MATH 466: Project 1

Dillon Stetler*, Justin Siao, Robert Mauban

November 2018

*Corresponding Author: Dillon Stetler

1.) The first computer program ever written was by Ada Lovelace who wrote a program for the Analytical Engine to compute Bernoulli numbers. The Bernoulli number B_n is given by $B_n = B_n(0)$ where $B_n(x)$ is the unique polynomial of degree n such that:

$\int_x^{x+1} B_n(t)dt = x^n$

Find B_n for $n = 0$, 1, 2 and 3 by substituting a polynomial of degree n into the integral and solving for the coefficients so that equality holds. You may use Maple or some other computer algebra system or do the calculation by hand.

SOLUTION:
The solution our group obtained is written in C programming language:

```c
#include <stdio.h>
#include <math.h>

double factorial(int x)
{
    double k = 1;
    while (x > 1){
        k *= x;
        x--;
    }
    return k;
}

double polyCoeff(int n, int r){
    if(r<1){
        return 0;
    }
    return factorial(n) / (factorial(r) * factorial(n-r));
}

int main(){
    //bernoulli numbers to be found
    int n=5;
    double poly[n][n];
    double calcArr[n][n];
    double scaleFactor;
    //generates matrix of pascal's triangle
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            poly[i][j] = polyCoeff(i+1,i-j+1)*1/(i+1);
        }
    }
    //generates bernoulli polynomials and numbers
    for(int i=0;i<n;i++){
        for(int j=i;j>=0;j--){
            for (int k=i;k>=0;k--){
                calcArr[j][k] = poly[j][k];
            }
        }
        //generates coefficients on terms of polynomial
        for(int j=i-1;j>=0;j--){
            scaleFactor = calcArr[i][j]*-1;
            for(int k=0;k<n;k++){
                calcArr[j][k] = calcArr[j][k]*scaleFactor;
                calcArr[i][k] = calcArr[j][k]+calcArr[i][k];
            }
        }
```

```c
        //reports bernoulli polynomials
        printf("The n=%d bernoulli polynomial is: \n",i);
        for(int j=i;j>=0;j--){
            if(j>1){
                printf("%fx^%d + ",calcArr[j][j],j);
            } else if(j==1){
                printf("%fx + ",calcArr[1][1]);
            } else {
                printf("%f\n",calcArr[0][0]);
            }
        }
        //reports bernoulli numbers
        printf("The n=%i bernoulli number is: %f\n",i,calcArr[0][0]);
        printf("\n");
    }
    return 0;
}
```

Output:

```
The n=0 bernoulli polynomial is:
1.000000
The n=0 bernoulli number is: 1.000000

The n=1 bernoulli polynomial is:
1.000000x + -0.500000
The n=1 bernoulli number is: -0.500000

The n=2 bernoulli polynomial is:
1.000000x^2 + -1.000000x + 0.166667
The n=2 bernoulli number is: 0.166667

The n=3 bernoulli polynomial is:
1.000000x^3 + -1.500000x^2 + 0.500000x + -0.000000
The n=3 bernoulli number is: -0.000000

The n=4 bernoulli polynomial is:
1.000000x^4 + -2.000000x^3 + 1.000000x^2 + -0.000000x + -0.033333
The n=4 bernoulli number is: -0.033333
```

2.) By the Fundamental Theorem of Calculus it follows that:

$\frac{d}{dx}\int_x^{x+1} B_n(t)dt = B_n(x+1) - B_n(x) = \int_x^{x+1} B_n'(t)dt.$

Use this fact to show that $B_n'(x) = nB_{n-1}(x)$.

SOLUTION:

First Fundamental Theorem: Let [a,b] be a closed bounded interval and let f be a function which is continuous on [a,b] and differentiable on (a,b) with f' integrable [a,b], then,

$\int_a^b f'(x)dx = f(b) - f(a).$

Proof:

We have that, $\int_x^{x+1} B_n'(t)dt = B_n(x+1) - B_n(x) = \frac{d}{dx}\int_x^{x+1} B_n(t)dt = \frac{d}{dx}x^n = nx^{n-1}$.

Then, $\int_x^{x+1} \frac{1}{n}B_n'(t)dt = x^{n-1}$.

Therefore, since $B_{n-1}(x)$ is the unique polynomial of degree $n-1$ such that $\int_x^{x+1} B_{n-1}(t)dt = x^{n-1}$ it follows that $\frac{1}{n}B_n'(t) = B_{n-1}(t)$.

3.) By the Fundamental Theorem of Calculus we also have,

$\int_0^x B_n'(t)dt = B_n(x) - B_n(0)$ or equivalently $B_n(x) = B_n + \int_0^x nB_{n-1}(t)dt$

Integrate the above equality in x from 0 to 1, then interchange the order of integration to obtain the relation that,

$B_n(x) = \int_0^1 tnB_{n-1}(t)dt$ for $n > 1$.

SOLUTION:

By integrating both sides of $B_n(x) = B_n + \int_0^x nB_{n-1}(t)dt$, we have,

$B_n(1) - B_n(0) = \int\limits_0^1 \int\limits_0^x nB_{n-1}(t)\,dt\,dx$

By interchanging the order of integration we have,

$B_n = \int\limits_0^1 \int\limits_t^1 nB_{n-1}(t)\,dx\,dt$

$B_n = \int_0^1 tnB_{n-1}(t)dt$

4.) Write $B_{n-1}(x) = \alpha_0 + \alpha_1 x + ... + \alpha_{n-1}x^{n-1}$ and use the identity,

$B_n(x) = \int_0^1 tnB_{n-1}(t)dt + \int_0^x nB_{n-1}(t)dt$

derived in the previous step to find formulas for B_n and $B_n(x)$ in terms of the α_k

SOLUTION:

5.) Starting with $B_1(x) = x_1 = 2$, write a program that computes the Bernoulli numbers by means of the formulas derived in the previous step. Use your program to print a table listing the values of n and B_n for n = 1,2,...,10.

SOLUTION:

```c
#include <stdio.h>
#include <math.h>

static double alpha[11];
static double temp[11];
static int n;

double generate_alpha(int n);
void generate_b(int);

int main(int argc, char **argv)
{
    alpha[0] = 1;
    alpha[1] = -0.5;
    for (int h = 2; h < 11; h++)
    {
        generate_alpha(h);
        generate_b(h);
        printf("%d\t\t", h);
        for (int y = 0; y <= h; y++)
        {
            printf("%3.6lf\t", alpha[y]);
        }
        putchar('\n');
    }
    return 0;
}

```

```c
double generate_alpha(int n)
{
    double return_val = 0;
    for (int u = 0; u < n; u++)
    {
        return_val += alpha[u] / (n-u+1);
    }
    return return_val * n;
}

void generate_b(int n)
{
    double r[n];
    for (int i = 0; i < n; i ++)
    {
        r[i] = alpha[i] * n / (n - i);
    }
    alpha[n] = generate_alpha(n);
    for (int i = 0; i < n; i ++)
    {
        alpha[i] = r[i];
    }
}
```

Output:

```
root@DESKTOP-OCN6246:/mnt/c/Users/Dillon/Desktop# clear
root@DESKTOP-OCN6246:/mnt/c/Users/Dillon/Desktop# ./Q5
2       1.000000        -1.000000       0.166667
3       1.000000        -1.500000       0.500000        0.000000
4       1.000000        -2.000000       1.000000        0.000000        -0.033333
5       1.000000        -2.500000       1.666667        0.000000        -0.166667       -0.000000
6       1.000000        -3.000000       2.500000        0.000000        -0.500000       -0.000000       0.023810
7       1.000000        -3.500000       3.500000        0.000000        -1.166667       -0.000000       0.166667-0.000000
8       1.000000        -4.000000       4.666667        0.000000        -2.333333       -0.000000       0.666667-0.000000       -0.033333
9       1.000000        -4.500000       6.000000        0.000000        -4.200000       -0.000000       2.000000-0.000000       -0.300000       -0.000000
10      1.000000        -5.000000       7.500000        0.000000        -7.000000       -0.000000       5.000000-0.000000       -1.500000       -0.000000       0.075758
root@DESKTOP-OCN6246:/mnt/c/Users/Dillon/Desktop#
```