

On the Simulation and Estimation of the Mean-Reverting Ornstein-Uhlenbeck Process

Especially as Applied to Commodities Markets and Modelling

William Smith, February 2010

Version 1.01

Abstract

Mean reverting processes are widely seen in finance. They are widely used to model interest rates, and are of particular use to those modelling commodities. The most popular model is the Ornstein and Uhlenbeck (1930) ('O-U') process, also known as the Vasicek (1977) process. I discuss the model briefly, including Matlab code to simulate the process. I discuss the estimation of the parameters, in particular the difficulty of estimating the speed-of-mean-reversion parameter. Again, I include extensive Matlab code for parameter estimation.

Use of the Ornstein Uhlenbeck Process in Commodity Modelling

Mean reverting processes are naturally attractive to model commodity *prices* since they embody the economic argument that when prices are 'too high', demand will reduce and supply will increase, producing a counter-balancing effect. When prices are 'too low' the opposite will occur, again pushing prices back towards some kind of long term mean.

Mean reverting processes are also useful for modelling other processes, observed or unobserved, such as interest rates or commodity 'convenience yield'.

The Ornstein Uhlenbeck process is widely used for modelling a mean reverting process. The process 'S' is modelled as

$$ds = \lambda(\mu - S)dt + \sigma dW_t$$

Where

- W_t is a Brownian- Motion, so $dW_t \sim N(0, \sqrt{dt})$,
- λ measures the speed of mean reversion
- μ is the 'long run mean', to which the process tends to revert.
- σ , as usual, is a measure of the process volatility

It is widely studied, has a number of well known closed form solutions, and has only 3 parameters to estimate. Its weakness is that nothing prevents the process from going negative. If this is undesirable, two approaches are:

1. Modify the process away from a pure O-U process, and modulate the volatility parameter as S tends towards zero, for example the Cox, Ingersoll, Ross (1985) model expresses variations

in interest rates 'r' as:

$$dr = \kappa(\theta - r)dt + \sigma\sqrt{r}dW_t$$

2. Model the log of the spot price, so a log-spot of below zero still corresponds to a spot price above zero.

Key commodity papers rely on the mean-reverting Ornstein-Uhlenbeck process, for example the widely-used Gibson and Schwartz (1990) model uses a mean-reverting process for the commodity convenience yield.

Modelling An O-U Process

In order to model the O-U process on a computer (for example using Matlab), it is usual to discretize time, and calculate samples at discrete timesteps of width Δt .

A naïve derivation is as follows:

$$ds = \lambda(\mu - S)dt + \sigma dW_t$$

$$S_t - S_{t-1} = \lambda(\mu - S_{t-1})\Delta t + \sigma dW_t$$

$$S_t = S_{t-1} + \lambda(\mu - S_{t-1})\Delta t + \sigma dW_t$$

For a Matlab implementation, see [SimulateOrnsteinUhlenbeckRough](#) below.

Gillespie (1996) points out that this simulation is only valid when the discrete Δt is sufficiently small. An exact formula¹ that holds for any size of Δt is:

$$S_t = e^{-\lambda\Delta t}S_{t-1} + (1 - e^{-\lambda\Delta t})\mu + \sigma\sqrt{\frac{(1 - e^{-2\lambda\Delta t})}{2\lambda}}dW_t$$

For a Matlab implementation, see [SimulateOrnsteinUhlenbeck](#) below. In particular, in my implementation, I note and handle the singularity in the above equation when the process is *not* mean reverting, i.e. $\lambda = 0$.

¹ See also "Monte Carlo Simulation of Stochastic Processes", http://www.puc-rio.br/marco.ind/sim_stoc_proc.html#mc-mrd

```

function [ S ] = SimulateOrnsteinUhlenbeckRough( S0, mu, sigma, lambda,deltat, t )
%% Approximate Ornstein-Uhlenbeck Generator.  A more accurate version is preferred
%% and available : SimulateOrnsteinUhlenbeck.

%% License
% Copyright 2010, William Smith, CommodityModels.com . All rights reserved.
%
% Redistribution and use in source and binary forms, with or without modification, are
% permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright notice, this list of
% conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright notice, this list
% of conditions and the following disclaimer in the documentation and/or other materials
% provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER, WILLIAM SMITH ``AS IS'' AND ANY EXPRESS
% OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
% MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
% THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
% SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
% OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
% HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
% OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
% SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

periods = floor(t / deltat);

S = zeros(periods, 1);

S(1) = S0;

dWt = sqrt(deltat) * randn(periods,1);
for t=2:1:periods

    dSt = lambda*(mu-S(t-1))*deltat + sigma*dWt(t);
    S(t) = S(t-1)+dSt;
end

% OPTIM Note : % Precalculating all dWt's rather than one-per loop makes this function
% approx 50% faster.  Useful for Monte-Carlo simulations.

% OPTIM Note : I tried calculating an array of dSt's and only doing a cumsum() at
% the end, but it doesn't speedup any more.

end

```

```

function [ S ] = SimulateOrnsteinUhlenbeck( S0, mu, sigma, lambda, deltat, t )
%% Simulate an ornstein uhlenbeck process.
%% Looks more complicated than expected, because if we don't include the
%% exp() terms, we are not accurate as deltat becomes large.

%% Reference
% Based on the equation described in see
% http://www.puc-rio.br/marco.ind/sim\_stoc\_proc.html#mc-mrd
% For a formal treatment, see
% Gillespie, D. T. 1996. 'Exact numerical simulation of the Ornstein-Uhlenbeck process
% and its integral.' Physical review E 54, no. 2: 2084-2091.

%% License
% Copyright 2010, William Smith, CommodityModels.com . All rights reserved.
%
% Redistribution and use in source and binary forms, with or without modification, are
% permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright notice, this list of
% conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright notice, this list
% of conditions and the following disclaimer in the documentation and/or other materials
% provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER, WILLIAM SMITH ``AS IS'' AND ANY EXPRESS
% OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
% MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
% THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
% SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
% OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
% HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
% OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
% SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.s

%% Code
periods = floor(t / deltat);

S = zeros(periods, 1);

S(1) = S0;

exp_minus_lambda_deltat = exp(-lambda*deltat);

% Calculate the random term.
if (lambda == 0)
    % Handle the case of lambda = 0 i.e. no mean reversion.
    dWt = sqrt(deltat) * randn(periods,1);
else
    dWt = sqrt((1-exp(-2*lambda* deltat))/(2*lambda)) * randn(periods,1);
end

% And iterate through time calculating each price.
for t=2:1:periods
    S(t) = S(t-1)*exp_minus_lambda_deltat + mu*(1-exp_minus_lambda_deltat) + sigma*dWt(t);
end

% OPTIM Note : % Precalculating all dWt's rather than one-per loop makes this function
% approx 50% faster. Useful for Monte-Carlo simulations.

% OPTIM Note : calculating exp(-lambda*deltat) makes it roughly 50% faster
% again.

% OPTIM Note : this is only about 25% slower than the rough calculation
% without the exp correction.

end

```

Estimating the Parameters of an Observed O-U Process

Well known techniques for parameter estimation are Least Square regressions, and Maximum Likelihood.

Least Squares

In the case of least-square regression, we can take the naïve updating formula above and simply turn it into a regression:

$$S_t - S_{t-1} = \lambda(\mu - S_{t-1})\Delta t + \sigma dW_t$$

$$S_t - S_{t-1} = \lambda\mu\Delta t + \lambda\Delta t S_{t-1} + \sigma dW_t$$

$$y = a + bx + \varepsilon_t$$

If we therefore regress a 'y' value of $S_t - S_{t-1}$ against an 'x' of S_{t-1} , we will recover $\hat{\lambda}$ as $\frac{b}{\Delta t}$, and

from there we can recover $\hat{\mu}$ as $\frac{a}{\hat{\lambda}\Delta t}$.

Finally, we can recover $\hat{\sigma}$ as $\frac{sd(\varepsilon_t)}{\sqrt{\Delta t}}$.

This procedure is written as the Matlab `CalibrateOrnsteinUhlenbeckRegress` below.

If we use the exact updating formula,

$$S_t = (1 - e^{-\lambda\Delta t})\mu + e^{-\lambda\Delta t}S_{t-1} + \sigma\sqrt{\frac{(1 - e^{-2\lambda\Delta t})}{2\lambda}}dW_t$$

$$y = a + bx + \varepsilon_t$$

we notice that we can now regress S_t against S_{t-1} , and derive

$$e^{-\hat{\lambda}\Delta t} = b$$

$$-\hat{\lambda}\Delta t = \ln(b)$$

$$\hat{\lambda} = -\frac{\ln(b)}{\Delta t}$$

$$(1 - e^{-\hat{\lambda}\Delta t})\hat{\mu} = a$$

$$(1-b)\hat{\mu} = a$$

$$\hat{\mu} = \frac{a}{(1-b)}$$

And finally,

$$\hat{\sigma} = sd(\varepsilon_t) / \sqrt{\frac{(1-e^{-2\hat{\lambda}\Delta t})}{2\hat{\lambda}}}$$

$$\hat{\sigma} = sd(\varepsilon_t) \sqrt{\frac{2\hat{\lambda}}{(1-e^{-2\hat{\lambda}\Delta t})}}$$

This more exact derivation of the parameters by least square is given by the Matlab function

`CalibrateOrnsteinUhlenbeckLeastSquares` below.

```
function [ mu, sigma, lambda ] = CalibrateOrnsteinUhlenbeckRegress(S, deltat, bigt)
%#ok<INUSD>
% Calibrate an OU process by a simple discrete time regression.
% Does not properly take the reversion into account, meaning this will
% become inaccurate for large deltat.
%
% Use CalibrateOrnsteinUhlenbeckLeastSquares if deltat is small.
%

%% License
% Copyright 2010, William Smith, CommodityModels.com . All rights reserved.
%
% Redistribution and use in source and binary forms, with or without modification, are
% permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright notice, this list of
% conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright notice, this list
% of conditions and the following disclaimer in the documentation and/or other materials
% provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER, WILLIAM SMITH ``AS IS'' AND ANY EXPRESS
% OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
% MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
% THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
% SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
% OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
% HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
% OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
% SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

% Regressions prefer row vectors to column vectors, so rearrange if
% necessary.
if (size(S,2) > size(S,1))
    S = S';
end

% Regress S(t)-S(t-1) against S(t-1).
[ k,dummy,resid ] = regress(S(2:end)-S(1:end-1),[ ones(size(S(1:end-1))) S(1:end-1) ] );
```

```

a = k(1);
b = k(2);

lambda = -b/deltat;
mu      = a/lambda/deltat;

sigma   = std(resid) / sqrt(deltat);

end

```

```

function [ mu, sigma, lambda ] = CalibrateOrnsteinUhlenbeckLeastSquares(S, deltat, bigt)
% Calibrate an OU process by least squares.
%
%% Reference.
% Based on the logic described at
% http://sitmo.com/doc/Calibrating\_the\_Ornstein-Uhlenbeck\_model

%% License
% Copyright 2010, William Smith, CommodityModels.com . All rights reserved.
%
% Redistribution and use in source and binary forms, with or without modification, are
% permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright notice, this list of
% conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright notice, this list
% of conditions and the following disclaimer in the documentation and/or other materials
% provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER, WILLIAM SMITH ``AS IS'' AND ANY EXPRESS
% OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
% MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
% THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
% SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
% OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
% HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
% OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
% SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

%% Code.

% Regressions prefer row vectors to column vectors, so rearrange if
% necessary.
if (size(S,2) > size(S,1))
    S = S';
end

[ k,dummy,resid ] = regress(S(2:end),[ ones(size(S(1:end-1))) S(1:end-1) ] );
a = k(1);
b = k(2);

lambda = -log(b)/deltat;
mu      = a/(1-b);
sigma   = std(resid) * sqrt( 2*lambda/(1-b^2) );

end

```

Maximum Likelihood

An alternative parameter estimation technique is maximum likelihood. I do not derive the maximum likelihood estimation here, but please see [http://sitmo.com/doc/Calibrating the Ornstein-Uhlenbeck model](http://sitmo.com/doc/Calibrating_the_Ornstein-Uhlenbeck_model) for a good description. A basic maximum likelihood implementation in Matlab based on that description is in the function `CalibrateOrnsteinUhlenbeckMaxLikelihood` below.

```
function [ mu, sigma, lambda ] = CalibrateOrnsteinUhlenbeckMaxLikelihood(S, deltat, T)
% Calibrate an OU process by maximum likelihood.

%% Reference
% Based on the algorithm and software described at :
% http://www.sitmo.com/doc/Calibrating_the_Ornstein-Uhlenbeck_model
n = length(S)-1;

Sx = sum( S(1:end-1) );
Sy = sum( S(2:end) );
Sxx = sum( S(1:end-1).^2 );
Sxy = sum( S(1:end-1).*S(2:end) );
Syy = sum( S(2:end).^2 );

mu = (Sy*Sxx - Sx*Sxy) / ( n*(Sxx - Sxy) - (Sx^2 - Sx*Sy) );

lambda = -(1/deltat)*log((Sxy - mu*Sx - mu*Sy + n*mu^2) / (Sxx - 2*mu*Sx + n*mu^2));
alpha = exp(- lambda*deltat);
alpha2 = exp(-2*lambda*deltat);
sigmahat2 = (1/n)*(Syy - 2*alpha*Sxy + alpha2*Sxx - ...
    2*mu*(1-alpha)*(Sy - alpha*Sx) + n*mu^2*(1-alpha)^2);
sigma = sqrt(sigmahat2*2*lambda/(1-alpha2));

end
```

Bias and Weaknesses in the Estimation Techniques

Both least-square minimization and maximum likelihood estimation techniques are known to be good at estimating σ and μ , but poor in estimating λ . See Yu (2009) for a recent treatment. By good, I mean firstly that the estimate is unbiased, and secondly that the standard deviation of estimates is low, i.e. the estimate is accurate.

To evaluate the degree that the parameters from estimation can be trusted, I the Matlab testbed 'MLE_Test'. This performs as follows:

1. Draw a random path based on the O-U process with known parameters.
2. Using the various estimation techniques described above, calculate estimates of those same parameters based on the random path.
3. Repeat many times.
4. Display the mean estimates, standard deviations of the estimates as well as histograms.

In order to improve the estimate of λ , Phillips and Yu (2005) propose a 'jackknife' technique, whereby λ is estimated over the whole sample, and denoted λ_T , as well as over m equal partitions of the data, dividing the time period in 2, call these $\lambda_1, \lambda_2, \dots, \lambda_m$. They advocate a low m of 2 or 3. The bias in the estimate is greatly reduced if we then create a new estimate

$$\lambda_{jack} = \frac{m}{m-1} \lambda_T - \frac{\sum_{i=1}^m \lambda_i}{m^2 - m}$$

I implement this technique in `CalibrateOrnsteinUhlenbeckMaxLikelihoodJackknife` below.

```
function [ mu, sigma, lambda ] = CalibrateOrnsteinUhlenbeckMaxLikelihoodJackknife(S, deltat, T)
%% Calibrate an O-U processes' parameters by maximum likelihood. Since the basic ML
%% calibration has a bias (resulting in frequent estimates of lambda which are much too
%% high), we perform a 'jackknife' operation to %% reduce the bias.

%% License
% Copyright 2010, William Smith, CommodityModels.com . All rights reserved.
%
% Redistribution and use in source and binary forms, with or without modification, are
% permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright notice, this list of
% conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright notice, this list
% of conditions and the following disclaimer in the documentation and/or other materials
% provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER, WILLIAM SMITH ``AS IS'' AND ANY EXPRESS
% OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
% MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
% THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
% SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
% OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
% HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
```

```
% OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS  
% SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

```
%% Reference.
```

```
% To get a less biased lambda, we just the jackknife procedure described in  
% Phillips, Peter C. B., and Jun Yu. 2005. 'Jackknifing Bond Option Prices.'  
% The Review of Financial Studies 18, no. 2 (Summer): 707-742.  
% http://www.jstor.org/stable/3598050
```

```
m = 2; % Number of partitions.  
partlength = floor(length(S)/m);
```

```
Spart = zeros(m,partlength);  
for i=1:1:m  
    Spart(i,:) = S(partlength*(i-1)+1:partlength*i);  
end
```

```
%fprintf('n = %d\n', length(S));
```

```
%% Calculate for entire partition.
```

```
[ muT, sigmaT, lambdaT ] = CalibrateOrnsteinUhlenbeckMaxLikelihood(S, deltat, T);
```

```
%% Calculate the individual partitions.
```

```
mupart = zeros(m,1);  
sigmapart = zeros(m,1);  
lambdapart = zeros(m,1);  
for i=1:1:m  
    [ mupart(i), sigmapart(i), lambdapart(i) ] = ....  
        CalibrateOrnsteinUhlenbeckMaxLikelihood(Spart(i,:), deltat, T/m);  
end
```

```
%% Now the jackknife calculation.
```

```
lambda = (m/(m-1))*lambdaT - (sum(lambdapart))/(m^2-m);
```

```
% mu and sigma are not biased, so there's no real need for the jackknife.
```

```
% But we do it anyway for demonstration purposes.
```

```
mu = (m/(m-1))*muT - (sum(mupart))/(m^2-m);  
sigma = (m/(m-1))*sigmaT - (sum(sigmapart))/(m^2-m);
```

```
end
```

Estimating the Gibson-Schwartz Convenience Yield Model

Gibson and Schwartz aim to estimate the mean reversion parameters of the stochastic convenience yield they observe in crude oil futures. Their entire sample is on weekly observations ($\Delta t = 1/50$), from January 1984 to November 1988 ($T \approx 5$, #observations=250), see Figure 1 below.

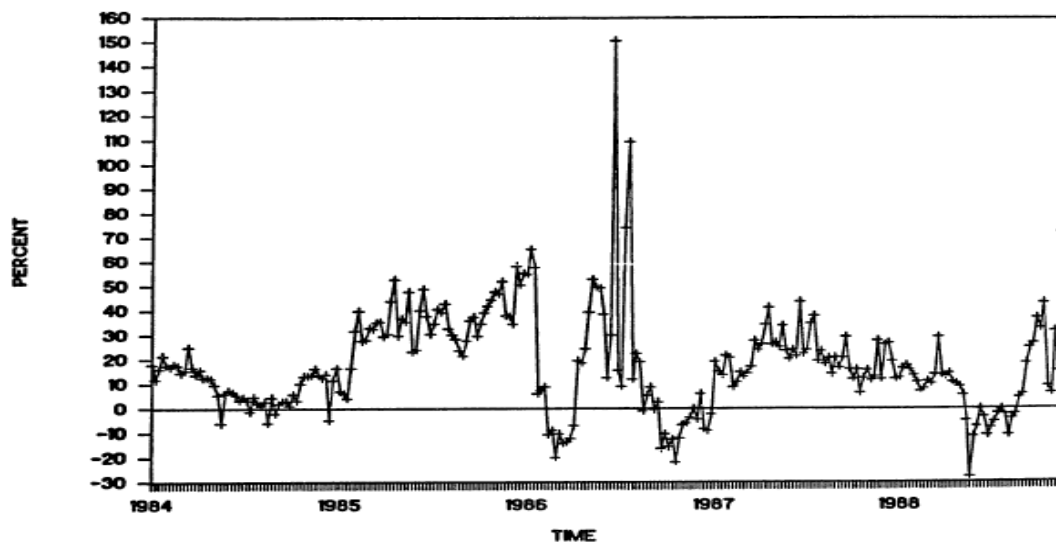


Figure 1 - Convenience Yield of Crude Oil, 1984-1988 (Gibson and Schwartz 1990)

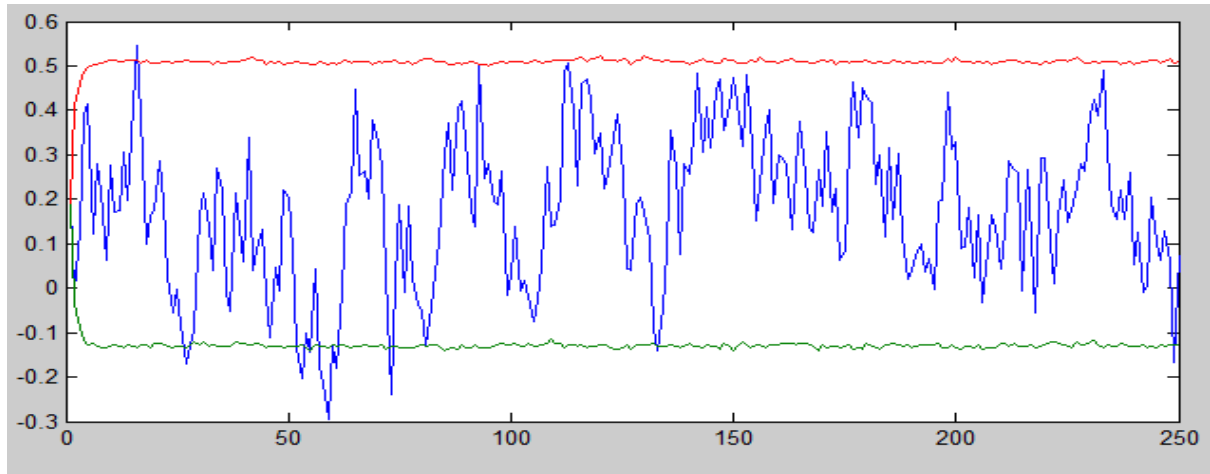
The estimates they obtain are as follows (with some rounding because the exact parameters are unimportant for the argument below):

$\lambda = 16$	$\mu = 0.19$	$\sigma = 1.1$
----------------	--------------	----------------

I then used these parameters as input to the simulation. What we are saying is, assuming the process *does* have those parameters, could our estimation technique have discovered this?

Results

A sample plot of the process, showing empirical 95% confidence intervals, is below. We see deviations from +50% to -20%, roughly in line with the empirical plot above. However, subjectively, the simulation plot mean-reverts faster than the empirical plot above.



The output of the simulation and estimation, with 10,000 draws from an O-U process with the above true parameters, is as follows:

Known Parameter	λ		μ		σ	
Actual	16		0.19		1.10	
Estimation Technique	$mean(\hat{\lambda})$	$sd(\hat{\lambda})$	$mean(\hat{\mu})$	$sd(\hat{\mu})$	$mean(\hat{\sigma})$	$sd(\hat{\sigma})$
Maximum Likelihood	17.06	3.237	0.1902	0.0307	1.104	0.057
Maximum Likelihood with Jack-knife	15.98	3.403	0.1902	0.0307	1.100	0.058
Least Squares (Simple Regression)	14.38	2.271	0.1902	0.0307	0.942	0.042
Least Squares	17.06	3.237	0.1902	0.0307	1.106	0.058

Observations / Comments

Estimates of μ , the long-term mean and σ , the process volatility, are very good whatever estimation technique we use (with the exception of the least square using the naïve regression), both accurate and with low standard deviation.

The least-squares simple regression is inaccurate for $\hat{\sigma}$, and also least accurate for $\hat{\lambda}$.

Estimates of λ are relatively poor. Even with 10,000 draws, we get a mean estimate quite far from the actual value, and with a wide deviation.

The mean of the jack-knife estimate $\hat{\lambda}$ is very close to the true λ , albeit at the expense of greater standard deviation.

It would initially appear that the 'jack-knife' is our saviour, giving very accurate (although quite widely dispersed) estimates. However, if we plot a histogram of the $\hat{\lambda}$ for each estimation method,

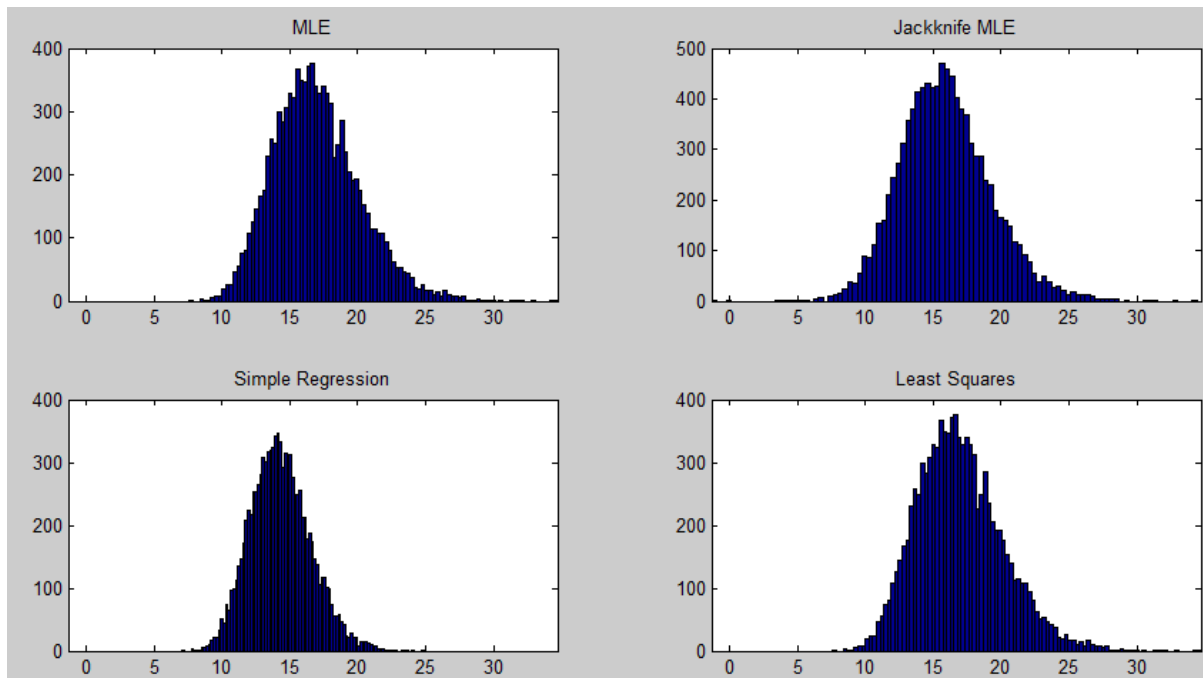


Figure 2 - Histogram of Mean Reversion Parameter Estimates based on Simulation

it is clear that the Jack-knife is no better than 'raw' MLE, it is just that MLE has a skewed with a fatter right tail. This biases the mean estimate upwards. Thus the jack-knife would appear no to help us in real life – we can't sample our process thousands of time, we only have a single empirical historical price process. No preference is observed between ML and LS. The 'simple' LS estimate has a lower standard deviation, but the estimate of λ is significantly too low.

It is possible (if we are unlucky) for our estimates of λ to be wildly out. A true λ of 16 can easily result in $\hat{\lambda} < 10$ or $\hat{\lambda} > 23$.

Improving Estimate of Mean Reversion

Let us now investigate what can give us a more accurate estimate for $\hat{\lambda}$, i.e. with lower standard deviation. If we increase observations to daily, i.e. $T=5$, $\Delta t=1/250$, $\lambda=16$, $\mu=0.19$, $\sigma=1.1$, #observations ≈ 1250 .

Known Parameter	λ		μ		σ	
Actual	16		0.19		1.10	
Estimation Technique	$mean(\hat{\lambda})$	$sd(\hat{\lambda})$	$mean(\hat{\mu})$	$sd(\hat{\mu})$	$mean(\hat{\sigma})$	$sd(\hat{\sigma})$
Maximum Likelihood	16.82	2.761	0.1900	0.0310	1.101	0.023
Maximum Likelihood with Jack-knife	15.95	2.902	0.1900	0.0310	1.100	0.023
Least Squares (Simple Regression)	16.26	2.574	0.1900	0.0310	1.065	0.021
Least Squares	16.82	2.761	0.1900	0.0310	1.101	0.023

Even with a 5-fold increase in samples, our estimates are barely improved, and the standard deviations of the estimate have barely fallen. Conclusion : estimating λ can never be a precise 'science'.

Behaviour with Weak Mean Reversion

We now return to the weekly Gibson-Schwartz model parameters, except we assume a much weaker mean reversion. For example, if we reduce λ to 2, and return to the Original Gibson-Schwartz parameters of $T=5$, $\Delta t=1/50$, $\mu=0.19$, $\sigma=1.1$, #observations ≈ 250 , we get the following:

Known Parameter	λ		μ		σ	
Actual	2		0.19		1.10	
Estimation Technique	$mean(\hat{\lambda})$	$sd(\hat{\lambda})$	$mean(\hat{\mu})$	$sd(\hat{\mu})$	$mean(\hat{\sigma})$	$sd(\hat{\sigma})$
Maximum Likelihood	2.949	1.295	0.1904	0.2484	1.105	0.051
Maximum Likelihood with Jack-knife	1.906	1.688	0.1514	2.1899	1.101	0.051
Least Squares (Simple Regression)	2.848	1.204	0.1904	0.2484	1.075	0.048
Least Squares	2.949	1.295	0.1904	0.2484	1.107	0.050

A sample draw of the process now looks like Figure 3 below, showing weaker reversion and, as expected, wider confidence intervals.

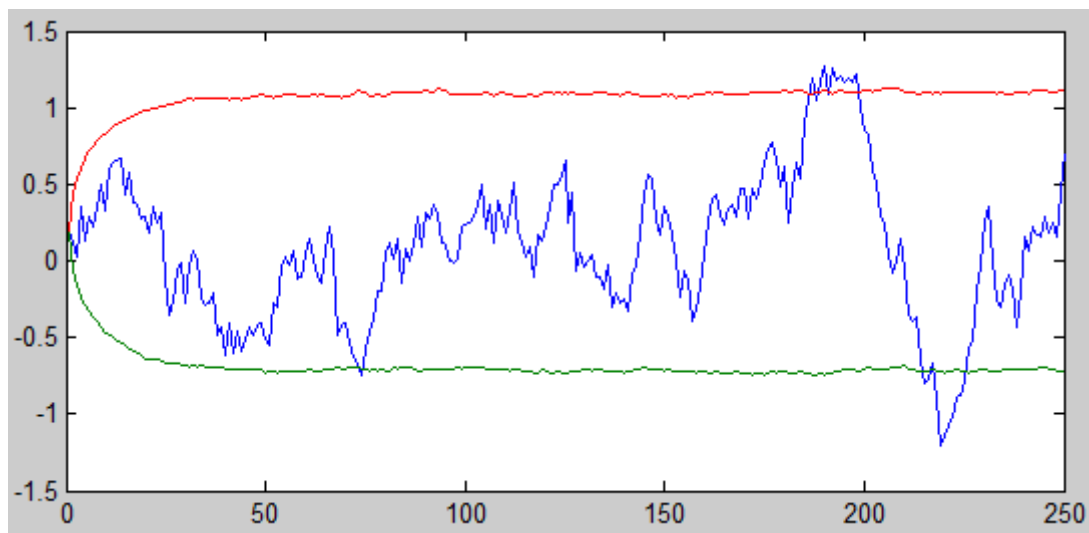


Figure 3 - Draw of the O-U process with weaker mean reversion.

We can see that we still have relatively poor estimates of λ , but in addition our estimate of μ has also seriously deteriorated for the jack-knife calculation. Since μ is known to have no bias, we should actually not use the jack-knife procedure to estimate μ or σ , but only λ , even if we do choose to do jack-knife at all. Viewing the distributions of the estimates (Figure 4), we see that the jack-knife estimates now include a significant chance of estimating a negative λ , i.e. an unstable

process, despite λ being positive. We then have to choose whether to discard (or set to 0) any $\lambda < 0$ estimates. Although the jack-knife estimate is unbiased, with a symmetrical distribution, it introduces more problems than it solves. We see that the other methods are broadly of a similar shape.

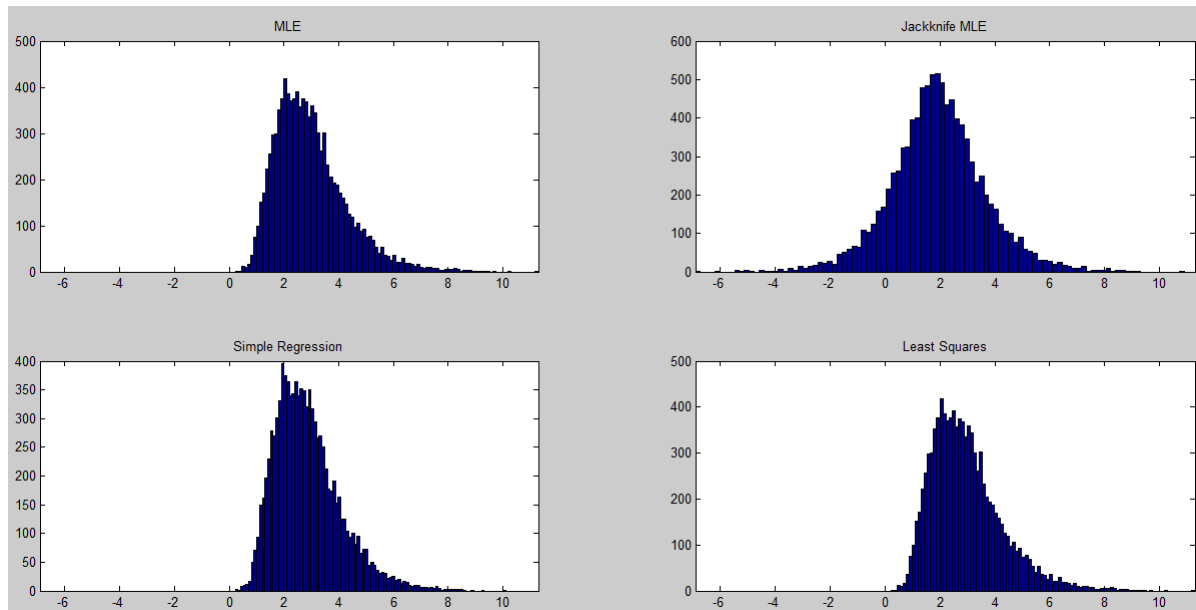


Figure 4 - Distributions of estimates with weaker mean reversion.

Timings

Approximate timings for the simulation and estimation techniques on my computer (quad core, 2.4GHz, 2008) are as follows, all for 10,000 simulations:

Simulate Paths	101 seconds
Estimate by Maximum Likelihood	1.2 seconds
Estimate by Maximum Likelihood + Jackknife	1.9 seconds
Estimate by Simple, Naïve Regression	36 seconds
Estimate by Accurate Regression	32 seconds

Clearly, more work should be done to vectorise the simulation, and the ML methods are preferred based on speed.

Conclusion

Estimation of a mean-reverting O-U process using either Maximum Likelihood or Least Squares gives accurate estimates of the mean μ and volatility σ , but only poor estimates of the mean reversion parameter λ . Taking a larger sample does not solve the problem, and nor do techniques such as the 'jackknife', which introduces more problems than it solves. Care should be taken to use exact

simulation and estimation formulae which work for even large Δt values, rather than the naïve formulae.

```

function [ output_args ] = MLE_Test( )

%% Create an Ornstein-Uhlenbeck mean reverting process with know
%% parameters, then try to estimate those same parameters using different
%% techniques.

%% License
% Copyright 2010, William Smith, CommodityModels.com . All rights reserved.
%
% Redistribution and use in source and binary forms, with or without modification, are
% permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright notice, this list of
% conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright notice, this list
% of conditions and the following disclaimer in the documentation and/or other materials
% provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER, WILLIAM SMITH ``AS IS'' AND ANY EXPRESS
% OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
% MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
% THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
% SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
% OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
% HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
% OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
% SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

%% Known parameters. Based on the parameters estimated for the mean
%% reverting convenience yield in the original 'Gibson Schwartz' model:
%% Gibson, R., and E. S. Schwartz. 1990. "Stochastic convenience yield and the pricing of
%% oil contingent claims" , Journal of Finance: 959-976.

S0 = 0.19;
mu = 0.19;
sigma = 1.1;
lambda =16;
deltat = 1/50;
T = 5;

%% Run and time the simulations and estimations.
fprintf('O-U\n');

simulations = 10000;
mu_hat = zeros(4,simulations);
sigma_hat = zeros(4,simulations);
lambda_hat= zeros(4,simulations);

timers = zeros(5,1);

% Run many simulations, re-estimate parameters in different ways.
% Saving each simulation is memory-intensive, but allows us to plot
% empirical confidence intervals.
for i=1:simulations
    tic;
    S(i,:) = SimulateOrnsteinUhlenbeck(S0, mu, sigma, lambda, deltat, T);
    timers(1) = timers(1) + toc;
    tic ;
    [ mu_hat(1,i), sigma_hat(1,i), lambda_hat(1,i) ] = ...
        CalibrateOrnsteinUhlenbeckMaxLikelihood (S(i,:), deltat, T);
    timers(2) = timers(2) + toc;
    tic;
    [ mu_hat(2,i), sigma_hat(2,i), lambda_hat(2,i) ] = ...
        CalibrateOrnsteinUhlenbeckMaxLikelihoodJackknife(S(i,:), deltat, T);
    timers(3) = timers(3) + toc;
    tic;
    [ mu_hat(3,i), sigma_hat(3,i), lambda_hat(3,i) ] = ...
        CalibrateOrnsteinUhlenbeckRegress (S(i,:), deltat, T);

```

```

timers(4) = timers(4) + toc;
tic;
[ mu_hat(4,i), sigma_hat(4,i), lambda_hat(4,i) ] = ...
    CalibrateOrnsteinUhlenbeckLeastSquares      (S(i,:), deltat, T);
timers(5) = timers(5) + toc;
end

% Plot two sample paths, the first two, just so we can
% visualise it.
subplot(3,2,1);
plot( 1:size(S,2) , S(1,:), ...
      1:size(S,2) , quantile(S,0.05,1), ...
      1:size(S,2) , quantile(S,0.95,1));
subplot(3,2,2);
plot( 1:size(S,2) , S(2,:), ...
      1:size(S,2) , quantile(S,0.05,1), ...
      1:size(S,2) , quantile(S,0.95,1));

fprintf('Simulation : %fs\n', timers(1));

fprintf('MLE : %fs\n', timers(2));
[ mean(mu_hat(1,:))      std(mu_hat(1,:))      ] %#ok<*NOPRT>
[ mean(sigma_hat(1,:))  std(sigma_hat(1,:))  ]
[ mean(lambda_hat(1,:)) std(lambda_hat(1,:)) ]

fprintf('Jackknife MLE : %fs\n', timers(3));
[ mean(mu_hat(2,:))      std(mu_hat(2,:))      ]
[ mean(sigma_hat(2,:))  std(sigma_hat(2,:))  ]
[ mean(lambda_hat(2,:)) std(lambda_hat(2,:)) ]

fprintf('Simple Regression : %fs\n', timers(4));
[ mean(mu_hat(3,:))      std(mu_hat(3,:))      ]
[ mean(sigma_hat(3,:))  std(sigma_hat(3,:))  ]
[ mean(lambda_hat(3,:)) std(lambda_hat(3,:)) ]

fprintf('Least Squares : %fs\n', timers(5));
[ mean(mu_hat(4,:))      std(mu_hat(4,:))      ]
[ mean(sigma_hat(4,:))  std(sigma_hat(4,:))  ]
[ mean(lambda_hat(4,:)) std(lambda_hat(4,:)) ]

% Intelligent axes.
lambdamin=min(min(lambda_hat));
lambdamax=max(max(lambda_hat));

subplot(3,2,3);
hist(lambda_hat(1,:),100);
xlim([lambdamin lambdamax]);
title('MLE');

subplot(3,2,4);
hist(lambda_hat(2,:),100);
xlim([lambdamin lambdamax]);
title('Jackknife MLE');

subplot(3,2,5);
hist(lambda_hat(3,:),100);
xlim([lambdamin lambdamax]);
title('Simple Regression');

subplot(3,2,6);
hist(lambda_hat(4,:),100);
xlim([lambdamin lambdamax]);
title('Least Squares');

end

```

References

- Cox, John C., Jonathan E. Ingersoll, and Stephen A. Ross. 1985. A Theory of the Term Structure of Interest Rates. *Econometrica* 53, no. 2 (March): 385-407.
- Gibson, R., and E. S. Schwartz. 1990. Stochastic convenience yield and the pricing of oil contingent claims. *Journal of Finance*: 959-976.
- Gillespie, D. T. 1996. Exact numerical simulation of the Ornstein-Uhlenbeck process and its integral. *Physical review E* 54, no. 2: 2084–2091.
- Ornstein, L. S., and G. E. Uhlenbeck. 1930. On the Theory of the Brownian Motion. *Physical Review* 36, no. 5: 823. doi:10.1103/PhysRev.36.823.
- Phillips, Peter C. B., and Jun Yu. 2005. Jackknifing Bond Option Prices. *The Review of Financial Studies* 18, no. 2 (Summer): 707-742.
- Vasicek, Oldrich. 1977. An equilibrium characterization of the term structure. *Journal of Financial Economics* 5, no. 2 (November): 177-188. doi:10.1016/0304-405X(77)90016-2.
- Yu, Jun. 2009. Bias in the Estimation of the Mean Reversion Parameter in Continuous Time Model. September. <http://www.mysmu.edu/faculty/yujun/Research/bias02.pdf>.