CONSTRAINED OPTIMIZATION APPROACHES TO ESTIMATION OF STRUCTURAL MODELS: COMMENT*

FEDOR ISKHAKOV
UNIVERSITY OF NEW SOUTH WALES

JINHYUK LEE
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

JOHN RUST†
GEORGETOWN UNIVERSITY

BERTEL SCHJERNING
UNIVERSITY OF COPENHAGEN

KYOUNGWON SEO
KOREA ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY

JANUARY 2015

Abstract

We revisit the comparison of mathematical programming with equilibrium constraints (MPEC) and nested fixed point (NFXP) algorithms for estimating structural dynamic models by Su and Judd (SJ, 2012). They used an inefficient version of the nested fixed point algorithm that relies on successive approximations. We re-do their comparison using the more efficient version of NFXP proposed by Rust (1987), which combines successive approximations and Newton-Kantorovich iterations to solve the fixed point problem (NFXP-NK). We show that MPEC and NFXP-NK are similar in performance when the sample size is relatively small. However, in problems with larger sample sizes, NFXP-NK outperforms MPEC by a significant margin.

KEYWORDS: Structural estimation, dynamic discrete choice, NFXP, MPEC, successive approximations, Newton-Kantorovich algorithm.

*The main results in this comment were obtained and submitted as two separate papers independently by Lee and Seo, and Iskahkov, Rust and Schjerning. These two papers were then combined together on the request of the editor into the current jointly authored comment. We are grateful to Harry J. Paarsch, Che-Lin Su, Kyoo-il Kim and Daniel Ackerberg for helpful comments. An early version of this paper was presented by Bertel Schjerning at the ZICE2014 workshop at the University of Zurich. We are grateful to participants of the ZICE2014 workshop for helpful feedback that lead to the current draft of this comment. This paper is part of the IRUC research project financed by the Danish Council for Strategic Research (DSF). Financial support is gratefully acknowledged.

†Corresponding author, email: jr1393@georgetown.edu.

1 Introduction

In "Constrained optimization approaches to estimation of structural models" Su and Judd (2012), hereafter SJ, proposed a constrained optimization approach for maximum likelihood estimation of infinite horizon dynamic discrete choice models — mathematical programming with equilibrium constraints (MPEC). They argued that MPEC is superior to the *nested fixed point* (NFXP) algorithm proposed by Rust (1987). NFXP uses the fact that the likelihood depends on the parameters via the *value function* to a dynamic programming (DP) problem. Under weak conditions, the value function is the unique fixed point to a contraction mapping defined by the *Bellman equation* to the DP problem and is a smooth implicit function of the underlying structural parameters of the problem. NFXP uses this to maximize the likelihood using standard *unconstrained* quasi-Newton optimization algorithms, except that each time the likelihood is evaluated, NFXP calls a fixed point subroutine to compute the value function corresponding to the current parameter values.

In contrast, MPEC method does not need a specialized inner loop algorithm to compute the fixed point. Instead, it recasts the problem of maximizing the likelihood function as a constrained optimization problem with respect to the K structural parameters plus N additional variables, which are the values of value function at a set of N grid points in the state space. These N additional variables must satisfy the Bellman equation, which can be recast as a N "side constraints". Thus MPEC also implicitly solves the fixed point problem while searching for structural parameter values that maximize the likelihood, but using a general-purpose *constrained* optimization algorithm[1].

SJ used the model of optimal replacement of bus engines of Rust (1987) to conduct a Monte Carlo study to compare the performance of the MPEC and NFXP algorithms. They found that MPEC outperformed NFXP in terms of CPU time by up to three orders of magnitude.

The point of this comment is to note that SJ used an inefficient version of the NFXP algorithm that results in misleading conclusions about the relative speed of MPEC and

[1]The specific implementation SJ use is KNITRO (see Byrd, Nocedal and Waltz 2006) which is run under AMPL software that provides analytic first and second order derivatives, and under Matlab with analytic gradients and the Hessian approximated numerically.

NFXP. SJ used a version of NFXP we refer to as "NFXP-SA" where the method of *successive approximations* (SA) is used to solve the inner fixed point problem. However it is well known that successive approximations is an inefficient algorithm for computing fixed points of contraction mappings, especially when the modulus of the contraction (which equals the discount factor in the underlying dynamic programming problem) is close to 1.

We redo the SJ Monte Carlo study using the more efficient version of NFXP that Rust (1987) employed. This version, NFXP-NK, combines successive approximations and Newton-Kantorovich (NK) iterations to solve the inner fixed point problem significantly faster and more reliably, especially when the discount factor is close to 1. We show that NFXP-NK and MPEC are roughly equivalent in their numerical performance when the sample size is relatively small, but NFXP-NK is significantly faster in larger datasets.

NK is the preferred method for computing contraction fixed points because it has guaranteed quadratic convergence rate in its domain of attraction. However, NK is only locally convergent, so the original design of the NFXP algorithm (Rust 1987, 2000) starts with successive approximations to ensure global convergence, and switches to NK iterations only after it detects that the domain of attraction has been reached.[2] This hybrid algorithm or "polyalgorithm" ensures that a highly accurate solution can be found after only a small combined number of iterations. In particular, the combination of these two approaches makes the performance of the NFXP-NK algorithm independent of the value of the discount factor β whereas the CPU times of the NFXP-SA algorithm steadily increase as $\beta \to 1$ as shown in Table 2 of SJ (p. 2228).

In the next section we redo the original comparison of MPEC and NFXP of SJ using NFXP-NK instead of NFXP-SA. We also demonstrate the effect of sample size on the relative performance of the two methods. Though we do not claim that the conclusions from this limited numerical comparison of the two methods in the bus engine replacement example necessarily applies to other problems, in the last section we discuss our own view

[2]Rust (2000, p.28) details the theory of when the algorithm has to switch from successive approximations to NK iterations. The idea is based on the fact that the ratio of tolerances in two successive iterations of SA algorithm approaches the modulus of contraction (β) as the algorithm progresses. When this ratio is close enough to β, the shape of the value function is nearly recovered (one can show that this ratio equals β exactly when the current iteration differs from the fixed point by a constant λ), which is a signification that the domain of attraction of NK method has been reached (λ will be "stripped away" in just a single iteration).

of the overall strengths and weaknesses of the two estimation approaches.

2 Results

Table 1 presents a comparison of CPU times for MPEC and NFXP that reproduces Table 2 of SJ. While we have developed our own code to implement NFXP-NK, we have been using the same setup and computer code for MPEC that was used to produce the results of the original paper by SJ.[3] We have replicated SJ's results for NFXP-SA, but since this version of NFXP is inefficient and is completely dominated by NFXP-NK, we only report results for the latter in Table 1 below. Similarly, we skip the results regarding MPEC-Maltab implementation which uses first order analytical derivatives only, because its run times were about two orders of magnitude larger that those of MPEC-AMPL in all our experiments.[4,5] To conserve space we also omit the table of structural estimates (Table 1 of SJ), which shows that conditional on convergence both methods were able to recover the structural parameters of the model. It is also the case in all our experiments.

It is clear from Table 1 that when the fast NFXP-NK algorithm is used, the CPU times are comparable to those of MPEC-AMPL, the implementation utilizing first and second order analytic derivatives of the objective function. It takes both methods about 0.07 to 0.08 of a second to structurally estimate the bus engine replacement model.

As the second panel of Table 1 shows, NFXP-NK uses far fewer successive approximation steps compared to the NFXP-SA results in Table 2 of SJ, and this is the main reason

[3] SJ have kindly provided well documented MPEC code for this problem via Che-Lin Su's website. We are thankful to Che-Lin Su for pointing out numerical instability of the original code due to the lack of recentering of the value functions in accordance with Rust (2000, p. 27). Recentering has been implemented in the numerically stable version of MPEC code we use here. Our NFXP code is available on request.

[4] We used the same specification of the bus engine replacement model, including the same true parameter values, same sample size and fixed point dimension, and the same number of Monte Carlo replications as in Table 2 of SJ. We fixed the stopping tolerance for the inner fixed point in the NFXP-NK algorithm at 10^{-13} as in Rust (1987), which is $1/1000$ of the stopping tolerance 10^{-10} that SJ used for NFXP-SA. Similarly to SJ, we estimated transition probabilities for mileage travelled by buses jointly with other parameters (replacement cost and maintenance cost) by maximizing the full likelihood function, following the partial likelihood optimization as described in Rust (1987, 2000). We used the BHHH algorithm on the outer loop of NFXP-NK and frequency based starting values for the transition probability parameters for both methods. For the polyalgorithm we use a minimum of $minstp = 2$ and maximum $maxstp = 20$ successive approximation iterations and a relative tolerance to switch from SA to NK iteration $rtol = 0.02$, see Rust (2000, p.28).

[5] Our hardware and software setup was the following: Mac Book Pro with 2.3 GHz Intel Core i7 processor and 8 GB of memory with OS X version 10.9.5, Matlab 2014a constrained to a single core with *-singleCompThread* startup option, AMPL Version 20131213 and KNITRO 9.0.1.

Table 1: MPEC versus NFXP-NK: sample size 6,000

β	Converged (out of 1250)	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Bellm. Iter.	# of N-K Iter.
			MPEC-AMPL			
0.975	1246	0.054	9.3	12.1		
0.985	1217	0.078	16.1	44.1		
0.995	1206	0.080	17.4	49.3		
0.999	1248	0.055	9.9	12.6		
0.9995	1250	0.056	9.9	11.2		
0.9999	1249	0.060	11.1	13.1		
			NFXP-NK			
0.975	1250	0.068	11.4	13.9	155.7	51.3
0.985	1250	0.066	10.5	12.9	146.7	50.9
0.995	1250	0.069	9.9	12.6	145.5	55.1
0.999	1250	0.069	9.4	12.5	141.9	57.1
0.9995	1250	0.078	9.4	12.5	142.6	57.5
0.9999	1250	0.070	9.4	12.6	142.4	57.7

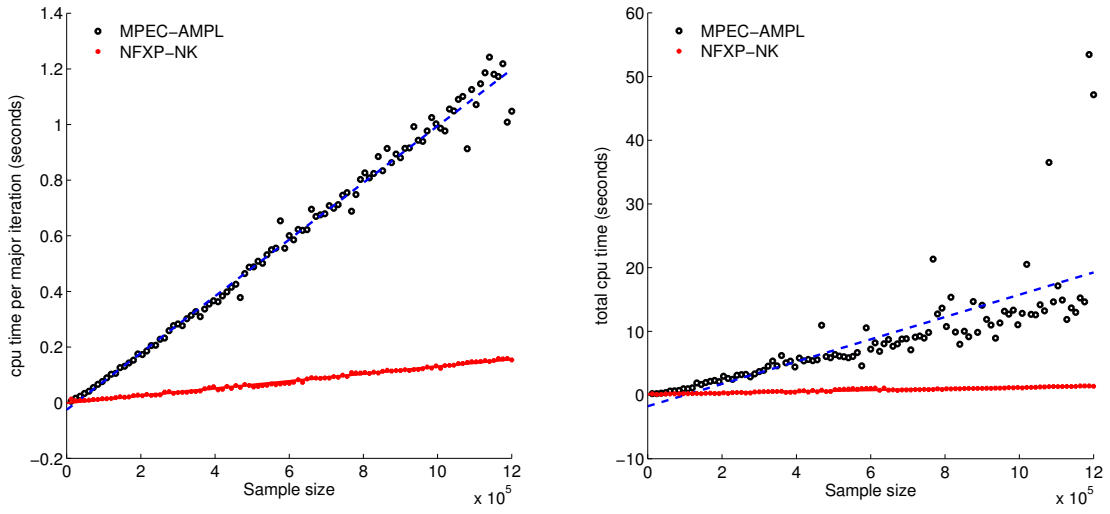
Notes: This table is a replication of Table 2 in Su and Judd (2012) with NFXP-SA replaced by NFXP-NK (section on MPEC-Matlab is skipped to conserve space). For each β, five starting points were used for each of the 250 simulated samples. CPU time, number of major iterations, number of function evaluations and number of inner loop iterations are the averages over the convergent runs. Inner loop iterations include both value function iterations and Newton-Kantorovich iterations.

why NFXP-NK is so much faster. For the highest discount factor $\beta = 0.995$ in Table 2 of SJ, an average of nearly 750,000 successive approximation iterations were required, compared to just 145 for NFXP-NK. With an average of 55 NK iterations per estimation, the average number of both types of inner loop iterations for NFXP-NK is remarkably insensitive to the discount factor. On average, it takes NFXP-NK only 12 successive approximation steps and 4 NK iterations per function evaluation to compute a highly accurate fixed point (to a tolerance of 10^{-13}) when $\beta \geq 0.9995$.

Our findings are thus completely contrary to the main conclusion of SJ (2012) that "MPEC is significantly faster than NFXP" (p. 2228). We believe that their conclusion is an artifact of their use of the inefficient NFXP-SA algorithm. We also find NFXP-NK method more robust with successful convergence in *every* case for the range of discount factors we used, whereas MPEC experienced a few cases of convergence failure.

There is a fundamental difference between how the two methods approach likelihood

Figure 1: CPU times for MPEC-AMPL and NFXP-NK as a function of sample size



Notes: Average CPU time is computed based on converging runs only. If we let x denote the sample size divided by 1.000.000, the plotted linear regressions for CPU time per iteration (left panel) are $T_{NFXP} = 0.001 + 0.13x$ ($R^2 = 0.991$) and $T_{MPEC} = -0.025 + 1.02x$ ($R^2 = 0.988$). The plotted linear regressions for total CPU time (right panel) are $T_{NFXP} = 0.129 + 1.07x$ ($R^2 = 0.926$) and $T_{MPEC} = -1.760 + 17.51x$ ($R^2 = 0.554$). For experimental design see note in Table 1, the only difference is varying sample size.

computation. In the case of NFXP the choice probabilities entering the likelihood function are computed independently of the data in the inner loop. For MPEC both the fixed point calculation and the maximization of the likelihood are done simultaneously, which not only implies that the gradient vector and the Hessian matrix are both high dimensional objects, but also that the whole data set needs to be processed multiple times when computing non-zero elements of these objects. Therefore, compared to NFXP, MPEC will be more computationally expensive in larger datasets.

To study the effect of a larger sample size on the relative run times for MPEC-AMPL and NFXP-NK we conducted an additional Monte Carlo experiment, where we kept the fixed point dimension the same ($N = 175$) but increased the sample size[6]. Figure 1 shows how the performance of the two algorithms depend on the sample size. The run time of both

[6]To increase the size of the sample we changed the number of buses and kept the number of periods fixed at 120 months.

MPEC-AMPL and NFXP-NK appear to be linear in the number of data points, but it is evident that the relative speed advantage of NFXP-NK over MPEC-AMPL monotonically increases up to an order of magnitude as the sample size grows up to millions of observations (which is not uncommon in the modern day register-based governmental datasets). Note also that while runtime is almost perfectly linear for NFXP, the total runtime varies much more for MPEC – due to varying number of iterations.

In addition to the desirable large sample performance, the separation between solving the model and computing the likelihood, enables NFXP method to use traditional unconstrained quasi-Newton/gradient search algorithms for likelihood maximization – such as the Berndt-Hall-Hall-Hausman (BHHH) algorithm (Berndt et al., 1974) – over a relatively small number of structural parameters. Unlike MPEC, NFXP recognizes the fact that the objective function is a sum of individual likelihoods each of which is computed from the set of value functions that are smooth in the structural parameters. BHHH algorithm exploits the information identity to approximate the Hessian of the likelihood with the negative of the outer product of the scores. Therefore, because the Hessian approximation is always negative semi-definite, BHHH always moves in the direction of the gradient (i.e. towards the maximum), even in convex areas of the likelihood function. Hence, beyond the advantage of avoiding computation of second order derivatives, BHHH has the major advantage of always moving uphill for small enough step size, and thus being globally convergent.

To sum up, the robustness of NFXP and its ability to handle larger problems comes from explicitly exploiting the structure of the problem and recognizing that the objective function is a sample sum over individual likelihoods.

3 Conclusion

We have shown that the main conclusion of SJ (2012), namely that "Monte Carlo results confirmed that MPEC is significantly faster than NFXP, particularly when the discount factor in the dynamic-programming model is close to 1." (p. 2228), appears to be an artifact of their use of an inefficient version of NFXP which we called NFXP-SA. We reran the comparison using the version of NFXP that Rust (1987) originally proposed, NFXP-

NK, and found that MPEC-AMPL and NFXP-NK are about equally fast on the same test problem that SJ considered. We also compared MPEC and NFXP for a larger range of discount factors β, and different sample sizes. We have shown that relative performance of NFXP-NK is independent of the discount factor. Further, we found that MPEC may be significantly slower than NFXP for problems with large samples.

Yet, we believe that MPEC has many desirable features, the most important of which is ease of use by people who are not interested in devoting time to the special-purpose programming necessary to implement NFXP-NK. As long as the problem to be estimated is not too big in terms of the number of observations, our results indicate that MPEC is very fast and competitive with NFXP-NK, and particularly in conjunction with the easy, intuitive AMPL language, it could save many users substantial programming time and enable them to structurally estimate many models of interest.

However for micro- or high frequency data applications with many observations, our results indicate that the MPEC method may not be as viable, at least until better implementations are available. It is also evident that high dimensionality of the constrained optimization problem solved by MPEC, which includes not only structural parameters, but also value functions computed on the grid over the state space, imposes certain restrictions on the hardware, for example on the size of available memory. Therefore, in problems with finite horizon where the value functions are time specific, the use of MPEC may be completely infeasible.[7] In these cases there may be no alternative but to incur the fixed costs of programming the special-purpose fixed point algorithm that can exploit the special structure of the dynamic programming problem at hand.

References

[1] Berndt, E., B. Hall, R. Hal, and J. Hausman (1974): "Estimation and Inference in Nonlinear Structural Models," *Annals of Economic and Social Measurement*, 3: 653-665.

[7] Other papers elaborating on the performance of MPEC include (Lee and Seo, 2014, 2105) and (Jørgensen, 2013).

[2] Byrd, R. H., J. Nocedal, and R. A. Waltz (2006): "KNITRO: An Integrated Package for Non- linear Optimization," in it Large-Scale Nonlinear Optimization/ ed. by G. di Pillo and M. Roma. New York: Springer, 35-59.

[3] Jørgensen, T. H. (2013): "Structural Estimation of Continuous Choice Models: Evaluating the EGM and MPEC," *Economics Letters*, 119, pp. 287-290.

[4] Lee, J. and K. Seo (2015): "Nested Fixed Point Algorithm Revisited in BLP Random Coefficients Demand Estimation," *unpublished manuscript*.

[5] Lee, J. and K. Seo (2014): "A Computationally Fast Estimator for Random Coefficients Logit Demand Models Using Aggregate Data," *RAND Journal of Economics*, forthcoming.

[6] Su, C.-L. and K. L. Judd (2012): "Constrained Optimization Approaches to Estimation of Structural Models," *Econometrica*, 80 (5), 2213-2230.

[7] Rust, J. (1987): "Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher," *Econometrica*, 55 (5), 999-1033.

[8] Rust, J. (2000): "Nested Fixed Point Algorithm Documentation Manual: Version 6," `https://editorialexpress.com/jrust/nfxp.html`