MATLAB: Read Me

Renzo Miguel Caballero Rosas

February 9, 2020

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

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

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

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

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

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

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):

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

```
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):

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 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**).

```
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 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

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

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 = log(1/beta(xi1,xi2)) + (xi1-1)*log((v-a)/(b-a)) + ...
        (xi2-1)*log((b-v)/(b-a));
%     disp(['xi1 = ',num2str(xi1),' and xi2 = ',num2str(xi2),'.']);
%     if val == Inf
%         disp('Some value was infinite in the log-likelihood.');
%         disp(['xi1 = ',num2str(xi1),' and xi2 = ',num2str(xi2),'.']);
%     end
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 $i \in \{1, \ldots, M\}$ 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
```

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

```
function [Table_Training, new_bat] = new_batch(Table_Training, batch_size, N)
    % 03/02/2020 17:41
    idx           = randperm(height(Table_Training), batch_size); % Sample indices.
    idx           = -sort(-idx); % We order the indices from large to small.
    Forecast      = Table_Training.Forecast;
    Forecast_Dot  = Table_Training.Forecast_Dot;
    Error         = Table_Training.Error;
    new_bat       = [];

    for i = 1:length(idx)
        Table_Training(idx(i),:) = []; % We remove the row that we sample from.
        for j = 2:N
            forecast(2*j-3:2*j-2)     = Forecast(idx(i),j-1:j);
            forecast_dot(2*j-3:2*j-2) = Forecast_Dot(idx(i),j-1:j);
            error(2*j-3:2*j-2)        = Error(idx(i),j-1:j);
        end
        new_bat = [new_bat, [forecast; forecast_dot; error]];
    end
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

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

```
function [theta_t] = theta_t(theta_0, p, p_dot) % 03/02/2020 13:58

    theta_t = max(theta_0, abs(p_dot)/(min(p,1-p)));

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

<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>
<tr><td>$\theta(t_{n-1})$</td><td>$\theta(t_n)$</td><td>$\theta(t_{n-1})$</td><td>$\theta(t_n)$</td><td>...</td><td>$\theta(t_{n-1})$</td><td>$\theta(t_n)$</td><td></td><td></td></tr>
</table>

```
function [batch_theta] = batch_with_theta(batch, theta_0) % 03/02/2020 18:32
    batch(4,1) = theta_t(theta_0, batch(1,1), batch(2,1));
    batch(4,end) = theta_t(theta_0, batch(1,end), batch(2,end));
    for i = 2:2:length(batch(1,:)-1)
        batch(4,i:i+1) = theta_t(theta_0, batch(1,i), batch(2,i));
    end
    batch_theta = batch;
end
```

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

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

$$\theta_0 \alpha \approx \frac{1}{2M\Delta t} \sum_{j=1}^{M} \frac{\sum_{i=1}^{N} (x_{i+1,j} - x_{i,j})^2}{\sum_{i=1}^{N} x_{i,j}(1 - x_{i,j})} \approx 0.021.$$

```matlab
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));
            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:

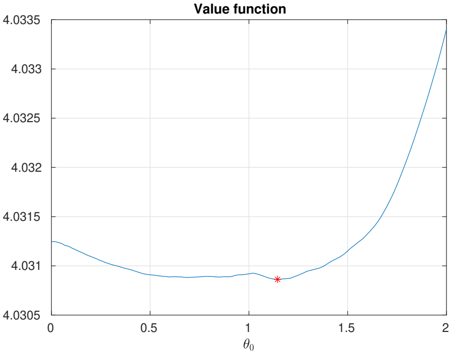
Recall we have M paths with $N+1$ measurements each.

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

```
function [val] = initial_theta(error, forecast, M, N, dt, theta)
    % 09/02/2020 09:58
    val = 0;
    for i = 1:M
        for j = 1:N
            p_dot   = (forecast(i,j+1) - forecast(i,j)) / dt;
            theta_T = theta_t(theta, forecast(i,j), p_dot);
            val     = val + (error(i,j+1)-error(i,j)+theta_T*error(i,j)*dt)^2;
        end
    end

end
```

Initial guess for θ_0:



$\theta_0 \approx 1.1450$ and $\theta_0 \alpha \approx 0.021 \implies \alpha \approx 0.018.$

Log-likelihood evaluation: CODE

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

    for i = 1:length(batch_complete(1,:))/2 % I can be changed to parfor.
        % However, parfor seems to be slower.

        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: Plot