Using a finite horizon numerical optimisation method for a periodic optimal control problem

Jeffrey D. Azzato and Jacek Krawczyk

Victoria University of Wellington

11. February 2007

USING A FINITE HORIZON NUMERICAL OPTIMISATION METHOD FOR A PERIODIC OPTIMAL CONTROL PROBLEM

JEFFREY D. AZZATO & JACEK B. KRAWCZYK

ABSTRACT. Computing a numerical solution to a periodic optimal control problem is difficult. A method of approximating a solution to a given (stochastic) optimal control problem using Markov chains was developed in [3]. This paper describes an attempt at applying this method to a periodic optimal control problem introduced in [2].

2007 Working Paper
School of Economics and Finance

JEL Classification: C63 (Computational Techniques), C87 (Economic Software).
AMS Categories: 93E25 (Computational methods in stochastic optimal control).
Authors' Keywords: Computational economics, Approximating Markov decision chains.

This report documents 2007 research into
Computational Economics Methods
directed by Jacek B. Krawczyk
supported by University Research Fund, GMS Grant Number 16378.

Correspondence should be addressed to:

Jacek B. Krawczyk. Faculty of Commerce and Administration, Victoria University of Wellington, PO Box 600, Wellington, New Zealand. Fax: +64-4-4712200; E-mail: Jacek.Krawczykatvuw.ac.nz; Website: http://www.vuw.ac.nz/staff/jacek_krawczyk

Contents

Introduction

In [2], Gaitsgory and Rossomakhine develop a numerical method for approximating a solution to a long run average problem of optimal control via linear programming. They consider optimal control problems of the form:

$$\inf_{(u(\cdot),\, y(\cdot))} \frac{1}{S}\int_0^S g(u(\tau), y(\tau))\, d\tau \tag{1}$$

subject to:

$$\dot{y}(\tau) = f(u(\tau), y(\tau)) \tag{2}$$

where:

- $f : U \times \mathbb{R}^m \to \mathbb{R}^m$ is continuous in (u, y) and satisfies Lipschitz conditions in y,
- The controls are Lebesgue measurable functions $u : [0, S] \to U$ and U is a compact metric space, and
- The infimum in (1) is taken over all admissible pairs $(u(\cdot), y(\cdot))$ on $[0, S]$. A pair $(u(\tau), y(\tau))$ is said to be *admissible* on $[0, S]$ if (2) holds for almost all $\tau \in [0, S]$ and $(\forall \tau \in [0, S])\ y(\tau) \in Y$, where Y is a given compact subset of $\mathbb{R}^m$.

If the infimum is taken over only those admissible pairs that are periodic, *i.e.* such that:

$$(\forall \tau \geqslant 0) \quad (u(\tau), y(\tau)) = (u(\tau + T), y(\tau + T))$$

for some $T > 0$, then (1) becomes a periodic optimisation problem of the form:

$$\inf_{(u(\cdot),\, y(\cdot))} \frac{1}{T}\int_0^T g(u(\tau), y(\tau))\, d\tau \tag{3}$$

where the infimum is now taken over T and all admissible pairs defined on $[0, T]$ that satisfy the periodicity condition $y(0) = y(T)$.

In this paper, we show how `SOCSol4L` (see [1]) can be used to solve a periodic optimal control problem of this type.

1. Periodic Optimal Control Problem

1.1. **Problem Formulation.** We examine *Example 1* of [2]. In this example, Gaitsgory and Rossamakhine take:

$$g(u, y) := u^2 - y_1^2 \tag{4}$$

and

$$f(u, y) := (y_2, -4y_1 - \tfrac{3}{10}y_2 + u) \tag{5}$$

in (3) and (2) respectively, where $y := (y_1, y_2)$ and $U := [-1, 1]$. Note that f results from the reduction of the differential equation:

$$\ddot{x}(\tau) + \tfrac{3}{10}\dot{x}(\tau) + 4x(\tau) = u(\tau)$$

to a first order system.

1.2. **Remarks on Analytical Solution.** This problem may be investigated analytically with the aid of, for example, the Maximum Principle. While we shall not attempt a full analytical solution here, it is instructive to consider how one might proceed.

The (simple) Hamiltonian for the problem is:

$$\begin{aligned} H(u, y, \pi) :&= -g(u, y) + \sum_{i=1}^{2} \pi_i f_i(u, y) \\ &= y_1^2 - u^2 + \pi_1 y_2 + \pi_2(-4y_1 - \tfrac{3}{10}y_2 + u) \end{aligned}$$

So the optimal control is:

$$\hat{u} := \arg\max_u H(u, y, \pi)$$

Clearly:

$$\hat{u}(\tau) = \begin{cases} 1 & \text{if } \pi_2(\tau) > 2 \\ \frac{\pi_2(\tau)}{2} & \text{if } |\pi_2(\tau)| < 2 \\ -1 & \text{if } \pi_2(\tau) < -2 \end{cases} \tag{6}$$

It follows that the optimal solution exhibits a “bang-bang control with a transition”. The system of adjoint equations for the problem is:

$$\begin{aligned} \dot{\pi}(\tau) &= \begin{bmatrix} -\frac{\partial H(u,y,\pi)}{\partial y_1} \\ -\frac{\partial H(u,y,\pi)}{\partial y_2} \end{bmatrix} \\ &= \begin{bmatrix} -2y_1 + 4\pi_2 \\ -\pi_1 + \frac{3}{10}\pi_2 \end{bmatrix} \end{aligned}$$

Combining these with the state evolution $\dot{y} = f(\hat{u}, y)$ yields the canonical system:

$$\frac{d}{d\tau}\begin{bmatrix} y_1 \\ y_2 \\ \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -4 & -\frac{3}{10} & 0 & 0 \\ -2 & 0 & 0 & 4 \\ 0 & 0 & -1 & \frac{3}{10} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \pi_1 \\ \pi_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \hat{u} \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

Solving this when $|\pi_2(\tau)| > 2$ reveals the period to be:

$$T = \frac{40\pi}{\sqrt{1591}} = 3.150 \quad (4\ \textit{s.f.})$$

Note that knowledge of the period provides two constraints for eliminating constants from the solution of (7), namely $y(0) = y(T)$.

2. Solution with SOCSol4L

2.1. **About SOCSol4L.**

SOCSol4L is a suite of MATLAB® routines for approximating solutions to stochastic optimal control problems using Markov chains. The method used is described in [3], while the use of SOCSol4L is explained in [1].

2.2. **Solution Syntax.**

This section describes the commands used to solve the problem using SOCSol4L.

The following functions are defined by the user and saved as MATLAB® .m files in a folder on the MATLAB® path. Each file is composed of a header followed by one or more commands.

Delta Function File:

This determines the system dynamics (f), and is written as follows:

```
function v = Delta(u, x, t)

v = [x(2), -4*x(1) - 0.3*x(2) + u];
```

Instantaneous Cost Function File:

This determines the instantaneous cost of the system (g), and is written as follows:

```
function v = Cost(u, x, t)

v = u^2 - x(1)^2;
```

Terminal State Function File:

As the problem has no scrap value, this is simply zero:

```
function v = Term(x)

v = 0;
```

The values chosen for the SOCSol4L parameters are explained below.

StateLB and StateUB: These were set to [-3, -4.7] and [3, 4.7] respectively to yield a state grid encompassing Gaitsgory and Rossamakhine's optimal trajectory (see p. 2020 of [2]).

`StateStep`: This was set to `[0.05, 0.05]`.

Naturally, both wide state bounds and fine state steps lead to increased computation times. Here, the state bounds were chosen based on prior knowledge of the problem solution. If such information is unavailable, a "calibration run" could be conducted with generous state bounds and coarse state steps so as to sketch the solution's behaviour.

The fine discretisation used reflects the "difficulty" of the problem: coarser state steps were not found to yield good results. Determining good discretisation for a given problem depends largely on experimentation.

`TimeStep`: This was set to `ones(1, 30*32)/32`, giving a finite time horizon of length 30, subdivided into thirty-seconds.

This long time horizon in fact spans several periods, facilitating easy estimation of T and identification of any anomalies near the time boundaries 0 and 30. Had periodicity been suspected but the period unknown, a calibration run could have been conducted with a generous time horizon to check for periodicity and provide a broad estimate of the period were it present.

`Problem File`: This is the name for the results files, say `'output'`.

`Options`: It is worthwhile tightening the termination tolerance on the function value, so this is set to {`'TolFun', '1e-12'`}.

`Initial Control Value`: This is given the value `0`. As this value is only an approximate starting point for the routine, it may be specified with some inaccuracy.

`A, b, Aeq` and `beq`: As there are no linear control constraints other than bounds, these are all passed as empty: `[]`.

`ControlLB` and `ControlUB`: These are set to `-1` and `1` respectively to define U.

`User Constraint Function File`: As there are no constraints requiring the use of this argument, it is also passed as empty: `[]`.

Consequently `SOCSol4L` could be called in MATLAB® as follows:

```
SOCSol('Delta', 'Cost', 'Term', [-3, -4.7], [3, 4.7], [0.05, 0.05],
       ones(1, 30*32)/32, 'output', {'TolFun', '1e-12'}, 0, [], [], [], [], -1,
       1, []);
```

However, it seems preferable to write and execute a script. This might look something like:

```
StateLB = [-3, -4.7];
StateUB = [3, 4.7];
StateStep = [0.05, 0.05];
TimeStep = ones(1, 30*32)/32;
Options = {‘TolFun’, ‘1e-12’};
InitialControlValue = 0;
A = [];
b = [];
Aeq = [];
beq = [];
ControlLB = -1;
ControlUN = 1;

SOCSol(‘Delta’, ‘Cost’, ‘Term’, StateLB, StateUB, StateStep, TimeStep,
       ‘output’, Options, InitialControlValue, A, b, Aeq, beq, ControlLB,
       ControlUB);
```

If this script were called `workspace.m` and placed in a directory visible to the MATLAB® path, it would then only be necessary to call `workspace` in MATLAB® to run `SOCSol4L`.

On running, `SOCSol4L` produces a pair of results files - in this case `output.DPS` and `output.DPP`. These are then interpreted with the aid of other package routines as described below.

2.3. **Results.**

2.3.1. *State and Control vs. Time.* `GenSim` uses the results files to derive a continuous-time, continuous-state control rule. It then simulates the system with this rule to produce state and control time paths starting from a given initial condition. The performance index is also returned.

A very fine simulation step is necessary to yield smooth time paths from this problem's control rule - here 75000 steps were used. Examination of Gaitsgory and Rossamakhine's optimal time paths (see Figure 5 of [2]) suggests $(2.1, 0.1)$ as an approximate initial condition. Consequently, `GenSim` was called as:

```
GenSim(‘output’, [2.1, 0.1], 30*ones(1, 75000)/75000)
```

yielding the time paths shown in Figure 1 (which correspond to Figures 5 and 4 of [2]). The transitions between control -1 and control 1 in Figure 4 of [2] occur faster than those in Figure 1 overleaf (approximately 0.35 time units versus approximately 0.50 time units). The latter decreases with the time step: the transitions would have taken approximately 0.70 time units had half as many time divisions been used.

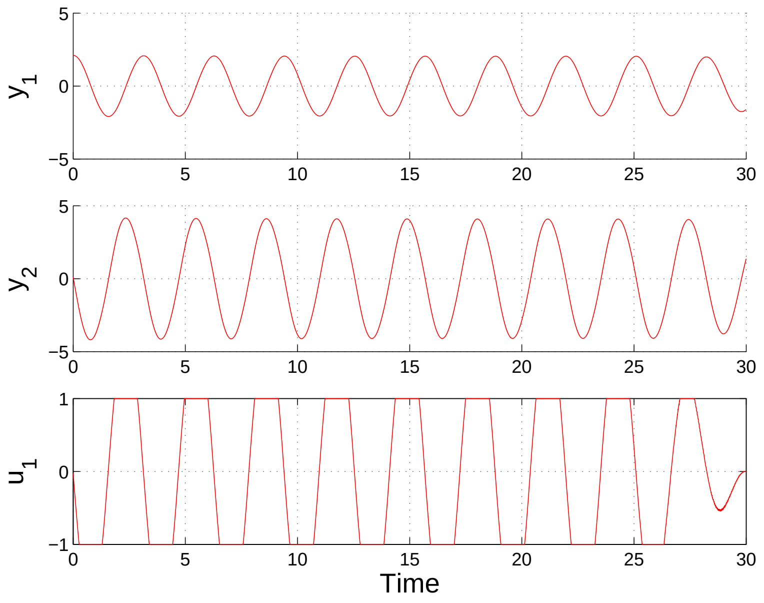

FIGURE 1. Approximated Time Paths.

T can be estimated from Figure 1 as 3.14. The first and last peaks in y_1 appear to occur at 3.15 and 28.23 respectively. This suggests calling `GenSim` again as:

```
( GenSim('output', [2.1, 0.1], 28.23*ones(1, 70575)/70575)
  - GenSim('output', [2.1, 0.1], 3.15*ones(1, 7875)/7875) )/25.08
```

yielding an estimate of -1.322 for the performance index, which closely agrees with Gaitsgory and Rossamakhine's estimate of -1.327 in [2]. The choice of `TimeStep` is important in deriving this value: halving the number of time divisions yields an estimate of only -1.218.

2.3.2. *State vs. State.* Plotting the pairs (y_1, y_2) from the time paths in Figure 1 yields the approximation of the optimal state trajectory shown in Figure 2. This corresponds closely to Gaitsgory and Rossamakhine's approximation in Figure 1 of [2].

2.3.3. *Control vs. State.* `ContRule` graphs the control rule against the selected state variable for a specified time and value of the other state variable.

For a graph of the control rule against y_1 at $(\tau, y_2) = (14.85, 3.904)$, `ContRule` would be called as:

```
ContRule('output', 14.85, [0, 3.904], 1);
```

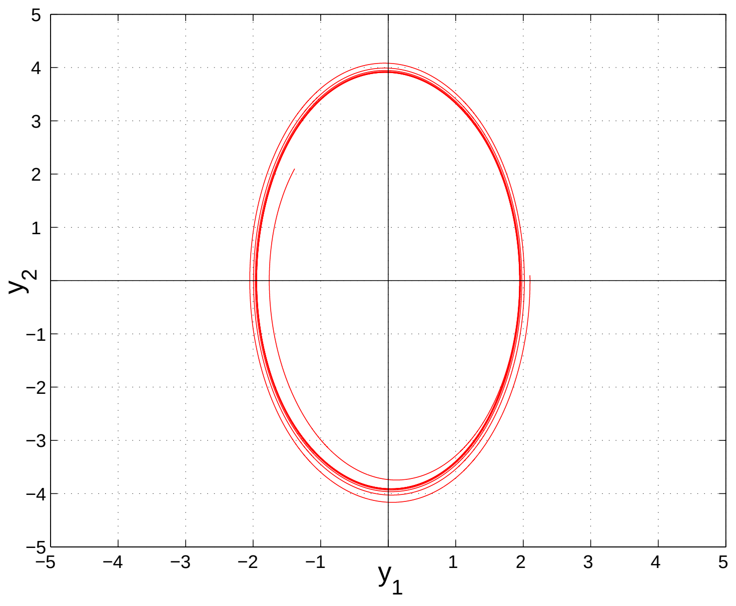

FIGURE 2. Approximated Trajectory.

Several similar calls were used to produce the control rules shown in Figures 3 and 4. The control rules are graphed against y_1 (on the left) and against y_2 (on the right) for four pairs (τ, y) taken from Figure 1.

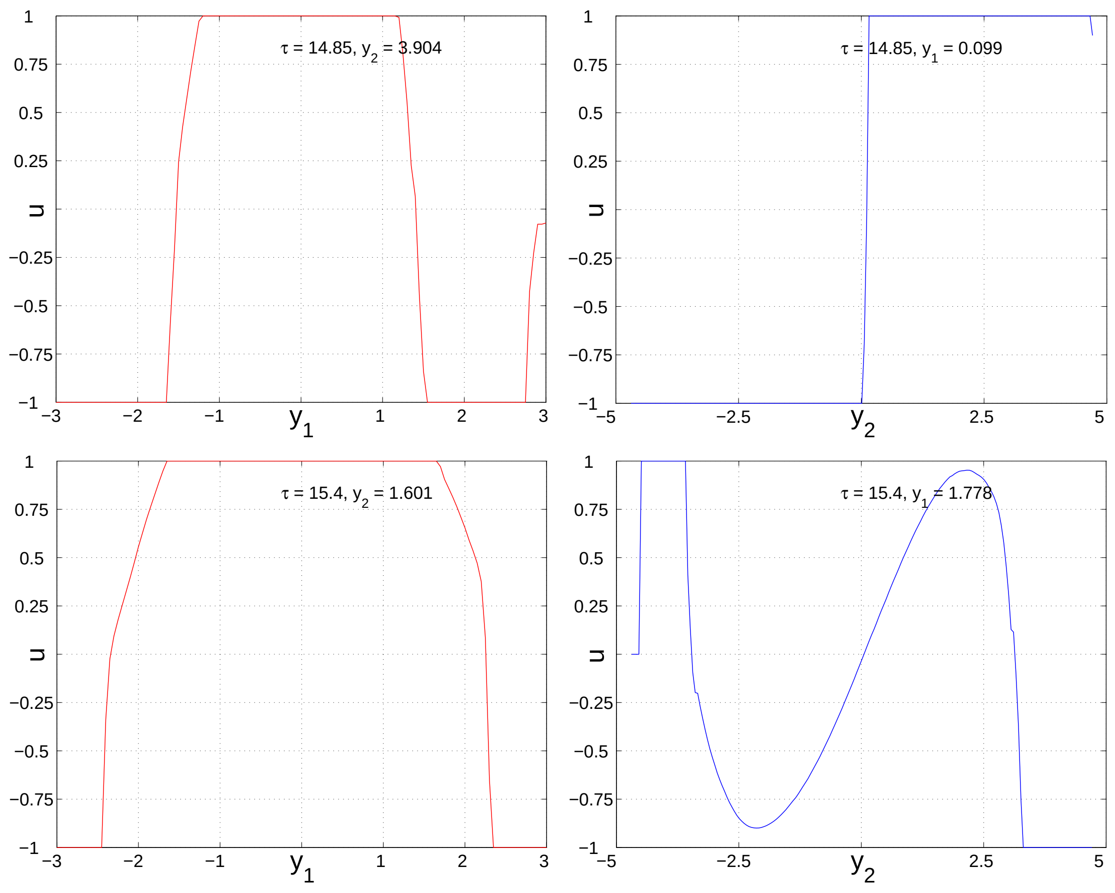

FIGURE 3. Control Rules.

As τ increases, a clear "flip" is evident in the control rules graphed against y_1, reflecting the form of (6). As y_2 is out of phase with y_1 by one quarter of a cycle, such a "flip" is absent in the control rules graphed against y_2.

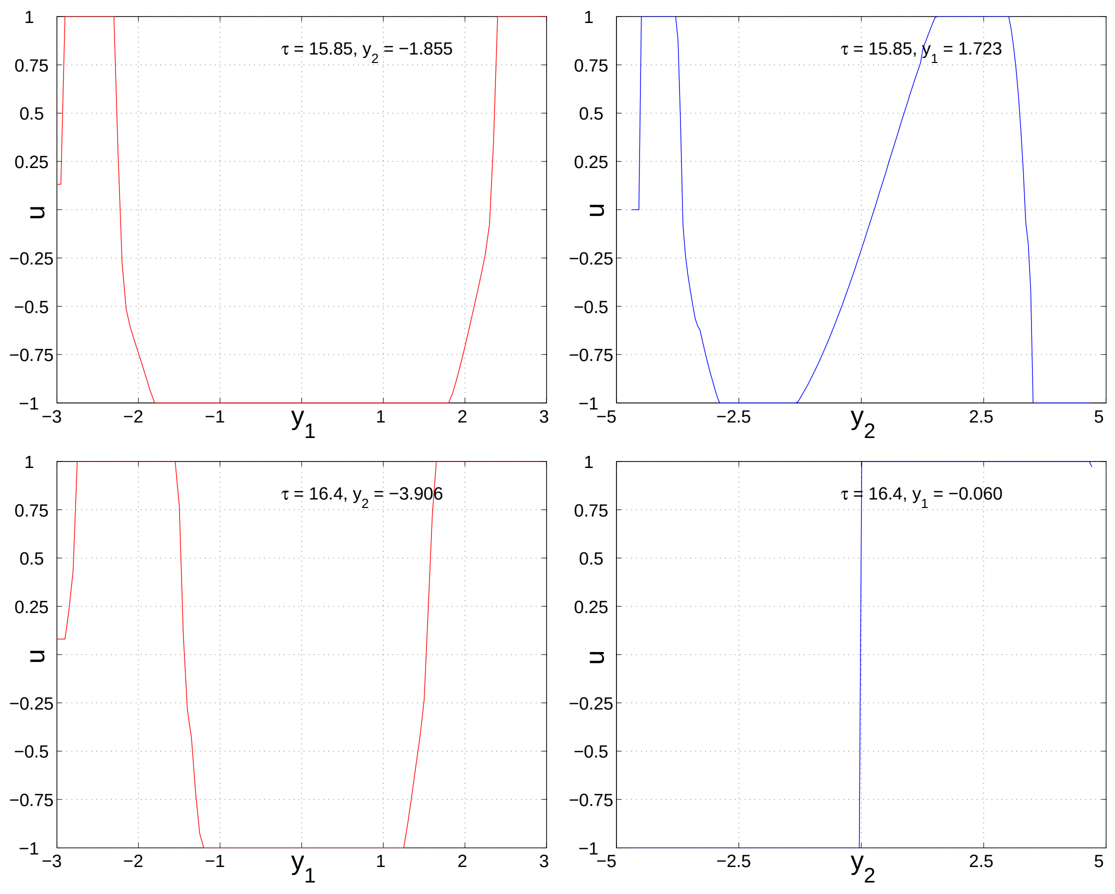

FIGURE 4. More Control Rules.

Note that Figure 2 of [2] is consistent with the upper-right plot in Figure 3, while Figure 3 of [2] is consistent with the lower-right plot in Figure 4.

2.4. System Specifications and Computation Time.

The results presented here were derived in 2 days 9.32 hours on a computer having a 2.80Ghz Pentium® 4 processor and 3Gb of memory. MATLAB® version 7.2.0.232 (R2006a) with Optimization Toolbox version 3.0.4 was run on a Windows XP Professional® operating system.

As computation time rises linearly with the number of time divisions (see [3]), results could have been derived considerably faster by reducing the length of the time horizon to, say, 10.

REFERENCES

[1] J.D. Azzato and J.B. Krawczyk. *SOCSol4L: An improved MATLAB® package for approximating the solution to a continuous-time stochastic optimal control problem.* School of Economics and Finance, VUW, 2006. MPRA: 1179; available at: http://mpra.ub.uni-muenchen.de/1179/ on 14/02/2007.

[2] V. Gaitsgory and S. Rossamakhine. Linear programming approach to deterministic long run average problems of optimal control. *SIAM J. Control Optim.*, 44:2006–2037, 2006.

[3] J.B. Krawczyk. A Markovian approximated solution to a portfolio management problem. *Inf. Technol. Econ. Manag.*, 1, 2001. Available at: http://www.item.woiz.polsl.pl/issue/journal1.htm on 14/02/2007.