Design of Computer Experiments for Packet Communication Networks

Ben M Parker

June 2, 2009

1 Introduction

1.1 Packet Communication Networks

Packet communication networks transmit information from a sender to a receiver by gathering the information into small fragments called packets; these packets are sent across a transmission medium where they are reassembled by the receiver according to a pre-determined protocol. The exact mechanisms used for this transfer are beyond the scope of this work, but suffice it to say that packet communication networks account for the vast majority of information transferred through the Internet, and other communication channels.

In an industrial context, manufacturers of networking equipment and also Internet service providers expend a great deal of resources in trying to measure networks, both in order to assess any faults with a service, and to provide accurate service level information to clients for use in billing, or assessing success against contractual agreements for service provision, for example.

Perhaps strangely, there is little understanding of how to measure these communications networks on a fundamental level. In our work, we show that by regarding measurement procedures on packet communication networks as experiments that can be optimised in the traditional sense, we can improve current industrial best practice in measurement.

Consider, for example, the case of a broadband user who wants to confirm that his Internet service provider is providing the agreed bandwidth: having no privileged access to the system, he may only send exploratory or probe packets at whichever rate he chooses, and measure the time that these packets take to arrive at their destination. What rate best allows him to determine the bandwidth? The problem of users seemingly not getting contractually agreed bandwidth has been recently discussed in the popular media, e.g. [1].

Essentially here, we have a classic bias-variance trade-off. By increasing the rate of the probe packets, our broadband user will gathers data which allows him to determine the bandwidth more precisely (lower variance). Conversely, sending more probe packets means that there will be more packets in the system; the system may become overloaded, and probe packets may become delayed, which decreases the accuracy of the probing regime (higher bias).

1.2 Computer Experiments

Computer experiments are often used in applications where it is impossible or expensive to perform real experiments; for example in calculating climate change or modelling a nuclear fallout, or in more mundane situations such as weather forecasting. We have a set of inputs, made up of controllable factors and environmental (uncontrollable) parameters, and a set of outputs which represent the real data we wish to model. In general we seek to provide a simulation which approximates the true behaviour of the underlying model accurately and precisely. [7] and [8] provide a good introduction to designing computer experiments.

Typically in computer experiments, a complex system is modelled by a simulator. Each run of the simulator may take many hours, or days, such that experimental runs are expensive. We must therefore use our experimental time effectively, and in particular choose the inputs of our simulation such that we accurately explore the input space.

We typically investigate an appropriate choice of simulator inputs by constructing an emulator, a simplified model of the original system; this emulator is very quick to get results, enabling us to quickly explore how our output might vary with our inputs, and allow us to choose useful inputs for our simulation.

1.3 Packet Networks as Computer Experiments

We present here a way of regarding measurement experiments on data networks as computer experiments.

We regard a complicated computer network as a simulator. In both cases, we have a large number of inputs, some of which we can control, and we wish to find some information about the system from the responses. The difference between computer experiments and network models is that, typically, computer experiments take a long time for each run, whereas network models may only be valid for a few seconds (while user behaviour is constant).

In order to find out about how a simulator runs, we build an emulator. In order to find how a complex network operates, we approximate it as a queue or series of queues which we can then "emulate".

One important difference in emulation of networks is that is common within the electronic engineering community to use stochastic models of queues; networks are deeply stochastic, and it is common within the electronic engineering literature to use a stochastic emulator; we will adopt this convention here.

2 Previous Work

There seems to be little research on designs for stochastic simulation. Recently van Beers and Kleijnen[9] have made an interesting contribution in optimally (and sequentially) designing queueing experiments; they seem to take some inspiration from the field of computer experiments in studying the M/M/1 queue.

They choose an initial design, and perform a number of experiments (simulations) at each design point. From these, they take a mean of the outputs, and use these means to fit a Kriging model across the design space, where the model is fitted by treating the means at each design point as if they were the outputs of a deterministic process at that design point. They now take a candidate set of design points from a space filling design (they take all points halfway between existing design points as candidates), and resample (bootstrap) from the data, using the existing simulated points, to work out which candidate point has the highest prediction variance, and this becomes the new design point, where a number of new experiments are performed. In total n experiments (simulations) are performed, encompassing both the initial experiments, and those at the design points found by Kriging. The method compares well to a simple space-filling design for a low number of simulations ($n = 10$), but that there is no significant improvement over space-filling designs for larger designs ($n = 50$).

The technique is detailed, and has much to recommend it: it uses ideas from computer experiments in a stochastic framework. Potential criticisms are that the space-filling design candidate points seem to be fairly arbitrarily chosen, and the range considered for load is limited to $0.1 \leq \rho \leq 0.9$; we see later that considering high loads may be important. Van Beers and Kleijnen's method is computationally complicated, and may be difficult to implement practically for a "real-time" system, which may be desirable in network monitoring applications.

2.1 Emulation as a Markov Chain

In previous work[5], we are interested in measuring the probability that a network is busy at any given time. For complex networks, this is a difficult problem.

We therefore constructed an emulator of a system, which represents any complex network, where we model the network as a system with two outputs- an empty/full binary state. The system evolves, according to a transition matrix P which we parameterise via $P = \begin{pmatrix} 1-p & p \\ r & 1-r \end{pmatrix}$. Our question was then, given we can observe the system at discrete time points $0, 1, 2, \ldots$, and we have a fixed number of observations T , at what time points should we measure to allow us to make the most inference about our parameters under study (here p and r)? We concluded that monitoring as frequently as possible was sub-optimal, and presented methods for determining what the optimal rate was. Some current research involves generalising this to an arbitrary Markov chain with any number of unknown parameters.

3 Stochastic Emulation by the Regenerative Method

Let us suppose we perform an emulation where we have a series of longitudinal observations on data y_1,y_2,..., which depend on some experimental factors, $\mathbf{x}$, which we may vary, and some unknown parameter θ, which we wish to estimate by making inference from the y_i. Typically, these observations will be generated by stochastic simulation.

Suppose further that the y_i are autocorrelated, and thus the problem of inferring θ from the data is tricky. Conventional techniques include aggregating the data into batches, and using the means of the data as approximately independent statistics from which to make inference (Batch Means); running the experiment several times, and forming a summary statistic from each run (Independent Replications); or indeed some combination of these. (See e.g. [6])

The weakness of these methods is that the summary statistics for each run or batch are only approximately independent, and in practice choosing the batch size, run length, and burn-in parameters is often tricky. Large burn-in periods are often chosen which result in inefficient simulations. The question of how to choose these simulation parameters is an open question in the literature.

Other techniques include imposing a model for the autocorrelation at the analysis stage, and seeking to remove the autocorrelation by some transformation of the data. If this model is not known, or is dependent on the unknown θ, this approach may break down.

We propose to revisit an existing method, called the regenerative method, to show how we can impose normality on aggregate statistics found. By appealing to the normality of these statistics, we can then present an optimal experimental design for systems we can use the regenerative method for.

3.1 Overview of the regenerative method

Instead of correcting for the autocorrelation found by experiment using such a transformation of the data, we seek to provide a method for removing it from the data gathered by measuring the physical process a different way. We adopt the regenerative method. This idea was first suggested in 1961 by Cox and Smith[2], and a clearly presented review of the method was described in 1977 by Crane and Lemoine [3].

For a given simulation, we pick a regenerative state, which is a state that our system returns to with non-zero probability. We then form our statistic(s) for the experiment by making calculations on data taken from each regenerative cycle in turn, now regarding each regenerative cycle as the unit level at which we gather data. As the regenerative cycles are by construction independent and identically distributed, we have removed any dependence in our data.

3.2 The regenerative method

Let us suppose we wish to find an estimate of some unknown Y from a stochastic process , which we estimate as $\tilde{Y}$. We follow the following algorithm:

1. Observe the system for n regenerative cycles. (A regenerative cycle is the period of time between regenerative points occurring in our system)
2. Compute Y_k and α_k for each cycle k, where Y_k is the sum of the waiting times, $\sum Y_{ij}$, over the k-th cycle. α_k is the length of the k-th cycle.
3. Compute the following statistics:

$$\begin{aligned}
\bar{Y} &= \frac{1}{n}\sum_{k=1}^{n} Y_k; \quad \bar{\alpha} = \frac{1}{n}\sum_{k=1}^{n} \alpha_k; \quad \tilde{Y} = \frac{\bar{Y}}{\bar{\alpha}} \\
s_{11} &= \frac{1}{n-1}\sum_{k=1}^{n} Y_k^2 - \frac{1}{n(n-1)}\sum_{k=1}^{n} Y_k^2 \\
s_{22} &= \frac{1}{n-1}\sum_{k=1}^{n} \alpha_k^2 - \frac{1}{n(n-1)}\sum_{k=1}^{n} \alpha_k^2 \\
s_{12} &= \frac{1}{n-1}\sum_{k=1}^{n} Y_j\alpha_j - \frac{1}{n(n-1)}(\sum_{k=1}^{n} Y_k)(\sum_{k=1}^{n} \alpha_k) \\
s^2 &= s_{11} - 2\tilde{Y}s_{12} + \tilde{Y}^2 s_{22}
\end{aligned}$$

4. Due to the fact that the lengths of cycles α_i are independent and identically distributed (IID) for $k = 1, \ldots, n$, and the values of Y_k are similarly IID (although α_k is not independent of Y_k for fixed k) we can use the central limit theorem and form a confidence interval for Y as

$$\tilde{Y} \pm \frac{s\Phi^{-1}(1-\frac{\delta}{2})}{\hat{\alpha}n^{\frac{1}{2}}},$$

where Φ is the standard normal cumulative density function.

Note that the estimate produced may be biased, but asymptotically (if we take large enough n), the distribution for $\tilde{Y}$ is correct.

3.3 Calculating the Fisher Information

We can shown using the regenerative method that our data is distributed normally with mean and variance depending on the queueing system.

$$y_i \sim N(m(x_i), v(x_i))$$

In the general case, we may not know the form of these mean and variance functions $m(.)$ and $v(.)$.

We omit the derivation here, but it can be shown that our Fisher Information for some unknown parameter μ with data Y is

$$\mathbb{E}_Y\left[\left(-\frac{\partial}{\partial\mu} log[L(\mu; y_i)]\right)^2\right] = \frac{[m'(x_i)]^2}{v(x_i)} + \frac{[v'(x_i)]^2}{2[v(x_i)]^2}, \tag{1}$$

where $m'(x_i) = \frac{\partial m(x_i)}{\partial \mu}$, and similarly $v'(x_i) = \frac{\partial v(x_i)}{\partial \mu}$.

Due to additivity under expectation, if we perform the experiment more than once, we can form the Fisher information in performing multiple experiments at $\mathbf{x} = (x_1, x_2, ..., x_n)$ from equation (1) as

$$I(\mathbf{x}) = \sum_{i=1}^{n} \mathbb{E}_Y \left[\left(-\frac{\partial}{\partial \mu} log[L(\mu; y_i) \right)^2 \right], \tag{2}$$

which we can take the reciprocal of to find our Cramer-Rao Lower Bound.

4 Example:The M/G/1 queue

Let us suppose that we have a complex problem, and for the sake of demonstration of our regenerative method, we emulate this as a single server queue, where arrivals occur such that the inter-arrival times are exponentially distributed with parameter x. Customers are served in the order they arrive, and if the server is busy, they must wait until the customer who arrives immediately before them has been served. The customers are served at mean rate μ. The service protocol may be further specified, but in general this is an M/G/1 queue.(If the service time is fixed, this is the M/D/1 queue. If the service time is exponentially distributed, this is the M/M/1 queue.)

We can the regenerative method to the M/G/1 queue. Crane and Lemoine show ([3], section 3.5) that as the number of regenerative cycles tends towards infinity , the choice of regenerative state is irrelevant to the confidence interval obtained. A natural regenerative state to choose is the one in which the queue is empty after a packet is serviced, as we know that all queues with $\rho < 1$ return to this state with non-zero probability (and indeed for the M/D/1 queue spend a proportion of their time equal to $1 - \rho$ in this state).

We find by simulation that the regenerative method is effective in practice in producing data with the correct mean, and which are also normally distributed. This simplifies our design problem as we now know by design that the error distribution of our data is normal.

We now consider the design problem for the case where the service rate μ is unknown, and we with to estimate it. We initially set our arrival rate x_i for one simulation (experiment), which will give us one data point y_i formed from the regenerative method; we can then form an estimate of μ for this simulation using an appropriate estimator.

For an M/G/1 queue, we find the Pollaczeq-Khintchine formula for the mean of the waiting time[4] as

$$m(x_i) = \frac{1}{\mu} \frac{x_i}{\mu - x_i} \frac{1 + (x_i \sigma_s)^2}{2} + \frac{1}{\mu}$$

where σ_s is the standard deviation of the service time such that, for example, $\sigma_s = 0$ for the M/D/1 with constant service rate, and $\sigma_s = \frac{1}{x_i}$ for the M/M/1 queue with exponentially distributed service times.

For the M/D/1 queue, with deterministic service time we can show

$$m(x_i) = \frac{x_i}{2\mu(\mu - x_i)} + \frac{1}{\mu} \tag{3}$$

and hence

$$m'(x_i) = \frac{x_i(x_i - 2\mu) - 2(\mu - x_i)^2}{2\mu^2(\mu - x_i)^2}. \tag{4}$$

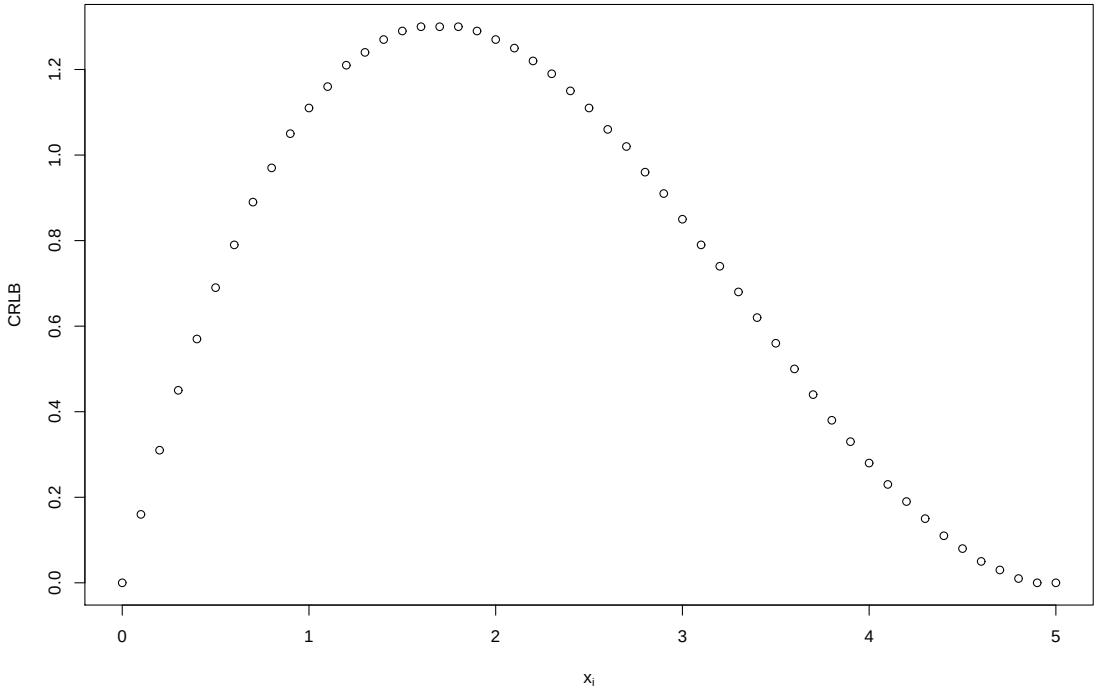
Similarly

$$v(x_i) = \frac{4x_i\mu - x_i^2}{12\mu^2(\mu - x_i)^2} \tag{5}$$

and hence

$$v'(x_i) = \frac{x_i(-6\mu^2 + 4\mu x_i - x_i^2)}{6\mu^3(\mu - x_i)^3}. \tag{6}$$

Figure 1: CRLB against x_i for $\mu = 5$



Substituting equations (3) and (5) into equation (1) provides the CRLB for the M/D/1 case with one observation. We can now see how the CRLB varies with x_i which we plot graphically as Figure 1, demonstrating for $\mu = 5$.

This shows clearly that the optimal design in the M/D/1 case is to take our observation at the point $x_i = \mu$ or at $x_i = 0$. Both of these are a physical nonsense; in the former, we take experimental readings at μ, which we are trying to determine by experiment! In the latter case, taking $x_i = 0$ would mean we gather no data as we are not inserting any packets into the queue.

Furthermore, performing more than one experiment would not make this difficulty disappear: from equation (2) it is clear that the optimal design is to take all observations at the start or end of the range.

Our result, although not practical in terms of finding an optimal design, agrees with our intuition; if we wish to find μ with certainty, we can either send packets into a queue as far apart as possible so that they are almost certain to reach the queue when the buffer is empty, meaning the length of time in the queue is simply the service time. We can thus determine this entirely accurately in this simple M/D/1 case where the service time is deterministic. Alternatively, we put packets into the queue as fast as possible, such that the queue is likely to become full. We can then calculate the service time from the difference between exit times of the packets.

4.1 Applying the results in practice

There are some criticisms when using the regenerative method as we have.

Firstly, all of the theory used above only applies to queues in equilibrium; the queueing theory used is not valid for $x_i \geq \mu$, and developing a general theory for this case is tricky as results will depend on the simulation parameters, such as the length of the simulation and the number of regenerative cycles examined. More simply, there will not be a stable mean to measure. As we do not know what the value of μ is before performing the experiment, we may have difficulty in constraining x_i to the appropriate region where the theory is valid. However, a similar criticism applies to other methods in the literature, and this we are generally only interested in measuring queues in equilibrium.

There are other criticisms sometimes made of the regenerative method. It is difficult in some circumstances to identify a regenerative point, although we know that queues operating with arrival rate less than their full capacity μ will always return to zero, so this is not a problem here.

A related criticism is that regenerative cycle times can be long; especially at high loads, it may take a very large amount of time to perform (or to simulate) enough regenerative cycles from a queue to ensure normality.

Note that the approach described in this work is not a sensible technique for evaluating μ in the specific example of an M/G/1 queue, A better estimator is to take the minimum time spent in the system by any packet, as each packet has a $(1 - \frac{x}{\mu})$ chance of finding the queue empty. If a packet enters a queue which is empty, it will proceed immediately to service and thus have a system time of $\frac{1}{\mu}$. We could thus find the waiting time with m packets with a $(1 - \frac{x}{\mu})^m$ level of confidence, which is maximised as $x \to 0$. We have presented the likelihood method of taking aggregated averages here as this would be a standard technique with more complex queues, and to demonstrate opportunities with the method.

The reader may question how [9] is able to present optimal designs. Firstly, a different question is being asked. The research is trying to find a regression model within a fixed class that accurately explains the behaviour of the queue, for example in CT-TH curves in industry. The regression problem is different, but related, to trying to estimate the parameters of the queue, on which the model will act. Also, the paper restricts the design space to $0.1 \leq \rho \leq 0.9$. Clearly, within these constraints, our method would put the optimal design point at one of the two limits of the design space.

5 Conclusions

We have described problems that emulation using simple Monte Carlo simulations may lead to, in particular when separating the effects of design parameters and simulation parameters. Although there are some criticisms, the regenerative methods seems to solve many of these problems.

This approach would seem to be a valuable method to use in finding designs with other stochastic emulations, which has not been considered in previous research on designed experiments.

This works seeks to explain how we can use some ideas in computer experiments to perform effective measurements on networks. We also have sought to introduce the regenerative method as one potential way of dealing with stochastic simulations for computer experiments.

For further work we intend to compare the total experimental effort from standard Monte Carlo techniques with that expended in the regenerative method; a regenerative cycle may be very long and it would be interesting to take the time to run this emulation into account.

References

[1] BBC. Broadband speeds under scrutiny. `http://news.bbc.co.uk/1/hi/technology/7003113.stm`, 2007.

[2] D. Cox and W. Smith. *Queues*. Methuen, 1961.

[3] A. A. Crane and J. Lemoine. *An Introduction to the Regenerative Method for Simulation Analysis*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1977.

[4] L. Kleinrock. *Queueing Systems: Theory Vol 1*. John Wiley & Sons Inc, 1975.

[5] B. Parker, S. Gilmour, and J. Schormans. Measurement of packet loss probability by optimal design of packet probing experiments. *IET Communications*, 3, 2009.

[6] K. Pawlikowski. Steady-state simulation of queueing processes: survey of problems and solutions. *ACM Computing Surveys (CSUR)*, 22(2):123–170, 1990.

[7] J. Sacks, W. Welch, T. Mitchell, and H. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

[8] T. Santner, B. Williams, and W. Notz. *The Design and analysis of Computer Experiments*. Springer, 2003.

[9] W. van Beers and J. Kleijnen. Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping. *European Journal of Operational Research*, 186(3):1099–1113, 2008.