

MATLAB: Read Me

Renzo Miguel Caballero Rosas

February 10, 2020

Questions:

- ▶ Why $dX = a(X; \theta)dt + b(X; \theta, \alpha)dW$ and not $dX = a(X; \theta)dt + b(X; \gamma)dW$? Where all $\theta, \alpha, \gamma \in \mathbb{R}^+$. Because in the way it is defined, θ controls the mean reversion and α the width 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

$$dV_t = -\theta_t V_t dt + \sqrt{2\theta_t \alpha(V_t + p_t)(1 - V_t - p_t)} dW_t,$$

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

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

We conclude that, whatever the normalization constant T is, it gets absorbed by the parameter θ_t (let $\hat{\theta}_t = \theta_t T$).

The E-M representation is

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

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

SDE first moment (1/2):

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

$$\begin{cases} dm_1(s) = [-m_1(s)\theta(s)] ds, \\ m_1(t_{n-1}) = v_{t_{n-1}}. \end{cases}$$

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

```
1 function m1 = moment_1(v,th1,th2,dt,n) % 02/02/2020 18:28
2
3     if th1 == th2 % We have the exact solution.
4         m1 = v*exp(-th1*dt);
5     else % Otherwise, we compute F-E.
6         m1(1) = v;
7         theta = @(i) th1 + (th2-th1) * i/n;
8         ds = dt/n;
9         for i = 2:n
10             m1(i) = m1(i-1) * (1 - theta(i-1)*ds);
11         end
12     end
13
14 end
```

The code is automatically imported from the MATLAB script.

SDE second moment (1/2):

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

$$\begin{cases} dm_2(s) &= [-2(1+\alpha)m_2(s)\theta(s) + 2\alpha\theta(s)m_1(s)(1-2p(s)) + 2\alpha\theta(s)p(s)(1-p(s))] ds, \\ &= 2\theta(s) [-(1+\alpha)m_2(s) + \alpha m_1(s)(1-2p(s)) + \alpha p(s)(1-p(s))] ds, \\ m_2(t_{n-1}) &= v_{t_{n-1}}^2. \end{cases}$$

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

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

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

SDE second moment (2/2): CODE

```
1 function m2 = moment_2(v, th1, th2, p1, p2, alpha, m1, dt, n) % 02/02/2020 18:28
2
3     if th1 == th2
4         theta = @(i) th2;
5         m1     = @(i) v*exp(-th1*dt*(i/n));
6     else
7         theta = @(i) th1 + (th2-th1) * i/n;
8     end
9     p      = @(i) p1 + (p2-p1) * i/n;
10    m2(1) = v^2;
11    ds    = dt/n;
12
13    for i = 2:n
14        m2(i) = m2(i-1) + 2*theta(i-1)*ds * (-(1+alpha)*m2(i-1) + ...
15            alpha*m1(i-1)*(1-2*p(i-1)) + alpha*p(i-1)*(1-p(i-1)));
16    end
17
18 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 5 and 7. 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**).

```
1 function [xi1,xi2] = moments_matching(m1,m2) % 02/02/2020 19:17
2 %      disp(['m1 = ',num2str(m1),' and m2 = ',num2str(m2),'.']);
3     mu    = m1;
4     sig2  = m2 - m1^2;
5     xi1   = - ((mu+1)*(mu^2+sig2-1)) / (2*sig2);
6     xi2   =  ((mu-1)*(mu^2+sig2-1)) / (2*sig2);
7
8 end
```

¹File: [matchingVerification.nb](#).

Log-density (1/2):

Recall the PDF $f_V(v)$ from slide 9. 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

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

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

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

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

where ξ_1 and ξ_2 depends on the SDE moments.

Log-density (2/2): CODE

```
1 function [val] = log_dist(v,xi1,xi2) % 03/02/2020 11:20
2     a = -1;
3     b = 1;
4     val = log(1/beta(xi1,xi2)) + (xi1-1)*log((v-a)/(b-a)) + ...
5         (xi2-1)*log((b-v)/(b-a));
6     % disp(['xi1 = ',num2str(xi1),' and xi2 = ',num2str(xi2),'.']);
7     % if val == Inf
8     %     disp('Some value was infinite in the log-likelihood. ');
9     %     disp(['xi1 = ',num2str(xi1),' and xi2 = ',num2str(xi2),'.']);
10    % end
11 end
```

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 $i \in \{1, \dots, M\}$ and $j \in \{2, \dots, N+1\}$. Then, the log-likelihood is

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

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

Data: CODE

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

```
1 function [Table_Training_Complete , Table_Testing_Complete , Table_Complete] = load_data()  
2  
3     % 03/02/2020 12:17  
4  
5     load(' ../../ Python/Represas_Data_2/Wind_Data/MTLOG_0100_and_Real_24h_Training_Data.mat');  
6     load(' ../../ Python/Represas_Data_2/Wind_Data/MTLOG_0100_and_Real_24h_Testing_Data.mat');  
7     load(' ../../ Python/Represas_Data_2/Wind_Data/MTLOG_0100_and_Real_24h_Complete_Data.mat');  
8  
9     %      Date           = Table_Training_Complete.Date;  
10    %      Time           = Table_Training_Complete.Time;  
11    %      Forecast       = Table_Training_Complete.Forecast;  
12    %      Forecast_Dot   = Table_Training_Complete.Forecast_Dot;  
13    %      real_ADME      = Table_Training_Complete.Real_ADME;  
14    %      Error          = Table_Training_Complete.Error;  
15    %      Error_Transitions = Table_Training_Complete.Error_Transitions;  
16    %      Lamparti_Data   = Table_Training_Complete.Error_Lamp;  
17    %      Lamparti-Tran   = Table_Training_Complete.Error_Lamp_Transitions;  
18  
19 end
```

Create a new batch (1/2):

To guarantee the data homogeneity, we sample per day and not per transition. This means that each batch is composed of data corresponding to some amount of days. If we sample a total of $Z \in \mathbb{N}$ days, the batch corresponding to this days is

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

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

```
1 function [Table_Training, new_bat] = new_batch(Table_Training, batch_size, N)
2     % 03/02/2020 17:41
3     idx = randperm(height(Table_Training), batch_size); % Sample indices.
4     idx = -sort(-idx); % We order the indices from large to small.
5     Forecast = Table_Training.Forecast;
6     Forecast_Dot = Table_Training.Forecast_Dot;
7     Error = Table_Training.Error;
8     new_bat = [];
9
10    for i = 1:length(idx)
11        Table_Training(idx(i), :) = []; % We remove the row that we sample from.
12        for j = 2:N
13            forecast(2*j-3:2*j-2) = Forecast(idx(i), j-1:j);
14            forecast_dot(2*j-3:2*j-2) = Forecast_Dot(idx(i), j-1:j);
15            error(2*j-3:2*j-2) = Error(idx(i), j-1:j);
16        end
17        new_bat = [new_bat, [forecast; forecast_dot; error]];
18    end
19 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

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

```
1 function [theta_t] = theta_t(theta_0 , p, p_dot) % 03/02/2020 13:58
2
3     theta_t = max(theta_0 , abs(p_dot)/(min(p,1-p)));
4
5 end
```

Complete batch:

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 15, the complete batch is

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

```

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

```

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

Recall we have M paths with $N + 1$ measurements each.

$$\theta_0 \alpha \approx \frac{1}{2M\Delta t} \sum_{j=1}^M \frac{\sum_{i=1}^N (x_{i+1,j} - x_{i,j})^2}{\sum_{i=1}^N x_{i,j}(1 - x_{i,j})} \approx 0.094.$$

```
1 function [est] = initial_guess(real_prod, M, N, dt)
2     % 09/02/2020 09:30
3     est = 0;
4
5     for i = 1:M
6         numerator = 0; denominator = 0;
7         for j = 1:N
8             numerator = numerator + (real_prod(i, j+1) - real_prod(i, j))^2;
9             denominator = denominator + real_prod(i, j)*(1 - real_prod(i, j));
10        end
11        est = est + numerator/denominator;
12    end
13    est = est / (2*M*dt);
14
15 end
```

Initial guess for θ_0 (1/2):

Recall we have M paths with $N + 1$ measurements each.

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

```
1 function [val] = initial_theta(error, forecast, M, N, dt, theta)
2     % 09/02/2020 09:58
3     val = 0;
4     for i = 1:M
5         for j = 1:N
6             p_dot = (forecast(i,j+1) - forecast(i,j)) / dt;
7             theta_T = theta_t(theta, forecast(i,j), p_dot);
8             val = val + (error(i,j+1)-error(i,j)+theta_T*error(i,j)*dt)^2;
9         end
10    end
11
12 end
```

Initial guess for θ_0 (2/2):

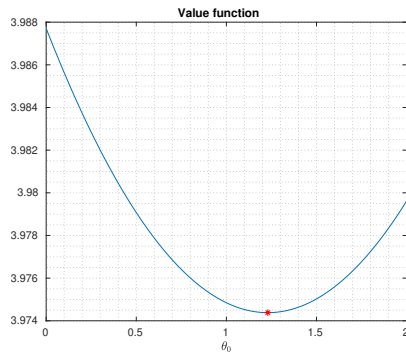
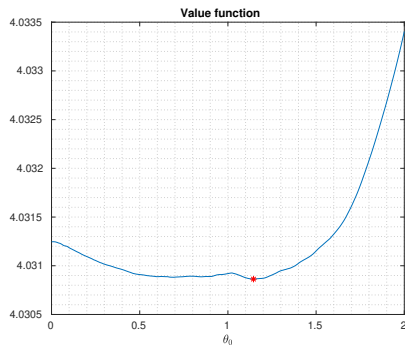


Figure 1: On the left, we use the value function defined in slide 20. On the right, we substitute in the value function θ_t by θ_0 and plot that *new value function*. We use the value from the left plot.

$$\theta_0 \approx 1.1450 \text{ and } \theta_0 \alpha \approx 0.094 \implies \alpha \approx 0.082.$$

Initial guess for δ (1/2):

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

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

We got $\delta \approx 80$ min.

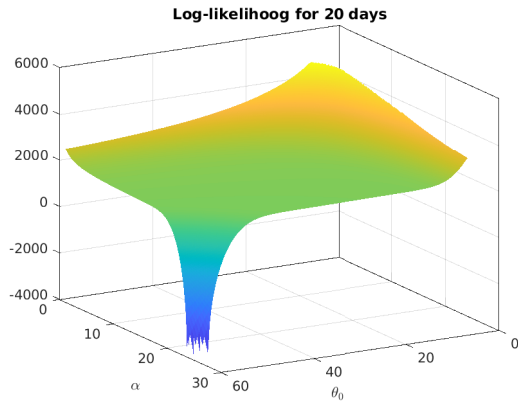
Initial guess for δ (2/2):

```
1 function [value] = first_log_LH_evaluation(batch_complete, alpha, delta, dt, N)
2
3 % 09/02/2020 18:35
4
5 for i = 0:length(batch_complete(1,:))/((N-1)*2)-1
6
7     p1 = batch_complete(1,1 + i*(N-1)*2);
8     p2 = batch_complete(1,12 + i*(N-1)*2); % This is the measurement at 01:10.
9     p0 = p1 - delta*(p2-p1)/(6*dt);
10    n = 10; % 10 discretizations for the ODEs.
11    th0 = batch_complete(4,1 + i*(N-1)*2); th1 = th0;
12    v0 = 0; v1 = batch_complete(3,1 + i*(N-1)*2);
13
14    m1 = moment_1(v0,th0,th1,delta,n);
15    m2 = moment_2(v0,th0,th1,p0,p1,alpha,m1,delta,n);
16    [xi1,xi2] = moments_matching(m1(end),m2(end));
17    val(i+1) = log_dist(v1,xi1,xi2);
18
19 end
20
21 value = sum(val);
22
23 end
```

Log-likelihood evaluation (1/2): CODE

```
1 function [value] = log_LH_evaluation(batch_complete, alpha, dt) % 03/02/2020 19:42
2
3     for i = 1:length(batch_complete(1,:))/2 % I can be changed to parfor.
4         % However, parfor seems to be slower.
5
6         j = i*2; % This is the real index (parfor must go one-by-one).
7         % Recall that: j is t_n and j-1 is t_{n-1}.
8         p1 = batch_complete(1,j-1); p2 = batch_complete(1,j);
9         v1 = batch_complete(3,j-1); v2 = batch_complete(3,j);
10        th1 = batch_complete(4,j-1); th2 = batch_complete(4,j);
11        n = 10; % 10 discretizations for the ODEs.
12
13        m1      = moment_1(v1,th1,th2,dt,n);
14        m2      = moment_2(v1,th1,th2,p1,p2,alpha,m1,dt,n);
15        [xi1,xi2] = moments_matching(m1(end),m2(end));
16        val(i)   = log_dist(v2,xi1,xi2);
17
18    end
19
20    value = sum(val);
21
22 end
```


Log-likelihood evaluation (2/2):



We are still analyzing why the log-likelihood keeps decreasing for large α and θ_0 .