Using the *schwartz97* package

David Lüthi, Philipp Erb & Simon Otziger

February 10, 2014

Abstract

The purpose of this document is to show how the R package *schwartz97* can be used. This is done by numerous examples and intuitive explanations.

1 Introduction

The package *schwartz97* provides a set of functions to work with the two-factor model of Gibson and Schwartz (1990)[1]. The two-factor model describes the joint dynamics of the state variables *spot price* and *spot convenience yield* (later simply called *convenience yield*).

We believe that the value of this package primarily lies in the parameter fitting routine `fit.schwartz2f`. Once the parameters of the two-factor model are estimated, a number of functions can be used to, e.g., draw samples from the model, price derivatives as European options, or filter a futures price series to get an estimate of the underlying state variables. The document is organized as follows: Section 2 briefly describes the model and section 3 gives an overview of the classes and functions. Then, section 4 to 8 give examples and a case study.

2 The Schwartz Two-Factor Model

The spot price of the commodity and the instantaneous convenience yield are assumed to follow the joint stochastic process:

$$dS_t = (\mu - \delta_t)S_t dt + \sigma_S S_t dW_S \quad (1)$$

$$d\delta_t = \kappa(\alpha - \delta_t)dt + \sigma_\epsilon dW_\epsilon, \quad (2)$$

with Brownian motions W_S and W_ϵ under the objective measure $\mathbb{P}$ and correlation $dW_S dW_\epsilon = \rho dt$. Under the pricing measure $\mathbb{Q}$ the dynamics are

$$dS_t = (r - \delta_t)S_t dt + \sigma_S S_t d\widetilde{W}_S \quad (3)$$

$$d\delta_t = [\kappa(\alpha - \delta_t) - \lambda]dt + \sigma_\epsilon d\widetilde{W}_\epsilon, \quad (4)$$

[1] Because the model was extended in Schwartz (1997) and Miltersen and Schwartz (1998) we call it the *Schwartz two-factor model* hereafter.

where the constant λ denotes the market price of convenience yield risk and $\widetilde{W}_S$ and $\widetilde{W}_\epsilon$ are $\mathbb{Q}$-Brownian motions. It may be handy to introduce a new mean-level for the convenience yield process under $\mathbb{Q}$

$$\tilde{\alpha} = \alpha - \lambda/\kappa. \tag{5}$$

The dynamics is then

$$d\delta_t = \kappa(\tilde{\alpha} - \delta_t)dt + \sigma_\epsilon d\widetilde{W}_\epsilon. \tag{6}$$

For more information on the model and pricing formulas we refer to the other package vignette *Technical Document*, or to Schwartz (1997) and Hilliard and Reis (1998).

3 Package Overview

This section gives an overview of the functions and classes contained in the package *schwartz97*. In addition, the object-oriented programming approach followed in this package is explained.

3.1 Functions

The core of the package *schwartz97* is built by the following functions[2]:

R-function	Description
`dstate`	Density of the bivariate state vector.
`pstate`	Distribution of the bivariate state vector.
`qstate`	Quantile of the bivariate state vector.
`rstate`	Sample from the state distribution at some future time.
`simstate`	Generate trajectories from the bivariate state vector.
`dfutures`	Density of the futures price.
`pfutures`	Distribution of the futures price.
`qfutures`	Quantile of the futures price.
`rfutures`	Sample from the futures price distribution.
`pricefutures`	Calculate the futures price.
`priceoption`	Calculate the price of European call or put options.
`filter.schwartz2f`	Filter a futures price series.
`fit.schwartz2f`	Fit the two-factor model to data.

Except the function `fit.schwartz2f` all the above functions are set to generic and accept three different signatures (see section 3.3).

[2]There are also a number of utility functions as *coef*, *mean*, *vcov*, *plot*, *resid*, and *fitted*.

3.2 Classes

The package *schwartz97* provides the class `schwartz2f`. This class contains all parameters which are needed to define the dynamics of the state variables *spot price* and *convenience yield* under the objective measure $\mathbb{P}$. The class `schwartz2f` has the following slots:

Slot name	Class	Symbol	Description
s0	numeric	s_0	Initial spot price.
delta0	numeric	δ_0	Initial convenience yield.
mu	numeric	μ	Drift parameter of the spot price.
sigmaS	numeric	σ_S	Diffusion parameter of the spot price.
kappaE	numeric	κ	Speed of mean-reversion of the convenience yield.
alpha	numeric	α	Mean-level of the convenience yield.
sigmaE	numeric	σ_ϵ	Diffusion parameter of the convenience yield.
rhoSE	numeric	ρ	Correlation between the two Brownian motions.
call	call		The function call.

The above set of parameters contains the symbols appearing in (1) and (2) as well as the initial values s_0 and δ_0. To create an object of class `schwartz2f` the constructor with the same name can be used (see section 4).

The function `fit.schwartz2f`, which estimates parameters of the two-factor model, returns an object of class `schwartz2f.fit`. This class inherits from the class `schwartz2f` and adds the following slots.

Slot name	Class	Symbol	Description
r	numeric	r	Risk-free interest rate.
alphaT	numeric	$\tilde{\alpha}$	Mean-value of the convenience yield under $\mathbb{Q}$.
lambda	numeric	λ	Market price of convenience yield risk.
deltat	numeric		Time-increment of the transition equation.
n.iter	numeric		Number of iterations.
llh	numeric		Log-likelihood value.
converged	logical		States whether the fit converged or not.
error.code	numeric		An error code or 0.
error.message	character		Contains the error message if any.
fitted.params	logical		States which parameters were fitted.
trace.pars	matrix		Parameter evolution during the estimation.
meas.sd	numeric		Standard deviation of the measurement equation.

These slots together with the ones contained in the class `schwartz2f` fully determine the dynamics of the model under both, the objective measure and the pricing measure. Notice that one of the parameters `lambda` and `alphaT` is redundant according to equation 5.

3.3 Object Orientation

As mentioned earlier most of the functions dealing with the state variables and futures prices are set to generic. The idea is to leave some freedom to the user, who can decide whether he wants to use an object-oriented approach or provide a fairly large set of arguments for each function-call.

Consider the function `dfutures` for example. The function headers for different signatures are:

```
## S4 method for signature 'ANY,ANY,ANY,numeric':
dfutures(x, time = 0.1, ttm = 1, s0 = 50, delta0 = 0,
         mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0,
         sigmaE = 0.5, rho = 0.75, r = 0.05, lambda = 0,
         alphaT = NULL, measure = c("P", "Q"), ...)

## S4 method for signature 'ANY,ANY,ANY,schwartz2f':
dfutures(x, time = 0.1, ttm = 1, s0, r = 0.05,
         lambda = 0, alphaT = NULL, measure = c("P", "Q"), ...)

## S4 method for signature 'ANY,ANY,ANY,schwartz2f.fit':
dfutures(x, time = 0.1, ttm = 1, s0, measure = c("P", "Q"), ...)
```

Without object-orientation (first header) the function has 15 arguments. Ten parameters are needed to describe the dynamics under both measures.

If a `schwartz2f.fit` object is provided for `s0` the only additional arguments required are `x` (quantiles), `time` (time where the futures process is evaluated), and `ttm` (time to maturity of the futures contract).

4 Object Initialization

A `schwartz2f` object with reasonable parameters is constructed in the following code chunk.

```
> s0 <- 100
> delta0 <- 0
> mu <- 0.1
> sigmaS <- 0.2
> kappa <- 1
> alpha <- 0.1
> sigmaE <- 0.3
> rho <- 0.4
> obj <- schwartz2f(s0 = s0, delta0 = delta0, alpha = alpha,
+                   mu = mu, sigmaS = sigmaS, sigmaE = sigmaE,
+                   rho = rho, kappa = kappa)
> obj
```

```
------------------------------------------------------------
Schwartz97 two-factor model:

SDE
d S_t     = S_t   * (mu - delta_t)    * dt + S_t * sigmaS * dW_1
d delta_t = kappa * (alpha - delta_t) * dt + sigmaE * dW_2
E(dW_1 * dW_2) = rho * dt

Parameters
s0    : 100
delta0: 0
mu    : 0.1
sigmaS: 0.2
kappa : 1
alpha : 0.1
sigmaE: 0.3
rho   : 0.4
------------------------------------------------------------
```

Objects of class `schwartz2f.fit` are constructed via the function `fit.schwartz2f` (see section 8).

5 Working with the state variables

As soon as a `schwartz2f` object is initialized, it can be passed to the functions `dstate`, `pstate`, `qstate`, `rstate`, and `simstate`. The distribution of the state variables depend on the horizon. Once this point in time is defined the above functions can be used like the standard R distribution functions for, e.g., the normal distribution (`dnorm`, `pnorm`, `qnorm`, `rnorm`).

In this example a sample of the spot price and the convenience yield in five years is generated by the function `rstate`. Then, the probability that the spot price is below 150 and the convenience yield is lower than 0 in five years is computed. The mean of the state variables in one and ten years is calculated next. Finally, trajectories of the state variables are plotted (see fig. 1).

```
> time <- 5
> sample.t <- rstate(n = 2000, time, obj)
> pstate(c(0, -Inf), c(150, 0), time, obj)

[1] 0.2243732
attr(,"error")
[1] 1e-15
attr(,"msg")
[1] "Normal Completion"
```

```
> mean(obj, time = c(1, 10))

          s.t    delta.t
[1,] 106.3906 0.06321206
[2,] 130.5386 0.09999546

> plot(obj, n = 30, time = 5, dt = 1 / 52)
```

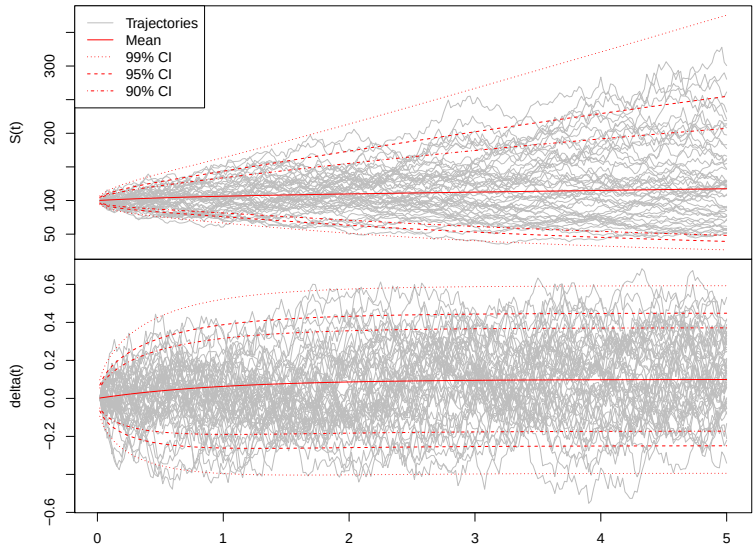

Figure 1: Thirty trajectories of the state variables are plotted on a weekly interval and a five years horizon. The initial values of the state variables are 100 for the spot price (s_0) and 0 for the convenience yield (δ_0). The spot price has a drift μ of 10% and a volatility σ_S of 20%. The *speed of mean-reversion* parameter κ of the convenience yield process is 1, and the long-term mean (α) is 10%. The volatility of the convenience yield σ_ϵ is 30% and the correlation ρ between the Brownian motions driving the state variables is 40%.

6 Working with derivatives

In this example we calculate some futures prices and plot the dynamics of the term structure ("forward curve"). In addition prices of European options are computed.

We work through this section by looking at corn and assuming all the parameters are known. The current price (s_0) of 1000 bushels of corn is assumed to be 80 USD. The convenience yield (δ_0) is zero at the moment

but it's long-term mean (α) is 5%. The drift (μ) of corn is 10% and the volatility is 30%. The speed of mean-reversion of the convenience yield (κ) is 1.5 and its volatility is 40%. Correlation is assumed to be 60%. The risk-free rate is 3% and the market price of convenience yield risk (λ) is zero.

First the object is initialized. Next a trajectory is generated based on weekly sampling over five years. Then futures prices are calculated with time to maturities ranging from zero (which is the spot) to two years. Finally, a call option which matures in one year written on a futures contract with time to maturity of two years is priced. Forward curves are shown in fig. 2.

```
> s0 <- 80
> delta0 <- 0.05
> mu <- 0.1
> sigmaS <- 0.3
> kappa <- 1.5
> alpha <- 0.05
> sigmaE <- 0.4
> rho <- 0.6
> lambda <- 0.04
> r <- 0.03
> set.seed(1)
> obj <- schwartz2f(s0, delta0, mu, sigmaS, kappa, alpha, sigmaE, rho)
> state.traj <- simstate(n = 52 * time, time, obj)
> pricefutures(seq(0, 2, by = 0.4), obj, lambda = lambda, r = r)

[1] 80.00000 79.28309 78.64870 78.17279 77.81715 77.53741

> priceoption(type = "call", time = 1, Time = 2, K = 85,
+             obj, r = r, lambda = lambda)

[1] 4.991482
```

7 Contango, Backwardation, and Hump Shapes

Fig. 2 shows the ability of the Schwartz two-factor model to generate contango and backwardation situations. Mixed shapes (humps and "inverse humps") are possible too. E.g. an upwards sloping forward curve at the short end which points downwards at the long end.

Looking at the $\mathbb{Q}$-dynamics in equations (3) and (4) it is obvious that, locally, the drift of the spot price is positive when $\delta_t < r$. This corresponds to a (local) contango situation. However, the long-term mean of δ_t, $\tilde{\alpha}$, defines the shape at the far end of the term structure. Four different shapes are generated in the following example and plotted in fig. 3:

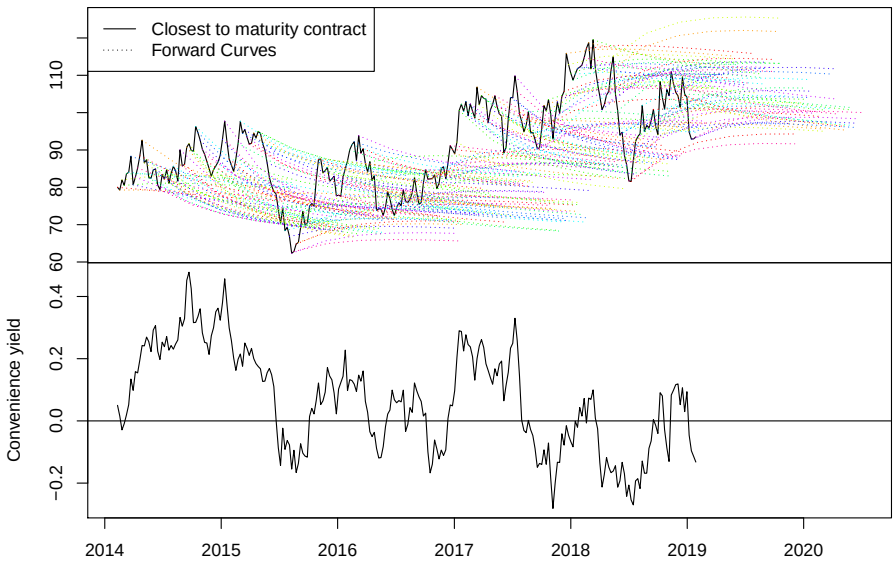

Figure 2: Forward curves with time to maturity up to two years are plotted for the trajectory *state.traj*. The closest to maturity contract is in fact the spot price because the time to maturity is zero.

Pure contango: If $\delta_t < r$ and $\tilde{\alpha} < r$.

Short end backwardation, long end contango: If $\delta_t > r$ and $\tilde{\alpha} < r$.

Pure backwardation: If $\delta_t > r$ and $\tilde{\alpha} > r$.

Short end contango, long end backwardation: If $\delta_t < r$ and $\tilde{\alpha} > r$.

```
> s0 <- 1
> delta0 <- 0.0
> sigmaS <- 0.3
> kappa <- 1
> sigmaE <- 0.4
> rho <- 0.5
> r <- 0.03
> ttm <- 0:4
> ## Pure contango
> pricefutures(ttm, s0 = s0, delta0 = 0, sigmaS = sigmaS,
+              kappa = kappa, sigmaE = sigmaE, rho = rho,
+              r = r, alphaT = 0)
```

```
[1] 1.000000 1.021605 1.054220 1.099526 1.152367

> ## Backwardation and then contango
> pricefutures(ttm, s0 = s0, delta0 = 2 * r, sigmaS = sigmaS,
+              kappa = kappa, sigmaE = sigmaE, rho = rho,
+              r = r, alphaT = 0)

[1] 1.0000000 0.9835835 1.0009214 1.0385929 1.0864517

> ## Pure backwardation
> pricefutures(ttm, s0 = s0, delta0 = r, sigmaS = sigmaS,
+              kappa = kappa, sigmaE = sigmaE, rho = rho,
+              r = r, alphaT = 2 * r)

[1] 1.0000000 0.9805302 0.9595804 0.9449587 0.9335775

> ## Contango and then backwardation
> pricefutures(ttm, s0 = s0, delta0 = -r, sigmaS = sigmaS,
+              kappa = kappa, sigmaE = sigmaE, rho = rho,
+              r = r, alphaT = 2 * r)

[1] 1.0000000 1.0184332 1.0106773 1.0003988 0.9902179
```

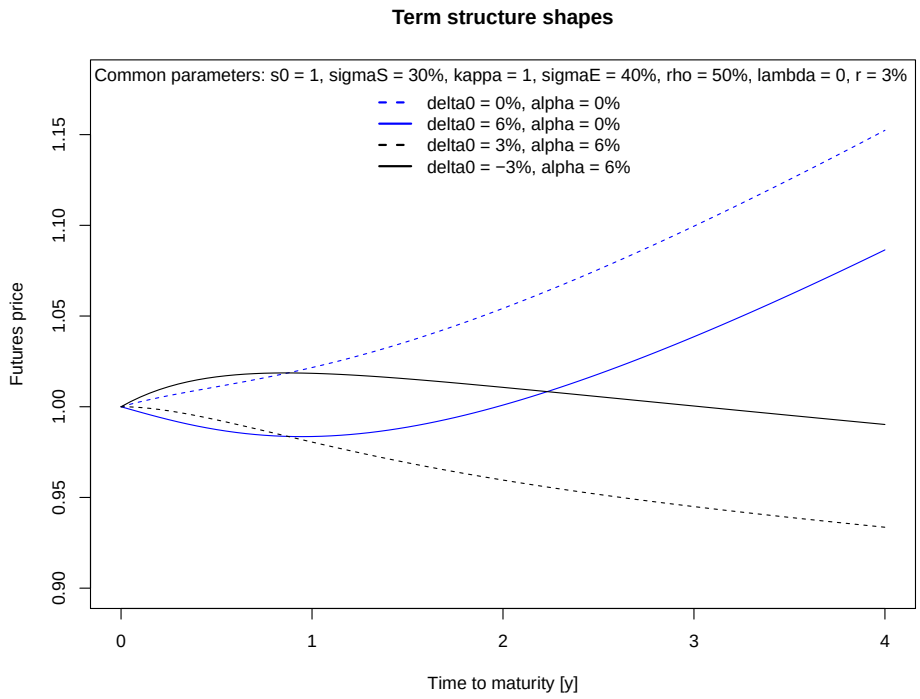

Figure 3: Four term structure shapes which can be generated by the Schwartz two-factor model. For a given risk-free interest rate r, the shape is determined by th value of the convenience yield δ_t and by the mean level of the convenience yield $\tilde{\alpha}$ under the pricing measure $\mathbb{Q}$ (see (3) and (4)).

8 Parameter Estimation

As mentioned in section 1, we believe that the package's most valuable piece of code is the function `fit.schwartz2f`. This function estimates the parameters involved in equations (1) - (4) including the initial values of the state variables s_0 and δ_0. Because log-futures prices linearly depend on a bivariate Gaussian random vector (the log-spot price and the convenience yield), it is most straightforward to use a linear state-space model. Therefore, the estimation procedure is based on the Kalman filter as proposed in Schwartz (1997).

The header of the function `fit.schwartz2f` looks like

```
>   args(fit.schwartz2f)

function (data, ttm, deltat = 1/260, s0 = data[1, 1], delta0 = 0,
    mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0, sigmaE = 0.3,
    rho = 0.7, lambda = 0, meas.sd = rep(0.1, ncol(data)), opt.pars = c(s0 = FALSE,
        delta0 = FALSE, mu = TRUE, sigmaS = TRUE, kappa = TRUE,
        alpha = TRUE, sigmaE = TRUE, rho = TRUE, lambda = FALSE),
    opt.meas.sd = c("scalar", "all", "none"), r = 0.03, silent = FALSE,
    ...)
NULL
```

The data inputs are `data` and `ttm`. `data` must be a regularly spaced time-series matrix of futures prices and `ttm` a matrix giving the time-to-maturity.

The time-to-maturity matrix admits the following interpretation: `data[i,j]` denotes the futures price whose time to maturity was `ttm[i,j]` when it was observed. The unit is defined by `deltat` which is the time between observations `data[i,j]` and `data[i+1,j]`.

The arguments from `s0` to `lambda` are initial values of the parameters.

`meas.sd` gives (initial) values of the measurement error standard deviations.

`opt.pars` states which parameters shall be estimated. Observe that some parameters are held constant by default.

`opt.meas.sd` specifies how measurement uncertainty is treated in the fit: According to the model there should be a one-to-one correspondance between the spot and the futures price. In reality, the term structure does not fully match for any set of parameters. This is reflected in the measurement uncertainty-vector `meas.sd`. All components of `meas.sd` can be fitted. However, it might be sufficient to fit only a scalar where the measurement uncertainty is parametrized by `scalar * meas.sd` (this is the default). In this case define the vector `meas.sd` and set `opt.meas.sd` to "scalar". `meas.sd` can be set to a vector with each component set to the same value, thereby giving

each point on the term structure equal weight. Another reasonable specification takes open interest or volumes into account: The higher the volume, the higher the weight and therefore the lower the corresponding component of `meas.sd`. If all components of `meas.sd` shall be fitted choose "all". If the measurement uncertainty is known set `meas.sd` to "none". Note that the measurement errors are assumed to be independent in this implementation (even though the model and the filter do not require independence). This is reflected in zero off-diagonals of the measurement error covariance matrix.

Finally, the risk-free rate `r` must be given.

8.1 Statistical and Computational Considerations

Estimation of the Schwartz two-factor model parameters is statistically fragile and computationally demanding. Multiple local maxima of the likelihood may exist which can result in absurd parameter estimates as, e.g., a yearly drift of 300% and or a market price of convenience yield risk of -200%. Therefore, a reasonable parameter estimation is most likely an iteration where several initial values are used and different combinations of parameters are held constant during estimation. Also, simulation studies showed that a fairly large sample is required to get adequate estimates (e.g. 20000 daily observations, depending on the number of parameters which shall be estimated). For this reason the default is to hold `s0`, `delta0`, and `lambda` constant.

Several utility functions as `fitted`, `resid`, `plot`, and `coef` may help to investigate the quality of the fit (see example below).

The fitting procedure generally requires a large number of iterations to achieve a reasonable tolerance level. Each optimization iteration involves the filtering of the data set by the Kalman filter. Therefore, an efficient implementation of the Kalman filter is key.

8.2 Example: Estimating Wheat Parameters

This section takes you through a "real-world" example of a Schwartz two-factor parameter estimation. There are daily observations of the five closest to maturity wheat futures prices from Jan. 1995 to April 2010 (approx. 4000 observations).

The default parameters of `fit.schwartz` are used, i.e., all parameters except the initial values of the state variables and the market price of convenience yield risk `lambda` are estimated. The maximum number of iterations is limited to 300 to save (build and check) time. Then the object is printed and the parameter evolution is plotted.

```
> data(futures)
> wheat.fit <- fit.schwartz2f(futures$wheat$price, futures$wheat$ttm / 260,
```

```
+                              deltat = 1 / 260, control = list(maxit = 300), silent =
> wheat.fit

------------------------------------------------------------------
Fitted Schwartz97 two-factor model:

SDE (P-dynamcis)
d S_t     = S_t   * (mu - delta_t) * dt    + S_t * sigmaS * dW_1
d delta_t = kappa * (alpha - delta_t) * dt + sigmaE * dW_2
E(dW_1 * dW_2) = rho * dt

SDE (Q-dynamcis)
d S_t     = S_t   * (r - delta_t) * dt      + S_t * sigmaS * dW*_1
d delta_t = kappa * (alphaT - delta_t) * dt + sigmaE * dW*_2
alphaT = alpha - lambda/kappa

Parameters
s0    : 395.5
delta0: 0
mu    : 0.250267865481006
sigmaS: 0.376161779933002
kappa : 0.0049924899685365
alpha : -0.12065663312003
sigmaE: 0.159902538320822
rho   : 0.904453926049466
r     : 0.03
lambda: 0
alphaT: -0.12065663312003
------------------------------------------------------------------
Optimization information
Converged:          FALSE
Fitted parameters:  mu, sigmaS, kappa, alpha, sigmaE, rho, meas.sd1; (Number: 7)
log-Likelihood:     -3343706937
Nbr. of iterations: 302
------------------------------------------------------------------

> plot(wheat.fit, type = "trace.pars")
```

Observe that the fit did not converge as the maximum of 300 iterations is not sufficient to achieve to required tolerance (here optim's default). Let's discuss the parameters: A μ of 25% is probably not far off and the spot price volatility σ_S of 37% seems to be fine too. The speed of mean reversion of the convenience yield process κ is alarmingly close to zero which means that the convenience yield can drift far away from it's mean. The mean level

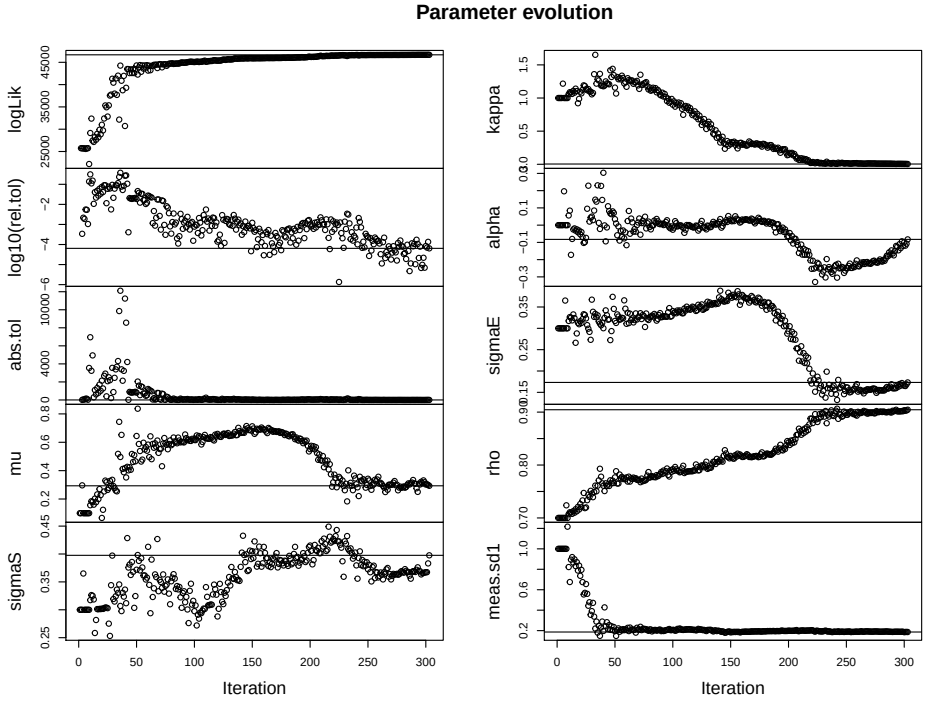

Figure 4: This figure shows the parameter evolution of the first 300 iterations of the unconstrained parameter estimation of wheat. While the relative tolerance gets below 10^{-4} after around 200 iterations the parameter values still fluctuate substantially.

of the convenience yield α of -12% is not intuitive to say the least. The correlation ρ of 90% is very high, making the parameters even harder to interpret and the process dynamic less intuitive. In addition, the parameter evolution shown in fig. 4 is concerning.

Two measures are introduced as an attempt to make parameters more appealing: κ is constrained to 1 and the measurement error standard deviations are set proportional to the average traded volumes. The average initial value of the measurement error standard deviations is set to 1%.

```
> vol.std <- colSums(futures$wheat$vol, na.rm = TRUE) / sum(futures$wheat$vol, na.rm
> wheat.fit.constr <- fit.schwartz2f(futures$wheat$price, futures$wheat$ttm / 260,
+                                    kappa = 1,
+                                    opt.pars = c(s0 = FALSE, delta0 = FALSE, mu = T
+                                      sigmaS = TRUE, kappa = FALSE, alpha = TRUE,
+                                      sigmaE = TRUE, rho = TRUE, lambda = FALSE),
+                                    meas.sd = 1 / vol.std  / sum(1 / vol.std) * len
+                                    deltat = 1 / 260, control = list(maxit = 300),
```

```
> wheat.fit.constr

------------------------------------------------------------------
Fitted Schwartz97 two-factor model:

SDE (P-dynamcis)
d S_t     = S_t   * (mu - delta_t) * dt    + S_t * sigmaS * dW_1
d delta_t = kappa * (alpha - delta_t) * dt + sigmaE * dW_2
E(dW_1 * dW_2) = rho * dt

SDE (Q-dynamcis)
d S_t     = S_t   * (r - delta_t) * dt      + S_t * sigmaS * dW*_1
d delta_t = kappa * (alphaT - delta_t) * dt + sigmaE * dW*_2
alphaT = alpha - lambda/kappa

Parameters
s0    : 395.5
delta0: 0
mu    : 0.0973917026862634
sigmaS: 0.315675278006403
kappa : 1
alpha : 0.0439034736965484
sigmaE: 0.280503762575771
rho   : 0.543979631300506
r     : 0.03
lambda: 0
alphaT: 0.0439034736965484
------------------------------------------------------------------
Optimization information
Converged:          FALSE
Fitted parameters:  mu, sigmaS, alpha, sigmaE, rho, meas.sd1; (Number: 6)
log-Likelihood:     -3927591873
Nbr. of iterations: 301
------------------------------------------------------------------

> plot(wheat.fit.constr, type = "trace.pars")
```

Parameters are more reasonable now: μ, α, and ρ seem to be fine at 10%, 4.3%, and 54%, respectively, as well as the parameter evolution presented in fig. 5. Also, as a quick check, simulated trajectories in fig. 6 look plausible. Real term structures are compared to model generated term structures in fig. 7.

```
> wheat.2007 <- lapply(futures$wheat,
+                      function(x)x[as.Date(rownames(x)) > "2007-01-01" & as.Date(ro
```

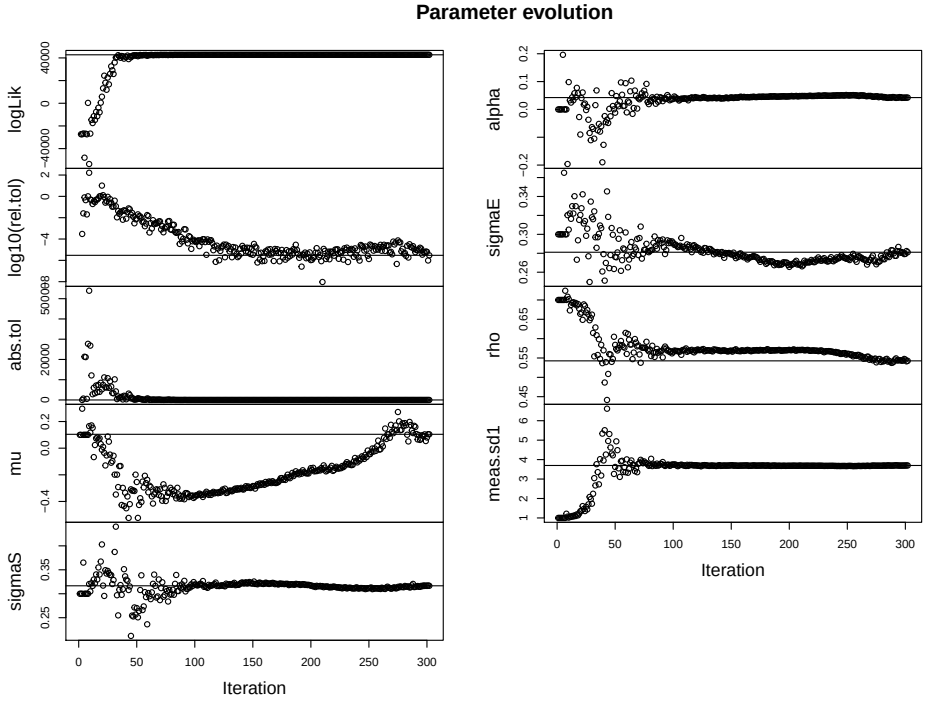

Figure 5: This figure shows the parameter evolution of the first 300 iterations of the constrained parameter estimation of wheat. The relative tolerance gets below 10^{-6} after 150 iterations and the parameter values get more and more stationary. "mu" is an exception: Between iteration number 250 and 300 it goes from negative territory to 20% and back to 10%.

```
> par(mfrow = c(1, 2))
> futuresplot(wheat.2007, type = "forward.curve")
> plot(wheat.fit.constr, type = "forward.curve", data = wheat.2007$price,
+      ttm = wheat.2007$ttm / 260)
> ##xx <- filter.schwartz2f(data = wheat.2007$price,
> ##                        ttm = wheat.2007$ttm / 260, wheat.fit.constr)
> ##plot(ts(xx$state, frequency = 260, start = 2007))
```

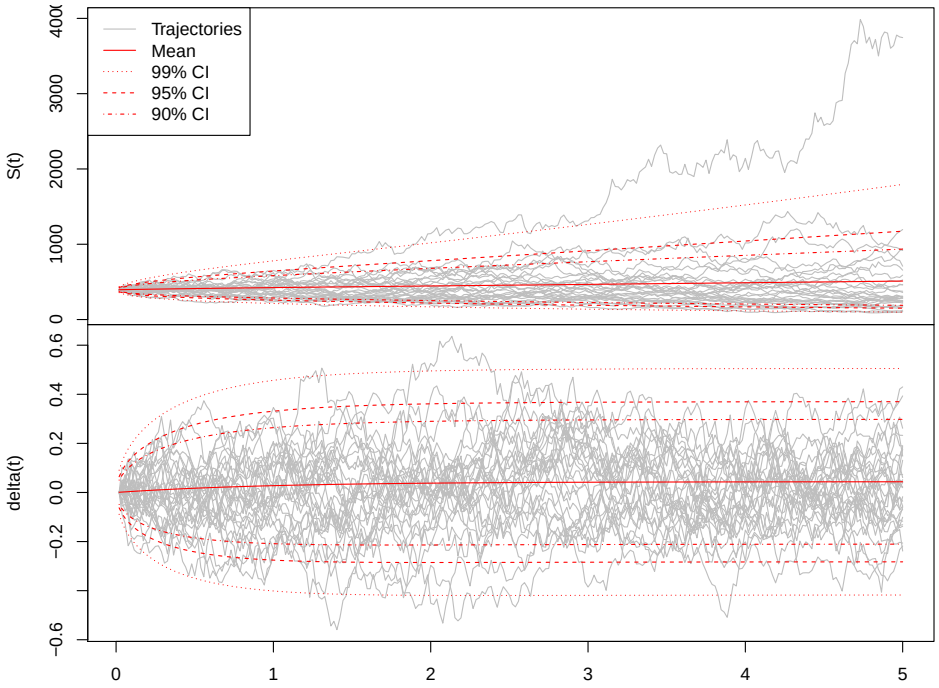

Figure 6: Thirty trajectories based on the constrained parameter estimates of wheat on a five years horizon and weekly sampling.

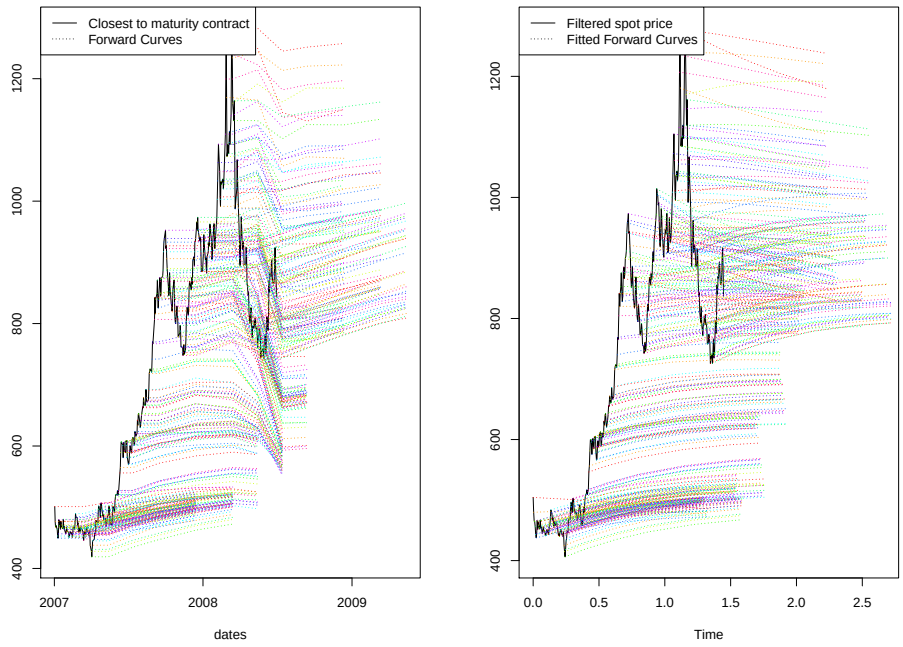

Figure 7: Real term structures (left panel) and model generated term structures (right panel) from Jan. 2007 to June 2008. The backwardation starting at mid of 2007 is not captured by the model initially, and underestimated later in 2007. The shape of the term structure at the peak starting in 2008 can not be produced by the Schwartz two-factor model. The model's prediction looks reasonable in Q1 2007 and Q2 2008.

8.2.1 Residual Analysis

Model validation is done here via graphical residual analysis. "Residuals" refer to prediction errors of the Kalman filter's measurement equation. According to the model residuals should be serially independent Gaussian random variables.

Different types of residulas can be obtained by the generic `resid` function and the specific argument `type`, which can be "filter" (raw filter residuals), "filter.std" (standardized filter residuals), and "real" (observed minus fitted prices). Standardized residuals are of interest here, hence the argument is "filter.std". First, serial independence is checked and then normality of residuals. Both assumptions are violated as shown in fig. 8 and fig. 9.

Before rejecting the two-factor model one should first increase the maximum number of iterations from 300 to, e.g., 3000 and investigate the parameters estimates again. Then, different settings for the measurement error standard deviations (argument `meas.sd`) could be tried, e.g. "all". Beside that different parameters could be hold constant.

```
> model.resid <- resid(wheat.fit.constr, data = futures$wheat$price, ttm = futures$w
+                      type = "filter.std")
> acf(model.resid, na.action = na.pass)
> par(mfrow = c(3, 2))
> invisible(apply(model.resid, 2, function(x)plot(density(na.omit(x)))))
```

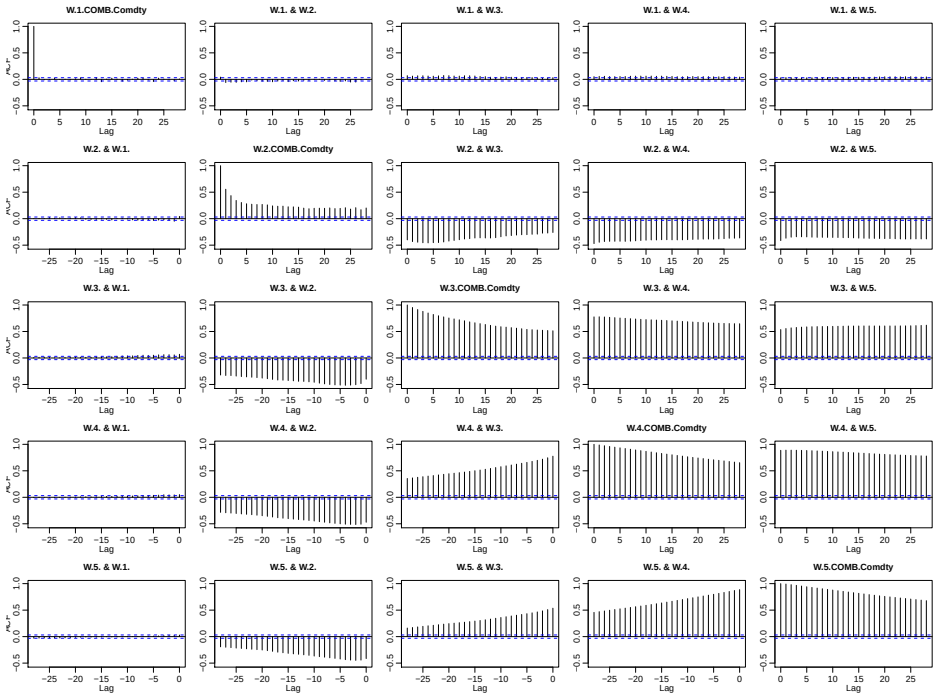

Figure 8: Residual's auto- and crosscorrelation estimates. Recall that off-diagonals of the measurement error covariance matrix are not estimated, hence the crosscorrelation is not relevant. Residuals of the closest to maturity futures show insignificant autocorrelation. However, residuals of all other futures are heavily autocorrelated. As the measurement error standard deviations are set proportional to the average traded volumes of the wheat futures, the two closest to maturity futures clearly get highest weights.

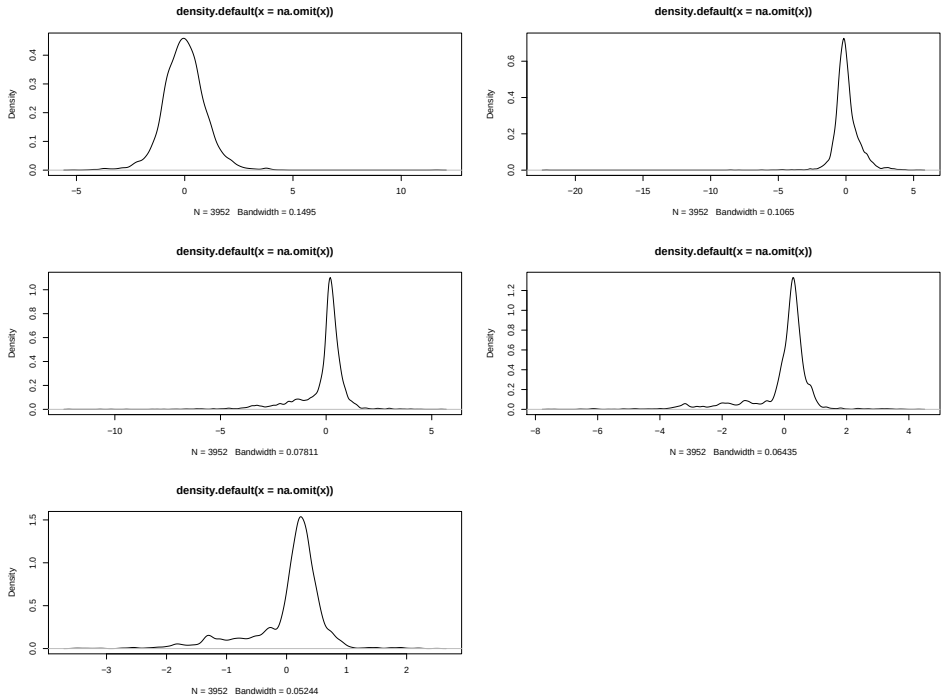

Figure 9: Non of the residual distributions look normal. The first distribution looks normally shaped but exhibits outliers on both tails. The second to fifth distributions look asymmetric and fat tailed.

8.2.2 Confidence Intervals and Value-at-Rsik

Confidence intervals and values-at-risk are computed here regardless of the misspecification of the model estimated above. In order to estimate the most recent values of the spot price and the convenience yield the function `filter.schwartz2f` is called first. Then, the 5% and 95% quantiles are computed and plotted (see fig. 10) for a one week horizon . Note that the 5% quantile is the 95% value-at-risk.

```
> state <- filter.schwartz2f(data = futures$wheat$price, ttm = futures$wheat$ttm / 2
> coefs <- coef(wheat.fit.constr)
> n <- nrow(futures$wheat$price)
> q.fut <- sapply(futures$wheat$ttm[n,] / 260, function(ttm, ...)
+                 qfutures(ttm = ttm, ...),
+                 p = c(0.05, 0.95), time = 5 / 260, s0 = state[n,1], delta0 = state
+                 mu = coefs$mu, sigmaS = coefs$sigmaS, kappa = coefs$kappa, alpha =
+                 sigmaE = coefs$sigmaE, rho = coefs$rho, r = coefs$r, lambda = coef
> plot(futures$wheat$ttm[n,], futures$wheat$price[n,], ylim = c(650, 850), type = "b
+      xlab = "Time to maturity [d]", ylab = "Price")
> points(futures$wheat$ttm[n,], q.fut[1,], col = "blue", type = "b")
> points(futures$wheat$ttm[n,], q.fut[2,], col = "blue", type = "b")
> legend("topleft", c("Current observed futures price", "One week ahead 90% confider
+        fill = c("black", "blue"))
> ## q.fut <- sapply(1:nrow(state), function(i, s0, delta0, ttm, ...)
> ##                 qfutures(s0 = s0[i], delta0 = delta0[i], ttm = ttm[i], ...),
> ##                 p = 0.05, time = 1 / 12, ttm = futures$wheat$ttm / 260, s0 = st
> ##                 delta0 = state[,2], mu = coefs$mu, sigmaS = coefs$sigmaS, kappa
> ##                 sigmaE = coefs$sigmaE, rho = coefs$rho, r = coefs$r, lambda = c
>
> ## par(mfrow = c(1, 2))
> ## plot(ts(cbind(state[,1], futures$wheat$price[,1], q.fut), freq = 260, start = 1
> ##      plot.type = "single", col = c("black", "blue", "red"))
> ## plot(ts(state[,2], freq = 260, start = 1995))
> ## abline(h = coefs$alphaT)
```

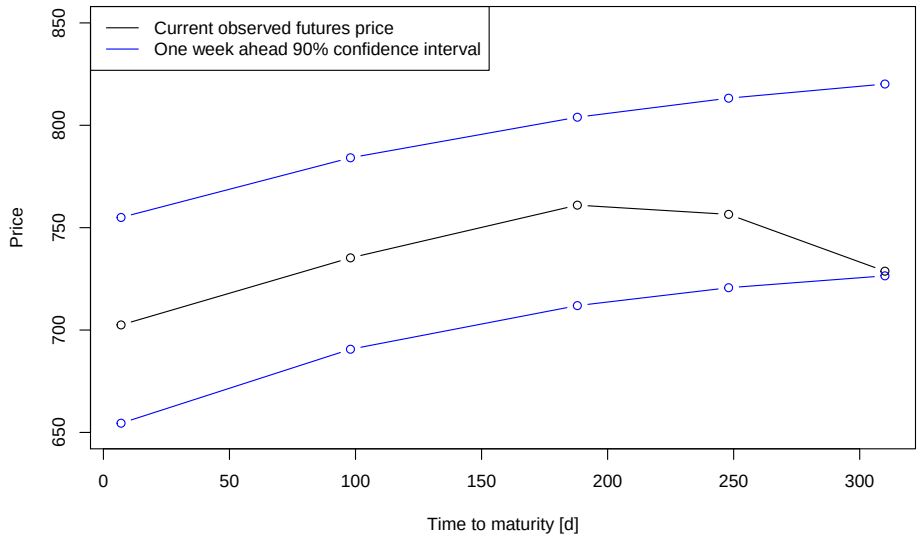

Figure 10: Current observed futures prices and one week ahead confidence intervals for the Sept. 2010, Dec. 2010, Mar. 2011, May 2011, and July 2011 wheat constracts as of Sept. 7, 2010. The July 2011 futures is already at the lower bound of the confidence interval. This is due to the fact that the current shape of the term structure cannot be captured fully by the Schwartz two-factor model

8.3 Example: Estimating Soybean Meal Parameters

Soybean meal parameters are estimated in this example based on weekly observations. All measurement error standard deviations are estimated (`opt.meas.sd` = "all"=), but kappa and lambda are held constant. Time-to-maturity (`ttm`) is divided by 260 as it is in unit of days and `deltat` is set to 1/52 because weekly price observations are used here.

First, data has to be made weekly. Wednesday observations are taken. Second, the estimation is carried out. Finally, real and fitted term structures are plotted in fig. 11. Residuals could be analysed as outlined in sec. 8.2.1.

```
> futures.w <- rapply(futures, function(x)x[format(as.Date(rownames(x)), "%w") == 3,
+                       classes = "matrix", how = "list")
> soybean.meal.fit <- fit.schwartz2f(data = futures.w$soybean.meal$price,
+                                    ttm = futures.w$soybean.meal$ttm / 260,
+                                    kappa = 1, mu = 0,
+                                    opt.pars = c(s0 = FALSE, delta0 = FALSE, mu = T
+                                      sigmaS = TRUE, kappa = FALSE, alpha = TRUE,
+                                      sigmaE = TRUE, rho = TRUE, lambda = FALSE),
+                                    opt.meas.sd = "all", deltat = 1 / 52,
+                                    control = list(maxit = 1000), silent = TRUE)
> soybean.meal.fit

------------------------------------------------------------------
Fitted Schwartz97 two-factor model:

SDE (P-dynamcis)
d S_t     = S_t   * (mu - delta_t) * dt    + S_t * sigmaS * dW_1
d delta_t = kappa * (alpha - delta_t) * dt + sigmaE * dW_2
E(dW_1 * dW_2) = rho * dt

SDE (Q-dynamcis)
d S_t     = S_t   * (r - delta_t) * dt      + S_t * sigmaS * dW*_1
d delta_t = kappa * (alphaT - delta_t) * dt + sigmaE * dW*_2
alphaT = alpha - lambda/kappa

Parameters
s0    : 150.7
delta0: 0
mu    : 0.249899307721834
sigmaS: 0.444902598205518
kappa : 1
alpha : -0.0891182726227762
sigmaE: 0.542971414981194
rho   : 0.912601358692617
```

```
r     : 0.03
lambda: 0
alphaT: -0.0891182726227762
-----------------------------------------------------------------
Optimization information
Converged:          FALSE
Fitted parameters:  mu, sigmaS, alpha, sigmaE, rho, meas.sd1, meas.sd2, meas.sd3, me
log-Likelihood:     -121733027
Nbr. of iterations: 1002
-----------------------------------------------------------------

> par(mfrow = c(1, 2))
> futuresplot(futures.w$soybean.meal, type = "forward.curve")
> plot(soybean.meal.fit, type = "forward.curve", data = futures.w$soybean.meal$price
+      ttm = futures.w$soybean.meal$ttm / 260)
```

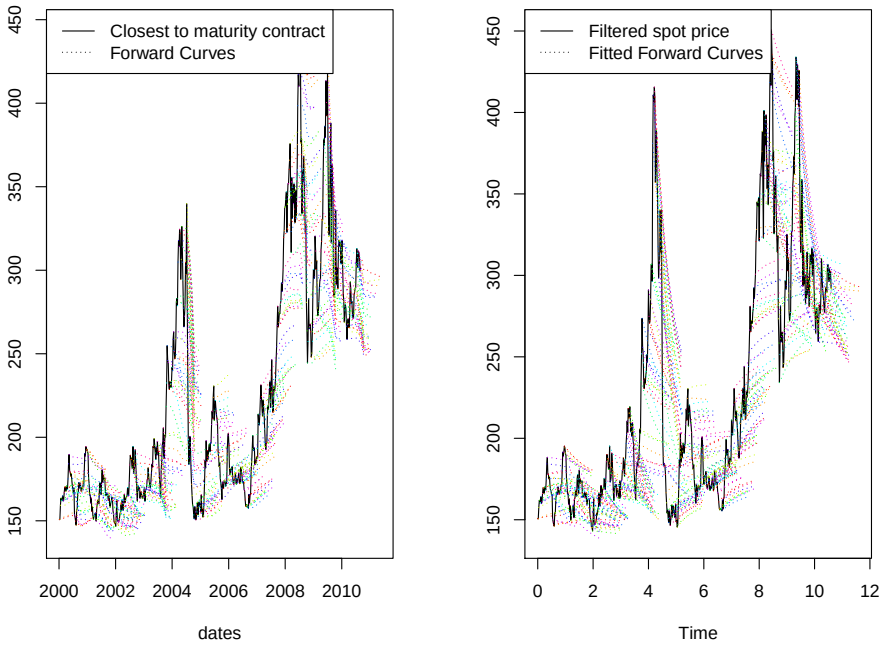

Figure 11: Real and fitted term structures of soybean meal data. Directionally fitted term structures mostly match observed term structures.

References

Rajna Gibson and Eduardo S. Schwartz. Stochastic convenience yield and the pricing of oil contingent claims. *The Journal of Finance*, 45(3):959–976, 1990.

Jimmy E. Hilliard and Jorge Reis. Valuation of commodity futures and options under stochastic convenience yields, interest rates, and jump diffusions in the spot. *Journal of Financial and Quantitative Analysis*, 33 (1):61–86, 1998.

Kristian R. Miltersen and Eduardo S. Schwartz. Pricing of options on commodity futures with stochastic term structures of convenience yields and interest rates. *Journal of Financial and Quantitative Analysis*, 33:33–59, 1998.

Eduardo S. Schwartz. The stochastic behavior of commodity prices: Implications for valuation and hedging. *Journal of Finance*, 52(3):923–973, 1997.