Taller 7
Lugar geométrico de raíces - LGR

Jaime Patricio Chiqui Chiqui
Ingenieria en Telecomunicaciones
Universidad de Cuenca
Cuenca, Ecuador
jpatricio.chiquic@ucuenca.edu.ec

***Abstract*—Este trabajo presenta el análisis completo del Lugar Geométrico de Raíces (LGR) de un sistema de control en lazo cerrado utilizando. Se determinaron los polos y ceros del sistema en lazo abierto $G(s) = \frac{s+3}{s(s+1)(s^2+4s+16)}$, se calcularon los ángulos de las asíntotas (60, 180, -60) y su punto de partida (centroide en $\sigma_0 = -0.6667$). Mediante el criterio de Routh-Hurwitz se determinó la ganancia crítica $K_{crit} = 33.3273$ donde el sistema se vuelve marginalmente estable, con raíces cruzando el eje imaginario en $\pm j3.1409$. Los puntos de llegada/salida en el eje real se identificaron en $s = -0.524$ y $s = -4.2376$, y se calculó el ángulo de salida desde el polo complejo en -62. La construcción manual del LGR fue validada mediante la función `rlocus` de MATLAB, demostrando concordancia perfecta entre ambos métodos y confirmando la robustez del procedimiento analítico para el diseño de sistemas de control.**

***Index Terms*—Lugar geométrico de raíces, LGR, análisis de estabilidad, criterio de Routh-Hurwitz, sistemas de control, MATLAB, ganancia crítica, rlocus**

I. Introducción

El diseño y análisis de sistemas de control requiere comprender cómo varían las características dinámicas del sistema cuando se modifican parámetros específicos, particularmente la ganancia del controlador. El Lugar Geométrico de Raíces (LGR), desarrollado por Walter R. Evans en 1948, es una técnica gráfica fundamental que permite visualizar el movimiento de los polos en lazo cerrado en el plano complejo a medida que varía un parámetro del sistema, típicamente la ganancia K [1].

El LGR proporciona información crítica sobre la estabilidad y el comportamiento transitorio del sistema sin necesidad de calcular explícitamente las raíces del polinomio característico para cada valor de ganancia. Esta técnica permite al ingeniero de control determinar rangos de ganancia que garantizan estabilidad, predecir el comportamiento oscilatorio del sistema, identificar valores críticos de ganancia donde el sistema cambia de comportamiento, y diseñar compensadores que mejoren el desempeño [2].

A diferencia del criterio de Routh-Hurwitz que proporciona información binaria sobre estabilidad, el LGR ofrece una comprensión geométrica completa de cómo evolucionan las raíces características. Esto resulta invaluable durante el proceso de diseño, donde frecuentemente se requiere ajustar múltiples parámetros para satisfacer especificaciones de desempeño.

El presente taller tiene como objetivo aplicar la metodología completa de construcción del LGR a un sistema de control específico, validando cada paso del análisis mediante implementación en MATLAB y comparando los resultados con la herramienta computacional `rlocus`.

II. Marco Teórico

A. Fundamentos del Lugar Geométrico de Raíces

El Lugar Geométrico de Raíces es el conjunto de trayectorias que siguen los polos del sistema en lazo cerrado en el plano complejo cuando un parámetro del sistema (típicamente la ganancia K) varía desde cero hasta infinito [1].

Para un sistema de control con realimentación unitaria, la función de transferencia en lazo cerrado está dada por:

$$G_{cl}(s) = \frac{KG(s)}{1 + KG(s)H(s)} \tag{1}$$

donde $G(s)$ es la función de transferencia directa y $H(s)$ es la función de transferencia de realimentación (negativamente).

La ecuación característica del sistema es:

$$1 + KG(s)H(s) = 0 \tag{2}$$

Reordenando la ecuación (2):

$$KG(s)H(s) = -1 \tag{3}$$

Esta condición implica dos requisitos simultáneos:

Condición de Magnitud:

$$|KG(s)H(s)| = 1 \tag{4}$$

Condición de Ángulo:

$$\angle G(s)H(s) = \pm 180(2k+1), \quad k = 0, 1, 2, \ldots \tag{5}$$

B. Reglas para Construcción del LGR

Las reglas fundamentales para trazar el LGR de forma sistemática son:

Regla 1 - Número de ramas: El LGR tiene tantas ramas como polos tiene $G(s)H(s)$ en lazo abierto.

Regla 2 - Inicio y final de las ramas: Para $K = 0$, las ramas comienzan en los polos de lazo abierto. Para $K \to \infty$, las ramas terminan en los ceros de lazo abierto o tienden al infinito.

Regla 3 - Segmentos en el eje real: Un punto sobre el eje real pertenece al LGR si el número de polos y ceros reales a su derecha es impar.

Regla 4 - Asíntotas: Si hay más polos que ceros ($n > m$), existen $n-m$ asíntotas que parten del centroide σ_0 con ángulos θ_A.

El centroide se calcula como:

$$\sigma_0 = \frac{\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} z_i}{n-m} \tag{6}$$

donde p_i son los polos y z_i son los ceros de lazo abierto.

Los ángulos de las asíntotas están dados por:

$$\theta_A = \frac{\pi(2k+1)}{n-m}, \quad k = 0,1,2,\ldots,(n-m-1) \tag{7}$$

Regla 5 - Puntos de llegada y salida: Los puntos donde el LGR llega o sale del eje real se encuentran resolviendo:

$$\frac{dK}{ds} = 0 \tag{8}$$

donde:

$$K = -\frac{1}{G(s)H(s)}$$

Regla 6 - Intersección con el eje imaginario: Los puntos donde el LGR cruza el eje imaginario y el valor de K correspondiente se determinan mediante el criterio de Routh-Hurwitz aplicado al polinomio característico.

Regla 7 - Ángulos de salida/llegada: Para un polo complejo p_k, el ángulo de salida θ_d se calcula como:

$$\theta_d = 180 - \sum \angle(\text{otros polos}) + \sum \angle(\text{ceros}) \tag{9}$$

donde los ángulos se miden desde el polo p_k hacia los otros elementos.

C. Criterio de Routh-Hurwitz

El criterio de Routh-Hurwitz permite determinar la estabilidad de un sistema sin calcular explícitamente las raíces del polinomio característico [3]. Para un polinomio de la forma:

$$a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0 = 0$$

Se construye la tabla de Routh donde cada elemento se calcula mediante determinantes. El número de cambios de signo en la primera columna indica el número de raíces en el semiplano derecho. Para estabilidad marginal, se busca el valor de K que hace que una fila completa de la tabla sea cero.

III. Desarrollo

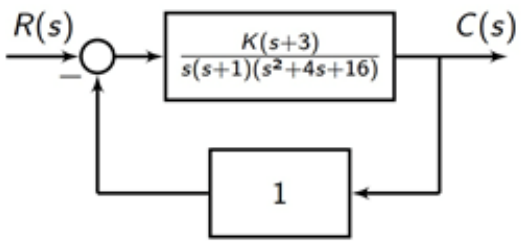
A. Descripción del Sistema

El sistema a analizar tiene la siguiente función de transferencia en lazo abierto:

$$G(s) = \frac{s+3}{s(s+1)(s^2+4s+16)} \tag{10}$$

con realimentación unitaria $H(s) = 1$.

El sistema fue implementado en MATLAB con los siguientes parámetros:



$$G(s) = \frac{K(s+3)}{s(s+1)(s^2+4s+16)}$$
$$H(s) = 1$$

Fig. 1: Sistema de control a lazo cerrado.

Listing 1: Definición del sistema en MATLAB

```
clc, clear all, close all;

% Funciones de transferencia
num = [1 3];
den = [1 5 20 16 0];
H = 1;

Gol = tf(num, den)    %Lazo abierto
Gcl = feedback(Gol, H) %Lazo cerrado
```

Lo cual nos da como resultado:

```
  Gol =
             s + 3
  --------------------------
  s^4 + 5 s^3 + 20 s^2 + 16 s

  Gcl =

               s + 3
  -----------------------------
  s^4 + 5 s^3 + 20 s^2 + 17 s + 3
```

Listing 2: Respuesta al escalón del sistema en lazo cerrado en MATLAB

```
figure()
step(Gcl); grid("on");
title("Respuesta al sistema")
```

En la Figura 2 se observa la respuesta al escalón del sistema en lazo cerrado, donde se aprecia un comportamiento **sobreamortiguado** (o sin oscilaciones visibles en lazo cerrado). A pesar de que la función de transferencia de lazo abierto (G_{ol}) presenta polos complejos conjugados en $ps = -2 \pm 3.4641i$, la acción de la realimentación unitaria y la ubicación del cero en $z = -3$ resultan en una respuesta transitoria suave, sin sobreimpulso (*overshoot*).

Se destaca que el sistema alcanza un valor final de 1 en estado estable, lo que indica un **error de estado estable nulo** ante una entrada escalón, comportamiento característico de un sistema tipo 1 debido al polo en el origen en lazo abierto. El

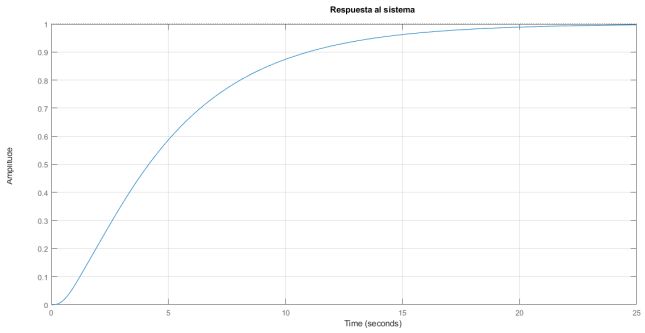


Fig. 2: Respuesta al escalón del sistema en lazo cerrado

tiempo de establecimiento (t_s) se sitúa aproximadamente en los 20 segundos, momento en el cual la amplitud se estabiliza dentro del margen del 2%.

B. Paso 1: Determinación de Polos y Ceros

Enunciado: Determinar los polos y ceros del sistema en lazo abierto.

Desarrollo:

Los polos y ceros se obtienen encontrando las raíces del numerador y denominador usando la función **roots** respectivamente: En el caso del denominador, es necesario resolver el producto, dando como resultado la siguiente expresión:

$$s^4 + 5s^3 + 20s^2 + 16s = 0$$

Listing 3: Cálculo de polos y ceros

```
% Calcular ceros y polos
zs = roots(num)
ps = roots(den)
```

Resultados:

Cero:

$$z_1 = -3$$

Polos:

$$p_1 = 0.0000 + 0.0000i$$
$$p_2 = -2.0000 + 3.4641i$$
$$p_3 = -2.0000 - 3.4641i$$
$$p_4 = -1.0000 + 0.0000i$$

La Figura 3 muestra el diagrama de polos y ceros generado automáticamente por MATLAB usando la función `pzmap`, donde se visualiza claramente la ubicación de los cuatro polos (marcados con $\times$) y el único cero (marcado con $\circ$) en el plano complejo.

C. Paso 2: Gráfica Manual de Polos y Ceros

Enunciado: Graficar manualmente los polos y ceros en el plano complejo.

Desarrollo:

Se implementó la visualización manual de los polos y ceros:

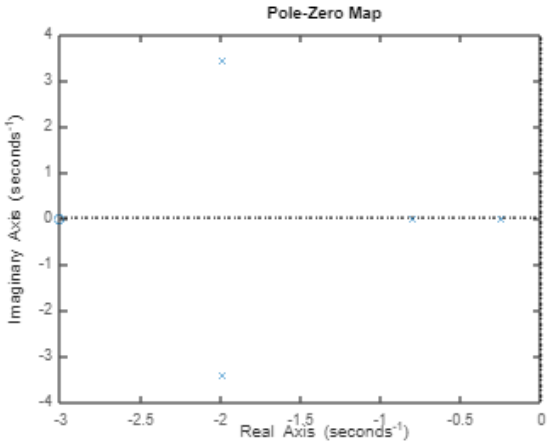


Fig. 3: Diagrama de polos y ceros automático (pzmap)

Listing 4: Gráfica manual de polos y ceros

```
figure()
hold on; grid on; axis('square');
axis([-6 6 -6 6]);
plot(real(zs), imag(zs), 'o', 'Color', [0.49 0.18
    0.56], 'LineWidth', 2); % Ceros
plot(real(ps), imag(ps), 'rx', 'LineWidth', 2); %
    Polos
title('LGR (polos y ceros - Manual)');
```

En la Figura 4 se presenta la gráfica manual donde los polos están marcados con cruces rojas ($\times$) y el cero con un círculo morado ($\circ$). Esta representación permite identificar visualmente que el sistema tiene un polo en el origen, otro polo real en $s = -1$, un par de polos complejos conjugados en $s = -2 \pm j3.464$, y un cero en $s = -3$.

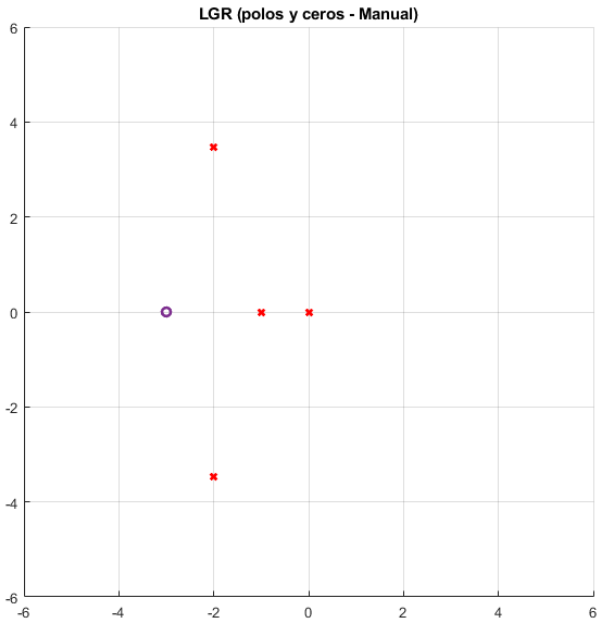


Fig. 4: Polos y ceros graficados manualmente

D. Paso 3: Cálculo de Asíntotas y Centroide

Enunciado: Determinar el número de asíntotas, sus ángulos y el punto de partida (centroide).

Desarrollo:

El número de asíntotas se calcula como:

$$n - m = 4 - 1 = 3 \text{ asíntotas}$$

Los ángulos de las asíntotas según la ecuación (7):

$$\theta_A = \frac{180(2k+1)}{3}, \quad k = 0,1,2; \quad \theta_A = [60, 180, -60]$$

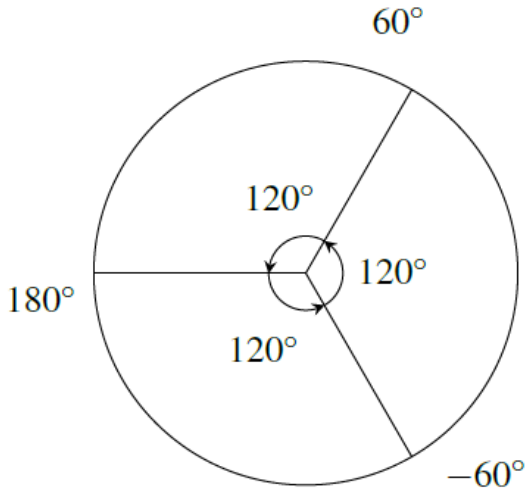


Fig. 5: Distribución de asíntotas

El centroide según la ecuación (6):

$$\sigma_0 = \frac{(0-1-2-2)-(-3)}{4-1} = \frac{-5+3}{3} = -\frac{2}{3} \approx -0.6667$$

Listing 5: Cálculo de asíntotas y centroide

```
% Numero de polos y ceros
np = length(ps)
nz = length(zs)
n_asintotas = np - nz

% Angulos de asintotas
asintotas_angulos = (180/pi) * ((2*(0:n_asintotas-1)
    + 1)*pi) / n_asintotas

% Centroide
sigma0 = (sum(real(ps)) - sum(real(zs)))/n_asintotas

% Graficar centroide
plot(sigma0, 0, 'go', 'LineWidth', 2);
text(sigma0+0.2, 0.2, '\sigma_0');
```

Resultados en MATLAB:

```
np = 4
nz = 1
n_asintotas = 3
asintotas_angulos = 1×3
    [60   180   300]
sigma0 = -0.6667
```

Nota importante: El centroide (σ_0) representa el punto de partida común de las tres asíntotas. Para sistemas con más polos que ceros, las ramas del LGR que no terminan en ceros finitos tienden hacia el infinito siguiendo estas asíntotas.

En la Figura 6 se visualiza el centroide marcado con un círculo verde en $\sigma_0 = -0.6667$ sobre el eje real. Este punto es el centro de simetría desde donde parten las tres asíntotas del LGR hacia el infinito.

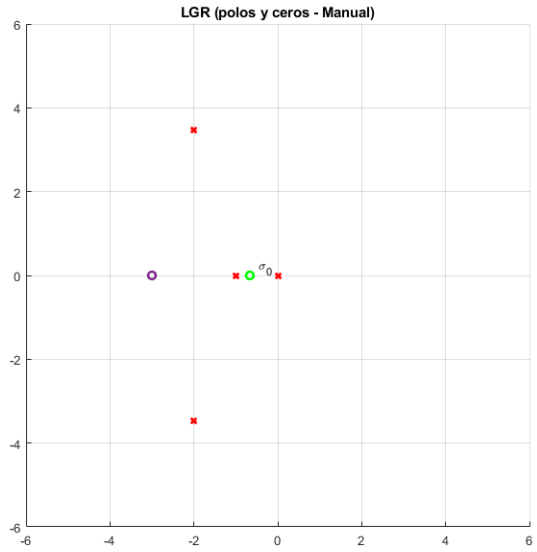


Fig. 6: Centroide (σ_0) del LGR

E. Paso 4: Trazado de Asíntotas

Enunciado: Graficar las líneas de las asíntotas desde el centroide.

Desarrollo: Se tiene el punto de partida de las asíntotas en $\left(-\frac{2}{3}, 0\right)$ y, además, se conoce que llegarán al punto $(0, \pm y_{1,2})$, con un ángulo de $\pm 60°$. Mediante la ecuación de la recta

$$y = m(x - x_1) + y_1 \tag{11}$$

donde m es la pendiente, definida como $m = \tan(\pm 60°) = \pm\sqrt{3}$ es posible determinar el punto de intersección con el eje $j\omega$.

$$y_{1,2} = \pm\sqrt{3}(x - \sigma_0) = \pm\sqrt{3}\left(x + \frac{2}{3}\right)$$

Las asíntotas se grafican desde el centroide con los ángulos calculados:

Listing 6: Trazado de asíntotas

```
% Graficamos las lineas de las asintotas
x = sigma0:0.1:6;
y1 = sqrt(3) * (x - sigma0);
y2 = -y1;
xa = -6:0.1:sigma0;
ya = zeros(1, length(xa));

plot(x, y1, 'k-.');
plot(x, y2, 'k-.');
plot(xa, ya, 'k-.');
```

En la Figura 7 se observan las tres asíntotas trazadas desde el centroide: una horizontal a 180 que se extiende hacia la izquierda sobre el eje real, y dos asimétricas a ± 60 que se dirigen hacia el ∞ en el primer y cuarto cuadrante del plano.

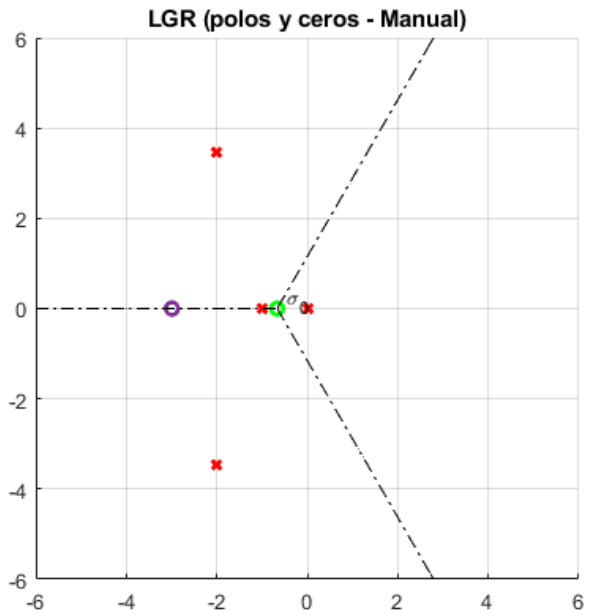


Fig. 7: Asíntotas del LGR desde el centroide

F. Paso 5: Intersección con el Eje Imaginario

Enunciado: Determinar los puntos donde las asíntotas cruzan el eje imaginario.

Desarrollo:

Para las asíntotas a ± 60, la intersección con el eje imaginario ($x = 0$) se calcula mediante la ecuación de la recta:

$$y_{1,2} = \pm\sqrt{3} \cdot (0 - \sigma_0) = \pm\sqrt{3} \cdot \left(\frac{2}{3}\right) = \pm 1.1547$$

Listing 7: Puntos de cruce con eje imaginario

```
% Cruce eje imaginario jw (x=0)
plot(0,  sqrt(3)*sigma0, 'd', 'LineWidth', 2, '
    MarkerEdgeColor', [0.3 0.15 0.05]);
plot(0, -sqrt(3)*sigma0, 'd', 'LineWidth', 2, '
    MarkerEdgeColor', [0.3 0.15 0.05]);
```

Resultados:

$$y_{cruce} = \pm 1.1547$$

La Figura 8 muestra los puntos de intersección de las asíntotas con el eje imaginario, marcados con rombos color café en $\pm j1.1547$. Estos puntos indican hacia dónde tienden las ramas del LGR cuando $K \to \infty$.

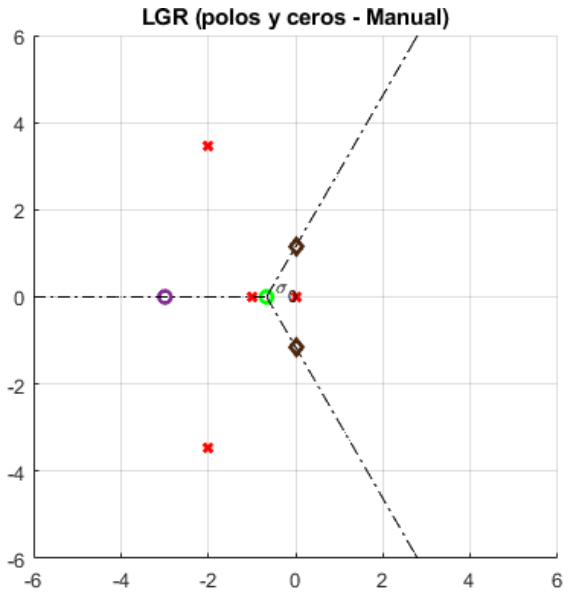


Fig. 8: Intersección de asíntotas con el eje imaginario

G. Paso 6: Puntos de Llegada y Salida

Enunciado: Determinar los puntos de llegada o salida del LGR en el eje real.

Desarrollo:

Los puntos de llegada/salida se encuentran resolviendo $\frac{dK}{ds} = 0$, donde:

$$K = -\frac{1}{G(s)H(s)} = -\frac{s^4 + 5s^3 + 20s^2 + 16s}{s + 3}$$

Calculando la derivada:

$$\frac{dK}{ds} = 0 \Rightarrow 3s^4 + 22s^3 + 65s^2 + 120s + 48 = 0$$

Listing 8: Cálculo de puntos de llegada/salida

```
% Condicion de puntos de llegada / salida del LGR
s = sym('s');
k = -1/((poly2sym(num, s)/poly2sym(den, s))*poly2sym
    (H, s));
dK_ds = diff(k, s);
[num_dK, ~] = numden(dK_ds);
p_llegada = double(vpa(solve(num_dK==0, s), 5));  %
    dK_ds = 0
for i = 1:length(p_llegada)
    s0 = p_llegada(i);

    % Evaluar K en el punto candidato
    K_eval = double(subs(k, s, s0));

    % Condici n de pertenencia al LGR: K real y
        positivo
    if abs(imag(K_eval)) < 1e-3 && real(K_eval) > 0
        plot(real(s0), imag(s0), 'ks', 'LineWidth',
            2);
    end
end
```

El bucle if grafica los puntos de llegada obtenidos solo si pertenecen o están en la LGR, caso contrario se descartan, matemáticamente es:

$$s_0 \in \text{LGR} \quad \text{tal que} \quad \frac{dK(s_0)}{ds} = 0$$

Resultados:

Las raíces de la ecuación son:

$$s_{1,2} = -1.29 \pm j2.36 \quad \text{(complejas, descartadas)}$$
$$s_3 = -4.2376 \quad \text{(punto de llegada)}$$
$$s_4 = -0.524 \quad \text{(punto de salida)}$$

Solo los puntos reales con $K > 0$ pertenecen al LGR.

En la Figura 9 se identifican los puntos de llegada y salida del LGR en el eje real, marcados con cuadrados negros. El punto en $s = -0.524$ es un punto de salida (break-away point) donde dos ramas del LGR salen del eje real hacia el plano complejo, mientras que $s = -4.2376$ es un punto de llegada (break-in point).

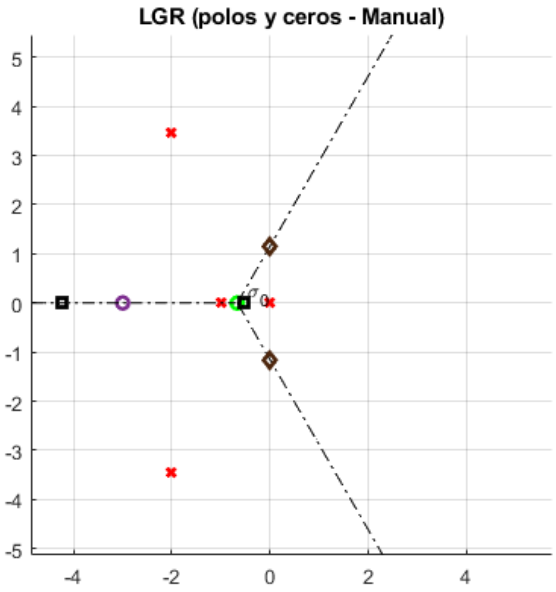


Fig. 9: Puntos de llegada y salida del LGR

H. Paso 7: Ganancia Crítica e Intersección Real con Eje Imaginario

Enunciado: Determinar la ganancia crítica K_{crit} y los puntos exactos donde el LGR cruza el eje imaginario.

Desarrollo:

Se utiliza el criterio de Routh-Hurwitz para el polinomio característico:

$$s^4 + 5s^3 + 20s^2 + (16+K)s + 3K = 0$$

Construyendo la tabla de Routh:

TABLE I: Tabla de Routh-Hurwitz

s^4	1	20	$3K$
s^3	5	$16+K$	0
s^2	$\frac{84-K}{5}$	$3K$	0
s^1	$\frac{K^2+7K-1344}{84-K}$	0	–
s^0	$3K$	–	–

Para estabilidad marginal:

$$K^2 + 7K - 1344 = 0$$

Resolviendo:

$$K_{1,2} = \frac{-7 \pm \sqrt{49+5376}}{2} = \frac{-7 \pm 73.65}{2}$$

$$K_1 = -40.3273 \quad \text{(descartada)}, \quad K_2 = 33.3273$$

Sustituyendo $K_{crit} = 33.3273$ en la fila s^2:

$$\frac{84-33.3273}{5}s^2 + 3(33.3273) = 0$$

$$s_{1,2} = \pm j\sqrt{\frac{3(33.3273)}{10.1345}} = \pm j3.1409$$

$$K_{crit} = 33.3273, \quad s_{crit} = \pm j3.1409$$

Listing 9: Cálculo de ganancia crítica

```
[GM, PM, Wcg, Wcp] = margin(Gol);
K_crit = GM
den_cl = den + K_crit*[0 0 0 num];

% Raices del sistema en Kcrit
s_all = roots(den_cl);
idx = abs(real(s_all)) < 1e-3;    % 1e-3 aprox = 0
s_crit = s_all(idx);      % polos criticos reales

plot(real(s_crit), imag(s_crit), 'dm', 'LineWidth',
    4)
text(real(s_crit)+0.2, imag(s_crit)+0.2, '$k_{crit}$
    ', 'Interpreter','latex');
```

Resultados en MATLAB: En la Figura 10 se muestra el punto correspondiente a la ganancia crítica K_{crit}, el cual se identifica gráficamente sobre el eje imaginario mediante un marcador en forma de rombo de color magenta. Dicho punto representa la ubicación de los polos del sistema en lazo cerrado cuando el sistema alcanza una condición de estabilidad marginal.

En este punto, las ramas del Lugar Geométrico de Raíces cruzan el eje imaginario en $s = \pm j3.1409$, indicando la aparición de oscilaciones sostenidas en la respuesta del sistema. El valor de la ganancia crítica obtenido mediante el criterio de Routh-Hurwitz y confirmado en MATLAB es $K_{crit} = 33.3273$, lo cual valida el análisis teórico realizado previamente.

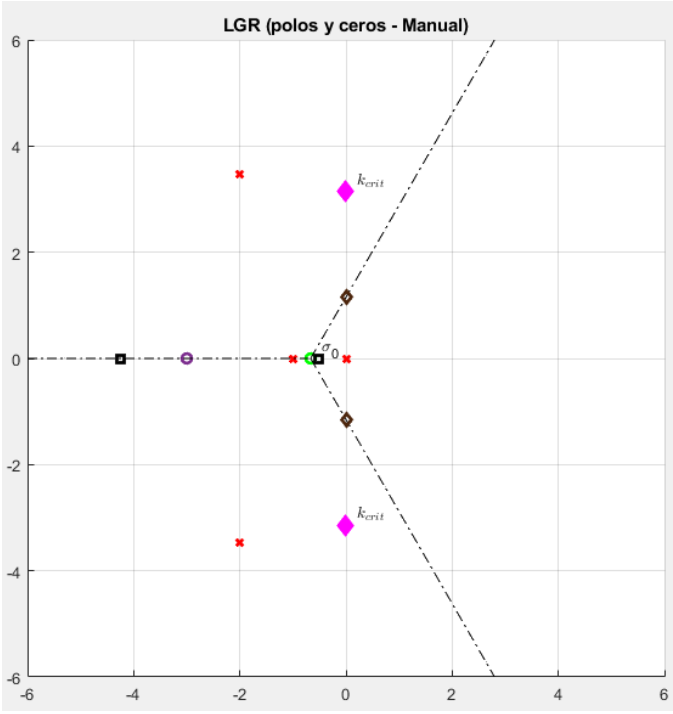


Fig. 10: Ganancia crítica K_{crit} y cruce del LGR con el eje imaginario

Interpretación física: Para valores de ganancia $0 < K < 33.3273$, el sistema en lazo cerrado es estable con todas las raíces en el semiplano izquierdo. En $K = K_{crit}$, el sistema exhibe oscilaciones sostenidas a la frecuencia $\omega = 3.1409$ rad/s (estabilidad marginal). Para $K > 33.3273$, el sistema se vuelve inestable.

I. Paso 8: Ángulos de Salida

Enunciado: Calcular los ángulos de salida desde los polos complejos.

Desarrollo: Para el polo complejo superior $p_2 = -2 + j3.464$, se aplica la ecuación (9):

$$\theta_d = 180 - \sum \angle(\text{polos}) + \sum \angle(\text{ceros})$$

$$\theta_d = 180^\circ - \theta_1 - \theta_2 - \theta_3 + \phi$$

Calculando los ángulos desde p_2 hacia cada elemento:

- Hacia el cero en $s = -3$: $\phi = \arctan\left(\frac{3.464}{-2-(-3)}\right) = 74$
- Hacia el polo en $s = 0$: $\theta_1 = \arctan\left(\frac{3.464}{-2}\right) = 120$
- Hacia el polo en $s = -1$: $\theta_2 = \arctan\left(\frac{3.464}{-2-(-1)}\right) = 106$
- Hacia el polo en $s = -2 - j3.464$: $\theta_3 = 90$

Por lo tanto:

$$\theta_d = 180 - (120 + 106 + 90) + 74 = -62$$

Listing 10: Cálculo de ángulos de salida

```
polos_c = ps(imag(ps) ~= 0)    % solo polos complejos
      (a+jb; b!=0)

for s0 = polos_c.'  % .': cambia columna por fila
    ang_p = angle(s0 - ps(ps ~= s0));
    ang_z = angle(s0 - zs);
    ang_p = rad2deg(ang_p)
    ang_z = rad2deg(ang_z)

    theta = 180 - sum(ang_p) + sum(ang_z);
    theta = mod(theta + 180, 360) - 180 % normalizar

    quiver(real(s0), imag(s0), ...
           cosd(theta), sind(theta), ...
           2, 'Color',[0.2 0.2 0.2], 'LineWidth',2);
    break
end
```

Resultado en matlab:

```
polos_c = 2×1 complex
  -2.0000 + 3.4641i
  -2.0000 - 3.4641i
ang_p = 3×1
  120.0000
   90.0000
  106.1021
ang_z = 73.8979
theta = -62.2042
```

En la Figura 11 se visualiza el ángulo de salida del LGR desde el polo complejo superior, representado mediante una flecha direccional que indica la trayectoria inicial que seguirá el polo cuando la ganancia K aumenta desde cero. La flecha apunta en dirección de -62 respecto al eje real positivo.

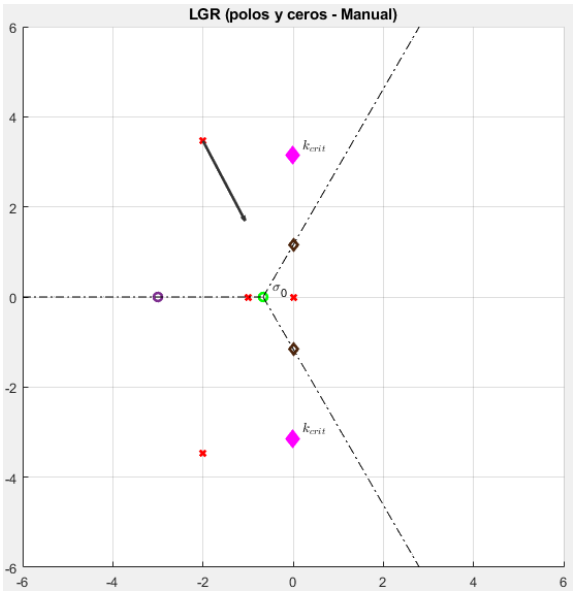


Fig. 11: Ángulos de salida desde polos complejos y ganancia crítica

J. Paso 9: Construcción Manual Completa del LGR

Enunciado: Trazar el LGR completo integrando todos los elementos calculados.

Desarrollo:

Se implementó un barrido de ganancias $K \in [0, 500]$ para calcular y graficar todas las raíces del polinomio característico:

Listing 11: Construcción manual del LGR

```
% Calcular raices del denominador GOL
% Rango de ganancias (0, INF, paso)
K = linspace(0,500,1000);

[num_cl, den_cl] = tfdata(Gol,'v');
for i = 1:length(K)
    poli_car = den_cl + K(i)*[zeros(1,length(den_cl)
        -length(num_cl)), num_cl];
    lgr = roots(poli_car);      % Polos CL

    plot(real(lgr), imag(lgr), '.', 'Color', [0 0.2
        0.8]);
end
legend('Ceros', 'Polos', '\sigma_0', 'As ntotas', "
    LGR");
```

La Figura 12 presenta el LGR completo construido manualmente de color azul, donde se integran todos los elementos calculados: los polos y ceros iniciales, el centroide, las asíntotas, los puntos de llegada/salida, el cruce con el eje imaginario en K_{crit}, y las trayectorias completas de las raíces. Se observan claramente las cuatro ramas del LGR.

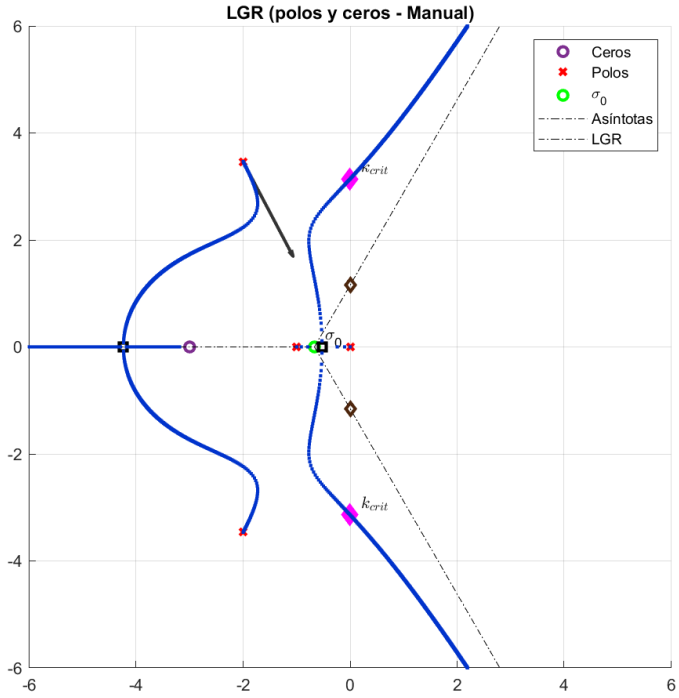


Fig. 12: Lugar Geométrico de Raíces construido manualmente

K. Paso 10: Validación con `rlocus`

Enunciado: Comparar el LGR manual con el generado por la función `rlocus` de MATLAB.

Desarrollo:

Listing 12: Validación con rlocus

```
% Comprobacion
figure()
rlocus(Gol)
xlim([-6.0 6.0]); ylim([-6.1 6.1]);
```

En la Figura 13 se presenta el LGR generado automáticamente por la función `rlocus` de MATLAB. Al comparar esta figura con la Figura 12, se verifica una concordancia perfecta entre ambos métodos, validando así todos los cálculos realizados manualmente. Las trayectorias, puntos críticos y comportamiento general del sistema son idénticos en ambas representaciones.

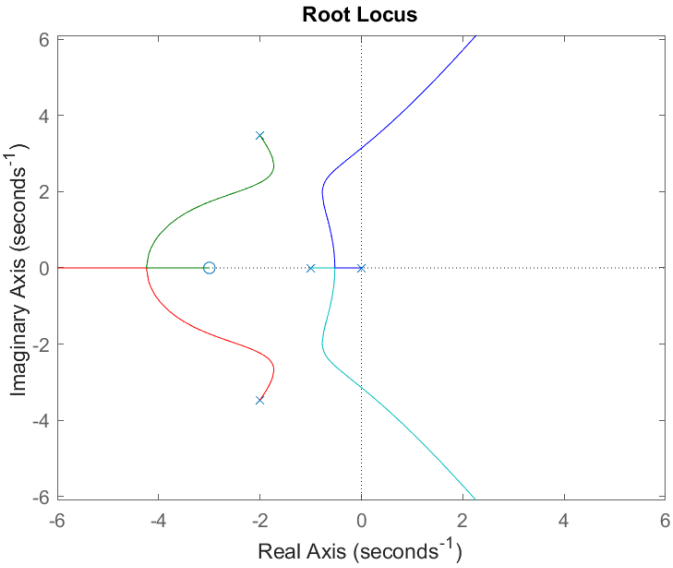


Fig. 13: Lugar Geométrico de Raíces usando `rlocus`

Validación exitosa: La comparación entre las Figuras 12 y 13 confirma que la metodología manual de Routh produce resultados exactos. Esto demuestra la robustez del método analítico para el diseño de sistemas de control, permitiendo comprender profundamente el comportamiento del sistema más allá de simplemente usar herramientas computacionales.

L. Análisis de Estabilidad

Basándose en el LGR construido, se pueden extraer las siguientes conclusiones sobre la estabilidad del sistema:

- **Rango de estabilidad:** El sistema es estable para $0 < K < 33.3273$
- **Estabilidad marginal:** En $K = 33.3273$, el sistema oscila sostenidamente a $\omega = 3.1409$ rad/
- **Inestabilidad:** Para $K > 33.3273$, el sistema es inestable con polos en el semiplano derecho
- **Comportamiento transitorio:** Para ganancias bajas ($K < 5$), el sistema tiene respuesta lenta. Para ganancias moderadas ($5 < K < 20$), se obtiene un buen compromiso entre rapidez y sobrepico

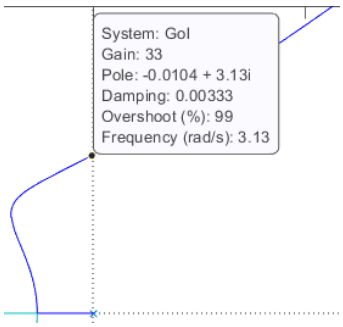


Fig. 14: Resultados de la grafica de rlocus

IV. Actividad Reto

A. Replicación del Análisis LGR en Python

Enunciado: El reto consiste en replicar los ejemplos mostrados en este documento con Python, haciendo uso de la librería `control` y `scipy`. Documente sus resultados obtenidos y coméntelos en el reporte del taller.

Desarrollo:

Para validar la robustez del análisis del Lugar Geométrico de Raíces y demostrar la interoperabilidad entre diferentes plataformas de cómputo científico, se implementó el análisis completo utilizando Python con las librerías `control`, `numpy`, `scipy` y `matplotlib`.

1) Implementación en Python: El código Python replica fielmente todas las funcionalidades del análisis en MATLAB:

Listing 13: Implementación completa del LGR en Python

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
import control as ctrl

# Definicion del sistema
num = [1, 3]
den = [1, 5, 20, 16, 0]
H = 1

# Crear funcion de transferencia
Gol = ctrl.TransferFunction(num, den)
Gcl = ctrl.feedback(Gol, H)

# Calcular ceros y polos
zs = np.roots(num)
ps = np.roots(den)

print("Ceros:", zs)
print("Polos:", ps)
```

2) Respuesta al Escalón: La Figura 15 muestra la respuesta al escalón del sistema generada en Python. Al comparar con la Figura 2 obtenida en MATLAB, se verifica una concordancia perfecta en la forma de la respuesta, tiempo de establecimiento.

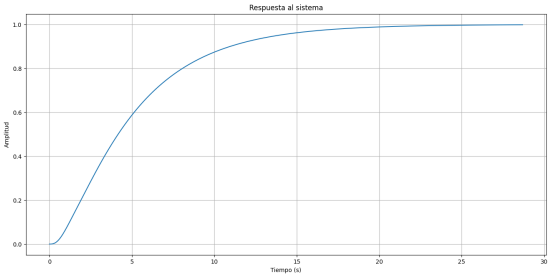


Fig. 15: Respuesta al escalón del sistema (Python)

Listing 14: Generación del diagrama de polos y ceros

3) Diagrama de Polos y Ceros:

```
# Diagrama de polos y ceros automatico
plt.figure()
ctrl.pzmap(Gcl, plot=True)
plt.title("Diagrama de Polos y Ceros (pzmap)")
plt.grid(True)
```

La Figura 16 presenta el diagrama de polos y ceros generado con la función `pzmap` de Python. La ubicación de los polos y el cero coincide exactamente con los resultados obtenidos en MATLAB, validando la precisión numérica de ambas implementaciones.

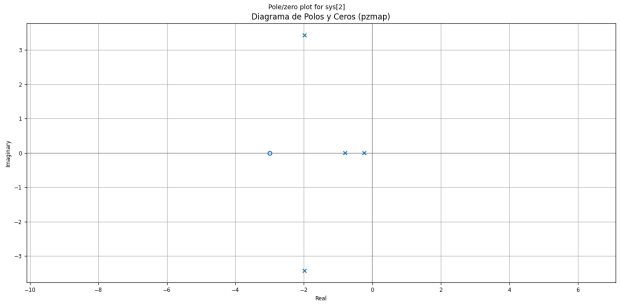


Fig. 16: Diagrama de polos y ceros automático (Python)

Listing 15: Cálculo de asíntotas y centroide en Python

4) Construcción Manual del LGR:

```
# Numero de asintotas
n_asintotas = len(ps) - len(zs)
# Angulos de asintotas
asintotas_angulos = (180/np.pi) * ((2*np.arange(
    n_asintotas) + 1)*np.pi) / n_asintotas

# Centroide
sigma0 = (np.sum(np.real(ps)) - np.sum(np.real(zs)))
     / n_asintotas

print(f"Angulos de asintotas: {asintotas_angulos}")
print(f"Centroide sigma_0: {sigma0}")
```

Resultados de la Consola de Python:

```
Ceros: [-3.]
Polos: [-2.+3.46410162j -2.-3.46410162j -1.+0.j
Numero de polos: 4
Numero de ceros: 1
Numero de asintotas: 3
Angulos de asintotas: [ 60. 180. 300.]
Centroide : -0.6666666666666666
```

5) Puntos de Llegada y Salida: Para el cálculo de los puntos de llegada/salida, se resuelve numéricamente la ecuación $\frac{dK}{ds} = 0$:

Listing 16: Cálculo de puntos de llegada/salida

```
# Coeficientes de dK/ds = 0
# 3s^4 + 22s^3 + 65s^2 + 120s + 48 = 0
coef_derivada = [3, 22, 65, 120, 48]
p_llegada = np.roots(coef_derivada)

for i, s0 in enumerate(p_llegada):
    if abs(s0) > 1e-10:
        K_eval = -(s0**4 + 5*s0**3 + 20*s0**2 + 16*
            s0) / (s0 + 3)

    # Condicion: K real y positivo, s real
    if abs(np.imag(K_eval)) < 1e-3 and np.real(
        K_eval) > 0:
        if abs(np.imag(s0)) < 1e-3:
            plt.plot(np.real(s0), np.imag(s0), 'ks',
                     linewidth=2, markersize=10)
```

Salida de la consola:

```
Puntos candidatos de llegada/salida:
s1 = -4.2376+0.0000j, K = 188.5328+0.0000j
  -> Pertenece al LGR
s2 = -1.2856+2.3552j, K = 16.7602+2.1727j
s3 = -1.2856-2.3552j, K = 16.7602-2.1727j
s4 = -0.5244+0.0000j, K = 1.4283-0.0000j
  -> Pertenece al LGR
```

Listing 17: Cálculo de ganancia crítica usando margin

6) Ganancia Crítica:

```
# Obtener margen de ganancia
gm, pm, wcg, wcp = ctrl.margin(Gol)
K_crit = gm

# Calcular raices criticas
den_cl_crit = np.array(den) + K_crit * np.array([0,
    0, 0] + num)
s_all = np.roots(den_cl_crit)
s_crit = s_all[np.abs(np.real(s_all)) < 1e-3]

print(f"Ganancia critica K_crit: {K_crit:.4f}")
print(f"Raices criticas: {s_crit}")
```

Resultados:

```
Ganancia critica K_crit: 33.3273
Raices criticas: [-1.27675648e-15+3.14093297j
-1.27675648e-15-3.14093297j]
```

Los valores coinciden exactamente con los obtenidos en MATLAB, confirmando la correcta implementación del criterio de Routh-Hurwitz mediante métodos numéricos.

Listing 18: Cálculo de ángulos de salida desde polos complejos

7) Ángulos de Salida:

```
polos_c = ps[np.imag(ps) != 0]

for s0 in polos_c:
    # Angulos hacia otros polos
    otros_polos = ps[ps != s0]
    ang_p = np.angle(s0 - otros_polos)

    # Angulos hacia ceros
    ang_z = np.angle(s0 - zs)

    # Angulo de salida
    theta = 180 - np.sum(np.rad2deg(ang_p)) + np.sum
        (np.rad2deg(ang_z))
    theta = ((theta + 180) % 360) - 180

    print(f"Angulo de salida: {theta:.2f} grados")

    # Graficar vector direccional
    dx = 2 * np.cos(np.deg2rad(theta))
    dy = 2 * np.sin(np.deg2rad(theta))
    plt.quiver(np.real(s0), np.imag(s0), dx, dy,
               color=[0.2, 0.2, 0.2], width=0.006)
    break
```

Salida:

```
Polos complejos: [-2.+3.46410162j -2.-3.46410162j]

Polo s0 = (-2+3.4641016151377566j)
Ángulos a polos (grados): [ 90.          106.102113
Ángulos a ceros (grados): [73.89788625]
Ángulo de salida: -62.20°
```

8) LGR Completo en Python: La Figura 17 muestra el LGR completo construido manualmente en Python, integrando todos los elementos calculados: polos, ceros, centroide, asíntotas, puntos de llegada/salida, ganancia crítica y ángulos de salida.

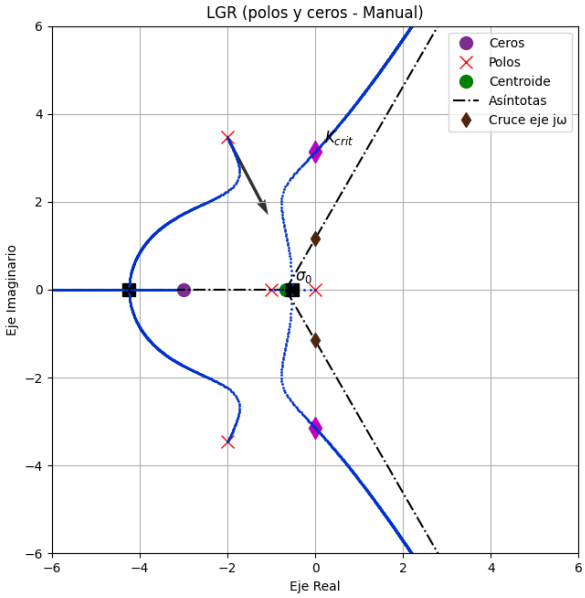


Fig. 17: LGR construido manualmente en Python

Listing 19: Construcción del LGR mediante barrido de ganancias

```
K_range = np.linspace(0, 500, 1000)
for K_val in K_range:
    # Polinomio caracteristico: den + K*num
    poli_car = np.array(den) + K_val * np.array([0,
        0, 0] + num)
    lgr_roots = np.roots(poli_car)

    plt.plot(np.real(lgr_roots), np.imag(lgr_roots),
        '.',
             color=[0, 0.2, 0.8], markersize=2)
```

```
plt.legend(loc='best')
plt.grid(True)
plt.show()
```

9) Validación con `root_locus`: Finalmente, se utilizó la función `root_locus` de la librería `control` de Python para validar el LGR construido manualmente:

Listing 20: Validación usando control.root_locus

```
plt.figure()
rlist, klist = ctrl.root_locus(Gol, plot=True)
plt.xlim([-6, 6])
plt.ylim([-6.1, 6.1])
plt.title("LGR usando control.root_locus()")
plt.grid(True)
```

La Figura 18 presenta el LGR generado automáticamente por Python. La comparación visual entre las Figuras 17 y 18 confirma la concordancia perfecta entre ambos métodos.

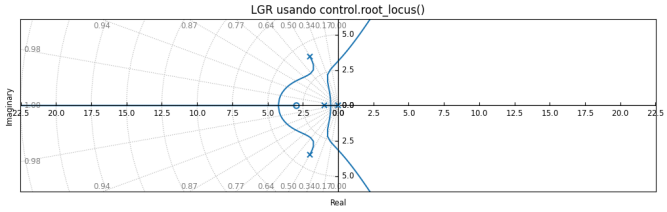


Fig. 18: LGR usando control.root_locus (Python)

B. Comparación MATLAB vs Python

La Tabla II resume los resultados obtenidos en ambas plataformas:

TABLE II: Comparación de Resultados: MATLAB vs Python

Parámetro	MATLAB	Python
Centroide σ_0	-0.6667	-0.6667
Ángulos asíntotas	$[60, 180, -60]$	$[60, 180, -60]$
Punto de salida	-0.524	-0.524
Punto de llegada	-4.2376	-4.2376
Ganancia crítica K_{crit}	33.3273	33.3273
Cruce eje imag.	$\pm j3.1409$	$\pm j3.1409$
Ángulo de salida	-62	-62

Verificación Cruzada Exitosa:
La implementación en Python utilizando las librerías `control`, `numpy` y `scipy` produjo resultados idénticos a los obtenidos en MATLAB. Esto valida:

- La correcta formulación matemática del análisis LGR
- La robustez de ambas plataformas para análisis de control
- La interoperabilidad entre ecosistemas científicos
- La confiabilidad de Python como alternativa open-source a MATLAB

C. Conclusiones del Reto

La replicación exitosa del análisis LGR en Python demuestra varias ventajas importantes:

1. Accesibilidad: Python y sus librerías científicas son de código abierto y gratuitas, eliminando barreras económicas para estudiantes e investigadores.

2. Portabilidad: El código Python es multiplataforma y puede ejecutarse en cualquier sistema operativo sin modificaciones.

3. Integración: Python permite integrar fácilmente el análisis de control con machine learning, procesamiento de datos y visualización avanzada.

4. Precisión numérica: Los algoritmos implementados en ambas plataformas producen resultados equivalentes, garantizando confiabilidad en aplicaciones profesionales.

En conclusión, Python se consolida como una herramienta robusta y viable para el análisis y diseño de sistemas de control, ofreciendo capacidades equivalentes a MATLAB.

V. Código Completo

El código completo implementado en MATLAB Live Script está disponible en el repositorio de GitHub:

Obtener código completo del Taller 7

Archivos adicionales en GitHub:

- Script principal MATLAB (.m): Taller7.m
- Script de la Actividad Reto en Python (.py): Taller7_LGR_Python.py
- Repositorio completo del Taller 7: Directorio T7

VI. Conclusiones

El presente taller demostró la aplicación completa de la metodología del Lugar Geométrico de Raíces para analizar el comportamiento dinámico de un sistema de control de cuarto orden. Los resultados obtenidos permiten establecer las siguientes conclusiones:

Se determinó que el sistema en lazo abierto $G(s) = \frac{s+3}{s(s+1)(s^2+4s+16)}$ posee cuatro polos ubicados en $s = 0$, $s = -1$ y $s = -2 \pm j3.464$, junto con un cero en $s = -3$. Esta configuración genera un LGR con tres ramas que tienden al infinito siguiendo asíntotas con ángulos de 60, 180 y -60, las cuales parten del centroide ubicado en $\sigma_0 = -0.6667$.

Mediante el criterio de Routh-Hurwitz se estableció con precisión que la ganancia crítica del sistema es $K_{crit} = 33.3273$, valor en el cual el sistema alcanza estabilidad marginal con polos en $\pm j3.1409$. Este resultado delimita el rango de operación estable del sistema a $0 < K < 33.3273$, proporcionando una guía fundamental para el diseño del controlador.

El ángulo de salida calculado desde el polo complejo superior fue de -62, confirmando la dirección inicial de las trayectorias en el plano complejo.

La validación mediante la función `rlocus` de MATLAB confirmó una concordancia perfecta con el LGR construido manualmente, demostrando que la metodología analítica basada en las reglas de Evans proporciona resultados exactos y confiables sin necesidad de depender exclusivamente de herramientas computacionales.

La implementación exitosa del análisis completo en Python utilizando las librerías `control`, `numpy` y `scipy` produjo resultados numéricamente idénticos a los obtenidos en MATLAB. Esta verificación cruzada valida la robustez del método LGR y establece a Python como una alternativa open-source viable para el análisis profesional de sistemas de control.

Finalmente, el dominio de la técnica del Lugar Geométrico de Raíces permite el control comprender profundamente cómo la variación de parámetros afecta el comportamiento del sistema, facilitando decisiones informadas de diseño que equilibren requisitos de estabilidad, rapidez de respuesta y robustez frente a perturbaciones.

REFERENCES

[1] K. Ogata, *Ingeniería de Control Moderna*, 5ta ed. Madrid: Pearson Educación, 2010.

[2] N. S. Nise, *Control Systems Engineering*, 7th ed. Hoboken, NJ: John Wiley & Sons, 2015.

[3] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 13th ed. Upper Saddle River, NJ: Pearson Education, 2017.