Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Matemática

Laboratorio 5

Principio de Pontryagin y Ecuaciones de HJB.

Estudiantes: Diego Olguín
Maximiliano S. Lioi
Profesor: Héctor Ramírez C.
Auxiliar: Matías V. Vera
Ayudante de laboratorio: S. Adrián Arellano
Santiago de Chile

Índice de Contenidos

Índice de Figuras

Índice de Códigos

1. Introducción

En el presente informe se muestran los resultados del laboratorio 5. Este consta de dos partes importantes. La primera, la parte A, se compone de 6 ejercicios, siendo principalmente de implementación numérica en Python de un modelo de ecuaciones diferenciales ordinarias controladas en un problema de pesca en donde se busca maximizar un funcional de costo tipo lagrange. Lo asociado a Python culmina con la implementación del método de tiro para la resolución del problema adjunto, lo que se inspira en lo que resulta del Principio del Máximo de Pontryagin, especialmente de las condiciones finales entregadas por las condiciones de transversalidad.
Se compara con lo obtenido por el software de resolución de problemas de control óptimo BOCOP.
La parte B consta de resolver un problema de control óptimo Bolza mediante ecuaciones de HJB con un ansatz cuadrático en espacio para la función valor, obteniendo ecuaciones diferenciales ordinarias para los coeficientes temporales de dicha ecuación. Luego utilizando síntesis del control óptimo en este caso se reconstruye el control en forma de feedback. Se compara nuevamente con lo obtenido con BOCOP para este problema.

2. Parte A. Métodos numéricos basados en Pontryagin

2.1. Contexto

En las costas de Chile habitan camarones nylon. Estos animales son consumidos al rededor de todo Chile, por lo tanto los pescadores locales se dedican a extraerlos, sin embargo, si los pescan todos, al periodo siguiente no podrán extraer más camarones y se terminaría su negocio. En este problema, trataremos de encontrar un modo de hacer que este negocio siga existiendo para los pescadores de las costas de Coquimbo. Localmente (en 2,000 km^2), podemos describir el sistema dinámico de los camarones, como sigue:

$$\begin{cases} \min -\displaystyle\int_0^{t_f} C(t)\cdot(CA(t)\cdot m_a + CC(t)\cdot m_c)\,dt \\ \dfrac{d}{dt}CA(t) = A\cdot\zeta(t-d)\cdot CC(t) - (M+C(t))\cdot CA(t) \\ \text{sa. } \dfrac{d}{dt}CC(t) = -A\cdot\zeta(t-d)\cdot CC(t) - (M+C(t))\cdot CC(t) + N\cdot\zeta(t)\cdot CA(t) \\ C(t)\in[0,C_{\max}] \quad \forall t\in[0,t_f] \end{cases}$$

Donde, las variables dependientes del tiempo son:

- $CA(t)$: la cantidad de camarones adultos en un determinado momento.
- $CC(t)$: la cantidad de crías de camarones en un determinado momento.

• $C(t)$: la intensidad de captura en un instante de tiempo (el control).

Y donde $\zeta(t) = 1 + \sin\left(\frac{2\pi t}{c}\right)$ junto a las constantes :

Variable	Valor	Descripción	Unidades
t_f	52	tiempo final	(semanas)
$C_{\max}$	1	captura máxima	(kg)
m_a	0,02	masa por camarón adulto	(kg)
m_c	0,005	masa por cría de camarón	
A	0,5	tasa de paso a la adultez	
N	0,85	tasa de natalidad	(semanas)
M	0,35	tasa de mortalidad	(semanas)
c	12	duración de un ciclo de reproducción	
d	4	desface del ciclo de reproducción con el de madurez	(semas.

De ahora en adelante consideramos las condiciones iniciales $CA(0) = 3000$ y $CC(0) = 1000$

2.2. Ejercicio 1

Se utiliza el Principio del Máximo de Pontryagin para encontrar la dinámica de los estados adjuntos $p = (p_1, p_2)$, las condiciones de transversalidad, y una caracterización del control óptimo.

Para ello consideramos el Hamiltoniano del sistema

$$H(CA, CC, C, p_1, p_2, t) = -C(CAm_a + CCm_c)$$

$$+p_1(A\zeta(t-d)CC - (M+C)CA) + p_2(-A\zeta(t-d)CC - (M+C)CC + N\zeta(t)CA)$$

Donde el término de tipo lagrangeano es $l(CA, CC, C, p_1, p_2, t) = -C(CAm_a + CCm_c)$

Se deduce que el estado adjunto $\dot{p} = -\partial_x H$ tiene la siguiente estructura

$$\begin{bmatrix} \dot{p_1}(t) \\ \dot{p_2}(t) \end{bmatrix} = - \begin{bmatrix} -C(t)m_a - p_1(t)(M + C(t)) + p_2(t)N\zeta(t) \\ -C(t)m_c + p_1(t)A\zeta(t-d) + p_2(-(M + C(t)) - A\zeta(t-d))) \end{bmatrix}$$

Puesto que el problema es de tipo Lagrangeano se deducen las condiciones de transversalidad

$$p_0 = 1 \ , \ p(t_f) = 0$$

Para caracterizar el control óptimo hacemos uso del principio de mínimo de Pontryagin

$$u(t) \in \arg \min_{w \in [0, C_{max}]} H(CA(t), CC(t), w, p_1(t), p_2(t), t)$$

$$\iff u(t) \in \arg \min_{w \in [0, C_{max}]} \{-C(CAm_a + CCm_c) + p_1(A\zeta(t-d)CC - (M+C)CA)$$

$$+p_2(-A\zeta(t-d)CC - (M+C)CC + N\zeta(t)CA)\}$$

Es equivalente a

$$\iff u(t) \in \arg \min_{C \in [0, C_{max}]} -C(CA(t)m_a + CC(t)m_c) - p_1(M + C(t))CA(t) - p_2(M + C(t))CC(t)$$

$$\iff u(t) \in \arg \min_{C \in [0, C_{max}]} C(-CA(t)m_a - CC(t)m_c - p_1(t)CA(t) - p_2(t)CC(t))$$

$$\implies u(t) = C_{max} * \mathbb{1}[-CA(t)m_a - CC(t)m_c - p_1(t)CA(t) - p_2(t)CC(t) \leq 0]$$

De donde obtenemos una caracterización para el control óptimo $u(t)$

2.3. Ejercicio 2

Se crea una función que le permite calcular el control óptimo en un determinado instante, en función de CA, CC, p, t haciendo uso de la caracterización anterior, para ello se escribe la función

$$u(t) = C_{max} * \mathbb{1}[-CA(t)m_a - CC(t)m_c - p_1(t)CA(t) - p_2(t)CC(t) \leq 0]$$

Siguiendo la regla

```
def C_fun(CC, CA, p1, p2, t):
    return Cmax*(-CA*ma-CC*mc-p1*CA-p2*CC <= 0)
```

2.4. Ejercicio 3

Se implementa el sistema dinámico acoplado (CC, CA, p_1, p_2) sin considerar las condiciones de transversalidad, haciendo uso de la caracterización anterior para el control en función de los estados de este sistema

Dinámica:

$$\begin{bmatrix} \dot{CA}(t) \\ \dot{CC}(t) \\ \dot{p_1}(t) \\ \dot{p_2}(t) \end{bmatrix} = \begin{bmatrix} A \cdot \zeta(t-d) \cdot CC(t) - (M + C(t)) \cdot CA(t) \\ -A \cdot \zeta(t-d) \cdot CC(t) - (M + C(t)) \cdot CC(t) + N \cdot \zeta(t) \cdot CA(t) \\ -(-C(t) \cdot m_a - p_1 \cdot (M + C(t)) + p_2 \cdot N \cdot \zeta(t)) \\ -(-C \cdot m_c + p_1 \cdot A \cdot \zeta(t-d) + p_2 \cdot (-(M + C(t)) - A \cdot \zeta(t-d))) \end{bmatrix}$$

Código 1: Sistema dinámico acomplado (CC, CA, p_1, p_2)

```
def F(t, X, args):
    CA, CC, p1, p2 = X # Estado
    C = C_fun(CC, CA, p1, p2, t) # Construye el control óptimo a partir del estado

    # Dinámica
    dCA = A*zeta(t-d)*CC - (M+C)*CA
    dCC = -A*zeta(t-d)*CC - (M+C)*CC + N*zeta(t)*CA
    dp1 = -(-C*ma - p1*(M+C) + p2*N*zeta(t))
    dp2 = -(-C*mc + p1*A*zeta(t-d) + p2*(-(M+C)-A*zeta(t-d)))

    return np.array([dCA, dCC, dp1, dp2])
```

2.5. Ejercicio 4: Método de tiro

Se utilizan las condiciones de transversalidad y la parte anterior, para determinar el control óptimo del sistema.

Para ello buscamos los valores $(p_1(0), p_2(0))$ asociados a la solución de la dinámica (CA, CC, p_1, p_2) sujeta a las condiciones de transversalidad $(p_1(t_f), p_2(t_f)) = (0, 0)$ mediante el método de tiro, para ello se hace uso del método *fsolve* considerando las funciones

$$f(x) = p_1(t_f) \ , \ g(x) = p_2(t_f)$$

Donde $(p_1(t), p_2(t))$ las soluciones de la dinámica y $x = (p_1(0), p_2(0)) \in \mathbb{R}^2$

$$\dot{X}(t) = f(t, X(t)) \quad X(0) = \begin{bmatrix} CA(0) \\ CC(0) \\ p_1(0) \\ p_2(0) \end{bmatrix}$$

Y donde

$$\dot{X}(t) = \begin{bmatrix} \dot{CA}(t) \\ \dot{CC}(t) \\ \dot{p_1}(t) \\ \dot{p_2}(t) \end{bmatrix} \ , \ f(t, X(t)) = \begin{bmatrix} A \cdot \zeta(t-d) \cdot CC(t) - (M + C(t)) \cdot CA(t) \\ -A \cdot \zeta(t-d) \cdot CC(t) - (M + C(t)) \cdot CC(t) + N \cdot \zeta(t) \cdot CA(t) \\ -(-C(t) \cdot m_a - p_1 \cdot (M + C(t)) + p_2 \cdot N \cdot \zeta(t)) \\ -(-C \cdot m_c + p_1 \cdot A \cdot \zeta(t-d) + p_2 \cdot (-(M + C(t)) - A \cdot \zeta(t-d))) \end{bmatrix}$$

Por lo que se resuelve en *fsolve* el sistema de ecuaciones

$$\begin{cases} f(x) = 0 \\ g(x) = 0 \end{cases} \quad , \quad x = (p_1(0), p_2(0)) \in \mathbb{R}^2$$

Código 2: Método de tiro

```
# Método de tiro
# Entrega valores de p_1(tf) y p_2(tf) para la condición inicial x = (p_1(0) y p_2(0))
def shooting(x):
    p10, p20 = x
    X0 = np.array([CA0, CC0, p10, p20])
    res = solve_ivp(F, (0, tf), X0, args=(args,)) # Resuelve sistema dinámico
    X = res["y"]
    CA, CC, p1, p2 = X

    return [p1[-1],p2[-1]]

# Entrega (p_1(0), p_2(0)) que resuelve el sistema de ecuaciones
res_sol = fsolve(shooting, x0=[-10,-10], epsfcn=1e-21, full_output=True)
```

Resuelto el sistema de ecuaciones se procede a computar la dinámica (CA, CC, p_1, p_2) con condiciones iniciales definidas por los valores de $p_1(0)$ y $p_2(0)$ encontrados mediante el método de tiro.

```
# Inicialización
nt = 100
t_eval = np.linspace(0, tf, nt)
p10, p20 = res_sol[0] # Método de tiro
X0 = np.array([CA0, CC0, p10, p20])

# Resuelve dinámica
res = solve_ivp(F, (0, tf), X0, t_eval=t_eval, args=(args,))

X = res["y"]
CA, CC, p1, p2 = X
```

Obteniendose los siguientes trayectorias óptimas asociadas al control óptimo

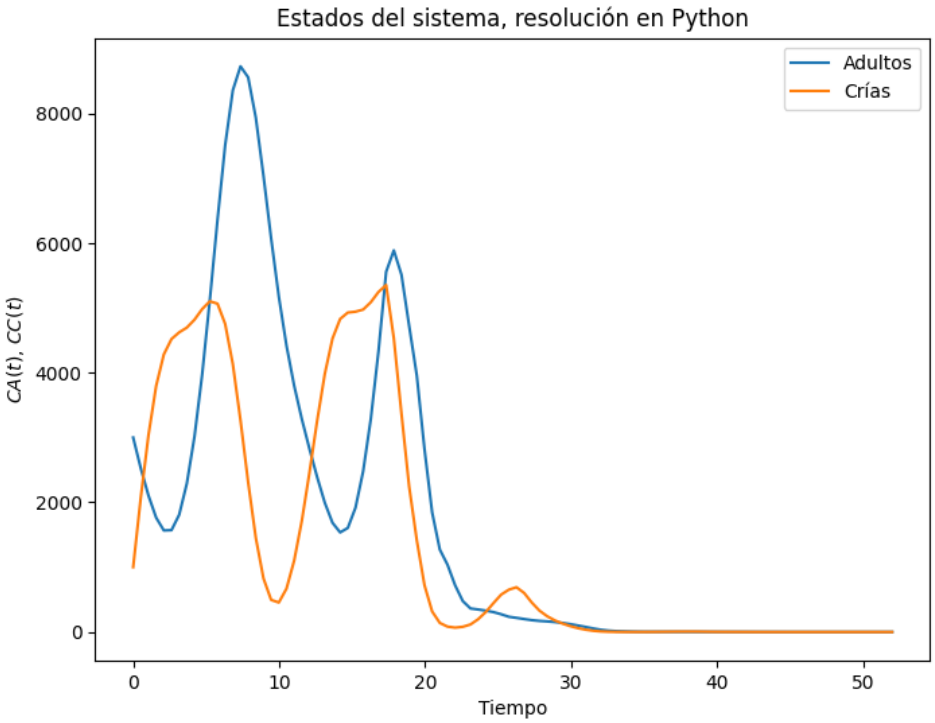

Figura 1: Estados del sistema asociados al control óptimo

Por otro lado, se reconstruye el control óptimo una vez conocidos los estados óptimos del sistema mediante el código

```
# Entrega el control óptimo conocidos los estados óptimos (CC,CA,p1,p2)
co_op = C_fun(CC, CA, p1, p2, t_eval)
```

Graficando el control óptimo se obtienen los resultados

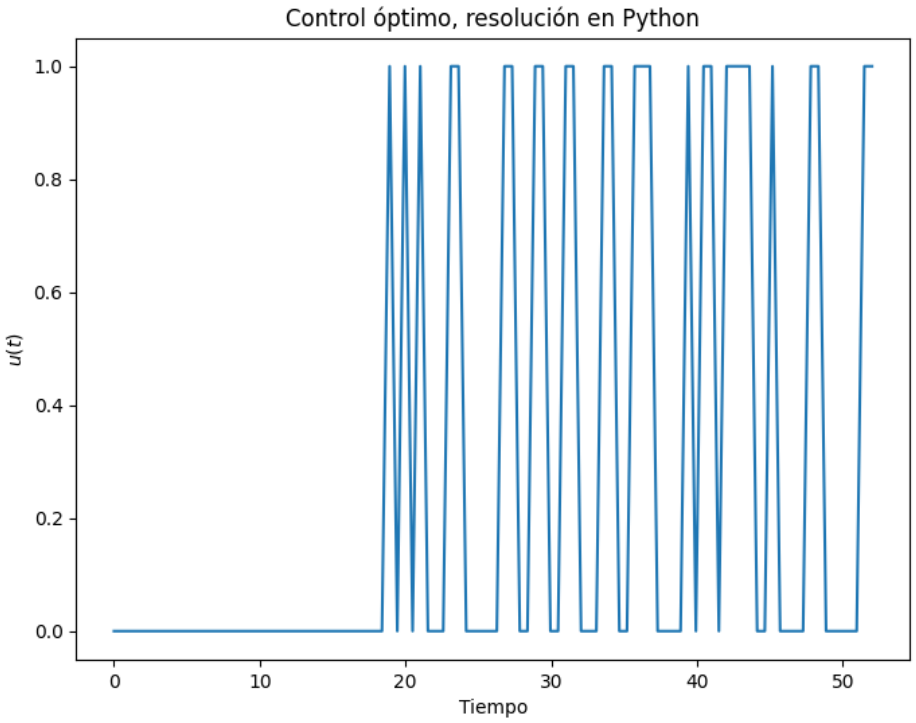

Figura 2: Control óptimo asociado al problema obtenido a partir de las trayectorias óptimas

Notamos que el control que muestra Python consiste en salir a pescar por cortos periodos de tiempo a maxima intensidad de manera muy frecuente, pasar de una intensidad de captura de 0 a C_{max} de manera frecuente durante cortos periodos de tiempo se traduce al hecho de salir a terreno múltiples veces que puede suponer costos adicionales.

2.6. Ejercicio 5

El objetivo de este ejercicio es implementar el problema en BOCOP y obtener una solución con este software. Para ello se convierte el problema Lagrange en uno Mayer como sigue. Sea

Entonces z satisface

$$\dot{z}(t) = C(t)(CA(t)m_a + CC(t)m_c); \quad z(0) = 0$$

Por lo que el problema queda

$$\begin{cases} \min -z(t_f) \\ \quad \frac{d}{dt}CA(t) = A \cdot \zeta(t-d) \cdot CC(t) - (M + C(t)) \cdot CA(t) \\ \text{sa. } \frac{d}{dt}CC(t) = -A \cdot \zeta(t-d) \cdot CC(t) - (M + C(t)) \cdot CC(t) + N \cdot \zeta(t) \cdot CA(t) \\ \quad C(t) \in [0, C_{\max}] \quad \forall t \in [0, t_f] \end{cases}$$

Así se puede introducir el problema a BOCOP. Se utilizaron 100 pasos de tiempo y el método Gauss II (implícito, 2 etapas, orden 4). Los gráficos obtenidos son los siguientes

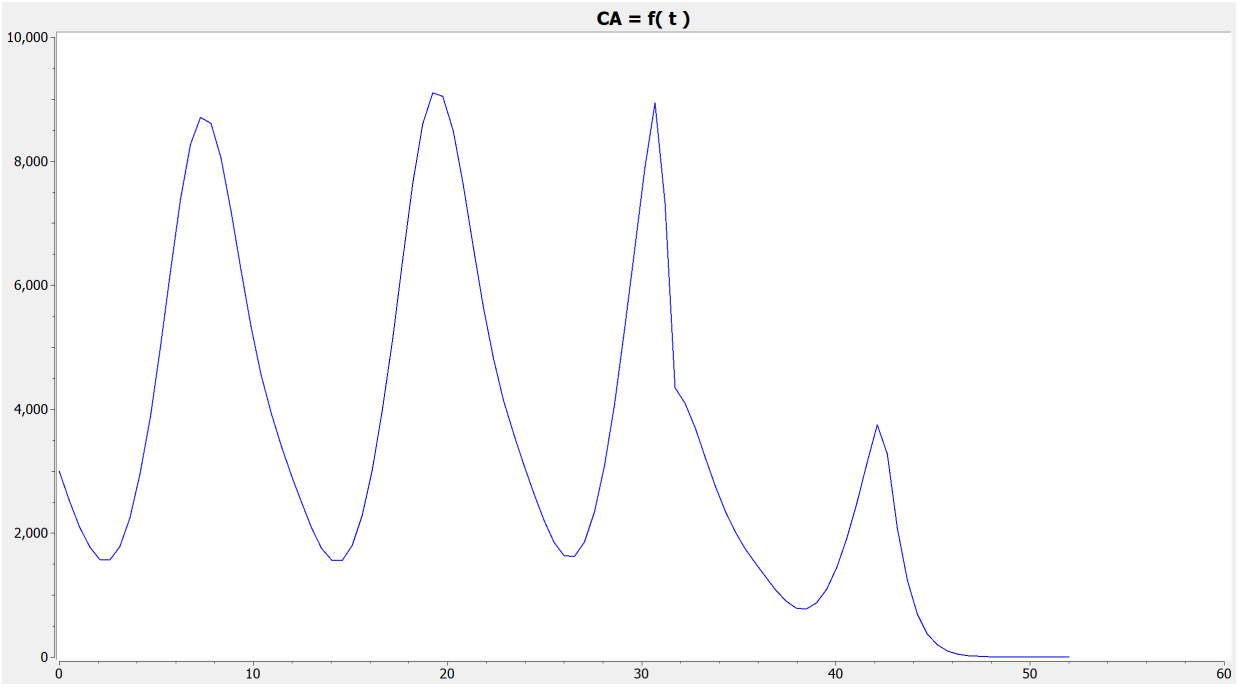

Figura 3: Trayectoria óptima de los camarones adultos obtenida en BOCOP

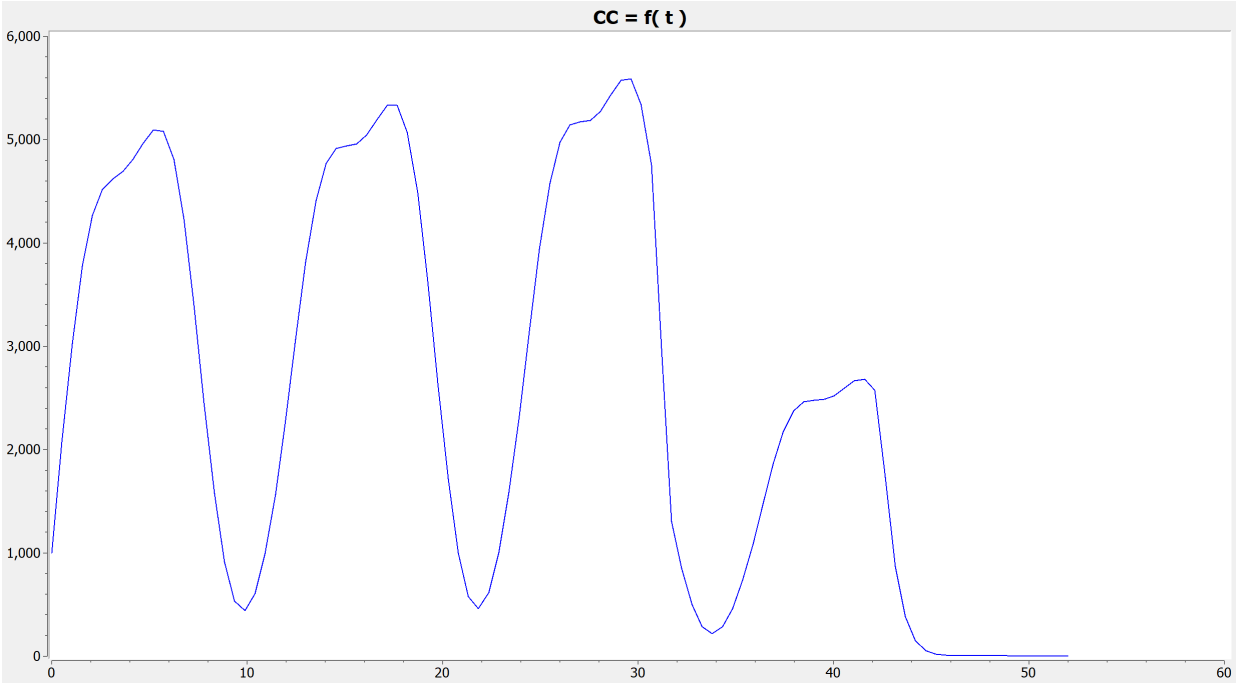

Figura 4: Trayectoria óptima de los camarones cría obtenida en BOCOP

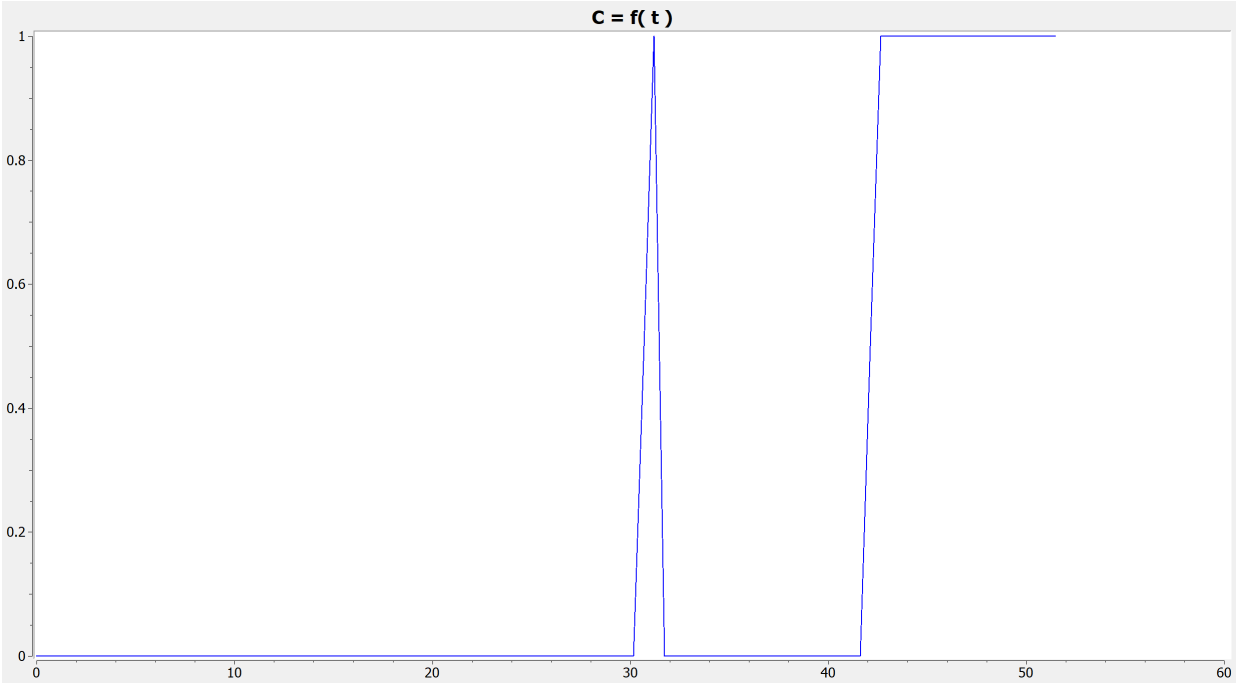

Figura 5: Control óptimo obtenido por BOCOP

Puesto que

$$z(t) = \int_0^t C(s)(CA(s)m_a + CC(s)m_c)ds$$

Estudiamos la evolución de la función objetivo viendo la evolución del estado $z(t)$

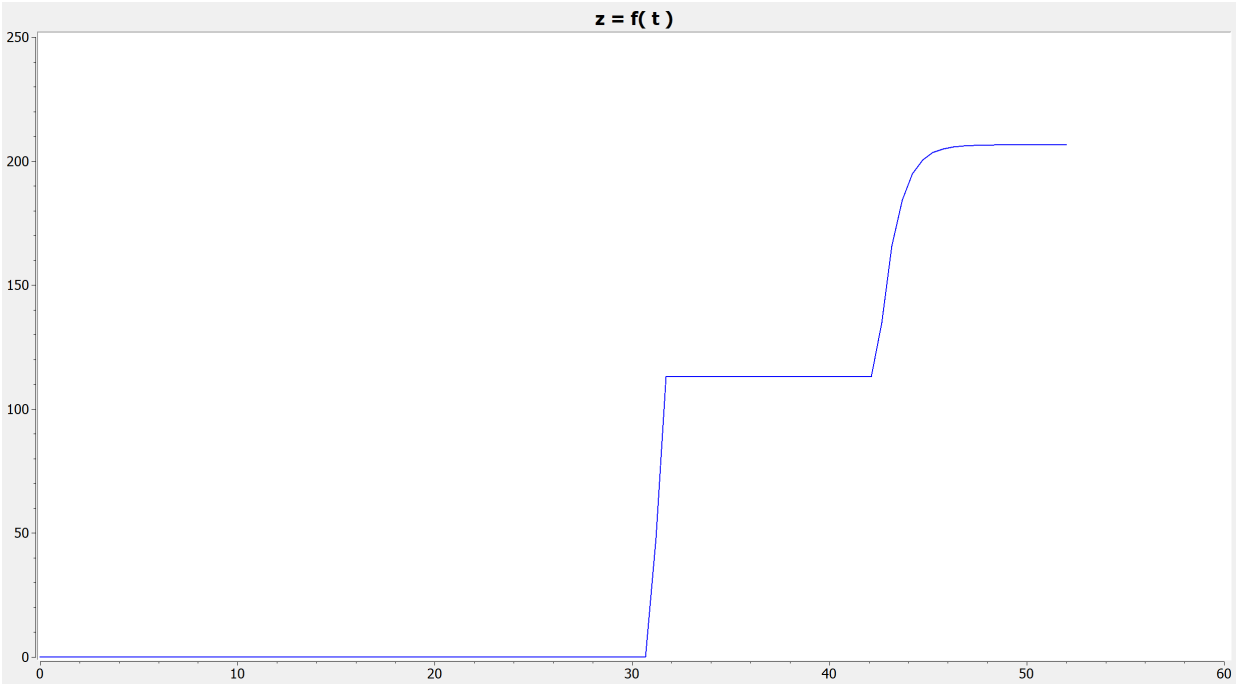

Figura 6: Evolución de la función objetivo obtenido por BOCOP

Notamos que la solución obtenida por BOCOP supera a aquella obtenida por Python en términos del valor objetivo alcanzado, además, el comportamiento que toma el control óptimo también es de tipo Bang-Bang, pero a diferencia del control óptimo obtenido en Python que consiste en un control con reiteradas oscilaciones entre 0 y C_{max}, el control obtenido en BOCOP propone una estrategia de trabajo mas afín a un proyecto de terreno pues consiste en un periodo de captura corto, para luego comenzar un periodo de captura hasta terminar la población de camarones.

2.7. Ejercicio 6

Se comparan ambos métodos, mostrando lado a lado, el control, la evolución de las variables de estado, la función de optimización, el valor objetivo obtenido en cada caso y se estiman errores.

Respecto de la evolución de las variables de estado, logramos apreciar comportamientos distintos a lo largo del tiempo

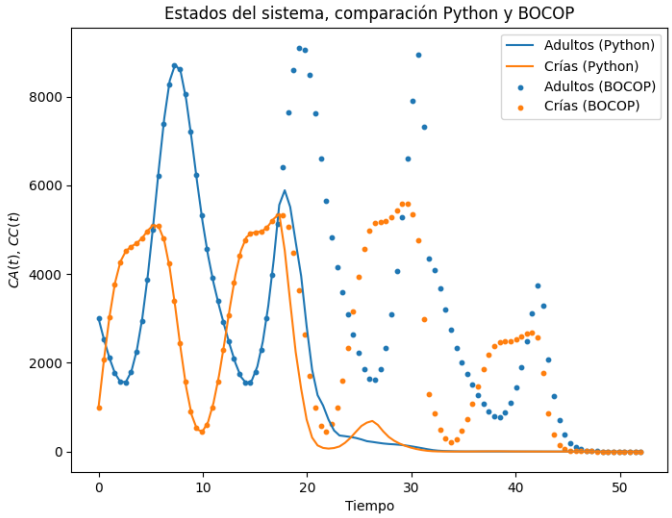

Figura 7: Comparativa estados del sistema asociados al control óptimo entre Python y BOCOP

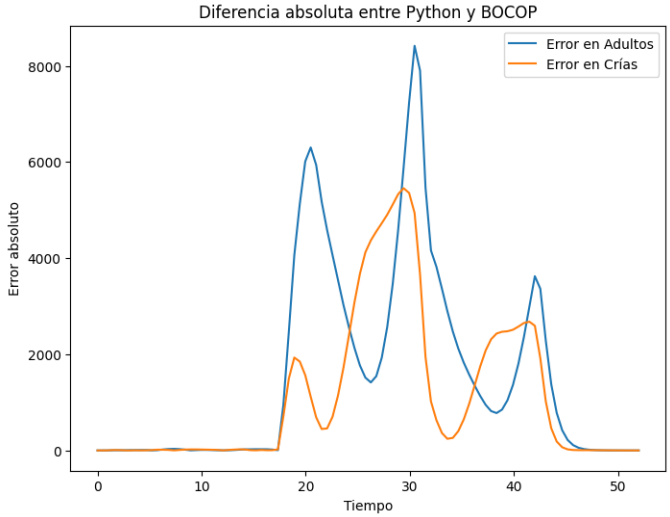

Figura 8: Diferencia absoluta de los estados entre Python y BOCOP a lo largo del tiempo

En efecto, los controles óptimos encontrados en Python y BOCOP toman formas distintas

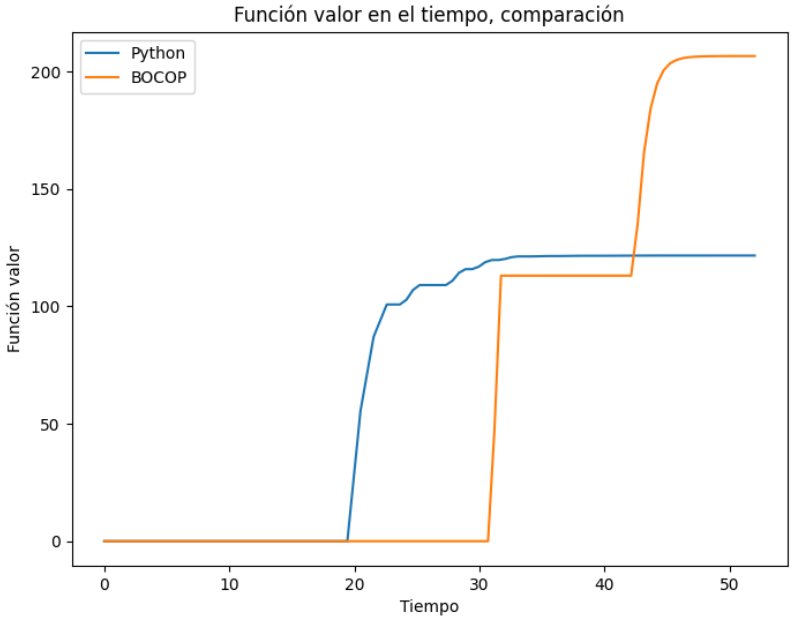

Figura 9: Comparativa entre los valores objetivos entre Python y BOCOP a lo largo del tiempo

Considerando las comparativas, se plantean las siguientes preguntas

• ¿Cuál de los controles utilizaría para resolver este problema en la vida real? ¿Por qué?

R: En primer lugar, el control obtenido en BOCOP alcanza un mayor valor para la función objetivo que aquel obtenido por Python, con una diferencia notable, posiblemente se deba a errores numéricos que se propagaron a partir de la solución de $(p_1(0), p_2(0))$ con el método de tiro, por lo que en principio se escoge la solución propuesta por BOCOP

Por otro lado, observando la forma que tienen los controles, notamos que el control que muestra Python resulta poco practico en la vida real pues consiste en salir a pescar por cortos periodos de tiempo a maxima intensidad de manera muy frecuente, pasar de una intensidad de captura de 0 a C_{max} de manera frecuente durante cortos periodos de tiempo se traduce en el mundo real al hecho de salir a terreno múltiples veces, lo cual implica a su vez gastos extras que afectan la utilidad real asociada al problema.

A diferencia de este comportamiento, el control que propone BOCOP se traduce en salir a capturar durante un corto periodo de tiempo, esperar, y luego capturar durante un largo periodo hasta que se acaben los camarones, que se traduce en dos salidas a terreno intensas, que abaratan a su vez los costos.

• ¿Y si fuera otro problema, bajo qué criterios escogería uno o el otro?

R: Los criterios que se considerarían para otro problema de naturaleza similar, sería tomar en cuenta el valor que toman las funciones objetivos, y además considerar los gastos que implica la estructura del control, pues dependiendo de la situación, un control de tipo Bang-Bang que se activa reiteradas veces durante cortos periodos de tiempo podría no ser implementable, como es el caso en que el control significa la intensidad de captura.

3. Parte B. Ecuaciones de Hamilton-Jacobi-Bellman

Consideremos el siguiente sistema dinámico controlado no lineal:

$$\min J(u(\cdot)) = \int_0^1 (x(t)u(t))^2 dt + x(1)^2; \quad \text{s.a.} \quad \dot{x}(t) = -x(t)u(t), \quad x(0) = 1.$$

Asumiremos que la solución a la ecuación de HJB asociada al problema es cuadrática, i.e.

$$V(t,x) = a(t)x^2 + b(t)x + c(t)$$

para ciertas funciones $a, b, c : \mathbb{R}_+ \to \mathbb{R}$ a determinar más adelante.

3.1. Ejercicio 7

Se escriben la ecuación de HJB del problema, considerando $V(t,x)$ anteriormente presentado.

El Hamiltoniano asociado al problema se escribe como

$$H(x,u,p) = x^2u^2 - pxu$$

Y la ecuación de HJB para la función valor $V(\cdot)$ se escribe una vez identificados los términos $\lambda = 0, g(e[x]) = x(t_f)^2$ como

$$\frac{\partial V}{\partial t} + \inf_{u\in\mathbb{R}} H(x,u,\frac{\partial V}{\partial x}) = 0$$

$$V(T,x) = x^2$$

Reemplazando se obtiene la ecuación diferencial parcial para $V(\cdot)$

$$\frac{\partial V}{\partial t} + \inf_{u\in\mathbb{R}} [x^2u^2 - \frac{\partial V}{\partial x}xu] = 0$$

$$V(T,x) = x^2$$

Notamos que la expresión

$$\inf_{u\in\mathbb{R}} [x^2u^2 - \frac{\partial V}{\partial x}xu]$$

Es convexa como función de u, pues es suma de una parte convexa (x^2u^2) con una parte lineal ($-\frac{\partial V}{\partial x}xu$), luego derivando con respecto de u e igualando a 0 se encuentra el u que realiza el ínfimo

$$\implies 2ux^2 - \frac{\partial V}{\partial x}x = 0$$

$$\implies u = \frac{\partial V}{\partial x}\frac{1}{2x}$$

Reemplazando entonces dicho u en el verdadero Hamiltoniano se obtiene

$$\inf_{u\in\mathbb{R}}[x^2u^2 - \frac{\partial V}{\partial x}xu] = x^2(\frac{\partial V}{\partial x})^2\frac{1}{4x^2} - \frac{\partial V}{\partial x}\frac{\partial V}{\partial x}\frac{x}{2x}$$

Luego, la ecuación de HJB se ve como

$$\frac{\partial V}{\partial t} + \frac{1}{4}(\frac{\partial V}{\partial x})^2 - \frac{1}{2}(\frac{\partial V}{\partial x})^2 = 0$$

Equivalentemente se tiene

$$\frac{\partial V}{\partial t} = \frac{1}{4}(\frac{\partial V}{\partial x})^2$$
$$V(T,x) = x^2$$

Suponiendo que $V(\cdot)$ es de la forma

$$V(t,x) = a(t)x^2 + b(t)x + c(t)$$

Tenemos de la condición de término

$$V(T,x) = a(T)x^2 + b(T)x + c(T) = x^2$$
$$\implies a(T) = 1\ , b(T) = 0, c(T) = 0$$

Computando $\frac{\partial V}{\partial t}$ y $\frac{\partial V}{\partial x}$ para $V(\cdot)$ con la forma anterior.

$$\frac{\partial V}{\partial t} = a'(t)x^2 + b'(t)x + c'(t)\ ,\ \frac{\partial V}{\partial x} = 2a(t)x + b(t)$$

$$\implies (\frac{\partial V}{\partial x})^2 = 4a(t)^2x^2 + 4a(t)b(t)x + b(t)^2$$

Del desarrollo anterior tenemos que $V(\cdot)$ cumple la ecuación diferencial

$$a'(t)x^2 + b'(t)x + c'(t) = a(t)^2x^2 + a(t)b(t)x + \frac{b(t)^2}{4} \quad \forall t \in [0, t_f] \quad \forall x \in \mathbb{R}$$

De donde se deduce que las funciones $a(t), b(t), c(t)$ deben cumplir la ecuación diferencial junto a las condiciones de término antes obtenidas

$$\begin{cases} a'(t) = a(t)^2 \ , \ a(T) = 1 \\ b'(t) = a(t)b(t) \ , \ b(T) = 0 \\ c'(t) = \frac{b(t)^2}{4} \ , \ c(T) = 0 \end{cases}$$

Consideramos entonces las funciones asociadas a la dinámica inversa

$$\hat{a}(t) = a(T-t) \ , \ \hat{b}(t) = b(T-t) \ , \ \hat{c}(t) = c(T-t) \ ,$$

Se tiene que la dinámica inversa cumple la ecuación diferencial con condiciones iniciales

$$\begin{cases} \hat{a}'(t) = -a(t)^2 \ , \ \hat{a}(0) = 1 \\ \hat{b}'(t) = -a(t)b(t) \ , \ \hat{b}(0) = 0 \\ \hat{c}'(t) = -\frac{b(t)^2}{4} \ , \ \hat{c}(0) = 0 \end{cases}$$

Se utiliza el método *solve ivp* para resolver la ecuación y a partir de las soluciones de $(\hat{a}(t), \hat{b}(t), \hat{c}(t))$ encontrar las funciones $(a(t), b(t), c(t))$

Código 3: Solución de la dinámica inversa de a(t), b(t) y c(t)

```
# Inicialización
x0 = 1
T = 1
nt = 100
t_eval_b = np.linspace(0, T, nt)
a0, b0, c0 = 1, 0, 0

# Dinámica sistema inverso
def F2(t, X):
    a, b, c = X

    da = -a**2
    db = -a*b
    dc = -(b**2)/4

    return np.array([da, db, dc])

res2 = solve_ivp(F2, (0,1), np.array([a0, b0, c0]), t_eval=t_eval_b)
```

Código 4: Obtención de funciones a(t), b(t) y c(t)

```
# Entrega solución del sistema inverso
a_til, b_til, c_til = res2['y']
a_t, b_t, c_t = interp1d(t_eval_b, a_til), interp1d(t_eval_b, b_til), interp1d(t_eval_b,
    ↪ c_til)

# Entrega funciones a(t), b(t) y c(t) a partir del sistema inverso
a_fun, b_fun, c_fun = lambda t: a_t(T-t), lambda t: b_t(T-t), lambda t: c_t(T-t)

# Entrega función valor a partir de las funciones (a(t), b(t), c(t))
V = lambda t, x: a_fun(t)*x**2 + b_fun(t)*x + c_fun(t)
Vx = lambda t, x: 2*a_fun(t)*x + b_fun(t)
Gamma = lambda x, p: p/(2*x)
```

3.2. Ejercicio 8

Se implementa la función tipo feedback que entregue el valor del control óptimo

$$u^*(\cdot) = \Gamma(x(\cdot), \partial x V(\cdot, x(\cdot)))$$

donde $\Gamma(x,p) := \text{argmin}_{w\in U}\{H(t,x,w,p)\}$. Puesto que

$$H(x,u,p) = x^2u^2 - pxu$$

Entonces tenemos que Γ se escribe como

$$\Gamma(x,p) := \text{argmin}_w\{x^2w^2 - pxw\}$$

Y como $H(x,w,p)$ es convexa para w, se tiene que el único valor de w que alcanza el mínimo satisface

$$2x^2w - px = 0 \implies w = \frac{p}{2x}$$

Por lo tanto

$$\Gamma(x,p) = \frac{p}{2x}$$

Código 5: Solución del problema a partir de control tipo feedback

```
# Dinámica de la función de estado como feedback
def F_opt(t, x):
    return -x*Gamma(x, Vx(t, x))

res_opt = solve_ivp(F_opt, (0,1), y0=np.array([x0]), t_eval=t_eval_b)
x_arr = res_opt["y"] # Solución óptima para trayectoria
x_opt = interp1d(t_eval_b, x_arr) # Entrega trayectoria óptima interpolada
u_opt = lambda t: Gamma(x_opt(t), Vx(t, x_opt(t))) # Construye el control óptimo
x, u = x_opt(t_eval_b).reshape(len(t_eval_b)), u_opt(t_eval_b).reshape(len(t_eval_b))
```

De donde se obtienen los siguientes resultados para la trayectoria óptima del problema

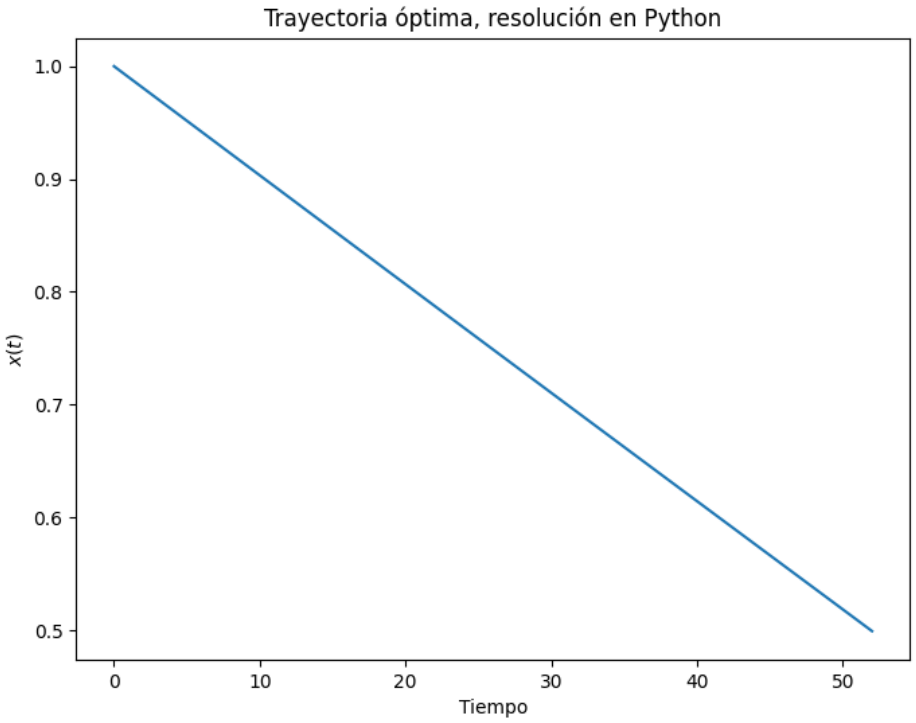

Figura 10: Trayectoria óptima del problema de control

Notamos que la trayectoria óptima tiene forma lineal, que resulta interesante para efectos del modelado.

Por otro lado, graficando el control óptimo feedback asociado que se obtiene a partir de la trayectoria óptima

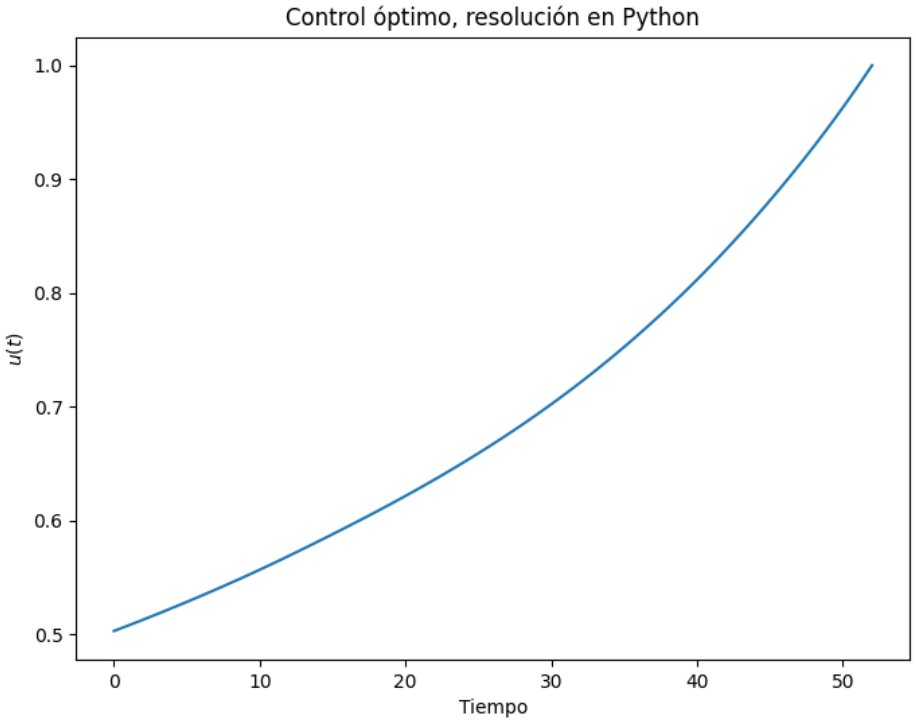

Figura 11: Control óptimo asociado al problema

En donde notamos un comportamiento no lineal para el control durante el tiempo de simulación

3.3. Ejercicio 9

Nuevamente se convertirá al problema en uno puramente Mayer llamando

$$z(t) = \int_0^t (x(s)u(s))^2 ds$$

Que satisface el problema

$$\dot{z}(t) = (x(t)u(t))^2; \quad z(0) = 0$$

Ahora el problema se escribe

$$\min J(u(\cdot)) = z(1)+x(1)^2; \quad \text{s.a.} \quad \dot{x}(t) = -x(t)u(t),\ \dot{z}(t) = (x(t)u(t))^2 \quad x(0) = 1,\ z(0) = 0.$$

Que ahora es resoluble en BOCOP, con lo que se utilizan 100 pasos temporales y el método Gauss II (implícito, 2 etapas, orden 4).

Resuelto el problema, se exportan los datos de BOCOP en Python para realizar una comparativa entre las soluciones, a continuación se grafica la trayectoria óptima obtenida en BOCOP para el problema de control

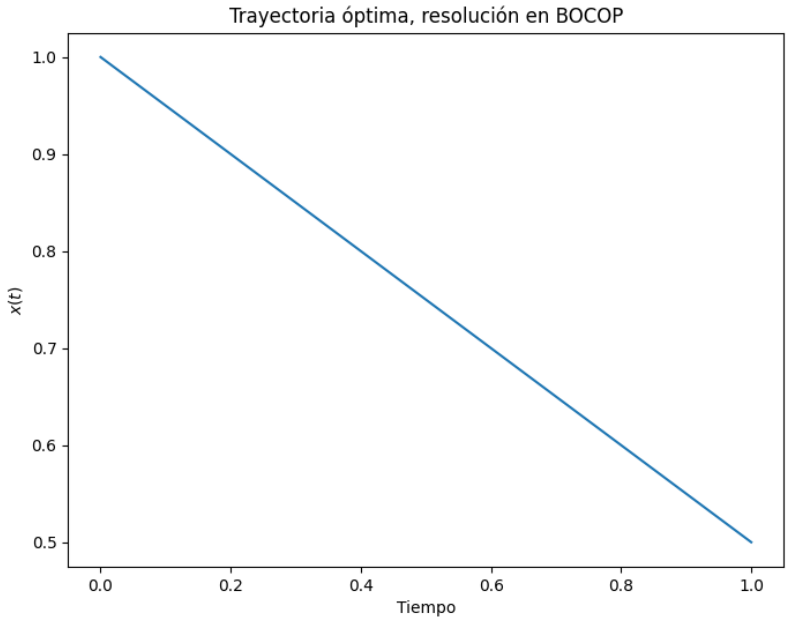

Figura 12: Trayectoria óptima obtenida en BOCOP

Notamos a simple vista que la solución coincide con aquella obtenida en Python.

Graficando el control óptimo asociado se tienen los siguientes resultados

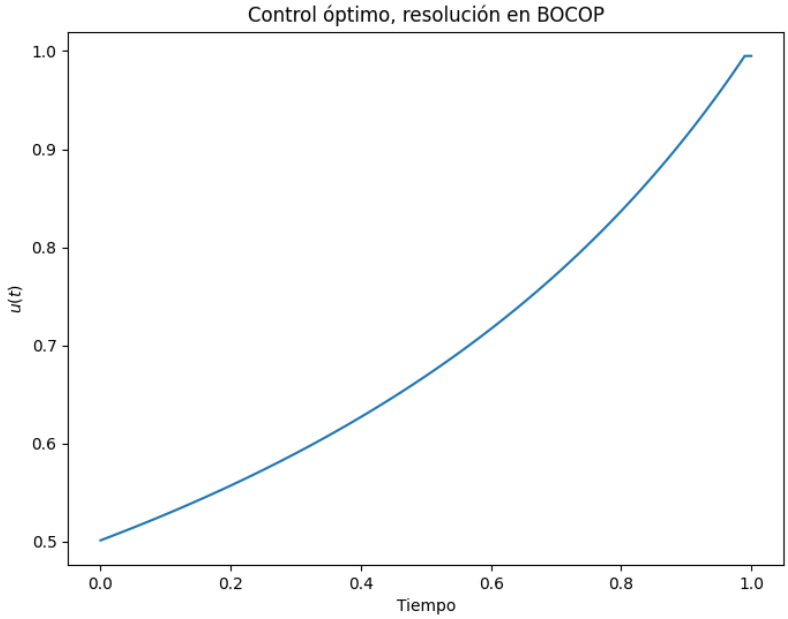

Figura 13: Control óptimo obtenido en BOCOP

El cual coincide también con la solución obtenida en Python, se procede a graficar las comparativas entre las soluciones para verificar la similitud que se tiene al resolver el problema de control con estos dos métodos distintos

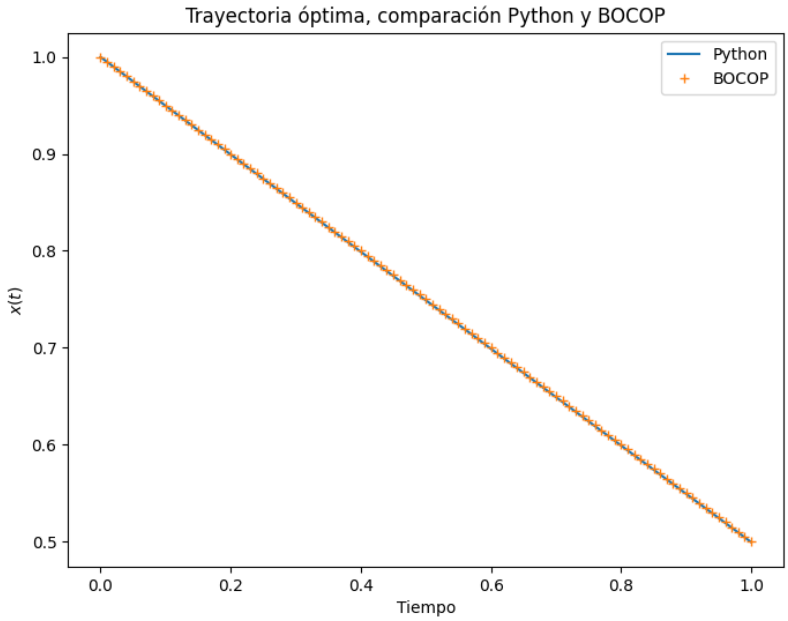

Figura 14: Comparativa entre la trayectoria óptima obtenida en Python y BOCOP

Graficando la diferencia de las trayectorias se obtienen los resultados

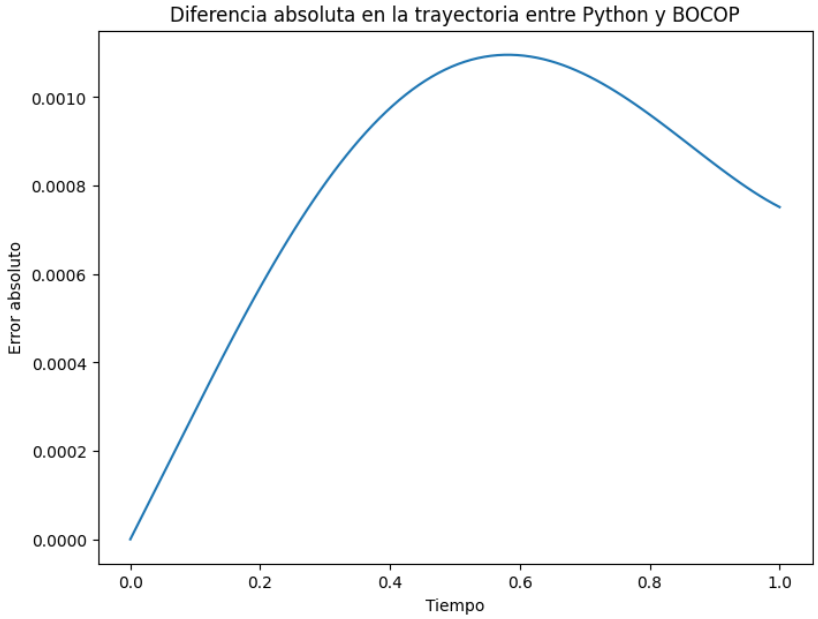

Figura 15: Diferencia absoluta entre las trayectorias obtenidas en Python y BOCOP

En donde se aprecia una diferencia máxima del orden de 10^{-3}
Realizando el mismo proceso con el control óptimo se obtienen los resultados

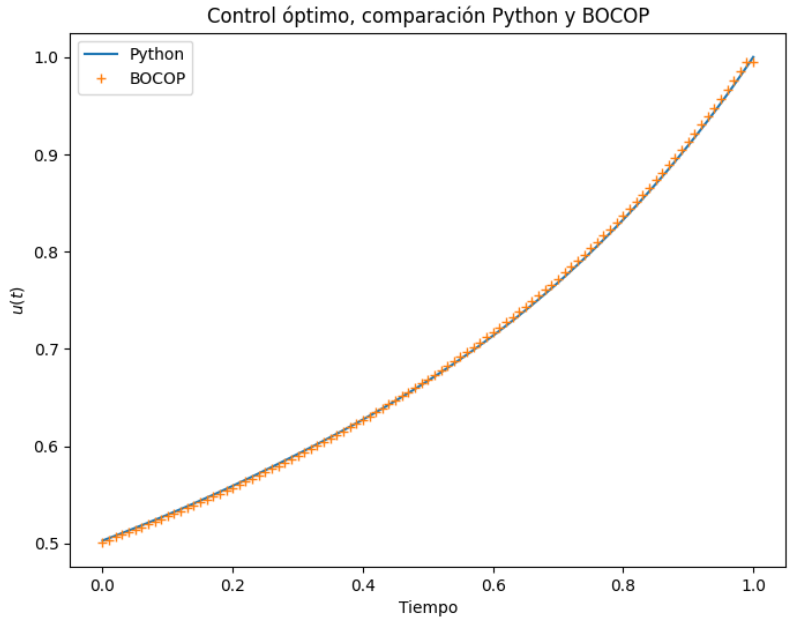

Figura 16: Comparativa entre los controles óptimos obtenidos en Python y BOCOP

Finalmente se comparan las funciones objetivo a lo largo del tiempo para ambas soluciones

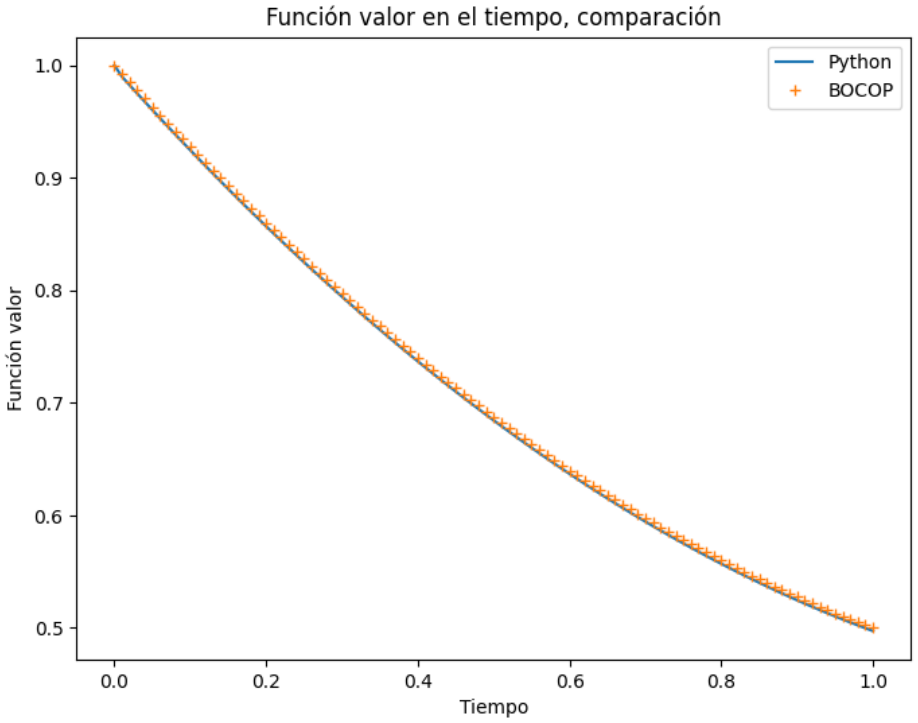

Figura 17: Comparativa entre las funciones objetivos a lo largo del tiempo obtenidas en Python y BOCOP

Concluyendo así que los métodos realizados en Python y en BOCOP convergen a la misma solución para el problema de control óptimo.

4. Conclusiones

En la parte A se logró implementar de manera parcial el método de tiro para obtener las condiciones iniciales de los estados adjuntos y así resolver el problema de control óptimo. Esto ya que el optimizador de Python no logró llegar a un cero de la función que dadas las condiciones iniciales de los estados adjunto entrega las condiciones finales, tarea que es equivalente a lo que se desea lograr. Esto fue probado con diferentes condiciones iniciales para el algoritmo de búsqueda de raíces, pero no tuvo buenos resultados, obteniendo que la función llegaba a un valor del orden de 10^{20}. Esto se puede deber a qué el problema de optimización es difícil en la práctica y probablemente el método de tiro no es adecuado en este contexto, considerando las escalas del problema y la no linealidad.
Esto es lo que se concluye luego de mirar los resultados obtenidos por BOCOP, los que son satisfactorios, ya que en efecto logran lo deseado y entregan algo coherente con para el problema que buscaba resolver.

Para la parte B se tiene todo lo contrario, ya que para los ejercicios entregados se obtiene de manera satisfactoria un control y trayectorias óptimas que además coinciden con la de BOCOP, lo que hace pensar que el método utilizado es eficaz. Ahora, este problema es mucho menos complejo que el anterior y además la función valor se obtiene en base a un ansatz sencillo y que da ecuaciones diferenciales ordinarias para los coeficientes temporales sencillas de resolver y que numéricamente no tienen ningún reto mayor.
En cuanto a los logros, en ambas partes del laboratorio se pudo implementar tanto lo pedido en Python como en BOCOP, comparando ambos, tanto en las trayectorias óptimas, controles óptimos y errores, que era lo pedido por el enunciado entregado.