Indirect Single Shooting

Optimal Control
Project Report

Serkan Burak Örs*

*orss19@itu.edu.tr

Contents

1 Introduction

In this project, the following problem is solved for the state response, costate response and control input using indirect method.

$$\min J = -y(t_f) \tag{1.1}$$

subject to

$$\dot{y} = yu - y - u^2 \tag{1.2}$$

for initial condition $y(0) = 1$ and final time $t_f = 5$.

2 Solution

The given optimal control problem is final state free and final time fixed problem.
The Hamiltonian can be defined for this problem as:

$$\mathcal{H} = p\left(yu - y - u^2\right) = pyu - py - pu^2 \tag{2.1}$$

Costate equation can be determined as

$$\dot{p}(t) = -\frac{\partial \mathcal{H}}{\partial y} = -pu + p = p(1-u) \tag{2.2}$$

Solution of costate equation is in form of

$$p(t) = ce^{(1-u)t} \tag{2.3}$$

Since there is no constraint on control input, necessary condition is

$$\frac{\partial \mathcal{H}}{\partial u} = py - 2pu = 0 \tag{2.4}$$

We need to check if $\partial^2\mathcal{H}/\partial y^2$ is positive definite or not

$$\frac{\partial^2 \mathcal{H}}{\partial u^2} = -2p(t) > 0 \tag{2.5}$$

Hence $\partial^2\mathcal{H}/\partial y^2$ is positive definite where $p < 0$. Hence optimal control input where $p(t) < 0$ is

$$u^*(t) = \frac{1}{2}y(t) \tag{2.6}$$

State response can be determined using the knowledge from Eq.[2.7]. A solution to the differential equation

$$\dot{y} = ay^2 - y \tag{2.7}$$

is

$$y = \frac{1/a}{1 + \beta e^t} \tag{2.8}$$

Eq.[2.7] can be rearranged as

$$\dot{y} + y = ay^2 \tag{2.9}$$

and Eq.[1.2] can be rearranged as

$$\dot{y} + y = yu - u^2 \tag{2.10}$$

Eq.[2.9] and [2.10] can be equated as

$$ay^2 = yu - u^2 \tag{2.11}$$

And finally, we can express variable, a in terms of state,y and control, u as

$$a = \frac{yu - u^2}{y^2} \tag{2.12}$$

We can substitute Eq.[2.12] into Eq.[2.7] to obtain

$$\dot{y}(t) = \left(\frac{yu - u^2}{y^2}\right) y^2 - y \tag{2.13}$$

Solution of Eq.[2.13] can be expressed using the hint from Eq.[2.8] as

$$y(t) = \left(\frac{y^2}{yu - u^2}\right)\left(\frac{1}{1 + \beta e^t}\right) \tag{2.14}$$

After several algebraic manipulations, we can obtain an expression for state, $y(t)$ as

$$y(t) = \frac{1 + \beta e^t}{u(1 + \beta e^t) - 1} u^2 \tag{2.15}$$

By substituting Eq.[2.14] into Eq.[2.6], we can obtain an expresssion for control input, $u(t)$ as

$$u(t) = \frac{1}{2} \frac{1 + \beta e^t}{u(1 + \beta e^t) - 1} u^2 \tag{2.16}$$

and by rearranging Eq.[2.16], we obtain a final expression for control input as

$$u(t) = \frac{1}{1 + \beta e^t} \tag{2.17}$$

State response can be determined by substituting Eq.[2.17] into Eq.[2.15]

$$y(t) = \frac{1 + \beta e^t}{\left(\frac{1}{1 + \beta e^t}\right)(1 + \beta e^t) - 1} \left(\frac{1}{1 + \beta e^t}\right)^2 = \frac{1}{1 - 1}\left(\frac{1}{1 + \beta e^t}\right) = \frac{1}{0}\left(\frac{1}{1 + \beta e^t}\right) = \text{indefinite} \tag{2.18}$$

Hence, state response cannot be determined analytically. But it may described using relation between control and state found in Eq.[2.17] as

$$y(t) = 2u^*(t) = \frac{2}{1 + \beta e^t} \tag{2.19}$$

Boundary conditions can be determined using Eq.[2.20]

$$\left[\frac{\partial h}{\partial y} - p(t_f)\right] \delta y(t_f) + \left[\mathcal{H} + \frac{\partial h}{\partial t}\right] \delta t_f = 0 \tag{2.20}$$

Since final time is fixed, δt_f goes to zero. Therefore, Eq.[2.20] becomes

$$\left[\frac{\partial h}{\partial y} - p(t_f)\right] \delta y(t_f) = 0 \tag{2.21}$$

$$\frac{\partial h}{\partial y} - p(t_f) = 0 \tag{2.22}$$

Since $h = -y(t_f = 5)$, Eq.[2.22] becomes

$$\frac{\partial}{\partial y}\left[-y(t_f = 5)\right] - p(t_f) = 0 \tag{2.23}$$

$$-1 - p(t_f) = 0 \tag{2.24}$$

$$p(t_f) = -1 \tag{2.25}$$

And by equating Eq.[2.3] to boundary condition found in Eq.[2.25] at $t_f = 5$, we obtain

$$p(t_f = 5) = ce^{(1-u)5} = ce^{5-5u} = -1 \tag{2.26}$$

Integration constant, c can be determined from Eq.[2.26] as

$$c = -e^{5u-5} \tag{2.27}$$

And by substituting control input found in Eq.[2.17] into Eq.[2.27], we obtain

$$c = -e^{\frac{5}{1+\beta e^t}-5} \tag{2.28}$$

And by substituting Eq.[2.17] and Eq.[2.28] into Eq.[2.3], we can obtain an expression for costate, p as

$$p(t) = -e^{\frac{5}{1+\beta e^t}-5} \cdot e^{\left(1-\frac{1}{1+\beta e^t}\right)t} = -e^{-5+\frac{5}{1+\beta e^t}+\left(1-\frac{1}{1+\beta e^t}\right)t} \tag{2.29}$$

And by applying condition found in Eq.[2.5] into Eq.[2.29], we obtain that,

$$p(t) = -e^{-5+\frac{5}{1+\beta e^t}+\left(1-\frac{1}{1+\beta e^t}\right)t} < 0 \tag{2.30}$$

Since Eq.[2.29] is always negative, the condition applied in Eq.[2.30] is satisfied
Until now, we have found the analytical results. In order to solve the given problem numerically, we can set up a two point boundary value problem(TPBVP) using with two equations we found already which are Eq.[1.2],Eq.[2.2] and Eq.[2.24]

$$\dot{y} = yu - u^* - (u^*)^2 \quad , \quad y(0) = 1 \tag{2.31}$$

$$\dot{p} = p - pu^* \quad , \quad p(t_f = 5) = -1 \tag{2.32}$$

Indirect single shooting method can be implemented in order to solve described TPBVP. In this implementation intial guess for the initial costate is 0. To integrate the system variable stepsize integrator is used such as CVODES integrator from the SUNDIALS suite, available in CasADi library. The resulting nonlinear system of equations are solved using IPOPT with a dummy objective function.

3 Results and Conclusion

In Fig.[1], results of algebraic and numerical are compared for state, costate, control input. Also error between algebraic and numerical calculations are determined for state, costate and control input.

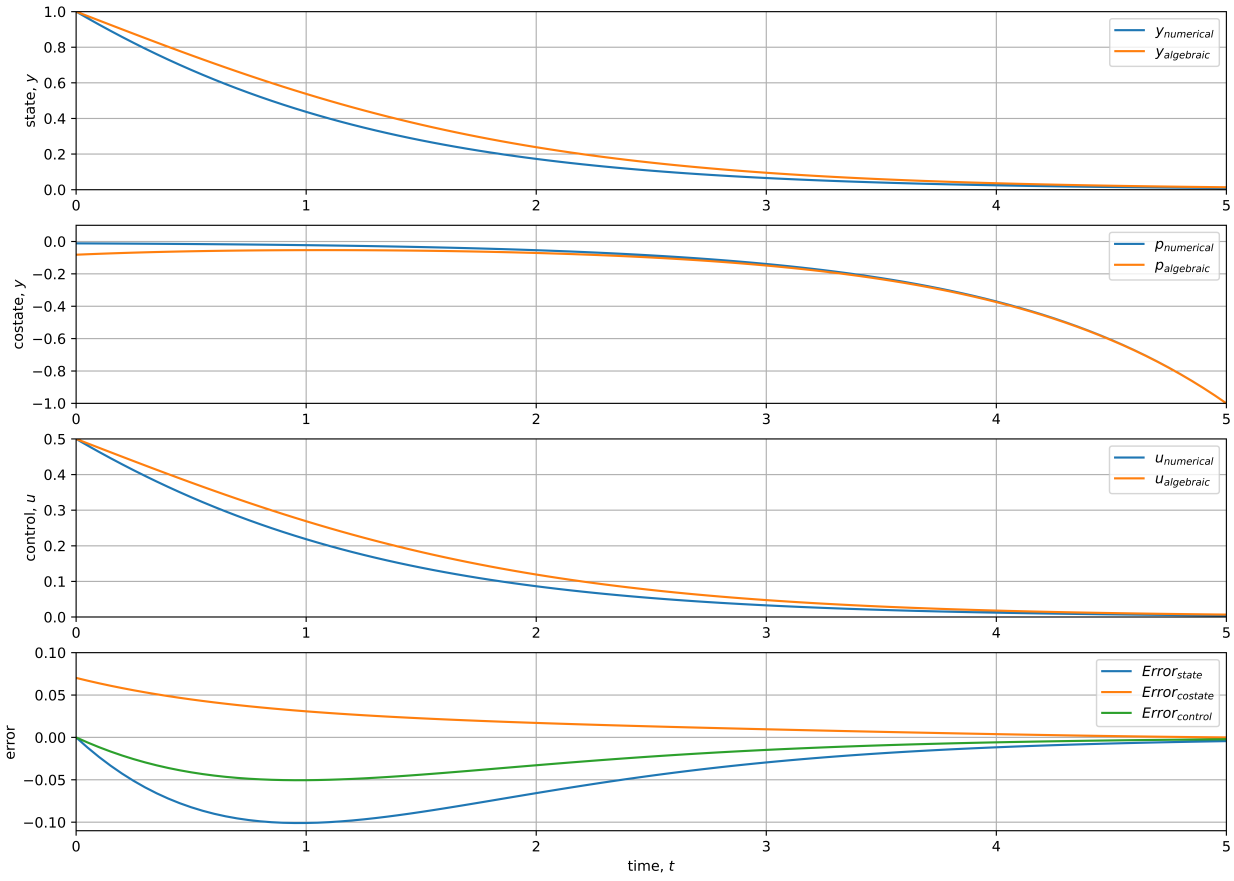

Figure 1: Graphical representation of admissible trajectories and the switching curve

Conclusion: The numerical results are obtained by simulating the system using an integrator function to approximate the state, costate, and control variables at discrete time intervals. These approximations are subject to errors arising from the discretisation of the time interval, numerical integration error, and solver tolerances. Therefore, the differences between the algebraic and numerical results may arise due to these errors and tolerances in the numerical simulation. Typically, the errors in numerical solutions decrease as the time step size is reduced, so using smaller time step sizes or more advanced integration methods may reduce the differences between the algebraic and numerical results.

4 Appendix A: Python Code

```
import casadi as ca
import matplotlib.pyplot as plt
import numpy as np
# for people who cannot see an interactive plot, uncomment the ...
    following lines
import matplotlib
if matplotlib.get_backend() == 'agg':
matplotlib.use('WebAgg')
print(f'backend: {matplotlib.get_backend()}')

T = 5                          # integration time
# state variables
x = ca.SX.sym('x', 1)          # state
lam = ca.SX.sym('lam', 1)      # costate
aug = ca.vertcat(x, lam)       # augmented state
# boundary values
x0bar = 1
lamTbar = -1
# optimal control
uopt = 0.5*x[0]
# state dynamics
xdot = x[0]*uopt-x[0]-uopt**2
# costate dynamics
lamdot = (-lam[0]*uopt)+lam[0];
# augmented dynamics
augdot = ca.vertcat(xdot, lamdot)
# integrator
# aim: create integrator function F(x_in, tf) that integrates the
```

```
28 # given dynamics starting at state x_in over a time interval tf and returns
29 # the resulting state.
30 tf = ca.SX.sym('tf')                    # integration interval
31 # create struct with information needed to define the integrator
32 # the dynamics are scaled by tf.
33 # this is because CVODES by default integrates over the time interval ...
       [0,1].
34 # integration of the scaled dynamics from 0 to 1 is equivalent to
35 # integration of the original unscaled dynamics from 0 to tf
36 dae = {'x': aug, 'p': tf, 'ode': tf * augdot}
37 # some options (precision, in the sense of absolut and relative error
38 # tolerance)
39 opts = {'abstol': 1e-8, 'reltol': 1e-8}
40 # create the desired integrator function
41 F = ca.integrator('F', 'cvodes', dae, opts)
42 ## build dummy NLP
43 # initial value of lambda (to be found)
44 lam0 = ca.MX.sym('lam0', 1)
45 # compute augmented state at time T dependend on lam0
46 intout = F(x0=ca.vertcat(x0bar, lam0), p=T)
47 augT = intout["xf"]                      # augmented state at T
48 lamT = intout["xf"][1:2]                 # final value lambda(T)
49
50 # terminal condition
51 # this residual should be zero
52 g = lamT - lamTbar
53 lbg = 0
54 ubg = 0
55 # NLP with dummy objective
56 nlp = {'x': lam0, 'f': 0, 'g': g}
57 solver = ca.nlpsol('solver', 'ipopt', nlp)
58 sol = solver(lbg=lbg, ubg=ubg)
59 lam0opt = sol["x"].full()
60
61 ## simulate solution
62 # integrate in N timesteps to get intermediate results
63 N = 100
64 DT = T/N
65 # integration loop
66 AUG0 = np.vstack((x0bar, lam0opt)).flatten()
67 AUG = np.empty((AUG0.size, N+1))
68 AUG[:, 0] = AUG0
69 for i in range(N):
```

```
intres = F(x0=AUG[:, i], p=DT)
AUG[:, i+1] = intres["xf"].full().flatten()
# split int state, costate, controls
xopt = AUG[0, :]
lamopt = AUG[1, :]
u_opt = 0.5*xopt
# time grid
tvec = np.linspace(0, T, num=N+1)

# algebraic results
beta = 1
uoptA = 1 / (1 + beta * np.exp(tvec))
xoptA = 2 * uoptA
lamoptA = (-np.exp(5 * uoptA - 5)) * (np.exp((1 - uoptA) * tvec))
# error evaluation
errorU = u_opt - uoptA;
errorX = xopt - xoptA;
errorLam = lamopt - lamoptA;
# plot section
plt.figure(1,dpi=1200)
plt.subplot(4,1,1)
plt.plot(tvec, xopt)
plt.plot(tvec, xoptA)
plt.legend([r'$y_{numerical}$',r'$y_{algebraic}$'],loc='upper right')
plt.xlabel(r'time, $t$')
plt.ylabel(r'state, $y$')
plt.grid(True)
plt.axis([0, 5,0, 1])

plt.subplot(4,1,2)
plt.plot(tvec, lamopt)
plt.plot(tvec, lamoptA)
plt.legend([r'$p_{numerical}$',r'$p_{algebraic}$'],loc='upper right')
plt.xlabel(r'time, $t$')
plt.ylabel(r'costate, $y$')
plt.grid(True)
plt.axis([0, 5,-1, 0.1])

plt.subplot(4,1,3)
plt.plot(tvec, u_opt)
plt.plot(tvec, uoptA)
plt.legend([r'$u_{numerical}$',r'$u_{algebraic}$'],loc='upper right')
plt.xlabel(r'time, $t$')
```

```
plt.ylabel(r'control, $y$')
plt.grid(True)
plt.axis([0, 5,0, 0.5])

plt.subplot(4,1,4)
plt.plot(tvec, errorX)
plt.plot(tvec, errorLam)
plt.plot(tvec, errorU)
plt.legend([r'$Error_{state}$',r'$Error_{costate}$',r'$Error_{control}$'],loc='upper right')
plt.xlabel(r'time, $t$')
plt.ylabel(r'error')
plt.grid(True)
plt.axis([0, 5,-0.11, 0.1])

plt.subplots_adjust(top=3,bottom=1,right=2)
plt.savefig('comparison.png', dpi=1200, bbox_inches='tight')
plt.show()
```

5 References

[1] Kirk D.E. *Optimal Control Theory*. Dover Publications. 1998.ulus

[2] Bryson, A.E., Ho, Y.*Applied Optimal Control:Optimization, Estimation and Control*. CRC Press, 1975.

[3] Liberzon, D. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2012.

[4] Burghes, D. and Graham, A. *Control and Optimal Control Theories with Applications*. Horwood Publishing, 2004.

[5] Başpınar, B. *Introduction to Optimal Control Lecture Notes*. Istanbul Technical University, 2022.

[6] Zdeněk, H. *Optimal and Robust Control Lecture Notes*. Czech Technical University, 2022.

[7] Gros, S. and Diehl, M. *Numerical Optimal Control Lecture Notes*. Freiburg University, 2020.