Extending the jPhase module of jMarkov project

Degree Project

Presented to the Industrial Engineering Department

by:

Andrés Sarmiento Romero

Adviser:

Raha Akhavan Tabatabaei

To qualify for Industrial Engineering degree

Universidad de los Andes

January 2013

A toda mi familia, especialmente a mi papá y a mi mamá, quienes toda la vida me han dado un apoyo incondicional en los momentos difíciles y me han sabido entender y formar. A mi hermana, quien siempre ha estado junto a mí brindándome su apoyo. A Juanita, por su paciencia y amor.

Contents

1. Introduction

Since 2002, the research group COPA of the Universidad de los Andes has been developing the jMarkov project. jMarkov is a stochastic modeling package developed in Java. In this project the principal characteristics of a Markov chain are defined in order tohelp the user focus on modeling instead of mathematical analysis [19].

The current version of jMarkov incorporates four modules: jPhase, which models Phase-tType (PH)distributions; jMarkov, which defines the components of a Markov chain; jQBD, which models quasi-birth-death (QBD) processes; and jMDP, which models Markov Decision Processes (MDP) [7].

The jPhase module is supported on a set of interface, abstract and implemented classes. The interface classes determine the behavior of a PH distribution, these classes define continuous and discrete PH distributions. The abstract classes implement the most general methods of the previous interfaces. Finally, the implemented classes extend the abstract classes and are useful if the user is going to use only the general properties of a PH distribution, such asthe dense and cumulative functions [14].

One special package of jPhase is jFit, which contains Maximum Likehood and Moment Matching algorithms for fitting data to a PH distribution. Additionally, there is a graphical interface where the user can fit data in order to create new variablesthat jPhase will show graphically [15].

The jMarkov module is a set of three packages: Build package, Basic package and Solver package. The Build package is the main package of the module since it contains the classes that take care of building the state-space and transition matrices. The Basic package contains the building blocks needed to describe a Markov Chain. It contains classes such as State and Event, which allow the user to code a description of the states and events, respectively. The Solver Package is a collection of Markov Chain Solvers. JAMA [10] and MTJ[9] algorithms are implemented as solvers. Interested readers can refer to the jMarkov manual [19] for more information.

jQBD package extends the basic capabilities of the jMarkov package to allow modeling highly structured infinite-space systems of the type quasi-birth-death processes (QBD), [18].

jMDP is used to build and solve Markov Decision Process (MDP). Probabilistic Dynamic Programming allows the analyst to design optimal control rules for a Markov Chain. jMDP works for discrete and continuous time MDPs. For details see [20].

This project is the result of some deficiencies or limitations found in the jMarkov project. Specifically, we found that jPhase graphical interface can be enhanced with new facilities to improve the user experience. The first limitation found is that the fitting algorithms only use files as input. These files may contain a list of data to which the distributions must be fitted. This means that the user cannot perform a fit if he does not possess a data file,even if he knows the theoretical distribution. The second deficiency found is that, after a fitting process, the user does not get any retrieval of the fitting quality. This means that the user does not have a measure ofhow good the new PH distributionrepresents the initial dataset. Finally, an enhancement opportunity is that, given the fact that jPhase graphical interface shows a list of PH distributions, the user could generate a queuing model using these distributionsas the inter-arrival or service times.

The objective of this project is to develop new features that will deal with the aforementioned limitations and improvements. These features are:

1. A random variate generator and moment calculator. In the jPhase graphical interface the user can generate random variates with a specific distribution or calculate the moments of thatdistribution to use them as input for the fitting process. As a complement to this, we implement a new tool to perform goodness-of-fit test, required to evaluate the quality of the generated variates. This features can be found in the Random Variate Generator and Moment Calculator Panel.
2. A tool for generating a queueing model where the inter-arrival and service time distribute PH, for a single-server queue(also called PH/PH/1). The package will solve the queueing model in order to calculate the performance measures.This features can be found in The PH/PH/1 Model Panel.
3. A goodness-of-fit tester for PH distributions. The package provides the facility to perform a goodness-of-fit test specific for PH distributions. This feature can be found in the Goodness-of-Fit Test for Phase-Type Distributions Panel.

The present document is organized as follows. Section 2 gives a brief theoretical review of the main concepts needed to develop and understandthe newfeatures. Readers who are not interested in the theoretical background of the features can skip this section. Section 3 is designed as a user manual of the new features. Also, the screenshots of these implementations are demonstrated in this section. Section 4 provides the methods that are used to verify the newly added features. Section 5concludes the document with the results of the project and future works for the jMarkov project.

2. Theoretical Background

The objective of this section is to summarize the theoretical background behind the implemented features. Additionally, there will be an explanatory section for the Stochastic Simulation in Java library (SSJ) which was used in this project. This project uses concepts of Phase-type distributions, QBD, queuing models and goodness-of-fit tests.

2.1 Phase-Type Distributions

A phase-type (PH) distribution is a combination of exponential variables with occurrence on sequence or phase[3]. The importance of PHdistributionsin queueing theory lies on the fact that theoretically, any distribution can be arbitrarily well approximated by a PH distribution [4].In adition to this, the PH distributions have very interesting properties with respect to other distributions. Some of theseproperties are that the maximum, the minimum and the sum of two or more PH distributions also distribute PH.

The parameters of a PH distribution are: a probability vectorand a characteristic matrix[16]. We use the notation $A \sim PH(\Lambda, \alpha)$ to describe that a variable$A$ distributes PH with a probability vector α anda characteristic matrixΛ.

There are seven subclasses of PH distributions: exponential PH, erlang PH, hipo-exponential PH, hyper-exponential PH, hyper-erlang PH, acyclic PH and general PH distribution. We will give a brief description of: exponential PH, Erlang PH andan example where a hyper-exponential PH distribution equals a exponential distribution. We describe these distributions because we use it later, but interested readers can see [23] for more information.

The exponential PH distribution is the representation of an exponential distribution but as a PH distribution. This exponential PH distribution is compused by the number one as its probability vector and the rate vector is the rate of the exponential distribution.Similar to the exponential PH distribution the erlang PH distribution represents anerlang distribution. An erlang PH distribution that represent anerlang (λ, k) has a probability vector of size k with the number 1 in the first position and 0 in the rest of the vector, and the characteristic matrix has $-\lambda$ in the diagonal and λ in the cells that are just over the diagonal.

A hyper-exponential PH distribution is represented by a group of N branches, each one with anasocciatedexponential variables with rate λ_i and a probability $p_i, \forall\, i \in N$ where $\sum_{i \in N} p_i = 1$. Anexponential variable is equal to ahyper-exponential PHvariable with one branch or to a hyper-exponential PHvariable with the same rate for all its branches. In the Figure 1 the reader can see an example of a hyper-exponential PH distribution. In this example the time before absobsion would distribute as: an exponential $(\lambda = 5)$ distribution with probability of 0.3, an exponential $(\lambda = 10)$ distribution with probability of 0.2 and so on.

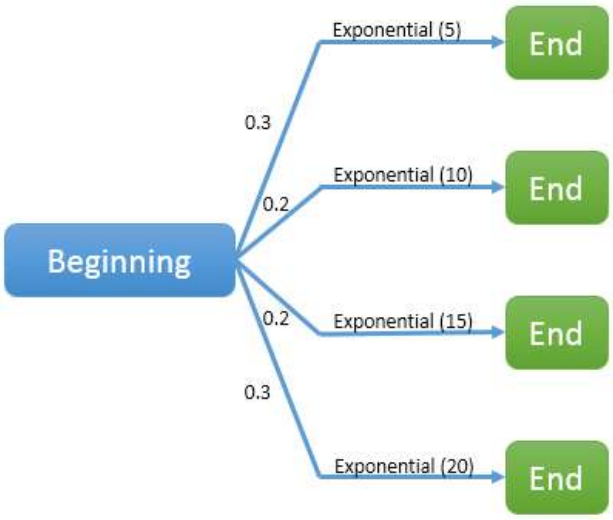


Figure 1. Hyper-Exponential Example

2.2 Quasi-Birth-Death Processes

In queuing theory a quasi-birth-death process (QBD)is a particular case of a continuous time Markov chain, where the state describes not only the number of entities in the system but also the current inter-arrival or service phase of an entity in the system. Having a PHdistributionthat describes the inter-arrival or service time, a QBD can be built in order to find the performance

measures of the system. Some QBD models and examples can be found in [13] and [16]. The generator matrix of a QBD process is of the form:

$$Q = \begin{bmatrix} B_1 & B_0 & 0 & 0 \\ B_2 & A_1 & A_0 & 0 \\ 0 & A_2 & A_1 & A_0 \\ 0 & 0 & A_2 & A_1 \end{bmatrix} \qquad \text{(Equation 1)}$$

As seen in Equation 1 the QBD generator matrix can be represented by six sub-matricesB_0, B_1, B_2, A_0, A_1, A_2. The first three matricesB_0, B_1, B_2 represent the transitions that affect the initial states. The second group of matrices refers to transition rates between non- initial states.

2.3 The PH/PH/1 Model

In this section we describe the process to build a queueing model where the inter-arrival and service time distribute PH (also called the PH/PH/1 queue).Building the model of the system allows us to find the performance measures of the PH/PH/1 model.

Similar to the M/M/1 queue(which has exponential inter-arrival and services times) with an associated birth and death process [25], PH/PH/1 systems have an associated QBD process. As we shown in Section 2.2 a QBD process is characterized by six sub matrices: A_0, A_1, A_2, B_0, B_1, B_2. Having two PH variables called A and B, where$A \sim PH(\Lambda, \alpha)$,and $B \sim PH(\mathrm{M}, \beta)$,that represent inter-arrival and service time respectively, the six sub matrices can be calculatedas follow [13] (The definition of the Kronecker product are also defined in Equation 3):

$$\begin{gathered} A_0 = \lambda\alpha \otimes I, \quad A_1 = \Lambda \oplus \mathrm{M}, \quad A_2 = I \otimes \mu\beta \\ B_0 = \lambda\alpha \otimes \beta, \quad B_1 = \Lambda, \quad B_2 = I \otimes \mu \\ \lambda = -\Lambda e, \quad \mu = -\mathrm{M}e \end{gathered} \qquad \text{(Equation 2)}$$

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{bmatrix} \qquad \text{(Equation 3)}$$

$$A \oplus \mathrm{B} = \mathrm{A} \otimes I_b + I_a \otimes B$$

Having the sub-matrices we can find the matricesR, U, G and Ndefined recursively as:

$$R = A_0 + RA_1 + R^2A_2 \qquad \text{(Equation 4)}$$

$$A_2 + A_1G + A_0G^2 = 0$$

$$U = A_1 + A_0 G$$

$$N = (I - U)^{-1}$$

In order to find the performance measures, these matricesare important because they define the balance and normalization equations. As some of these matricesare defined recursively (see Equation 4) there is no direct way to calculate them. Latouche and Ramaswami [17]define a very efficient iterative algorithm to find an approximate matrix that solves the previous recursive equations. This method is called Logarithmic Reduction [17].There are other algorithms like Modified Boundary [16]that can be used to find these matrices as well. Currently jMarkov implements these two algorithms.

Since the existing classes in jMarkov receive as input a Geometric Process (this is the set of events, states and rates that define a Markov Chain)and not the six characteristic matrices or the PHdistribution, it isnecessary to generate new classes for them.The reason why these classes receive a Geometric process is because they are designed to solve general MarkovChains and not specific queueing models. For more information about the existing classes or Geometric Processes see[18].The new classes receive as parameters two PH distributions, calculate the six characteristic matrices and return the R matrix.

We define the π_i vector $\forall i \geq 0$, as the set of probabilities of being in the states where there are i entities in the system. For example,π_0 is the vector where there are no entities in the system. The number of states for a specific number of entities depends on the number of phases of the inter-arrival and service time.

In order to get the performance measures it is necessary to find the $\pi_0$probabilities. We define the vectorπ_i(Equation 5) and the balance and normalization equations (Equations 6and7 respectively)as follows:

$$\pi_i \mathbf{1} = the\ probaility\ that\ \ the\ system\ contains\ i\ units \qquad \forall i \geq 0 \quad \text{(Equation 5)}$$

$$\pi_0(B_1 + B_0 N B_2) = \mathbf{0} \quad \text{(Equation 6)}$$

$$\pi_0(1 + B_0 N (I - R)^{-1}\mathbf{1}) = 1 \quad \text{(Equation 7)}$$

Equation 6 has n equations (where n is the size of the π_0 vector) and Equation 7 has 1. Thiscauses that the previous system has $n + 1$ equations and n variables. As in the first system of

equations there is at least one linearly dependent equation, we can take off any of them and replace it with the normalization equation. We need$\pi_0$because we use it to find anyπ_i using Equations 8 and 9. We solve this linear system in order to getπ_0.

$$\pi_i = R^i \pi_o \quad \forall i > 1 \qquad \text{(Equation 8)}$$

$$\pi_0 = \pi_0 B_1 + \pi_1 B_2 \qquad \text{(Equation 9)}$$

The final step is to calculate performance measures such as: the expected number of units in queue, in service and in the system, and the average time in queue, in service and in the system.We use Equations 10-15 to find the performance measures.

$$L = \sum_{i \in Z^+} i\pi_i \mathbf{1} = Expected\ number\ of\ entities\ in\ the\ system \qquad \text{(Equation 10)}$$

$$L_q = \sum_{i \in Z^+} (i-1)\pi_i \mathbf{1} = Expected\ number\ of\ entities\ in\ queue \qquad \text{(Equation 11)}$$

$$L_s = (1 - \pi_0 \mathbf{1}) = Expected\ number\ of\ entities\ in\ service \qquad \text{(Equation 12)}$$

$$E[A] * L_q = Average\ time\ in\ queue \qquad \text{(Equation 13)}$$

$$E[B] = Average\ time\ in\ service \qquad \text{(Equation 14)}$$

$$E[B] + E[A] * L_q = Average\ time\ in\ the\ system \qquad \text{(Equation 15)}$$

As the reader can see, the summationsof theEquations 10 and 11are defined in the integer domain (which is not finite) we sum until a number $j$when the cumulative function in j (see Equation 16) is greater than 99.99%.

$$\sum_{i=0}^{j} \pi_i \mathbf{1} \qquad \text{(Equation 16)}$$

2.4 Goodness-of-Fit Tests

A goodness of fit test is an analysis to testthe hypothesis that a dataset distributes as a selected distribution. In assessing if a distribution is suited to a dataset there are many different tests that can be performed. Some of these tests are: the Kolmogorov-Smirnov test [8], the Chi-Square test [12], Crámer-Von Mises test [1], Shapiro-Wilk [22] and Andreson-Darling test [2]. We only perform the Kolmogorov-Smirnov and the Chi-Square testsin this work.

The Kolmogorov-Smirnov test measure the distance of the data sample distribution with the theoretical distribution. The test builds an empirical cumulative distribution function and measures

its distance with the expected one. Interested readers can refer to[8] for more information. The Chi-Square test divides the data sample in groups and measures the distance to the expected distribution. This measure is a function of the difference between the observed number of occurrences in each group against the expected number of occurrences. For more information see [12].

2.5 Stochastic Simulation in Java

The SSJ package is a Java library for stochastic simulation [11]. Among many features, SSJ provides facilities for generating random variates for the common distributions and performing goodness-of-fit tests. Since SSJ has codedmany distributions and also many methods to generate them, we use this package in our work.This is very useful not only for generating variates but also for performing goodness-of-fit tests.

The process to perform a goodness-of-fit test in SSJ is composed of two steps:

1) Transforming all data in the file to an uniform distribution that represents the behavior of the dataset. To do this, it is necessary to use the cumulative distributionfunction for each data point in the file, and this array of probabilities is used as a uniform distribution.
2) Performing a Chi-Square test and a Kolmogorov-Smirnov test with the new set of uniform variates to test that the transformed data distributes uniformly.

3. Manual for the New Features

This section is designed to serve as amanual for the newly addedfeaturesto the jPhase graphical interface.

To run the jPhase graphical interface the user needs to initialize a ***FramePrincipal*** object and set it visible (see Figure 2).

```
14 public class JPhaseStarter {
15
16     /**
17      * @param args
18      */
19     public static void main(String[] args) {
20         FramePrincipal fp = new FramePrincipal();
21         fp.setVisible(true);
22     }
23
24 }
25
```

Figure 2. jPhase graphical interface starting code

After setting the ***FramePrincipal*** visible, a new frame will appear. The features are in the ***Phase Var***menu (see Figure 3).All the features have an associated panel and we will present a description of each of themfollowed by an example.

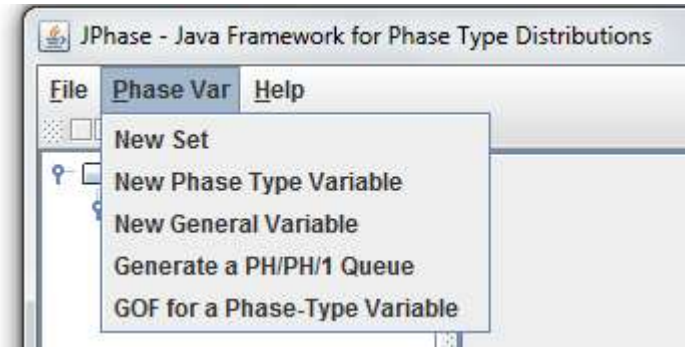


Figure 3. JPhase "Phase Var" menu

3.1 Random Variate Generator andMoment Calculator Panel

Thisfeature can be found in the ***Phase Var*** menu, in the ***New General Variable*** option (see Figure 3). After selecting thisoption, the software generates a new panel which contains two tabs: one for the variate generation and moment calculator (see Figure 4) and one for the goodness-of-fit tests (see Figure 8).Each tab is explained below.

3.1.1 Random Variate Generator and Moment CalculatorTab

3.1.1.1 Description

As described in Section 2,Version 2.5 of the SSJ library is implemented for the random variate generation and the goodness of fit test performer.

In SSJ for generating random variates it is necessary to define a random uniform generator (from a list of possible generators), a distributionand agenerator method for that distribution. In the developed interface the user can define (See Figure 4):

- The random uniform variates generator.In the first *change* button the user can change the generation method which uses a default one.
- The expected distribution and its parameters, which can be changed in the second *change* button.
- The generator method for the specificdistribution. Only the normal distribution has more than one option and it is located in the third *change* button.
- A location folder for saving the data filethat will serve asthe input for jPhase. The browserfor selecting the destination folder is located in the *select* button.

The software can either generate the variates or calculate the moments of a distribution. In that case the user only needs to define a distribution and a location for the filethat will contain the moments.

As a result of the generation, the software shows a message with a summary of the process, which contains the number of variates generated, the distribution, and the location of the file (Figure 7). After that, the software automatically performs a goodness-of-fit test comparing the generated data filewith the distribution (Figure 9).

3.1.1.2 Example

In this section we will describe the process to generate 1000 Gammavariates with rate $\lambda = 5\ and\ \alpha = 1$.

1) Select the***New General Variable***option in the jPhase graphical interface.
2) In this example we will change the uniform random generator method (the user can use the default one). First,click on the first***change***button (next to the uniform random variate generator label) and the menu shown inFigure 5 will appear. Choose***Well607*** option (if the user wants more information about any algorithm she must click on the button next to the algorithm and a new window will be visible with the description) andthen click on the***accept*** button.
3) In order to change the distribution, click on the second***change***button (next to the distribution label) and the window shown in Figure 6will appear. The user must select the

Gammaoption and click ***accept***.In the *Parameter* windows write the $\alpha = 1 \; and \; \lambda = 5$ rates.Click on ***accept***.

4) Click on the ***Select*** button and choose a location for the data file. In this case we choose "C:\\jMarkov\".
5) Click on ***Generate Variates*** button. After this, the Figure 7 message should appear. Click ***Ok***.
6) Finally the Figure 9 panel should appear. The user can save the histogram image or click on ***Ok***to go back to the *Generator* tab.

The result of this procedure is a file in the selected location with 1000 variates.

If instead of generating variates the user wants to calculate the moments she can **skipstep 2** and in **step 5**click on***Calculate Moments.***

3.1.2 Goodness-of-Fit Test Tab

3.1.2.1 Description

To perform a goodness-of-fit test the user has to define the number of groups for the Chi-Square test, the theoretical distribution, and alocations for the data file (See Figure 7). In Section 2 the goodness-of-fit test procedures are briefly described. For this sections we also perform the tests using SSJ. In Section 2 the reader can find a brief explanation of the procedure to perform the tests developed by SSJ.

As a result of the test, the software shows a new panel with a summary of the fitting and a histogram comparing the Cumulative Distribution Function (CDF) against the data histogram. The summary contains the P-Values for Kolmogorov-Smirnov and Chi-Square tests.

3.1.2.2 Example

To perform a goodness-of-fit test for the dataset that we created on the previous section, we have to:

1) Open the Random Variate Generator and Moment Calculator Panel.
2) Change to the***Goodness-of-Fit*** **tab** (see Figure 8).
3) Set the number of groups for the Chi-Square test to**10**.
4) Click on ***Select*** button and browse the data file.

5) To change the theoretical distribution click on ***Change*** button and select ***Gamma***.
6) To perform the test, click on the ***Accept*** button. A results windows shown inFigure 9shouldappear.

As the result of thistest we can see that both Kolmogorov-Smirnov P-value and Chi-Square P-value are greater than 5%. Therefore,we cannot reject the hypothesis that the dataset follows aGamma distribution.

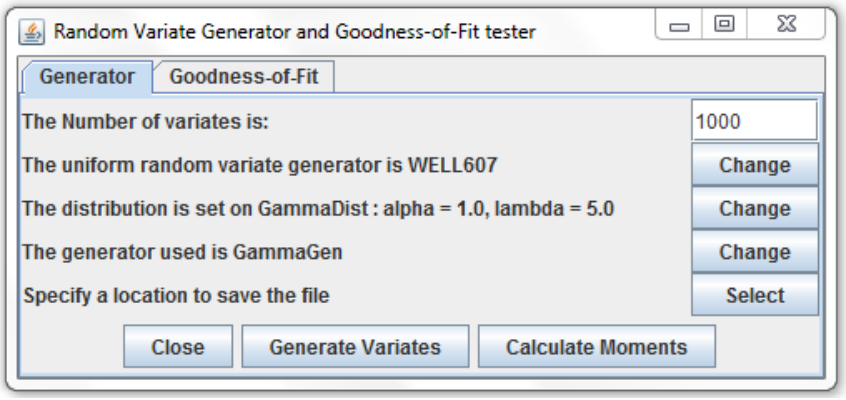


Figure 4. Variate Generator and Moment Calculator panel

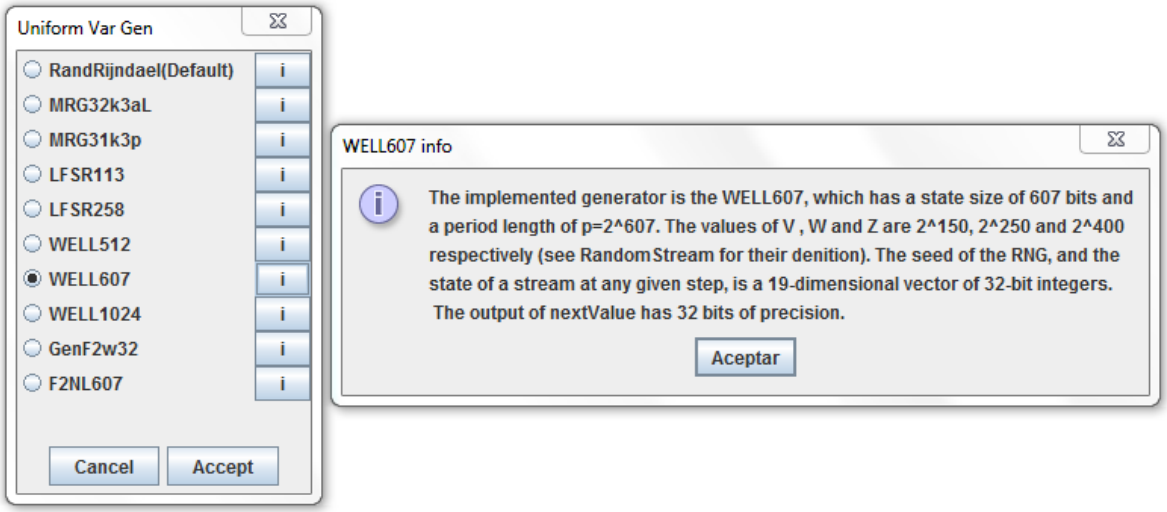


Figure 5. Uniform Random Variate Generator Panel (make sure the caption falls on the same page as the figure)

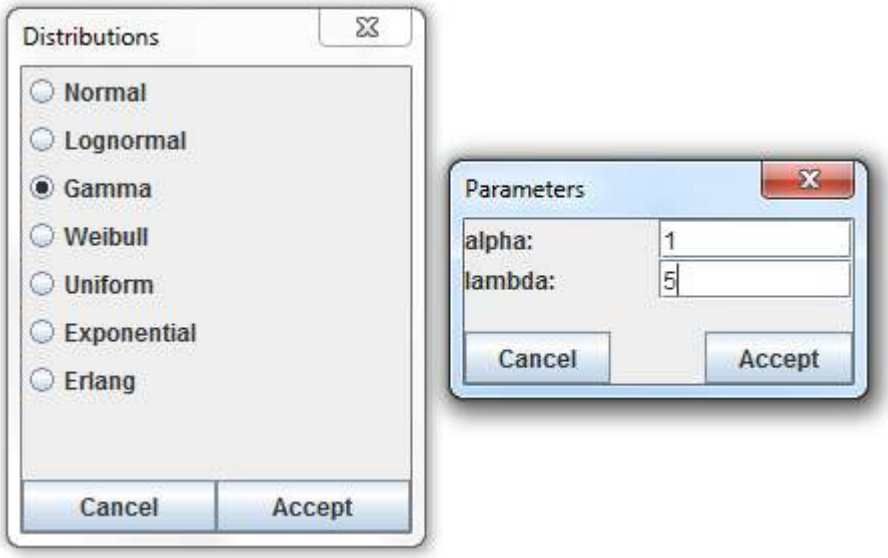


Figure 6. Distribution and Parameters Selection Panel

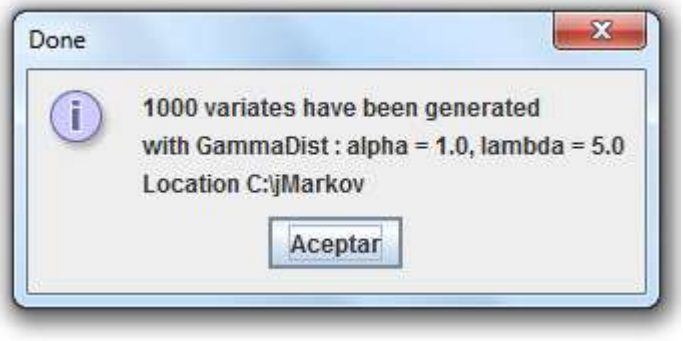


Figure 7. Generation Summary

Figure 8. Goodness-of-Fit Test Panel

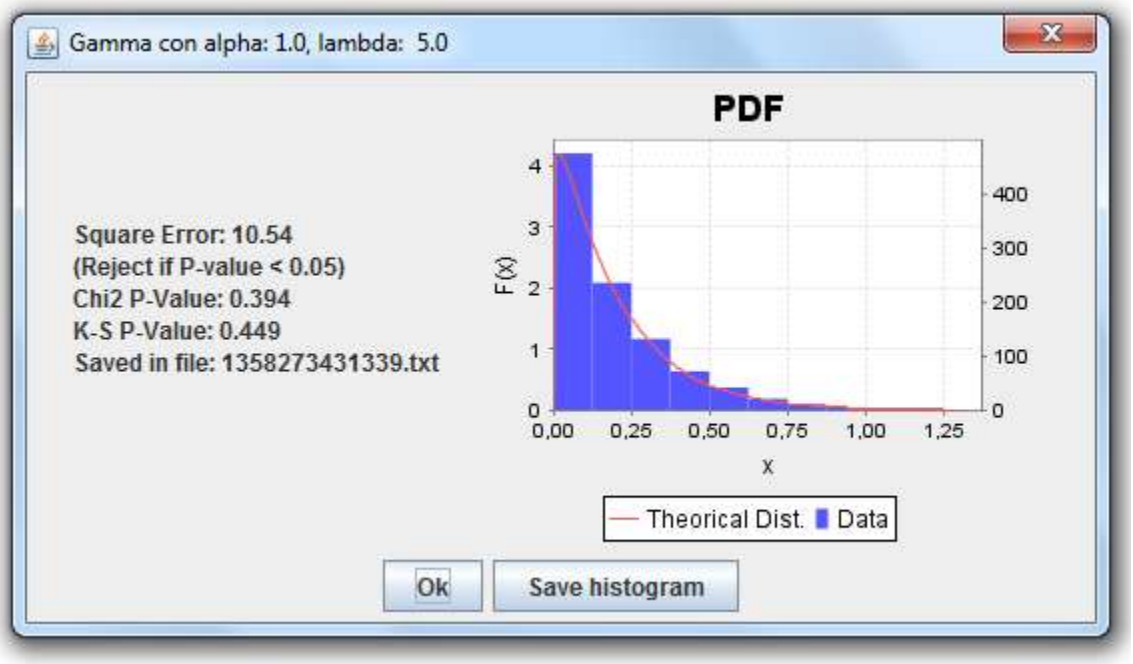


Figure 9. Goodness-of-Fit Test Result Panel

3.2 Goodness-of-Fit Test for Phase-TypeDistributions Panel

3.2.1 Description

The goodness-of-fit test for PHdistributionsis similar to theone implementedfor the common distributions. We complemented the tests for common distributions provided by SSJ by adding the methods necessaries to perform tests for PH distributions. For more information about how the test isperformed see Section 2.5.

This features can be found in the jPhase graphical interface in the ***Phase Var*** menu, option ***GOFfor a Phase-Type variable*** (see Figure 3).This feature is designed in a panel similar to the goodness-of-fit tests for common distributions. The panel is composed of a text field to set the number of groups for the Chi-Square test,a button to select the location of the data and a list selector to choosethe theoretical PH distribution (See Figure 10). The test is performed after clicking on *Accept* button.

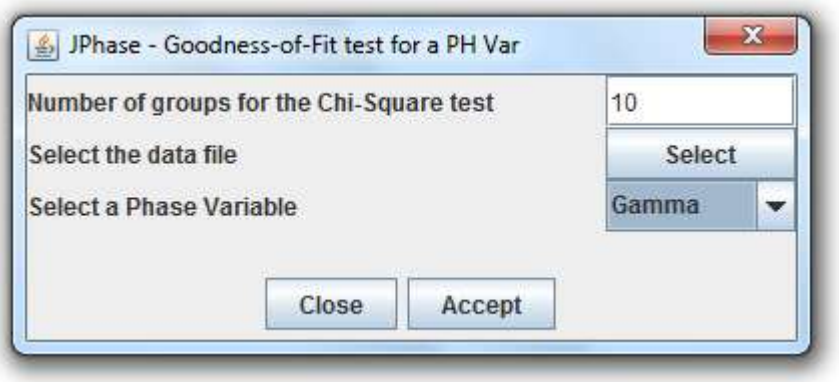


Figure 10. GOF test parameters panel

3.2.2 Example

In this example we are going to test if a dataset distributes as a specific PH distribution. This dataset canbe the same one that we createas example in Section 3.1.1and the theoretical PH distribution is a new PH variable that we fit with jPhase usingthe dataset as input. To perform the test we have to follow these steps:

1) In the jPhase graphical interface create a PH variable with the name Gamma. To do it we open New Phase-Type optionset a name and in the fitted variable menu we select EMPhaseFit. Click on accept.
2) Open the panel of this feature.
3) We are going to use **10** as the number of groups for the Chi-Square test, so it is not necessary to change this parameter.
4) Click on ***Select*** button and choose the location of the file(in this case, the file that we created in Section 3.1.1).
5) Select the ***Gamma*** option.
6) Click on the ***Accept*** button.

After this, the panel of the Figure 11 will appear. We can see that in this case the result is that the Chi-Square P-Value and the Kolmogorov-Smornov P-Value are greater than 5%. This means that the we cannot reject the hypotesis that the dataset distributes as the PH distribution.

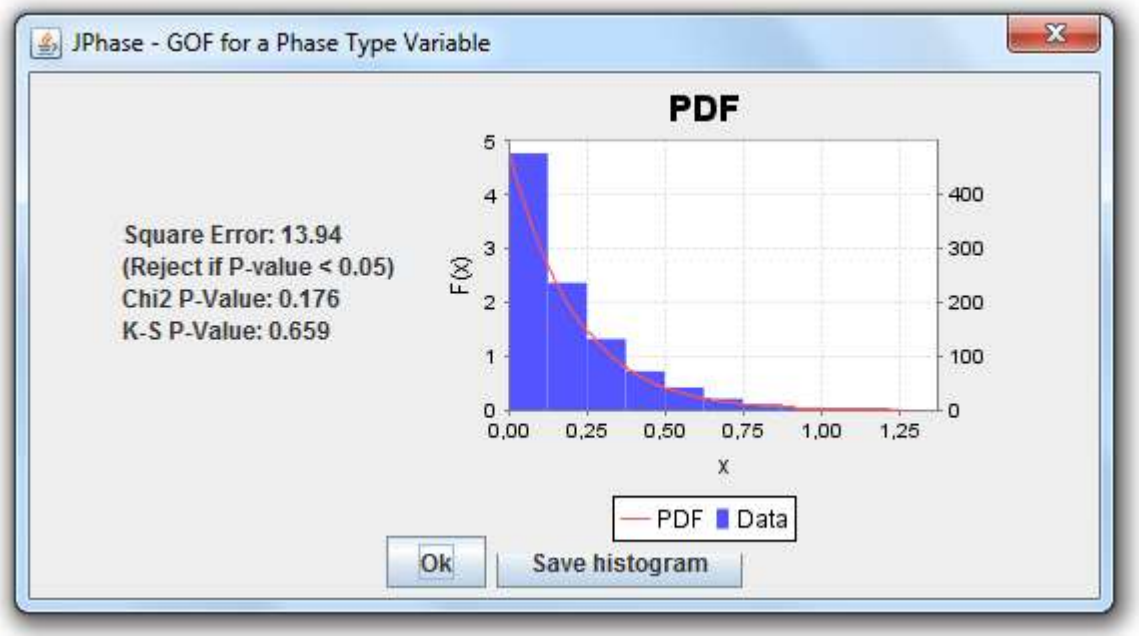


Figure 11. GOF test result panel

3.3 The PH/PH/1 Model Panel

3.2.1 Description

jPhase's graphical interface allows users to define any number of PHdistributionsin order to list them, show some statistics and draw theirprobability density and cumulative distribution functions. In order to enhancethe jPhase graphical interface this new functionality allows users to build a PH/PH/1 model which uses the list of distributionsas input. The process by which the model is built is shown in Section 2.

The panel is composed of two listsfrom where the user should choose the inter-arrival time and service time.

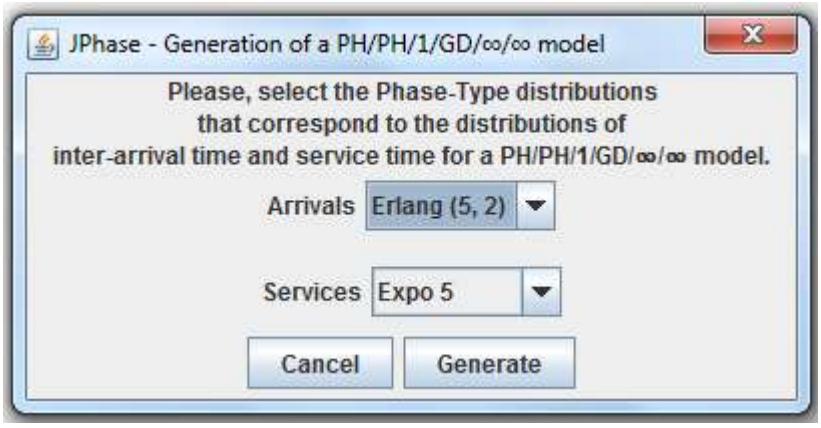


Figure 12. Generation of a PH/PH/1 model

3.2.2 Example

In this example we will generate and analyze a PH/PH/1 queue. The time between arrivals distributes erlang with parameters $\lambda = 5,\ k = 2$ and the service time distributes exponentially with rate $\lambda = 5$. See Section 2 for more information about these distributions. To generate the model we need to follow these steps:

1) In the ***Phase Var*** menu, select ***Generate a PH/PH/1Queue*** option. After this, the panel shown in Figure 12should appear.
2) This panel is composed of twolists to choose from. The first list refers to the inter-arrival time distribution and the second to the service time distribution. In the first list select $\boldsymbol{Erlang}\ (\mathbf{5}, \mathbf{2})$ and in the second select$\boldsymbol{Expo}\ \mathbf{5}$.
3) Click on ***Generate***

As result, a panel as shown in Figure 13 appears. In it, we can see the performance measures described in Section 2 which are the performance measures of the PH/PH/1 model.

As an additional example we perform a PH/PH/1 module using the variables of the Figures 16 and 17 named A and B respectively. In Figure 14 we show the Panel with these variables and in Figure 15 we present the performance measures of the model.

Figure 13. Performance Measures

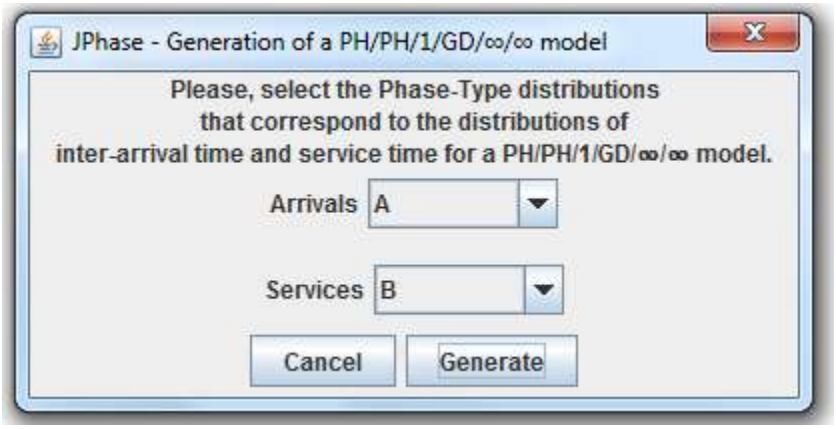


Figure 14. Generation Model Panel

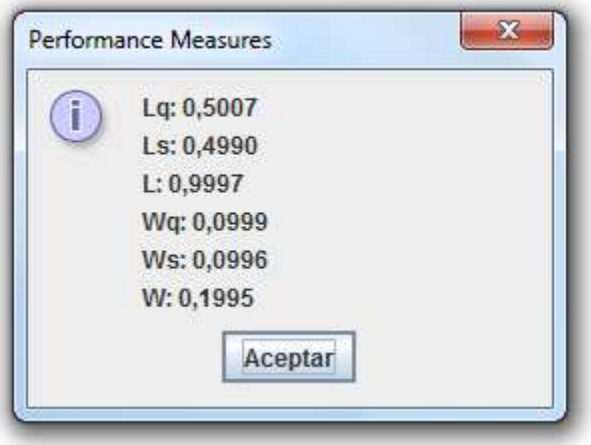


Figure 15. Performance Measures

4. Validation of the New Features

In this section we explain the methods that we used to test each newly developed feature.

4.1 Random Variate Generator, Moment Calculator and Goodness-of-Fit Tester

We verified the random variate generatorby performing goodness-of-fit tests for 10 generated datasets. 5 of these dataset contained 1000 variates with Gamma ($\alpha = 1, \lambda = 5$) distribution, the rest of datasets countained 1000 variates with Lognormal($\mu = 0.2, \sigma = 1$) distribution.The tests wereperformed usingthree software packages: Crystal Ball, Arena Input Analyser and also with our goodness-of-fit tester in order to validate it too. As the reader can see in the Table 1,the result of the test was that, for the100% of the datasets, the Kolmogorov-Smirnov and Chi-Square P-Values were greater than 5%. This means that all thedatasets distributeproperly. This 100% of well distributed datasets was useful to validate the variate generator and confirm that the test developed by our performer was also valid, because its results are the same that the other two softwares. With these tests we can conclude that the variate generator and the goodness-of-fit tests are valid.

Test #	Distribution	Crystal Ball		Arena Input Analyzer		jMarkov	
		Chi Square	K-S	Chi Square	K-S	Chi Square	K-S
1	Gamma (α= 1, λ= 5)	0,203	>0.15	0,234	0,141	0.225	0.638
2	Gamma (α= 1, λ= 5)	0,446	>0.15	0,122	0,416	0.670	0.759
3	Gamma (α= 1, λ= 5)	0,615	>0.15	0,444	0,311	0.155	0.709
4	Gamma (α= 1, λ= 5)	0,821	>0.15	0,655	0,862	0.430	0.945
5	Gamma (α= 1, λ= 5)	0,75	>0.15	0,527	0,507	0.570	0.916
11	Lognormal (μ = 0.2, σ = 1)	0,475	>0.15	0,303	0,523	0.691	0.924
12	Lognormal (μ = 0.2, σ = 1)	0,651	>0.15	0,965	0,886	1.000	0.993
13	Lognormal (μ = 0.2, σ = 1)	0,276	>0.15	0,306	0,246	0.603	0.765
14	Lognormal (μ = 0.2, σ = 1)	0,653	>0.15	0,181	0,147	0.420	0.644
15	Lognormal (μ = 0.2, σ = 1)	0,085	>0.15	0,394	0,111	0.866	0.637

Table 1. Goodness-of-Fit tests results

4.2 The PH/PH/1 Model

We used two different verification and validation methods. The first one consists of manually verifying that the algorithms to solve the model are properly implemented. The second is comparingthe results of one specific model with someone that we had its performance measures and expectthe same results.

In the first method we found the characteristic matricesof a PH/PH/1 model where the inter-arrival time distributes $A$and the service time distributes B. The variable A is the variable presented in Figure 14 and variable B is presented in figure 15.At the same time we followedthe procedure presented in Section 2 and found the characteristic matrices of a PH/PH/1 model.

```
Phase-Type Distribution
Number of Phases: 2
Vector:
        0,6533  0,3467
Matrix:
        -6,0978 0,0253
        3,8663  -6,1902
```

Figure 16. Phase-Type Variable A

```
Phase-Type Distribution
Number of Phases: 5
Vector:
        0,1044  0,1287  0,1898  0,2564  0,3207
Matrix:
        -1,6818 0,1726  0,0133  0,0013  0,0001
        0,5637  -0,6393 0,0073  0,0007  0,0001
        1,6413  0,1204  -1,7902 0,0010  0,0001
        1,8715  0,1200  0,0101  -2,0049 0,0001
        1,8888  0,1200  0,0101  0,0010  -2,0203
```

Figure 17. Phase-Type Variable B

Since the performance measures arethe same we can conclude that at least for this specific case the software gives validresults.

For the second method, as we show in Section 2.1, in some specific cases the hyper-exponential PH distribution is related to the exponential distribution.We created a hyper-exponential PHvariable$\Gamma$with 5 branches, each branch with a rate$\lambda = 5$ and a probability of 0.2 for each branch. The variableΓ should be equal to an exponential variable with rate $\lambda = 5$. Knowing this, we found the performance measures of a PH/PH/1 model where the inter-arrival time distributes as $\Gamma$and the service time distributes exponential PH with rate $\lambda = 10$, using jPhase.Additionally we found the performance measures for an M/M/1 model where the inter-arrival time distributes exponential with rate $\lambda = 5$ and the service time distributes exponential with rate $\lambda = 10$. In the case of M/M/1 queues the performance measures can be drawn theoretically, hence we were able to verify our results with the theoretical output. For more information about M/M/1 models see $[25]$. Since we obtain the expected result (both models

have the same performance measures), we verified that at least in this case the software provides correct results.

4.3 Goodness-of-Fit Tests for Phase-Type Distributions

To verify the goodness-of-fit tester for PH distributions we created a dataset of 1000 exponential variates with rate $\lambda = 5$.After verifying with Arena Input Analyzer that the new dataset followed the exponential distribution, we definedtwonew PH distributions. We provide a brief explanation of the exponential PH and hyper-exponential PH distributions in Section 2.The first variable $\Psi$was a hyper-exponential PHthat replicated an exponentialvariable with rate$\lambda = 5$ (see Section 2.1 for more information) and the second variable $\Theta$was anexponential PH with rate $\lambda = 5$. Given the fact that the dataset distributes exponentially we expect that it also distributes as $\Psi$and Θbecause as we explained in the Section 2, theoretically the three variables are the same. The result of the test was that the dataset distributes as the two new variables, we conclude that at least in this case the software provides correct results.

5. Results and Future Work

As a result of this project a new version of jMarkov was compiled with the described features.

As shown in Section 3 all the features were tested and proven to be functional. Withthe new version of jMarkov users can generate PH distributionshaving the theoretical distribution; perform goodness-of-fit tests not only for the common distributions but also for phase type distributions; and find the performance measures of a PH/PH/1 queue.

During this project we found three new opportunities to enhance the jMarkov project.

1. With the idea of taking advantage of the PH distribution's benefits mentioned in Section 2, it would be desirable to connect jMarkov with the R project $[24]$. We want to develop a connection between jPhase and the R project. With this connection the jMarkov user can perform calculus using the R project, and the R project user can define PH distributions in jPhase and use them in the R project.
2. As we mentioned in the introduction, jPhase implements two kinds of algorithms: Moment Matching and Maximum Likehood. Other fitting algorithms can be added to the jMarkov

package to help the user perform better fittings. [5]Presents a Discrete PH fitting algorithm using Machine Learning, this could be a new approximation that jPhase does not possess, not only because it is for discrete distributions but because it includes concepts of machine learning. We can also complement the actual EM algorithms with the proposed in [21] and compare its results with [3] (which is already implemented) given the fact that both fits data to hyper-erlang distributions.

3. Additionally, an Approximate Dynamic Programming module (jADP) can be built similar to the jMDP module. This new module would facilitate the performance of Approximate Dynamic Programming as an alternative to MDP for larger and more complex problems.

6. Bibliography

[1] Anderson, T. W. (1962). On the Distribution of the Two-Sample Cramer-von Mises Criterion. *The Annals of Mathematical Statistics*Volume 33, Number 3 (1962), 1148-1159.

[2] Anderson, T. W., & Darling, D. A. (1954). A Test of Goodness of Fit. *Journal of the American Statistical Association*, Volume 49, Number 268 (Dec., 1954),765-769.

[3] Asmussen, S., Nerman, O., & Olsson, M. (1996). Fitting Phase-Type Distributions via the EM Algorithm. *Scandinavian Journal of Statistics*, Volume 23, Number 4 (Dec., 1996).419-441.

[4] Bolch, G., Greiner, S., de Meer, H., & Trivedi, K. S. (2006). *Applications, Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science.* John Wiley & Sons.

[5] Callut, J., Dupont, P. (2006). Sequence discrimination using phase-type distributions. *Proceedings of the 17th European conference on Machine Learning*, 78-89.

[6] Cote, M. S. (June 2011). *jMarkov package: A Stochastic Modeling Tool.*Para optar al título de Ingeniería Industrial, Universidad de los Andes, Bogotá.

[7] Cote, M., Riano, G., Akhavan-Tabatabaei, R., Perez, J. F., Sarmiento, A., & Goez, J. (2012). jMarkov package: a stochastic modeling tool. *SIGMETRICS Perform. Eval. Rev.,*Volume 39, Number 4. (2012). 48-48.

[8] Frank, J., & Massey, J. (1951). The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, Volume 46, Issue 253. 68-78.

[9] Heimsund, B. (2005, Diciembre Last checked on 05-Dec-2005.). *Matrix Toolkits for Java (MTJ*. Retrieved from http://rs.cipr.uib.no/mtj/,

[10] Hicklin, J., Moler, C., Webb, P., Boisvert, R. F., Miller, B., Pozo, R., & Remington, K. (2005, July). JAMA: A java matrix package. Retrieved from http://math.nist.gov/javanumerics/jama/

[11] L'Ecuyer, P. (2002). SSJ: A Framework for Stochastic Simulation In Java. *Proceedings of the 2002 Winter Simulation Conference*, 234-242.

[12] Lewis, D., & Burke, C. J. (1949). The use and misuse of the chi-square test. *Psychological Bulletin*, Volume. 94, Number 1. (1983).433-489.

[13] Neuts, M. F. (1981). *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach.* Phenix: Courier Dover Publications, 1981.

[14] Perez, J. F. (May 2006). *An Object-Oriented tool for modeling Phase distribution and a computational benchmarking of fitting algorithms.*Para optar a titulo de Maestría en Ingeniería Industrial, Universidad de los Andes, Bogotá.

[15] Perez, J. F., & Riaño, G. (2006). *jPhase User's Guide.* Bogotá, Colombia: Universidad de los Andes.

[16] Ramaswami, V., & Latouche, G. (1987). *Introduction to Matrix Analytic Methods in Stochastic Modeling.* SIAM.

[17] Ramaswami, V., & Latouche, G. (1993). A Logarithmic Reduction Algorithm for Quasi-Birth-Death Processes. *Journal of Applied Probability*, Volume 30, Number 3 (Sep., 1993),650-674.

[18] Riaño, G., & Goez, J. (2006). *jMarkov: an object-oriented framework for modeling and analyzing Markov chains and QBDs.* Pisa, Italy: ACM.

[19] Riaño, G., & Goez, J.. *jMarkov User's Guide*. Retrieved from http://copa.uniandes.edu.co

[20] Riaño, G., & Sarmiento, A. (2005). *jMDP user's guide.*Bogotá, Colombia: Industrial Engineering, Universidad de los Andes.

[21] Riska, A. (2002). Efficient fitting of long-tailed data sets intoPH distributions.Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE. Volume 30 (Sep., 2002),2513-2517.

[22] Shapiro, S., & Wilk, M. (1965). An Analysis of Variance Test for Normality. *Journal of the American Statistical Association*, Volume. 67, Number 337. (1972).591-611.

[23] Telek, A. T. (2006). A Novel Approach for PhaseType Fitting with the EM Algorithm. *IEEE Transactions on Dependable and Secure Computing*, Volume. 3, Number 3. (2006).245-258.

[24] Venables, W. N., & Smith, D. M. (2012, 10 26). *An Introduction to R.* Retrieved from http://cran.r-project.org/doc/manuals/r-release/R-intro.pdf

[25] Winston, W. (2004). *Introduction to Probability Models.* Fourth edition, Thomson.

Universidad de los Andes

NIT: 860.007.386-1

SISTEMA DE BIBLIOTECAS
IDENTIFICACIÓN TRABAJO DE GRADO

FECHA DE ELABORACIÓN

DD	MM	AAAA
14	01	2013

1. IDENTIFICACIÓN AUTOR(ES) DEL TRABAJO DE GRADO

CÓDIGO	DOCUMENTO DE IDENTIDAD		APELLIDOS	NOMBRES	CORREO ELECTRÓNICO
	TIPO	NÚMERO			
200822561	CC	1015412365	Sarmiento Romero	Andrés	a.sarmiento64@uniandes
	CC				
	CC				
	CC				
	CC				
	CC				

PROGRAMA Pregrado
FACULTAD Facultad de Ingeniería
DEPARTAMENTO Departamento de Ingeniería Industrial

ENTREGÓ FORMATO:

☑ SB-10 "Entrega trabajo de grado y autorización de uso a favor de la Universidad de los Andes".

SB-10: Documento con el cual, el autor permite que su trabajo sea utilizado por la Universidad, para fines de consulta y de mención en sus catálogos bibliográficos, tanto físicos como en línea.

1.1 IDENTIFICACION DE TRABAJO DE GRADO PARA DOBLE TITULACIÓN

PROGRAMA No Aplica
FACULTAD No Aplica
DEPARTAMENTO No Aplica

TESIS PARA DOBLE TITULACIÓN:

☐ Si el trabajo de grado presentado aplica para obtener dos (2) titulaciones, por favor marque esta casilla y diligencie la información de esta sección.

2. INFORMACIÓN GENERAL DEL TRABAJO DE GRADO

TÍTULO DEL TRABAJO DE GRADO:
Extending the jPhase module of jMarkov project

DESCRIPCIÓN FÍSICA
Número de páginas: 26
Ilustraciones: 17

MATERIAL ACOMPAÑANTE (Cantidad):
Casetes Audio:
Discos compactos:
Casetes Video:
Diapositivas:
Disquetes:
Otros: ¿Cuáles?

FECHA DE ELABORACIÓN

DD	MM	AAAA
14	01	2013

*RESUMEN DEL TRABAJO DE GRADO:
This project is the result of some deficiencies or limitations found in the jMarkov project. Specifically, we found that jPhase graphical interface can be enhanced with new facilities to improve the user experience.

OBJETIVOS DEL TRABAJO DE GRADO:
The objective of this project is to develop new features that will deal with the limitations and improvements found in the jPhase module.

METODOLOGÍA DEL TRABAJO DE GRADO:

CONCLUSIONES DEL TRABAJO DE GRADO:

As shown in Section 3 all the features were tested and proven to be functional. With the new version of jMarkov users can generate PH distributions having the theoretical distribution; perform goodness-of-fit tests not only for the common distributions but also for phase type distributions; and find the performance measures of a PH/PH/1 queue.

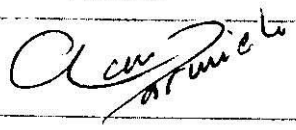
*PALABRAS CLAVES (TEMAS) DEL TRABAJO DE GRADO:
jMarkov, jPhase, stochastic modeling, java, Phase-Type distribution.

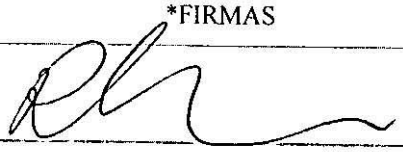
ACUERDOS DE CONFIDENCIALIDAD: ° NO TIENE ACUERDO(S) ·› TIENE ACUERDO(S)

Si selecciona tener acuerdo de confidencialidad, por favor diligencie el siguiente cuadro:

Persona natural o jurídica	Desde			Hasta			
	DD	MM	AAAA	DD	MM	AAAA	

3. FIRMAS

AUTORES (Nombre completo)	*FIRMAS
Andrés Sarmiento Rovero	

DIRECTORES / ASESORES (Nombre completo)	*FIRMAS
Raha Akhavan	

JURADO / LECTOR (Nombre completo)	*FIRMAS
Universidad de los Andes Facultad de Ingeniería DEPARTAMENTO DE INGENIERÍA INDUSTRIAL COORDINACIÓN DE PREGRADO	

Las firmas de Autor y Director/Asesor son obligatorias. Si tiene inconvenientes con el registro de la firma del Jurado/Lector, deberá tramitar ante la respectiva Facultad la autorización para registrar las firmas de pares o un sello que justifique la ausencia de la firma faltante.

SB-09

Verificar Información Imprimir

Universidad de los Andes
Colombia

ENTREGA EJEMPLAR TRABAJO DE GRADO Y AUTORIZACIÓN DE SU USO A FAVOR DE LA UNIVERSIDAD DE LOS ANDES

Yo **Andrés Sarmiento Romero** , mayor de edad, vecino de Bogotá D.C., identificado con la Cédula de Ciudananía N° **10.151.412.365** de **Bogotá D.C.** , actuando en nombre propio, en mi calidad de autor del trabajo de tesis, monografía o trabajo de grado denominado:

Extending the jPhase module of jMarkov project

, hago entrega del ejemplar respectivo y de sus anexos del ser el caso, en formato digital o electrónico (CD-ROM) y autorizo a LA UNIVERSIDAD DE LOS ANDES, para que en los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia, utilice y esu en todas sus formas, los derechos patrimoniales de reproducción, comunicación pública, transformación y distribución (alquiler, préstamo público e importación) que me corresponden como creador de la obra objeto del presente documento.

PARÁGRAFO: La presente autorización se hace extensiva no sólo a las facultades y derechos de uso sobre la obra en formato o soporte material, sino también para formato virtual, electrónico, digital, óptico, usos en red, internet, extranet, intranet, etc., y en general para cualquier formato conocido o por conocer.

EL AUTOR - ESTUDIANTES, manifiesta que la obra objeto de la presente autorización es original y la realizó sin violar o usurpar derechos de autor de terceros, por lo tanto la obra es de su exclusiva autoría y tiene la titularidad sobre la misma.
PARÁGRAFO: En caso de presentarse cualquier reclamación o por acción por parte de un tercero en cuanto a los derechos de autor sobre la obra en cuestión, EL ESTUDIANTE - AUTOR, asumirá toda la responsabilidad, y saldrá de defensa de los derechos aquí autorizados; para todos los efectos la Universidad actúa como un tercero de buena fe.

Para constancia se firma el presente documento en dos (2) ejemplares del mismo valor y tenor, en Bogotá D.C.,

a los **diecisiete** **17** días del mes de **Enero** de Dos Mil **Trece** **2013**

EL AUTOR - ESTUDIANTE.

(Firma) .....

Nombre **Andrés Sarmiento Romero**

Cédula **1.015.412.365** de **Bogotá D.C.**

SB-10