Supporting Materials for: Working with Data using Filtering and State Resetting

This archive contains the models and other files necessary to replicate the results in the chapter on filtering and state resetting. The material herein, with the exception of the ExternalFunction archive (which has its own license) is released under the following license:

The models contained here are all Vensim models and were developed using Vensim DSS. They have been saved in a binary format so that it will be possible to open them in the Vensim Model Reader. Not all of the results can be obtained in this way however as the model reader does not directly support some of the operations used in creating the results.

Figure 1

This graph is from a simulation of the model ProductionDistributionAppendixK.vmf in the ProductionDistribution directory simulated with the constant NOISE SEED set to 0 and 8.

Figure 2

This graph was created by running a sensitivity simulation with the constant “noise stream” varying between 1 and 2,000,000 by 100. This results in 20,001 simulations which is larger than

needed to see the spread so can be reduced. The .vsc and .lst files for this are also contained in the directory.

Figure 3

Using the model DampedPendulum00.vmf in the directory Pendulum this is run with the default parameters and NOISE SEED changed to 0. The latter run was called NoiseSeed0

Figure 4 (and parameter estimates)

Again using DampedPendulum00.vmf it is calibrated against NoiseSeed0 using the payoff file DampedPendulum01.vpd and the optimization control file DampedPendulum01.voc. The weight file PayoffWeights.cin is created by first calibrating without the changes file, then executing the program in err2weight.exe contained in the ExternalFunction archive. This file requires a .err file (so that payoff report needs to be checked) and uses the command line

err2weight calibrate_1step.err PayoffWeights.cin

to create the file. All this program does is create a weight equal to 1/standard deviation of error term (the sample standard deviation). You can also make the same computation other ways.

If you want to use the err2weight program on a Mac you will need to recompile the source code err2weight.c. After doing this the calibration is repeated using PayoffWeights.cin. The results are the same (since there is only one variable in the payoff), but the computed confidence bounds using Payoff Sensitivity of 4 now represent 95% confidence bounds. Note that in this case, because the fit is so poor, the confidence bounds are not very meaningful. This general technique can be used elsewhere – for example the regression results are computed using this (and also a linear regression in R for comparison).

Regression Results

There are two ways to compute these. One is with Vensim using the model DampedPendulumRegression02.vmf in exactly the same way as described in Figure 4 but with a different payoff definition file (DampedPendulumRegress.vpd). The file PayoffWeights.cin will need to be constructed again – it uses acceleration and not position for calibration.

To perform a linear regression using R the .Excel file ComputedVelocityAcceleration.xlsx uses the position measurement to compute velocity and acceleration. A truncated version of this (the last rows are missing otherwise) ComputedVelocityAcceleration.txt is the used by the R file Regression.R (you will need to change the directory for the file to load to use this). The R regression is linear and you will need to transform the resulting coefficients to the drag and length values (actually drag is just a sign change). You could also run a nonlinear regression but this is more complicated to set up.

Figure 5 (and parameter estimates)

This time using DampedPendulumResetting05.vmf we repeat the same steps as for figure 4. Note that this model will not run unless the run NoiseSeed0 exists.

Figure 6 (and parameter estimates)

This run is created by turning on Kalman filtering. To do this you will need to use the file kalman.prm which contains

```
:DEBUG 1
Angular Velocity/2.44/1
Angular Position/0/1
```

The first line causes the file kalman.err to be created which reports (among other things) the Kalman Gain. The other lines have the state name, driving variance and initial variance. The driving variance is all against velocity and 2.44 = (50 * 0.03125)^2 where 50 is the standard deviation of the driving noise and 0.03125 is TIME STEP. There is no driving noise for position. The initial values of 1 are arbitrary but relatively small. The payoff definition DampedPendulumKalman01.vpd is used there. The “weight” element of the payoff is the variance of the error term – since no measurement error was explicitly introduced here a small number is used.

To run the model with Kalman filtering active check the box for Kalman filtering in the advanced tab of the simulation control dialog. To do the parameter estimates optimize with the same settings. Note that the computation of the confidence bounds should have the Payoff Value sensitivity set to 2, and not 4. Use 4 for non Kalman Filtering calibration.

Figure 7

This is the model Project08.vmf in the directory Project. A number of elements have been hidden to make this easier to read. The model with all hidden elements showing appears as:

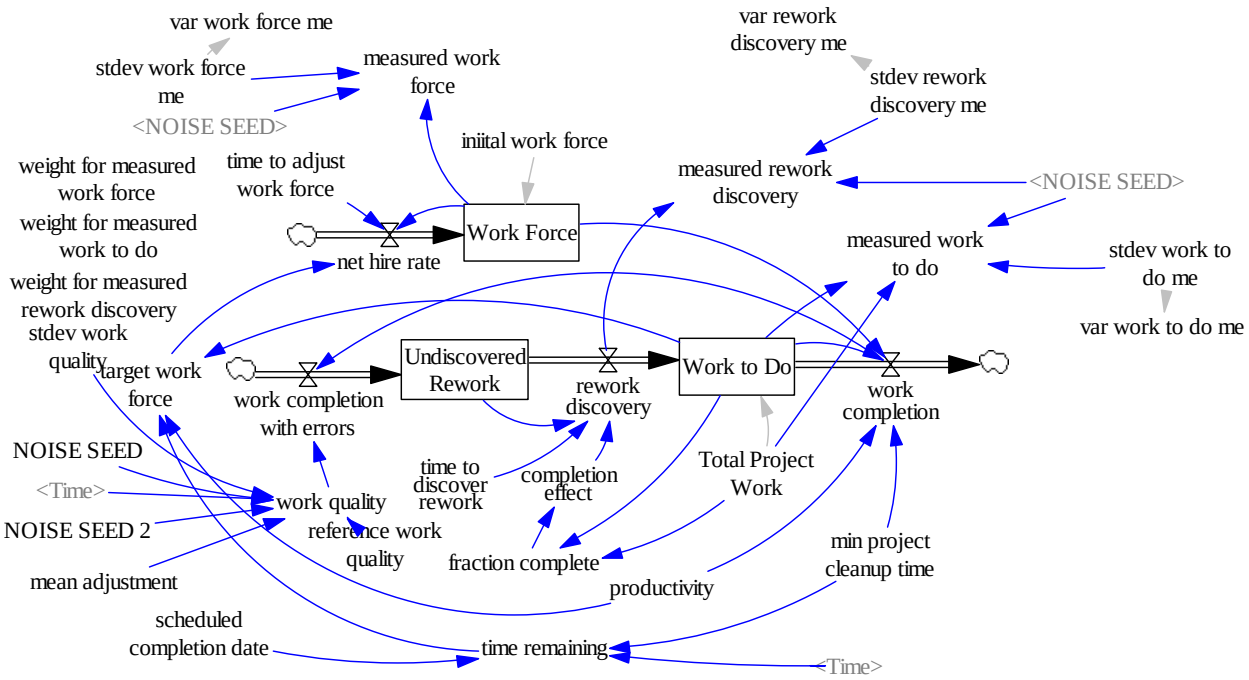

Figure 8

This is the model Project08.vmf run with “reference work quality” set to 1 and 0.85

Figure 9 and calibration

This is a run of Project08.vmf with FINAL TIME set to 30 and NOISE SEED set to 0. Note that when NOISE SEED is < 0 the model is set to run without any noise terms. Once that run has been made it us used as the data source.

The simple calibration is done by using that as the data source with the payoff file Project02.vpd and the optimization control file Project02.voc. Note that you will need to optimize once, create PayoffWeights.cin and optimize again. This time the optimization results will be different.

The Kalman Filter optimization uses ProjectKalman.vpd as the payoff file. This file has variances computed as supplementary variables in the model proper (and hidden in the diagram for the model). You will also need the file Kalman.prm which contains

```
Undiscovered Rework/1.7/0.1
Work Force/0/0.1
Work to Do/0/0.1
```

The driving variance in undiscovered rework 1.7 = (0.2 * 26 * 0.25)^2 where 0.2 is the standard deviation of the error on quality, 26 is an approximate average for the work completion rate and 0.25 is TIME STEP. There is no other driving variance, and the initial values are again small but approximations.

Figure 10

These are just values from the two previous calibration runs.

Figure 11

This sensitivity run is created by setting NOISE SEED to 0 (so that noise is active) and then varying NOISE SEED 2 which changes the trajectory after time 30. The sensitivity control file for this is Project06.vsc, but you will need to change NOISE SEED to 0 before launching sensitivity. The save list is in Project06.lst, but you can add to that.

Figure 12 and averages

This uses the model PortfolioValue00.vmf in the directory PortfolioValue. The deterministic run uses a NOISE SEED of -1, and the noisy run 99999. Note that this particular noise seed ends up demonstrating visibly volatile behavior toward the end of the simulation and this is why it was chosen. Repeating the experiments discussed in the paper with other noise seeds will usually results in estimates that are closer to one another for the different techniques.

Be sure to call the noisy run NoiseRun – this will be needed to run the 07 model.

The arithmetic average can be computed using Vensim's stats tool, in Excel or by hand. The geometric mean is simply the 100^{th} root of final/initial.

Figure 13

This is the model PortfolioValue07.vmf calibrated against the noise seed 99999 run (NoiseRun). It uses the payoff definition PortfolioValueNoWeight.vpd and the optimization control file PortfolioValue01.voc. Not that the noweight optimization control file uses 1 for a weight not a weight based on the error standard deviation. Since we are not going to be computing confidence intervals, this is all that is necessary.

Figure 14 and optimization

Zoomed in on the first 50 years for Figure 13.

The calibration with weights is done using the PortfolioValueWeight.vpd. The resetting optimizations are done by setting the variable "reset switch" to 1 and then using the weight or noweight payoff definition files.

Figure 15

This compares the weighted and unweighted state resetting optimizations as described above by zooming in on the last 15 years.

Figure 16 & 17

This work uses the model BasicPopulationPhysics05.vmf in the Population directory. To use this model you will need to load the external function file cohort_control08.dll that is contained in the ExternalFunctions archive that is part of this archive. There is documentation in that archive describing how to do this.

Before you can use the population model you will need to first run the model DataPrepJapan06.vmf. This model reads data from the Excel spreadsheet JapanDataOrganized01.xlsx and manipulates into a form useful for the population model. The resulting dataset should be called JapanOrganizedData.vdf.

The payoff definition file is BasicPopulationPhysics01.vpd and the optimization control file is BasicPopulationPhysics02.voc. This is again a 2 stage optimization in which the payoff weights are computed using err2weight after the first stage as described above. The two opzimizations are run by setting the constant “reset switch” to either 0 or 1.

Once the optimizations have been completed the projection is run by again setting the “reset switch” to 0 or 1, then setting Final Time to 2050 and including runname.out as a changes file where runname is the name of the optimization with or without state resetting. The .out file is the output of the optimization process and this file is configured so that it can also be used directly as a changes file.