

MÉTODOS NUMÉRICOS - LISTA DE EXERCÍCIOS II

SÉRGIO CORDEIRO

SUMÁRIO

1. Exercícios	2
2. Análise e conclusões	21
3. Anexos	21
Referências	23

1. EXERCÍCIOS

Desenvolva algoritmos em C ou MATLAB (escolha livre) para:

1. Solução de SL de ordem $n \times n$ usando decomposição LU com pivotação parcial. Calcular o determinante.

O programa **exercmat.c**, em anexo, escrito em C, lê um sistema de equações gerado pelo MATLAB, resolve-o pelo método de decomposição LU com pivotação parcial, e calcula o determinante da matriz e a norma 2 da matriz-resultado, para conferência. Basta digitar:

exercmat 4 n

onde n é o tamanho do sistema ($n \times n+1$). Ele também calcula o número de operações em ponto flutuante necessário para cada etapa. Os resultados obtidos estão tabelados a seguir. Foi usada apenas precisão simples, de forma a favorecer a ocorrência de erros de arredondamento.

n	custo ¹			determinante	norma 2
	solução	determinante ²	norma 2		
3	63	3	6	-0.112721	4.464723
30	19701	30	60	26.168304	1.938657
300	18138546	300	600	<i>overflow</i>	25.915476

O sistema foi gerado pelo MATLAB na solução da primeira lista de exercícios.

¹Número de operações de ponto flutuante necessárias.

²Adicional ao custo da solução.

2. Decomposição de Cholesky.

O programa `exercmat.c`, em anexo, escrito em C, lê um sistema de equações gerado pelo MATLAB, resolve-o pelo método de decomposição de Cholesky com pivotação parcial, e calcula o determinante da matriz e a norma 2 da matriz-resultado, para conferência. Basta digitar:

`exercmat 5 n`

onde n é o tamanho do sistema ($n \times n+1$). Ele também calcula o número de operações em ponto flutuante necessário para cada etapa.

A matriz precisa ser simétrica e definida positiva para que esse método possa ser empregado. O programa abaixo gera, no MATLAB, um sistema adequado, grava os valores em disco, e calcula o determinante da matriz e a norma 2 do resultado, para conferência.

LISTING 1. `gerachol.m`

```

1 function gerachol(n)
2 %Gera um sistema a partir de valores aleatórios na faixa ]0 - 1[, grava-o em disco,
   resolve-o e calcula o determinante e a norma 2 do resultado.
3 %A matriz gerada é simétrica e definida positiva.
4 % Gera matriz não singular
5 test = 0;
6 while test == 0
7     A = rand(n);
8     test = det(A);
9 end
10 % Gera matriz simétrica definida positiva
11 S = A * A';
12 % Gera o sistema
13 b = rand(n,1);
14 C = [S, b];
15 save('C', 'C');
16 disp(sprintf("Sistema gerado e gravado."));
17 disp(sprintf("Determinante = %f", test^2));
18 % Resolve o sistema
19 INVS = S ^ (-1);
20 x = INVS * b;
21 disp(sprintf("Norma 2 do resultado = %f", norm(x, 2)));
22 end

```

De acordo com [WEISSTEIN 2016],

A real symmetric matrix A is positive definite iff there exists a real nonsingular matrix M such that

$$A = MM^{(T)}$$
where $M^{(T)}$ is the transpose [...].

por isso a matriz foi gerada como o quadrado de outra matriz que se garantiu não ser singular. Com isso, a condição mencionada é satisfeita. Os resultados obtidos estão tabelados a seguir. Foi usada apenas precisão simples, de forma a favorecer a ocorrência de erros de arredondamento.

n	det. MATLAB	norma 2 MATLAB	custo ³		det.	norma 2
			solução	det. ⁴		
3	8.4343	6197.920598	116	4	0.000026	6191.667480
30	0.00002643	904.156803	12500	31	8.460002	901.309753
300	2.96×10^{148}	4121.211866	9278000	301	<i>overflow</i>	4364.376465

³Número de operações de ponto flutuante necessárias.

⁴Adicional ao custo da solução.

3. Refinamento de solução de SL (nxn).

O programa **exercmat.c**, em anexo, escrito em C, lê um sistema de equações gerado pelo MATLAB, calcula a solução pelo método de decomposição LU e refina-a, se necessário, bem como calcula a norma 2 do resultado, para conferência. Basta digitar:

```
exercmat 11 n [e]
```

onde n é o tamanho do sistema ($n \times n+1$) e e (opcional), é o erro tolerado. Ele também calcula o número de operações em ponto flutuante necessário para cada etapa.

No refinamento, foi empregada a norma infinita do resíduo

$$b - Ax$$

como critério de parada. Quando o procedimento diverge ou converge muito devagar, adota-se o melhor valor disponível até então.

Como o refinamento exige a solução sucessiva de diversos sistemas lineares com a mesma matriz, o mais recomendável é usar um método de decomposição, em lugar do de eliminação Gaussiana, por exemplo. Este é o mais eficiente na solução de um sistema único, mas o custo é muito alto para empregos sucessivos.

Os resultados obtidos estão tabelados a seguir. Foi usada apenas precisão simples, de forma a favorecer a ocorrência de erros de arredondamento.

n	norma 2	custo ⁵	iterações	tolerado	erro inicial	erro final	norma 2
3	4.464723	75	0	10^{-6}	0.000000	0.000000	4.464723
30	1.938658	20631	0	10^{-6}	0.000000	0.000000	1.938657
300	25.915637	18501847	1	10^{-5}	0.000054	0.000009	25.915562
300	25.915637	18774848	2 ⁶	10^{-6}	0.000054	0.000009	25.915701

O sistema foi gerado pelo MATLAB na solução da primeira lista de exercícios.

⁵Número de operações de ponto flutuante necessárias.

⁶O método divergiu neste caso.

4. Cálculo da inversa de uma matriz ($n \times n$).

O programa **exercmat.c**, em anexo, escrito em C, lê um sistema de equações gerado pelo MATLAB e calcula a matriz inversa e o determinante pelo método de eliminação de Gauss com pivotação parcial, bem como a norma 2 do resultado, para conferência. Basta digitar:

exercmat 9 n

onde n é o tamanho do sistema ($n \times n+1$). Ele também calcula o número de operações em ponto flutuante necessário para cada etapa.

Os resultados obtidos estão tabelados a seguir. Foi usada apenas precisão simples, de forma a favorecer a ocorrência de erros de arredondamento.

n	custo ⁷			determinante	norma 2
	inversão ⁸	solução	norma 2		
3	105	9	6	-0.112721	4.464724
30	74391	900	600	26.168293	1.938658
300	72226446	90000	600	<i>overflow</i>	25.915840

O sistema foi gerado pelo MATLAB na solução da primeira lista de exercícios.

⁷Número de operações de ponto flutuante necessárias.

⁸Cálculo do determinante incluído.

5. Solução de SL de ordem $n \times n$ usando o método iterativo de Jacobi.

O programa **exercmat.c**, em anexo, escrito em C, lê um sistema de equações gerado pelo MATLAB, resolve-o pelo método iterativo de Jacobi e calcula a norma 2 do resultado, para conferência. Basta digitar:

exercmat 12 n

onde n é o tamanho do sistema ($n \times n+1$). Ele também calcula o número de operações em ponto flutuante necessário para cada etapa.

Como o sistema gerado pelo MATLAB na solução da primeira lista de exercícios é muito mal condicionado, o método falhou para todos os tamanhos testados. Por isso, desenvolveu-se uma função no MATLAB, listada abaixo, para gerar uma matriz diagonalmente dominante.

LISTING 2. geradsis.m

```
1 function geradsis(n)
2 % Gera um sistema 'n' x 'n' diagonalmente dominante a partir de valores aleatórios,
   grava-a em disco, resolve-o e calcula a norma 2 do resultado.
3 A = rand(n);
4 for i = 1:n
5     A(i,i) = 10 + A(i,i) * 10;
6 end
7 b = rand(n,1);
8 S = [A, b];
9 save('D', 'S');
10 disp(sprintf("Sistema gerado e gravado."));
11 INVA = A^(-1);
12 X = INVA * b;
13 disp(sprintf("Norma 2 da solução = %f", norm(X, 2)));
14 end
```

Os resultados obtidos estão tabelados a seguir. Foi usada apenas precisão simples, de forma a favorecer a ocorrência de erros de arredondamento.

n	norma 2	custo ⁹	iterações	norma 2	tolerância
3	0.606234	204	8	0.606233	10^{-5}
		225	9	0.606234	10^{-6}
30	0.166238	1831980	1000	0.166224	10^{-5}
		2318760	1266	0.166236	10^{-6}
300	0.389537	- ¹⁰			10^{-5}

⁹Número de operações de ponto flutuante necessárias.

¹⁰O método divergiu neste caso.

6. Solução de SL de ordem $n \times n$ usando o método iterativo de Gauss Seidel.

O programa **exercmat.c**, em anexo, escrito em C, lê um sistema de equações gerado pelo MATLAB, resolve-o pelo método iterativo de Gauss-Seidel e calcula a norma 2 do rtado, para conferência. Basta digitar:

exercmat 13 n

onde n é o tamanho do sistema ($n \times n+1$). Ele também calcula o número de operações em ponto flutuante necessário para cada etapa.

Foram usados os mesmos sistemas gerados para o problema anterior.

Os resultados obtidos estão tabelados a seguir. Foi usada apenas precisão simples, de forma a favorecer a ocorrência de erros de arredondamento.

n	norma 2	custo ¹¹	iterações	tolerância	norma 2
3	0.606234	99	3	10^{-5}	0.606234
		120	4	10^{-6}	0.606234
30	0.166238	14790	7	10^{-5}	0.166238
		16620	8	10^{-6}	0.166237
300	0.389537	12802800	70	10^{-5}	0.389532
		16228500	89	10^{-6}	0.389537

¹¹Número de operações de ponto flutuante necessárias.

7. Determinação do maior e do menor auto-valor de uma matriz e cálculo do número de condicionamento.

O programa **exercmat.c**, em anexo, escrito em C, lê uma matriz gerada pelo MATLAB e calcula seus autovalores extremos pelo método das potências (direta e inversa). Basta digitar:

exercmat 14 n

onde n é o tamanho da matriz ($n \times n$). Ele também calcula o número de operações em ponto flutuante necessário para cada etapa.

Foram usadas matrizes geradas para a lista de exercícios anterior.

Os resultados obtidos estão tabelados a seguir. Foi usada apenas precisão simples, de forma a favorecer a ocorrência de erros de arredondamento.

n	Autovalores		custo ¹²		iterações		tolerância
	menor	maior	menor	maior	menor	maior	
3	0.499284	1.361155	265	232	3 ¹³	7	10 ⁻⁵
3	0.499284	1.361152	265	292	3 ¹³	9	10 ⁻⁶
30	0.057670	15.053532	34897	4000	7	3 ¹³	10 ⁻⁵
30	0.057670	15.053532	37066	4000	8	3 ¹³	10 ⁻⁶
300	0.181859	149.791565	18505777	454849	2 ¹³	4	10 ⁻⁵
300	0.181859	149.791534	18505777	546058	2 ¹³	5	10 ⁻⁶

¹²Número de operações de ponto flutuante necessárias.

¹³O método divergiu nestes casos.

8. Determinação de todos os autovalores e autovetores de uma matriz.

O programa **exercmat.c**, em anexo, escrito em C, lê uma matriz gerada pelo MATLAB e calcula seus autovalores pelo método de Jacobi. Basta digitar:

```
exercmat 15 n
```

onde n é o tamanho da matriz ($n \times n$). Ele também calcula o número de operações em ponto flutuante necessário para cada etapa.

Foram usadas matrizes geradas para a lista de exercícios anterior.

Os resultados obtidos estão tabelados a seguir. Foi usada apenas precisão simples, de forma a favorecer a ocorrência de erros de arredondamento.

n	custo ¹⁴	iterações	tolerância
3	228	1	10^{-6}
30	22800	19	10^{-6}
300	2188800	191	10^{-5}
300	3123600	273	10^{-6}

¹⁴Número de operações de ponto flutuante necessárias.

9. Pesquise sobre sistemas tridiagonais e desenvolva um método para sua solução.

Um sistema linear tridiagonal é aquele em que a matriz dos coeficientes A é tridiagonal. Uma matriz tridiagonal é uma matriz quadrada em que os coeficientes são $a_{i,j}$ nulos para $|j - i| > 1$; em outras palavras, apenas coeficientes na diagonal principal e nas diagonais adjacentes não são nulos.

Uma matriz tridiagonal genérica pode ter seus coeficientes expressos da forma usual:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_{3,2} & a_{3,3} & a_{3,4} & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

ou da forma alternativa:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & a_n & b_n \end{bmatrix}$$

Esta última facilita o armazenamento eficaz de grandes matrizes, pois é preciso gravar apenas os 3 vetores a , b e c , todos com comprimento n . Note-se que sempre $a_1 = c_n = 0$.

Lançando-se mão da forma alternativa, o sistema tridiagonal $Ax = d$, cuja equação genérica é:

$$\sum_{j=1}^n a_{i,j} x_j = d_i$$

pode ser escrito simplesmente como:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

o que facilita a solução por meio de uma modificação da eliminação Gaussiana conhecida como **algoritmo de Thomas**. O custo computacional deste método é linear em n : $\mathcal{O}(n)$. A matriz inversa V pode ser obtida

por meio das expressões seguintes:

$$v_{i,j} = \begin{cases} (-1)^{i+j} \frac{\theta_{i-1}\phi_{j+1}}{\theta_n} \prod_{k=i}^{j-1} c_k & i \leq j \\ (-1)^{i+j} \frac{\theta_{j-1}\phi_{i+1}}{\theta_n} \prod_{k=j}^{i-1} a_k & i > j \end{cases}$$

$$\theta_i = \begin{cases} 1 & i = 0 \\ b_1 & i = 1 \\ b_i\theta_{i-1} - a_{i-1}c_{i-1}\theta_{i-2} & 2 \leq i \leq n \end{cases}$$

$$\phi_i = \begin{cases} 1 & i = n+1 \\ b_n & i = n \\ b_i\phi_{i+1} - a_i c_i \phi_{i+2} & n-1 \geq i \geq 1 \end{cases}$$

V é sempre uma matriz semi-separável. Toda matriz tridiagonal é um tipo de matriz de Hessenberg, por isso pode ser reduzida a uma matriz triangular por meio de processos iterativos, como a decomposição QR. Esses processos são muito estáveis numericamente.

Existem algoritmos para tridiagonalizar matrizes genéricas.

10. Pesquisa sobre a técnica de solução de SL chamada GMRES.

O algoritmo GMRES (*Generalized Minimal Residual Method*) é um método iterativo para solução de sistemas lineares genéricos que minimiza a norma Euclidiana do resíduo:

$$r = Ax - b$$

Ele foi desenvolvido por Saad e Schultz em 1986, a partir dos algoritmos MINRES e DIIS, e seu custo computacional está entre $\mathcal{O}(pn^2 + p^2n)$ para matrizes densas e $\mathcal{O}(p^2 + pn)$ para matrizes muito esparsas, onde p é o número de iterações necessárias.

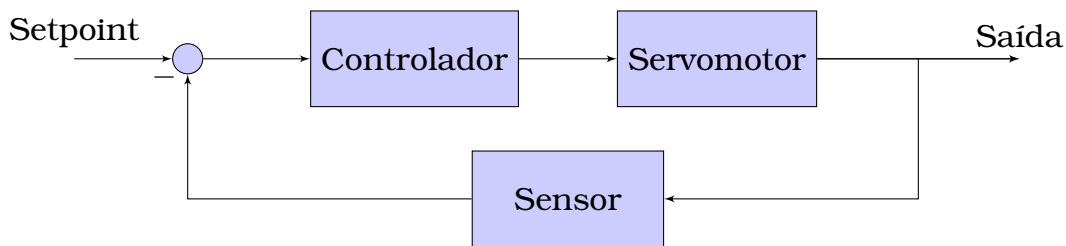
A convergência é garantidamente monotônica e $p < n$; em geral, $p \ll n$, o que torna o custo muito atrativo. Outra característica do método é que a velocidade da convergência não depende do número de condição, e sim do valor absoluto do menor autovalor.

Outra grande vantagem do algoritmo é que ele não exige que A seja diagonalmente dominante ou definida positiva.

A minimização do resíduo, que consiste na solução de um problema de mínimos quadrados, baseia-se na decomposição QR não de A e sim da matriz H , que tem dimensão $(p+1 \times p)$ e que é obtida por meio de iteração de Arnoldi.

11. Estude a resposta dinâmica (estável/não estável (encontrar e avaliar os autovalores)) de um sistema real (o aluno deverá propor e descrever completamente o sistema).

A seguir apresenta-se uma visão simplificada do funcionamento de um sistema de posicionamento de antena terrestre para comunicação com satélite, seguindo o apresentado por [PHILLIPS 1995] Modelos mais sofisticados podem ser encontrados em [KAWAKAMI 1989] e [NASA 1980]. Um sistema de posicionamento de antena de satélite consiste de um controlador digital, um sensor de posição e um servomotor de potência, como ilustrado no diagrama de blocos abaixo.



O servomotor em geral é um motor C.C., com corrente de campo constante, que apresenta resposta linear dentro da faixa de operação. A equação diferencial que o modela é a seguinte:

$$(1) \quad J \frac{d^2 y}{dt^2} + \frac{BR_a + K_T K_b}{R_a} \frac{dy}{dt} - \frac{K_T}{R_a} V_a = 0$$

onde y é o deslocamento angular em relação à posição de repouso, V_a é a tensão aplicada à armadura, J é o momento de inércia total do conjunto, B é o amortecimento total devido a fricção, R_a é a resistência elétrica da armadura, K_T é a relação de torque do motor, dada por $\frac{\tau}{i}$, onde τ é o torque e i , a corrente na armadura, e K_b é a relação de velocidade do servomotor, dada por $\frac{\epsilon}{\omega}$, onde ϵ é a força eletromotriz e ω , a velocidade angular. Todos os parâmetros são constantes e independentes do tempo e a indutância da armadura, L_a , em geral pode ser desprezada.

Assim, a função de transferência do servomotor pode ser expressa por:

$$(2) \quad \mathcal{G}(s) = \frac{\mathcal{Y}(s)}{\mathcal{V}_a(s)} = \frac{A}{s(s+a)}$$

$$\text{com } A = \frac{K_T}{JR_a} \text{ e } a = \frac{BR_a + K_T K_b}{JR_a}.$$

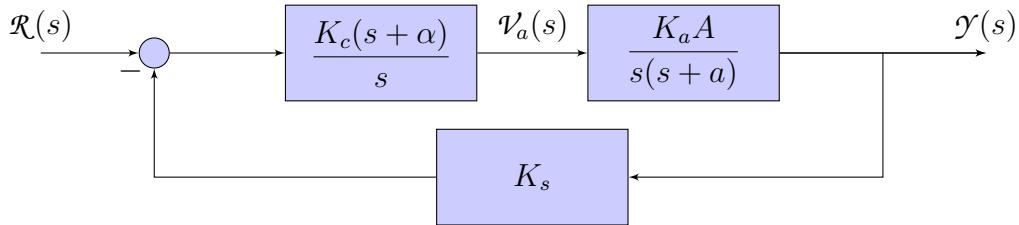
Note-se que tanto A quanto a são sempre positivos, porque o servomotor é um sistema fisicamente estável.

Segundo [KAWAKAMI 1989], o erro máximo tolerado para y é de 0.015 graus, devido à pequena largura do feixe de ondas a ser transmitido e recebido. [NASA 1980] limita o erro de posicionamento a 3'.

O sensor normalmente é do tipo *encoder* de posição, e apresenta resposta linear e baixo ganho. Sua função de transferência $\mathcal{H}(s)$, portanto, pode ser expressa como uma constante K_s , menor que 1. Além disso, um amplificador de potência, de ganho K_a , é sempre colocado para alimentar o motor, para aumentar a sensibilidade e melhorar a resposta.

Para um satélite não-geoestacionário o motor gira continuamente, dando uma volta a cada 24 h. Assim, a função $\mathcal{D}(s)$ deve prover pelo menos mais um polo em $s = 0$ de forma a zerar o erro em regime estacionário. Vamos assumir, por simplicidade, uma função proporcional + integral ideal: $\mathcal{D}(s) = \frac{K_c(s+\alpha)}{s}$.

A partir desses dados, podemos redesenhar o diagrama de blocos:



Os dados de placa do servomotor TSM3304 da Tamagawa são os seguintes [?]:

Tensão	V_a	200 V
Potência	P	750 W
Torque	τ	2.39 N · m
Velocidade	ω	3000 min ⁻¹

Servomotores de mercado com essa potência são normalmente motores C.A., não C.C. Esses dispositivos são construídos de forma a apresentar resposta linear, o que não acontece com os motores de indução usuais; [?] traz gráficos que demonstram essa característica. Neste trabalho continuaremos usando as equações e os termos empregados para motores C.C., apesar de não se aplicarem estritamente a essa classe de servomecanismos.

Com esses dados, podemos calcular os parâmetros do motor, de acordo com o roteiro apontado por [NASA 1980], e considerando uma perda de

5% na resistência da armadura:

$$\begin{aligned} i_a &= \frac{P + \Delta P}{V_a} \\ &= \frac{750 \text{ W} \times 1.05}{200 \text{ V}} \\ &= 3.94 \text{ A} \end{aligned}$$

$$\begin{aligned} R_a &= \frac{\Delta P}{i_a^2} \\ &= \frac{750 \text{ W} \times 0.05}{(3.94 \text{ A})^2} \\ &= 2.42 \text{ } \Omega \end{aligned}$$

$$\begin{aligned} K_T &= \frac{\tau}{i_a} \\ &= 2.39 \text{ N} \cdot \text{m} / 3.94 \text{ A} \\ &= 0.607 \text{ V} \end{aligned}$$

$$\begin{aligned} \epsilon &= V - R_a i_a \\ &= 200 \text{ V} - 2.42 \text{ } \Omega \times 3.94 \text{ A} \\ &= 190 \text{ V} \end{aligned}$$

$$\begin{aligned} \omega &= 3000 \text{ min}^{-1} \\ &= \frac{3000}{60 \text{ s}} \\ &= 50.0 \text{ s}^{-1} \end{aligned}$$

$$\begin{aligned} K_b &= \frac{\epsilon}{\omega} \\ &= \frac{190 \text{ V}}{50 \text{ s}^{-1}} \\ &= 3.80 \text{ V} \cdot \text{s} \end{aligned}$$

O momento de inércia da antena é dado aproximadamente por $J = \frac{mR^2}{4}$, onde R é o raio do disco, se a considerarmos como um disco plano e sua espessura for desprezada [MOON 2008]. Para uma antena típica [GATR 2015], com raio de 2,4 m e pesando 45,4 kg, teremos:

$$J = \frac{45.4 \text{ kg} \times (2.4 \text{ m})^2}{4} = 260 \text{ N} \cdot \text{m}^2$$

O atrito devido à fricção, segundo [NASA 1980], é altamente não-linear; os autores optaram por considerá-lo nulo para cálculo da função de transferência e tratar seu efeito como um distúrbio $\mathcal{T}(s)$ aplicado ao processo. Aqui adotaremos um procedimento mais simples: fazendo, na equação 1, a aproximação $\frac{d^2y}{dt^2} = 0$, podemos estimar um limite máximo para B :

$$\begin{aligned} \frac{B_{max}R_a + K_T K_b}{R_a} \frac{dy}{dt} &= \frac{K_T}{R_a} V_a \\ \implies B_{max} &= \frac{K_T(V_a - \omega K_b)}{R_a} \\ &= \frac{0.607 \text{ V} \cdot \text{s} (200 \text{ V} - 50.0 \text{ s}^{-1} \times 3.80 \text{ V} \cdot \text{s})}{2.42 \Omega} \\ &= 2.51 \text{ N} \end{aligned}$$

Assumiremos, então, por simplicidade, $B = 2.00 \text{ N}$ em condições típicas de funcionamento. Agora, os coeficientes da função de transferência em malha aberta podem ser calculados:

$$\begin{aligned} A &= \frac{K_T}{JR_a} \\ &= \frac{0.607 \text{ V} \cdot \text{s}}{260 \text{ N} \cdot \text{m}^2 \times 2.42 \Omega} \\ &= 9.65 \times 10^{-4} \\ a &= \frac{BR_a + K_T K_b}{JR_a} \\ &= \frac{2.00 \text{ N} \times 2.42 \Omega + 0.607 \text{ V} \cdot \text{s} \times 3.80 \text{ V} \cdot \text{s}}{260 \text{ N} \cdot \text{m}^2 \times 2.42 \Omega} \\ &= 1.14 \times 10^{-2} \end{aligned}$$

O pequeno valor de A deixa mais clara a necessidade do amplificador na entrada. Para efeito de análise, consideraremos $K_s = 0.4$, $K_a = 20$ e tentaremos encontrar valores satisfatórios para K_c e α . Lembremos que, como o amplificador precisa ter alta potência na saída, da ordem de 750 W, o ganho não pode ser muito elevado.

A função de transferência em malha fechada do sistema será:

$$\begin{aligned}
 \mathcal{F}(s) &= \frac{\mathcal{D}(s)\mathcal{H}(s)\mathcal{G}(s)}{1 + \mathcal{D}(s)\mathcal{H}(s)\mathcal{G}(s)} \\
 &= \frac{K_c K_s K_a A(s + \alpha)}{s^2(s + a) + K_c K_s K_a A(s + \alpha)} \\
 (3) \quad &= \frac{K_c C(s + \alpha)}{s^3 + as^2 + K_c Cs + K_c C\alpha}
 \end{aligned}$$

com

$$\begin{aligned}
 C &= K_s K_a A \\
 &= 0.4 \times 20 \times 9.65 \times 10^{-4} \\
 &= 7.72 \times 10^{-3}
 \end{aligned}$$

A equação diferencial correspondente a 3 é:

$$(4) \quad \frac{d^3 y}{dt^3} + a \frac{d^2 y}{dt^2} + K_c C \frac{dy}{dt} + K_c C \alpha y = K_c C \frac{df}{dt} + K_c C \alpha f$$

Na formulação de variáveis de estado:

$$\begin{aligned}
 x_1 &= y \\
 x_2 &= \frac{dy}{dt} = \dot{x}_1 \\
 x_3 &= \frac{d^2 y}{dt^2} = \dot{x}_2 \quad \Rightarrow \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -K_c C \alpha & -K_c C & -a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ K_c C \alpha & K_c C \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\
 v_1 &= f \\
 v_2 &= \frac{df}{dt} = \dot{v}_1
 \end{aligned}$$

ou, mais brevemente, $\dot{\mathbf{x}} = \mathbf{L}\mathbf{x} + \mathbf{M}\mathbf{v}$.

Os polos de $\mathcal{F}(s)$ são os autovalores da matriz \mathbf{L} . Foram geradas, no MATLAB, algumas matrizes para diversos valores de K_c e α , através da função listada abaixo. Ela também calcula os autovalores.

Para todos os valores testados há um autovalor puramente real e dois complexos, obviamente conjugados. Para $\alpha = 0.1$, a parte real desse par conjugado é positiva, independentemente do valor de K_c , o que resulta num sistema instável. Aparentemente, à medida que α aumenta, o valor da parte real dos polos complexos se aproxima assintoticamente de -0.006 , e a do polo real tende a 0 . O aumento de K_c diminui a estabilidade relativa do sistema, mas não parece suficiente para provocar instabilidade estrita. A melhor combinação parece ser $K_c = 10000$ e $\alpha = 0.001$, que resulta no seguinte conjunto de autovalores: $\{-0.00100$ e $-0.00520 \pm j8.77$, para uma função de transferência do controlador $\mathcal{D}(s) = 10000 + \frac{10}{s}$, cuja implementação é perfeitamente viável.

2. ANÁLISE E CONCLUSÕES

Alguns algoritmos utilizam extração de raiz quadrada e divisões, que têm um custo computacional dependente da arquitetura do processador usado. Aqui adotaram-se os valores recomendados por [FAH 2013] e [INTEL 2016].

3. ANEXOS

Os seguintes arquivos constam do anexo (arquivo **exercmat1.zip**):

- arquivo fonte em C **exercmat.c**
 - arquivos fonte em MATLAB:
 - **gerachol.m**: problema 2
 - arquivos de dados:
 - **Sn**: problemas 1, 3 e 4
 - **Cn**: problemas 2 e 8
 - **Dn**: problemas 5 e 6
 - **MatAn**: problema 7
 - **Lm**: problema 11
-

REFERÊNCIAS

- [FAH 2013] Folding@home, Pande Lab, Stanford University, **FAQ: FLOPS**. Disponível em <https://folding.stanford.edu/home/faq/faq-flops/>, acesso em 17/03/2016.
- [GATR 2015] GATR Technologies, **2.4m Antenna System Datasheet**. Disponível em <http://www.gatr.com/products/2-4-antenna-system>, acesso em 21/03/2016.
- [INTEL 2016] INTEL Corporation, **Intel 64 and IA-32 Architectures Optimization Reference Manual**, 2016, Tab. 15-3, pag. 15-9 a 15-10.
- [KAWAKAMI 1989] Y. KAWAKAMI, H. HOJO e M. UEBA, **Control Design of an Antenna Pointing Control System with Large On-board Reflectors**, in T. NISHIMURA - Automatic Control in Aerospace 1989: Selected Papers from The IFAC Symposium.
- [MOON 2008] Francis C. MOON, **Applied Dynamics: With Applications to Multi-body and Mechatronic Systems**, 2nd Ed., 2008, Weinheim, Wiley-VCH, ISBN 978-3-527-40751-4, Chap. 5, pp. 228 a 231.
- [NASA 1980] National Aeronautics and Space Administration, **Satellite Power System (SPS) Antenna Pointing Control**, <http://www.nss.org/settlement/ssp/library/NASACR3350-AntennaPointingControl.pdf>, 21/03/2016
- [PHILLIPS 1995] Charles L. PHILLIPS e H. Troy NAGLE, **Digital Control System Analysis and Design**, Prentice Hall, Englewood Cliffs, 1995, 3rd Ed..
- [TAMAGAWA 2016] TAMAGAWA Seiki Corporation, **TBL-iVseries AC Servomotor**. Disponível em http://www.tamagawa-seiki.com/pdf/download/1699N1EJ_shusei.pdf, acesso em 21/03/2016.
- [WEISSTEIN 2016] Eric W. WEISSTEIN, **Positive Definite Matrix MathWorld**. Disponível em <http://mathworld.wolfram.com/PositiveDefiniteMatrix.html>, acesso em 17/03/2016.

Programas testados com **Octave** 4.0.0 e **MinGW** C 4.8.2:

<https://www.gnu.org/software/octave/>

<https://www.mingw.org>

Texto formatado com **pdflatex** em ambiente **MiKTeX** 2.9:

<http://miktex.org/download/>