Calibrated Projection in MATLAB: User's Manual*

Hiroaki Kaido† Francesca Molinari‡ Jörg Stoye§ Matthew Thirkettle¶

March 8, 2019

Abstract

We present the calibrated-projection MATLAB package implementing the method to construct confidence intervals proposed by Kaido, Molinari, and Stoye (2019). This manual provides details on how to use the package for inference on projections of partially identified parameters and instructions on how to replicate the empirical application and simulation results in the paper. The version of this code included in this ZIP file is what was used to carry out the empirical application in Section 4 of Kaido et al. (2019) and the Monte Carlo simulations in Appendix C. Please visit `https://molinari.economics.cornell.edu/programs.html` for the most up-to-date version of the code.

Keywords: Partial identification; Inference on projections; Moment inequalities; Uniform inference.

*We gratefully acknowledge financial support through NSF grants SES-1230071, SES-1357643 and SES-1824344 (Kaido), SES-0922330 and SES-1824375 (Molinari), and SES-1260980 and SES-1824375 (Stoye).

†Department of Economics, Boston University, hkaido@bu.edu.

‡Department of Economics, Cornell University, fm72@cornell.edu.

§Departments of Economics, Cornell University stoye@cornell.edu.

¶Department of Economics, Cornell University, mkt68@cornell.edu.

1 Introduction

This manual details the structure of the Calibrated Projection Interval (CPI) algorithm and `MATLAB` Package. It accompanies the paper "Confidence Intervals for Projections of Partially Identified Parameters" (Kaido et al., 2019) and it assumes familiarity with that paper.[1] The CPI algorithm uses an EAM (evaluate, approximate, maximize) algorithm to solve:

$$\begin{aligned} &\inf / \sup_{\theta \in \Theta} p'\theta \\ &\text{s.t.} \sqrt{n} \frac{\bar{m}_j(\theta)}{\hat{\sigma}_j(\theta)} \leq \hat{c}(\theta) \quad j = 1, \cdots, J, \end{aligned}$$

where $\hat{c}(\theta)$ is the calibrated critical value (Jones, Schonlau, & Welch, 1998; Jones, 2001). This version of the CPI algorithm is optimized for basis projection $p = (0, \cdots, 0, 1, 0, \cdots, 0)$ with hyperrectangle parameter constraints $\Theta = \{\theta \in \mathbb{R}^d : \theta_{LB} \leq \theta \leq \theta_{UB}\}$. We also allow for p to be in the unit sphere and polytope constraints on the parameter space, so that $\Theta = \{\theta \in \mathbb{R}^d : \theta_{LB} \leq \theta \leq \theta_{UB}, A_\theta \theta \leq b_\theta\}$. Additional care is required within these extensions (see Appendix B for further details). The current version of the package is written for moment (in)equalities that are separable in data W and parameter θ, so that $E_P[m_j(W_i, \theta)] = E_P[f_j(W_i)] + g_j(\theta)$.[2] Future releases of the package will include:

- Non-separability of $E_P[m_j(W_i, \theta)]$ in W_i and θ.
- Objective function $h(\theta)$ not necessarily equal to $p'\theta$.

We have structured the code so that it is portable. In order to implement a user-specified model, the user needs only input the data, algorithm options, the function that defines the estimators for the moment (in)equalities, as well as the gradients and standard deviation

[1] Some notation differs between this paper and (Kaido et al., 2019). This is made clear throughout this manual. Unless otherwise specified, we use notation from the earlier version of the paper (Kaido, Molinari, & Stoye, 2016). The table numbering references (Kaido et al., 2019).

[2] In this manual and in the CPI MATLAB package data is defined as W. The function f and g are the two components of the separable moment (in)equality $E_P[m_j(W_i, \theta)]$. This is in contrast to Kaido et al. (2019), where data is X, $f(\theta)$ refers to the objective function, and $\bar{g}(\theta)$ appears in the EAM algorithm. The subscript n has also been dropped from all estimators.

estimators of the moment functions. Section 2 provides an overview of the key files in the package and explains how to set up numerical solvers. Section 3 provides instructions on how to replicate the simulations and the empirical application in Kaido et al. (2019).

2 Using the Calibrated Projection Interval Algorithm

This section is organized as follows. Section 2.1 briefly describes the key files in the package. Section 2.2 details how to set up `CVXGEN` and `CVX`, both are fast disciplined convex solvers that we use to compute the calibrated critical value $\hat{c}(\theta)$ (Mattingley & Boyd, 2012; Grant & Boyd, 2014, 2008).

2.1 Overview of Important Files and Folders

First, we briefly describe the key `MATLAB` files and folders.

- `KMS_Simulation.m`. This executes the simulations in Kaido et al. (2019). The DGP, method (Calibrated Projection, Andrew and Soares (AS), or Bugni, Canay, and Shi (BCS)-Profiling),[3] nominal significance level, projection directional vector, number of observations, and number of simulations are set by the user here. The data is generated and passed to either `KMS_0_Main.m` or `BCS_Main`, which computes the Calibrated or AS Projection Interval, or the BCS-Profiled Interval, respectively.

- `KMS_0_Main.m`. This is the file that the user calls to execute the CPI algorithm and compute the Projection Interval (either Calibrated or AS). The user specifies data `W`, the initial guess for a feasible parameter `theta_0`, the projection direction `p`, a set of pre-specified feasible points `theta_feas`, the lower bound on parameter space `LB_theta`, the upper bound on parameter space `UB_theta`, the polytope constraints on the parameter space `A_theta` and `b_theta` so that $A_\theta \theta \leq b_\theta$, the nominal significance

[3]The code implementing BCS is the code provided by these authors and is available at `http://qeconomics.org/ojs/index.php/qe/article/view/431`.

level `alpha`, a one-sided or two-sided confidence interval `type`, the projection method (calibrated or AS) `CI_method`, the GMS tuning parameter `kappa`, the GMS function `phi`, the name of the `MEX` files for `CVXGEN` (discussed in Section 2.2 below) `CVXGEN_name`, and a structure of algorithm options `KMSoptions`.

The package assumes that the moment (in)equalities are separable, so that $E_P[m_j(W_i, \theta)] = E_P[f_j(W_i)] + g_j(\theta)$.

- `moments_w.m` is the user-specified function for the estimator of $E_P[f_j(W_i)]$, namely $\hat{f}_j$. We allow for both moment inequalities and equalities, as well as paired moment inequalities. If $f_j(W_i)$ is a Bernoulli random variable and if its expectation is too close to 0 or 1, then the corresponding moment (in)equalities are dropped. The output `f_ineq_keep` and `f_eq_keep` defines the moment (in)equalities that are not discarded.

- `moments_theta.m` is the user-specified function for $g_j(\theta)$.

- `moments_gradient.m` is the user-specified function for the gradient of $g_j(\theta)$, which is denoted $D_\theta g_j(\theta)$.

- `moments_stdev.m` is the user-specified function for the estimator for the standard deviation $\sigma_j(W_i)$.

- `KMSoptions.m` defines a structure of algorithm options. `KMSoptions` is also passed to the four user-specified functions above, so the user can pass additional parameters through `KMSoptions` to the user-specified functions (e.g., the support for data W_i). The function `KMSoptions.m` is called before running `KMS_0_Main.m`, and is passed through the last argument of `KMS_0_Main.m`, which is `KMSoptions`.

- `Rho_Polytope_Box.m` and `bound_transform.m` are additional user-written functions needed when polytope constraints on the parameter space are provided (see the arguments `A_theta` and `b_theta` in `KMS_0_Main.m`) or when p is not a basis vector. If p is a

non-basis vector or if polytope constraints on the parameter space are included, then sensitivity in the estimate for the projection interval can arise.

The disciplined convex solver `CVXGEN` is used to check whether the set

$$\Lambda^b(\theta, \rho, c) = \{\lambda \in \sqrt{n}(\Theta - \theta) \cap \rho B^d : \mathbb{G}_j^b + D_\theta g_j(\theta)\lambda + \varphi_j(\hat{\xi}_j(\theta)) \leq c, j = 1, \cdots, J\}$$

is empty for each bootstrap repetition $b = 1, \cdots, B$. In order to run `CVXGEN`, the user first compiles a `MEX` file that defines the parameters of the problem (details in Section 2.2).

- The compiled `MEX` files are stored in the subfolder `\CVXGEN`. The file name for this is chosen by the user. For example, we choose `csolve_DGP8.mex64` for the BCS Entry Game. The file name must also be defined when `KMS_0_Main.m` is called. The name is passed via the argument `CVXGEN_name`.

2.2 CVXGEN and CVX Setup

The calibrated critical value $\hat{c}(\theta)$ is computed using a fixed-point algorithm. The fixed-point mapping is computed by checking whether the following set is empty:

$$\Lambda^b(\theta, \rho, c) = \{\lambda \in \sqrt{n}(\Theta - \theta) \cap \rho B^d : \mathbb{G}_j^b(\theta) + D_\theta g_j(\theta)\lambda + \varphi_j(\hat{\xi}_j(\theta)) \leq c, j = 1, \cdots, J\}. \quad (1)$$

This amounts to solving many linear programs (LP), which is done using the fast disciplined convex solver `CVXGEN` (Mattingley & Boyd, 2012) or `CVX` (Grant & Boyd, 2014, 2008).

2.2.1 CVXGEN Setup

To set up `CVXGEN`, the user needs to: 1) install a `MEX` Compiler; 2) generate `C` code at `https://cvxgen.com`; 3) compile and save the `MEX` file; 4) Instruct the CPI algorithm to use `CVXGEN` rather than `CVX`.

The first step is to install a `MEX` compiler. We use the `MinGW-w64 Compiler` on a Windows machine, which is an add-on in `MATLAB`. To install: open `MATLAB`, go to `Home` tab, go to `Add-Ons`. An add-on search window appears on the screen. Search `MinGW-w64 Compiler` and install `MATLAB Support for MinGW-w64 C/C++ Compiler v`. On a Mac, a `C` compiler is supplied with `Xcode`. On a Linux based system, one can use `GCC` (GNU Compiler Collection).

The next step is to generate the `C` code for a specific problem. First, create an account at `https://cvxgen.com` and log in. Next, navigate to the `edit` tab under `problem`. Copy-and-paste the following:

```
dimensions
    dim_p      = XX
    J1         = YY
    J2         = ZZ
    S          = VV
end
parameters
    A    (J1 + 2*J2 + 4*dim_p + 2 + S , dim_p)
    b    (J1 + 2*J2 + 4*dim_p + 2 + S , 1)
end
variables
    x(dim_p,1)
end
minimize
    0
subject to
    A*x<= b
end
```

Replace `XX` with the dimension of the parameter θ, `YY` with the number of moment inequalities, `ZZ` with the number of moment equalities (do not double count $E_P[m_j(W_i, \theta)] \leq 0$ and $-E_P[m_j(W_i, \theta)] \leq 0$ here), and `VV` with the number of polytope box constraints. If no

polytope constraints $A_\theta\theta \leq b_\theta$ are included, set `VV`$= 0$.

Next, navigate to the `generate C` tab under `CODEGEN`. Click `Generate code`. As a result, a list of files populate the webpage. Download the `cvxgen.zip` file and extract. Run `make_csolve.m`. The file `csolve.mex64` should appear in the folder (if on a Linux or Mac machine, the extension is slightly different).[4] Rename `csolve.mex64` to `CVXGEN_name.mex64` (where `CVXGEN_name` is specified by the user) and move the file to the subfolder \`CVXGEN`.

Last, set `KMSoptions.CVXGEN = 1` to instruct CPI algorithm to use `CVXGEN`.

There is an upper bound of $4,000$ non-zero Karush-Kuhn-Tucker matrix entries for the linear program in `CVXGEN`. The size of the problem is determined jointly by J_1, J_2, and d. As an example, `CVXGEN` can handle $\theta \in \mathbb{R}^{10}$ with $J_1 = 55$ and $J_2 = 55$.

2.2.2 CVX Setup

An alternative solver to `CVXGEN` is `CVX`. This solver is slower than `CVXGEN`, but can handle significantly larger LPs and, in our experience, is significantly faster than `MATLAB`'s LP solver `LINPROG`. `CVX` is a `MATLAB` "wrapper" for five different disciplined convex solvers (Grant & Boyd, 2014, 2008). Among these, the solver `MOSEK` is the fastest for our problem. To run `CVX` with `MOSEK`:

1. Ensure that there is a copy of `CVX` is located in the subfolder \`CVX`. If not, navigate to `http://cvxr.com/cvx/` and deposit a copy in the subfolder \`CVX`.

2. Request a license from `http://cvxr.com/cvx/` and deposit it in the same folder.

3. Run `cvx_setup.m`.

4. Set solver using the command `cvx_solver MOSEK` in the `MATLAB` command window.

5. Set `KMSoptions.CVXGEN = 0`.

6. Set `CVXGEN_name` to the empty set.

[4] If an error occurs here, it is likely that the `MEX` compiler is not installed correctly.

Once CVXGEN or CVX is set up, either a simulation model (Section 3.1) or a user-specified model can be called via the CPI algorithm.[5]

3 Simulations & Empirical Application

In this section we discuss how to replicate the empirical application and simulation results on the calibrated projection confidence intervals reported in Kaido et al. (2019) (see Tables 1-3 in the paper). Section 3.1 provides instructions on how to replicate the simulations to reproduce Tables 2-3 in Kaido et al. (2019). Section 3.3 explains how to replicate the empirical application to reproduce Table 1.

3.1 Running Simulations: Calibrated Projection CIs

As per CVXGEN policy, we are unable to distribute the MEX files for these simulations. So the first step is to generate the relevant MEX files, see Section 2.2 for instructions and Table 1 for CVXGEN parameters and naming conventions.

The next step is to set parameters in KMS_Simulation.m. Open an instance of KMS_Simulation.m and set the following:

- `method = 'KMS'` to compute the Calibrated Projection Interval; or `method = 'AS'` to compute the AS Projection Interval.

- `DGP=k` where `k`$\in \{1, 2, 3, 4, 5, 6, 7, 8\}$. This parameter selects the data-generating process. $k = 8$ is the Entry Game used to evaluate the performance of the calibrated projection CIs (Tables 2-3 in Kaido et al. (2019)).[6]

[5] For additional help with CVXGEN or CVX, please visit `https://cvxgen.com` and `http://cvxr.com/cvx/`.

[6] $k = 1 - 4$ corresponds to the rotated box described in the earlier version Kaido et al. (2016). $k = 5 - 7$ corresponds to the Entry Games described in another earlier version (Kaido, Molinari, & Stoye, 2017). $k = 5$ is the point-identified Entry Game with zero correlation (Table 3); $k = 6$ is the partially-identified Entry Game with zero correlation (Tables 4, 6 and 7); $k = 7$ is the partially-identified Entry Game with $Corr(u_1, u_2) = 0.5$ (Table 5).

- `KMS=1` or `KMS=0` determines if `KMS_0_Main` or `BCS_Main` is called. `KMS=0` is a valid input only if `DGP=8`, and `component=1` or `component=2`.

- `component=k` where `k`$\in \{1, \cdots, $`dim_p`$\}$ selects the projection direction. That is, the projection vector is p with $p_i = 1$ if $i = k$ and $p_i = 0$ otherwise.

- `n` is the sample size. `n` is set to 4000 for Tables 2-3.

- `Nmc` is the number of Monte Carlo simulations requested. `Nmc` is set to 300 in Table 2 and 1000 in Table 3.

- `sim_lo` and `sim_hi` determine which simulations are run. These parameters are used to split the simulations into batches if needed.

Among other things, convergence criteria are set in `KMSoptions_Simulation`. All DGPs use what we call the baseline options. The baseline options are:

- `KMSoptions.EAM_maxit=20`. This sets the maximum number of EAM iterations to 20.

- `KMSoptions.h_rate=1.8`. This determines the contraction rate of the parameter space for the M-step.

- `KMSoptions.h_rate2=1.25`. This determines the contraction rate of the parameter space for additional points

- `KMSoptions.EAM_obj_tol = 0.005`. One requirement for convergence is that the absolute difference between the expected improvement projection and the current feasible point $\theta^{*,L}$ is less than `EAM_obj_tol`.

- `KMSoptions.EI_points=10`. The M step is initialized with a set of starting points. The algorithm selects `EI_points` points around the current feasible point $\theta^{*,L}$ that have positive expected improvement. Additional points are also selected.

The number of bootstrap repetitions is also set in `KMSoptions_Simulation.m`. Table 2 sets this number equal to 301, so that `KMSoptions.B=301`. For Table 3, set `KMSoptions.B=1001`.

Finally, run `KMS_Simulation` to run a simulation with the parameters and options specified above. The results are saved in the subfolder `\Results`.

The file `Analysis.m` carries out post analysis for a particular set of simulations. To run the post analysis, load a results file and run `Analysis.m`. The output includes the median lower bound for the Calibrated Projection Interval; the median upper bound for the Calibrated Projection Interval; coverage percent at the end points of the identification region, as well as at the true parameter; average $\hat{c}(\theta)$; and average computational time.

3.2 Running Simulations: BCS-Profiling CIs

The BCS-profiling CIs can be calculated with or without the EAM algorithm. To calculate them without the EAM algorithm, set `KMS=0` in `KMS_Simulation.m` and run simulations as described in the previous section.

To calculate them with the EAM algorihm, the next step is to set the following parameters in `BCS_Simulation.m`.

- `method = 'KMS'` (not AS) to compute the BCS-profiling CI with the EAM algorithm;
- `DGP=8` to generate data from the Entry Game (Tables 2-3 in Kaido et al. (2019)).
- `KMS=0` to run the BCS simulations.
- `component=k` where `k`$\in \{1, 2\}$ selects the projection direction. When `KMS=0` is used, `k=1` and `k=2` are the only valid options for the projection direction.

The remaining parameters (`n,Nmc,sim_lo,sim_hi`) are the same as the ones in the previous section. The parameters for convergence criteria follow the baseline options and are set in `KMSoptions_BCS_Simulation.m`. Once these parameters are set, run `BCS_Simulation.m` to conduct simulation experiments. The file `Analysis_BCS.m` is an analog of `Analysis.m`, which carries out post analysis for a particular set of BCS simulations.

3.3 Replicating the Empirical Application

The empirical exercise applies the CPI algorithm to the airline application in Kline and Tamer (2016). The main replication file is `KMS_Application.m` and the set of options are located in `KMSoptions_Application.m`. To create the `MEX` files, set `J1=J2=16`, `dim_p=9`, and `S=0`.

In addition to increasing the maximum number of allowable iterations to 200, there are six additional options/modifications imposed that require further discussion:

1. Set `KMSoptions_app.FeasAll = 1`
2. Set `KMSoptions_app.direct_solve = 1`
3. Set the GMS function equal to $\phi(x) = \min(x, 0)$.
4. Set the upper bound on the correlation parameter to 0.85.
5. Set `KMSoptions_app.boundary = 0`.
6. Set `KMSoptions_app.CVX_resid_tol = []`.

The option `KMSoptions_app.FeasAll = 1` initiates the CPI algorithm in parallel from each feasible point obtained in the feasible search rather than from only the feasible point that maximizes $q'\theta$ for $q \in \{-p, p\}$. Each of the solutions as well as a subset of feasible points obtained from the CPI algorithm are then passed as starting points to the vanilla `fmincon` algorithm to fine tune the solution. This is accomplished by setting `KMSoptions_app.direct_solve = 1`. We find that the CPI algorithm and the vanilla `fmincon` algorithm complement each other in finding the solution to this relatively difficult DGP. In particular, feeding `fmincon` a randomly chosen starting point returns an infeasible solution, so that a standard application of the `multistart-fmincon` algorithm does not suffice. In contrast, passing `fmincon` the solution to the CPI algorithm yields an improvement in the objective value and completes in a reasonable amount of time.

We specify the GMS function to be the continuous function $\min(x, 0)$ rather than the discontinuous "hard thresholding" function. This is both theoretically appealing and seems to be the better choice for this application. Using "hard thresholding", we find that the CPI algorithm returns solutions that strictly satisfy the constraint $\sqrt{n}\frac{\bar{m}_j(\theta)}{\hat{\sigma}_j(\theta)} \leq \hat{c}(\theta)$ for all j, thus signally that the algorithm terminated early. The continuous GMS function does not experience this problem.

The gradient of the moment function, $D_\theta g_j(\theta)$, is a necessary ingredient for computing the calibrated critical value. The gradient is not well-defined when the correlation parameter is close to one. To avoid numerical issues we restrict the parameter space so that $\theta_9 \in [0, 0.85]$.

A final potential issue is that the identified region in the application includes values of θ on the boundary. That is, there exists $\theta \in \Theta_I(P)$ such that $\theta_9 = 1$ where θ_9 is the correlation parameter. This is accounted for when computing the calibrated critical value $\hat{c}(\theta)$ by specifying additional constraints that recenter the parameter space, see Lines 124-129 in `KMS_33_Coverage.m`. To turn these constraints on, set `KMSoptions_app.boundary = 1` and `KMSoptions_app.CVX_resid_tol = 1e-4`. The reported results in Kaido et al. (2019) set `KMSoptions_app.boundary = 0` and `KMSoptions_app.CVX_resid_tol = []` (i.e., the CVX residual tolerance is set to default settings). We do not find that CPI is sensitive to whether the boundary constraints are imposed. We do, however, find that the boundary constraints generate numerical issues in the CVXGEN program, which are overcome by setting the CVX residual tolerance to a larger value (e.g., 10^{-4}).

References

Grant, M., & Boyd, S. (2008). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, & H. Kimura (Eds.), *Recent advances in learning and control* (pp. 95–110). Springer-Verlag Limited. (`http://stanford.edu/~boyd/graph_dcp.html`)

Grant, M., & Boyd, S. (2014, March). *CVX: Matlab software for disciplined convex programming, version 2.1.* `http://cvxr.com/cvx`.

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, *21*(4), 345–383.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, *13*(4), 455–492.

Kaido, H., Molinari, F., & Stoye, J. (2016). Confidence intervals for projections of partially identified parameters. *arXiv preprint arXiv:1601.00934v1*.

Kaido, H., Molinari, F., & Stoye, J. (2017). Confidence intervals for projections of partially identified parameters. *arXiv preprint arXiv:1601.00934v2*.

Kaido, H., Molinari, F., & Stoye, J. (2019). Confidence intervals for projections of partially identified parameters. *Econometrica*.

Kline, B., & Tamer, E. (2016). Bayesian inference in a class of partially identified models. *Quantitative Economics*, *7*(2), 329–366.

Mattingley, J., & Boyd, S. (2012). Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, *13*(1), 1–27.

Appendices

A Tables

DGP	dim_p	J1	J2	S	CVXGEN_name
1-3	2	4	0	0	csolve_DGP1
4	2	8	0	0	csolve_DGP4
5-6	8	8	8	0	csolve_DGP5
7	9	8	8	0	csolve_DGP7
8	5	8	4	13	csolve_DGP8

Table 1: List of parameters for creating the `CVXGEN MEX` files for simulations in Kaido et al. (2016), Kaido et al. (2017), and Kaido et al. (2019). The first column corresponds to the parameter `DGP` in `KMS_Simulation`.

B Polytope Constraints and Non-basis Directional Vectors

In this appendix we describe the numerical issues that arise when either p is a non-basis directional vector or polytope constraints are imposed on the parameter space. We also propose a method on how to resolve these issues. The key issue is how to draw points from the contracted parameter space, see Equation (**??**). If the constraints $A_\theta \theta \leq b_\theta$ are included or if p is not a basis vector, then the contracted parameter space is a polytope but not a hyperrectangle (henceforth, called a non-basis polytope). In either case the numerical problem amounts to drawing points uniformly from a non-basis polytope.

We have identified three methods that can be used to draw points from a non-basis polytope. We, however, find that only the third method is reliable.

1. Hit-and-Run (HR) sampling. HR sampling uses Monte Carlo Markov Chain methods to draw points uniformly from the non-basis polytope $\Theta(h^{\text{counter}}_{\text{rate}}) \subset \mathbb{R}^d$. The method is, however, numerically unstable if the non-basis polytope is thin. The contracted pa-

rameter space in the EAM algorithm converges to a polytope in $\mathbb{R}^{d-1}$ as the contraction counter increases Therefore, HR sampling is unreliable for our problem.

2. Weighted average of vertices. In this method, the vertices of the contracted parameter space $\Theta(h_{\text{rate}}^{\text{counter}})$ are computed. A randomly generated point can be generated from a random weighted average of the vertices. Uniform weights do not guarantee that the point is uniformly drawn from $\Theta(h_{\text{rate}}^{\text{counter}})$. This, never-the-less, does not violate convergence assumptions for the EAM algorithm provided that there is positive mass at all points $\theta \in \Theta(h_{\text{rate}}^{\text{counter}})$. The algorithm that computes the vertices suffers from numerical issues as the parameter space becomes thin, and so this method is not appropriate for the CPI algorithm.

3. Draw-and-Discard sampling (DD). The algorithm first draws points uniformly from a box $B \supset \Theta(h_{\text{rate}}^{\text{counter}})$. It then discards any points that are not in $\Theta(h_{\text{rate}}^{\text{counter}})$. The volume of B relative to $\Theta(h_{\text{rate}}^{\text{counter}})$ must be small for this method to work well. If not, then a large number of initial points are required in order to achieve a target number of points. Therefore, the box B needs to be carefully defined.

In the current version of the CPI algorithm, the DD method only works for when p is a basis vector and the parameter space is a non-basis polytope. Modifications to the user-written function `bound_transform.m` are required. We explain the modifications with an example. The parameter space for DGP 8 is the polytope:

$$\Theta = \{\theta \in \mathbb{R}^5 : \theta_1 \in [0,1], \theta_2 \in [0,1], \theta_k \in [0, \min\{\theta_1\, \theta_2\}], k = 3,4,5\}.$$

First, to run DD sampling set `KMSoptions.HR=0` (to use hit-and-run sampling set `KMSoptions.HR=1`). To draw points from this space we use the draw-and-discard sampling method. The file `bound_transform.m` defines the box B above. It is not advised to set B to be the parameter bounds θ_{LB} and θ_{UB}, as the volume of this box relative to the contracted parameter space $\Theta(h_{\text{rate}}^{\text{counter}})$ quickly diverges. The inputs of `bound_transform` are: `LB_in`, `UB_in`, and

`KMSoptions`. The inputs `LB_in` and `UB_in` define the contracted parameter space (contracted in direction p). The outputs are the modified bounds `LB_out` and `UB_out` . Points drawn from $\{\theta \in \mathbb{R}^5 : LB_{\text{in}} \leq \theta \leq UB_{\text{in}}\}$ are unlikely to satisfy the polytope constraints. In particular, if

$$LB_{\text{in}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad UB_{\text{in}} = \begin{bmatrix} 10^{-4} \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

then it is likely that components $3-5$ violate the condition $\theta_k \in [0, \min\{\theta_1\, \theta_2\}]$. To resolve this issue the upper bound is modified, so that $UB_{\text{out},1} = UB_{\text{in},1}$, $UB_{\text{out},2} = UB_{\text{in},2}$, and $UB_{\text{out},k} = \min\{UB_{\text{in},1}, UB_{\text{in},2}, UB_{\text{in},k}\}$ for $k = 3, 4, 5$ (see `Lines 39-44` in `bound_transform.m`). The lower bound is unchanged. The box B defined by LB_{out} and UB_{out} contains the contracted parameter space and retains a good volume ratio. The modifications to `bound_transform.m` are model specific, and depend on the constraints $A_\theta \theta \leq b_\theta$.

If the parameter space is a polytope, then additional constraints for the linear program that computes $\hat{c}(\cdot)$ are required. These constraints are determined by the user-specified function `Rho_Polytope_Box`. Recall that we require $\lambda \in \sqrt{n}(\Theta - \theta) \cap \rho B^d$. The constraint $\lambda_k \in [-\rho, \rho]$ is already included in `KMS_33_Coverage`. For DGP 8, the following constraints need to be added:

$$\begin{aligned}
&\lambda_k \leq \sqrt{n}(1 - \theta_k), k = 1, 2 \\
&-\lambda_k \leq \sqrt{n}(0 - \theta_k), k = 1, 2, 3, 4, 5 \\
&-\lambda_1 + \lambda_k \leq -\sqrt{n}(-\theta_1 + \theta_k), k = 3, 4, 5 \\
&-\lambda_2 + \lambda_k \leq -\sqrt{n}(-\theta_2 + \theta_k), k = 3, 4, 5.
\end{aligned}$$

Observe that the constraint $-\lambda_1 + \lambda_k \leq -\sqrt{n}(-\theta_1 + \theta_k)$ is implied by the condition $\theta_k \leq \min\{\theta_1, \theta_2\}$. These $S = 13$ constraints are specified in `Rho_Polytope_Box`. In the `CVXGEN C` code generator, we set $S = 13$ for this DGP.