Alexandre Nascimento, Jr.

Alocação de Polos Em Regiões do Plano Complexo Para Sistemas Discretos via LMIs

Campo Grande, MS
Novembro 2022

Alexandre Nascimento, Jr.

Alocação de Polos Em Regiões do Plano Complexo Para Sistemas Discretos via LMIs

Universidade Federal de Mato Grosso do Sul

Orientador: Prof. Dr. Victor Yoshimura

Campo Grande, MS
Novembro 2022

Alexandre Nascimento, Jr.

Alocação de Polos Em Regiões do Plano Complexo Para Sistemas Discretos via LMIs

Prof. Dr. Victor Yoshimura
Orientador

Prof. Dr. Fábio Iaione
Convidado 1

Campo Grande, MS
Novembro 2022

Resumo

Este trabalho implementou em *software* algoritmos desenvolvidos nos últimos anos na Teoria de Controle que aproximam em LMI regiões do plano z que não são convexas. A partir de parâmetros de projetos informados, o algoritmo verifica a factibilidade da solução e, caso possível, retorna um compensador que estabiliza o sistema discretizado.

Sumário

1 Introdução

Notação:

- $\mathcal{V}(A) = A + A'$;
- $\bar{\mathcal{V}}(A) = A - A'$
- $\operatorname{loc}(v_1, v_2)$ determina o ponto que a reta que passa por v_1 e v_2 cruza o eixo real;
- $\operatorname{ang}(v_1, v_2)$ refere-se ao ângulo entre aquela reta e o eixo real.

2 Região de Desempenho Garantido

A alocação de polos é uma das principais ferramentas da teoria de controle, pois a partir desta, é possível projetar um sistema que seja estável e que tenha um bom desempenho **??**). A operação de alocar polos de um sistema linear dentro de uma região específica é chamada $\mathscr{D}$-estabilidade **??**).

Entende-se por estável o sistema que, em termos de resposta a estímulos, possui uma convergência ao zero da resposta natural, restando apenas a reposta forçada (**??**). Assim, para um intervalo de tempo determinado, espera-se que o sistema apenas tenha dinâmica referente à entrada aplicada. Neste contexto, a estabilidade é o ponto de partida para projetos de compensadores.

A resposta ao impulso no domínio do tempo de um sistema linear e invariante no tempo tem as características da figura 1. O máximo da curva é o máximo sobressinal (*overshoot* em inglês), onde é o pico da saída, dado em porcetagem. O parâmetro t_s representa o tempo de acomodação, onde é o instante de tempo que o sinal entra em um intervalo definido e não sai (normalmente 2%).

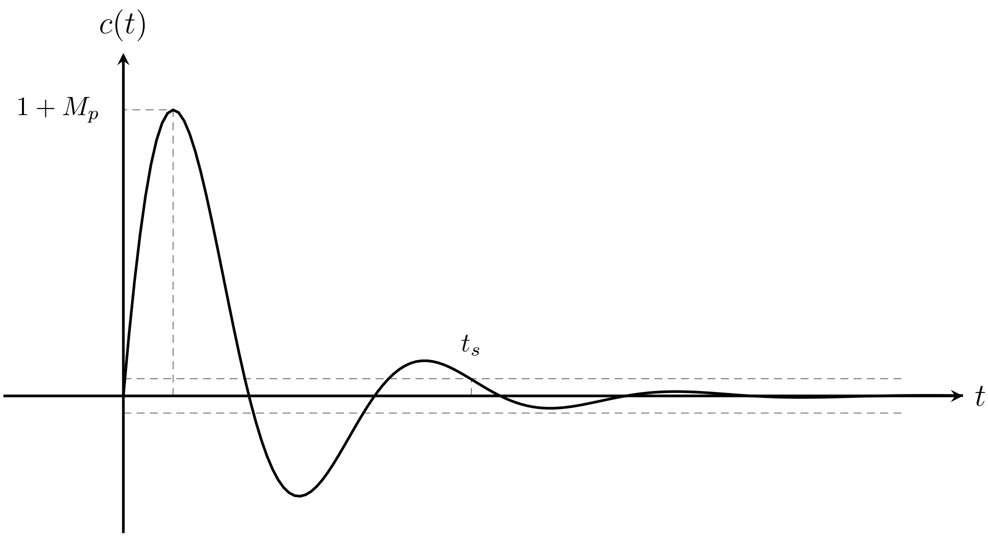


Figura 1 – Reposta ao impulso de um sistema de segunda ordem.

2.1 Regiões de polos para sistemas contínuos

Na $\mathscr{D}$-estabilidade, a região referente à estabilidade em sistema contínuos e invariantes no tempo é o semi-plano esquerdo do plano complexo. Dado um ponto genérico no plano s, representado por:

$$\mathrm{s} = x + \jmath y \tag{2.1}$$

este estará na região estável somente se a parte real de tal ponto estiver à esquerda do eixo imaginário, ou em números:

$$\mathrm{Re}\,(\mathrm{s}) < 0 \implies x < 0 \tag{2.2}$$

Assim, um sistema com n polos é dito estável se todos os seus polos estão localizados à esquerda do eixo imaginário. A partir deste conceito, é possível definir estabilidade relativa: se um sistema é estável para um valor $\sigma < 0$, então aquele é dito estável relativo (ao valor de σ).

A figura 2a mostra um esboço da região comentada. À medida que o valor de σ aumenta em valor absoluto, mais a esquerda a reta limitante se encontra e menor o plano estável relativo se torna. Além disso, as equações de tais retas podem ser generalizadas via:

$$x = -|\sigma| \tag{2.3}$$

Outros parâmetros de desempenho importantes para projetos de compensadores são o fator de amortecimento ζ e a frequência natural não-amortecida ω_n. São caracterizados pela resposta de sistemas de segunda ordem à função degrau (**??**)(**??**) e representam as oscilações não-amortecidas do modelo físico. Dado um par de polos conjugados via (2.1), é possível reescrevê-los em termos daqueles parâmetros:

$$s = -\zeta\omega_n \pm \jmath\omega_n\sqrt{1-\zeta^2} \tag{2.4}$$

com $\sigma = -\zeta\omega_n$. As regiões de $\mathscr{D}$-estabilidade referente a tais parâmetros são obtidos fixando um deles em (2.4) e variando o outro em um certo intervalo. Por esse motivo, (2.4) pode ser entendido como uma função de duas varáveis, dado como:

$$s(\zeta, \omega_n) = -\zeta\omega_n \pm \jmath\omega_n\sqrt{1-\zeta^2} \tag{2.5}$$

Com isso, é possível analisar as regiões geradas a partir de tais parâmentros. A região ζ-constante é obtida fixando-se um valor para ζ e variando-se o valor de ω_n. As características obtidas são representadas na figura 2b. Os ângulos formados entre as retas e o eixo imaginário têm valores absolutos $\beta = -\operatorname{cotg}(\arccos(\zeta))$ e diminuem à medida que o valor de ζ aumenta, tornando a região estreita.

Já as regiões ω_n-constante possuem as características esboçadas na figura 2c. Os raios das semi-circunferências formadas possuem valores iguais à ω_n e aumentam ou diminuem à medida que se varia tal parâmetro. Conforme abordado em (**??**), a intersecção das regiões comentadas formam a Região Ω de Desempenho garantido. Todos os polos dentro de tal região possuem um mínimo valor de σ, ζ e ω_n.

2.2 Regiões de polos para sistemas discretos

As regiões de $\mathscr{D}$-estabilidade para sistemas discretos são obtidas seguindo os mesmos métodos abordados nos contínuos, com a diferença de serem descritos no plano z. A transformada de um ponto do plano s para z é dada por:

$$\mathrm{z} = \exp(\mathrm{s}\, T_s) \tag{2.6}$$

onde T_s é o período de amostragem, parâmetro importante para sistemas discretos (**??**). Substituindo (2.4) em (2.6), chega-se a seguinte relação:

$$\mathrm{z} = \exp\left(-\zeta\omega_n T_s \pm j\omega_n T_s\sqrt{1-\zeta^2}\right) \tag{2.7}$$

A σ-estabilidade nos contínuos foi encontrada verificando a parte real dos polos. Utilizando-se da mesma ideia, ao analisar apenas a parte real de (2.7) (igualando a parte imaginária igual a zero), chega-se na seguinte relação:

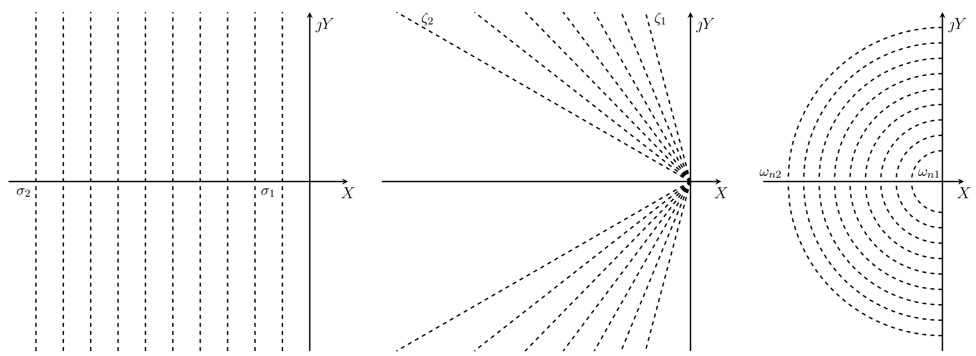
$$\mathrm{z} = \exp\left(-|\sigma| T_s\right) \tag{2.8}$$

onde $\sigma = -\zeta\omega_n$. Tal função descreve uma circunferência com raio $r = |\sigma|$ no plano complexo. Recordando (2.2) e (2.3), quando $\sigma = 0$ em (2.8), a circunferência gerada possui raio unitário. Dessa maneira, o região estável nos sistemas discretos é o interior de uma circunferência unitária. A figura 3a mostra esboços para vários valores de σ.

Assim como realizado nos sistemas contínuos, (2.7) pode ser enxergada em função de ζ e ω_n:

$$\mathrm{z}(\zeta, \omega_n) = \exp\left(-\zeta\omega_n T_s \pm j\omega_n T_s\sqrt{1-\zeta^2}\right) \tag{2.9}$$

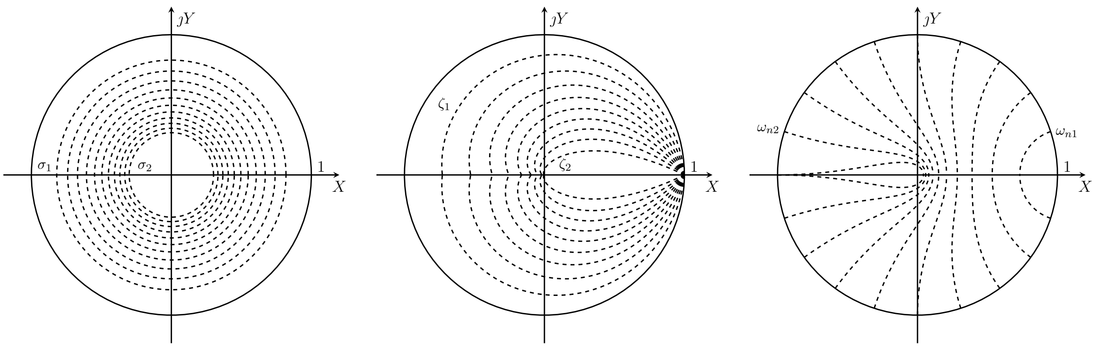
e a partir desta, é possível descrever as regiões geradas a partir de tais parâmentros. A região ζ-constante possui o formato aprensentado na figura 3a. Devido ao exponencial,



(a) (b) (c)

Figura 2 – Regiões de $\mathscr{D}$-estabilidade do plano s. Em (a) encontram-se retas verticais para vários valores de σ, sendo $|\sigma_2| > |\sigma_1|$. Em (b) encontram-se retas para vários valores de ζ, sendo $\zeta_2 > \zeta_1$. Em (c) encontram-se circunferências de raios $r = \omega_n$, sendo $\omega_{n2} > \omega_{n1}$.

as curvas geradas assemelham-se a cardioides, mas não o são, pois denominam-se espirais logarítmicas.



(a) (b) (c)

Figura 3 – Regiões de $\mathscr{D}$-estabilidade do plano z. Em (a) encontram-se circunferências com valores de raios crescentes, sendo $|\sigma_2| > |\sigma_1|$. Em (b) encontram regiões ζ-constantes com áreas decrescentes em relação à ζ, sendo $\zeta_2 > \zeta_1$. Em (c) encontram-se regiões semelhantes à cardioides que possuem áreas crescentes em relação à ω_n, sendo $\omega_2 > \omega_1$.

Ambos os ramos começam a ser desenhadas a partir de $(1, 0)$ (quando $\omega_n = 0 = \omega_{nmin}$), e se deslocam no sentido anti-horário, até cruzarem o eixo real primeira vez. À medida que o valor de ω_n aumenta, mais voltas o contorno dá. E a cada n meia-volta, o contorno cruza o eixo real pela n-ésima vez, conforme esboçado na figura 4. Como a espiral tende para dentro da região limitada pela primeira volta, somente a primeira volta é considerada no plano z.

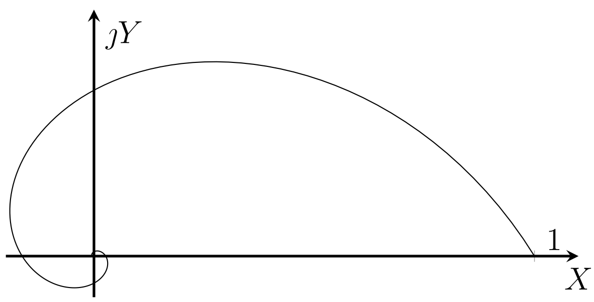


Figura 4 – Espiral logarítmica com 3 meia-voltas gerada a partir de (2.9) com $\zeta = 0.5$ constante.

O valor de ω_n no qual os ramos cruzam o eixo imaginário a cada n meia-volta é encontrado quando o argumento de (2.9) é igual à π (meia volta da espiral, em radianos), isto é:

$$\arg\left(\mathrm{z}(\zeta, \omega_{nmax})\right) = \pi \implies \omega_{nmax} = \frac{\pi}{T_s\sqrt{1-\zeta^2}} \tag{2.10}$$

Assim, ambos os ramos cruzam o eixo imaginário pela primeira vez quando $\omega_n = \omega_{nmax}$, para o respectivo valor fixado de ζ. Ainda, outra característica que pode ser citada é a influência do ζ na região: quanto maior seu valor, menor a área da região ζ-constante equivalente, assim como ocorre com seu dual nos contínuos.

Em relação às regiões ω_n-constante, como o estudo de alocação de polos se restringe a sistemas se segunda ordem subamortecidos (**??**) (**??**), os valores possíveis para a taxa de amortercimento está no intervalo $0 < \zeta < 1$. Dito isso, os ramos esboçados na figura 3c começam a ser desenhadas a partir da circunferência unitária, onde o ponto é dado por z(ζ_{min}, ω_n) (com $\zeta = 0 = \zeta_{min}$) e vão em direção ao ponto z(ζ_{max}, ω_n), com $\zeta_{max} = 1$.

Com tais pontos extremos, é possível aproximar as regiões do plano z utilizando Desigualdes Matriciais Lineares (LMIs, em inglês). Esse estudo será abordado na subseção a seguir.

2.3 Aproximação das regiões do plano z via LMIs

Em estudos anteriores, foram abordadas técnicas utilizando LMIs para mapear as regiões de $\mathscr{D}$-estabilidade no plano s. Tal feito foi realizado devido à convexidade de tais regiões, requisito para o uso de LMIs. Conforme visto na subseção 2.2, as regiões ζ-constante e ω_n-constante no plano z podem possuir características não-convexas, o que impossibilita o mapeamento exato via LMIs.

Estudos foram desenvolvidos para contornar a não-convexidade de algumas regiões do plano z, aproximando-os em regiões convexas. Em 2014, no artigo (**??**), a autora mapeou a região ζ-constante utilizando a maior elipse ou circunferência inscrita possível. Mas foi em (**??**) que foi desenvolvido um algoritmo que traz várias aproximações utilizando elipses, para aproveitar da melhor forma a área daquela região.

Já em (**??**) foi abordada uma aproximação cônica, utilzando-se apenas de quatro pontos e, consequentemente, dois setores cônicos. Apesar de simples, a ideia poderia facilmente ser estendida para n pontos, o que foi feito em (**??**). Ao aumentar a área a cada iteração, o algoritmo verifica a solução proposta.

E finalmente, utilizando-se das ideias anteriores, (**??**) trouxe aproximações cônica, elíptica e poligonal da região ω_n-constante.

2.4 Modelagem via espaço de estados

O espaço de estados é uma representação matemática desenvolvida na Teoria de Controle para estudar sistemas físicos, relacionando saídas, entradas e variáveis de estado internas, através do uso de Equações Diferenciais Ordinárias (EDOs).

Se o sistema a ser analisado for contínuo, linear e invariante no tempo, então as variáveis de estado deste podem ser representados através de vetores, enquanto que as

EDOs correspondentes são descritas de forma matricial, como segue:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{2.11a}$$

$$\mathbf{y} = C\mathbf{x} + D\mathbf{u} \tag{2.11b}$$

onde:

- $\dot{\mathbf{x}}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ e $\mathbf{y} \in \mathbb{R}^p$;
- $A \in \mathbb{R}^{n \times n}$ é a matriz de estado;
- $B \in \mathbb{R}^{n \times m}$ é a matriz de entrada;
- $C \in \mathbb{R}^{p \times n}$ é a matriz de saída;
- $D \in \mathbb{R}^{p \times m}$ é a matriz de transmissão direta.

Já a representação no espaço de estados de um sistema discreto, linear e invariante no tempo é:

$$\mathbf{x}^{+} = A\mathbf{x} + B\mathbf{u} \tag{2.12a}$$

$$\mathbf{y} = C\mathbf{x} + D\mathbf{u} \tag{2.12b}$$

Em ambos os domínios, os polos do sistema são os autovalores da matriz de estado. Logo, um sistema representado via espaço de estados é dito estável se todos os polos da matriz A estão no semi-plano esquerdo ou no interior de uma circunferência unitária.

2.5 Condição de Liapunov

Seja $\mathcal{D}$ um subconjunto do plano contínuo, localizado no semi-plano esquerdo e um sistema representado como em (2.11) ou (2.12). Este conjunto é dito $\mathcal{D}$-estável se todos os autovalores da matriz de estado correspondente estiverem contidos em $\mathcal{D}$.

Caso $\mathcal{D}$ for uma região convexa, é possível mapeá-la utilizando Desigualdes Matriciais Lineares (LMIs em ingîes) (**??**), através da condição de Liapunov. A partir disso, uma matriz A é estável se, e somente se, existir uma matriz simétrica P, tal que:

$$AP + PA' \prec 0, \qquad P \succ 0 \tag{2.13}$$

onde o "$P \succ 0$"lê-se P é positiva definida.

2.6 Regiões LMIs de interesse

Definidos LMIs, algumas regiões LMIs são apresentadas neste trabalho para as aproximações estudadas. Uma primeira região que pode ser citada é todo o interior de uma circunferência de raio r, dado no plano z:

$$\begin{bmatrix} -rP & * \\ PA + Z'B & -rP \end{bmatrix} \prec 0 \tag{2.14}$$

com $r = \exp(-|\sigma| T_s)$. Outra região de interesse é um setor cônico com centro em α e ângulo θ. Tal plano é simétrico em relação ao seu centro. Portando, o mapeado LMI leva em consideração tal simetria. Contudo, através de manipulações algébricas, a região LMI de um setor cônico voltado para a esquerda é definida como:

$$\begin{bmatrix} \text{sen}\,(\theta)(\mathcal{V}\,(AP) + \mathcal{V}\,(BZ) - 2\alpha P) & * \\ \cos(\theta)(\bar{\mathcal{V}}\,(PA') + \bar{\mathcal{V}}\,(Z'B')) & \text{sen}\,(\theta)(\mathcal{V}\,(AP) + \mathcal{V}\,(BZ) - 2\alpha P) \end{bmatrix} \prec 0 \tag{2.15}$$

com seu lado direito dado como:

$$\begin{bmatrix} \text{sen}\,(\theta)(2\alpha P - \mathcal{V}\,(AP) - \mathcal{V}\,BZ) & * \\ \cos(\theta)(\bar{\mathcal{V}}\,(PA') + \bar{\mathcal{V}}\,Z'B') & \text{sen}\,(\theta)(2\alpha P - \mathcal{V}\,(AP) - \mathcal{V}\,(BZ)) \end{bmatrix} \prec 0 \tag{2.16}$$

Para impedir a simetria do setor cônico direito, é definido uma reta limitante em α, dado por:

$$AP + BZ + Z'B' + PA' - 2\alpha P \succ 0 \tag{2.17}$$

onde considera a área a direita de tal reta. Isso é necessário apenas para o setor cônico direito, pois o voltado para a esquerda é limitado por (2.14), como pode ser observado na figura 3. Uma última região LMI é apresentada: uma elipse, que possui a seguinte equação (**??**):

$$\mathbb{E}: \frac{(x-1)^2}{\left(1 - \exp\left(-\frac{2\pi}{N_y}\right)\right)^2} + \frac{\left(\left(1 - \exp\left(-\frac{2\pi}{N_y}\right)\right)^2 - \left(\cos\left(\frac{2\pi}{N_y}\right) - 1\right)^2\right) y^2}{\left(1 - \exp\left(-\frac{2\pi}{N_y}\right)\right)^2 \text{sen}\left(\frac{2\pi}{N_y}\right)} = 1 \tag{2.18}$$

onde N_y é relação entre a frequência de amostragem e a frequência natural não-amortecida, dada por:

$$N_y = \frac{\omega_s}{\omega_n} = \frac{2\pi}{\omega_n T_s} \tag{2.19}$$

A LMI correspondente é dada por:

$$
\begin{bmatrix} -P & * \\ -\frac{1}{a}P + \frac{1}{2}\left(\frac{1}{a}+\frac{1}{b}\right)AP + \frac{1}{2}\left(\frac{1}{a}-\frac{1}{b}\right)PA' & -P \end{bmatrix} \prec 0 \tag{2.20}
$$

onde:

$$
a = \left(1 - \exp\left(\frac{-2\pi}{N_y}\right)\right) \tag{2.21a}
$$

$$
b = a \operatorname{sen}\left(\frac{2\pi}{Ny}\right) \tag{2.21b}
$$

3 Algoritmo

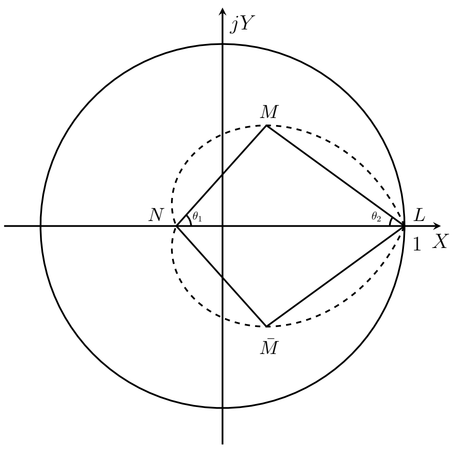
O algoritmo apresentado neste trabalho é um compilado de algoritmos desenvolvidos anteriormente, utilizando-se das aproximações cônica, elíptica e poligonal das regiões de $\mathscr{D}$-estabilidade do plano z. O objetivo deste trabalho é desenvolver em *software* tais algoritmos e, ao informar parâmetros de projeto, determinar se é possível implementar um compensador que respeite os requisitos.

Para tal, o algoritmo pode ser divido em três partes, uma para cada aproximação, sendo a aproximação desejada escolhida via chamada da função. O *software* utilizado foi o MATLAB, juntamente com o interpretador de LMIs YALMIP em conjunto com o solucionador númerico MOSEK.

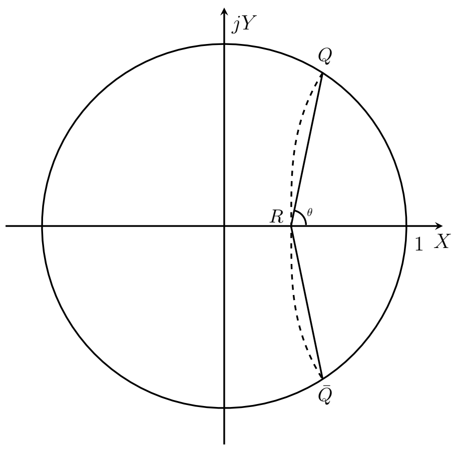
3.1 Aproximação cônica

Para o mapeamento cônico das curvas ζ-constante e ω_n-constante, são utilizados os setores cônicos determinados via (2.16) e (2.15), e retas verticais como apresentado em 2.17.

Para a primeira curva, a ideia consiste em utilizar os pontos extremos calculados na seção 2.2, onde serão os centros dos setores cônicos. Os ângulos, medidos no sentido anti-horário, são determinados a partir de um terceiro ponto, conforme a figura 5a. A escolha do ponto M é feita de maneira que a área do triângulo $\widehat{LMN}$ seja a maior possível. Um algoritmo linear foi usado para encontrar este ponto.



(a)



(b)

Figura 5 – Esboços da aproximação cônica das regiões ζ e ω_n-constantes.

Uma que vez conhecidas tais informações, é possível aplicar o algoritmo 1. Um setor cônico voltado para a direita, com centro em N e ângulo θ_1, e outro voltado para a esquerda, com centro em L e ângulo θ_2, são aplicados para a aproximação inicial. Além

Algoritmo 1 Aproximação cônica da taxa de amortecimento

Entrada: σ, ζ, T_s
Saída: K
1: $L \leftarrow \mathrm{z}(\zeta, \omega_{nmin})$
2: $N \leftarrow \mathrm{z}(\zeta, \omega_{nmax})$
3: $M \leftarrow \mathrm{z}(\zeta, \omega_n)$, onde a área do triângulo formado é a maior possível
4: $F \leftarrow P \succ 0$
5: $F \leftarrow F \cap$ (2.14) com $r = \exp(-|\sigma| T_s)$ ▷ Taxa de amortecimento
6: $F \leftarrow F \cap$ (2.15) com $\alpha = L$ e $\theta = \mathrm{ang}(M, N)$ ▷ Setor cônico esquerdo
7: $F \leftarrow F \cap$ (2.16) com $\alpha = N$ e $\theta = \mathrm{ang}(L, M)$ ▷ Setor cônico direito
8: $F \leftarrow F \cap$ (2.17) com $\alpha = N$ ▷ Reta vertical
9: Verificar se o problema é factível
10: $K \leftarrow ZP^{-1}$

disso, para limitar a simetria do setor cônico com centro em N, uma reta que passa por este ponto é aplicada.

A região de $\mathscr{D}$-estabilidade resultante é a intersecção das regiões descritas. Ao ser unida com a restrição da taxa de decaimento, o setor cônico com centro em L é limitado por esta região. Após finalizado, o algoritmo determina a factibilidade da solução encontrada e retorna a matriz K que estabiliza o sistema com os parâmetros de projeto informados.

Em relação à região ω_n-constante, a mesma ideia é aplicada (**??**). Contudo, neste caso, somente um setor cônico com centro em R, que é limitado pela direita por uma reta que passa neste ponto são usados, conforme a figura 5b. Os pontos Q e R são determinados via (2.9), com $\zeta = \zeta_{min}$ e $\zeta = \zeta_{max}$, respectivamente. Além disso, o ângulo θ é determinado através de $\mathrm{ang}(Q, R)$.

Após determinadas essas informações, é possível utilizar o algoritmo 2. Um detalhe que é facilmente observado é a rápida perda de convexidade da curva N_y. Logo, caso a constante N_y seja menor que 4.86 (**??**), o algoritmo retorna um alerta devido informando à falta de convexidade. Assim, para fins práticos, a pouca e a falta de convexidade de tais curvas não foram tratadas.

Algoritmo 2 Aproximação cônica da curva N_y

Entrada: σ, ω_n
Saída: K
1: $Q \leftarrow \mathrm{z}(\zeta_{min}, \omega_n)$
2: $R \leftarrow \mathrm{z}(\zeta_{max}, \omega_n)$
3: $F \leftarrow P \succ 0$
4: $F \leftarrow F \cap$ (2.14) com $r = \exp(-|\sigma| T_s)$ ▷ Taxa de amortecimento
5: $F \leftarrow F \cap$ (2.16) com $\alpha = R$ e $\theta = \mathrm{ang}(Q, R)$ ▷ Setor cônico direito
6: $F \leftarrow F \cap$ (2.17) com $\alpha = R$ ▷ Reta vertical
7: Verificar se o problema é factível
8: $K \leftarrow ZP^{-1}$

3.2 Aproximação elíptica

Para a aproximação elíptica, apenas a região ω_n-constante foi aproximada. A ideia consiste em encontrar a maior elipse inscrita, a fim de aproveitar melhor a área. A figura 6 mostra um esboço da ideia descrita. Para tal, é preciso verificar se o valor escolhido para ω_n e T_s resultem em uma área convexa (**??**), através da constante N_y. Caso os parâmetros informados atendam às restrições, o algoritmo 3 pode ser aplicado.

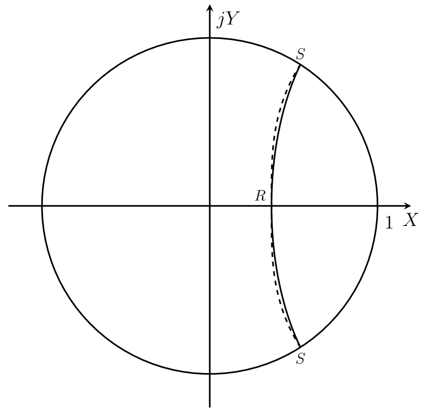


Figura 6 – Aproximação elíptica da região ω_n-constante.

Algoritmo 3 Aproximação elíptica da curva N_y

Entrada: σ, T_s, N_y
Saída: K
1: $F \leftarrow P \succ 0$
2: $F \leftarrow F \cap (2.14)$, com $r = \exp(-|\sigma| T_s)$ ▷ Taxa de amortecimento
3: $F \leftarrow (2.20)$, com $a = (2.21a)$ e $b = (2.21b)$ ▷ Elipse
4: Verificar se o problema é factível
5: $K \leftarrow ZP^{-1}$

3.3 Aproximação poligonal

A aproximação poligonal consiste na ideia de aproximar as regiões de interesse em um polígono com o maior número de lados possível. Para isto, o algoritmo irá partir de uma aproximação cônica simples. A partir daí, entre os dois pontos usados para definir o setor, um ponto intermediário é calculado e dois novos setores cônicos são definidos. Sob a ótica do número de lados, a cada iteração, um novo lado é acrescentado e, consequentemente, a área é incrementada. Em um número grande de iterações, a região aproximada tende a área total.

Para a região ζ-constante, um setor cônico voltado para esquerda e centro em L é usado como aproximação inicial (**??**). Contudo, devido à cúspide daquela, uma reta em N é usada para eliminar tal convexidade. Dito isso, surge a necessidade de calcular o ponto M, localizado entre os pontos máximo e mínimo, onde possui a mesma parte real que N, conforme a figura 7a.

Algoritmo 4 Aproximação poligonal da região ζ-constante

Entrada: σ, ζ, T_s
Saída: K

$l \leftarrow 0$
$L \leftarrow \mathrm{z}(\zeta, \omega_{nmin})$
$N \leftarrow \mathrm{z}(\zeta, \omega_{nmax})$
$M \leftarrow \mathrm{z}(\zeta, \omega_n)$ tal que $\mathrm{Re}\,(M) = N$
$pts1 \leftarrow [0\ \omega_{ne}]$
$pts2 \leftarrow pts1$
$vec1 \leftarrow [L\ M]$
$vec2 \leftarrow vec1$
$F \leftarrow P \succ 0$
$F \leftarrow$ (2.14) com $r = \exp(-|\sigma| T_s)$ ▷ Taxa de amortecimento
$F \leftarrow F \cap$ (2.15), com $\alpha = L$ e $\theta = \mathrm{ang}(L, M)$ ▷ Voltado para a esquerda
$F \leftarrow F \cap$ (2.17), com $\alpha = N$ ▷ Reta vertical
Verificar se o problema é factível
enquanto Problema for infactível **faça**
  se $l <$ número de elementos em $vec1 - 1$ **então**
    $l \leftarrow l + 1$
  senão
    $l \leftarrow 1$
    $vec1 \leftarrow vec2$
    $pts1 \leftarrow pts2$
  fim se
  $F \leftarrow \emptyset$ ▷ Descarta as restrições anteriores
  $F \leftarrow P \succ 0$
  $F \leftarrow$ (2.14) com $r = \exp(-|\sigma| T_s)$ ▷ Taxa de amortecimento
  $pt_{new1} \leftarrow (pts1(l) + pts1(l+1))/2$
  $V_{new1} \leftarrow \mathrm{z}(\zeta, pt_{new1})$
  $pts2 \leftarrow [pts2\ pt_{new1}]$
  Orderna de forma decrescente $pts2$
  $vec2 \leftarrow [vec2\ V_{new1}]$
  Orderna de forma decrescente $vec2$
  $F \leftarrow F \cap$ (2.17), com $\alpha = N$ ▷ Reta vertical
  para $m = 1$ até número de elementos de $vec1 - 1$ **faça**
    $u_1 \leftarrow \mathrm{loc}(vec2(m), vec2(m+1))$
    se $u_1 < 0$ **então**
      $F \leftarrow F \cap$ (2.16), com $\alpha = u_1$ e $\theta = \mathrm{ang}(vec2(m+1), u_1)$ ▷ Voltado para a direita
    senão
      $F \leftarrow F \cap$ (2.15), com $\alpha = u_1$ e $\theta = \mathrm{ang}(vec2(m+1), u_1)$ ▷ Voltado para a esquerda
    fim se
  fim para
  Verificar se o problema é factível
fim enquanto
$K \leftarrow ZP^{-1}$

Para tal, utiliza-se o cálculo numérico para encontrar uma solução aproximada. Com tal ponto calculado e a não convexidade da cúspide tratada, é possível utilizar o algoritmo 4. Os vértices iniciais L e N do ramo são guardados em um vetor de vértices. Em paralelo a isso, os valores de ω_n que geram tais pontos também são armazenados. A cada iteração, o algoritmo calcula o ponto médio entre dois vértices consecutivos em cada vetor, já que a realização do primeiro depende do segundo.

Contudo, antes do algoritmo voltar para o início dos vetores, é preciso percorrer todos

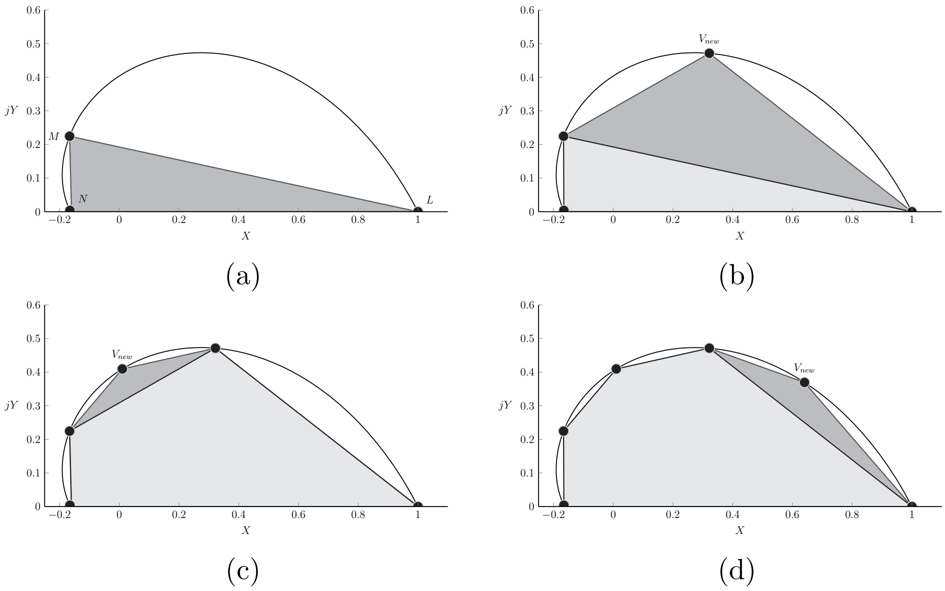


Figura 7 – As primeiras quatro aproximações poligonais da curva ζ-constante.

os pontos do vetor da iteração anterior[1]. Para isso, cópias dos vetores de vértices e de pontos são inicializados, a fim de controlarem tal fluxo. Assim, quando o algoritmo terminar de percorrer o "vetor anterior", tal conjunto é atualizado com os novos pontos e vértices calculados ao final deste processo.

Em relação ao cálculo dos pontos intermediários, a inclinação da reta que passa pelos pontos extremos locais pode ser positiva ou negativa. O uso da função loc se torna enssencial, pois caso o ponto resultante for menor que zero, a reta que passa pelos pontos possui inclinação positiva, e negativa caso contrário (**??**). Assim, os setores cônicos gerados seguem a orientação desta, com centro naquele ponto calculado.

A figura 7 mostra as quatro primeira iterações do algoritmo 4. É possível observar o sentido horário do fluxo para o cálculo dos vértices intermediários. Também é notória a rápida abrangência da região ζ-constante.

Para a região ω_n-constante, a ideia é similar. A aproxmição inicial utiliza-se de apenas um setor cônico voltado para a direita com centro em $R = \mathrm{z}(\zeta_{max}, \omega_n)$. O ângulo θ em (2.16) é definido a partir do ângulo entre a reta $\overline{QR}$ e o eixo real, onde $Q = \mathrm{z}(\zeta_{min}, \omega_n)$.

Caso esta região não seja factível, basta calcular o ponto intermediário entre Q e R e definir dois novos setores cônicos, a fim de aumentar a área. Novamente, para o novo ponto calculado, é utilizado a função loc para determinar o centro do setor cônico correspondente e, em seguida, usa-se ang para determinar seu ângulo. Assim, a cada iteração, o algoritmo adiciona dois novos setores cônicos e os intersecta com as regiões previamente definidas. A figura 8 mostra as quatro primeiras iterações do algoritmo 5.

Assim como ocorre com a aproximação poligonal da região ζ-constante, há uma rápida

[1] Como os vetores são atualizados com os novos elementos, o tamanho daqueles aumenta ($(n-1)$ pontos são adicionados a cada varredura, com n a quantidade de pontos do vetor) durante a iteração. Assim, o execução termina antes de atingir o final do vetor.

Algoritmo 5 Aproximação poligonal da região ω_n-constante

Entrada: σ, ω_n, T_s
Saída: K

$l \leftarrow 0$
$Q \leftarrow \mathrm{z}(\zeta_{min}, \omega_n)$
$R \leftarrow \mathrm{z}(\zeta_{max}, \omega_n)$
$pts3 \leftarrow [\zeta_{min}\ \zeta_{max}]$
$pts4 \leftarrow pts3$
$vec3 \leftarrow [R\ Q]$
$vec4 \leftarrow vec3$
$F \leftarrow P \succ 0$
$F \leftarrow$ (2.14) com $r = \exp(-|\sigma|T_s)$ ▷ Taxa de amortecimento
$F \leftarrow F \cap$ (2.16), com $\alpha = R$ e $\theta = \mathrm{ang}(Q, R)$ ▷ Voltado para a direita
$F \leftarrow F \cap$ (2.17), com $\alpha = R$ ▷ Reta vertical
Verificar se o problema é factível
enquanto Problema for infactível **faça**
  se $l <$ número de elementos em $vec3 - 1$ **então**
    $l \leftarrow l + 1$
  senão
    $l \leftarrow 1$
    $vec3 \leftarrow vec4$
    $pts1 \leftarrow pts2$
  fim se
  $F \leftarrow \emptyset$ ▷ Descarta as restrições anteriores
  $F \leftarrow P \succ 0$
  $F \leftarrow$ (2.14) com $r = \exp(-|\sigma|T_s)$ ▷ Taxa de amortecimento
  $pt_{new2} \leftarrow (pts3(l) + pts3(l+1))/2$
  $V_{new2} \leftarrow \mathrm{z}(pt_{new2}, \omega_n)$
  $pts4 \leftarrow [pts4\ pt_{new2}]$
  Orderna de forma decrescente $pts4$
  $vec4 \leftarrow [vec4\ V_{new2}]$
  Orderna de forma decrescente $vec4$
  $F \leftarrow F \cap$ (2.17), com $\alpha = N$ ▷ Reta vertical
  para $m = 1$ até número de elementos de $vec3 - 1$ **faça**
    $u_2 \leftarrow \mathrm{loc}(vec4(m), vec4(m+1))$
    $F \leftarrow F \cap$ (2.16), com $\alpha = u_2$ e $\theta = \mathrm{ang}(vec4(m), u_2)$
  fim para
  Verificar se o problema é factível
fim enquanto
$K \leftarrow ZP^{-1}$

cobertura da região. Com poucas iterações, a área é quase totalmente coberta.

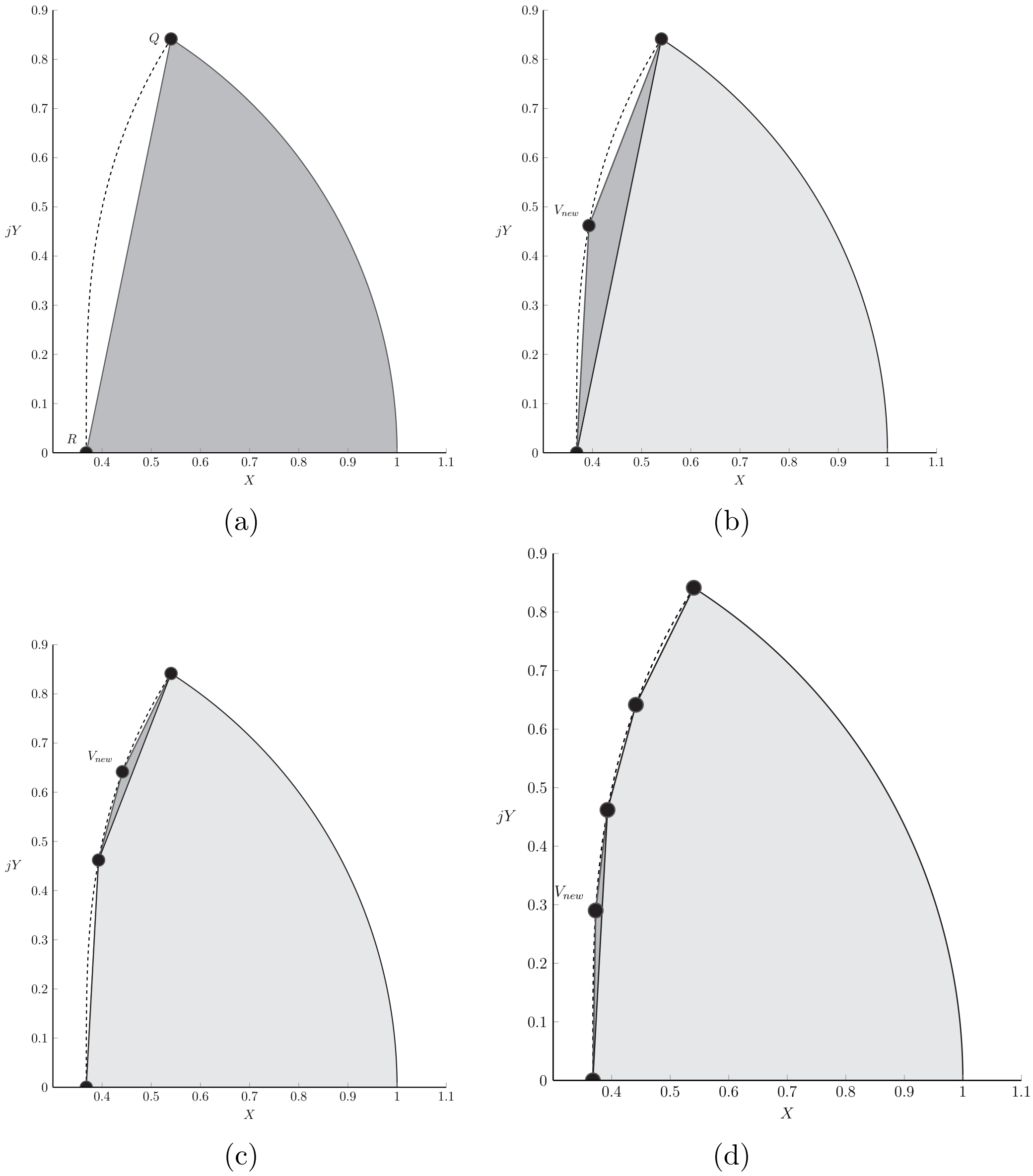


Figura 8 – As primeiras quatro aproximações poligonais da curva ω_n-constante.

4 Testes e Simulações

Para ilustrar o funcionamento dos algoritmos, foi proposta uma planta hidráulica composta por um sistema de tanques comunicantes. A figura 9 mostra um esboço de tal sistema. O tanque com capactância C_1 é interligado com um de capacitância C_2. Aquele é alimentado por uma vazão q e é drenado por uma vazão q_1. Tal grandeza é controlada por um registro, que pode ser enxergado como um resistor de resistência R_1.

Ainda, devido à ligação, a vazão de saída do primeiro tanque é a entrada do segundo. Este é drenado por uma vazão q_2, onde é controlado por um registro R_2. A variável controlada é a diferença $h_1 - h_2$.

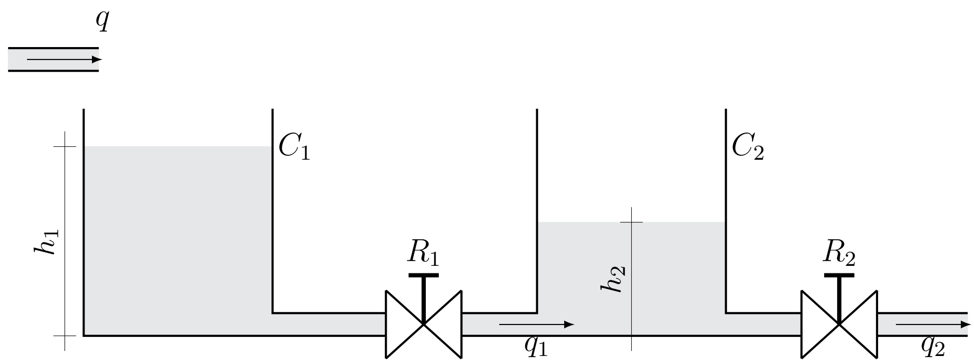


Figura 9 – Tanques comunicantes.

4.1 Modelagem via espaço de estados

Para a representação via espaço de estados, define-se as variáveis de estado $x_1 = q_1$ e $x_2 = q_2$. A partir das relações entre capacitância e vazão, chega-se a seguinte representação no espaço de estados:

$$\dot{\mathbf{x}} = \begin{bmatrix} -(R_1C_{eq})^{-1} & (R_1C_2)^{-1} \\ (R_2C_2)^{-1} & -(R_2C_2)^{-1} \end{bmatrix} \mathbf{x} + \begin{bmatrix} (R_1C_1)^{-1} \\ 0 \end{bmatrix} \mathbf{u} \tag{4.1a}$$

$$\mathbf{y} = \begin{bmatrix} R_1 & 0 \end{bmatrix} \mathbf{x} \tag{4.1b}$$

onde $C_{eq} = C_1C_2/(C_1 + C_2)$. Para discretizar o sistema, é preciso de um valor para o período de amostragem T_s. A transformada usada será a bilinear de Tustin, dada por:

$$\mathrm{s} = \frac{2}{T_s}\frac{\mathrm{z}-1}{\mathrm{z}+1} \tag{4.2}$$

onde o semi-plano esquerdo dos contínuos é mapeado no círculo unitário dos discretos.

4.2 Parâmetros de projeto

Com posse do espaço de estados, é possível sintetizar uma matriz de ganho K que possa estabilizar o sistema. Antes, é necessário atribuir valores para a planta. Em um primeiro projeto, serão escolhidas arbitrariamente tais valores, como segue:

- $C_1 = C_2 = 5$;
- $R_1 = R_2 = 1$;
- $T_s = 10\,\text{s}$.

Assim, a representação via espaço de estados da planta nos contínuos é:

$$\dot{\mathbf{x}} = \begin{bmatrix} -0.4 & -0.2 \\ 0.2 & -0.2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} \mathbf{u} \tag{4.3a}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x} \tag{4.3b}$$

Através do função `c2d` disponibilizada no MATLAB, a transformação bilinear é realizada, resultando em:

$$\mathbf{x}^{+} = \begin{bmatrix} -0.4286 & -0.2857 \\ 0.2857 & -0.1429 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0.5714 \\ 0.2857 \end{bmatrix} \mathbf{u} \tag{4.4a}$$

$$\mathbf{y} = \begin{bmatrix} 0.2857 & -0.1426 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0.2857 \end{bmatrix} \mathbf{u} \tag{4.4b}$$

A característica notória obtida é a presença da matriz D no sistema discretizado. É um consequência da transformação bilinear: surgimento de transmissão direta. Com posse das matrizes obtidas na representação via espaço de estados, é possível escolher os parâmetros de projeto:

- $t_s = 50\,\text{s}$;
- $\zeta = 0.5 \implies M_p \leq 0.16$;
- $\omega_n = 0.5\,\text{rad/s}$.

Neste caso, o raio da circunferência relativo à estabilidade possui valor igual à 0.4493. Ainda, a constante N_y possui o valor de 6.2832, acima do recomendado. Além disso, a maior frequencia natural não-amortecida de malha aberta do sistema discretizado possui o valor igual a $0.3780\,\text{rad/s}$.

Ao executar o algoritmo utilizando a aproximação cônica, a solução proposta é infactível. Ao checar a solução, apenas houve uma infactibilidade em relação à taxa de amortecimento. Tal fenômeno é comum em solucionadores numéricos, uma vez que podem admitir uma certa infactibilidade. A figura 10a mostra o diagrama de polos do

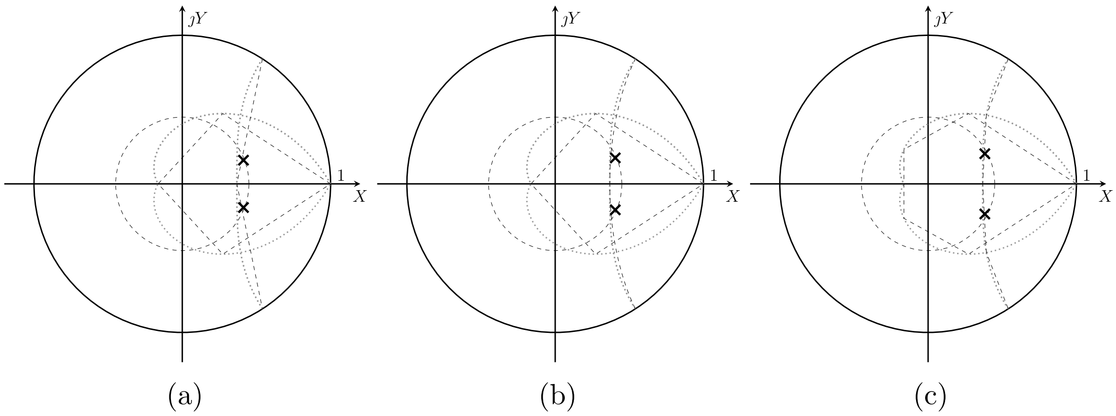


Figura 10 – Diagrama de polos do sistema compensado com a matriz de ganho K obtidos na (a)aproximação cônica, (b) elíptica e (c) poligonal.

sistema compensado. É possível notar que o sistema é estável, mesmo com a negativa do algoritmo.

O valor da matriz K que estabiliza o sistema é:

$$K = [2.5358 \quad -0.1842] \tag{4.5}$$

A resposta ao impulso para o sistema em malha fechada está representada na figura 11. O máximo sobressinal é de aproximadamente $5,21\%$, bem abaixo do requisito de projeto. Como os parâmentros de projeto foram escolhidas arbitrariamente, o tempo de acomodação exigido é maior do que o sistema em malha aberta. Mesmo assim, o algoritmo conseguiu respeitar o requisito.

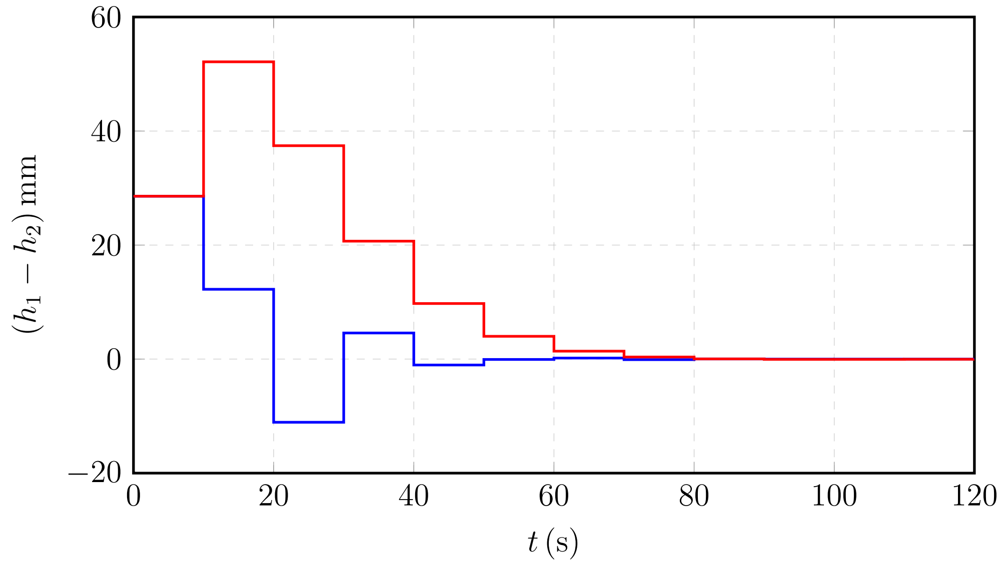


Figura 11 – Resposta ao impulso do sistema compensado a partir da aproximação cônica.

Para a aproximação elíptica, o algoritmo retornou uma solução factível. É esperado devido o melhor aproveitamento da região ω_n-constante. A figura 12 mostra o diagrama de polos do sistema compensando. A matriz de ganho K possui o seguinte valor:

$$K = [2.5076 \quad -0.1807] \tag{4.6}$$

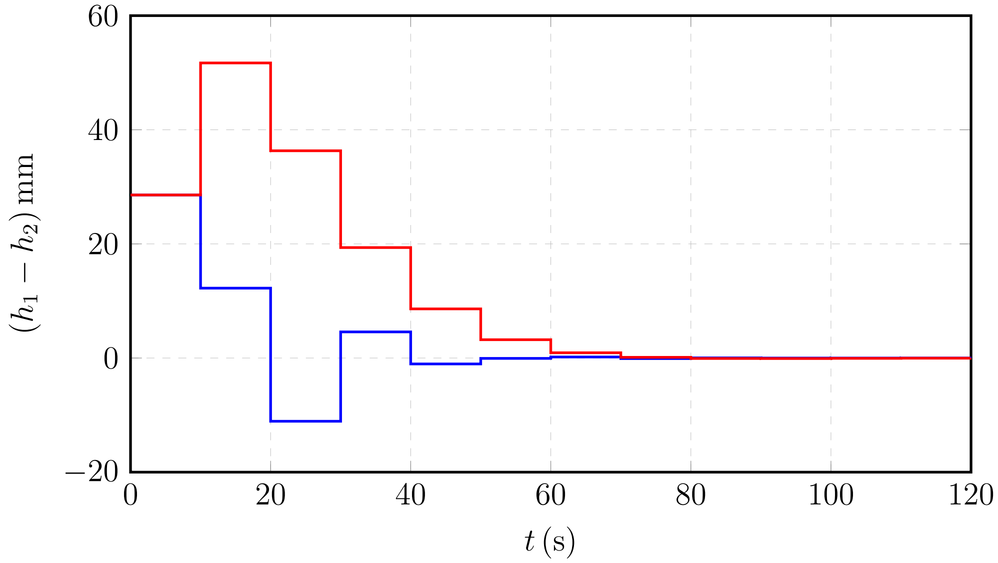


Figura 12 – Resposta ao impulso do sistema compensado a partir da aproximação elíptica.

Já para a aproximação polígonal, o algoritmo retornou uma solução factível com duas iterações. As regiões aproximadas foram representadas na figura 13 e a matriz de ganho K é dado por:

$$K = [2.4159 \quad -0.1554] \tag{4.7}$$

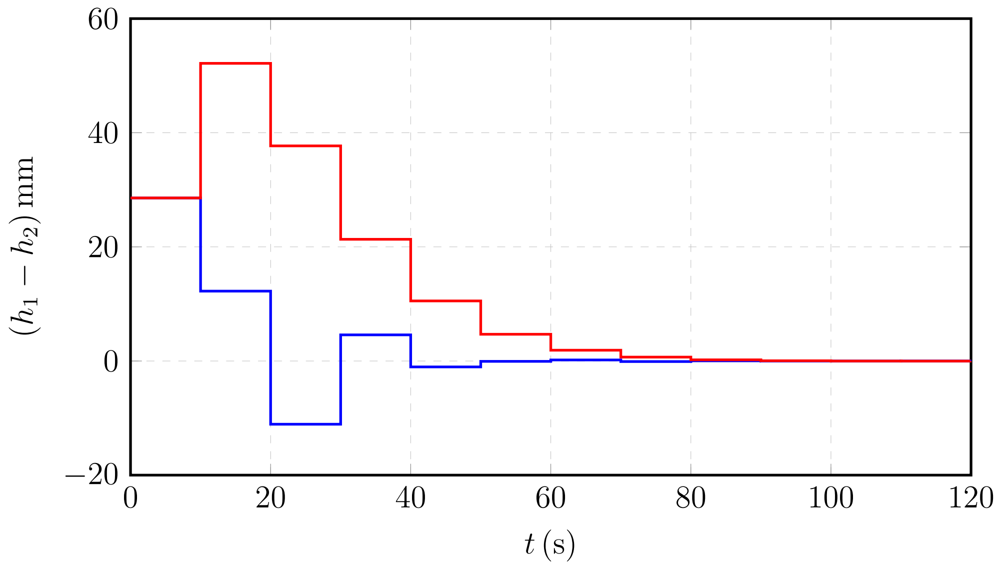


Figura 13 – Resposta ao impulso do sistema compensado a partir da aproximação poligonal.

5 Conclusão

A proposta de unir diferentes aproximações para regiões diferentes em uum único algoritmo foi cumprida. As principais dificuldades encontradas foram verificar a fatibilidade da solução, uma vez que solucionadores númericos podem aceitar infactibilidades no conjunto. Dessa maneira, o projeto que venha usar o algoritmo proposto deve ser conservaodr. Contudo, foi implementado a opção de verificar a solução proposta pelo interpretador, caso não seja factível. Caso os resíduos sejam ínfimos (na ordem de 10^{20}), é possível utilizar tal solução. Cabe ao usuário aceitar ou não a infactibilidade, nesses casos.

Ainda, como a resolução de LMIs foi feita utilizando programação semi-definida, a presença de uma função objetiva tornou dificultoso vários projetos. A função proposta foi o traço de P, onde minimiza os auto-valores da matriz de estado. Contudo, como a ideia do algoritmo é determinar um ponto de factibilidade, optou-se por não incluir a função objetiva. Mesmo assim, as soluções propostas, caso atendem ao parâmetros de projeto, se mostraram corretas.

Em relação às aproximações estudadas, foi possível notar que as cônicas, por serem muito simples, permitem soluções infactíveis. No caso do exemplo apresentado, foi possível tornar o sistema $\mathscr{D}$-estável, apesar do interpretador retornar uma negativa. Logo, apesar de simples e rápida, tal método é limitado por não aproveitar bem as regiões de interesse.

Já a aproximação elíptica se mostrou uma alternativa poderosa. As áreas são bem aproveitadas, o que reflete num projeto mais relaxado em relação à aproximação anterior. Como comentado no estudo, a grande dificuldade deste método está na computação da LMI correspondente.

Finalmente, as aproximações poligonais são as que melhor aproveitam as áreas de interesse. Em contrapartida, são mais lentos. Para projetos restritivos, podem levar segundos papra retornar uma solução. Ademais, nos estudos propostos, não foi determinada uma condição de desistência ao algoritmo. Neste trabalho, foi proposta a seguinte condição de parada: caso o incremento de área seja menor que 1%, interrompa o fluxo. Apesar de simples, a média tempo de execução para projetos que não são possíveis foi de 10.3 s.

Para trabalhos futuros, pode ser implementada a minimização da norma H_∞, pois há literatura disponível e como há computação nas dimensões das matrizes na representação via espaço de estados, é possível incluir tal restrição no algoritmo, tarefa não realizada devido ao prazo de entrega.

ANEXO A – Códigos em MATLAB desenvolvidos neste trabalho

Código I – Código em MATLAB que, dados dois pontos v_1 e v_2, determina o ponto que a reta que passa por estes cruza o eixo real.

```
function loc = loc(v1,v2)
  if v2 > v1
    loc = real(v1)-imag(v1)/((imag(v2)-imag(v1))/(real(v2)-real(v1)));
  else
    loc = real(v2)-imag(v2)/((imag(v1)-imag(v2))/(real(v1)-real(v2)));
  end
```

Código II – Código em MATLAB implementa (2.9).

```
function z = pontoplanoz(zeta,wn,Ts)
  % PONTOPLANOZ calcula pontos das curvas da taxa de amortecimento
  % (zeta) e da frequência natural no plano z.
  %
  % PONTOPLANOZ(ZETA,WN,TS) recebe como parâmetros zeta, a frequência
  % natural não-amortecida e o período de amostragem Ts.
  %
  % PONTOPLANOZ retorna o ponto z calculado por (6).
  %
  % Dado um ponto no plano s (contínuo) representado por:
  %     s = x + jy                                       (1)
  % a representação no plano z é dada pela transformação
  %     z = exp(sTs)                                     (2)
  % onde Ts é a taxa de amostragem. Substituindo (1) em (2),
  % obtém-se a seguinte relação:
  %     z = exp(xTs)*exp(j*y*Ts)                         (3)
  % Ainda, parte real pode ser representada como
  %     Re(s) = -zeta*wn                                 (4)
  % e a parte imaginária como:
  %     Im(s) = wn*sqrt(1-zeta^2)                        (5)
  % assim, (3) pode ser reescrito como:
  %     z = exp(-zeta*wn*Ts).*exp(j*wn*sqrt(1-zeta^2)*Ts)   (6)
  z = exp(-zeta.*wn.*Ts).*exp(1i.*wn.*sqrt(1-zeta.^2)*Ts);
```

Código III – Código em MATLAB que calcula o terceiro ponto de um triângulo onde possui a maior área.

```
function Q = determinarmaiorarea(Xa,Xb,zeta,Ts)
  fa = 0;
  fb = pi/(Ts*sqrt(1-zeta^2));
  fc = fb/2;

  while 1
    Q1 = pontoplanoz(zeta,(fc+fa)/2,Ts);
    Q2 = pontoplanoz(zeta,(fb+fc)/2,Ts);
```

```
    area1 = 1/2*abs(det([real(Xa),imag(Xa),1; ...
      real(Xb),imag(Xb),1; ...
      real(Q1),imag(Q1),1]));
    area2 = 1/2*abs(det([real(Xa),imag(Xa),1; ...
      real(Xb),imag(Xb),1; ...
      real(Q2),imag(Q2),1]));

    if area1 > area2
      fb = fc;
      fc = (fb+fa)/2;
    elseif area2 > area1
      fa = fc;
      fc = (fb+fa)/2;
    else
      Q = Q1;
      break
    end
  end
```

Código IV – Código em MATLAB que estima uma ponto sobre a curva $\zeta - constante$ que possui a mesma parte real que N.

```
function resultado = realwn(zeta,Ts)
  fun = @(h) cos(h) == -exp(zeta*(h-pi)/(sqrt(1-zeta^2)));
  h0 = pi/(1.72742*Ts*sqrt(1-zeta^2));
  resultado = fzero(fun,h0);
end
```

Código V – Código em MATLAB que preenche (2.14).

```
function LMI = lmiestabilidade(sigma,sys,Ts,P,Z)
  a11 = -exp(-abs(sigma)*Ts)*P;
  a21 = (P*sys.A'+Z'*sys.B');

  LMI = [a11 a21';
    a21 a11];
```

Código VI – Código em MATLAB que preenche (2.16) ou (2.15).

```
function LMI = lmisetorconico(a,phi,sys,P,Z,direcao)
  arguments
    a
    phi
    sys
    P
    Z
    direcao {mustBeMember(direcao,['E','D'])} = 'E'
  end

  switch direcao
    case 'D'
      a11 = sin(phi)*(2*a*P-sys.A*P-sys.B*Z-P*sys.A'-Z'*sys.B');
      a21 = cos(phi)*(P*sys.A'+Z'*sys.B'-sys.A*P-sys.B*Z);

      LMI = [a11 a21';
        a21 a11];
```

```
    case 'E'
      a11 = sin(phi)*(sys.A*P+sys.B*Z+P*sys.A'+Z'*sys.B'-2*a*P);
      a21 = cos(phi)*(P*sys.A'+Z'*sys.B'-sys.A*P-sys.B*Z);
      LMI = [a11 a21';
        a21 a11];
  end
```

Código VII – Código em MATLAB que verifica a factibilidade dos parâmetros de projeto informado.

```
function K = factibilidade(SYS,TS,SIGMA,ZETA,WN,METODO,PLOTAR)
  % FACTIBILIDADE determina se há uma matriz de ganho K capaz de estabilizar o
  % sistema informado com os parâmetros desejados.
  %
  % K = FACTIBILIDADE(SYS,TS,SIGMA,ZETA,WN,METODO,PLOTAR) verifica se um sistema
  % discreto representado via espaço de estados é factível dentro dos parâmetros
  % de projeto informados. São eles:
  %
  %   - SYS       representação do modelo via espaço de estados;
  %   - TS        período de amostragem;
  %   - SIGMA     valor de estabilidade relativa;
  %   - ZETA      taxa de amortecimento;
  %   - WN        frequência natural-amortecida;
  %   - METODO    (OPCIONAL) método de aproximação usada, onde:
  %       - 'C'     aproximação cônica (VALOR-PADRÃO)
  %       - 'E'     aproximação elíptica
  %       - 'P'     aproximação poligonal
  %   - PLOTAR    (OPCIONAL) opção booleana que plota as regiões
  %               aproximadas. Valor-padrão é FALSO.
  arguments
    SYS
    TS
    SIGMA
    ZETA
    WN
    METODO {mustBeMember(METODO,['C','E','P'])} = 'C'
    PLOTAR {mustBeNumericOrLogical} = false
  end
  A = SYS.A;
  B = SYS.B;
  C = SYS.C;
  D = SYS.D;
  [n,m] = size(B);

  NY = 2*pi/(WN*TS);

  if NY < 4.86
    error('Curva NY não convexa.');
  end

  P = sdpvar(n,n);
  Z = sdpvar(m,n,'f');
  options = sdpsettings('verbose',0);

  F = [];
```

```
F = [F, (P>=0):'Positividade'];

%% Estabilidade Relativa
F = [F, (lmiestabilidade(SIGMA,SYS,TS,P,Z)<=0):'Taxa de amortecimento'];

%% Verifica o método escolhido para a aproximação das regiões
switch METODO
  case 'C'   % Aproximação Cônica
    L = pontoplanoz(ZETA,0,TS);
    N = pontoplanoz(ZETA,pi/(sqrt(1-ZETA^2)*TS),TS);
    M = determinarmaiorarea(L,N,ZETA,TS);
    theta1 = acos(abs(real(M)-L)/abs(M-L));
    theta2 = acos(abs(real(M)-N)/abs(M-N));

    F = [F, (lmisetorconico(real(L),theta1,SYS,P,Z,'E')<=0):['Setor cônico esquerdo ZETA']];
    F = [F, (lmisetorconico(real(N),theta2,SYS,P,Z,'D')<=0):['Setor cônico direito ZETA']];
    F = [F, (A*P+B*Z+Z'*B'+P*A'-2*real(N)*P>=0):['Limitação à direita ZETA']];

    Q = pontoplanoz(0,WN,TS);
    R = pontoplanoz(1,WN,TS);
    theta = acos(abs(real(Q)-R)/abs(Q-R));

    F = [F, (lmisetorconico(R,theta,SYS,P,Z,'D')<=0):['Setor cônico direito WN']];
    F = [F, (A*P+B*Z+Z'*B'+P*A'-2*real(R)*P>=0):['Limitação à direita WN']];

  case 'E'  % Aproximação Elíptica
    L = pontoplanoz(ZETA,0,TS);
    N = pontoplanoz(ZETA,pi/(sqrt(1-ZETA^2)*TS),TS);
    M = determinarmaiorarea(L,N,ZETA,TS);
    theta1 = acos(abs(real(M)-L)/abs(M-L));
    theta2 = acos(abs(real(M)-N)/abs(M-N));

    F = [F, (lmisetorconico(real(L),theta1,SYS,P,Z,'E')<=0):['Setor cônico esquerdo ZETA']];
    F = [F, (lmisetorconico(real(N),theta2,SYS,P,Z,'D')<=0):['Setor cônico direito ZETA']];
    F = [F, (A*P+B*Z+Z'*B'+P*A'-2*real(N)*P>=0):['Limitação à direita ZETA']];

    a = 1-exp((-2*pi)/NY);
    b = (a^2*sin((-2*pi)/NY))/(sqrt(a^2-(cos((-2*pi)/NY)-1)^2));

    e11 = -P;
    e21 = -1/a*P+1/2*(1/a+1/b)*(A*P+B*Z)+1/2*(1/a-1/b)*(P*A'+Z'*B');

    E = [e11 e21';
      e21 e11];

    F = [F, (E<=0):'Elipse'];

  case 'P'  % Aproximação Poligonal
    l = 0;
    L = pontoplanoz(ZETA,0,TS);
    N = pontoplanoz(ZETA,pi/(sqrt(1-ZETA^2)*TS),TS);
    M = double(pontoplanoz(ZETA,realwn(ZETA,TS),TS));
    Q = pontoplanoz(0,WN,TS);
    R = pontoplanoz(1,WN,TS);

    pts1 = [0 pi/(1.5*sqrt(1-ZETA^2)*TS)];
    pts2 = pts1;
```

```
  vec1 = [L M];
  vec2 = vec1;
  pts3 = [0 1];
  pts4 = pts3;
  vec3 = [R Q];
  vec4 = vec3;
  theta = acos(abs(real(vec4(m+1))-R)/abs(vec4(m+1)-R));

  Satual = polyshape(real([vec2 N]),imag([vec2 N])).area;

  F = [F, (A*P+B*Z+Z'*B'+P*A'-2*real(N)*P>=0):['Limitação à direita ZETA ' METODO]];
  F = [F, (lmisetorconico(loc(L,M),acos(abs(L-real(M))/abs(L-M)), ...
    SYS,P,Z,'E')<=0):['Setor cônico esquerdo ZETA' METODO]];
  F = [F, (lmisetorconico(R,theta,SYS,P,Z,'D')<=0):['Setor cônico direito NY ' METODO]];
  F = [F, (A*P+B*Z+Z'*B'+P*A'-2*real(R)*P>=0):['Limitação à direita NY ' METODO]];

  optimize(F,[],options);

  [primalres,dualres] = check(F);
  primalres = sort(primalres,'ascend');
  dualres = sort(dualres,'ascend');

  while primalres(1) < 0 || dualres(1) < 0
    if l < length(vec1)-1
      l = l+1;
    else
      l = 1;
      vec1 = vec2;
      pts1 = pts2;
      vec3 = vec4;
      pts3 = pts4;
    end

    F = [];
    F = [F, (P>=0):'Positividade'];
    F = [F, (lmiestabilidade(SIGMA,SYS,TS,P,Z)<=0):['Taxa de amortecimento ' METODO]];

    Vnew1 = pontoplanoz(ZETA,(pts1(l)+pts1(l+1))/2,TS);
    pts2 = sort([pts2 (pts1(l)+pts1(l+1))/2],'descend');
    vec2 = sort([vec2 Vnew1],'descend');

    Vnew2 = pontoplanoz((pts3(l)+pts3(l+1))/2,WN,TS);
    pts4 = sort([pts4 (pts3(l)+pts3(l+1))/2],'ascend');
    vec4 = sort([vec4 Vnew2],'descend');

    F = [F, (A*P+B*Z+Z'*B'+P*A'-2*real(N)*P>=0):['Limitação à direita ZETA ' METODO]];
    F = [F, (A*P+B*Z+Z'*B'+P*A'-2*real(R)*P>=0):['Limitação à direita NY ' METODO]];

    for m=1:length(vec1)-1
      u1 = loc(vec2(m),vec2(m+1));
      if u1 < 0
        phi = acos(abs(real(vec2(m+1))-u1)/abs(vec2(m+1)-u1));
        F = [F, (lmisetorconico(u1,phi,SYS,P,Z,'D')<=0):['Setor cônico direito ZETA '
METODO ' ' num2str(l)]];
      else
        phi = acos(abs(u1-real(vec2(m+1)))/abs(u1-vec2(m+1)));
        F = [F, (lmisetorconico(u1,phi,SYS,P,Z,'E')<=0):['Setor cônico esquerdo ZETA '
```

```
  METODO ' ' num2str(l)]];
      end
    end

    for m=1:length(vec3)-1
      u2 = loc(vec4(m),vec4(m+1));
      theta = acos(abs(real(vec4(m))-u2)/abs(vec4(m)-u2));
      F = [F, (lmisetorconico(u2,theta,SYS,P,Z,'D')<=0):['Setor cônico direito NY ' METODO
  ' ' num2str(m)]];
    end

    optimize(F,[],options);
    [primalres,dualres] = check(F);
    primalres = sort(primalres,'ascend');
    dualres = sort(dualres,'ascend');

    Sant = Satual;
    Satual = polyshape(real(vec2),imag(vec2)).area;

    if (Sant/Satual) < 1 && (Sant/Satual > 0.999999)
      break;
    end
  end
end

optimize(F,[],options);
[primalres,dualres] = check(F);

if any(primalres < 0) || any(dualres < 0)
  check(F);
  disp('Infactível!');
else
  disp('Factível!');
end

K = value(Z)/value(P);

if PLOTAR == true
  epsilon = 1e-3;
  wnv = 0:epsilon:pi/(TS*sqrt(1-ZETA^2));
  zetav = 0:epsilon:1;

  cdr = pontoplanoz(ZETA,wnv,TS);
  nfc = pontoplanoz(zetav,WN,TS);
  drc = taxadedecaimento(SIGMA,TS);

  hold on
  plot(real(cdr),imag(cdr),':k', ...
    real(cdr),-imag(cdr),':k', ...
    real(nfc),imag(nfc),':k', ...
    real(nfc),-imag(nfc),':k', ...
    real(drc),imag(drc),'--m')

  switch METODO
    case 'C'
      pgon = polyshape([real(L) real(M) real(N) real(M)], ...
        [imag(L) imag(M) imag(N) -imag(M)]);
```

```
      plot(pgon,'LineStyle','--', ...
        'FaceAlpha',0, ...
        'EdgeColor','m')

      plot([real(pontoplanoz(0,WN,TS)), exp(-2*pi/NY), real(pontoplanoz(0,WN,TS))], ...
        [imag(pontoplanoz(0,WN,TS)), 0, -imag(pontoplanoz(0,WN,TS))], ...
        'LineStyle','--', ...
        'Color','m')

     case 'E'
      pgon = polyshape([real(L) real(M) real(N) real(M)], ...
        [imag(L) imag(M) imag(N) -imag(M)]);
      plot(pgon,'LineStyle','--', ...
        'FaceAlpha',0, ...
        'EdgeColor','m')

      syms u v
      fimplicit((u-1)^2/a^2+(a^2-(cos((-2*pi)/NY)-1)^2)*v^2/(a^2*sin((2*pi)/NY)^2)==1, ...
        [exp(-2*pi/NY) real(pontoplanoz(0,WN,TS)) -imag(pontoplanoz(0,WN,TS))
    imag(pontoplanoz(0,WN,TS))], ...
        'LineStyle','--', ...
        'Color','m')

     case 'P'
      plot(real([vec2 vec2(length(vec2))]),imag([vec2 0]),'--m', ...
        real([vec2 vec2(length(vec2))]),-imag([vec2 0]),'--m')
      plot(real(vec4),imag(vec4),'--m', ...
        real(vec4),-imag(vec4),'--m')
    end

    xlabel('Re')
    ylabel('Im')
    SYSCOMP = ss(A+B*K,B,C+D*K,D,TS);
    pzmap(SYSCOMP,'r')
    zgrid(ZETA,WN,TS)
    hold off
  end
end
```