hybrid scheme of Bennedsen, Lunde, and Pakkanen (2017) fixes the deficiencies of forward Riemann sums by using:

$$\widetilde{W}^{\alpha,n}_{i/n} := \underbrace{\sum_{k=1}^{\kappa} \int_{\frac{i}{n}-\frac{k}{n}}^{\frac{i}{n}-\frac{k-1}{n}} \left(\tfrac{i}{n}-s\right)^{\alpha} dW_s}_{\text{exact for } \kappa \text{ slices}} + \underbrace{\sum_{k=\kappa+1}^{i} \left(\tfrac{b_k}{n}\right)^{\alpha} \left(W_{\frac{i}{n}-\frac{k-1}{n}} - W_{\frac{i}{n}-\frac{k}{n}}\right)}_{\text{Riemann sum for the rest}},$$

where $b_k \in [k-1, k] \setminus \{0\}$ can be chosen (MSE) optimally.

The hybrid scheme

The hybrid scheme of Bennedsen, Lunde, and Pakkanen (2017) fixes the deficiencies of forward Riemann sums by using:

$$\widetilde{W}^{\alpha,n}_{i/n} := \underbrace{\sum_{k=1}^{\kappa} \int_{\frac{i}{n}-\frac{k}{n}}^{\frac{i}{n}-\frac{k-1}{n}} \left(\frac{i}{n}-s\right)^{\alpha} dW_s}_{\text{exact for } \kappa \text{ slices}} + \underbrace{\sum_{k=\kappa+1}^{i} \left(\frac{b_k}{n}\right)^{\alpha} \left(W_{\frac{i}{n}-\frac{k-1}{n}} - W_{\frac{i}{n}-\frac{k}{n}}\right)}_{\text{Riemann sum for the rest}},$$

where $b_k \in [k-1, k] \setminus \{0\}$ can be chosen (MSE) optimally.

Usually $\kappa = 1$ suffices.

The hybrid scheme

The hybrid scheme of Bennedsen, Lunde, and Pakkanen (2017) fixes the deficiencies of forward Riemann sums by using:

$$\widetilde{W}^{\alpha,n}_{i/n} := \underbrace{\sum_{k=1}^{\kappa} \int_{\frac{i}{n}-\frac{k}{n}}^{\frac{i}{n}-\frac{k-1}{n}} \left(\frac{i}{n}-s\right)^{\alpha} dW_s}_{\text{exact for } \kappa \text{ slices}} + \underbrace{\sum_{k=\kappa+1}^{i} \left(\frac{b_k}{n}\right)^{\alpha} \left(W_{\frac{i}{n}-\frac{k-1}{n}} - W_{\frac{i}{n}-\frac{k}{n}}\right)}_{\text{Riemann sum for the rest}},$$

where $b_k \in [k-1, k] \setminus \{0\}$ can be chosen (MSE) optimally.

Usually $\kappa = 1$ suffices.

The variates $\widetilde{W}^{\alpha,n}_0, \widetilde{W}^{\alpha,n}_{1/n}, \ldots, \widetilde{W}^{\alpha,n}_{\lfloor nt \rfloor/n}$ can be generated by sampling $\lfloor nt \rfloor$ iid draws from a $\kappa + 1$-dimensional Gaussian distribution and computing a discrete convolution.

The hybrid scheme

The hybrid scheme of Bennedsen, Lunde, and Pakkanen (2017) fixes the deficiencies of forward Riemann sums by using:

$$\widetilde{W}^{\alpha,n}_{i/n} := \underbrace{\sum_{k=1}^{\kappa} \int_{\frac{i}{n}-\frac{k}{n}}^{\frac{i}{n}-\frac{k-1}{n}} \left(\frac{i}{n}-s\right)^{\alpha} dW_s}_{\text{exact for } \kappa \text{ slices}} + \underbrace{\sum_{k=\kappa+1}^{i} \left(\frac{b_k}{n}\right)^{\alpha} \left(W_{\frac{i}{n}-\frac{k-1}{n}} - W_{\frac{i}{n}-\frac{k}{n}}\right)}_{\text{Riemann sum for the rest}},$$

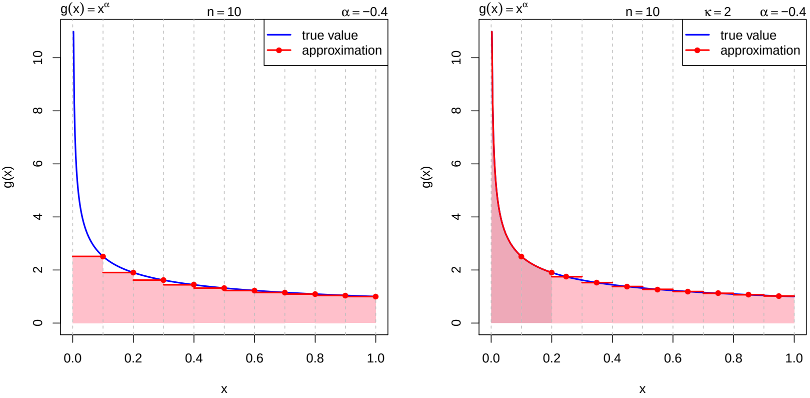
where $b_k \in [k-1, k] \setminus \{0\}$ can be chosen (MSE) optimally.

Usually $\kappa = 1$ suffices.

The variates $\widetilde{W}^{\alpha,n}_{0}, \widetilde{W}^{\alpha,n}_{1/n}, \ldots, \widetilde{W}^{\alpha,n}_{\lfloor nt \rfloor/n}$ can be generated by sampling $\lfloor nt \rfloor$ iid draws from a $\kappa + 1$-dimensional Gaussian distribution and computing a discrete convolution.

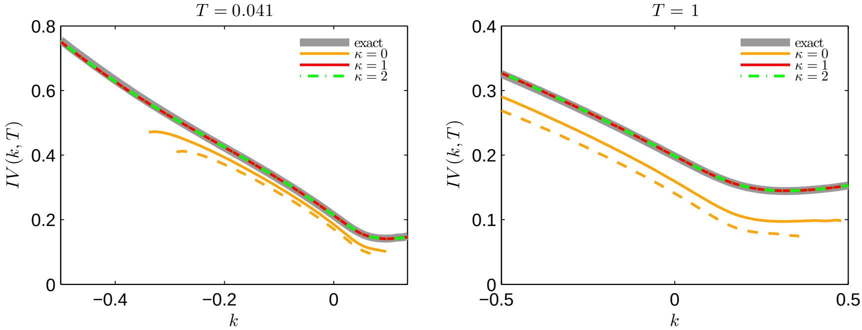
Again, this requires only $\mathcal{O}(n \log n)$ flops.

Approximating $x \mapsto x^{\alpha}$



Forward Riemann sums vs. the hybrid scheme

Numerical results: implied volatility smiles



Solid/patterned line: optimal b_k
Dashed line: $b_k = k$.

S_0	$\xi_0(t)$	η	α	ρ	n	paths
1	0.235^2	1.9	−0.43	−0.9	500	10^6

Pricing under rough volatility

Simulating rough volatility

Variance reduction methods

Towards variance reduction

While the hybrid scheme simulates the variance process V efficiently, ceteris paribus it does essentially nothing to the variance of the Monte Carlo pricer.

Towards variance reduction

While the hybrid scheme simulates the variance process V efficiently, ceteris paribus it does essentially nothing to the variance of the Monte Carlo pricer.

Indeed there is scope for improving the efficiency of the pricer by deploying a "cocktail" of variance reduction methods.

Towards variance reduction

While the hybrid scheme simulates the variance process V efficiently, ceteris paribus it does essentially nothing to the variance of the Monte Carlo pricer.

Indeed there is scope for improving the efficiency of the pricer by deploying a "cocktail" of variance reduction methods.

To this end, we work with price estimators of the form

$$\hat{P}_n(k,t) := \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{\alpha}_n Y_i) - \hat{\alpha}_n \mathbf{E}[Y],$$

where $(X_1, Y_1), \ldots, (X_n, Y_n)$ are identical copies of a random vector (X, Y) and $\hat{\alpha}_n$ is a free parameter, to be defined shortly.

Base estimator

Our reference estimator, which we call the Base estimator, uses

$$X := f(S_t) := \begin{cases} (S_t - S_0 e^k)^+, & k \geq 0, \\ (S_0 e^k - S_t)^+, & k < 0, \end{cases}$$

$$Y := 0.$$

Base estimator

Our reference estimator, which we call the Base estimator, uses

$$X := f(S_t) := \begin{cases} (S_t - S_0 e^k)^+, & k \geq 0, \\ (S_0 e^k - S_t)^+, & k < 0, \end{cases}$$
$$Y := 0.$$

This is really just the "naive" estimator, except that we price the out-of-the-money European call/put, which is less noisy, and derive implied volatility from its price.

Base estimator

Our reference estimator, which we call the Base estimator, uses

$$X := f(S_t) := \begin{cases} (S_t - S_0 e^k)^+, & k \geq 0, \\ (S_0 e^k - S_t)^+, & k < 0, \end{cases}$$
$$Y := 0.$$

This is really just the “naive” estimator, except that we price the out-of-the-money European call/put, which is less noisy, and derive implied volatility from its price.

Without loss of generality, assume $S_0 = 1$.

Mixing formula

The well-known result of Romano and Touzi (1997) implies that

$$\mathbf{E}[f(S_t)] = \mathbf{E}\left[\mathrm{BS}\left((1-\rho^2)\int_0^t V_s ds, S_t^W, k\right)\right],$$

where BS is the appropriate Black–Scholes function, $dS_t^W/S_t^W = \sqrt{V_t}\,\rho\, dW_t$

Mixing formula

The well-known result of Romano and Touzi (1997) implies that

$$\mathbf{E}[f(S_t)] = \mathbf{E}\left[\mathrm{BS}\left((1-\rho^2)\int_0^t V_s ds, S_t^W, k\right)\right],$$

where BS is the appropriate Black–Scholes function,
$dS_t^W/S_t^W = \sqrt{V_t}\,\rho\,dW_t$

This suggests that we could use

$$X := \mathrm{BS}\left((1-\rho^2)\int_0^t V_s ds, S_t^W, k\right).$$

This method alone is rather effective in reducing variance when $\rho \approx 0$, but its benefits evaporate as $\rho \to -1$.

Control variate

Inspired by the idea of Bergomi (2016) of using a timer option as a control variate, we choose

$$Y := \mathrm{BS}\left(\rho^2\left(\hat{Q}_n - \int_0^t V_s ds\right), S_t^W, k\right),$$

where $\hat{Q}_n$ is a free parameter, "variance budget", to be chosen post simulation.

Control variate

Inspired by the idea of Bergomi (2016) of using a timer option as a control variate, we choose

$$Y := \mathrm{BS}\left(\rho^2\left(\hat{Q}_n - \int_0^t V_s ds\right), S_t^W, k\right),$$

where $\hat{Q}_n$ is a free parameter, "variance budget", to be chosen post simulation.

By a martingale argument,

$$\mathbf{E}[Y] = \mathrm{BS}(\rho^2 \hat{Q}_n, 1, k)$$

Mixed estimator

Our "turbocharged" Mixed estimator (McCrickerd and Pakkanen, 2017) is given by

$$X := \mathrm{BS}\left((1-\rho^2)\int_0^t V_s ds, S_t^W, k\right),$$
$$Y := \mathrm{BS}\left(\rho^2\left(\hat{Q}_n - \int_0^t V_s ds\right), S_t^W, k\right).$$

Mixed estimator

Our "turbocharged" Mixed estimator (McCrickerd and Pakkanen, 2017) is given by

$$X := \mathrm{BS}\Big((1-\rho^2)\int_0^t V_s ds, S_t^W, k\Big),$$

$$Y := \mathrm{BS}\Big(\rho^2\Big(\hat{Q}_n - \int_0^t V_s ds\Big), S_t^W, k\Big).$$

We set, post simulation,

$$\hat{\alpha}_n := -\frac{\sum_{i=1}^n (X_i - \overline{X}_i)(Y_i - \overline{Y}_i)}{\sum_{i=1}^n (Y_i - \overline{Y}_i)^2}, \quad \hat{Q}_n := \max_{i=1,\dots,n} \Big(\int_0^t V_s ds\Big)_i.$$

Mixed estimator

Our “turbocharged” Mixed estimator (McCrickerd and Pakkanen, 2017) is given by

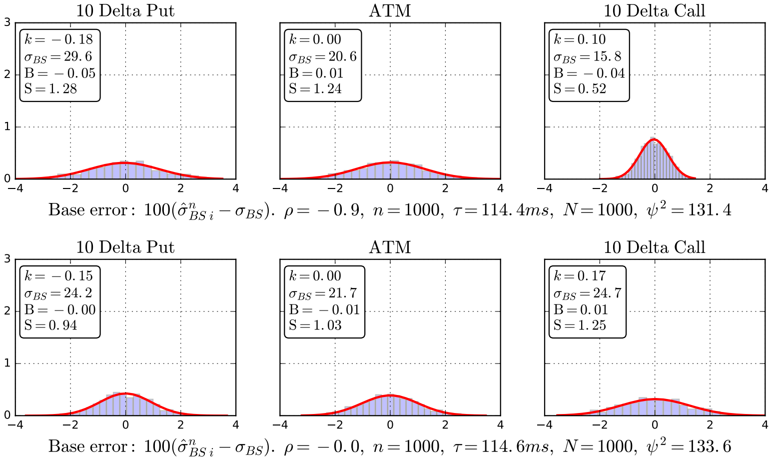
$$X := \mathrm{BS}\Big((1-\rho^2)\int_0^t V_s ds, S_t^W, k\Big),$$
$$Y := \mathrm{BS}\Big(\rho^2\Big(\hat{Q}_n - \int_0^t V_s ds\Big), S_t^W, k\Big).$$

We set, post simulation,

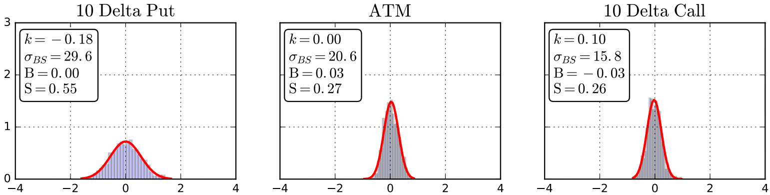
$$\hat{\alpha}_n := -\frac{\sum_{i=1}^n (X_i - \overline{X}_i)(Y_i - \overline{Y}_i)}{\sum_{i=1}^n (Y_i - \overline{Y}_i)^2}, \quad \hat{Q}_n := \max_{i=1,\dots,n} \Big(\int_0^t V_s ds\Big)_i.$$

Additionally, we couple (X_{2i-1}, Y_{2i-1}) and (X_{2i}, Y_{2i}) for any $i \geq 1$ by using antithetic pairs (B, W) and $(-B, -W)$ as drivers.

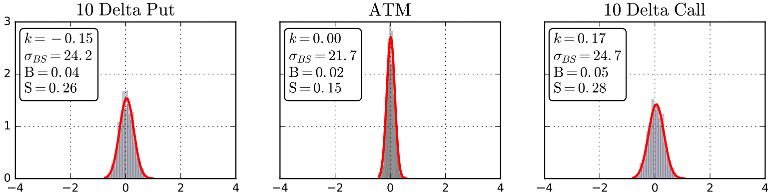
Numerical results: Base estimator



Numerical results: Mixed estimator



Mixed error: $100(\hat{\sigma}^n_{BS\,i}-\sigma_{BS})$. $\rho=-0.9$, $n=1000$, $\tau=70.6ms$, $N=1000$, $\psi^2=10.4$



Mixed error: $100(\hat{\sigma}^n_{BS\,i}-\sigma_{BS})$. $\rho=-0.0$, $n=1000$, $\tau=69.8ms$, $N=1000$, $\psi^2=3.9$

Calibration experiment

Ultimately, the goal of this simulation methodology is to calibrate the rough Bergomi model to a volatility surface.

Calibration experiment

Ultimately, the goal of this simulation methodology is to calibrate the rough Bergomi model to a volatility surface.

We conduct a simple experiment to demonstrate what difference using the Mixed estimator in calibration makes.

Calibration experiment

Ultimately, the goal of this simulation methodology is to calibrate the rough Bergomi model to a volatility surface.

We conduct a simple experiment to demonstrate what difference using the Mixed estimator in calibration makes.

We calibrate the parameters η and ρ to a 3M rough Bergomi reference smile at 19 points, minimising RMSE using L-BFGS-B.

Calibration experiment

Ultimately, the goal of this simulation methodology is to calibrate the rough Bergomi model to a volatility surface.

We conduct a simple experiment to demonstrate what difference using the Mixed estimator in calibration makes.

We calibrate the parameters η and ρ to a 3M rough Bergomi reference smile at 19 points, minimising RMSE using L-BFGS-B.

We initialise the solver at the true parameter values and let it run for 700 milliseconds.

Calibration experiment

Ultimately, the goal of this simulation methodology is to calibrate the rough Bergomi model to a volatility surface.

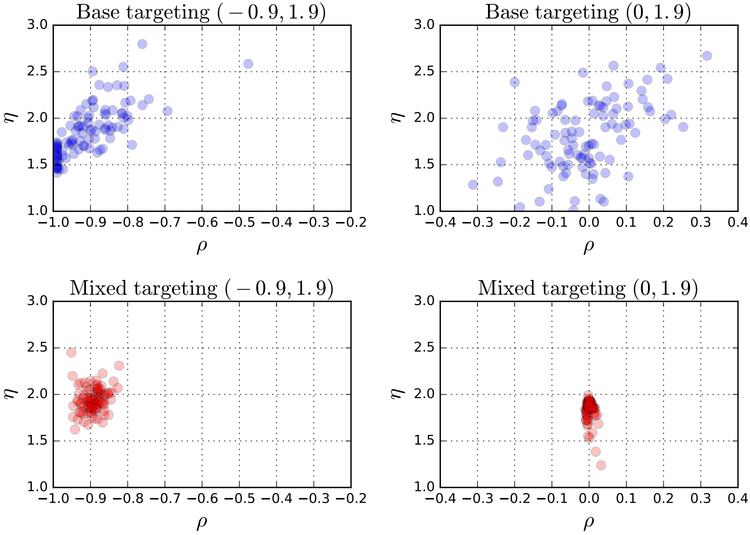
We conduct a simple experiment to demonstrate what difference using the Mixed estimator in calibration makes.

We calibrate the parameters η and ρ to a 3M rough Bergomi reference smile at 19 points, minimising RMSE using L-BFGS-B.

We initialise the solver at the true parameter values and let it run for 700 milliseconds.

Throughout the experiment, we use $n = 1\,000$.

Numerical results: calibration experiment



Some alternative variance reduction methods

- Multilevel Monte Carlo — compatible with the hybrid scheme, but does not appear to effective in reducing variance in this setting (Mamallan, 2017).

Some alternative variance reduction methods

- Multilevel Monte Carlo — compatible with the hybrid scheme, but does not appear to effective in reducing variance in this setting (Mamallan, 2017).
- Importance sampling — seems unattractive as it would need to be tuned strike by strike.

Some alternative variance reduction methods

- Multilevel Monte Carlo — compatible with the hybrid scheme, but does not appear to effective in reducing variance in this setting (Mamallan, 2017).
- Importance sampling — seems unattractive as it would need to be tuned strike by strike.
- Quasi Monte Carlo (Sobol sequences etc) — applicable and useful here, albeit the speed-up appears not to be dramatic.

References

- C. Bayer, P. K. Friz, and J. Gatheral (2016): Pricing under rough volatility. *Quant. Finance* **16**(6), 887–904.
- M. Bennedsen, A. Lunde, and M. S. Pakkanen (2017): Hybrid scheme for Brownian semistationary processes. *Finance Stoch.* **21**(4) 931–965.
- L. Bergomi (2005): Smile dynamics II, *Risk* October 2005, 67–73.
- L. Bergomi (2016): *Stochastic Volatility Modeling*. CRC Press, Boca Raton.
- C. Mamallan (2017): *Efficient implementation of the rBergomi model with comparison to the Heston model.* Unpublished MSci dissertation, Imperial College London.
- R. McCrickerd and M. S. Pakkanen (2017): Turbocharging Monte Carlo pricing for the rough Bergomi model. Preprint: `http://arxiv.org/abs/1708.02563`
- M. Romano and N. Touzi (1997): Contingent claims and market completeness in a stochastic volatility model. *Math. Finance* **7**(4), 399–412.

Implementation

Python implementation of turbocharged pricing along with a Jupyter notebook are available from:

https://github.com/ryanmccrickerd/roughbergomi

Finally...

Happy birthday Jim!