Parameter and Confidence Interval Estimation in Dynamic Models: Maximum Likelihood and Bootstrapping Methods

Read Me File

Jeroen Struben, John Sterman, David Keith

This archive contains the R-scripts, as well as the files (models, runs, data, etc.) and instructions to replicate the examples and proposed exercises (challenge) of chapter 1. Details on how to use the utilities are provided in the "MLE APPROACH TO SIMULATION – Appendix.pdf" file in this archive as well as in the main text. For setup instructions see below (as well as the appendix)

Table A1. Documents, R-scripts, and data included in the online appendix

Document	Contents	Main R-script	Subordinate R-scripts used	Data
Main document	Theory Application Challenge	CH1_MLE_BOOT_ Application.R	CH1_MLE_Functions.R CH1_BOOT_Functions.R CH1_LR_Interval_Functions.R	ServiceQuality Data2.csv
Appendix	Further detailing of MLE theory, using linear model as an example	CH1_MLE_BOOT_ LinEx.R	CH1_LinEx_Functions.R	-
Challenge Solutions	Solutions to the challenge in the main document	CH1_MLE_BOOT_CHALLENGE.R	CH1_MLE_Functions.R CH1_BOOT_Functions.R CH1_LR_Interval_Functions.R CH1_Challenge_Functions.R	Beer Game Subject 1.csv
Required folder to save:		Scripts	Scripts	Data

All documents except the main document are provided in the electronic supplement on the publishers' handbook website. The electronic documents can also be requested from one of the authors (jeroen.struben@mcgill.ca).

Start instructions:

1. Create a work folder for your analysis. (You may use different work folders for the application and the challenge).
2. Within the work folder create three subfolders: "Scripts", "Data", and "BookChapterOutput".
3. Save the provided documents, listed above, in the appropriate subfolders ("Scripts" or "Data"). Save any of your R-script files in the "Script" folder.

Note: Figures will be saved in the folder "BookChapterOutput".

Chapter 2 Online Appendix

In the online appendix, full Matlab codes are provided to replicate the applied example and solve the exercise problem. The codes can be used for a typical MSM estimator and are based on the log of Matlab programming language; however, they are written in a way which is understandable for users of other programming languages as well. Files included in the online appendix are:

- *MSM_Applied_Example.zip*; including full matlab codes and a Vensim model (Figure 2.2).
- *MSM_Exercise_Obesity.zip*; including full matlab codes, data, a Vensim model, and a sample solution.

To run the Matlab codes, execute *RUN_MSM.m* file. Codes are written in different script files (files with the extension '*.m*') presented in Table 2.7.

[Table 2.7 near here]

Table 2.7: Matlab script files for the firms example

Script file	**Action**
RUN_MSM.m	Follows the MSM steps and saves estimated parameters, confidence intervals and J-test results

	in a '*.mat*' file.
UserInput_MSM.m	Includes the number of simulations, *K* in equation (2), and the number of simulations to estimate W^*, L_1 and L_2 in equation (4).
*UserInput_Model.m**	Includes model constants (e.g. the number of firms and Reference Resources in the firms example). It also includes the true values of the parameters (see Table 2.3). These values are only used to generate a data set out of which actual moments are extracted.
OptimizationInitiation.m	Includes the optimization tolerance (as a stopping criterion), the lower and the upper bounds for the unknown parameters, and the initial points to be used by the optimization solver. It also includes the choice of solver ('GlobalSearch' or 'MultiStart'). See Matlab Help for more information about optimization solvers.
*MomentSelection.m**	Selects the moments (e.g. mean of profits) from the data.
*EmpiricalMoments.m**	Executes the *FirmExample.m* and *MomentSelection.m* to capture the empirical

	moments.
*SimulatedMoments.m**	Executes the *FirmExample.m* and *MomentSelection.m* to capture the simulated moments.
FirmExample.m†	Executes two functions to run the firms example: *PinkNoise.m* and *FirmsModel.m*.
PinkNoise.m†	Generates the pink noise used in the firms example.
FirmsProfits.m†	Generates profits of the firms.
W1.m	Calculates a weighting matrix with diagonal elements of $W_{ii} = 1/(M_i)^2$. This weighting matrix is used in the first round of optimization.
Optimization.m	Runs the optimization solver based on user-provided information in the *OptimizationInitiation.m*. Note that the objective function for the optimization solver is *MSM_Obj_Fn.m* file.
MSM_Obj_Fn.m	Estimates simulated moments and then follows Equation (3) in the first round of optimization—it follows Equation (5) when W^* is estimated.
Weight.m	Estimates the weighting matrix (W^*) based on

	estimated parameters in the first round of optimization.
EstimatedVar.m	Estimates the variance-covariance matrix of the estimated parameters, see Equation (7).
ChangeParameters.m	Shifts estimated parameters one epsilon up and down. The output of this function is used in *Delta.m*.
Delta.m	Estimates the sensitivity of the simulated moments to the estimated parameters based on the outputs of *ChangeParameters.m*.
ConfInt.m	Calculates confidence intervals of the estimates parameters based on a confidence level (e.g. 95%).
J_test.m	Runs the J-test, see Equation (8).
SingularityWarningFlag.m	Checks for singularity and near singularity of the matrix that is being inverted. If the matrix is singular or nearly singular, a flag with value 1 is saved.
NumOfMomWarningFlag.m	Checks the number of moments vs the number of unknown parameters. Note that the number of moments should not be less than the number of

	unknown parameters; otherwise, a flag with value 1 is saved.

[*] Functions which are customized for the firms example.
[†] Functions which excusively present the dynamic model of the firms example.

Chapter 3: Structural Equation Modeling

This archive contains the R code example for the SEM package and results.

Contents

READ ME.doc
: This file.

Chapter 3 Appendix
: Code and explanation for exercise and results

R Code

Examples.R
: Code for importing data and estimating the models using `sem` package

Exercise 3.1.R
: Actual R code for the exercise

Vensim files

Simple regression.mdl
: Example of simple regression model shown in Figure 3.2

Simple regression.vdf
: Vensim data file from simulation of model shown in Figure 3.2

Simple regression.tab
: Tab delimited file of data from simulation of model shown in Figure 3.2 that can be imported into R.

Latent SEM model.mdl
: Example of simple regression model shown in Figure 3.3

Latent.vdf
: Vensim data file from simulation of model shown in Figure 3.3

Latent.tab
: Tab delimited file of data from simulation of model shown in Figure 3.3 that can be imported into R.

Latent SEM feedback simultaneous.mdl
: Example of simple regression model shown in Figure 3.4

Simultaneous.vdf
: Vensim data file from simulation of model shown in Figure 3.4

Simultaneous.tab
: Tab delimited file of data from simulation of model shown in Figure 3.4 that can be imported into R.

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:

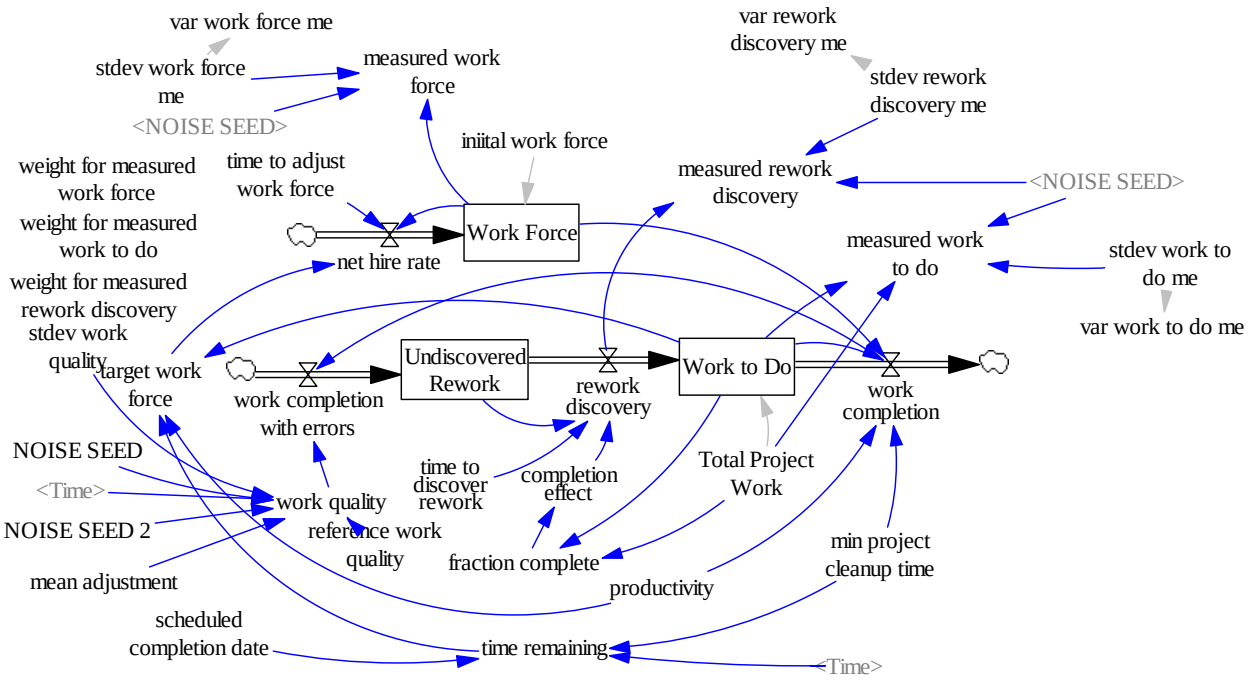

Figure 8

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.

Chapter 5: Combining Markov Chain Monte Carlo Approaches and Dynamic Modeling

The supplemental materials within this folder and its subfolders provide the steps associated with the example provided in the chapter and with the exercises posed, and supporting mechanisms for the chapter on using MCMC with Dynamic models. Broadly, these materials include a library bridging between R and Vensim, R code to ease the application of MCMC with Vensim models, definition of the example model in that framework, and R scripts for the specific steps undertaken in the chapter.

Contents:

- READ ME.pdf
  - This file.
- RVensimInterface
  - The bridge between R and Vensim, permitting 32-bit R to load, parameterize, run, and read values from Vensim models under Windows. This library has been verified to work with Vensim 5.1 and R.2.14 under Microsoft Windows 7 64-bit.
  - RVensimInterface8.dll
    - A dynamic link library (DLL) providing the low-level support for the bridge.
  - RVensimInterface8.c
    - The source code for the above. To be compiled using the following command from 32-bit R
      C:\Usask\Research\VensimMCMC\VensimRCompatibleDLL>"c:\Program Files\r\R-2.14.1\bin\R.exe" --arch i386 CMD SHLIB -L./ -lVenDLL32 RVensimInterface6.c
  - RVensimInterface5.R
    - The R stubs needed to interface to the C code.
- RVensimMCMCUtilities
  - Provides utility functions to ease the process of using R's MCMC libraries with dynamic models specified in Vensim. They are designed to simply multiple steps of the process, including finding the initial parameter vector, performing the chain, and summarizing and depicting results. Use of such functions can greatly reduce the amount of code required to perform MCMC in this way for a wide variety of MCMC problems, and allow for a considerably cleaner specification of such MCMC problems. Such utilities are used in the example provided
  - MCMCUtility v3.R
    - Defines the functions
- SEIRExample
  - Specifes the particular example model used in the chapter, as well as related exercises.
  - VensimSEIRModel
    - Defines the Vensim model used for both the example and exercises.
      SEIR Model v7.vpm
  - SEIRProbabilisticModel
    - Defines the probabilistic model accompanying the example. This defines the prior, sampling, and posterior distributions, and relates the empirical data to the SD model output as needed to compute the density of the latter two such distributions.
      Bounded SEIR MCMC v11.R
  - SEIR Chapter Commands v12 Finalizing
    - The commands used to actually perform the example and related exercises in the chapter.

Chapter 6. Pattern Recognition Electronic Supplement

Gönenç Yücel, Yaman Barlas

This archive contains the files and software necessary to conduct pattern-based model calibration, testing and behavior analysis as discussed in the chapter, as well as the instructions to setup and use these. Besides, the test model that is used in the chapter for demonstrative purposes can be found in this electronic supplement.

Contents

READ ME.pdf
: This file

Challenge.pdf
: Document providing the answer to the pattern-based model testing exercise and how SIS software can be used for that purpose.

Test Model (folder)
: This folder contains the test model in Vensim® format, as well as its detailed documentation

  TestModel.mdl
  : Model file

  TestModel Documentation (folder)
  : Folder that contains the model documentation prepared using the SDM-Doc tool.

Testing – SIS (folder)
: Folder that contains the SIS software that is used in the pattern-based model testing section of the chapter.

  SIS.zip
  : Archive file that contains SIS.exe

Calibration – POPS (folder)
: Folder that contains the Matlab® files that constitute the POPS model calibration system, as well as a manual that describes how to configure and use POPS

  Manual_POPS.pdf
  : Manual that describes how to configure and use POPS

  POPS.zip
  : Archive file that contains the Matlab® functions (.m files) that are used by for pattern-oriented model calibration

Behavior Analysis – BPS (folder)
: Folder that contains Pyhton files that are used for pattern-oriented behavior analysis with the BPC algorithm, as well as a manual that describes how to configure and use the BPC algorithm.

Manual_BPC.pdf

Manual that describes how to configure and use the BPC algorithm

BPC.zip

Archive that contains the Python code files (.py files) that are used by the BPC algorithm

Chapter 7. Eigenvalue Elasticity Analysis Electronic Supplement
Rogelio Oliva

This archive contains the Mathematica® packages and utilities to perform Eigenvalue Elasticity Analysis, as well as all the files (models, runs, etc.) and instructions to replicate the examples and proposed exercises in chapter 7. Details on how to install and used the utilities are on the Appendix.pdf file in this archive as well as in the main text of the chapter.

Contents

READ Me.pdf
: This file.

Appendix.pdf
: Document describing the use of the Mathematica® utilities to perform the EEA analysis as well as instructions on how to access the utility to translate Vensim® *.mdl files into documents readable by the Mathematica® utilities.

Challenge.pdf
: Document providing answers to the questions posed in the challenge section of the chapter and how the EEA tools could be use to address them.

Mathematica Tools (folder)
: This folder contains the functions and utilities to perform the EEA. The folder contains three files.

  FeedbackLoops.m
  : Mathematica® package with the core functions called by the LEEA and DDWA notebooks. It needs to be installed in a directory that is accessible to Mathematica®.

  LEEA.nb
  : Mathematica® notebook to perform the Loop Eigenvalue Elasticity Analysis.

  DDWA.nb
  : Mathematica® notebook to perform the Dynamic Decomposition Weight Analysis.

Base model (folder)
: This folder contains the files to replicate (assess) the LEEA of the base model as described in the chapter's main text. The folder contains five files.

  NF_model.mdl & nf_model.nb
  : Vensim® and Mathematica® readable version of the base model.

  Base.vdf & Base.tab
  : Vensim® data file and tab delimited version (readable by Mathematica®) with the values for the simulation of the base model.

  nf_base_LEEA.nb
  : Mathematica® notebooks with the results of the base model's LEEA.

Full model (folder)
: This folder contains the files to replicate (assess) the LEEA and DDWA of the full model as described in the chapter's main text. The folder contains eight files.

  NF_model_full.mdl & nf_model_full.nb
  : Vensim® and Mathematica® readable version of the full model.

  Full.vdf & Full.tab
  : Vensim® data file and tab delimited version (readable by Mathematica®) with the values for the simulation of the full model.

(cont. next page)

Policy.vdf & Policy (tai).vdf

Vensim® data files with the values for the Policy and Policy (tai) simulations of the full model.

nf_full_LEEA.nb

Mathematica® notebooks with the results of the full model's LEEA.

nf_full_DDWA.nb

Mathematica® notebooks with the results of the full model's DDWA.

Chapter 8. An Introduction to Stochastic Optimization

This archive contains the Powersim Studio (PS) and the compiled SOPS files for the optimization models in the chapter. The archive also contains the Users Manual for SOPS and all Powersim Studio models and compiled SOPS files used in the Users Manual. Powersim Studio and the SOPS program are available from Powersim Software: http://powersim.com/main/products-services/sops/

Contents

READ ME.pdf
- This file

Chapter models, folder
- FishContSimple2
  - ~si files, to be ignored
  - FishContSimple2.sip, PS fishery model
  - FishContSimple2, folder
    - FishContSimple2.sops, SOPS optimization model
    - FishContSimple2.csim, stored settings for optimization
    - model.results.txt, to be ignored
- Inverted pendulum
  - ~si files, to be ignored
  - Inverted pendulum.sip, PS standard pendulum model
  - Inverted pendulum, folder
    - Inverted pendulum.sops, SOPS optimization model
    - Inverted pendulum.csim, stored settings for optimization
    - model.results.txt, to be ignored
  - Inverted pendulumExtra.sip, PS pendulum model with feedforward
  - Inverted pendulumExtra, folder
    - Inverted pendulumExtra.sops, SOPS optimization model
    - Inverted pendulumExtra.csim, stored settings for optimization
    - model.results.txt, to be ignored
  - Inverted pendulumOfTime.sip, PS pendulum model with policy over time
  - Inverted pendulumOfTime, folder
    - Inverted pendulumOfTime.sops, SOPS optimization model
    - Inverted pendulumOfTime.csim, stored settings for optimization
    - model.results.txt, to be ignored

Users Manual.pdf, gives an introduction to Powersim Studio and SOPS

SOPS Users Manual models, folder

- FishCapacityCont.sip, PS models for continuous fishery model
- FishCapacityCont, folder
  - FishCapacityCont.sops, SOPS optimization model
  - FishCapacityCont.csim, stored settings for optimization
  - FishCapacityCont.xls, Excel file with graphs of results
  - model.results.txt, to be ignored
- FishCapacityDisc.sip, PS models for discrete time fishery model
- FishCapacityDisc, folder
  - FishCapacityDisc.sops, SOPS optimization model
  - FishCapacityDisc.csim, stored settings for optimization
  - FishCapacityDisc.xls, Excel file with graphs of results
  - model.results.txt, to be ignored
- FishContSimple.sip, PS models for continuous simplified model
- FishContSimple, folder
  - FishContSimple.sops, SOPS optimization model
  - FishContSimple.csim, stored settings for optimization
  - FishContSimple.xls, Excel file with graphs of results
  - model.results.txt, to be ignored

Chapter 9: Addressing Dynamic Decision Problems Using Decision Analysis and Simulation

The supplemental materials within this folder and its subfolders provide the java-based tool to integrate decision trees and Vensim models, the XML file encoding the decision tree presented, and the associated Vensim model, and output from that model.

Contents:

READ ME.pdf
: This file.

WNVModelForDecisionTreeV55.vpm
: The application example Vensim model.

DecisionTree-Java1-5PackagedJar.jar
: The software to integrate decision trees and Vensim. This software supports defining decision trees or loading such trees (where already specified), pruning such trees with a System Dynamics model specified in Vensim, and browsing the results. This software has been verified to work with Vensim 5.1 under Microsoft Windows 7 64-bit.

Example.xml
: Example decision tree for use with the software. This tree can be loaded into the software, and the tree evaluated.

Chapter 10 Electronic Supplement by Tan and Anderson

This archive contains the files that are used to create the examples in Chapter 10. Specifically, the contents include system dynamics model and sensitivity files (in Vensim®), decision tree files (in DPL®), Macro-enabled Excel template files with embedded VBA programs, and other Excel files (some with embedded @Risk functions) to calculate the middle steps of the algorithms.

Contents

ReadMe.pdf

This file.

System Dynamics Model Files (folder)

Chapter 10 analyzes two different versions of a renewable energy capital investment model. This folder contains the Vensim® files for the corresponding SD models as well as the sensitivity analysis files.

Chapter10-SDModel.mdl

Vensim® model file for the base model analyzed in Section 10.3.

Chapter10-ModifiedSDModel.mdl

Vensim® model file for the modified version analyzed in Section 10.4.

sens3.vsc

The file that contains the list of uncertain variables and their distributions for the Monte Carlo simulation of the SD model

final.lst

The list of output variables for the Monte Carlo simulation

newbatchfile.cmd

Vensim script file to run different decision sequences of the project model automatically. The file contains the changes to be made in the model for each decision sequence.

Excel Template Files with VBA code (folder)

This folder includes the Macro-enabled Excel template files to calculate the discrete distribution approximations for the SD-based decision tree algorithm. There are two templates because the code is different when the decision sequence starts with "Invest" (in which case, use "templateIEE.xltm") versus when it starts with "Delay" (in which case, use "templateDIE.xltm").

templateIEE.xltm

Macro-enabled Excel template file to calculate the discrete distribution approximations for the continous cash flow distribution obtained from the SD model. Used for the decision sequences that start with the decision "Invest" (see Table 10.3).

templateDIE.xltm

Macro-enabled Excel template file to calculate the discrete distribution approximations for the continous cash flow distribution obtained from the SD model. Used for the decision sequences that start with the decision "Delay" (see Table 10.3).

Decision Tree Files (folder)

This folder includes the decision tree files created in DPL® for the examples in Chapter 10.

SD-basedDecisionTreeApproach.da

The DPL® file for the *SD-based decision tree algorithm* corresponding to the base model in Section 10.3.

SD-basedDecisionTreeApproach-ModifiedModel.da

The DPL® file for the *SD-based decision tree algorithm* corresponding to the modified model in Section 10.4.

DiffusionApproximationApproach.da

The DPL® file for the *diffusion approximation approach* corresponding to the base model.

DiffusionApproximationApproach-ModifiedModel.da

The DPL® file for the *diffusion approximation approach* corresponding to the modified model.

Diffusion Approximation Algorithm Files (folder)

This folder includes the files to calculate the middle steps of the diffusion approximation algorithm.

CalculationForPVandCashFlowPayout.xlsx

This file is used to estimate the present value (PV) of the cash flow, the remaining project value (V_t) for period *t* as defined in Equation 10.1, and the cash flow payout rate (δ_t) as defined in Equation 10.2.

FileToSimulateGBMprocess.xlsx

This file uses the software @Risk to simulate a Geometric Brownian Motion (GBM) process using the PV, and δ_t calculated in the file CalculationForPVandCashFlowPayout.xlsx. The analyst has to enter a

value for volatility (sigma) to simulate the process. The goal is to simulate GBM processes for different values of volatility and then finding the volatility that minimizes the difference between the GBM process and the cash flow distribution obtained from the SD model.

CalculationForVolatilityEstimation.xlsx

This file includes a template to estimate the volatility of the GBM approximation. Specifically, for different values of volatility, the file compares the 10^{th}, 50^{th}, and 90^{th} percentiles of the GBM approximation and the cash flow distribution from the SD model. Then, the squared errors are calculated and summed. The volatility that gives the minimum total squared error is chosen. Note that this is not an automated process. The analyst has to obtain the GBM percentiles for each value of volatility one by one using the "FileToSimulateGBMprocess.xlsx" template and then paste the data into the corresponding worksheet.

Chapter 11. Optimal Control Theory Electronic Supplement

Edward G. Anderson Jr., Nitin R. Joglekar

This archive contains the Mathematica® file (termed a notebook) that solves the example problem in the exercise at the end of the chapter as well as the instructions on how to use it.

The notebook file is self-documented to guide you through using optimal control theory techniques discussed in the chapter to solve the exercise. The notebook was built in Mathematica v. 8.0.4.0. If you are at an academic institution, you are already likely to have free access to it. However, if not, Wolfram (www.Wolfram.com), which publishes Mathematica, also has free software to read notebook files, although you cannot change any of the operations. Their current reader is called CDF-Player and can be found at http://www.wolfram.com/cdf-player/.

Contents

READMe.pdf

This file.

Example Ricatti.nb

This contains the self-documented file on how to use the Ricatti equation to solve the example problem in the exercise section. You can adapt this file to solve many other problems using optimal control theory as well by changing the system and control matrices.

Ch 13 Online Appendix Contents