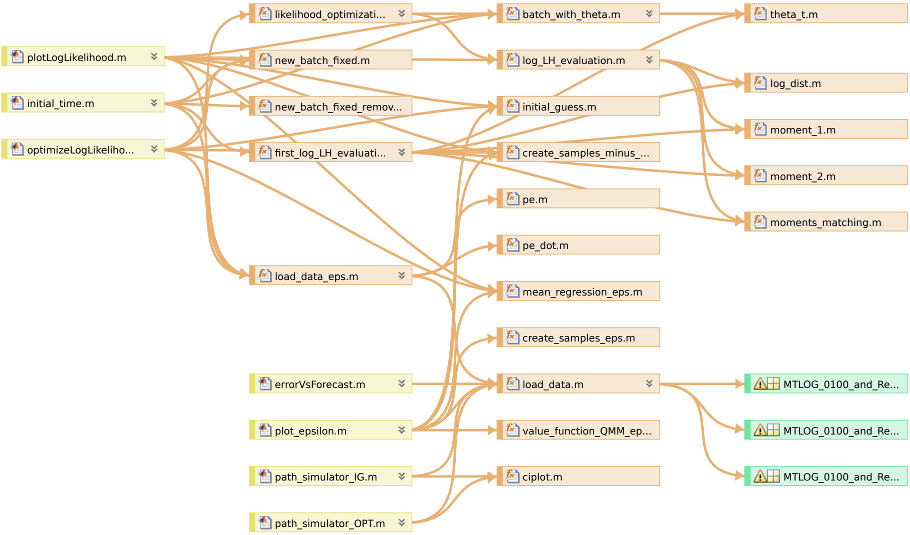
MATLAB: Read Me

Renzo Miguel Caballero Rosas

March 29, 2020

Code diagram:



Questions:

- Why $\mathrm{d}X = a(X;\theta)\,\mathrm{d}t + b(X;\theta,\alpha)\,\mathrm{d}W$ and no $\mathrm{d}X = a(X;\theta)\,\mathrm{d}t + b(X;\gamma)\,\mathrm{d}W$? Where all $\theta,\alpha,\gamma \in \mathbb{R}^+$. Because in the way it is defined, θ controls the mean reversion and α the wide of the confidence band. However, maybe it is better to optimize over θ and γ? Because of the relative dimension. After, trivially we can compute $\alpha = \gamma/\theta$.
- Which is Beta, the measurements or the transitions?
- Which data in the histograms? Measurements or transitions?

Some keywords:

- Our process is: **High-frequency in a fixed time-interval**.

Normalization:

Given the SDE

$$\mathrm{d}V_t = -\theta_t V_t \,\mathrm{d}t + \sqrt{2\theta_0 \alpha (V_t + p_t)(1 - V_t - p_t)}\,\mathrm{d}W_t,$$

we consider the normalized differentials $\mathrm{d}\hat{t} = \frac{\mathrm{d}t}{T}$, and $\mathrm{d}\hat{W}_t = \frac{\mathrm{d}W_t}{\sqrt{T}}$. Then, we can write the SDE as

$$\mathrm{d}V_t = -\theta_t T V_t \,\mathrm{d}\hat{t} + \sqrt{2\theta_0 T \alpha (V_t + p_t)(1 - V_t - p_t)}\,\mathrm{d}\hat{W}_t.$$

We conclude that, whatever the normalization constant T is, it gets absorbed by the parameter θ_t or θ_0 (let $\hat{\theta}_t = \theta_t T$ and $\hat{\theta}_0 = \theta_0 T$).
The E-M representation is

$$V_{t_{n+1}} = V_{t_n} - \left[\hat{\theta}_{t_n} V_{t_n}\right] \Delta s + \left[\sqrt{2\hat{\theta}_0 \alpha (V_{t_n} + p_{t_n})(1 - V_{t_n} - p_{t_n})}\right] \sqrt{\Delta s}\Delta \hat{W}_{t_n}, \;\; V_{t_0} = v_0,$$

for $n \in \{0 \ldots, m-1\}$, $t_j = t_0 + j\Delta s$, $t_0 = 0$, $t_m = 1$, and $\Delta \hat{W}_{t_n}$ normal (0,1) for each t_n.

SDE first moment (1/2):

Check if still correct for the new model!!!

Given some measurement $v_{t_{n-1}}$, we want to compute the first moment at time t_n. The exact first moment $m_1(s)$ for $s \in [t_{n-1}, t_n]$ is the solution of the ODE

$$\begin{cases} \mathrm{d}m_1(s) = \left[-m_1(s)\theta(s)\right]\mathrm{d}s, \\ m_1(t_{n-1}) = v_{t_{n-1}}. \end{cases}$$

If $\theta(t_{n-1}) = \theta(t_n) = \theta$, the solution is $m_1(t_n) = m_1(t_{n-1})e^{-\theta(t_n - t_{n-1})}$.

Otherwise, we compute a linear interpolation for $\theta(s)$ and solve the ODE using Forward-Euler:

$$m_1(s_n) = m_1(s_{n-1})(1 - \theta(s_{n-1})\Delta s).$$

SDE first moment (2/2): CODE

```matlab
function m1 = moment_1(v,th1,th2,dt,n) % 02/02/2020 18:28

    if th1 == th2 % We have the exact solution.
        m1 = v*exp(-th1*dt);
    else % Otherwise, we compute F-E.
        m1(1) = v;
        theta = @(i) th1 + (th2-th1) * i/n;
        ds    = dt/n;
        for i = 2:n
            m1(i) = m1(i-1) * (1 - theta(i-1)*ds);
        end
    end

end
```

The code is automatically imported from the MATLAB script.

SDE second moment (1/2):

Check if still correct for the new model!!!

Given some measurement $v_{t_{n-1}}$, we want to compute the second moment at time t_n. The exact second moment $m_2(s)$ for $s \in [t_{n-1}, t_n]$ is the solution of the ODE

$$\begin{cases} \mathrm{d}m_2(s) & = \left[-2(1+\alpha)m_2(s)\theta(s) + 2\alpha\theta(s)m_1(s)(1-2p(s)) + 2\alpha\theta(s)p(s)(1-p(s))\right]\mathrm{d}s, \\ & = 2\theta(s)\left[-(1+\alpha)m_2(s) + \alpha m_1(s)(1-2p(s)) + \alpha p(s)(1-p(s))\right]\mathrm{d}s, \\ m_2(t_{n-1}) & = v_{t_{n-1}}^2. \end{cases}$$

We compute a linear interpolation for the functions $\theta(s)$ and $p(s)$. After, we solve the ODE using Forward-Euler:

$$m_2(s_n) = m_2(s_{n-1}) + 2\theta(s_{n-1})\left[-(1+\alpha)m_2(s_{n-1}) + \alpha m_1(s_{n-1})(1-2p(s_{n-1})) + \alpha p(s_{n-1})(1-p(s_{n-1}))\right]\Delta s.$$

We use the same discretization points for both $m_1(s)$ and $m_2(s)$.

SDE second moment (2/2): CODE

```
function m2 = moment_2(v,th1,th2,p1,p2,alpha,m1,dt,n) % 02/02/2020 18:28

    if th1 == th2
        theta = @(i) th2;
        m1    = @(i) v*exp(-th1*dt*(i/n));
    else
        theta = @(i) th1 + (th2-th1) * i/n;
    end
    p     = @(i) p1  + (p2-p1)   * i/n;
    m2(1) = v^2;
    ds    = dt/n;

    for i = 2:n
        m2(i) = m2(i-1) + 2*theta(i-1)*ds * (-(1+alpha)*m2(i-1) + ...
            alpha*m1(i-1)*(1-2*p(i-1)) + alpha*p(i-1)*(1-p(i-1)));
    end

end
```

Density next measurement (1/2):

We want the next measurement $V_{t_n}|V_{t_{n-1}}$ to have a Beta distribution, but with support in $[a,b] = [-1,1]$. Given $X \sim \beta(\xi_1, \xi_2)$ a Beta distributed random variable, we define the new random variable $V = a + (b-a)X$ with support in $[-1,1]$, and PDF $f_V(v)$.

We can compute:

- $\mathbb{E}[V] = a + (b-a)\mathbb{E}[X] = a + (b-a)\frac{\xi_1}{\xi_1+\xi_2} = \mu_V$.
- $\mathbb{V}[V] = (b-a)^2\mathbb{V}[X] = \frac{(b-a)^2\xi_1\xi_2}{(\xi_1+\xi_2)^2(\xi_1+\xi_2+1)} = \sigma_V^2$.

Then, we want the SDE and our new PDF $f_V(v)$ to have the same moments at each $t \in \{\text{some appropriate domain}\}$, i.e., $\mu(t) = m_1(t)$ and $\sigma^2(t) = m_2(t) - m_1^2(t)$. $\mu(t)$ and $\sigma^2(t)$ refers to the mean and variance of $V_{t_n}|V_{t_{n-1}}$, following the structure described for $f_V(v)$.

Density next measurement (2/2):

For each measurement $V_{t_{n-1}}$, we can find the analytical moments for the SDE at time t_n solving the ODEs from slides 6 and 8. Then, we can find the parameters ξ_1 and ξ_2 such that both the SDE and the PDF of $V_{t_n}|V_{t_{n-1}}$ have the same first and second moments at time t_n.

- $\xi_1 = -\frac{(1+\mu)(\mu^2+\sigma^2-1)}{2\sigma^2}$ all evaluated at time t_n (verified in **Mathematica 11.0**[1]).
- $\xi_2 = \frac{(\mu-1)(\mu^2+\sigma^2-1)}{2\sigma^2}$ all evaluated at time t_n (verified in **Mathematica 11.0**).

```
function [xi1,xi2] = moments_matching(m1,m2) % 02/02/2020 19:17
%       disp(['m1 = ',num2str(m1),' and m2 = ',num2str(m2),'.']);
        mu    = m1;
        sig2  = m2 - m1^2;
        xi1   = - ((mu+1)*(mu^2+sig2 -1)) / (2*sig2);
        xi2   =   ((mu-1)*(mu^2+sig2 -1)) / (2*sig2);

end
```

[1] File: matchingVerification.nb.

Log-density (1/2):

Recall the PDF $f_V(v)$ from slide 10. We will use this density to model the random variables $V_{t_n}|V_{t_{n-1}}$. For $[a,b]=[-1,1]$, we have that

$$f_V(v) = f_X(g^{-1}(v))\left|\frac{\mathrm{d}}{\mathrm{d}v}g^{-1}(v)\right| \quad \text{where} \quad f_X(x) = \mathrm{Beta}(\xi_1,\xi_2) \quad \text{and} \quad g(x) = a+(b-a)x.$$

Then, $f_V(v) = \frac{1}{|(b-a)|}\frac{1}{B(\xi_1,\xi_2)}\left(\frac{v-a}{b-a}\right)^{\xi_1-1}\left(1-\frac{v-a}{b-a}\right)^{\xi_2-1}$ because $g^{-1}(v)=\frac{v-a}{b-a}$.

Also, we have that (up to some constant values)

$$\log\left(f_V(v)\right) = \log\left(\frac{1}{B(\xi_1,\xi_2)}\right) + (\xi_1-1)\log\left(\frac{v-a}{b-a}\right) + (\xi_2-1)\log\left(\frac{b-v}{b-a}\right),$$

where ξ_1 and ξ_2 depends on the SDE moments.

Log-density (2/2): CODE

```
function [val] = log_dist(v,xi1,xi2) % 03/02/2020 11:20
     a    = -1;
     b    =  1;
     val  = -betaln(xi1,xi2) + (xi1-1)*log((v-a)/(b-a)) + ...
            (xi2-1)*log((b-v)/(b-a));
end
```

Notice we use the function **betaln(a,b)**. It is important to compute the log directly for the beta function, and no first the beta, and after apply log.

Log-likelihood (1/2):

We introduce the number of paths M, and the number of measurements per path $N+1$ (N transitions). Then, we have a total of $M \times N$ samples to use. Notice that each pair (ξ_1, ξ_2) depends on $i \in \{1, \ldots, M\}$ and $j \in \{2, \ldots, N+1\}$. Then, the log-likelihood is

$$\mathfrak{L}\left(\{V\}_{M,N}\right) = \sum_{i=1}^{M} \sum_{j=2}^{N+1} \log\left[\rho_{i,j}\left(V_{i,j} | V_{i,j-1}\right)\right],$$

where $\rho_{i,j}\left(V_{i,j} | V_{i,j-1}\right) = \rho_{i,j}\left(V_{i,j} | V_{i,j-1}; \xi_{1_{i,j}}, \xi_{2_{i,j}}\right)$, and where we assumed a non-informative prior.

Data: CODE

We load our three tables: **Table_Training_Complete**, **Table_Testing_Complete**, and **Table_Complete**.

```
function [Table_Training_Complete, Table_Testing_Complete, Table_Complete] = load_data()

    % 03/02/2020 12:17

    load('../../Python/Represas_Data_2/Wind_Data/MTLOG_0100_and_Real_24h_Training_Data.mat');
    load('../../Python/Represas_Data_2/Wind_Data/MTLOG_0100_and_Real_24h_Testing_Data.mat');
    load('../../Python/Represas_Data_2/Wind_Data/MTLOG_0100_and_Real_24h_Complete_Data.mat');

%     Date                = Table_Training_Complete.Date;
%     Time                = Table_Training_Complete.Time;
%     Forecast            = Table_Training_Complete.Forecast;
%     Forecast_Dot        = Table_Training_Complete.Forecast_Dot;
%     real_ADME           = Table_Training_Complete.Real_ADME;
%     Error               = Table_Training_Complete.Error;
%     Error_Transitions   = Table_Training_Complete.Error_Transitions;
%     Lamparti_Data       = Table_Training_Complete.Error_Lamp;
%     Lamparti_Tran       = Table_Training_Complete.Error_Lamp_Transitions;

end
```

From θ_0 to θ_t:

To ensure that the analytical solutions is always in $]0,1[$, we choose the drift parameter to be

$$\theta(t) = \max\left(\theta_0, \frac{\theta_0\alpha + 2|\dot{p}^{\varepsilon}(t)|}{2\min(p^{\varepsilon}(t), 1 - p^{\varepsilon}(t))}\right), \quad \theta_0 > 0.$$

```
function [theta_t] = theta_t(theta_0, alpha, pe, pe_dot) % 13/03/2020 19:09

    theta_t = max(theta_0, (theta_0*alpha + 2*abs(pe_dot))/(2*min(pe,1-pe)));

end
```

Recall that we normalized w.r.t. time T. The normalized $\hat{\theta}_t$ is $\hat{\theta}_t = \theta_t T$. Notice that in the definition of θ_t, if we use $\hat{\theta}_0 = \theta_0 T$ and the derivative of p_t w.r.t. $\mathrm{d}\hat{t}$ instead of w.r.t. $\mathrm{d}t$, then we automatically have $\hat{\theta}_t$. See slide 5.

From p_t to p_t^ε:

To ensure that $\theta(t)$ does not blow up, we truncate the forecast so it is always in $[\varepsilon, 1-\varepsilon]$ for some $\varepsilon \in (0, 1/2)$.

Where $p^\varepsilon(t) = \begin{cases} \varepsilon & \text{if} \quad p(t) < \varepsilon \\ p(t) & \text{if} \quad \varepsilon \le p(t) < 1-\varepsilon \\ 1-\varepsilon & \text{if} \quad p(t) \ge 1-\varepsilon \end{cases}.$

```
function [pe] = pe(p, eps) % 18/03/2020 09:51
    for i = 1:length(p)
        if p(i) < eps
                pe(i) = eps;
            elseif p(i) >= eps && p(i) < 1-eps
                pe(i) = p(i);
            elseif p(i) >= 1-eps
                pe(i) = 1-eps;
        end
    end
end
```

$\dot{p}_t^\varepsilon = \frac{\mathrm{d}p^\varepsilon}{\mathrm{d}t}$. We use finite differences.

```
function [pe_dot] = pe_dot(pe, dt) % 18/03/2020 10:01

    pe_dot = (pe(2:end) - pe(1:end-1)) / dt;

end
```

Adapting the data:

We use this function to truncate the forecast, calculate its derivative, and to define the new errors $(X_t - p_t^{\varepsilon})$ associated with the new truncated forecast.

We call to all this new set of data, the truncated data.

```
function [Table_Training_Complete] = load_data_eps(ep)

    % 18/03/2020 10:41

    [Table_Training_Complete, ~, ~] = load_data();

    Time      = Table_Training_Complete.Time;
    Forecast  = Table_Training_Complete.Forecast;
    Real_ADME = Table_Training_Complete.Real_ADME;

    dt          = Time(1,2);
    [M, ~] = size(Forecast);

    for i = 1:M
        Forecast(i,:)             = pe(Forecast(i,:), ep);
        Forecast_Dot(i,:)         = pe_dot(Forecast(i,:), dt);
        Error(i,:)                = Real_ADME(i,:) - Forecast(i,:);
        Error_Transitions(i,:) = Error(i,2:end) - Error(i,1:end-1);
    end

    Table_Training_Complete.Forecast          = Forecast;
    Table_Training_Complete.Forecast_Dot      = Forecast_Dot;
    Table_Training_Complete.Error             = Error;
    Table_Training_Complete.Error_Transitions = Error_Transitions;

end
```

Create a new batch (1/2):

This function is independent of if we are using the real data or the truncated one.

If we take a total of $Z \in \mathbb{N}$ days, the batch corresponding to this days is

<table>
<tr><td colspan="7">PATH 1</td><td>...</td><td>PATH Z</td></tr>
<tr><td colspan="2">$t_n = 01:10$</td><td colspan="2">$t_n = 01:20$</td><td>...</td><td colspan="2">$t_n = 00:50$</td><td rowspan="4">...</td><td rowspan="4">...</td></tr>
<tr><td>$p(t_{n-1})$</td><td>$p(t_n)$</td><td>$p(t_{n-1})$</td><td>$p(t_n)$</td><td>...</td><td>$p(t_{n-1})$</td><td>$p(t_n)$</td></tr>
<tr><td>$\dot{p}(t_{n-1})$</td><td>$\dot{p}(t_n)$</td><td>$\dot{p}(t_{n-1})$</td><td>$\dot{p}(t_n)$</td><td>...</td><td>$\dot{p}(t_{n-1})$</td><td>$\dot{p}(t_n)$</td></tr>
<tr><td>$V(t_{n-1})$</td><td>$V(t_n)$</td><td>$V(t_{n-1})$</td><td>$V(t_n)$</td><td>...</td><td>$V(t_{n-1})$</td><td>$V(t_n)$</td></tr>
</table>

with dimensions $3 \times (2Z(N-1))$. As an example: If we have 145 measurements (N+1), then $N = 144$ and $N - 1 = 143$. We use 143 samples because we need to ignore the initial measurement (because we do not have data at time t_{-1}) and the final one (because it does not have $\dot{p}$). Then, each day has 143 samples. In this implementation, we are duplicating the data. In case of a lack of RAM, we can reduce the dimensions to $3 \times (Z(N-1))$.

Create a new batch (2/2): CODE

```
function [Table_Training, new_bat] = new_batch_fixed(Table_Training, batch_size, N)
    % 11/02/2020 09:35

    Forecast     = Table_Training.Forecast;
    Forecast_Dot = Table_Training.Forecast_Dot;
    Error        = Table_Training.Error;
    new_bat      = [];

    for i = 1:batch_size
        for j = 2:N
            forecast(2*j-3:2*j-2)     = Forecast(i,j-1:j);
            forecast_dot(2*j-3:2*j-2) = Forecast_Dot(i,j-1:j);
            error(2*j-3:2*j-2)        = Error(i,j-1:j);
        end
        new_bat = [new_bat, [forecast; forecast_dot; error]];
    end
end
```

We call it **fixed** because, at some point, we were going to optimize the Likelihood taking random batches and increasing the size at each iteration. This idea was discarded since we have computational power enough to use all data at once.

Complete batch:

This function is independent of if we are using the real data or the truncated one. After we created a batch with the elements $(p(t), \dot{p}(t), V(t))$, we want to add the parameter $\theta(t)$ to use in the likelihood. Following the same idea than in slide 19, the complete batch is

PATH 1							...	PATH Z
$t_n = 01:10$		$t_n = 01:20$		...	$t_n = 00:50$			
$p(t_{n-1})$	$p(t_n)$	$p(t_{n-1})$	$p(t_n)$	...	$p(t_{n-1})$	$p(t_n)$		
$\dot{p}(t_{n-1})$	$\dot{p}(t_n)$	$\dot{p}(t_{n-1})$	$\dot{p}(t_n)$	...	$\dot{p}(t_{n-1})$	$\dot{p}(t_n)$	...	...
$V(t_{n-1})$	$V(t_n)$	$V(t_{n-1})$	$V(t_n)$	...	$V(t_{n-1})$	$V(t_n)$		
$\theta(t_{n-1})$	$\theta(t_n)$	$\theta(t_{n-1})$	$\theta(t_n)$	...	$\theta(t_{n-1})$	$\theta(t_n)$		

```
function [batch_theta] = batch_with_theta(batch, alpha, theta_0) % 09/02/2020 18:51
    batch(4,1)   = theta_t(theta_0, alpha, batch(1,1), batch(2,1));
    batch(4,end) = theta_t(theta_0, alpha, batch(1,end), batch(2,end));
    for i = 2:2:length(batch(1,:))-1
        batch(4,i:i+1) = theta_t(theta_0, alpha, batch(1,i), batch(2,i));
    end
    batch_theta = batch;
end
```

Initial guess for $\theta_0 \cdot \alpha$: Quadratic variation

Recall we have M paths with $N+1$ measurements each. Depending on the SDE we are using, we have the two approximations:

$$X_t : \theta_0^* \alpha^* \approx \frac{1}{2M\Delta t} \sum_{j=1}^{M} \frac{\sum_{i=1}^{N} (\Delta X_{i,j})^2}{\sum_{i=1}^{N} (X_{i,j})(1 - X_{i,j})}, \quad V_t : \theta_0^* \alpha^* \approx \frac{1}{2M\Delta t} \sum_{j=1}^{M} \frac{\sum_{i=1}^{N} (\Delta V_{i,j})^2}{\sum_{i=1}^{N} (V_{i,j} + p_{i,j})(1 - V_{i,j} - p_{i,j})}$$

```
function [est] = initial_guess(real_prod, M, N, dt)
    % 09/02/2020 09:30
    est = 0;

    for i = 1:M
        numerator = 0; denominator = 0;
        for j = 1:N
            numerator   = numerator + (real_prod(i,j+1) - real_prod(i,j))^2;
            denominator = denominator + real_prod(i,j)*(1-real_prod(i,j));
        end
        est = est + numerator/denominator;
    end
    est = est / (2*M*dt);

end
```

Initial guess for θ_0 (1/2): Mean least squares

Recall we have M paths with $N+1$ measurements each. If we assume $\theta_t^\varepsilon = \theta_0$ for $t \in [0, T]$, then we have

$$\theta_0^* \approx \arg\min_{\theta_0} \left[\sum_{j=1}^{M} \sum_{i=1}^{N} \big(V_{i+1,j} - V_{i,j}(1 - \theta_0 \Delta t) \big)^2 \right]. \tag{1}$$

As problem (1) is convex, we can formulate the equivalent problem using derivatives

$$\theta_0^* \approx \frac{1}{\Delta t \cdot M} \sum_{j=1}^{M} \frac{\sum_{i=1}^{N} V_{i,j}(V_{i,j} - V_{i+1,j})}{\sum_{i=1}^{N} V_{i,j}^2}.$$

Initial guess for θ_0 (2/2): Mean least squares

As we need to assume that $\theta_t^{\varepsilon} = c \in \mathbb{R}^+$ in order to use correctly the estimator (1), we define $\varepsilon, \gamma \in \mathbb{R}^+$, and estimate $\frac{\theta_0 \alpha}{\delta}$ using data which original forecast satisfies $p_i \notin (\varepsilon, 1-\varepsilon)$ and θ_0 using data which original forecast satisfies $p_i \in [\gamma, 1-\gamma]$. The full explanation can be found in slides **20200308 - Initial Guess.pdf**.

```matlab
function [val, accum] = mean_regression_eps(error, dt) % 16/03/2020 11:08
    [M, N_ini]  = size(error);
    num   = 0; den = 0;
    accum = 0;
    for i = 1:M
        for j = 1:N_ini-1
            if error(i,j+1) ~= -1
                num   = num + error(i,j) * (error(i,j) - error(i,j+1));
                den   = den + error(i,j)^2;
                accum = accum + 1;
            end
        end
    end
    val = num/(den*dt);
end
```

To create this specific data sets, we use the functions **create_samples_eps.m** (ε-data) and **create_samples_minus_eps.m** (γ-data).

Initial guess for δ:

We have that, for almost all days, at time $t = t_0$, $X(t_0) \neq p(t_0)$ and then $V(t_0) \neq 0$. However, by forecast construction, there should exist a time $t_{-\delta} < t_0$ such that $V(t_{-\delta}) = 0$.

We extrapolate linearly $p(t)$ so we can evaluate $p(t_{-\delta})$, and assume that $V(t_{-\delta}) = 0$. Then, for each day j, we have an initial transition $(V_{j,t_0} | V_{j,t_{-\delta}}; \boldsymbol{\theta}, \delta)$. We assume again that it is Beta and apply the same moment matching as for the rest of transitions. With our initial guess for $\boldsymbol{\theta}$, we can construct our initial guess for δ solving the problem

$$\delta \approx \arg\min_{\delta} \mathscr{L}_\delta(\boldsymbol{\theta}, \delta; V^{M,1}) = \arg\min_{\delta} \prod_{j=1}^{M} \rho_0(V_{j,t_0} | V_{j,t_{-\delta}}; \boldsymbol{\theta}, \delta) .$$

See slides **20200319 - Time delta** for very detailed information.

We use $\delta = 5.5\,\mathrm{h}$.

Log-likelihood evaluation (1/2): CODE

```
function [value] = log_LH_evaluation(batch_complete, alpha, dt) % 03/02/2020 19:42

    for i = 1:length(batch_complete(1,:))/2

        j = i*2; % This is the real index (parfor must go one-by-one).
        % Recall that: j is t_n and j-1 is t_{n-1}.
        p1 = batch_complete(1,j-1); p2 = batch_complete(1,j);
        v1 = batch_complete(3,j-1); v2 = batch_complete(3,j);
        th1 = batch_complete(4,j-1); th2 = batch_complete(4,j);
        n = 10; % 10 discretizations for the ODEs.

        m1        = moment_1(v1,th1,th2,dt,n);
        m2        = moment_2(v1,th1,th2,p1,p2,alpha,m1,dt,n);
        [xi1,xi2] = moments_matching(m1(end),m2(end));
        val(i)    = log_dist(v2,xi1,xi2);

    end

    value = sum(val);

end
```

Log-likelihood evaluation (2/2):

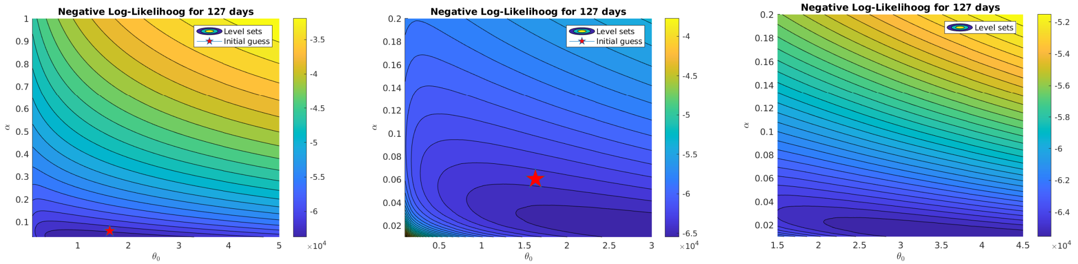


Figure 1: We use the code **plotLogLikelihood.m** to create this plots. We used about 18 thousand transitions, and we can also see the initial guess.

Summing-up:

1. Initial guess: From quadratic variation and using all the data we get $\hat{\theta}_0 \times \hat{\alpha}$. Choosing some appropiate γ, from mean least squares we get $\hat{\theta}_0$, then we also get $\hat{\alpha}$.
   Also, from mean least squares we can estimate ε^*, using that the least squares estimator estimates $\frac{\theta_0 \alpha}{\varepsilon}$ for some appropriate ε.
   Finally, using the pair $(\hat{\theta}_0, \hat{\alpha})$, we can obtain $\hat{\eta}$ and $\hat{\delta}$.
2. Optimal parameters: Using the training data and the initial point $(\hat{\theta}_0, \hat{\alpha})$, we can find the optimal pair (θ_0^*, α^*) using the MLE. With this new pair, we can find η^* and δ^*.

Values for the parameters:

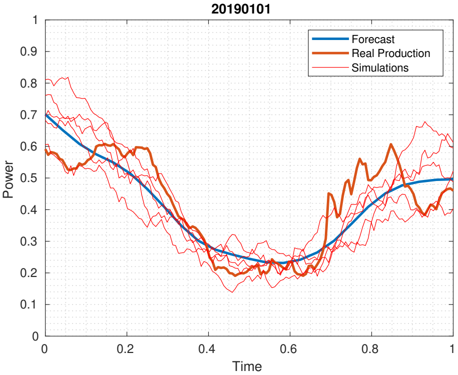
$\hat{\theta}_0$	$\hat{\alpha}$	$\hat{\eta}$	$\hat{\delta}$	γ	ε
1.63	0.06	*not apply*	220 min	0.3	0.018

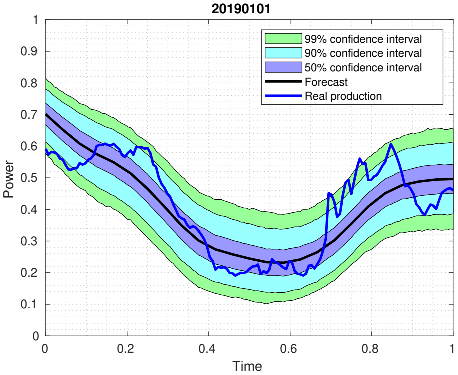
η^{ini} is not relevant here because we do not need to remove any data to find δ^{ini}.

θ_0^*	α^*	η^*	δ^*
3.91	0.02	0.1	220 min

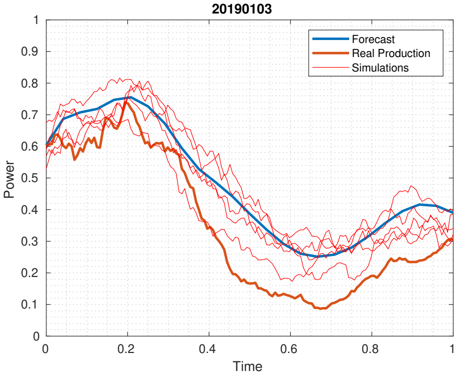
Notice that $\hat{\delta} = \delta^*$. However, to obtain δ^* we had to remove all initial errors larger than η^*, and in this way, we reduced the variance of the initial error. This has sense since $\theta_0^* \alpha^* < \hat{\theta}_0 \hat{\alpha}$.

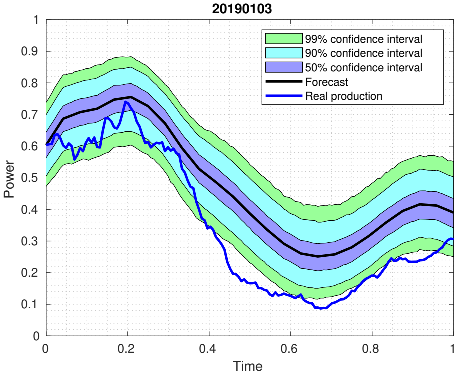
Paths and bands for optimal values (1/4):



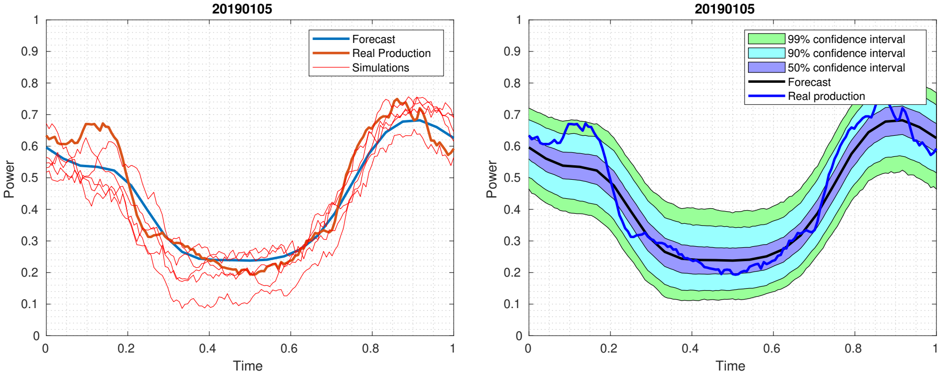


Paths and bands for optimal values (2/4):

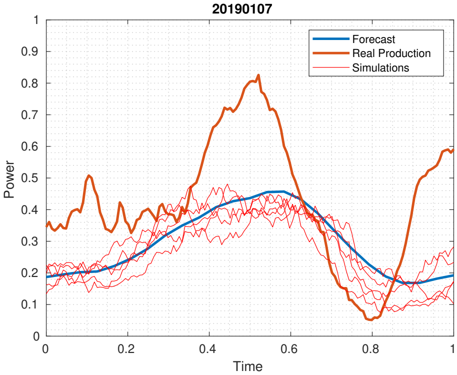


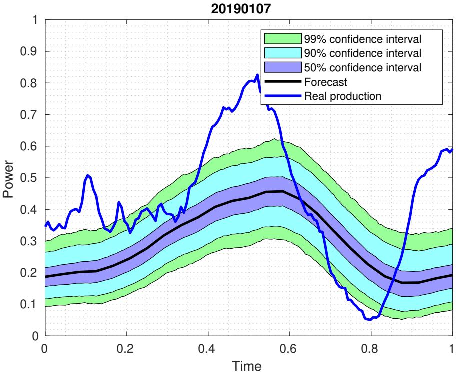


Paths and bands for optimal values (3/4):



Paths and bands for optimal values (4/4):





Lamperti Transform

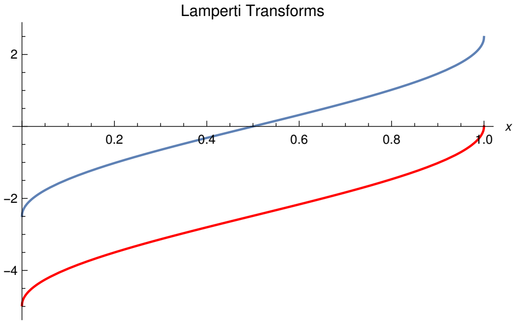
Two candidates:

Recall $X_t = V_t + p_t$.

We have the candidates:

- $Z_t = \frac{1}{\sqrt{2\alpha\theta_0}} \arcsin(2(V_t + p_t) - 1)$.
- $Z_t = \sqrt{\frac{2}{\alpha\theta_0}} \arcsin(\sqrt{1 - V_t - p_t})$.

They have the same partial derivatives. The main difference appears in the resulting SDE for Z_t. In the first case, we have a $\sin(\sqrt{2\alpha\theta_0} Z_t)$ in the drift, while in the second case, we have $\cos(\sqrt{2\alpha\theta_0} Z_t)$.



For now, we will use the first candidate.

Lamperti data:

We are following the first candidate. Then, we caa

Lamperti moments:

The Lamperti SDE is:

$$\mathrm{d}Z_t = \underbrace{\left[\frac{(\alpha\theta_0 - \theta_t)\sin(\sqrt{2\alpha\theta_0}Z_t) - \theta_t(1-2p_t) + 2\dot{p}_t}{\sqrt{2\alpha\theta_0}\cos(\sqrt{2\alpha\theta_0}Z_t)}\right]}_{:=b(Z_t)}\mathrm{d}t + \mathrm{d}W_t.$$

Here we follow Bayesian Filtering and Smoothing, Chapter 9. We use the linearization-based approximation for the moments.

SDE first Lamperti moment (mean) (1/2):

Given some measurement $z_{t_{n-1}}$, we want to compute the first moment at time t_n. The linearly approximated first moment $\mu_Z(s)$ for $s \in [t_{n-1}, t_n]$ is the solution of the ODE

$$\begin{cases} \mathrm{d}\mu_L(s) & = b(\mu_L(s))\,\mathrm{d}s, \\ \mu_L(t_{n-1}) & = z_{t_{n-1}}. \end{cases}$$

We solve numerically the ODE using Forward-Euler:

$$\mu_L(s_n) = \mu_L(s_{n-1}) + b(\mu_L(s_{n-1}))\Delta s.$$

SDE first Lamperti moment (mean) (2/2):

```
function m1 = moment_1_L(z,theta,alpha,p1,p2,dt,n)
    % 29/03/2020 16:28
     p_dot    = (p2-p1) / dt;
     m1(1)    = z;
     p_t      = @(i) p1 + (p2-p1) * i/n;
     ds       = dt/n;

     for i = 2:n
          Theta_t = theta_t(theta, alpha, p_t(i), p_dot);
          m1(i)   = m1(i-1) + ds * ...
               ((alpha*theta-Theta_t) * sin(m1(i-1)*sqrt(2*alpha*theta)) - ...
               Theta_t*(1-2*p_t(i)) + 2*p_dot) / ...
               (sqrt(2*alpha*theta) * cos(m1(i-1)*sqrt(2*alpha*theta)));
     end

end
```

SDE Lamperti variance (1/2):

Given some measurement $z_{t_{n-1}}$, we want to compute the variance at time t_n. The linearly approximated variance $v_Z(s)$ for $s \in [t_{n-1}, t_n]$ is the solution of the ODE

$$\begin{cases} \mathrm{d}\sigma_L^2(s) & = \left(2\sigma_L^2 b'(\mu_L(s)) + 1\right) \mathrm{d}s, \\ \sigma_L^2(t_{n-1}) & = 0. \end{cases}$$

We solve numerically the ODE using Forward-Euler:

$$\sigma_L^2(s_n) = \sigma_L^2(s_{n-1}) + \Big(1 + 2\sigma_L^2(s_{n-1}) b'(\mu_L(s_{n-1}))\Big) \Delta s.$$

We have that (computed with Mathematica):

$$b'(z) = \frac{\mathrm{d}b}{\mathrm{d}z}(z) = \frac{\alpha\theta_0 - \theta_t + (2\dot{p}_t + (2p_t - 1)\theta_t)\sin(z\sqrt{2\alpha\theta_0}))}{\cos^2(z\sqrt{2\alpha\theta_0})}.$$

SDE Lamperti variance (2/2):

```
function m2 = moment_2_L(m1,theta,alpha,p1,p2,dt,n)
    % 29/03/2020 16:50
    p_dot   = (p2-p1) / dt;
    m2(1)   = 0;
    p_t     = @(i) p1 + (p2-p1) * i/n;
    ds      = dt/n;

    for i = 2:n
        Theta_t = theta_t(theta, alpha, p_t(i), p_dot);
        b_prime = (alpha*theta - Theta_t + (2*p_dot+(2*p_t(i)-1)*Theta_t) * ...
            sin(m1(i-1) * sqrt(2*alpha*theta))) / ((cos(m1(i-1) * sqrt(2*alpha*theta)))^2);
        m2(i)   = m2(i-1) + ds * ...
            (1+2*m2(i-1)*b_prime);
    end

end
```

Density next measurement:

We want the next measurement $Z_{t_n}|Z_{t_{n-1}}$ to have a Gaussian with mean μ_Z and variance σ_Z^2 ($f_Z(z) = \mathcal{N}(\mu_Z, \sigma_Z^2)$).

Then, we want the SDE and our new PDF $f_Z(z)$ to have the same moments at each $t \in \{\text{some appropriate domain}\}$, i.e., $\mu_Z(t) = \mu_L(t)$ and $\sigma_Z^2(t) = \sigma_L^2(t)$.

The density and log-density (natural logarithm) are:

$$f_Z(z) = \frac{e^{-\frac{1}{2}\left(\frac{z-\mu_Z}{\sigma_Z}\right)^2}}{\sigma_Z\sqrt{2\pi}} \iff \log(f_Z(z)) = -\frac{1}{2}\left(\frac{z-\mu_Z}{\sigma_Z}\right)^2 - \log(\sigma_Z\sqrt{2\pi}).$$

```
function [val] = log_dist_L(z,mu,sig) % 29/03/2020 17:49

    val = -(1/2)*((z-mu)/sig)^2 - log(sig*sqrt(2*pi));

end
```