Solving Dynamic Discrete Choice Models Using Smoothing and Sieve Methods

Dennis Kristensen† Patrick K. Mogensen‡ Jong Myun Moon§ Bertel Schjerning¶

March 4, 2019

Abstract

We propose to combine smoothing, simulations and sieve approximations to solve for the value function in a general class of dynamic discrete choice (DDC) models. We use Monte Carlo methods to approximate the Bellman operator defining the solution. The random Bellman operator is not smooth which complicates the search for the corresponding solution which is generally non-differentiable. To circumvent this issue, we introduce a smoothed version of the random Bellman operator and solve for the corresponding smoothed value function using projection-based methods where the unknown solution is approximated by a set of basis functions. We provide an asymptotic theory for the approximate solution and show that it converges with $\sqrt{N}$-rate, where $N$ is number of Monte Carlo draws, towards a Gaussian process. We examine its performance in practice through a set of numerical experiments and find that the new method is computationally fast and reliable and provides a good approximation to the unknown solution.

Keywords: Dynamic discrete choice; numerical solution; Monte Carlo; sieves.

---

*We acknowledge helpful comments from John Rust, Víctor Aguirregabiria, Lars Nesheim, Aureo de Paula and many other people. Kristensen gratefully acknowledges financial support from the ERC (through starting grant No 312474 and advanced grant no. GEM 740369). Schjerning gratefully acknowledges the financial support from the Independent Research Fund Denmark (grant no. DFF – 4182-00052) and the URBAN research project financed by the Innovation Fund Denmark (IFD).

†Department of Economics, University College London, Gower Street, London, United Kingdom. E-mail: d.kristensen@ucl.ac.uk. Website: https://sites.google.com/site/econkristensen.

‡Department of Economics, University of Copenhagen, Øster Farimagsgade 5, Building 35, DK-1353 Copenhagen K, Denmark. E-mail: Patrick.Kofod.Mogensen@econ.ku.dk. Webpage: http://www.economics.ku.dk/staff/phd_kopi/?pure=en/persons/374766.

§PIMCO

¶Department of Economics, University of Copenhagen, Øster Farimagsgade 5, Building 35, DK-1353 Copenhagen K, Denmark. E-mail: Bertel.Schjerning@econ.ku.dk. Webpage: http://bschjerning.com/.
1 Introduction

Discrete Decision Processes (DDPs) are widely used in economics to model forward-looking discrete decisions. For their implementation, researchers are required to solve the model which generally cannot be done in closed form. Instead, a number of methods have been proposed for solving the model numerically; see, e.g., Rust (2008) for an overview. We propose two novel methods for approximating the solutions to a general class of Markovian DDP models in terms of either the so-called integrated or expected value function. Our framework allows for both continuous and discrete state variables, non-separable utility functions and unrestricted dynamics. As such, we cover most relevant models used in empirical work. The proposed implementation of model and estimators are found to be computationally very efficient, and at the same time providing precise results with small approximation errors due to the use of simulations and sieve methods.

Our first proposal proceeds in three steps: First, we develop smoothed simulated versions of the Bellman operators that returns the integrated and expected value functions as fixed points. Next, we approximate the unknown value function by a sieve, that is, a parametric function class, thereby turning the problem into a finite-dimensional one. Finally, we solve for the parameters entering the chosen sieve using projection-based methods. When the chosen sieve is linear in the parameters, the solution can be found iteratively where each step is on closed form.

As an alternative to the above sieve-based method, we also adapt and generalize the approximation method proposed in Rust (1997b) to our setting: We design the sampling procedure used for the simulated Bellman operators in such a way that the corresponding expected and integrated value functions can be solved for directly without the use of sieves. In comparison with the sieve approach, this “self-approximating” procedure has the advantage that it will not suffer from any biases due to function approximations. But at the same time, the simulated Bellman operator used in its implementation will generally have a larger variance compared to the one that can be used for the sieve-method. This larger variance also translates into a larger simulation bias of the self-approximating value function due to the non-linear nature of the problem.

Our two procedures, the sieve-based and self-approximating one, differ from existing proposals in three important aspects: First, we solve for either the integrated or expected value function instead of the value function itself. This generally reduces the dimensionality of the problem since we integrate out any i.i.d. shocks appearing in the model before solving it. Second, we allow for a general class of importance samplers to be used in the implementation of the simulated Bellman operator that can be used to reduce variances due to simulations. Third, we smooth the simulated Bellman operator by replacing the max-function appearing in the Bellman operator by a smoothed version where the degree of smoothing is controlled by a parameter akin to the bandwidth in kernel smoothing methods. This is similar to the logit-smoothed accept-reject simulator of probit models as proposed by McFadden (1989); see also Fermanian and Salanie (2004), Kristensen and Shin (2012) and Iskhakov, Jørgensen, Rust and Schjerning (2017). The smoothing turns the problem of solving for the value function into a differentiable one. In particular, derivative-based solvers can be used which helps reducing computation time significantly. Moreover, the approximate value function also become smooth in any underlying structural pa-
rameters which means that, if used in estimation, standard numerical solvers can be employed in computing estimators of these. The smoothing entails an additional bias but this can be controlled for by the aforementioned smoothing parameter. The smoothing device also facilitates the theoretical analysis of the approximate value functions since it allows us to use functional Taylor expansions. This expansion is then used to analyze the leading numerical error terms of the approximate value functions due to simulations, smoothing and function approximations. In particular, under regularity conditions, we show that the approximate value function will converge weakly towards a Gaussian process which is the first result of its kind to our knowledge. These results allow researchers to, for example, build confidence intervals around the estimated value function and should be useful when analyzing the impact of value function approximation when used in estimation. The results may also be potentially helpful in building algorithms for selecting the relevant number of basis functions and the smoothing parameter.

A numerical study investigates the performance of the sieve-based and self-approximating methods in practice. We implement the proposed methods for the engine replacement model of Rust (1987) and investigate how number of basis functions and simulations affect the approximation error. We also investigate how the procedures are affected when the number of state variables increases, and how derivative-based solvers affect computation times. We find that the sieve method performs best of the two methods: It is generally computationally faster and provides a better approximation in terms of bias and variance. Moreover, it appears to be little affected by an increase in the number of state variables while the self-approximating method suffers from a severe curse-of-dimensionality in this particular setting. Moreover, the errors due to simulations and function approximation behave according to theory and are found to vanish at the expected rates. The proposed method share similarities with the ones developed in, amongst others, Arcidiacono, Bayer, Bugni and (2013), Keane and Wolpin (1994), Munos and Szepesvari (2008), Rust (1997b) and Pal and Stachurski (2013) who also use simulations and/or sieve methods to solve dynamic discrete choice models. However, their methods solve for the value function while we focus on the integrated or expected value function which tend to be more well-behaved (smooth) and therefore easier to approximate. Moreover, in contrast to the cited papers, we employ importance sampling and smoothing in our implementation which comes with the aforementioned computational advantages. From a theory perspective, we provide a more complete asymptotic analysis of the approximate integrated and expected value functions. On the other hand, Munos and Szepesvari (2008) and Rust (1997b) provide an analysis of the computational complexity of solving for the value function and so the theories of this paper and these studies complement each other.

The remains of the paper are organized as follows: Section 2 introduces a general class of DDP’s and their “smoothed” versions. In Section 3, we develop our smoothed simulated versions of the Bellman operators that the integrated and expected value functions are fixed points to. We then show how to (approximately) solve these simulated Bellman equations in Section 4. An asymptotic theory of the approximate value function is presented in Section 5, while the results of the numerical experiments are found in Section 6. All proofs and lemmas have been relegated to Appendix B and C, respectively.
2 Model

We consider the following dynamic discrete choice model where the agent at time \( t \geq 1 \) chooses

\[
d_t = \arg \max_{d \in D} \{ u(S_t, d) + \beta E [ \nu(S_{t+1}) | S_t, d_t = d] \}, \tag{2.1}
\]

where \( D = \{1, \ldots, D\} \) is the set of alternatives, \( u(S_t, d) \) is the per-period utility, \( 0 < \beta < 1 \) is the discount factor, \( S_t \) is a set of state variables that follows a controlled Markov process with transition kernel \( F_s (S_t | S_{t-1}, d_{t-1}) \) and the so-called value function \( \nu \) solves the following fixed-point problem,

\[
\nu(S_t) = \max_{d \in D} \{ u(S_t, d) + \beta E [ \nu(S_{t+1}) | S_t, d_t = d] \}. \tag{2.2}
\]

In many of the specifications of the above model found in empirical work, \( S_t \) contains an i.i.d. component, \( S_t = (Z_t, \varepsilon_t) \in Z \times \mathcal{E} \subseteq \mathbb{R}^d \times \mathbb{R}^d \) where \( Z_t \) and \( \varepsilon_t \) satisfy the following conditional independence condition,

\[
F_s (Z_t, \varepsilon_t | Z_{t-1}, \varepsilon_{t-1}, d_{t-1}) = F_{\varepsilon} (\varepsilon_t | Z_t) F_Z (Z_t | Z_{t-1}, d_{t-1}).
\]

If no i.i.d. component is present, we can always choose \( \varepsilon_t = \emptyset \) to be an empty variable so that \( S_t = Z_t \). Throughout, we will assume that \( Z \) is a compact set. This is done to simplify the theoretical analysis since it, for example, implies that all relevant functions will lie in the space \( B(Z) \) of bounded functions on \( Z \) equipped with the sup-norm, \( \|v\|_{\infty} = \sup_{z \in Z} |v(z)| \).

In the current formulation, the model is characterized in terms of \( \nu(s) \). However, it is possible to rewrite the models in terms of either the so-called integrated value function or the expected value function and solve for these instead. These are defined as

\[
v(Z_t) = E [\nu(Z_t, \varepsilon_t) | Z_t] = \int_{\varepsilon} \nu(Z_t, \varepsilon) dF_{\varepsilon} (d\varepsilon | Z_t),
\]

and

\[
V(Z_t, d_t) = E [\nu(Z_{t+1}, \varepsilon_{t+1}) | Z_t, \varepsilon_t, d_t] = E [v(Z_{t+1}) | Z_t, d_t] = \int_{\varepsilon} v(z') dF_Z (dz' | Z_t, d_t),
\]

respectively, where we have used the conditional independence assumption. Eq. (2.1) can now be written as

\[
d_t = \arg \max_{d \in D} \{ u(Z_t, \varepsilon_t, d) + \beta V(Z_t, d) \}. \tag{2.3}
\]

Similarly, conditional choice probabilities, which are needed for counterfactuals and for estimation, take the form

\[
P(d_t = d | Z_t = z) = M_{u,d}(\beta V(z') | z), \quad M_{u,d}(r | z) = \frac{\partial M_{u}(r | z)}{\partial r(d)},
\]

which again take as input the expected value function. Thus, for any implementation of the above class of dynamic discrete choice models, such as the computation of estimators and/or counterfactual analysis, we need to be able to compute either \( V \) directly, or first \( v \) and then

\[
V(Z_t, d_t) = E [v(Z_{t+1}) | Z_t, d_t].
\]

Except for a few special cases, an analytical expression of \( V \) is not available and so numerical approximations have to be employed.
We will here develop numerical methods for solving for \( v \) or \( V \) instead of \( \nu \) for the following reasons: First, \( \nu \) is a function of \( s = (z, \bar{z}) \) while \( V \) and \( v \) are functions of \( z \) alone and therefore tend to be easier to approximate. Second, \( \nu \) is non-differentiable due to the max-function in (2.2); in contrast, \( v(z) \) and \( V(z, d) \) are both smooth functions of \( z \) if \( F_{\varepsilon}(\cdot|z) \) and \( F_{z}(\cdot'|z, d) \) are. If there is no i.i.d. component in the model, \( \varepsilon_t = 0 \), then \( \nu(s) = \nu(z) = v(z) \), while \( V(z, d) \) remains smooth even in this case. We allow for both countable and continuously distributed random variables. However, many of the ideas and results only have real bite in the continuous case. For example, the smoothness property is irrelevant in the discrete case.

The integrated and expected value function each solves their own fixed-point problem: Taking conditional expectations on both sides of eq. (2.2), we see that \( V \) is the solution to the following fixed point problem,

\[
V(z, d) = \Gamma(V)(z, d),
\]

where

\[
\Gamma(V)(z, d) = E \left[ \max_{d' \in D} \left\{ u(Z_{t+1}, \varepsilon_{t+1}, d') + \beta V(Z_{t+1}, d') \right\} \mid Z_t = z, d_t = d \right]
= \int_{\mathcal{Z}} \int_{\mathcal{E}} \max_{d' \in D} \left\{ u(z', \epsilon, d') + \beta V(z', d') \right\} dF_{\varepsilon}(de|z')dF_{z}(dz'|z, d)
= \int_{\mathcal{Z}} M_u(\beta V(z')|z')dF_{z}(dz'|z, d),
\]

and \( M_u(r|z) \) is a generalized version of the so-called social surplus function defined as, for any \( r = (r(1), \ldots, r(D)) \),

\[
M_u(r|z) = \int_{\mathcal{E}} \max_{d \in D} \left\{ u(z, \epsilon, d) + r(d) \right\} dF_{\varepsilon}(de|z).
\]

In the following, we will let \( V(z) \) denote the \( D \times 1 \)-vector function \( V(z) = (V(z, 1), \ldots, V(z, D))' \) and similar for other object. With this notation, we can represent the fixed-point problem on vector form, \( V(z) = \Gamma(V)(z) \), where

\[
\Gamma(V)(z) = \int_{\mathcal{Z}} M_u(\beta V(z')|z')dF_{z}(dz'|z).
\]

Next, to derive the fixed-point problem that \( v \) solves, again take conditional expectations on both sides of eq. (2.2) but now only condition on \( Z_t \) to obtain

\[
v(z) = M_u(\beta V(z)|z).\tag{2.5}
\]

Combining this with eq. (2.3),

\[
v(z) = M_u \left( \beta \int_{\mathcal{Z}} M_u(\beta V(z')|z')dF_{z}(dz'|z) \bigg| z \right) = \bar{\Gamma}(v)(z). \tag{2.6}
\]

where

\[
\bar{\Gamma}(v)(z) = M_u \left( \beta \int_{\mathcal{Z}} v(z') dF_{z}(dz'|z) \bigg| z \right).
\]

Under great generality, \( \Gamma \) and \( \bar{\Gamma} \) are contraction mappings on \( \mathcal{B}(\mathcal{Z}) \) and so \( V \in \mathcal{B}(\mathcal{Z})^D \) and \( v \in \mathcal{B}(\mathcal{Z}) \) are well-defined and unique. However, closed-form expressions of any of these components
are generally not available.

**Example 1.** Consider the special case where \( u(Z_{t+1}, \varepsilon_{t+1}, d) = \tilde{u}(Z_{t+1}, d) + \varepsilon_{t+1} (d) \) and \( F_{\varepsilon}(e|z) = F_{\varepsilon}(e) \) in which case

\[
M_o(r|z) = \int_{\mathcal{D}} \max_{\varepsilon \in \mathcal{D}} \{ \tilde{u}(z, d) + e + r(d) \} \, dF_{\varepsilon}(de) = G_{\varepsilon}(\tilde{u}(z) + r),
\]

where \( G_{\varepsilon}(r) = \int_{\mathcal{D}} \max_{\varepsilon \in \mathcal{D}} \{ e + r(d) \} \, dF_{\varepsilon}(de). \) Thus,

\[
\tilde{\Gamma}(v)(z) = G_{\varepsilon} \left( \tilde{u}(z) + \beta \int_{\mathcal{Z}} v(z') \, dF_{\varepsilon}(dz'|z) \right).
\]

### 3 Simulated Bellman operators

As a first step towards a computationally feasible method for solving for either the integrated or expected value function, we introduce simulated versions of their two Bellman operators, \( \Gamma \) and \( \tilde{\Gamma} \), defined in the previous section. We first develop basic Monte Carlo simulators of the operators and then introduce their smoothed counterparts.

To allow for added flexibility in the implementation and to cover as special case a modified version of Rust’s method, we employ importance sampling for the sampling of \( S_{t+1}|S_t \): Let \( \Phi_z(z'|z, d) \) and \( \Phi_{\varepsilon}(e|z) \) be conditional importance sampling distribution functions as chosen by the researcher. These have to be chosen such that \( F_{z} (\cdot|z, d) \) and \( F_{\varepsilon} (\cdot|z) \) are absolutely continuous w.r.t. \( \Phi_z (\cdot|z, d) \) and \( \Phi_{\varepsilon} (\cdot|z) \), respectively, with Radon-Nikodym derivatives \( w_z (\cdot|z, d) \geq 0 \) and \( w_{\varepsilon} (\cdot|z) \geq 0 \) so that

\[
\frac{dF_{z}(z'|z, d)}{d\Phi_{z}(z'|z, d)} = w_z (z'|z, d) \, d\mu_z (z'), \quad \frac{dF_{\varepsilon}(e|z)}{d\Phi_{\varepsilon}(e|z)} = w_{\varepsilon} (e|z) \, d\mu_{\varepsilon} (e), \tag{3.1}
\]

for some measures \( \mu_z \) and \( \mu_{\varepsilon} \). The leading case is where \( dF_{z} = f_z d\mu_z \) and \( d\Phi_{z} = \phi_z d\mu_z \) in which case \( w_z = f_z/\phi_z \) and similar for the sampling of \( \varepsilon_t \). The above covers the case where \( z_{t+1}|z_t \) has a continuous distribution (in which case \( \mu_z \) is the Lesbesque measure), a discrete distribution (in which case \( \mu_z \) is the counting measure) and the mixed case. With discrete finite support, we could in principle solve the exact Bellman equation and so would not need to resort to numerical methods. But if the discrete support is large this may still be computationally very demanding and so even in this case the numerical methods developed below may be computationally attractive, c.f. Arcidiacono, Bayer, Bugni and James (2013).

Given the chosen importance sampler, we can rewrite \( \Gamma(V)(z, d) \) as

\[
\Gamma(V)(z, d) = \int_{\mathcal{Z}} \int_{\mathcal{E}} \max_{d' \in \mathcal{D}} \left\{ u(s', d') + \beta V(z', d') \right\} \, w(s'|z, d) \, d\Phi(ds'|z, d),
\]

where \( s' = (z', e') \) and

\[
w(s'|z, d) = w_z (e'|z') \, w_z (s'|z, d), \quad \Phi (s'|z, d) = \Phi_{\varepsilon}(e'|z') \Phi_{z} (z'|z, d).
\]

For a given choice of \( V \), we can then approximate this integral by Monte Carlo methods: First
generate $N \geq 1$ i.i.d. draws, $Z_i (z, d) \sim \Phi_z (\cdot | z, d)$ and $\varepsilon_i (z, d) \sim \Phi_\varepsilon (\cdot | Z_i (z, d)), i = 1, ..., N,$ and then compute

$$
\Gamma_N (V) (z, d) = \sum_{i=1}^{N} \max_{d' \in D} \{ u (S_i (z, d), d') + \beta V (Z_i (z, d), d') \} w_{N,i} (z, d),
$$

where $S_i (z, d) = (Z_i (z, d), \varepsilon_i (z, d))$ and

$$
w_{N,i} (z, d) = \frac{w (S_i (z, d) \mid z, d)}{\sum_{i=1}^{N} w (S_i (z, d) \mid z, d)}.
$$

Note here that we normalize the importance weights so that $\sum_{i=1}^{N} w_{N,i} (z, d) = 1$. This is done to ensure that $\Gamma_N$ is a contraction mapping on $\mathbb{B} (\mathcal{Z})^D$. Similarly, we approximate $\bar{\Gamma} (v)$ by

$$
\bar{\Gamma}_N (v) (z) = \sum_{j=1}^{N} \max_{d' \in D} \left\{ u (z, \varepsilon_j (z, d'), d') + \beta \sum_{i=1}^{N} v (Z_i (z, d')) w_{z,N,i} (z, d') \right\} w_{z,N,j} (z, d'),
$$

where again we normalize the weights to ensure $\bar{\Gamma}_N$ is a contraction on $\mathbb{B} (\mathcal{Z})$,

$$
w_{z,N,i} (z, d) = \frac{w_z (Z_i (z, d) \mid z, d)}{\sum_{i=1}^{N} w_z (Z_i (z, d) \mid z, d)}, \quad w_{z,N,i} (z, d) = \frac{w_{\varepsilon,i} (\varepsilon_i (z, d) \mid z)}{\sum_{i=1}^{N} w_{\varepsilon,i} (\varepsilon_i (z) \mid z)}.
$$

When $\varepsilon_t = \emptyset$, the simulated Bellman operator $\bar{\Gamma}_N$ includes as special cases the ones considered in Rust (1997b) (who chooses $\Phi_z$ as the uniform distribution uniform on $\mathcal{Z}$) and Pal and Stachurski (2013) (who chooses $\Phi_z = F_z$).

**Example.** Considering again the special case where $u (z_{t+1}, \varepsilon_{t+1}, d) = \bar{u} (z_{t+1}, d) + \varepsilon_{t+1} (d)$ and $F_\varepsilon (e \mid z) = \Phi_\varepsilon (e \mid z) = F_\varepsilon (e)$, the above simplifies to

$$
\bar{\Gamma}_N (v) (z) = \frac{1}{N} \sum_{j=1}^{N} \max_{d' \in D} \left\{ \bar{u} (z, d') + \varepsilon_j (d') + \beta \sum_{i=1}^{N} v (Z_i (z, d')) w_{z,N,i} (z, d') \right\}.
$$

If $F_\varepsilon (e \mid z)$ and $F_z (z' \mid z, d)$ are smooth functions w.r.t. $z$ then $\Gamma (V) (z)$ and $\bar{\Gamma} (v) (z)$ will be smooth functions of $z$ as well. In contrast, $\Gamma_N (V) (z, d)$ and $\bar{\Gamma}_N (v) (z)$ are non-smooth due to the presence of the max-function in their definitions which does not get smoothed for finite $N$. This in turn implies that their corresponding fixed points, $V_N = \Gamma_N (V_N)$ and $v_N = \bar{\Gamma}_N (v_N)$, will be non-differentiable which complicates solving for them (approximately).

We propose to smooth the simulated Bellman operators above by replacing the “hard” max-function appearing in the expressions of $\Gamma_N$ and $\bar{\Gamma}_N$ by the following smoothed version,

$$
G_\lambda (r) = \lambda \log \left[ \sum_{d \in D} \exp \left( \frac{r (d)}{\lambda} \right) \right], \quad (3.2)
$$

where $r = (r (1), ..., r (D))$ and $\lambda > 0$ is a smoothing parameter that plays a similar role as the bandwidth in kernel regression estimation. Elementary calculations yield that

$$
0 \leq G_\lambda (r) - \max_{d \in D} r (d) \leq \lambda \log D, \quad (3.3)
$$

where
so that \( G_\lambda (r) \to \max_{d \in D} r(d) \), as \( \lambda \to 0 \), uniformly in \( r \in \mathbb{R}^D \). Substituting in \( G_\lambda (r) \) yields the following smoothed simulated operators,

\[
\Gamma_{N,\lambda}(V)(z, d) = \sum_{i=1}^{N} G_\lambda \left( u \left( S_i(z, d) \right) + \beta V \left( Z_i(z, d) \right) \right) w_{N, i}(z, d), \tag{3.4}
\]

\[
\tilde{\Gamma}_{N,\lambda}(v)(z) = \sum_{j=1}^{N} G_\lambda \left( u(z, \varepsilon_j(z)) + \beta \sum_{i=1}^{N} v \left( Z_i(z) \right) \circ w_{N, j}(z) \right) w_{\varepsilon, N, j}(z), \tag{3.5}
\]

where \( u(z, \varepsilon_j(z)) = (u(z, \varepsilon_j(z, 1), 1), ..., u(z, \varepsilon_j(z, D), D)) \) and \( \circ \) denotes the so-called Hadamard product. These will be employed below to obtain smooth approximations to the expected and integrated value function, respectively. The use of \( G_\lambda (r) \) in place of \( \max_{d \in D} r(d) \) generates an additional bias in the approximate solutions of order \( O_\mu(\lambda) \) as we shall see in the theory section, but this can be controlled for by suitable choice of \( \lambda \).

In an important class of models \( G_\lambda(r) \) appears in the Bellman operators as a direct implication of the model specification in which case no smoothing bias will be present: Suppose that in addition to the i.i.d. shock \( \varepsilon_t \) there is another shock variable \( \eta_t = (\eta_t(1), ..., \eta_t(D)) \) present in the model which is independent of \( \{S_t, S_{t-1}, ...\} \) and enters the per-period utility additively so that the model of interest takes the form

\[
d_t = \arg \max_{d \in D} \{ u(S_t, d) + \lambda \eta_t(d) + \beta E[\nu(S_{t+1}, \eta_{t+1})|S_t, \eta_t, d_t = d] \}, \]

where \( \lambda > 0 \) is now a structural scale parameter that determines the impact of \( \eta_t(d) \) on the utility. Choosing the components of \( (\eta_t(1), ..., \eta_t(D)) \) to be i.i.d. extreme-value distributed random variables, we obtain after suitable normalization of the distribution that

\[
\int_{\mathbb{R}^D} \max_{d \in D} \{ r(d) + \lambda \varepsilon(d) \} dF_{\varepsilon}(d\varepsilon) = \log \left( \sum_{d \in D} \exp \left( \frac{r(d)}{\lambda} \right) \right) = G_\lambda (r), \tag{3.6}
\]

c.f. Rust, Traub and Wozniakowski (2002). So another interpretation of \( G_\lambda(r) \) is that of the social surplus function of a discrete choice problem with an additive extreme-value term. It is now easily checked, combining the arguments employed in the previous section with eq. (3.6), that the expected value function in this case solves \( \Gamma_\lambda (V) = V \) where

\[
\Gamma_\lambda (V)(z) = \int_{\mathcal{E}} \int_{\mathcal{Z}} G_\lambda \left( u(z', e') \right) + \beta V(z') \right) dF_{\varepsilon}(de'|z)dF_{Z}(dz'|z, d),
\]

and so \( \Gamma_{N,\lambda} \) in eq. (3.4) is an unbiased simulated version of \( \Gamma_\lambda \). Thus, if the original model of interest contains an additive extreme value term, which is the case in many empirical papers, \( G_\lambda \) appears as part of the model and so no smoothing bias will be present in our proposed simulator.
4 Approximate value functions

The simulated Bellman operators \( \Gamma_{N,\lambda}(V) \) and \( \Gamma_{N,\lambda}(v) \) in eqs. (3.4)-(3.5) are contraction mappings on \( \mathbb{B}(Z)^D \) and \( \mathbb{B}(Z) \), respectively, and so they have unique fixed points defined as

\[
V_{N,\lambda} = \Gamma_{N,\lambda}(V_{N,\lambda}), \quad v_{N,\lambda} = \Gamma_{N,\lambda}(v_{N,\lambda}). \tag{4.1}
\]

However, solving these two simulated Bellman equations are not generally feasible since these are infinite-dimensional problems. We here present two ways to reduce the problems to a finite-dimensional ones. The first method is a generalization of the so-called self-approximating method proposed in Rust (1997b) while the second one uses projection-based methods as advocated by Pal and Stachurski, 2013.

4.1 Self-approximating method

Rust (1997b) proposed to turn the infinite-dimensional problems in eq. (4.1) into a finite-dimensional ones by choosing the importance sampling to be based on marginal, instead of conditional distributions. In our generalised version this corresponds to restricting \( \Phi_z(z'|z,d) = \Phi_z(z') \) for some marginal distribution \( \Phi_z(\cdot) \) so that the draws \( Z_i \sim \Phi_z(\cdot) \) and \( \epsilon_i \sim \Phi_z(\cdot|Z_i) \), \( i = 1,...,N \) no longer depend on \( z \). In this case, the fixed-point problems in eq. (4.1) reduce to solving one of the following two sets of \( N \) nonlinear equations,

\[
V_{N,\lambda,k} = \sum_{i=1}^{N} G_{\lambda}(u(S_i) + \beta V_{N,\lambda,i}) w_{N,i}(Z_k), \tag{4.2}
\]

\[
v_{N,\lambda,k} = \sum_{j=1}^{N} G_{\lambda} \left( u(Z_k, \epsilon_j) + \beta \sum_{i=1}^{N} v_{N,\lambda,i} w_{z,N,i}(Z_k) \right) w_{\epsilon,N,j}(Z_k), \tag{4.3}
\]

for \( k = 1,...,N \), w.r.t. \( \{V_{N,\lambda,k} : k = 1,...,N\} \) and \( \{v_{N,\lambda,k} : k = 1,...,N\} \), respectively. Here, \( V_{N,\lambda,k} = V_{N,\lambda}(Z_k) \) and \( v_{N,\lambda,k} = v_{N,\lambda}(Z_k), k = 1,...,N \). Each of the two sets of equations have a unique solution due to the contracting property of \( \Gamma_{N,\lambda} \) and \( \Gamma_{N,\lambda} \). Once, for example, eq. (4.2) has been solved, the approximate expected value functions can be evaluated at any other value \( z \) by

\[
V_{N,\lambda}(z) = \sum_{i=1}^{N} G_{\lambda}(u(S_i) + \beta V_{N,\lambda,i}) w_{N,i}(z).
\]

Note that \( V_{N,\lambda}(z) \) is a smooth function even if \( \lambda = 0 \) as long as \( w_{N,i}(z) \) is smooth and smoothing here becomes unnecessary for this property to hold. However, without smoothing, the set of equations (4.2) become non-smooth w.r.t. the variables \( \{V_{N,\lambda,k} : k = 1,...,N\} \) and so cannot be solved using derivative-based methods. Thus, the numerical implementation of the self-approximating method still benefits from smoothing.

In addition to smoothing, the above self-approximating method differs from Rust’s original proposal in two other ways: First, while Rust (1997b) solved for the value function \( \nu(z,\epsilon) \), we here solve for either \( V(z) \) or \( v(z) \). As explained earlier, the latter are the more relevant ones in most applications. Moreover, our formulation allows for the following generalized version of the
simulated Bellman equations for $v_{N,\lambda}$,

$$v_{N,\lambda,k} = \sum_{j=1}^{\tilde{N}} G_\lambda \left( u(Z_k, \varepsilon_j) + \beta \sum_{i=1}^{N} v_{N,\lambda,i} w_{z,N,i}(Z_k) \right) w_{e,N,j}(Z_k),$$

(4.4)

where we allow for different number of draws from $\Phi_z(\tilde{N})$ and $\Phi_z(N)$. In particular, we can choose $\tilde{N}$ as large as we wish (thereby decreasing the variance of the problem) without increasing the number of variables that need to be solved for ($N$). A similar generalization of the simulated Bellman equations for $V_{N,\lambda}$ is possible. Second, we here only require that the state dynamics together with the chosen importance sampler satisfy (3.1); in contrast, Rust (1997b) assumed that $S_t$ was continuously distributed with compact support and chose as importance sampler the uniform distribution with same support. Thus, our version allows for a broader class of models and samplers.

The self-approximating method may not always work well: First, finding a marginal distribution $\Phi_z(\cdot)$ so that (3.1) holds can be difficult in some models. And even if (3.1) does hold, the use of marginal samplers instead of conditional ones will generally lead to a larger variance of the solutions since the “marginal” draws $Z_1, \ldots, Z_N$ do not adapt to the changing shape of $F_z(\cdot|z,d)$ as a function of $z$. In particular, many of the draws may fall outside of the support of $F_z(\cdot|z,d)$ and so are “wasted” in which case a large $N$ is required to achieve a reasonable approximation; see Section 6 for an example of this. This issue tends to become more severe in higher-dimensions (when $d_z$ is large), and so the self-approximating method will generally suffer from a built-in curse-of-dimensionality. This curse-of-dimensionality does not appear in the subclass of models that Rust (1997b) focused on as described above. Finally, given that $V_{N,\lambda}$ and $v_{N,\lambda}$ are solutions to non-linear equations, a large variance in the simulated Bellman operator translates into a large bias as is well-known from non-linear GMM estimators. But choosing $N$ to be large means that numerically solving either (4.2) or (4.3) becomes computationally very costly. These issues motivate us to pursue a projection-based solution strategy.

### 4.2 Projection-based method

We now return to the general versions of the simulated Bellman operators and so again allow for conditional importance samplers. Let $\mathcal{V} \subseteq \mathcal{B}(\mathcal{Z})$ be a (scalar) function space that $v_{N,\lambda}$ defined in (4.1) lies in. We then choose a finite-dimensional function space (commonly called a sieve in the econometrics literature) $\mathcal{V}_K = \{v_K(\cdot;\alpha) : \mathcal{Z} \mapsto \mathbb{R} | \alpha \in \mathcal{A}_K \} \subseteq \mathcal{V}$, where $\mathcal{A}_K \subseteq \mathbb{R}^K$ is a parameter set with $K < \infty$, and will then search for an approximate solution to $v_{N,\lambda}$ within this. Similarly, we let $\mathcal{V} \subseteq \mathcal{B}(\mathcal{Z})^D$ be a space of $D$-dimensional vector functions that the solution to the simulated expected Bellman equation in (4.1) lie in and $\mathcal{V}_K = \{V_K(\cdot;\alpha) : \mathcal{Z} \mapsto \mathbb{R}^D | \alpha \in \mathcal{A}_K \} \subseteq \mathcal{V}$ our sieve for this space. Let

$$\Pi_K(v) = \arg \min_{v' \in \mathcal{V}_K} \|v - v'|_{\mathcal{V}}$$

$$\Pi_K(V) = \arg \min_{V' \in \mathcal{V}_K} \|V - V'|_{\mathcal{V}}.$$  

(4.5)
be the corresponding projections for given (pseudo-) norms \( \| \cdot \|_{\mathcal{P}} \) and \( \| \cdot \|_{\mathcal{V}} \) as chosen by us as well. We then approximate \( V_{N,\lambda} \) and \( v_{N,\lambda} \) by the solutions to the projected Bellman equations,

\[
\hat{v}_{N,\lambda} = \arg \min_{v \in \mathcal{V}_K} \left\{ \| v - \Pi_K \Gamma_{N,\lambda}(v) \|_{\mathcal{V}} \right\}, \quad \hat{V}_{N,\lambda} = \arg \min_{V \in \mathcal{V}_K} \left\{ \| V - \Pi_K \Gamma_{N,\lambda}(V) \|_{\mathcal{V}} \right\}.
\]

These are finite-dimensional problems of size \( K \). When \( K \) is small relative to \( N \), which will generally be the case, the above problems are computationally much more tractable compared to the corresponding self-approximating ones. Note here that these projection-based approximations are different from the least-squares approximations that would solve \( \min_{V \in \mathcal{V}_K} \| V - \Gamma_{N,\lambda}(V) \|_{\mathcal{V}} \) and \( \min_{v \in \mathcal{V}_K} \| v - \Gamma_{N,\lambda}(v) \|_{\mathcal{V}} \), respectively. In particular, by suitable choice of the projection operators, \( \Pi_K \Gamma_{N,\lambda} \) and \( \Pi_K \Gamma_{N,\lambda} \) will be contraction mappings w.r.t. \( \| \cdot \|_{\infty} \) guaranteeing that \( \hat{V}_{N,\lambda} \) and \( \hat{v}_{N,\lambda} \) exist and are unique. The following discussion focuses on the expected value function approximation since it carries over with only minor modifications to the one of the integrated value function. We discuss their numerical implementation in further detail in the subsection below.

The projection operator \( \Pi_K \) can be thought of as a function approximator with the approximation error being \( V - \Pi_K (V) \) for a given function \( V \). Roughly speaking, the projection-based method approximates \( V_{N,\lambda} \) by \( \hat{V}_{N,\lambda} = \Pi_K (V_{N,\lambda}) \) which incurs an additional sieve approximation error, \( V_{N,\lambda} - \Pi_K (V_{N,\lambda}) \). The smoothness of \( V_{N,\lambda} \) here proves helpful since many well-known sieves are able to provide good approximations of smooth functions using a low-dimensional space so that \( K \) is small. Due to these features, our proposed projection-based solutions will generally suffer from quite small additional biases relative to the exact simulated solution. This is in contrast to existing projection-based solution methods, such as the one in Pal and Stachurski (2013), that approximate the value function \( \nu (s) \) which is non-differentiable.

The smoothness properties of \( V_{N,\lambda} \) help guiding us in choosing the sieve: It allows us to restrict \( \mathcal{V} \) to a suitable smoothness class and then import existing approximation methods for smooth functions as developed in the literature on numerical methods and nonparametric econometrics. A leading example is the class of linear function approximations where the finite-dimensional function space takes the form of \( V_K = \{ \alpha' B_K (z) : \alpha \in \mathbb{R}^K \} \) for a set of basis functions \( B_K (z) = \{ b_k (z) : k = 1, \ldots, K \} \). The basis functions can be chosen as, for example, Chebyshev interpolation and B-splines that are able to approximate smooth functions well. However, other non-linear function space are possible such as wavelets, artificial neural networks and shrinkage-type function approximators such as LASSO, where the additional constraints are imposed on \( \mathcal{A}_K \); we refer to Chen, 2007 for a general overview of different function approximators and constrained sieve estimators. We also allow for flexibility in terms of the chosen norms \( \| \cdot \|_{\mathcal{V}} \) with a leading example being \( \| V \|_{\mathcal{V}} = \sum_{i=1}^M V (z_i)' V (z_i) \) for a set of design points \( z_1, \ldots, z_M \in \mathcal{Z} \). Very often the design points will be chosen in conjunction with the sieve.

The above procedure does not suffer from any of the above mentioned issues of the self-approximating method: We can choose the importance samplers freely and so these can be designed to control the variance of the simulated Bellman operators; and the dimension of the problem remains \( K \) irrespectively of the number of draws \( N \). The main drawback is that unique solutions to eqs. (4.6) do not necessarily exist for a given choice of \( N \) and \( K \).
A sufficient condition for this to hold is that $\Pi_K$ is a non-expansive operator w.r.t $\|\cdot\|_\infty$, $\|\Pi_K (V_1) - \Pi_K (V_2)\|_\infty \leq \|V_1 - V_2\|_\infty$ since this translates into $\Pi_K \Gamma_{N,\lambda}$ being a contraction mapping. By definition, $\Pi_K$ is non-expansive w.r.t. $\|\cdot\|_V$ but this does not necessarily translate into non-expansiveness w.r.t $\|\cdot\|_\infty$. Pal and Stachurski (2013) provide some examples of projections that are non-expansive w.r.t. $\|\cdot\|_\infty$, but these are unfortunately computationally more expensive to implement in general. However, $\Pi_K$ will generally be non-expansive w.r.t $\|\cdot\|_\infty$ asymptotically as $K \to \infty$ for a wide range of sieves and pseudo-norms since

$$\|\Pi_K\|_{op,\infty} := \sup_{V \in \mathcal{V}, \|V\|=1} \|\Pi_K (V)\|_\infty \leq \sup_{V \in \mathcal{V}, \|V\|=1} \|\Pi_K (V) - V\|_\infty + 1,$$

where the first term in the last expression will go to zero in great generality as $K \to \infty$ for suitably choices of $\mathcal{V}$ and $\mathcal{V}_K$ (see next section for details). Given that $\Gamma_{N,\lambda}$ is a contraction with Lipschitz coefficient $\beta < 1$, this in turn implies that $\Pi_K \Gamma_{N,\lambda}$ will be a contraction mapping for all $K$ large enough. This will be used in our asymptotic analysis of the algorithm. The above argument is not completely satisfactory in practice, however, since it does not say how large $K$ should be chosen to ensure non-expansiveness. But in our numerical experiments we generally did not experience any such issues.

4.3 Numerical implementation of the two methods

We here discuss in more detail the numerical implementation of the self-approximating and projection-based methods. We here focus on solving for the integrated value function since most results and arguments for this case carries over with very minor modifications to the expected value function. First, the researcher has to choose the importance sampling distributions, the smoothing parameter $\lambda$ and, in the case of the projection-based method, the function approximation $\Pi_K$. Second, given these choices, either eq. (4.3) or (4.6) has to be solved for.

4.3.1 Importance sampler

The choices of $\Phi_\epsilon$ and $\Phi_\Delta$ determine the variance of the $\tilde{\Gamma}_{N,\lambda}$ and should ideally be tailored to minimize it. In the case of projection-based methods, where we can choose $\Phi_\Delta$ and $\Phi_\epsilon$ as conditional distributions, we can rely on the already existing theory for efficient importance sampling for how to do so; see Chapter 3 in Robert and Casella, 2013 for an introduction. In our numerical experiments, we did not experiment with different choices and throughout set $\Phi_\epsilon = F_\epsilon$ and $\Phi_\Delta = F_\Delta$.

In the case of the self-approximating method, the choice of $\Phi_\Delta$ is restricted to the class of marginal distributions. Generally, this entails a large variance of the corresponding simulated Bellman operators. It will hold in great generality that $(z_t, d_t)$ has a stationary distribution, say, $F_\Delta (z, d)$. In this case, a suitable choice would be the marginal, $\Phi_\Delta (z) = \sum_{d \in \mathcal{D}} F_\Delta (z, d)$. However, the stationary distribution depends on the value function and so is rarely available on closed form; so this strategy requires an initial solution and exploration of the model. Alternatively, one can try to construct a good approximation of the stationary approximation through a mixture Markov model on the form $F_\Delta (z'|z) = \sum_{d \in \mathcal{D}} \omega_d (z) F_\Delta (z'|z, d)$ for a set of pre-specified mixture
weights $\omega_d(z) \geq 0$ with $\sum_{d \in D} \omega_d(z) = 1$. Simulating from this model does not require solving the model and one could now choose $\Phi_z(z)$ as the stationary solution to this auxiliary model (assuming it exists). In the numerical experiments, we chose $\Phi_z(z)$ as the uniform distribution on $Z$ which is far from optimal, and so more research in this direction is needed.

### 4.3.2 Smoothing

The use of $G_\lambda(r)$ in place of $\max_{d \in D} r(d)$ generally generates an additional bias in the corresponding integrated value function of order $O(\lambda)$. At the same time, the variance of $v_{N,\lambda}$ is an increasing function of $\lambda$. Thus, ideally we would like to choose $\lambda$ to balance these two effects. A natural criterion would be to minimize the so-called integrated mean-square-error, $\lambda^* = \arg \min_{\lambda \geq 0} E \left[ \int_Z \| v_{N,\lambda}(z) - v(z) \|^2 dF(z) \right]$, where $F(z)$ is a suitably chosen distribution such as the stationary one of $Z_t$. Since $v(z)$ is unknown and we cannot evaluate the expectations, $\lambda^*$ cannot be solved for but cross-validation methods can be used instead. This could in principle be done along the same lines as bandwidth selection for smoothed empirical cdfs, see Bowman, Hall and Prvan (1998). However, this is computationally somewhat burdensome and so in practice we recommend choosing $\lambda = 0.01$ since this choice works well in simulations.

### 4.3.3 Function approximation

As mentioned earlier, many approximation architectures are available in the literature. In our numerical experiments we focus on the class of linear function approximators where $\mathcal{V}_K = \{ \alpha' B_K(z) : \alpha \in \mathbb{R}^{K \times D} \}$ for a set of pre-specified basis functions $B_K(z) \in \mathbb{R}^K$. For a given set of $M \geq 1$ design points in $Z, z_1, ..., z_M$, eq. (4.5) then becomes

$$
\Pi_K(v)(z) = B_K(z)' \left[ \sum_{i=1}^M B_K(z_i) B_K(z_i)' \right]^{-1} \sum_{i=1}^M B_K(z_i) v(z_i). \quad (4.7)
$$

The design points may either be random or deterministic and can be chosen relative to the basis functions to ensure that $\Pi_K$ is easy to compute and provides a good approximation for a broad class of functions. The performance of most function approximations will depend on the smoothness of the function of interest. Due to smoothing, $v_{N,\lambda}$ will be $s \geq 1$ times differentiable if the elements of the model of interest are $s$ times differentiable. Let $\mathcal{C}_s(Z)$ denote the space of $s \geq 0$ times continuously differentiable functions on the domain $Z$; we may then choose $\mathcal{V}$ as

$$
\mathcal{V}_{s,r} = \{ v \in \mathcal{C}_s(Z) : \| v \|_{s,\infty} < r \}, \quad (4.8)
$$

for some $r < \infty$ where, with $\alpha = (\alpha_1, ..., \alpha_d) \in \mathbb{N}_0^d$,

$$
\| v \|_{s,\infty} = \sup_{|\alpha| \leq s} \| D^\alpha v \|_{\infty}, \quad D^\alpha v(z) = \frac{\partial^{\alpha_1 + \cdots + \alpha_d} v(z)}{\partial^{\alpha_1} \cdots \partial^{\alpha_d}}. \quad (4.9)
$$

We can now employ existing results for approximating functions in $\mathcal{V}_{s,r}$ to control the error due to projection:
Example 2. Polynomial interpolation using tensor products. Suppose we use Jth order Chebyshev interpolation with M ≥ J nodes in each of the d_z dimensions, or a Jth order B-spline interpolation with M ≥ J number of nodes in each of the d_z dimensions. Let p_1, ..., p_J denote the J polynomials; we then have

\[ B_K (z) = \left\{ p_{j_1} (z_1) \cdots p_{j_{d_z}} (z_{d_z}) : j_1, ..., j_{d_z} = 1, ..., J \right\}, \]

which is of dimension K = J^{d_z}. Assuming that J ≥ s, where s ≥ 1 denotes the number of derivatives of v(z) both interpolation schemes satisfy, for any radius r < ∞,

\[ \sup_{v \in \mathcal{V}_{s,r}} \| \Pi_K (v) - v \|_\infty = O \left( \frac{\log (J)}{J^{s+1}} \right) = O \left( \frac{\log (K)}{K^{(s+1)/d}} \right); \]


As can be seen from the above example, standard polynomial tensor product approximations suffer from the well-known curse of dimensionality: For a given error tolerance, the total number of basis functions K has to grow exponentially as d_z increases. This issue can be partially resolved by using more advanced function approximation methods:

Example 3. Interpolation with sparse grids. Instead of using tensor-product basis functions to approximate a given function, where the total number of basis function and interpolation points will have to grow exponentially with d_z to control the approximation error, one can instead use so-called Smolyak sparse grids; see, e.g., Judd, Maliar, Maliar and Valero (2014) and Brumm and Scheidegger (2017). Using these, the number of grid points needed to obtain a given error tolerance are reduced from O \left( M^{d_z} \right) to O \left( M (\log M)^{d_z} \right) with only slightly deteriorated accuracy.

Example 4. Variable selection, shape constraints, shrinkage estimators, and machine learning. An alternative way of breaking the curse of dimensionality is to select the basis functions judiciously. This could, for example, be done using standard variable selection methods; one example of this approach can be found in Chen (1999). Alternatively, one can in some cases show that the value functions satisfy certain shape constraints that can then be imposed on , for example, Cai and Judd, 2013. Other automated selection methods include shrinkage methods where a penalization term is added to the least-squares criterion. Again this leads to a more sparse representation which is able to break the curse-of-dimensionality. Finally, machine learning algorithms, such as neural networks, can be employed to obtain good approximations of the value function that suffer from only a moderate curse-of-dimensionality; see, for example, Chen and White, 1999.

As noted earlier, there is no guarantee that a given function approximator is non-expansive. But this can, in principle, be checked for a given choice. For the least-squares projection, this amounts to solving, for a given choice of basis functions and grid points,

\[ \left\| \Pi_K \right\|_{op, \infty} = \sup_{v \in \mathbb{R}^M, \|v\| = 1} \sup_{z \in Z} \left| B_K (z)^\prime \left[ \sum_{i=1}^{M} B_K (z_i) B_K (z_i)^\prime \right]^{-1} \sum_{i=1}^{M} B_K (z_i) v_i \right|. \]
When $M$ and/or $\dim Z$ is large this may be computationally demanding and instead one can obtain a lower bound by restricting $z$ to only take values on the chosen set of grid points: With $B_{K,M} \in \mathbb{R}^{K \times M}$ containing the basis functions evaluated at the grid points, we can represent $\Pi_K$ when only evaluated at chosen grid points $z_1, \ldots, z_M$ in terms of

$$P_{K,M} = B_{K,M}' \left[ B_{K,M} B_{K,M}' \right]^{-1} B_{K,M} \in \mathbb{R}^{M \times M}.$$  

In particular, it is easily checked that with the supremum in (4.10) being only taken over $z \in \{z_1, \ldots, z_M\}$, $\|\Pi_K\|_{\text{op,}\infty} = \|P_{K,M}\|_{\text{op,}\infty}$. Furthermore, $\|P_{K,M}\|_{\text{op,}\infty} \leq 1$ if and only if

$$\max_{i=1,\ldots,M} \sum_{j=1}^{M} |p_{ij}| \leq 1,$$

where $p_{ij}$ is the $(i,j)$th element of $P$, c.f. Lizotte (2011).

### 4.3.4 Solving for the approximate value functions

Computing the simulated self-approximating solution or the projection-based one can be done using three different numerical algorithms: Successive approximation, Newton-Kantorovich, or a combination of the two. The latter corresponds to the hybrid solution method proposed in Rust, 1988. We here discuss the implementation of these algorithms for the projection-based method approximation of the $v$; their implementations for the projection-based approximation of $V$ and the self-approximating solutions of either of the two follow along the same lines. The main difference between solving for $V$ or $v$ is that the latter involves smaller computational burden since it is a scalar function while the former is a $D$-dimensional vector function.

Successive approximation utilizes that (for $K$ chosen large enough), $\Pi_K \Gamma_{N,\lambda}$, is a contraction mapping which guarantees that the following algorithm will converge towards the solution to (4.2),

$$v_{N,\lambda}^{(k)} = \Pi_K \Gamma_{N,\lambda}(v_{N,\lambda}^{(k-1)}),$$

for $k = 1, 2, \ldots$, given some initial guess $\hat{v}_{N,\lambda}^{(0)}$. In the leading case of (4.7), this can be expressed as a sequence of least-squares problems that are easily computed: $v_{N,\lambda}^{(k)}(z) = \hat{\alpha}_k B_K(z)$ where

$$\hat{\alpha}_k = \left[ \sum_{i=1}^{M} B_K(z_i) B_K(z_i)' \right]^{-1} \sum_{i=1}^{M} B_K(z_i) \Gamma_{N,\lambda}(\hat{\alpha}_{k-1} B_K)(z_i)' \in \mathbb{R}^{K},$$

for $k = 1, 2, \ldots$, given some initial guess $\hat{\alpha}_0$. Here, observe that

$$\Gamma_{N,\lambda}(\alpha' B_K)(z_i) = \sum_{k=1}^{N} G_{\lambda} \left( u(z, \epsilon_k(z_i)) + \beta \alpha' \sum_{j=1}^{N} B_K(Z_j(z_i)) \circ w_{z,N,j}(z_i) \right) w_{z,N,k}(z_i),$$

and so $\sum_{j=1}^{N} B_K(Z_j(z_i)) \circ w_{z,N,j}(z_i), i = 1, \ldots, M$, only need to be computed once and then recycled in each iteration; in contrast, the simulated averages appearing in $\Gamma_N(\hat{\alpha}_{k-1} B_K)(z_i)$, $i = 1, \ldots, M$, have to be recomputed in each step of the successive approximation algorithm.
This implies that it is generally faster to (approximate) solve for \( v_{N,\lambda} \) instead of \( V_{N,\lambda} \). While the successive approximation method is guaranteed to converge if \( \Pi_K \) is non-expansive, rate of convergence will be slow with the error vanishing at rate \( \beta^k \),

\[
\left\| v_{N,\lambda}^{(k)} - v_{N,\lambda} \right\|_\infty \leq \frac{\beta^k (1 + \beta)}{1 - \beta} \left\| v_{N,\lambda}^{(0)} - v_{N,\lambda} \right\|_\infty.
\]

To speed up computations, we therefore follow Rust (1988) and propose to combine successive approximations with Newton-Kantorovich (NK) iterations to speed up convergence. While successive approximation only converges with rate \( \beta^k \), NK iteration converges with polynomial rate once a given guess of the value function is close enough to the fixed point. Moreover, in situations where \( \Pi_K \) is expansive, NK iteration is still guaranteed to converge locally. Thus, a hybrid method where successive approximation and NK iterations are combined can achieve convergence even in situations where non-expansiveness of \( \Pi_K \) fails. Since both the self-approximating and projection-based methods solve finite-dimensional problems, the NK algorithm for these are equivalent to the standard Newton-Raphson method. First consider the projection-based method where we focus on the least-squares projection as given in (4.7). We are then seeking to find \( \hat{\alpha} \) solving the following \( K \) equations,

\[
S_{N,K} \left( \alpha \right) = 0,
\]

with

\[
S_{N,K} \left( \alpha \right) = \alpha - \left[ \sum_{i=1}^{M} B_K (z_i) B_K (z_i)' \right]^{-1} \sum_{i=1}^{M} B_K (z_i) \bar{\Gamma}_{N,\lambda} \left( \alpha' B_K \right) (z_i).
\]

The corresponding derivatives of the left-hand side as a function w.r.t. \( \alpha \) can be expressed in terms of the the Hadamard differential of \( \bar{\Gamma}_{N,\lambda} \) w.r.t. \( v \),

\[
\nabla \bar{\Gamma}_{N,\lambda} (v) [m] (z) = \beta \sum_{d \in D} \sum_{j=1}^{N} \partial_i G_{d,\lambda} \left( u (z_i, \varepsilon_j (z)) + \beta \sum_{i=1}^{N} v (Z_i (z)) \circ w_{z,N,i} (z) \right)
\times \left( \sum_{k=1}^{N} dm (Z_k (z,d)) \circ w_{z,N,k} (z,d) \right) w_{\varepsilon,N,j} (z),
\]

where \( m : Z \mapsto \mathbb{R} \) is the direction and

\[
\partial_i G_{d,\lambda} (r) = \frac{\partial \bar{G}_\lambda (r)}{\partial d} (d) = \frac{\exp \left( \frac{r(d)}{\lambda} \right)}{\sum_{d' \in D} \exp \left( \frac{r(d')}{\lambda} \right)}.
\]

The partial derivatives of \( S_{N,K} \left( \alpha \right) \) then becomes

\[
H_{N,K} \left( \alpha \right) = I_K - \left[ \sum_{i=1}^{M} B_K (z_i) B_K (z_i)' \right]^{-1} \sum_{i=1}^{M} B_K (z_i) \nabla \bar{\Gamma}_{N,\lambda} \left( \alpha' B_K \right) [B_K] (z_i)' \in \mathbb{R}^{K \times K}.
\]

With these definitions, the NK algorithm takes the form

\[
\hat{\alpha}_k = \hat{\alpha}_{k-1} - H_{N,K}^{-1} \left( \hat{\alpha}_{k-1} \right) S_{N,K} \left( \hat{\alpha}_{k-1} \right).
\]
The NK algorithm for the self-approximating method is on the same form, except that we now solve directly for the value function at the \( N \) draws. With slight abuse of notation, let \( v_N = \{ v_{N,\lambda}(Z_k) : k = 1, \ldots, N \} \) be the vector of integrated values across the set of draws solving \( S_N(v_N) = 0 \) where

\[
S_{N,k}(v_N) = v_{N,k} - \sum_{j=1}^{N} G_{\lambda}(u(Z_k, \varepsilon_j) + \beta \sum_{i=1}^{N} v_{N,i} \circ w_{z,N,i}(Z_k)) w_{\varepsilon,N,j}(Z_k),
\]

for \( k = 1, \ldots, N \). The corresponding derivatives is \( H_N(\alpha) = (H_{N,1}(\alpha), \ldots, H_{N,N}(\alpha))' \in \mathbb{R}^{N \times N} \) where, with \( \mathbf{1}_N = (1, \ldots, 1)' \in \mathbb{R}^N \),

\[
H_{N,k}(\alpha) = I_N - \nabla \Gamma_{N,\lambda}(v_N)[\mathbf{1}_N](z_k) \in \mathbb{R}^N.
\]

Finally, we note that the NK algorithm for the expected value function also takes a similar form with the functional differential of \( \Gamma_{N,\lambda} \) being given by

\[
\nabla \Gamma_{N,\lambda}(v)[M](z) = \beta \sum_{d \in \mathcal{D}} \sum_{i=1}^{N} \hat{G}_{d,\lambda}(u(S_i(z,d)) + \beta V(Z_i(z,d))) M(Z_i(z,d)) w_{N,i}(z,d).
\]

Comparing the NK algorithm for the self-approximating and the projection-based method, we note that the former involves inverting a \( N \times N \)-matrix while the latter a \( K \times K \)-matrix. As pointed out earlier, the self-approximating method generally needs \( N \) to be chosen quite large to achieve a precise simulated version of the Bellman operator, in particular in higher dimensions, and so the NK algorithm for this method is not feasible beyond 2-3 dimensions. While the projection-based method also suffers from a curse of dimensionality, since the number of basis functions, \( K \), has to be quite large in higher dimensions to achieve a reasonable approximation, it is less severe and is implementable for higher-dimensional models. If more advanced function approximation methods are employed, even better performance can be achieved.

\section{Theory}

We here develop an asymptotic theory for the self-approximating and projection-based methods. We first establish some fundamental results for the simulated Bellman operators and their exact solutions. These are then used to provide results for the approximate solutions. The theory will be based on two general results for estimated solutions to fixed point problems stated in Theorems 6 and 7 in the appendix. Our analysis will mostly focus on \( V_{N,\lambda} \) and \( \hat{V}_{N,\lambda} \) since these easily translate into similar results for, for example, \( v_{N,\lambda}(z) = M_{N,\lambda,\alpha}(\beta V_{N,\lambda}(z)|z) \), where

\[
M_{N,\lambda,\alpha}(r|z) = \sum_{j=1}^{N} G_{\lambda}(u(z, \varepsilon_j(z)) + r) w_{\varepsilon,N,j}(z),
\]
Theorem 1. Under Assumption 1, for all $\lambda \geq 0$ and all $N \geq 1$, the operators $(\Gamma_\lambda, \Gamma_{N, \lambda})$ and $(\tilde{\Gamma}_\lambda, \Gamma_{N, \lambda})$ are almost surely contraction mappings on $B(Z)^D$ and $B(Z)$, respectively. In
particular, the solutions $V_{N,\lambda}$ and $v_{N,\lambda}$ to the simulated Bellman equations (4.1) exist and are unique. If in addition Assumption 2 holds, then $V_\lambda(z)$, $V_{N,\lambda}(z)$, $v_\lambda(z)$ and $v_{N,\lambda}(z)$ are $s \geq 0$ times continuously differentiable w.r.t. $z$ for any $\lambda > 0$.

Next, we show that the bias due to smoothing vanishes with rate $\lambda$:

**Theorem 2.** Under Assumption 1, the following hold: $\|V_\lambda - V_0\|_\infty = O(\lambda)$ and $\|V_{N,\lambda} - V_N\|_\infty = O_P(\lambda)$ for any given $N \geq 1$.

These two results show that the smoothing can be controlled for by suitable choice of $\lambda$ both asymptotically ($N = +\infty$) and for a given set of simulations ($N < \infty$). Also note that these result hold independently of the smoothness properties of the unsmoothed exact and simulated solutions. In the following, we will now focus on the error due to simulations, $V_{N,\lambda} - V_\lambda$, uniformly in $\lambda$. These results can then be combined with Theorem 2 to obtain the full error that also accounts for the smoothing bias.

### 5.1 Generalized Rust’s method

In this section, we provide an analysis of $V_{N,\lambda}$ allowing for general importance samplers. As a special case, we obtain an asymptotic theory for the self-approximating solution method (where $\Phi(z' | z, d)$ is restricted to be marginal distribution). The general results will then in turn be used in the analysis of the corresponding projection-based methods in the next section. First, we obtain the convergence rates of the simulated version of the expected value function:

**Theorem 3.** Suppose that Assumption 1 holds. Then $V_{N,\lambda}$ solving $\Gamma_{N,\lambda}(V_{N,\lambda}) = V_{N,\lambda}$ satisfies $\sup_{\lambda \in \Lambda} \|V_{N,\lambda} - V_\lambda\|_\infty = O_p(1/\sqrt{N})$ for any $\bar{\lambda} > 0$. If furthermore Assumption 2 holds with $s \geq 1$, then $\sup_{\lambda \in \Lambda} \|\partial V_{N,\lambda} / (\partial z) - \partial V_\lambda / (\partial z)\|_\infty = O_P \left( 1/\sqrt{N} \right)$.

The first part of this theorem covers models with both continuous, discrete and deterministic components and is analogous to results in Rust (1997b) and Pal and Stachurski (2013) who also show $\sqrt{N}$-convergence of their value function approximation. The second part is new and utilizes the smoothness of our problem; this requires Assumption 2 and so rules out discrete components. Importantly, the two convergence results hold uniformly over the smoothing parameter $\lambda$ and so there is no first-order effect from smoothing. In particular, if $\lambda$ satisfies $\sqrt{N}\lambda \to 0$, then Theorems 2 and 3 yield $\|V_{N,\lambda} - V\|_\infty = O_p(1/\sqrt{N})$. This is similar to convergence of smoothed empirical cdf where the indicator function is replaced by a smoothed version; this also does not affect the convergence rate. However, we conjecture that the higher-order derivatives of $V_{N,\lambda}(z)$ will not converge with $O_P \left( 1/\sqrt{N} \right)$ because these correspond to ill-posed problems.

The above result is then in turn used to derive the asymptotic distribution of $V_{N,\lambda}(z)$ uniformly in $(z, \lambda) \in Z \times (0, \bar{\lambda})$. Here, the smoothing proves important since it allows us to generalise the standard arguments used in the analysis of finite-dimensional extremum estimators to our setting. We can expand the “first-order condition”, $V_{N,\lambda} - \Gamma_{N,\lambda}(V_{N,\lambda}) = 0$, around $V_\lambda = \Gamma_\lambda(V_\lambda)$ to obtain

$$0 = \Gamma_\lambda(V_\lambda) - \Gamma_{N,\lambda}(V_\lambda) + \{I - \nabla \Gamma_{N,\lambda}(V_\lambda)\} [V_{N,\lambda} - V_\lambda] + o_p \left( 1/\sqrt{N} \right),$$

18
Theorem 4. Suppose that Assumption 1 hold together with either $Z$ being finite or Assumption 2 with $s \geq 1$. Then, $\sqrt{N}\{V_{N,\lambda} - V_\lambda\} \rightsquigarrow G_\nu$ on $B \left( Z \times (0, \bar{\lambda}) \right)^D$ where $G_\nu(z, \lambda) = \{I - \nabla \Gamma_{N,\lambda}(V_\lambda)\}^{-1}[G](z)$ is a $D$-dimensional Gaussian process. Here, $G$ has covariance kernel

$$
\Omega(z_1, \lambda_1, z_2, \lambda_2) = E_U \left[ g_i(z_1, \lambda_1) g_i(z_2, \lambda_2)^T \right],
$$

$$
g_i(z, \lambda, d) = \{G\lambda(u(S_i(z, d)) + \beta V_\lambda(Z_i(z, d))) w_i(z, d) - \Gamma_\lambda(V_\lambda)\} w_i(z, d),
$$

We conjecture that a similar result may hold for the non-smooth value function approximation. However, the proof of such a result would require different tools and stronger assumptions. In particular, the current proof only requires the following empirical process $\{z, \lambda\} \rightarrow \Gamma_{N,\lambda}(V_\lambda)$ to converge weakly. To allow for non-smooth value function approximation ($\lambda = 0$), our proof would have to be modified and would instead require that the empirical process $V \rightarrow \Gamma_N(V)$ converges weakly. For this to hold, we would need to be able to find a suitable function set that the estimated non-smooth solution, $V_N$; would be situated in and then verify that the entropy of this function set was suitably bounded. Standard choices of function sets are smooth classes, but $V_N$ is non-smooth and so the proof would be much more delicate.

Finally, for a complete analysis that takes into account the smoothing bias, state the following corollary to Theorem 4: For any $\lambda = \lambda_N \rightarrow 0$ such that $\lambda \sqrt{N} \rightarrow 0$,

$$
\sqrt{N}\{V_{N,\lambda} - V_0\} \rightsquigarrow G_\nu.
$$

5.2 Projection-based approximation of value functions

We now analyze the asymptotic properties of the projection-based estimator, $\hat{V}_{N,\lambda}$. To this end, we use the following decomposition of the over-all error,

$$
\hat{V}_{N,\lambda} - V_\lambda = \{\hat{V}_{N,\lambda} - V_{N,\lambda}\} + \{V_{N,\lambda} - V_\lambda\},
$$

where the second term converges weakly towards a Gaussian process, c.f. Theorem 4. What remains is to control the first term which is due to use of projection. This is done by imposing the following high-level assumption on the projection operator when applied to the function set $\mathcal{V}_{s,r} = \tilde{V}_{s,r}^D$ where $\tilde{V}_{s,r}$ was defined in (4.8):

**Assumption 3.** The projection operator $\Pi_K$ satisfies $\sup_{V \in \mathcal{V}_{s,r}} \|\Pi_K(V) - V\|_\infty = O_P(\rho_K)$, for some sequence $\rho_K \rightarrow 0$, where $s \geq 0$ was given in Assumption 2 and $r \geq \|V_\lambda\|_{s,\infty}$.

This is a high-level condition that requires the chosen function approximation method to have a uniform error rate over the function class $\mathcal{V}_{s,r}$. It allows for most known function approximations. As a particular example, we showed in Section 4.3 that $\rho_K = \log(K) / K^{(s+1)/d}$ for any value of $r < \infty$ when polynomials are employed. Compared to results on sieve approximations of
value functions found elsewhere in the literature, ours is weaker since we are here allowed to restrict attention to the smooth function class \( \mathcal{V}_{s,r} \). In contrast, sieve-based approximations developed in other papers, such as Munos and Szepesvari (2008) and Pal and Stachurski (2013), will tend suffer from bigger biases since the underlying value function being approximated is at most Lipschitz. In the case of \( \mathcal{Z} \) being discrete, we have \( \sup_{V \in \mathcal{V}_{s,r}} \| \Pi_K (V) - V \|_\infty = 0 \) for \( K > |\mathcal{Z}| \) under great generality and so there will be no asymptotic bias component due to sieve approximations in this case.

Theorem 6 together with the fact that \( \Gamma_{N,\lambda}(V_\lambda) - \Gamma_{\lambda}(V_\lambda) = O_P \left( \frac{1}{\sqrt{N}} \right) \), c.f. Proof of Theorem 3, now yield the following result:

**Theorem 5.** Suppose that Assumptions 1 and 3 hold. Then \( \hat{V}_{N,\lambda} \), defined as the solution to \( \Pi_K \Gamma_{N,\lambda}(\hat{V}_{N,\lambda}) = \hat{V}_{N,\lambda} \), satisfies \( \sup_{\lambda \in (0, \lambda_{\min})} \| \hat{V}_{N,\lambda} - V_\lambda \|_\infty = O_P(1/\sqrt{N}) + O_P(\rho_K) \). Suppose in addition that either \( \mathcal{Z} \) is finite or Assumption 2 holds with \( s \geq 1 \). Then, if \( \sqrt{N} \rho_K \to 0 \),

\[
\sqrt{N} \{ \hat{V}_{N,\lambda} - V_\lambda \} = \sqrt{N} \{ V_{N,\lambda} - V_\lambda \} + o_P(1) \to \mathcal{G}_V.
\]

As with the analysis of \( V_{N,\lambda} \), the rate result still goes through when no smoothing is employed \( (\lambda = 0) \) but the asymptotic distribution result requires smoothing \( (\lambda > 0) \). Compared to the rate results for \( V_{N,\lambda} \), the projection-based method suffers from an additional error due to the sieve approximation, \( O_P(\rho_K) \). This can be interpreted as a bias term, while \( O_P(1/\sqrt{N}) \) is its variance component which is shared with \( V_{N,\lambda} \). The requirement that \( \sqrt{N} \rho_K \to 0 \) is used to kill the sieve bias term so that \( \hat{V}_{N,\lambda} \) is centered around \( V_\lambda \).

The above result provides a refinement over existing results where a precise rate for the bias is not available; see, e.g., Lemma 5.2 in Pal and Stachurski (2013). This result also shows that there is an inherent curse-of-dimensionalilty built into our projection-based value function approximation when polynomial interpolation is employed: In high-dimensional models, a large number of basis functions are needed which in turn increases the computational effort. In the case of polynomial approximations, the rate condition becomes \( \sqrt{N} \log(K)/K^{(s+1)/d} \to 0 \) and so, as \( d \) increases, we need \( K \) to increase faster with \( N \) to kill the sieve bias component. However, also note that \( K \) as no first-order effect on the variance and so there is no bias-variance trade-off present. In particular, we can let \( K \) increase with \( N \) as fast as we wish and so our procedure should in principle also work for models with high-dimensional state space. However, this requires choosing both \( N \) and \( K \) large in order to control variance and bias which may lead to prohibitively expensive computations.

### 6 Numerical results

In this section we examine the numerical performance of the proposed solution algorithms with focus on how the theoretical results derived in the previous sections translate into practice. Our discussion will connect the theoretical results with verification in an actual model, and discuss some of the issues the researcher might come across. We demonstrate how different features of the model affect the performance. As a benchmark, to compare the sieve-based approximate
solutions against, we also implement the self-approximating method of Rust (1997b), which corresponds to $\hat{V}_R$ with $d\Phi (z'|z) = \phi (z')\,dz'$.

We measure the performance of a given approximate solution, say, $\hat{V} (z)$ in terms of its point-wise bias and variance defined as $Bias (z) := E[\hat{V} (z)] - V_0 (z)$ and $Var (z) := Var \left( \hat{V} (z) \right) = E \left[ (\hat{V} (z) - E[\hat{V} (z)])^2 \right]$, respectively, across different choices of $N$ and $K$. As overall measures of bias and variance, we will use $\|Bias\|_\infty = \sup_z |Bias (z)|$ and $\|Var\|_\infty = \sup_z |Var (z)|$. Given that the exact solution $V_0 (z)$ is unknown, we replace this by a very finely approximated solution computed as the average over 100 sieve approximations, where each approximate solution was based on using $M = 500$ evaluation points in the projection and $N = 2000$ pseudo-random draws. Each approximate solution was computed by successive approximation until a contraction tolerance of machine precision was reached. For the integrated value function the standard deviation across the 100 approximations were less than 0.017 which is tiny compared of the overall range of the integrated value function. We then approximate the point-wise bias and variance of a given method through $S \geq 1$ independent replications: Let $\hat{V}_1 (z), \ldots, \hat{V}_S (z)$ be the solutions obtained across the $S$ replications. We then compute

$$\hat{E}[\hat{V} (z)] = \frac{1}{S} \sum_{s=1}^{S} \hat{V}_s (z)$$

which in turn is used to approximate bias and variance by $Bias (z) = \hat{E}[\hat{V} (z)] - V_0 (z)$, and

$$Var (z) = \frac{1}{S} \sum_{s=1}^{S} (\hat{V}_s (z) - \hat{E}[\hat{V} (z)])^2.$$

### 6.1 A model of optimal replacement

To provide a test bed for comparison of the sieve-based approximation method, we use the well-known engine replacement model by Rust (1987). Rust’s model has become the basic framework for modeling dynamic discrete-choice problems and has been extensively used in other studies to evaluate the performance of alternative solution algorithms and estimators. While the model and its solution is well described in many papers, for completeness we briefly describe our variation of it below.

We consider the optimal replacement of a durable asset (such as a bus engine) whose controlled state $z_t \in \mathbb{R}_+$ is summarized by the accumulated utilization or mileage since last replacement. In each period, the decision maker faces the binary decision $d_t \in \mathcal{D} = \{0, 1\}$ whether to keep ($d_t = 0$) or replace ($d_t = 1$) the durable asset with a fixed replacement cost. If the asset is replaced, accumulated usage regenerates to zero. We let $\Delta z' = z' - z(1 - d)$ denote the change in utilization and assume that the per period change in usage (in absence of the replacement decision) is a mixture of a discrete distribution with a probability mass $\pi > 0$ at zero and a linearly transformed Beta distribution with shape parameters $a$ and $b$ and scale parameter $\sigma > 0$. Thus,

$$F (z'|z, d) = \pi I \{\Delta z' = 0\} + (1 - \pi) F_+ (z'|z, d),$$

where $F_+ (z'|z, d)$ has density $f_+ (z'|z, d) = \frac{1}{\sigma} f_\beta (\Delta z'/\sigma; a, b)$, $\pi \geq 0$ is the probability of no usage
Notes: The transition density $f_+ (z'|z,d)$ plotted on a subset of the state space, $(z, z') \in [0; 30]^2$. Parameters are $\sigma = 15$, $a = 2$, $b = 5$ and $\pi = 0$.

and $f_{\beta}(x; a, b)$ is the probability density function of the Beta distribution with shape parameters $a, b$.

We have chosen the scaled Beta distribution because of its flexibility and because it has bounded support between zero and $\sigma$, i.e. $f_+ (z'|z,d) = 0$ for $\Delta z' < 0$ or $\Delta z' > \sigma$. This is in line with the discretized model in the original formulation in Rust (1987) where monthly mileage were only allowed to take a few discrete values and monthly mileage is naturally bounded above and below (busses never drives backwards and there are limits how far a bus can drive with a month). We introduce probability mass $\pi$ at $\Delta z' = 0$ to allow for the possibility that the asset is not used in a given period and thereby can end in the same state with positive probability when $\pi > 0$. As explained below, this feature turns out to be quite important for the applicability of the self-approximating method of Rust (1997).

Let the expected per period operating costs be linear in usage, $c(z) = \theta_c \cdot 0.001 \cdot z$, and let $RC > 0$ denote the replacement cost if installing a new asset. Given the usage $z$ and the replacement decision $d$, the state and decision dependent per period utility is the given by $u(z,d) + \lambda \eta(d)$ where

$$u(z,d) = \begin{cases} RC + c(0) & \text{if } d = 0 \\ c(z) & \text{if } d = 1 \end{cases}$$

and the utility shocks $\eta = [\eta(0), \eta(1)]$ are i.i.d. extreme value.

In the numerical illustrations below we use the following set of benchmark parameter values unless otherwise specified. These parameters are chosen to illustrate the properties of the solution methods under study, and so will not mimic Rust (1987) exactly. For example, we set replacement cost to $RC = 10$ and the cost function parameter to $\theta_c = 2$ so that $RC$ is 5 times as large as $c(1000)$. This implies a large variation in the probability of replacement over $z$ compared to Rust (1987) and a more curved value function. The parameters indexing the transition density $f_+ (z'|z,d)$ are $\sigma = 15$, $a = 2$, $b = 5$ and $\pi = 0$. This implies a quite sparse transition density as illustrated in Figure 1, which is similar to the fitted model in Rust (1987). Note that $f_+ (z'|z,d)$
is so peaked and sparse that in order to show the functional form we only plot $f_+ (z'|z,d)$ for $(z,z') \in [0; 30]^2$ while the actual support of $z_t$ is the positive half line and so unbounded. Thus, the theory does not apply directly, since we throughout assumed bounded support, but we expect that the theory extends to the unbounded case after suitable modifications.

We have chosen a discount factor $\beta = 0.95$ which is quite small relative to Rust (1987) where $\beta = 0.9999$. The main reason for choosing a smaller discount factor is to avoid overly long computation times for the implementation of successive approximation algorithm whose convergence rate crucially depends on $\beta$. We will later illustrate how the convergence properties are affected by the value of this parameter.

We set $\lambda = 1$ throughout, and do not numerically study the role of the scale of the random utility shocks below. Instead we refer to Iskhakov, Jørgensen, Rust and Schjerning (2017) who present extensive numerical experiments where Extreme Value taste shocks are used as a logit smoothing device of an underlying model of interest. They work with a slightly different model that combines both discrete and continuous choices, and their primary focus is on how discrete choice specific Extreme Value taste shocks can be used to smooth out kinks in the value functions and discontinuities in the optimal policy rules. However, their numerical results clearly illustrate how the original problem with $\lambda = 0$ can be approximated by an augmented smoothed model to any desirable degree of precision. They show that that even in the case where the addition of taste shocks results in a misspecification of the model, the implied smoothing improves the numerical accuracy of the solution and reduces computation time without increasing the approximation bias significantly.

In Figure 2 we plot the corresponding “exact” solution as described earlier. Importantly, the value function is smooth and so well-approximated by polynomial interpolation methods.

### 6.2 Implementation of Simulated Bellman operators.

The simulated Bellman operators in (3.4) and (3.5) require the user to choose a importance sampling distribution. We consider two choices: First, for the implementation of the self-approximating solution method proposed in Rust (1997b), we choose $\Phi (dz'|z,d) = \phi (z') dz'$ with $\phi (z')$ being a uniform density with support support $[0, z^{\max}]$, where $0 < z^{\max} < \infty$ is
a truncation point chosen by us. This entails that the simulated Bellman operator used for
the self-approximating value function is biased since we do not sample from the full support
\( Z = (0, \infty) \). We will explain below why we do not choose \( \phi(z') \) to have unbounded support.

Second, for the implementation of the projection-based versions given in (6.1) and (??), we
choose \( \Phi(dz'|z,d) = F(dz'|z,d) \) and so the second version of our simulated Bellman operator is
unbiased.

One issue with using a marginal density in the importance sampling is that fails to adapt to the
particular shape of the support of \( F(z'|z,d) \). Suppose that \( F \) and \( \Phi \) have densities; it is then
well-known from the literature on importance sampling that our simulated Bellman operator only
provides a good estimate of the exact one if \( \phi(z'|z,d) \propto G_\lambda(u(z') + \beta V(z')) f(z'|z,d). \) When
\( \phi(z'|z,d) = \phi(z') \) and \( f(z'|z,d) \) has bounded support for a given value of \( z \), this condition
evidently fails. In particular, for a given choice of \( z \), many of the draws from \( \phi(z') \) will fall outside
the support of \( f(z'|z,d) \) and so will not contribute. In contrast, when \( \phi(z'|z,d) = f(z'|z,d) \),
the draws from \( \phi \) will by construction fall within the support of \( f(z'|z,d) \). This can be seen in
Figure 3 where we have plotted the random grids for the two implementations together with the
actual support of \( f(z'|z,d) \). In the left-hand side panel we have plotted pairs of the uniform
draws, \( (Z_i, Z_j) \) for \( i,j = 1, ..., N \), used for Rust’s self-approximating method with \( N = 400 \) and
\( z_{max} = 1,000 \), while in the right-hand side we have plotted \( (Z_{0,i}, Z_{1,j} (Z_{0,i}, d) \) where \( Z_{0,i} \) are
uniform draws and \( Z_{1,j} (z,d) \sim f(z'|z,d) \). In both cases, we have marked the pairs for which the
corresponding density, \( f(Z_j|Z_i,d) \) and \( f(Z_{1,j} (Z_{0,i}, d)|Z_{0,i}, d) \), respectively, is positive. Clearly,
the use of a marginal importance sampling density, leads to very poor coverage of the actual
support of \( f(z'|z,d) \) as \( z \) varies while by construction \( \Phi(dz'|z,d) = F(dz'|z,d) \) does an excellent
job. This translates into the former simulated Bellman operator exhibiting much larger variance
compared to the latter.

This issue is further amplified when we introduce the normalization given in eq. (??): When
\( \phi(z') = I \{ 0 < z' < z_{max} \} \) is uniform, we have

\[
W_{N,i}(Z_j,d) = \frac{f(Z_i|Z_{j},d)}{\sum_{k=1}^{N} f(Z_k|Z_{j},d)}
\]

If \( f(z'|z,d) \) has bounded support, it often happens that \( \sum_{k=1}^{N} f(Z_k|Z_{j},d) = 0 \) for even large
values of \( N \) and so the simulated Bellman operator is not even well-defined. One could be
tempted to simply set \( W_{N,i}(Z_j,d) = 0 \) when \( \sum_{k=1}^{N} f(Z_k|Z_{j},d) = 0 \), but this tends to lead to
poor results since we then risk setting the approximate value function to zero in the state \( Z_j \)
when in fact it is non-zero. This issue will of course vanish as \( N \to \infty \), but this on the other hand
increases the computational burden when solving for the value function. One way to circumvent
having to choose a very large \( N \) is to design the transition distribution and the sampler so that
there exists a “special draw”, say \( Z_1 \), so that \( f(Z_1|Z_{j},d) > 0 \) for all \( j \). In our application, we
ensure this by introducing a probability point mass \( \pi > 0 \) at \( \Delta z = 0 \), c.f. (6.1). We then split up
the integral into the discrete and continuous component and evaluate the continuous component
using importance sampling while the discrete one can be evaluated analytically. Thus, \( \pi > 0 \)
functions as a regularization device.

This brings us to the reason why we do not choose \( \phi(z') \) as a density with unbounded support
Notes: In the left panel we present the grids used for Rust’s self-approximate random Bellman operator. We have uniformly sampled a random grid, \{Z_1, ..., Z_N\} on the interval [0; 1000] with N = 400. Dots (.) mark sampled grid points in \(R^2: Z_N \times Z_N\), plus (+) mark grid points where \(f(z_j|z_i, d = 0) > 0\) and circles (o) mark points where \(f(z_j|z_i, d = 1) > 0\). In the right panel, we plot the grid the projected random Bellman operator, where we have sampled directly from the conditional transition density in each of the \(M = 400\) uniformly spaced evaluation points. To have equally many grid-points with non-zero transition density we only need \(N = 400 \times \sigma / \max(Z_N) = 9\) random grids for each of the \(M = 400\) evaluation points. Both figures show only a subset of the state space, \((z, z') \in [0; 100]^2\). Parameters are \(\sigma = 15\), \(a = 2\), \(b = 5\) and \(\pi = 0.0000000001\).
to avoid the issue of truncation. In our initial experimentation, we did try out sampling from distributions with unbounded support, but the above numerical issues were even more severe since the draws become even more spread out in this case. Figure 4 shows how the solution depends on \( z_{\text{max}} \). The effect of the truncation \( z_{\text{max}} \) will be model specific and in practice experimentation is required. We set \( z_{\text{max}} = 1000 \) to limit the source of this bias.

We mention this for practical reasons, as it will affect the quality of the fixed point differently for different methods proposed in this paper. We cannot simply set \( z_{\text{max}} = 1000.000 \), as this would make the grid less dense for a given sample size, \( N \). For Rust’s self-approximating method, increasing \( z_{\text{max}} \) will amplify the problems with undefined sample weights \( w_{N,i}(z,d) \) mentioned above.

As can be seen from the right-hand side panel of Figure 3, the above mentioned issues are not present in the second version of the simulated Bellman operator, which is used for the implementation of the projection-based value function approximation. In particular, since we sample directly from the conditional transition density, we have by construction that \( f(Z_{1,j}(z,d)|z,d) > 0 \) for all \( j \) and all \( z \), which translates into the sample weights satisfying \( w_{N,i}(z,d) > 0 \).

### 6.3 Sieve spaces used in projection-based method

For the projection-based method, we need to choose the sieve space used in constructing \( \Pi_K \). We here focus on polynomial interpolation as discussed in Example 1 using either Chebyshev basis functions or B-Splines as described in further detail below.

**Chebyshev basis functions** Interpolation and approximation by Chebyshev polynomials of the first kind have well-known good properties when approximating functions on bounded intervals. Recall that Chebyshev polynomials are defined on \([-1,1]\). We then choose \( 0 \leq z_{\text{min}} < z_{\text{max}} < \infty \) and define the \( k \)th basis function as follows for any \( z \in \mathbb{R} \),

\[
B_{c,k}(z) = \begin{cases} 
\cos((k-1) \arccos(T(z))) & |T(z)| \leq 1 \\
\text{sign}(T(z))^k & |T(z)| > 1 
\end{cases},
\]

where

\[
T(z) = \frac{2z - z_{\text{min}}}{z_{\text{max}} - z_{\text{min}}} - 1
\]

linearly transforms our \( z \)'s to the interval \([-1,1]\). In particular, the basis functions are “truncated” and are set to one outside the interval \([z_{\text{min}}, z_{\text{max}}]\). This is done to avoid any erratic extrapolation. For interpolation it is natural to use the Chebyshev nodes to minimize the presence of Runge’s phenomenon. For approximation, that is \( M > K \), it is less clear what to do, though one possibility is to augment the Chebyshev nodes associated with the order \( K - 1 \) Chebyshev polynomial with random numbers distributed uniformly across the interval we’re considering.
B-Splines  We use cardinal B(asis)-splines to form our B-spline spaces, so they are represented by a knot vector with equidistant entries \((0, \frac{1}{M+1}, \frac{2}{M+1}, \ldots, \frac{M}{M+1}, 1)\), and the Cox-de Boor recursion

\[
\bar{B}_{i,0}(z) = \begin{cases} 
1 & \text{if } t_i \leq z < t_{i+1} \\
0 & \text{otherwise}
\end{cases}
\]

\[
\bar{B}_{i,k}(z) = \frac{z-t_i}{t_{i+k} - t_i} \bar{B}_{i,k-1}(z) + \frac{t_{i+k+1} - z}{t_{i+k+1} - t_{i+1}} \bar{B}_{i+1,k-1}(z).
\]

For interpolation purposes we use the so-called Universal (Parameters) Method by Tjahjowidodo et al. (2017). This amounts to choosing the \(M\)-grid to consist of the unique maximizers of all B-splines of degree \(k \geq 1\), or any point if \(k = 0\) in which case we set it to the first \(K\) elements of the knot vector. Since the points given by the Universal Method makes the interpolation quite well-behaved, we also use these points for approximation, and augment them the same way as we described for the Chebyshev Sieve spaces.

The above are defined on the unit interval \([0, 1]\) and so the final basis functions are chosen as

\[
B_{c,k}(z) = \begin{cases} 
\bar{B}_k(T(z)) & 0 \leq T(z) \leq 1 \\
\text{(sign}(T(z))) & \text{otherwise}
\end{cases}
\]

where now

\[
T(z) = \frac{z - z_{\text{min}}}{z_{\text{max}} - z_{\text{min}}}.
\]

6.4 Effect of adding a smoothing device

In the theory section we considered two different interpretations of the extreme value type I taste shocks. One interpretation is that the shock acts as smoothing device, and another is a structural error that is present to facilitate maximum likelihood estimation to obtain logit-style choice probabilities. If we are in the world where the taste shocks are structural, we can simply use the usual closed form solution for the value functions and choice probabilities. No further smoothing is needed, and all the terms involving the scale parameter on the smoothing shocks simply drop out. Another situation could be that we wanted to include some other type of shocks, such as normally distributed taste shocks with some correlation structure. Then, we don’t have closed form solutions for the choice probabilities and value functions, and we have to simulate them. Then, as described in the theory section, we run into trouble, because none of the theoretical results apply, because the max operator is not differentiable everywhere. As argued, we can then control the problems introduced by simulating an integral over a the max operator, by introducing a smoothing device. Here, we briefly, numerically study the results that state that in our setup, the mean squared error (MSE) does not explode for small smoothing shocks. Actually, small amounts of smoothing may even lower MSE, because it can mitigate some of the behavior that can show up when evaluating the monte Carlo simulation over the
In Figure 5 we look at $\|\text{MSE}(z)\|_\infty$ for different combinations of $K = M$ and $\lambda$ (the smoothing scale parameter) in the model described above. To emulate a situation where we do not have a closed form solution for the log-sum. Instead we use Monte Carlo simulation to simulate the taste shocks. This is problematic for our theory, so we add an extreme value type I taste shock on top of the simulations, and vary the scale parameter $\lambda$. The merit of doing this exercise, is that for this model we can actually chose not to simulate the taste shocks, so we have a good numerical reference solution for $N \uparrow \infty$ and $\lambda \downarrow 0$. In the figure, we see that for large $\lambda$ the error grows as expected. We also see that for $\lambda \downarrow 0$ the smoothing error disappears, and the only error that is left comes from simulation of the structural taste shocks and state transitions. This is promising if all we want to achieve with the smoother, is to be on solid ground with respect to the results from the theory. In some cases, for $K \in \{6, 15\}$ we get that a positive $\lambda$ actually minimizes the mean squared error. We have no theory or heuristics for choosing an optimal $\lambda$, but at least some smoothing doesn’t seem to hurt.

6.5 Convergence of iterative solutions methods

Contraction properties of projection-based Bellman operator

As demonstrated in Theorem 1, $\Gamma_N$ and $\bar{\Gamma}_N$ are contractions and so Rust’s self-approximating method is guaranteed to converge using successive approximations. In contrast, the projection-based successive approximation method is only guaranteed to converge if the projection is non-expansive, c.f. discussion in Section 5.2. For a given choice of basis functions, $\|P_K\|_{\text{op,}\infty}$ as defined in Section 5.2 provides us with a lower bound for $\|\Pi_K\|_{\text{op,}\infty}$. In particular, $\|P_K\|_{\text{op,}\infty} > 1$
Figure 6: Fixed points of four approximate Bellman operators, $M = 64$, and their associated lower bounds of on Lipschitz constants $L$ of the projection, based on approximating $v$ (integrated value function) or $V$ (expected value function).

![Graph showing fixed points and lower bounds](image)

implies $||\Pi_K||_{op,\infty} > 1$. However, even if $||P_K||_{op,\infty} > 1$, successive approximation may still converge. For example, all our implementations of the projection-based successive approximation algorithm converged despite the fact that most of the projections were expansive. This is of course model and sieve basis specific, so no general promises can be made in this regard.

At the same time, a non-expansive projection operator may not provide a good function approximation. That is, the approximation bias $\Pi_K (V_0) - V_0$ may be “large” for a given choice of $K$. Unfortunately, many guaranteed non-expansive projections tend to require choosing $K$ large in order for it to provide a good approximation. Thus, in the implementation one has to weight these two features, numerical convergence and approximation quality, when choosing a suitable sieve basis.

We illustrate this point in in Figure 6 by plotting four different approximate solutions obtained by projection-based successive approximation. The four approximate solutions were obtained using $M = 64$ evaluation points, but with different number of Chebyshev basis functions, $K$. We show results for two variants: one where we approximate the expected value function, and one where we approximate the integrated value function. Clearly, for the two versions with $K = 4$ we obtain the best approximations, but as we see, the lower bounds on the Lipschitz constants, $L := ||\Pi_K||_{op,\infty}$, presented earlier are both larger than 1, so a priori we could not have been sure that successive approximations would converge (while in fact they did). With a Lipschitz constants of the projection of $L = 1.00$, the projection with $K = 1$ is obviously non-expansive. Nevertheless, the method of successive approximations did in fact converge to a tolerance of $10^{-12}$ for $K = 1$ and $K = 4$.

**Convergence properties of successive approximation and Newton’s method**

As shown in Section 3, the smoothing induced by the extreme value type I taste shock means that the Frechet derivative of the simulated Bellman operator exists, and so we can use Newton’s method. It is well-known that the method of successive approximations will perform very poorly when the modulus of the contraction mapping (which is proportional to the discount factor) is
close to 1. Therefore, Newton’s method is often the preferred method for computing contraction fixed points because it has guaranteed quadratic convergence rate once we are in its domain of attraction\(^1\).

The computational advantages of Newton’s method are illustrated in Figure 7 where we have plotted the log residual error of the current value function approximation (relative to the “exact” solution) against the iteration count for successive approximations and Newton steps, respectively. The convergence of successive approximations is bound by the discount factor and clearly slows down as \(\beta\) approaches 1, whereas Newton steps converges very rapidly regardless of \(\beta\) (see also Iskhakov, Lee, Rust, Schjerning and Seo (2016) for a comparison in the context of structural estimation based on the Nested Fixed Point (NFXP) algorithm). This highlights the advantages of the smoothing device since this allows for the use of Newton’s method.

6.6 Approximation quality

We will now look at how the approximate value function is affected by the number of draws and the chosen projection basis. The goal is to demonstrate the rate results of the theoretical sections, and to compare the two types of basis functions spaces that we described above. We will take a partial approach and fix \(N\) to study the role of \(K\), and fix \(K\) to study the role of \(N\). We here only report results for the approximation of the integrated value function defined in eq. (??) since the results for the corresponding expected value function were qualitatively the same.

Effect of varying \(K\) for projection-based value function approximation

The theory for the projection-based value function approximation informs us that the choice of the basis functions will have a first-order effect on the bias while only a second-order effect on

\(^1\) Since Newton’s method is only locally convergent, Rust (1987) used a “poly-algorithm” that starts with successive approximations to ensure global convergence, and switches to newton iterations only after it detects that the domain of attraction has been reached.
the variance. In particular, we expect $\text{Bias}(z)$, as defined in the beginning of this section, to satisfy $\text{Bias}(z) \cong \Pi_K (V_0)(z) - V_0(z)$, c.f. discussion following Theorem 5, while $\text{Var}(z)$ should be much less affected by $K$. The actual size of the bias obviously depends on the curvature of the unknown value function $V_0$, and the particular choice of basis functions. As can be seen from Figure 2, $V_0$ is quite smooth and so we expect it to be well-approximated by a small number of polynomial basis functions. The theory is confirmed by the pointwise bias and standard deviation ($\sqrt{\text{Var}(z)}$) reported in Figure 8.

First, from looking the left-hand side panel of Figure 8, we see that using first-order B-splines lead to significantly more point-wise bias than the other two sieve bases, namely second-order B-splines and Chebyshev polynomials. This is accordance with theory since we know that a smooth function is better approximated by higher-order polynomials, c.f. the error rates reported in Example 1 as a function of $s$. At the same time, second-order B-splines and Chebyshev polynomials exhibit very similar biases for a given $K$.

The right-hand side panel of Figure 8 shows the point-wise standard deviation across different choices of $K$ for the three different sieve bases. Consistent with the theory, the standard deviation of the value function approximation is not very sensitive to the particular choice of the sieve basis and the number of basis functions uses. That is, the sieve basis mostly affect the bias with only minor impact on the variance.

Finally, we examine how the bias behaves as we further increase $K$. Figure 9 plots $\parallel \text{Bias} \parallel_{\infty}$ as a function of $K$. Similar to Figure 8 we see much more rapid convergence when smooth basis functions are used, and with little improvement for $K$ greater than 9. This is not surprising given the reported shape of $V_0$. The second-order B-splines and Chebyshev basis functions produce very similar fits and errors, even if they’re evaluated on different grids and the B-splines are much more local in the sense described above. Indeed, the curves are practically overlapping.

This is in accordance with the asymptotic theory that predicts that higher-order B-splines and Chebyshev polynomials should lead to similar biases. Moreover, the theory informs us that if $V_0$ is analytic, and this is the case in this particular implementation, we should expect the bias to vanish with rate $O(\text{log}K)$ when using polynomial interpolation. The bias indeed does go to zero very quickly and so the numerical results support the theory.

**Simulation error, rates of convergence and asymptotic normality**

We now compare the rates of convergence for Rust’s self-approximating method and the projection based method. For both methods, theory tells us that $N$ should have a first-order effect on the variance of the approximate value function which is supposed to vanish at rate $1/N$, c.f. Theorems 5 and 3. The approximated value functions suffer from potential three biases: Smoothing bias that vanishes with rate $\lambda$, sieve approximation bias that vanish with rate $\rho_K$, and simulation bias due to approximation error in the simulated bellman operator. Our asymptotic theory is silent about the size of simulation bias and the rate with which it should vanish with $N$. However, we can think of both the projection based method and Rust’s method as a nonlinear GMM-estimator where the simulated Bellman operator defines the sample moments. Importing results for GMM estimators, see, e.g., Newey and Smith (2004), we should expect a higher-order bias of order $1/N$. 

31
Figure 8: Point-wise bias and standard deviation of solutions for various choices of $K$ using different interpolation schemes, $N = 200$, $S = 200$, $\sigma = 15$. 

- **Chebyshev**
- 1st order B-splines
- 2nd order B-splines
Figure 9: Sup-norm of bias of solutions for various choices of $K$ using various interpolation schemes, $N = 200$, $S = 200$.

Figure 10: Convergence results

Notes: Discount factor is $\beta = 0.95$, utility function parameters are $\theta = 2$, $RC = 10$, $\lambda = 1$ and transition parameters are $a = 2$, $b = 5$ and $\pi = 0.00000001$. Uniform bias and variance were estimated using 500 evaluation points and $S = 2000$ implementations.
In Figure 10 we investigate this prediction by plotting $\|\text{Bias}\|_\infty$ and $\|\sqrt{\text{Var}}\|_\infty$ (defined above) as a function of $N$. We do this for both the projection based method (left panels) and for Rust’s self-approximating method (right panels) for two different choices of $\sigma$. To examine the rate with which the simulation bias and variance vanish we estimate the following an exponential regression by NLS $\|\sqrt{\text{Var}}\|_\infty = \exp(\alpha_{SD} + \rho_{SD} \ln(N))$ and $\|\text{Bias}\|_\infty = \exp(\alpha_{Bias} + \rho_{Bias} \ln(N))$ where $N^{\rho_{SD}}$ and $N^{\rho_{Bias}}$ measures the rate for $\|\sqrt{\text{Var}}\|_\infty$ and $\|\text{Bias}\|_\infty$ respectively. The resulting regression fit estimates are reported in both Figure 10 as well as in Table 1. In Table 1 we present bias and standard deviation for $N = 500$ as well as their rates of convergence both methods; with various values of $K$ for the sieve approximation method. According to the theory, the variance should vanish with rate $1/N$ for both methods and we therefore expect $\rho_{SD} = -0.5$ so that $\|\sqrt{\text{Var}}\|_\infty$ vanish with $1/\sqrt{N}$. For the projection based method, we see that the rate with which the standard deviation shrinks to zero is indeed close to $-0.5$ for all values of $K > 1$ and irrespectively of the value of $\sigma$. For Rust’s method we estimate the rate to $1/N^{0.541}$ when $\sigma = 100$, which is in line with the theory. However, $\|\sqrt{\text{Var}}\|_\infty$ is found to vanish with rate $1/N^{3.6}$ for $\sigma = 15$. This seems to indicate that the asymptotic theory developed in Theorems 3 and 4 do not provide a very accurate approximation of the performance of Rust’s method for small and moderate choices of $N$ when the support of $m_{z_t|z_{t-1}} = z$ is small ($\sigma = 15$). We conjecture that the discrepancy between theoretical predictions and numerical results for Rust’s method is due to the aforementioned issues with the marginal importance sampler discussed in Section 6.2: Many of the draws are not used in the computation of the simulated Bellman operator because they fall outside the support of $z|z_{t-1} = z$ for a given choice of $z$. Thus, the effective number of draws is smaller than $N$ and changes as $z$ varies.

For the projection based method, the main source of bias is due to the sieve projection. From Figure 8, we see that, with $N = 200$ and $K = 9$, the projection-based methods using second-order B-splines or Chebyshev polynomials have virtually no bias, and both Figure 10 as well as in Table 1 also confirms that we practically eliminate by approximate the value function using Chebyshev polynomials with $K = 20$. However, there still remains a small simulation bias for that decays with $N$. For small $K$, we see that the bias is roughly independent of $N$. As $K$
increases so does the dependence on $N$. However, even for $K = 20$ we estimate $\rho_{\text{Bias}}$ to be 0.21 and 0.30 for $\sigma = 15$ and $\sigma = 100$ respectively, the rate of convergence is far from $1/N$. This is probably due to the presence of higher-order bias components that our asymptotic theory doesn’t account for or because there is still some remaining sieve approximation bias left even with $K = 20$.

For Rust’s method, there is no sieve projection bias but a larger simulation induced bias that decreases with $N$. We obtain rate estimates of $1/N^{1.7}$ and $1/N^{1.4}$ for the bias when $\sigma = 15$ and $\sigma = 100$ respectively; these are slightly faster than expected but not too far from the theoretical predictions of $1/N$. For Rust’s method, bias constitute more than half of RMSE when $N < 600$ for $\sigma = 15$ (or $N < 400$ for $\sigma = 100$), but since $||\text{Bias}||_\infty$ decays faster than $||\sqrt{\text{Var}}||_\infty$, the simulation bias eventually becomes second order for large $N$.

Comparing $||\text{MSE}||_\infty = ||\text{Bias}^2 + \text{Var}||_\infty$ for $N = 500$ we find that the our projection-based method clearly dominates Rust’s self-approximating method when $\sigma = 15$, whereas $||\text{MSE}||_\infty$ is better for Rust’s method when $\sigma = 100$. As discussed in Subsection 6.2, a key requirement for successful implementation of Rust’s method is that $\sum_{k=1}^{N} f(Z_k|Z_j, d) > 0$, for $j = 1, ..., N$. In our application, $f(z'|z, d)$ has bounded support for all $z$ with the size of the support being determined by the scale parameter $\sigma$. Thus, we expect the error of Rust’s self-approximating method will tend to be larger when $\sigma$ is low. The opposite is the case for the projection based method which becomes more precise for smaller value of $\sigma$, since no draws are wasted when sampling directly from the transition density. When $f(z'|z, d)$ has bounded support Rust’s method completely break down if $N$ is too small since we easily have that $\sum_{k=1}^{N} f(Z_k|Z_j, d) = 0$ for at least one grid point $Z_j$. This also happens when $\sigma = 100$ if $N$ is small enough.

Theorems 4 and 5 state that when $N$ is large, the approximate value functions should be normally distributed. We here investigate whether this asymptotic approximation is useful in practice by looking at the pointwise distribution of the approximate solution of integrated value function obtained through both methods. In Figure 11, we plot the distribution of $r_N(z) = (\hat{\nu}_N(z) - E[\hat{\nu}_N(z)])/\sqrt{\text{Var}(\hat{\nu}_N(z))}$ for $z = 500$ together with the standard normal distribution. It is here important to note we do not center the estimate around $v(z)$ but instead around $E[v_N(z)]$; this is due to the sizable bias of Rust’s method. For the projection based method, we see that the distribution of $r_N(500)$ is quite close to the standard normal irrespectively of the value of $\sigma$. In contrast, the normal distribution appears to be a poor approximation for Rust’s method when $\sigma = 15$. As expected the approximation gets better as $N$ and $\sigma$ increases.

### 6.7 Performance in the bivariate case

We now examine how Rust’s method and the projection-based method perform in the bivariate case ($d_z = 2$). To this end, we follow the approach of Arcidiacono, Bayer, Bugni and James (2013) and Rust (1997a) and build a $d_z$-dimensional model by adding up $d_z$ independent versions of the univariate model considered so far. That is, we define

$$u(z_t, d_t) = \sum_{i=1}^{d_z} u_{uni}(z_{t,i}, d_{t,i})$$
Figure 11: Asymptotic Normality

\[
\left( V_N - V \right) / \sqrt{\text{var}(V_N)}
\]

Notes: Each panel shows the densities of \( r_N(z) = \left( \hat{v}_N(z) - E[\hat{v}_N(z)] \right) / \sqrt{\text{var}(\hat{v}_N(z))} \) for \( z = 500 \). The kernel density was estimated using \( S = 2000 \) solutions for each sample size \( N \). Discount factor is \( \beta = 0.95 \), utility function parameters are \( \theta_c = 2, \ RC = 10, \lambda = 1 \) and transition parameters are \( \sigma = 100, \ a = 2, \ b = 5 \) and \( \pi = 0.000000001 \).
\[ F(z_t|z_{t-1}, d_{t-1}) = \prod_{i=1}^{d_z} F_{uni}(z_{t,i}|z_{t-1,i}, d_{t-1,i}) \]

\[ D = D_1 \times D_2 \times \cdots \times D_{d_z} \]

where \( F_{uni}(z_{t,i}|z_{t-1,i}, d_{t-1,i}) \) and \( u_{uni}(z,d) \) are identical to the state transition and per-period utility in the univariate case, c.f. Section 6.1. Note here that \( z_{t,i} \) and \( z_{t,j} \) are fully independent of each other, \( i \neq j \) and \( d_t = (d_{t,1}, \ldots, d_{t,d_z}) \) so that number of alternatives are \( 2^{d_z} \). Thus, the model considers the joint replacement decision of \( d_z \) assets whose stochastic usages \( (z_{t,1}, \ldots, z_{t,d_z}) \) are mutually independent. Conveniently, the solution (value function) of this multidimensional problem is simply the sum of the solutions to each of the underlying univariate models

\[ V(z_1, \ldots, z_{d_z}) = \sum_{i=1}^{d_z} V_{uni}(z_i), \]  

(6.2)

where \( V_{uni}(z_i) \) is the solution to the model in Section 6.1. This is a rather simplistic multivariate model but it comes with the major advantage that we can obtain a very accurate approximation of the exact solution by simply adding up the “exact” solution found for the univariate case. With a more complicated multidimensional structure, the computational cost of finding the “exact” solution would be much higher.

In the following, we report results for Rust’s method and the projection-based method for the bivariate case. For both solution methods, we forgo the knowledge of the additive structure of the solution and so treat the above model as a “proper” multivariate problem.

**Simulation error**

Given the issues with Rust’s method for small values of \( \sigma = 15 \), we here focus exclusively on the case \( \sigma = 100 \). To get a sense of the pointwise performance of the method, we have plotted the pointwise error of \( \hat{V}_R(z_1, z_2) \) using \( N = 3000 \) state draws in Figure 12 together with the pointwise errors of the corresponding replacement probabilities. The overall shape and level of the value function is quite well captured, and the same is true for the policy. We see some noise in both value function and the choice probabilities, and some of this comes from the fact that the issues described above regarding sparseness of grids and bounded support for the transition densities are amplified here. The problems are especially present in the off-grid evaluations, where it can sometimes be hard to have sufficiently many nodes in the area where we want to evaluate the value function or policies.

Next, we examine the uniform bias, variance and RMSE of \( \hat{V}_R(z_1, z_2) \) as we increase \( N \). These are plotted in Figure 13. As in the univariate case, all three vanish as \( N \) increases, but at the same time the magnitude of all three is much larger. Note in particular, that we here only report results for \( N \geq 1400 \) since for \( N \) below this threshold the Rust’s method became numerically unstable. This seems to indicate a certain type of curse-of-dimensionality in this particular application of Rust’s method. This was caused by the aforementioned inability of the uniform importance sampler to adapt to the changing conditional support of \( z_t \) which becomes even more
Notes: Discount factor is $\beta = 0.95$, utility function parameters are $\theta_c = 2$, $RC = 10$, $\lambda = 1$ and transition parameters are $\sigma = 100$, $a = 2$, $b = 5$ and $\pi = 0.0000001$. The “exact” solution was computed by averaging over $S = 100$ solutions, each found using the smoothed random Bellman operator with $N = 3000$ pseudo random draws. Each fixed point was found using a contraction tolerance of machine precision.
Sieve approximation error

In the implementation of the projection-based method, we use as bivariate sieve basis the tensor product of the univariate Chebyshev polynomials presented in Section 6.3. That is, given, say, $J$ univariate Chebyshev polynomials, say, $p_1, ..., p_J$, we construct our bivariate basis functions as $B_{i,j}(z_1, z_2) = p_i(z_1)p_j(z_2)$ for $i, j = 1, ..., J$ yielding a total of $K = J^2$ bivariate basis functions. One could potentially utilize the particular sparse structure of the value function of the model under consideration when choosing the bivariate sieve basis. Specifically, one could reduce the dimension of sieve basis from $K = J^2$ to $2J$ by setting all cross-product parameters to zero, since these are irrelevant due to the additive structure of $V$ in eq. (6.2). This is illustrated in Table 2, where we report the coefficients for one particular projection-based bivariate value function estimate using a tensor product of $J = 5$ Chebyshev polynomials. As can be seen, the coefficients associated with cross-product terms are effectively zero and so the effective number of basis functions is $2J = 10$ instead of $J^2 = 25$. This highlights the potential benefits of using more sparse sieve bases, such as complete polynomials or sparse grids, instead of the tensor product sieve space considered here.

That the cross-terms to turn out to be zero for Chebyshev basis functions, where each basis function has full support over our individual states, does not generalize to other basis functions. B-Splines that we have also used above, have much more limited support. Higher order B-Splines technically have full support as well, but they very quickly converge to zero when moving away from their maximum. This can be an advantage, as B-Splines can potentially better capture functions with surfaces that are harder to approximate. As we see in 3, this means that we need...
Table 2: Coefficients on tensor product Chebyshev basis functions in the 2D model of engine replacement for $K = J^2 = 25$, $N = 200$.

<table>
<thead>
<tr>
<th>$J_1 \backslash J_2$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-38.4713</td>
<td>-4.4754</td>
<td>1.6176</td>
<td>-0.256420</td>
<td>-0.064960</td>
</tr>
<tr>
<td>2</td>
<td>-4.4754</td>
<td>1.9662e-14</td>
<td>-6.2341e-15</td>
<td>-1.1318e-15</td>
<td>2.5392e-15</td>
</tr>
<tr>
<td>3</td>
<td>1.6176</td>
<td>7.2256e-15</td>
<td>5.6179e-14</td>
<td>-2.0548e-14</td>
<td>-3.3049e-15</td>
</tr>
<tr>
<td>4</td>
<td>-0.2564</td>
<td>-1.0110e-14</td>
<td>-8.8673e-15</td>
<td>7.5672e-15</td>
<td>-2.9838e-15</td>
</tr>
<tr>
<td>5</td>
<td>-0.0649</td>
<td>4.0869e-15</td>
<td>-1.5251e-14</td>
<td>3.4571e-15</td>
<td>1.4218e-15</td>
</tr>
</tbody>
</table>

Table 3: Coefficients on tensor product 2nd order B-Spline basis functions in the 2D model of engine replacement for $K = J^2 = 25$, $N = 200$.

<table>
<thead>
<tr>
<th>$J_1 \backslash J_2$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-31.5060</td>
<td>-36.6455</td>
<td>-39.5844</td>
<td>-40.6285</td>
<td>-41.0321</td>
</tr>
</tbody>
</table>

all the cross-terms.

Since we do not exploit the structure of the problem in any way, we will ignore it in the following, and use a tensor product sieve basis, not complete polynomials, since we are interested in the practical contents of Theorems 5 and ??, where no particular sparsity/special structure of the model is assumed to be known. As mentioned in the introduction to this section, this is very much in line with other papers like our own.

In Figure 14, we report the uniform bias of the projection-based method with $N$ chosen very large. We observe that the bias vanishes as $K$ increases as in the one-dimensional model. However, convergence is obviously slower in $K$ in the bivariate case relative to the univariate case. This is consistent with the error rates reported in Example 1 where the rate slows down as the dimension of the problem increases.

**Model with interaction**

Consider a model with interaction between the two busses’ contributions in the utility function. Such a structure could reflect the fact that there is limited time, space, man power and so on. If the both busses are very worn down, they’re effectively more to maintain, and setting aside time to change the engine of both busses at once may require paying overtime or temporarily hiring more mechanics.

The main reason to do this extra exercise is to verify that the solution converges in a similar way as above, and that the fast convergence observed for the Chebyshev basis functions was not just due to the simple, additive structure. Note, that there’s a reason we’re using this modified model to nuance the conclusions, and that this wasn’t just the only model we solve, and show convergence results for. The additive model above had the distinct advantage, that we could solve the univariate model and get a very credible solution out by essentially adding...
Figure 14: Bias of value function in two-dimensional replacement model for varying $K$.

Table 4: Some of the coefficients on the basis functions upon convergence with $J = 50^2$.

<table>
<thead>
<tr>
<th>$J_1 \backslash J_2$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-52.200</td>
<td>-5.070</td>
<td>2.700</td>
<td>-1.170</td>
<td>0.478</td>
<td>-0.145</td>
<td>-0.006</td>
<td>0.038</td>
<td>-0.024</td>
<td>0.007</td>
</tr>
<tr>
<td>2</td>
<td>-5.070</td>
<td>-1.560</td>
<td>0.135</td>
<td>0.225</td>
<td>-0.057</td>
<td>-0.023</td>
<td>0.013</td>
<td>-0.006</td>
<td>0.004</td>
<td>0.001</td>
</tr>
<tr>
<td>3</td>
<td>2.700</td>
<td>0.135</td>
<td>-0.176</td>
<td>0.032</td>
<td>0.046</td>
<td>-0.021</td>
<td>-0.004</td>
<td>0.005</td>
<td>-0.002</td>
<td>0.001</td>
</tr>
<tr>
<td>4</td>
<td>-1.170</td>
<td>0.225</td>
<td>0.032</td>
<td>-0.087</td>
<td>0.019</td>
<td>0.019</td>
<td>-0.012</td>
<td>0.002</td>
<td>0.001</td>
<td>-0.001</td>
</tr>
<tr>
<td>5</td>
<td>0.478</td>
<td>-0.057</td>
<td>0.046</td>
<td>0.019</td>
<td>-0.038</td>
<td>0.010</td>
<td>0.009</td>
<td>-0.008</td>
<td>0.002</td>
<td>0.000</td>
</tr>
<tr>
<td>6</td>
<td>-0.145</td>
<td>-0.023</td>
<td>-0.021</td>
<td>0.019</td>
<td>0.010</td>
<td>-0.018</td>
<td>0.005</td>
<td>0.005</td>
<td>-0.004</td>
<td>0.002</td>
</tr>
<tr>
<td>7</td>
<td>-0.006</td>
<td>0.013</td>
<td>-0.004</td>
<td>-0.012</td>
<td>0.009</td>
<td>0.005</td>
<td>-0.009</td>
<td>0.003</td>
<td>0.002</td>
<td>-0.003</td>
</tr>
<tr>
<td>8</td>
<td>0.038</td>
<td>-0.006</td>
<td>0.005</td>
<td>0.002</td>
<td>-0.008</td>
<td>0.005</td>
<td>0.003</td>
<td>-0.005</td>
<td>0.002</td>
<td>0.001</td>
</tr>
<tr>
<td>9</td>
<td>-0.024</td>
<td>0.004</td>
<td>-0.002</td>
<td>0.001</td>
<td>0.002</td>
<td>-0.004</td>
<td>0.002</td>
<td>0.002</td>
<td>-0.003</td>
<td>0.001</td>
</tr>
<tr>
<td>10</td>
<td>0.007</td>
<td>0.001</td>
<td>0.001</td>
<td>-0.001</td>
<td>0.0002</td>
<td>0.002</td>
<td>-0.003</td>
<td>0.001</td>
<td>0.001</td>
<td>-0.002</td>
</tr>
</tbody>
</table>

the value functions for the two univariate solutions together. In the bivariate, and in general in the multivariate, case, we’re slightly more constrained in the extent to which we can max out all the parameters that control the fineness of the approximation.

Nevertheless, the coefficients on the basis functions tend to zero quite quickly. The sup-norm of the difference in the value function at of 40000 evaluation grids is on the order of $10^{-5}$ when comparing the solutions with $J = 50^2 = 2500$ and $J = 30^2 = 900$ basis functions, and individual coefficients fall below $10^{-6}$ for univariate basis functions and cross products beyond the 22nd univariate basis functions, and below $10^{-8}$ around the 30th basis functions. In 4 we see the first coefficients on the first ten basis functions in each dimension and their interactions. Compared to the Chebyshev-based solution earlier, we see quite significant coefficients on the coefficients for the cross-terms. We will now see if this affects the fast convergence rates from before.
6.8 Comparison of run-times for Newton’s Method and Successive Approximations

The theory we provide in this paper includes a choice specific shock that acts to smooth out the max operator, and the conditional choice probabilities. With such shocks we get a Bellman operator that is differentiable, and we can calculate the Jacobian to apply Newton’s method as an alternative to successive approximation (SA). The relative merits of the two approaches are somewhat well-understood and studied. Still, we will now show some partial results on the relative performance in terms of run-times between the method. We do this to validate what can be expected, but also to highlight some of the choices we’ve made. Before looking at the results, we want to emphasize that other choices of basis spaces, and other simulators of the state transitions might change some of these conclusions. We’ll come back to this.

First, let’s determine what can be expected. If the Bellman operator is a contraction, SA is known to be robust. The Bellman operator presented in this paper is only a contraction in the limit, but at no point did we encounter failure to converge or solutions that looked out of the ordinary. This has held for any type of initialization of coefficients, parameter choices, and so on. For practical reasons, assume that the Bellman operator is a contraction, then the cost of the robustness is the rate of convergence. Convergence will be linear, and given by the Lipshitz constant of the Bellman operator: the discount factor. The closer the discount factor is to one, the slower convergence. A related disadvantage of SA is, that for models where the Bellman operator is expensive to evaluate, say if simulating the taste shocks, the many iterations lead to long total run-times. Newton’s Method (NM) on the other hand, obtains quadratic convergence rates as long as we are in the so-called basin of attraction. For general non-linear systems of equations, or their fixed point formulations, we cannot say what the shape or volume of this basin of attraction is. The conclusion is, that as long as we are close enough to the fixed point, we will make rapid progress towards an accurate solution measured by the number of algorithm iterations.

Compared to SA, each NK iteration is of course much more expensive, given the Bellman operator is not too expensive to apply. We need to solve a linear system of equations to find the search direction, and solving a linear system of equations without special structure has the same computational complexity as matrix multiplication, so between $O(n^{2.373})$ (for the asymptotically fastest known algorithm) and $O(n^3)$ (for the naïve matrix multiplication). However, we explicitly try to alleviate this by representing our problem as a $K$-by-$K$ system instead of an $M$-by-$M$ system. If we chose to work with the expected value function, each dimension would increase by a factor equal to the number of choices. Given the complexity of the step, performance would be affected significantly as a result.

In Figure 15 we see best of 10 run-times for various levels of $K$ and the discount factor $\beta$. The idea is to highlight the specific pros and cons of the two competing solution methods. In the top row, we see a clear winner for all values of the discount factor: NM is fastest. We also see that setup time is not negligible if we compare the run-time of NM with the setup time.

---

2 See Le Gall (2014)
Figure 15: Run-times including setup time for Newton’s Method (NM) and Successive Approximations (SA) across specifications of $K$ and $\beta$ for $N = 10$ for the 1-dimensional bus engine problem.

$\beta = 0.9, K = 5$

$\beta = 0.95, K = 5$

$\beta = 0.99, K = 5$

$\beta = 0.9, K = 100$

$\beta = 0.95, K = 100$

$\beta = 0.99, K = 100$

$\beta = 0.9, K = 500$

$\beta = 0.95, K = 500$

$\beta = 0.99, K = 500$
The second row shows the same overall picture as the first, but with a larger fraction of time spent setting up the projections, etc. Remember, we are pre-calculating all expectations. If we didn’t do this, setup time would fall significantly, but each iteration would be slower. For SA this would be a problem because the Bellman operator needs to be evaluated so many times. We’ve increased $K$ from 5 to 100 here, so we’re far beyond the case where the numerical results showed that most of the bias was gone in the 1D case. In the bottom two rows, we see that setup time continues to increase as we set $K = 500$, and the complexity of solving the linear system of equations begins to kick in, and, now, SA is actually faster than NM for moderate discount factors. For the highest specification, we still see NM being fastest for $\beta = 0.99$. Rust’s original preferred empirical specification had $\beta = 0.9999$. In that case we’d really have to increase $K$ for SA to be the better choice. This means that for high discount rate models, even higher dimensional models, where $K$ is the product of a $K_d$ for each dimension, Newton’s method might be the only viable solution method.

While we don’t show a full set of results here, it should be noted that the actual time to do one SA iteration is one more margin to consider when choosing solution method. We have focused on the case of with a closed-form expression for the expectation over the iid extreme value type shocks here, but the theory allows for much more general shocks. In the section where we studied the effects of adding a smoothing device, we simulated the taste shocks as any other state. This was slightly contrived, but in a real life setting there may be other factors that force the researcher to simulate taste shocks. Such examples could be: multiplicative shocks, gaussian shocks with correlation across choices, and more. This is all within the theory presented here, but will generally increase the computational cost of evaluating the Bellman operator, and as a result SA will be hurt relatively more than NM, because NM only needs very few evaluations of the Bellman operator.

7 Conclusion

We have proposed a projection-based value function approximation that combines smoothing, simulations and sieve approximations. The asymptotic theory shows that compared to the exact solution to the simulated Bellman operator, the use of sieves leads to an additional bias component. This bias component can be controlled by using a large enough number of sieve basis functions. Moreover, we derive the large-sample distribution of the projection-based value function and find that, with suitably bias control, it is first-order equivalent to the exact solution to the simulated Bellman operator.

A number of numerical experiments investigate the performance of the projection-based value function approximation using as benchmark Rust’s method. We find that the projection-based method performs much better than Rust’s with smaller bias and variance.
References


A Auxiliary Results

We derive a general result for projection-based approximate solutions to functional fixed-points. Let \((\mathcal{X}, \|\cdot\|)\) be a normed vector space and \(\Psi : \mathcal{X} \to \mathcal{X}\) be some contraction mapping w.r.t. \(\|\cdot\|\) so that there exists a unique solution \(x_0 \in \mathcal{X}\) to \(x = \Psi(x)\). Let \(\Psi_N\) be an approximation to \(\Psi\) and let \(\Pi_K\) be a projection operator, \(K, N \geq 1\); the following theorem characterizes the (set of) solution(s) to \(x = (\Pi_K \Psi_N)(x)\) as \(K, N \to \infty\):

**Theorem 6.** Suppose that (i) \(\|\Psi_N(x_0) - \Psi(x_0)\| = O_p(\rho \Psi_N)\) for some \(\rho \Psi_N \to 0\); (ii) for some \(\beta < 1\), \(\|\Psi_N(x) - \Psi_N(y)\| \leq \beta \|x - y\|\) for all \(N\) large enough and all \(x, y\); (iii) \(\Pi_K\) satisfies \(\sup_{\|x - x_0\| < \delta} \|\Pi_K(x) - x\| = O_p(\rho_{\Pi,K})\) for some \(\rho_{\Pi,K} \to 0\). Then there exists a unique solution \(\hat{x} \in \mathcal{X}\) to \(x = (\Pi_K \Psi_N)(x)\) with probability approaching one (w.p.a.1) satisfying, with \(x_N = \Psi_N(x_N)\),

\[
\|\hat{x} - x_0\| \leq \|\hat{x} - x_N\| + \|x_N - x_0\| = O_p(\rho_{\Pi,K}) + O_p(\rho \Psi_N).
\]

as \(N \to \infty\).

**Proof.** We first observe that due to (ii), there exists a unique solution \(x_N = \Psi_N(x_N)\) which satisfies

\[
\|x_N - x_0\| = \|\Psi_N(x_N) - \Psi(x_0)\| \leq \|\Psi_N(x_N) - \Psi_N(x_0)\| + \|\Psi_N(x_0) - \Psi(x_0)\| \\
\leq \beta \|x_N - x_0\| + \|\Psi_N(x_0) - \Psi(x_0)\|,
\]

and so \(\|x_N - x_0\| \leq \|\Psi_N(x_0) - \Psi(x_0)\| / (1 - \beta) = O_p(\rho \Psi_N).\) Next, combining (ii) and (iii), we see that \(\Pi_K \Psi_N\) is a contraction mapping w.p.a.1. with Lipschitz coefficient \(\beta\), and so \(\hat{x}\) defined in the theorem exists and is unique w.p.a.1. Moreover, by the same arguments employed in the analysis of \(x_N\) together with the fact that \(\|x_N - x_0\| < \delta\) w.p.a.1,

\[
\|\hat{x} - x_N\| \leq \|\Pi_K \Psi_N(x_N) - \Psi_N(x_N)\| / (1 - \beta) \leq \sup_{\|x - x_0\| < \delta} \|\Pi_K(x) - x\| = O_p(\rho_{\Pi,K}).
\]

**Theorem 7.** Suppose that in addition to the conditions stated in Theorem 6, the following ones are satisfied: (i) \(\rho_{\Psi,N} \{\Psi_N(x_0) - \Psi(x_0)\} \sim \mathcal{G}\) in \((\mathcal{X}, \|\cdot\|)\) and (ii) \(\Psi_N(x_0)\) is almost surely Fréchet differentiable at \(x_0\) with Fréchet differential \(\nabla \Psi_N(x_0)\) such that \(\|\Psi_N(x) - \Psi_N(x_0) - \nabla \Psi_N(x_0)(x - x_0)\| = o_P(\|x - x_0\|)\) for all \(x\) in a neighbourhood of \(x_0\); (iii) it satisfies \(\|\nabla \Psi_N(x_0) - \nabla \Psi(x_0)\|_{op} = o_p(1)\), where \(\|\nabla \Psi(x_0)\|_{op} = \sup_{\|dm\|_1 = 1} \|\nabla \Psi(x_0)\|_{dm}\|\). Then \(\{I - \nabla \Psi(x_0)\} \rho_{\Psi,N} \{x_N - x_0\} \sim \mathcal{G}\). If furthermore \(dm \mapsto \{I - \nabla \Psi(x_0)\} [dm]\) has a continuous inverse, then \(\rho_{\Psi,N} \{x_N - x_0\} \sim \{I - \nabla \Psi(x_0)\}^{-1} \mathcal{G}\).

**Proof.** Combining assumption (ii) with Lemma 6,

\[
0 = x_N - \Psi_N(x_N) = x_0 - \Psi_N(x_0) + \{I - \nabla \Psi_N(x_0)\} [x_N - x_0] + o_p(\|x_N - x_0\|) \\
= \Psi(x_0) - \Psi_N(x_0) + \{I - \nabla \Psi_N(x_0)\} [x_N - x_0] + o_p(\rho_{\Psi,N}),
\]

47
where, using (iii),
\[
\| \{ I - \nabla \Psi_N(x_0) \} [x_N - x_0] - \{ I - \nabla \Psi(x_0) \} [x_N - x_0] \| \leq \| \nabla \Psi_N(x_0) - \nabla \Psi(x_0) \|_{op} \| x_N - x_0 \| = o_P(1) O_P(\rho_N) = o_P(\rho_N).
\]

The first part now follows from (i), while the second part follows by the continuous mapping theorem.

**B  Proofs**

**Proof of Theorem 1.** By the quasi-linearity of the social surplus function, for any \( V_1, V_2 \in \mathbb{B}(Z)^D \),

\[
\Gamma_N(V_1)(z, d) = \sum_{i=1}^{N} G_{\lambda} (u (S_i(z)) + \beta V_2 (Z_i(z)) + \beta [V_1 (Z_i(z)) - V_2 (Z_i(z))] ) w_{N,i}(z, d)
\]

\[
\leq \sum_{i=1}^{N} G_{\lambda} (u (S_i(z)) + \beta V_2 (Z_i(z)) + \beta \| V_1 - V_2 \| \infty \mathbf{1}_D ) w_{N,i}(z, d)
\]

\[
= \sum_{i=1}^{N} G_{\lambda} (u (S_i(z)) + \beta V_2 (Z_i(z)) ) w_{N,i}(z, d) + \beta \| V_1 - V_2 \| \infty \sum_{i=1}^{N} w_{N,i}(z, d)
\]

\[
= \Gamma_N(V_2)(z, d) + \beta \| V_1 - V_2 \| \infty,
\]

where \( \mathbf{1}_d = (1, \ldots, 1) \in \mathbb{R}^D \) and we have used that \( \sum_{i=1}^{N} w_{N,i}(z, d) = 1 \) by construction. Similarly, for any \( v_1, v_2 \in \mathbb{B}(Z) \),

\[
\begin{aligned}
\bar{\Gamma}_N(v_1)(z) &\leq \sum_{j=1}^{N} G_{\lambda} \left( u (z, t_j(z)) + \beta \sum_{i=1}^{N} v_2 (Z_{1,i}(z)) \circ w_{z,N,i}(z) + \beta \| v_1 - v_2 \| \infty \mathbf{1}_D \right) w_{\varepsilon,N,j}(z)
\leq \sum_{j=1}^{N} G_{\lambda} \left( u (z, t_j(z)) + \beta \sum_{i=1}^{N} v_2 (Z_{1,i}(z)) \circ w_{z,N,i}(z) \right) w_{\varepsilon,N,j}(z) + \beta \| v_1 - v_2 \| \infty
= \bar{\Gamma}_N(v_2)(z) + \beta \| v_1 - v_2 \| \infty.
\end{aligned}
\]

Next, we prove that \( V_{N,\lambda}(z) \) is \( s \geq 1 \) times continuously differentiable: We know that \( \Gamma_N \) is a contraction mapping on \( \mathbb{B}(Z) \). But the set of \( s \geq 0 \) continuously differentiable functions \( \mathcal{C}_s(Z) \) is a closed subset of \( \mathbb{B}(Z) \) and so the result will follow if \( \Gamma_N \left( \mathcal{C}_s(Z)^D \right) \subseteq \mathcal{C}_s(Z)^D \). But for any \( V \in \mathcal{C}_s(Z)^D \), it follows straightforwardly by the chain rule in conjunction with the stated assumptions that \( \Gamma_N(V)(z) = \sum_{i=1}^{N} G_{\lambda} (u (S_i(z)) + \beta V (Z_i(z))) w_{N,i}(z) \) is \( s \geq 0 \) continuously differentiable w.r.t. \( z \).

For later use, we derive an expression of the first-order derivative \( \partial V_{N,\lambda}/(\partial z_j) \). To this end, first note that \( 0 = \{ I - \Gamma_{N,\lambda} \} (V_{N,\lambda})(z) \). The implicit function theorem then implies that, assuming that \( M \mapsto [I - \nabla \Gamma_{N,\lambda}]^{-1}(V)[M] \) is well-defined and continuous,

\[
\frac{\partial V_{N,\lambda}(z)}{\partial z_j} = \{ I - \nabla \Gamma_{N,\lambda} \}^{-1} \left[ \Gamma_{N,j}(V_{N,\lambda}) \right] (z), \tag{B.1}
\]
where, with \( \dot{G}_{d,\lambda}(r) = \frac{\partial G_{d}(r)}{\partial r(d)} = \exp \left( \frac{r(d)}{\lambda} \right) / \sum_{d' \in D} \exp \left( \frac{r(d')}{\lambda} \right) \),

\[
\dot{\Gamma}_{N,j}(V)(z) = \sum_{i=1}^{N} \sum_{d \in D} \dot{G}_{\lambda,d}(u(S_{i}(z)) + \beta V(Z_{i}(z))) \frac{\partial w_{N,i}(z)}{\partial z_{j}} + \sum_{i=1}^{N} \dot{G}_{\lambda}(u(S_{i}(z)) + \beta V(Z_{i}(z))) \frac{\partial w_{N,i}(z)}{\partial z_{j}}.
\]

But it is easily checked that \( M \mapsto \nabla \Gamma_{N,\lambda}(V) \) is a continuous linear operator with norm \( \| \nabla \Gamma_{N,\lambda} \|_{op} \leq \beta \). By the Banach inverse theorem, the inverse of \( I - \nabla \Gamma_{N,\lambda} \) is therefore well-defined and continuous. For later use, also observe that

\[
\left\| \frac{\partial \hat{\Gamma}_{N}(z)}{\partial z_{j}} \right\| \leq \left\| \left\{ I - \nabla \hat{\Gamma}_{N}(\hat{V}_{N}) \right\}^{-1} \left\| \hat{\Gamma}_{N,j}(\hat{V}_{N}) \right\| \leq (1 - \beta)^{-1} \left\| \hat{\Gamma}_{N,j}(\hat{V}_{N}) \right\|,
\]

where

\[
\left\| \hat{\Gamma}_{N,j}(V) \right\| \leq \sum_{i=1}^{N} \sum_{d \in D} \dot{G}_{\lambda,d}(u(S_{i}(z)) + \beta V(Z_{i}(z))) w_{N,i}(z) \left\| \frac{\partial u(\cdot)}{\partial z_{j}} \right\|_{\infty} + \sum_{i=1}^{N} \dot{G}_{\lambda}(u(S_{i}(z)) + \beta V(Z_{i}(z))) \left\| \frac{\partial w_{N,i}(z)}{\partial z_{j}} \right\|_{\infty} \leq \left\| \frac{\partial u(\cdot)}{\partial z_{j}} \right\|_{\infty} + \sum_{i=1}^{N} \left\| \frac{\partial w_{N,i}(z)}{\partial z_{j}} \right\|_{\infty} \times \left\{ \| u(\cdot) \|_{\infty} + \beta \| V(\cdot) \|_{\infty} \right\}.
\]

\( \square \)

**Proof of Theorem 2.** We only show the result for \( \hat{V}_{\lambda} \); the proof for \( V_{N,\lambda} \) is analogous. Applying (3.3), the following holds for any \( V \),

\[
|\Gamma_{\lambda}(V)(z,d) - \Gamma(V)(z,d)| \leq \int_{\mathcal{D} \times \mathcal{E}} \max_{d' \in D} \left\{ u(s',d') + \beta V(z',d') \right\} - G_{\lambda}(u(s') + \beta V(z')) \, dF_{s}(ds'|z,d)
\]

\[
\leq \sup_{r \in \mathcal{D}} \left| G_{\lambda}(r) - \max_{d \in D} r(d) \right| \int_{\mathcal{D} \times \mathcal{E}} dF_{s}(ds'|z,d)
\]

\[
= \lambda \log D.
\]

The result now follows from Theorem 6 with \( \Pi_{K}(V) = V \) and \( \Psi_{N} = \Gamma_{\lambda,N} \). \( \square \)

**Proof of Theorem 3.** We apply Theorem 6 with \( \Pi_{K}(V) = V \) and \( \Psi_{N} = \Gamma_{N} \). We define

\[
\tilde{u}(U;z) := u(\psi_{z}(z,U),\psi_{e}(z,U)), \quad \tilde{w}(U;z) = w(\psi(z,U)|z), \quad \tilde{V}(U;z,\lambda) = V_{\lambda}(\psi_{z}(z,U)),
\]

so that we can write \( \Gamma_{\lambda}(V_{\lambda})(z) = E_{U} \left[ G_{\lambda} \left( \tilde{u}(U;z) + \beta \tilde{V}(U;z,\lambda) \right) \tilde{w}(U;z) \right] \) and

\[
\Gamma_{N,\lambda}(V)(z) = \frac{\dot{\Gamma}_{N,\lambda}(V)(z)}{W_{N}(z)}.
\]

(B.2)
where
\[
\tilde{\Gamma}_{N,\lambda}(V_\lambda)(z) = \frac{1}{N} \sum_{i=1}^{N} G_\lambda \left( \bar{u}(U_i; z) + \beta \bar{V}(U_i; z, \lambda) \right) \bar{w}(U_i; z), \tag{B.3}
\]
\[
W_N(z) = \frac{1}{N} \sum_{j=1}^{N} \bar{w}(U_i; z), \tag{B.4}
\]

Due to Assumption 1(ii), it follows from Theorem 2.7.11 of van der Vaart and Wellner (1996) that the bracketing number of
\[
\mathcal{W} := \{ u \mapsto \bar{w}(u; z) \mid z \in \mathcal{Z} \} \tag{B.5}
\]
satisfies \( N((2\varepsilon C, \mathcal{W}, \| \cdot \|_\infty) \lesssim N(\varepsilon, \mathcal{Z}, \| \cdot \|) \lesssim \varepsilon^{-d_\mathcal{W}}. \) Thus, \( \mathcal{W} \) is a \( P_U \)-Donsker class since
\[
\int_0^{\infty} \sqrt{\log N((\varepsilon, \mathcal{W}, \| \cdot \|_\infty) d\varepsilon \lesssim \int_0^{\infty} \sqrt{\log N(\varepsilon, \mathcal{Z}, \| \cdot \|) d\varepsilon < \infty}, \text{ and so}
\]
\[
\sup_{z \in \mathcal{Z}} |W_N(z) - 1| = O_P \left( \frac{1}{\sqrt{N}} \right), \tag{B.6}
\]

Next, to analyze \( \tilde{\Gamma}_{N,\lambda} \), first observe that the function class \( \mathcal{G} \) defined in Lemma 1 also is a Donsker class. Given that both \( \mathcal{G} \) and \( \mathcal{W} \) are both bounded classes, \( \mathcal{G} \cdot \mathcal{W} \) is also Donsker and so
\[
\sup_{\lambda \in (0, 1)} \left\| \tilde{\Gamma}_{N,\lambda}(V_\lambda) - \Gamma_\lambda(V_\lambda) \right\|_\infty = O_P \left( \frac{1}{\sqrt{N}} \right) \text{ which combined with (B.6) yield sup}_{\lambda \in (0, 1)} \| \Gamma_{N,\lambda}(V_\lambda) - \Gamma_\lambda(V_\lambda) \|_\infty = O_P \left( \frac{1}{\sqrt{N}} \right). \]

We conclude from Theorem 6 that \( \sup_{\lambda \in (0, 1)} \| V_{N,\lambda} - V_\lambda \|_\infty = O_P(1/\sqrt{N}) \).

To show convergence of the first-order derivatives of the approximate value function, recall that \( \frac{\partial V_{N,\lambda}}{\partial z_j} \) satisfies (B.1) while \( \frac{\partial V_\lambda}{\partial z_j} \) solves
\[
\frac{\partial V_\lambda(z)}{\partial z_j} = \left( I - \nabla \Gamma_\lambda(V_\lambda) \right)^{-1} \left[ \tilde{\Gamma}_{\lambda,j}(V_\lambda) \right](z),
\]

where
\[
\nabla \Gamma_\lambda(V_\lambda)[M](z) = \beta \sum_{d \in \mathcal{D}} E_U \left[ \hat{G}_{d,\lambda} \left( \bar{u}(U; z) + \beta \bar{V}(U; z, \lambda) \right) M(U; z, d) \bar{w}(U; z) \right]
\]
and
\[
\hat{\Gamma}_{\lambda,j}(V)(z) = \sum_{d \in \mathcal{D}} E_U \left[ \hat{G}_{d,\lambda} \left( \bar{u}(U; z) + \beta \bar{V}(U; z, \lambda) \right) \frac{\partial \bar{u}(U; z)}{\partial z_j} \bar{w}(U; z) \right] + E_U \left[ G_\lambda \left( \bar{u}(U; z) + \beta \bar{V}(U; z, \lambda) \right) \frac{\partial \bar{w}(U; z)}{\partial z_j} \right].
\]

Here note that \( M \mapsto \{ I - \nabla \Gamma_\lambda(V_\lambda) \}[M] \) and \( M \mapsto \{ I - \nabla \Gamma_{N,\lambda}(V_{N,\lambda}) \}[M] \) are both bounded linear operators with operator norm \( 1 - \beta \). In particular, they are both contraction mappings. Thus, we can again apply 6 with \( \Psi_N(M) = (I - \nabla \Gamma_{N,\lambda}(V_{N,\lambda}))[M] \). To show convergence of \( \Psi_N(\partial V_\lambda/\partial z_j) \), first note that \( \hat{G}_{\lambda,d}(r) \) is Lipschitz in \( r \) uniformly in \( \lambda \), c.f. the proof of Lemma
1. This combined with $W$ being a Donsker class implies
\[
\|\nabla \Gamma_{N,\lambda}(V_{N,\lambda}) [\partial V_{\lambda} / (\partial z_j)] - \nabla \Gamma_{\lambda}(V_{\lambda}) [\partial V_{\lambda} / (\partial z_j)]\|_\infty = O_P \left( \| V_{N,\lambda} - V_{\lambda} \|_\infty \right) = O_P \left( 1/\sqrt{N} \right),
\]
while, using arguments similar to the ones used in the first part of the proof in conjunction with Lemma 1,
\[
\|\nabla \Gamma_{N,\lambda}(V_{\lambda}) [\partial V_{\lambda} / (\partial z_j)] - \nabla \Gamma_{\lambda}(V_{\lambda}) [\partial V_{\lambda} / (\partial z_j)]\|_\infty = O_P \left( 1/\sqrt{N} \right).
\]
This completes the proof.

Proof of Theorem 4. We apply Theorem 7. First recall that $M \mapsto \{ I - \nabla \Gamma_{\lambda} \}^{-1} (V_{\lambda}) [M]$ is well-defined and continuous and that
\[
\sqrt{N} \left( \Gamma_{N,\lambda}(V_{\lambda}) - \Gamma_{\lambda}(V_{\lambda}) \right) \sim (G_1, G_2), c.f. \text{ proof of Theorem } 3.
\]
The latter result implies that
\[
\sqrt{N} \left( \Gamma_{N,\lambda}(V_{\lambda}) - \Gamma_{\lambda}(V_{\lambda}) \right) = \sqrt{N} \left( \Gamma_{N,\lambda}(V_{\lambda}) - \Gamma_{\lambda}(V_{\lambda}) \right) - \Gamma_{\lambda}(V_{\lambda}) \sqrt{N} \{ W_N - 1 \} + o_P (1)
\sim G := G_1 - \Gamma_{\lambda}(V_{\lambda}) G_2.
\]
It is easily seen that the influence function of $\Gamma_{N,\lambda}(V_{\lambda})$ takes the form
\[
g(U; z, \lambda) = \left\{ G_{\lambda} \left( \tilde{u}(U; z) + \beta \tilde{V}(U; z, \lambda) \right) - \Gamma_{\lambda}(V_{\lambda}) \right\} \tilde{w}(U; z),
\]
and so $G(z, \lambda)$ has covariance kernel
\[
\Omega(z_1, \lambda_1, z_2, \lambda_2) = E_U \left[ g(U; z_1, \lambda_1) g(U; z_2, \lambda_2) \right].
\]
What remains to be shown is the uniform convergence of $M \mapsto \nabla \Gamma_{N,\lambda}(V_{\lambda}) [M]$ over some suitable function set $\mathcal{M}$ chosen such that $V_{N,\lambda} - V_{\lambda} \in \mathcal{M}$ w.p.a.1. In the case where $Z$ is finite, $\mathcal{M}$ is also finite-dimensional and so has finite bracketing number. In the case where $Z$ is not finite, we instead invoke Assumption 2 which implies that $V_{N,\lambda}$ and $V_{\lambda}$ are both smooth, c.f. Theorem 2. Moreover, from Theorem 3, $\sup_{\lambda \in (0, \lambda)} \| V_{N,\lambda} - \partial V_{\lambda} \|_\infty = O_P \left( 1/\sqrt{N} \right)$ and $\sup_{\lambda \in (0, \lambda)} \| \partial V_{N,\lambda} / (\partial z) - \partial V_{\lambda} / (\partial z) \|_\infty = O_P \left( 1/\sqrt{N} \right)$. Thus, we can choose $\mathcal{M} = \{ M \in C_1 (Z) : \| M \|_{1, \infty} < r \}$ for some $r < \infty$, where $\| M \|_{s, \infty}$ was defined in (4.9). By Theorem 2.7.1 in van der Vaart and Wellner (1996), $N \| (\epsilon, \mathcal{M}, L_1 (P_U)) \| < \infty$ for any given $\epsilon > 0$ which combined with the fact that the bracketing numbers of $\hat{G}_d$ and $W$ are also finite imply that $\mathcal{F} = \hat{G}_d \cdot W \cdot \mathcal{M}$ has finite bracketing number (where we have used that each of the three functions classes are uniformly bounded in $U$). It now follows from the Glivenko-Cantelli Theorem that
\[
\sup_{M \in \mathcal{M}} \| \nabla \Gamma_{N,\lambda}(V_{\lambda}) [M] - \nabla \Gamma_{\lambda}(V_{\lambda}) [M] \|_\infty = o_p (1) \]
which together with (B.6) yield the desired result.

Proof of Theorem 5. The rate result is an immediate consequence of Theorem 6 together with Assumption 3. For the weak convergence result, we use the decomposition (??) where, by Theorem 6, $\| \hat{V}_{N,\lambda} - V_{N,\lambda} \|_\infty = O_P (\rho_K) = o_p \left( 1/\sqrt{N} \right)$ while the second term converges weakly according to Theorem 4.
C Lemmas

Lemma 1. Suppose that Assumption 2 hold. Then, for any $\bar{\lambda} < \infty$,

$$\mathcal{G} \equiv \left\{ G_\lambda (u(\psi(z,\cdot)) + \beta V_\lambda (\psi_z(z,\cdot))) \mid (z, \lambda) \in Z \times (0, \bar{\lambda}) \right\} \quad (C.1)$$

$$\mathcal{G}_d \equiv \left\{ G_{\lambda,d} (u(\psi(z,\cdot)) + \beta V_\lambda (\psi_z(z,\cdot))) \mid (z, \lambda) \in Z \times (0, \bar{\lambda}) \right\}, \quad (C.2)$$

satisfy $\mathcal{N} (\varepsilon, \mathcal{G}, \|\cdot\|_2) \preceq \varepsilon^{-v}$ and $\mathcal{N} (\varepsilon, \mathcal{G}_d, \|\cdot\|_2) \preceq \varepsilon^{-v}$, $d \in \mathcal{D}$, for some $v \geq 1$.

Proof. We first analyze the properties of $V_\lambda (z)$: It is easily seen that $\Gamma_\lambda (V)(z)$ is Lipschitz w.r.t $z$ uniformly over $\lambda \in (0, \bar{\lambda})$ for any function $V(z)$ which is Lipschitz. Thus, by the same arguments as used in the proof of Theorem 1, $V_\lambda (z)$ is Lipschitz in $z$ uniformly over $\lambda \in (0, \bar{\lambda})$. Moreover, $\lambda \mapsto \Gamma_\lambda (V)(z) = EU \left[ G_\lambda (u(U; z) + \beta V(\psi(z, U))) \right] \bar{w}(U; z)$ is continuously differentiable by the dominated convergence theorem which, by the implicit function theorem, in turn implies that $\lambda \mapsto V_\lambda (z)$ is continuously differentiable with its derivative being on the form

$$\frac{\partial V_\lambda (z)}{\partial \lambda} = \{ I - \nabla \Gamma_\lambda (V_\lambda) \}^{-1} [\partial \Gamma_\lambda (V_\lambda)] (z), \quad (C.3)$$

where

$$\partial \Gamma_\lambda (V) (z) = E \left[ \frac{\partial G_\lambda (\bar{u}(U; z) + \beta \bar{V}(U; z, \lambda))}{\partial \lambda} \bar{\omega}(U; z) \right],$$

and

$$\frac{\partial G_\lambda (r)}{\partial \lambda} = \log \left[ \sum_{d \in \mathcal{D}} \exp \left( \frac{r(d)}{\lambda} \right) \right] - \frac{\sum_{d \in \mathcal{D}} \exp \left( \frac{r(d)}{\lambda} \right) r(d)}{\lambda \sum_{d \in \mathcal{D}} \exp \left( \frac{r(d)}{\lambda} \right)}.$$ 

Write

$$G_\lambda (r) = \lambda \log \left[ \sum_{d \in \mathcal{D}} \exp \left( \frac{r(d)}{\lambda} \right) \right] = \max_{d \in \mathcal{D}} r(d) + \lambda \log \left[ \sum_{d \in \mathcal{D}} \exp \left( \frac{\bar{r}(d)}{\lambda} \right) \right],$$

where $\bar{r}(d) = r(d) - \min_{d \in \mathcal{D}} r(d) \leq 0$, $d \in \mathcal{D}$, to obtain

$$\frac{\partial G_\lambda (r)}{\partial \lambda} = \log \left[ \sum_{d \in \mathcal{D}} \exp \left( \frac{\bar{r}(d)}{\lambda} \right) \right] - \frac{\sum_{d \in \mathcal{D}} \exp \left( \frac{\bar{r}(d)}{\lambda} \right) \bar{r}(d) / \lambda}{\sum_{d \in \mathcal{D}} \exp \left( \frac{\bar{r}(d)}{\lambda} \right)}.$$ 

Since $1 \leq \sum_{d \in \mathcal{D}} \exp \left( \frac{\bar{r}(d)}{\lambda} \right) \leq D$ and $-De^{-1} \leq \sum_{d \in \mathcal{D}} \exp \left( \frac{\bar{r}(d)}{\lambda} \right) \bar{r}(d) / \lambda \leq 0$ for all $\lambda > 0$ and all $r \in \mathbb{R}^D$, we conclude that $|\partial G_\lambda (r) / \partial \lambda| \leq \log (D) + De^{-1}$ and so is Lipschitz uniformly in $\lambda \in (0, \bar{\lambda})$. This in turn implies that $\sup_{z, \lambda} \left\| \frac{\partial V_\lambda (z)}{\partial \lambda} \right\| < \infty$. This combined with Assumption 1 implies that $G_\lambda (u(\psi(z, \cdot)) + \beta V_\lambda (\psi_z(z, \cdot))) \mid z \in \mathbb{Z}$ is Lipschitz w.r.t. $(z, \lambda)$. It now follows from Theorem 2.7.11 of van der Vaart and Wellner (1996) that the $\varepsilon$-covering number of $\mathcal{G}$ is of order $\varepsilon^{-v}$ for some $v \geq 1$.

The proof of the bound of $\mathcal{N} (\varepsilon, \mathcal{G}_d, \|\cdot\|_2)$ is analogous except that we now use that $\mathcal{G}_\lambda(d) (r) \in$
(0, 1) so that
\[
\left\{ \left( u, t \right) \mid \hat{G}_{\lambda,d}(u(\psi(z,u)) + \beta V_{\lambda}(\psi_{z}(z,u))) < t \right\} \mid (z, \lambda) \in \mathcal{Z} \times (0, \tilde{\lambda})
\]
\[
= \left\{ \left( u, t \right) \mid \hat{G}^{-1}_{\lambda,d}(t) - u(\psi(z,u)) + \beta V_{\lambda}(\psi_{z}(z,u)) > 0 \right\} \mid (z, \lambda) \in \mathcal{Z} \times (0, \tilde{\lambda})
\]
and it now follows from Lemmas 2.6.15 and 2.6.18(iii) of van der Vaart and Wellner (1996) that the \( \varepsilon \)-covering number of \( \hat{G}_{d} \) is of order \( \varepsilon^{-v} \) for some \( v \geq 1 \). □