DETERMINISTIC MODELS

Where the output of the model (the final system state) is solely dependent on the input (the initial system state). Thus every time the model is run it will produce the same result for the same input. In system engineering deterministic models produce definite results that are easier to handle in terms of designing the system and therefore are generally preferred if they can be justified.

STOCHASTIC MODELS

A stochastic model contains some probability function so the output system state is not solely dependent upon the input system state but also on some random component with in it (the opposite of Deterministic models). Thus it will not produce the same result every time it is run even if the input state is the same.

Since the output of a stochastic model is a probability function it makes handling the design that much more difficult so systems engineers looking for an easy life try to avoid them. However real life is often stochastic and the use of probabilities within a model is often unavoidable if it is to represent the system accurately. This is particularly true if humans are involved in the modelling.

Historical Note: The word stochastic is from the Greek word for skill at aiming a bow and arrow - presumably leading to a probability function?

DEALING WITH UNCERTAINTY

As we have seen generally it is a good idea to neglect the random (stochastic) nature of real systems in modelling and use deterministic models to support analysis and design activities. For the perfectly acceptable reason it makes life easier.

However if the stochastic nature of the system (or the environment in which it operates) significantly impacts its performance that must be included in the modelling in some way. The options are:

Budget Margins.
We have already seen the use of margins within budgets to allow for uncertainties in systems design and performance.

Worst Case Analysis.
Run models with values that represents the extremes of probability function.

Monte Carlo Analysis.
With either random inputs to a deterministic model or fixed inputs to a stochastic model.

Direct Analysis.
Mathematically combine the probability functions in the model to obtain total model probability function using techniques such as Markov Chains.

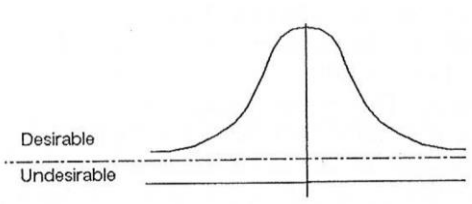
This lecture deals in more detail with handling uncertainty with Stochastic Models.

WORST CASE ANALYSIS

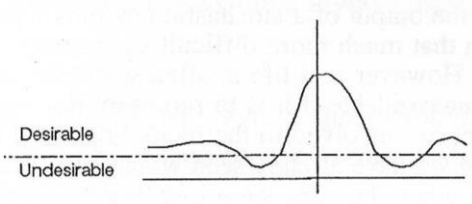
This approach requires a good definition of the probability functions within the model.

The model is then run with a deterministic function in place of the stochastic probability function which corresponds to an extreme probability (e.g. 90% or greater) - note it is not strictly "worst case" as there is always a probability it could be worse in any probability distribution.

Danger: If the model has its extremes of behaviour within the middle of its probability bands rather than at the edge then worst cases will not in fact be found. Therefore there must be a good understanding of the models behaviour across the range for examination of the extreme probability ranges to work.



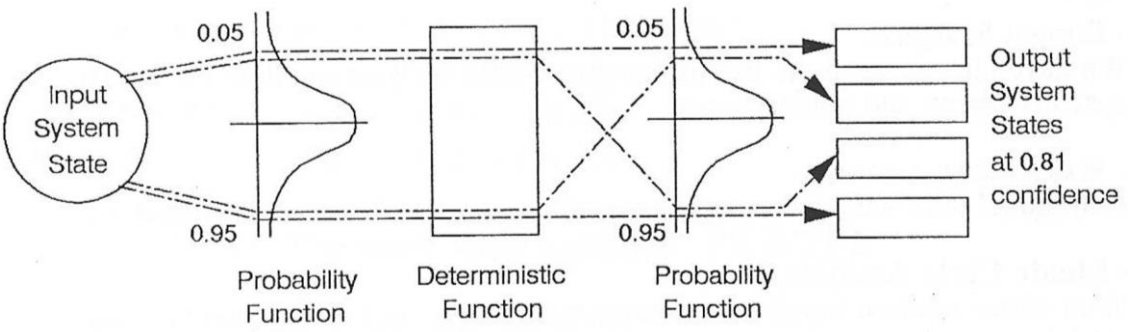
OK Extremes are the worst case



Problem - Extremes are not the worst case

CHAINING FUNCTIONS

A model may have more than one probability function in it. Worst case analysis can still be performed as shown in the example. There will be 2n paths (n = number of probability functions in the model) through the worst case (assuming both extremes are of interest) and every output goes to both the next functions extremes.



Note how the confidence level at the end is the probability of staying within the range of all the probability functions i.e. the more probability functions involved, the lower the probability the analysis will be valid in any real life state.

The Deterministic function is just there to show you can mix deterministic and stochastic functions.

MONTE CARLO ANALYSIS

Where a proper analytical understanding of the models between the extremes is not possible (or just too much work) an alternative is to just keep running the model again and again. Over many runs the results will form a Probability Function. One simply has to obtain a statistically significant number of results to obtain the probability function to the required accuracy.

This is based on something called Ergodic Theorem which says that the probability distribution over many runs will be equal to the probability distribution of a single run over time.

In complex models this is by far the easiest approach (thanks to computers) and given enough inputs is always convincing.

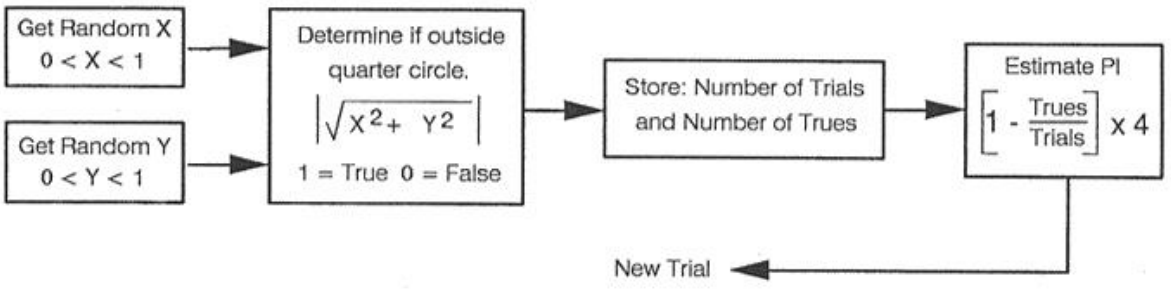
HISTORY OF MONTE CARLO ANALYSIS

The theory was developed by the earliest of computer pioneers Von Neumann, Ulam and Metropolis at the end of the Second World War. This was part of the work on atomic bombs as a means to model the fission and fusion processes. The first computer based Monte Carlo runs were at Los Alamos in 1953 on very early computers. Note the first uses were really more science than engineering, and the technique is still more widely used in science than engineering.

Monte Carlo Example

Using a Monte Carlo model to get an estimated value for Pi.

Method: Randomly generate points in a unit square and see if the point lies beyond a unit radius. Collect statistics for number of trials which are outside and use that to estimate ratio of circle area to square area and hence Pl.

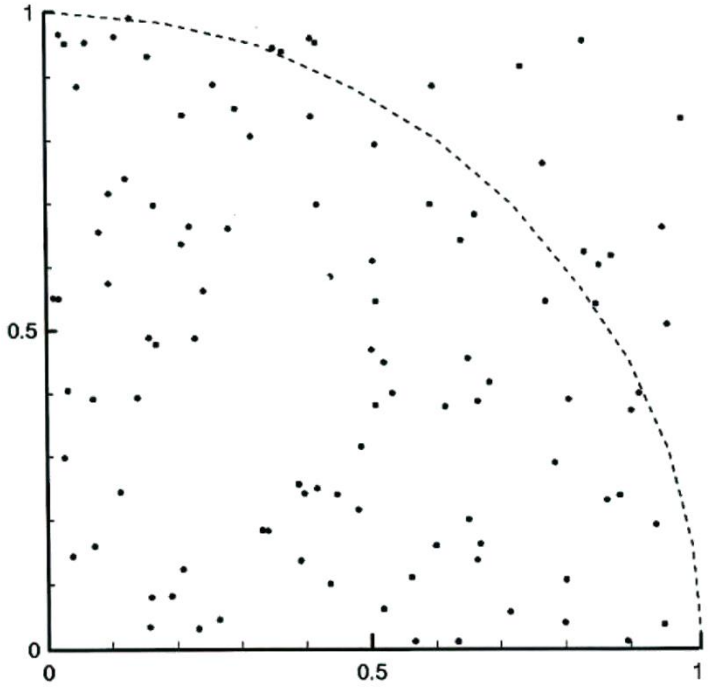


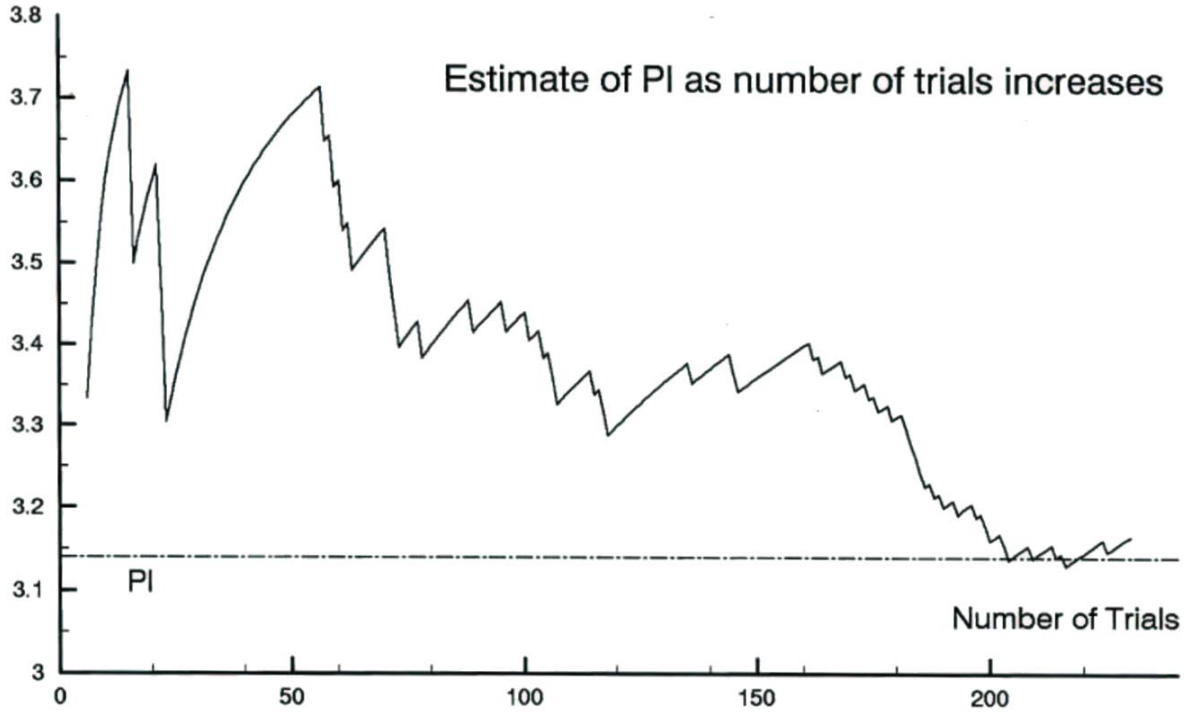
Warning: This is only an example, not a sensible way to find Pi!

Position of the first 100 trials to determine PI using a Monte Carlo model.

Overall 237 trials were conducted to obtain an estimate of PI.

237 was limit of spreadsheet capacity.





Final estimate after 237 trials was 3.190 to an accuracy of around 0.1

RANDOM NUMBERS

Monte Carlo analysis depends upon the generation of random numbers within the computer. These are normally generated by using Pseudo-random numbers generated by a series like:

$$I_{i+1} = a.\ (I_i - [I_i/M].\ M)$$

Where a and M are large non-negative integer numbers and [ ] denotes the integer part of the division result. This results in Pseudo-random integers between a and a. M all divisible by a. Therefore random numbers (Xi) in the range 0 - 1 can be obtained from:

$$X_i = I / (M\ .\ a)$$

As an example if a = 13, M = 89 and the "seed" Io = 5 then we get the sequence

I = 65 845 572 494 637 182 52 676 689

hence

X = 0.056 0.730 0.494 0.427 0.550 0.157 0.045 0.584 0.596

The sequence repeats after M (89) numbers which is the best that such series can provide.

In practice on computers M is the largest number the machine can hold and a is also very large. As a result and careful selection of the "seed" very good Pseudo-random series can be generated.

Warnings on the Use of Random Number Generators:

- The quality of random number generation varies between languages and spreadsheets etc. In some cases when many random numbers are needed the results can be significantly affected by the inherent bias within the Pseudo-random numbers generator.

- In some cases the language or program always starts from the same point and therefore whenever the program is run the random numbers appear in the same order. Some languages offer a command like "RANDOMISE" to enter the generator at some random point. Alternatively it may be necessary to produce some randomising routine within the code.

3 – If you are trying to be the next lottery winner somebody will almost certainly be using the same random number generator as yourself and get the same "random" numbers and you will have to share the prize.

GETTING THE RIGHT DISTRIBUTION

The numbers produced by a computer language or spreadsheet are almost universally an even distribution between 0 and 1.

However the input in real life is nearly always some probability function (Poisson, normal, etc.). So code is often needed to turn the uniform distribution into the actual distribution of the input.

e.g. A method to get a Gaussian or Normal distribution.

(with Thanks to Dr Abhijit Guha)

Get 2 Random Xs (X_1,X_2) from 2 x RANDOM(i) - 1.0

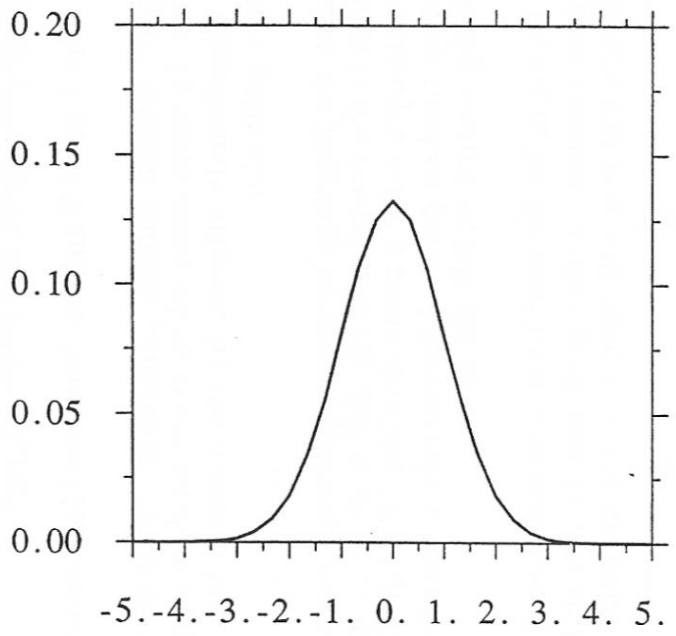
Get a composite random X^2 (Xc^2) from $X_1^2 + X_2^2$

Check if Xc is >= 1 or = 0: If yes get more numbers

Get factor (F) from SQRT(-2 x Ln(Xc^2) / Xc^2)

Get two new Random Numbers $Xg_1 = X_1 \times F$ and $Xg_2 = X_2 \times F$

The Xg's will have a normal distribution with a mean of 0 and a Variance of 1



NORMALLY DISTRIBUTED RANDOM NUMBERS GENERATED FROM UNIFORMLY DISTRIBUTED NUMBERS.

UNIFORMLY DISTRIBUTED NUMBERS 2×10^6
THOSE WITHIN UNIT CIRCLE 1.57×10^6

DR GORDON REECE'S GENERAL METHOD

You need to know the Probability Density Function over the range of numbers (X) you are interested in (to get a given confidence level for example).

METHOD

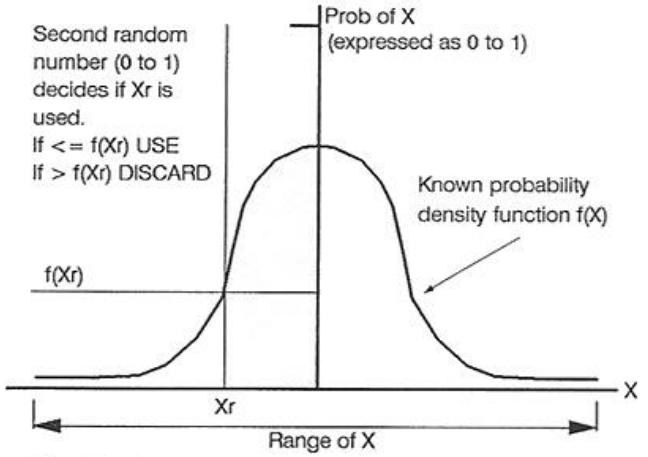
Generate first random number

Normalise it over the range of interested [Xr]

Find probability of Xr [f(Xr)]

Generate second random number between 0 and 1 [Decider]

If Decider > = f(Xr) Then use Xr otherwise discard



Xr = First Random Number normallised over desired range of X

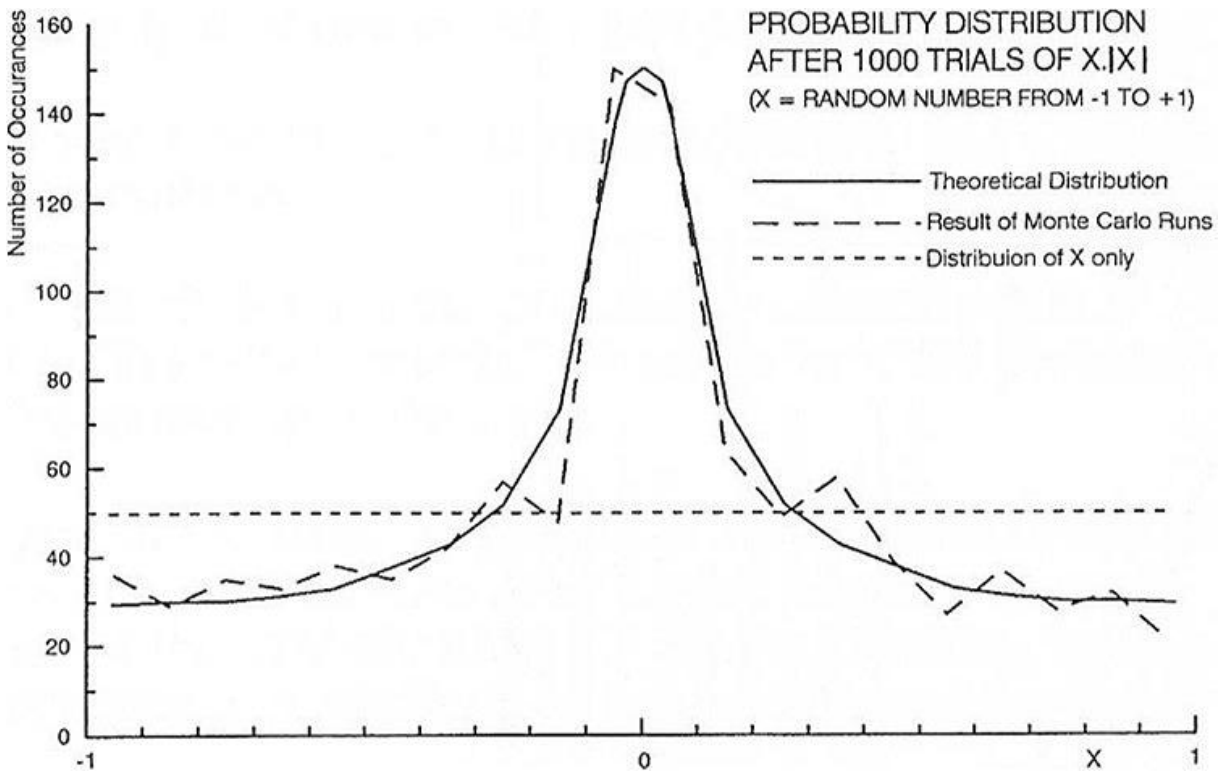
This method is code efficient and can be used to generate any definable probability distribution. It might prove a little slow as, at best, less than a quarter of the Random numbers generated by the Pseudo-Random routine contribute numbers to the distribution. The actual efficiency in use of the generated random numbers depends upon the selected range and tends to zero as range tends to infinity. So the range needs to be chosen with care.

MY OWN LITTLE DODGE

This trick does not produce a proper Normal Distribution, but it is easy to implement and if a precise probability distribution is not an issue - say in a training simulation - it is better than a uniform distribution obtained by the random number generator (actually what I used it for was games programs.)

Assuming the random numbers X_r have been spread about zero (in the example here between -1 and +1). Then multiplying X_r by its absolute value gives an X_r^2 distribution but with negative numbers mapping to negative numbers.

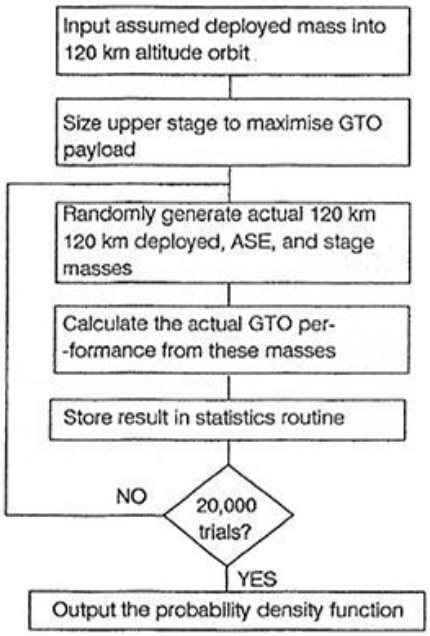
The graph here shows the theoretical probability density for 1000 trials and superimposed is an actual Monte Carlo run with 1000 trials.

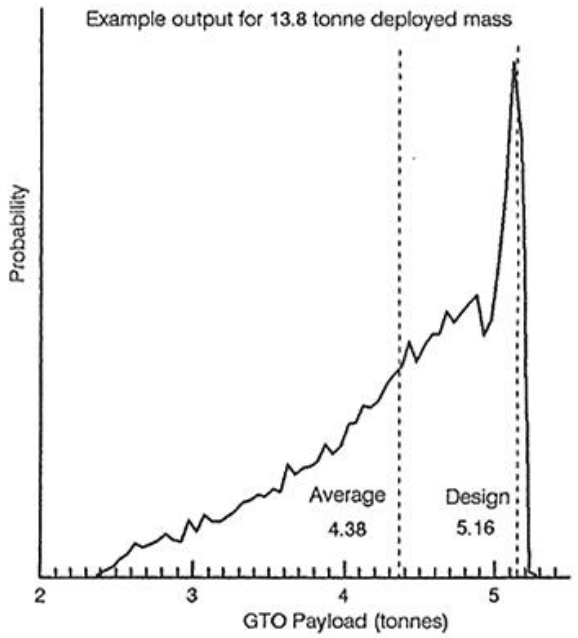


EXAMPLE OF MONTE CARLO ANALYSIS

This model was used to find optimum sizing of a launch vehicle upper stage given when there are uncertainties in the launch system performance.

MONTE CARLO MODEL OF STAGE PAYLOAD PROBABILITY DISTRIBUTION





Monte Carlo modelling enabled real life nonlinear factors to be included.

Note that despite using 20,000 trials the resulting plot still has a lot of random noise on it. The examples below uses half a million trials and gives a smooth plot.

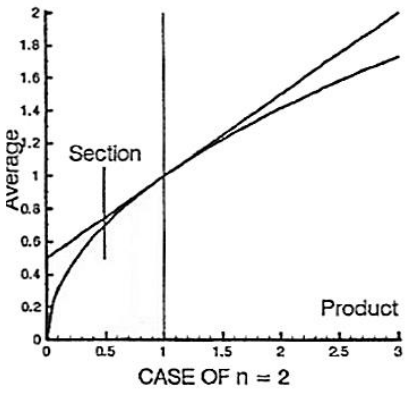
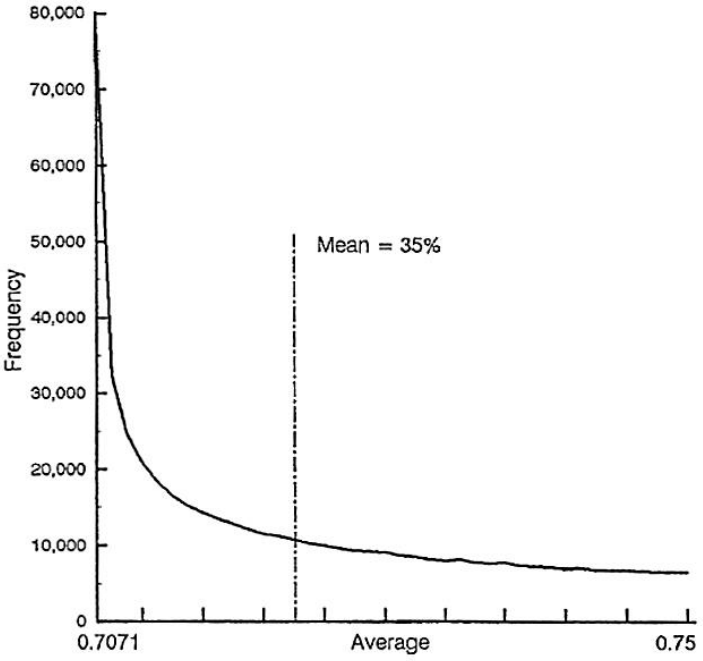


Figure 2a: Upper and Lower Boundaries For Average of Two Factors Against Product.

Figure 2b: Section at Product = 0.5 Showing The Distribution of Averages Between The Boundaries As Determined By Monte Carlo Modelling.



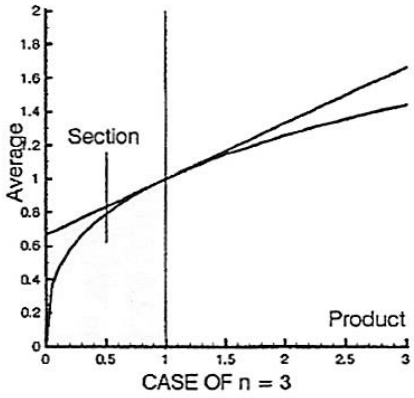
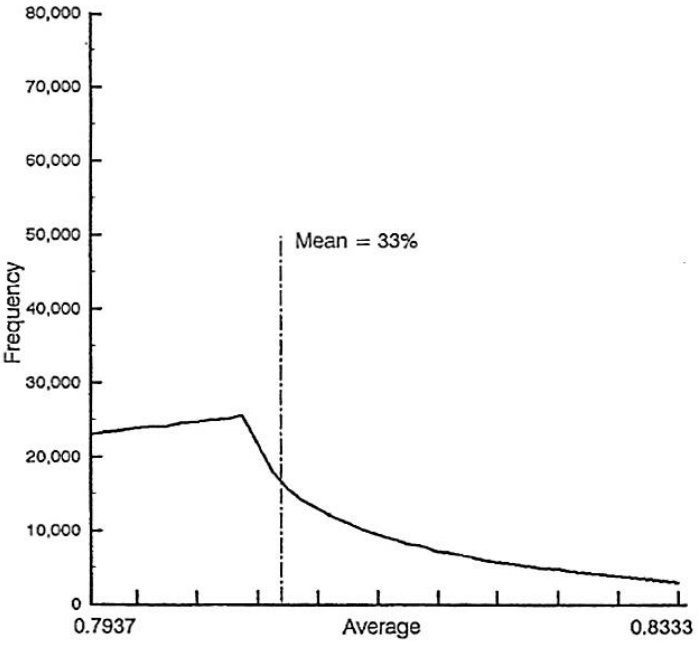


Figure 3a: Upper and Lower Boundaries For Average of Three Factors Against Product.

Figure 3b: Section at Product = 0.5 Showing The Distribution of Averages Between The Boundaries As Determined By Monte Carlo Modelling.



But it highlights a problem with Monte Carlo modelling as the analytical reason for discontinuities with n<2 are not yet known.

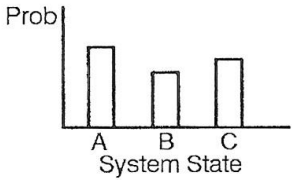
DIRECT ANALYSIS

The Worst Case and Monte Carlo approaches to stochastic modelling both avoid actually mathematically handling the probability function itself. - Generally a good idea, given how difficult probability mathematics can become.

However there is a body of mathematics that enables defined probability functions to be combined to produce a total system behaviour.

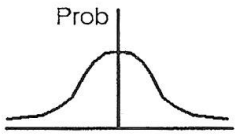
Two sorts of Probability distribution Continuous and Discrete.

Discrete



$$\sum_{i=1}^{n} P_i = 1$$

Continuous



$$\int_{-\infty}^{\infty} P(x)\,dx = 1$$

We will look at one example of combining discrete functions: Markov Chains.

MARKOV CHAINS

A **Markov Chain** or **Markov Process** (there is a theoretical difference dependant on precisely what is being modelled and how – but for our purposes they are the same). These are where the probability of a certain system state as an output depends upon the system state inputted and there is a number of such probability functions in the model with the output of one as the input of the other.

Markov Chain analysis works when the system states are discrete, and a matrix defining the probabilities can be built up.

Markov Chains use a "vector" which defines the probability of each of the system states at any time and a "Transition matrix" which defines the probability of the various system states dependent upon the input

A Markov Chain can be looped in the same way as a dynamic model. Thus it can be used to establish the various probabilities after any number of runs through the model. In many cases the Probabilities will reach a limiting value as time -> ∞ enabling time independent predictions on probabilities of system states to be made.

MARKOV CHAIN - SIMPLE EXAMPLE

e.g. For a two state system; called A and B, one can define a matrix of transition probability output dependent upon input called a transition matrix.

If input state is A then output is 0.6 chance of remaining A and 0.4 chance of going to B

If input state is B then output is 0.8 chance of going to A and 0.2 chance of remaining B

This produces the Transition Matrix P

Input / Output	A	B
A	0.6	0.4
B	0.88	0.2

(note: input rows must always total 1)

We can also define a probability vector (PA, PB), Where PA is the probability of finding the system in state A and PB is the probability of finding the system in State B. (note: The sum of vector probabilities must always equal 1)

Start with input state A defined by a vector of the probability state (1, 0) which is multiplied by the Transition matrix

Then output is Probability vector (0.6, 0.4), this can then be fed back for the next transition giving an output of (0.68, 0.32), Repeating again gives (0.664, 0.336). So the system seems to be converging on (2/3, 1/3)

This result can be confirmed by analysis. In the limit if the function is converging the n^{th} (PA, PB) must equal the n^{th}+l (PA, PB). Therefore:

PA = 0.6 PA + 0.8 PB PA (1 - 0.6) = 0.8 PB

PB = 0.4 PA + 0.2 PB PB (1 - 0.2) = 0.4 PA

i.e. They are the same basic equation PA = 2 PB

Given PA + PB = 1 then PA = 2/3 and PB = 1/3

Markov Chains can also be used to sum together several transition matrices as all must be n sided square matrices where n is the number of possible system states. Multiplying the matrices together produces a single n by n matrix with can be solved as shown above.