Ingeniería Matemática
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

Reporte

Laboratorio 5

Arturo Lazcano
Sebastián Tapia

28 de noviembre de 2022

Ejercicio 1

Antes de implementar el método de tiro en Python, se buscó una expresión para el control óptimo de este problema utilizando el principio del máximo de Pontryagin.

Por enunciado, $\dot{x} = x(t)u(t)$ con la condición inicial $x(0) = x_0 > 0$. También se define $c(t) = (1 - u(t))x(t)$ a tiempo final fijo $T > 1$ y $u(t) \in [0, 1]$. Así, el problema resulta ser:

$$\max_{u(\cdot)} \int_0^T c(t)dt,$$

o equivalentemente,

$$-\min_{u(\cdot)} \int_0^T -c(t)dt.$$

Para utilizar el PMP, se verá el Hamiltoniano

$$H(t, x(t), u(t), p(t)) = l(x, u) + p(t)f(x, u) = (u(t) - 1)x(t) + p(t)x(t)u(t)$$

Así, la dinámica del sistema adjunto viene dada por (simplificando la notación) $\dot{p} = -\partial_x H = 1 - u + pu$ con la condición final $p(T) = 0$.

Luego, para caracterizar el control óptimo, notar que, $\text{argmin}_u[x(u - 1) + pxu] = \text{argmin}_u[ux(1 + p)]$ pues solo interesa los términos que incluyan a $u(t)$.

Por lo tanto, para minimizar la expresión $ux(1 + p)$, se analiza el signo de $x(1 + p)$. Sin embargo, como $\dot{x} = x(t)u(t)$ (lineal en x y $u(t) \geq 0$), al resultar una solución de x del tipo exponencial y con $x(0) = x_0 > 0$, entonces $x(t) > 0$ para todo t. Por lo que realmente interesa el signo de $1 + p$. Entonces,

$$u(t) = \begin{cases} 1, & 1 + p(t) < 0 \\ 0, & 1 + p(t) > 0 \\ \tilde{u}, & 1 + p(t) = 0 \end{cases}$$

Donde $\tilde{u}$ es libre. Se verá ahora que no se puede cumplir $1 + p = 0$. Sea $\phi(t) = 1 + p(t)$. Entonces, $\dot{\phi}(t) = \dot{p}(t) = 1 - u + pu = 1 - u(1 + p) = 1 - u\phi$.

$$\implies \dot{\phi}(t) = \begin{cases} 1, & 1 + p(t) > 0 \\ 1, & 1 + p(t) = 0 \\ -p, & 1 + p(t) < 0 \end{cases}$$

Pero en el caso que $1 + p < 0$, entonces $p < -1 \Leftrightarrow p > 1$. Por lo que se concluye que $\dot{\phi} > 0$. Debido a lo anterior, si la trayectoria pasa por el cero, lo hará solo una vez. Así, el control óptimo del problema resulta:

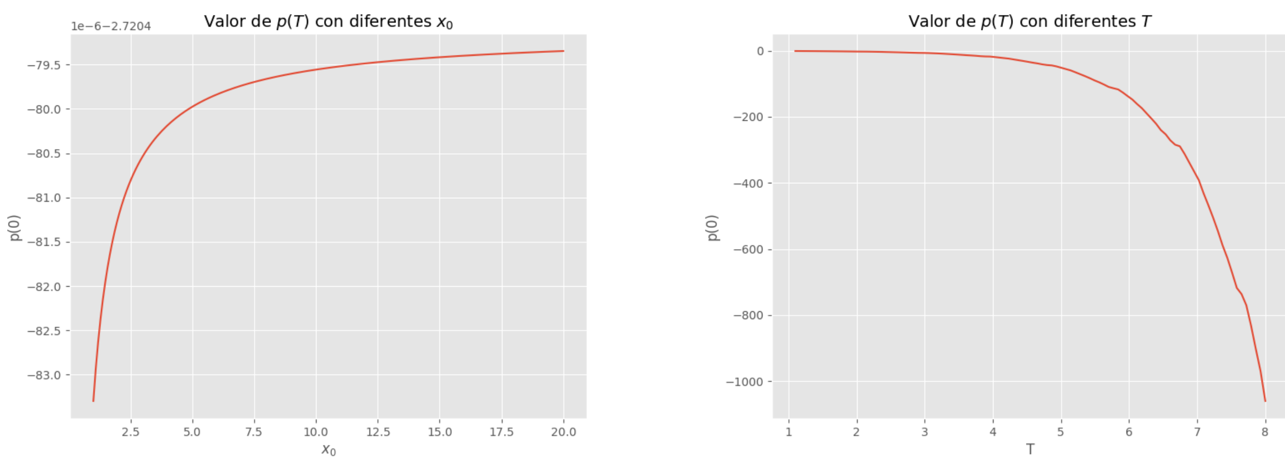
$$u(t) = \begin{cases} 1, & 1 + p(t) < 0 \\ 0, & 1 + p(t) \geq 0. \end{cases}$$

Método de tiro

Dado el control óptimo, se desea resolver la ecuación diferencial para p. Para esto se utilizó el método de tiro, que para este caso consiste en encontrar el valor $p(0)$ de modo que $p(T) = 0$. Se define $R(p_0, x_0, T)$ una función que resuelve la EDO para x y el estado adjunto y retorna el valor de $p(T)$. Se cumple que las raíces de R corresponden a los $p(0)$ buscados.

Se exploraron la relación que hay entre $p(0)$ y T, x_0. En primer lugar, dado que $\dot{p}$ no depende de x en ningún instante, es esperable que los valores de $p(0)$ no varíen para x_0. En el caso de T puede haber una dependencia dado que para T's grandes el consumo total se puede maximizar aún más y por tanto la variable adjunta tendría que partir de puntos más chicos.

A continuación se muestran gráficos de los $p(0)$ obtenidos para varios x_0 y varios T:



(a) Variación de $p(0)$ en función de x_0.

(b) Variación de $p(0)$ en función de T.

Figura 1: Variación de $p(0)$ con x_0 y T

Se observa un crecimiento logaritmico para el caso de x_0. Sin embargo, el resultado es engañoso pues la escala en el eje y es de 10^{-6}. Más aún, la diferencia entre el mayor valor y el menor valor obtenido es de 3.95×10^{-6}, que es una diferencia despreciable y se puede concluir que x_0 no altera los resultados. Para el caso de T se observa un decrecimiento exponencial, donde a mayores valores de T, $p(0)$ decrece hacia menos infinito.

Utilizando el método de tiro y la expresión encontrada para el control óptimo, se obtuvieron, para todas las combinaciones posibles entre $x_0 = 5, x_0 = 20, T = 2$ y $T = 5$, los siguientes resultados :

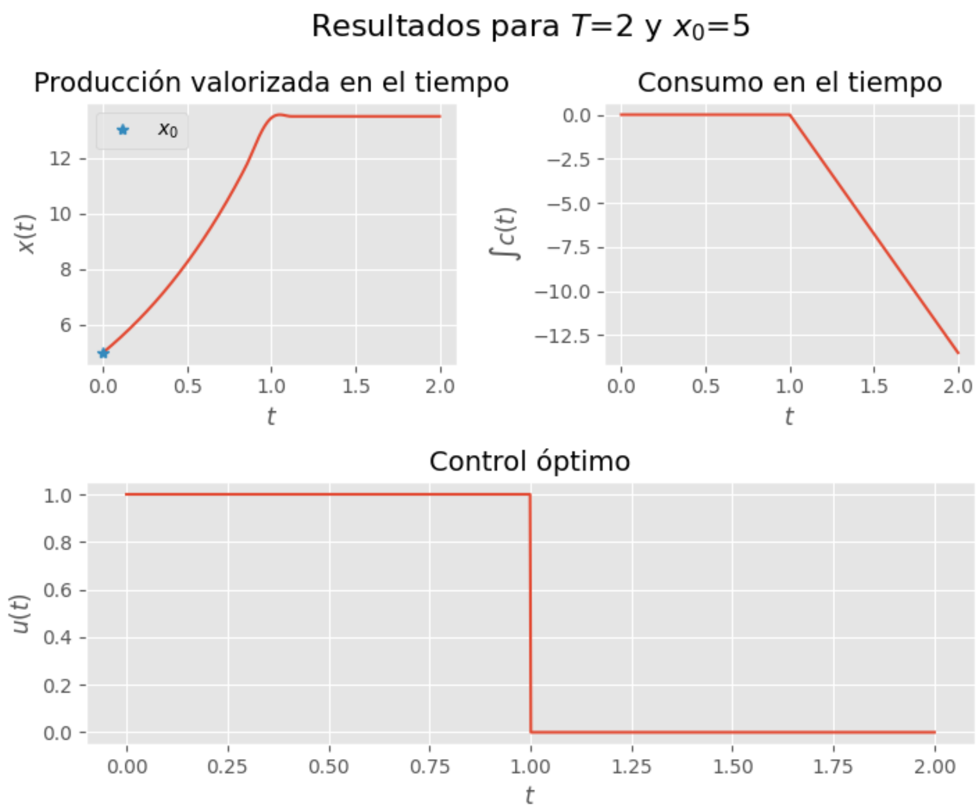


Figura 2: Trayectoria, consumo hasta t y control óptimo para $T = 2$ y $x_0 = 5$.

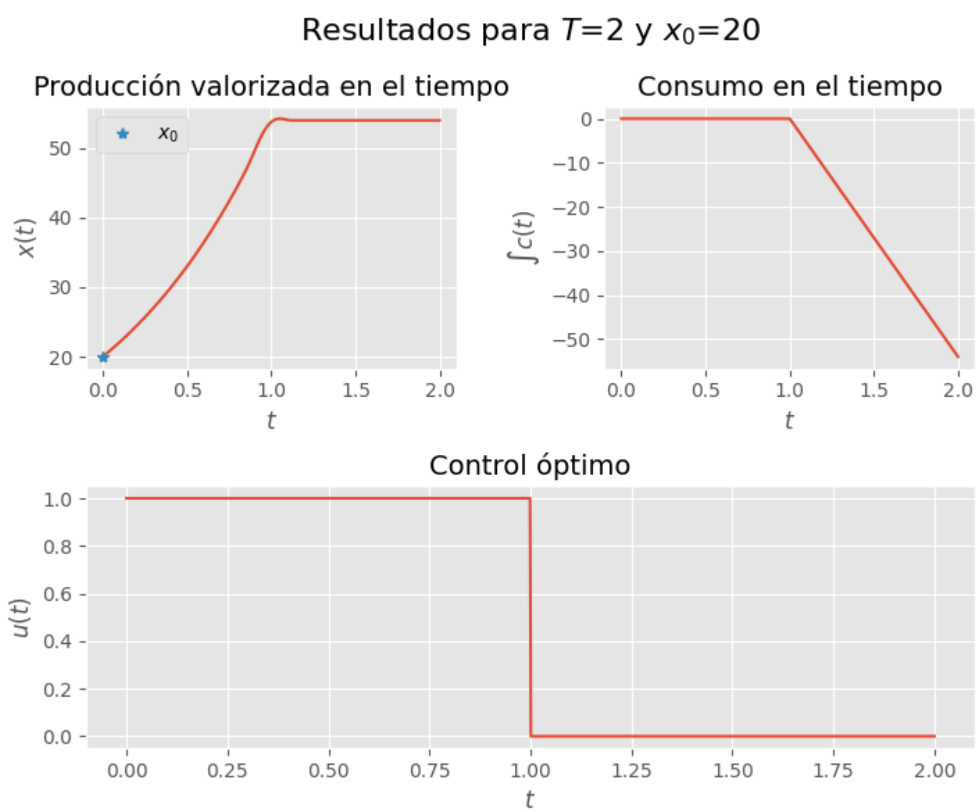


Figura 3: Trayectoria, consumo hasta t y control óptimo para $T = 2$ y $x_0 = 20$.

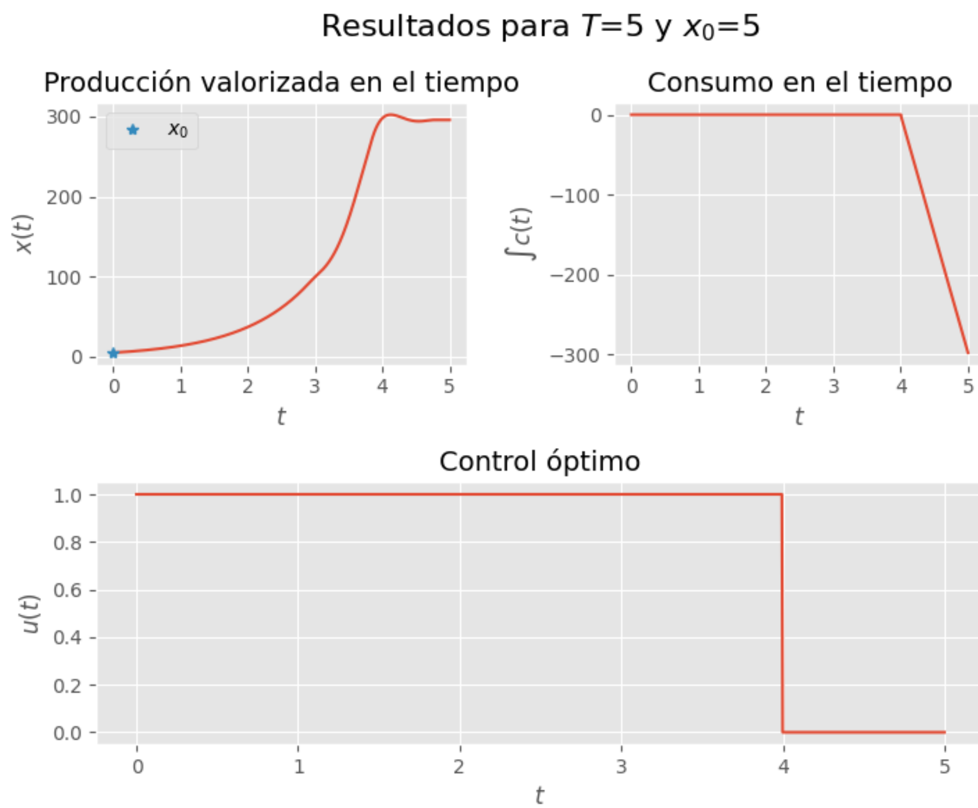


Figura 4: Trayectoria, consumo hasta t y control óptimo para $T = 5$ y $x_0 = 5$.

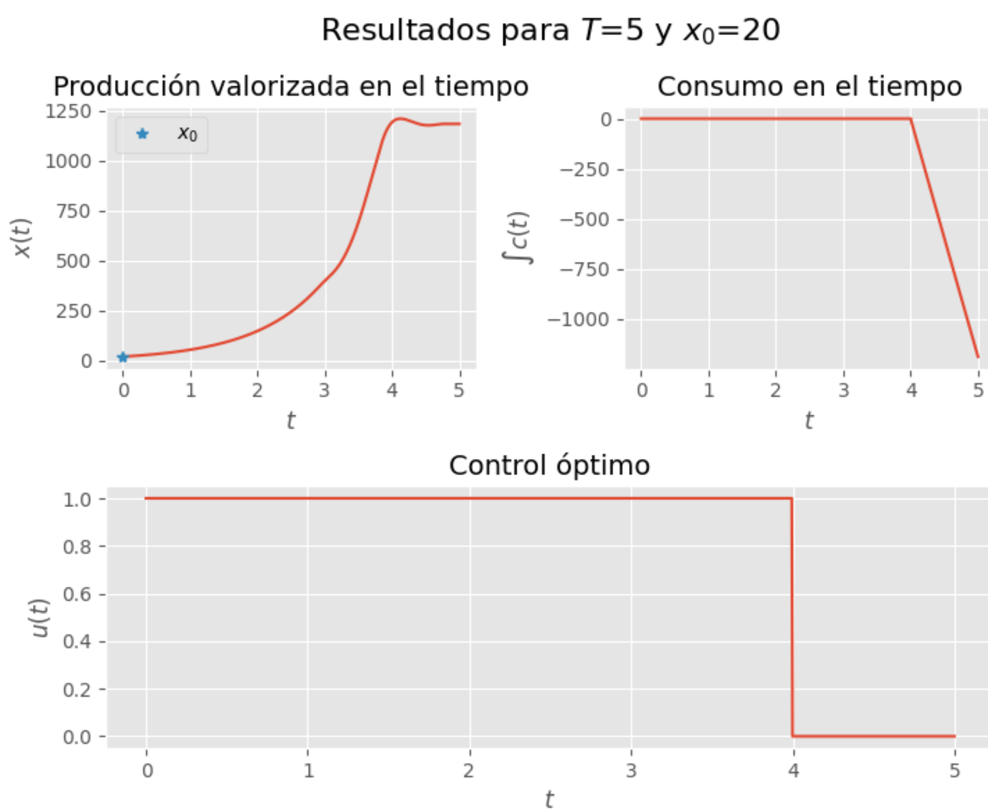


Figura 5: Trayectoria, consumo hasta t y control óptimo para $T = 5$ y $x_0 = 20$.

Ejercicio 2

El objetivo de esta pregunta es verificar computacionalmente que la función $V(s, y)$ definida como

$$V(s,y) = \begin{cases} ye^{-s+T-1}, & (s,y) \in [0, T-1) \times (0, \infty) \\ y(T-s), & (s,y) \in [T-1, T] \times (0, \infty), \end{cases}$$

cumple la ecuación de Hamilton-Jacobi-Bellman dada por:

$$\partial_t V(s,y) + y + y \max_{w \in [0,1]} \{w(\partial_x V(s,y) - 1)\} = 0, \quad V(T,y) = 0.$$

Para realizar esto, se programan las derivadas con respecto a t y x manualmente en Python y así, tomando una grilla en cada una de las componentes anteriores y evaluando V, se obtiene un arreglo de soluciones. El criterio para verificar que se cumple la ecuación anterior fue tomar valor absoluto a la lista de soluciones y retornar el valor máximo. Así, si ese máximo es cercano a 0, todos los otros valores también lo serán.

A continuación, se ven distintos gráficos en donde cambia la grilla de y, esto pues en la función $V(\cdot)$ la variable y puede tomar valores en $(0, \infty)$ por lo que se simula ese intervalo con uno del estilo $[y_{\min}, y_{\max}]$. En la ejecución, se usa $y_{\min} = 0.001$. Se prueba para $T = 2$ y $T = 5$.

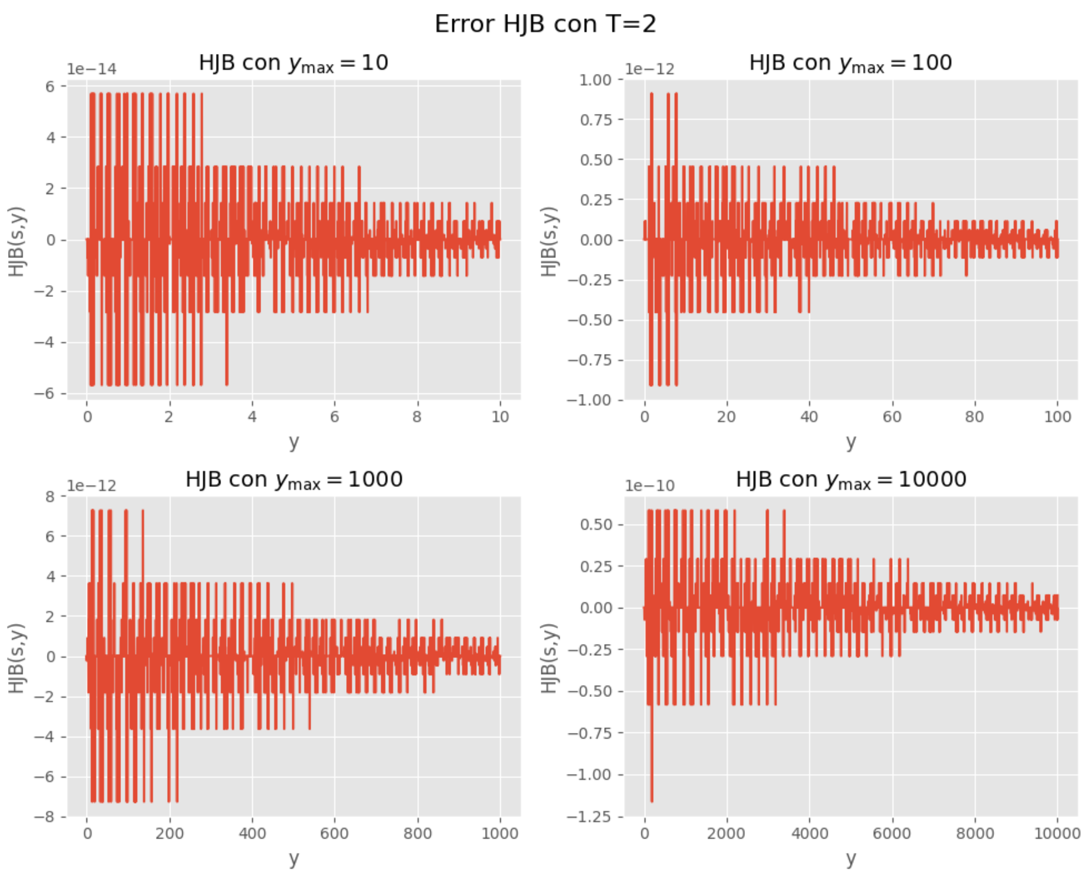


Figura 6: Error para $T = 2$.

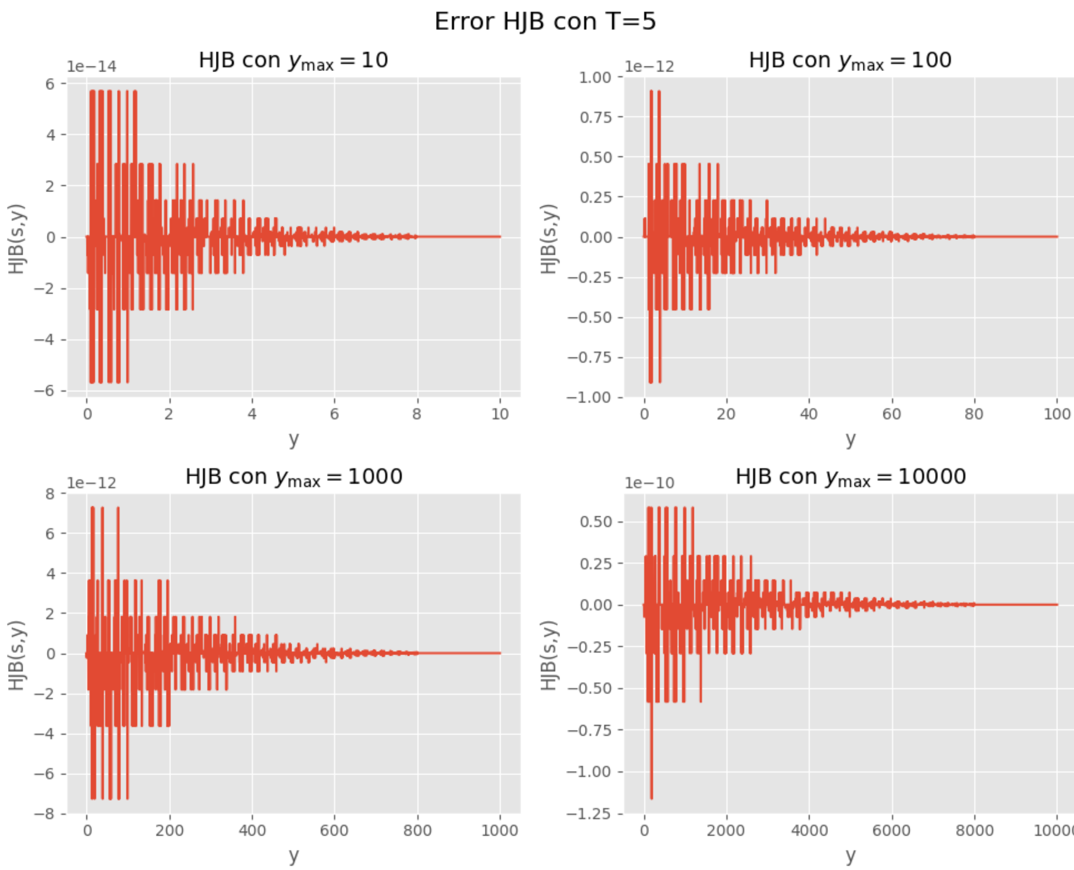


Figura 7: Error para $T = 5$.

Se observa que, para ambos casos y para todo valor de la grilla implementada para s, y, la función $V(s, y)$ cumple con la ecuación de Hamilton-Jacobi-Bellman, esto pues cada simulación del lado izquierdo de la ecuación es bastanta cercano al cero (del orden de 10^{-12}), inclusive su máximo, por lo que se concluye que la función cumple HJB.

Ejercicio 3

Se tiene que $H(x, w, \partial_x V) = (w-1)x + \partial_x V x w = wx(1 + \partial_x V) - x$. Luego, el control feedback dado queda como

$$\begin{aligned} u &= \underset{w\in[0,1]}{\arg\min}\{H(x, w, \partial_x V)\} = \underset{w\in[0,1]}{\arg\min}\{wx(1+\partial_x V) - x\} \\ &= \underset{w\in[0,1]}{\arg\min}\{wx(1+\partial_x V)\}, \end{aligned}$$

donde en el último paso se eliminó x porque no depende de w. Además, en el ejercicio 1 se obtuvo que $x > 0$, por lo que no influirá en el cálculo de u y se obtiene que $u = \underset{w\in[0,1]}{\arg\min}\{w(1+\partial_x V)\}$. Nuevamente se obtiene una función lineal con respecto a w, por lo que el control óptimo puede definirse como

$$u = \begin{cases} 0 & \text{si } 1 + \partial_x V \geq 0 \\ 1 & \text{si } 1 + \partial_x V < 0. \end{cases}$$

Del ejercicio 2 se tiene que $\partial_x V$ no depende de x, por lo que la definición de u se simplifica aún más. Se obtuvieron los siguientes resultados utilizando todas las combinaciones entre $x_0 = 5, x_0 = 20, T = 2$ y $T = 5$:

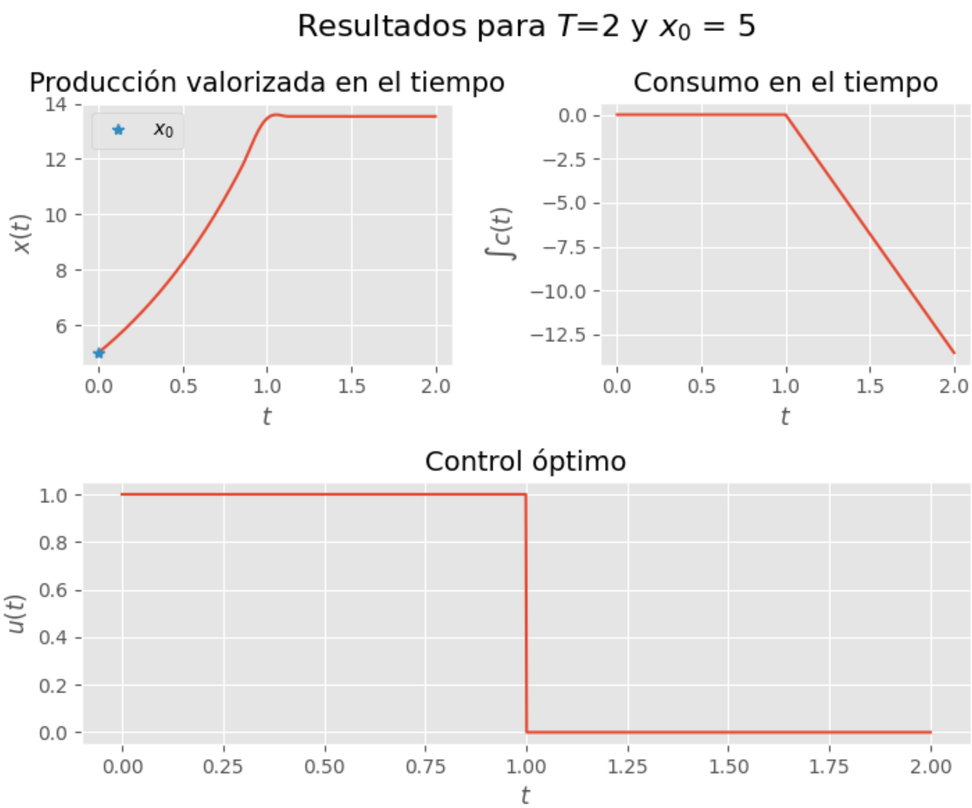


Figura 8: Trayectoria, consumo hasta t y control óptimo para $T = 2$ y $x_0 = 5$.

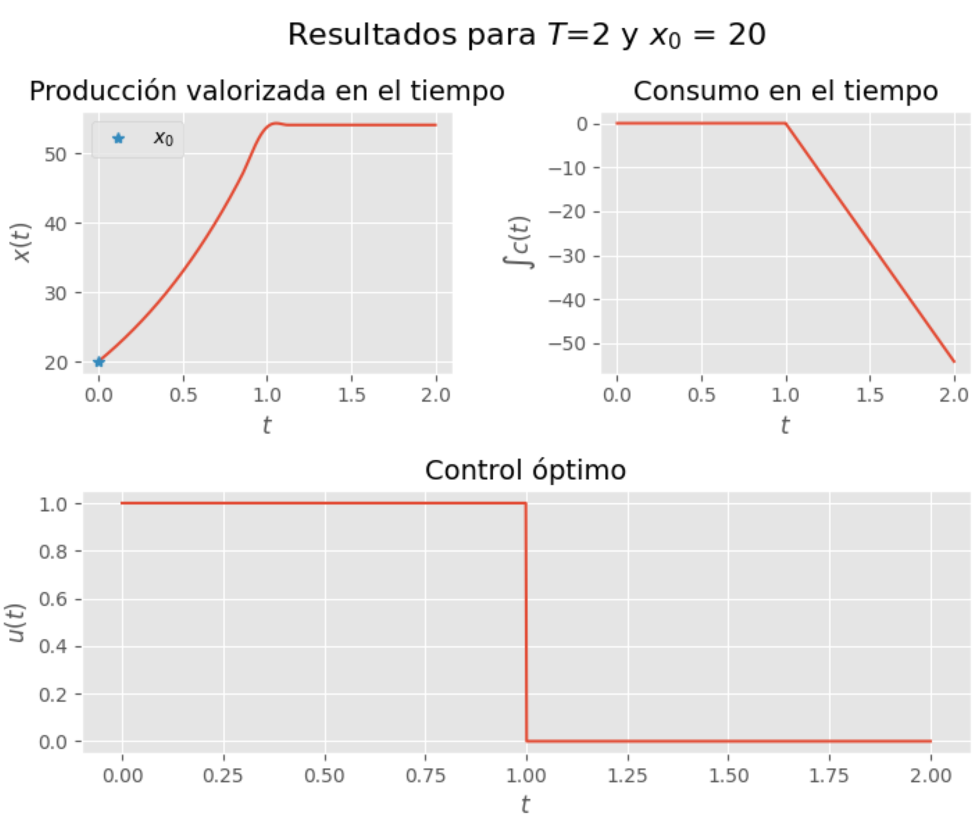


Figura 9: Trayectoria, consumo hasta t y control óptimo para $T = 2$ y $x_0 = 20$.

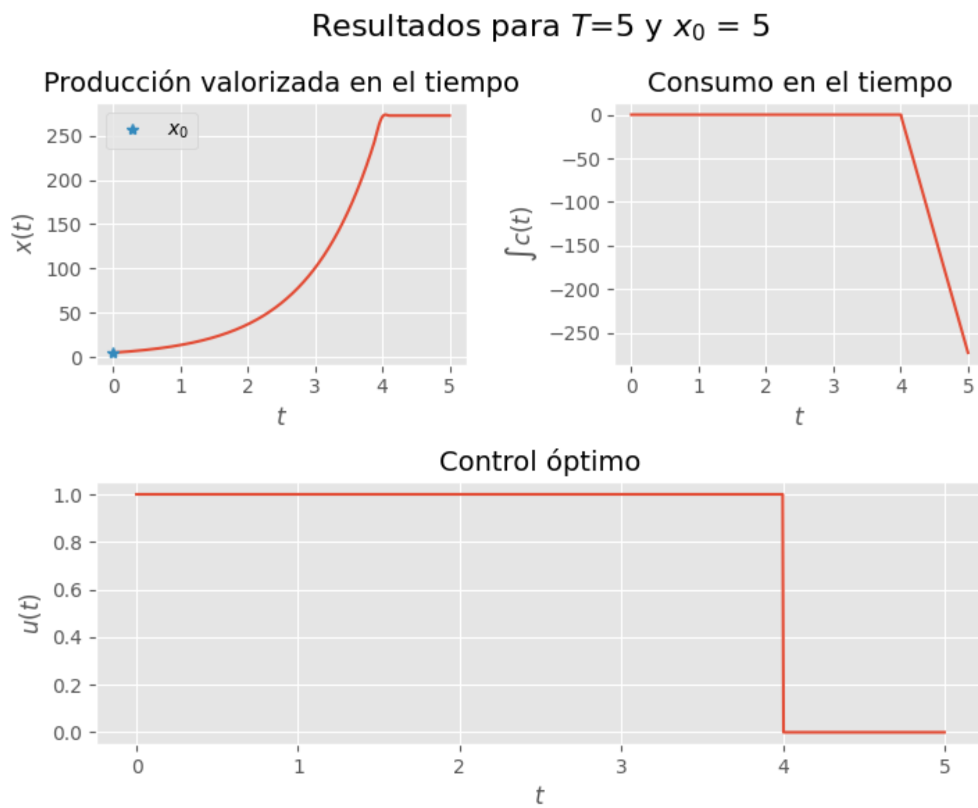


Figura 10: Trayectoria, consumo hasta t y control óptimo para $T = 5$ y $x_0 = 5$.

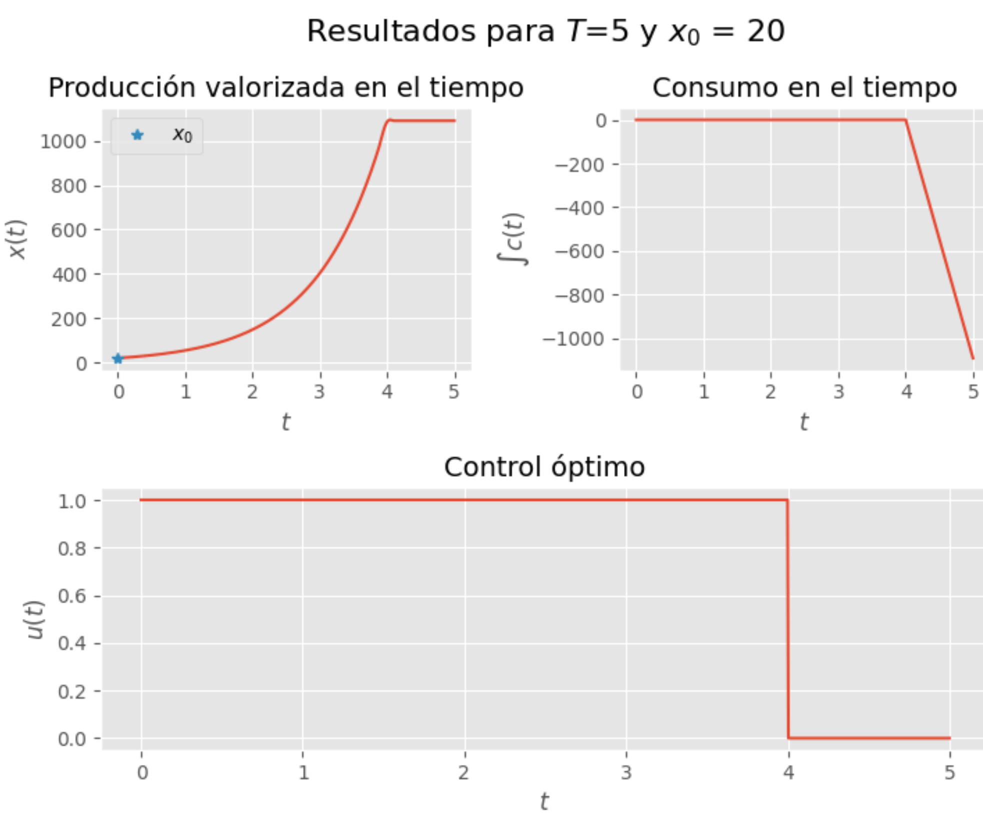


Figura 11: Trayectoria, consumo hasta t y control óptimo para $T = 5$ y $x_0 = 20$.

Los resultados obtenidos se condicen con los resultados del ejercicio 1, siendo bastante similares las trayectorias, control y consumo. El control usado en este ejercicio resuelve el problema dado que en el ejercicio 2 se demostró numéricamente que la función $V(s, y)$ definida resuelve la ecuación de Hamilton-Jacobi-Bellman y, por lo tanto, el control definido como en este ejercicio resulta ser el control óptimo.

Ejercicio 4

A continuación se muestran los resultados que se obtuvieron usando BOCOP con todas las combinaciones entre $x_0 = 5, x_0 = 20, T = 2$ y $T = 5$:

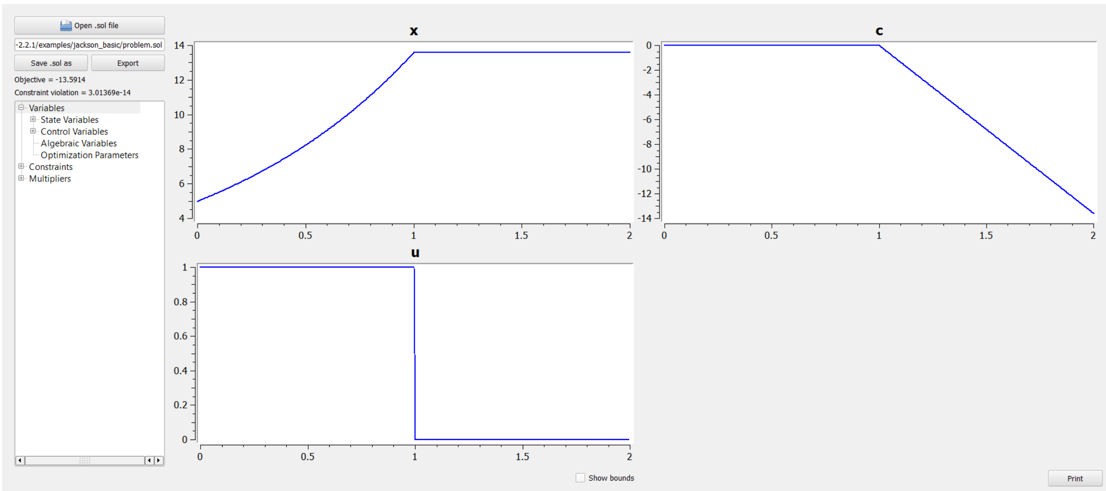


Figura 12: $x_0 = 5$ y $T = 2$.

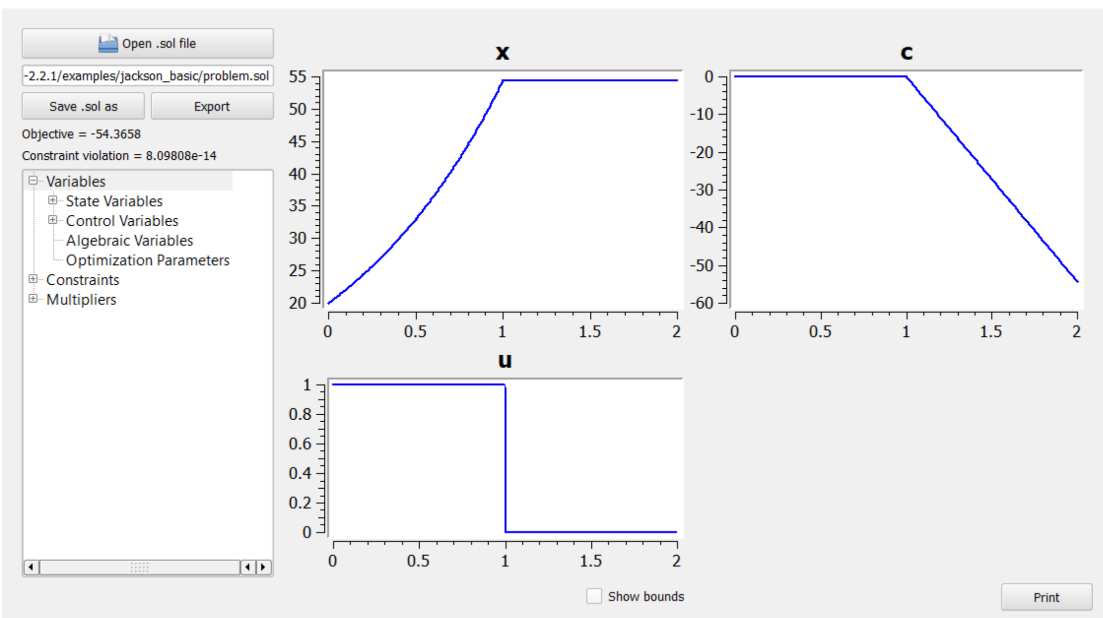


Figura 13: $x_0 = 20$ y $T = 2$.

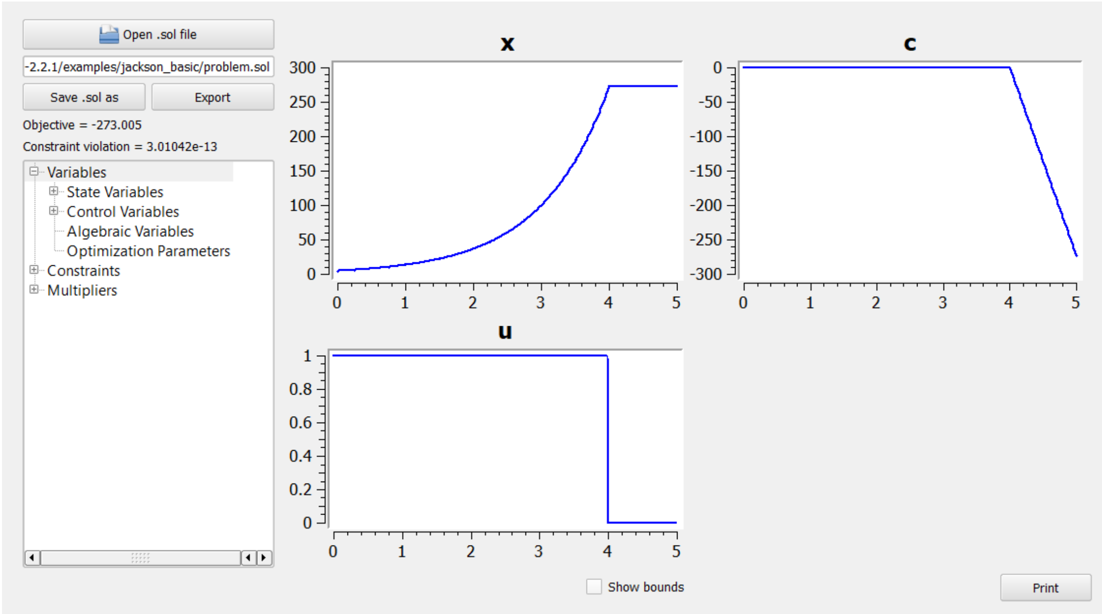


Figura 14: $x_0 = 5$ y $T = 5$.

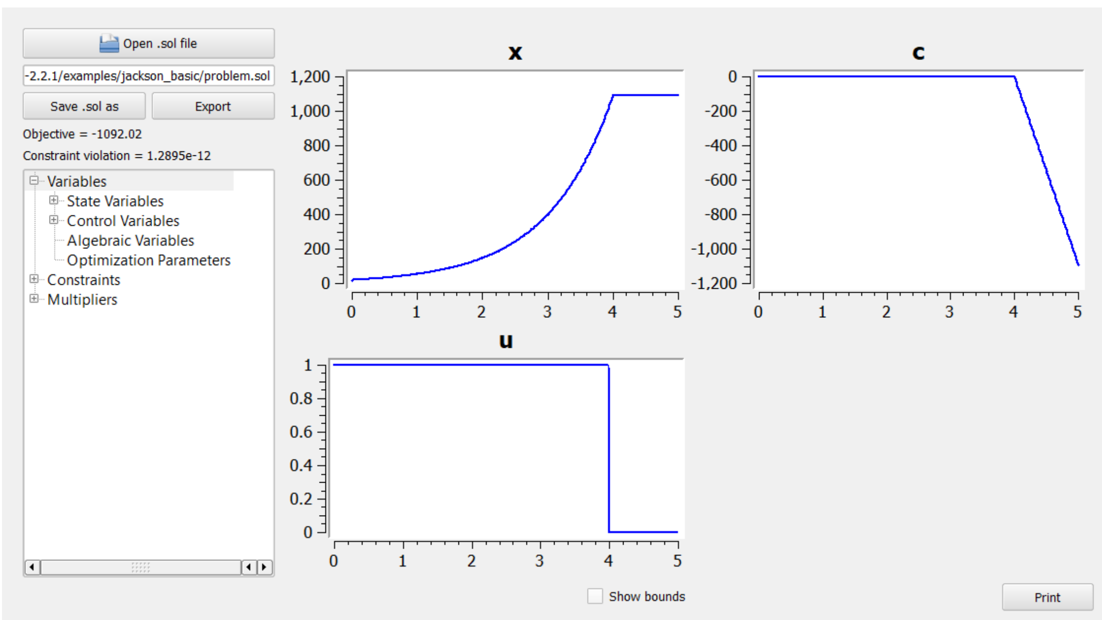


Figura 15: $x_0 = 20$ y $T = 5$.

En las figuras anteriores, se puede ver como el valor objetivo es negativo, donde este valor representa el $min(-f)$ por lo cual para obtener el equivalente al máximo se multiplica por -1.

Ejercicio 5

A continuación, se comparan los métodos implementados junto con el programa de BOCOP para todas las condiciones iniciales y tiempos finales utilizados en los ejercicios anteiores. Estos valores toman en cuenta la capacidad que se tiene en Python y en BOCOP ya que, por ejemplo, $T = 20$ excede el poder computacional de estos programas. Se obtuvieron las siguientes tablas con los valores objetivos obtenidos y tiempos de ejecución de cada método.

Los valores objetivos y tiempos de ejecución para el método de tiro fueron los siguientes:

Parámetros	Valor objetivo	Tiempo de ejecución (s)
$T = 2$, $x_0 = 5$	13.48	0.0014
$T = 2$, $x_0 = 20$	53.95	0.0016
$T = 5$, $x_0 = 5$	297.52	0.0017
$T = 5$, $x_0 = 20$	1190	0.0015

Los valores objetivos y tiempos de ejecución con el control feedback fueron los siguientes:

Parámetros	Valor objetivo	Tiempo de ejecución (s)
$T = 2$, $x_0 = 5$	13.54	0.0034
$T = 2$, $x_0 = 20$	54.16	0.0038
$T = 5$, $x_0 = 5$	272.55	0.0044
$T = 5$, $x_0 = 20$	1090.12	0.0038

Los valores objetivos y tiempos de ejecución en BOCOP fueron los siguientes:

Parámetros	Valor objetivo	Tiempo de ejecución (s)
$T = 2$, $x_0 = 5$	13.59	0.18
$T = 2$, $x_0 = 20$	54.37	0.36
$T = 5$, $x_0 = 5$	273.01	0.5
$T = 5$, $x_0 = 20$	1092.02	0.41

A modo de conclusión, se puede ver que los tres métodos tuvieron un rendimiento similar en las trayectorias y controles óptimos, dado que estos no poseen mayores diferencias en sus gráficos. En los valores objetivos, todos los métodos entregaron resultados similares, con diferencias en las décimas, salvo en los casos $T = 5, x_0 = 5$ y $T = 5, x_0 = 20$ para el método de tiro, donde la diferencia con Feedback y Bocop fue mayor. Este aumento pudo deberse a la dependencia de $p(0)$ con respecto a T en el método de tiro, ya que esta diferencia en los resultados ocurre cuando se aumentó de $T = 2$ a $T = 5$.

Los tiempos de ejecución resultaron ser muy variados entre cada método, habiendo una diferencia significativa entre los métodos de Python y Bocop. Esto puede deberse a la complejidad del problema que resuelve Bocop, ya que se tuvo que definir una variable extra con el término de la integral. Luego, Bocop debe resolver esta integral en varias iteraciones, lo que ralentizaría el proceso. En cambio los métodos de Python solo deben resolver EDO's simples que incurren en pocos tiempos de ejecución.

El uso de condición inicial x_0 pequeña junto con un tiempo final T pequeño se debe a que, para este problema, la complejidad aumenta de manera abrupta, donde tanto BOCOP como Python no terminarán su proceso de ejecución. Se comprueba entonces la eficiencia de los métodos en Python y la verificación por el programa BOCOP.