EXPERIMENT – IX

Anirban Mukherjee

16BIT0200

POPULATION DYNAMICS

Aim:

Analyzing and solving Difference equation for the population growth.

Problem Statements:

We divide this experiment in two parts

i) Formulation of population growth (Human population) by first order difference equation.

ii) Formulation of population growth of rabbits in terms of second order difference equation (through Fibonacci numbers).

Matlab Commands:

- ztrans – z-transformation
- iztrans – Inverse z-transformation
- stem – Plots discrete sequence data
- strcat – Concatenate strings horizontally

Description of Physical Experiment:

Modelling of Human Population growth:

In adopting a growth model to human populations, we must take into account births as well as deaths. A plausible assumption is that births and deaths both occur at a rate which is proportional to the size of N of the population at any time t.

The governing equation of Human population can be written in Difference form as:

$$\frac{dN}{dt} = \frac{N_{n+1} - N_n}{\Delta t} = \alpha N_n - \beta N_n = kN_n$$

where $k = (\alpha - \beta), \Delta t = 1$

Fibonacci Numbers

The original problem that Fibonacci investigated was about how fast rabbits could breed in ideal circumstances. It is not only rabbits but also cows, bees and birds and perhaps several other animals and insects breed the same way. It follows a particular pattern:

In case of Rabbits

t=0	1 month	2	3	. . .
1 pair	1 pair	2	3	. . .

If we change months into years, we get how cow pair population changes.

<u>Example: Rabbit Population</u>

Consider this problem, which was originally, posed by Leonardo Pisano, also known as Fibonacci, in the thirteenth century in his book Liber abaci. A young pair of rabbits (one of each sex) is placed on an island. A pair of rabbits does not breed until they are 2 months old.

After they are 2 months old, each pair of rabbits produces another pair each month, as shown in Figure 1. Find a difference equation for the number of pairs of rabbits on the island after n months, assuming that no rabbits ever die.

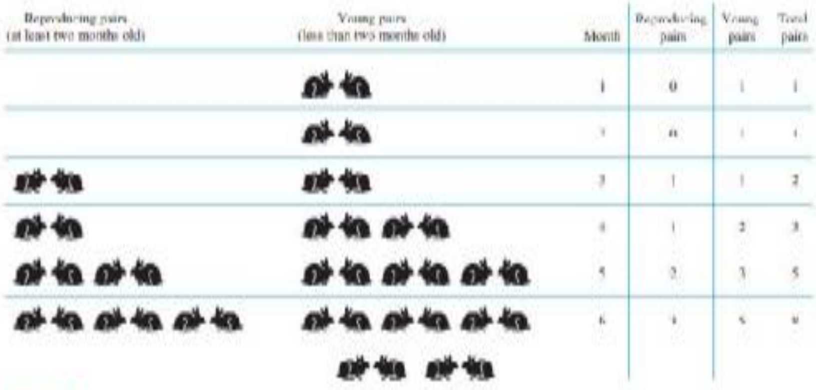


FIGURE 1 Rabbits on an Island.

Solution:

Denote by f_n the number of pairs of rabbits after n months. We will show that f_n, n = 1, 2, 3,..., are the terms of the Fibonacci sequence.

The rabbit population can be modelled using a recurrence relation. At the end of the first month, the number of pairs of rabbits on the island is $f_1 = 1$. Because this pair does not breed during the second month, $f_2 = 1$ also. To find the number of pairs after n months, add the number on the island the previous month, f_{n-1}, and the number of newborn pairs, which equals f_{n-2}, because each newborn pair comes from a pair at least 2 months old. Consequently, the sequence $\{f_n\}$ satisfies the recurrence relation $f_n = f_{n-1} + f_{n-2}$ for $n \geq 3$ together with the initial conditions $f_1 = 1$ and $f_2 = 1$. Because this recurrence relation and the initial conditions uniquely determine this sequence, the number of pairs of rabbits on the island after n months is given by the nth Fibonacci number.

Note: (Petals on a flower)

On many plants, the number of petals is a Fibonacci number. Leaf arrangement like branches on a tree-the sun flower, vegetable like

Cauliflowers follow Fibonacci sequence. In fact, the Fibonacci numbers are natural numbers.

Note: (Golden Number)

Fibonacci numbers and the golden number: 1,1,2,3,5,8,13,…

If we take the ratio of two successive numbers in Fibonacci series, we obtain the following series of numbers.

1/1=1; 2/1=2; 3/2=1.5; 5/3=1.666; 8/5=1.6,…

The ratio seems to be settling down to a particular value, which we call the golden ratio or the golden number. It has a value approximately equal to 1.618034.

Connection to Mathematics:

The Human Population Growth can be solved by:

First order difference equation:

Where N_0 represents the initial population

A general method for solving such difference equations may now be explained. Consider the same equation

$$N_{n+1} = (1 + k)N_n$$

We assume $N_n=a^n$. Substituting into the given equation, we get

$$a^{n+1} = (1 + k)a^n$$

Thus, the solution satisfying the condition $N(0) = N_0$ is:

$$N_n = N_0(1 + k)^n$$

Differential form: (The problem is now discussed in Differential form)

$$\frac{dN}{dt} = \alpha N - \beta N = (\alpha - \beta)N = kN, k > 0$$

Where α and β are positive constants denoting the average rate of births and deaths.

This simple equation was proposed in 1798 by the English economist Thomas Mathews as a basic model for population growth. If

$N(t_0) = N_0$, we have:

$$N(t) = N_0 e^{k(t-t_0)}$$

Solving Rabbit Population Problem using Second Order Difference Equations

The Rabbit population can be predicted by solving the following second order Fibonacci difference equation:

$F_n = F_{n-1} + F_{n-2}$

Or

$F_{n+2} = F_n + F_{n+1}$, with $F_0 = 0$, $F_1 = 1$

The complete solution of the Fibonacci equation is:

$$F_n = \frac{1}{2^n\sqrt{5}}\left[\left(1+\sqrt{5}\right)^n - \left(1-\sqrt{5}\right)^n\right]$$

Code:

For finding solution of Fibonacci Difference equation using z-transform

The codes are for finding population growth for the given input when y(n+2): 1, y(n+1): -1 and y(n): -1

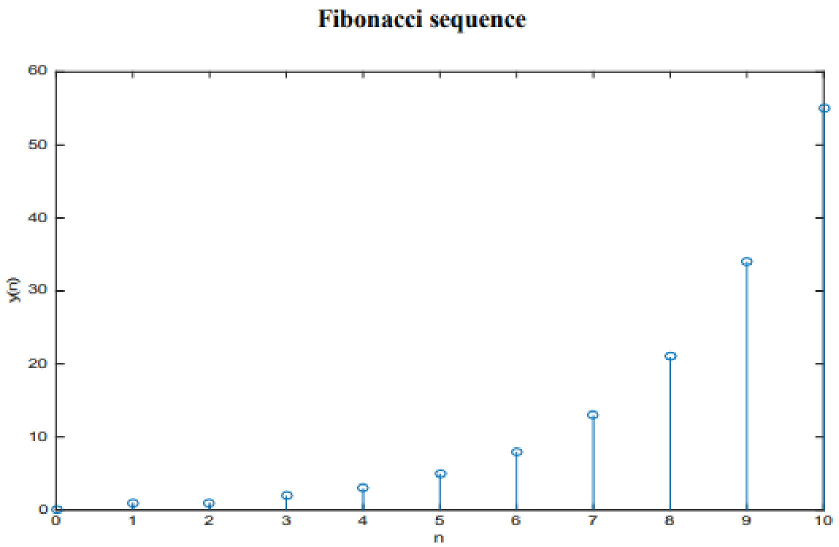
```
clc
clear all
syms z Y n positive
LHS=ztrans(sym('y(n+2)')-sym('y(n+1)')-sym('y(n)'),n,z);
RHS=ztrans(0,n,z);
newLHS=subs(LHS,{'ztrans(y(n),n,z)','y(0)','y(1)'},{Y,0,1});
Y=solve(newLHS-RHS,Y);
y=iztrans(Y,z,n)

y =

(5^(1/2)/5)*(5^(1/2)/2 + 1/2)^n + (-5^(1/2)/5)*(1/2 -
5^(1/2)/2)^n
```

For finding solution of linear second order Non-homogenous Difference equation

```
clc
clear all
symsnk1k2m
assume(n,'integer')
a = input('Enter the coefficient of y(n+2): ');
b = input('Enter the coefficient of y(n+1): ');
c = input('Enter the coefficient of y(n): ');
g = input('Enter the non-homogeneous part: ');
r = subs(solve(a*m^2+b*m+c,m));
ifimag(r)~=0
rho = sqrt(real(r(1))^2 + imag(r(1))^2);
theta = atan(abs(imag(r(1)))/real(r(1)));
 y1 = (rho^n)*cos(n*theta);
 y2 = (rho^n)*sin(n*theta);
elseif r(1)==r(2)
 y1 = r(1)^n;
 y2 = n*r(1)^n;
else
 y1 = r(1)^n;
 y2 = r(2)^n;
end
Co = det([y1, y2;subs(y1,n,n+1), subs(y2,n,n+1)]); %Casoratian of the
solutions
y_c = k1*y1 + k2*y2;
disp('Complementary Solution is: ');
disp(y_c);
if(g ~= 0)
 y11 = subs(y1,n,n+1);
```

```
 y21 = subs(y2,n,n+1);
 Co1 = subs(Co,n,n+1);
 u1 = simplify(symsum(-g*y21/Co1,n))
 u2 = simplify(symsum(g*y11/Co1,n))
y_p = simplify(u1*y1+u2*y2);
 y = y_c + y_p;
else
 y = y_c;
end
check = input('If the given problem has initial conditions then enter 1
else enter 0: ');
if (check == 1)
 yval1 = input('Enter the initial condition at n = 0: ');
 yval2 = input('Enter the initial condition at n = 1: ');
 cond1 = strcat(char(subs(y,n,0)),'=',num2str(yval1));
 cond2 = strcat(char(subs(y,n,1)),'=',num2str(yval2));
 [k1,k2] = solve(cond1,cond2);
 y = subs(y);
end
disp('Complete Solution is: ')
disp(collect(collect(y,y1),y2))
if(check ~= 0)
nrange = 0:10;
 Y = subs(y,n,nrange);
 stem(nrange,Y);
 set(gca,'XTick',linspace(0,10,11))
xlabel('n');
ylabel('y(n)');
end
```

```
Enter the coefficient of y(n+2): 1

Enter the coefficient of y(n+1): -1

Enter the coefficient of y(n): -1

Enter the non-homogeneous part: 0

Complementary Solution is:

k1*(1/2 - 5^(1/2)/2)^n + k2*(5^(1/2)/2 + 1/2)^n

If the given problem has initial conditions then enter 1 else enter 0: 1

Enter the initial condition at n = 0: 0

Enter the initial condition at n = 1: 1

Complete Solution is:

(5^(1/2)/5)*(5^(1/2)/2 + 1/2)^n + (-5^(1/2)/5)*(1/2 - 5^(1/2)/2)^n
```



My Work:

This program can be used for data analytics, we can use the program for Fibonacci equation for calculating the rate of population growth of a country and we can find the golden ratio which we can develop into tools using Python language for architecture purposes.

Fibonacci heaps have a faster amortized running time than other heap types for sorting of data as well. Fibonacci heaps are similar to binomial heaps but Fibonacci heaps have a less rigid structure.

Binomial heaps merge heaps immediately but Fibonacci heaps wait to merge until the extract-min function is called. While Fibonacci heaps have very good theoretical complexities, in practice, other heap types such as pairing heaps are faster. This is because even in the simplest implementation, Fibonacci heaps require four pointers for each node, other heaps need two or three.

This can allow sorting of huge amounts of data at a very less time when implemented.

(heap is a specialized tree-based data structure)