TÉCNICO LISBOA

INSTITUTO SUPERIOR TÉCNICO

Modelação e Simulação

2021 / 2022 - 2º Período

TRABALHO 2

Simulação de Monte Carlo do jogo do Monopólio

Grupo 18
96201 – Fábio Miguel Magalhães Dias - fabio.dias@tecnico.ulisboa.pt
96219 – Gonçalo Pinto Midões - goncalomidoes@tecnico.ulisboa.pt
96305 – Ravi Alexandre Da Silva Regalo - ravi.regalo@tecnico.ulisboa.pt
96334 – Tiago Miguel Branco Ferreira - tiagombferreira@tecnico.ulisboa.pt

O grupo de alunos acima identificado garante que o texto deste relatório e todo o software e resultados entregues foram inteiramente realizados pelos elementos do grupo, com uma participação significativa de todos eles, e que nenhuma parte do trabalho ou do software e resultados apresentados foi obtida a partir de outras pessoas ou fontes.

20 de janeiro de 2022

Introdução

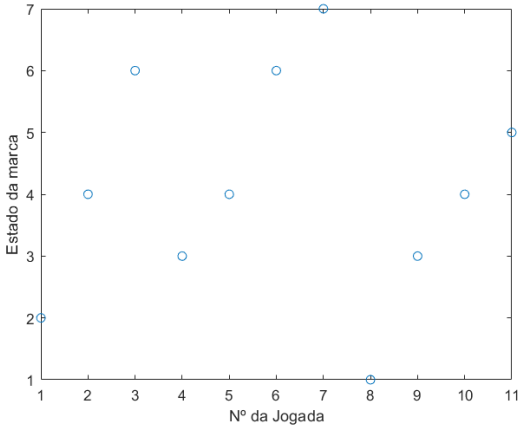
O principal objetivo deste trabalho é simular o jogo de Monopólio simplificado de forma a obter a distribuição de probabilidade de equilíbrio de cair nas várias "casas" usando o Método de Monte Carlo. Estas simulações foram implementadas com recurso ao MatLab®, e a sua análise detalhada segue as perguntas do guia de laboratório.

P1. Simulação para estudar a distribuição de probabilidades

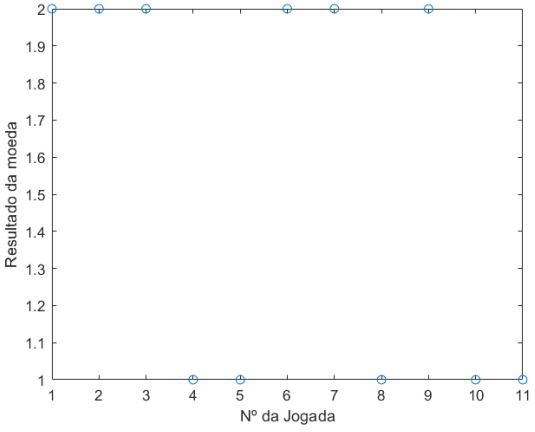
O programa desenvolvido para esta simulação encontra-se nos Anexos e as variáveis usadas no programa estão de acordo com o enunciado. Considera-se 2000 jogadas e 3 runs onde se descarta as primeiras 100 jogadas. Considera-se ainda que segundo a implementação escolhida e como é visível na figura 1b um resultado de moeda igual a 1 corresponde a sair "cara" e um resultado igual a 2 é "coroa". Para todo o trabalho o gerador de dados aleatórios usado é o predefinido (também chamado de *twister*).

a. Estados e resultado do lançamento da moeda

Para facilitar a análise das figuras nesta alínea corre-se o programa com 20 jogadas. Podemos observar, por inspeção, que quando a marca corresponde a cara os estados transitam para o estado seguinte (com excepção do estado 6). Quando a marca corresponde a coroa os estados transitam de dois em dois (com excepção do estado 5).



(a) Estados percorridos numa *run*.



(b) Resultado de lançamento de moedas sucessivas.

Fig. 1 – Gráficos significativos para a pergunta 1a.

b. Frequência relativa

A frequência relativa de cada casa obteve-se contando o número de ocorrências de cada casa e dividindo pelo o número total de jogadas (retirando o número de jogadas descartadas). Na figura 2 temos a frequência relativa de cada casa, como se pode observar a casa mais visitada é a 3 (prisão).

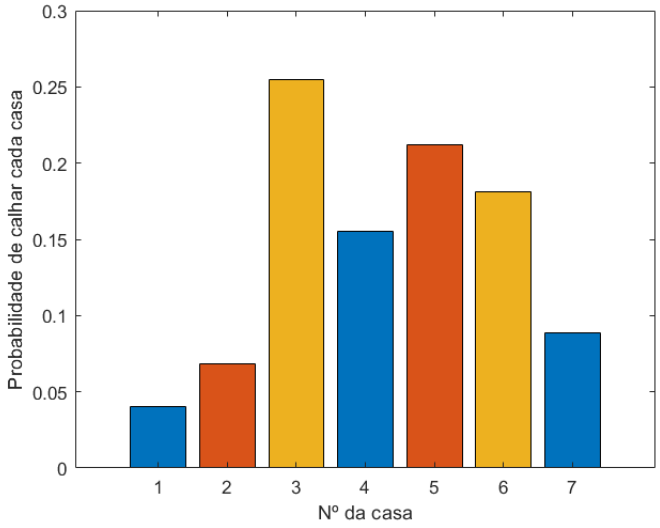


Fig. 2 – Frequência relativa das diferentes casas para uma *run*.

c. Renda média

Para obtermos o lucro médio em cada casa do jogo multiplica-se o vetor que contêm a frequência relativa pelo vetor que contém o custo de cada casa. Pela figura 3 percebemos que a casa mais lucrativa é a sexta.

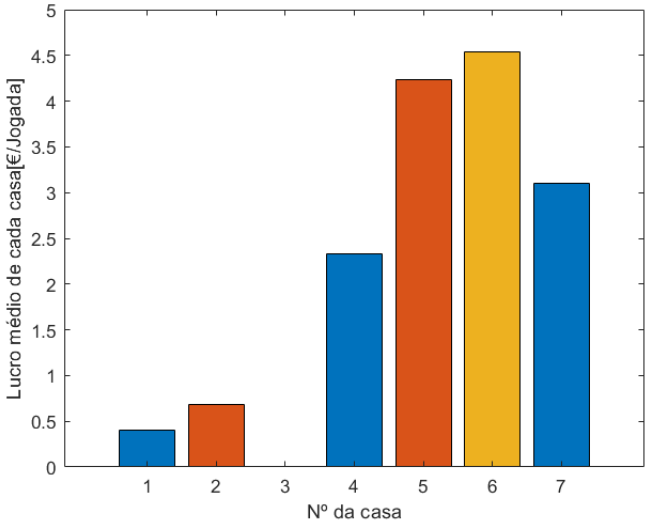


Fig. 3 – Lucro médio de cada casa.

P2. Validação do programa desenvolvido

Nesta secção o programa desenvolvido foi testado com o objetivo de confirmar todas as suas decisões e estudar a validade dos seus outputs.

i. Análise da sequência de estados e da sequência de eventos

Como foi feita referência na secção P1, por inspecção dos vetores *y* e *coinflips* (figura 1) podemos observar que o programa parece estar a funcionar como o desejado. No entanto, para confirmar para um número de jogadas elevado, escrevemos um loop (código P2, linha 22) onde se verifica se os saltos entre jogadas consecutivas têm valores plausíveis (1,2,3,5 ou 6). Para além disso, nas jogadas onde os saltos são de 3, 5 ou 6 verifica-se também o valor de *coinflip* que deu origem a essa transição.

ii. Distribuição das probabilidades em função do número de *runs*

Para este tópico decidiu-se calcular a probabilidade acumulada para vários valores de *runs*. Para tal executa-se o código realizado para a secção P1, com $Njogadas = 100$, $NMC = 100$ e $Ndiscard = 10$. O valor de *Ndiscard* utilizado foi obtido através da análise executada em P2.iii. Depois dessa execução, procede-se ao cálculo de probabilidades.

Dessa forma, para uma *run* calcula-se a quantidade de ocorrência de cada casa (valor diretamente lido da primeira linha da matriz *z*) e divide-se pelo número total de jogadas ($Njogadas - Ndiscard$). No estudo de 2 *runs*, o mesmo processo é feito, mas são somadas as quantidades de ocorrência até então (ou seja, as somam-se as primeiras 2 linhas da matriz *z*) e a divisão é feita pelo número de jogadas total (neste caso, $2 * (Njogadas - Ndiscard)$). Este processo é feito até o número de *runs* que foi efetivamente executado obtendo o gráfico representado na figura 4.

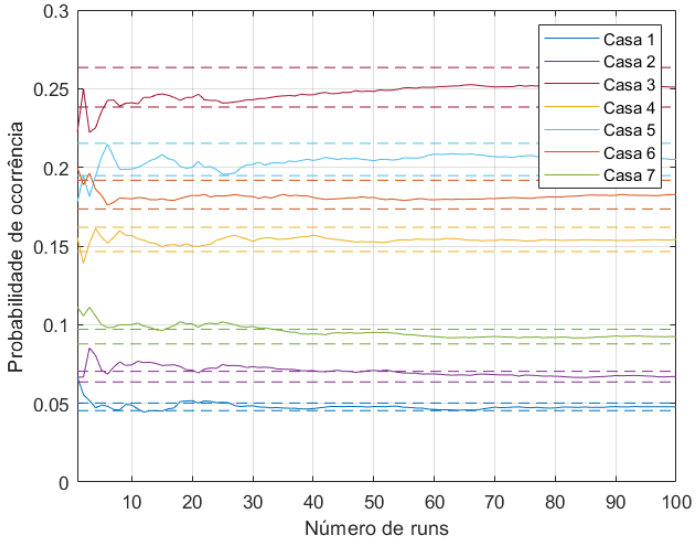
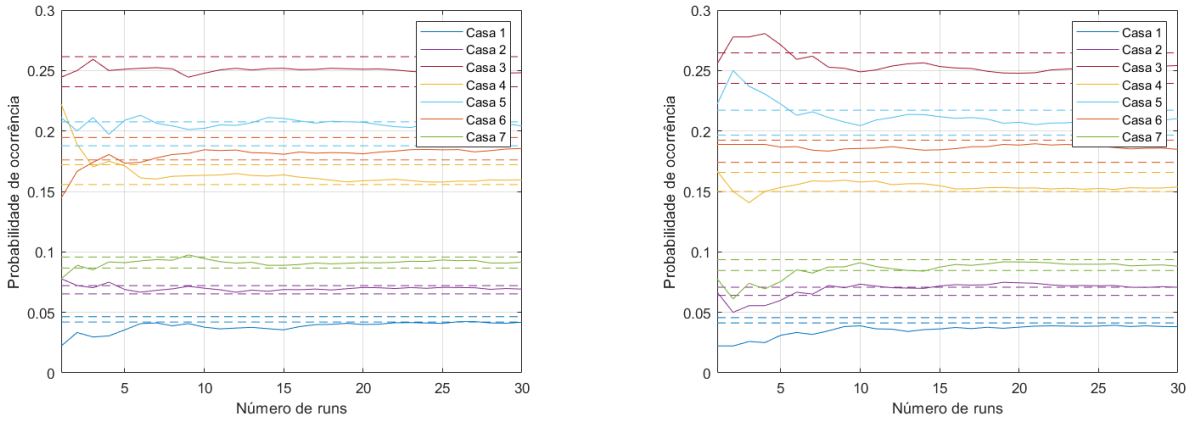


Fig. 4 – Probabilidade de ocorrência das diferentes casas em função do número de *runs*.

De facto, a convergência dá-se com a raiz quadrada do número de *runs*, como é visível na figura 4, uma vez que a probabilidade de ocorrência das diferentes casas estabiliza próximo das 10 *runs* ($\sqrt{\text{NMC}} = \sqrt{100} = 10$), ou seja, que a maioria das probabilidades se encontram a menos de 5% do seu valor final (linhas a tracejado).

Foi ainda feito o estudo das diferentes convergências das probabilidades consoante o gerador de números aleatórios usado. Existem 9 geradores diferentes na versão R2020b do MatLab®. Achou-se relevante fazer a comparação entre a convergência das probabilidades usando o gerador *twister* (figura 4), o gerador *combRecursive* (figura 5a) e o gerador *threefry* (figura 5b). Os limites do gráfico foram adaptados de modo a obter maior detalhe na análise.



(a) Gerador *combRecursive*.

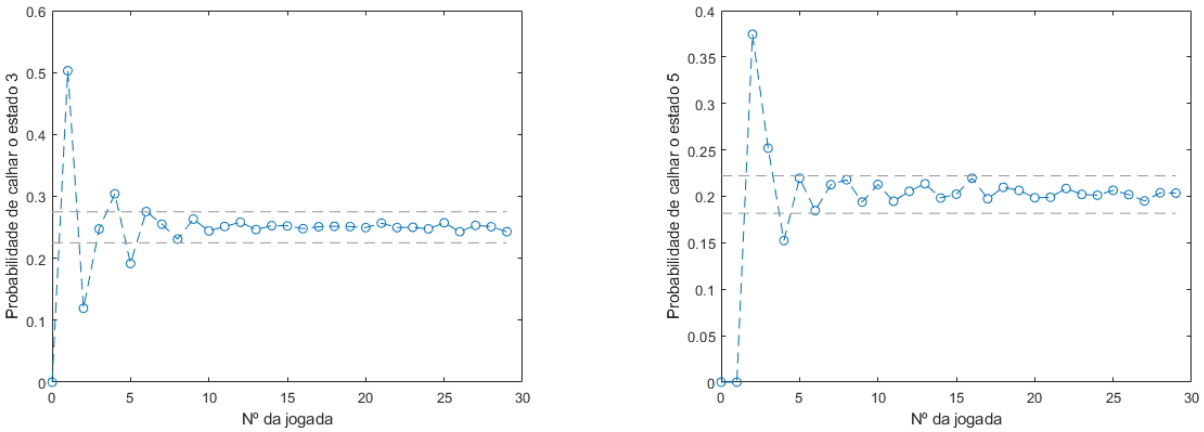
(b) Gerador *threefry*.

Fig. 5 – Probabilidade de ocorrência das diferentes casas em função do número de *runs*, usando diferentes geradores de números aleatórios.

Decidiu-se analisar estes dois gráficos por revelarem comportamentos nitidamente diferentes do gerador padrão. O gerador *combRecursive* aparenta ter a velocidade de convergência mais elevada e o gerador *threefry* assume um comportamento ruidoso, em que o valor das probabilidades oscilam bastante em torno do valor final.

iii. Valor adequado da variável *Ndiscard*

A variável *Ndiscard* representa o número de jogadas iniciais que não devem ser usadas para cálculos estatísticos. Para determinar qual o valor mais adequado a atribuir a esta variável realizou-se um estudo parecido com o que será feito na pergunta P3. Estudou-se a probabilidade da marca estar em cada casa ao longo do decorrer do jogo, abaixo estão representados os gráficos para as casas 3 e 5. Tanto para estes gráficos como para os das outras casas é possível verificar que a partir da 10ª jogada a probabilidade já se encontra a menos de 10% do seu valor final (linhas a tracejado). Assim, conclui-se que o valor ótimo para *Ndiscard* é 10.



(a) Casa 3.

(b) Casa 5.

Fig. 6 – Probabilidade da marca estar na casa *x* no fim de cada jogada.

iv. Validação extra

Nas simulações feitas em P1, os valores de *Njogadas*, *runs* e *Ndiscard* foram escolhidos arbitrariamente, sem nenhum resultado que os justificasse. Neste parágrafo discutem-se os valores ideais de *Njogadas* e *run* que foram encontrados à posteriori. Para P1 simulou-se apenas 1 *run* pelo que um número elevado de jogadas era necessário para obter várias amostras para o método de Monte Carlo. Ao aumentar o número de *runs* verificou-se que deixava de ser relevante simular durante muitas jogadas, sendo mais eficaz aumentar o número de *runs*. Para P3 verificou-se que as probabilidades aproximavam-se do seu valor final ao fim de poucas jogadas pelo que se decidiu simular apenas 50 jogadas. Quanto ao número de *runs*, começou-se por usar 100 para o P2 por considerarmos um valor razoável. No entanto, como estudo adicional, decidimos ver o erro relativo do valor $floor(\sqrt{\text{NMC}})$ em relação ao valor final da probabilidade no final das *runs* todas. O gráfico que obtivemos está presente na figura 7, podemos observar que para a casa 2, por exemplo, o erro relativo a partir da 25ª *run* já não é significativo. Assim, concluí-se que não é necessário valores muito elevados do número de *runs* para obtermos resultados com pouco erro.

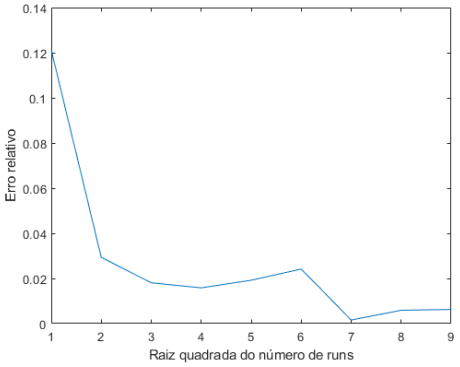


Fig. 7 – Erro relativo em função da raiz quadrada do número de runs.

P3. Estudo do posicionamento da marca ao longo do jogo

Nesta secção, foi utilizado o Método de Monte Carlo (MCM) para estimar a probabilidade da marca se encontrar na posição 4 no fim de cada jogada, até ao final do jogo. Para este efeito modificou-se o programa anterior para produzir o representado no anexo P3. Usou-se um vetor de contadores com dimensão *Njogadas* para, ao longo das várias *runs*, se contar o número de vezes que a marca calhava na posição 4 em cada jogada. Escolheu-se simular o jogo durante 50 jogadas para não sobrecarregar o gráfico e verificou-se que 100 *runs* já era suficiente para obter bons resultados. Para obter ainda melhores resultados, o gráfico da figura 8 foi obtido com 1000 *runs*.

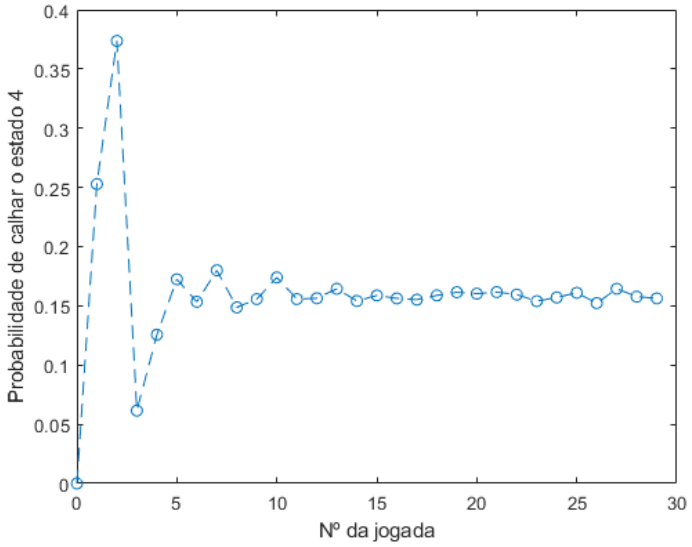


Fig. 8 – Probabilidade da marca estar na posição 4.

Analisando o gráfico nota-se que a probabilidade da peça estar na posição 4 no fim da jogada 0 é nula pois após o primeiro lançamento da moeda a marca apenas se pode deslocar para as posições 1 ou 2. A probabilidade da marca se encontrar na posição 4 no fim da primeira jogada é de cerca de 25%. Considerando o seguinte conjunto para os dois primeiros lançamentos: $[1,1]; [1,2]; [2,1]; [2,2]$ apenas o último permite que a peça se encontre na posição 4 o que corresponde a 25%. Fazendo o mesmo raciocínio para as 8 combinações possíveis dos 3 primeiros lançamentos, facilmente se conclui que apenas 3 resultariam na posição 4 o que corresponde á probabilidade de cerca de 37.5% que se verificou para a jogada 2. Aplicando novamente o mesmo raciocínio chega-se a $1/16 = 6.25\%$ para a jogada 3 o que corrobora o gráfico. A partir da jogada 4 deixa de ser prático fazer a mesma análise, no entanto é possível concluir que com o decorrer do jogo a probabilidade da marca calhar na posição 4 estabiliza em cerca de 15%, o que representa .

P4. Diagrama alterado para nova regra de jogo da prisão

Nesta secção foi pedido a manipulação do diagrama de transição de estados de forma a que a ficha em jogo permanecesse na prisão durante uma jogada. Como resposta à pergunta 4 criou-se o diagrama da figura 9 que visa a acomodar a nova regra. É então possível de observar o aparecimento de um novo estado, *P*, para o qual a ficha é enviada quando aterra na casa de ir para a prisão. Este estado funciona como uma casa intermédia, ficando a peça assim retida durante um lançamento. Na jogada seguinte a marca é movida para a casa três, independentemente do resultado obtido no lançamento da moeda, de forma a dar continuidade ao jogo apartir da posição correta.

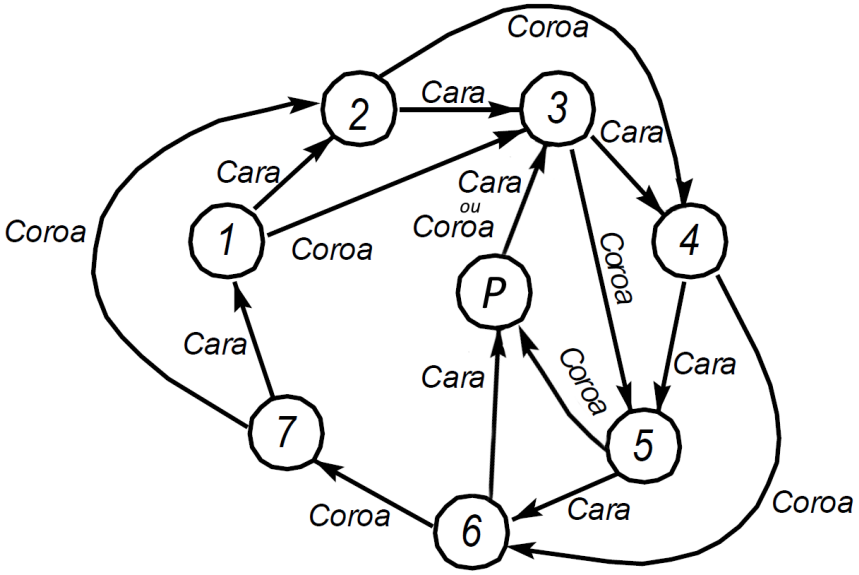


Fig. 9 – Novo diagrama de transição de estado.

Conclusão

Neste trabalho, após as simulações realizadas, foi possível estudar as propriedades das cadeias de Markov recorrendo ao método de Monte Carlo. O sistema de acontecimentos discretos (jogo simplificado de monopólio) constitui uma cadeia de Markov, como pudemos constatar no relatório.

Dos gráficos que obtivemos podemos concluir que a casa que é mais visitada é a casa número 3 e a casa que dá mais lucro é a casa número 6.

Para além disso, foi estudado o número de jogadas a discartar de um transiente inicial para que a probabilidade estabilize. Com o número de jogadas a ignorar estudadas, foi possível, através das respostas às diferentes perguntas, observar que a probabilidade de ocorrência de cada casa converge para um número (sendo que a convergência dá-se com a raiz quadrada do número de runs), o que indica que há uma certa memória num processo estocástico.

Anexos

P1

```
%(...) Inicialização de variáveis%

for i=1:1:NMC
    x = 0; %variável escalar que indica o número do estado em que a marca do jogador está em cada instante;
    espera = espera + 1/NMC;
    waitbar(espera,hh,"Doing stuff");
    for k=1:1:Njogadas+1 % a inicial n conta para o nº de jogadas
        avanca = randi([1 2]); % lançamento da moeda
        coinflips(k)=avanca;
        if x == 0              % no inicio está "fora do tabuleiro"
            x = avanca;        % ou vai para o 1 ou para o 2
        else
            x = estados(avanca,x);  % avança segundo o diagrama
        end
        y(k) = x;
        if k > Ndiscard+1
            z(i,x) = z(i,x)+1;
        end
    end
    if first == 1
        figure(10)
        plot(1:1:Njogadas+1, y, 'o')
        xlabel('Nº da Jogada')
        ylabel('Estado da marca')
        figure(11)
        plot(1:1:Njogadas+1, coinflips, 'o')
        xlabel('Nº da Jogada')
        ylabel('Resultado da moeda')
        first = 0;
    end
end
for i=1:1:Ncasas
    zfreq(i) = sum(z(:,i))/((Njogadas-Ndiscard)*NMC);
    Lucro_av(i) = Aluguer(i) * zfreq(i);
end
%(Plots)%
```

P2

```
%(...) Inicialização de variáveis%
generator = ["twister" "simdTwister" "combRecursive" "multFibonacci" "philox" "threefry" "v5uniform" "v5normal" "v4"];
espera = 0;
%hh = waitbar(espera,"Doing stuff");

for jj=1:1:9
    rng('shuffle', generator(jj));
    coinflips = zeros(1,Njogadas);
    zfreq = zeros(NMC,Ncasas);
    z = zeros(NMC,Ncasas);
    y = zeros(1,Njogadas);
    %(...)%
    %(JOGO)%
    %(...)%
    %testing if transitions are right. First bullet point on question P2.
    for i = 1:1:(length(y)-1)
        salto = y(i+1) - y(i);
        if(salto ~= 1 && salto ~= 2)
           salto = abs(salto);
           if(salto == 3)
               if(~((y(i) == 6 && y(i+1) == 3) &&  coinflips(i+1) == 1))
                    disp("There's a wrong move in this run" )
                    disp(i)
               end
           end
           if(salto == 5)
               if(~((y(i) == 7 && y(i+1) == 2)  &&  coinflips(i+1) == 2))
                    disp("There's a wrong move in this run"  )
               end
           end
           if(salto == 6)
               if(~((y(i) == 7 && y(i+1) == 1) &&  coinflips(i+1) == 1))
                    disp("There's a wrong move in this run"  )
```

```
                end
            end
        end
    end

    %testing if transitions are right.
    for n = 1:1:NMC
        for i=1:1:Ncasas
            zfreq(n,i) = sum(z(1:n,i))/((Njogadas-Ndiscard)*n);
        end
         % espera = espera + 1/NMC;
          %waitbar(espera,hh,"Doing stuff");
    end
    figure(1 + jj)
    plot_legends = zeros(1,Ncasas);
    for i = 1:1:Ncasas
        extr_sup = zeros(1, 100)+ 1.05*zfreq(NMC,i);
        extr_inf = zeros(1, 100)+ 0.95*zfreq(NMC,i);
        %(Plots)%
    end
end
```

P3

```
%(...) Inicialização de variáveis%
for casa = 3:1:5
    counter4 = zeros(1, Njogadas);
    for i=1:1:NMC
        x = 0; %variável escalar que indica o número do estado em que a marca do jogador está em cada instante;
        espera = espera + 1/NMC/7;
        waitbar(espera,hh,"Doing stuff");
        for k=1:1:Njogadas %+1 %se se contar com a jogada 0 +50 jogadas
            avanca = randi([1 2]); % lançamento da moeda
            if x == 0              % no inicio está "fora do tabuleiro"
                x = avanca;        % ou vai para o 1 ou para o 2
            else
                x = estados(avanca,x);  % avança segundo o diagrama
            end
            if x == casa
                counter4(k) = counter4(k)+1; %assinala que na jogada k calhou a casa 4 na run i
            end
        end
    end
    freq4 = counter4/NMC; % calcula a frequencia com que em cada jogada calhou a casa 4
    final = sum(freq4(21:30))/10;
    extr_sup = zeros(1, Njogadas)+ 1.1*final;
    extr_inf = zeros(1, Njogadas)+ 0.9*final;
    %(Plots)%
```