Appendix A: The PSP Algorithm

The Search Problem

The PSP search problem can be characterized in the following way. By adopting a particular definition of a data pattern we have induced an unknown finite partition on the parameter space, and our goal is to design a search procedure that visits each element of the partition at least once.

The Intuition

We adopt a sampling-based search procedure for finding regions. In this kind of procedure we randomly sample parameter values from some distribution, and check to see if they produce a new region. In a naïve, ``simple Monte Carlo'' (SMC) approach, we would specify a uniform distribution over the whole parameter space. As the number of samples drawn grows arbitrarily large, this procedure will uncover all regions (of non-zero volume) with probability 1. However, in high dimensional spaces this kind of procedure is inefficient (as discussed in the main text). As a result, we need to think about more efficient search methods. In essence, this amounts to finding a better probability distribution from which to sample.

One of the major inefficiencies in SMC search is that it takes no account of the fact that some regions are much larger than others. The goal in the PSP search is to visit each region at least once, in which case it is inefficient to spend most of one's time in a few big regions. Thus, an ideal sampling distribution would be one that is assigns equally probability to every region in the partition. In short, a natural way to improve on SMC would be to ``rescale'' the space so that all regions ``look the same size''. Our PSP algorithm uses a Markov Chain Monte Carlo (MCMC) procedure to attempt this kind of rescaling

PSP Sampling by Markov Chain Monte Carlo

MCMC is a powerful method for sampling a complicated *target distribution,* denoted $f(\theta)$. The assumption is that we know how to calculate the value of $f(\theta)$, but have no simple method of sampling from it. In the Metropolis-Hastings algorithm, we specify a *jumping distribution,* denoted $q(\theta_k | \theta_{k-1})$, used to sample candidate points. With probability $a = f(\theta_k)\, q(\theta_k | \theta_{k-1}) \,/\, f(\theta_{k-1})\, q(\theta_{k-1} | \theta_{k-1})$, the next sampled point is θ_k. However, with probability 1-a, we stay in the same spot and the next point is θ_{k-1}. Assuming that certain regularity conditions are met (discussed later), the values of θ converge to samples from the target distribution.

Recall that our goal was to specify a target distribution that is uniform over the regions. Obviously it is impossible to specify this distribution directly, since the partition itself is unknown. However, we can specify a series of approximations to this ideal distribution that will (under certain regularity conditions, discussed later) eventually converge to it. At any given point in the search, the PSP algorithm will have discovered some number of regions. In order to ensure that each of the discovered regions is treated equally, we run a separate Metropolis-Hastings algorithm for each of these regions, each with a target distribution that is uniform on that region (i.e. $f(\theta) \propto 1$ if θ belongs to the region, and $f(\theta) = 0$ otherwise).

On any given trial, we randomly pick one of these chains to draw the next sample from (actually, we systematically sweep through all of them in order to be more efficient, but the principle is the same), thereby ensuring that each of the discovered regions is treated equally irrespective of size.

``Growing'' a Partition

The procedure outlined above has the rather awkward property of converging to the ideal target distribution (uniform on the regions) so long as the algorithm has already discovered all the regions, which rather defeats the point of the search. In order to address this problem, we need to ensure that the algorithm looks outside the discovered regions often enough to ensure that it finds new regions. Our approach to doing this is based on the assumption that the structure of the parameter space varies smoothly (formal regularity conditions are discussed later). The idea is, firstly, that new regions will be found near the edges of old regions, and secondly, that the sizes of adjacent regions will be correlated.

A natural way to find regions that satisfy these constraints is to adapt the jumping distribution to ensure that the Markov chains jump outside of their region at an ideal rate. To see why this is useful, recall that the target distribution for a given MCMC routine is uniform on a particular region, so points that lie outside the region will always be rejected. Since we adopt a jumping distribution that is uniform on a small hyper-sphere (again, details to follow), the way to ensure a particular rejection rate is to adapt the size of the hyper-sphere. The corollary of this adaptation is that each of the Markov chains will (with some specified rate) jump just outside of its borders, looking for new regions. If the space varies smoothly in the sense discussed later, this will be guaranteed to find all the regions.

The other aspect of our approach is that, whenever a new region is encountered, the PSP algorithm concentrates solely on that region until it has ``caught up'' to the pre-existing regions in terms of the number of samples drawn from it (if multiple new regions are found, they are queued until they are all caught up). The reason for this is that the discovery of a new region is essentially a source of information about a potentially informative area of the parameter space that has not yet been fully explored. Accordingly, since we wish to treat all regions equally, the algorithm concentrates on a new region until it is on an equal footing with all the regions that were previously discovered. By deliberately ensuring that the various chains frequently jump outside their borders, the PSP algorithm is constantly ``looking'' for new regions.

The Jumping Distribution

As the previous discussion indicates, the choice of jumping distribution is critical to the success of the PSP algorithm. As discussed, we employed a uniform distribution over a hypersphere centered on the current location as the jumping distribution for PSP. The reason for using a hyper-sphere is that it is widely used in cases where the sampling region is constrained and no prior information is available about the shape of the region; the hyper-spherical shape of a jumping distribution is not likely to favor a particular shape of the sampling domain *a priori*.

In PSP, the size of the jumping distribution (*i.e.*, the radius of the hyper-sphere) must adapt to each region. If it is too small, almost all candidate points will be accepted, but every jump will be so small that it will take too many jumps for an exhaustive search of a region. Also, rejected points will rarely be generated. In contrast, if the size of the jumping distribution is too large, candidate points will be rejected

too often, and the granularity of the jumps will not be small enough to define the edges of a region, which requires a properly sized jumping distribution to succeed. Unfortunately, unless one is dealing with a normal distribution, no theory exists that defines the optimal jumping distribution. With the PSP algorithm, we have found it best to use an adapt the size of the jumping distribution so that the Markov chain on average accepts 20% to 25 % of the candidate points. In other words, each chain spends 20-25% of its time moving around inside the region, and the rest of its time searching just outside the borders. This range was identified heuristically by testing the algorithm on real and toy models many times using jumping distribution that varied widely in size.

This was accomplished by using the standard errors of volume estimates as a criterion of Markov chain efficiency. ***[DJN: I don't know what this sentence refers to]***

Formal Description of the PSP Algorithm

We now present a formal description of the PSP algorithm.

Step 0. Given θ_1 and Pattern 1, set $m = i = 1$.

Step 1. Establish $q_m(\cdot|\cdot)$ by adapting the size of its hyper-spherical domain. Go to step 2.

Step 2. Set $i = \text{mod}(i, m) + 1$. Go to step 3.

Step 3. Sample θ_y from $q_i(\cdot|\theta_i)$. If θ_y generates a new valid pattern, set $m = m + 1$, set $\theta_m = \theta_y$, record the new pattern as Pattern m, and then go to step 1. If θ_y generates Pattern i, set $\theta_i = \theta_y$ and go to step 2. Otherwise, go to step 2.

[DJN: This doesn't explain how the adaptation takes place, nor how the new region "catches up". Since I don't know exactly what procedure Woojae developed for doing this, I can't fix this myself.]

In the algorithm describe above, $q_i(\cdot|\theta_i)$ denotes the jumping distribution of the region corresponding to pattern i, centered at θ_i. The subscript $1 \le i \le m$ indexes the region from which we are currently sampling, and m represents the number of region (or number of data patterns) found so far. The algorithm terminates when a preset number of search trials (or size of an MCMC sample) is obtained for *each* of the discovered regions. To compensate for the fact that regions discovered early in the search process are likely to be sampled more than regions discovered later, the algorithm concentrates more on the newly discovered regions in such a way that the total number of trials in the search history will eventually be the same for all regions. An implication of this is that the algorithm aggressively searches for new regions in the vicinity of a newly discovered region.

Note that the initial conditions (step 0) require an initial parameter set (i.e, point in parameter space) in order to start the algorithm. These can be provided in one of two ways. Firstly, we could run a preliminary SMC search to find a parameter set that yields a valid pattern. Alternatively, if the modeler has a strong intuition about which parameter sets make sense, hey may be able to supply an initial set ``by hand''. In addition to supplying the initial parameter set, these two methods can help the PSP algorithm to overcome the potential problem of discontinuity in the partition (*i.e.*, disconnected groups of regions), as will be discussed in the next section.

Regularity Conditions

All numerical methods that work in high dimensions require that regularity conditions be satisfied to perform satisfactorily ***[DJN: I've removed the words "accurately" and "optimally", since as yet, we haven't proved either]***. This is also true of PSP. The following six conditions must be met:

1. *Connectedness*. A data pattern occupies a single connected region in the parameter space, such that any two points that produce the pattern can be joined by a path that passes only through points that produce that same pattern.
2. *Continuity*. Regions are contiguous with one another in the sense that any two patterns can be joined by a path that passes only through points that produce a valid data pattern.
3. *Smoothness*. The size of the regions changes smoothly in the parameter space, so small regions tend to cluster together
4. *Stationarity*. Model behavior is stationary in the sense that a given parameter set always generates a single, fixed data pattern. This means that the boundaries of the regions are fixed, not varying every time the model generates a data pattern
5. *Finite Space*. The volume of the parameter space to be searched is finite.
6. *Finite Partition*. The partition must be finite, in that there exist only a finite number of elements to be found.

[DJN: It should be possible to prove that, given these conditions, the PSP algorithm converges asymptotically to an ergodic Markov chain whose stationary distribution is uniform on the regions, and for each region is uniform on the region. That is, it finds all regions, and achieves the desired rescaling. Moreover, there's no point in specifying regularity conditions if you don't say what they guarantee (and so far, we haven't done this). In fact, if I were a reviewer, and I saw a whole lot of regularity conditions and a few general statements about MCMC, I'd be VERY suspicious].

[DJN: Technically, condition 3 is probably unnecessary for this proof, but will probably be needed if we try to show that the rate of convergence is faster than SMC].

[DJN: I haven't done much with the rest of this section. Just typos and a few minor edits].

If these conditions are violated, the PSP algorithm may not work particularly well, so it is useful to ensure that they are met, and to take the appropriate action when they are not. Although some conditions may be self-evident before applying PSP (4, 5), others can be more difficult to assess (1, 2, 3, and sometimes 6). In these latter cases, in the absence of proofs to test for violations, the best way to identify and deal with them is through numerical diagnosis using multiple PSP runs. This solution is not only common when solving high-dimensional search problems, which is what PSP is doing, but has proven to be successful (e.g., solving the local minima problem in nonlinear optimization). By running the algorithm multiple times with different starting values, tolerance levels, maximum number of function evaluations, etc, one can ascertain whether a consistent solution is obtained. In the same way, the consistency of a PSP solution can be established by performing multiple runs with different starting points in the parameter space, size of the sample used for an adaptation cycle, and the maximum number of search cycles before termination. Analysis of the resulting data should provide an answer to question. Below we discuss more thoroughly how potential violations of PSP's regularity conditions can be handled.

The assumptions of continuity and smoothness of the partitions in the parameter space are reasonable to make if the computational model being analyzed is composed of mathematical functions that are known

to exhibit these properties (i.e., the functions are continuous and smooth). If these assumptions are suspected of being violated (e.g., there are breaks in the range of a parameter), the modeler can check the accuracy and consistency of the PSP solution through multiple runs of the algorithm. Estimated center (or mean) points and covariance matrices of discovered regions, which are outputted by the algorithm, will reveal inconsistencies, such as discrepant volume estimates or the number of data patterns found. Inspection of these values across multiple runs should validate any suspicions and reveal the nature of the inconsistency if it exists. In addition, the results of these multiple PSP runs can be combined to create a more accurate picture of model behavior.

For example, if one suspects disconnected regions exist in the model, then at least two methods can be used to ensure PSP finds and records both. One is to supply different initial starting points. The modeler, who is most familiar with the behavior of the model, may be able to provide parameter values that generate typical patterns as well as less typical patterns that could come from another region in the parameter space. The more sets of parameter values the modeler can provide, the more likely the algorithm will find the disconnected regions if they are present. Alternatively, another way to find disconnected regions is to use SMC. That is, randomly sample the parameter space to find these "islands." Large regions are of most importance because the data from them could influence the PSP analysis. Fortunately, size works to the modelers advantage in this circumstance because a reasonably long run of SMC should be able to find at least one point in the island, which is all that is needed to discover its existence. The PSP algorithm can take over from there and find the other elements of the partition that belong to the island. In simulation tests in which islands of data patterns were placed in various regions of the parameter space, we have found SMC to work quite well and in a reasonable amount of time with models that have up to seven dimensions. At higher dimensions, the region becomes such a tiny speck in the model's parameter space that exponentially longer runs are needed, which can quickly become impractical and the success rate drops.

This same approach can be used to identify disjoint regions that correspond to the same data pattern. Multiple runs with different initial points or a preliminary SMC run can help detect such discontinuous regions, again, especially if they occupy considerable individual volumes. Output analysis using estimated center points, volumes, and the covariance matrix of discovered regions will reveal their existence (and location) through discrepancies across repeated measurements.

The stationarity assumption implies that PSP is not applicable to simulation-based, probabilistic models (e.g., distributed connectionist). For this reason, a probabilistic model can be analyzed by PSP only if its simulational component can be replaced with a closed-form probability density (or mass) function, so that data patterns are defined on the space of probability distributions. Study of the application of PSP to probabilistic models is currently underway. As for the requirement that the range of parameters must be finite, if some unconstrained parameters are unavoidable (i.e., plausible data patterns could still be generated from their extreme values), it may be useful to reparameterize the model utilizing log, inverse logistic, or other transformation function.

As with some of the preceding assumptions, the finiteness of the partition is not always guaranteed. To promote the successful application of PSP, it is wise to restrict the model's prediction space by imposing restrictions on possible data patterns when they are reasonable, as was done with ALCOVE ***[DJN: I don't understand this]***. In most cases, such restrictive definition of a data pattern will be justifiable from experimental data or prior work with the model. By precluding nonsensical data patterns, the model analysis can be fair, realistic, and thus informative.

The preceding discussion is intended to alert the potential user to the issues that must be considered when applying PSP. It also serves as demonstration of PSP's current capabilities. The sampling-based approach to partitioning the parameter space makes it a powerful tool for model analysis. There is room for improvement, and enhancements to improve its accuracy and efficiency are being developed. Current and future updates of the algorithm, including Matlab source code and a tutorial, can be found at http://quantrm2.psy.ohio-state.edu/PSP/

A Test of the Accuracy of Volume Estimation

[DJN: waiting on the new simulations, so I've not written anything about this. However, I do think that it should go after the regularity conditions, since those conditions are conceptually related to the search process, whereas the volume estimation is not].