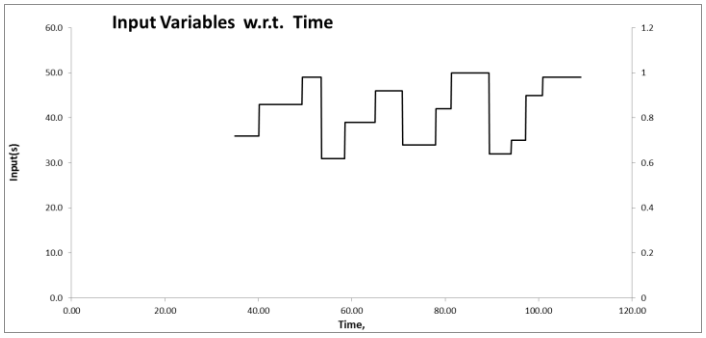
User Guide for FOPDT and SOPDT Model Regression
7 August 2016 – R. Russell Rhinehart

User Guide for "2016 Generic LF Dynamic Model Regression FOPDT.xlsm"
And
"2016 Generic LF Dynamic Model Regression SOPDT.xlsm"

This guide explains First-Order Plus Dead Time (FOPDT) and Second-Order Plus Dead Time (SOPDT) models, how to generate data needed to determine model coefficient values, and how to use an optimizer to determine best model values. These are alternately called first-order lag or delay is used for deadtime, with diverse acronyms such as FOLPDT, FOLPD, etc.

Empirical Modeling – Skyline Input

First, you need data. Generate a skyline pattern in U the controller output. For example:



U makes step-and-hold moves for a variety of hold durations and step-to values.

Step the manipulated variable (MV) to random values within a desired range, hold for a random time interval between 0.1 and 1 times process open loop setting time. Repeat. The range of Umin to Umax should be large enough to create significant deviations in the controlled variable (CV), relative to noise and disturbances, but within the local range of CV values for which the model will be used.

```
If Time > changepoint Or Time = changepoint Then
    changepoint = changepoint + (0.1 + 0.9 * r) * ST
    U = Umin + r * (Umax - Umin)
End If
```

You don't have to watch and choose when to change after detecting that the process has achieved steady state. The time duration and CV deviation away from nominal value is much less w.r.t the standard up-down-down-up method. Within a same interval of 4 settling times, the skyline input provides much greater richness of signal/influence to minimize confounding influences of noise and disturbances.

During the trial period, external disturbances may make the CV drift away from the nominal operating conditions. If so, the skyline MV values could be biased by a value to keep the CV within the desired operating range.

The skyline pattern does not have to be step-and-holds. It could be include any pattern including ramps.

Collect the MV(t) and CV(t) data.

Empirical FOPDT Modeling – Model

The FOPDT model is in deviation variables

$$\tilde{\tau}\frac{d\tilde{y}'(t)}{dt} + \tilde{y}'(t) = \tilde{K}u'(t-\tilde{\theta}) \quad (1)$$

Where

$\tilde{y}'(t) = \tilde{y}(t) - \tilde{y}_{base}$
$\tilde{u}'(t) = \tilde{u}(t) - \tilde{u}_{base}$
$t = time\ after\ the\ user\ defined\ start\ point$

The initial condition is

$$\tilde{y}'(t=0) = \tilde{y}(0) - \tilde{y}_{base} = \begin{Bmatrix} 0\ if\ starting\ at\ y_{base} \\ \tilde{y}_{initial} - \tilde{y}_{base}\ otherwise \end{Bmatrix}$$

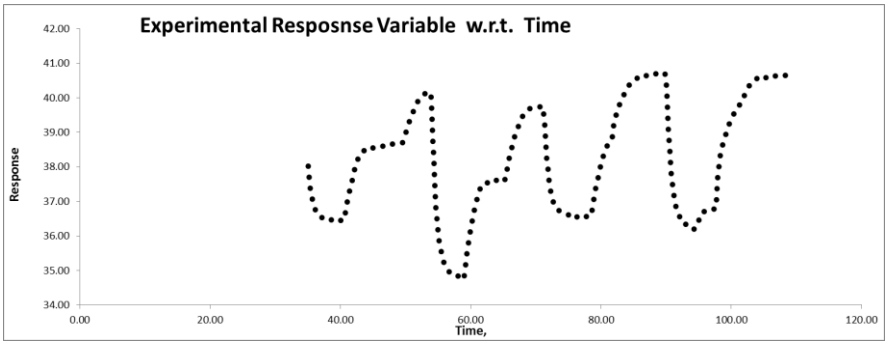
A common presumption is that the process starts from an initial steady state. But, in my experience, a real process usually does not. If it starts at an initial SS then $\tilde{u}_{base}$ and $\tilde{y}_{base}$ are the initial values and $\tilde{y}'_{initial} = 0$. The user can specify those values. But, if the process does not start at an initial SS, you cannot presume to know those values.

It appears that there are 6 model coefficients: $\tilde{\tau}, \tilde{K}, \tilde{\theta}, \tilde{y}_{initial}, \tilde{y}_{base}, \tilde{u}_{base}$. However, there are only 5 independent model coefficients. There is a steady-state relation between y and u. The FOPDT model indicates that at SS: $\tilde{y}'_{SS} = \tilde{K}\tilde{u}'_{SS}$. Converting the steady-state deviation variables to CV and MV reveals the relation between $\tilde{K}, \tilde{y}_{base}, \tilde{u}_{base}$.

$$\tilde{y}_{base} - \tilde{K}\tilde{u}_{base} = y_{SS} - \tilde{K}u_{SS} = c \quad (2)$$

Where c is some value defined by the process and the modeled process gain. You could find the c-value by holding the MV at the u_{SS} value and waiting to steady-state to find the y_{SS} value. But, it might take forever to find a steady-state period. Fortunately, you don't need to do so. Realizing that Equation (2) places a constraint on the three model coefficients $\tilde{K}, \tilde{y}_{base}, \tilde{u}_{base}$, permits you to choose the value of one, reducing the number of DVs to 5: $\tilde{\tau}, \tilde{K}, \tilde{\theta}, \tilde{y}_{initial}, \tilde{y}_{base}$.

My preference is to choose the $\tilde{u}_{base}$ value, and to choose it as the nominal u-value about which the skyline function is centered, the average of u_{max} and u_{min}. Alternately, you could choose initial u-value. After generating the skyline MV input, use regression to adjust the 5 model coefficients to best make model match measured Y response using sum-of-squared deviation as the OF. The CV response might appear as:



Empirical FOPDT Modeling – Regression

The traditional optimization statement is:

$$\min_{\{\tilde{\tau},\tilde{K},\tilde{\theta},\tilde{y}_{initial},\tilde{y}_{base}\}} J = \sum_{i=1}^{N}(y_i - \tilde{y}_i)^2 \qquad (3)$$

Note that the regression occurs at discrete time-sampling points, not continuous for all time. In the optimization statement of Equation (3) the index i is the time counter with $t = i\Delta t$. And note, although the model delay, $\tilde{\theta}$, could have a continuum of values, when time is discretized into Δt intervals, the delay must have an integer number of Δt intervals. So, the delay becomes $\tilde{n} = INT(\frac{\tilde{\theta}}{\Delta t} + 0.5)$. Because of the delay, this is a nonlinear regression; and because the delay is constrained to be integer multiples of the sample interval, the optimization is mixed integer (integer and continuum) – Mixed Integer, Nonlinear, and Constrained.

To calculate the optimization Objective Function value, J, for each trial solution of the decision variables:

```
Ndelay=INT(thetamodel/dt + 0.5)        'discretized model delay
clambda = Exp(-dt / Taumodel)          'model coefficient
lambda = 1 – clambda                   'complement to model coefficient
ymodeldev = Yinitial – Ybase           'initial deviation variable value for model
SSD = 0
For DataNumber = Ndelay + 1 To Ndata   'start with first value after the delay
    influence = udata(DataNumber - Ndelay) – Ubase                  'calculate influence
    ymodeldev = lambda * Kmodel * influence + clambda * ymodeldev   'model
    ymodel(DataNumber) = ymodeldev + Ybase                          'convert to CV
    SSD = SSD + (ymodel(DataNumber) – ydata(DataNumber))^2          'sum d^2
Next DataNumber
```

Empirical SOPDT Modeling – Model

The SOPDT model is in deviation variables

$$\tilde{\tau_1}\frac{d\tilde{y_1}'(t)}{dt} + \tilde{y_1}'(t) = \tilde{K}u'(t - \tilde{\theta}) \qquad (4)$$

$$\tilde{\tau}_2 \frac{d\tilde{y}_2'(t)}{dt} + \tilde{y}_2'(t) = \tilde{y}_1'(t) \quad (5)$$

Where

$\tilde{y}_i'(t) = \tilde{y}_i(t) - \tilde{y}_{base}$
$i = \{1,2\}$ to indicate first or second lag. The second lagged variable is the measurement.
$\tilde{u}'(t) = \tilde{u}(t) - \tilde{u}_{base}$
t = time after the user defined starting point

The initial condition for the second lag is the initial measurement and for the first lag is an initial value.

$$\tilde{y}_2'(t=0) = y_{data}(t=0) - \tilde{y}_{base}$$
$$\tilde{y}_1'(t=0) = \tilde{y}_{1,initial} - \tilde{y}_{base}$$

A common presumption is that the process starts from an initial steady state. But, in my experience, a real process usually does not. If it starts at an initial SS then $\tilde{u}_{base}$ and $\tilde{y}_{base}$ are the initial values and $\tilde{y}_1'_{initial} = \tilde{y}_2'_{initial} = 0$. The user can specify those values. But, if the process does not start at an initial SS, you cannot presume to know those values.

It appears that there are 8 model coefficients: $\tilde{\tau}_1, \tilde{\tau}_2, \tilde{K}, \tilde{\theta}, \tilde{y}_{1,initial}, \tilde{y}_{2,initial}, \tilde{y}_{base}, \tilde{u}_{base}$. However, there are only 7 independent model coefficients. There is a steady-state relation between y and u. The model indicates that at SS: $\tilde{y}'_{SS} = \tilde{K}\tilde{u}'_{SS}$. Converting the steady-state deviation variables to CV and MV reveals the relation between $\tilde{K}, \tilde{y}_{base}, \tilde{u}_{base}$.

$$\tilde{y}_{base} - \tilde{K}\tilde{u}_{base} = y_{SS} - \tilde{K}u_{SS} = c \quad (6)$$

Where c is some value defined by the process and the modeled process gain. You could find the c-value by holding the MV at the u_{SS} value and waiting to steady-state to find the y_{SS} value. But, it might take forever to find a steady-state period. Fortunately, you don't need to do so. Realizing that Equation (6) places a constraint on the three model coefficients $\tilde{K}, \tilde{y}_{base}, \tilde{u}_{base}$, permits you to choose the value of one, reducing the number of DVs from 8 to 7: $\tilde{\tau}_1, \tilde{\tau}_2, \tilde{K}, \tilde{\theta}, \tilde{y}_{1,initial}, \tilde{y}_{2,initial}, \tilde{y}_{base}$.

My preference is to choose the $\tilde{u}_{base}$ value, and to choose it as the nominal u-value about which the skyline function is centered, the average of u_{max} and u_{min}. Alternately, you could choose initial u-value.

There are two initial values for the state variables, $\tilde{y}_{1,initial}, \tilde{y}_{2,initial}$. I have chosen to set $\tilde{y}_{2,initial} = y_{data(t=0)}$. This reduces the number of regression variables from 7 to 6: $\tilde{\tau}_1, \tilde{\tau}_2, \tilde{K}, \tilde{\theta}, \tilde{y}_{1,initial}, \tilde{y}_{base}$.

After generating the skyline MV input, use regression to adjust the 6 model coefficients to best make model match measured Y response using sum-of-squared deviation as the OF. The CV response might appear as:

Empirical SOPDT Modeling – Regression

The traditional optimization statement is:

$$\min_{\{\tilde{\tau}_1,\tilde{\tau}_2,\tilde{K},\tilde{\theta},\tilde{y}_{1,initial},\tilde{y}_{base}\}} J = \sum_{i=1}^{N}(y_i - \tilde{y}_i)^2 \qquad (7)$$

Note that the regression occurs at discrete time-sampling points, not continuous for all time. In the optimization statement of Equation (7) the index i is the time counter with $t = i\Delta t$. And note, although the model delay, $\tilde{\theta}$, could have a continuum of values, when time is discretized into Δt intervals, the delay must have an integer number of Δt intervals. So, the delay becomes $\tilde{n} = INT(\frac{\tilde{\theta}}{\Delta t} + 0.5)$. Because of the delay, this is a nonlinear regression; and because the delay is constrained to be integer multiples of the sample interval, the optimization is mixed integer (integer and continuum) – Mixed Integer, Nonlinear, and Constrained.

To calculate the optimization Objective Function value, J, for each trial solution of the decision variables:

```
Ndelay=INT(thetamodel/dt + 0.5)        'discretized model delay
clambda1 = Exp(-dt / Taumodel1)        'model lag coefficient for y1
lambda1 = 1 - clambda1
clambda2 = Exp(-dt / Taumodel2)        'model lag coefficient for y2
lambda2 = 1 - clambda2
ymodeldev2 = ydata(Ndelay) - Ybase     'start y2 at first measurement
ymodeldev1 = yInitial - Ybase          'initialize y1, not necessarily at steady state
SSD = 0
For DataNumber = Ndelay + 1 To Ndata
    influence = udata(1, DataNumber - Ndelay) - Ubase
    ymodeldev2 = lambda2 * ymodeldev1 + clambda2 * ymodeldev2          'second lag first
    ymodeldev1 = lambda1 * Kmodel * influence + clambda1 * ymodeldev1  'first lag after
    ymodel(DataNumber) = ymodeldev2 + Ybase
    SSD = SSD + (ymodel(DataNumber) – ydata(DataNumber))^2             'sum d^2
Next DataNumber
```

Empirical Modeling & Skyline Response Comments

This regression method seeks to fit the model to all data points (after the delay, actually N-minus-Ndelay number of data points), not just the selected several initial, final, and two intermediate points in a parametric fit. So, it better rejects noise and disturbances over classical reaction curve or parametric approaches.

The skyline and regression method does not require operator attention and judgment as to when steady state has been achieved, which lessens the possibility for operator error or bias.

While running the process tests, the skyline method keeps the CV closer to the target value, because it changes the MV to a new value prior to steady state. It also takes less time and has many more than just a few influences.

The skyline input has many ups and downs, accordingly this tempers the influence that environmental drifts have on confounding the CV response to the MV.

Because of these advantages, your choice in generating an empirical model should be the skyline MV influence and the regression approach to determining model coefficient values, not the old-school (pre-computer) best practice "reaction curve" approach that still dominates the direction given in textbooks.

Using the classical reaction curve approach or a parametric approach to get a FOPDT is relatively simple. And, it can generate models that are fully functional for control. However, to determine the two time-constant and delay values for a SOPDT model is not nearly as easy. The equations are more complicated and the propagation of error due to noise is larger. But, this is not a problem with the skyline and regression approach to get a SOPDT model.

Here are additional comments on skyline and regression as opposed to classic reaction curve approaches to obtain FOPDT or SOPDT models:

In the classic reaction curve methods, the up-down-down-up response should start with an initial steady state, make equal step changes in the MV, and wait for the CV to reach steady state before making the next MV step. The first up and down changes should return the CV to initial location. If it doesn't, something else is affecting the process. Then you should connect the initial and final SS CV values as the base-line, and use the differences between CV and base line as the measure of the CV response to the MV. The second down and up is to push the process to the other side of the base value so that you can see if there is any impact of nonlinearity, and also to provide a CV balance to the first up-down upset. Again, the CV should return to the initial location.

Continuing discussion of the classic reaction curve method, the steps should be large enough to clearly reveal the CV change, but small enough so that the CV remains in the local region. Usually 10% changes in MV are made: For instance, start at 25% then go to 35% then 25% then 15% then 25%.

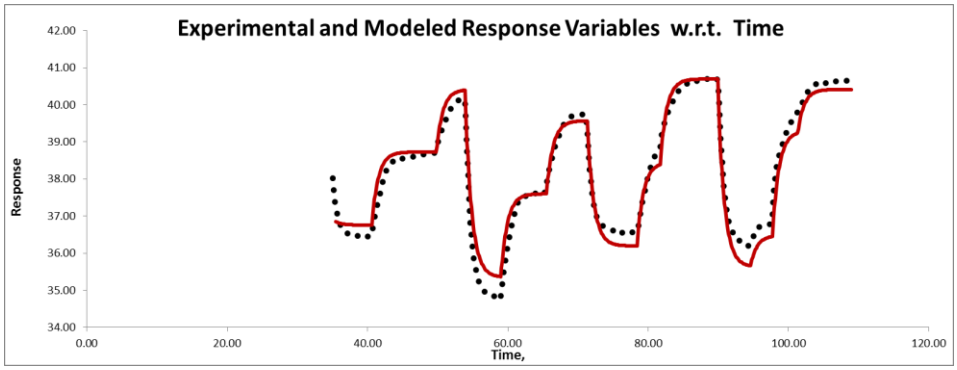
Continuing: If the process is linear, the steps up and down should each provide identical response. The two steps up should be identical. The two steps down should be identical. If the process is linear, then if you flip the down responses (turn the page over and look at them through the paper), they should be exactly the same as the up responses.

Continuing: Real processes have noise and disturbances. When doing classic step tests, this makes it difficult to know when the process transient from the MV change is over, and when the operator can implement the new MV change. This also makes it difficult to determine the ΔCV at SS for the process gain calculation. This procedure requires starting at an initial SS and waiting for 4 additional SS points, which makes the test period last about 5*5 = 25 time-constants – a long period. During the test period the process is pushed off of the nominal CV value for about 2*5 = 10 time-constants, and then pushed off nominal in the other direction for another 10 time-constants – a long time for the quality manager to accept. After you experience these difficulties, you can see that the classic reaction curve techniques are not as good as using the skyline function and regression to get your model.

By contrast, the skyline and regression method has many steps not just one or a few, which provides greater tempering of noise and disturbances than does classic reaction-curve modeling techniques. The skyline and regression method does not extend the period of off-nominal production, each "+" or "-" period is shorter, creating less objection by a quality manager. The skyline and regression method does not require an initial SS, the entire test period takes less time.

As a comment on the models, for a nonlinear process, gains, delays and time-constants change with MV and CV. This FOPDT or SOPDT model is linear, and may not provide a great match to the process over a wide operating range. Just because the optimizer converges, and consistently converges, on a best model, does not mean that that model actually fits the data. As a similar concept, one can seek to fit a line through curved data. There is a best line, a line that minimizes the SSD. But the fact that there is a best line, does not mean that it adequately corresponds to the data. So, after the regression, look to see if the fit is satisfactory for your model use purposes.

The following graph reveals the data and best FOPDT model. The model is the solid line, the data are the dots. Note that early-time data is usually below the model, but late-time data is usually above. Perhaps some drifting influence was affecting the data. Also note the kinks in the model at times a bit after 80 and 100. These are expected responses to small changes in the MV at those times, but these are not expressed in the data. The data is from a pilot-scale unit, and models flowrate response to signal to the valve. There is no positioner on the valve, and perhaps valve sticktion prevented the valve from moving even though the MV made changes. Finally, flow rate response to the signal to the valve is not expected to be a linear response. Although the linear FOPDT model does not perfectly match the data, it is a very good representation of the process dynamics.



Empirical FOPDT or SOPDT Modeling – Regression Program

I have written a generic VBA program to find FOPDT and SOPDT model coefficients. The files are **"2016 Generic LF Dynamic Model Regression FOPDT.xlsm"** and **"2016 Generic LF Dynamic Model Regression FOPDT.xlsm"**. The method uses Leapfrogging as the optimizer (a multi-player direct search algorithm) because it is robust to many surface features (such as the flat spots due to the delay discretization) and has a high probability for finding the global optimum. It uses steady-state identification as the convergence criterion because that is a generic approach, universal to any application regardless of scale or values, and does not need user defined tolerance or precision values. It uses multiple starts from random initial model coefficient values, to reveal local optima and the confidence that the global has been found. Although this write-up describes a Single-Input-Single-Output (SISO) process, the generic optimizer can handle a Multiple-Input-Single-Output (MISO) situation.

The data entry is on the "DataIN" tab, Columns 5, 6, and 7. Press the "DeleteOldData" button to delete the existing data. Enter your new data (CV, time, MV) in the columns and time increment in

cell(4,6). Columns 21 and beyond contain data that I've generated in our labs. The sample data in the green fields represents tests of air flow rate to signal to the valve for our absorber. Notice in the data that there is not one step-and-hold from an initial steady state. There are multiple steps in the MV. In regression, one does not need a single step-and-hold influence, and multiple steps provide replicate data for a model that is not influenced by the vagaries of one trial.

The "Main" tab reveals the key features. The graph on the left side reveals data (dotted black line) and model (solid red line). Below it, the graph shows both MV and CV.

The graph on the upper right indicates approach to convergence. The abscissa is the optimizer iteration. The blue dots represent the rms deviation of model-to-data from a random sampling. When the regression has converged the blue dots will be a noisy steady state. The solid red line in the graph is the filtered value of the random-set rms. When the solid red line relaxes to the middle of the blue dots, there is no visible improvement of the model with iterations, and the optimization stops. The green line is an indication of the player number of the Leapfrogging player that has the best OF-value. As the worst player leaps over the best, it might become the new best.

The graph in the lower middle is a parity plot of modeled w.r.t. experimental data. Ideally, with a perfect fit, all of the points are on the 1:1 line.

The graph in the lower right is a cumulative distribution function (CDF) of the objective function value, the root-mean of the sum of squared deviations (rms). If every randomized optimization run stopped at the same rms, then the curve would be a vertical line. If there are local optima that trap the optimizer, then the CDF curve will show steps that represent the local optima. The initial steep rise represents that the optimizer repeatedly found that same best optimum, increasing confidence that the global has been found. The clear steps (as opposed to ramps, or other smooth transitions) indicate that the optimizer repeatedly found the same optima – that the local and global were definitive. Steps indicate that there are several optima, a characteristic of a nonlinear application.

To run the optimizer enter what you think are reasonable ranges for the model coefficients, and press the "Start Regression Optimizer" button.

If you want to enter your own values for the coefficients in Column 8, you can, and then press the "Display Model" button to see how well your model works.

The FOPDT model needs 5 coefficient values. Three are obvious, gain, time-constant, and delay. But, the FOPDT model is in deviation variables, and needs the base case values for both the MV and the CV. However, the base case values for the deviations are not independent. They are related by the data steady state and the model gain. So, I set up this optimizer for the user to choose either the MV-base value or the CV base value. Note, I think choosing an MV base value is easier, since the CV is a response to the user choices, so I choose to set the base MV value. My convenient method to do this was to make the high and low range u-base values in Columns 9 and 10 identical, and let the optimizer think there are 6 DVs.

The SOPDT model needs 6 coefficient values. Again, the steady state relation constrains Km, Ubase, and ybase. And I prefer to choose the ubase value. I also initialized the y2 value, the output state with the initial measurement value, and initialized the y1 value with an alternate to best make it match the data.

Really, however, the initial values are relatively unimportant, because their influence fades as time progresses.

You can change the operation by entering values in the green fields. The yellow highlighted fields are for the program to output progress.

As optimizer independent runs are finished the results are presented in Rows 16 and below on the "Main" tab and on the "DataRESULT" tab. In the "DataRESULT" worksheet, they are sorted in ascending SSD order.

The program is limited to a delay of 50 time increments. And to 3,000 data points.

This VBA macro uses leapfrogging (LF) to best fit a model to data. LF seeks model coefficient values to minimize the root-mean-square deviation (square root of the sum of squared deviations divided by number of data) between data and model. Convergence is based on the rms-value of a random selected subset coming to steady-state with iteration progress, and the optimizer starts N times (from randomized DV-values) and reports the best of N.

Where to find things

- The worksheet tab "Main" is for user inputs and to observe results.
- The worksheet tab "DataIN" is for user to input experimental data.
- The worksheet tab "DataRESULT" summarizes the N optimization trials, sorted by OF value.
- Subroutine "Response_Model" in the VBA module contains the model, and also recognizes if the argument of the square root function could be negative.
- Subroutine "ConstraintTest" in the VBA module tests to see if DV values violate constraints.
- Subroutine "Assign" in the VBA module assigns LF player DV values to the model coefficients.
- Declarations in the VBA Code are where you define variable names.

If you wish to use this program for your individual application you'll need to work with those 6 items.

Disclaimer

The user accepts all responsibility for the use of this program or the use of results from this program.

User Guide

Main Worksheet

You can enter your own model coefficient values in Column 8 and then click on the "Display model" button. The data are the red dots on the lower left graph and the modeled values are the black astericks connected by the black connect-the-dots curve. The middle graph is a parity plot of modeled w.r.t. experimental y-data, and permits a view of the distribution of residuals. If you enter variable values that violate a constraint "FAIL" will appear in Cells(8, 6). The rms value will appear in Cells(12, 6).

To run the optimizer, first enter a hypothesized range of coefficient values in the green cells of Columns 9 and 10. (The optimizer is not constrained by that range and will seek an external optimum.) Then choose an option in the green Column 15 cells, then press the "Start Regression Optimizer" button. You can also change optimizer criteria in Column 4 cells.

Optimizer results are placed in the cells associated with the yellow background. Column 6 contains optimizer progress identifiers. Once converged, DV*, OF* and associated data (trial number, number of function evaluations, number of iterations) are listed in Columns 1 to 14.

If you place a desired confidence (0<c<1) basis in Cells(3, 15) and a desired best fraction (0<f<1) in Cells(4, 15), Cells(5, 15) will reveal the number of trials needed to be c confident that at least one of the N trials will have landed in an rms value that represents one of the best f possible.

I like to choose a "Y" (for "yes") in Cells(8, 15) to observe the optimizer progress. The upper right graph reveals the random subset rms value w.r.t. iteration number (blue dots) and the red filtered value. When the red filtered value relaxes to a noisy steady state convergence is declared. A threshold for the R-statistic in steady-state identification of 0.85 is a good value. This is a scale independent, application independent left-hand critical value of a ratio of variances. The green line in the upper right graph is the number of the lead LF player (associated with the secondary axis). As long as it keeps jumping between players, all is well. Noteworthy, there is an outlier I the data. As a result, when it is sampled in the random set, it makes a blue dot that is definitely not consistent with the bulk of the blue dots. Such a pattern provides insight to the data.

After N trials, the graph on the lower right indicates the CDF (cumulative distribution function) of OF* values. This can provide insight on the solution. Steps in the CDF of OF* graph reveal local optima that trapped the optimizer, and horizontal axis values indicate the OF range of local traps. Desirably, the first rise will be steep and long, indicating that many trials found the same OF* value. However, any rise is not perfectly vertical. A numerical optimizer stops in the close proximity of the optimum, not exactly on it. So, if there is only one optimum, found by all trials, then they will not all have the exactly the same OF value, and the CDF of OF curve will not be a vertical line. But in this case, it will have a very small range on the horizontal axis.

DataIN Worksheet

You can enter your data here. You can use the "DeleteOldData" button to delete old data or you can use any method. Place your modeled output data in Column 5, and associated input(s) in Columns 6 or after. Do not skip data. Sort the data by input value (Column 6) for viewing convenience in the graphs.

The VBA program places the modeled output in Column 4. Do not place your data there.

Sub Response_model() in the VBA Module

You can enter VBA code for your model here.

Sub ConstraintTest() in the VBA Module

You can enter VBA code to look at constraints here.

Sub Assign() in the VBA Module

There are three decision variables DVs in this application, one for each model coefficient. The LP players have a player number and a dimension index. The array name is “PlayerPosition”. To determine the OF-value of the player, the rms-value of model to data, you need to assign player position values, DV values, to the model coefficients.

Declarations in the VBA Module

I choose to use the “Option Explicit” declaration in the header. This means that all variables need to be explicitly defined. Accordingly, if you decide to use a model with different variable or coefficient names you need to declare them here.