**Time Series Forecasting**

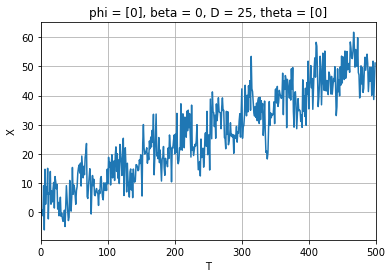
Let’s look at a few examples.

**Example 1**

Say we have the following series:



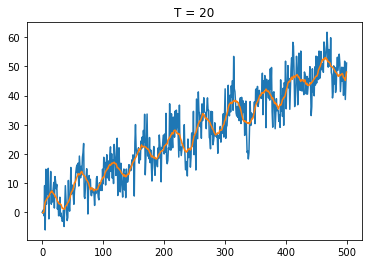
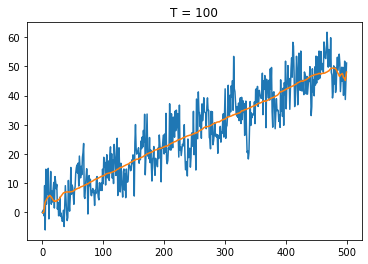
D is the variance of the white noise fluctuations. So amplitude of the fluctuations is basically the same as the amplitude of the seasonality. Looks like this:



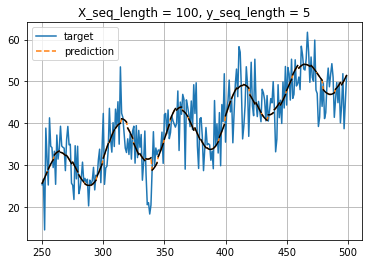
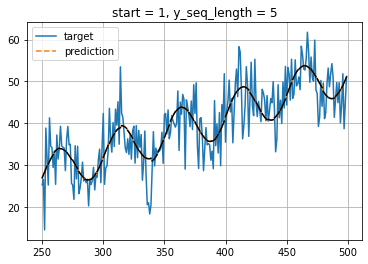
This series has no correlations, so it’s purely a regression problem. We wouldn’t necessarily know that a priori, though.

**Regression**

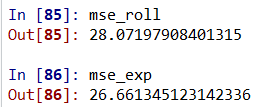
We can attempt a regression. It looks like a linear function plus a sinusoidal. But we could allow for a power law plus sinusoid. Let’s do a regression on xn = an+ b + Asin(ωn). Let’s do a rolling average to estimate the period and power.

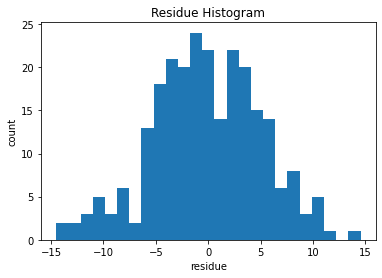
Might guess a period of around T = 50, so ω = 2π/50. Also the amplitude would be around A = 3. The trend looks pretty linear. Let’s say, a = 0.1, and b = 0. We can do rolling and expanding window regression. And say, we’re comitted to predicting 5 steps out. Then we have:

These are the mean square errors (technically, the average of the mse’s for each fold, i.e., each set of 5 predictions),



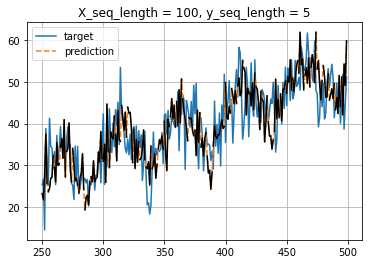
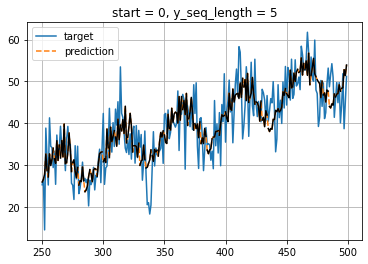
These are pretty close to the best we can do, as the actual mse should be 25 (since D = 25 up above). A histogram of the errors for the expanding window guy is:



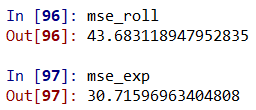
Which looks fairly Gaussian.

**Exponential Smoothing**

Let’s try exponential smoothing. Here’s a rolling window, and expanding window side by side, with automatically chosen parameters.

Here’s the mean square errors,



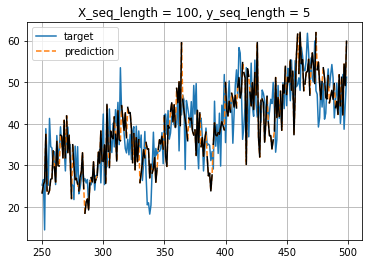
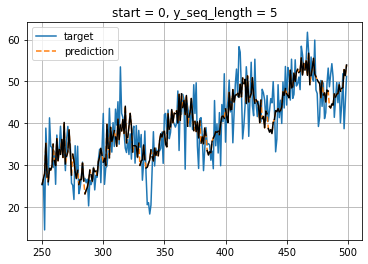
The expanding window guy is pretty good, but the regression is still the best.

**SARIMA**

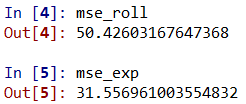
There is clearly a seasonality to the data, and there might be correlations between points beyond white noise. We can try a SARIMA model. Consider taking the d = 1, D = 1 difference,



So we should be able to model this with: p = 0, d = 1, q = 1, P = 0, D = 1, Q = 1, s = 50. And we get:

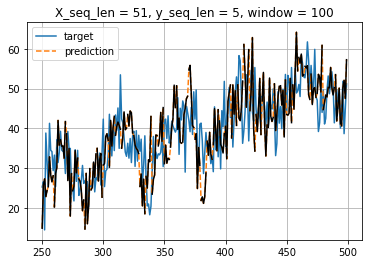
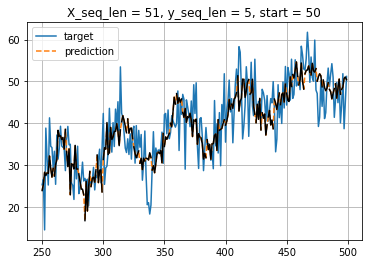
Errors are:



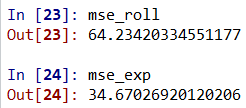
So this is the worst yet. Not sure why as an exact solution of the difference equation should yield the exact result. Maybe it has trouble finding the solution appropriate to the initial conditions, since the difference equation is an effectively 51st order equation? Not sure. But note then, that SARIMA won’t necessarily get all power law/sinusoidal behaviors.

**DE Forecaster**

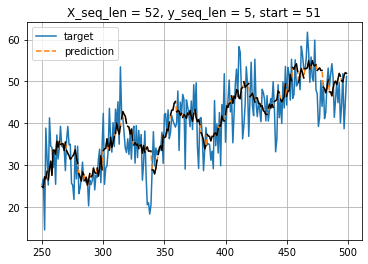
Decided to try a general linear difference equation, well, AR difference equation, with n = 50 regressive terms (since period is 50). It looks better than what I got with just n = 2 regressive terms. Got this:

Results are:



So that’s not too good either. The SARIMA does better, as we’d expect, really, since it can focus on optimizing just the parameters that *actually* matter. Kind of relatedly, those plots were with α = 0.1. If I increase the L2 regularization parameter to α = 5, I get:

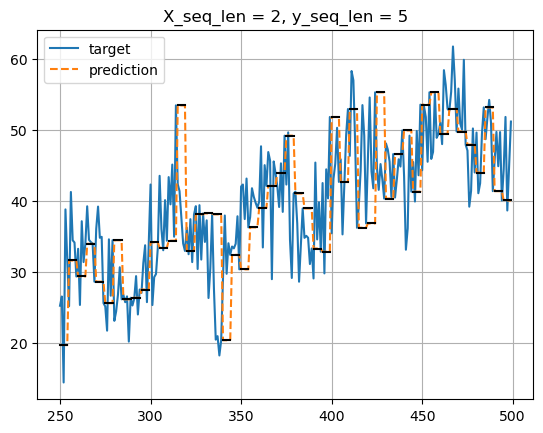




which is on par with SARIMA. This kind of makes sense in because a higher regularization will drop many of the terms/coefficients which don’t really matter, and thus prevent overfitting. And when I print out the coefficients (all 50 of them), the non-zero ones are indeed clustered at the beginning and at the end. When I slice out the first three and last three columns, however, and rerun the regression, I don’t get as good results. Perhaps this is because the real difference equation has MA terms (as saw in the SARIMA analysis) which I can’t model with this guy.

**DES Forecaster**

Always worthwhile just trying a random walk model. I got this:

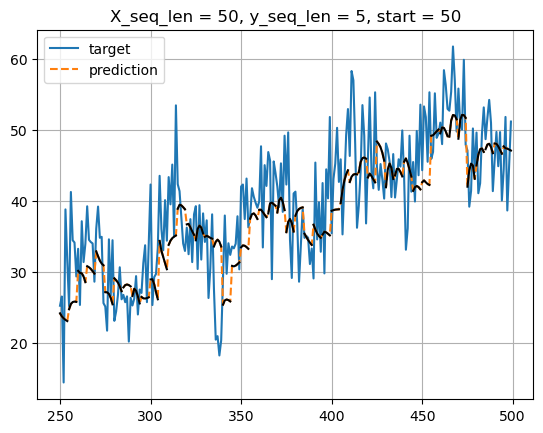




So obviously got much better fits elsewhere, which suggests this is not capturing the real behavior.

**LSTM Forecaster**

Can also try a machine learning model. The more-or-less best fit I could find was a one with a sequence length of 50, using an expanding window. Can see that there is somethting of a sin wave going on, but fit isn’t as a good as the classical models.





I made an LSTM Encoder-Decoder with unit\_size = 5, and just a single layer. Doing more or less didn’t seem to help, and often made it much worse. I used a step size of α = 0.005, and ran it for 250 epochs. Generally, speaking the losses would level out after a while; definitely had a hard time getting any better.

**Upshot**

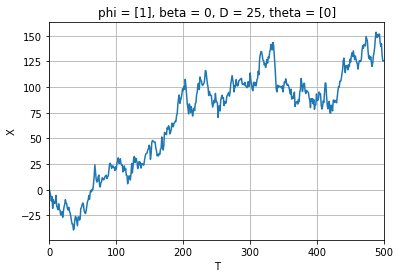
Well, the regression does best, as I’d hoped, honestly. So good results so far.

**Example 2**

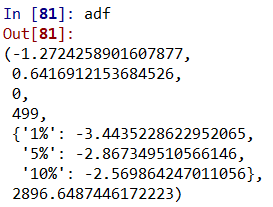
Let’s consider a random walk model:



Looks like this, in this case:

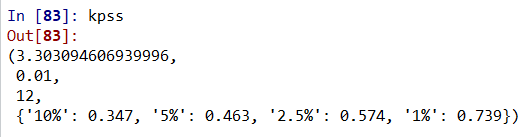


Let’s look at the adfuller and kpss tests. We find:



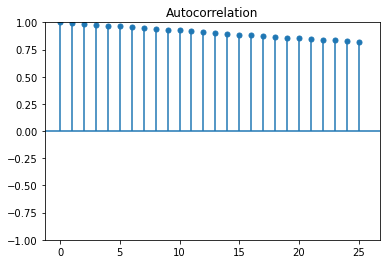
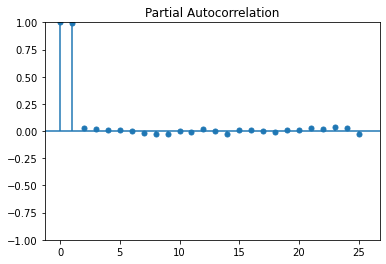


This says the probability it’s un-stationary is about 64%.



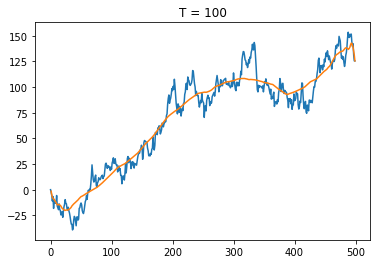


This says the probability it’s stationary is about 1%. So that’s all consistent with what we know, since a random walk is unstationary. Even though it’s time-dependent, which makes doing the correlation stuff not useful, we’ll do it anyway. The partial autocorrelation suggests (correctly as it turns out) that xn-1 alone is in this DE.

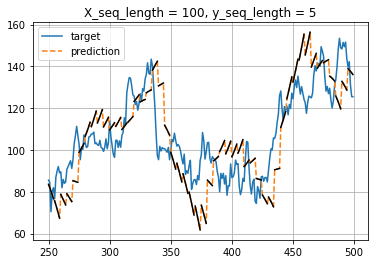
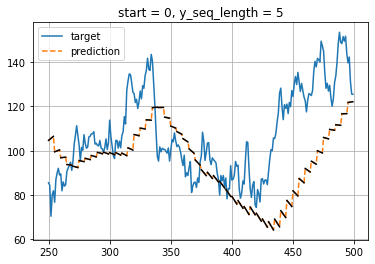
 

**Regression**

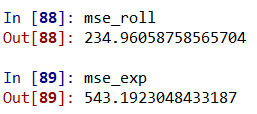
Just going through the motions, we can attempt a regression. Let’s look at a rolling average:



This doesn’t conform to any obvious curve. Nonetheless, let’s try a general cubic polynomial.

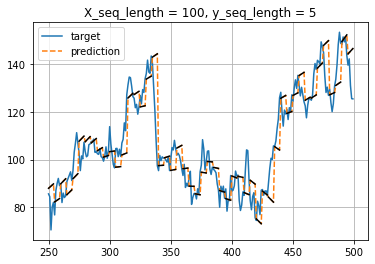
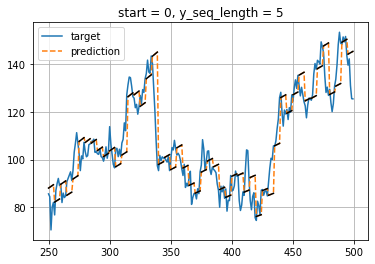
Neither looks very good. Errors are:



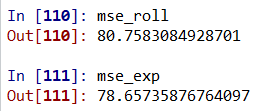
Which looks fairly Gaussian.

**Exponential Smoothing**

Let’s try exponential smoothing. Here’s a rolling window, and expanding window side by side, with automatically chosen parameters.

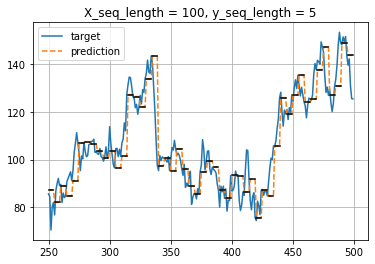
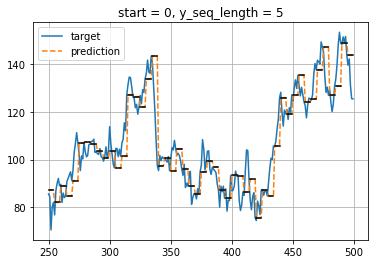
Here’s the mean square errors,



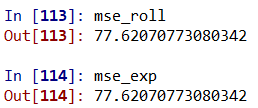
This looks pretty close to a pure random walk forecast. So this is a pretty good result!

**ARIMA**

Since our series is literally xn+1 = xn + ΔWn, ARIMA should perfectly capture the data with d = 1, p = 0, q = 0. Let’s try.

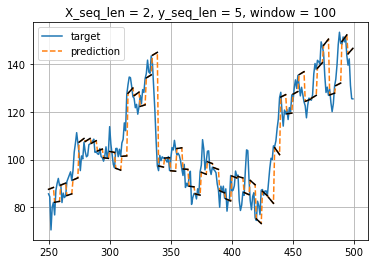
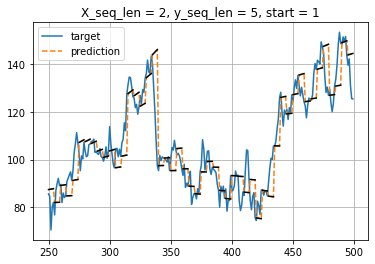
Errors are:



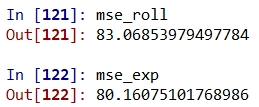
This is the best so far. And it should be the best possible. But we’ll see.

**DE Forecaster**

Decided to try a general linear difference equation, well, AR difference equation, with n = 1 regressive terms, i.e., a model of the sort xn+1 = φ1xn + βΔt + ΔWn. When I have L2 regularization term α = 0, I get a non-zero drift term βΔt. When I set α = 1, the drift term goes away and φ1 -> 1, so practically same as ARIMA, naturally.

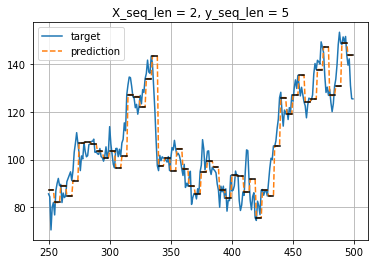
Results are:



Interestingly, letting n = 10 (so including 10 regressive terms) in the difference equation, doesn’t change the shape of the fit very much.

**DES Forecaster**

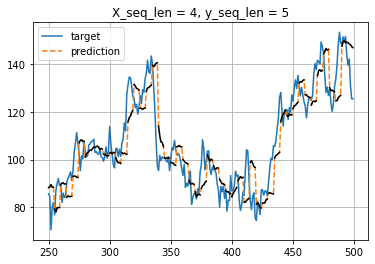
The DE Forecaster uses regularized linear regression to try to find the best parameers of an AR model. This isn’t perhaps the most accurate approach. Let’s try an explicit random walk model, using the DES Forecaster.



We get:



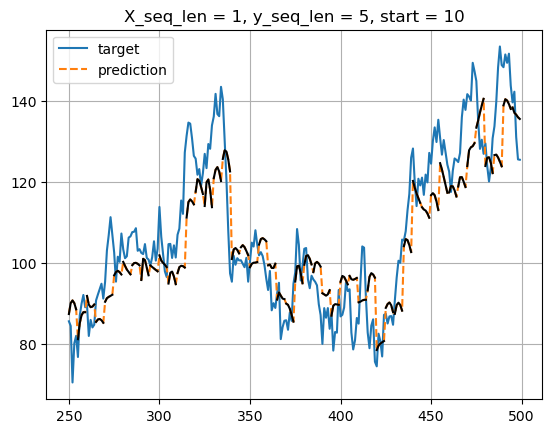
This is what ARIMA got too, which is reassuring. What if we tried a kind of rolling average approach. I’ll average the previous 4 terms, i.e., xn = (1/4)(xn-1 + xn-2 + xn-3 + xn-4)



Not as good,

**LSTM Forecaster**

Going the machine learning route again. I used X\_seq\_length = 1, as befits a random walk and got the best results. But did an expanding window, rather than a rolling window; that seems to work best b/c, I suppose, the process is the same at all times and so making the training range as large as possible is the best practice.





Even so, the results are worse than the classical results.

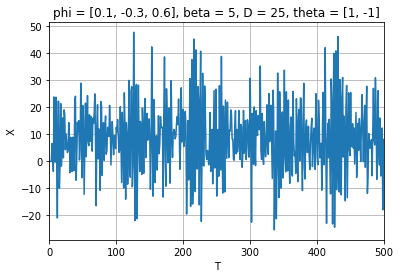
**Upshot**

So we find the random walk model does the best, as we would hope. Everything is looking good so far!

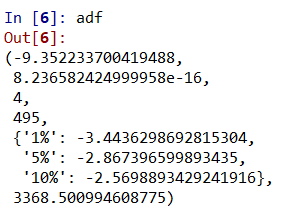
**Example 3**

Here’s an ARMA process. And the white noise has variance D = 25.



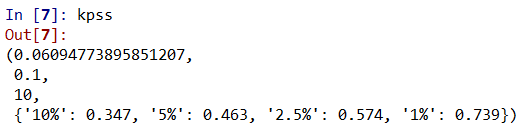


Obviously an ARMA process stochastic model should fit it best, but let’s check out the other options. Let’s look at the adfuller and kpss tests. We find:



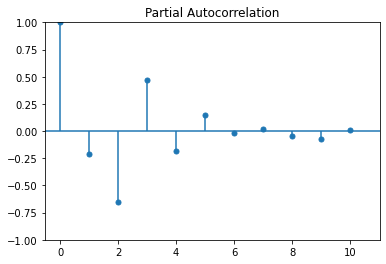


This says the probability it’s un-stationary is very negligible.



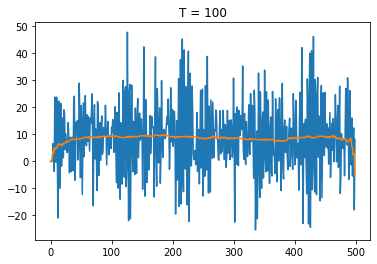


This says the probability it’s stationary is about 10%. Huh. Well it is stationary. The partial autocorrelation suggests there could be up to around n = 5 autoregressive terms.

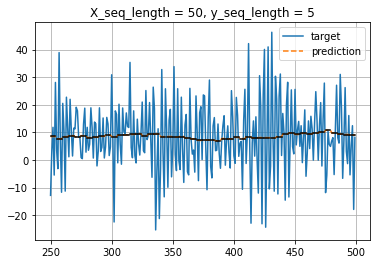
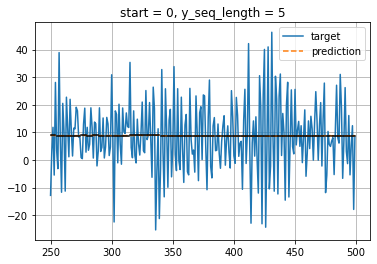


**Regression**

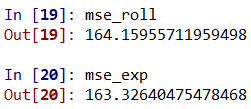
Just going through the motions, we can attempt a regression. Let’s look at a rolling average:



This suggests that the best regression curve is probably just x = 10. But we’ll do a formal fit of a constant. The first graph is basically my rolling average again. The second graph is a global average.

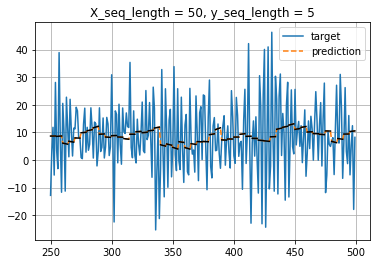
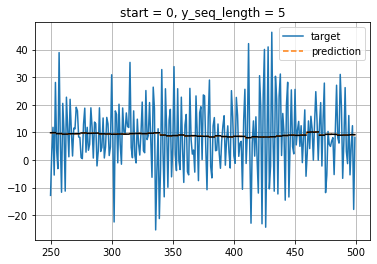
Errors are:



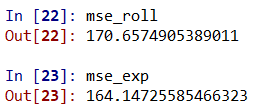
It’s not clear, a priori, that one can do better than this.

**Exponential Smoothing**

Let’s try exponential smoothing. Here’s a rolling window, and expanding window side by side, with automatically chosen parameters.

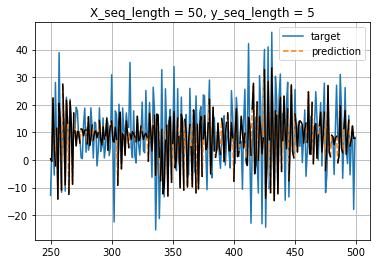
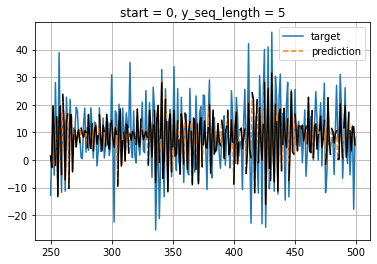
Here’s the mean square errors,



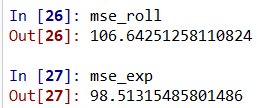
The last one is kind of like our global average. Global (expanding window) averaging works best so far.

**ARIMA**

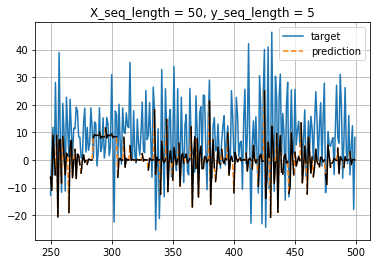
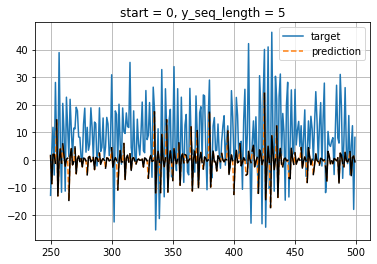
Of course ARIMA should do the trick. Let’s use p = 3, q = 2, d = 0.

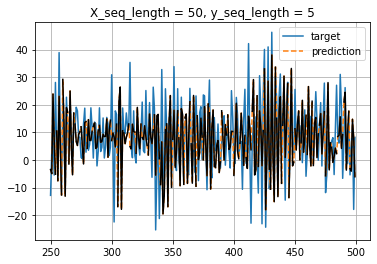
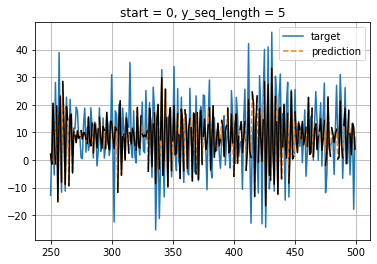
Errors are:



These are a lot better! What if we change the parameters away from what they should be. Let p = 2, q = 1.

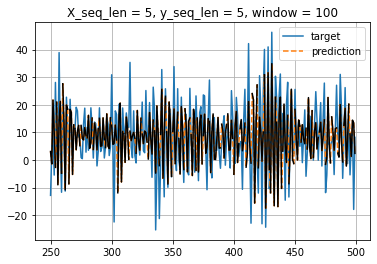
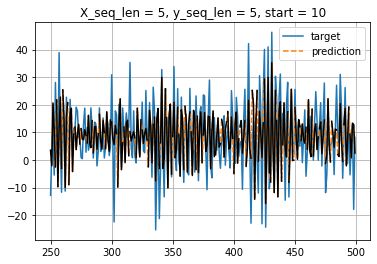
Obviously not good. There were some convergence issues, though. What about p = 5, q = 5?

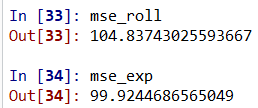
These are close to as good as p = 3, q = 2, but their mse’s are bit higher. So a grid search would probably evince the proper p, q value.

**DE Forecaster**

Decided to try a general linear difference equation, well, AR difference equation, with n = 4 regressive terms, i.e., a model of the sort xn = φ1xn-1 + φ2xn-2 + φ3xn-3 + φ4xn-4 + βΔt + ΔWn. When I have L2 regularization term α = 0.1, I get:

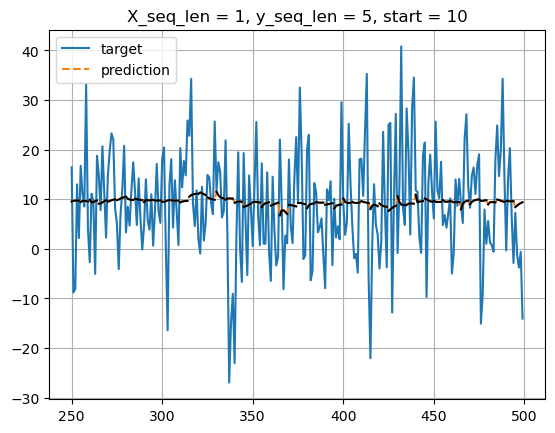
Results are:



Increasing number of auto regressive terms to n = 8 decreases mse\_exp to 98.6, which is close to the best ARMA model fit.

**LSTM Forecaster**

Now trying the LSTM encoder-decoder. I get:





So this isn’t as good as the classical models, generally, but did do better than the global average.