**Abstract**

While calibration is an important element of the System Dynamics modeling process, traditional calibration techniques exhibit significant limitations. Many such techniques are limited to providing point estimates of calibrated values, sometimes together with information on uncertainty around such estimates. Such techniques also impose assumptions concerning the error distributions and privilege a specific dynamic model structure. Markov Chain Monte Carlo (MCMC) techniques offer a powerful, general, and versatile alternative approach. Bayesian MCMC approaches eschew point estimates, and instead provide a means of sampling from a full ("posterior") distribution of parameter vectors. Such techniques can further express the relative likelihood of different model structures. Finally, MCMC approaches allow a modeler to explicitly specify a general probabilistic model giving the likelihood that observed empirical data would be produced by a certain parameter vector. While MCMC approaches offer strong benefits, it can be daunting for System Dynamics modelers to secure even a basic understanding of the MCMC process, and there is only a small extant literature concerning applications of MCMC to simulation models, largely using language unfamiliar to most System Dynamics practitioners. Within this paper, we seek to provide a gentle introduction to the use of Bayesian MCMC techniques for System Dynamics parameter estimation.

# Bayesian Parameter Estimation of System Dynamics Models Using Markov Chain Monte Carlo Methods: An Informal Introduction

This paper describes a Bayesian approach to estimation of the parameters of System Dynamics simulation models.

The context of this work lies in limitations associated with a key component of the dynamic modeling process – calibration. Classically, calibration of a simulation model seeks to estimate little-known model parameters by comparing the emergent behavior exhibit of that model with corresponding behavior observed from the external world. For example, we may be seeking to use calibration to estimate the values of parameters concerning contact rates on which we have limited data directly from studies or from surveys that had been conducted. Within this calibration, we will observe how the simulation model as a whole – or large subpieces thereof – behaves in terms of its emergent behavior. With simulation models, the complexity of such behavior is generally such that we can't simply "back-calculate" the values for parameters such that model output will match the empirical data. Instead, we compare the emergent behavior of the model against empirical data on corresponding quantities to which we have recourse to, and try to use an optimization algorithm to estimate the values of model parameters that yield model behavior most closely corresponding to that empirical data.

The calibration process traditionally gives us point estimates for parameter values, that is, it gives us a set of values, one for each of the less well known parameters, whereby the model best matches the empirical data that we have at hand. In more advanced forms of calibration, the approaches we use may seek to additionally yield confidence intervals. Such confidence intervals help characterize the degree of uncertainty in the point estimate.

While a very important and powerful component of the modeling process, this traditional form of calibration does exhibit some important limitations. One of the most fundamental limitations is that we are performing the calibration on a single dynamic model at a time. That is, we are assuming for the point of view of deriving model parameters that we know the model structure exactly, and we are trying to identify the values of the parameters such that, *conditional on that assumed model structure*, the model behavior best matches our empirical data. This is a strong assumption. Often we are not exactly sure about the underlying structure. Perhaps there are several competing hypotheses as to the nature of that underlying structure involves. And yet for the purpose of calibration, there is an important set of strictures that limits us to calibrating one model at a time.

A second important limitation of traditional calibration is that – in many but not all cases cases – we are emerging with just a single point estimate for parameter values, an investment that doesn't really capture the degree of variance associated with model parameter in isolation, or the covariance between the calibrated value of multiple parameters. For example, such point estimates don't capture systematic

variation in the uncertainty associated with, say, one parameter versus another – the fact that, for example, this model may closely match the empirical data either if both parameter A and parameter B are high together, or parameters A and B are both low together, but exhibits poor matches otherwise. Or, for example, the fact that the model can only match the empirical data closely if either parameter A is high and parameter B is low or vice versa. These sort of systematic variations – which may be associated with "valleys" in the error function – in the uncertainty are not really captured within single point estimates that emerge through many calibrations – despite the fact that these uncertainties (variances and covariances in parameter estimates) may have important searing on the outcomes of interventions.

Even when we do have estimates of covariance for parameters within the underlying system, these are local estimates, traditionally around the single point estimate – around, for example, the maximum peak (mode) of the distribution. There is thus limited understanding of the global shape of the distribution. Such local estimates will not, for example, capture the fact, for example, that there is another peak nearly as high somewhere else within parameter space, where some other combination of parameters offers emergent behavior nearly as good, and that may have quite significant implications for interventions for example we may seek to design. In some cases – such as when calibrations are being simultaneously performed on many parameters – computation of the covariance matrices around the point estimate may be highly challenging. It can also be very challenging go from a covariance estimate for the parameters themselves to an estimate for functions of those parameters of particular interest (most notably, for model output given those parameters).

Some assumptions associated with classic methods of optimization are often less well articulated or less explicit. For example, the discrepancy metrics whose value is optimized will incorporate assumptions about the distribution of discrepancies between the model results and the empirical data. Our imposition of these assumed distributions for errors may come with significant challenges for particular practical problems when those distributions are not appropriate – that is, when the actual error distribution does not accord with what was implicitly assumed to obtain during calibration.

## *1.1 A Different Approach*

This paper describes a different approach to seeking to enhance the accordance between System Dynamics models and data – one that incorporates elements of both calibration and validation. Within this approach, we will be moving beyond the assumption of privileged point estimates and possibly confidence intervals centered around such point estimates that measure only local features associated with the distribution. Instead, we are going to be dealing with an approach that seeks to understand the shape of the probability distribution associated with parameter values – that is, which helps us understand the relative likelihood of parameters combinations drawn from across that parameter space.

## *1.2 Operational Representation of the Distributions*

The "knowledge" of the posterior distribution that emerges from MCMC using simulation models is not characterization of such distribution using explicit functional form (e.g. as a familiar distribution). With nonlinear simulation models (such as are common in System Dynamics – e.g. variants on Bass Diffusion

and SIR models), we generally don't have the option of arriving at an analytic form for the output of the simulation model, and the shape of the posterior distribution will depend critically on such output. As a result, when using MCMC together with simulation models, there is generally no option for analytically (using a formula) representing the posterior distribution. Moreover, in a high-dimensional space, explicit representation can also be awkward. Instead, we will be working with an "operational knowledge" of the distribution – rather than having a reified representation of the distribution, MCMC provides "knowledge" of the distribution in the sense that we can *draw samples* from this posterior distribution, such that the values of those samples are (approximately[1]) distributed as given by the posterior distribution. And for each such sample, for example, we could sampling from various model outputs, compute summary statistics such as the mean, etc. So the idea is that MCMC provides us a way of specifying or arriving at the posterior distribution, not by giving us an explicit reified specification for it, but instead by allowing us to numerically sample from it[2].

# 2 Essential Background

To accomplish the goals laid out here, we will be appealing to classic Bayesian[3] concepts, but will apply them specifically to the MCMC process. That is, we will allow the user to express a "prior" distribution and – following incorporation of information relating the model to empirical data – seek to draw samples from a "posterior" distribution. For readers unfamiliar with Bayesian approaches, we attempt here to provide some intuition for such terms, and then discuss the application to MCMC.

## *2.1 Some Common Notation*

For consistency with the literature, this document will adhere to some common notation conventions from Bayesian statistics. We will use the symbol $\theta$ to indicate a parameter vector – that is, a vector each of whose elements specifies the value to be used for a particular parameter. For example, perhaps we have a model of infectious disease transmission, and you'd like to impose an assumption that the contact rate is 10 people per day, that the hand hygiene compliance rate is 50%, and that the mean infectious period – that is, the mean duration of time until recovery from infectiousness – is 3.2 days. These assumptions could be specified using a particular parameter vector maybe $\theta_1$ – a vector that denotes a particular point within the parameter space associated with $\theta$. Another distinct value of $\theta$ (call it $\theta_2$) might be associated with the assumption of a contact rate of 20 people per day, handwashing compliance also 50%, and also

[1] This approximation will be achieved as a large number of samples are accumulated, and the distribution of the Markov Chain converges to the distribution of the posterior.

[2] We note that a reified, explicit representation of the distribution is not necessarily to be preferred to this operational ability to sample from that distribution. In some cases, researchers who have an explicit representation of the distribution will instead use MCMC to sample from it, as sampling can simplify certain tasks.

[3] While we have taken a Bayesian perspective here, we note in passing that it is possible to make use of MCMC methods in a non-Bayesian context. This would put aside dealing with a prior distribution, and instead deal only with a likelihood function.

3.2 days average days of infectiousness. As noted above, the framework we are describing here is very general, in that the parameter values can be parameters within the simulation model, within a probabilistic model that's going to express the likelihood that a given set of simulation model output could explain the observed data, or even parameters indicating which model structure is to be assumed.

We will further denote the observed data using a vector *y*. This would describe the empirical data that we have in hand – perhaps time series, perhaps particular data points to which we want the simulation model to match. (This would be the sort of data that we are traditionally trying to match within a calibration experiment). A key point here is that while we will denote them using single letters, both θ and y are generally vectors, and thus can include many components.

## *2.2 Understanding Bayes Rule*

Within this discussion, we provide a brief background in Bayes' Rule, so that the reader can understand some of the basics for how the MCMC algorithm operates. As noted above, Bayes' Rule serves as a critical conceptual underlying component of the Bayesian MCMC approach. Bayes Rule specifies that[4]:

$$P(\theta|y)=P(y|\theta)P(\theta)/P(y)$$

Readers may be familiar with the fact that Bayes' rule relates the likelihood of A given B (*i.e.* P(A|B)) to the likelihood of B given A (*i.e.* P(B|A)).

To link expressions in the equation up with the terms introduced earlier, P(θ|y) is posterior probability of θ, P(y|θ) is the likelihood function (which would generally take into consideration not only θ directly, but also the results associated with running the simulation model on θ), and P(θ) is the prior distribution. So Bayes' Rule allows us essentially to take a situation where we know a prior defined over parameter vectors (P(θ)) and where we know a likelihood that the empirical *y* could be explained given a set of parameter values and their resulting model output P(y|θ), and "flip it around" to get an estimate of what we want – the relative probability (density) of those parameter values θ obtaining in light of the known *y*, the information that describes the posterior P(θ|y). While we've given them name, here further how each of these components functions.

The likelihood P(y|θ) is what we bring to the table in terms of our likelihood rules. I have mentioned it here, this likelihood formula. We define a likelihood formula that indicate the relative likelihood of seeing particular values of empirical data, such as particulars counts of emergency room intake for H1N1 in light of particular values for θ, for the parameters in the corresponding model output. As noted above, this formula typically depends on quantities that emerging from the simulation model on those θ. For instance, for a particular situation, P(y|θ) might represent the likelihood that 5 people presented to the emergency room in week 2 of the H1N1 pandemic in your city, of this pandemic with H1N1 for a

[4] Within this rule, P(A|B) is read as "Probability of A given B" (for discrete values A and B) or "Probability density of A given B" (for continuous values A and B). By definition, P(A|B)=P(AB)/P(B)

particular set of parameter values θ that you know – a contact rate of 10 people per day, hand washing compliance of 50%, a mean of 3.2 days average infectiousness. To assess this (relative) the relative likelihood, you would here have to run the simulation model because looking at those parameter values in and of themselves is not going to tell you the implications for how many incident cases are there within that week, for example, because that's an emergent property of the model. The values of the parameters alone are not going to directly tell you the key information you're going to need to assess the relative likelihood of the empirical data *y*. Instead, a simulation model (possibly one chosen by θ) will need to be run using any simulation parameters specified by θ, and we would then obtain the incidence rate (in terms of count *c* of people per week) at week 2. We would then use a probabilistic model to calculate the desired probability P(y|θ). For our example, we might consider that each person has a certain chance of *p* of presenting (more or less immediately) for care given H1N1. The probability model would then calculate the likelihood of receiving a value of 5 (the observed count of cases) from a Binomial distribution with *c* trials, each of which has a likelihood *p* of success. Thus, in the first step here, we are going to run the simulation model. In the second step, we compute the relative posterior where we are going to, we are going to take this this assessment, we can compute the likelihood P(y|θ) of the observed y in light the associated simulation model output implied by θ.

It bears noting that P(y|θ) will often be a multiplicative expression that successively multiplies several likelihood terms. A given one of those terms would define the likelihood that a particular component of vector empirically observed data *y* is explained by the simulation model given parameter vector θ. The multiplication of the terms is a reflection of the treatment of these events as conditionally independent (that is, the view that the variability in the occurrence of particular values of *y* is independent given θ).

As noted above, P(θ) is the prior. In the context of Bayesian MCMC, this reflects the fact that we often have some sense as to where parameter values fall, and which of several possible competing model structures may be most appropriate. This varying degree of belief that we bring to the table regarding possible values of parameters can be captured within a Bayesian context using the prior. This is something that we specify ahead of time (*prior to actually observing the empirical data* or *considering the matching of the model to that data*) given our perception regarding the relative likelihood of different parameter values. In a Bayesian context, we start with the option of imposing either weaker or stronger initial guesses as to the relative likelihood of different parameter values – a weaker or stronger prior. Sometimes we just impose an informative prior, or a prior that is – for example –uniform over some bounded θ space. In other cases, we may come up with rather strong assumptions based on rough knowledge confided by experts, or based on observations of ranges of values seen in the literature. Generally, the choice of prior should be based on how much priori knowledge is available. When there is limited prior information available, one may use the methods described here (running MCMC algorithms for different prior) to check how the posterior would be affected by different priors. As well known, if the data contains rich information (e.g. large sample size), the posterior will be dominated by the likelihood and thus will be insensitive to the choice of priors. However, when the data do not contain much information, the posterior may heavily depend on the prior. For cases with small amounts of data, the choice of prior is crucial and should be made with caution and exploration.

The denominator of the equation contains only P(*y*) – the likelihood of different empirical data occuring. This term is notable because it's just a constant with respect to θ – it doesn't vary at all with parameter vectors θ. So when we are assessing the relative likelihoods of different values of θ, on we don't have to worry that this denominator term will favor some θ over others. Instead, it holds the same value across all parameter vectors θ, and one can think of 1/P(*y*) serves as a sort of coefficient. While knowing the value for this would still be important if we were trying to evaluate the numerical likelihood of P(θ|y), we will just talk about the *relative* likelihood of θ given *y*, that is, P(θ|*y*). To put it in another way, to compute the relative value of the probabilities of 2 parameter vectors, $\theta_1$ and $\theta_2$ (that is, the relative values of P($\theta_1$|*y*) and P($\theta_2$|*y*)), all we have to do is compute the values of P(*y*|$\theta_1$)P($\theta_1$) and P(*y*|$\theta_2$)P($\theta_2$) – both values are multiplied by the identical constant 1/P(*y*). Because it's a coefficient in front of both, we don't have to worry about its particular value. There are some coefficient in front of it, but as long as we are drawing samples from θ in a way that's proportional to the relative likelihood of θ, we will be sampling from the P(θ|*y*) appropriately.

The posterior, P(θ|y), is what we are seeking. In light of the empirical data *y*, we want to have some understanding of the relative probability that we are dealing with an parameter vector θ. And following the use of Markov Monte Carlo techniques, we arrive at this updated distribution – the posterior – which takes into account the empirical data (in light of model structure(s)). This posterior distribution (from which we sample) considers, as it were, not only our initial sense as to the likely values, but also – critically – takes into account the plausibility of different possible sets of parameters values by actually running the model and observing the consistency of various model outputs with empirical data. Providing that there is sufficient empirical data, the insights from the observed data and its consistency with model results will typically overwhelm our initial guesses (in the form of the prior) – our original guesses recede into the background in terms of the shape of the posterior.

## *2.3 Using the Data from MCMC*

This section builds on the core concepts introduced in the previous section to describe the use of the results emerging from the MCMC algorithm, with a particular emphasis on contrasting the deliverables from MCMC with those emerging from the traditional calibration widely applied within System Dynamics.

The posterior distributions which emerge from MCMC are distinctive in several respects from the sort of point estimates that often come out of calibration. Firstly, far from privileging particular points in posterior space – as are the point estimates emerging from traditional calibration – the method instead allow us to appreciate global features of the distribution. Secondly – although application of this approach can require advanced understanding – such posterior distributions can permit us to express confidence in different the likelihood of different models. Thirdly, we can consider the relative probability not only of simulation parameters values, but also of parameters within a formal probabilistic model used to express the consistency between the model results and empirical data. Finally, given the capacity to sample from the posterior distribution, it is generally far easier to reason about the

distributions associated with other quantities (such as model outputs) than is the case with traditional confidence intervals surrounding calibration point estimates.

This final point reflects the fact that once we arrive at a posterior distribution, there are diverse useful things we can do with it. For example, the capacity to draw samples from the posterior distribution over parameters also allows us to draw samples from distributions (either joint or marginal) associated with computed System Dynamics model quantities ("outputs"), such as would be of interest for baseline projections, policy analyses, system vulnerability assessments, etc. It is straightforward to sample not only from the marginal distributions associated with such outputs, but also the joint distributions – allowing us to capture the fact that in situations when one output is low, another is high, etc.

We can also use the MCMC approach to sample from distributions comparing model results in different scenarios – we can sample from distributions over, for example, how much better intervention A is than intervention B with respect to some particular metrics – for example, in terms of quality adjusted life years lived, cost, deaths averted, etc. As before, we can sample from such distributions either on a marginal or joint basis. Alternatively, we can ask, for example, what's the posterior probability that intervention A is better than intervention B in terms of various System Dynamics model outcome measures (e.g. Quality Adjusted Lifeyears Lived, Deaths Avoided, Cost, etc.).

Another option that is sometimes exercised and which applies to any such distributions is to use the capacity to sample from the posterior distribution to derive point estimates. For example, we can derive a median or mean value for parameters, or a maximum a posteriori estimate of parameter values (associated with the peak of the distribution). By imposing a "flat[5]" prior or a weakly informative prior (essentially, only considering the likelihood function), we could also use MCMC approaches to arrive at maximum likelihood estimate. So the posterior can give us the types of point estimates provided by traditional calibration techniques, but the further option of securing other point estimates not traditionally provided by calibration (e.g. medians, means, etc.).

Where we seek interval estimates around the maximum a posteriori estimate of parameters or for diverse System Dynamics model output, MCMC makes it straightforward to derive credible intervals. This stands in contrast to the difficulties that can be encountered in trying to go from confidence intervals around parameters point estimates arising from traditional calibration to confidence intervals around model outputs or other functions of those parameters. A credible interval is an interval estimate for a parameter in Bayesian paradigm that is based on the posterior distribution of the parameter while a confidence interval is based on the likelihood. The interpretation of credibility intervals is more straightforward than confidence intervals because the coverage level of the former stands for probability while not for the latter case.

[5] While the application of uniform priors is limited to parameters exhibiting bounded range of variation ("support"), a flat prior is a more general construct that can handle parameters associated with unbounded ranges. When such an "improper" prior (i.e. a prior that does not integrate or sum up to 1) is used, care must be taken to ensure that the resulting posterior is itself proper (is integrable).

Thus, broadly speaking, Markov Chain Monte Carlo (MCMC) techniques helps us to reliably derive (in the form of reliably sampling from) posterior distributions for parameters – parameters over simulation model parameters, over probabilistic model parameters, and even over models. What's really notable here is that MCMC is a very principled and general way for arriving at these approximations to these distributions. We have a very regularized way of generating samples in particular from this distribution. And we can use that flexibility to apply the approach to a wide variety of models, and a wide variety of types of challenges.

# 3 Bayesian MCMC

## *3.1 Why "Markov Chain Monte Carlo"?*

What lies behind the name "Markov Chain Monte Carlo"? MCMC provides us a way of sampling from the posterior distribution, and approximations thereto. The fact that we sample randomly from it again and again and again places it in a category of algorithms known as "Monte Carlo" techniques.

And what accounts for the second MC in MCMC – the "Markov Chain" in "Markov Chain Monte Carlo"? One question that a lot of individuals new to MCMC bring to bear when thinking about these techniques is confusion about the associated Markov Chain. There is often an assumption that because this is "Markov Chain" Monte Carlo technique, that Markov Chain must be reified in very noticeable way – e.g. visualized, constructed and laid out. It turns out that the Markov Chain that's involved in this process is indeed extremely important, but it's really not typically reified. It's left implicit within the code. As we will see, a Markov Chain lies *implicitly* behind the ability to generate samples – while it does lie at the heart of the sampling methodology, there is no reification of that chain. Specifically, we will be hopping from one possible sample probabilistically – in what can be viewed as a type of Markov Chain. This implicit Markov Chain offers an extremely flexible means of drawing values from a posterior distribution, but is not represented explicitly.

Now, this may seem like that a puzzling thing. Why do we need a Markov Chain technique to do this? Well, it's worth reflecting on the fact that drawing number from general distributions is not easy. Standard pseudo-random number generators make it straightforward to draw a random number from a uniform distribution. This is enough to draw from some other basic distributions. Specifically, if we have a distribution which is associated with some analytic expression, often we can arrive at an analytic expression for the cumulative distribution function (CDF). Knowing the CDF allows us to draw a random number from the desired distribution. Specifically, if we know the CDF, we can draw a random number from a uniform distribution, and then we can invert the CDF – we can return the parameter value that gives accumulative probability that is equal to the value we drew from the uniform distribution.

The problem is that, while this technique is very powerful, it only generalizes somewhat awkwardly to higher number of dimensions, and is also more cumbersome in cases where we only know numerically (rather than analytically) the distribution from which we wish to sample. More fundamentally, this approach exhibits severe problems when we don't know the CDF up front. If we can't arrive at a full specification of the CDF, we cannot proceed in this manner. In cases where MCMC is used, it is

typically not feasible to numerically depict the "shape" of the distribution we wish to sample – much less to identify its CDF.

So in such cases, we need another way besides this traditional way to draw values from arbitrary distributions. This trick that we use with the CDF works great for simple distributions, for the cumulative distribution function, the CDF that is, is known exactly, but it does not work well we can't formulate that distribution explicitly, the cumulative distribution function.

Implicit use of an implicit Markov Chain will be key to generate samples from the posterior distribution without knowing the shape of that posterior or its corresponding CDF.

## *3.2 A Brief Overview of MCMC*

This section provides a glimpse of the high level steps involved in using Markov Chain Monte Carlo techniques for parameter estimation using simulation models. This information will be useful to keep in mind as a structure for the remainder of this document – perhaps even for reference while reading the balance of the document. For some readers, the information presented here may be all that is sought.

There is a set of tasks required to initially prepare a simulation model for use with Markov Chain Monte Carlo methods.

One of first task is to decide on the parameter space to be sampled by the MCMC algorithm – that is, over what parameters we wish to derive distributions, and the ranges of values to use. In some cases, we may have several choices of parameters from the simulation model which should be calibrated; while it's not always possible, to decrease covariance in the posterior distribution (see Section 5.1), it is ideal to examine parameters that have distinct types of impact on the model. We should be sure to include in our plans parameters drawn from the probabilistic model. We may further encode which model we want to use, with a categorical parameter. Finally, to ease framing the MCMC problem in a fashion suitable for existing MCMC packages, we will often create convenience variables that a represent transformations the parameters such that these transformed variables vary over an unbounded space (from minus to plus infinity rather than over a defined interval).

Secondly, we need to assign prior distributions for parameters. As noted above, such distributions will capture our best guesses as to where parameter values might lie before we actually start rigorously comparing the model output to empirical data. These prior distributions for parameters could be specified in a parametric fashion – for example, using normal, uniform, log normal, Gamma, Gumbel, etc. distributions. In other cases, we may wish to use nonparametric distributions.

Thirdly – and this is probably the most novel, critical and conceptually challenging component of the work – we need to formulate a likelihood function – an expression for the (relative) likelihood of seeing particular values of empirical data in light of values that parameters and – and this is the key component when we come to simulation – resultant simulation model output. This element is the key component of what we term the "probabilistic model" associated with the process laid out here. Because is so critical, we will now make further elaborate upon it.

Suppose that we have some values of parameters that have yielded corresponding output from the simulation model. We need to specify a likelihood function – a formula when given parameter values and the simulation model output that comes from assuming those parameter values and that model will return the relative likelihood[6] that such a situation (as depicted by the parameters and model results) could in fact yield the empirical data. Suppose, for example, that we have observed data on the number of emergency room admissions for each successive week during an H1N1 output, and that we further we have a System Dynamics model that simulates the number of incident cases of H1N1 within the population over time. Suppose that when we run that simulation model for a particular assignment of parameters values (i.e. for a specific vector $\theta$ in parameter space) it gives the number of incident cases for each successive week. To specify the probabilistic model in this case, we would be seeking a formula that would give the likelihood that the sequence of empirically observed cases over time would result from those parameter values. *It is critical to recognize that this formula would here consider the System Dynamics model's indicated rate of incidence over time when that model is parameterized with those parameter values.* In short, we would need to provide an expression for the relative likelihood that we would see the empirical data if we were to assume those parameter values – a calculation that typically depends critically on the results emerging simulation model when it assumes those parameter values. Typically this calculation would also depend importantly on parameters that lie within the probabilistic model itself such as the likelihood of a given person who develops flu symptoms would in fact present for care at the emergency room rather than convalescing at home.

Now given these components, we implicitly apply Bayes rule to arrive at a calculation for the relative probability of a given parameter vector in terms of the likelihood formula and prior. Specifically, by multiplying the value of the likelihood function and the prior at a given point, we can calculate the relative probability density of a particular parameter value within the posterior distribution.

The output of MCMC is dependent samples for the parameters from the posterior distribution. The details of how MCMC algorithm can be implemented to draw these samples are presented in Section 7.

## *3.3 Understanding the Algorithm*

This section will focus on the internals of the MCMC algorithm – the mechanisms by which we sample values from the posterior distribution.

As noted earlier, we use a markov chain to accomplish the sampling from the posterior distribution. There are a number of commonly used algorithms used for Markov Chain Monte Carlo. In this section, we employ one of the simplest algorithms – what is called a random Walk Metropolis Hastings algorithm. Essentially, we will generate a candidate sample with a random perturbation from the current (last accepted) sample. Given this candidate, we will then calculate the relative posterior probability (density)

[6] We note that the calculation of this likelihood makes the assumption that the sampling distribution of the data is characterized by the composition of the probabilistic mode and the dynamic model. Here the dynamic model will map the parameter values to a dynamic model outputs that are "closer" in character to the empirical data. The likelihood function would consider that dynamic model output as well as other parameters and return a likelihood.

of this candidate value of θ, P(θ|y). Given P(θ|y), we will accept that sample with a probability that reflects the relative posterior for this candidate value of θ compared to the (relative) value of the posterior at the last generated sample. If this candidate is not accepted (i.e. is "rejected"), we re-issue the last accepted value. This strategy will yield a sampling pattern that eventually approaches the desired posterior distribution.

We begin the algorithm by choosing an initial point (parameter vector θ) in parameter space and assess and remember its posterior distribution. We will then begin the central Monte Carlo iteration. Firstly, we obtain a candidate sample generated by adding a random disturbance Δθ to the current point θ. We then assess P(θ|y), the relative value of the posterior of that candidate sample, using the technique described in the previous section – applying Bayes' rule by multiplying the value of the likelihood function P(y|θ) by the value of the Prior P(θ). As noted above, calculation of the likelihood P(y|θ) will typically involve running the simulation model and applying a probabilistic model.

Having calculated this posterior value for the candidate parameter vector, we will either accept or reject it. That is, we either emit it or just throw it out based on the ratio between the relative posterior at the candidate parameter vector and the value of their relative posterior of the last generated sample[7]. If we accept it, we will release the candidate as our new sample from the posterior distribution, and our current location within parameter space will shift to the candidate point. If we instead reject the sample, we just emit the previously omitted value as our new sample, and we don't change position. Having generated the sample, we then continue on to the next iteration of the loop. For the algorithm shown here, the idea here is that if we find a higher relative likelihood point than the current one, we accept that point (and transition there) automatically. If it's higher than our current relative likelihood, we automatically go there and emit that sample. If we find a lower likelihood point than the current one, we only use it as a new sample with a certain probability as given by the ratio of relative posteriors involved.

It's important to recognize that the greater the value of the relative posterior at this candidate θ (relative to the previously emitted sample), the greater the probability that the algorithm will accept it, and emit that new candidate as a sample. So if the candidate is extremely likely, we are much more likely to accept it and emit it as a sample. Similarly, if the current point θ, the chances are higher that we will remain at this point (rather than drawing a sample from a new candidate point). As a result of both of these influences, these points will tend to be sampled more – that is, emitted more. By contrast, if our candidate sample is, unlikely to occur, then the ratio of posteriors would be typically smaller, we are more typically reject it. If (by luck) we do make it to an unlikely point in space, the ratios of many other possible candidate points will look favorable compare to this one. For both of those reasons, we'll tend to sample less likely points less frequently.

So this, this algorithm provides sort of its internal throttling such that it will emit more frequently things that are more likely to occur, and less frequently things that are less likely to occur.

[7] You'll note that the P(y) term in the definition for the formula for P(θ|y) totally drops out when computing the ratio between $P(\theta_1|y)$ and $P(\theta_2|y)$, as both have the identical term P(y) in their denominator.

The algorithm discussed here is guaranteed to (dependently) sample from the posterior distribution. However, it frequently starts from an arbitrary point in space – not from particularly high likelihood points. We don't want it to depend on the vagaries of where we happened to start within parameter space. In order to apply this algorithm appropriately, we are going to need to run through some burn-in time before starting to record the samples. That is, in order to arrive at confidence that the samples will be representative of the underlying posterior we are trying to approximate, we need to run it enough times so that it "forgets" about the vagaries of where it started, and begins mixing well within the parameter space – in a fashion as given by the underlying distribution.

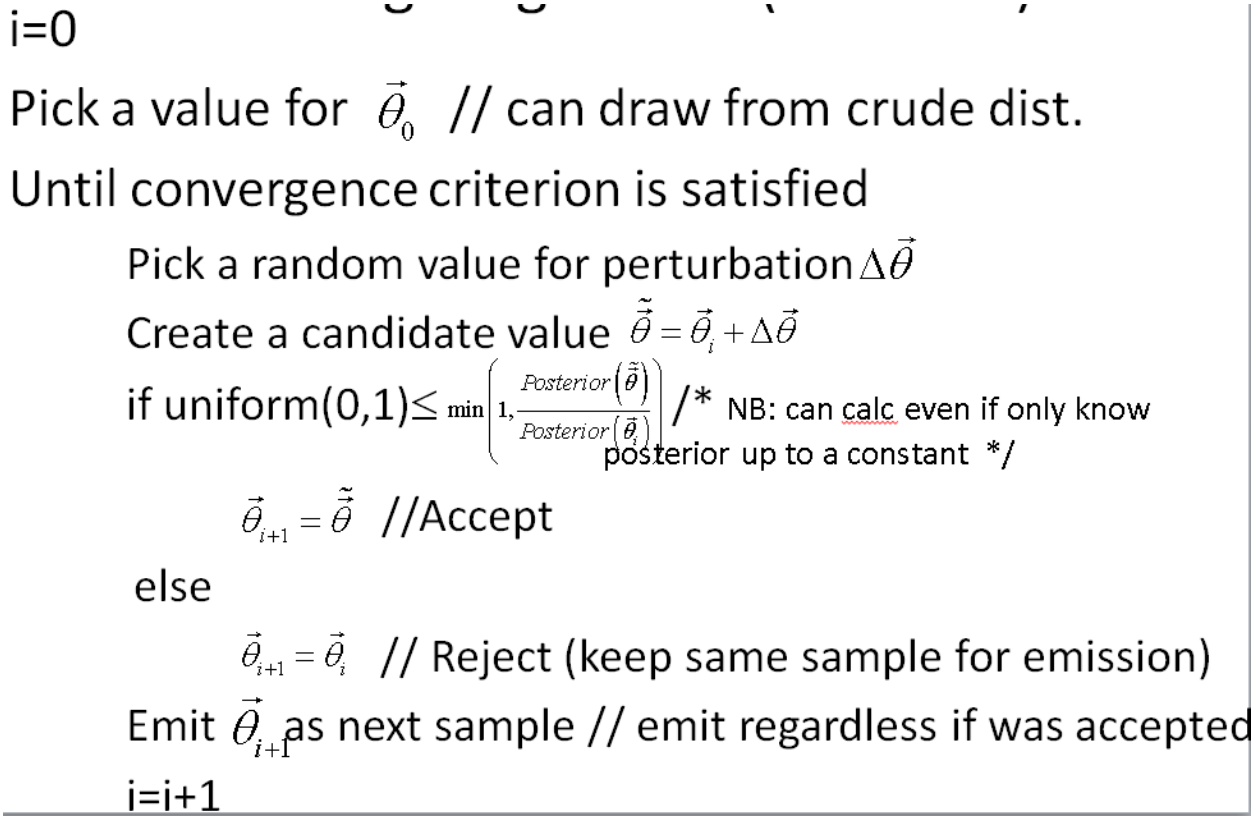


**Figure 1: Pseudocode for Random Walk Metropolis Hastings algorithm**

## *3.4 The Implicit Markov Chain*

Where is the Markov Chain in the algorithm? At a given point during the body of the algorithm, we have a particular value associated with $\theta_i$, associated with our current point. We then have a certain chance of transition to other possible values of θ. We have many possible values for θ we can get to from this state (i.e. from this point in parameter space), and we go to a given such possible next states with certain probabilities, based on the ratio of posteriors.

The entire process can be abstracted to transitions along a Markov Chain, where each state of the Markov Chain is defined by being at a certain point in parameter space, and the transition probabilities associated with a transition from state A to state B is given by the ratio of the posterior at state A and at state B. The

Markov Chain is implicit within this algorithm; this algorithm is capturing the behavior of Markov Chain even though the chain is not explicitly represented.

## *3.5 Brief Comments on Why MCMC Works*

While this paper does not have time to explain it, the reader should recognize that there is an extensive theoretical edifice that explains why MCMC work. At a fundamental level, this is a result of two proven facts:

- The actual posterior is a stationary distribution of the algorithm shown above.
- There is only one stationary distribution for the Markov Chain.

The implication of these facts is that the actual posterior is the unique stationary distribution of the algorithm shown.

More technical readers may have some interest in understanding the design principles underlying the many MCMC algorithms. When constructing an MCMC algorithm (rather than merely applying one), the key thing is to design the transitional kernel (for continuous state space) in a way that the markov chain has a unique stationary distribution which is the posterior (or target distribution). An easy way to verify this is detailed balance equations. In other words, one wants to have the transitional kernel satisfy the detailed balanced equation with the stationary distribution being the posterior. The Random Walk Metropolis Hastings algorithm sketched in Section 3.3 satisfies this properly, as do many other MCMC algorithms.

## *3.6 The Role of the System Dynamics Model*

Where does the simulation model fit in to this whole approach? At base, it serves as a critical mediator between the parameters and the observed data, and thereby greatly eases the formulation of the likelihood function. Typically, in order to assess the likelihood that a given set of empirical data would have resulted from a particular value particular parameter vector, we need to reason about the emergent behavior of the model based on $\theta$. For most models, that capturing emergent behavior is going to require us to simulate the simulation model. In many cases, it may be a nonlinear simulation model, perhaps for examples, associated with infection transmission, the transition of norms and ideas, etc. And we are not going to be able to arrive an understanding of the emergent behavior – or its discrepancy from the observed data $y$ – without plugging in a value of data and running it, and then observing the consistency in a probabilistic sense – assessing the relative likelihood that that model output together perhaps with considerations of the actual values of parameters would yield the observed empirical data.

So the model helps us project behavioral consequences of parameter assumptions, in light of dynamic hypotheses captured by those models. We typically need the emergent output from the model given assumption of parameter values $\theta$ to assess the likelihood $P(y|\theta)$ that the observed data would occur for these parameter values.

# 4 Computational Effort

One thing to recognize with regards to this process is that running a simulation model often takes a very large amount of computational effort and regardless whether we accept or reject the candidate sample (thereby simply repeating the current sample), we are going to be running it. So to calculate these quantities, throughout this whole iteration, we are going to be running the simulation model for each iteration regardless of whether we accept or reject, and often we may run it hundreds of thousands of times to arrive at those sufficient set of samples from the posterior.

As noted above, we also typically run a burn-in period, which can amount to tens or hundreds of thousands of runs.

Fortunately, much of this work can be done in parallel – for example, by making use of different walkers so that we will have several walks of this sort going on in parallel, for example, on different cores, on different machines. This approach takes advantage of the "embarrassing parallelism" associated with random walk processes. While there are dependencies within a given use of this walker (i.e. later steps depend on earlier), but it is possible to have several walkers operating at different areas of the space simultaneously, each yielding samples from the posterior distribution.

# 5 Additional Practical Considerations

This section briefly highlights some practical concerns beyond the raw computational effort. The first issue is burn-in time. Typically we do need to run the simulation for long enough so that starts generating values approximately distributed as posteriors before initiating the recording of those values. Some insight into the required length can be assessed by examining the autocorrelation in sampled values over time.

A second issue is the acceptance rate. We don't want to have 1% only if a candidate samples is accepted. Now different authors will make different suggestions for what the ideal acceptance rate should be; one set of recommendations can be found in [1]. Others will argue for example for higher acceptance rates. It's not a correctness issue. This is an efficiency issue. You certainly want the acceptance rate to be high enough to explore so you don't just remain in the same space. That is, you do not want the algorithm to be stuck in a location that makes it extremely difficult to move anywhere, because you won't be getting samples that are representative for the entire space. Instead, you want well mix, with movement around the parameter space. But you don't want the acceptance rate to be so high that just going around in one area of space, without traveling elsewhere. So in another words, you want a large enough perturbation you get out of your comfort zone sometimes. For the algorithm presented – Metropolis Hastings with a Symmetric Random Walk – we want to tune the algorithm parameters such as the perturbation size used for the random walk, in other words to assess how far this $\Delta\theta$ brings us, how far do we move in $\Delta\theta$, till we find an suitable acceptance rate. Generally speaking, if the acceptance rate is too high, we wish to increase the size of the perturbation. If the acceptance rate is too low, we need to lower the size of the perturbation.

## 5.1 Handling High Parameter Covariance

Another practical concern relates to covariance in the posterior distribution. Such covariance can adversely impact the efficiency of the algorithm. If two parameters, for example, exhibit high covariance in the posterior distribution, essentially the random walk algorithm may needing to walk along a locally 1-D ridge in 2-D space, and that's going to take a lot of rejections to stick on that ridge. So there is going to be a lot of candidate samples that are found outside of the zone of high posterior probability (density), and end of up being rejected. If you have this high covariance between parameters and the posterior space, it really can lower the efficiency of the algorithm. There are several strategies that can be attempted to address this issue. The first – and ideal – strategy would be to make use of a different set of parameters for analysis. Often there is some latitude in choosing the set of parameters to consider. for example, there may be a variety of parameters that can be expressed as functions of one another, and any identically-sized subset those might be used.

A second strategy for dealing with such correlation, involves using regression on some initial outputs from MCMC on the original (untransformed) parameters to define a new set of transformed parameters. For example, if one is considering a parameter space consisting of ($p_1$,$p_2$) which exhibits high covariance, parameter $p_2$ might be expressed as a function of parameter $p_1$ plus some residual $\varepsilon$ (that is, $p_2=f(p_1)+\varepsilon$). A subsequent MCMC could then be conducted in a transformed parameter space where some of the parameters use residuals from the regression (that is, in a space involving $p'_2 = p_1 - f(p_1)$). In essence, we seek to remove the correlation with $p_1$ from parameter $p_2$ by operating in this transformed space.

## 5.2 Providing an Unbounded Space

Many algorithms implementing Markov Chain Monte Carlo algorithms in libraries such as MCMCpack either presuppose or make it much easier if you're operating in an unbounded space. So often what this leads to is, if you have a parameter that exhibits only bounded variation, you will transform it so that the transformed parameter varies without bounds (that is, between minus and plus infinity). Such a transformation often further allows for exploration of a larger range of dynamic values for the parameter, and saves some computational effort when compared to strategies that simply retreat regions outside the bounds as being of likelihood 0.

For example, suppose that we have a parameter only want to vary between 0 and 1 – say, the transmission probability beta within an infectious disease simulation model. In order to transform this parameter that so that its transformed counterpart is associated with an unbounded space, one might take a logistic transform of it. Or if we have a parameter that can only be zero and higher, one may want to take a log transform of it, so that the transformed value will vary from minus infinity to plus infinity.

## 5.3 Dealing with Correlated Samples

Another issue is that – particularly if one is seeking to use the successive samples in some serial way – one may really place a premium on reducing correlation between successive samples. If successive samples generated from the MCMC algorithm exhibit high degrees of serial correlation, they will not

represent legitimate *independent* samples from the posterior distribution –each successive sample isn't going to be telling you a lot independently about the distribution. So the *effective* sample size will generally be much lower than the nominal sample size. If you have high auto correlation between successive outputs from the simulation model, you are typically going to you need to sample further to secure the same count of effective samples. In some cases, we may wish to seek to reduce such autocorrelation to acceptable levels by performing a "thinning" the successive samples accepted by the MCMC algorithm. A thinning factor of *n* will only retain 1 out of every *n* accepted samples.

Such thinning is not required in all cases. Generally just you can sample a lot more, just be very aware you effective sample size maybe much lower than the nominal size (that is, than the count of numbers returned in the vectors). You may be really fooling yourself if you are treating them as independent samples.

How many MCMC samples are enough? [1] has some nice discussion using, between-walker and within-walker variance to assess how much will be gained by further sampling. Essentially, if within-walker statistics are similar to those between walkers, it can be a sign that you've done a pretty good job exploring the space for each walker (that the "mixing" has been fairly complete), and there's not much additional need to each walker further. Please consult with this valuable reference for greater detail.

The R Package *MCMCPack* has some nice tools for assessing effective sample size, plotting autocorrelations. The basic mechanism used for the sampling *metrop1,* also provides a way of thinning the results which is very common, only accepting one out of every *n* samples.

# 6 Other MCMC Algorithms

This article has it only highlighted some of the basics of MCMC algorithms in a highly preliminary and non-representative way. Readers should be aware that there are actually many variants of MCMC algorithms. These different algorithms offer different combinations of generality, simplicity, and sophistication. For example, some of the more advanced models – such as reverse jump MCMC – permit choosing between models with a different numbers of parameters that are applicable. Some algorithms allow for asymmetric random perturbations (in contrast to assuming symmetric perturbations). Other algorithms – such as Gibbs sampling – secure additional efficiencies. For example, Gibbs sampling can be used where you have conditional formulation of the model that you build up, using successive conditionals in a way that avoids the risk of rejection of samples.

# 7 Batch and Sequential Monte Carlo Methods

Monte Carlo methods offer different sorts of needs when it comes to incorporating new data. What we've been talking about here is an example of a "batch" MCMC method. Such batch methods are what might be termed "offline" methods by the standards of other algorithms – they process all required empirical data *y* at a single time to yield the resulting posterior. If new data arrives to be incorporated into *y*, we can run the entire algorithm again.

In some cases, data may be arriving on a frequent basis, and the computational cost involved may rule out the possibility of running a full batch MCMC on each new data point that arrives. Within such cases, *Sequential Monte Carlo* methods [3] permit us to more quickly update our previous estimate for the posterior to incorporate consideration of the new data. Such algorithms exhibit similarity to Extended Kalman Filtering (EKF) [2] in the sense they allow you to incorporate new data into previous estimates in a recursive fashion – you don't have to go back and recalculate the entire state estimate from scratch, in light of all data. In both techniques, you can use results (for Extended Kalman Filtering, point estimates and covariance matrices for states; for Sequential Monte Carlo methods, the posteriors) from the data you had earlier, and update those results in light of the new data that's available. However, Sequential Monte Carlo methods are far less restrictive in terms of their assumptions and far more extensive in their results than is Extended Kalman Filtering. Because they build atop the posterior estimates previously derived (rather than re-deriving it from scratch), the Sequential Monte Carlo techniques can be much more efficient than the sort of batch MCMC techniques looked here. The primary downside is that they are somewhat more restrictive, and can exhibit different types of pathologies. We anticipate describing the use of Sequential Monte Carlo methods with System Dynamics models in a separate, later, contribution.

# 8 Sequential Monte Carlo Methods

This paper has provided a very brief sketch of Bayesian estimation of System Dynamics modeling parameters via Markov Chain Monte Carlo techniques. Use of MCMC with dynamic modelers seems likely to rise rapidly over coming years in other areas of the modeling community, and System Dynamics modelers would do well to learn from this technique.

MCMC is a very rich, powerful and flexible technique, and one that's challenging to present in a brief fashion, and for brevity this article has had to go light on a lot of details and gloss over many things. Interested readers are strongly advised to consult the references below.

Despite the brevity of this treatment, we hope that you will find this material useful in your evaluation of the suitability of this powerful technique for incorporating rich statistical, statistical techniques with your dynamic models.

# References


1. Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2003). *Bayesian data analysis*. Chapman & Hall/CRC.
2. Gelb, A. (Ed.). (1974). *Applied optimal estimation*. MIT press
3. Doucet, A., & De Freitas, N. (2001). *Sequential Monte Carlo methods in practice* (Vol. 1). N. Gordon (Ed.). New York: Springer.