EA721 - Princípios de Controle e Servomecanismo
Turma A

Trabalho Computacional 03

Thiago Maximo Pavão - 247381
Vinícius Esperança Mantovani - 247395

Códigos desenvolvidos:

calculaRegioes.m:

```
syms k1 k2;

n_pontos = 200;

% Definição dos polinômios
polinomio1 = sym([1 1 k1*k2+2 k2+3 k1+4]);
polinomio2 = sym([1 7 2*k2-k1 5+k1*k2 1]);
polinomio3 = sym([1 2 k1+2 3+k2 k1*k2]);
polinomio10 = sym([1 k1 k2 k1*k2 sin(k1) cos(k2) k1+3 k2*log(k1) k1^k2
k2^(1/k1)]);

% Calculando e salvando os pontos estáveis para cada polinômio
% Polinômio 1
tic;
tabela_reg1 = regiaoEstavel_k1k2(polinomio1);
regiao1 = regiaoEstavel(tabela_reg1, [k1 k2], [-3, 3], [-3, 3],
n_pontos, 'Polinômio 1');
save('regiao_polinomio1.mat', 'regiao1');
tempoPolinomio1 = toc;
fprintf('Tempo para Polinômio 1: %.2f segundos.\n', tempoPolinomio1);

% Polinômio 2
tic;
tabela_reg2 = regiaoEstavel_k1k2(polinomio2);
regiao2 = regiaoEstavel(tabela_reg2, [k1 k2], [-3, 3], [-3, 3],
n_pontos, 'Polinômio 2');
save('regiao_polinomio2.mat', 'regiao2');
tempoPolinomio2 = toc;
fprintf('Tempo para Polinômio 2: %.2f segundos.\n', tempoPolinomio2);

% Polinômio 3
tic;
tabela_reg3 = regiaoEstavel_k1k2(polinomio3);
regiao3 = regiaoEstavel(tabela_reg3, [k1 k2], [-3, 3], [-3, 3],
n_pontos, 'Polinômio 3');
```

```
save('regiao_polinomio3.mat', 'regiao3');
tempoPolinomio3 = toc;
fprintf('Tempo para Polinômio 3: %.2f segundos.\n', tempoPolinomio3);

% Polinômio 10
tic;
tabela_reg10 = regiaoEstavel_k1k2(polinomio10);
regiao10 = regiaoEstavel(tabela_reg10, [k1 k2], [-5, 5], [-5, 5],
n_pontos, 'Polinômio 10');
save('regiao_polinomio10.mat', 'regiao10');
tempoPolinomio10 = toc;
fprintf('Tempo para Polinômio 10: %.2f segundos.\n', tempoPolinomio10);

disp('Cálculo e salvamento das regiões concluído.');

%-------------------------------------------------------------------------
-----------------------------------------------------

function routh = regiaoEstavel_k1k2(polinomio)
    routh = sym([]);

    for i = 1:size(polinomio,2)
        if i == 1
            for j = 1:2:size(polinomio,2)
                routh(1, (j+1)/2) = polinomio(j);
            end
        elseif i == 2
            for j = 2:2:size(polinomio, 2)
                routh(2, j/2) = polinomio(j);
            end
        else
            for j = 1:(size(routh, 2)-1)
                routh(i, j) = sym((routh(i-1, 1))*(routh(i-2,j+1)) -
(routh(i-2, 1))*(routh(i-1,j+1)))/(routh(i-1, 1));
            end
        end
    end
end

%-------------------------------------------------------------------------
-----------------------------------------------------

function [ptsEstaveis] = regiaoEstavel(tabela, vars, limk1, limk2,
numPontos, nomePolinomio)
    k1 = linspace(limk1(1), limk1(2), numPontos);
    k2 = linspace(limk2(1), limk2(2), numPontos);

    % Estimativa inicial do tamanho máximo de ptsEstaveis, que pode ser
no pior caso
    maxPts = length(k1) * length(k2);
```

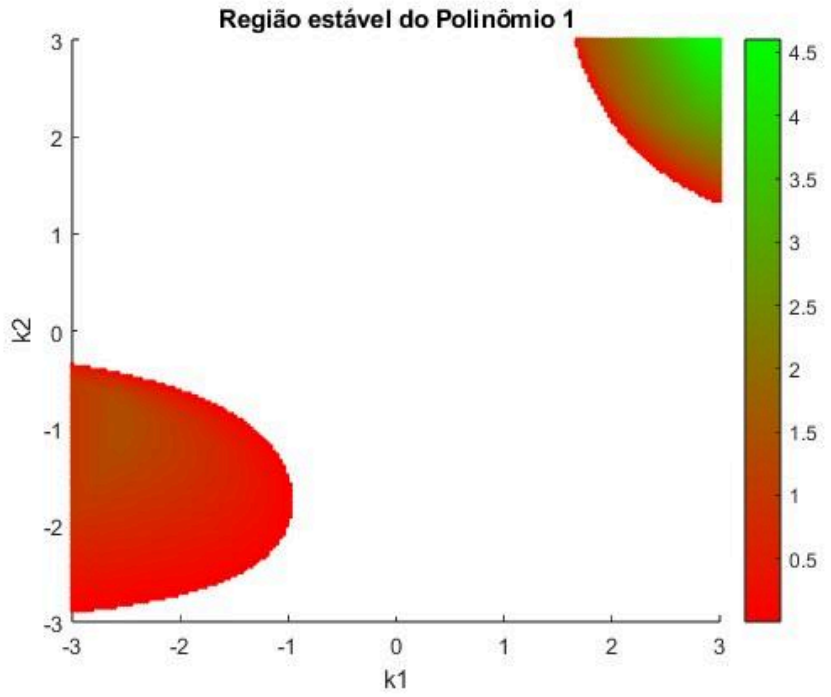
```
    % Pré-aloca a matriz ptsEstaveis
    ptsEstaveis = zeros(maxPts, 3);
    idx = 0;  % Índice para controlar a inserção de pontos estáveis

    totalIteracoes = length(k1) * length(k2);  % Número total de
iterações
    contador = 0;  % Inicializa o contador para a barra de progresso

    % Cria a barra de progresso com o título do polinômio
    h = waitbar(0, sprintf('Calculando %s...', nomePolinomio));

    for i = 1:length(k1)
        for j = 1:length(k2)
            estavel = 1;
            minV = 1e10;

            for k = 1:size(tabela, 1)
                elemento = tabela(k, 1);
                if ~isempty(symvar(elemento))
                    valor = subs(elemento, vars, [k1(i), k2(j)]);
                    if valor <= 0
                        estavel = 0;
                        break;
                    end
                    if valor < minV
                        minV = valor;
                    end
                end
            end

            if estavel
                idx = idx + 1;  % Incrementa o índice
                ptsEstaveis(idx, :) = [k1(i), k2(j), minV];
            end

            % Atualiza a barra de progresso
            contador = contador + 1;
            progresso = contador / totalIteracoes;
            waitbar(progresso, h, sprintf('Calculando Região (%s)',
nomePolinomio));
        end
    end

    % Remove linhas não utilizadas em ptsEstaveis
    ptsEstaveis = ptsEstaveis(1:idx, :);

    % Fecha a barra de progresso após a conclusão
    close(h);
end
```

plotaRegioes.m:

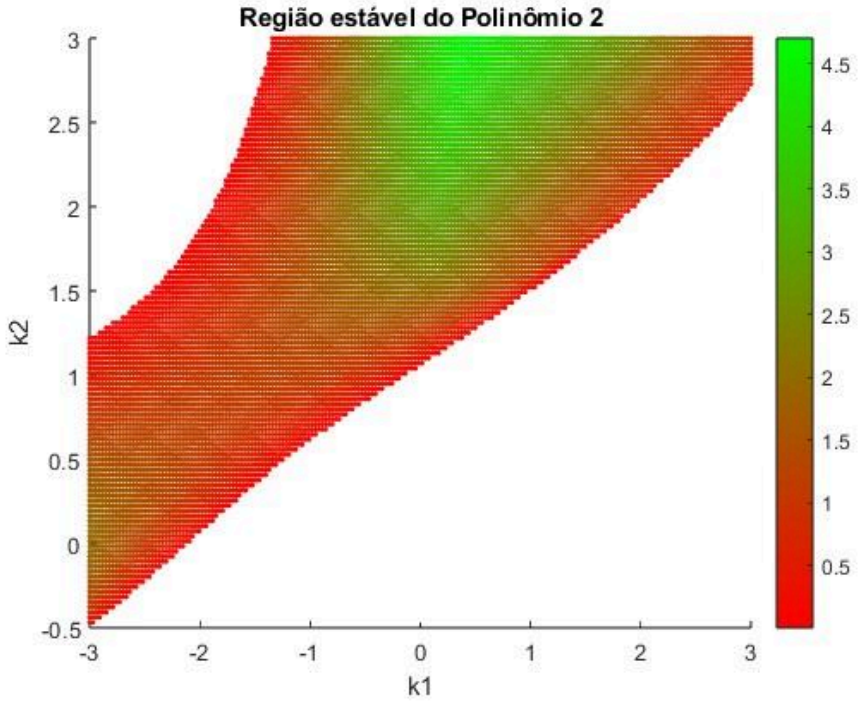
```
% Carregando as regiões salvas
load('regiao_polinomio1.mat');
load('regiao_polinomio2.mat');
load('regiao_polinomio3.mat');
load('regiao_polinomio10.mat');

% Plotando cada polinômio usando a função auxiliar
plotaRegiaoEstavel(regiao1, 'Região estável do Polinômio 1');
plotaRegiaoEstavel(regiao2, 'Região estável do Polinômio 2');
plotaRegiaoEstavel(regiao3, 'Região estável do Polinômio 3');
plotaRegiaoEstavel(regiao10, 'Região estável do Polinômio 10');

function plotaRegiaoEstavel(regiao, titulo)
    figure;
    scatter(regiao(:,1), regiao(:,2), 5, regiao(:,3), 'fill');
    colormap([linspace(1,0,256)', linspace(0,1,256)', zeros(256,1)]);
    colorbar;           % Exibe a barra de cores
    title(titulo);      % Define o título do gráfico
    xlabel("k1");
    ylabel("k2");
end
```

Plot polinômios:

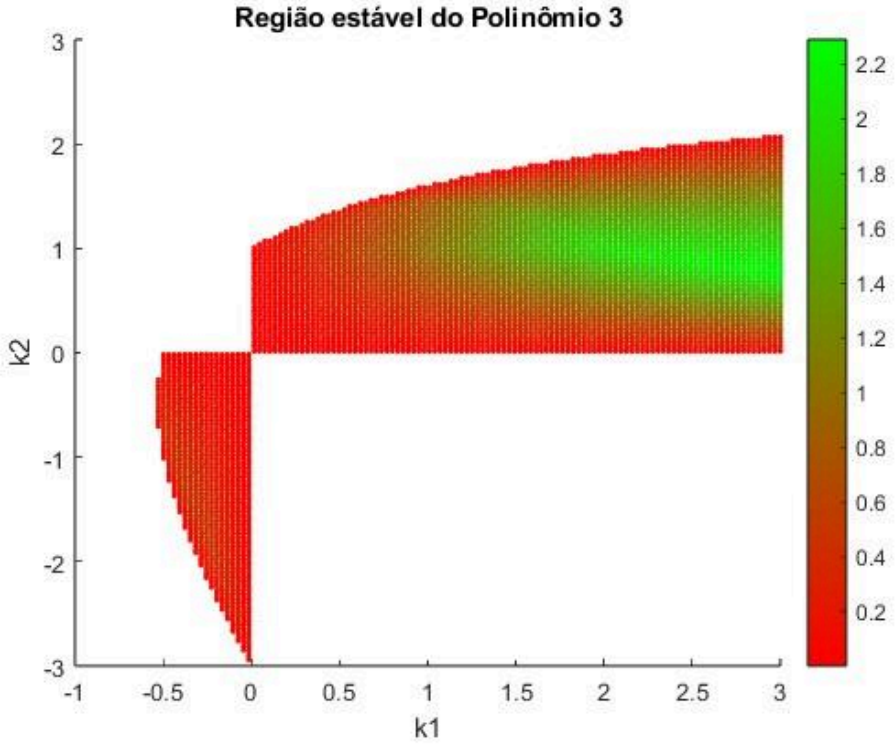
Polinômio 1 → $\Delta(s) = s^4 + s^3 + (k_1k_2 + 2)s^2 + (k_2 + 3)s + k_1 + 4$



Polinômio 2 → $\Delta(s) = s^4 + 7s^3 + (2k_2 - k_1)s^2 + (5 + k_1k_2)s + 1$

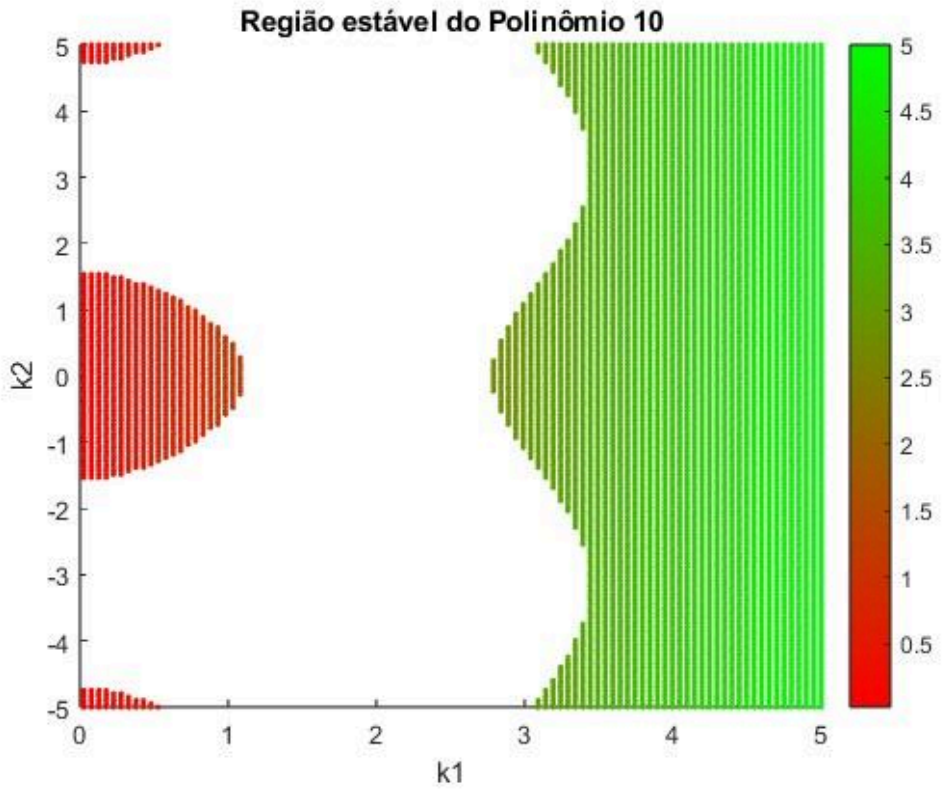


Polinômio 3 → $\Delta(s) = s^4 + 2s^3 + (k_1 + 2)s^2 + (3 + k_2)s + k_1k_2$



Polinômio 10 →

$$P(s) = s^9 + k_1 \cdot s^8 + k_2 \cdot s^7 + k_1 \cdot k_2 \cdot s^6 + \sin(k_1) \cdot s^5 + \cos(k_2) \cdot s^4 + (k_1 + 3) \cdot s^3 + k_2 \log(k_1) \cdot s^2 + k_1^{k_2} \cdot s^1 + k_2^{\frac{1}{k_1}}$$



Tempos de processamento:

Processamento realizado com 200 pontos, totalizando 200x200 = 40000 pares k1, k2 por polinômio.

Tempo para Polinômio 1: 365.11 segundos.
Tempo para Polinômio 2: 368.67 segundos.
Tempo para Polinômio 3: 584.25 segundos.
Tempo para Polinômio 10: 506.58 segundos.