QuantMinds International

Intercontinential O2 London, November 17, 2025

Rough volatility workshop

Lecture 4: Quadratic Rough Heston

Outline of lecture

- The quadratic rough Heston (QRH) model
- Forward variance and forward volatility curves
- A microstructural foundation for the QRH model
- The QRH simulation scheme
  - Simulating SPX
  - Simulating VIX
- Estimation of the skew stickiness ratio (SSR)

Set up the environment

In [1]:
```
# Download .zip from GitHub
release_url <- "https://github.com/jgatheral/RoughVolatilityWorkshop2025/raw/main/QM_2025.zip'
local_zip <- "QM_2025.zip"

download.file(release_url, local_zip, mode = "wb")
unzip(local_zip)
```

In [2]:
```
# The required packages

library(colorspace)
library(doParallel)
library(foreach)
library(gsl)
library(MittagLeffleR)
library(statmod)
library(stinepack)
library(repr)

options(repr.plot.width = 10, repr.plot.height = 7, repr.plot.res = 150)
```

```
Loading required package: foreach

Loading required package: iterators

Loading required package: parallel
```

In [3]:
```
bl <- "royalblue"
rd <- "red2"
pk <- "hotpink1"
gr <- "green4"
```

```
br <- "brown"
pu <- "purple"
```

Some R-code

In [4]:
```
source("afv_simulation.R")
source("black_scholes.R")
source("gamma_kernel.R")
source("plot_ivols_mc.R")
source("qrh_gamma_y_from_xi.R")
source("qrh_simulation.R")
```

The quadratic rough Heston model

The quadratic rough Heston (QRH) model of [QRH][9] may be written as

$$\begin{aligned}\frac{dS_t}{S_t} &= -\sqrt{V_t}\,dW_t,\\ V_t &= Y_t^2 + \underline{v},\end{aligned} \tag{1}$$

where the minimum variance $\underline{v} \geq 0$, and

$$Y_t = \bar{Y} + \int_{-\infty}^t \kappa(t-s)\,\sqrt{V_s}\,dW_s = \bar{Y} - \int_{-\infty}^t \kappa(t-s)\,\frac{dS_s}{S_s} \tag{2}$$

is a weighted average of historical returns and $\kappa(\cdot)$ a kernel function.

Dynamics in martingale form

Taking the conditional expectation of Y_u for $u > t$ gives,

$$Y_u = y_t(u) + \int_t^u \kappa(u-s)\,\sqrt{V_s}\,dW_s \tag{3}$$

where

$$y_t(u) := \mathbb{E}_t\left[Y_u\right] = \bar{Y} + \int_{-\infty}^t \kappa(u-s)\,\sqrt{V_s}\,dW_s.$$

Then, $y_\cdot(u)$ is a martingale and

$$dy_t(u) = \kappa(u-t)\,\sqrt{V_t}\,dW_t.$$

The variance of Y_u

Thus $\mathbb{E}_t\left[Y_u\right] = y_t(u)$ and from (1) and Itô's isometry,

$$\mathrm{var}_t[Y_u] = \int_t^u \xi_t(s)\,\kappa(u-s)^2\,ds,$$

where $\xi_t(s) := \mathbb{E}_t\left[V_s\right]$ for all $s \geq t$.

- Under QR Heston, we have the mean and variance of Y_u in (quasi-) closed form.
- We will sometimes refer to the curve $y_t(u)$ as the *forward volatility curve*.

The forward variance curve

From the model definition, for $u > t$, $V_u = y_u(u)^2 + \underline{v}$ so applying Itô's formula, we have

$$\begin{aligned}\xi_t(u) &= \mathbb{E}_t\left[y_u(u)^2\right] + \underline{v}\\ &= y_t(u)^2 + \mathbb{E}_t\left[\int_t^u d\langle y_\cdot(u)\rangle_s\right] + \underline{v}\\ &= y_t(u)^2 + \int_t^u \xi_t(s)\,\kappa(u-s)^2\,ds + \underline{v}.\end{aligned} \tag{4}$$

Alternatively,

$$y_t(u)^2 = \xi_t(u) - \int_t^u \xi_t(s)\,\kappa(u-s)^2\,ds - \underline{v},$$

- Thus $y_t(u)$ may be easily imputed from the forward variance curve.

The stationary state

In the stationary state, $\mathbb{E}\left[Y_t\right] = \bar{Y}$ and

$$\begin{aligned}\bar{V} := \mathbb{E}\left[V_t\right] &= \mathbb{E}\left[\underline{v} + Y_t^2\right]\\ &= \underline{v} + \bar{Y}^2 + \int_{-\infty}^t \kappa(t-s)^2\,\mathbb{E}\left[V_s\right]ds\\ &= \underline{v} + \bar{Y}^2 + \|\kappa^2\|\,\bar{V}.\end{aligned} \tag{5}$$

where

$$\|\kappa^2\| = \int_0^\infty \kappa(s)^2\,ds.$$

Admissibility condition on the kernel

- A necessary condition on the kernel then follows:

$$\|\kappa^2\| < 1.$$

- Violated by the power-law kernel!

Pricing with the QRH model

- Like all rough volatility moodels, to compute quantities of interest, we need to define the state vector and the model (kernel) parameters.
  - The (infinite) state vector is the forward variance curve $\xi_t(u)$.
  - The model parameters are the parameters of the chosen kernel.

Choice of kernel

- For implementation purposes, for flexibility as well as the existence of a closed-form expression for the resolvent kernel K, we choose the gamma kernel. we choose the gamma kernel given by

$$\kappa(\tau) = \frac{\nu}{\Gamma(\alpha)}\,\tau^{\alpha-1}\,e^{-\lambda\tau}, \tag{2}$$

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

- With this choice, the QR Heston model has only four parameters: H, ν, λ, and $\underline{v}$.

Kernel parameters

- The QRH model was calibrated to SPX and VIX smiles as of 15-Feb- 2023 by Florian Bourgey of Bloomberg.
  - A non-trivial task.
  - Efficient calibration of the model is work in progress.
- He found the following QRH parameters (with $c = \underline{v}$):

In [5]:
```
params0 <- list(al = 0.568, lam = 9.68, nu = 0.572, c = 8.1e-3)
```

Kernel code

In [6]:
```
kGamma
```

```
function (params)
{
    Vectorize(function(tau) {
        al <- params$al
        lam <- params$lam
        nu <- params$nu
        res <- (nu/gamma(al)) * tau^(al - 1) * exp(-lam * tau)
        return(res)
    })
}
```

Admissibility condition for the gamma kernel

- The admissibilty condition then evaluates to

$$\|\kappa^2\| = \int_0^\infty \kappa(s)^2\,ds = \mathcal{L}[\kappa^2](0) = \frac{\hat{\nu}^2}{(2\lambda)^{2H}} < 1,$$

where $\hat{\nu} := \nu\sqrt{\Gamma(2H)}/\Gamma(\alpha)$.

With our fitted parameters

We check the necessary condition with our parameters.

In [7]:
```
converge.test <- function(params) {
  al <- params$al
  H <- al - 1 / 2
  lam <- params$lam
  nu <- params$nu
  return(((nu / gamma(al))^2 * gamma(2 * H)) / (2 * lam)^(2 * H))
}
```

In [8]:
```
converge.test(params0)
```

```
0.613702132553213
```

So our paraameter set is admissible.

Plot of the kernel

In [9]:
```
curve(kGamma(params0)(x),from=0,to=,col="red2",xlab=expression(tau),ylab=expression(kappa),lw
```

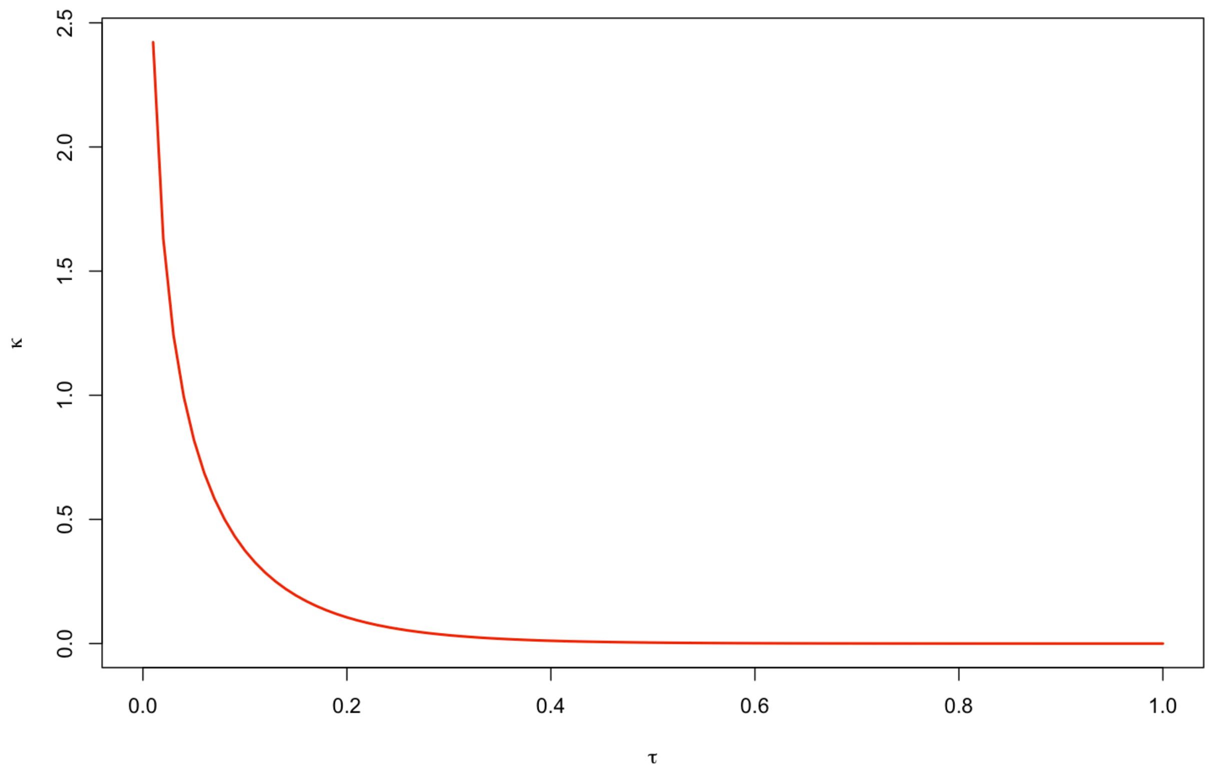


Figure 1: Plot of the kernel

Log-log plot of the kernel

In [10]:
```
curve(log(kGamma(params0)(exp(x))),from=-10,to=log(.2),col="red2",xlab=expression(tau),ylab=e
```

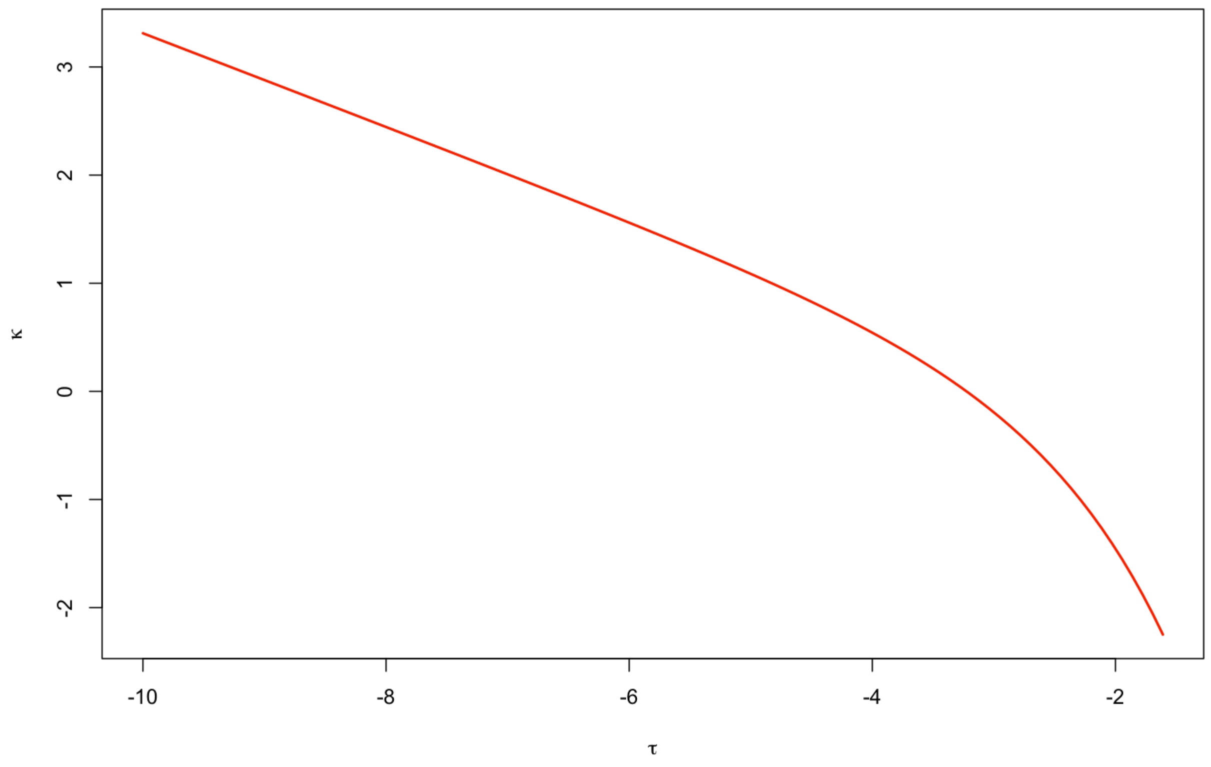


Figure 2: Log-log plot of the kernel.

The forward variance curve $\xi_t(u)$ as of 15-Feb 2023

- We use our usual forward variance curve from 15-Feb 2023.

In [11]:
```
load(file = "xi_20230215.rData")
xi <- xiCurveObj$getForwardVarCurve()
```

Plot $\xi_t(u)$

In [12]:
```
curve(xi,
  from = 0, to = .2, col = "red2", n = 100, lwd = 2,
  xlab = "Maturity u", ylab = expression(xi(u))
)
```

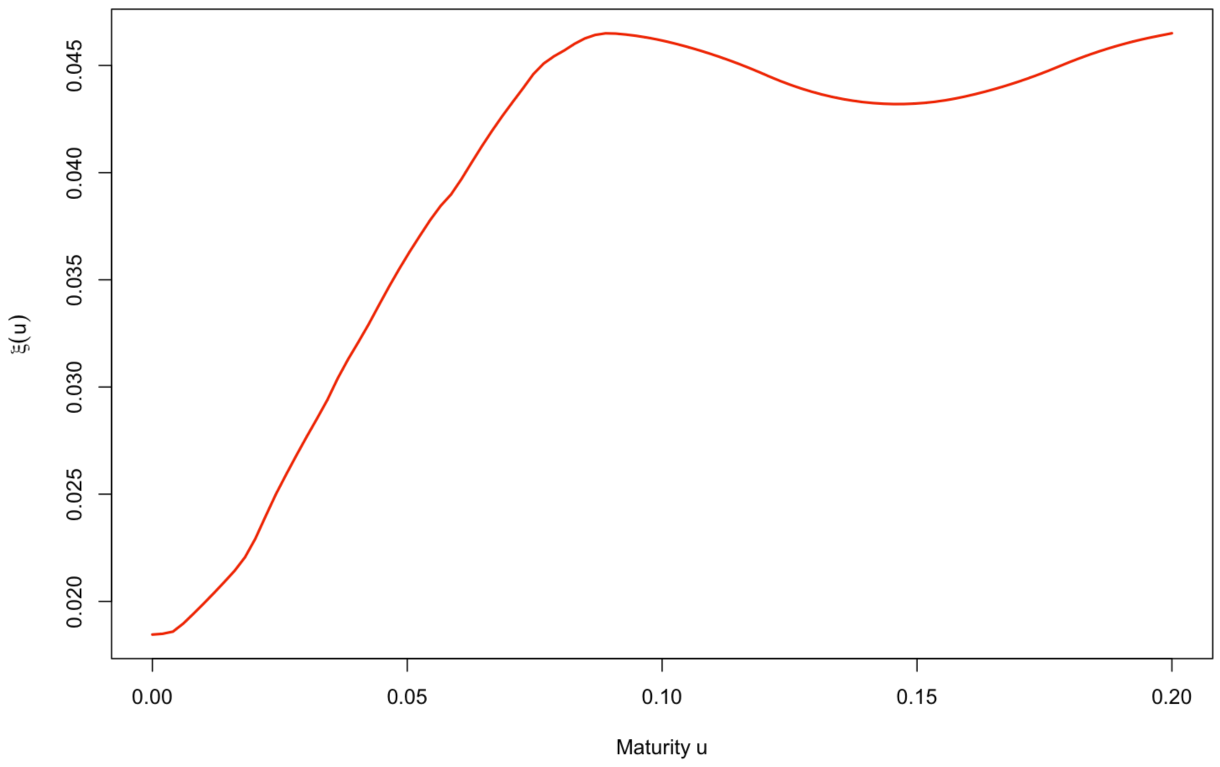


Figure 3: The forward variance curve as of 15-Feb-2023.

$y_t(u)$ from $\xi_t(u)$

- Given the kernel, model parameters, we can get $y)t(u)$.
- Recall that

$$y_t(u)^2 = \xi_t(u) - \underline{v} - \int_t^u \xi_t(s)\,\kappa(u-s)^2\,ds.$$

- We perform the integration using Gauss-Jacobi quadrature.

$y_t(u)$ from $\xi_t(u)$ code

In [13]:
```
yFromXi
```

```
function (params, xi)
{
    al <- params$al
    H <- al - 1/2
    nu <- params$nu
    lam <- params$lam
    gj <- gauss.quad(n = 10, kind = "jacobi", alpha = 2 * H -
        1, beta = 0)
    y.raw <- function(u) {
        f <- function(x) {
            prefactor <- (u/2)^(2 * H) * nu^2/gamma(al)^2
            prefactor * xi(u/2 * (1 + x)) * exp(-lam * u * (1 -
                x))
        }
        int.u <- ifelse(u > 0, sum(gj$weights * f(gj$nodes)),
            0)
        yu.2 <- xi(u) - params$c - int.u
        return(sqrt(pmax(yu.2, 0)))
    }
    return(Vectorize(y.raw))
}
```

Plot $y_t(u)$

In [14]:
```
ey <- yFromXi(params0, xi)


curve(ey,
  from = 0, to = .2, n = 100, col = "green4", lwd = 2,
  xlab = "Maturity u", ylab = expression(y(u))
)
```

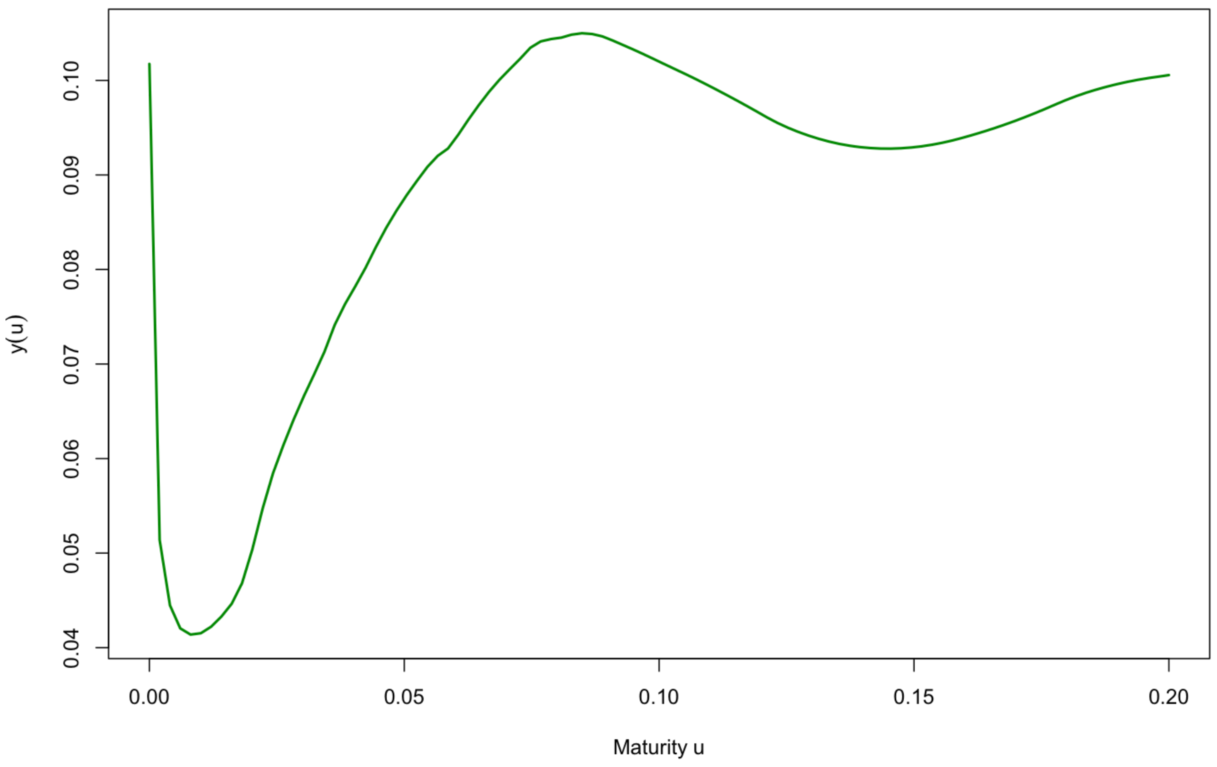


Figure 4: $y_t(u)$ ('the forward variance curve') as of 15-Feb-2023.

$\xi_t(u)$ from $y_t(u)$

- Recall that

$$\xi_t(u) = y_t(u)^2 + \int_t^u \xi_t(s)\,\kappa(u-s)^2\,ds + \underline{v}.$$

- This is a Wiener-Hopf equation for $\xi_t(u)$ whose solution may be written as

$$\xi_t(u) = y_t(u)^2 + \underline{v} + \int_t^u K(u-s)\;\left[y_t(s)^2 + \underline{v}\right]\;ds.$$

  where K is the *resolvent kernel* of κ^2.

Solving the Wiener-Hopf integral equation

- We write the original equation in convolution notation as:

$$\xi = y^2 + (\xi \star \kappa^2) + \underline{v}.$$

- The Laplace transform of a convolution is the product of the Laplace transforms. Thus

$$\mathcal{L}[\xi] = \mathcal{L}[y^2 + \underline{v}] + \mathcal{L}[\xi]\,\mathcal{L}[\kappa^2].$$

- Rearranging gives

$$\mathcal{L}[\xi]\;\left(1 - \mathcal{L}[\kappa^2]\right) = \mathcal{L}[y^2 + \underline{v}].$$

- So,

$$\mathcal{L}[\xi] = = \mathcal{L}[y^2 + \underline{v}] + \mathcal{L}[K]\,\mathcal{L}[y^2 + \underline{v}],$$

  where the Laplace transform of the resolvent kernel K is given by

$$\mathcal{L}[K] = \frac{\mathcal{L}[\kappa^2]}{1 - \mathcal{L}[\kappa^2]}.$$

- Inverting the Laplace transforms gives

  \xi_t(u) = y_t(u)^2+\uv+ \int_{t}^u\,K(u-s)\,\left[y_t(s)^2+\uv\right]\,ds.
  \label{eq:xiu2}

  as required.

The resolvent of the gamma kernel squared

- The Laplace transform of κ^2 is given by

$$\mathcal{L}[\kappa^2](p) = \frac{\eta^2\,\Gamma(2H)}{(2\lambda + p)^{2H}} =: \frac{\hat{\nu}^2}{(2\lambda + p)^{2H}},$$

- Thus

$$\mathcal{L}[K](p) = \frac{\mathcal{L}[\kappa^2](p)}{1 - \mathcal{L}[\kappa^2](p)} = \frac{\hat{\nu}^2}{(2\lambda + p)^{2H} - \hat{\nu}^2}.$$

- This turns out to be the Laplace transform of

$$K(\tau) = \hat{\nu}^2\, e^{-2\lambda\tau}\, \tau^{2H-1}\, E_{2H,2H}(\hat{\nu}^2\, \tau^{2H}),$$

  where $E_{\bullet,\bullet}$ is the two-parameter Mittag-Leffler function.

- Thus for the gamma kernel, we have a closed-form expression for the resolvent kernel K.

Dynamics of forward variance

- Recall that

  \xi_t(u) = y_t(u)^2+\uv+ \int_{t}^u\,K(u-s)\,\left[y_t(s)^2+\uv\right]\,ds.
  \label{eq:xiu2}

- Applying Itô's formula and using that $\xi_\cdot(u)$ is a martingale,

$$\begin{aligned} d\xi_t(u) &= 2\,y_t(u)\,dy_t(u) + 2\int_t^u y_t(s)\,dy_t(s)\,K(u-s)\,ds \\ &= 2\left\{y_t(u)\,\kappa(u-t) + \int_t^u y_t(s)\,\kappa(s-t)\,K(u-s)\,ds\right\}\sqrt{V_t}\,dW_t \\ &= -2\left\{\kappa(u-t)\,y_t(u) + \int_t^u \kappa(s-t)\,K(u-s)\,y_t(s)\,ds\right\}\frac{dS_t}{S_t}. \end{aligned} \tag{6}$$

- If $y_t(u) > 0$, dS_t and $d\xi_t(u)$ are perfectly anti-correlated in the QR Heston model.
  - However, sometimes $y_t(u)$ is negative for short-dated u.

  - This implies in particular that shorter dated smiles should be slighly more symmetric, as pbserved for example in [Rømer][14].

A microstructural foundation for the QRH model

- We now show how the QRH model can be obtained as the diffusion limit of a nnonlinear Hawkes process.
  - Just as rough Heston can be obtained as the diffusion limit of a linear Hawkes process.
  - Or more generally, AFV models can be obtained as diffusion limits of AFI models.

A Hawkes process

We adopt the same Hawkes model of order flow as in Lecture 3 ([AFV][10]) , but make it quadratic.

- J_t^+ (buys) and J_t^- (sells) are marked jump processes with (possibly different) mark laws ζ_+ and ζ_-.

- Just as in the AFI case, the log-price process is given in terms of the $J^\pm$ as

$$dX_t = -\lambda_t m_X dt + dJ_t^+ - dJ_t^-,$$

  where m_X is determined by the martingale condition on $S = e^X$.

Modeling V_t

- In order to complete the definition of the Hawkes order flow model, we need to specify the dynamics of λ_t or equivalently V_t.

- We would like a jump model whose diffusion limit is the quadratic rough Heston model (1).

The quadratic Hawkes order flow (QHO) model

- Motivated by the form of the QRH V process, we write

$$V_t = v_J\,\lambda_t \ = c + Y_t^2, \tag{7}$$

  where, with $d\tilde J_t^Y = \gamma_+\,d\tilde J_t^+ + \gamma_-\,d\tilde J_t^-$, and $\gamma^\pm$ constant,

```
\begin{align}
Y_t &= \bar Y + \int_{-\infty}^t\,\kappa_H(t-s)  \,d\tilde J^Y_s\notag\\
&= \bar Y + \left(\kappa_H\star  \tilde J^Y\right)_t.
\label{eq:Yu}
\end{align}
```

- Thus, when a trade arrives, X and λ (or V) jump simultaneously.

- We refer to this model as the QHO (quadratic Hawkes order flow) model.

Diffusion limit of the QHO model

- In complete analogy with the AFI case, with

$$\frac{d}{dt}\langle Z, W\rangle_t =: \rho = \frac{1}{\sqrt{v_+ + v_-}}\,\frac{\gamma_+\,v_+ - \gamma_-\,v_-}{\sqrt{{\gamma_+}^2\,v_+ + {\gamma_-}^2\,v_-}}.$$

  and

$$\eta^2 = \frac{{\gamma_+}^2\, v_+ + {\gamma_-}^2\, v_-}{v_+ + v_-},$$

we obtain

> **The limiting model**
>
> $$\begin{aligned} dX_t &= -\tfrac{1}{2}\, V_t\, dt + \sqrt{V_t}\, dZ_t, \\ V_t &= \underline{v} + \left(\bar{Y} + \int_{-\infty}^t \kappa(t-s)\, \sqrt{V_s}\, dW_s \right)^2, \end{aligned} \tag{8}$$
>
> with $d\langle Z, W \rangle_t = \rho\, dt$, and $\kappa = \eta\, \kappa_H$.

- This is a generalization of the QRH model with $\rho > -1$ in general.

Related models

- The QHO model is related to the ZHawkes model of [Blanc et al.][1] and the AFI model of [AFV][10].
- The diffusion limit of the QHO model is the quadratic rough Heston model of [Rømer][14].
- When $\gamma_- = \gamma > 0$ and $\gamma_+ = -\gamma$,

$$\rho = -1; \qquad \eta = \gamma,$$

  and we retrieve the quadratic rough Heston model of [QRH][9] and [Bourgey and Gatheral][4].

  - In this special case, $d\tilde{J}^Y_t = -\eta\, \left(d\tilde{J}^+_t - d\tilde{J}^-_t \right)$. So every time X jumps up, Y jumps down, and vice versa.

The QRH simulation scheme

- Recall the dynamics of $y_\bullet(u)$:

$$dy_t(u) = \kappa(u-t)\, \sqrt{V_t}\, dW_t.$$

- The dynamics of y are formally identical to those of ξ in an affine forward variance (AFV) model!
- Thus, we may use the HQE scheme of [Effiicient simulation][7] to simulate y.

Simulation of Y

- Following the derivation of the HQE algorithm closely, we have that

$$Y_T = y_T(T) = y_t(T) + \int_t^T \kappa(T-s)\,\sqrt{V_s}\,dW_s.$$

- Let $t = 0.$% and the time step $\Delta = T/N$ where N is the number of steps.
- Then, with more-or-less obvious notation,

$$Y_n = y_n + \sum_{k=1}^{n} \int_{(k-1)\Delta}^{k\Delta} \kappa(n\Delta - s)\,\sqrt{V_s}\,dW_s =: \hat{y}_n + u_n,$$

- The $\mathcal{F}_{n-1}$-adapted variable $\hat{y}_n$ is given by

$$\hat{y}_n := \mathbb{E}\left[Y_n \middle| \mathcal{F}_{n-1}\right] = y_n + \sum_{k=1}^{n-1} \int_{(k-1)\Delta}^{k\Delta} \kappa(n\Delta - s)\,\sqrt{V_s}\,dW_s,$$

- The martingale increment u_n is given by

$$u_n := \int_{(n-1)\Delta}^{n\Delta} \kappa(n\Delta - s)\,\sqrt{V_s}\,dW_s.$$

The X-process

We also need to simulate the nth increment of the component of the log-stock price process $X = \log S$:

$$\chi_n := \int_{(n-1)\Delta}^{n\Delta} \sqrt{V_s}\,dW_s.$$

Notation

- For convenience, define for every $i, j \in \mathbb{N}$,

$$\mathcal{K}_i(\Delta) := \int_0^\Delta \kappa(s + i\Delta)\,ds$$
$$\mathcal{K}_{i,j}(\Delta) := \int_0^\Delta \kappa(s + i\Delta)\kappa(s + j\Delta)\,ds.$$

- For the gamma kernel,

$$\mathcal{K}_i(\Delta) = \frac{\nu}{\lambda^{H+1/2}\Gamma(\alpha)}\left[\Gamma(H+1/2, i\lambda\Delta) - \Gamma(H+1/2, (i+1)\lambda\Delta)\right],$$
$$\mathcal{K}_{i,i}(\Delta) = \frac{\nu^2}{(2\lambda)^{2H}\Gamma(\alpha)^2}\left[\Gamma(2H, 2i\lambda\Delta) - \Gamma(2H, 2(i+1)\lambda\Delta)\right],$$

where $\Gamma(s,x) = \int_x^\infty t^{s-1}e^{-t}\,dt$ is the upper incomplete gamma function.

Code for $\mathcal{K}_0$ and $\mathcal{K}_{jj}$

In [15]:
```
K0
```

```
function (params)
{
    Vectorize(function(tau) {
        al <- params$al
        lam <- params$lam
        nu <- params$nu
        prefactor <- (nu/gamma(al))/(lam^al)
        bkt <- gamma(al) - gamma_inc(al, lam * tau)
        res2 <- (nu/gamma(al))/al * tau^(al)
        res <- ifelse(lam > 0, prefactor * bkt, res2)
        return(res)
    })
}
```

In [16]:
```
Kjj
```

```
function (params)
{
    Vectorize(function(j, tau) {
        kGamma2 <- function(s) {
            (kGamma(params)(s + j * tau))^2
        }
        res <- integrate(kGamma2, lower = 0, upper = tau)$value
        return(res)
    }, vectorize.args = c("j", "tau"))
}
```

Computation of $\mathrm{var}\left[\left. u_n \right| \mathcal{F}_{n-1}\right]$

The conditional variance may be approximated by

$$\begin{aligned}\mathrm{var}\left[\left. u_n \right| \mathcal{F}_{n-1}\right] &= \mathrm{var}\left[\left. Y_n \right| \mathcal{F}_{n-1}\right] \\ &\approx \left[\theta\,\hat{y}_n^2 + (1-\theta)\,Y_{n-1}^2 + c\right]\,K_0(\Delta) \\ &=: \bar{V}_n\,\mathcal{K}_{0,0}(\Delta),\end{aligned}$$

with $\theta := \frac{1}{2H+1}$ and

$$K_0(\Delta) := \int_0^\Delta K(s)\,ds, \tag{9}$$

where K is the resolvent kernel of κ^2.

Simulation of u_n and χ_n

- As in [Effiicient simulation][7], we make the hybrid approximation $u_n \approx \beta_{u\chi}\,\chi_n + \varepsilon_n$, where $\beta_{u\chi} = \mathcal{K}_0(\Delta)/\Delta$.
- We simulate χ_n and ϵ_n as \beas \chi_n \sim \sigma_\chi,Z_\chi; \quad \epsilon_n \sim \sigma_\epsilon,Z_\epsilon, \eeas where Z_χ and Z_ϵ are two independent standard normal random variables, and

$$\sigma_\chi^2 = \bar V_n\,\Delta; \quad \sigma_\epsilon^2 = \bar V_n\,\left[\mathcal{K}_{0,0}(\Delta) - \mathcal{K}_0(\Delta)^2/\Delta\right],$$

- In the large n limit, $\mathrm{var}[u_n]$, $\mathrm{var}[\chi_n]$ and $\mathrm{cov}[u_n\,\chi_n]$ are matched.

Simulation of $\hat y_n$

- For any $j \in \mathbb{N}$, let $b_j^\star$ be the root mean square average of the kernel κ over the jth interval
  - T his choice ensures that quadratic variations are matched exactly.

$$b_j^\star = \sqrt{\frac{1}{\Delta}\int_{(j-1)\Delta}^{j\Delta} \kappa^2(s)\,ds} = \sqrt{\frac{1}{\Delta}\,\mathcal{K}_{j-1,j-1}(\Delta)}.$$

  - At the nth step of the algorithm, we need $\hat y_n = \mathbb{E}\left[y_n \middle| \mathcal{F}_{n-1}\right]$:

$$< p > \hat y_n \approx y_n + \sum_{k=1}^{n-1} b_{n-k+1}^\star\,\chi_k.$$

    - Note the explicit path-dependence!

Simulation of V_n and X_n

Having now simulated $\hat y_n$ and u_n, we simulate V_n and $X_n = \log(S_n)$ as

$$\begin{aligned} V_n &= Y_n^2 + c = (\hat y_n + u_n)^2 + c, \\ X_n &= X_{n-1} - \tfrac{1}{4}\,(V_n + V_{n-1})\,\Delta - \chi_n. \end{aligned}$$

- At this point, we have simulated X and its quadratic variation, so we can, in principle, price all SPX derivatives, including variance swaps.

Simulation of VIX

- Let $\delta = 30/365$ (30 days).
- Given $y_T(u)$, $u > T$, we can show that, under quadratic rough Heston,

$$\mathrm{VIX}_\mathrm{T}^2 = \frac{1}{\delta}\int_0^\delta \left(y_T(T+u)^2 + c\right)\,\{1 + K_0(\delta - u)\}\;du, \tag{3}$$

  where $K_0(\tau) = \int_0^\tau K(s)\,ds$.

Simulation of the $y_T(u)$

For any $u > T$,

$$\begin{aligned} y_T(u) &= y(u) + \int_0^T \kappa(u-s)\,\sqrt{V_s}\,dW_s && (10)\\ &\approx y(u) + \sum_{k=1}^m a_k(u)\int_{(k-1)\Delta}^{k\Delta} \sqrt{V_s}\;dW_s && (11)\\ &= y(u) + \sum_{k=1}^m a_k(u)\,\chi_k, && (12) \end{aligned}$$

where the $a_k(u)$ are chosen so as to match the variances of the $y_T(u)$.

Final VIX simulation recipe

- Plugging into (3) gives

$$
\begin{aligned}
\mathrm{VIX}_{\mathrm{T}}^2 &= \frac{1}{\delta}\int_0^\delta \left(y_T(T+u)^2+c\right)\,\left(1+K_0(\delta-u)\right)\,du\\
&= \frac{1}{\delta}\int_0^\delta \left[\left(y(T+u)+\sum_{k=1}^n a_k(T+u)\,\chi_k\right)^2+c\right]\\
&\qquad\qquad\qquad\times\left[1+K_0(\delta-u)\right]du.
\end{aligned}
$$

- A numerical integral that may be easily computed.
  - For example, using the trapezoidal rule or a Gauss--Legendre quadrature.

QRH simulation code

In [17]:
```
QRH.sim
```

```
function (params, xi)
{
    function(paths, steps, expiries, output = "all", delvix = 1/12,
        nvix = 10) {
        library(gsl)
        c <- params$c
        H <- params$al - 1/2
        Z.eps <- matrix(rnorm(steps * paths), nrow = steps, ncol = paths)
        Z.chi <- matrix(rnorm(steps * paths), nrow = steps, ncol = paths)
        v0 <- xi(0)
        ey <- yFromXi(params0, xi)
        bigK0p <- bigK0(params)
        K00p <- K00(params)
        K0p <- K0(params)
        sim <- function(expiry) {
            dt <- expiry/steps
            K0del <- K0p(dt)
            K00del <- K00p(dt)
            bigK0del <- bigK0p(dt)
            tj <- (1:steps) * dt
            yj <- ey(tj)
            K00j <- c(0, K00p(tj))
            bstar <- sqrt(diff(K00j)/dt)
            chi <- array(0, dim = c(steps, paths))
            v <- rep(v0, paths)
            Y <- rep(ey(0), paths)
            yhat <- rep(yj[1], paths)
            rho.uchi <- K0del/sqrt(K00del * dt)
            beta.uchi <- K0del/dt
            X <- numeric(paths)
            w <- numeric(paths)
            for (j in 1:steps) {
                alp <- 1/(2 * H + 1)
                varu <- bigK0del * (alp * yhat^2 + (1 - alp) *
                  Y^2 + c)
                vbar <- varu/K00del
                sig.chi <- sqrt(vbar * dt)
                sig.eps <- sqrt(vbar * K00del * (1 - rho.uchi^2))
                chi[j, ] <- sig.chi * Z.chi[j, ]
                eps <- sig.eps * Z.eps[j, ]
                u <- beta.uchi * chi[j, ] + eps
                Y <- yhat + u
                vf <- Y^2 + c
                dw <- (v + vf)/2 * dt
                w <- w + dw
                X <- X - dw/2 - chi[j, ]
                btilde <- rev(bstar[2:(j + 1)])
                if (j < steps) {
                  yhat <- yj[j + 1] + as.numeric(btilde %*% chi[1:j,
                    ])
                }
                v <- vf
            }
            vix2 <- 0
            ds <- delvix/nvix
            if ((output == "vix") || (output == "all")) {
                for (k in 1:nvix) {
                  tk <- expiry + k * ds
                  Ku <- c(K00p(tk), K00p(tk - tj))
                  ck.vec <- sqrt(-diff(Ku)/dt)
                  dyTu <- as.numeric(ck.vec %*% chi[1:steps,
                    ])
                  yTu <- ey(tk) + dyTu
                  vix2 <- vix2 + (yTu^2 + c) * (1 + bigK0p((nvix -
                    k) * ds))/nvix
                }
                vix2 <- vix2 + v * (1 + bigK0p(delvix))/(2 *
                  nvix) - (yTu^2 + c)/(2 * nvix)
                vix <- sqrt(vix2)
            }
            res.sim <- switch(output, v = v, X = X, w = w, vix = vix,
                all = list(v = v, X = X, w = w, vix = vix))
            return(res.sim)
        }
        sim.out <- NULL
        if (output != "all") {
            sim.out <- t(sapply(expiries, sim))
        }
        else {
            for (j in 1:length(expiries)) {
                sim.out[[j]] <- sim(expiries[j])
            }
        }
        return(sim.out)
    }
}
```

Load SPX and VIX implied volatility data

In [18]:
```
load("spx_vix_vols_20230215.rData")
```

Choose expiries

In [19]:
```
(spxExpiries <- c(7, 14, 20, 28) / 365)
```

```
0.0191780821917808 · 0.0383561643835616 · 0.0547945205479452 · 0.0767123287671233
```

In this particular case, we choose identical SPX and VIX expiries.

In [20]:
```
(vixExpiries <- spxExpiries)
```

```
0.0191780821917808 · 0.0383561643835616 · 0.0547945205479452 · 0.0767123287671233
```

Run the HQR simulation

In [21]:
```
paths <- 1e5
steps <- 100
```

The list `spx.all` has all variables of interest for each expiry.

In [22]:
```
system.time(
  qrh.all <- QRH.sim(params0, xi)(
    paths, steps, nvix = 10, expiries = spxExpiries, output = "all"
  )
)
```

```
   user  system elapsed 
  9.396   0.755  10.156 
```

In [23]:
```
n.all <- length(qrh.all)
```

Extract log-SPX values from list `spx.all`

In [24]:
```
qrh.spx <- matrix(nrow = n.all, ncol = paths)
for (j in 1:n.all) {
  qrh.spx[j, ] <- qrh.all[[j]]$X
}
```

In [25]:
```
spxVolDataExtract <- spxIvols20230215[round(spxIvols20230215$Texp, 3) %in% round(spxExpiries,
```

Plot SPX smiles with QRH fits

In [26]:
```
options(repr.plot.width = 13, repr.plot.height = 8, repr.plot.res = 150)


plot.spx <- plotIvolsMC(spxVolDataExtract, mcMatrix = exp(qrh.spx), colnum = 2)
```

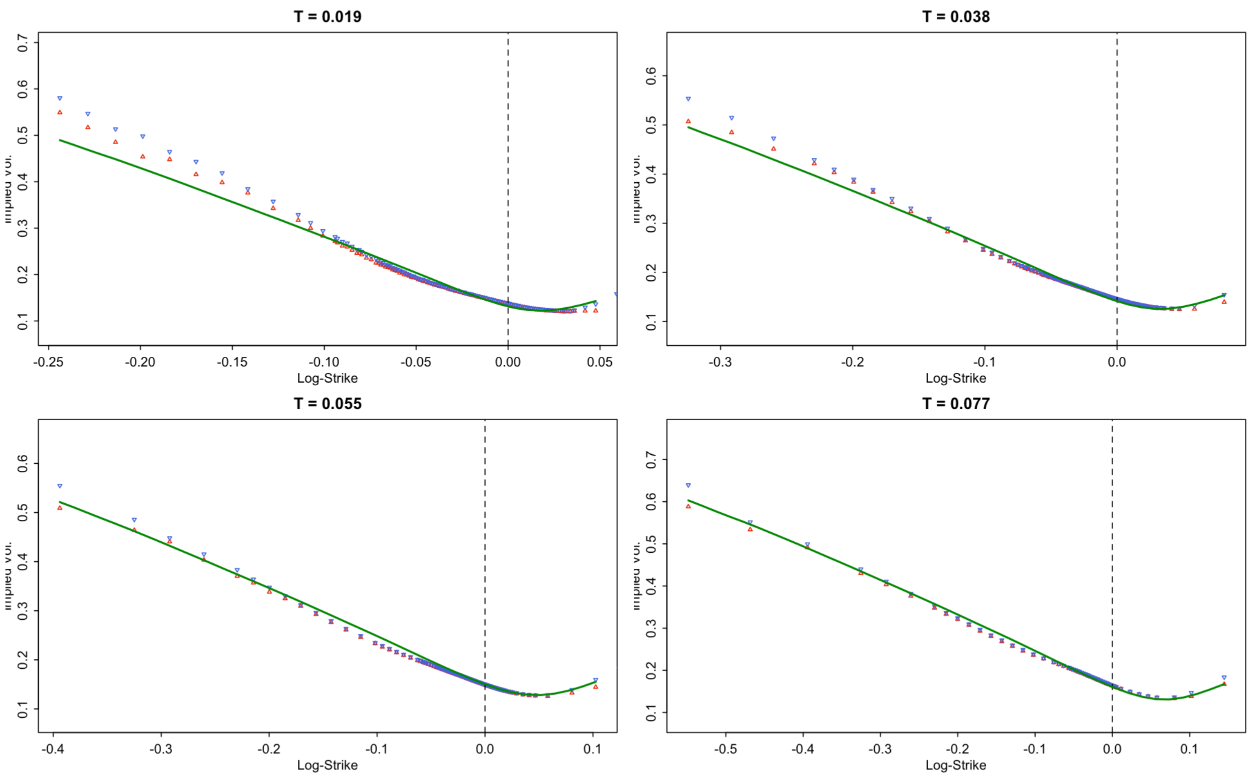


Figure 5: SPX smiles fits

VIX smiles

Extract VIX values from list `spx.all`

In [27]:
```
qrh.vix <- matrix(nrow = n.all, ncol = paths)
for (j in 1:n.all) {
  qrh.vix[j, ] <- qrh.all[[j]]$vix
}
```

In [28]:
```
vixVolDataExtract <- vixIvols20230215[vixIvols20230215$Texp < .09, ]
```

In [29]:
```
plot.vix <- plotIvolsMC(vixVolDataExtract, colnum = 2, mcMatrix = qrh.vix)
```

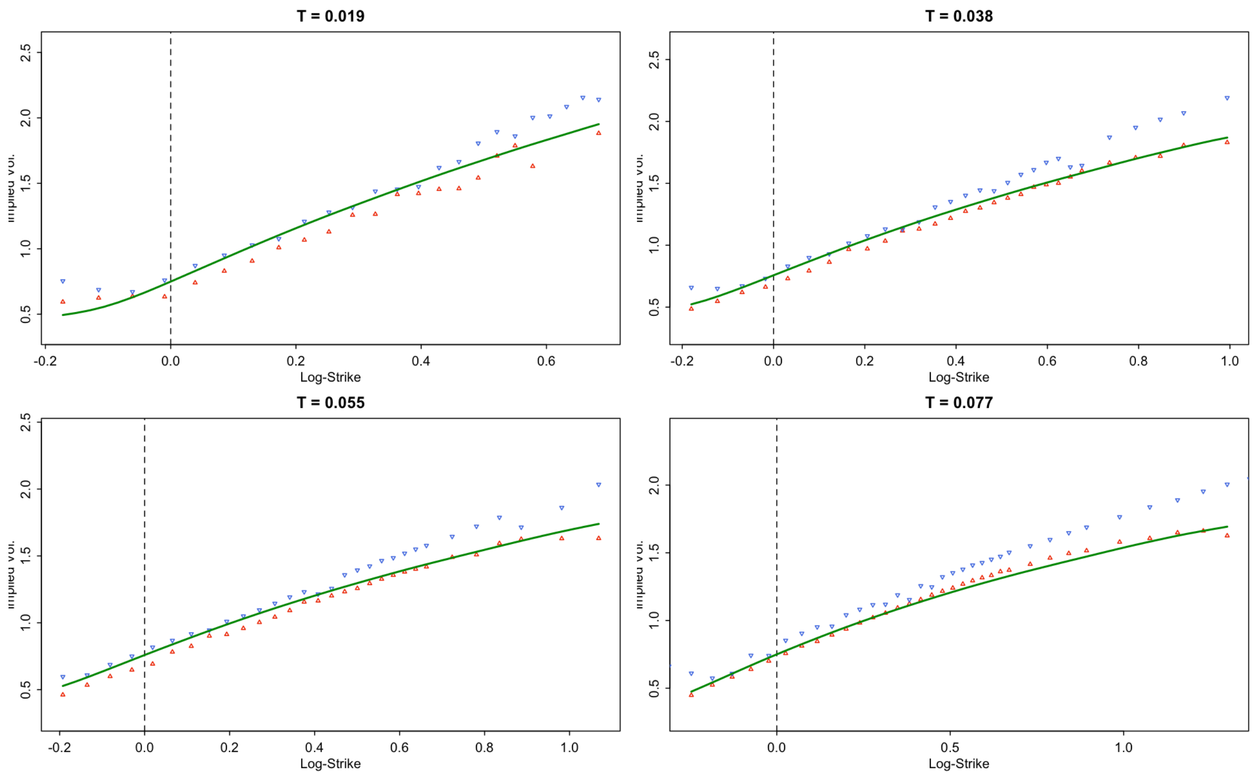


Figure 6: VIX smiles fits

Options on VIX: An example

- We consider a 1 month option on VIX.
- We use our 15-February 2023 parameters.
- We select 5 Monte Carlo paths and plot the forward variance curves as of time $T = 1m$ corresponding to these paths.
- Recall that

$$VIX_T^2 = \frac{1}{\delta} \int_T^{T+\delta} \xi_T(u)\, du.$$

  - Valuation depends on the forward variance curve between $T = 1m$ and $T = 2m$.
- We show this graphically in Figure 7.

Options on VIX: Graphical illustration of dependence on $\xi_T(u)$

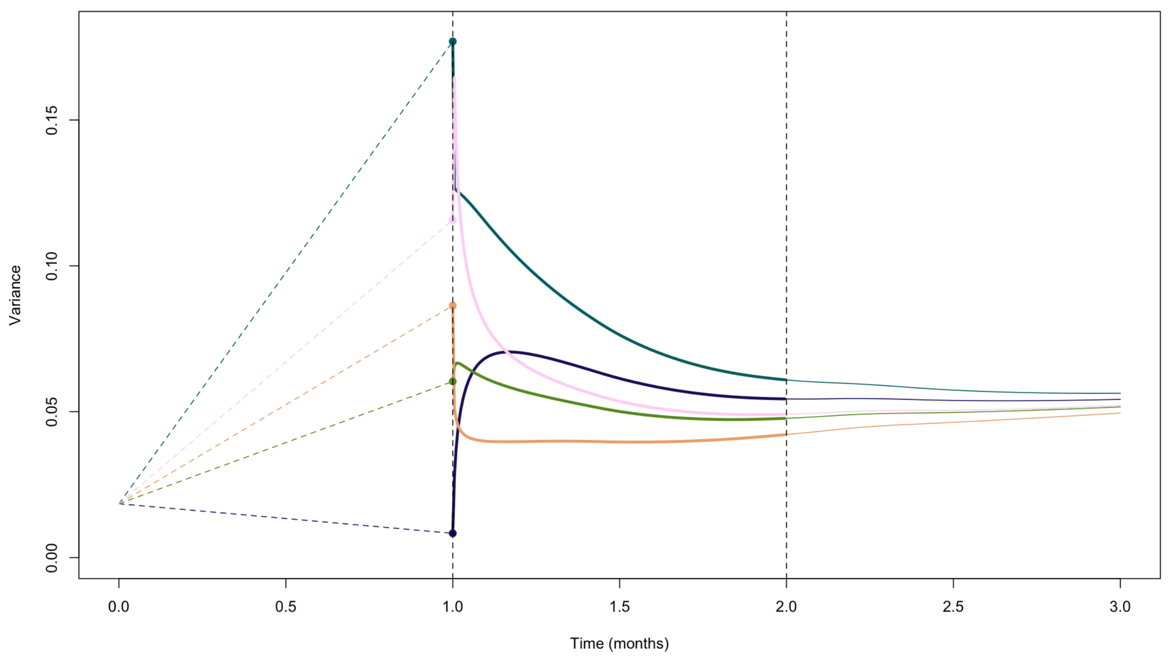


Figure 7: Five forward variance curves $\xi_t(u)$ contributing to the valuation of a 1-month option on VIX.

Path-dependence of the QR Heston model

- From (3), changes in the forward variance curve are directly expressible in terms of changes in the underlying.
  - The QR Heston model is purely path-dependent.
- Explicitly, using $dy_t(u) = \kappa(u-t)\,\sqrt{V_t}\,dW_t$, we may write

$$y_t(u) = \bar{Y} + \int_{-\infty}^{t} \kappa(u-r)\,\sqrt{V_r}\,dW_r = \bar{Y} - \int_{-\infty}^{t} \kappa(u-r)\,\frac{dS_r}{S_r}.$$

  - The forward volatility curve is thus a weighted average of historical stock returns.
- The fair values of SPX options, VIX options, and the SSR are all functionals of the forward volatility curve.

Examples of path-dependence

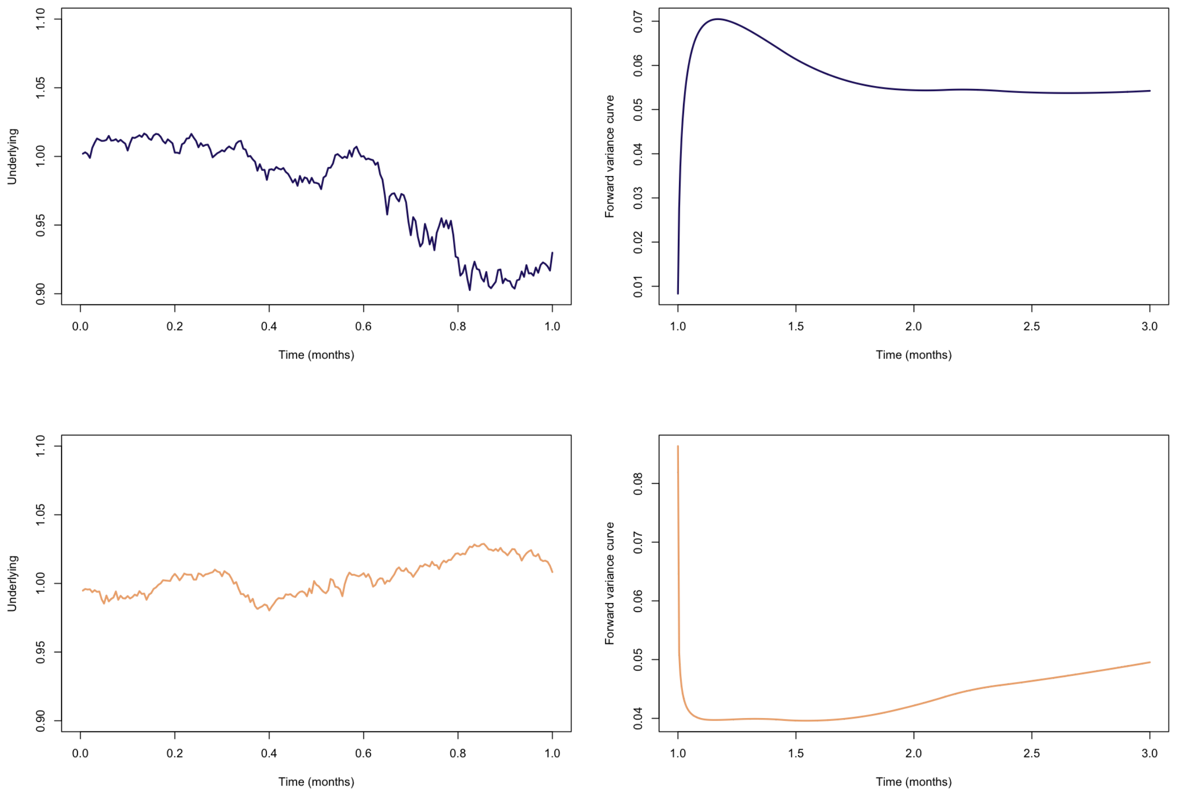


Figure 8: Two price paths and corresponding forward variance curves. With our fitted parameters, the forward variance curve is most sensitive to the very recent past.

The skew-stickiness ratio (SSR)

- Recall that the SSR is given by

$$\mathcal{R}_t(T) = \frac{\beta_t(T)}{\mathcal{S}_t(T)}$$

  where $\beta_t(T)$ is the regression coefficient in

$$\delta\sigma_t(T) = \beta_t(T)\,\delta X_t + \text{noise},$$

  where $\sigma_t(T)$ is ATM vol. and $\mathcal{S}_t(T)$ is the ATM skew.

- In practice whole smiles are tied to the ATM vol. So this is a recipe for real-time updating of the whole vol. surface.

Estimation of the regression coefficient $\beta_t(T)$ under QRH

- Recall that in martingale form, the QR Heston model reads:

$$dy_t(u) = \kappa(u-t)\,\sqrt{V_t}\,dW_t.$$

- We apply the finite difference methodology of [Bourgey et al.][2].
- Thus, the regression coefficient is estimated as

$$\beta_t(T) \approx \frac{1}{h}\left\{\sigma_t\left(T, y_t^h(\cdot)\right) - \sigma_t\left(T, y_t(\cdot)\right)\right\},$$

  where h is a small parameter, and for $t \leq u \leq T$,

$$y_t^h(u) := y_t(u) - h\,\kappa(u-t). \tag{13}$$

Blipped forward volatility curves

- Experiment shows that scaling the blip by time to expiration works well.
- Specifically, we choose $h = -0.01\sqrt{T}$.
- $h < 0$ ensures a positive blipped forward variance curve.

Run `QRH.blip` in parallel

Note that Figure 3 in the paper has more expiries, 300,000 paths, and 300 time steps. We have reduced the computational burden here for illustration.

In [30]:
```r
paths <- 100000
steps <- 100
h <- -.01
expiries <- 2^(((-13):(0)) / 2)
```

In [31]:
```r
t0 <- proc.time()

# Number of iterations
iters <- max(1, floor(paths / 10000))

# Setup parallel backend to use all processors
(cl.num <- detectCores()) # Display number of processors on your machine
cl <- makeCluster(cl.num)
registerDoParallel(cl)

# Load packages on all workers
invisible({
  clusterEvalQ(cl, {
    library(statmod)
    library(MittagLefflerR)
  })
})

# Loop
ls <- foreach(icount(iters)) %dopar% {
  QRH.blip(params0, xi, h)(paths = 10000, steps = steps, expiries = expiries)
}

stopCluster(cl)
mcMatrix <- do.call(cbind, ls) # Bind all of the submatrices into one big matrix

print(proc.time() - t0)
```

```
14
   user  system elapsed 
  0.171   0.111   9.016 
```

Compute base and blipped implied volatilities

First we construct a matrix of S_T values from the output of `QRH.blip`.

In [32]:
```
nn <- length(expiries)

ST <- matrix(nrow = nn, ncol = paths)
ST.h <- matrix(nrow = nn, ncol = paths)

for (j in 1:nn) {
  ST[j, ] <- exp(-mcMatrix[4 + 8 * (j - 1), ] - mcMatrix[2 + 8 * (j - 1), ] / 2)
  ST.h[j, ] <- exp(-mcMatrix[8 + 8 * (j - 1), ] - mcMatrix[6 + 8 * (j - 1), ] / 2)
}
```

Illustrate for one expiration

In [33]:
```
kk <- seq(-.4, .2, .01)

vol.kk <- bsOut(ST[9, ], expiries[9], exp(kk))$BSV
vol.h.kk <- bsOut(ST.h[9, ], expiries[9], exp(kk))$BSV


plot(kk, vol.kk, pch = 20, col = bl, lwd=1.5,type = "l", xlab = "Log-strike k", ylab = "ATM vo
lines(kk, vol.h.kk, col = rd,lwd=1.5)
```

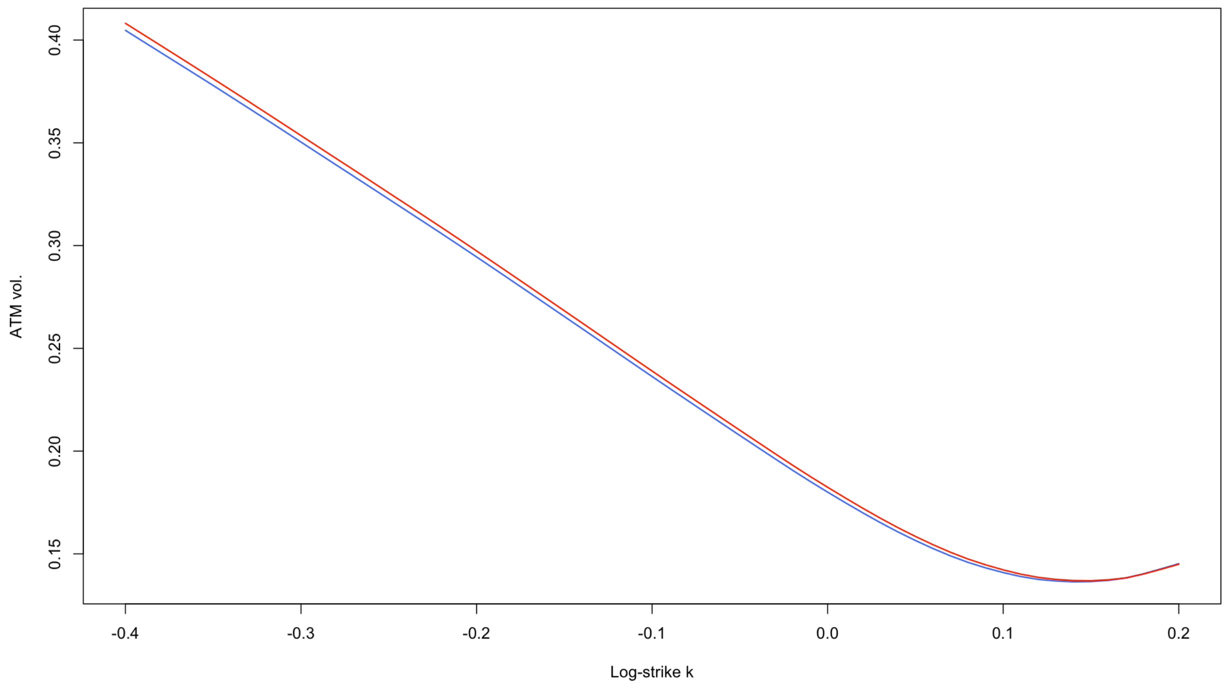


Figure 9: Base smile in blue; blipped smile in red.

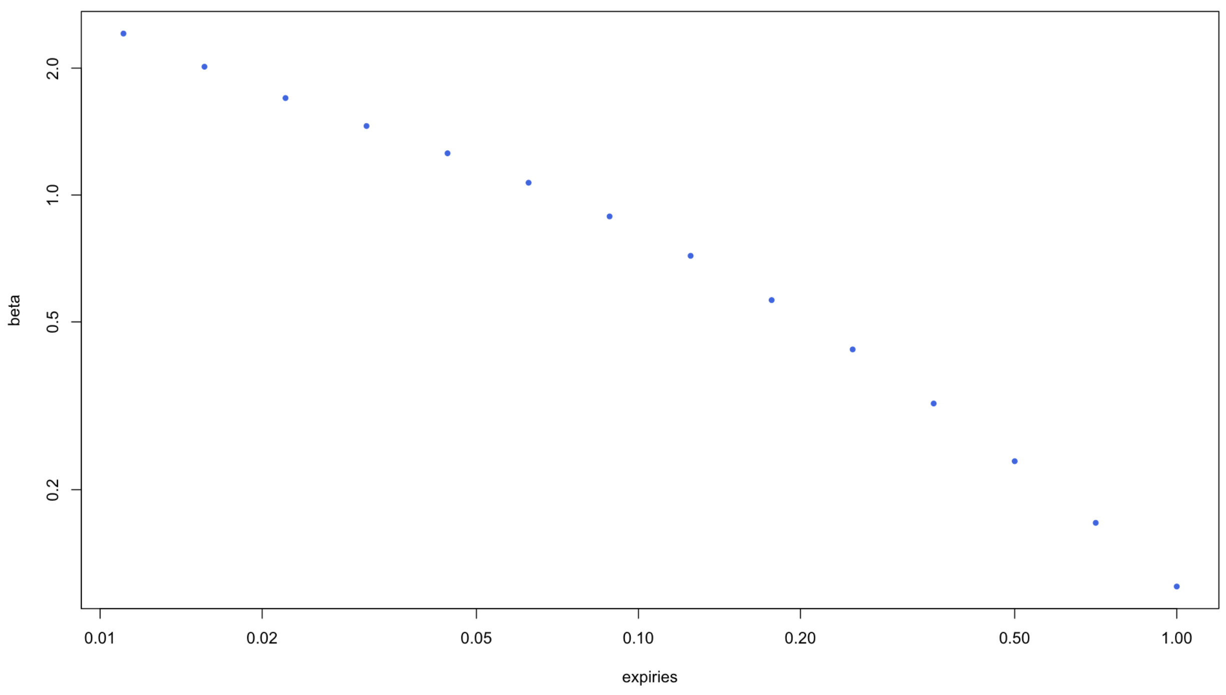
Computation of the regression coefficient β

In [34]:
```
beta <- numeric(nn)

for (j in 1:nn) {
  beta[j] <- (bsOut(ST.h[j, ], expiries[j], 1)$BSV - bsOut(ST[j, ], expiries[j], 1)$BSV) / .0
}
```

Log-log plot of $\beta_t(T)$

In [35]:
```
plot(expiries, beta, pch = 20, col = bl, log = "xy")
```



Estimation of the ATM skew $\mathcal{S}_t(T)$

- Recall that the ATM skew

$$\mathcal{S}_t(T) = \left.\frac{\partial \sigma_{\text{BS}}(k,T)}{\partial k}\right|_{k=0}.$$

- We estimate the ATM skew using a trick from [Bourgey, Friz et al.][3]: for any $k \in \mathbb{R}$,

$$\frac{\partial \sigma_{\text{BS}}(k,T)}{\partial k} = \left.\frac{\Phi\left(-\frac{k}{\sqrt{w}} - \frac{\sqrt{w}}{2}\right) - \mathbb{P}\left(X_T \geq k\right)}{\sqrt{T}\phi\left(-\frac{k}{\sqrt{w}} - \frac{\sqrt{w}}{2}\right)}\right|_{w=T\sigma_{\text{BS}}^2(k,T)}$$

where $X_T = \log(\frac{S_T}{S_0})$ and ϕ is the p.d.f.~of a standard normal distribution.

Code to compute the ATM skew $\mathcal{S}_t(T)$

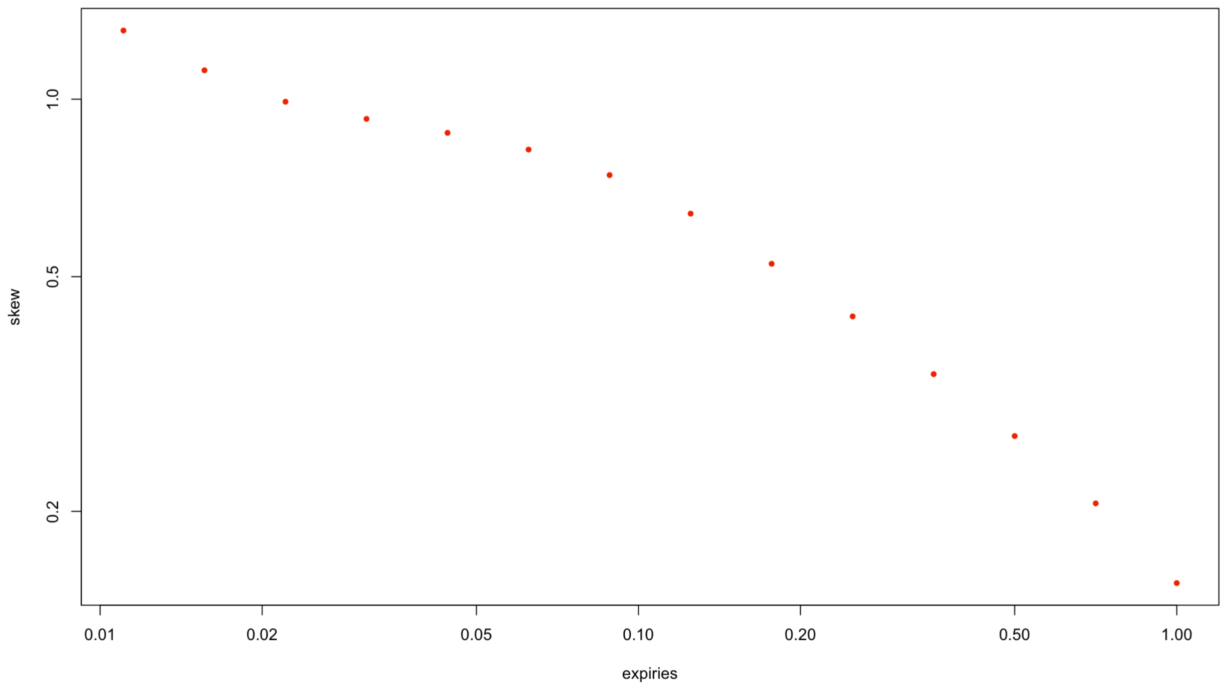
In [36]:
```
pr <- numeric(nn)
skew <- numeric(nn)
pr.h <- numeric(nn)
skew.h <- numeric(nn)
```

In [37]:
```
for (j in 1:nn) {
  Xj <- mcMatrix[4 + 8 * (j - 1), ] - mcMatrix[2 + 8 * (j - 1), ] / 2
  pr[j] <- mean(Xj >= 0)
  Xj <- mcMatrix[8 + 8 * (j - 1), ] - mcMatrix[2 + 8 * (j - 1), ] / 2
  pr.h[j] <- mean(Xj >= 0)
}
```

In [38]:
```
for (j in 1:nn) {
  wj <- mean(mcMatrix[2 + 8 * (j - 1), ])
  Phij <- pnorm(-sqrt(wj) / 2)
  phij <- dnorm(-sqrt(wj) / 2)
  skew[j] <- (Phij - pr[j]) / sqrt(expiries[j]) / phij
  wj <- mean(mcMatrix[6 + 8 * (j - 1), ])
  Phij <- pnorm(-sqrt(wj) / 2)
  phij <- dnorm(-sqrt(wj) / 2)
  skew.h[j] <- (Phij - pr[j]) / sqrt(expiries[j]) / phij
}
```

Log-log plot of $\mathcal{S}_t(T)$

In [39]:
```
plot(expiries, skew, pch = 20, col = rd, log = "xy")
```



Estimation of the SSR

- Finally, the SSR is estimated as

$$\mathcal{R}_t(T) \approx \frac{\beta_t(T)}{\mathcal{S}_t(T)}. \tag{4}$$

- The estimator (4) only requires only the ATM implied volatilities for two different forward volatility curves $y_t(u)$, and one estimate of the ATM skew.

In [40]:
```
(ssr <- beta / skew)
```

1.84719190631819 · 1.80002296728499 · 1.71505174143957 · 1.57425911928033 · 1.43218809699522 · 1.30188700794479 · 1.19658308907814 · 1.12159361774969 · 1.07106002031001 · 1.00481917238011 · 0.936988245029928 · 0.870669493155682 · 0.808800682046376 · 0.779867593301061

SSR term structure as of 15 February 2023

In [41]:
```
plot(c(0, expiries), c(params0$al + 1, ssr),
  pch = 20, col = gr, type = "b", lwd = 2, cex = .5,
  ylab = "SSR", xlab = "Expiry T"
)
    grid( col="darkblue", lty = "dotted")
```

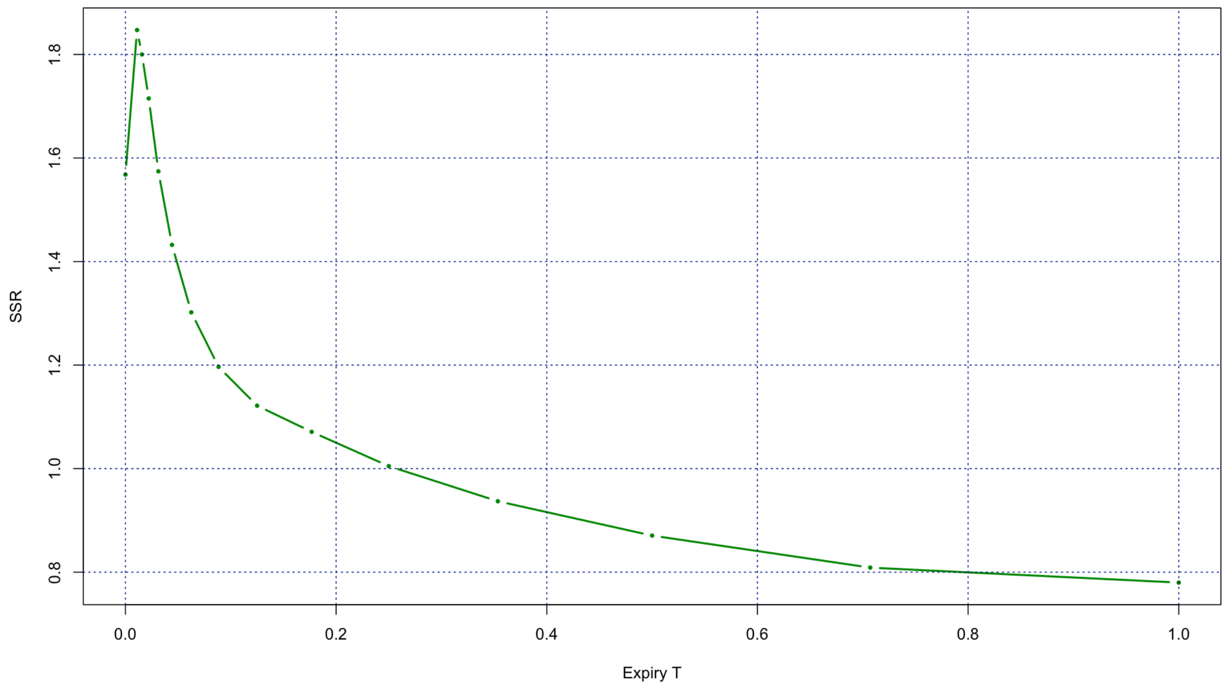


Figure 10. The SSR as a function of maturity τ.

Add historical estimates of SSR as of 15-Feb-2023

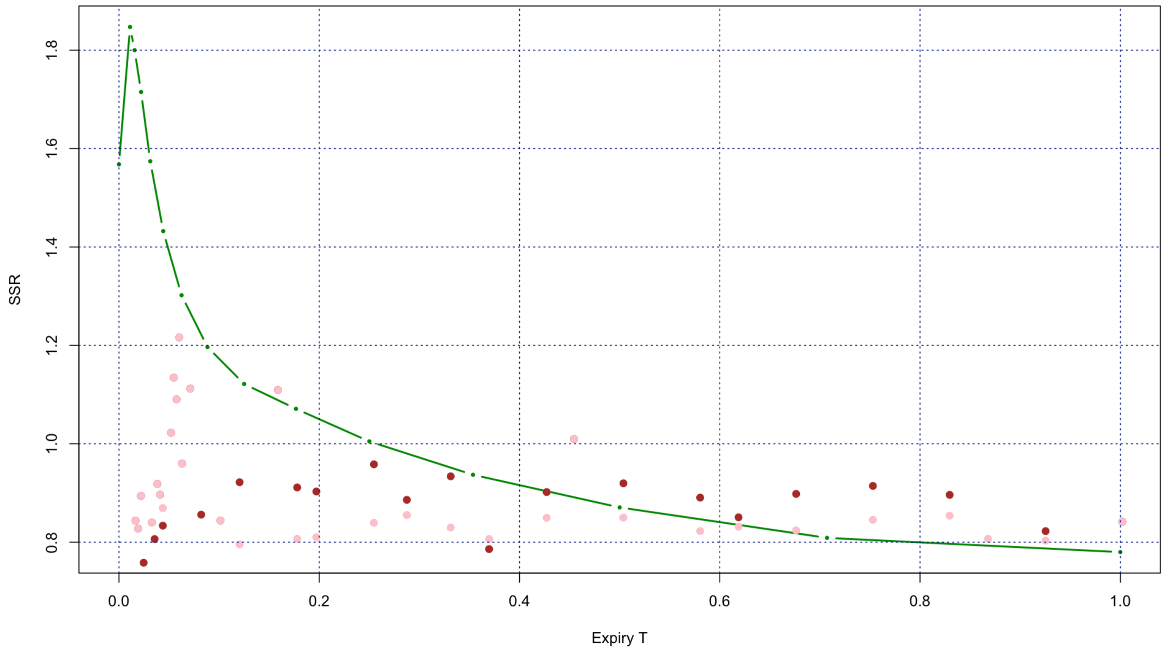


Figure 11. The SSR as a function of maturity τ with historical estimates of the SSR added; 21 day average in brown; 60 day average in pink.

Remarks on the fits

- Despite the impressive joint fits to short-dated SPX and VIX smiles, fits deteriorate for longer periods.
- The very short-time behavior of the SSR shown in Figure 10, where the SSR increases sharply before declining, is consistent with the general results derived in [Friz and Gatheral][6].
  - And consistent with empirical estimates of the SSR.
- $c > 0$ is really necessary.
  - Fits with $\underline{v} = 0$ are much less impressive.

Why does the QR Heston model fit so well?

- In view of its extreme parsimony, it may at first seem surprising that the QR Heston model can generate such impressive fits.
- By taking a historical perspective and relating the QR Heston model to earlier models, we may begin to understand its strong fitting performance.

QR Heston from Quadratic Hawkes

- The QR Heston model in (1) is a special case of the quadratic rough Heston model of [Dandapani et al.][5].
- This model, in turn, emerges as the continuous-time limit of a non-Markovian extension of the quadratic Hawkes model originally proposed in [Blanc et al.][4].
- It, in turn, can be viewed as a limit of the QARCH (quadratic ARCH) model.
  - Calibration of QARCH to 5-minute intra-day US stock yielded very impressive results.

- The QUARCH model has been used in practice for a long time, in particular by Bouchaud at CFM.
- Thus the QR Heston model can be seen as a non-Markovian extension of a long-standing successful statistical model of financial markets.

Markovian QR Heston

- The QR Heston model can also be seen as the non-Markovian extension of various realistic models in the classical stochastic volatility literature.
- Consider the QR Heston model in the Markovian case ($H = 1/2$) with the gamma kernel (2):

$$\kappa(\tau) = \nu\, e^{-\lambda\, \tau},$$

and $\underline{v} = 0$.

- In this case, $\sigma_t = Y_t$ is the instantaneous volatility, and $Y_t = \bar{Y} + \int_{-\infty}^{t} \kappa(t-s)\, \sqrt{V_s}\, dW_s$ may be rewritten as

$$d\sigma_t = -\lambda\, (\sigma_t - \bar{Y})\, dt + \nu\, \sigma_t\, dW_t, \tag{5}$$

QR Heston and the Inverse Gamma model

- (5) are the dynamics of volatility in the Inverse Gamma model of [Langrené et al.][13].
- As noted in that paper, the Inverse Gamma model is a special case of various other models that have desirable properties consistent with empirical studies of stocks and their volatilities.
- For example, the Inverse Gamma model has an inverse gamma stable distribution, which was shown to be a good fit to the unconditional distribution of volatility by Bouchaud and Potters and Artur Sepp amongst others.
  - In fact, this observation was one of the reasons that Sepp chose his particular model dynamics.

QR Heston and the Double Lognormal model

- In terms of $V_t = \sigma_t^2 = {Y_t}^2$, (5) may be written in the form

$$\begin{aligned} dV_t &= -\hat{\lambda}\, \left(V_t - {V_t}'\right)\, dt + 2\,\nu\, V_t\, dW_t, \\ d{V_t}' &= -\lambda\, \left({V_t}' - \bar{V}'\right)\, dt + \nu\, {V_t}'\, dW_t, \end{aligned}$$

where

$$\hat{\lambda} = 2\,\lambda - \nu^2; \qquad {V_t}' = \frac{2\lambda}{\hat{\lambda}}\, \bar{Y}\, \sigma_t; \qquad \bar{V}' = \frac{2\lambda}{\hat{\lambda}}\, \bar{Y}^2.$$

- This is a special case of the Double Lognormal (DL) model of [Bachelier 2008][8]
  - DL generates positively sloping VIX smiles
  - DL parameters are reasonable stable.

Lifting the Double Lognormal model}

- Viewing the gamma kernel as an infinite sum of exponential kernels with different characteristic timescales in the spirit of Abi Jaber, the QR Heston model may be seen as an infinite sum of Double

Lognormal models.
- Accordingly, the QR Heston model inherits various properties of the Double Lognormal model such as positively sloped VIX smiles.

Summary

- The Quadratic Rough Heston (QR) Heston model can successfully capture the joint behavior of SPX and VIX implied volatility smiles, whilst producing realistic values for the SSR.
  - Thus statically and dynamically reasonable.
- It may be derived as the limit of a quadratic Hawkes proces describing order flow.
- The QR Heston model with the gamma kernel is:
  - Extremely parsimonious with only four parameters -Purely path-dependent
- The QRH scheme is efficient and easy to implement.
  - The QR Heston model is thus a viable candidate for practical trading applications.

References

1. ^ Pierre Blanc, Jonathan Donier, and Jean-Philippe Bouchaud, uadratic Hawkes processes for financial prices, *Quantitative Finance* **17**(2), 171-188 (2017).
2. ^ Florian Bourgey, Jules Delemotte, and Stefano De Marco, Smile dynamics and rough volatility, *SSRN* 4911186} (2024).
3. ^ Florian Bourgey, Stefano De Marco, Peter K Friz, and Paolo Pigato, Local volatility under rough volatility, *Mathematical Finance* **33**(4) 1119--1145 (2023).
4. ^ Florian Bourgey and Jim Gatheral, The SSR under Quadratic Rough Heston, *SSRN* 4911186 (2025).
5. ^ Aditi Dandapani, Paul Jusselin, and Mathieu Rosenbaum, From quadratic Hawkes processes to super-Heston rough volatility models with Zumbach effect, *Quantitative Finance* **21**(8) 1235-1247 (2021).
6. ^ Peter Friz and Jim Gatheral, Computing the SSR, *Quantitative Finance* **25**(5) 701-710 (2025).
7. ^ Jim Gatheral, Efficient Simulation of Affine Forward Variance Models, *Risk.net*, *SSRN* 3876680, February (2022).
8. ^ Jim Gatheral, Consistent modeling of SPX and VIX options, *Bachelier Congress*, London (2008).
9. ^ Jim Gatheral, Paul Jusselin and Mathieu Rosenbaum, The quadratic rough Heston model and the joint S&P 500/VIX smile calibration problem, *Risk.net*, (May 2020).
10. ^ Jim Gatheral and Martin Keller-Ressel, Affine forward variance models, *Finance and Stochastics*, **23**(3) 501-533 (2019).
11. ^ Jim Gatheral, Thibault Jaisson and Mathieu Rosenbaum, Volatility is rough, *Quantitative Finance* **18**(6), 933-949 (2018).
12. ^ Thibault Jaisson and Mathieu Rosenbaum, Rough fractional diffusions as scaling limits of nearly unstable heavy tailed Hawkes processes, *The Annals of Applied Probability*, **26**(5) 2860-2882 (2016).
13. ^ Nicolas Langrené, Geoffrey Lee, and Zili Zhu, Switching to non-affine stochastic volatility: A closed-form expansion for the inverse gamma model, *International Journal of Theoretical and Applied Finance* **19**(5) 1650031 (2016).
14. ^ Sigurd Emil Rømer, Empirical Analysis of Rough and Classical Stochastic Volatility Models to the SPX and VIX Markets, *Quantitative Finance* **22**(10) 1805-1838 (2022).