CHAPTER-2
BERNOULLI NUMBERS AND THEIR COMPUTER PROGRAMS*

* Published:

“A Note on Bernoulli Numbers and Their Computer Programs”, in Ultra Scientist of Physical Sciences, Vol. 25 (3) A, 2013, pp. 399-404
CHAPTER-2

Bernoulli Numbers and Their Computer programs

2.1. **Introduction:** The Bernoulli numbers are a set of numbers discovered by Jacob Bernoulli (1654-1705). This set of numbers holds a deep relationship with the Riemann Zeta function. The Riemann Zeta Function has been found to have a relationship with Prime numbers. The Bernoulli numbers have also been found to be useful for proofs of a restricted version of Fermat’s Last Theorem. In this chapter, we give there interesting computer program for power sum problem for generating the Bernoulli numbers.

Bernoulli’s numbers play an important role and quite mysterious role in mathematics and in various places like analysis, number theory and differential topology. They first appeared in Ars Conjectandi, Page 97 [6], a famous (and posthumous) treatise published in 1713, by Jacob Bernoulli (1654-1705) when he studied the sums of Powers of consecutive integers

\[ S_r(n) = \sum_{k=1}^{n-1} k^r \]  

(2.1. 1)

where, \( r \) and \( n \) are two given positive integers.

Bernoulli numbers are a sequence of rational numbers that arise in a dazzling variety of applications in analysis, numerical analysis and number theory. When Charles Babbage designed the Analytical
Engine in the 19th century, one of the most important tasks he hopped the Engine would perform was the calculation of Bernoulli numbers [53].

The Bernoulli numbers [6, 78] also appear in many mathematical expressions, such as –

(i) The Taylor expansion in a neighborhood of the origin of the circular and hyperbolic tangent and co-tangent functions;
(ii) The sums of power of natural numbers;
(iii) The residual term of Euler-Maclaurin quadrature rule.

Bernoulli numbers also appear in the computation of the numbers

\[ \xi(2p) = \sum_{k=1}^{\infty} \frac{1}{k^{2p}} \]  \quad (2.1.2)

And in the expansion of many usual functions as \( \tan x, \tanh (x), \frac{1}{\sin(x)}, \) and others [79].

The Bernoulli Polynomials have important applications in number theory and classical analysis. They appear in the integral representation of differentiable periodic function since they are employed for approximating such function in terms of polynomials. They are also used for representing the remainder term of the composite Euler-Maclaurin quadrature rule [90].

2.2. A Power Sum Problem: -The great Swiss mathematician Jacob Bernoulli (1654-1705) has used these numbers in the power sum problem. The power sum problem is to find a formula for the sum of the \( r \)-th powers of the first \( n \) natural numbers for positive integer, [6].
\[ S_r(n) = 1^r + 2^r + 3^r + \ldots + (n-1)^r = \sum_{k=1}^{n-1} k^r \]  

(2.2.1)

\[ S_r(n) \] satisfies the difference equation

\[ S_r(n+1) - S_r(n) = n^r \]  

(2.2.2)

with initial condition

\[ S_r(0) = 0 \]  

(2.2.3)

It is easy to derive an expression for \( S_r(n) \) for the first few values of \( r \).

\[ S_1(n) = \frac{n(n-1)}{2} \]  

(2.2.4)

\[ S_2(n) = \frac{n(n-1)(2n-1)}{6} \]  

(2.2.5)

\[ S_3(n) = \frac{n^2(n-1)^2}{4} \]  

(2.2.6)

For higher values of \( r \) the sum formula \( S_r(n) \) becomes very complicated. For computation of these sums of power series of consecutive integers we give the following computer program and are verified.

**Program to generate Series:**

```csharp
private void button1_Click(object sender, EventArgs e)
{
    double n = Convert.ToDouble(textBox1.Text);
```
double r = Convert.ToDouble(textBox2.Text);

double sum = 0;

for (double i = 0; i <= n - 1; i++)
{
    sum = sum + powerof(i, r);
}

MessageBox.Show(sum.ToString(), "Value of S");

public double powerof(double i, double r)
{
    double p = 1;

    for (double j = 1; j <= r; j++)
    {
        p = p * i;
    }

    return p; [77]

2.3. Generating Function Of Bernoulli Numbers: - Jacob Bernoulli then, empirically, noticed that the polynomials $S_r(n)$ have the form

$$S_r(n) = \frac{1}{r+1} n^{r+1} - \frac{1}{2} n^r + \frac{r}{12} n^{r-1} + 0 \times n^{r-2} \quad (2.3.1)$$
In this expression, the numbers \((1, -1/2, 1/12, 0, \ldots)\) are appearing and do not depend on \(r\). More generally, the sums \(S_r(n)\) can be written in the form

\[
S_r(n) = \sum_{k=0}^{r} \frac{B_k}{k!} \cdot \frac{r!}{(r + 1 - k)!} n^{r+1-k}
\]

(2.3.2)

Where, the \(B_k\), \(k=0, 1, 2,\) are the numbers which are independent of \(r\) and called Bernoulli’s numbers. An explicit formula for Bernoulli numbers is given by

\[
B_n = \sum_{k=0}^{n} \frac{1}{k+1} \sum_{r=0}^{k} (-1)^r \binom{k}{r} r^n
\]

(2.3.3)

The first few Bernoulli’s numbers are –

\[
B_0 = 1
\]

\[
B_1 = -1/2
\]

\[
B_2 = 1/6
\]

\[
B_4 = -1/30
\]

\[
B_6 = 1/42
\]

\[
B_8 = -1/30
\]

\[
B_{10} = 5/66,
\]

\[
B_{12} = -691/2730
\]

\[
B_{14} = 7/6
\]

\[
B_{16} = -3617/510
\]

More numbers are given in [1] and in [78]
After $B_1$ all Bernoulli’s number with odd index are zero and the non-zero ones alternate in sign.

To illustrate the usefulness of his formula, Bernoulli computed the astonishing value of $S_{10}(1000)$ with little effort (in less than “half a quarter of an hour” he says ….. [83]

$$1^{10}+2^{10}+\ldots+1000^{10} = 91409924241424243424241924242500 \quad (2.3.4)$$

To Verify the formula (2.3.2) for $S_r(n)$ and in particular value,

We give the following computer program.

(i) **Program for computing $S_r(n)$ from (2.3.2):**

```csharp
public partial class Bernoulli : Form
{
    Series er = new Series();

    double[] b = { 1, -1.0 / 2.0, 1.0 / 6.0, 0, -1.0 / 30.0, 0, 1.0 / 42.00, 0, -1.0 / 30.0, 0, 5.0 / 66.0, 0, -691.0 / 2730.0, 0, 7.0 / 6.0, 0, -3617.0 / 510.0, 0, 43867.0 / 798.0, 0, -174611.0 / 330.0, 0, 854513.0 / 138.0, 0, -236364091.0 / 2730.0, 0, 8553103.0 / 6.0, 0, -23749461029.0 / 870.0, 0, 8615841276005.0 / 14322.0, 0, -7709321041217.0 / 510.0, 0, 2577687858367.0 / 6.0, 0, -26315271553053477373.0 / 1919190.0, 0, 2929993913841559.0 / 6.0, 0, -261082718496449122051.0 / 13530.0 };

    public Bernoulli()
    {

        InitializeComponent();

        for (int j = 0; j <= 40; j++)
```
private void button1_Click(object sender, EventArgs e)
{
    double r = Convert.ToDouble(textBox1.Text);
    double n = Convert.ToDouble(textBox2.Text);
    double sum = 0;
    double s1 = 0;
    for (int i = 0; i <= r; i++)
    {
        double db = powerof(n + 1, r + 1 - i);
        s1 = (b[i] * factorial(r) * db) / (factorial(i) * factorial(r + 1 - i));
        sum = sum + s1;
    }
    MessageBox.Show(sum.ToString(), "Value of S");
}

public double factorial(double k)


double p = 1;

for (double i = 1; i <= k; i++)
{
    p = p * i;
}

return p;

}

public double powerof(double i, double k)
{

double p = 1;

for (double j = 1; j <= k; j++)
{
    p = p * i;
}

return p;
}

(ii) **Program for computing the Bernoulli number $B_n$:** - To compute the different values of Bernoulli number we give the following computer program using the formula (2.3.3)
public partial class GenerateBernoullics : Form
{
    public GenerateBernoullics()
    {
        InitializeComponent();
    }
}
Bernoulli br1 = new Bernoulli();

private void button1_Click(object sender, EventArgs e)
{
    double br=0,n,k,r,sum=0,a=0;
    n=Convert.ToDouble(textBox1.Text);
    if (n % 2 == 0 || n==1)
    {
        for (k = 0; k <= n; k++)
        {
            for (r = 0; r <= k; r++)
            {
                a = (br1.factorial(r) * br1.factorial((k - r)));
                br = br + (br1.powerof(-1, r) * br1.powerof(r, n) / a * br1.factorial(k));
            }
        }
    }
}
a = 0;
}
br = br / (k + 1);
sum = sum + br;
br = 0;
}
textBox2.Text = Convert.ToString(sum);
}
else
{

textBox2.Text = "0";
}

2.4. Concluding Remarks: In this chapter, we presents the computer programs for generating Bernoulli numbers from their generating function. The program for $s_r(n)$ (The power sum of $n$ natural numbers) have also been given. We have verified the programs by taking different examples.