**SME0121 – Processos Estocásticos**

**Trabalho Prático**

**ALUNO : Ubiratan Soares (563242)**

**Data : 24/06/2011**

**Introdução**

Conforme proposta do docente, este trabalho busca aplicar na prática os conceitos vistos ao longo do curso de Processos Estocáticos. O desenvolvimento foi realizado com a linguagem Java, usando a ferramenta de desenvolvimento Eclipse. O suporte às classes que implementam as distribuições de probabilidade foi obtido através do framework *Apache Commons Math*, uma vez que tais distribuições não possuem implementação nativa nessa linguagem. Maiores detalhes podem ser obtidos em :

<http://commons.apache.org/math/>

Nesse trabalho, foi requisitado que fosse simulado um sistema composto por componente sujeitos à falhas que opera ao longo de um ano. Os componentes tem tempo de vida que segue uma distribuição exponencial.

Assim, inicialmente foi implementada a classe **ComponentSimulation.java**, cujo código segue a seguir. Essa classe executa uma simulação única nas condições propostas, ou seja, com componentes falhando em média a cada 1/ λ = 60 horas, em um período de simulação de 8760 horas(1 ano).

**// ComponentSimulation.java**

**package** org.usp.stochastic;

**import** org.apache.commons.math.distribution.\*;

**import** java.util.\*;

**public** **class** ComponentSimulation{

**private** ArrayList<Double> simulatedFailures = **new** ArrayList<Double>();

**private** **double** mean = 60;

**private** **double** hours = 8760.0;

**private** **double** accumulated = 0.0;

ComponentSimulation() **throws** Exception{

ExponentialDistributionImpl exp = **new** ExponentialDistributionImpl(mean);

**while**(accumulated < hours){

**try**{

**double** rexp = exp.sample();

simulatedFailures.add(rexp);

accumulated += rexp;

}

**catch**(Exception e){

System.*out*.println("ERROR : CANT GENERATE DISTRIBUTION!");

}

} //end while

}//end Constructor

**public** ArrayList<Double> getSimulation(){

**return** **this**.simulatedFailures;

}

}// end ComponentSimulation

De posse dessa classe foram implementadas outras duas classes, que tomam a classe acima como base : **ComponentSimulationTest.java** e **ComponentSimulationTestBattery.java**.

A primeira executa uma simulação única e faz todos os cálculos associados, ou seja, automaticamente gera os resultados para os exercícios 2 e 3 desse trabalho. A segunda executa uma bateria de um número arbitrário de testes, a ser definido pelo usuário.

A intenção dessa implementação adicional é verificar de forma estatística o comportamento médio do sistema em questão, ainda que a mesma não seja utilizada nos exercícios aqui resolvidos. Uma execução da classe **ComponentSimulationTestBattery** é colocada na sequência para efeitos de constatação de que, por exemplo, os componentes não falharão mais do que 200 vezes em um ano (estimado empiricamente nos meus testes), por exemplo. Essa classe pode ser estendida para obter resultados mais confiáveis para os problemas propostos nesse trabalho.

**// ComponentSimulationTestBattery.java**

**package** org.usp.stochastic;

**import** java.util.\*;

**public** **class** ComponentSimulationTestBattery {

**public** **static** **void** main(String[] args) **throws** Exception{

ArrayList<Double> meanResults = **new** ArrayList<Double>();

**int** runs = 100;

**for**(**int** i = 0; i < 200; i++) meanResults.add(0.0);

**for**(**int** i = 0; i < runs; i++){

ComponentSimulation c = **new** ComponentSimulation();

ArrayList<Double> results = c.getSimulation();

**for**(**int** j = 0; j < results.size(); j++){

meanResults.set(j, meanResults.get(j) + results.get(j));

}

}

**for**(**int** i = 0; i < 200; i++) meanResults.set(i, meanResults.get(i) / runs);

System.*out*.println("[BATTERY TEST RUN]\n");

**for**(**int** i = 1; i < meanResults.size() + 1; i++){

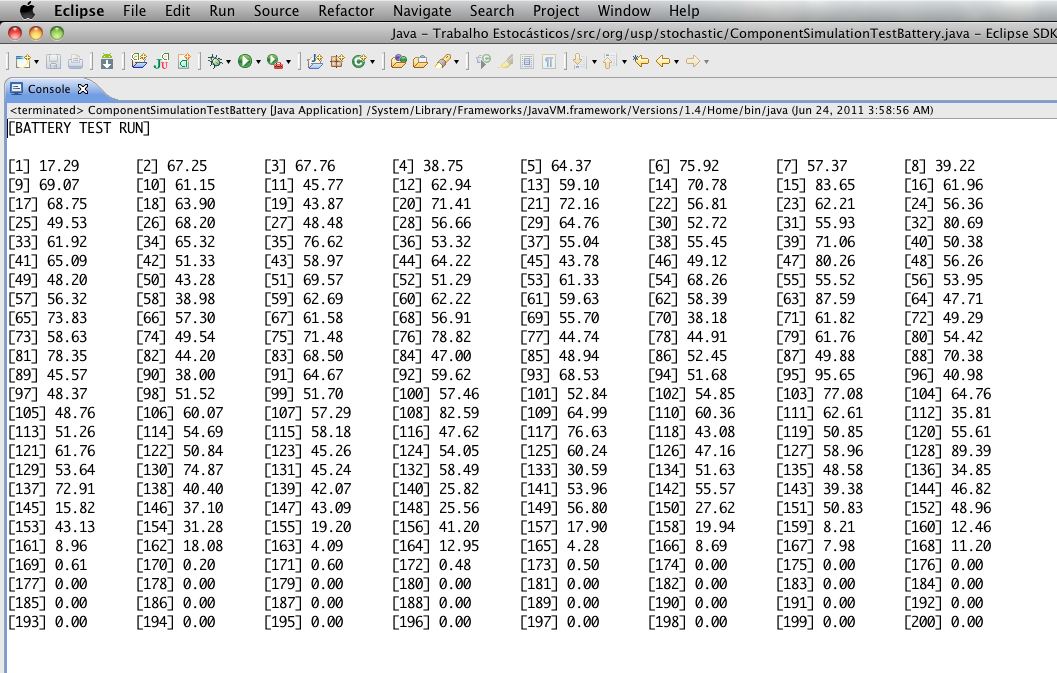
**if**(i % 8 == 0) System.*out*.printf("[%d] %.2f \n", i, meanResults.get(i - 1));

**else** System.*out*.printf("[%d] %.2f \t", i, meanResults.get(i - 1));

}

}//end Main

}//end CompomentSimulationTestBattery



A classe principal que realiza as tarefas do trabalho tem seu código listado a seguir :

**// ComponentSimulationTest.java**

**package** org.usp.stochastic;

**import** java.util.\*;

**public** **class** ComponentSimulationTest {

**public** **static** **void** main(String[] args) **throws** Exception{

ComponentSimulation c = **new** ComponentSimulation();

ArrayList<Double> simulation = c.getSimulation();

System.*out*.println("[START] \n\nUNIQUE SIMULATION FOR EACH COMPONENT FAILURE\n");

**for**(**int** i = 1; i < simulation.size() + 1; i++){

**if**(i % 8 == 0) System.*out*.printf("[%d] %.2f \n", i, simulation.get(i - 1));

**else** System.*out*.printf("[%d] %.2f \t", i, simulation.get(i - 1));

}

System.*out*.println("\n\n[END]");

**int** k = 10;

**double** k10time = 0.0;

**for**(**int** i = 0; i < k; i++) k10time += simulation.get(i);

System.*out*.println("\n -> TIME UNTIL 10th COMPONENT FAIL = " + k10time);

ArrayList<Double> costs = **new** ArrayList<Double>();

**double** r = 0.05;

**double** beta = 10;

**double** lambda = (**double**) 1/60;

**double** alpha = 1 / (1 + r);

**for**(**int** i = 0; i < simulation.size(); i++){

**double** cost = Math.*exp*(-alpha \* simulation.get(i)) \* beta;

costs.add(cost);

}

**double** totalCost = 0.0;

System.*out*.println("\n -> MAINTENANCE COST UNTIL 10th COMPONENT FAIL = " + costs.get(10));

**for**(**int** i = 0; i < costs.size(); i++) {totalCost += costs.get(i);}

System.*out*.println("\n -> TOTAL SYSTEM MAINTENANCE COST = " + totalCost);

**double** estimatedParameter = totalCost / simulation.size();

System.*out*.println("\n -> ESTIMATED PARAMETER FOR COST DISTRIBUTION = " + estimatedParameter);

System.*out*.println("\n -> ESTIMATED LAMBDA FOR COST DISTRIBUTION = " + 1/estimatedParameter);

**double** teorethicalExpectedValueCost = beta \* lambda / alpha;

System.*out*.println("\n -> TEORETHICAL EXPECTED VALUE FOR COST DISTRIBUTION = " + teorethicalExpectedValueCost);

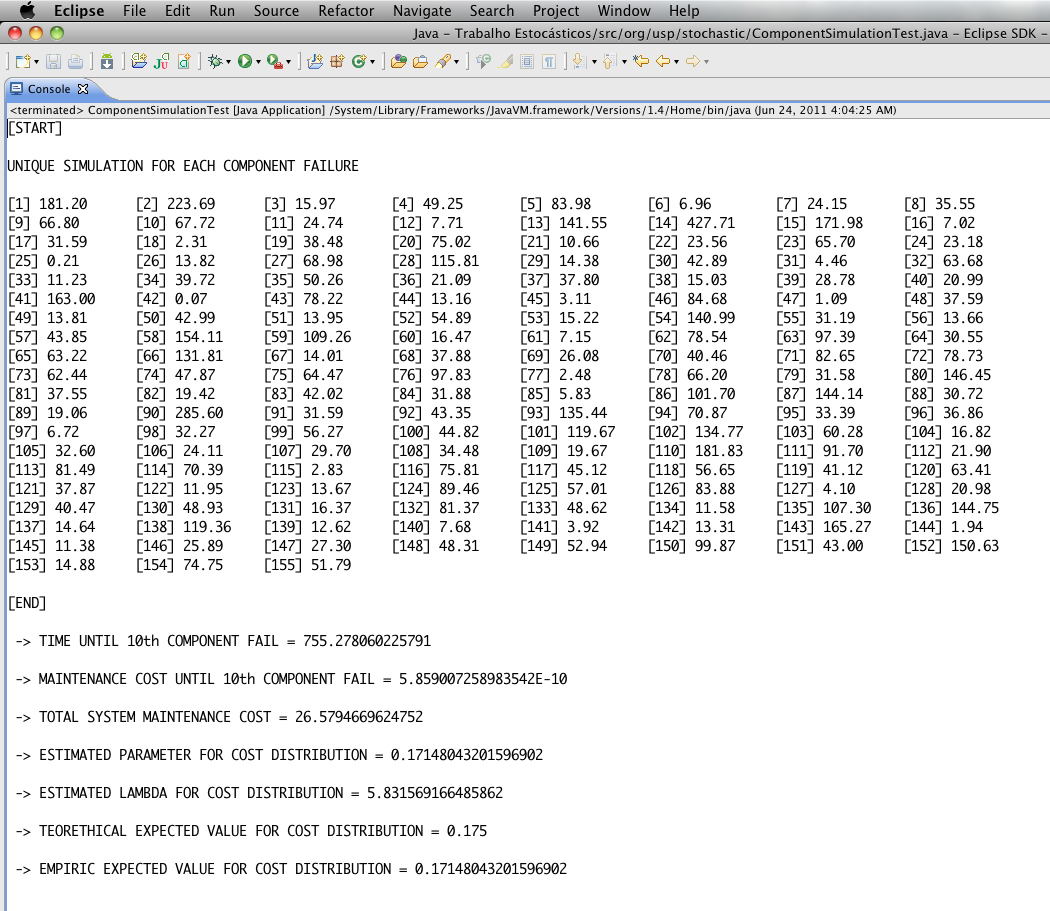
**double** praticalExpectedValueCost = totalCost / costs.size();

System.*out*.println("\n -> EMPIRIC EXPECTED VALUE FOR COST DISTRIBUTION = " + praticalExpectedValueCost);

}//end Main

}//end CompomentSimulationTest

Uma execução dessa classe é colocada abaixo e é utilizada para responder as questões do trabalho :



**Exercício 01**

Conforme pode ser conferido na simulação acima retratada, 155 componentes apresentaram falhas durante um ano de operação do sistema. Em particular, o tempo em horas até a falha do décimo componente é de **67.72 horas**.

Sabemos que esse componente tem distribuição **x ~ Exp(λ).** Dessa forma, para esse componente temos **1** / **λ = 67.72** ou seja **λ = 0.1476.**

Além disso, da distribuição exponencial sabemos que :

* **E(x) =** **1/λ = 67.72**
* **σ(x) =**  **67.72**

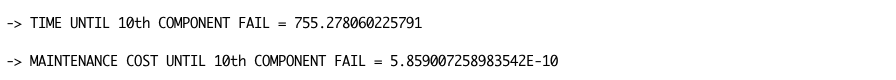
Finalmente, para estimar o tempo de vida com probabilidade de 95%, tomamos a inversa da função de distribuição acumulada da exponencial, fazendo :

**= - 20.29**

**Exercício 02**

Para essa questão, tem-se que o tempo de substituição do k-ésimo componente é dado por:

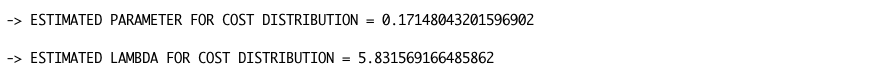
Sendo cada positivo, o custo do k-ésimo componente é uma função exponencial negativa, ou seja, apresenta um valor cada vez menor conforme k aumenta, e no infinito esse custo tende a zero. Para o décimo componente podemos esperar um custo bem baixo, e conforme a execução da simulação



Pelos resultados acima, vemos que decorreram mais de 755 horas até que o décimo componente fosse substítuido, ou seja, apresentasse falha. O custo dessa substituição é próximo de zero, conforme o esperado da natureza da fórmula do custo. A implementação também obteve o custo total de manutenção do sistema para essa simulação :

Estamos interessados em estimar a distribuição do custo total de manutenção do sistema, que como calculado acima é cado por

Assim, olhamos para o custo médio dos componentes para obter um estimador do parâmetro da distribuição exponencial que rege o comportamento de **C**, ou seja, fazemos = **0.1714**



Logo, o parâmetro estimado é .

Adicionalmente, sabemos que a distribuição exponencial apresenta o fenômeno de falta de memória. Dessa maneira, tantos os custos individuais como o custo total tem mesmo parâmetro para a distribuição exponencial, ou seja **ck ~exp(** e **C~exp(** de maneira que temos :

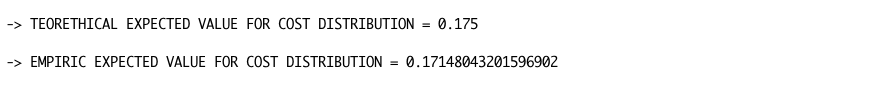
* **E(ck) = E(C) = 0.1714**
* **σ(ck)= σ(C) =**  **0.1714**
* **FDA :**

**Exercício 03**

Nesse exercício, comparamos os resultados práticos com as expectativas teóricas. Em particular, estamos interessados no valor esperado,de maneira que o resultado teórico diz que :

**,** com , **r** e **β** dados

De fato, a simulação mostra a proximidade dos resultados :

Por questões de limitação desse editor de texto (MS Word), a demonstração desses resultados se encontra em anexo, escrita à mão.