control-toolbox: solving control problems within Julia

Joseph Gergaud, Jean-Baptiste Caillau, Olivier Cots, Pierre Martinon

control-toolbox: solving optimal control problems within Julia

Jean-Baptiste Caillau, Olivier Cots, Joseph Gergaud, Pierre Martinon, Sophia Sed

What it's about

- Nonlinear optimal control of ODEs:

$$g(x(t_0), x(t_f)) + \int_{t_0}^{t_f} f^0(x(t), u(t))\,\mathrm{d}t \to \min$$

subject to

$$\dot{x}(t) = f(x(t), u(t)), \quad t \in [t_0, t_f]$$

plus boundary, control and state constraints

- Our core interests: numerical & geometrical methods in control, applications

Where it comes from

- BOCOP: the optimal control solver
- HamPath: indirect and Hamiltonian pathfollowing

- Coupling direct and indirect solvers, examples

OptimalControl.jl

- Double integrator
- Batch processing
- Goddard problem

Wrap up

- High level modelling of optimal control problems
- Efficient numerical resolution coupling direct and indirect methods
- Collection of examples

Future

- New applications (biology, space mechanics, quantum mechanics and more)
- Additional solvers: direct shooting, collocation for BVP, Hamiltonian pathfollowing...
- ... and open to contributions!
- CTProblems.jl

control-toolbox.org

- Open toolbox
- Collection of Julia Packages rooted at OptimalControl.jl

control-toolbox

The control-toolbox ecosystem gathers `Julia` packages for mathematical control and applications. It is an outcome of a research initiative supported by the Centre Inria of Université Côte d'Azur and a sequel to previous developments, notably Bocop and Hampath. See also: ct gallery. The root package is `OptimalControl.jl` which aims to provide tools to solve optimal control problems by direct and indirect methods.

Documentation

doc OptimalControl.jl

Installation

See the installation page.

Partners

Université de Toulouse

UNIVERSITÉ CÔTE D'AZUR

Inria

cnrs

FRANCE 2030 PROGRAMME DE RECHERCHE INTELLIGENCE ARTIFICIELLE

Credits (not exhaustive!)

anr agence nationale de la recherche

- ADNLPModels.jl

- DifferentiationInterface.jl
- DifferentialEquations.jl
- MLStyle.jl

« Catenoid solution

JuliaCon 2024 »

Double integrator: time minimisation

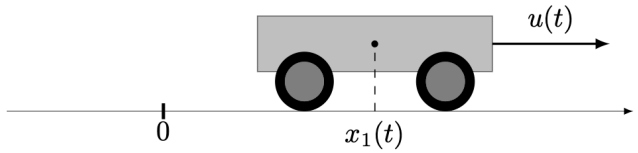
The problem consists in minimising the final time t_f for the double integrator system

$$\dot{x}_1(t) = x_2(t), \quad \dot{x}_2(t) = u(t), \quad u(t) \in [-1, 1],$$

and the limit conditions

$$x(0) = (1, 2), \quad x(t_f) = (0, 0).$$

This problem can be interpretated as a simple model for a wagon with constant mass moving along a line without fricton.



First, we need to import the `OptimalControl.jl` package to define the optimal control problem and `NLPModelsIpopt.jl` to solve it. We also need to import the `Plots.jl` package to plot the solution.

```
using OptimalControl
```

```
using NLPModelsIpopt
using Plots
```

Then, we can define the problem

```
@def ocp begin

    tf ∈ R,          variable
    t ∈ [ 0, tf ],   time
    x = (q, v) ∈ R², state
    u ∈ R,           control

    tf ≥ 0
    -1 ≤ u(t) ≤ 1

    q(0)  == 1
    v(0)  == 2
    q(tf) == 0
    v(tf) == 0

     0 ≤ q(t) ≤ 5,          (1)
    -2 ≤ v(t) ≤ 3,          (2)

    ẋ(t) == [ v(t), u(t) ]

    tf → min

end
```

Nota bene

In order to ensure convergence of the direct solver, we have added the state constraints labelled (1) and (2):

$$0 \leq q(t) \leq 5, \quad -2 \leq v(t) \leq 3, \quad t \in [0, t_f].$$

Solve it

```
sol = solve(ocp)
```

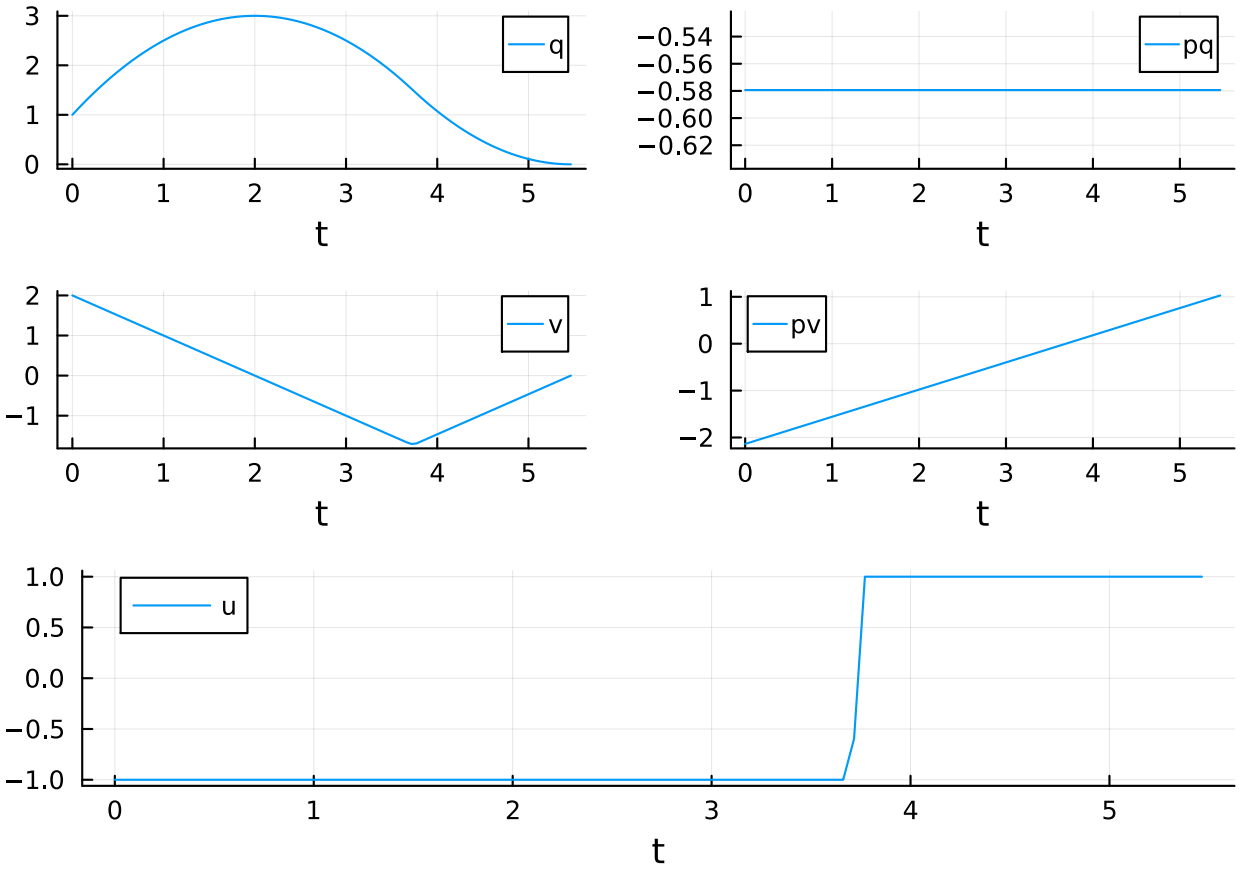
```
Method = (:direct, :adnlp, :ipopt)
This is Ipopt version 3.14.14, running with linear solver MUMPS 5.6.2.

Number of nonzeros in equality constraint Jacobian...:     1004
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:      202

Total number of variables............................:      304
                     variables with only lower bounds:        1
                variables with lower and upper bounds:      303
                     variables with only upper bounds:        0
Total number of equality constraints.................:      204
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0
```

```
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  1.0000000e-01 1.90e+00 0.00e+00   0.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1  9.9999010e-04 1.88e+00 2.32e+01  -5.7 1.10e+01    -  8.53e-02 8.99e-03h  1
   2  9.9900010e-06 1.88e+00 6.44e+03  -5.7 1.84e+03    -  5.27e-04 1.07e-04h  1
   3r 9.9900010e-06 1.88e+00 9.99e+02   0.3 0.00e+00    -  0.00e+00 2.68e-07R  3
   4r 1.8899160e-03 1.74e+00 4.65e+02  -0.7 4.77e-01    -  4.73e-01 5.34e-01f  1
   5r 1.4658984e-01 1.73e+00 2.09e+01  -1.1 1.45e-01   2.0 9.79e-01 1.00e+00f  1
   6r 2.9193546e-01 1.70e+00 4.08e+01  -0.8 1.70e-01   2.4 1.00e+00 8.53e-01f  1
   7r 3.5836667e-01 1.70e+00 4.73e+01  -0.9 6.64e-02   2.9 1.00e+00 1.00e+00f  1
   8r 5.9834814e-01 1.70e+00 7.36e+01  -0.9 3.75e-01   2.4 9.65e-01 6.40e-01f  1
   9r 7.2398061e-01 1.70e+00 9.10e+01  -1.7 1.48e-01   2.8 1.00e+00 8.51e-01f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  10r 8.4184963e-01 1.71e+00 4.59e+02  -1.7 6.10e-01   2.3 1.00e+00 1.93e-01f  1
  11r 9.4835839e-01 1.72e+00 7.81e+02  -1.3 3.80e-01   1.8 1.00e+00 2.80e-01f  1
  12r 1.0283813e+00 1.71e+00 8.78e+02  -1.6 5.91e-01   1.4 1.00e+00 1.67e-01f  1
  13r 1.0666091e+00 1.66e+00 2.42e+02  -2.8 1.71e-01   1.8 1.00e+00 5.19e-01f  1
  14  5.7360280e-01 1.56e+00 5.29e+01   0.3 1.05e+01    -  2.58e-01 6.17e-02h  1
  15  6.1993703e-01 1.56e+00 7.19e+01  -5.7 4.08e+02    -  2.38e-03 2.47e-03h  1
  16  7.2554684e-01 1.55e+00 7.04e+01  -5.7 1.98e+02    -  6.46e-03 2.79e-03h  1
  17  9.7142594e-01 1.53e+00 5.07e+01  -5.7 6.37e+01    -  1.85e-02 1.07e-02h  1
  18  1.6662909e+00 1.45e+00 1.50e+01   0.5 1.28e+01    -  1.10e-01 5.58e-02f  1
  19  7.3295731e+00 1.70e-01 4.17e+02   0.3 6.42e+00    -  6.24e-01 8.83e-01f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  20  6.1520840e+00 1.78e-02 1.48e+02   0.1 1.41e+00    -  1.00e+00 9.21e-01h  1
  21  6.2752208e+00 2.09e-04 1.72e+01  -0.5 2.95e-01    -  9.95e-01 9.90e-01f  1
  22  6.2660417e+00 1.86e-06 6.24e+00  -2.2 1.23e-02    -  1.00e+00 9.91e-01h  1
  23  5.6695137e+00 1.84e-03 2.90e-02  -2.8 5.97e-01    -  1.00e+00 1.00e+00f  1
  24  5.5118889e+00 7.92e-04 6.45e+02  -3.3 5.19e-01    -  9.78e-01 1.00e+00h  1
  25  5.4861365e+00 1.97e-04 6.37e+03  -3.8 5.63e-01    -  1.00e+00 8.15e-01h  1
```

```
  26  5.4663359e+00 4.94e-05 6.71e+01  -4.8 5.00e-01    -  1.00e+00 9.97e-01h  1
  27  5.4649557e+00 7.68e-08 1.39e-07  -5.8 6.24e-03    -  1.00e+00 1.00e+00h  1
  28  5.4648096e+00 1.42e-09 3.60e-01 -11.0 1.33e-03    -  9.97e-01 1.00e+00h  1
  29  5.4648096e+00 8.88e-16 4.76e-16 -10.3 1.55e-06    -  1.00e+00 1.00e+00h  1

Number of Iterations....: 29

                                   (scaled)                 (unscaled)
Objective...............:   5.4648095606409361e+00    5.4648095606409361e+00
Dual infeasibility......:   4.7600990651224600e-16    4.7600990651224600e-16
Constraint violation....:   8.8817841970012523e-16    8.8817841970012523e-16
Variable bound violation:   9.5654382192833509e-09    9.5654382192833509e-09
Complementarity.........:   5.0382564513062877e-11    5.0382564513062877e-11
Overall NLP error.......:   5.0382564513062877e-11    5.0382564513062877e-11


Number of objective function evaluations             = 34
Number of objective gradient evaluations             = 21
Number of equality constraint evaluations            = 34
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 31
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 29
Total seconds in IPOPT                               = 0.448

EXIT: Optimal Solution Found.
```

and plot the solution

```
plot(sol)
```



« Basic example (functional version)

Initial guess options »

Batch processing

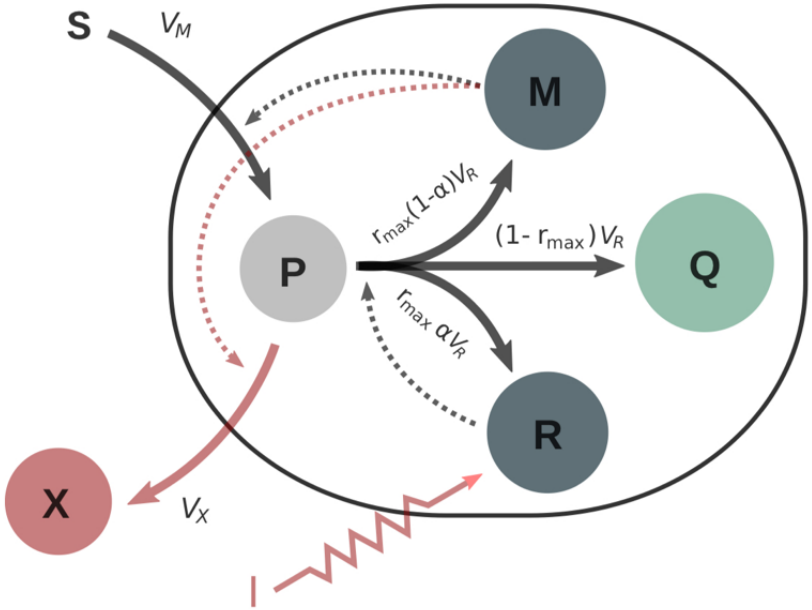
Introduction

Let us consider a *self-replicator* model [1] describing the dynamics of a microbial population growing inside a closed bioreactor. The bacterial culture has a constant volume. At the beginning of the experience, there is an initial mass of substrate S inside the bioreactor, that is gradually consumed by the bacterial population, and transformed into precursor metabolites P. These precursors are intermediate metabolites used to produce proteins—such as ribosomes and enzymes—responsible for specific cellular functions; and metabolites of interest X which are excreted from the cell. The proteins forming bacterial cells are divided into three classes M, R and Q, associated with the following cellular functions:

- class M proteins of the metabolic machinery, responsible for the uptake of nutrients,
- class S from the medium (substrate),
- the production of precursor metabolites P,
- and the synthesis of metabolites of interest X.

Class R proteins of the gene expression machinery (such as ribosomes) actively involved in protein biosynthesis (*i.e.* in the production of proteins of classes M, Q and R). Class Q Growth are rate-independent proteins, such as housekeeping proteins responsible for cell maintenance, and ribosomes not involved in protein synthesis. There is an *internal* control, α, accounting for the behaviour of each

individual cell that has to decide between two pathways (transforming precursors P into metabolites M or into ribosomes R), and one *external* control leveraging the production of the product X (for instance using a light induced control of bioengineered cells).



After some normalisations, a simplified version of the system (not describing the production of metabolite as we will focus on volume maximisation on this example) can be written in terms of the concentrations of the substrate s, precursors p, ribosomes r, and the of the volume V of the bacterial population.[2] Accordingly, the state $\varphi = (s, p, r, V)$ is four-dimensional, and there is only one control, α:

$$
\begin{aligned}
\dot{s} &= -w_M(s)(1-r)V, \\
\dot{p} &= w_M(s)(1-r) - w_R(p)r(p+1), \\
\dot{r} &= (\alpha - r)w_R(p)r, \\
\dot{V} &= w_R(p)rV,
\end{aligned}
$$

where velocities are taken linear in the concentrations, and where Michaelis-Menten kinetics are assumed:

$$v_R := V_R/V = w_R(p)r, \quad v_M := V_M/V = w_M(s)m,$$

with

$$w_R(p) = \frac{k_R\, p}{K_r + p}, \quad w_M(s) = \frac{k_m s}{K_m + s}.$$

Biomass maximisation

We first import the needed packages.

```
using OptimalControl
using NLPModelsIpopt
using Plots
```

We are interested in maximising the biomass production [3] (final volume of the bacterial population) over a finite time horizon $[0, t_f]$. To solve the problem, we first set up the boundary values,

```
t0 = 0
```

```
V0 = 0.003
```

together with parameters and auxiliary functions definnig the synthesis rates:

```
kᵣ = 1.1
kₘ = 1.2
Kᵣ = 1.3
Kₘ = 1.4

wᵣ(p) = kᵣ * p / (Kᵣ + p)
wₘ(s) = kₘ * s / (Kₘ + s)
```

Then we define the optimal control problem setting time, state, control, boundary conditions, state and control constraints, dynamics and Mayer cost:

```
@def batch begin

    t ∈ [ t0, tf ], time
    φ = (s, p, r, V) ∈ R⁴, state
    α ∈ R, control

    s(t0) == s0
    p(t0) == p0
    r(t0) == r0
    V(t0) == V0

    s(t) ≥ 0
    p(t) ≥ 0
```

```
    0 ≤ r(t) ≤ 1
    V(t) ≥ 0
    0 ≤ α(t) ≤ 1

    φ̇(t) == F0(s(t), p(t), r(t), V(t)) + α(t) * F1(s(t), p(t), r(t), V(t))

    V(tf) → max

end
```

The dynamics is indeed affine in the control, $\dot{\varphi} = F_0(\varphi) + \alpha F_1(\varphi)$, with vector fields

```
F0(s, p, r, V) =
    [ -wₘ(s) * (1 - r) * V
       wₘ(s) * (1 - r) - wᵣ(p) * r * (p + 1)
      -wᵣ(p) * r^2
       wᵣ(p) * r * V ]

F1(s, p, r, V) = [ 0, 0, wᵣ(p) * r, 0 ]
```

Direct solve

Since the following result holds,[4]

Proposition. *The Lie bracket F_{101} belongs to the span of F_1 and F_{01}, so singular controls are at least of local order two.*

one expects singular arcs connected with bang arcs through Fuller phenomenon (accumulation of switching times).

We first solve the problem using a uniform discretisation:

```
sol0  = solve(batch; grid_size=1000, print_level=0)
println("Objective ", sol0.objective, " after ", sol0.iterations, " iterations")
```

```
Method = (:direct, :adnlp, :ipopt)
Objective 0.07683200113875356 after 111 iterations
```

Although convergence is obtained, it is actually more efficient to first solve on a raw grid, then use a *warm start* to solve again on a finer (still uniform) grid:

```
sol1 = solve(batch; grid_size=20, print_level=0)
println("Objective ", sol1.objective, " after ", sol1.iterations, " iterations")
```

```
Method = (:direct, :adnlp, :ipopt)
Objective 0.0769994219213667 after 31 iterations
```

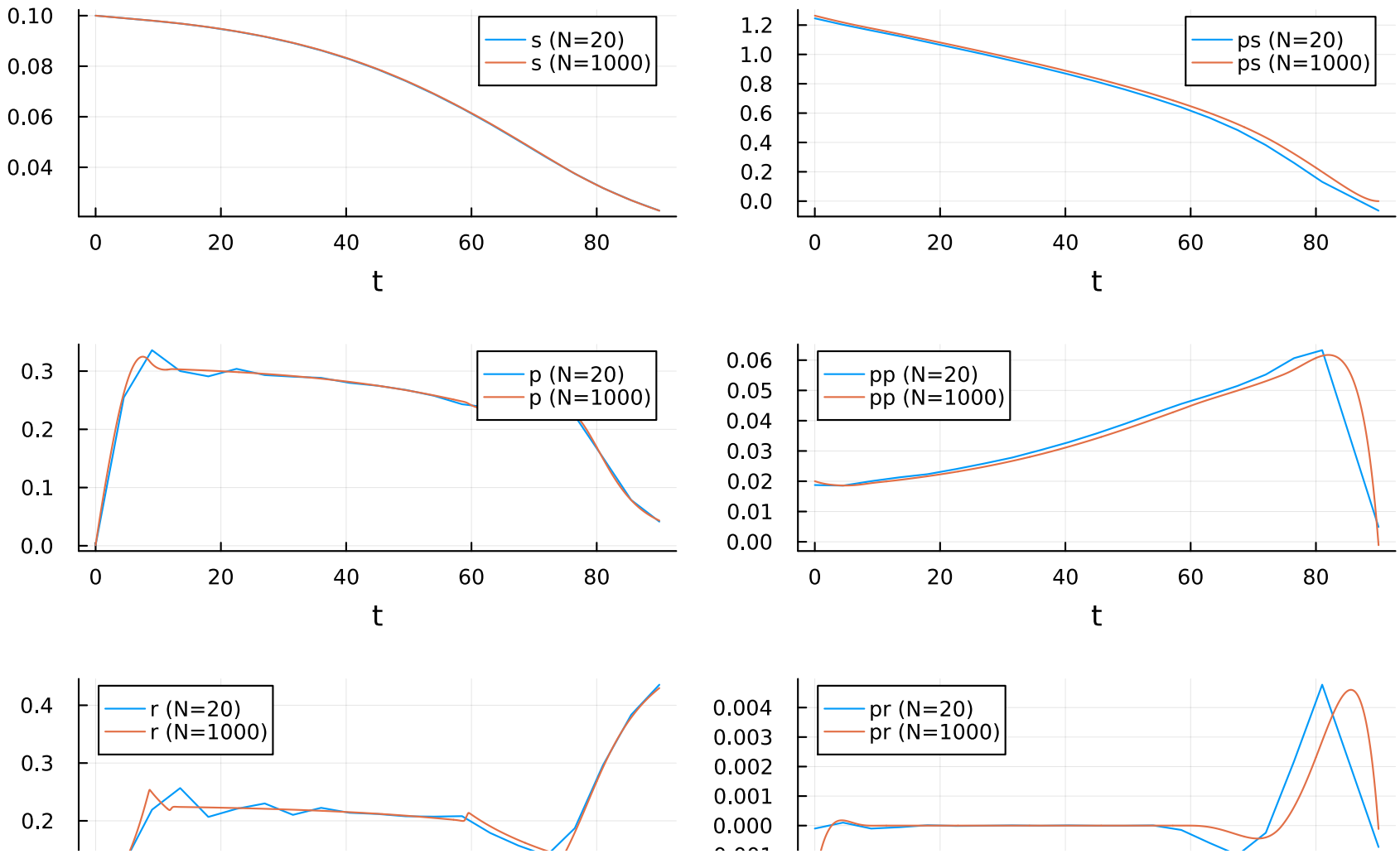
```
sol2 = solve(batch; grid_size=1000, print_level=0, init=sol1)
println("Objective ", sol2.objective, " after ", sol2.iterations, " iterations")
```

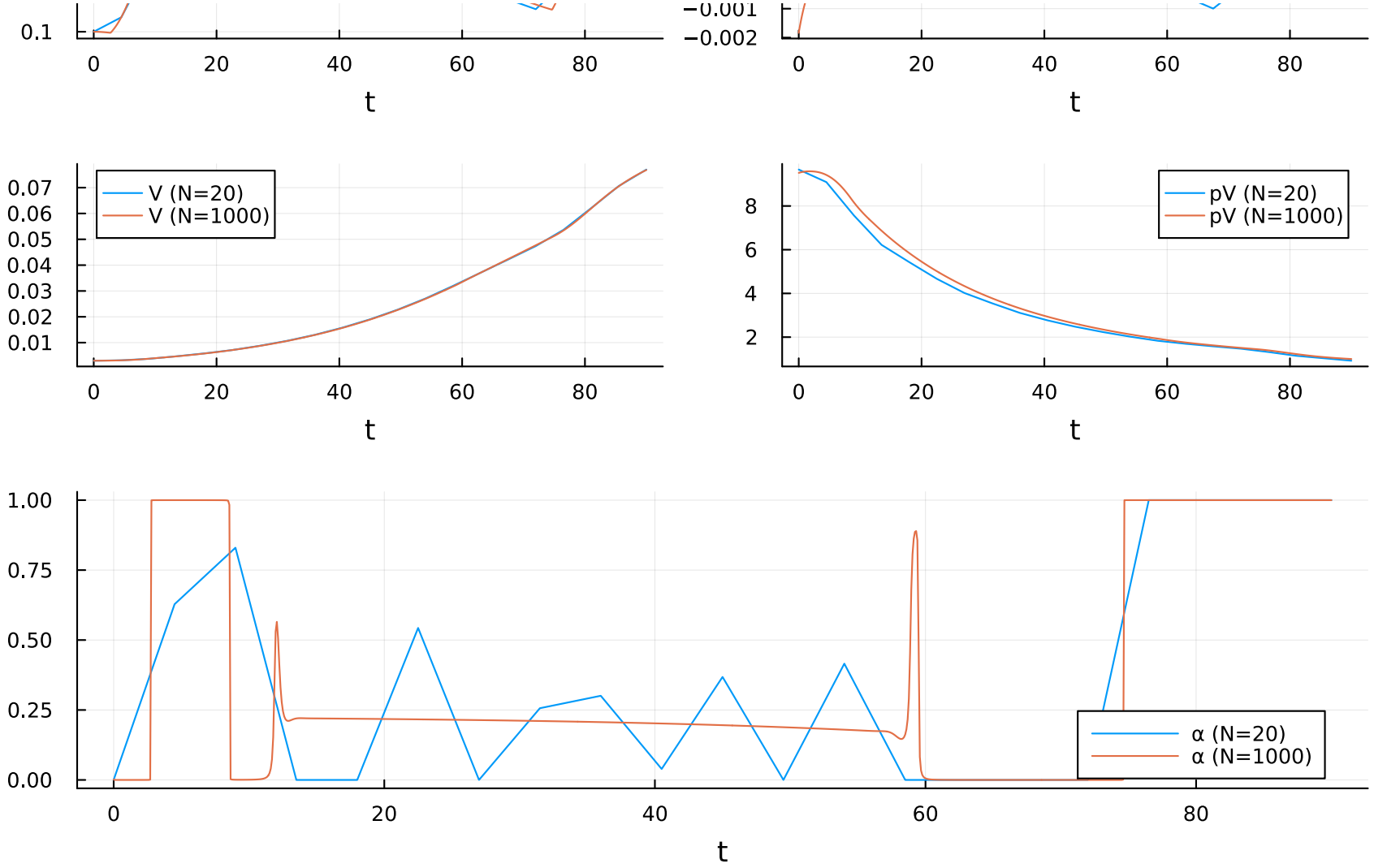
```
Method = (:direct, :adnlp, :ipopt)
Objective 0.07683200113876136 after 22 iterations
```

Plotting

We eventually plot the solutions (raw grid + finer grid) and observe that the control exhibits the expected structure with a Fuller-in arc followed by a singular one, then a Fuller-out arc:

```
plot(sol1; solution_label="(N=20)", size=(800, 1000))     # N is the grid size
plot!(sol2; solution_label="(N=1000)")
```




0.1
0
20
40
60
80
t
−0.001
−0.002
0
20
40
60
80
t
0.07
0.06
0.05
0.04
0.03
0.02
0.01
V (N=20)
V (N=1000)
0
20
40
60
80
t
8
6
4
2
pV (N=20)
pV (N=1000)
0
20
40
60
80
t
1.00
0.75
0.50
0.25
0.00
α (N=20)
α (N=1000)
0
20
40
60
80
t

References

- 1 Giordano, N.; Mairet, F.; Gouzé, J.-L.; Geiselmann, J.; De Jong, H. Dynamical allocation of cellular resources as an optimal control problem: novel insights into microbial growth strategies. *PLoS comp. biol.* **12** (2016), e1004802.
- 2 Yabo, A. G.; Caillau, J.-B.; Gouzé, J.-L.; de Jong, H.; Mairet, F. Dynamical analysis and optimization of a generalized resource allocation model of microbial growth. *SIAM J. Appl. Dyn. Syst.* **21** (2022), no. 1, 137-165.
- 3 Yabo, A. G.; Caillau, J.-B.; Gouzé, J.-L. Optimal bacterial resource allocation strategies in batch processing. *SIAM J. Appl. Dyn. Syst.*, to appear.
- 4 Astruc, L.; Edery, N. Optimal allocation of bacterial resources in a bioreactor. Project report, Polytech Nice Sophia, Université Côte d'Azur (2023).

« Goddard problem

Solar sail »