Desempenho e Dimensionamento de Redes

João Génio, 88771
Ruben Menino, 89185

Exemplos de Aplicação de Probabilidades, Variáveis Aleatórias e Cadeias de Markov

17 de abril 2021

Exercício 4

4a.

```
% normal state

state6 = 1 / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 + 8/600*5/200*2/50*1/5);
state5 = (8/600) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);
state4 = (8/600*5/200) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);

probNormal = state6 + state5 + state4

% interference state
state3 = (8/600*5/200*2/50) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);
state2 = (8/600*5/200*2/50*1/5) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);

probInterference = state3 + state2
```

Probabilidade do link estar no estado normal: 0.999984 (código matlab arredonda para 1)

Probabilidade do link estar no estado de interferência: 1.5784e-05

4b.

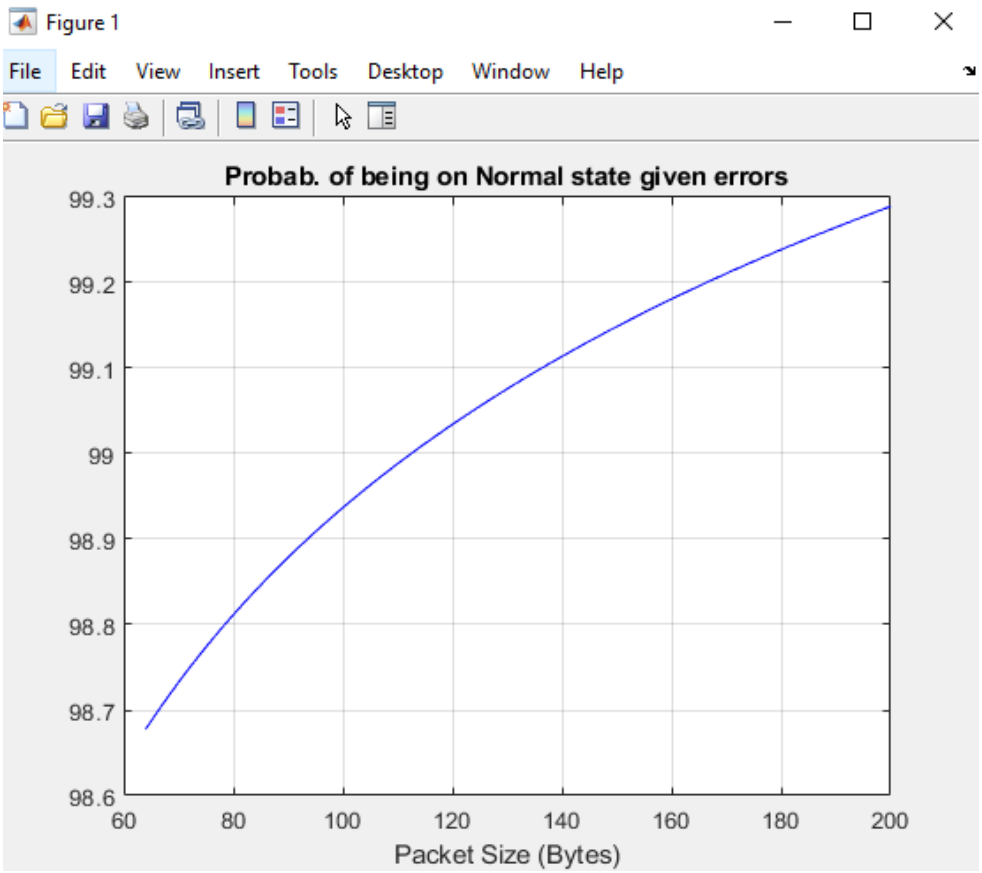
```
% normal state
state6 = 1 / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 + 8/600*5/200*2/50*1/5);
state5 = (8/600) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);
state4 = (8/600*5/200) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);

% interference state
state3 = (8/600*5/200*2/50) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);
state2 = (8/600*5/200*2/50*1/5) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);

ber6 = state6 * 10^-6;
ber5 = state5 * 10^-5;
ber4 = state4 * 10^-4;

% normal average
normalAverage = (ber6 + ber5 + ber4)/ ( state6 + state5 + state4)

ber3 = state3 * 10^-3;
ber2 = state2 * 10^-2;

% interference average
interferenceAverage = (ber3 + ber2) / (state3 + state2)
```

Média do bit error rate quando está no estado normal: 1.1509e-06

Média do bit error rate quando está no estado de interferência: 2.5e-03

4c.

```
packetsize = linspace(64*8,200*8);

% normal state
state6 = 1 / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 + 8/600*5/200*2/50*1/5);
state5 = (8/600) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);
state4 = (8/600*5/200) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);

% interference state
state3 = (8/600*5/200*2/50) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);
state2 = (8/600*5/200*2/50*1/5) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);

% 1 - ( nchoosek(n,i) * p^i * (1-p)^(n-i) )
% i = 0 (zero erros)
probEestado6 = 1 - ( 1 * (10^-6)^0 * (1-(10^-6)).^packetsize )
probEestado5 = 1 - ( 1 * (10^-5)^0 * (1-(10^-5)).^packetsize );
probEestado4 = 1 - ( 1 * (10^-4)^0 * (1-(10^-4)).^packetsize );
probEestado3 = 1 - ( 1 * (10^-3)^0 * (1-(10^-3)).^packetsize );
probEestado2 = 1 - ( 1 * (10^-2)^0 * (1-(10^-2)).^packetsize );

prob6dadoE = ( probEestado6 .* state6 ) ./ ( ( probEestado6 .* state6 ) + (
probEestado5 .* state5 ) + ( probEestado4 .* state4 ) + ( probEestado3 .* state3 )
+ ( probEestado2 .* state2 ) )
prob5dadoE = ( probEestado5 .* state5 ) ./ ( ( probEestado6 .* state6 ) + (
probEestado5 .* state5 ) + ( probEestado4 .* state4 ) + ( probEestado3 .* state3 )
+ ( probEestado2 .* state2 ) )
prob4dadoE = ( probEestado4 .* state4 ) ./ ( ( probEestado6 .* state6 ) + (
probEestado5 .* state5 ) + ( probEestado4 .* state4 ) + ( probEestado3 .* state3 )
+ ( probEestado2 .* state2 ) )

probNdadoE = ( prob6dadoE + prob5dadoE + prob4dadoE)

figure(1)
plot(packetsize/8,probNdadoE * 100,'b-')
grid on
title('Probab. of being on Normal state given errors')
xlabel('Packet Size (Bytes)')
```



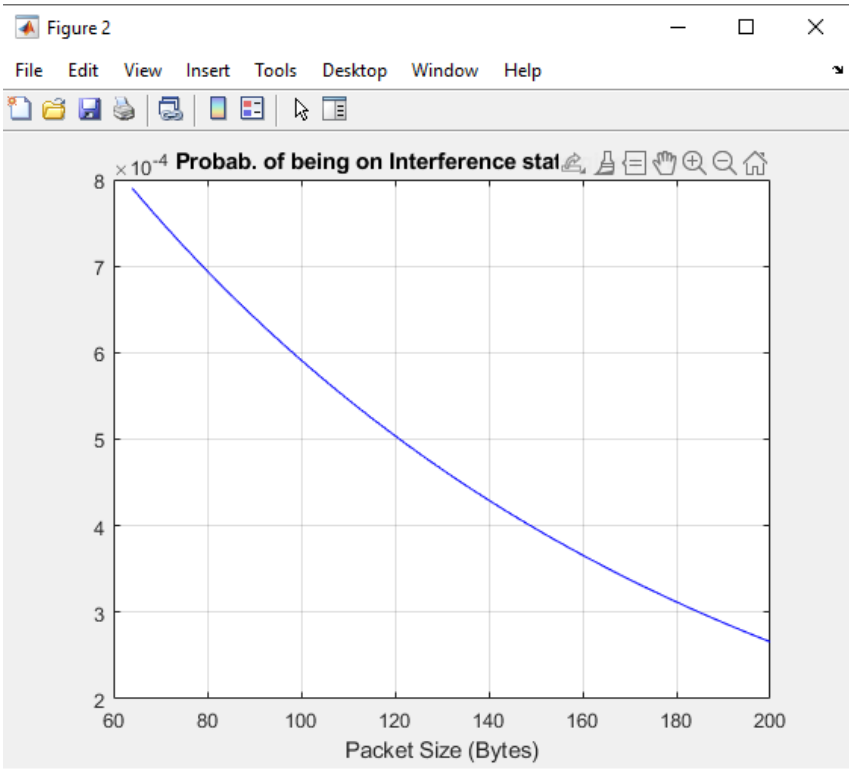
A partir das cadeias de Markov, conseguimos arranjar um sistema de nascimento e morte em tempo contínuo, em que existem taxas de chegada e também as taxas de partida. Inicialmente para conseguir calcular a probabilidade de um link estar num estado normal sabendo que é recebido com erros, foi necessário calcular a probabilidade de estar em cada um dos estados a partir das probabilidades limite de processos de nascimento e morte. Posteriormente, sabendo que o bit error rate para o link estar no estado de interferência teria de ser pelo menos 10e-03 e para se encontrar no estado normal teria de ser 10e-06, 10e-05 ou 10e-04, calculamos a partir da função $f(i) = \frac{n!}{i!(n-i)!} * p^{i} * (1-p)^{n-i}$ a probabilidade de ter ocorrido a receção dos data frames sem nenhum número de erros. Como o principal objetivo era calcular a probabilidade da receção de erros, pela função de variáveis aleatórias binomial, subtraímos ao resultado calculado para cada estado e bit error rate, o valor de 1. Após serem calculados estes valores, foi aplicada a Regra de Bayes para cada um dos estados normal.

Analisando o gráfico e sabendo que a probabilidade de um link estar num estado normal é de 0.999984, podemos observar que quando existe ocorrência de erros, essa mesma probabilidade

continua a ser bastante elevada. Podemos observar também que consoante o aumento do packet size, aumenta também a probabilidade de estar nesse estado, de uma forma logarítmica.

4d.

```
packetsize = linspace(64*8,200*8);

% normal state
state6 = 1 / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 + 8/600*5/200*2/50*1/5);
state5 = (8/600) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);
state4 = (8/600*5/200) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);

% interference state
state3 = (8/600*5/200*2/50) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);
state2 = (8/600*5/200*2/50*1/5) / (1 + 8/600 + 8/600*5/200 + 8/600*5/200*2/50 +
8/600*5/200*2/50*1/5);

% 1 - ( nchoosek(n,i) * p^i * (1-p)^(n-i) )
% i = 0 (zero erros)
probEestado6 = ( 1 * (10^-6)^0 * (1-(10^-6)).^packetsize )
probEestado5 = ( 1 * (10^-5)^0 * (1-(10^-5)).^packetsize );
probEestado4 = ( 1 * (10^-4)^0 * (1-(10^-4)).^packetsize );
probEestado3 = ( 1 * (10^-3)^0 * (1-(10^-3)).^packetsize );
probEestado2 = ( 1 * (10^-2)^0 * (1-(10^-2)).^packetsize );

prob3dadoE = ( probEestado3 .* state3 ) ./ ( ( probEestado6 .* state6 ) + (
probEestado5 .* state5 ) + ( probEestado4 .* state4 ) + ( probEestado3 .* state3 )
+ ( probEestado2 .* state2 ) )
prob2dadoE = ( probEestado2 .* state2 ) ./ ( ( probEestado6 .* state6 ) + (
probEestado5 .* state5 ) + ( probEestado4 .* state4 ) + ( probEestado3 .* state3 )
+ ( probEestado2 .* state2 ) )


probNdadoE = ( prob3dadoE + prob2dadoE)

figure(2)
plot(packetsize/8,probNdadoE * 100,'b-')
grid on
title('Probab. of being on Interference state given errors')
xlabel('Packet Size (Bytes)')
```



Na alínea d) foi-nos proposto calcular a probabilidade de um link estar no estado de interferência sabendo que não existiam quaisquer erros na data frame de um estado para outro. Neste exercício foi utilizada a mesma estratégia que no exercício anterior, porém não foi necessário subtrair o valor de 1, visto que estamos a calcular para um número nulo de erros.

Após esses cálculos foi utilizada a regra de Bayes para os estados de interferência.

Analisando o gráfico podemos observar que à medida que o número de pacotes aumenta, a probabilidade de estar num estado de interferência, com uma recepção sem erros é menor.

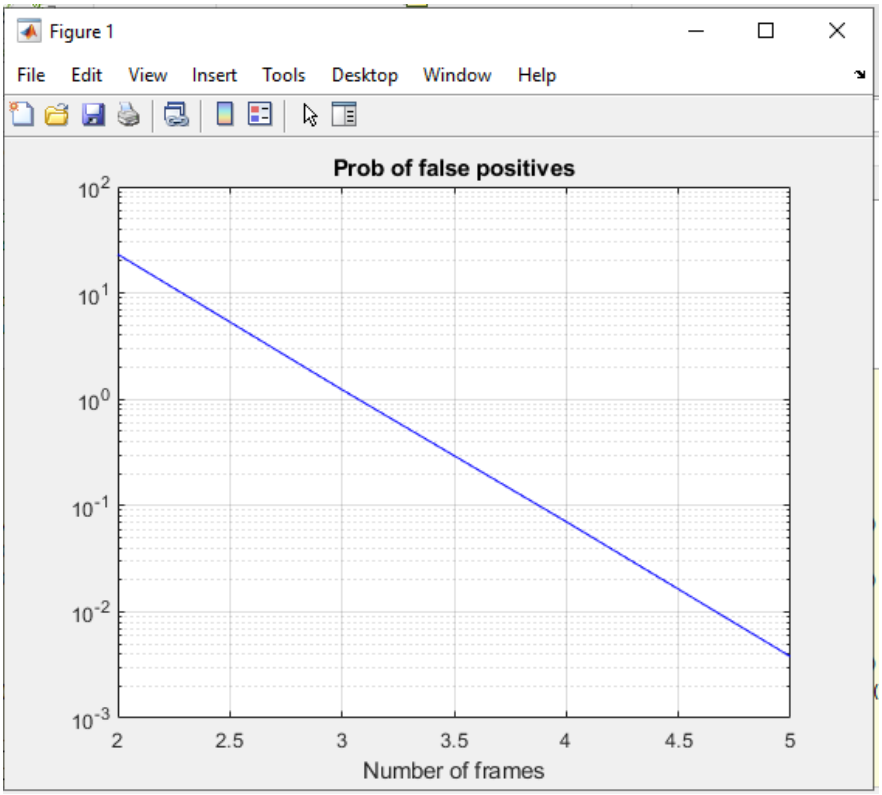
A razão de essa probabilidade ser tão baixa deve-se ao facto de que se o bit error rate for pelo menos 10e-3, ele está já no estado de interferência., logo à medida que esse ber aumenta, e o packet size também, vai haver mais probabilidade de ocorrer um erro na transmissão.

Exercício 5

5a.

```
frameSize = (64 * 8);

% normal state
state6 = 1 / (1 + 8/600 + (8/600) * (5/200) + (8/600) * (5/200) * (2/50) + (8/600)
* (5/200) * (2/50) * (1/5));
state5 = (8/600) / (1 + (8/600) + (8/600)*(5/200) + (8/600)*(5/200)*(2/50) +
(8/600)*(5/200)*(2/50)*(1/5));
state4 = ((8/600)*(5/200)) / (1 + (8/600) + (8/600)*(5/200) +
(8/600)*(5/200)*(2/50) + (8/600)*(5/200)*(2/50)*(1/5));

% intereference state
state3 = ((8/600)*(5/200)*(2/50)) / (1 + (8/600) + (8/600)*(5/200) +
(8/600)*(5/200)*(2/50) + (8/600)*(5/200)*(2/50)*(1/5));
state2 = ((8/600)*(5/200)*(2/50)*(1/5)) / (1 + (8/600) + (8/600)*(5/200) +
(8/600)*(5/200)*(2/50) + (8/600)*(5/200)*(2/50)*(1/5));

probEestado6 = 1 - ( 1 * (10^-6)^0 * (1-(10^-6)) ^ frameSize );
probEestado5 = 1 - ( 1 * (10^-5)^0 * (1-(10^-5)) ^ frameSize );
probEestado4 = 1 - ( 1 * (10^-4)^0 * (1-(10^-4)) ^ frameSize );
probEestado3 = 1 - ( 1 * (10^-3)^0 * (1-(10^-3)) ^ frameSize );
probEestado2 = 1 - ( 1 * (10^-2)^0 * (1-(10^-2)) ^ frameSize );

probFalsePositives = [0,0,0,0];
i = 1;
for n = 2:1:5
    n6 = probEestado6 ^ n;
    n5 = probEestado5 ^ n;
    n4 = probEestado4 ^ n;
    n3 = probEestado3 ^ n;
    n2 = probEestado2 ^ n;

    prob6dadoE = ( n6 .* state6 ) ./ ( ( n6 .* state6 ) + ( n5 .* state5 ) + ( n4
.* state4 ) + ( n3 .* state3 ) + ( n2 .* state2 ) )
    prob5dadoE = ( n5 .* state5 ) ./ ( ( n6 .* state6 ) + ( n5 .* state5 ) + ( n4
.* state4 ) + ( n3 .* state3 ) + ( n2 .* state2 ) )
    prob4dadoE = ( n4 .* state4 ) ./ ( ( n6 .* state6 ) + ( n5 .* state5 ) + ( n4
.* state4 ) + ( n3 .* state3 ) + ( n2 .* state2 ) )

    probNdadoE = ( prob6dadoE + prob5dadoE + prob4dadoE)
```

```
    probFalsePositives(i) = probNdadoE
    i = i + 1;
end


figure(1)

semilogy(linspace(2,5,4), probFalsePositives * 100, 'b-')
grid on
title('Prob of false positives')
xlabel('Packet Size (Bytes)')
```



Novamente, calculámos a probabilidade do link se encontrar em cada estado, de acordo com o seu “bit error rate”, e a probabilidade do link ter pelo menos um erro, dado que se encontra num determinado estado.

Depois, iterámos um valor n (de 2 a 5) e aplicámos a função probabilidade de variáveis aleatórias binomiais em cada estado, para obtermos a probabilidade de todos os “n” pacotes terem pelo

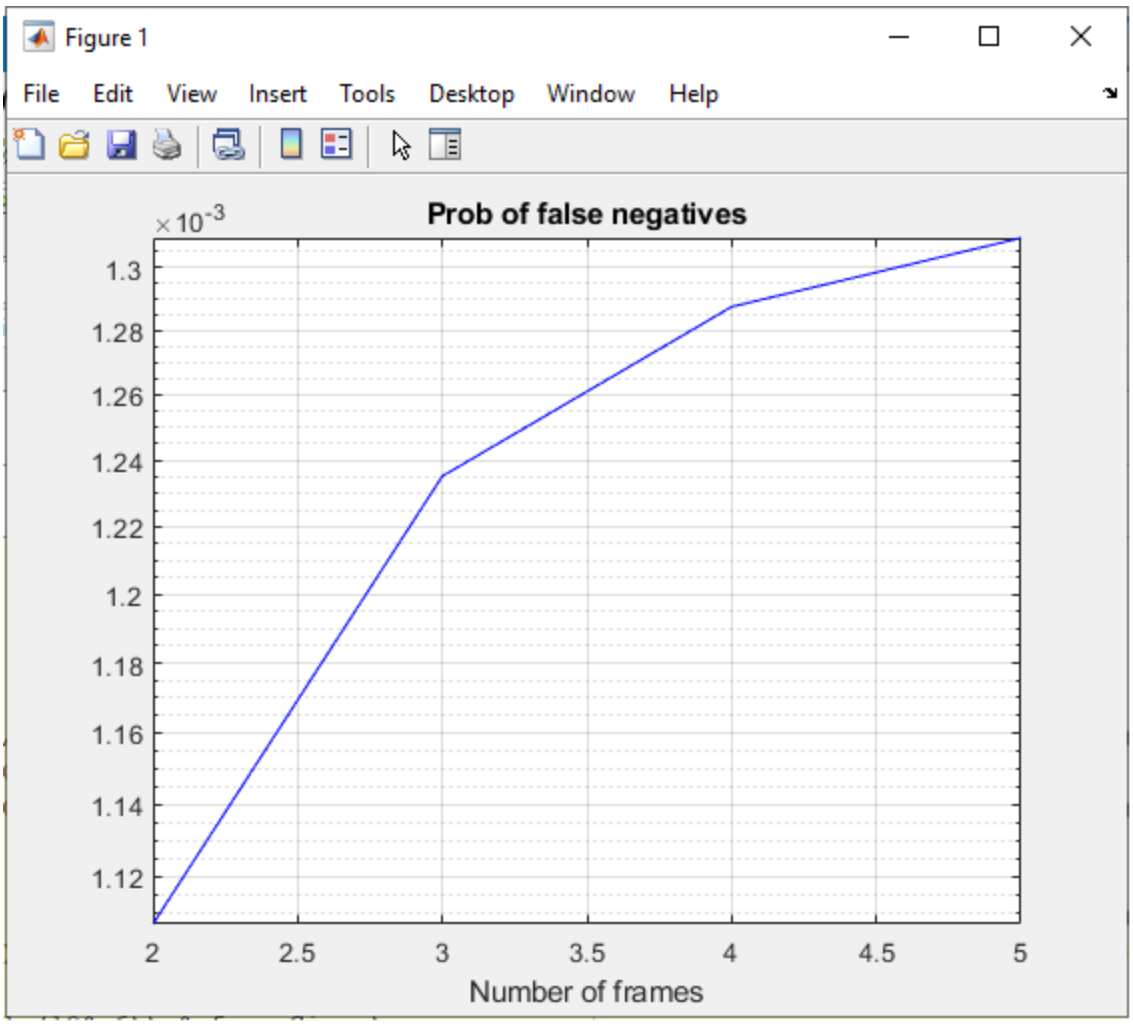
menos um erro cada. Por fim, aplicámos a Regra de Bayes para obter a probabilidade de falsos positivos em cada “n”.

É fácil de assumir que a probabilidade de ocorrer um falso positivo diminua com o aumento do número de pacotes pois, para isso acontecer, todos os pacotes teriam de ter pelo menos um erro, o que é, por si só, improvável num link no estado normal (bit error rate baixo).

5b.

```
frameSize = (64 * 8);

% normal state
state6 = 1 / (1 + 8/600 + (8/600) * (5/200) + (8/600) * (5/200) * (2/50) + (8/600)
* (5/200) * (2/50) * (1/5));
state5 = (8/600) / (1 + (8/600) + (8/600)*(5/200) + (8/600)*(5/200)*(2/50) +
(8/600)*(5/200)*(2/50)*(1/5));
state4 = ((8/600)*(5/200)) / (1 + (8/600) + (8/600)*(5/200) +
(8/600)*(5/200)*(2/50) + (8/600)*(5/200)*(2/50)*(1/5));

% intereference state
state3 = ((8/600)*(5/200)*(2/50)) / (1 + (8/600) + (8/600)*(5/200) +
(8/600)*(5/200)*(2/50) + (8/600)*(5/200)*(2/50)*(1/5));
state2 = ((8/600)*(5/200)*(2/50)*(1/5)) / (1 + (8/600) + (8/600)*(5/200) +
(8/600)*(5/200)*(2/50) + (8/600)*(5/200)*(2/50)*(1/5));

probEestado6 = 1 - ( 1 * (10^-6)^0 * (1-(10^-6)) ^ frameSize );
probEestado5 = 1 - ( 1 * (10^-5)^0 * (1-(10^-5)) ^ frameSize );
probEestado4 = 1 - ( 1 * (10^-4)^0 * (1-(10^-4)) ^ frameSize );
probEestado3 = 1 - ( 1 * (10^-3)^0 * (1-(10^-3)) ^ frameSize );
probEestado2 = 1 - ( 1 * (10^-2)^0 * (1-(10^-2)) ^ frameSize );

probFalsePositives = [0,0,0,0];
i = 1;
for n = 2:1:5
    n6 = 1 - probEestado6 ^ n;
    n5 = 1 - probEestado5 ^ n;
    n4 = 1 - probEestado4 ^ n;
    n3 = 1 - probEestado3 ^ n;
    n2 = 1 - probEestado2 ^ n;

    prob3dadoE = ( n3 .* state3 ) ./ ( ( n6 .* state6 ) + ( n5 .* state5 ) + ( n4
.* state4 ) + ( n3 .* state3 ) + ( n2 .* state2 ) )
    prob2dadoE = ( n2 .* state2 ) ./ ( ( n6 .* state6 ) + ( n5 .* state5 ) + ( n4
.* state4 ) + ( n3 .* state3 ) + ( n2 .* state2 ) )
```

```
    probIdadoE = ( prob3dadoE + prob2dadoE)


    probFalseNegatives(i) = probIdadoE
    i = i + 1;
end
semilogy(linspace(2,5,4), probFalseNegatives * 100, 'b-')
grid on
title('Prob of false negatives')
xlabel('Packet Size (Bytes)')
```



Mais uma vez, calculámos a probabilidade do link se encontrar em cada estado, de acordo com o seu “bit error rate”, e a probabilidade do link ter pelo menos um erro, dado que se encontra num determinado estado.

A seguir, iterámos um valor n (de 2 a 5) e aplicámos a função probabilidade de variáveis aleatórias binomiais em cada estado, para obtermos a probabilidade de pelo menos um dos “n” pacotes terem zero erros. Por fim, aplicámos a Regra de Bayes para obter a probabilidade de falsos negativos em cada “n”.

Assumimos facilmente que a probabilidade de ocorrer um falso positivo aumente com o aumento do número de pacotes pois, para isso acontecer, pelo menos um dos pacotes teria de ter zero erros.

Uma vez que o estado de interferência tem chances de erro mais elevadas, a probabilidade de um pacote não sofrer erros é reduzida, assim como a probabilidade de falso negativo.

5c.

Observando o gráfico do 5.a, podemos observar que à medida que o número de frames aumenta, a probabilidade de erros vai diminuir. Logo existe uma descida na probabilidade de ocorrer um falso positivo.

Relativamente ao gráfico do 5.b, vai existir uma probabilidade baixa de ocorrer um falso negativo, aumentando significativamente com o aumento do número de frames.

A escolha é fácil porque para n=5 conseguimos diminuir significativamente a probabilidade de falsos positivos sem “sacrificar”, isto é, aumentar demasiado a probabilidade de falsos negativos. Uma vez que são igualmente importantes, diminuir significativamente um deles, sem aumentar, de forma igual, o outro, encontramo-nos no melhor cenário.