INT. J. BIOL. BIOTECH., 17 (2): 195-218, 2020.

RANDOM NUMBERS AND MONTE CARLO SIMULATION: APPLICATIONS IN GENETIC DRIFT AND RANDOM WALK MODELS OF PREDATOR-PREY INTERACTION

S. Shahid Shaukat[1*], Asma Zafar[2], Noreen Noor[3] and Moazzam Ali Khan[1]

[1]*Institute of Environmental Studies, University of Karachi, Karachi-75270. Pakistan.*
[2]*Department of Mathematics, Sir Syed University of Engineering & Technology, Karcahi-75300, Pakistan.*
[3]*Bahria University, 13 National Stadium Road, Karachi-75260,Pakistan.*
*corresponding author: syed.shaukat2004@gmail.com

ABSTRACT

The paper attempts to briefly review the methods of random number generation and such numbers from some commonly used probability distributions such as Normal, Binomial and Poisson distributions in biological and particularly in ecological work. Besides, Monte Carlo simulation theory is briefly introduced. The Monte Carlo simulation is applied to i) Genetic drift and ii) Random walk models of predator-prey interaction. Simulation of genetic drift is accomplished for haploid and diploid populations. In addition, genetic drift is also examined together with mutation. For a high mutation rate an example of peppered moth *Bistson betularia* is chosen in which mutation occurs from typical form c (*typica*) to C (*carbonaria*).Computer programs were developed for linear congruential and multiplicative congruential methods of random number generation and the two methods were tested and compared. Both linear congruential and multiplicative congruential methods of random number generation gave almost equally good results in terms of period length, uniformity of distribution and with regard to other tests of randomness. Monte Carlo simulations of genetic drift for both haploid and diploid populations showed that smaller population size showed marked fluctuations and greater number of fixation (as well as loss) of alleles either A or a. Genetic drift scenarios incorporating high mutation rate (0.041) gave upward trends of increasing frequency of *carbonaria* C gene with increasing number of generations suggesting predominance of *carbonaria* population with the passage of time. These results support the field studies. The random walk model of predator-prey interaction involves random movement of predator (random walk) on a square grid having a prey population ($N \geq 100$) where they search, encounter and capture (or fail to capture) the prey. These stochastic events are simulated through a Monte Carlo technique suggested earlier with modifications. The method involves defining an $r \times r$ grid that contains the uniformly distributed prey population. The predators (a few in number) walk over the grid randomly feeding on the prey when encountered and captured. It is assumed that the predator is unable to capture and eat all of them in a cell. Each prey has a certain probability of escape or hiding from the predator. The probability of capture PC is fixed at the beginning. Furthermore, the maximum consumption of the predator in one cell (C_x) is also fixed prior to simulation. The predator is allowed to move on the grid in all four directions of movement with equal probability. When the maximum time units of the simulation are completed, average and standard deviation of both encounter and consumption are computed for each of the K predators. The simulations gave some interesting results with regard to encounter, consumption and escape of prey. The random walk stochastic model given here is certainly an improvement over the deterministic model since the former captures a number of stochastic events involved in the process beginning with the movement of predator to the encounter,capture (or escape) and consumption of the prey.

Kay-words: Random numbers, genetic drift, simulation, predatory-prey interaction.

INTRODUCTION

Random numbers generation (RNG) are the basic ingredients for performing stochastic simulation and for the development of Monte Carlo procedures (O’Neill, 2002; Robert and Casella, 2004; DelMoral *et al*., 2006; Kroese, 2011; Acevedo, 2012; Thomopoulos, 2013; Kroese *et al.,* 2014; Frellsen *et al*., 2016). Simulation involves experimenting with a model and observing the consequences. Monte Carlo methods utilize random numbers for various stochastic events in the concerned model. Monte Carlo simulations are quantitative models that predict each outcome in the model and its probability. There exists a wide range of applications of Monte Carlo methods in the field of genetics and ecology (Swartzman and Kaluzny, 1987; Mode *et al*., 2011; Sanford and Nelson, 2000; Nuin and Otto, 2000; Stewart, 2006; Acevedo, 2012; Rubenstein and Kroese, 2016). Thus the generation of random numbers and their quality is of paramount importance in these fields. In general, a sequence of random numbers which are real numbers is generated that behaves like random, unpredictable variable within the range 0 and 1 as independent and identically distributed (i.i.d.) variate from a uniform distribution, usually abbreviated as U[0,1] (L'Ecuyer, 1990, 1994, 1996; L'Ecuyer *et al*., 1993; Schneir, 1996; Knuth, 1997,1998; Law, 2014). Other probability distributions can be generated by transforming the uniform random variate and applying appropriate methods to convert these to random numbers from required probability distributions.

Random numbers can be generated in fundamentally two different ways: 1) by a physical phenomenon that is considered to be random. A physical random number generator can be based on an essentially random atomic or subatomic physical phenomenon that are perceived to be unpredictable. The sources of randomness include radioactive decay, thermal noise, shot noise, noise in Zener diodes, radio noise, etc (e.g. Intel's hardware generated random number). These physical phenomena generally introduce asymmetries and systematic biases with the result that they are not uniformly randomly distributed. We often call these as hardware generation of random numbers, while the other alternative procedure 2) involves some deterministic computational algorithm. The latter sequence of computer generated random numbers is not truly random, since each number is completely determined from the previously generated number. Therefore, these are often referred to as pseudo-random numbers as they are necessarily generated by a deterministic procedure. Thus random numbers generated by any computer program are likely to be pseudo-random numbers, not 'actual' random numbers though they are apparently random. This could be an important drawback when simulations being performed are sensitive to subtle patterns in the "random" numbers employed, but for most demonstration programs or for class-room use this is not a serious issue. In fact, this process of computer generated random numbers is popularly used by programmers. Some of the characteristics of a random sequence are statistical that may be readily evaluated by suitable statistical tests that disclose the quality of such numbers. In this paper, Pseudo-random numbers are employed in the suit of computer programs presented. The prime source of randomness in pseudo- random number generators (PRNGs) is the initial fixed value, called seed, which is extended by means of a recursive formula for the generation of a sequence of random numbers; the entire procedure can readily be performed by developing a suitable software. The reproducibility level depends on the unpredictability of the seed value, which is usually taken from a true random sequence. Because of some good qualities of these random numbers such as high speed of generation , good statistical properties and no additional requirement for hardware/physical devices makes these generators very handy in Monte Carlo simulations and are most widely utilized by biologists. Many algorithms for generating pseudo-random numbers are available that automatically generate long sequences of random numbers of good quality but eventually the sequence repeats (Anderson, 1990, Brent, 1992, 1994; Zeeb *et al*., 1999). Apparently, the earliest method is the Mid-square method (von Neuman, 1951). It begins with a four-digit positive integer $Z_{0..}$ This is squared ($Z_0^2 = Z_0 \times Z_0$) to get an eight digit integer number and the middle four are chosen which are random numbers and provide the next four digits for the next random integers. The process can be expressed by an equation as follows:

$$X_{n+1} = [X_n^2 / b^a] - [X_n^2 / b^{3a}]b^{2a}$$

Where X_n is a 2a digit number, X_n is squared to yield a 4a digit value then 'a' digits are removed from right and left retaining the middle 2a digits which gives X_{n+1}. The numbers are then scaled to return random numbers in the interval [0,1].

While simple to implement it has several drawbacks; the random numbers tend to zero and the period is short. One of the most common PRNG is linear congruential method, (LCM) which uses the following recurrence formula to generate numbers.

$$X_{i+1} = (aX_i + c) \bmod m \quad i=1,2,3,\ldots..$$

$$m > 0,\ a < m,\ c < m,\ x_0 > 0$$

where*a*, *c* and *m* are large integers, X_{i+1} is the next in *X* as a series of pseudo-random numbers. The values of a, c and m determine the characteristics of the sequence. The maximum number of random numbers the formula can produce is one less than the modulus, *m*-1. When c = 0, we have the special case of Park-Miller algorithm or Lehmer algorithm (see Park and Miller, 1988).

$$X_{i+1} = (16807X_i + 0) \bmod 2^{31} - 1 \qquad i=1,2,3,\ldots..$$

To avoid certain non-random properties of a single linear congruential generator, several such random number generators with slightly different values of the multiplier coefficient, *a*, can be used in parallel, with a "master" random number generator that selects from among the several different generators. The choice of the values of a, c, m and X_0 (the seed value) are known to have a great impact on the statistical properties and on the length of the period. A good choice of these constants can lead to a good quality sequence while a poor choice would necessarily yield a poor quality sequence. Integer pseudo-random numbers are generated within the interval [0,m-1]. The integers X_i are readily transformed to [0,1] as $R_i = X_i / m$. The modulus m should be sufficiently large as the period cannot be larger than m, infact it is m-1. Usually it is chosen as the largest number that can be represented in a computer, e.g. for a 32 bit machine it would be 2^{31}. The sequence will have a period m under the following conditions : i) c is relatively prime to m, ii) a-1 is a multiple of p for every prime p dividing m and iii) a-1 is a multiple of 4 when m is a multiple of 4. Some good sequences, using LCM, with large periods and good statistical properties can be obtained with following modifications:

$X_{i+1} = (65539 \times Xi) \bmod 2^{31} i=1,2,3,\ldots\ldots\ldots$ (X_0, initial seed must be an odd number)

$X_{i+1} = (2^{24} + 1)X_i + 1 \bmod 2^{35}$

$X_{i+1} = (2^{18} + 1)X_i + 1 \bmod 2^{35}$

Multiplicative congruential generators, also known as Lehmer random number generator (LRNG), is a type of linear congruential generator for generating pseudorandom numbers in U[0,1]. The multiplicative congruential generator, usually denoted as MCG, is computed by a recurrence formula like that of LCG with c=0

$X_{i+1} = aX_i \bmod m$

Unlike the LCG, the parameters a and m for multiplicative congruential generator have certain restrictions : the seed value X_0 must be relatively prime to the modulus m (that is, the greatest common factor of c and m is 1) while m is $2^{31} - 1 = 2147483647$ and $a = 7^5 = 16807$. Later on, since 1990 the value of ‘a’ has been changed to a=48271 (Park *et al*., 1988; Park and Miller, 1988; Saucier, 2000).

The kth order additive congruential random number generator was defined by Wikramaratna (2008) from an integer modulus an integer seed satisfying and an arbitrary set of *k* integer initial values , each satisfying $0 < X_0^0 < m$

X_0^m , m=1,………,k each

$0 \le X_0^m < m$ conforming to the following equations

$X_n^0 = X_{n-1}$ $\quad n \ge 1$

$X_n^m = (X_n^{m-1} + X_{n-1}^m) \bmod M$ $\quad n \ge 1$, m=1,. . . k

$R_n^k = X_n^k / M$ $\quad n \ge 1$

Where (x) mod M implies the remainder of dividing X by M. The numbers can be scaled to return pseudo random numbers in the interval [0.1] by division of the number by the modulus M.

There are some other random number generators like Lagged Fibonacci Generator (LFG) (see Anderson, 1990) which are widely used. The basic equation is as follows:

$X_n = X_{n-r}$ - op X_{n-s} $\quad n \ge r$

Where 0 < s < rare the lags m is a base, op is a binary operator and X {0. . . s-1}is a sequence of r initial values. The usual values of (r,s) are (55,24), (607, 273), (1279,418) (Brent, 1992).

Four possible operations are as follows:

+ addition mod *m*,- subtraction mod *m*,* multiplication mod *m*, exclusive or if *m* is a power of 2. (Brent, 1992; Zeeb *et al*, 1999)

A particular case of this kind of generators developed by Knuth (2002) is as follows:

$X_n = (X_{n-37} + X_{n-100}) \bmod 2^{30}$;

which is a Fibonacci-lagged generator. The period of this generator is about 2129.

Various modifications of Fibonacci generators have been developed and applied such as Addition and Subtraction Lagged Fibonacci Generators (ASLF).Most recent random number generator is apparently the ‘Mersenne twister’ (MT)(Matsumoto and Nishimura, 1998)which has theoretically maximal cycle, passes all the tests commonly used for random number generators, it is computationally light and generated with high speed. For a *w*-bit word length, this method returns integers in the range [0, $2^w - 1$]. The MT generator was particularly designed to circumvent most of the demerits inherent in earlier pseudo-random number generators (PRNGs).

Period length is chosen to be a Mersenne prime, and have high order of dimensional equidistribution, reliability. Some workers in the field believe that the linear congruential methods, additive methods, etc. are so simplistic that they cannot generate sufficiently random sequences. Reasonable methods exist for combining two sequences into a third one that should be able to provide a good looking random sequence. Views on combination generators are somewhat mixed. Marsaglia (1985), a strong supporter of such generators, provides some theoretical basis and gave further support later in this respect (Marsaglia and Zaman, 1994).

By contrast, Brent (1992, 1994) argued against combination generators on the grounds that such generators are slow and they are not guaranteed to posses good statistical properties.

Likewise, Coddington (1992) advocated that the mixed generators though often perform well but they almost lack theoretical basis and the mixing of two methods would probably result in new errors.

Good random numbers should be able to satisfy certain desirable properties, such asi) the generated numbers should be uniformly distributed on [0,1]. If we divide the interval [0,1] into *n* sub-classes of equal length, the expected frequency of number in each interval should be E[*N/n*] where *N* equals the total number of observations.The total observations should be sufficiently large. The pdf of the distribution is

$$f(x) = \begin{Bmatrix} 1{,}0 \ \le x \le 1 \\ 0, \text{otherwise} \end{Bmatrix}$$

for which the expectation is

$$E(R) = \int_0^1 x\,dx = \frac{1}{2}$$

With a variance of

$$\text{Var }(R) = \int_0^1 x^2\,dx - [E(R)]^2 = \frac{1}{12}$$

ii) They must be independent and identically distributed (i.i.d.) random variables. The probability of observing a value within a particular interval is independent of any previous selections. iii) The numbers must be replicable (for debugging and other purposes). iv) They should have long cycle length (period) before the repetition of the sequence occurs. It is good to have a cycle length much larger than the random numbers required in a particular simulation.
v) They should be generated with high speed (i.e., the generation algorithm should not be cumbersome). vi) The generated numbers should not use large memory. vii) They should be portable to different computers.
Various statistical tests are employed to assess the quality of random numbers, whether a sequence such as u_1, u_2, u_3, . . . , u_N can be regarded as independent and identically distributed (i.i.d.) random uniform variate; here we are concerned with tests on the interval [0,1] (see L'Ecuyer and Simard, 2007).i) The most important test is the goodness of fit test that compares the cumulative empirical distribution Fn of u_i with the distribution of U[0,1]. The popular tests in this regard are Kolmogorov-Smirnov test, Cramer von Mises test and Anderson-Darling test. Frequency classes of random numbers can also be compared with the expected frequencies using a chi-square test statistic.ii) The gap test is performed to detect the significance of the interval between recurrence of the same digit. A gap of length x occurs between the reappearance of some digit. The probability of a certain gap length is then determined using the Bernoulli trials. The gap lengths between particular digits can also be compared using the Kolmogorov-Smirnov test. For example, the special pattern such as that of zeros in the u_i can be tested. The theoretical frequency of gap length of randomly ordered digits is computed by :

$F(x) = 0.1\sum (0.9)^n = 1\text{-}0.9^{x+1}$ $\quad$ n=0, ... , x

Subsequently the observed frequencies can be tested against expected (theoretical) using Kolmogorov-Smirnov-test.
iii) The runs test is performed to check the arrangement of numbers in a sequence in order to test the hypothesis of independence of random numbers. A run, by definition, is a succession of similar events of which the previous and the next runs are different. If total number of runs is ‘a’ then mean and variance of runs are given by:

Mean = $\mu_a = (2N\text{-}1)/3$
Variance = $V_a = (16N\text{-}29)/90$

iv) The test for auto-correlation is performed to check the dependence between numbers in a sequence. The test computes the auto-correlation between every *m* numbers (*m* also refers to lag) beginning with the *i* thrandom number, as follows:

$$. \quad R_i, R_{i+m}, R_{i+2m}, \ldots R_{i+(M+1)m}$$

The value of *M* is the largest integer such that i + (*M* +1) m ≤ *N* where *N* is the total number of values in the sequence. v) The Poker test is performed which tests the independence based on the frequency in which various digits are repeated in a sequence of numbers. vi) Finally, Spectral test determines how densely k-tuples fill the k-dimensional space.

Other probability distributions can be generated by transforming the uniform variate and applying various methods to obtain random numbers from other distributions. In general we require random numbers x € [a,b] with a probability density f(x) as follows:

$p(x_1 < X \le x_2) = f(x)dx$

from a reference distribution $F(x)=F(X < x)=\int_0^x f(x)dx$

The most common methods for this purpose are the inversion and acceptance-rejection methods. Inverse transform method involves computation of cumulative frequency distribution function (CDF), u=F(x). Generate random numbers u_i[0.1]; then generate the required random variate as $x_i = F(u_i)$. The acceptance –rejection technique, an alternative technique, is particularly useful when inverse CDF does not exist. Acceptance-rejection method begins with uniform random numbers, but also requires an additional random number generator. When generating a random number from a continuous distribution with pdf *f*, the method first generates a random number from a continuous distribution with pdf g such that $f(x) \le c\,g(x)$ for some c and all x.The steps of algorithm are: First choose a density *g;* find a constant c so that $f(x)/g(x) \le c$ for every x . Next, generate a uniform random variate *u* and generate a random number *v* from the distribution *g*. The random variate *v* is accepted as the desired random number when $cu \le f(v) / g(v)$ otherwise it is discarded; a fresh *u* is generated and the above sequence is repeated.

The acceptance-rejection procedure for discrete variate is similar. Another useful method to generate random numbers from a probability distribution is the Metropolis method which is a universal sampling algorithm. This technique can generate samples from any probability distribution. However, it is slower than direct methods. The Metropolis algorithm is based on a Markov chain which has an equilibrium distribution. Many problems in science, biology, ecology, engineering, and statistics involving computation and mathematical solutions these days are approached through Monte Carlo methods; that invariably employ the use of random numbers and are solved using computer simulation (Fishman, 1996; Mooney,1997; O'Neill,2002;Robert and Casella,2004;Kroese, 2011, Kroese *et al*., 2014).In mathematical analysis of biological/ecological problems often simplifying assumptions have to be made. On the other hand, use of organisms in unnatural situations, for example work on animals in captivity in cages or vials, yield results not readily comparable to those that are obtained in natural environment. Computer simulation by means of Monte Carlo method is a good practical compromise. Genetic changes in natural populations are usually extremely slow and therefore cannot be investigated in detail in experimental or natural populations. The population geneticists, therefore, resort to either theoretical studies with the use of mathematical models or by computer simulations. Investigation of population level changes is particularly difficult as a multitude of factors may vary in the study area and different processes work simultaneously. One solution to this problem is to examine one or two processes at a time using Monte Carlo simulation. Such a simplified approach, of course, does not provide results comparable to those for realistic situation of natural populations, since other factors operating on the population are ignored, despite these shortcomings, simplistic simulations elegantly provide some fundamental and productive information. In this approach, some important interactions are disregarded, obscuring certain aspects of the dynamics of populations. Before the development of Monte Carlo methods, there were no means to gain full insight into a holistic view of population changes and fluctuations. As more and more complex simulation models are developed more comprehensive picture of the population changes would be forthcoming. However, simple simulation programs in this respect are useful as they demonstrate some fundamental processes adequately and are more readily comprehended, particularly by students and newcomers in the field. On the other hand, more comprehensive numerical simulation programs provide the understanding and prediction of how populations undergo changes under natural conditions. The Monte Carlo model is readily applied to simulate genetic drift using a random number generator to sample genes from a small parental population and passes them on to offspring. In the model presented here population size is held constant from generation to generation and the gene frequencies changes are only due to random sampling process. Natural populations are constantly under flux. Hardy-Weinberg equilibrium which assumes that allelic frequencies remain stable and in genetic equilibrium from generation to generation (in large randomly mating populations) when no forces are acting to change their frequency. This assumption is almost never met with since populations are seldom in equilibrium, and many random events (e.g., mutations, population size fluctuations, immigration and emigration, natural selection and environmental perturbations irrevocably and radically alter the genetic structure of populations.

When changes in gene frequencies are favorable to a population, the population size increases. On the other hand, such changes may be unfavorable and the affected population can then either decline or even become locally extinct. Changes in the population that occur as a consequence of natural selection are known as '*adaptive evolution'* since natural selection favors the survival of organisms that have high 'fitness'. By contrast, '*nonadaptive evolution'* refers to changes induced by factors that are independent of 'fitness'of the organism concerned such as random genetic drift or mutation pressure. Dynamics of population is the result of a multitude of factors, including genetic and random events that make the study of Population dynamics not only interesting but also challenging (Halliburton, 2004; Liu *et al*, 2006; Charlesworth and Charlesworth, 2017). Before the development of comprehensive numerical simulation programs we had no means of comprehending the complexities of population dynamics and predict the dynamics in a comprehensive way (Nuin and Otto, 2000; Stewart, 2006; Excoffier and Heckel, 2006; Mode and Gallop, 2008; Sanford and Nelson, 2012). These programs that are mostly appropriately and elaborately designed circumvent this problem and provide clear understanding of population changes, including the genetic structure, in an integrated manner and with appropriate parameters may even predict the population changes in future.

The process of genetic drift is defined as one in which allele frequencies within a population change by chance events alone as a consequence of sampling error from generation to generation. Genetic driftor sampling drift is purely a random process that after certain number of generations can lead to fixation of an allele (where only one allele remains). The fixation of an allele or it being ultimately vanished from the population depends on its selection coefficient and also chance fluctuations and on the proportion of the alleles (Kimura, 1983).

Three programs have been developed to demonstrate genetic drift: i) in a haploid population, ii) in a diploid population and iii) genetic drift in a diploid population with mutation.

A random walk is a stochastic process of movement which describes a path that comprises of successive random steps on an N-dimensional lattice. Modeling of the movement of animals, micro-organisms, cells or cell organelles has a great role in the disciplines of biology, ecology and medicine. Models of movement can be built using various methods but the commonly employed models of movements are generally extensions of simple random walk process (Codling *et al*., 2008). Random walks are uncorrelated which implies that the direction of movement is fully independent of the previous movement and probability of each step taken in the random walk is dependent solely on the current position; this simply means that the process is Markovian with respect to the location (Berg, 1993; Weiss, 1994).Unbiased refers tono preferred direction, *i.e*., at each step the direction of movement is fully random. Such movement that allows for any direction is said to be Brownian motion.

Predator–prey dynamics are an essential system in mathematical ecology and, in particular it helps to comprehend interactions between populations in the natural environment (Brauer and Castillo–Chavez, 2010; Jansen, 2001; Narayan, 2006; Baggio *et al*., 2011; Li *et al*., 2016). The dynamics of predator–prey interactions is affected by many factors, but even in the simplest possible biologically relevant form, the models usually indicate complex behavior. Most deterministic models of predator prey interaction either assume spatial homogeneity or give a spatially averaged response on the dynamics of prey and the predator. In the real systems, however, the encounters between predator and prey occur at certain random points in time and space. Furthermore, the pursuit of predator and the capture or escape of prey a real random events. Such chance events in particular the movement of predator can be incorporated in a random walk model. The random walk stochastic model, for which a software is developed represents an improvement over the deterministic predator-prey model since the former captures a number of random events in the process that include movement, encounter, escape, consumption, even though we do not have details of the behavior of the predator. The randomness of predator search, its encounter with the prey and the capture (with certain probability) can be simulated by a Monte Carlo simulation (Swartzman and Kaluzny, 1987).

This paper provides some computer programs (in BASIC language) for pseudo random number generation (PRNG), random numbers from some important probability distributions such as normal, binomial and Poisson distribution; also Monte Carlo simulation of genetic drift and a random walk model of predator-prey interaction.

MATERIALS AND METHODS

Random numbers generation: The methods of generating random numbers are given in sufficient detail in the introduction. Computer programs were developed for linear congruential method (LCONGR.BAS) and multiple congruential methods (MCONGR.BAS) (see Knuth, 1997). The recursive equations for these methods are given above.

Generating Random Numbers from Probability distributions: For generation of random numbers two methods are commonly used: inverse method and acceptance-rejection method. Algorithms of the programs developed for generating random numbers from different distributions are outlined in the sequel.

Normal distribution:

Box-Muller method: First generate two random numbers $U_1[0,1]$ and U_2 [0,1] from a uniform distribution then find two independent normal random numbers as follows:

$$Z_1 = \sqrt{-2ln\mathrm{U1}}\ \ \mathrm{Cos}\ (2\Pi U_1)$$

$$Z_2 = \sqrt{-2ln\mathrm{U2}}\ \ \mathrm{Sin}\ (2\Pi U_2)$$

Random normal by Polar method: The steps are as follows;

1. Generate two random numbers U_1 and U_2 from U[0,1]
2. Set $V_1 = 2U_1 - 1$; $V_2 = 2U_2 - 1$
3. If S>1 , go to step 1 otherwise move to step 4
4. Yield two independent normal variates

$$Z1 = \sqrt{-2lnS/S}\, V_1$$

$$Z2 = \sqrt{-2lnS/S}\quad V_2$$

Inversion method: The necessary steps are:

1. Generate a random number U from U∼ [0,1]
2. Output Z as follows:

$$Z = \left[-\ln\frac{1}{U} - 1\right] /1.702$$

The algorithm is comparatively fast because it requires only one random number from uniform distribution U[0.1].

Poisson random number generation
One efficient method is that of Atkinson (1979):The steps of the algorithm are as follows:
1.Set r= e^{-m} (m=mean), N=0 and S=1
2. Generate a standard normal random variate U(0,1) and make S=SU
3. If S $\geq$ r then N=N+1 goto 2 else output N
The merit of this algorithm is that it requires the computation of only one constant e^{-m} , this makes it sufficiently speedy.
Binomial random number generation
Algorithm of Bocu (2000) is used for which the steps are as follows:
1. Generate N random numbers from uniform distribution U[0,1]
2. Count U_i that are $\leq$ P; the count is then Binomial.

Genetic Drift:
Genetic drift or sampling drift is the random variation in the relative frequency of different genotypes in a small population, owing to the chance disappearance of particular genes as individuals die or do not reproduce and after certain number of generations can lead to fixation of an allele. The fixation of an allele or being ultimately eliminated from the population depends on its selection coefficient and also chance fluctuations and on the proportion of the alleles. Programs related to genetic drift are developed both for haploid and diploid populations. Also a program that incorporates mutation is also developed. We assume that the entire population reproduces simultaneously. This implies that the population is discrete and also non-overlapping. For a haploid population consider that there are only two type of individuals (having alleles A and a) as in the one locus two allele haploid model. It is also assumed that the two types of individuals are equally fit (W_A and W_a). Consider a population that has a constant size N over the given time period. Initially the frequency of the two alleles are the same: of allele A is p=0.5 and that of 'a' is q=0.5.It can be imagined that individuals are reproducing and generating an enormous number of propagules from which only a small number N of the surviving offspring are sampled. This experiment implies that the number of copies of allele A among the offspring should conform to binomial distribution. These frequencies change from generation to generation and for small population e.g., N=20 fixation of one or the other allele generally occurs after a certain number of generations. The sampling error of the frequencies for haploid case is as follows:

$$\sqrt{[p(1-p)/N]}$$

The program DRIFT1.BAS is the relevant genetic drift program for a haploid population. We begin the program with equal frequencies of A and a, e.g., for N=200 the allele frequencies are A=100,a=100.

In the diploid population case the initial frequencies of genotypes AA, Aa and aa are chosen in the ratio 1:2:1 such that p=0.5 and q=0.5. Choosing two kinds of alleles A and a among the gametes the adults will reflect the frequency at which they occur with a certain standard deviation owing to sampling process. If the total population size is N and gene frequencies p and q=(1-p) the sampling error is;

$$\sqrt{[p(1-p)/2N]}$$

The relevant program for a diploid population is DRIFT2.BAS.

Genetic drift with mutation:
This program involves genetic drift in a diploid population with mutation which in general follows Mode and Gallop (2008). To keep the mutation rate high we have chosen a particular mutation that occurs in Peppered moth *Biston betularia*. As an example, take the mutation $c \dashrightarrow C$ in *Biston betularia* in Manchester (U.K.) and nearby industrial areas Northwest of England where the normal (light coloured form *typica* having allele c) was replaced rapidly to coal-black form *carbonaria* C. The allele C is dominant over c. Before the mid19th century the form *typica* (grey) was dominant but was replaced at a dramatic rate by coal black form *carbonria* in about 50 years or so.. Manchester was the early centre of the gene while the pattern elsewhere in the country was of the migration of melanic (*cabonaria*) form to the north-west, followed by selection. Taking the example of *B. betularia* permits a simple example of genetic drift along with high mutation rate. The per generation rate of mutation in *B. betularia* can be as high as 0.041%, which means that to obtain a rapid evolution, as fast as what we can observe in the nature, mutation should occur in about 4 gametes per 100 at each generation from *typical* (c) to *carbonaria* (C). Note that this is an unusually fast rate of mutation with regard to spontaneous mutation rate observed in nature which is about 10^{-5} to 10^{-6}. The program DRIFTMR.BAS simulates genetic drift with mutation.
Random walk predator prey model:

Here an example of predator prey interaction is provided in which the predators move randomly (in a random walk) on a grid where they search, encounter and capture (or fail to capture) the prey. These stochastic events are simulated through a Monte Carlo technique (Swartzman and Kaluzny, 1987). The method begins with the construction of an r × r grid that contains the prey population (created by a program PREYFIL.BAS, that produces the file PREY containing the random location of each of the N prey). In the prey population, individuals are uniformly randomly distributed on the grid. Thus each cell of the grid may contain 0,1,2,3, . . . , m number of prey. The predator walks over the grid randomly feeding on the prey when encountered and captured. The word 'encounter' literally means 'to meet by chance or unexpectedly'; here it implies entering of the predator in a grid cell where certain numberof prey exist.It is reasonable to believe that the predator is unable to capture and eat all of them. Each prey has a certain probability of escape or hiding from the predator. The prey can avoid the predator by hiding or running (or flying) swifter than the predator with a certain probability of escape; the probability of capture is PC and that of not being captured is NC=1- PC. Furthermore, the maximum consumption of the predator (C_{max}) in one cell is fixed prior to simulation. The predator is allowed to move on the grid unit either up or down, left or right. Each direction of movement has equal probability. When the maximum time units of the simulation are completed, average and standard deviation of both encounter and consumption are computed for each of the K predators. The relevant programs are PREYFIL.BAS that creates the file PREY for prey distribution on the grid and RNDWALK.BAS that performs random walk of the predator and predator-prey interaction and makes use of the file PREY.

Listings of all the necessary programs written in GWBASIC by the first author (S.S.S.) are given in APPENDIX A. For implementation of these programs the user requires a file GWBASIC.EXE.

RESULTS and DISCUSSION

Random numbers

Computer programs of the two common methods for generating pseudo random numbers (PRN) namely, linear congruent method (LCONGR.BAS) and multiple congruent method (MCONGR.BAS) were tested. Using each program 500 uniform PRN U[0,1]were generated. The range was divided into ten equal class interval(each 0.1width) and the frequencies within each part counted. Thus, each part should have approximately a frequency of 50 for a uniform distribution. Fig.1a and 1b show the frequency distribution for each of the method. Several simulations were performed but the result for only one representative simulation for each method is given. It is apparent that the classes show more or less equal frequency owing to uniform distribution. Multiple congruent method shows marginally better results compared to linear congruent method. These two random number generators are commonly used in simulations in various disciplines.

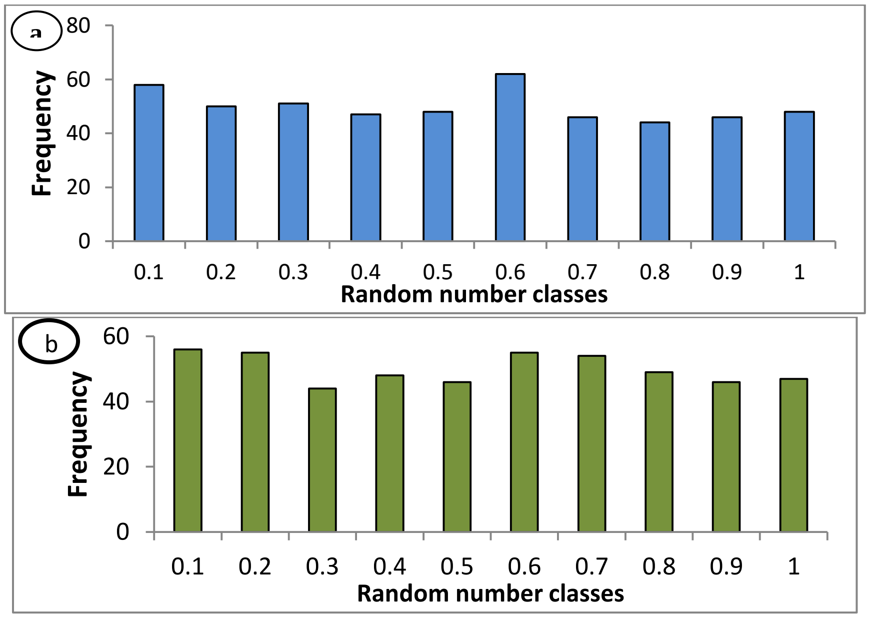


Fig.1. Frequency distributions for Linear Congruent method and Multiplicative Congruent method of generating random numbers. (a) Linear Congruent (b) multiplicative Congruent method.

The programs developed for random number generation from various probability distributions such as Normal distribution NORPROB.BAS, Binomial distribution BIPROB.BAS and Poisson distribution POPROB.BAS are not given here for the sake of brevity but they can be obtained from the first author. However they were tested for accuracy and were found to be sufficiently effective to represent a particular distribution with the given parameters (results not shown). The results of these programs will be presented elsewhere and the random numbers from these distributions will be employed in other applications of Monte Carlo simulations.

These programs (except that of binomial distribution) are not used in the present experimental simulations.

Random numbers from probability distributions

Genetic drift

Simulations of gene frequency under genetic drift alone for a haploid population followed up to 20 generations are given in Figs. 2a and 2b. Ten (or more) populations were run for N=20 and N=200 each. The populations with smaller sample size (N=20) showed greater tendency to dispersion (Fig.2a) compared to populations with greater sample size(Fig.2b). In the smaller population size scenario (Fig.2a) there was 1 fixation (of allele A) while there were four losses (of allele A). Obviously, for a small population there are greater chances of fixation or loss.

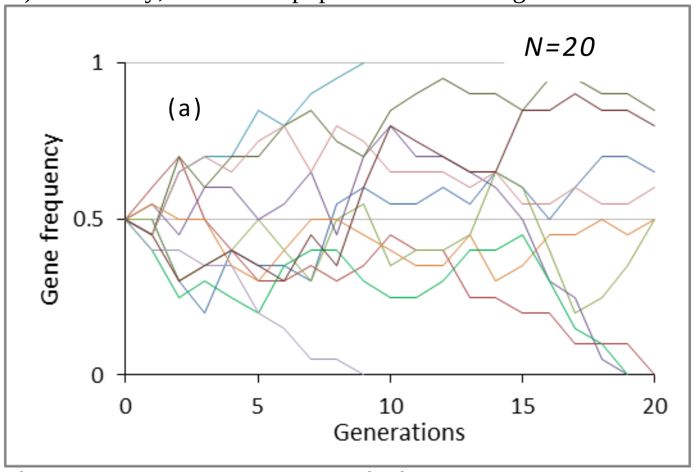


Fig. 2(a). Simulation of gene frequency changes under genetic drift of haploid population with the alleles A,a for 20 generations at each starting sample size. *N=20*

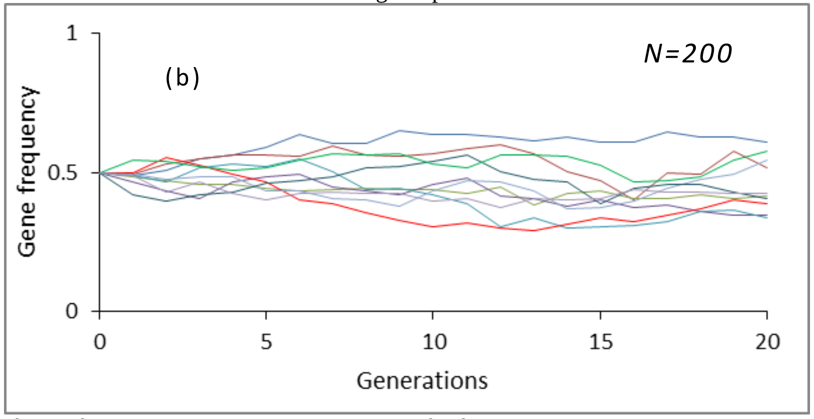


Fig. 2(b). Simulation of gene frequency changes under genetic drift of haploid population with the alleles A,a for 20 generations at each starting sample size. *N=200*

The simulations for large population (N=200) no fixation or loss was observed (Fig.2b) when populations were followed up to 20 generations, although there was some dispersion and the gene frequencies slightly diverted from the initial frequencies of p=0.5 and q=0.5. Simulation scenarios for diploid populations are shown in Fig.3a and 3b for N=20 and N=200 respectively. A fair degree of dispersion can be seen for the smaller population (N-20) where 4 fixation(of A) two losses occurred (Fig.3a). Whereas in the larger population (N=200) no fixation or loss was observed) (Fig.3b).

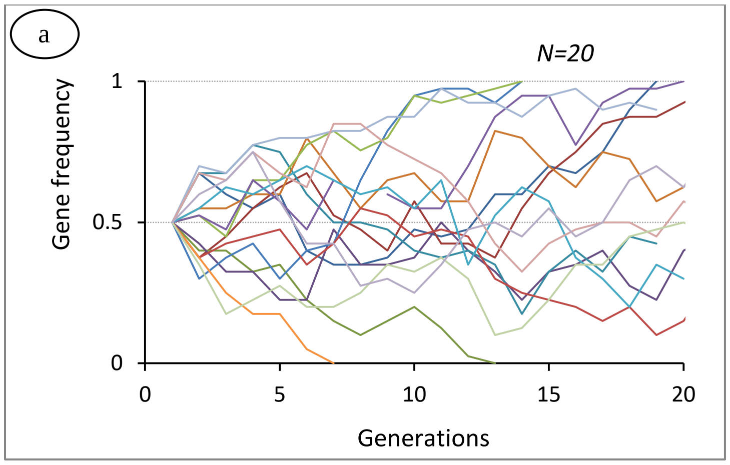


Fig. 3(a). Simulation of gene frequency changes under genetic drift of diploid population with the alleles Aa for 20 generations at each starting sample size.

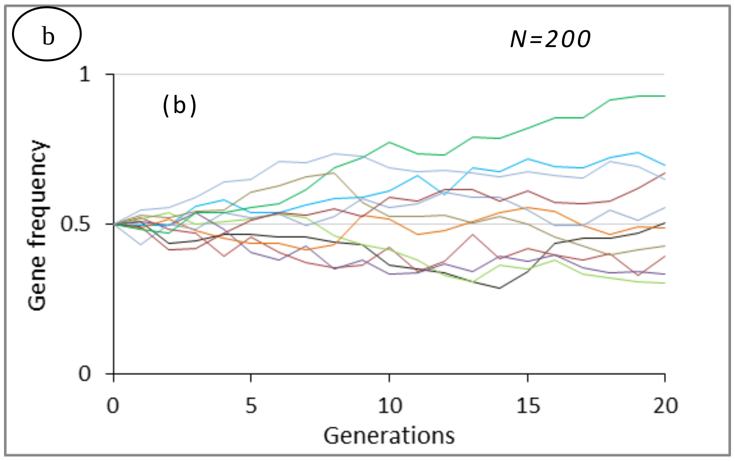


Fig. 3(b). Simulation of gene frequency changes under genetic drift of diploid population with the alleles Aa for 200 generations at each starting sample size.

Evidently, the rate of fixation is greater in smaller populations compared to larger populationswhich accords well with the theory of genetic drift (Crow and Kimura, 1970; Cook, 1976;Hartl and Clark,2007).Another way of representing gene frequencies is that of Wright(1969) and Kimura and Ohta (1971) in which theoretical distribution is shown with certain initial gene frequency, such p=0.5 or 0.1. These curves represent the expected frequency distributions for a certain number of generations and with given N. Such frequency distributions mimic the probability distributions showing the behavior of each of the population with known parameters. For plants and animals that breed annually the genetic drift alone causes considerable loss of variation in a few years duration. Furthermore, this effect becomes more pronounced for the small sized populations. It has been suggested that a population size of at least 500 is adequate to effectively decrease the effect of genetic drift (Stiling, 1996). Accordingly, the '50/500' rule of thumb has often been recommended in the field of conservation as a "magic" number (Franklin, 1980; Simberloff, 1988). Fifty, being the critical population size to prevent excessive inbreeding and 500 the critical size to prevent genetic drift.

The results of genetic drift with mutation (at a rate 0.041) for population sizes of N=20 and N=200 are give in Figs.4a,b respectively for diploid populations; simulations were run for 20 generations. The simulation scenario for smaller populations (N=20) showed greater dispersion along with 3 fixations of *carbonaria* (C) and one loss (Fig.4a). On the other hand, low dispersion was observed for the large populations (N=200) and no fixation or loss occurred (Fig.4b). Many field studies have been conducted on *Biston betularia* populations in the United Kingdom (Kattlewell, 1958; Clarke and Sheppard, 1966; Bishop, 1972). Kattlewell (1958) reported *carbonaria* (C) frequency of 87%, Clarke and Sheppard (1966) found 94% *carbonaria* while Bishop (1972) also reported 94% from Birmingham, Liverpool, Wirral, Hawarden and Loggerhead, respectively. The frequency of *carbonaria* mutation elevated rapidly in the industrial areas of U.K. and eventually the populations of *B. betularia* were predominately that of *carbonaria* form. Thus the process of genetic drift tends to eliminate genetic variation while mutation tends to enhanced variation. Fig. 5 shows the number of mutation that occurred for each simulation (1-10) for populations of N=20 and N=200. Obviously, the number of mutations in populations of N=200 are much higher than those of small populations (N=20).

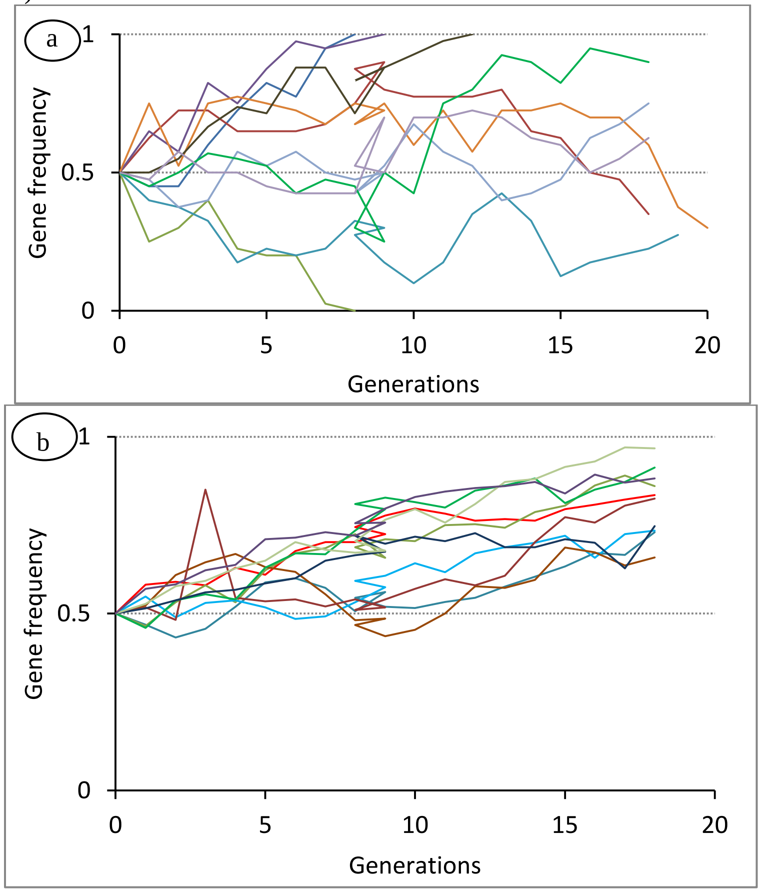


Fig.4 (a, b). Simulation of genetic drift with mutation (mutation rate = 0.041). Simulation for N=20, (b) Simulation of N=200.

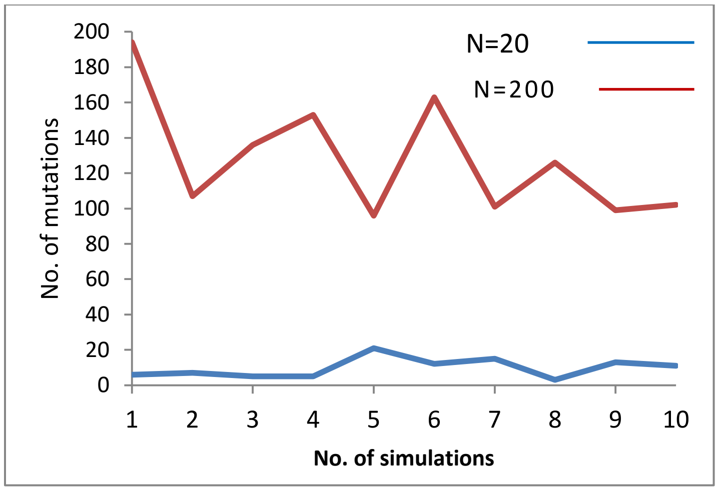


Fig.5. Number of mutations recorded in each of the ten simulations for populations of N=20 and N=200 (Mutation rate = 0.041).

Predator-Prey model

The results of computer simulation of predator prey interaction are given in Figs. 6,7and 8 for 1,2 and 3 predators and 100 uniformly distributed prey scenarios. The maximum consumption in any one grid unit for 1 time unit for a predator was set at 5 and maximum time was 10 units. The results for a single predator simulation (Fig.6),in general, show close values for predator’s encounter and consumption in different simulation runs since there is only one predator and plenty of prey is available even though the prey can escape from the predator with a probability of (1-0.8)=0. 2. Additionally, both the number of encounters and consumption are in most of the simulation runs found higher than those found for 2 and 3 predators simulations (compare Fig 6 with Figs 7 and 8). For 2 and 3 predator simulation scenarios, both the number of encounters and consumption were slightly lower compared to those of one predator simulation as the same number of prey (100) were available to greater number of predators. The difference between encounter and consumption was also closer owing to the fact that the limit of maximum intake was set at 5 and with decreasing number of prey with time almost all those encountered were consumed by the predators.

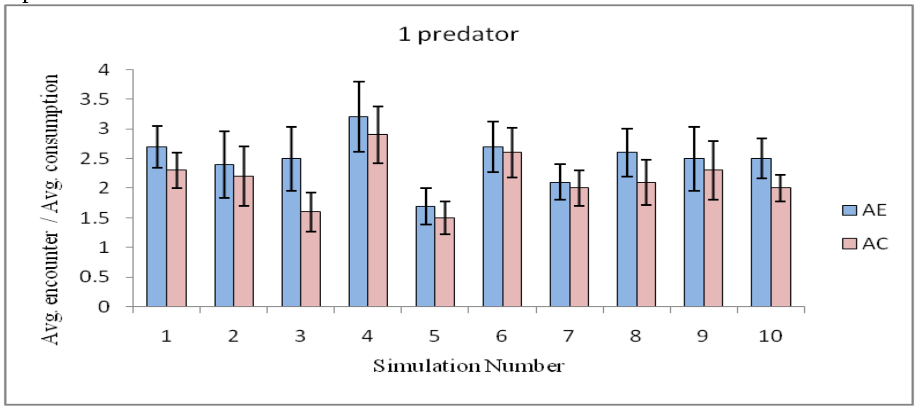
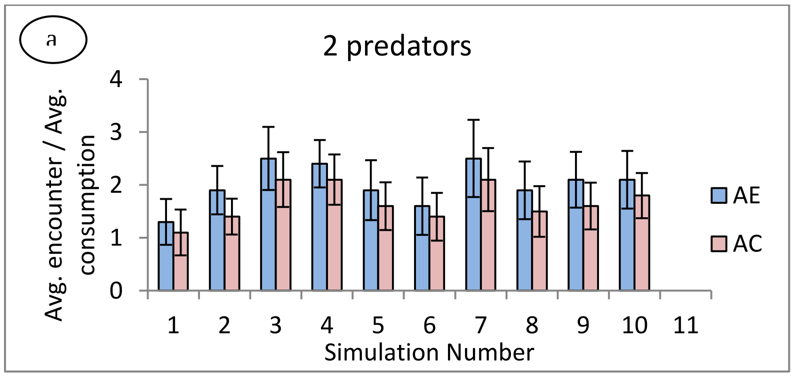


Fig.6. Random Walk simulations using 1 predator and 100 prey. AE = Average encounter and AC= Average consumption. Standard errors are attached for each average.



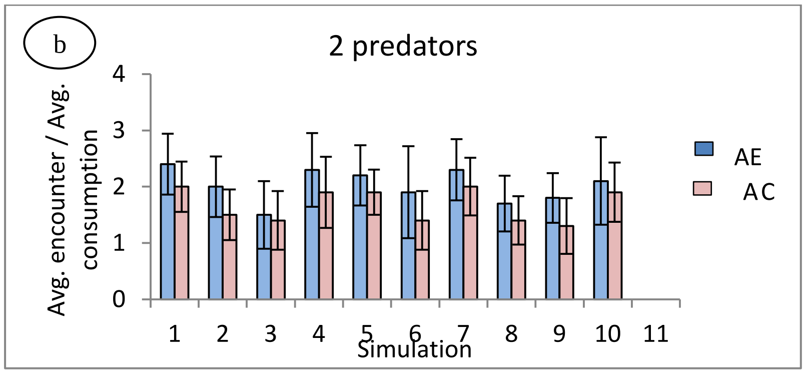


Fig.7. Random Walk simulations using 2 predators and 100 prey. AE= Average encounter and AC= Average consumption. Standard errors are attached for each average. (a) Predator 1, (b) Predator 2.

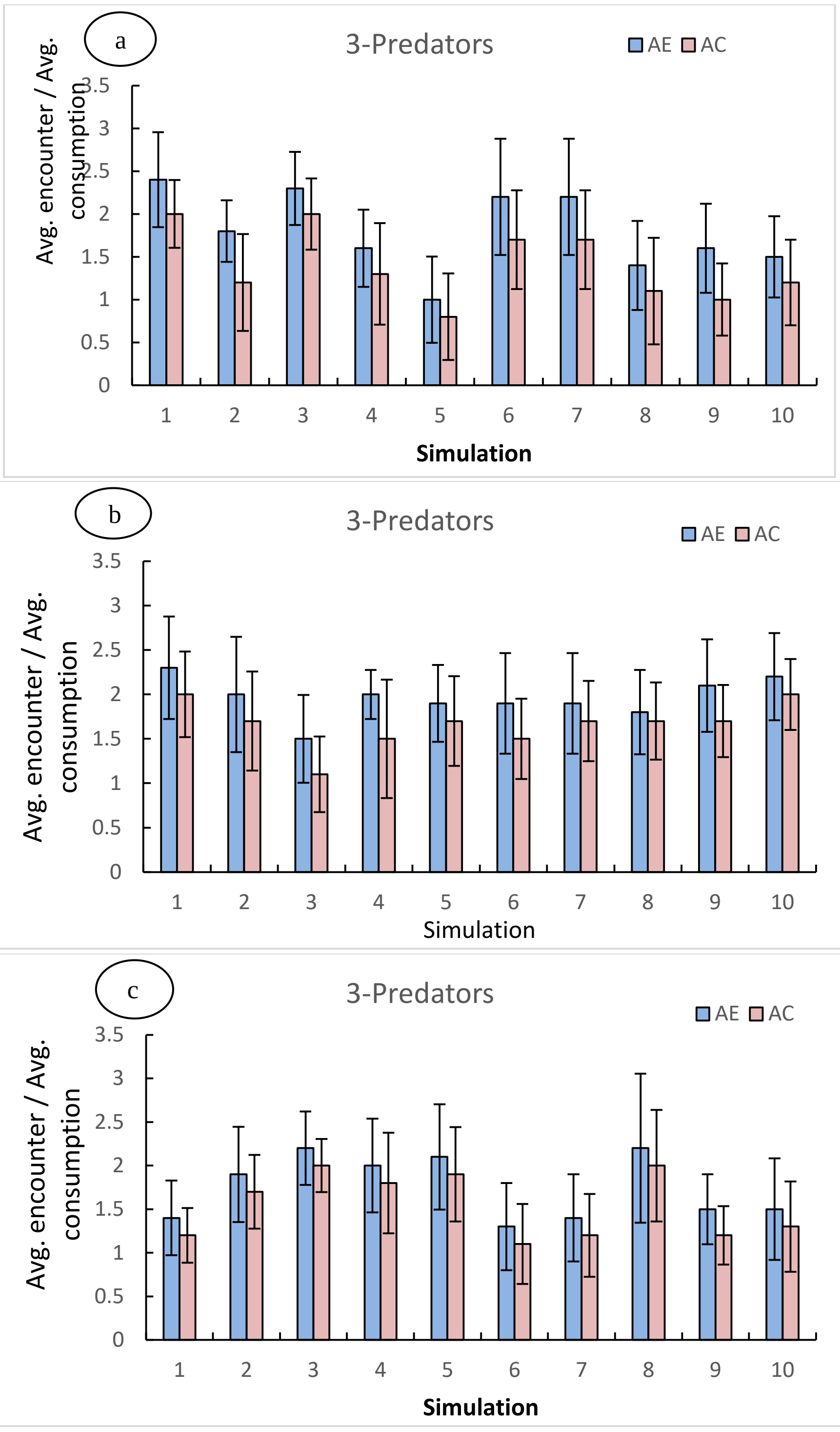


Fig.8. Random Walk simulations using 3 predators and 100 prey. AE= Average encounter and AC= Average consumption. Standard errors are attached for each average.(a) Predator 1, (b) Predator 2, (c) Predator 3.

Various variants of RNDWALK predator-prey model can easily be conceived, such as 1) the movement of predator to high density patches of prey, 2) creating 'refugia' in the landscape (grid) where the prey can hide from the predator and rest for a while, 3) two types of predators with unequal satiation capacity,4)one predator and multiple prey with varying density, size of prey and predator preference . Evidently, a great variety of stochastic events of predator-prey interactions can be modeled in this manner. In fact, software for these have been developed by the first author (S.S.S.) and will be presented elsewhere.

The deterministic predator-prey models (e.g., Brauer and Castillo–Chavez, 2010; Jansen, 2001; Narayan, 2006; Baggio *et al*., 2011; Li *et al*., 2016) either assume spatial homogeneity or give an spatially average response on the dynamics of prey and the predator. However, in real systems, the encounters of predator with the prey, occur in space and time. Furthermore, the pursuit of predator over the land (random walk) and the capture or escape of the prey are allstochastic events. Thus the random walk predator-model developed and demonstrated here is more appropriate as stochastic events are incorporated into the model compared to deterministic models that fail to account for randomness of various events in the predator-prey interaction and their outcome is fixed for given parameter values. For stochastic models, the same set of parameter values and initial conditions will lead to an ensemble of different outputs. Here, for simulation purpose probabilistic events are simulated by generating random numbers from uniform distribution U[0,1], however, random numbers can easily be generated from other probability distributions but such numbers can be employed when we are certain regarding a specific probability distribution of a particular process or event. The paper briefly reviewed the generation of pseudo-random numbers and their tests for randomness. Computer programs for PRNG were developed and tested for two common methods. In addition, this paper successfully demonstrated the application of Monte Carlo techniques in genetics and ecology. The programs clearly illustrated the biological systems under investigation and to an appreciable extent mimic the processes occurring in nature.

REFERENCES

Acevedo, M.F. (2012). *Simulation of Ecological and Environmental Models*. Wiley, New York.

Anderson, S. (1990). Random number generators. *SIAM Reviews*, 32:221-251.

Atkinson, A.C. (1979). The computer generation of Poisson random variables. *Appl. Statist*., 28: 29-35.

Baggio, J.A., K. Salau, M.A. Janssen, M.L. Schoon and O. Bodin (2011). Landscape connectivity and predator-prey population.*Landscape Ecol*., 26: 33-45.

Balloux, F. (2001). EASYPOP (Version 1.7): A computer program for population and genetics simulation. *J. Heredity*, 92: 301-302.

Berg, B.A. (1993). Locating global minima in optimization problems by a random-cost approach. *Nature*, 361:708-710.

Bishop, J.A. (1972). An experimental study of the cline of industrial melanism in *Biston betularia* (L.)(Lepidoptera) between urban Liverpool and rural north Wales. *J Anim Ecol*., 41:209–243.

Bocu, D. (2000). An algorithm for generating random numbers with binomial distribution.*Yugoslav Jour. Operation Research*, 10:99-108.

Brauer, F. and C. Castillo-Chavez (2010). *Mathematical Models in Population Biology and Epidemiology*. Springer-Verlag, Berlin.

Brent, R. P. (1992). Uniform random number generators for supercomputers. *Proceed. Fifth Australian Supercomputing Conf.,* SASC Organizing Committee, pp. 95-104. Melbourne, Australia.

Brent, R. P. (1994). On the periods of generalized Fibonacci recurrences. *Math Comput.,* 63: 389 401.

Charlesworth, B. and D. Charlesworth (2017). *Evolution: A Very Short Introduction*. 2nd.ed., Oxford University Publishers, Oxford.

Coddington, P.D. (1092). Analysis of random number generators using Monte Carlo simulation. *Int. J. Modern Phys*., 6:1-14.

Codling, E.A., M.J. Plank and S. Benhamo (2008). Random walk models in biology. *J. Royal Soc. Interface*, 5: 813-834.

Cook, L.M. (1976). *Population Genetics*. Chapman and Hall, New York.

Clarke, C.A. and P.M. Sheppard (1966). A local survey of distribution of industrial melanic forms in the moth *Biston betularia* and estimates of the selective valuesof these in an industrial environment. *Proced. Royal Soc., Series B* 165: 424-437.

Crow, J.F. and M. Kimura (1970). *An Introduction to Population Genetics Theory*. Harper & Row, New York.

Del Moral, P., A. Doucet and A. Jasra (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society*, *Series B*., 68: 411–436.

Excoffier, L. and G. Heckel (2006). Computer programs for population genetics data analysis: a survival guide. *Nat. Rev. Genet*. 7:745-758.

Fishman, G.S. (1996). *Monte Carlo: Concepts, Algorithms and Applications*. Springer-Verlag, Berlin.

Franklin, I.R. (1980). Evolutionary change in small population. *In*: M.E. Soule and B.A. Wilcox (eds.). *Conservation Biology: an Evolutionary-Ecological Perspective*. Sinauer Associates, Sunderland, Massachusetts.

Frellsen, J., O. Winther, Z. Ghahramani and J. Ferkinghoff-Borg (2016). *Bayesian Generalized Ensemble Markov Chain Monte Carlo*. Cambridge, UK.

Halliburton, R. (2004). *Introduction to Population Genetics*. Prentice Hall, Upper Saddle River.

Hartl, D.L. and A.G. Clarke (2007). Principles of Population Genetics. Sinauer Assoc. Publishers, Sunderland, Massachusetts.

Jansen, V.A.A. (2001). The dynamics of two diffusively coupled predator prey populations. *Theor. Popul. Biology*, 58: 119-131.

Kattlewell, B. (1958). A survey of the frequencies of *Biston betularia* (L.) (Lepidoptera) and its melanic form in Great Britain. *Heredity*, 12:51-72.

Kimura, M. (1983). *The Neutral Theory of Molecular Evolution*. Cambridge University Press, Cambridge.

Kimura, M. and T. Ohta (1971).*Theoretical Aspects of Population Genetics.* Princeton University Press, Princeton, New Jersey.

Knuth, D. (1997). *The Art of Computer Programming, Vol. 1, Seminumerical Algorithms*, 2nd ed., Addison-Wesley, Cambridge, MA, USA.

Knuth, D. (1998). *The Art of Computer Programming, Vol. 2 Seminumerical Algorithms*, 2nd ed., Addison-Wesley, .Cambridge, MA , USA.

Knuth, D. (2002). *The Art of Computer Programming: Seminumerical Algorithms*, Vol. 2, 3rd ed., Addison Wesley., Cambridge, MA, USA

Kroese, D.P. (2011). *Monte Carlo Methods*. Summer School of the Australian Mathematical Sciences Institute (AMSI), Qeensland, Australia.

Kroese, D. P., T. Brereton, T. Taimre and Z.I. Botev (2014). Why the Monte Carlo method is so important today. *Comput. Stat*., 6: 386-392.

Law, A.M. (2014). *Simulation Modeling and Analysis*, 5th ed., McGraw-Hill, New York, USA.

L’Ecuyer, P. (1990). Random numbers for simulation. *Commun. ACM* ,, 33: 86–97.

L'Ecuyer, P. (1994). Uniform random number generation, *Annals of Operations Research*, 53: 77-120.

L'Ecuyer, P. (1996). Combined multiple recursive random number generators. *Operations Research,* 44: 816-822.

L’Ecuyer, P., R. Blouin and A. Couture (1993). A search for good multiple recursive random number generators, *ACM Trans. Modell. Comput.Simul*., 3: 87-98.

L’Ecuyer, P. and R. Simard (2007). Test U01: A C library for empirical testing of random number generators. *ACM Trans. Math. Softw*., 33:1-35.

Li, B., S. Liu, J. Cui and J. Li (2016). A simple predator-prey population model with rich dynamics. *Appli. Sci*., 6:1-14.

Liu, D., J.T. Trumble and R. Stauthamer (2006). Genetic differentiation between eastern populations and recent introductions of potato psyllid (*Bactericera cockerelli*) into western North America. *Entomol. Exp. Appl.*, 18: 177-183.

Marsaglia, G. (1985). A current view of random number generators, 3-10p. In: Marsaglia, G (Ed).*Computer Science and Statistics, Sixteenth Symposium on the Interface*, Elsevier Science Publishers, North-Holland, Amsterdam.

Marsaglia, G. and A. Zaman (1994). Some portable very-long-period random number generators. *Computers in Physics,* 8: 117-121.

Matsumoto, M. and T. Nishimura (1998). Merseme twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. on Model. & Comp. Simul.,* 1: 3–30.

Mode, C.J. and R. J. Gallop (2008). A review on Monte Carlo simulation methods as they apply to mutation and selection as formulated in Wright–Fisher models of evolutionary genetics. *Mathemat. Biosci*., 211: 205-226.

Mode, C.J. T. Raj and C.K. Sleeman (2011).Monte Carlo implementation of two sex density dependent branching processes and their applications in evolutionary genetics. *In*: Mode, C.J. (Ed.). *Applications of Monte Carlo Methods in Biology, Medicine and other Fields of Science.* Intech Publishers, Carotia.

Mooney, C. Z. (1997). *Monte Carlo Simulation*. Sage Publications, Thousand Oaks, CA, USA.

Narayan, K.L. (2006). A prey - predator model with cover for prey and an alternate food for the predator, and time delay**.** *Caribb. J. Math. Comput. Sci*.9, 56-63.

Nuin, P. A. S. and P.A. Otto (2000).A program for representing and simulating population genetic phenomena. *Genetics Mol. Biol*., 23: 53–60.

O’Neill, P.D. (2002). A tutorial introduction to Bayesian inference for stochastic epidemic models using Markov chain Monte Carlo methods. *Math. F. Biosci*., 180: 103-111.

Park, S.K. and K.W. Miller(1988). Random number generators: Good ones are hard to find, *Commun. ACM*, 36: 1192-1201.

Park, S.K. , K. W. Miller and P. K. Stockmeyer (1988). Technical Correspondence on random numbers. *Commun. ACM.*, 36 : 105–110.

Robert, C. P. and G. Casella (2004). *Monte Carlo Statistical Methods*.2nd ed. Springer, Secaucus, NJ, USA.

Rubenstein, R.Y. and Kroese (2016). *Simulation and the Monte Carlo Methods*.3rd ed., Wiley, NY, USA.

Sanford, J.C. and C.W. Nelson (2000). The next step in understanding population dynamics: Comprehensive numerical simulation. *In*: M. C. Fuste (Ed.). *Studies in Population Genetics*. Intech Publishers., Croatia.

Sanford, J.C. and C.W. Nelson (2012).Understanding population dynamics: Comprehensive numerical simulation. *In*: J.C. Sanford (Ed.) *Population Genetics*.2nd ed., Intech Publishers, Croatia.

Saucier, R. (2000). *Computer Generation of Statistical Distributions* (1st ed.). Aberdeen, MD. Army Research Lab., UK.

Schneir, B. (1996). *Applied Cryptography*. John Wiley & Sons, New York.

Simberloff, D. (1988). The contribution of population and community biology to conservation science. *Annl. Rev. Ecol. &Systemat*., 5:161-179.

Stewart, R.D. (2006). Fast Monte Carlo simulation of DNA damage formed by Tectron and light ions. *Phys. in Medicine & Biol*., 51: 1693-1706

Stiling, P.D. (1996). *Ecology: Theories and Applications*. Prentice Hall, Upper Saddle River, New Jersey.

Swartzman, G.L. and S.P. Kaluzny (1987).*Ecological Simulation Primer.* MacMillan, New York, USA.

Thomopoulos, N. T. (2013). *Essentials of Monte Carlo Simulation: Statistical Methods for Building Simulation Models*. Springer, Berlin.

von Neumann, J. (1951). Various techniques used in connection with random digits. Pp 36-38.(Eds.) A. S. Householder, G. E. Forsythe and H. H. Germond (Eds.) *Monte Carlo Methods*, National Bureau of Standards Applied Mathematics Series, 12, Washington, D.C.: U.S. Government Printing Office..

Weiss, G.H. (1994). *Aspects and Applications of random Walk*. North-Holland Publ., Amsterdam.

Wikramaratna. R.S. (2008). The additive congruential random number generator – A special case of a multiple recursive generator. *J. Comput. & Appl. Maths*., 216: 371- 387.

Wright, S. (1969). *Evolution and the Genetics of Populations*. Vol.2. *The Theory of Gene Frequencies*. Chicago University Press, Chicago

Zeeb, C. N., P.J. Burns., K. Branner, and J. Dolaghan (1999).*User's Manual forMONT3D* - Version 2.4, Department of Mechanical Engineering, Colorado State University,Fort Collins, Colorado, USA.

(*Accepted for publication February 2020*)

APPENDIX A

Listings of various programs developed by the first author (S. Shahid Shaukat).

LISTING OF PROGRAM LCONGR.BAS (FOR RANDOM NUMBER GENERATION USING LCN.

```
10 REM LINEAR CONGRUENT METHOD FOR PRNG (LCN)
20 PRINT “THE INITIAL (SEED VALUE SHOULD BE ODD) “
30 DIM RAN (500)
40 M=2^16-1
50 INPUT “PROVIDE OUTFILE NAME “;F$
60 OPEN “O”,1,F$
70 INPUT “HOW MANY NUMBERS (50-500) “;N
80 INPUT “SEED VALUE e.g. 7391,7937,7953 “;X0
90 DIM X(N)
100 C=273
110 RANDOMIZE TIMER
120 FOR I=1 TO N
130 IF I>=2 THEN X0=MODD
140 A=(16807*X0+C)
150 A=INT(A)
160 MODD=A-(M*INT(A/M))
170 RAN- MODD/M
180 PRINT RAN;”  “;
190 PRINT #1, RAN
200 NEXT I
210 CLOSE #1
220 PRINT
230 PRINT “OUTFILE NAME “;F$
```

MCONGR.BAS (RANDOM NUMBER GENERATIONUSING MCN)

```
10 REM MULTIPLICATIVE CONGRUENTIAL NUMBERS (MCN)
20 PRINT “THE INITIAL (SEED) VALUE SHOULD BE ODD”
30 M=2^31-1
40 INPUT “HOW MANY NUMBERS (50-500) “;N
50 INPUT  “SEED VALUE e.g. 3791,9173,713; X0
60 XZ=X0
70 INPUT “OUTFILE NAME “;F$
90 DIM X(N)
100 FOR I=1 TO N
110 IF I>=2 THEN X0=MODD
120 A=((16807*X0))
130 A=INT(A)
140 MODD=A=(M*INT(A/M))
150 RAN= MODD/M
160 PRINT RAN;” “;
170 PRINT #1, RAN
180 NEXT I
190 CLOSE#1
200 PRINT
210 PRINT “OUTFILE NAME “;F$
220 PRINT “NUMBER OF RANDOM NUMBERS “;N
230 PRINT “X0=”;XZ
240 END
```

LISTING OF GENETIC DRIFT PROGRAM FOR HAPLOID POPULATION

```
DRIFT1.BAS
10 REM GENETIC DRIFT:HAPLOID POPULATION:PROGRAMMER:S.S.SHAUKAT
20 INPUT “INITIAL POPULATION SIZE “;N
30 INPUT “HOW MANY GENERATIONS “;K
40 DIM G(N)
50 FOR I=1 TO N
60 IF INT(I/2)=I/2 THEN G(I)=1 ELSE G(I)=2
70 NEXT I
80 PRINT
90 PRINT
100 RANDOMIZE TIMER
110 FOR I=1 TO N
120 R1=INT((N*RND(1)+1)
130 TEMP=G(I)
140 G(I)=G(R1)
150 G(R1)=TEMP
160 NEXT I
170 PRINT “INITIAL POPULATION SHUFFLED”
180 FOR I=1 TO N: PRINT G(I);” “;: NEXT I
190 A=0:B=0
200 FOR I=1 TO N
210 IF G(I)=1THEN A=A+1
220 IF G(I)=2 THEN B=B+1
230 NEXT I
240 P=A/(A+B) : Q=B/(A+B)
250 PRINT “INITIAL FREQUENCY OF A=”;P
260 PRINT “INITIAL FREQUENCY OF a=”;Q
270 FOR J=1 TO K
280 PRINT “GENERATION “;J
290 X=0
300 FOR J=1 TO N
310 RANDOMIZE TIMER
320 R=RND(1)
330 IF R<=P THEN X=X+1
340 NEXT I
350 A=0 : B=0
360 FOR I=1 TO X
370 G(I)=1: A=A+1: NEXT I
380 FOR I= X+1 TO N
390 G(I)=2 : B=B+1 : NEXT I
400 X=0
410 FOR I=1 TO N:PRINT G(I);” “; : NEXT I
420 PRINT “ALLELE A=”;A
430 PRINT “ALLELE a=”;B
440 p=A/(A+B) : Q=B/(A+B)
450 PRINT “P=”;P ;”  Q=”;Q
460 E=SQR((P*Q)/N) : PRINT TAB(7);”SAMPLING ERROR=”;E
470 IF P=0 THEN PRINT “LOSS OF A, FIXATION OF a “: GOTO 530
480 IF P=1 THEN PRINT “FIXATION OF A, LOSS OF a “: GOTO 530
490 X=0
500 INPUT Z
510 NEXT J
520 PRINT “POPULATION SIZE N “;N
530 END
```

LISTING OF GENETIC DRIFT PROGRAM FOR DIPLOID POPULATION

<u>DRIFT2.BAS</u>

```
10 REM GENETIC DRIFT: DIPLOID POPULATION:PROGRAMMER;S.S.SHAUKAT
20 PRINT “INITIAL FREQUENCIES “
30 DIM G(1000)
40 INPUT “FREQUENCY OF AA INDIVIDUALS “;N1
50 INPUT “FREQUENCY OF Aa INDIVIDUALS “;N2
60 INPUT “FREQUENCY OF aa INDIVIDUALS “;N3
70 INPUT “HOW MANY GENERATIONS “;NG
80 N=N1+N2+N3 :NN=N*2:PRINR “N=”;N; “  2*N=”;NN
90 PRINT “TOTAL INDIVIDUALS=”;N
100 FOR I=1 TO  (N1*2)+N2
110 G(I)=1
120 NEXT I
130 FOR I= (N1*2)+(N2+1) TO (N1+2)+(N2*2) +(N3*2)
140 G(I)=2
150 NEXT I
160 PRINT:PRINT
170 RANDOMIZE TIMER
180 FOR I=1 TO NN
190 RM=INT((NN*RND(1)+1))
200 TEMP=G(I)
210 G(I)=G(RM)
220 G(RM)=TEMP
230 NEXT I
240 PRINT “GENE FREQUENCY IN THE INITIAL POPULATION”
250 K=1
260 FOR I=1 TO NN:PRINT G(I);” “;:NEXT I
270 RANDOMIZE TIMER
280 FOR I=1 TO NN
290 R1=INT((NN*RND(1)+1))
300 R2=INT((NN*RND(1)+1))
310 IF G(R1)=1 AND G(R2)=1 THEN A2=A2+1
320 IF G(R1)=2 AND G(R2)=2 THEN B2=B2+1
330 IF G(R1)<>G(R2) THENAB=AB+1
340 NEXT I
350 p\PRINT:PRINT
360 A=2*A2+AB : B=2*B2+AB
370 PRINT “FREQUENCY OF A=”;A : PRINT “FREQUENCY OF a=”;B
380 PRINT “AA=”;A2; “Aa=”;AB;”aa=”;BB
390P=A/(A+B): Q=B/(A+B)
400 PRINT “P=”;P;”  Q=”;Q
410 X=0
420 FOR I=1 TO NN
430 R=RND(1)
440 IF R<P THEN X=X+1
450 NEXT I
460 PRINT “X=”;X
470 IF X=0 THEN PRINT “LOSS OF A”:GOTO 920
480 FOR I=1 TO NN
490 IF I<=X THEN G(I)=1 ELSE G(I)=2
500 NEXT I
510 X=0
520 FOR I=1 TO NN:PRINT G(I);” “; : NEXT I
530 FOR I=1 TO NN
```

```
540 RANDOMIXE TIMER
550 RM=INT((NN*RND(1)+1))
560 TEMP=G(I)
570 G(I)=G(RM)
570 G(RM)=TEMP
590 NEXT I
600 PRINT: PRINT "**********************"
610 FOR I=1 TO NN:PRINT G(I);"  ";:NEXT I
620 A2=0:B2=0:AB=0
630 RANDOMIZE TIMER
640 FOR I=1 TO NN
650 R1=INT((NN*RND(1)+1))
660 R2=INT((NN*RND(1)+1))
670 IF G(R1)=1 AND G(R2)=1 THEN A2=A2+1
680 IF G(R1)=2 AND G(R2)=2 THEN B2=B2+1
690 IF G(R1)<>G(R2) THEN AB=AB+1
700 NEXT I
710 PRINT "AA=";A2;"  Aa=";AB;"  AA=";B2
720 A=(2*A2)+AB :B=(2*B2)+AB:P=A/(A+B):Q=B/(A+B)
730 PRINT "P=";P;"  Q=";Q
740 E=SQR((P+Q)/(2*N))
750 PRINT "SAMPLING ERROR="; E
760 IF P=1 THEN PRINT "FIXATION OF A " : GOTO 920
770 IF Q=1 THEN PRINT "FIXATION OF a ":GOTO 920
780 RANDOMIZE TIMER
790 FOR I=1 TO NN
800 R-RND(1)
810 IF R<P THEN X=X+1
820 NEXT I
830 IF X=0 THEN PRINT "LOSS OF A ":GOTO 840
840 FOR I=1 TO NN
850 IF I<=X THEN G(I)=1 ELSE G(I)=2
860 NEXT I
870 FOR I=1 TO NN: PRINT G(I);" "; :NEXT I
880 K=K+1
890 PRINT "GENERATION=";K
900 NEXT I
910 IF K<NG THEN 620
920 END
```

LISING OF PRGRAM FOR GENETIC DRIFT WITH MUTATION
DRIFTMR.BAS   (GENETIC DRIFT WITH MUTATION)

```
10 REM GENETIC DRIFT WITH MUTATION:PROGRAMMER:S.S.SHAUKAT
20 REM DIPLOID POPULATION C,c WHERE C IS CARBONARIA
30 PRINT "INITIAL GENEOTYPIC FREQUENCIES e.g., CC=5,Cc=10,cc=5 "
40 DIM G(1000)
50 INPUT "FREQUENCY OF CC INDIVIDUALS= ";N1
60 INPUT "FREQUENCY OF Cc INDIVIDUALS= ";N2
70 INP[UT "FREQUENCY OF cc INDIVIDUALS= ";N3
80 INPUT "HOW MNY GENERATIONS=";NG
90 INPUT "MUTATION RATE=";MR
100 INPUT "OUTFILE NAME ";F$
110 OPEN "O",1,F$
120 N=N1+N2+N3 :NN=N*2:PRINT "N=";N;" 2*N=";NN
130 PRINT "TOTAL NUMBER OF INDIVIDUALS=";N
140 REM A=1 , a=2
```

```
150 FOR I=1 TO (N1*2)+N2
160 G(I)=1
170 NEXT I
180 FOR I=(N1*2)+(N2+1) TO (N1*2)+(N2*2)+(N3*2)
190 G(I)=2
200 NEXT I
210 PRINT :PRINT
220 REM SHUFFLE OF ALLELES (YATES METHOD)
230 RANDOMIZE TIMER
240A1=0 : B1=0
250 FOR I=1 TO NN
260 RF=INT((NN*RND(1)+1))
270 TEMP=G(I)
280 G(I)=G(RF)
290 G(RF)=TEMP
300 NEXT I
310 PRINT “SHUFFLED ALLELES “
320 FOR I=1 TO NN :PRINT G(I) : NEXT I
330 FOR I=1 TO NN
340 IF G(I)=1 THEN A1=A1+1
350 IF G(I)=2 THEN B2=B2+1
360 NEXT I
370 PRINT “GENE FREQUENCY IN INITIAL POPULATION “
380 PRINT TAB(7); “INITIAL GENE FREQUENCY “
390 FOR I=1 TO NN : PRINT G(I): NEXT I
400 S1=0
410 K=1
420 P=A1/NN :Q=B1/NN:PRINT “P=”;P; “  Q=”;Q
430 REM BINOMIAL SAMPLING
440 PRINT “GENERATION “;K
450 X=0
460 RANDOMIZE TIMER
470 FOR I=1 TO NN
480 R=RND(1)
490 IF R<P THEN X=X+1
500 NEXT I
510 IF X=0 THEN P=0:PRINT “P=”;0;”  Q=”;1;” LOSSS OF C “:GOT 800
520 FOR I=1 TO X:G(I)=1:NEXT I
530 FOR I=X+1 TO NN:G(I)=2: NEXT I
540 REM MUTATION FROM c TO C
550 RANDOMIZE TIMER
560 FOR J=1 TO NN
570 RR=RND(1)
580 IF RR <= MR AND G(J)=2 THEN G(J)=1 : IF G(J)=1 THEN S1=S1+1
590 NEXT J
600 RANDOMIZE TIMER
610 FOR I=1 TO N
620 R1=INT((NN*RND(1)+1))
630 R2=INT((NN*RND(1)+1))
640 IF G(R1)=1 AND G(R2)=1 THEN A2=A2+1
650 IF G(R1)=2 AND G(R2)=2 THEN B2=B2+1
660 IF G(R1)<>G(R2) THEN AB=AB+1
670 NEXT I
680 PRINT “CC=”;A2;” cc=”;B2;” Cc=”;AB
690 A=(2*A2)+AB: B=(2*B2)+AB: P=A/(A+B)”Q=B/(A+B): PRINT “C=”;A;”  c=”;B
700 PRINT “P=”;P; “ Q=”;Q
```

```
710 PRINT #1, NG, P
720 E=SQR((P*Q)/(2*N))
730 PRINT “SAMPLING ERROR=”; E
740 INPUT “PRESS ANY NUMBER “;Z
750 IF P=1  THEN PRINT “FIXATION OF C “ : GOTO 800
760 IF Q=1 THEN PRINT “FIXATION OF c “ : GOTO 800
770 K=K+1 :IF K> NG THEN 800
780 A=0:B=0:A2=0:B2=0:AB=0
790 IF K<= NG THEN 430
800 PRINT “OUTFILE NAME= ”;F$
810 CLOSE #1
820 PRINT “NUMBER OF TIMES MUTATION OCCURRED “;S1
830 PRINT “MUTATION RATE “;MR
840 END
```

LISTING OF PROGRAM PREYFIL.BAS THAT CREATES FILE PREY FOR RNDWALK

```
10 REM PROGRAM PREYFIL TO CREATE DATA FILE ‘PREY’
20 INPUT “PROVIDE OUTFILE NAME VIZ. PREY “;F$
30 INPUT “HOW MANY PREY IN THE GRID “;N
40 OPEN “O”,1,F$
50 DIM X(N),Y(N)
50 REM CREATING RANDOM LOCATIONS OF THE PREYS
60 RANDOMIZE TIMER
70 FOR I=1 TO N
80 X(I)=RND(1)
90 Y(I)=RND(1)
100 NEXT I
110 FOR I=1 TO N
120 WRITE #1, X(I),Y(I)
130 NEXT I
140 CLOSE #1
150 END
```

LISTING OF RNDWALK.BAS (RANDOM WALK PREDATOR-PREY MODEL)

```
10 REM RANDOM WALK PREDATOR-PREY MODEL: PROGRAMMER:S.S.SHAUKAT
20 INPUT “NUMBER OF ROWS=NUMBER OF COLUMNS IN SQUARE GRID=”;R
30 INPUT “PREY FILE NAME “;F$
50 INPUT “TOTAL NUMBER OF PREY IN THE AREA “;M
60 INPUT “NUMBER OF PREDATORS E.G, 1-4 “;NP
70 INPUT “MAXIMUM INTAKE OF A PREDATOR IN A GRID UNIT E.G., 5 “;MXINT
80 INPUT  “MAXIMUM TIME “;TMAX
90 INPUT “PROBABILITY OF CAPTURE E.G., >=0.8 “;PC
100 S=1/R :NC=1-PC
110 DIM X(M),Y(M), N(R,R),P(10),Q(10),P1(10),Q1(10)
120 OPEN “I”,1,F$
130 FOR I=1 TO M
140 INPUT #1,X(I),Y(I)
150 NEXT I
160 CLOSE #1
170 L1=0: L2=0:M1=0M2=0:T1=0
180 FOR I=1 TO R
190 M2=M2+S :M1=M2-S
200 FOR I=1 TO R
210 L2=L2+S: L1=L2-S
220 FOR K=1 TO M
230 IF X(K)< L1 AND X(K) <=L2 THEN 250
```

```
240 GOTO 260
250 IF Y(K)>M1 AND Y(K)<=M2 THEN NB=NB+1
260 NEXT K
270 N(I,J)=NB
280 T1=T1+N(I,J)
290 NB=0
300 NEXT J
310 L2=0:L1=0
320 NEXT J
330 PRINT “TOTAL NUMBER OF PREY “;T1
340 FOR I=1 TO R
350 FOR J=1 TO R
360 PRINT N(I,J);” “;
370 NEXT j: PRINT :NEXT I
380 RANDOMIZE TIMER
390 PRINT “INITIAL PREDATOR LOCATION “
400 FOR Z=1 TO NP
410 P1(Z)= INT (RND(1)*R)+1
420 Q1(Z)= INT (RND(1)*R)+1
430 PRINT “INITIAL LOCATION OF PREDATOR “;Z;” “;P1(Z),Q1(Z)
440 NEXT Z
450 T=0: NOTCAP=0
460 REM TIME LOOP
470 FOR H=1 TO NP
480 PRINT “PREDATOTR “;H
490 IF T=0 THEN I=P1(I):J=Q1(J)
500 REM TO SUBROUTINE MOVE
510 GOSUB 830
520 ENC(H)=ENC(H)+N(I,J)
530 ENC2(H)=ENC2(H)+N(I,J)^2
540 Z1=0
550 FOR L=1 TO N(I,J)
560 RQ=RND(1):IF RQ<=NC THEN NOTCAP=NOTCAP+1 :Z1=Z1+1
570 REM TOTAL PREY NOT CAPTURED  NC FOR 1 PREDATOR
580 NEXT L
590 REM SUBTRACT THOSE NOT CAPTURED
600 N(I,J)=N(I,J)-Z1
610 IF N(I,J) <=MXINT THEN 640
620 INTK(H)=INTK(H)+MXINT
630 INTK2(H)=INTK2(H)+MXINT^2
640 INTK(H)=INTK(H)+N(I,J)
650 INTK2=INTK2(H)+N(I,J)^2
660 REM INTAKE & ENCOUNTER OF EACH PREDATOR
670 PRINT “INTAKE”;INTK(H);” ENCOUNTER”;ENC(H);” I=”;I;” J=”;J
680 P(H)=I :Q(H)=J
690 N(I,J)=N(I,J)+Z1
700 NEXT H
710 T=T+1
720 IF T< TMAX THEN 400
730 PRINT TAB(8);”**************SUMMARY OF RESULTS*************”
740 FOR I=1 TO NP
750 AVENC(I)=ENC(I)/T:SDENC(I)=SQR((ENC2(I)-ENC(I)^2/T/(T-1))
760 AVINTK(I)=INTK(I)/T:SDINT(I)=SQR((INTK2(I)-INTK(I)^2/T/(T-1))
770 PRINT TAB(5);”AVERAGE ENCOUNTER OF PREDATOT “;I;”=”;AVENC(I)
780 PRINT TAB(5);”SD OF ENCOUNTER OF PREDATOR “;I;”=”;SDENC(I)
790 PRINT TAB(5);”AVERAGE CONSUMPTION OF PREDATOR “;I;”=”;AVINTK(I)
```

```
800 PRINT TAB(50;"SD OF CONSUMPTION OF PREDATOR ";I;"=";SDINT(I)
810 NEXT I
820 PRINT TAB(6);"TOTAL NUMBER OF TIMES PREY WAS NOT CAPTURED ";NOTCAP
830 END
840 REM MOVING THE PREDATOR ON GRID (RANDOM WALK) SUBROUTINE)
850 PR=RND(1)
860 IF PR<=0.25 THEN 990
870 IF PR>0.25 AND PR<=0.5 THEN 930
880 IF PR >0.5 AND PR<=0.75 THEN 960
890 IF PR >0.75 THEN 990
900 I=I-1
910 IF I<1  THEN I=R
920 GOTO 1010
930 I=I+1
940 IF I>R THEN I=1
950 GOTO 1010
960 J=J+1
970 IF J>R THEN J=1
980 GOTO 1010
990 J=J-1
1000 IF J<1 THEN J=R
1010 RETURN
```

All programs are available from the first author (SSS) on request.