Meeting 12 Prep

# Initialization

We are looking for the smallest L1-penalty parameter such that the estimated W is a DAG. Question is how we optimize when we have this L1-parameter, as there is no closed form solution.

Approaches:

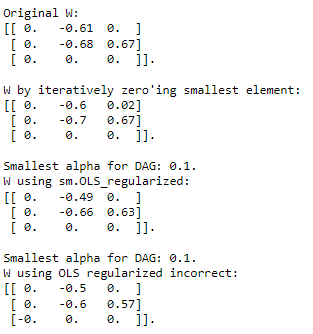
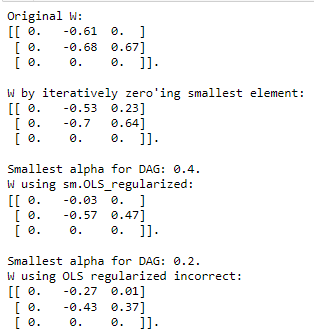
* LINGNAM (Iteratively set smallest non-zero element to zero)
* LASSO-CLOSED-FORM-INCORRECT (Use [this formula,](https://en.wikipedia.org/wiki/Lasso_(statistics)#Orthonormal_covariates) which should only work on orthonormal data, so not really suitable here, but is closed form).
* LASSO-LARS / SQUARE ROOT LASSO / LASSO (Iteratively increase L1-penalty).

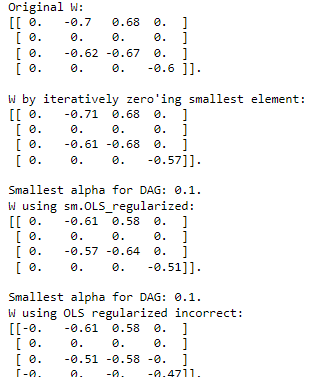
Conclusion:

* LASSO-CLOSED-FORM-INCORRECT does something very similar to the LINGNAM one, as it iteratively shrinks one value at the time. The only difference is that with LINGNAM, all values are shrunk equally, but for LASSO-CLOSED-FORM-INCORRECT, this scale also depends on the absolute value of the entry.

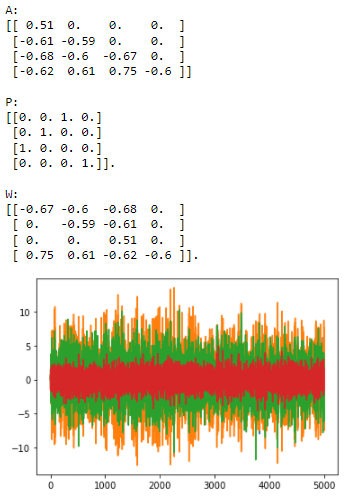
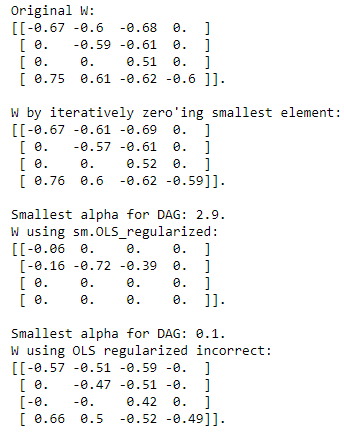
A larger *T* seems to improve the performance of all three methods. This makes sense, as all the parameters will be closer to their true parameters, so all the non-true edges are closer to zero and hence easier to get the true *W*.

*T = 50 T = 5000*





For a very dense *W*, the LASSO method seems to regularize out a lot of the true parameters before it is a DAG. E.g. here, for a fully dense W, the regularization regularizes out 7/11 true parameters. The other methods do not do this. This holds both for large and small *T*.



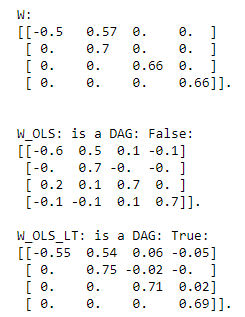
# Optimizing A When we KNOW the permutation

Now that we know the permutation, we can easily estimate the values for A. We need to constrain A to be lower triangular, but this can be done easily using least squares. We first estimate the first variable X\_\pi(1). For this variable, we can only use this variable as a regressor, so we use least squares with only this variable as regressor.

For the second variable X\_\pi(2), we use all the previous variables (X\_\pi(1)) and the current variable as regressor. We keep doing this until the end, and then we have a closed form solution for our problem.

Works well, much better than gradient descent (especially for large T). Also closed form, mathematically more sound.

However, we do need to know the permutation beforehand.



# Orthogonal Matching Pursuit

Iteratively, pick the index i = 1, …, n that minimizes the error. Normally is used to solve Ax = b, or equivalently, y = Xbeta + epsilon (regression, our scenario).

We want to solve X\_t = X\_{t-1} W + epsilon.

This is often used when the number of variables is much larger than the number of measurements, but this is not necessary the case in our scenario.

For each regression scenario, we only pick a certain number of variables X that we use as regressors. We pick these regressors using Orthogonal Matching Pursuit.

# NOTEARS for VAR

Works very well.