Maths Club: Project DiceForgec

Pseudorandom Distributions # 1

Student's Name, Roll Number

January 23, 2024

Contents

List of Figures

List of Tables

1 Problem 1

The infinite series

$$f(n) = \sum_{n=1}^{n} \frac{1}{i^4} \tag{1}$$

converges on a value of $f(n) = \pi^4/90$ as n approaches infinity.

(a) Write a program in single precision to calculate $f(n)$ for $n = 100$ by computing the sum from $i = 1$ to 100.

(b) Then repeat the calculation but in reverse order—that is, from $i = 100$ to 1 using increments of -1.

(c) In each case, compute the true percent relative error. Explain the results.

(d) Plot the relative error as a function of number of terms used. Explain the results.

1.1 Approach

Mention the name and description of the algorithm/approach used to solve the problem.
Example: In this problem, we use a simple `for` loop to sum the terms in $f(n)$.

1.2 Algorithm

In this section, present the pseudocode/flowchart of the algorithm used to solve the problem.
Example: The pseudocode for the summation is provided in Algorithm 1.

Algorithm 1: Approximating $f(n)$

$n \leftarrow 100, s \leftarrow 0$
$s^* \leftarrow \frac{\pi^4}{90}$
$err[n] = \mathbf{0}$
for $i = 1$ *to* n **do**
  $s \leftarrow s + i^{-4}$;
  $err[i] = \frac{|s - s^*|}{s} \times 100$;
end

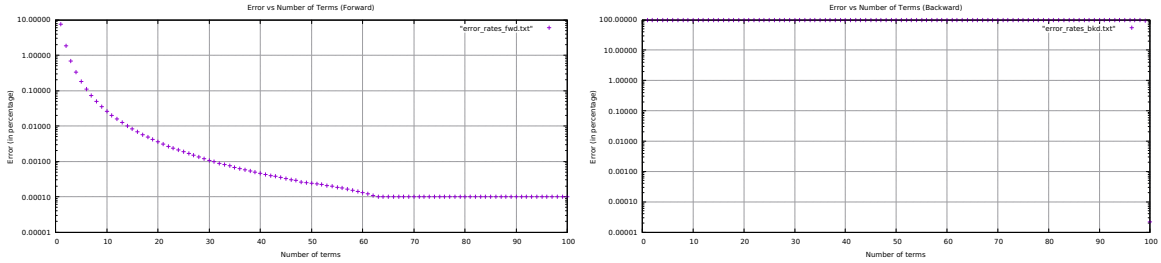
1.3 Results

In this section, plot the graphs/table that provide an overview of your findings.
Example: We plot the graphs showing the error percentage as a function of the number of terms considered while approximating $f(n)$ In Figure 1 and 2. The results are also summarized in Table 1.

1.4 Inferences

Present the key findings/highlights of the experiments in this section in a bulleted manner.
Example: We deduce the following inferences in this experiment:

- The sharp decline in the error percentage, even in the logarithmic scale, as a function of n in Figure 1a shows that the sum converges to the true value *extremely fast.*



(a) Summing terms from $n = 1$ to 100.

(b) Summing terms from $n = 100$ to 1.

Figure 1: Error vs number of terms considered while approximating $f(n)$. This is an example of adding images side-by-side for comparison.

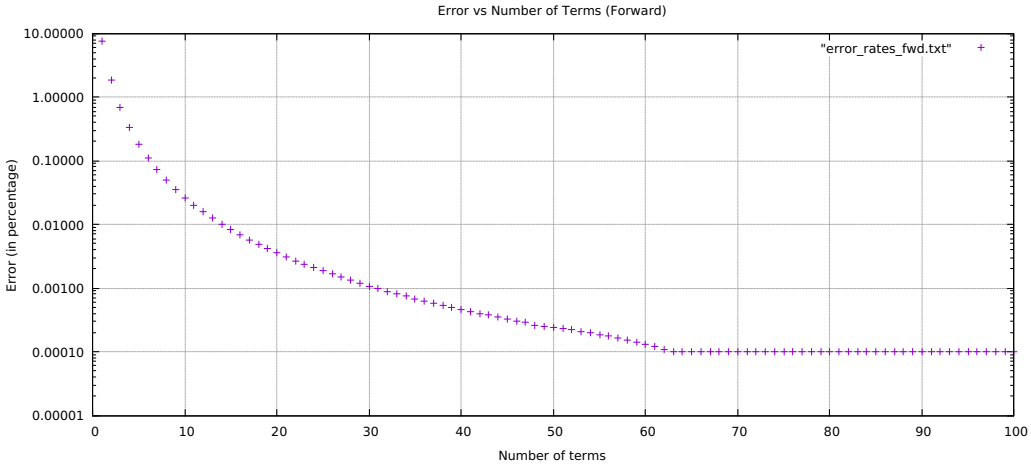


Figure 2: Error vs number of terms considered while approximating $f(n)$. This is an example of inserting a single image.

Table 1: Error percentage vs number of terms

n	**Forward**	**Backward**
1	7.61	100
5	0.18	100
10	0.03	99.99
50	2.4×10^{-4}	99.99
95	9.9×10^{-5}	99.81
99	9.9×10^{-5}	92.39
100	9.9×10^{-5}	2.2×10^{-5}

- Although, there is absolutely no difference in the order which the terms are added mathematically, we see there is a difference in both in Table 1 due to using `float` (single precision value) in C. The results indicate that it is more precise to calculate the sum in a reverse manner confirming that floating point arithmetic is **not commutative**.

1.5 Code

The code used for the experiments is mentioned in Listing 1.

Add comments in the code and name the variables appropriately.

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

#ifndef M_PI
#    define M_PI 3.14159265358979323846
#endif

int main() {
   int i, N = 100;
   float sum, true_sum;
   sum = 0.0;
   FILE *fp = NULL;

   /* the file to which the data is saved.
    It will be used later for plots. */
   fp = fopen("error_rates_fwd.txt","w");
   true_sum = pow(M_PI, 4) / 90.0;
   printf("true sum:%f\n", true_sum);

   // use "for (i = N; i >= 1; i--)" for the backward loop
   for (i = 1; i <= N; i++)
//   for (i = N; i >= 1; i--)
   {
   sum = sum + 1 / pow(i, 4);

   float err;

   // compute the error
   err = ((true_sum - sum) / true_sum) * 100;

   // debug on terminal
   printf("value of i:%d, sum: %f, err:%f\n", i, sum, err);
   fprintf(fp,"%d\t %f\n",i, err);
   }

   return 0;
}
```

Listing 1: Code snippet used in the experiment.

1.6 Alternate Methods (optional)

In this section, mention if you used any alternate (efficient) method to the assignment. Also, mention how this approach is better than the one described in the class (in terms of space/time complexity).