Journal of Economic Dynamics & Control 34 (2010) 4–27

Contents lists available at ScienceDirect

Journal of Economic Dynamics & Control

journal homepage: www.elsevier.com/locate/jedc

Comparison of solutions to the incomplete markets model with aggregate uncertainty

Wouter J. Den Haan [*]

Department of Economics, University of Amsterdam, Roetersstraat 11, 1018 WB Amsterdam, The Netherlands and CEPR

ARTICLE INFO

Article history:
Received 20 October 2008
Accepted 20 December 2008
Available online 11 August 2009

JEL classification:
C63

Keywords:
Numerical solutions
Approximations

ABSTRACT

This paper compares numerical solutions to the model of Krusell and Smith [1998. Income and wealth heterogeneity in the macroeconomy. Journal of Political Economy 106, 867–896] generated by different algorithms. The algorithms have very similar implications for the correlations between different variables. Larger differences are observed for (i) the unconditional means and standard deviations of individual variables, (ii) the behavior of individual agents during particularly bad times, (iii) the volatility of the per capita capital stock, and (iv) the behavior of the higher-order moments of the cross-sectional distribution. For example, the two algorithms that differ the most from each other generate individual consumption series that have an average (maximum) difference of 1.63% (11.4%).

1. Introduction

This paper compares different algorithms to solve the model of Krusell and Smith (1998), a popular model with a continuum of heterogeneous agents, idiosyncratic as well as aggregate risk, incomplete markets, and an inequality constraint on the chosen capital level.[1] Models with heterogeneous agents and incomplete markets are becoming more and more important in both macro and finance. Consequently, it is important that we know how to solve them well. This is not a trivial task, because (i) the set of state variables includes the cross-sectional distribution of income and wealth levels, a high dimensional object, (ii) the amount of idiosyncratic uncertainty is quite high so nonlinearities are likely to matter, and (iii) the occasionally binding constraint results in non-differentiable policy functions.

There are now several algorithms that can solve this type of model and in this paper I investigate whether the different algorithms generate similar answers when solving the model of Krusell and Smith (1998). The solutions turn out to differ substantially in several dimensions. This is especially true for the individual choices. Not only do the generated series differ during exceptional periods, such as particularly bad times, but there are even nontrivial differences between the implied first moments. Several accuracy checks are performed. Overall, the algorithm of Reiter (2009b) performs best in terms of accuracy. It clearly performs the best in terms of the accuracy of the individual policy rules and the accuracy of its aggregate law of motion is close to the most accurate aggregate laws of motion, which are the ones obtained with the Krusell–Smith algorithm. The performance of the algorithm of den Haan and Rendahl (2009) is close to the performance of Reiter (2009b) in terms of accuracy, but slightly worse. Interestingly, the algorithms of den Haan and Rendahl (2009) and

[*] Tel.: +31 20 5255237; fax: +31 20 5254254.
E-mail address: wdenhaan@uva.nl
[1] The actual version of the model considered and its parameter values can be found in den Haan, Judd and Juillard et al. (2009).

0165-1889/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.jedc.2008.12.010

Reiter (2009b) are also the fastest, with the algorithm of den Haan and Rendahl (2009) roughly seven times as fast as the algorithm of Reiter (2009b).[2]

The model considered here is a nontrivial model, but there are much more complex models considered in the literature. The fact that the different algorithms generate results that are not that similar, should motivate us to be careful in numerically solving these models. There are some useful lessons that can be learned from this comparison project. Those are the following:

- It is essential to have an algorithm for the individual problem that does well in terms of accuracy as well as speed. Standard lessons from the numerical literature, for example, that time iteration is typically faster and more reliable than fixed-point iteration, should not be ignored.[3] Also, the lower and upper bounds of the grid should be chosen with care in order not to exclude useful grid points. Grid points should also not be wasted; for this project, I find that the algorithms that use the largest range for individual capital also have lower accuracy. Finally, the method of endogenous grid points, proposed in Carroll (2006), is recommended. It is not clear whether this leads to a more accurate solution, but it is definitely faster and makes it, for example, easy to implement time iteration.
- It is important to realize that the properties of an algorithm found when solving for individual policy rules in the model *without* aggregate uncertainty, i.e., for a fixed capital stock level, do not carry over to the model *with* aggregate uncertainty, even when taking as given the law of motion for aggregate capital. In particular, this paper shows that it is more difficult to get accurate individual policy rules in the model with than in the model without aggregate uncertainty (even when taking the aggregate law of motion as given).
- In solving models with a representative agent, it is typically possible to achieve arbitrary accuracy. None of the algorithms considered here do extremely well in terms of all the accuracy tests. Especially the outcomes of the accuracy test for the aggregate policy rule are somewhat disappointing.[4] The maximum errors in a simulation of 10,000 observations vary across algorithms from 0.156% to 1.059%. Ideally, these should be at least a factor 10 smaller than the lowest values generated here.
- Given that it is not (yet) easy to generate numerical solutions with arbitrary accuracy, it is important to perform accuracy tests. The role of a good accuracy procedure consists not only of providing a measurement of the accuracy of the solution, but also of making clear which aspect of the solution is inaccurate when and whether the inaccuracies found matter.
- Algorithms and computers will get better and the model considered in this paper will hopefully soon be solved with arbitrary accuracy. But models are likely to get more complex at a faster rate, so numerical solutions will be generated that—like those considered in this paper—do reasonably well, but not exceptionally well in all accuracy tests considered. The question is what to do in such cases. Numerical solutions can fail accuracy tests and still give the right answer to the question the researcher is interested in.[5] The best accuracy test is, therefore, to "play around" with different choices, such as different classes of approximating functions and/or different grids, and to see whether the results of interest change. It would be even more convincing if the results do not change if a different algorithm is used to solve the model. Algorithms differ substantially in their programming burden. But both the popular Krusell–Smith algorithm and the recently developed algorithm of den Haan and Rendahl (2009) are quite simple to program, so the researcher does no longer have a good excuse not to try more than one algorithm.

This paper is organized as follows. In Section 2, I give a brief overview of the different algorithms used. In Section 3, I discuss the differences between the algorithms in solving a model without aggregate uncertainty and in Section 4 I discuss the differences between the algorithms in solving the full model, that is, the model with aggregate uncertainty. The last section concludes.

2. Different types of algorithms

Recursive numerical solutions of DSGE models consist of functions of the state variables. Existing algorithms are based on either the projection method or the perturbation method, sometimes on both. The projection method consists of two steps. In the first step, a grid of the state variables is constructed and one defines at each grid point error terms that provide a measure for the fit of any approximating function.[6] The second step consists of choosing the coefficients of the numerical approximation to obtain the best fit for a given loss function of the error terms.[7] The perturbation approach solves for the

[2] The algorithm of Kim et al. (2009) is even faster, but this algorithm does not solve the actual model specified.

[3] See Chapter 16 in Judd (1998) for a discussion.

[4] This paper uses a more demanding (but much better) accuracy test than the R^2 that is typically used in the literature.

[5] Some accuracy tests are directly linked to properties of interest. Santos (2000) relates the Euler equation residual to errors in the policy function. Reiter (2001) and Santos and Peralta-Alva (2005) construct a relationship between the size of the errors found and upperbounds on the errors of objects that economists are interested in such as the obtained utility level or moments.

[6] Numerical procedures, such as quadrature methods to calculate conditional expectations, may still be needed to calculate the value of the error terms.

[7] Or obtain a perfect fit if there are as many grid points as unknown coefficients.

Table 1
Algorithms and participants.

Abbreviation	Participants
BInduc	Michael Reiter
KS-num	Eric Young
KS-sim	Lilia Maliar, Serguei Maliar, Fernando Valli
Param	Olivier Allais, Yann Algan, Wouter den Haan
Xpa	Wouter den Haan, Pontus Rendahl
Penal	Sunghyun Kim, Robert Kollmann, Jinill Kim

coefficients of the Taylor expansion of the true set of policy functions, $h(x)$, around the steady state. Using $h(x)$, the choice variables can be substituted out of the model equations and one obtains a system of equations with x as the only variable, that is, $F(x) \equiv 0$. The unknown coefficients of the Taylor expansion are found by *sequentially* differentiating this system of equations and evaluating the obtained equations at the steady state.

Three of the participating algorithms are projection methods. Those are the backward induction procedure, referred to as BInduc throughout this paper, developed in Reiter (2009b), the parameterized distribution procedure, referred to as Param, of Algan et al. (2009), and the explicit aggregation algorithm, referred to as Xpa, developed in den Haan and Rendahl (2009). These algorithms are summarized in the subsections on projection methods. The popular Krusell–Smith algorithm, referred to as KS, was developed in Krusell and Smith (1997, 1998). Two versions of this algorithm are used in this project. They differ in how the panel of individual data is simulated. The algorithm of Young (2009), referred to as KS-num, uses a numerical procedure to simulate an economy with a continuum of agents, and the algorithm of Maliar et al. (2009), referred to as KS-sim, follows Krusell and Smith (1997, 1998) and uses a finite number of agents. The KS algorithm is a hybrid procedure, because it uses a standard projections approach to solve for the individual policy rule, but a simulation step to solve for the law of motion of the aggregate variables. The algorithm of Kim et al. (2009), referred to as Penal, is based on the perturbation approach. To be able to use the perturbation method, Kim et al. (2009) replace the inequality constraint by a smooth penalty function.

In the remainder of this section, I give a very brief description of the different algorithms that are part of this comparison project and of the algorithms of Preston and Roca (2006) and Reiter (2009a), that are not. For a more detailed description, I refer the reader to the papers in this issue and to Algan et al. (2008). The participants and the abbreviations of the algorithms are given in Table 1.

KS algorithm: The different algorithms have in common that they follow den Haan (1996, 1997), Krusell and Smith (1997, 1998), and Ríos-Rull (1997) in summarizing the cross-sectional distribution of capital and employment status with a limited set of moments. The KS algorithm specifies a law of motion for these moments and finds the approximating function using a simulation procedure. That is, given a set of individual policy rules, a time series of cross-sectional moments is generated and new laws of motion for the aggregate moments are estimated using the simulated data. KS-sim simulates the economy using a finite number of agents, whereas KS-num uses a numerical procedure to simulate using a continuum of agents. The latter has the advantage of avoiding cross-sectional sampling error.[8] Given the aggregate law of motion, the laws of motion of the individual variables are then updated using standard projection methods.

Projection methods I: *Parameterization of the cross-sectional distribution*. BInduc and Param solve the model using tools from the projection approach. They do not obtain next period's cross-sectional moments by simulation techniques, but by explicitly integrating the individual choices. This allows them to avoid a disadvantage of the algorithm of Krusell and Smith (1998), namely that the points at which the aggregate law of motion is determined are chosen inefficiently.[9]

BInduc and Param avoid using simulation techniques to solve for the coefficients of the approximating functions by parameterizing the cross-sectional distribution. BInduc uses a histogram and Param a flexible polynomial. The time-varying coefficients of the cross-sectional distribution are then the state variables. Both BInduc and Param, still use moments as state variables, but specify a mapping from cross-sectional moments to the coefficients of the cross-sectional distribution. It is not easy to implement this pure projections procedure. The main reason is that describing the cross-sectional distribution accurately requires several coefficients and, thus, quite a few state variables. To deal with this problem, both BInduc and Param reduce the number of coefficients and, thus, the number of state variables by using a simulation to learn about the appropriate shape of the cross-sectional distribution. The simulation part of the algorithm fulfills, however, only a supporting role in the algorithm.

Another problem of these two algorithms is that it is not always clear how to construct a sensible density. For example, if moments are used as state variables, then some combinations of moments may not correspond to an actual density.

Projection method II: *No parameterization of the cross-sectional distribution*: Xpa derives the aggregate laws of motion directly from the individual policy rules by simply aggregating them. Xpa avoids numerical integration techniques and the

[8] The KS algorithm relies on the idea that next period's moments are perfectly forecastable, which is at best approximately true in a simulation with a finite number of agents.

[9] Just like in standard regression analysis, one would like the explanatory variables to be spread out, whereas in simulated data, they tend to be clustered around the mean.

Table 2
Computation times.

Algorithm	Programming language	Time
BInduc	Matlab	47 min
KS-num	Fortran	324 min
KS-sim	Matlab	310 min
Param	Fortran	2739 min
Xpa	Matlab	7 min
Penal	Matlab	<1 s!

Notes: This table reports the time it takes to solve the model when $\gamma = 1.1$, starting at the solution for $\gamma = 1$.

need to specify a cross-sectional distribution, as is done by BInduc and Param, by writing the individual policy functions as a linear combination of basis functions of the individual state variables.[10] This analytic approach makes immediately clear that at least the cross-sectional averages of all the basis functions that enter the individual policy functions should be state variables. In fact, a strict implementation of the idea of explicit aggregation implies that *all* higher-order cross-sectional moments, i.e., an infinite set, should be included, unless the individual policy function is exactly linear. Xpa deals with this infinite dimensionality issue as follows. Suppose that the individual policy rule for x_{t+1} is a second-order polynomial of x_t. Explicit aggregation implies that the cross-sectional means of x_t and x_t^2 should definitely be included as a state variable. Predicting the cross-sectional average of x_{t+1}^2 requires an approximation for x_{t+1}^2. Instead of using the square of the policy function for x_{t+1}, which would include fourth-order terms, Xpa uses an *auxiliary* policy rule that is also of second-order. The cross-sectional average of x_{t+1}^2 can then be obtained by explicitly aggregating this auxiliary policy rule.

The individual policy function used to obtain the law of motion of the aggregate variables does not have to be the same as the individual policy function used to describe individual behavior. In particular, Xpa uses splines for the individual policy, but uses a smooth approximation to the spline to obtain the aggregate law of motion.[11] Xpa is a standard projection technique and can use all the standard tools such as optimal location of the grid points.

Perturbation approaches: The approximating functions used by perturbation techniques are continuous and differentiable. Consequently, they are not well suited for a model like the one considered here with occasionally binding constraints. Preston and Roca (2006) solve a model very similar to the one considered here, but with the borrowing constraint replaced by a penalty function and the finite-state Markov process replaced by a stochastic process with continuous support. Kim et al. (2009), like Preston and Roca (2006), replace the borrowing constraint by a penalty function and use a stochastic process with continuous support, but they do not use the perturbation approach to obtain the aggregate laws of motion the way Preston and Roca (2006) do. Instead, they simply use the solution from a representative-agent economy without a constraint or penalty function as the aggregate law of motion. Consequently, they solve a model that differs in two aspects from the model solved by the other participants. First, the penalty function limits borrowing in a different way than the borrowing constraint. Second, the law of motion for aggregate capital is not derived from the law of motion for the individual variable. Comparison of the solution of Penal with the others is like comparing apples and oranges. Nevertheless, it will be interesting to see how far off the solution obtained using this method is, because this algorithm solves the model in less than 1 s, which is much faster than any of the other algorithms.

There are two important algorithms that are not part of this comparison project. The first is the "pure" perturbation approach of Preston and Roca (2006). Unlike Kim et al. (2009), they derive both the individual and the aggregate laws of motion using a standard perturbation approach. A nice feature of the approach of Preston and Roca (2006) is that there is a very direct link between the individual policy rule and the corresponding aggregates, similar to the link established by Xpa.

A fair comparison of the algorithm of Preston and Roca (2006) with the ones considered here would require using a penalty function instead of a non-negativity constraint on capital. The same is, of course, true for the Penal algorithm of Kim et al. (2009). The model of this project has a non-negativity constraint, because it is commonly used in the literature. The use of this type of constraint in the literature seems to be mainly motivated by common practice and by its convenience in terms of calibration, *not* by realism. Penalty functions imply that there are costs associated with lower levels of capital and with short positions in capital, that increase as capital decreases. The borrowing constraint also corresponds to a penalty function, namely one that is zero for non-negative capital stocks and infinite for negative values. It is obviously not sensible to argue that this extreme choice is to be generally preferred to smooth penalty functions.[12]

[10] There is no restriction on how aggregate variables enter.

[11] The requirement that the cross-sectional moment of each basis function has to be a state variables makes clear that explicit aggregation could be quite expensive if splines are used to approximate the individual policy function. Splines can still be written as a linear combination of basis functions (B-splines), but typically quite a few nodes and, thus, quite a few basis functions are used. Consequently, quite a few cross-sectional moments would have to be included as state variables.

[12] Perturbation approaches use the derivatives at the steady state to solve the model. Consequently, low-order perturbation approaches require that the penalty function is not too flat at the steady state. den Haan and de Wind (2008) investigate the ability of the perturbation approach to solve models with different types of penalty functions accurately.

Table 3
Computational intensity in solving individual problem.

Algorithm	# Nodes	Functional form	Range	Location
BInduc	500	Spline	[0, 392.9]	Extra nodes around constraint
KS-num	150	Spline	[0, 1200]	Extra nodes around constraint
KS-sim	100	Spline	[0, 1000]	Extra nodes around constraint
Param	50	27th-order Chebyshev pol.	[0, 99]	Chebyshev nodes
Xpa	250	Spline	[0, 200]	Extra nodes around constraint

Notes: The range given is not exactly comparable across methods, because BInduc and Xpa use the endogenous grid points procedure of Carroll (2006) in which case the range is for k_{t+1}, whereas for the other procedures the range is for k_t.

The second missing algorithm is the hybrid perturbation/projection approach of Reiter (2009a). This algorithm starts out by solving the model without aggregate uncertainty using projection methods. Since projection methods are used, the individual policy functions can easily deal with non-differentiabilities caused by features such as borrowing constraints. The next step is a perturbation step. The solution of the first step consists of the coefficients of the individual policy function, ψ_1, and the coefficients that describe the cross-sectional distribution, ψ_2. The idea is to perturb this solution for ψ_1 and ψ_2 around the case of no aggregate uncertainty. The coefficients ψ_1 and ψ_2 are like the variables in a standard perturbation analysis and the question is how they change if aggregate uncertainty is being introduced. That is, the procedure of Reiter (2009a) describes how individual behavior changes with changes in the amount of aggregate uncertainty.

Computational speed: Table 2 reports the time it takes for the different algorithms to solve the model when γ is equal to 1.1, taking as initial conditions the solution of the model when the coefficient of relative risk aversion, γ, is equal to 1.[13] One always should be careful in interpreting these numbers. They may say more about the work the different participants put into writing a fast algorithm, than about the lowest possible computing time that is attainable with the different algorithms. For a project like this one, in which only one model is solved, speed is not that important. Despite these caveats, the differences in computing times are so large that some useful lessons can be drawn.

There are enormous differences in speed across the different algorithms. Penal takes less than one second, but this algorithm only solves two individual problems. Nevertheless, perturbation algorithms are likely to outperform projection procedures in speed. Therefore, future research should compare solutions to complex models with heterogeneous agents solved with perturbation approaches, like the one developed by Preston and Roca (2006), with the solutions obtained with projection procedures.

Even when we only focus on the algorithms that solve the actual model with heterogeneous agents, then there are enormous differences. The fastest algorithm is Xpa that takes less than 7 min and the slowest is Param that takes 2,739 min, almost 400 times as slow as Xpa. BInduc is second with 47 min and the two KS algorithms are similar with computing times of 324 and 310 min for KS-num and KS-sim, respectively.

It is clear, why Xpa is fast. The most important reason is that explicit aggregation makes it possible to get the aggregate policy rules from the individual policy rules almost instantaneously. But another important reason is that the individual problem is solved efficiently. In particular, Xpa (i) uses time iteration instead of fixed-point iteration, (ii) uses endogenous grid points so that there is not a complex non-linear problem at each grid point, and (iii) uses an efficient grid.

It is also clear why Param is so slow. The first reason is that it uses the largest number of aggregate state variables. In particular, it uses four aggregate state variables in addition to the current aggregate state, a_t. The second reason is that in addition to the individual and the aggregate problem it also has an outer loop to obtain information about the correct shape of the cross-sectional distribution. The third reason is that it has to solve a non-linear problem in determining which cross-sectional distribution corresponds to the set of moments. Moreover, the individual problem is solved inefficiently. The first time the algorithm solves for the individual policy rule, for a given aggregate law of motion, it takes roughly 590 min.[14] It takes so long, not only because of the many state variables, but also because Param uses fixed-point iteration and to ensure convergence, it has to use a very low updating coefficient.

3. Results for the model without aggregate uncertainty

In this part of the paper, I focus on the model without aggregate uncertainty. In particular, the aggregate capital stock is equal to 43 (not the equilibrium value), the probability of moving out of employment (unemployment) is equal to 0.4 (0.955555), and the unemployment rate is equal to 0.10.

[13] The programs were run on a Dell Latitude D410 with an Intel Pentium M processor (2.00 GHz, 798 MHz FSB).

[14] The later individual problems take less time, because they have better starting values, but the speed of this first individual problem is still indicative of the slowness of Param.

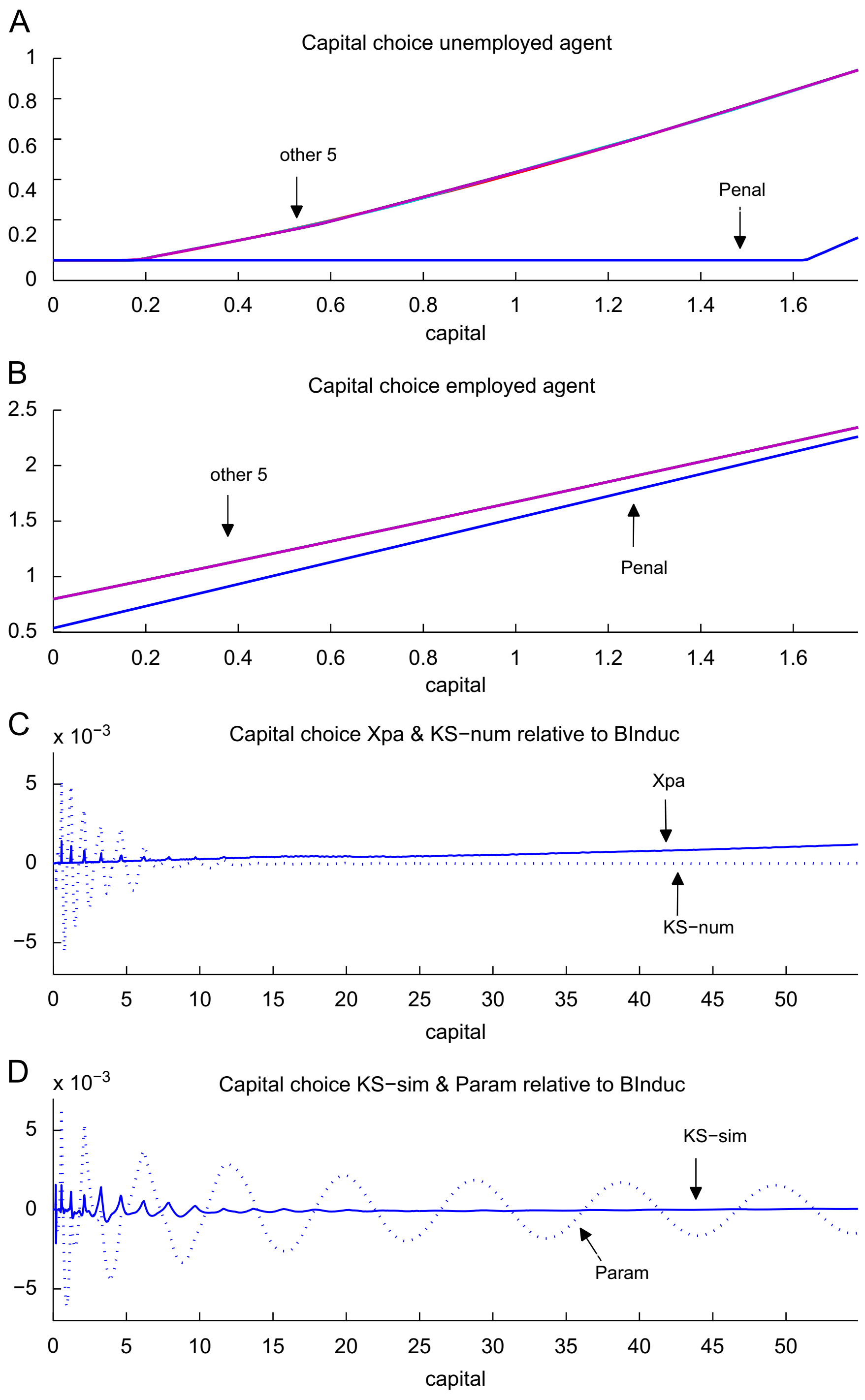

Fig. 1. Policy rule (no aggregate uncertainty). Notes: Panels A and B plot the policy rule for capital at low values of capital. Panels C and D plot the policy choices relative to the policy choice according to BInduc. In the simulation, the mean capital stock is 12.5; the maximum is equal to 43 (the initial condition); the maximum when the initial transition from the high initial value is excluded is equal to 18.7.

3.1. Information about computational intensity

The solution of this version of the model consists of an individual capital choice as a function of the beginning-of-period capital stock, for the employed and the unemployed. Participants were required to solve this individual problem with the same computational complexity as they used to solve the individual problem in the model *with* aggregate uncertainty. For

Table 4
Value of k at which constraint binds.

	BInduc	KS-num	KS-sim	Param	Xpa	Penal
$k' = 0$ for k equal to	0.177	0.184	0.145	0.176	0.177	1.62

Table 5
Properties policy function—no aggregate uncertainty.

	BInduc	KS-num	KS-sim	Param	Xpa	Penal
Mean c_t^i	2.632	2.632	2.632	2.632	2.632	2.705
Mean k_t^i	12.48	12.49	12.48	12.47	12.50	27.34
St. dev. c_t^i	0.193	0.193	0.193	0.193	0.192	0.143
St. dev. k_t^i	3.738	3.737	3.738	3.731	3.742	7.229
Correlation c_t^i and k_t^i	0.824	0.824	0.824	0.825	0.825	0.961

Notes: These properties are based on the outcomes of a simulation of 10,000 observations using identical realizations for the exogenous driving process.

example, if 100 equidistant grid points for k in the interval $[0, 100]$ were used when solving the model with aggregate uncertainty, then they had to do the same for the version without aggregate uncertainty.

Information about these choices is given in Table 3. The differences in the number of nodes and the range considered are enormous and these differences turn out to be important. None of the algorithms used equidistant nodes to construct the grid for k. Most used a transformation of k to obtain a grid with more nodes close to the constraint.[15] BInduc and KS-num use value function iteration to solve for the individual policy rule and the other algorithms use a method that is based on the Euler equation.

3.2. Policy functions

Panels A and B of Fig. 1 plot the policy functions for the capital choice of the unemployed and the employed agent, respectively. The figure only plots the policy function for low values of capital to highlight the behavior around the kink. The policy functions generated by BInduc, KS-num, KS-sim, Param, and Xpa are indistinguishable from each other. The policy function generated by Penal, however, clearly differs from the others. According to the Penal algorithm, the capital choices of both the unemployed and the employed lie below those generated by the other algorithms. These differences do not disappear when a broader range of values for the capital input are considered. The different predictions of the Penal solution also show up in Table 4, that reports the value of capital at which the constraint is just binding for an unemployed agent.

Panels C and D of Fig. 1 zoom in on the differences between the policy functions (excluding Penal) by plotting the differences between the indicated policy function and the policy function of BInduc. The largest differences are typically (but not always) observed for low values of capital, which is not surprising given the nearby presence of the constraint. Panel C plots the differences between Xpa and BInduc and between KS-num and BInduc. For low values of capital the differences between Xpa and BInduc are very small, whereas they are substantially larger between KS-num and BInduc. For larger values of the capital stock the differences between KS-num and BInduc disappear. Between Xpa and BInduc, however, there remains a systematic difference that increases slightly with the capital level. The differences remain small, however. At $k = 50$ the difference is equal to 0.00118, which is equal to 0.0024% of the capital choice made.

Panel D plots the differences between KS-sim and BInduc and between Param and BInduc. The differences between KS-sim and BInduc are similar to the differences between KS-num and BInduc. In particular, for larger values the differences disappear. The differences between the BInduc and Param policy function do not disappear for larger values of the capital stock. In contrast to the differences between Xpa and BInduc, which also did not disappear for larger capital stocks, they are not systematic; instead, they oscillate around zero. Again, the magnitudes of the differences are small.

3.3. Simulated series

For a given sequence of 10,000 realizations of the employment status, time series for consumption and capital are generated. Table 5 reports some summary statistics for the generated series. The statistics of the different algorithms

[15] For example, BInduc uses equidistant nodes for $\ln(k + 0.1w)$, where w is the steady state wage rate.

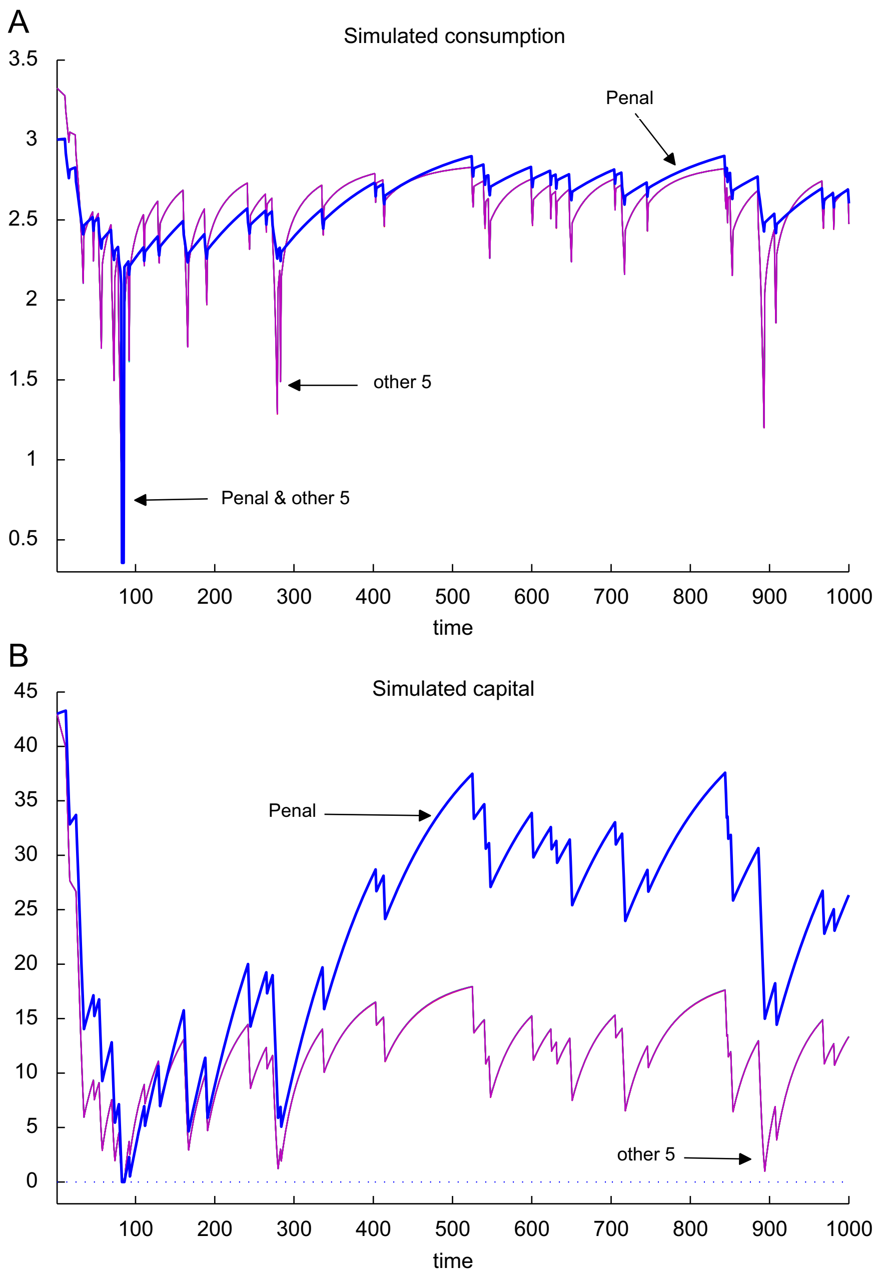

Fig. 2. Simulated individual variables (no aggregate uncertainty). Notes: This graph plots the initial part of simulated series generated by the six algorithms. Note that the initial value chosen is extreme (at least according to all algorithms except Penal) and such a high value of k is not again observed in the later part of the simulation.

are extremely similar except for those generated with Penal. For example, Penal generates a standard deviation for consumption (capital) that is 26% (93%) lower (higher), than the corresponding value generated by the other algorithms. The mean capital stock generated with Penal is more than twice as high as the mean implied by the other algorithms. The penalty function used by Penal provides a strong incentive for agents to accumulate savings. The borrowing constraint imposed by the other algorithms also induce agents to save more, but much less so.

Panels A and B of Fig. 2 plot the first 1,000 observations for consumption and capital, respectively. The series generated by the Penal algorithm are clearly different, whereas the series generated by the other five algorithms are indistinguishable from each other. The consumption series generated by the Penal algorithm increase (decrease) by too little when the agent is employed (unemployed). For capital, the series generated by the Penal algorithm are more volatile than the series generated by the other algorithms. The increased popularity of perturbation techniques will make it more likely that

Table 6
Euler equation errors—no aggregate uncertainty.

	BInduc	KS-num	KS-sim	Param	Xpa	Penal
Average errors						
Unemployed (%)	2.2E−04	1.7E−04	2.1E−03	6.2E−02	1.4E−05	1.4E−02
Employed (%)	4.6E−05	4.0E−05	1.7E−04	6.4E−04	1.1E−05	1.4E−03
Maximum errors						
Unemployed (%)	3.1E−02	3.0E−02	6.8E−01	1.6	4.0E−03	9.2E−01
Attained at $k=$	0.57	0.19	0.20	0.56	99.8	1.63
Employed (%)	2.6E−04	9.3E−04	1.3E−03	1.6E−03	2.9E−05	2.1E−02
Attained at $k=$	0.81	3.58	6.82	0	0.57	0

Notes: Errors are calculated on a grid for k where k varies from 0 to 100 and the step size is equal to 0.01.

authors will modify models as is done by Kim et al. (2009). The results reported here make clear that such modifications can lead to substantially different model properties.[16]

3.4. Accuracy tests

Standard Euler-equation accuracy test: Table 6 reports information about the errors of the Euler equation. Euler-equation errors are calculated on a grid for capital ranging from 0 to 100 with a step size equal to 0.01. The error is calculated as follows. Let $c(k)$ be the consumption value according to the proposed numerical solution and let $\bar{c}(k)$ be the consumption choice implied by the explicitly calculated conditional expectation. The Euler-equation error is then given by

$$100\left|\frac{c(k)-\bar{c}(k)}{\bar{c}(k)}\right|.$$

Xpa attains the smallest errors, but the errors of BInduc and KS-num are of similar magnitude. Note that Xpa uses the second-largest number of nodes for k, namely 250, and the second-smallest range for k. The largest errors are attained by either Param when one looks at the errors for an unemployed agent or by Penal when one looks at the errors for an employed agent. The errors of KS-sim are uniformly better than those of Param and Penal, but they are not substantially smaller.

This accuracy test only checks for inaccuracies by looking one-period ahead and so would not detect the possibility of tiny errors accumulating to larger errors. The next two tests are better equipped to detect this possibility.

Dynamic Euler-equation accuracy test: The idea behind the dynamic Euler-equation accuracy test is to compare a time series for k_t, that is generated with a numerical solution, $k_{t+1}=k(k_t,\varepsilon_t)$, with an alternative series $\tilde{k}_t$, that is constructed as follows.

- $\tilde{k}_1=k_1$,
- $\hat{k}_{t+1}=k(\tilde{k}_t,\varepsilon_t)$, where $\hat{k}_{t+1}$ is only a temporary variable,
- use $\hat{k}_{t+1}$ and $k(\hat{k}_{t+1},\varepsilon_{t+1})$ for the two possible realizations for ε_{t+1} to calculate the conditional expectation, and
- use the Euler equation and the budget constraint to calculate $\tilde{k}_{t+1}$ and the corresponding level of consumption, $\tilde{c}_t$.[17]

That is, $\tilde{k}_{t+1}$ is calculated each period as it is directly implied by the budget constraint and the Euler equation; the numerical solution is only used indirectly, namely to calculate the conditional expectation.

The results are reported in Table 7. For BInduc, KS-sim, and Xpa, the errors, even the maximum errors, are low. For KS-num, Param, and Penal, however, the maximum errors are much higher for this dynamic Euler-equation accuracy test, than for the standard Euler-equation test; for consumption they are equal to 3.8%, 48%, and 499%, respectively.

The dynamic Euler-equation test is a very stringent test and an occasional large error does not necessarily mean that the solution is generally bad. For example, Fig. 3 plots c_t and $\tilde{c}_t$ according to Param for the first 100 observations, which includes the observation for which the maximum error of 48% is attained. The two series are virtually indistinguishable from each other, except when c_t takes on an extremely low value. Moreover, around and at the observation where the maximum error occurs, the series for c_t generated by Param are virtually identical to the series for c_t generated by BInduc, one of the algorithms that has very low errors. That is, the inaccuracy of Param lies in calculating at extreme low points the conditional expectation, which integrates over some states that are even more extreme.

[16] den Haan and de Wind (2008) investigate how properties (true properties and those obtained with low-order perturbation approximations) of models with a penalty function differ from the corresponding properties of models with an inequality constraint.

[17] Strictly speaking one does not need the temporary variable $\hat{k}_{t+1}$. An alternative would be to solve for $\tilde{k}_{t+1}$ directly from the budget constraint and the Euler equation, but this would require solving a somewhat complex non-linear problem each period.

Table 7
Percentage errors from dynamic Euler accuracy test—no aggregate uncertainty.

	BInduc	KS-num	KS-sim	Param	Xpa	Penal
Capital (*scaled error*)						
Average error (%)	0.0000	0.005	0.001	0.015	0.004	11.2
Maximum error (%)	0.0004	0.118	0.011	1.607	0.015	20.1
Consumption (% *error*)						
Average error (%)	0.0001	0.010	0.001	0.019	0.0004	2.39
Maximum error (%)	0.0024	3.830	0.154	47.96	0.2066	499

Notes: The error from this test is the difference between the values of the capital and consumption paths generated with the individual policy functions and the values of the paths that are obtained when each period the values of capital and consumption that are implied by the explicitly calculated conditional expectation are used. Since capital values can be close to zero, the error for capital is calculated as the absolute difference divided by the mean capital stock.

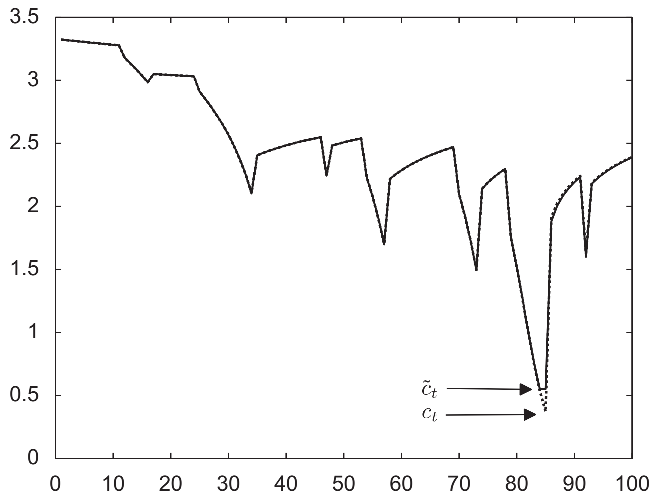

Fig. 3. Individual consumption—simulated using Param. Notes: This graph plots the initial part of the simulated observations for individual consumption using Param and the corresponding value according to the dynamic Euler equation test.

One might conclude from the discussion above that this accuracy test is too stringent. But some algorithms do manage to generate very small errors, so it is possible to pass this demanding test. Moreover, the size of the errors reported here for the model *without* aggregate uncertainty is an indication of how well the algorithm solves the individual problem of the model *with* aggregate uncertainty, but then the errors turn out to be substantially larger, even though the complexity of solving the individual problem is kept the same.

DHM statistic: The den Haan–Marcet (DHM) statistic, proposed in den Haan and Marcet (1994), checks whether the Euler-equation error is orthogonal to elements in the agents' information set using simulated data. Like the dynamic Euler-equation test, it checks whether errors accumulate, puts less weight on those observations that occur less frequently, and puts no weight on those parts of the state space that are not part of the simulation. The latter two properties can in some cases be advantages, but definitely not in all. For example, if a solution incorrectly diverts the simulated time path towards a part of the state space where the errors are small, one would get a misleading answer from this test.

Table 8 reports the results for the DHM accuracy test. Because of the constraint, we focus on the Euler-equation error times the capital choice; this product should be orthogonal to elements of the information set in each period t. The choice of instrument does not seem to matter and we simply choose the constant. That is, we check whether the average of this product is equal to zero. We use a sample set of 500 observations, i.e., $T = 500$. Given that the participants provided a sample of 10,000, we can do the test 20 times.

In constructing the earlier accuracy test statistics for Penal, the modified version of the model, i.e., with the penalty function instead of the inequality constraint, is used. Here we check whether the Penal algorithm generates an accurate solution for the model without the penalty function, but with the inequality constraint. The DHM statistic makes clear that the Penal solution is extremely inaccurate as a solution for the model with the inequality constraint. A value of T equal to

Table 8
den Haan–Marcet statistic—no aggregate uncertainty.

	BInduc	KS-num	KS-sim	Param	Xpa	Penal
Average statistic	1.76	1.75	1.75	1.75	1.77	47.4
Times failed (out of 20)	1	1	1	1	1	20

Notes: Each test is based on a sample of 500 observations. The null that the sample average of the error term is equal to zero is rejected at the 5% level, if the statistic is above 3.84. For Penal, I test the accuracy of the approximation as a solution to the original model, i.e., with the inequality constraint and without the penalty function.

500 is a relatively low value for the DHM statistic,[18] but even at this relatively low value for T, the Penal solution is rejected in each of the 20 samples. The statistics for the other algorithms are much better. Using a 5% confidence level, they fail in exactly 5% of the cases.

Discussion: Getting better accuracy for the version of the model without aggregate uncertainty is simply a matter of adding and/or relocating grid points. This version of the model is still cheap to solve, so grid points can easily be added. But choices like the number of grid points had to be the same in the versions of the model with and without aggregate uncertainty. This means that it is easier to add grid points for algorithms like Xpa that can also solve the model with aggregate uncertainty fast. That Param and Penal perform worse is not that surprising. Param uses smooth polynomials, only 50 grid points for k, and even less coefficients. Even though Param only fits the polynomial on that part of the state space where the constraint is not binding, splines are likely to do better than polynomials for problems with inequality constraints.[19] That Penal performs worse is not surprising given that it uses a lower-order approximation, whereas the other algorithms use much more intricate approximations.

4. Results for the model with aggregate uncertainty

4.1. Properties of the individual policy rules

4.1.1. Individual policy rules and the implied behavior of aggregates

For a given sequence of realizations for the aggregate shock, a time series of 10,000 observations for the cross-sectional distribution of the capital level and employment status are generated using the individual policy rules of the different algorithms. The simulation procedure simulates the economy using a continuum of agents and, thus, avoids cross-sectional sampling variation.[20] It is important to note that only individual policy rules are used; if an algorithm also solves for an aggregate law of motion, then it is not used. The values for the aggregate moments that are needed to determine prices or arguments of individual policy functions are calculated from the simulated cross-sectional distribution.

Moments: The simulation provides time series for the cross-sectional nth-order uncentered moment of the capital stock for the employed and unemployed.[21] Table 9 reports the mean and the standard deviation of the generated time series for the cross-sectional moments. For the sample mean of the first-order cross-sectional moment, the value generated by the Penal algorithm is somewhat less than the values generated by the other algorithms. This makes sense. The aggregate law of motion of Penal is simply the solution to the representative-agent economy in which there is no idiosyncratic risk and there is no borrowing constraint. This representative agent has less reason to build up a capital stock as a buffer against large idiosyncratic shocks than the agents in the economy with idiosyncratic shocks. When the results for Penal are excluded, then the differences between the mean capital stocks are quite small. For the mean (across time) of the cross-sectional mean capital stock of the employed (unemployed) the largest difference between the algorithms is equal to 0.13% (0.19%). For the standard deviation of the cross-sectional mean capital stock, the largest differences are substantially larger, namely 3.2% and 2.3% for the employed and unemployed, respectively.

When we look at sample averages of higher-order cross-sectional moments, then substantially larger differences are observed and when we look at the generated volatility in these higher-order cross-sectional moments, then the differences are huge. For example, for the (scaled) 5th-order moment the standard deviation generated by KS-sim is almost *five* times as large as the standard deviation generated by Param. These differences are solely due to differences in the individual policy rule, because everything else in the calculation of these moments is exactly the same across procedures.

The small differences in the sample means of the cross-sectional mean do not reveal that the time series of the cross-sectional means can be quite different and often do not follow each other that closely. This is documented in Table 10, that

[18] For large enough T, the DHM statistic will reject any numerical solution, even if it suffers from very minor inaccuracies.

[19] When the idiosyncratic shock has discrete support, then the constraint is likely to induce non-differentiabilities in the policy function of the unemployed, not only at the value of k when the constraint becomes binding, but also at the levels of k that are such that the constraint will be binding in l periods when the agent is unemployed in the next l periods. An example can be found in Fig. 2 of den Haan (1997).

[20] See the appendix for details.

[21] For $n>1$, the nth-order moment, $M(n)$, is expressed as $M(n)^{1/n}/M(1)$.

Table 9
Means and standard deviations of cross-sectional moments.

	BInduc	KS-num	KS-sim	Param	Xpa	Penal
Means						
1st-order moment of the unemployed	37.67	37.74	37.70	37.67	37.69	37.56
2nd-order moment of the unemployed	1.13	1.15	1.12	1.12	1.13	1.02
3nd-order moment of the unemployed	1.27	1.29	1.23	1.24	1.25	1.04
4th-order moment of the unemployed	1.41	1.44	1.33	1.34	1.37	1.05
5th-order moment of the unemployed	1.54	1.59	1.42	1.42	1.48	1.07
1st-order moment of the employed	39.45	39.49	39.50	39.45	39.47	39.33
2nd-order moment of the employed	1.12	1.13	1.11	1.11	1.11	1.02
3nd-order moment of the employed	1.25	1.27	1.21	1.21	1.23	1.03
4th-order moment of the employed	1.38	1.40	1.30	1.31	1.34	1.05
5th-order moment of the employed	1.50	1.54	1.39	1.39	1.44	1.06
Standard deviations						
1st-order moment of the unemployed	1.4324	1.4356	1.4031	1.4334	1.4303	1.4708
2nd-order moment of the unemployed	0.0167	0.0128	0.0193	0.0068	0.00145	0.0130
3nd-order moment of the unemployed	0.0345	0.0227	0.0411	0.0122	0.0283	0.0234
4th-order moment of the unemployed	0.0543	0.0307	0.0663	0.0168	0.0425	0.0320
5th-order moment of the unemployed	0.0760	0.0369	0.0944	0.0207	0.0570	0.0391
1st-order moment of the employed	0.964	0.9683	0.9532	0.9383	0.9608	0.9961
2nd-order moment of the employed	0.0135	0.0096	0.0159	0.0055	0.0114	0.0098
3nd-order moment of the employed	0.0292	0.0176	0.0354	0.0100	0.0233	0.0182
4th-order moment of the employed	0.0473	0.0242	0.0587	0.0138	0.0360	0.0254
5th-order moment of the employed	0.0675	0.0294	0.0851	0.0172	0.0493	0.0313

Notes: These statistics are based on the outcomes of a simulation of 10,000 observations using identical realizations for the exogenous random shocks. The procedure to simulate a continuum of agents is discussed in the appendix. For $n>1$, the nth-order moment, $M(n)$, is expressed as $M(n)^{1/n}/M(1)$.

reports the average and the maximum absolute deviation between the cross-sectional means in the simulated data generated by the different algorithms. KS-num and KS-sim turn out to have simulated series for the cross-sectional mean capital stock that are quite different from the other algorithms and from each other. Fig. 4 displays the simulated time series for the mean capital stock of the employed in that part of the sample where the largest differences are obtained. It confirms what is reported in the table, namely that KS-num and KS-sim are typically further away from the other algorithms. In this small part of the sample, it looks as if the differences are quite systematic, but that is not true. The relative ranking of the series generated by the different algorithms changes frequently. The graph for the simulated mean capital stock of the unemployed displays a very similar pattern.

Percentiles: Table 11 reports the means of the 5th and 10th percentiles for the capital holdings of the employed and the unemployed. The table reports the outcomes both unconditionally and conditionally on the aggregate state. This table makes very clear that the penalty function used by the Penal algorithm induces a completely different cross-sectional distribution, namely one with much less mass at lower capital stocks than is found by the other algorithms. When the results of the Penal algorithm are excluded, then there are still noticeable differences between the algorithms. In contrast to the results discussed for the means above, the results for the percentiles are very systematic. That is, the relative ranking across algorithms observed for the sample means of the percentiles also holds for each observation in the sample. In particular, KS-num implies the fattest left tail and KS-sim the thinnest right tail (when Penal is excluded).

4.1.2. *Properties of simulated individual data*

Next, simulated time series for individual consumption and capital, calculated using a common set of realizations for the exogenous random variables, are compared. Some procedures solve for an aggregate law of motion, but this is not used in generating these individual variables; all aggregate moments needed to calculate the individual's choices are obtained from the generated cross-sectional distribution. Table 12 reports summary statistics for the behavior of individual consumption and capital.

Penal versus the other algorithms: When we compare Penal with the other algorithms, then the results are much more similar for this case than they are for the case without aggregate uncertainty. In fact, most of the statistics generated by Penal are now similar to the statistics generated by the other algorithms. Possibly, the additional variation in aggregate prices helps in aligning the Penal solution closer to the solution generated by the other algorithms. Another explanation is that the series in the model without aggregate uncertainty are closer to the constraint (because of the low value of the interest rate) and replacing the inequality constraint with a penalty function has then larger effects on the results. However, there are still some important differences. The standard deviation of individual capital implied by Penal is

Table 10
Differences between simulated cross-sectional mean capital stocks.

	BInduc (%)	KS-num (%)	KS-sim (%)	Param (%)	Xpa (%)
Average differences—mean capital stock of unemployed					
BInduc	0				
KS-num	0.176	0			
KS-sim	0.116	0.178	0		
Param	0.018	0.179	0.106	0	
Xpa	0.018	0.124	0.103	0.058	0
Average across algorithms	0.091	0.164	0.128	0.090	0.085
Average differences—mean capital stock of employed					
BInduc	0				
KS-num	0.098	0			
KS-sim	0.146	0.108	0		
Param	0.017	0.101	0.142	0	
Xpa	0.056	0.047	0.113	0.058	0
Average across algorithms	0.079	0.088	0.127	0.079	0.068
Maximum differences—mean capital stock of unemployed					
BInduc	0				
KS-num	0.403	0			
KS-sim	0.436	0.816	0		
Param	0.093	0.490	0.383	0	
Xpa	0.107	0.336	0.522	0.169	0
Average across algorithms	0.260	0.511	0.593	0.284	0.283
Maximum differences—mean capital stock of employed					
BInduc	0				
KS-num	0.235	0			
KS-sim	0.299	0.505	0		
Param	0.081	0.310	0.247	0	
Xpa	0.102	0.154	0.380	0.159	0
Average across algorithms	0.179	0.301	0.358	0.199	0.199

Notes: Using the simulated cross-sectional mean capital stocks, this table reports the average and maximum differences between the algorithms. Differences are calculated using log differences to ensure symmetry.

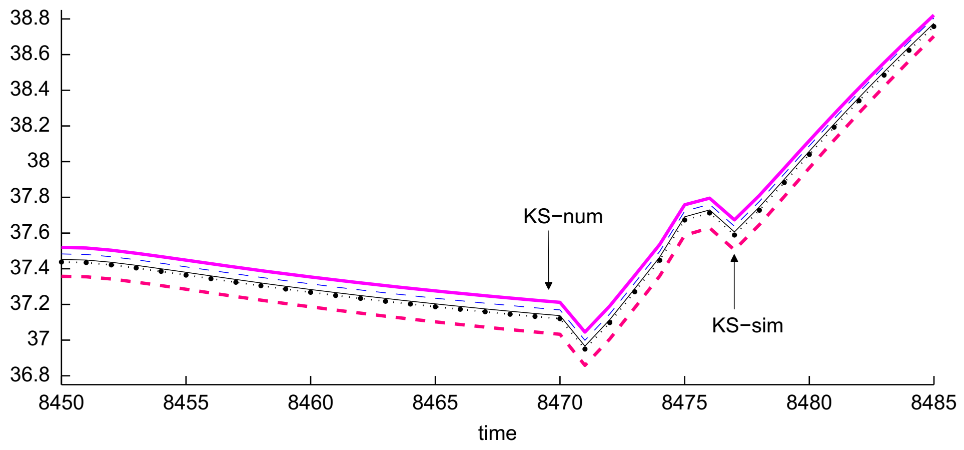

Fig. 4. Simulated mean capital stock of the employed. Notes: This graph plots the simulated cross-sectional mean capital stock of the employed in that part of the sample where the differences between the algorithms take on the largest values.

26–30% below the corresponding numbers of the other algorithms and the mean of individual capital is 23–33% higher depending on the alternative algorithm considered.

Differences in moments of individual variables: Although for most statistics the differences are small, there are some substantial differences between the statistics generated by the algorithms, even when Penal is excluded. Excluding Penal, the highest generated mean of the individual capital stock, obtained by BInduc, is 8.1% higher than the lowest generated

Table 11
Means of the 5th and 10th percentiles.

	BInduc	KS-num	KS-sim	Param	Xpa	Penal
Unemployed						
5%	11.91	11.28	12.28	11.80	12.07	23.86
5%, boom	12.77	12.15	13.16	12.65	12.93	25.32
5%, recession	11.10	10.45	11.44	10.99	11.26	22.47
10%	15.68	15.04	16.12	15.60	15.89	27.38
10%, boom	16.49	15.83	16.95	16.41	16.70	28.76
10%, recession	14.91	14.28	15.32	14.83	15.12	26.08
Employed						
5%	14.05	13.43	14.44	13.92	14.20	26.24
5%, boom	14.26	13.63	14.66	14.13	14.41	26.74
5%, recession	13.85	13.24	14.23	13.12	14.00	25.77
10%	17.68	17.05	18.14	17.58	17.89	29.66
10%, boom	17.88	17.24	18.35	17.78	18.08	30.10
10%, recession	17.49	16.86	17.95	17.40	17.70	29.24

Notes: These properties are based on the outcomes of a simulation of 10,000 observations using identical realizations for the exogenous random shocks.

Table 12
Properties of individual policy rules.

	BInduc	KS-num	KS-sim	Param	Xpa	Penal
Correlations						
c_t^i and C_t	0.269	0.269	0.233	0.263	0.267	0.297
c_t^i and Y_t	0.196	0.194	0.127	0.191	0.195	0.210
c_t^i and K_t	0.264	0.263	0.238	0.257	0.262	0.291
c_t^i and k_t^i	0.921	0.914	0.919	0.926	0.923	0.967
Autocorrelations						
c_t^i and c_{t-1}^i	0.985	0.982	0.980	0.986	0.986	0.992
c_t^i and c_{t-2}^i	0.971	0.966	0.963	0.971	0.972	0.984
c_t^i and c_{t-3}^i	0.957	0.948	0.948	0.957	0.958	0.974
k_t^i and k_{t-1}^i	0.999	0.998	0.999	0.999	0.999	0.997
k_t^i and k_{t-2}^i	0.996	0.995	0.996	0.996	0.996	0.991
k_t^i and k_{t-3}^i	0.992	0.991	0.992	0.992	0.992	0.984
Δc_t^i and Δc_{t-1}^i	−0.084	−0.084	-0.094	−0.081	−0.0793	0.090
Means						
c_t^i	2.75	2.73	2.77	2.75	2.75	2.82
k_t^i	31.14	28.80	31.03	30.37	30.69	38.22
Standard deviations						
c_t^i	0.151	0.150	0.155	0.152	0.151	0.151
k_t^i	10.73	10.29	10.98	10.95	10.86	7.65

Notes: These properties are based on the outcomes of a simulation of 10,000 observations using identical realizations for the exogenous random shocks.

mean capital stock, obtained by KS-num. Similarly, the highest standard deviation of the individual capital stock, generated by KS-sim, is 6.7% higher than the lowest standard deviation, generated by KS-num. These are huge differences.

Differences in simulated individual variables: Table 13 reports the average and maximum deviation between the simulated individual capital and consumption series. For capital as well as for consumption, the largest differences are attained by KS-num and KS-sim when the differences are averaged across algorithms.[22] This is true for both average and maximum differences. The reason for the deviating behavior of KS-num and KS-sim is likely to be related to the fact that they use a much larger range for individual capital in the grid and less grid points.

[22] When the differences are considered across pairs of algorithms, then either KS-num or KS-sim is always one of the two algorithms for which the largest differences are observed.

Table 13
Differences between simulated capital and consumption series across methods.

	BInduc (%)	KS-num (%)	KS-sim (%)	Param (%)	Xpa (%)
Average differences—capital (*scaled difference*)					
BInduc	0				
KS-num	4.54	0			
KS-sim	2.90	7.44	0		
Param	1.27	5.34	2.33	0	
Xpa	1.81	6.35	1.10	1.41	0
Average across algorithms	2.10	4.73	2.75	2.07	2.13
Maximum differences—capital (*scaled difference*)					
BInduc	0				
KS-num	13.1	0			
KS-sim	5.52	17.5	0		
Param	5.96	18.8	3.82	0	
Xpa	3.82	16.5	2.89	2.42	0
Average across algorithms	5.68	13.2	5.94	6.20	5.12
Average differences—consumption (*% difference*)					
BInduc	0				
KS-num	0.49	0			
KS-sim	1.15	1.63	0		
Param	0.15	0.58	1.08	0	
Xpa	0.20	0.69	0.97	0.17	0
Average across algorithms	0.40	0.68	0.97	0.39	0.41
Maximum differences—consumption (*% difference*)					
BInduc	0				
KS-num	7.74	0			
KS-sim	9.81	9.51	0		
Param	1.79	8.96	9.03	0	
Xpa	3.66	11.4	10.8	3.37	0
Average across algorithms	4.60	7.52	7.83	4.63	5.84

Notes: Using the simulated consumption series, this table reports the average and maximum differences between the algorithms; differences are calculated using log differences to ensure symmetry. Since capital values can be close to and equal to zero, the table reports for capital the absolute difference scaled by a weighted average of the means of the two series.

Fig. 5 plots a subsample of the series generated by these five algorithms. It plots the series over that part of the sample when consumption and capital do not take on extreme values and the differences are relatively small. The graph documents the differences in the simulated series and shows that they can be quite persistent. It also indicates that the consumption series generated by KS-sim has some deviating dynamics when the aggregate state changes. Consider a change from the good to the bad aggregate state. All algorithms, except KS-sim, generate an initial drop followed by a less severe drop in the subsequent period (if the low value for the aggregate state persists). KS-sim, however, displays an *increase* for individual consumption when the aggregate state switches from the good to the bad outcome; individual consumption only starts to decrease in the subsequent period.

The largest differences between the series are observed when the constraint is almost binding. An example is given in Fig. 6 that plots the series when the largest difference between the consumption values of KS-num and Xpa are observed.[23] Around the trough, the series follow each other quite closely. In period 6734, the period before the unemployment spell starts, consumption values are 2.45 and 2.47 according to KS-num and Xpa, respectively. But according to KS-num it drops to 1.40 whereas according to Xpa it drops to 1.57. The graph also shows that the other algorithms display a drop close to the value predicted by Xpa.

4.1.3. *Comparing means of individual and aggregate series*

As documented in Table 12, the sample mean of the individual capital stock varies between 10.29 and 10.98 (excluding Penal), whereas according to Table 9 the sample mean of the *per capita* capital stock varies much less. Moreover, the sample means of the individual capital stock are substantially below the sample means of the per capita values.[24] If the sample is long enough, then the sample mean of the individual capital stock should get arbitrarily close to the sample mean

[23] For consumption, the largest difference across all possible pairs of algorithms is found between KS-num and Xpa.
[24] The unconditional mean can be easily calculated from the conditional means. Averaged across the first five algorithms, the sample average of the cross-sectional mean is equal 39.35 and the sample mean of the individual capital stock is equal to 30.4.

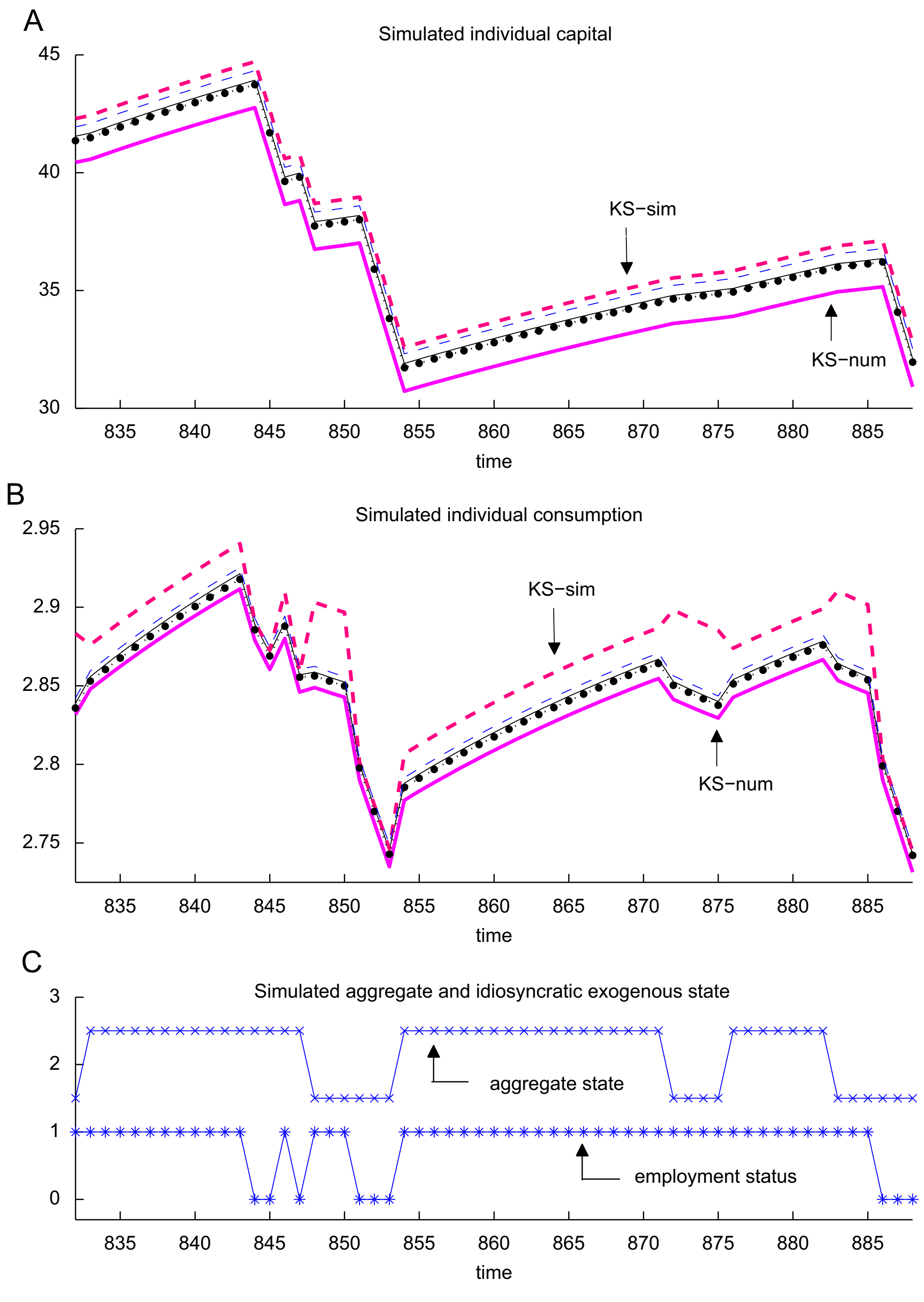

Fig. 5. Simulated individual series I. Notes: This graph plots the indicated series in a part of the sample where substantial (but not the largest) differences between the different algorithms are observed. The scale of the lower panel is meaningless; a higher value for the aggregate state simply indicates that aggregate productivity takes on the higher value and a higher value for the employment status indicates that the agent is employed.

of the cross-sectional average. A sample length of 10,000 observations, which is used here, is apparently not long enough for the law of large numbers to kick in. The explanation is the enormous persistence in the individual capital series. The first-order autocorrelation coefficient varies between 0.997 for Penal and 0.999 for most of the other algorithms. The enormous persistence implies that the sample mean is estimated with a large standard error. In fact, the near unit-root type behavior means that the sample mean is almost not well defined. The high persistence is without doubt important in explaining why the individual capital series can diverge so much from each other, as is documented in Figs. 5 and 6, and why the same, although to a lesser extent is true for cross-sectional means, as is documented in Fig. 4.

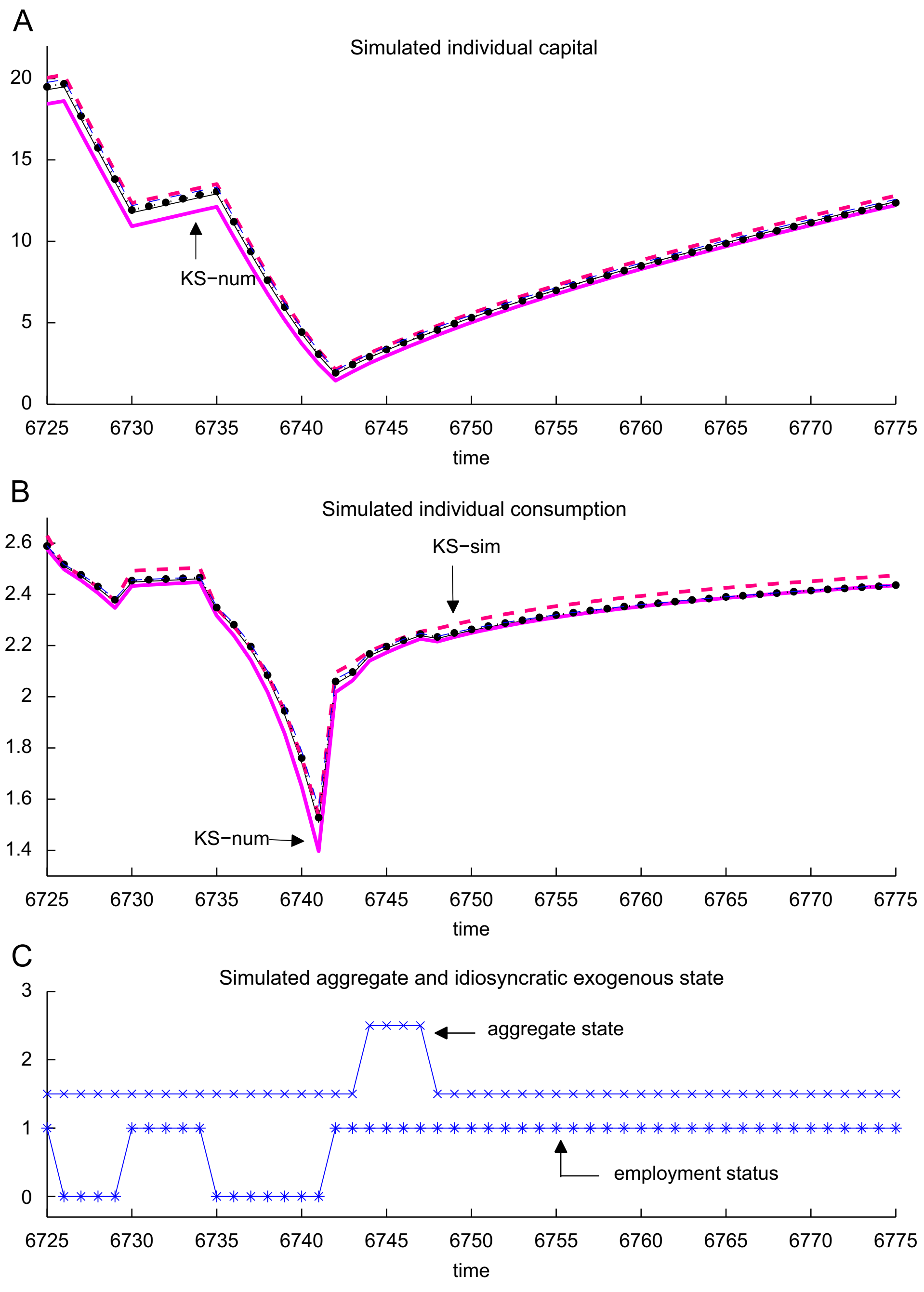

Fig. 6. Simulated individual series II. Notes: This graph plots the indicated series in that part of the sample where the largest differences between the different algorithms are observed. The scale of the lower panel is meaningless; a higher value for the aggregate state simply indicates that aggregate productivity takes on the higher value and a higher value for the employment status indicates that the agent is employed.

The high persistence is a typical feature of many economic models. Driving processes are typically persistent and the desire to smooth consumption only adds to the persistence. In the presence of so much persistence, one should be extremely careful in using regression analysis based on moments from simulated data as part of a numerical solution; the outcomes of regression analysis depend on sample means and when the series are highly persistent, then it typically requires extremely long samples to estimate these precisely.

4.1.4. Accuracy of the individual policy rule

In this section, the accuracy of the individual policy rule is discussed. It is possible to check for accuracy using standard Euler-equation errors. These are difficult to compare across methods, however, because the different algorithms use

Table 14
Percentage errors from dynamic Euler accuracy test.

	BInduc	KS-num	KS-sim	Param	Xpa	Penal
Capital (*scaled error*)						
Average error (%)	0.005	4.61	33.2	0.78	0.050	0.29
Maximum error (%)	0.016	17.2	68.8	2.96	0.103	1.52
Consumption (% *error*)						
Average error (%)	0.006	0.54	5.52	0.11	0.006	0.12
Maximum error (%)	0.004	24.6	67.5	22.3	0.099	1.10

Notes: The error from this test is the difference between the values of the capital and consumption paths generated with the individual policy functions and the values of the paths that are obtained when each period the values of capital and consumption that are implied by the explicitly calculated conditional expectation are used. Since capital values can be close to zero, the error for capital is calculated as the absolute difference divided by the mean capital stock.

different specifications for the state space. Therefore, the dynamic Euler-equation test discussed in Section 3.4 is used. The purpose here is to test only the accuracy of the individual policy rule. Thus, $\tilde{k}_{t+1}$ is based on the same values of the aggregate moments as those used to calculate k_{t+1}, namely those from the simulated panel. In terms of making predictions for the aggregate variables (which are needed to calculate the conditional expectation), the choices made are identical to those the algorithm makes in solving for the individual policy rule.

Table 14 reports the average and maximum percentage differences between the alternative series, generated by explicitly calculating the conditional expectation in each period, and the series directly calculated using the numerical solution for the individual capital choice. If the solution is accurate, then the differences between the two series should be small. The relative performance of the different algorithms is similar to the one found for the case without aggregate uncertainty. For the case with aggregate uncertainty, however, the size of the errors attained by the algorithms that perform less well are larger. The exception is Penal and to some extent Param. The accuracy errors for Penal are enormous for the case without aggregate uncertainty, but in the mid-range of the observed outcomes for the case with aggregate uncertainty. The aggregate capital level for the case without aggregate uncertainty was set at a level above the equilibrium level to make the constraint bind more often. In the version with aggregate uncertainty, the aggregate capital level is the equilibrium level and the constraint binds less often. This is likely to be the reason behind the improved overall performance of the Penal algorithm and possibly also behind the reduction in the maximum consumption error for Param. It is an interesting question why the error for KS-num increases by so much for the case with aggregate uncertainty, while this is not the case for BInduc and Xpa.[25] As mentioned above, the fact that the constraint is less binding can only make it easier to solve the individual model accurately. The higher errors for KS-num suggest that solving the individual policy rules accurately is more difficult in the case with aggregate uncertainty, than in the case without, *even* if the law of motion for the aggregate capital stock is taken as given. But the results for BInduc and Xpa make clear that it is possible to have errors that are low for the version with and for the version without aggregate uncertainty. The larger number of nodes over a more relevant part of the state space is likely to be an important reason behind the ability of BInduc and Xpa to keep on performing well when aggregate uncertainty is introduced.

4.2. *Properties of the law of motion for aggregate variables*

All algorithms used here, except BInduc, solve for a numerical approximation of the law of motion for a set of moments of the joint distribution of capital holdings and employment status.[26] Param solves for an approximation to the aggregate law of motion, but this is an unnecessary step; given the parameterized cross-sectional distribution, the individual policy function could have been solved for without an explicit parameterization of the aggregate law of motion.

In this section, we investigate the properties of the generated aggregate laws of motion.[27] The first subsection discusses the differences in the implied means and standard deviations for aggregate capital. The second subsection discusses the accuracy of the aggregate laws of motion.

[25] The accuracy errors for KS-sim are likely to pick up the deviating behavior of the individual capital choice when the economy switches from one aggregate state to another. Maliar et al. (2009) report that their original program contained a typo. After correcting the typo, the average (maximum) errors are equal to 0.032% (0.093%) and 0.009% (0.436%) for capital and consumption, respectively.

[26] den Haan (1996) also does not solve for a separate law of motion for aggregate capital. This algorithm updates the coefficients of the individual policy rules using the data from the simulated panel, which contains the required information about the endogenous aggregate state variables *without* relying on an explicit parameterization of the aggregate law of motion.

[27] As explained in Section 2.6 of Reiter (2009b), the closest analogue to simulating with an aggregate law of motion is to simulate with the proxy distribution and the data of BInduc are based on this simulation.

Table 15
Moments of K_t in panel and according to aggregate law of motion.

	BInduc	KS-num	KS-sim	Param	Xpa	Penal
Sample means of K_t						
In panel	39.307	39.348	39.357	39.306	39.329	39.187
According to aggregate law of motion	39.338	39.351	39.346	39.375	39.340	39.254
% difference	0.078	0.088	0.027	0.175	0.030	0.171
Standard deviations of K_t						
In panel	1.001	0.989	1.033	1.005	0.997	0.976
According to aggregate law of motion	1.018	0.992	1.024	1.011	1.027	0.941
% difference	1.700	0.385	0.798	0.611	2.953	3.575

Notes: These statistics are based on the outcomes of a simulation of 10,000 observations using identical realizations for the exogenous random shocks.

Table 16
Accuracy aggregate policy rule.

	BInduc (%)	KS-num (%)	KS-sim (%)	Param (%)	Xpa (%)	Penal (%)
Average differences						
Unemployed	–	–	–	0.256	0.117	0.360
Employed	–	–	–	0.168	0.104	0.343
Total	0.105	0.072	0.050	0.175	0.105	0.345
Maximum differences						
Unemployed	–	–	–	0.497	0.439	1.131
Employed	–	–	–	0.341	0.339	1.056
Total	0.280	0.239	0.156	0.347	0.343	1.059

Notes: This table reports the difference between the mean capital stock according to the aggregate law of motion and the value that is obtained if the individual policy rules are used to simulate a cross-sectional distribution. BInduc, KS-num, and KS-sim only calculate an aggregate law of motion for the total capital stock.

4.2.1. *Means and standard deviations of K_t*

Table 15 reports the mean and standard deviation of K_t according to the aggregate law of motion. The table documents that the differences across algorithms are similar to the differences between the corresponding moments that are based on the simulated panel.

If the aggregate law of motion is very accurate, then moments of the capital series generated by the aggregate law of motion should be very close to the corresponding moments of the aggregate capital series from the simulated panel. Those corresponding moments are also reported in Table 15. For the mean, the differences are relatively small. The largest differences are equal to 0.175% and 0.171% for Param and Penal, respectively. The smallest difference is 0.03% which is attained by Xpa. For the standard deviation, the differences are substantially larger. The smallest difference is attained by KS-num and it is equal to 0.385%. Although the smallest value, it is still a non-trivial difference and indicates that not all properties of the aggregate law of motion are accurately measured.[28] Excluding Penal, the largest difference is observed for Xpa. The standard deviation according to the aggregate law of motion is equal to 1.027 and according to the simulated panel it is equal to 0.997. This difference is equal to almost 3%. Note that these moments are based on the exact same set of realizations and the differences are, thus, not due to sampling variation.

The differences between the moments of K_t generated with and without the aggregate law of motion can be interpreted as an accuracy test of the aggregate law of motion, because for the true aggregate law of motion the two time series for aggregate capital would be identical. The next subsection discusses an accuracy test, that is based on this idea.

4.2.2. *Accuracy of the aggregate policy rule*

Papers in the literature that use the KS algorithm typically evaluate the accuracy of the law of motion for endogenous aggregate variables using the R^2 and the standard error of the regression. In den Haan (2009), I document, however, that these measures are completely inadequate accuracy tests. In particular, I consider simulated data for the aggregate capital stock that is generated by the KS-num algorithm. Taking these as the true data, I then consider several approximating laws

[28] The values of the standard R^2 test are in excess of 0.9999 for both KS-num and KS-sim. This supports the view expressed in den Haan (2009) that the R^2 gives a misleading idea about the accuracy of the aggregate law of motion.

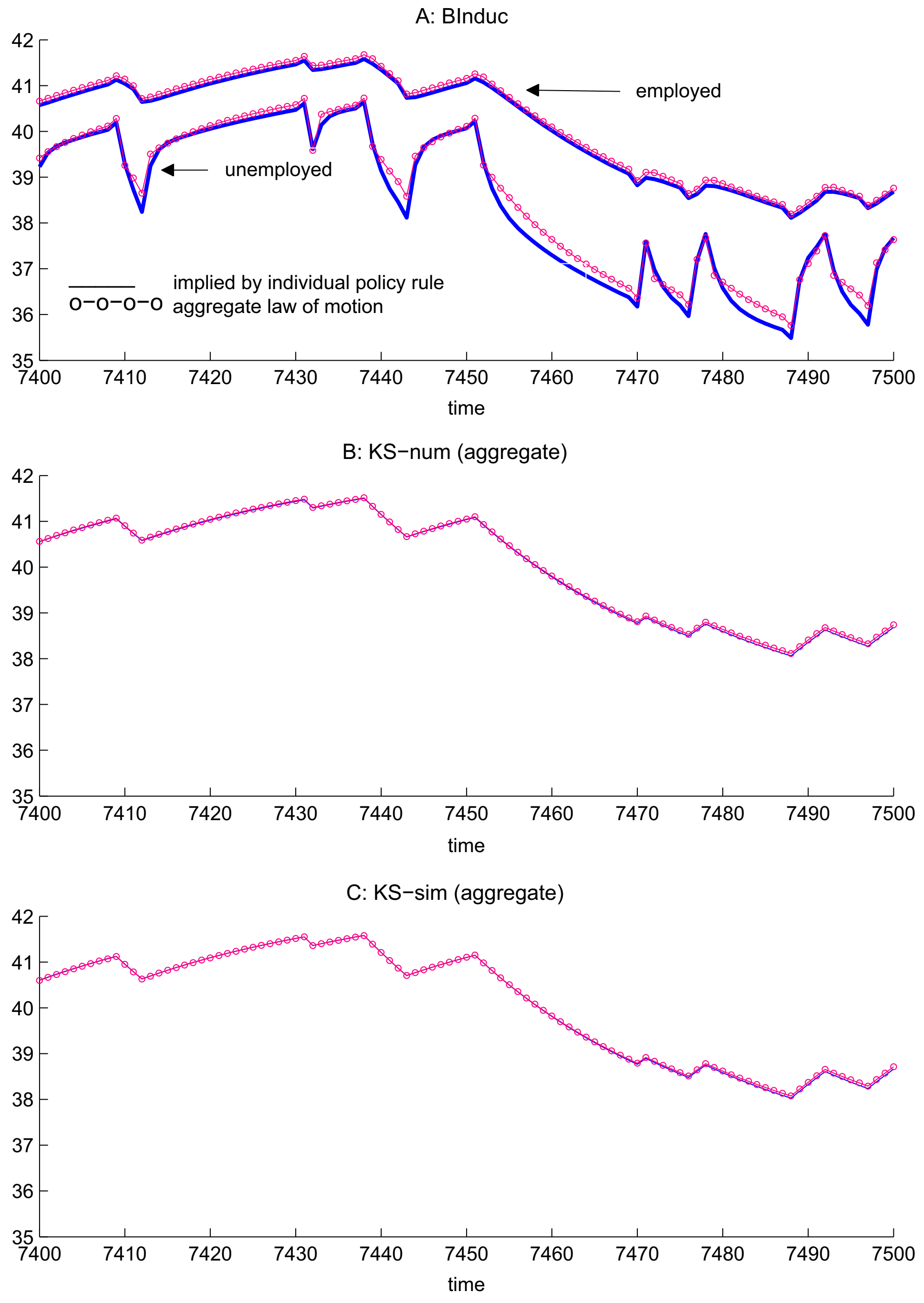

Fig. 7. Accuracy aggregate law of motion. Notes: This graph plots the indicated mean capital stock according to the aggregate law of motion (line with open circles) and the value that is obtained if the individual policy rules are used to simulate a cross-sectional distribution (solid line). Note that the figures for KS-num and KS-sim do plot the two lines, but that they are basically indistinguishable.

of motion. The true standard deviation of aggregate capital is up to 14% (119%) higher than the value implied by approximating laws of motion with R^2's as high as 0.9999 (0.99). Thus, even approximating laws of motion with high R^2 values can be clearly inaccurate in that they generate, for example, quite different standard deviations.[29]

[29] The R^2 has several problems, but one of the most important ones is that—in evaluating the fit of the aggregate law of motion—it uses as the explanatory variable the value of aggregate capital obtained from the simulated panel. That is, each period the aggregate law of motion is corrected using what has to be explained.

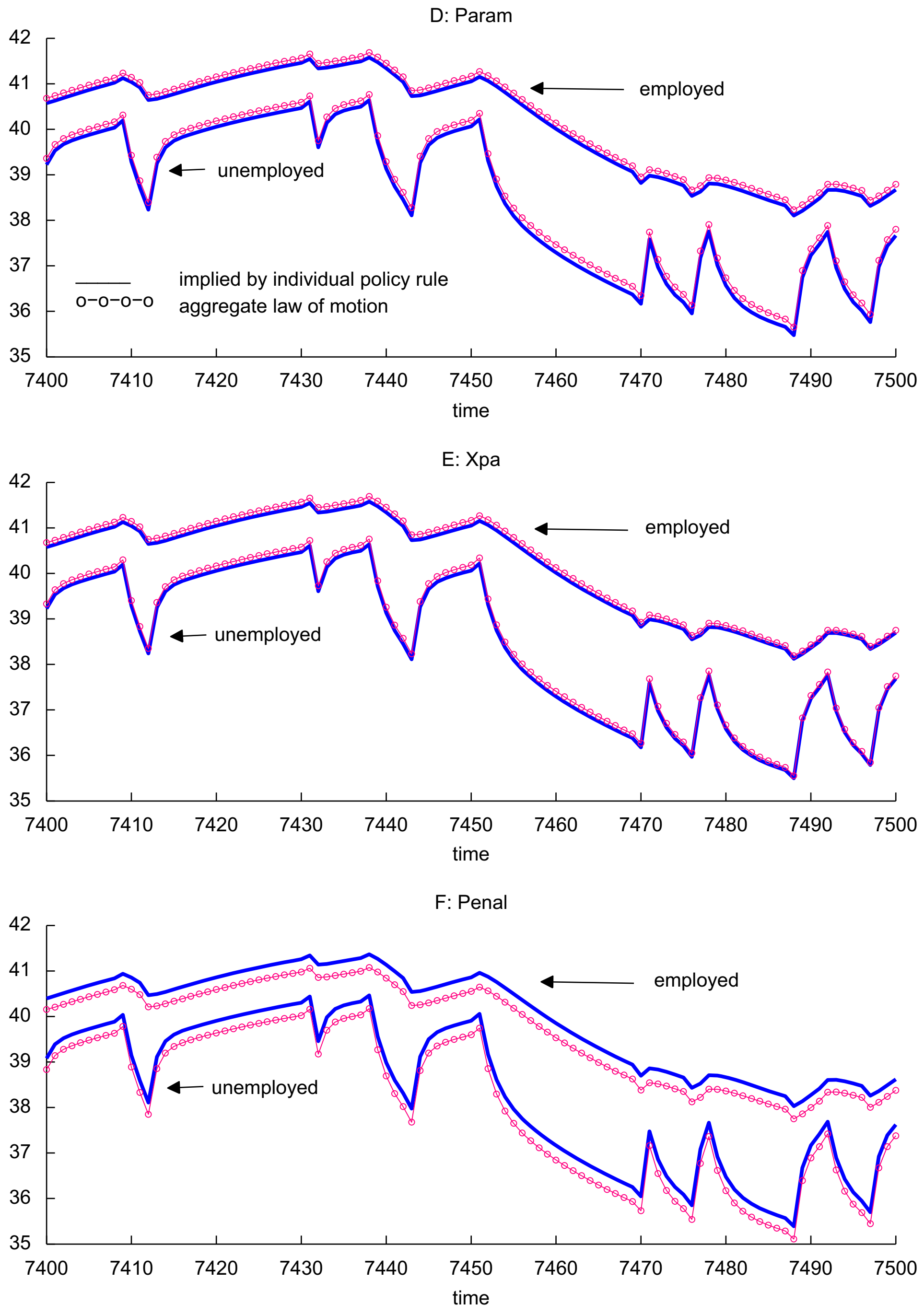

Fig. 7. (*Continued*)

Description of the accuracy test: den Haan (2009) discusses two alternative accuracy tests that are much more powerful. The first accuracy test is the maximum forecast error of 100-period ahead forecast errors observed in a long simulation. This measure was first mentioned in Krusell and Smith (1998), but it is now rarely used. The second accuracy test is the logical extension of the first and is the one considered in this paper. It calculates the maximum percentage error between the following two aggregate series. The first is the aggregate capital stock that comes out of the simulated panel; these observations are calculated using *only* the individual policy rules. The second is the series that is obtained when only the aggregate law of motion is used to generate a time series for the corresponding aggregate series. That is, if one uses a sample with T observations, then one considers all t-period ahead forecast errors for $1 \leq t \leq T$, starting at period 1 and conditional on the realization of aggregate shocks. By considering such large forecast horizons, one allows tiny errors to accumulate.

Outcomes of the accuracy test: Table 16 reports the outcome of this test using a time series of 10,000 observations. For BInduc, KS-num, and KS-sim, I only report the results for the aggregate capital stocks. For the other algorithms, I also report the results for the aggregate capital stocks conditional on the employment status. The reason is that BInduc, KS-num, and KS-sim only generate a law of motion for aggregate capital.[30] Finding an accurate solution for the law of motion of the average capital stock across all agents is obviously easier, than doing the same for the average capital stock of the unemployed, but should be comparable to obtaining the law of motion for the average capital stock of the employed. The best performance is by KS-num and KS-sim. In particular, the maximum error found by KS-sim (KS-num) for the aggregate capital stock is 0.16% (0.24%), whereas the values are 0.28%, 0.34%, 0.35%, and 1.06% for the average capital according to BInduc, Xpa, Param, and Penal, respectively.

Fig. 7 compares the data generated by the aggregate law of motion with the corresponding time series from the simulated panel.[31] The graph clearly documents the excellent fit for KS-num and KS-sim. The errors of Param and Xpa are small, but the aggregate law of motion generates data that are consistently above the simulated series in this part of the sample. The aggregate laws of motion of BInduc do well during a boom, but the aggregate law of motion for the average capital stock of the unemployed clearly does poorly during a downturn. For Penal the aggregate law of motion consistently lies below the one implied by the simulation, which makes sense given that this law of motion is simply the capital choice of a representative agent that does not face idiosyncratic risk and incomplete markets.

5. Concluding comments

The lessons learned from this comparison project have already been summarized in the introduction. In the remainder of this paper, I will discuss some open questions.

The algorithms considered in this paper differ in terms of how they solve the individual problem and in how they solve the aggregate problem. Somewhat surprisingly, there are large differences in the accuracy of the individual policy rules. These (differences in) inaccuracies are without doubt part of the reason why there are such large differences in several properties of the simulated data. But there are also differences in the (accuracy of the) aggregate laws of motion. Therefore, it is unclear how much of the observed differences are due to differences in how the individual problem is solved and how much is due to differences in how the aggregate law of motion is calculated. It would be a worthwhile exercise to investigate this in more detail.

The second open question is how the different algorithms will compare when more complex models are solved. The advantage of the KS algorithm is that it is easy to program and there are now many papers in the literature that use it. The additional programming burden of some of the other algorithms may make it more difficult to use them in more elaborate models. The recently developed Xpa algorithm, however, is even easier to program than the KS algorithm, because it avoids the simulation step. At least these two algorithms should be compared using more interesting models.

There is no algorithm that uniformly does better in all accuracy tests. It would, therefore, be nice if an algorithm could be constructed that does better than the algorithms considered here in each test considered and in particular has substantially better results for the accuracy test of the aggregate law of motion.

Acknowledgments

I would like to thank the participants of the comparison project for comments and stimulating conversations and acknowledge financial support of the NWO.

Appendix A. Simulating without cross-sectional sampling variation

Information used: The beginning-of-period t distribution of capital holdings is fully characterized by the following:

- the fraction of unemployed agents with a zero capital stock, $p_t^{u,0}$,
- the fraction of employed agents with a zero capital stock,[32] $p_t^{e,0}$,

[30] For BInduc the series are generated as described in footnote 27.

[31] The figure plots the series in that part of the sample where BInduc obtains its largest errors (excluding the initial period) for the average capital stocks conditional on employment status. BInduc does not automatically generate a law of motion for the average capital stocks conditional on employment status, but it is possible to do so. The errors for the conditional means are substantially larger than the errors for the per capita capital stock. The proxy distribution in BInduc takes the role of the aggregate law of motion in the other algorithms. The proxy distribution does not take care well of how capital is split between employed and unemployed, but does predict aggregate capital well.

[32] Employed agents never choose a zero capital stock, but some unemployed agents that chose a zero capital stock last period are employed this period.

- the distribution of capital holdings of unemployed agents with positive capital holdings, and
- the distribution of capital holdings of employed agents with positive capital holdings.

Overview: The goal is to calculate the same information at the beginning of the next period. Besides these four pieces of information regarding the cross-sectional distribution one only needs (i) the realizations of the aggregate shock this period and next period and (ii) the individual policy function.

Grid: Construct the following grid and define the beginning-of-period distribution of capital as follows:

- $\kappa_0 = 0$ and $\kappa_j = 0.1j, j = 1, \ldots, 1000$.
- Let $p_t^{\varepsilon,0}$ be the fraction of agents with employment status ε with a zero capital stock at the beginning of period t.
- For $j > 0$, let $p_t^{\varepsilon,j}$ be equal to the mass of agents with a capital stock bigger than κ_{i-1} and less than or equal to κ_i. This mass is assumed to be distributed uniformly between grid points.
- We have

$$\sum_{j=0}^{1000} p_t^{u,j} = 1, \quad \sum_{j=0}^{1000} p_t^{e,j} = 1.$$

Denote this beginning-of-period distribution function by $P_t^{\varepsilon}(k)$.

End-of-period distribution: The first step is to calculate the end-of-period distribution of capital.

For the unemployed calculate the level of capital holdings at which the agent chooses κ_j. If we denote this capital level by $x_t^{u,j}$, then it is defined by[33]

$$k'(x_t^{u,j}, \cdot) = \kappa_j. \tag{1}$$

Now compute the end-of-period distribution function at the grid points as

$$F_t^{u,j} = \int_0^{x_t^{u,j}} dP_t^u(k) = \sum_{j=0}^{\overline{j_u}} p_t^{u,j} + \frac{x_t^{u,j} - \kappa_{\overline{j_u}}}{\kappa_{1+\overline{j_u}} - \kappa_{\overline{j_u}}} p_t^{u,\overline{j_u}+1}, \tag{2}$$

where $\overline{j_u} = \bar{j}(x_t^{u,j})$ is the largest value of j such that $\kappa_j \le x_t^{u,j}$. The second equality follows from the assumption that P_t^u is distributed uniformly between grid points.

A similar procedure is used to calculate the end-of-period distribution for the employed.

$$F_t^{e,j} = \int_0^{x_t^{e,j}} dP_t^e(k) = \sum_{j=0}^{\overline{j_e}} p_t^{e,j} + \frac{x_t^{e,j} - \kappa_{\overline{j_e}}}{\kappa_{1+\overline{j_e}} - \kappa_{\overline{j_e}}} p_t^{e,\overline{j_e}+1},$$

where $\bar{i}_j = \bar{i}(x_t^{e,j})$ is the largest value of j such that $\kappa_j \le x_t^{e,j}$.

Next period's beginning-of-period distribution: Let $g_{\varepsilon_t \varepsilon_{t+1} a_t a_{t+1}}$ stand for the mass of agents with employment status ε^t that have employment status ε_{t+1}, conditional on the values of a_t and a_{t+1}. For each combination of values of a_t and a_{t+1} we have

$$g_{u_t u_{t+1} a_t a_{t+1}} + g_{e_t u_{t+1} a_t a_{t+1}} + g_{u_t e_{t+1} a_t a_{t+1}} + g_{e_t e_{t+1} a_t a_{t+1}} = 1. \tag{3}$$

We then have

$$P_{t+1}^{\varepsilon,j} = \frac{g_{u_t \varepsilon_{t+1}}}{g_{u_t \varepsilon_{t+1}} + g_{e_t \varepsilon_{t+1}}} F_t^{u,j} + \frac{g_{e_t \varepsilon_{t+1}}}{g_{u_t \varepsilon_{t+1}} + g_{e_t \varepsilon_{t+1}}} F_t^{e,j} \tag{4}$$

and

$$p_{t+1}^{\varepsilon,0} = P_{t+1}^{\varepsilon,0}, \tag{5}$$

$$p_{t+1}^{\varepsilon,j} = P_{t+1}^{\varepsilon,j} - P_{t+1}^{\varepsilon,j-1}. \tag{6}$$

To implement this procedure and to ensure that differences in the simulated output are only due to differences in the policy functions used, we have to use the same interval length, which is set equal to 0.1. Since setting the upper bound can be an important part of the program, participants are free to choose their own upper bound.

References

Algan, Y., Allais, O., den Haan, W.J., 2009. Solving the incomplete markets model with aggregate uncertainty using parameterized cross-sectional distributions. Journal of Economic Dynamics & Control, doi:10.1016/j.jedc.2009.03.010.

Algan, Y., Allais, O., den Haan, W.J., Rendahl, P., 2008. Solving and simulating models with heterogeneous agents. Unpublished Manuscript, University of Amsterdam.

Carroll, C.D., 2006. The method of endogenous gridpoints for solving dynamic stochastic optimization problems. Economics Letters 91, 312–320.

den Haan, W.J., 1996. Heterogeneity, aggregate uncertainty and the short-term interest rate. Journal of Business & Economic Statistics 14, 399–411.

[33] This is a non-linear problem (and has to be calculated at many nodes), but it should be a well behaved problem.

den Haan, W.J., 1997. Solving dynamics models with aggregate shocks and heterogeneous agents. Macroeconomic Dynamics 1, 355–386.
den Haan, W.J., 2009. Assessing the accuracy of the aggregate law of motion in models with heterogeneous agents. Journal of Economic Dynamics & Control, doi:10.1016/j.jedc.2008.12.009.
den Haan, W.J., de Wind, J., 2008. Solving DSGE models when penalty functions are used instead of inequality constraints. Unpublished Manuscript, University of Amsterdam.
den Haan, W.J., Judd, K.L., Juillard, M., 2009. Computational suite of models with heterogeneous agents: model specifications. Journal of Economic Dynamics and Control, doi:10.1016/j.jedc.2009.07.001.
den Haan, W.J., Marcet, A., 1994. Accuracy in simulations. Review of Economic Studies 61, 3–18.
den Haan, W.J., Rendahl, P., 2009. Solving the incomplete markets model with aggregate uncertainty using explicit aggregation. Journal of Economic Dynamics & Control, doi:10.1016/j.jedc.2008.12.008.
Judd, K.L., 1998. Numerical Methods in Economics. MIT Press, Cambridge, MA.
Kim, S., Kollmann, R., Kim, J., 2009. Solving the incomplete markets model with aggregate uncertainty using a perturbation method. Journal of Economic Dynamics & Control, doi:10.1016/j.jedc.2008.11.011.
Krusell, P., Smith Jr., A.A., 1997. Income and wealth heterogeneity, portfolio choice, and equilibrium asset returns. Macroeconomic Dynamics 1, 387–422.
Krusell, P., Smith Jr., A.A., 1998. Income and wealth heterogeneity in the macroeconomy. Journal of Political Economy 106, 867–896.
Maliar, L., Maliar, S., Valli, F., 2009. Solving the incomplete markets model with aggregate uncertainty using the Krusell–Smith algorithm. Journal of Economic Dynamics & Control, doi:10.1016/j.jedc.2009.03.009.
Preston, B., Roca, M., 2006. Incomplete markets, heterogeneity and macroeconomic dynamics. Unpublished Manuscript, Columbia University.
Reiter, M., 2001. Estimating the accuracy of numerical solutions to dynamic optimization problems. Unpublished Manuscript, Universitat Pompeu Fabra.
Reiter, M., 2009a. Solving heterogeneous-agent models by projection and perturbation. Journal of Economic Dynamics & Control, 33, 649–665.
Reiter, M., 2009b. Solving the incomplete markets economy with aggregate uncertainty by backward induction. Journal of Economic Dynamics & Control, doi:10.1016/j.jedc.2008.11.009.
Ríos-Rull, J.V., 1997. Computation of equilibria in heterogeneous agent models. Federal Reserve Bank of Minneapolis Staff Report, vol. 231, pp. 238–264.
Santos, M., 2000. Accuracy of numerical solutions using the Euler equations residual. Econometrica 68, 1377–1402.
Santos, M.S., Peralta-Alva, A., 2005. Accuracy of simulations for stochastic dynamic models. Econometrica 73, 1939–1976.
Young, E.R., 2009. Solving the incomplete markets model with aggregate uncertainty using the Krussell-Smith algorithm and non-stochastic simulations. Journal of Economic Dynamics & Control, doi:10.1016/j.jedc.2008.11.010.