Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER Journal of Economic Dynamics & Control 30 (2006) 569–593
JOURNAL OF Economic Dynamics & Control
www.elsevier.com/locate/jedc

# Optimal time aggregation of infinite horizon control problems

Nedim M. Alemdar[a], Sibel Sirakaya[b,*], Farhad Hüsseinov[a]

[a]*Department of Economics, Bilkent University, 06800 Bilkent, Ankara, Turkey*
[b]*Departments of Economics and Statistics, Center for Statistics and Social Sciences, University of Washington, Seattle, WA 98105, USA*

Received 20 February 2004; accepted 17 February 2005
Available online 3 May 2005

**Abstract**

This paper proposes a novel method that enhances numerical approximation of infinite horizon optimal control problems. For direct numerical optimization, a continuous-time infinite horizon model needs to be first recast as a discrete-time, finite-horizon control problem. The very transformation itself may significantly degrade the quality of the optimization results, if due care is not taken to preserve the salient features in the original model. Mercenier and Michel (1994. Econometrica 62, 635–656, 2001. Journal of Economic Dynamics and Control 25, 1179–1191), for instance, propose time aggregation methods that minimize approximation errors at the steady-state. Using their scheme, we show that overall optimization performance can be further improved if the discretization of the transient phase is optimal as well. Three sample problems are numerically solved to demonstrate the potential benefits.


*JEL classification:* C61; C63

*Keywords:* Infinite-horizon optimal control; Time aggregation; Direct numerical optimization

*Corresponding author. Tel.: +1 206 221 6981.
*E-mail address:* sirakaya@u.washington.edu (S. Sirakaya).

0165-1889/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.jedc.2005.02.005

## 1. Introduction

Infinite horizon continuous-time optimization problems appear frequently in various fields of economics. Generally, the nonlinearities in cost functionals and/or system dynamics and the presence of path constraints do not allow for an explicit solution. Despite the challenges posed by high-dimensional nonlinearity, recent advances in computing technology have enabled economists to study richer and more complex sets of economic dynamics rather than a compromise on the modeling exercise to obtain analytical solutions. In this vein, there has been a growing interest in computational aspects of infinite horizon optimal control problems.[1]

Often, an approximation is adopted at the outset that assumes quadratic objective and linear dynamics which affords an analytical solution. In cases where linear/quadratic approximation is not appropriate, the solution has to be computed numerically. For given initial states, open-loop solutions can be computed either by a numerical approximation of the necessary conditions by the Pontryagin's Maximum Principle, or by direct optimization of the objective functional using nonlinear programming techniques. If a feedback solution is more appropriate to the problem at hand then, numerical dynamic programming algorithms can be used.

Numerical methods for open-loop solutions can be classified into two broad classes: indirect and direct methods (Von Stryk and Bulirsch, 1992; Kraft, 1994). Indirect methods iterate on solutions until they satisfy the necessary conditions by the Pontryagin's Maximum Principle. Generally, gradient and shooting methods are applied (Bryson, 1999; Bryson and Ho, 1975). Direct methods, on the other hand, first approximate the original problem by discretizing the system dynamics in time and thus transforming them into a set of equality and inequality constraints. Then, a suitable nonlinear programming algorithm, local or global, can be used to numerically optimize the objective functional subject to the constraints.[2]

Dynamic programming yields optimal policies as feedback solutions. However, in practice dynamic programming methods are limited by the size of the problem. The term curse of dimensionality (Bellman, 1961) refers to the fact that an increase in the dimensionality of the problem causes an exponential increase in computational and memory requirements to find a solution.[3] There exists two broad approaches to dynamic programming: higher order approximations and adaptive gridding techniques. High-order approximations, which can be very efficient when the optimal value function is sufficiently smooth, include approximations using smooth functions like Chebyshev polynomials (Rust, 1996; Judd, 1996; Jermann, 1998), Splines (Daniel, 1976; Johnson et al., 1993; Trick and Zin, 1993, 1997) or piecewise high-order approximations (Falcone and Ferretti, 1998) and other high-order strategies, like in finite difference approximations (Candler, 2001), Gaussian

[1]See among others, Flam and Wets (1987), Grüne (1997), Judd (1992), Kehoe and Levine (1992) and Mercenier and Michel (1994a, b, 2001).

[2]See Hull (1997) and Betts (1998) for surveys.

[3]There exists approaches such as randomly distributed grid points (Rust, 1997) or so called low discrepancy grids (Rust, 1996; Reiter, 1999) which are able to break the curse of dimensionality.

Quadrature discretization (Tauchen and Hussey, 1991; Burnside, 2001) and perturbation techniques (Judd, 1996). Adaptive gridding schemes are based on flexible grids generated using local and/or global error estimates (Marcet, 1994; Grüne, 1997; Munos and Moore, 2002; Grüne and Semmler, 2003).[4]

In this paper, our focus will be on the direct methods of optimization. More specifically, we aim to contribute to direct optimization methods by showing that an optimal choice of time intervals in the discrete representation of the problem will enhance optimization performance. Direct methods transform the original optimal control problem into a nonlinear programming problem through discretization in time thereby relieving the computational burden relative to the methods that rely on the necessary conditions for numerical approximation. Consequently, an optimal choice of time grids will further improve the efficiency of direct optimization.

Mercenier and Michel (1994a) propose a time aggregation method for optimal control programs that have steady-steady solutions. The method ensures that discrete models have the same steady-states as the infinite-horizon continuous-time counterparts. The steady-state invariance property is achieved by imposing some simple restrictions on the discount factor in the time aggregated model. Later, Mercenier and Michel (2001), extend their results to optimal control programs that exhibit endogenously generated constant steady-state growth.

The time aggregation method in Mercenier and Michel (1994a, 2001) works for any given partition of the decision horizon provided that the truncation date is sufficiently late, and that the discount factor in the time-aggregated model obeys a certain recursion. That is, the steady-state in the discrete model will closely approximate that of the continuous-time analog, independent of how exactly the transient phase is divided. Thus, the proposed method is geared towards minimizing steady-state approximation errors from a discrete transformation.

Unlike in continuous models where controls are active at every instant, in discrete models, control frequency is also subject to a choice. An arbitrary choice of time intervals between decisions may significantly worsen the numerical results. This aspect of discrete approximation is often ignored in computational exercises where the frequency of decision making is assumed to be uniform; or, even if it is recognized, the dates so chosen are not necessarily optimal.

Mercenier and Michel (1994b), for instance, develops a measure which determines decision dates with frequencies that die out as time elapses (referred as M–M gridding hereafter). This criterion rests crucially upon the assumption that the speed of convergence along the transient phase is constant.[5] We show in our sample problems, the above uneven sampling method does indeed improve the optimization accuracy over uniform time steps when it is used in conjunction with the steady-state invariance restrictions. Unfortunately, however, in order for this idea to be useful,

[4]In passing, we note that adaptive 'state' gridding is similar in spirit to the optimal 'time' gridding suggested in this paper.

[5]Unless the system dynamics are linear, the speed of convergence will not generally be constant. Thus, time intervals given by this scheme will be suboptimal with nonlinear dynamics that start far from steady-state.

one needs to know the largest stable root of the linearized system around the stationary equilibrium. This is a significant drawback as it is computationally inefficient. After having derived the necessary conditions, linearized the system dynamics around the stationary equilibrium and computed the stable root, what would be the advantage of going back to approximate it once again, this time, with the direct methods?

Hence, the important question as to how 'best' to divide the decision horizon is still left unresolved. In fact, not only that time grids themselves should be best in some sense, but also that the method to discover them should also be efficient. This is the specific problem we tackle in this paper. The answer is simple enough: Use direct methods to also optimize the time intervals in the discrete models aggregated as in Mercenier and Michel (1994b, 2001). Indeed, given the number of decisions, the computational tradeoff that exists between the minimization of approximation errors at the steady-state and the transient phase, is an integral part of the optimization approach we propose. This way, the most important advantage of the direct methods namely, computational efficiency will not be compromised. We show that if time aggregation is carried out with a view to minimizing the overall approximation errors, the optimization performance, accuracy and speed, will significantly improve.

We take up three sample optimal control problems that frequently appear in the literature and compare our method with approximations using uniform decision dates and nonuniform decision dates as suggested by Mercenier and Michel (1994b, 2001). For numerical optimization, we use genetic algorithms (GAs) as a global search heuristic.[6] First, a simple regulator problem is adopted for its analytical tractability. Given a fixed number of inputs, optimal and M–M gridding enhance optimization accuracy over uniform gridding equivalently. Direct optimization of the time intervals is however more efficient in terms of better average performance and lower standard deviation over a number of runs than both M–M and uniform discretization. Next, we reconsider the Ramsey growth model in Mercenier and Michel (1994a). Again, if investments take place at optimal dates as per our method or, at unequally spaced M–M dates, welfare significantly improves over equally spaced investment dates. Our direct method has a slight performance advantage over M–M, but the real gain again is in terms of computational efficiency. Over a number of runs, the average performance is substantially higher with a lower variance compared to both approximations with uniform and M–M time grids. Finally, Lucas' (1988) growth model is numerically solved with the same parametrization as in Mercenier and Michel (2001). Since the model involves an economy wide externality, approximation involves both optimization and solution to a fixed point problem. We develop parallel GAs to tackle both tasks simultaneously. With the given parameters the approach to steady-state is almost linear so that the manner in which the transient phase is partitioned does not matter much. However, our method

[6]GA implementation, however, is not essential to our results; any other direct numerical optimization routine could do just as well.

approximates more efficiently, and captures better the qualitative features of the transient phase.

## 2. Optimal step sizes in time aggregated models

Generally speaking, two critical decisions need to be made for any discrete representation. First, the length of the transient phase and the stationary conditions thereafter have to be considered. Assuming that a continuous-time problem becomes stationary after some date, say $T$, then the objective functional can be truncated at $T$ with the stationary tail adjoined. Analogously, the discrete objective functional can also be appended with the stationary tail as a terminal criterion to approximate the model at the steady-state. Obviously, if additional accuracy is desired in approximating the transition dynamics, this will require a relatively larger $T$ with, of course, incremental computational costs.

Second, given the length of the transient phase, $T$, one must then decide on a specific sequence of dates, $t_n \in [0, T]$ and $n \in \{0, 1, \ldots, N-1\}$, at which controls will be activated. Any sequence of dates will partition $[0, T]$ such that $\sum_{n=0}^{N-1} \triangle_n = T$, where $\triangle_n = t_{n+1} - t_n$ and $t_0 = 0$. Obviously, the closer the dates, (the smaller $\triangle_n$) the better will be the approximation. At this juncture, the experimenter has to again weigh the enhanced numerical accuracy against the increased computational costs associated with a more frequent decision making.

Note that for any given $T$, a specific sequence of decision dates is comprised of the number, $N$, and the frequency of control actions, $\triangle_n$. For instance, if $T$ is arbitrarily fixed and decision intervals are assumed uniform, as is usually done in computational exercises, then, $\triangle_n = \triangle$ for all $n \in \{0, 1, \ldots, N-1\}$ and $N = T/\triangle$. As an alternate paradigm, we propose to fix the number of control actions, $N$, and treat the intervals, $\triangle_n$, also as control parameters. Thus, if $T$ is fixed ahead of time, then a sequence of optimal intervals, $\triangle_n^*$, must obey the constraint, $\sum_{n=0}^{N-1} \triangle_n^* = T$. Otherwise, the time at which the system becomes stationary is approximated as $\sum_{n=0}^{N-1} \triangle_n^* = T^*$.

### 2.1. Steady-state invariance

Consider the following generic multi-dimensional infinite horizon control problem, with a state vector $x(t) \in \mathscr{R}^n$ and $h(t) \in \mathscr{R}$ and a control vector $u(t) \in \mathscr{R}^m$:

$$\begin{aligned} J = \max \quad & \int_0^\infty \mathrm{e}^{-\rho t} h(t)^a g(x(t), u(t))\,\mathrm{d}t, \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t)), \quad x(0) = \bar{x}, \\ & \frac{\dot{h}(t)}{h(t)} = \varphi(x(t), u(t)), \quad h(0) = \bar{h}. \end{aligned} \tag{1}$$

Under standard differentiability conditions Pontryagin's maximum principle will hold and a balanced-growth path will exist. Assuming the maximized functional in

Eq. (1) is concave, Mercenier and Michel (2001) propose the following discrete transformation of Problem (1):

$$\begin{aligned} J_{\mathrm{d}} = \max \quad & \sum_{n=0}^{\infty} \frac{\triangle_n}{y_{t_{n+1}}} g(x_{t_n}, u_{t_n}), \\ \text{s.t.} \quad & x_{t_{n+1}} - x_{t_n} = \triangle_n f(x_{t_n}, u_{t_n}), \quad x_{t_0} = \bar{x}, \\ & y_{t_{n+1}} = y_{t_n}[1 + \triangle_n(\rho - a\varphi(x_{t_n}, u_{t_n}))], \quad y_{t_0} = \bar{y} \end{aligned} \tag{2}$$

with no restriction on the choice of $\triangle_n$.

The discrete approximation above preserves the steady-state invariance property in the sense that the constant optimal values of the control, state and the co-state variables for problem (2) will also be the optimal values for the continuous time problem (1) at the steady-state. Moreover, since the optimal values of $x_{t_n}$ and $u_{t_n}$ do not depend on $y_0$, it can arbitrarily be set to one. Finally, defining $\alpha_n = 1/y_{t_{n+1}}$ and substituting in the equation governing the motion of $y_{t_{n+1}}$, Mercenier and Michel (2001) gets

$$\alpha_n = \frac{\alpha_{n-1}}{1 + \triangle_n(\rho - a\varphi(x_{t_n}, u_{t_n}))}. \tag{3}$$

The recursion (3) is a generalization of the results in Mercenier and Michel (1994a) where growth is assumed to have ceased in the long-run, i.e., $\varphi = 0$.

Now, assuming that problem (1) reaches a stationary equilibrium $(\hat{x}, \hat{u})$ at some $T$, it can be restated as

$$J = \max \int_0^T \mathrm{e}^{-(\rho - a\gamma(t))t} g(x(t), u(t))\,\mathrm{d}t + G(\hat{x}),$$

where, by definition,

$$G(\hat{x}) = \int_T^{\infty} \mathrm{e}^{-(\rho - a\hat{\gamma})t} g(\hat{x}, \hat{u})\,\mathrm{d}t = \frac{\mathrm{e}^{-(\rho - a\hat{\gamma})T}}{\rho - a\hat{\gamma}}\, g(\hat{x}, \hat{u}).$$

The time aggregated version of the truncated problem becomes

$$J_{\mathrm{d}} = \max \sum_{n=0}^{N-1} \frac{\triangle_n}{y_{t_{n+1}}} g(x_{t_n}, u_{t_n}) + \beta_N G(x_N). \tag{4}$$

Mercenier and Michel (2001) shows that if $G(\hat{x}) = G(x_N)$ and recursion (3) holds for $n = 0, 1, \ldots, N-2$ with $\gamma_N = \varphi(u_{t_N}, x_{t_N})$ and $\beta_N = 1/y_{t_N}$ then $(\hat{x}, \hat{u})$ will be a solution also to (4).

Note that the steady-state invariance holds for any sequence of time intervals, $\triangle_n, n = 0, 1, \ldots, N-1$. However, in passing from continuous to discrete time, a new state variable, decision dates, is introduced which evolves according to $t_{n+1} = t_n + \triangle_n$, with the initial condition $t_0 = 0$. If $T$ is fixed ahead of time, the evolution of decision dates, $t_n$, will terminate with $t_N = T$ otherwise, $t_N$ is also freely chosen. In Mercenier and Michel (1994b, 2001), this state evolution is ignored as it is immaterial to the steady-state (or steady-growth) invariance results. Hence, by

choosing the decision intervals optimally, the decision dates can be optimized without nullifying the recursion (3).

Given any $N$, assuming $T$ is free, we constrain the intervals by, $0 \leqslant \triangle_n \leqslant \triangle_{\max}$ so that $\sum_{n=0}^{N-1} \triangle_n \leqslant T^u$ where $T^u = \sum_{n=0}^{N-1} \triangle_{\max}$. Now, if we can show the existence of a sequence of time intervals that maximizes problem (4), then that sequence will also satisfy the steady-state invariance (steady-growth invariance) restrictions in problem (1).

The following argument establishes the existence of a sequence, $\{\triangle_n^*\}_{n=0}^{N-1}$, that maximizes problem (4). Since $x_{t_1} = x_{t_0} + \triangle_0 f(x_{t_0}, u_{t_0})$ and $y_{t_1} = y_{t_0}(1+\triangle_0(\rho - a\varphi(x_0, u_{t_0})))$, $x_{t_1}$ depends continuously on $u_{t_0}$, $\triangle_0$ and $x_{t_0}$, and $y_{t_1}$ depends continuously on $u_{t_0}, \triangle_0$, $x_{t_0}$ and $y_{t_0}$. Also, because $x_{t_2} = x_{t_1} + \triangle_1 f(x_{t_1}, u_{t_1})$ and $y_{t_2} = y_{t_1}(1 + \triangle_1(\rho - a\varphi(x_1, u_{t_1})))$, $x_{t_2}$ is a continuous function of $u_{t_1}, u_{t_0}, \triangle_1, \triangle_0$ and $x_{t_0}$ and $y_{t_2}$ is a continuous function of $u_{t_1}, u_{t_0}, \triangle_1, \triangle_0$ and $x_{t_0}$, and $y_{t_0}$. Proceeding in this manner, it can be shown that $x_{t_1}, x_{t_2}, \ldots, x_{t_N}$ all depend continuously on $u_{t_0}, u_{t_1}, \ldots, u_{t_{N-1}}$, $\triangle_0, \triangle_1, \ldots, \triangle_{N-1}$ and $x_{t_0}$. Similarly, $y_{t_1}, y_{t_2}, \ldots, y_{t_N}$ are all continuous functions of $u_{t_0}, u_{t_1}, \ldots, u_{t_{N-1}}, \triangle_0, \triangle_1, \ldots, \triangle_{N-1}$, $x_{t_0}$ and $y_{t_0}$. Thus, the objective function in problem (4) is a continuous function of those $2N$ variables since $x_{t_0}$ and $y_{t_0}$ are fixed. Since the constraint set

$$\left\{(u_{t_0}, u_{t_1}, \ldots, u_{t_{N-1}}, \triangle_0, \triangle_1, \ldots, \triangle_{N-1}) | u_{t_n} \in U, \right.$$
$$\left. 0 \leqslant \triangle_n \leqslant \triangle_{\max}, n = 0, 1, \ldots, N-1 \text{ and } \sum_{n=0}^{N-1} \triangle_n \leqslant T^u \right\}$$

is compact, we conclude that problem (4) possesses a solution.[7]

## 3. Numerical solution method

Once the original problem is transformed into a nonlinear programming problem as per our method, then any local/or global search algorithm can be used for numerical optimization. Though, it is not essential for our main argument, we use GAs to approximate the solutions. There are several good reasons why we choose GAs to implement our solution method. GAs are global search algorithms which start completely blind but, learn as they solve. Starting from a random initial population of candidate solutions, they ensure convergence to an approximate global optimum by exploiting the domain space and relatively better solutions through genetic operators. Gradient based optimization methods, on the other hand, may get stuck in a local optimum or fail to converge at all, depending on the initial parameters. Furthermore, in addition to the inherent parallelism in a single GA,

[7]Note that for the steady-state invariance to hold, the continuous-time problem has to be concave. If the original control problem is nonconcave, neither the steady-state invariance nor optimization of time intervals will be valid.

several GAs can be implemented in parallel to approximate dynamic games or distribute computational tasks that require task-specific learning. Indeed, in one of our sample problems, we ease the computational burden by assigning one GA to optimize while another one to solve a fixed point problem. Against the aforementioned advantages, however, GAs are considerably slower than gradient based local search routines.

### 3.1. Genetic algorithms as an optimization heuristic

A GA, is an iterative search heuristic based on the mechanics of natural evolution. At any generation, say $s$, the algorithm maintains a constant-size population, $Pop(s)$, of candidate solutions to breed yet better solutions. By iterating on a population of well-adopted sample points rather than a single point, the probability of getting stuck at a local optimum is reduced. The GA operators are simple enough, involving no more than random number generation, string copying and partial string exchanging; yet, the resulting search performance is impressive (Goldberg, 1989).

The first step of optimization with a GA is to code the parameter set of the optimization problem as a finite-length string, usually over the binary alphabet ($\{0,1\}$). The initial population, $Pop(0)$, is generally random. To approximate the control problem (4), GA evolves $\{u^1_{s,t_n}, \triangle^1_{s,n}, \ldots, u^k_{s,t_n}, \triangle^k_{s,n}\}_{n=0}^{N-1}$, a $k$-sized population of candidate solutions, which is random at $s=0$.

Next, given an objective function, each string, $l$, is evaluated by computing its performance relative to the population, namely, its fitness score. For the control problem (4) this is

$$prob_l = \frac{J_d(x_{t_0}, y_{t_0}, u^l_{s,t_n}, \triangle^l_{s,n})}{\sum_{j=1}^{k} J_d(x_{t_0}, y_{t_0}, u^j_{s,t_n}, \triangle^j_{s,n})} \quad \text{for all } l \text{ and } s.$$

Based on the fitness scores, the individuals in $Pop(s-1)$ are copied into $Pop(s)$ by a randomized selection procedure. The probability of a certain schemata, a given binary configuration, in $Pop(s-1)$ being present in $Pop(s)$ is proportional to that schemata's fitness score in $Pop(s-1)$. By selecting structures for reproduction in such a way that the number of offspring of a given structure is proportional to that structure's performance (relative to the population), a GA achieves the important property of inherent parallelism: the number of structures that contain any given schemata, changes at a rate roughly proportional to the observed average performance of all structures that contain the given schemata (Holland, 1975; Theorem 6.2.3, p. 102.).[8] This operation is an artificial version of natural selection, a Darwinian survival of the fittest among individuals. GA searches the space of binary combinations by generating and testing points in the space of structures. Since the latter space is exponentially smaller than the former, a GA is a highly efficient search procedure (Holland, 1975).

[8]A schema is a similarity template: it is the set of all potential strings (chromosomes) with prespecified characters occupying designated positions.

Next, in order to breed fitter individuals (explore other points in the search space), some variation is introduced into the new population by means of genetic recombination operators without disrupting the selection process. Crossover is the primary recombination operator in generating new populations. Crossover recombines two binary strings at a random point by exchanging the segments to the right of this point. Thus crossover provides for further exploration within the search space. The secondary search operator, mutation, also introduces diversity in the population by randomly changing 1's or 0's of the structure at particular locations. Finally, the evolution terminates after a specified number of generations. The general outline of a GA is

```
procedure GA;
begin
initialize population Pop(0);
evaluate Pop(0);
t = 1;
repeat
  select Pop(s) from Pop(s-1);
  recombine Pop(s);
  evaluate Pop(s);
until (termination condition);
```

Our third sample problem requires parallelization of the above procedure as it involves both optimization and solution to a fixed point problem.[9] There, a representative agent takes a sequence of externality $\bar{w}_{t_n}$ as given and optimizes by choosing a sequence of actions $w_{t_n}$ and $v_{t_n}$. In optimizing, she is, however, unaware that in equilibrium $w_{t_n} = \bar{w}_{t_n}$. We construe this as a Nash equilibrium between two artificially intelligent players.

Player one is a GA, $GA^{\mathrm{O}}$, which evolves $\{w^1_{s,t_n}, v^1_{s,t_n}, \triangle^1_{s,n}, \ldots, w^k_{s,t_n}, v^k_{s,t_n}, \triangle^k_{s,n}\}^{N-1}_{n=0}$, a $k$-sized population of candidate solutions, based on the fitness score

$$prob_l = \frac{J_{\mathrm{d}}(x_{t_0}, y_{t_0}, w^l_{s,t_n}, v^l_{s,t_n}, \triangle^l_{s,n}, \bar{w}^{\star}_{s-1,t_n})}{\sum_{j=1}^{k} J_{\mathrm{d}}(x_{t_0}, y_{t_0}, w^j_{s,t_n}, v^j_{s,t_n}, \triangle^j_{s,n}, \bar{w}^{\star}_{s-1,t_n})} \quad \text{for all } l \text{ and } s,$$

where $\bar{w}^{\star}_{s-1,t_n}$ is the best performing candidate solution in the previous generation communicated by the second player.

The second player is also a GA, $GA^{\mathrm{F}}$, evolving a $k$-sized population of potential solutions, $\{\bar{w}^1_{s,t_n}, \ldots, \bar{w}^k_{s,t_n}\}^{N-1}_{n=0}$, based on the fitness score,

$$prob_l = \frac{J_f(\bar{w}^l_{s,t_n}, w^{\star}_{s-1,t_n})}{\sum_{j=1}^{k} J_f(\bar{w}^j_{s,t_n}, w^{\star}_{s-1,t_n})} \quad \text{for all } l \text{ and } s,$$

[9]The details of the parallel GAs can be found in Alemdar and Özyıldırım (1998). This parallelism is later extended to an online approximation of Stackelberg equilibria in Alemdar and Sirakaya (2003) and feedback Nash equilibria in Sirakaya and Alemdar (2003).

where $w^{\star}_{s-1,t_n}$ is the the best performing candidate solution in the last generation exchanged by $GA^{\mathrm{O}}$. The raw fitness of the second player is equal to,

$$J_f = \min\left(\sum_{n=0}^{N-1} |\bar{w}_{t_n} - w_{t_n}|^2\right)^{1/2}.$$

The exchange of best-to-date candidate solutions takes place via the computer shared memory synchronously at every generation. Essentially, $GA^{\mathrm{O}}$, optimizes while $GA^{\mathrm{F}}$ forecasts. Under evolutionary pressure $GA^{\mathrm{O}}$ learns to optimize better for any given forecast $\bar{w}^{\star}_{s-1,t_n}$. $GA^{\mathrm{F}}$, on the other hand, learns from past forecasts to better predict given any $w^{\star}_{s-1,t_n}$. Thus, the equilibrium of this game between the optimizer, $GA^{\mathrm{O}}$, and the forecaster, $GA^{\mathrm{F}}$, is the approximation of $J_{\mathrm{d}}$ and also the equilibrium $w_{t_n} = \bar{w}_{t_n}$. The following pseudo code outlines the steps involved in the parallel GA search.

```
procedure GA^O;
begin
  Randomly initialize Pop^O(0);
  copy initial w^l_{0,t_n} to shared memory;
  synchronize;
  evaluate Pop^O(0);
  s = 1;
  repeat
    select Pop^O(s) from Pop^O(s − 1);
    copy best w^l_{1,t_n} to shared memory;
    synchronize;
    crossover and mutate Pop^O(s);
    evaluate Pop^O(s);
    s = s + 1;
  until(termination condition);
end;
```

```
procedure GA^F;
begin
  Randomly initialize Pop^F(0);
  copy initial w̄^l_{0,t_n} to shared memory;
  synchronize;
  evaluate Pop^F(0);
  s = 1;
  repeat
    select Pop^F(s) from Pop^F(s − 1);
    copy best w̄^l_{1,t_n} to shared memory;
    synchronize;
    crossover and mutate Pop^F(s);
    evaluate Pop^F(s);
    s = s + 1;
  until(termination condition);
end;
```

## 4. Numerical experiments

For numerical optimization, we use the genetic operators in Genesis 5.0 (Grefenstette, 1990), a GA package. We fine tune search parameters and/or distribute computational tasks as the problem demands it. Each experiment starts with a randomly initialized constant size, 50, population. In Genesis 5.0, the selection of individuals is done by a stochastic procedure whereby each structure is allocated a portion of a spinning wheel proportional to that structure's relative fitness. A single spin of the wheel determines the number of offsprings assigned to every structure. The selection pointers are then randomly shuffled, and the selected structures are copied into the new population. This selection procedure is quite successful when the

search terrain is relatively free of noise; it leads to a fast convergence. If, however, the search terrain is highly erratic as is the case in the third sample problem where optimization and fixed point iterations go in parallel, it may lead to a premature convergence. Consequently, to avoid false convergence, we use *rank* selection in parallel GAs. Furthermore, we use a selection strategy that ensures that the best performing structures always survive intact from one generation to the next. In the absence of such an *elitism*, it is possible that the best structure may disappear after a mutation or a crossover operation. In all simulation, we adopt a crossover rate of 0.60 and a mutation rate of 0.001 which are both suggested by Grefenstette (1986) after experimenting on a number of optimization problems. We gradually increase the number of generations until no substantial improvement in performance is observed. Genesis 5.0 is compiled on a IBM RS/6000 running AIX 5.2.

In order to compare our discretization method with those using uniform and nonuniform M–M step sizes, we take up three sample problems. All our sample problems involve equality and inequality constraints. We simply substitute the equality constraints. If inequality constraints are violated, the fitness is reduced by a large penalty. The simple infinite-horizon discounted cost continuous-time regulator problem admits a unique closed-form solution, thus allowing for a direct comparison with the exact solution. Indeed, given $N$, optimal and M–M time-gridding equally improves performance accuracy over uniform gridding. Next, we reconsider the Ramsey growth problem. Though, there is a small advantage in favor of our method over M–M, both methods improve welfare over uniform gridding substantially. The improvement becomes more pronounced when the initial capital stock is relatively small compared to the steady-state. As a last experiment, we numerically solve Lucas' (1988) human capital model using the parametrization in Mercenier and Michel (2001). Here, rapid dynamics are further compounded by the presence of an externality. Our solution algorithm is novel, and successfully handles both complications. However, since the approach to steady-state is very fast and almost linear, transient dynamics do not matter much in terms of overall performance: The welfare gain from a more accurate approximation of the transition path is small. Nonetheless, the qualitative features of the transition, such as early full allocation of labor to production, is only captured by optimal time gridding.

### 4.1. Example I: a simple regulator

As a benchmark for comparison, consider the following continuous-time regulator problem,

$$J = \min \int_0^\infty \mathrm{e}^{-\rho t}((x(t)-1)^2 + u(t)^2)\,\mathrm{d}t,$$
$$\text{s.t.} \quad \dot{x}(t) = x(t) + u(t) - 1,$$

and $x(0) = \bar{x}$. Particularly, assume an initial state, $x(0) = 0.1$ and a discount rate, $\rho = 0.1$. In this problem, the optimal state and control trajectories are, $x(t) =$

$1-0.9e^{-1.3293t}$ and $u(t)=2.0964e^{-1.3293t}$, respectively. The feedback control is, $u=2.3293(1-x)$, yielding a minimum cost, $J=1.8869$.

For direct numerical optimization, the regulator problem is first recast in the time aggregated form as

$$J_d=\min \quad \sum_{n=0}^{N-1}\alpha_n\triangle_n((x_{t_n}-1)^2+u_{t_n}^2)+\beta_N((x_{t_N}-1)^2+u_{t_N}^2),$$

$$\text{s.t.} \quad x_{t_{n+1}}-x_{t_n}=\triangle_n(x_{t_n}+u_{t_n}-1)$$

and the initial condition $x(0)=0.1$. The discount factor, $\alpha_n$, obeys Eq. (3) for $n=0,1,\ldots,N-2$ and terminates with $\beta_N=\alpha_{N-1}$. When approximating with uniform step sizes, we fix the number of inputs, $N$, at 25 and assume the system to have reached the steady-state at $T=10$ so that $\triangle=0.40$. After $T=10$, the terminal cost, $(x_{t_N}-1)^2+u_{t_N}^2$, becomes zero. When control intervals are also optimized, they are constrained by $0.01\leqslant\triangle_n\leqslant 10$, so that $\sum_{n=0}^{24}\triangle_n\leqslant 250$. M–M decision dates are given by $t_n=(1/\lambda)\log(1-(n/N)(1-\exp(\lambda T))$ for $n=0,\ldots,N$, where $\lambda$ is the stable eigenvalue which is $-1.3293$ in this example.

Table 1 and Fig. 1 summarize our numerical results. From Table 1, observe the improvement in performance that comes along with nonuniform sampling times. The minimum total cost is reduced from 2.4630 with uniform steps to 1.9601 with optimized decision intervals, and to 1.9650 with M–M time gridding. In both cases, there is a gain in performance of around 20 percent. This reduction in costs is largely achieved by a more assiduous use of the fixed number of inputs, which calls for a more frequent control action early on to steer the system more rapidly towards the steady-state. Recall however, that in order for M–M time gridding method to realize this gain in accuracy, it would require the knowledge of the negative eigenvalue whereas our interval selection method assumes no such information and it is completely blind.

Reported in Table 1 are also the optimal and M–M decision dates. Note that in our method $T$ is not fixed ahead of time so that the final decision date is optimally chosen weighing the incremental reduction in the transient versus the steady-state costs. The final M–M decision date, on the other hand, is always equal to $T$, the assumed length of transient phase. In Fig. 1, the exact trajectory of the state variable is also displayed to better gauge the gain in accuracy from optimal time-aggregation.

### 4.2. Example II: transient growth

In order to demonstrate the computational payoffs and the potential welfare gains from an optimal sequencing of decision dates, we next adopt the one sector optimal growth model in Mercenier and Michel (1994a).[10]

[10]The growth model in Mercenier and Michel (1994a) is standard and adopted from Manne (1991) for numerical comparisons.

Table 1
A simple regulator[a]

| Uniform grids | | | | | Optimal grids | | | | M–M grids | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $\triangle_n$ | $t_n$ | $xt_n$ | $ut_n$ | $\triangle_n$ | $t_n$ | $xt_n$ | $ut_n$ | $\triangle_n$ | $t_n$ | $xt_n$ | $ut_n$ |
| 0 | 0.40 | 0.00 | 0.1000 | 1.6982 | 0.0100 | 0.0000 | 0.1000 | 2.1574 | 0.0307 | 0.0000 | 0.1000 | 2.0941 |
| 1 | 0.40 | 0.40 | 0.4193 | 1.0959 | 0.0341 | 0.0100 | 0.1126 | 2.0640 | 0.0320 | 0.0307 | 0.1367 | 2.0092 |
| 2 | 0.40 | 0.80 | 0.6254 | 0.7069 | 0.0356 | 0.0474 | 0.1527 | 1.9700 | 0.0334 | 0.0627 | 0.1733 | 1.9240 |
| 3 | 0.40 | 1.20 | 0.7583 | 0.4562 | 0.0372 | 0.0865 | 0.1927 | 1.8770 | 0.0350 | 0.0962 | 0.2100 | 1.8387 |
| 4 | 0.40 | 1.60 | 0.8441 | 0.2942 | 0.0387 | 0.1284 | 0.2325 | 1.7846 | 0.0367 | 0.1312 | 0.2467 | 1.7529 |
| 5 | 0.40 | 2.00 | 0.8994 | 0.1899 | 0.0405 | 0.1716 | 0.2719 | 1.6933 | 0.0386 | 0.1679 | 0.2834 | 1.6676 |
| 6 | 0.40 | 2.40 | 0.9351 | 0.1224 | 0.0423 | 0.2170 | 0.3111 | 1.6008 | 0.0407 | 0.2065 | 0.3201 | 1.5816 |
| 7 | 0.40 | 2.80 | 0.9581 | 0.0792 | 0.0446 | 0.2641 | 0.3496 | 1.5125 | 0.0430 | 0.2471 | 0.3568 | 1.4961 |
| 8 | 0.40 | 3.20 | 0.9730 | 0.0510 | 0.0467 | 0.3118 | 0.3880 | 1.4232 | 0.0456 | 0.2901 | 0.3934 | 1.4106 |
| 9 | 0.40 | 3.60 | 0.9826 | 0.0329 | 0.0492 | 0.3622 | 0.4258 | 1.3345 | 0.0486 | 0.3357 | 0.4301 | 1.3251 |
| 10 | 0.40 | 4.00 | 0.9888 | 0.0212 | 0.0520 | 0.4153 | 0.4632 | 1.2477 | 0.0519 | 0.3843 | 0.4668 | 1.2400 |
| 11 | 0.40 | 4.40 | 0.9928 | 0.0134 | 0.0552 | 0.4718 | 0.5002 | 1.1613 | 0.0557 | 0.4362 | 0.5035 | 1.1537 |
| 12 | 0.40 | 4.80 | 0.9953 | 0.0090 | 0.0587 | 0.5319 | 0.5367 | 1.0764 | 0.0602 | 0.4919 | 0.5401 | 1.0687 |
| 13 | 0.40 | 5.20 | 0.9970 | 0.0058 | 0.0630 | 0.5947 | 0.5727 | 0.9928 | 0.0655 | 0.5521 | 0.5767 | 0.9830 |
| 14 | 0.40 | 5.60 | 0.9981 | 0.0037 | 0.0684 | 0.6614 | 0.6083 | 0.9094 | 0.0717 | 0.6176 | 0.6134 | 0.8975 |
| 15 | 0.40 | 6.00 | 0.9988 | 0.0022 | 0.0745 | 0.7335 | 0.6437 | 0.8269 | 0.0793 | 0.6893 | 0.6500 | 0.8126 |
| 16 | 0.40 | 6.40 | 0.9992 | 0.0016 | 0.0823 | 0.8111 | 0.6788 | 0.7450 | 0.0886 | 0.7686 | 0.6867 | 0.7275 |
| 17 | 0.40 | 6.80 | 0.9995 | 0.0010 | 0.0920 | 0.8964 | 0.7137 | 0.6639 | 0.1005 | 0.8572 | 0.7234 | 0.6418 |
| 18 | 0.40 | 7.20 | 0.9997 | 0.0005 | 0.1044 | 0.9909 | 0.7484 | 0.5830 | 0.1160 | 0.9576 | 0.7601 | 0.5568 |
| 19 | 0.40 | 7.60 | 0.9998 | 0.0004 | 0.1211 | 1.0970 | 0.7830 | 0.5029 | 0.1372 | 1.0736 | 0.7969 | 0.4717 |
| 20 | 0.40 | 8.00 | 0.9999 | 0.0001 | 0.1448 | 1.2199 | 0.8176 | 0.4222 | 0.1679 | 1.2107 | 0.8337 | 0.3867 |
| 21 | 0.40 | 8.40 | 0.9999 | 0.0003 | 0.1805 | 1.3651 | 0.8523 | 0.3410 | 0.2164 | 1.3786 | 0.8708 | 0.3018 |
| 22 | 0.40 | 8.80 | 1.0000 | 0.0000 | 0.2400 | 1.5458 | 0.8872 | 0.2597 | 0.3050 | 1.5950 | 0.9081 | 0.2167 |
| 23 | 0.40 | 9.20 | 1.0000 | 0.0000 | 0.3573 | 1.7870 | 0.9224 | 0.1777 | 0.5214 | 1.9000 | 0.9462 | 0.1314 |
| 24 | 0.40 | 9.60 | 1.0000 | 0.0000 | 0.7085 | 2.1448 | 0.9582 | 0.0940 | 7.5785 | 2.4215 | 0.9866 | 0.0151 |
| 25 | | 10.00 | 1.0000 | 0.0000 | | 2.8519 | 0.9953 | 0.0047 | | 10.0000 | 0.9992 | 0.0008 |

[a]Respective minimum costs with uniform, optimal and M–M intervals are 2.4630, 1.9601 and 1.9650.

The model assumes logarithmic preferences, a Cobb–Douglas production technology, no capital depreciation, a constant population growth at rate $g$, and a post terminal growth at rate $g$ over an infinite horizon. The time-aggregated economic model is

$$\max \quad \sum_{n=0}^{N-1}\left\{\frac{1+g}{g}\,[1-(1+g)^{-\triangle n}]\alpha_n \log(c_{t_n})\right\} + \frac{\alpha_{N-1}}{\rho}\log(c_{t_N})$$

s.t.

$$k_{t_{n+1}} - k_{t_n} = \frac{1}{g}\,[1-(1+g)^{-\triangle n}][i_{t_n} - gk_{t_n}],$$

$$c_{t_n} + i_{t_n} = ak_{t_n}^{b},$$

$$c_{t_n} \text{ and } k_{t_n} \geqslant 0 \text{ and } k_{t_0} = \bar{k}.$$

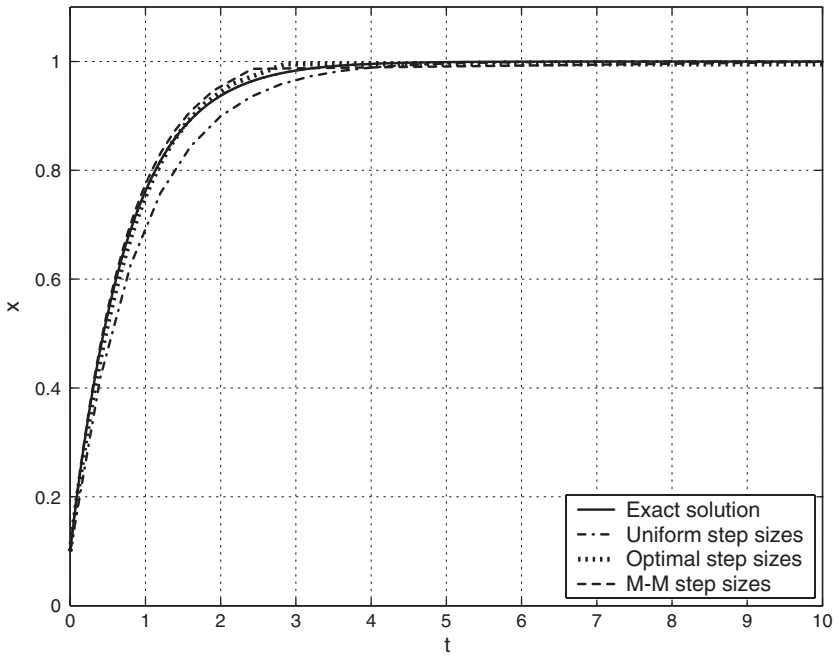


Fig. 1. Exact and approximate state trajectories.

The discount rate follows the recursion

$$\alpha_n = \frac{\alpha_{n-1}}{1 + \rho \frac{1+g}{g}[1 - (1+g)^{-\triangle n}]}$$

for any $\alpha_0 > 0$, and $n \leqslant N - 1$, and the terminal welfare becomes

$$\frac{\alpha_{N-1}}{\rho}\log(c_{t_N}) = \frac{\alpha_{N-1}}{\rho}\log(ak_{t_N}^b - gk_{t_N}),$$

where $\rho$ is the rate of pure time preference in the continuous time analog.

Undoubtedly, the best test of the strength of our method would be to weigh our numerical results against the exact solution to the continuous-time growth model as in the previous section. Unfortunately, however, because of the inherent nonlinearities, no closed form solution exists to allow for such a comparison. To proceed further, we fix the parameter values, $\rho = 0.05/4$, $g = 0.03/4$, $a = 0.2$, $b = 0.24$ as in Mercenier and Michel and approximate the growth model using uniform and optimal time intervals assuming two different initial capital stocks, $k_{t_0} = 2.4$ and $k_{t_0} = 0.01$.

In approximations with uniform grids, we assume $T = 350$ and $N = 35$ so that $\triangle_n = 10$. Thus, whether starting poor, $k_{t_0} = 0.01$, or with a head-start, $k_{t_0} = 2.4$,

economic growth has to be optimized by a choice of 35 investment rates at equally distanced dates of 10 periods. In contrast, when the time intervals are optimized, as per our method, GAs also search for the optimal investment dates, subject to the constraints, $0.01 \leqslant \triangle_n \leqslant 350$, so that $\sum_{n=0}^{34} \triangle_n \leqslant 12,250$. M–M grids, on the other hand, are given by $t_n = (1/\log(\lambda))\log(1-(1-\lambda^T)(n/N))$ for $n = 0, \ldots, N$, where $\lambda$ is the stable root and equal to 0.972.[11]

Table 2 reports our numerical results. Note that a major source of approximation error using time-aggregation along the transient phase is the assumption that controls are constant within given time intervals. Consequently, the sharper are the optimal adjustments during the transition, the larger will be the approximation errors with uniform step sizes. When the economy starts relatively well-off, with the initial capital stock, $k_{t_0} = 2.4$, only 25 percent below the steady-state, the desired adjustments along the state trajectory are relatively small. Nonetheless, even with such a 'smooth' transition to the steady-state, our solution algorithm improves the optimization performance from $-131.8716$ with uniform time intervals to $-117.6428$, a welfare gain of more than 10 percent. Improvement over approximation with M–M time intervals is relatively small from $-119.1260$ to $-117.6428$, a gain of about 1 percent. In contrast, when the economy begins relatively poor, $k_{t_0} = 0.01$, the desired initial adjustments along the state trajectories are sharp, and so is the performance enhancement that comes along with an optimal choice of discretization steps. Our method improves welfare from $-166.4245$ to $-139.0410$ compared to uniform time intervals, a gain of more than 15 percent, and from $-141.3519$ to $-139.0410$ compared to M–M time intervals, a gain of about 2 percent. However, to able to use M–M method, one would again need the dominant stable root from the linearized system dynamics.

From Table 2, it is also apparent that the M–M sequence of investment dates are suboptimal. In contrast, when investment dates are also optimized, we observe a more frequent investment activity at earlier stages when the desired adjustments are larger. As the capital stock approaches its steady-state value, investment becomes less frequent and eventually dies out.

Also noteworthy from Figs. 2 and 3 is the fact that the steady-state invariant optimal time aggregation method captures the stationary equilibrium reasonably well. Though, the steady-state performance worsens a bit when $k_0 = 0.01$, this is more than offset with the superior performance along the transient phase. Obviously, in order to improve the steady-state performance, either $T$ can be fixed at a relatively larger number, e.g., $T = 350$, or the number of sample points $N$ can be increased, or both. While the former is suboptimal, the latter is simply costly. Ultimately, given $N$, optimal choice of time aggregation with steady-state equivalence better captures the transition dynamics thus providing a better approximation to the continuous model.

[11] Mercenier and Michel (1994a) find the stable root by assuming equal unit grids and linearizing the discrete dynamics around the steady-state with the above parameters.

Table 2
Optimal investment, consumption and capital stock[a]

| Uniform grids | | | | | Optimal grids | | | | Uniform grids | | | | Optimal grids | | | | M–M grids | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k_{t_0} = 2.4000$ | | | | | | | | | $k_{t_0} = 0.0100$ | | | | | | | | $k_{t_0} = 0.0100$ | | | | $k_{t_0} = 2.4000$ | |
| $n$ | $\triangle_n$ | $t_n$ | $k_{tn}$ | $c_{tn}$ | $\triangle_n$ | $t_n$ | $k_{tn}$ | $c_{tn}$ | $\triangle_n$ | $t_n$ | $k_{tn}$ | $c_{tn}$ | $\triangle_n$ | $t_n$ | $k_{tn}$ | $c_{tn}$ | $\triangle_n$ | $t_n$ | $k_{tn}$ | $c_{tn}$ | $k_{tn}$ | $c_{tn}$ |
| 0 | 10 | 0 | 2.4000 | 0.2121 | 0.01 | 0.0000 | 2.4000 | 0.2061 | 10 | 0 | 0.0100 | 0.0439 | 0.0100 | 0.0000 | 0.0100 | 0.0161 | 1.0207 | 0.0000 | 0.0100 | 0.0293 | 2.4000 | 0.2077 |
| 1 | 10 | 10 | 2.5602 | 0.2183 | 1.8585 | 0.0100 | 2.4002 | 0.2079 | 10 | 10 | 0.2241 | 0.0925 | 0.2343 | 0.0100 | 0.0105 | 0.0217 | 1.0511 | 1.0207 | 0.0473 | 0.0434 | 2.4214 | 0.2086 |
| 2 | 10 | 20 | 2.6866 | 0.2230 | 1.9823 | 1.8685 | 2.4385 | 0.2097 | 10 | 20 | 0.6610 | 0.1286 | 0.3186 | 0.2443 | 0.0211 | 0.0277 | 1.0835 | 2.0718 | 0.1020 | 0.0550 | 2.4428 | 0.2096 |
| 3 | 10 | 30 | 2.7860 | 0.2267 | 2.1115 | 3.8508 | 2.4773 | 0.2114 | 10 | 30 | 1.1167 | 0.1551 | 0.4134 | 0.5629 | 0.0373 | 0.0341 | 1.1179 | 3.1553 | 0.1664 | 0.0651 | 2.4642 | 0.2105 |
| 4 | 10 | 40 | 2.8641 | 0.2296 | 2.2331 | 5.9623 | 2.5162 | 0.2131 | 10 | 40 | 1.5186 | 0.1749 | 0.5227 | 0.9763 | 0.0605 | 0.0410 | 1.1545 | 4.2731 | 0.2370 | 0.0742 | 2.4857 | 0.2115 |
| 5 | 10 | 50 | 2.9252 | 0.2318 | 2.3664 | 8.1954 | 2.5550 | 0.2148 | 10 | 50 | 1.8531 | 0.1897 | 0.6289 | 1.4990 | 0.0920 | 0.0484 | 1.1937 | 5.4277 | 0.3121 | 0.0825 | 2.5071 | 0.2124 |
| 6 | 10 | 60 | 2.9731 | 0.2336 | 2.4634 | 10.5618 | 2.5936 | 0.2165 | 10 | 60 | 2.1244 | 0.2011 | 0.7469 | 2.1279 | 0.1319 | 0.0557 | 1.2356 | 6.6213 | 0.3908 | 0.0903 | 2.5286 | 0.2133 |
| 7 | 10 | 70 | 3.0107 | 0.2349 | 2.6068 | 13.0252 | 2.6311 | 0.2181 | 10 | 70 | 2.3416 | 0.2098 | 0.8842 | 2.8748 | 0.1811 | 0.0632 | 1.2805 | 7.8569 | 0.4721 | 0.0977 | 2.5501 | 0.2143 |
| 8 | 10 | 80 | 3.0400 | 0.2360 | 2.7594 | 15.6320 | 2.6682 | 0.2198 | 10 | 80 | 2.5140 | 0.2165 | 1.0065 | 3.7590 | 0.2409 | 0.0720 | 1.3288 | 9.1374 | 0.5556 | 0.1046 | 2.5717 | 0.2153 |
| 9 | 10 | 90 | 3.0630 | 0.2368 | 2.9188 | 18.3914 | 2.7045 | 0.2213 | 10 | 90 | 2.6502 | 0.2217 | 1.1087 | 4.7655 | 0.3091 | 0.0791 | 1.3809 | 10.4662 | 0.6411 | 0.1114 | 2.5932 | 0.2162 |
| 10 | 10 | 100 | 3.0810 | 0.2374 | 3.0404 | 21.3102 | 2.7400 | 0.2229 | 10 | 100 | 2.7574 | 0.2257 | 1.2628 | 5.8742 | 0.3856 | 0.0870 | 1.4373 | 11.8472 | 0.7280 | 0.1179 | 2.6148 | 0.2172 |
| 11 | 10 | 110 | 3.0950 | 0.2379 | 3.0858 | 24.3506 | 2.7738 | 0.2243 | 10 | 110 | 2.8416 | 0.2288 | 1.4122 | 7.1370 | 0.4722 | 0.0959 | 1.4985 | 13.2845 | 0.8163 | 0.1241 | 2.6362 | 0.2181 |
| 12 | 10 | 120 | 3.1059 | 0.2383 | 3.2516 | 27.4364 | 2.8054 | 0.2256 | 10 | 120 | 2.9077 | 0.2312 | 1.5364 | 8.5492 | 0.5669 | 0.1042 | 1.5651 | 14.7830 | 0.9058 | 0.1301 | 2.6577 | 0.2191 |
| 13 | 10 | 130 | 3.1145 | 0.2386 | 3.4429 | 30.6880 | 2.8359 | 0.2270 | 10 | 130 | 2.9594 | 0.2331 | 1.6513 | 10.0856 | 0.6675 | 0.1116 | 1.6379 | 16.3481 | 0.9963 | 0.1359 | 2.6792 | 0.2200 |
| 14 | 10 | 140 | 3.1211 | 0.2389 | 3.5265 | 34.1309 | 2.8650 | 0.2282 | 10 | 140 | 2.9999 | 0.2345 | 1.7900 | 11.7369 | 0.7736 | 0.1194 | 1.7179 | 17.9860 | 1.0879 | 0.1416 | 2.7007 | 0.2210 |
| 15 | 10 | 150 | 3.1263 | 0.2391 | 3.5696 | 37.6574 | 2.8919 | 0.2293 | 10 | 150 | 3.0316 | 0.2357 | 1.9362 | 13.5269 | 0.8849 | 0.1272 | 1.8060 | 19.7039 | 1.1802 | 0.1473 | 2.7222 | 0.2220 |
| 16 | 10 | 160 | 3.1303 | 0.2392 | 3.7294 | 41.2270 | 2.9166 | 0.2303 | 10 | 160 | 3.0563 | 0.2366 | 2.0445 | 15.4631 | 1.0005 | 0.1345 | 1.9036 | 21.5099 | 1.2732 | 0.1526 | 2.7436 | 0.2229 |
| 17 | 10 | 170 | 3.1334 | 0.2393 | 3.8189 | 44.9564 | 2.9400 | 0.2313 | 10 | 170 | 3.0756 | 0.2373 | 2.2432 | 17.5076 | 1.1177 | 0.1419 | 2.0125 | 23.4135 | 1.3669 | 0.1578 | 2.7649 | 0.2238 |
| 18 | 10 | 180 | 3.1358 | 0.2394 | 3.9802 | 48.7753 | 2.9615 | 0.2322 | 10 | 180 | 3.0906 | 0.2378 | 2.3867 | 19.7508 | 1.2399 | 0.1495 | 2.1345 | 25.4260 | 1.4614 | 0.1631 | 2.7862 | 0.2247 |
| 19 | 10 | 190 | 3.1376 | 0.2395 | 4.1389 | 52.7555 | 2.9815 | 0.2330 | 10 | 190 | 3.1023 | 0.2382 | 2.5177 | 22.1375 | 1.3618 | 0.1566 | 2.2723 | 27.5604 | 1.5563 | 0.1682 | 2.8076 | 0.2257 |
| 20 | 10 | 200 | 3.1390 | 0.2395 | 4.1088 | 56.8944 | 3.0000 | 0.2338 | 10 | 200 | 3.1116 | 0.2386 | 2.6065 | 24.6552 | 1.4826 | 0.1631 | 2.4291 | 29.8327 | 1.6517 | 0.1732 | 2.8290 | 0.2266 |
| 21 | 10 | 210 | 3.1401 | 0.2396 | 4.2247 | 61.0032 | 3.0163 | 0.2345 | 10 | 210 | 3.1186 | 0.2388 | 2.7844 | 27.2617 | 1.5998 | 0.1693 | 2.6092 | 32.2618 | 1.7476 | 0.1782 | 2.8502 | 0.2276 |
| 22 | 10 | 220 | 3.1408 | 0.2396 | 4.3907 | 65.2279 | 3.0312 | 0.2351 | 10 | 220 | 3.1242 | 0.2390 | 2.9224 | 30.0461 | 1.7167 | 0.1751 | 2.8181 | 34.8710 | 1.8439 | 0.1830 | 2.8714 | 0.2285 |
| 23 | 10 | 230 | 3.1415 | 0.2396 | 4.3935 | 69.6186 | 3.0448 | 0.2356 | 10 | 230 | 3.1286 | 0.2392 | 3.1555 | 32.9685 | 1.8311 | 0.1812 | 3.0634 | 37.6890 | 1.9407 | 0.1877 | 2.8925 | 0.2294 |
| 24 | 10 | 240 | 3.1419 | 0.2397 | 5.1597 | 74.0121 | 3.0568 | 0.2361 | 10 | 240 | 3.1320 | 0.2393 | 3.4363 | 36.1240 | 1.9439 | 0.1867 | 3.3555 | 40.7524 | 2.0378 | 0.1924 | 2.9135 | 0.2303 |
| 25 | 10 | 250 | 3.1421 | 0.2397 | 4.8255 | 79.1718 | 3.0692 | 0.2366 | 10 | 250 | 3.1345 | 0.2394 | 3.4945 | 39.5603 | 2.0566 | 0.1923 | 3.7093 | 44.1079 | 2.1355 | 0.1970 | 2.9346 | 0.2312 |
| 26 | 10 | 260 | 3.1424 | 0.2397 | 5.1351 | 83.9973 | 3.0793 | 0.2370 | 10 | 260 | 3.1367 | 0.2394 | 3.7609 | 43.0548 | 2.1600 | 0.1974 | 4.1465 | 47.8172 | 2.2335 | 0.2016 | 2.9556 | 0.2321 |
| 27 | 10 | 270 | 3.1424 | 0.2397 | 5.4035 | 89.1324 | 3.0886 | 0.2375 | 10 | 270 | 3.1382 | 0.2395 | 4.3269 | 46.8157 | 2.2598 | 0.2022 | 4.7008 | 51.9638 | 2.3320 | 0.2061 | 2.9766 | 0.2330 |
| 28 | 10 | 280 | 3.1425 | 0.2396 | 6.0808 | 94.5359 | 3.0966 | 0.2378 | 10 | 280 | 3.1394 | 0.2395 | 4.6224 | 51.1426 | 2.3620 | 0.2063 | 5.4265 | 56.6646 | 2.4309 | 0.2105 | 2.9975 | 0.2338 |
| 29 | 10 | 290 | 3.1429 | 0.2397 | 6.1936 | 100.6167 | 3.1043 | 0.2381 | 10 | 290 | 3.1404 | 0.2396 | 4.9753 | 55.7650 | 2.4609 | 0.2109 | 6.4179 | 62.0911 | 2.5303 | 0.2149 | 3.0185 | 0.2347 |
| 30 | 10 | 300 | 3.1431 | 0.2397 | 7.1529 | 106.8103 | 3.1112 | 0.2384 | 10 | 300 | 3.1413 | 0.2396 | 5.8064 | 60.7403 | 2.5530 | 0.2148 | 7.8543 | 68.5090 | 2.6302 | 0.2193 | 3.0394 | 0.2356 |
| 31 | 10 | 310 | 3.1431 | 0.2397 | 8.2371 | 113.9632 | 3.1174 | 0.2386 | 10 | 310 | 3.1421 | 0.2396 | 7.2861 | 66.5467 | 2.6462 | 0.2193 | 10.1249 | 76.3634 | 2.7308 | 0.2236 | 3.0602 | 0.2365 |
| 32 | 10 | 320 | 3.1432 | 0.2397 | 10.3165 | 122.2003 | 3.1235 | 0.2389 | 10 | 320 | 3.1426 | 0.2396 | 9.588 | 73.8328 | 2.7413 | 0.2235 | 14.2673 | 86.4883 | 2.8323 | 0.2279 | 3.0812 | 0.2373 |
| 33 | 10 | 330 | 3.1433 | 0.2396 | 14.9238 | 132.5168 | 3.1291 | 0.2391 | 10 | 330 | 3.1432 | 0.2396 | 14.6531 | 83.4208 | 2.8403 | 0.2278 | 24.3774 | 100.7556 | 2.9354 | 0.2322 | 3.1023 | 0.2382 |
| 34 | 10 | 340 | 3.1438 | 0.2397 | 31.2014 | 147.4406 | 3.1349 | 0.2394 | 10 | 340 | 3.1436 | 0.2397 | 30.4822 | 98.0739 | 2.9492 | 0.2325 | 224.8670 | 125.1330 | 3.0421 | 0.2376 | 3.1241 | 0.2393 |
| 35 | | 350 | 3.1443 | 0.2397 | | 178.6420 | 3.1414 | 0.2397 | | 350 | 3.144 | 0.2397 | | 128.5561 | 3.0861 | 0.2388 | | 350.0000 | 3.1250 | 0.2395 | 3.1409 | 0.2397 |

[a]Total discounted utilities under uniform, optimal and M–M time intervals are respectively, −131.8716, −117.6428 and −119.1260 when $k_0 = 2.4000$, and −166.4245, −139.0410 and −141.3519 when $k_0 = 0.0100$.

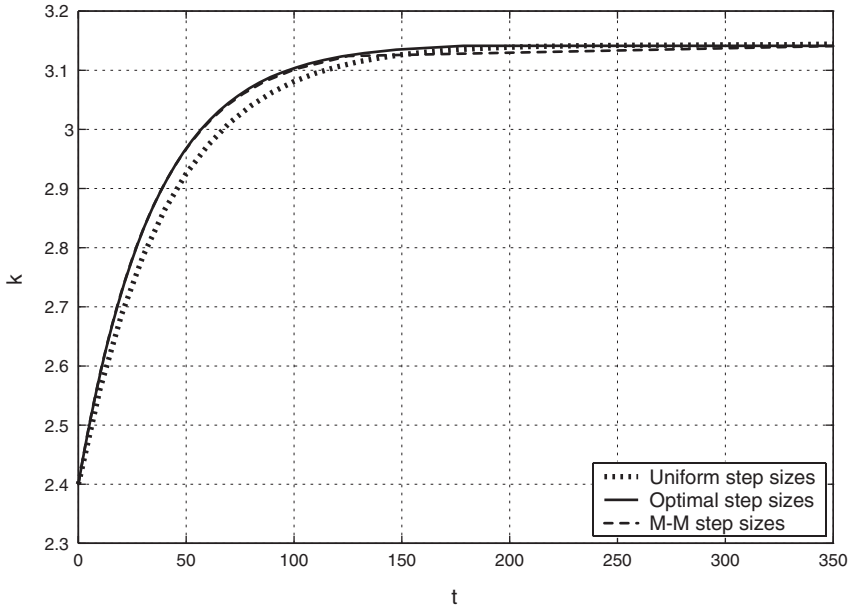


Fig. 2. $k_0 = 2.4000$: capital stock.

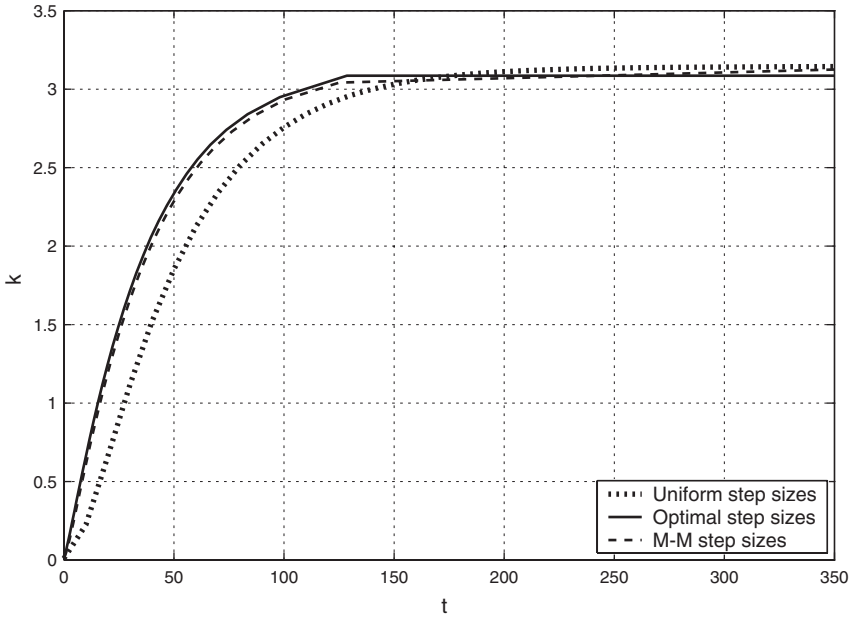


Fig. 3. $k_0 = 0.0100$: capital stock.

### 4.3. Example III: permanent growth

Consider now the Lucas' (1988) model with a constant population that is normalized to unity:

$$J = \max \int_0^\infty \mathrm{e}^{-\rho t} c(t)^a \,\mathrm{d}t$$

s.t.

$$\dot{K}(t) = F(K(t), w(t)H(t)\bar{H}(t)^\gamma) - c(t) - \mu K(t), \quad K(0) = K_0 \text{ given},$$
$$\dot{H}(t) = \delta(1 - w(t))H(t), \quad H(0) = H_0 \text{ given},$$

where $c$ is consumption, $K$ and $H$ are physical and human capital, $w$ is the share of non-leisure time devoted to the production of goods and $1 - w$ the share devoured to human capital accumulation, an overbar denotes an externality, $F(.)$ has constant returns, and $\gamma$, $\delta$ and $\mu$ are all positive scalars. Using the following transformations:

$$h(t) = H(t)\bar{H}(t)^\gamma, \quad x(t) = K(t)/h(t), \quad v(t) = c(t)/h(t)$$

and rearranging we get

$$J = \max \int_0^\infty \mathrm{e}^{-\rho t} h(t)^a v(t)^a \,\mathrm{d}t$$

s.t.

$$\dot{x}(t) = F(x(t), w(t)) - c(t) - \mu x(t) - \delta(1 - w(t))x(t) - \gamma\delta(1 - \bar{w}(t))x(t) - v(t),$$
$$\frac{\dot{h}(t)}{h(t)} = \delta(1 - w(t)) + \gamma\delta(1 - \bar{w}(t)),$$
$$x(0) = x_0 \text{ given}, \quad h(0) = h_0 \text{ given}.$$

For direct numerical optimization, the problem is first recast in the time aggregated form as

$$J_\mathrm{d} = \max \sum_{n=0}^{N-1} \frac{\triangle_n}{y_{t_{n+1}}} v_{t_n}^a + \beta_N \frac{v_{t_N}^a}{\rho - a(\delta(1 - w_{t_N}) + \gamma\delta(1 - \bar{w}_{t_N}))}$$

s.t.

$$x_{t_{n+1}} - x_{t_n} = \triangle_n(F(x_{t_n}, w_{t_n}) - c_{t_n} - \mu x_{t_n} - \delta(1 - w_{t_n})x_{t_n} - \gamma\delta(1 - \bar{w}_{t_n})x_{t_n} - v_{t_n}), \quad x_0 = 0.5,$$
$$y_{t_{n+1}} = y_{t_n}[1 + \triangle_n(\rho - a(\delta(1 - w_{t_n}) + \gamma\delta(1 - \bar{w}_{t_n})))], \quad y_0 = 1.0$$

with $\beta_N = 1/y_{t_N}$.

Assuming temporal aggregation with uniform intervals, we fix the number of inputs, $N$, at 25 and assume the system to have reached the steady-state at $T = 500$ so that $\triangle = 20$. If time aggregation is optimal, on the other hand, control intervals must also obey, $0.01 \leqslant \triangle_n \leqslant 500$ so that $\sum_{n=0}^{24} \triangle_n \leqslant 12,500$. A quarterly model serves

as benchmark. We use the following parameter values: $a = 0.1$, $\gamma = 0.01$, $\delta = 0.05/4$, $\rho = 0.04/4$, $\mu = 0.04/4$ and $\beta = 0.25$. We generate the M–M grids using $t_n = \log(1 - n/(N+1))/(1/T)\log(1/(N+1))$ for $n = 0, \ldots, N$. This formula builds on the criterion in Mercenier and Michel (1994b) and is proposed by Mercenier and Michel (2001) for endogenous growth models.[12]

As mentioned earlier we use parallel GAs to approximate the model. Every generation, $GA^{\mathrm{F}}$ evolves a population of externalities, $\bar{w}_{t_n}$s best of which are passed to the $GA^{\mathrm{O}}$ which breeds a population of $w_{t_n}$, $v_{t_n}$ and $\triangle_n$ to optimize economic growth. Simultaneously, $GA^{\mathrm{O}}$ copies the best performing individual $w_{t_n}$ to the computer shared memory so that $GA^{\mathrm{F}}$ can evaluate fitness of the individual forecasts in its population. $GA^{\mathrm{F}}$ ranks forecasts in the population, $\bar{w}_{t_n}$s, according to their fitness based on, $J_f = \min(\sum_{n=0}^{N-1} |\bar{w}_{t_n} - w_{t_n}|^2)^{1/2}$. The Nash equilibrium of this game between the optimizer and the forecaster are: $w_{t_n}$, $v_{t_n}$ and $\triangle_n$ that maximize economic growth and $\bar{w}_{t_n}$ that solves the fixed point problem $\bar{w}_{t_n} = w_{t_n}$.

From Table 3, observe that optimal policy calls for a full allocation of time to production until physical capital grows sufficiently large. This is so since labor can immediately increase production while the physical capital is still relatively low. Subsequently, more time is devoted to accumulate human capital thereby increasing the efficiency of now reduced labor time. Moreover, as also depicted in Figs. 4 and 5, all these transient adjustments are rather fast so that the economy can get on the balanced growth path as quickly as possible. Consequently, on balance, welfare that accrues along the steady-state path weighs relatively more than the welfare enjoyed along the transient phase. Though, the transition dynamics do not feature prominently in this model, nonetheless, optimal time aggregation improves welfare albeit, by small margins. More significantly, qualitative features of the transition dynamics, such as initial full allocation of time to production, are better captured under optimal time aggregation.

## 5. Computational efficiency

Thus far, we have emphasized the improved accuracy due to the optimal time-aggregation of the transient phase. Enhanced accuracy, however, also comes with higher computational costs. Since, the intervals are also optimized, the number of variables subject to search increases by $N - 1$. It can be argued that perhaps the same, or maybe a better degree of accuracy can be achieved with the equivalent computational resources by simply adding $N - 1$ sample points to a transient phase that is divided either by a uniform or M–M gridding scheme thereby shrinking the time intervals. A theoretical exposition of this query is beyond the scope of this

[12]Note that this discretization scheme does not require the stable root of the linearized system. When this measure is used for discretization, the regulator problem has a minimum cost of 2.1254 while the transient growth problem has a maximum welfare of $-121.7689$ when $k(0) = 2.4$ and $-146.6402$ when $k(0) = 0.01$. Since, the general performance is substantially worse than for the other criterion, we do not report the results in detail.

Table 3
Permanent growth model[a]

| | Uniform grids | | | | | | Optimal grids | | | | | | M–M grids | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $\triangle_n$ | $t_n$ | $x_{tn}$ | $v_{tn}$ | $w_{tn}$ | $\bar{w}_{tn}$ | $\triangle_n$ | $t_n$ | $x_{tn}$ | $v_{tn}$ | $w_{tn}$ | $\bar{w}_{tn}$ | $\triangle_n$ | $t_n$ | $x_{tn}$ | $v_{tn}$ | $w_{tn}$ | $\bar{w}_{tn}$ |
| 0 | 20 | 0 | 0.5000 | 0.4860 | 0.7760 | 0.7780 | 0.4813 | 0.0000 | 0.5000 | 0.2933 | 1.0000 | 0.9999 | | 0.0000 | 0.5000 | 0.4510 | 0.9894 | 0.9894 |
| 1 | 20 | 20 | 4.5660 | 0.9130 | 0.8030 | 0.8010 | 0.5796 | 0.4813 | 0.7611 | 0.3469 | 1.0000 | 0.9997 | 6.0190 | 6.0190 | 2.7762 | 0.7406 | 0.9916 | 0.9918 |
| 2 | 20 | 40 | 9.9760 | 1.1750 | 0.7960 | 0.7950 | 0.7060 | 1.0608 | 1.0970 | 0.4012 | 1.0000 | 0.9995 | 6.2647 | 12.2837 | 5.9963 | 0.9531 | 0.9391 | 0.9392 |
| 3 | 20 | 60 | 13.9290 | 1.3200 | 0.7880 | 0.7880 | 0.8257 | 1.7668 | 1.5285 | 0.4634 | 1.0000 | 0.9996 | 6.5314 | 18.8150 | 9.1000 | 1.1068 | 0.8913 | 0.8913 |
| 4 | 20 | 80 | 16.3200 | 1.3990 | 0.7840 | 0.7850 | 0.9676 | 2.5926 | 2.0514 | 0.5279 | 1.0000 | 0.9992 | 6.8217 | 25.6368 | 11.7124 | 1.2179 | 0.8556 | 0.8555 |
| 5 | 20 | 100 | 17.6700 | 1.4410 | 0.7810 | 0.7800 | 1.1205 | 3.5601 | 2.6787 | 0.5956 | 1.0000 | 0.9998 | 7.1391 | 32.7759 | 13.7781 | 1.2982 | 0.8302 | 0.8303 |
| 6 | 20 | 120 | 18.4020 | 1.4620 | 0.7790 | 0.7790 | 1.3261 | 4.6806 | 3.4148 | 0.6647 | 0.9999 | 0.9995 | 7.4875 | 40.2634 | 15.3514 | 1.3561 | 0.8128 | 0.8279 |
| 7 | 20 | 140 | 18.7960 | 1.4740 | 0.7780 | 0.7780 | 1.4279 | 6.0067 | 4.2905 | 0.7380 | 0.9998 | 0.9997 | 7.8717 | 48.1351 | 16.5212 | 1.3972 | 0.8007 | 0.8007 |
| 8 | 20 | 160 | 19.0070 | 1.4810 | 0.7780 | 0.7780 | 1.5945 | 7.4346 | 5.2301 | 0.8118 | 1.0000 | 0.9995 | 8.2974 | 56.4325 | 17.3725 | 1.4264 | 0.7927 | 0.7927 |
| 9 | 20 | 180 | 19.1230 | 1.4850 | 0.7780 | 0.7780 | 1.6062 | 9.0290 | 6.2636 | 0.8757 | 1.0000 | 0.9990 | 8.7717 | 65.2042 | 17.9819 | 1.4463 | 0.7871 | 0.7870 |
| 10 | 20 | 200 | 19.1820 | 1.4860 | 0.7780 | 0.7780 | 1.7206 | 10.6352 | 7.3710 | 0.9474 | 0.9999 | 0.9991 | 9.3037 | 74.5079 | 18.4130 | 1.4614 | 0.7834 | 0.7837 |
| 11 | 20 | 220 | 19.2150 | 1.4880 | 0.7780 | 0.7760 | 1.8233 | 12.3557 | 8.5131 | 1.0134 | 0.9991 | 0.9985 | 9.9043 | 84.4122 | 18.7003 | 1.4704 | 0.7810 | 0.7809 |
| 12 | 20 | 240 | 19.2310 | 1.4860 | 0.7770 | 0.7770 | 1.9629 | 14.1790 | 9.7176 | 1.0741 | 1.0000 | 0.9998 | 10.5879 | 95.0001 | 18.8966 | 1.4780 | 0.7801 | 0.7796 |
| 13 | 20 | 260 | 19.2420 | 1.4860 | 0.7770 | 0.7750 | 1.9799 | 16.1419 | 10.9247 | 1.1329 | 1.0000 | 0.9995 | 11.3729 | 106.3730 | 19.0244 | 1.4807 | 0.7789 | 0.7789 |
| 14 | 20 | 280 | 19.2490 | 1.4870 | 0.7780 | 0.7790 | 2.0314 | 18.1217 | 12.0549 | 1.1868 | 0.9825 | 0.9810 | 12.2837 | 118.6567 | 19.1162 | 1.4835 | 0.7782 | 0.7782 |
| 15 | 20 | 300 | 19.2770 | 1.4880 | 0.7770 | 0.7780 | 2.2829 | 20.1531 | 13.2619 | 1.2395 | 0.9476 | 0.9487 | 13.3531 | 132.0098 | 19.1730 | 1.4857 | 0.7779 | 0.7777 |
| 16 | 20 | 320 | 19.2560 | 1.4870 | 0.7770 | 0.7780 | 2.4475 | 22.4360 | 14.3682 | 1.3010 | 0.9033 | 0.9041 | 14.6267 | 146.6365 | 19.2032 | 1.4865 | 0.7777 | 0.7777 |
| 17 | 20 | 340 | 19.2510 | 1.4900 | 0.7780 | 0.7780 | 3.0678 | 24.8835 | 15.4166 | 1.3300 | 0.8817 | 0.8794 | 16.1690 | 162.8055 | 19.2210 | 1.4870 | 0.7775 | 0.7774 |
| 18 | 20 | 360 | 19.2290 | 1.4880 | 0.7780 | 0.7780 | 3.7250 | 27.9513 | 16.5182 | 1.3808 | 0.8430 | 0.8427 | 18.0754 | 180.8809 | 19.2286 | 1.4874 | 0.7776 | 0.7776 |
| 19 | 20 | 380 | 19.2490 | 1.4880 | 0.7780 | 0.7780 | 4.1939 | 31.6762 | 17.3353 | 1.4089 | 0.8171 | 0.8169 | 20.4922 | 201.3732 | 19.2339 | 1.4872 | 0.7775 | 0.7777 |
| 20 | 20 | 400 | 19.2580 | 1.4870 | 0.7770 | 0.7770 | 5.6623 | 35.8701 | 18.0785 | 1.4470 | 0.7924 | 0.7934 | 23.6566 | 225.0297 | 19.2431 | 1.4875 | 0.7775 | 0.7773 |
| 21 | 20 | 420 | 19.2570 | 1.4890 | 0.7780 | 0.7790 | 6.9911 | 41.5324 | 18.4750 | 1.4583 | 0.7848 | 0.7849 | 27.9798 | 253.0095 | 19.2486 | 1.4877 | 0.7775 | 0.7774 |
| 22 | 20 | 440 | 19.2540 | 1.4890 | 0.7780 | 0.7790 | 7.7180 | 48.5234 | 18.7488 | 1.4639 | 0.7830 | 0.7827 | 34.2445 | 287.2539 | 19.2499 | 1.4882 | 0.7776 | 0.7775 |
| 23 | 20 | 460 | 19.2510 | 1.4890 | 0.7780 | 0.7780 | 11.6085 | 56.2414 | 19.0890 | 1.4877 | 0.7889 | 0.7904 | 44.1488 | 331.4027 | 19.2363 | 1.4876 | 0.7776 | 0.7776 |
| 24 | 20 | 480 | 19.2490 | 1.4890 | 0.7780 | 0.7780 | 13.2025 | 67.8499 | 19.1215 | 1.4805 | 0.7787 | 0.7761 | 62.2242 | 393.6270 | 19.2479 | 1.4880 | 0.7776 | 0.7777 |
| 25 | 20 | 500 | 19.2490 | 1.4890 | 0.7780 | 0.7770 | | 81.0523 | 19.2314 | 1.4874 | 0.7774 | 0.7774 | 106.3730 | 500.0000 | 19.2377 | 1.4876 | 0.7775 | 0.7774 |

[a]Respective maximum utilities with uniform, optimal and M–M time intervals are 103.9175, 104.9467 and 104.6046. Respective forecast errors are 0.005 and 0.005 and 0.0150.

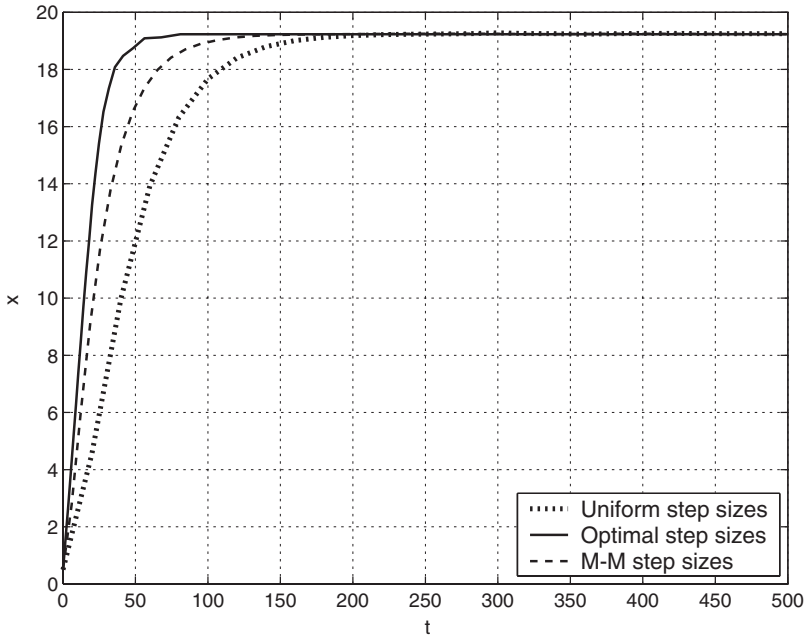


Fig. 4. Transformed capital stock.

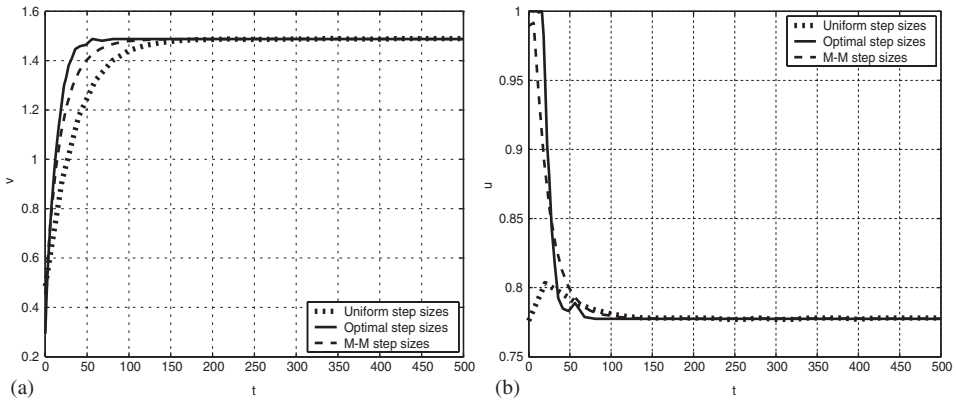


Fig. 5. (a) transformed consumption (b) labor time[13].

paper. Instead, we experiment on the sample problems to test this proposition. For each test problem, we repeat the optimization experiment from different random populations for 30 times.

[13]The observed non-monotonicity in the labor time is due to approximation and certainly not optimal.

Table 4
Comparison of computational efficiency

| | Uniform grid | Optimal grid | M–M grid |
|---|---|---|---|
| *Regulator* | | | |
| # of generations | 1000 | 1000 | 1000 |
| Average performance | 3.4351 | 2.4678 | 6.5350 |
| Standard deviation | 0.3029 | 0.0634 | 1.8537 |
| CPU time | 45.39s | 42.71s | 44.72s |
| *Transient growth* | | | |
| # of generations | 7000 | 7000 | 7000 |
| Average performance | −124.3065 | −117.3030 | −120.1896 |
| Standard deviation | 1.4233 | 1.0107 | 1.7124 |
| CPU time | 1071.04s | 872.14s | 1323.33s |
| *Permanent growth* | | | |
| # of generations | 2000 | 2000 | 2000 |
| Average performance | 104.1450 | 104.6165 | 104.5313 |
| Standard deviation | 0.0141 | 0.0436 | 0.0611 |
| CPU time | 737.14s | 547.10s | 735.11s |

When decision intervals are objects of search, we keep the number of sample points, $N$, unchanged. However, $N$ is increased for uniform and M–M gridding so that all three methods search for the same number of optimal values. We increase $N$ to 50 in the regulator, 70 in the transient growth and 37 in the permanent growth problems so that the respective uniform step sizes are now reduced to 0.2, 5 and 13.5135. We increase numbers of sample points in the aforementioned formulas to generate the M–M grids which are now denser. All sample problems are approximated 30 times each of which starts with a randomly initialized population of 50. For each problem, we increase the number of generations by increments of 1000 until all 30 runs contain at least one feasible solution in the population. A summary of the numerical findings is reported in Table 4.

Numerical experiments indicate that with comparable computational effort, optimal time-aggregation of transient phase yields better average performance. Furthermore, this conclusion is robust since optimal time-aggregation also reduces the variance across random initial populations. From Table 4, it is also noteworthy that an added advantage of direct optimization of discretization steps is its speed. The method proposed in this paper not only improves accuracy, but it computes faster than the other alternative methods. Given the extra time needed for paper and pencil derivation and offline computation of the stable root for the M–M method, we conclude that our direct optimization method complements the steady-state invariant time aggregation by increasing its efficiency.

We attribute the increased efficiency in our method to the fact that our search algorithm evolves not only a fixed number of sample points on the state trajectories, but also the associated distances between them. Thus, the fitness of a candidate solution in the population depends not only on how well it approximates the 'level'

of the sample points, but also on how accurately it represents the 'speed' with which a sample point is reached from another. Consequently, step-sizes are accordingly adjusted in evolutionary steps to improve overall performance thereby speeding up learning of the transient phase. Ultimately, our method reduces approximation errors not only at the steady-state, but also during transition phase. Of course, further numerical accuracy can be achieved by increasing the number of control actions, $N$, at the expense of larger computational costs.

## 6. Conclusion

This paper has shown that an optimal choice of discretization steps in time-aggregated models with steady-state invariance would significantly enhance the numerical approximation of the transient dynamics. Conversely, if the length of the transient phase is arbitrarily set and uniformly partitioned as is usually done, numerical results can deteriorate dramatically. An alternative scheme that sets nonuniform intervals is the M–M method. Since, this approach relies on the stable root of the linearized dynamics around the steady-state, it is less efficient than the direct method we propose.

We have demonstrated the benefits from our method in three sample problems: a simple regulator, a transitory and an endogenous growth models. All experiments indicate that optimal timing of control actions also becomes important in time-aggregated discrete models. Furthermore, repeated experiments from random initial populations, have shown the proposed method to be numerically efficient. Extension of our results to time-optimal control problems is immediate.

## Acknowledgements

An earlier version of this paper was presented under the title 'Optimal Discretization of Continuous Time Control Problems' at the Seventh International Conference of the Society for Computational Economics at Yale University. We would like to thank Tarık Kara and Süheyla Özyıldırım for their helpful suggestions and comments. Of course, any remaining errors are ours.

## References

Alemdar, N.M., Özyıldırım, S., 1998. A genetic game of trade, growth and externalities. Journal of Economic Dynamics and Control 22, 811–832.

Alemdar, N.M., Sirakaya, S., 2003. On-line computation of Stackelberg equilibria with synchronous parallel genetic algorithms. Journal of Economic Dynamics and Control 27 (8), 1503–1515.

Bellman, R.E., 1961. Adaptive Control Processes: A Guided Tour. Princeton University Press, Princeton.

Betts, J., 1998. Survey of Numerical Methods for Trajectory Optimization. Journal of Guidance, Control and Dynamics 21 (2), 193–207.

Bryson Jr., A.E., 1999. Dynamic Optimization. Addison-Wesley, Reading, MA.

Bryson, Jr., A.E., Ho, Y.Ch., 1975. Applied optimal control. Optimization, Estimation and Control, Hemisphere.

Burnside, C., 2001. Discrete state-space methods for the study of dynamic economies. In: Marimon, R., Scott, A. (Eds.), Computational Methods for the Study of Dynamic Economies. Oxford University Press, Oxford, pp. 95–113.

Candler, G.V., 2001. Finite-difference methods for continuous-time dynamic programming. In: Marimon, R., Scott, A. (Eds.), Computational Methods for the Study of Dynamic Economies. Oxford University Press, Oxford, pp. 172–194.

Daniel, J.W., 1976. Splines and efficiency in dynamic programming. Journal of Mathematical Analysis and Applications 54, 402–407.

Falcone, M., Ferretti, R., 1998. Convergence analysis for a class of high-order semi-Lagrangian advection schemes. SIAM Journal of Numerical Analysis 35, 909–940.

Flam, S.D., Wets, R.J.-B., 1987. Existence results and finite approximates for infinite horizon optimization problems. Econometrica 55, 1187–1209.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Grefenstette, J.J., 1986. Optimization of control parameters for genetic algorithm. IEEE Transactions on Systems, Man and Cybernetics 16, 122–128.

Grefenstette, J.J., 1990. A user's guide to GENESIS Version 5.0.

Grüne, L., 1997. An adaptive grid scheme for the discrete Hamilton–Jacobi–Bellman equation. Numerische Mathematice 75 (3), 319–337.

Grüne, L., Semmler, W., 2003. Solving asset pricing models with stochastic dynamic programming. Working Paper No. 54, Center for Empirical Macroeconomics, Universität Bielefeld.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI.

Hull, D., 1997. Conversion of optimal control problems into parameter optimization problems. Journal of Guidance, Control and Dynamics 20 (1), 57–60.

Jermann, U.J., 1998. Asset pricing in production economies. Journal of Monetary Economics 41, 257–275.

Johnson, S.A., Stedinger, J.R., Shoemaker, C.A., Li, Y., Tejada-Guibert, J.A., 1993. Numerical solution of continuous-state dynamic programs using linear and spline interpolation. Operations Research 41, 484–500.

Judd, K.L., 1992. Projection methods for solving aggregate growth models. Journal of Economic Theory 58, 410–452.

Judd, K.L., 1996. Approximation, perturbation, and projection methods in economic analysis. In: Amman, H.M., Kendrick, D.A., Rust, J. (Eds.), Handbook of Computational Economics. Elsevier, Amsterdam, pp. 511–585 (Chapter 12).

Kehoe, T.J., Levine, D.K., 1992. On characterizing equilibria of economies with externalities and taxes as solutions to optimization problems. Economic Theory 2, 43–68.

Kraft, D., 1994. TOMP-Fortran modules for optimal control calculations. ACM Transactions on Mathematical Software 20 (3), 262–281.

Lucas, R.E., 1988. On the mechanics of economic development. Journal of Monetary Economics 22, 3–42.

Marcet, A., 1994. Simulation analysis of stochastic dynamic models: applications to theory and estimation. In: Sims, C.A. (Ed.), Advances in Econometrics, Sixth World Congress of the Econometric Society. Cambridge University Press, Cambridge, pp. 81–118.

Mercenier, J., Michel, P., 1994a. Discrete-time finite horizon approximation of infinite horizon optimization problems with steady-state invariance. Econometrica 62, 635–656.

Mercenier, J., Michel, P., 1994b. A criterion for time aggregation in intertemporal dynamic models. Mathematical Programming 64, 179–197.

Mercenier, J., Michel, P., 2001. Temporal aggregation in a multi-sector economy with endogenous growth. Journal of Economic Dynamics and Control 25, 1179–1191.

Munos, R., Moore, A., 2002. Variable resolution discretization in optimal control. Machine Learning 49, 291–323.

Reiter, M., 1999. Solving higher-dimensional continuous-time stochastic control problems by value function regression. Journal of Economic Dynamics and Control 23, 1329–1353.

Rust, J., 1996. Numerical dynamic programming in economics. In: Amman, H.M., Kendrick, D.A., Rust, J. (Eds.), Handbook of Computational Economics. Elsevier, Amsterdam, pp. 620–729.

Rust, J., 1997. Using randomization to break the curse of dimensionality. Econometrica 65, 478–516.

Sirakaya, S., Alemdar, N.M., 2003. Genetic Neural Networks to Approximate Feedback Nash Equilibria in Dynamic Games. Computers and Mathematics with Applications 46 (10–11), 1493–1509.

Tauchen, G., Hussey, R., 1991. Quadrature-based methods for obtaining approximate solutions to nonlinear asset-price models. Econometrica 59, 371–396.

Trick, M.A., Zin, S.E., 1993. A linear programming approach to solving stochastic dynamic programs. Working paper, Carnegie-Mellon University.

Trick, M.A., Zin, S.E., 1997. Spline approximations to value functions: a linear programming approach. Macroeconomic Dynamics 1, 255–277.

Von Stryk, O., Bulirsch, R., 1992. Direct and indirect methods for trajectory optimization. Annals of Operations Research 37, 357–373.