**Transfer Function Models (Lectures 8 and 9)**

*Linear Filters*

Thus far we have only done a univariate analysis where the variable is regressed upon itself with a leftover error term which may also have an autoregressive structure. The transfer function model adds a second time series variable. We can then regress our primary variable against the new variable (as well as its past samples) to see if the new variable affects the value of our primary variable.

We will use *Yt* to indicate the primary variable of interest and *Xt* to indicate the variable thought to influence *Yt*. The simplest model is just a regression of Y against X which omits the error term:

This model is called a **linear filter** and the operator is the **transfer function** which modifies the input signal (*Xt*) to obtain the output response (*Yt*). Without the error term, all stochastic or non-deterministic behavior much be represented by the *Xt* samples. However, we do not necessarily assume that the *Xt* process is stochastic at all. Instead, it can be any process – even one which we control – which is modified or transformed to produce the output *Yt*.

One way to examine this system is to imagine a single “impulse” as input to the system. If *X0* = 1 and all other *Xt*’s are 0, we would observe that the *vi* weights indicate the state of the system at time *i*. For this reason the collection of coefficients, *vi*, can also collectively be called the **impulse response function**. With weights we could graph the coefficients like this:



Each spike also indicates the state of *Yt* at times 0 through 7 if we input a single impulse at time 0. If, instead of a single impulse, we feed a constant signal into the system with all *Xt* = 1, we will find the above pattern repeated and overlayed with itself indefinitely. Each point will have a contribution from every coefficient so that the output will be a constant signal with magnitude:

This is called the **steady state gain**. If the result is greater than 1, the system in general acts as an amplifier, increasing the input signal. If the result is less than 1, the input signal in general will be dampened.

*Dynamic Regression Functions*

The linear filter model presented above is similar to an MA model with the error terms replaced by the independent variables, *Xt*. Continuing this analogy, we might wonder why we don’t add autoregressive terms (AR) to the equation to get:

In fact, we can do this, but we usually leave *Yt* alone on the left-hand side of the equation and instead divide through by to get:

Notice that in *w*-form, the backshift polynomial in the numerator now has minus signs instead of the positive signs we saw in the linear filter representation. The denominator also uses the same convention but also fixes the first coefficient as 1. This does not sacrifice generality as scaling can still be performed by the *w*-weights. Fixing the coefficient in the denominator instead removes a variable that would give us an infinite set of equivalent parameter estimates.

The important point of this formula is that is a rational function that could be evaluated to get an infinite set of coefficients that would equal *v(B)*. This means that the above equation is simply another way to write the linear filter. The advantage is that while the linear filter may have infinitely many weights (if any coefficient in the denominator is non-negative), and thus infinitely many parameters to estimate, we may be able to work with considerably less parameters (and much more parsimonious models) by intelligently using only the relevant parameters in the numerator and denominator. Ultimately, we are back to working with ARMA models where the MA coefficients are in the numerator and the AR coefficients are in the denominator. An extra coefficient (w0) is added to allow arbitrary scaling, and we have not yet introduced any error terms of stochastic component other than *Xt*. Nothing prevents us from doing this, however, and if we choose to add an autoregressive error term we could easily attach it as:

Notice that the rational function expression of coefficients still allows us to create ARMA models as we would have if we multiplied through to get and on the left-hand side of the equation. The notation above is a little more convenient, however, as it allows us to easily add additional terms without interfering with other parts of the equation. In fact, there is nothing stopping us from regressing *Yt* against as many time series as we’d like, so that in general we could have:

where *Yt* is regressed against *M* independent time series variables. We also include an autoregressive error term with the white noise process *at*. The presence of denominators in the rational functions, while only added as a convenience (we could instead use polynomials of infinite degree) also make it clear that *Yt* can be regressed against itself if it has any internal autoregressive structure.

*Feedback Tests*

It is important to note that the models discussed assume that the *Xt* variables are independent processes that drive the value of *Yt*. The model does not provide a means for the independent variables to influence each other of for *Yt* to influence them. The latter problem is one of **feedback**: X influences Y which influences X, and so on. If feedback is present in the system, we would not want to use the models presented so far.

A simple test for feedback (what Levy calls the “poor man’s test”) is to perform a regression of the supposedly independent *Xt* process against all other variables in the system. An arbitrary number of time lags, *p*, can be used (a large number like 15 or 20 is suggested) with *Xt* regressed against itself at lags 1 through *p*, all other independent (*Xt*) processes at lags 1 through *p*, and against *Yt* at lags 1 through *p*. We won’t care if the regression produces significant correlations with *Xt* and itself or with other dependent variables, but if the parameters regressing *Xt* on *Yt* are significant, we have evidence of feedback in the system and should choose another model (like a vector auto-regressive model).

In the “poor man’s test” we are left to check whether any of the *Yt* coefficients are statistically significant. We can choose to examine them individually or to do a “portmanteau” test that takes all the *Yt* coefficients as a whole and checks for statistical significance. A more advanced test is the Granger Causality Test. This is similar, and begins by performing the regression:

It then also performs the regression without the presence of *Y*:

An F-Test compares the regressions to see if adding the *Y* variables significantly improves the model.

Reference: <http://support.sas.com/rnd/app/examples/ets/granger/index.htm>

*Estimating Parameters of a Linear Filter: Xt is uncorrelated with errors*

Model identification and estimation is significantly easier if we can make the assumption that *Xt* is uncorrelated with the error terms. In this case we can use the cross covariances between *Y* and *X* to estimate the model parameters since the error terms will drop out of our equations. We begin by writing the model:

as linear filter:

By taking cross-correlations of X with Y at different lags, we will be able to determine the theoretical cross-correlations that should result for different types of models. As with ARMA models, this will help us to identify the type of model we want to fit based on a plot of the estimated cross-correlations, and we will then be able to estimate the parameters of such a model.

First, we define the **cross covariance coefficient** at lag *k*:

Note that . This is not a symmetrical relationship as we had in the autocovariance coefficient. The autocovariances were symmetrical, with . However the cross covariances require us to switch the subscripts when we negate the lag parameter. In general, we will find that .

We can estimate the covariance with:

and normalize the value into a cross correlation coefficient by dividing by the (estimated) standard deviations of each variable:

Also, notice that we divide by *n* even though we only have *n - k* terms in the summation. This is done because it guarantees that the cross-correlation matrix produced will be positive definite and thus invertible. If we instead divide by *n* or *n*-1, this condition will not always be met.

Now, if *Y* is given as a linear filter of *X* with *k* coefficients in the impulse response function (with both variables assumed to be centered), we can multiply the linear filter:

by to get the cross-covariances:

(This is where the assumption that *Xt* and are uncorrelated allow the term to drop out of the expectation and leave behind only correlations between *Y* and *X*.)

Taking these for successive lags, k, produces a system of equations similar to the Yule-Walker equations.

which can be inverted to find estimates for the coefficients of the impulse response function.

*Estimating Parameters of Linear Filter: General Case*

Having found a simple way to estimate the parameters of a linear filter when *Xt* is uncorrelated with the terms of the error process, we now wish to drop that restriction. The simplest way to do this is to find a transformation of *Xt* that leaves behind nothing but white noise. The residual white noise would then be uncorrelated with the terms of the error process and we could fit the model using these residuals instead of the *Xt* themselves. This method of analysis is called “**pre-whitening**” as we transform our series to white noise before fitting the model.

Assuming we can find a satisfactory univariate model for *Xt*, the residuals of the model would be assumed to be white noise. If we fit the ARMA model:

we could isolate the residuals to get:

which is exactly the filter we are looking for. When applied to the transfer model:

we can multiply through by the filter to get:

We see that we also have to multiply the *Yt* series by the pre-whitening filter. The resulting model now fits our requirement that the independent variables are not correlated with the error terms (white noise is not correlated with anything), and we can use the methods discussed earlier to estimate the impulse response function.

TODO: LTF

*Simplifying Linear Filter Models*

We have already noted that the generic linear filter can be written as:

In theory, we can have infinitely many coefficients, but we can sometimes simplify this by using a rational function for the weights:

|  |  |  |
| --- | --- | --- |
|  | (or) |  |

While we can estimate the linear filter coefficients directly, estimating those in the rational function representation can be a little trickier. However, the reward is a model with fewer parameters that is much less subject to over-fitting.

We begin by noting that changes in *X* may take a while to affect *Y*. The linear filter assumes that directly affects at any moment through the coefficient . However, if this means it takes at least 1 time period before changes in *X* affect *Y*. In general, we may find that the first *b* coefficients are all zero, indicating that it takes *b* time periods before *X* affects *Y*. *b* is called the inertial parameter of the system, and we can simplify our analysis by back-shifting *X* to remove this lag and line up the series. We thus, are really interested in modeling:

After allowing for this lag, we will then find that the graph of the estimated impulse response function is similar to the ACF in an autoregressive estimation. Significant spikes indicate the presence of MA terms (in this case coefficients in the polynomial), and geometric decay patterns indicate the present of AR terms (in this case coefficients in the polynomial). We do not have a plot corresponding to the PACF, so we instead have to recognize both patterns using just the ACF alone. The next section describes Pankratz’s rules for analyzing the impulse response function and trying to identify the type of rational function we would like to fit to it.

*Pankratz’s Rules for Simplifying v(B)*

1. Determine *b* by looking at the number of v-weights that are insignificant starting at *v0*. If *vi* is the first significant weight, we assume *b = i*.
2. Determine r, by looking for patters in the v-weights:
	1. Simple exponential decay indicates r = 1
	2. Compound exponential decay (or damped sinusoid) indicates r = 2
	3. r >= 2 is rare
	4. If we cannot find any pattern, r = 0
3. If r > 0, determine the start-up lag for the AR pattern.
	1. Find the largest lag ()
	2. If r = 1, this is the start-up lag
	3. If r = 2, this lag, and the adjacent lag with the highest magnitude are the start-up lags
4. Determine the number of parameters in the numerator. We want one parameter for each significant spike which is not accounted for by the AR decay.
	1. If r = 0, we simply have 1 parameter for each significant spike
	2. If r > 0, we count the number of significant lags that appear before the AR start-up lags (calling this *u*), and we require one parameter for each of these. We also add parameters for each of the start-up lags (in case there is over-lapping activity at these points). We thus have u + r parameters.

NOTE: the constant coefficient in the numerator polynomial is allowed to vary (unlike the denominator). This means that a polynomial of degree h has h+1 free parameters. We thus calculate the number of parameters needed to model the significant spikes and subtract 1 to get the polynomial degree: h = u + r – 1

*Examples*

|  |  |
| --- | --- |
|  |  |
|  |  |

*Steps in Identifying, Fitting, and Diagnosing TF Models*

1. Difference *Xt* and *Yt* (the same number of times) to achieve stationarity
2. Identify, estimate, and check an ARMA model for *Xt*
3. Pre-whiten both *Xt* and *Yt*

|  |  |
| --- | --- |
|  |  |

1. Cross correlate with to calculate estimates for the impulse response function,
2. From the form of , make preliminary guesses as to model type (*b,r,h*) where:

b = lag of input and output

r = order of AR operator,

h = order of MA operator,

1. Fit the model:

(assuming that is white noise)

1. Take the residuals of the model, , and check for correlation with *Xt*. That is: calculate and test that these are all equal to zero. If we fail this test we have to go back to step 4 and look for other models that fit the estimated impulse response function.
2. Find an ARMA model for the residuals, , using univariate methods to produce:
3. Fit the “full” model:

This involves estimating 1 + (h + 1) + r + q + p parameters. (1 for the mean of *y*, h+1 in , r in , q in , and p in .

1. Estimate the residuals of the model, , and check to make sure they are white noise. If the model fails this test we have to go back to step 7 and find a different model for th residuals.
2. Make sure that the residuals, , (which are assumed to be white noise) are uncorrelated with *Xt*. If correlation exists, we have to go back to step 4 and fit a different model.
3. Perform additional tests on the residuals to make sure that they are white noise.