Estimate running variance using a particle filter

Assume we live in a world where the observed variable (luminance, say) is iid Gaussian at every time point with a variance determined by σ. σ is first-order Markov.

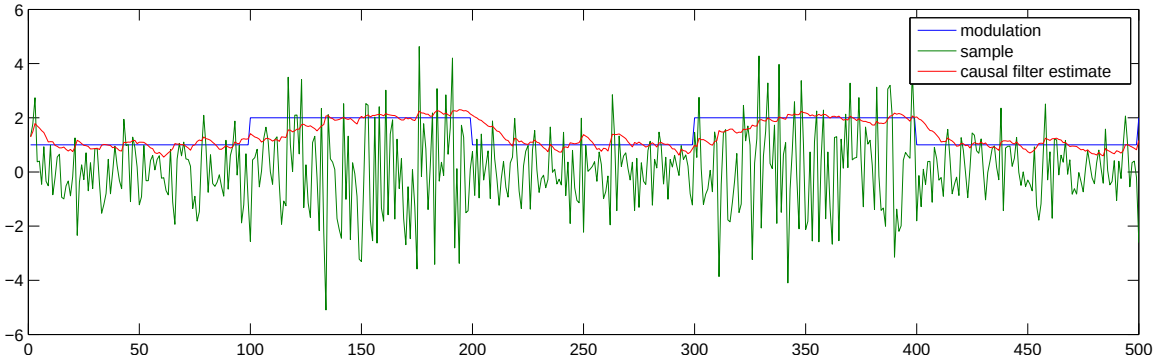
Setup: $p(x_t|x_{t-1}) = N(x_t|x_{t-1}, \sigma^2)$ $p(z_t) = N(0,1)$ $p(y_t|x_t) = x_t z_t$

We can estimate the variance causally using a stochastic variant of the optimal filter. Here I am using a vanilla version of a particle filter.

View the result of the particle filter acting on an input composed of step function variations in sigma times a random modulation

```
y1 = floor(mod((1:500)/100,2))'+1;
z1 = y1.*randn(size(y1));
xhat = runningVariance(z1,.1);

clf;
plot([y1,z1,xhat]);
legend('modulation','sample','causal filter estimate');
set(gcf,'Position',[5 408 901 251]);
```

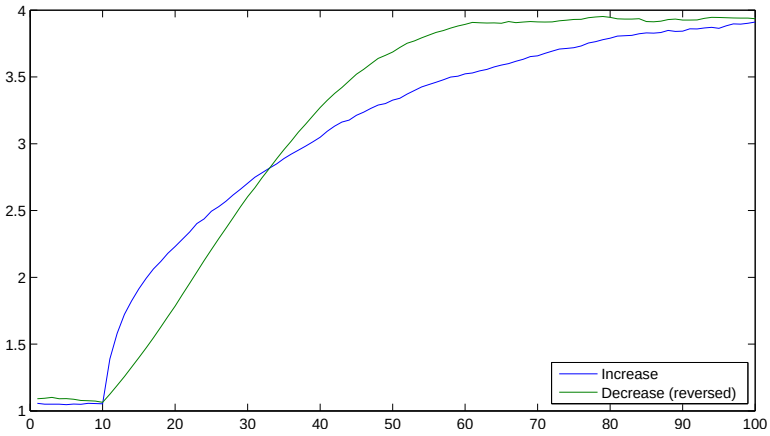
```
Iteration 500
```



Show the estimate of the variance right after a step change

```
y = 3*floor(mod((1:50000)/100,2))'+1;
z = y.*randn(size(y));
xhat = runningVariance(z,.1);

xhat = reshape(xhat(90:end-11),100,499);
m1 = mean(abs(xhat(:,1:2:end)),2);
m2 = mean(abs(xhat(:,2:2:end)),2);
```

```
clf;
plot([m1,5-m2]);
legend('Increase','Decrease (reversed)');
legend('Location','SouthEast');
set(gcf,'Position',[5 313 658 346]);
```



Analysis

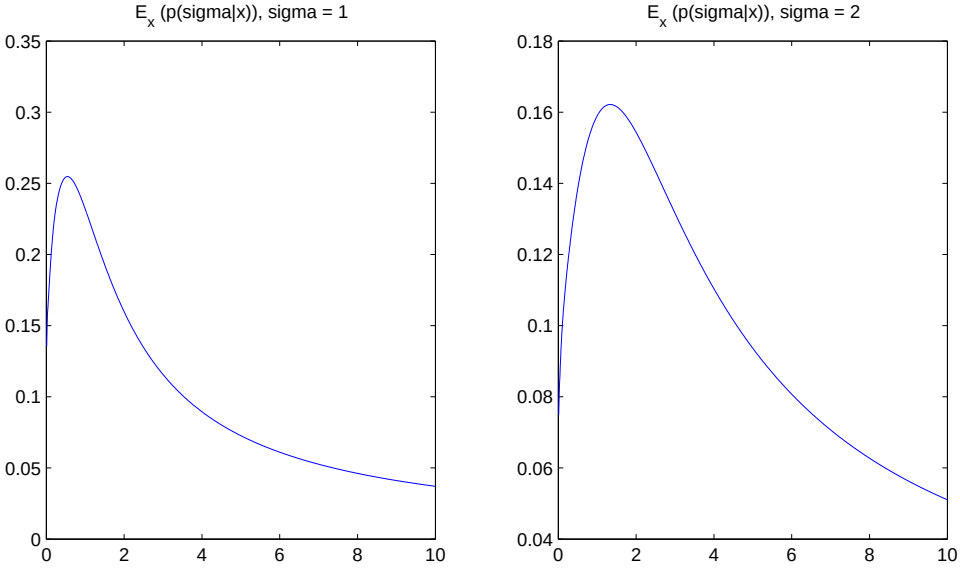
After an increase in the variance, the estimate of the variance increases rapidly at first, while after a decrease the change is slower. This is because after an increase in the variance, the next samples are outliers under the current estimate, but this is not the case after a decrease.

On the other hand, the estimate takes longer to asymptote after an increase. This is a little harder to explain, let's try and illustrate.

Compute $\mathbf{E}_x p(\sigma|x)$ for $\sigma = 1$.

```
ns = 10000;
xs = randn(ns,1);
sigmas = .01:.01:10;
ps = bsxfun(@times,1./sigmas,exp(-1/2.*bsxfun(@times,1./sigmas.^2,xs.^2)));
ps = bsxfun(@times,ps,1./sum(ps,2));
ps = sum(ps,1);
clf;
subplot(1,2,1);
plot(sigmas,ps./sum(ps)*100);
title('E_x (p(sigma|x)), sigma = 1');
```

```
xs = randn(ns,1)*2;

ps = bsxfun(@times,1./sigmas,exp(-1/2.*bsxfun(@times,1./sigmas.^2,xs.^2)));
ps = bsxfun(@times,ps,1./sum(ps,2));
ps = sum(ps,1);

subplot(1,2,2);
plot(sigmas,ps./sum(ps)*100);
title('E_x (p(sigma|x)), sigma = 2');
```



Analysis

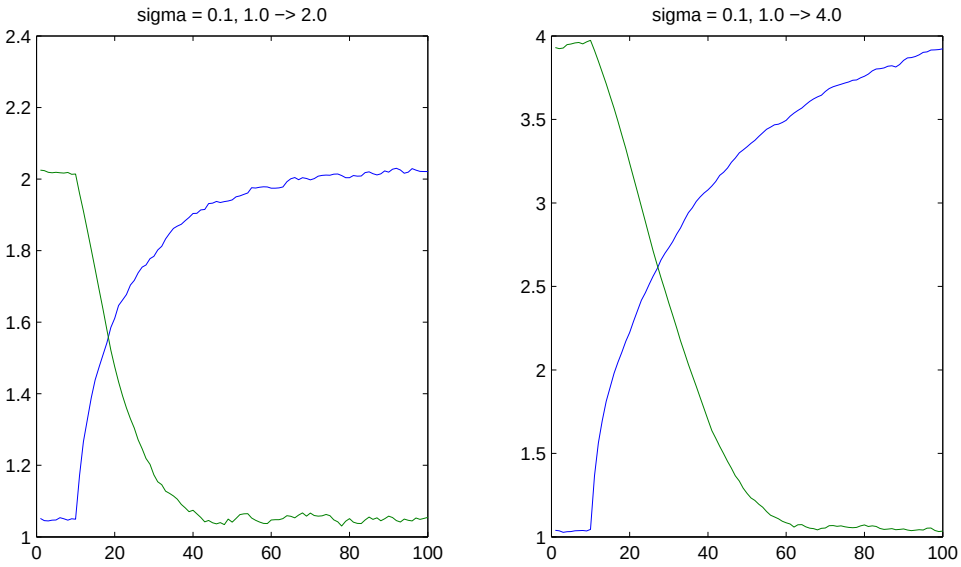
Larger values are compatible with a wider range of σ, which means that each large value is paradoxically less informative about σ. Hence, it takes longer for the estimate to asymptote. See DeWeese & Zador (1998) for more details.

Scaling properties

The filter associated with $\alpha\sigma$ is such that $f(x, \alpha\sigma) = y\alpha$. The filter is otherwise rather nonlinear and has unintuitive scaling properties. Let's compare its action for $\sigma : 1 \to 2$ and $\sigma : 1 \to 4$.

```
paramsigma = .1;
paramlevel = [1,2;
              1,4];
```

```
clf;
simulateFilterForParams(paramsigma,paramlevel);
```



Analysis

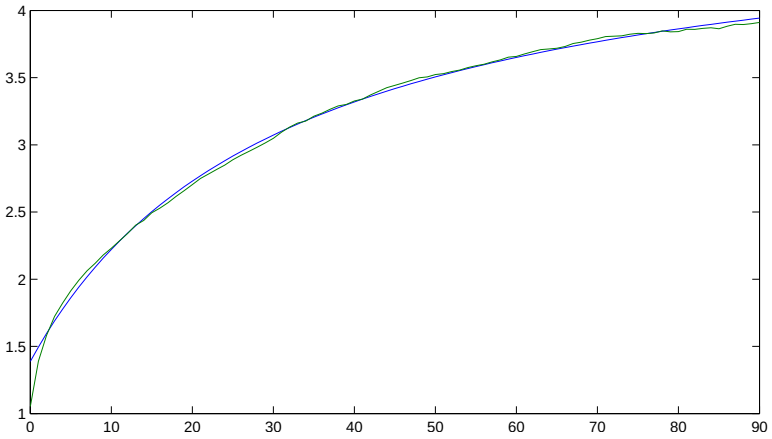
The time it takes for the filter to convergence increases with increasing step size. This is not the case for a running mean estimator. Similarly, uniformily scaling the input changes the properties of the estimator, so $f(\alpha y) \neq \alpha f(y)$

Parametric form

The response to a step increase in variance is well modeled by a $1/(1+x)$ function

```
clf;
thefun = @(p,x) p(1)./(1+p(3)*x) + p(2);
p0 = [-1,4,.2];
clf;
[ps mse] = lsqcurvefit(thefun,p0,1:90,m1(11:end)')
plot(-9:90,[thefun(ps,-9:90)',m1]);
xlim([0,90]);
```

```
Local minimum possible.

lsqcurvefit stopped because the final change in the sum of squares relative to
its initial value is less than the default value of the function tolerance.



ps =

   -3.4446    4.8330    0.0319


mse =

    0.0539
```

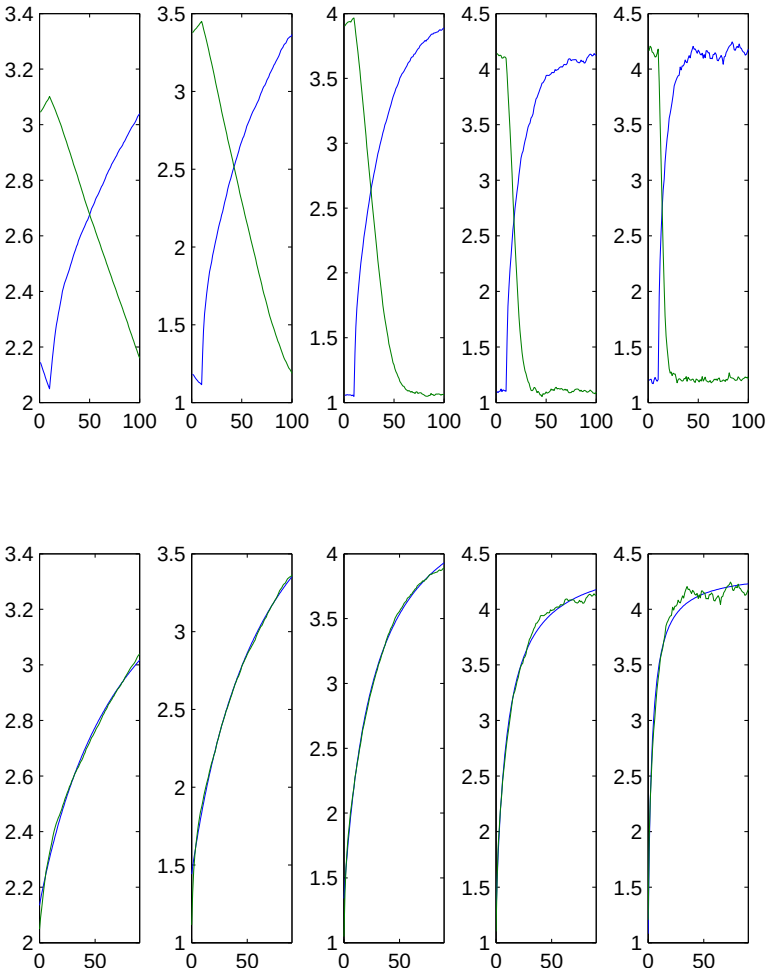


Another scaling property

Using this, we can look at how the filter scales in time as function of σ

```
paramsigma = [.025,.05,.1,.2,.4];
paramlevel = [1,4];

A = simulateFilterForParams(paramsigma,paramlevel);

clf
for ii = 1:size(A,1)
    for jj = 1:1
        subplot(2,size(A,1),ii);
```

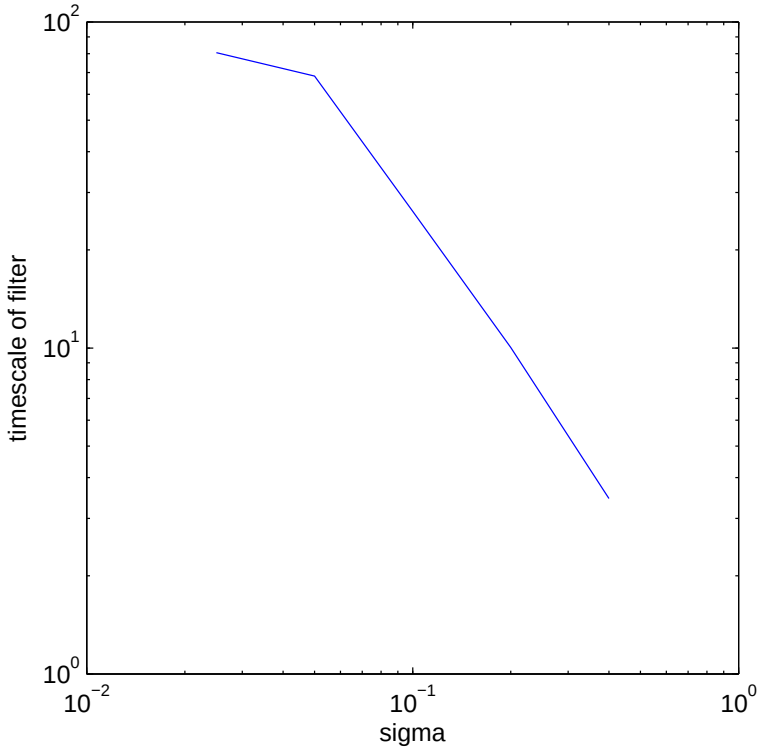
```
        plot(squeeze(A(ii,jj,:,:))');

        subplot(2,size(A,1),ii+size(A,1));
        a = squeeze(A(ii,jj,1,:));
        %fit exponential
        p0 = [-3,3,.1];
        [ps mse] = lsqcurvefit(thefun,p0,1:90,a(11:100)');
        plot(-9:90,[thefun(ps,-9:90)',a]);
        xlim([0,90]);
        pss(ii) = ps(3);
    end
end

set(gcf,'Position',[5 84 471 575]);
```



Analysis

The timescale of the filter if we increase the variance 2-fold decreases by a factor of more than two fold. This is a really nonlinear filter.

```
clf;
loglog(paramsigma,1./pss);
xlabel('sigma');
axis square;
ylabel('timescale of filter')

set(gcf,'Position',[5 324 590 335]);
```


10^2
10^1
10^0
timescale of filter
10^{-2}
10^{-1}
10^0
sigma