Mathematical Literacy for Software Engineers – Part 5

Praseed Pai K.T.

A Formula for Computing Sum of the GP

$$a,\ ar,\ ar^2,\ ar^3,\ ar^4,\ \ldots$$

$$\sum_{k=1}^{n} ar^{k-1} = ar^0 + ar^1 + ar^2 + ar^3 + \cdots + ar^{n-1}.$$

$$\begin{aligned}
(1-r)\sum_{k=1}^{n} ar^{k-1} &= (1-r)(ar^0 + ar^1 + ar^2 + ar^3 + \cdots + ar^{n-1}) \\
&= ar^0 + ar^1 + ar^2 + ar^3 + \cdots + ar^{n-1} - ar^1 - ar^2 - ar^3 - \cdots - ar^{n-1} - ar^n \\
&= a - ar^n
\end{aligned}$$

$$\sum_{k=1}^{n} ar^{k-1} = \frac{a(1-r^n)}{1-r}.$$

How to Compute EMI?

$$P = A \cdot \frac{1 - (1 + r)^{-n}}{r}$$

$$A = P \cdot \frac{r(1 + r)^n}{(1 + r)^n - 1}$$

Equated monthly Installment (aka EMI)

```java
import java.lang.*;



public class Emi {
  public static double Get_Emi( double principal , double rate , long n ){
       double percent_rate = rate/(12.0*100.0);
       return  principal/(( 1 - Math.pow(1+percent_rate,-n*12.0))/percent_rate);
  }
}
```

$$P = A \cdot \frac{1-(1+r)^{-n}}{r}$$

$$A = P \cdot \frac{r(1+r)^n}{(1+r)^n - 1}$$

Derivation of the EMI formulathe EMI formula

P1 = P x (1 + r) - E ----- [1]

P2 = P1 x (1 + r) - E ----- [2]

Substituiting [1] in [2]
P2 = (P x (1 + r) - E) x (1 + r) - E

Distributing (1 + r) into (P x (1 + r) - E)
P2 = (P x (1 + r)^2 - E x (1+r) - E

After Reduction
= (P x (1 + r)^2) - Ex((1+r) + 1)

Let (1 + r) => t
= (Pxt^2 - Ex(t + 1)

Expand the Algebraic equation for ith Period
Pi = P x t^i - Ex(1+t + t2 + ..+ t^(i-1))

Expand the Algebraic equation for the nth Period
Pn = P x t^n - Ex(1+t + t^2 + t^3 +....+ t^(n-1)) = 0

Pxt^n = E x (1 + t + t^2 + ... + t^(n-1))

Simplyfying the above equation using GP formula
Pxt^n = E x (t^n -1)/(t-1)
After Transposition of Factors
E = P x t^n x (t -1)/(t^n - 1)

Substituite t => (1 + r)
E = P x (1+r)^n x ((1+r)-1) / ((1+r)^n - 1)
After Suitable Reduction
= P + (1 + r)^n x (r) / ((1+r)^n - 1)
= P x r x (1 + r)^n / (1 + r)^n - 1)

$$E = P \cdot r \cdot \frac{(1+r)^n}{((1+r)^n - 1)}$$

IRR for two Periods?

$$100 = \frac{60}{1+r} + \frac{60}{(1+r)^2}.$$

$$60x^2 + 60x - 100 = 0,$$

$$x = \frac{-60 \pm \sqrt{60^2 + 4(60)(100)}}{120}.$$

$$x = \frac{\sqrt{27{,}600} - 60}{120} \approx .8844.$$

$$1 + r^* \approx \frac{1}{.8844} \approx 1.131.$$

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
public static double Quadratic(double a, double b, double c,
                               ref double rt1 , ref double rt2 )
{
    double disc = b*b - 4*a*c;

    if (disc  < 0.0 )
        return Double.NaN;


    double root1 = (-b + Math.Sqrt(disc)) / (2*a) ;
    double root2 = (-b - Math.Sqrt(disc)) / (2*a) ;
    rt1 = root1;
    rt2 = root2;
    return 0.0;
 }
```

IRR – The math

$$\mathrm{NPV} = \sum_{n=0}^{N} \frac{C_n}{(1+r)^n} = 0$$

If an investment may be given by the sequence of cash flows

Year (n)	Cash flow (C_n)
0	-123400
1	36200
2	54800
3	48100

then the IRR r is given by

$$\mathrm{NPV} = -123400 + \frac{36200}{(1+r)^1} + \frac{54800}{(1+r)^2} + \frac{48100}{(1+r)^3} = 0.$$

In this case, the answer is 5.96% (in the calculation, that is, r = .0596).

How to Compute IRR for longer periods?

```
public static double CashFlowPVDiscreteAnnual(List<double> arr,
                double rate, double period) {
        int len = arr.Count;
        double PV = 0.0;
        for (int i = 0; i < len; ++i){
            PV += arr[i] / Math.Pow((1+rate/100), i);
        }
            return PV;
    }
    public static double CashFlow_IRR_Annual(List<double> arr,
                double rate,double period) {
        const double ACCURACY = 1.0e-5;
        const int MAX_ITERATIONS = 50;
        const double ERROR = -1e30;
        double x1 = 0.0; double x2 = 20.0;
        // create an initial bracket,
        // with a root somewhere between bot,top
        double f1 = CashFlowPVDiscreteAnnual(arr,x1,period);
        double f2 = CashFlowPVDiscreteAnnual(arr,x2,period);
        for (int j = 0; j < MAX_ITERATIONS; ++j){
            if ( (f1*f2) < 0.0) { break; }
            if (Math.Abs(f1) < Math.Abs(f2)) {
              f1 = CashFlowPVDiscreteAnnual(arr,x1+= 1.06*(x1-x2),period );
            }
            else {
              f2 = CashFlowPVDiscreteAnnual(arr,x2+=1.06*(x2-x1),period);
            }
        }

        if (f2*f1>0.0) { return ERROR; }
        double f = CashFlowPVDiscreteAnnual(arr,x1,period);
        double rtb; double dx=0;
        if (f<0.0)  { rtb = x1; dx=x2-x1;} else { rtb = x2; dx = x1-x2; }
        for (int i=0;i<MAX_ITERATIONS;i++) {
              dx *= 0.5; double x_mid = rtb+dx;
              double f_mid = CashFlowPVDiscreteAnnual(arr,x_mid,period);
              if (f_mid<=0.0) { rtb = x_mid; }
              if ( (Math.Abs(f_mid)<ACCURACY) || (Math.Abs(dx)<ACCURACY) ) { return x_mid;}
        }
        return ERROR; // error.

    }
}
```

Put Everything together

```
public class EntryPoint {

    public static void Main(String [] args ) {
        List<double> returns = new List<double>();
        returns.Add(-100);
        returns.Add( 60 );
        returns.Add( 60 );
        double dcfactor =  PV.CashFlow_IRR_Annual(returns,0,2);
        Console.WriteLine(dcfactor);
        List<double> returns2 = new List<double>();
        returns2.Add(-123400);
        returns2.Add( 36200 );
        returns2.Add( 54800 );
        returns2.Add( 48100 );
        double dcfactor2 =  PV.CashFlow_IRR_Annual(returns2,0,3);
        Console.WriteLine(dcfactor2);
    }

  }
```

Output of the Console

```
D:\cpp>IRR
13.066234588623
5.96163749694824
```

Q&A

- If any!
- https://github.com/praseedpai/ElementaryMathForProgrammingSeries/blob/master/AlgebraNArith/EMI/Emi.cs
- https://github.com/praseedpai/ElementaryMathForProgrammingSeries/blob/master/AlgebraNArith/EMI/Emi.java
- https://github.com/praseedpai/ElementaryMathForProgrammingSeries/blob/master/AlgebraNArith/EMI/emi.cpp
- https://github.com/praseedpai/ElementaryMathForProgrammingSeries/blob/master/AlgebraNArith/IRR/PV.cs
- https://github.com/praseedpai/ElementaryMathForProgrammingSeries/blob/master/AlgebraNArith/NthQuad/QuadNth.cs