Generating Pi+ Decay Experiments

Your Name

1 Introduction

This paper presents an analysis of the decay times of pi+ particles using a C++ program. The program generates a set of random decay times by simulating N_{exp} experiments based on an exponential distribution. It then exports the decay times to a file for further analysis.

2 C++ Code Explanation

The following C++ code snippet simulates the decay times of pi+ particles:

```
#include <iostream>
#include <fstream>
#include <random>
#include <iomanip>
#include <vector>
#include "decay_analysis.h"

int main(int argc, char *argv[]) {
    int Nexp = 10;
    double lambda = 38412783.7744;
    // ...
}
```

The code starts by including necessary libraries and a header file for a decay analysis function. The 'main' function initializes the number of experiments, N_{exp}, and λ, the inverse of the pi+ particle lifetime.

The following for loop allows the user to provide N_{exp} and λ values as command-line arguments:

```
for (int i = 0; i < argc; i++) {
    if (strncmp(argv[i], "-Nexp", 4) == 0) {
        i++;
        int Ne = std::stoi(argv[i]);
        if (Ne > 0)
            Nexp = Ne;
    }
    if (strncmp(argv[i], "-lambda", 4) == 0) {
        i++;
        double lambda_temp = std::stoi(argv[i]);
        lambda = lambda_temp;
    }
}
```

Next, the code initializes a random number generator with a fixed seed and an exponential distribution with the given λ value:

```
unsigned seed = 4589698390;
std::default_random_engine generator(seed);
std::exponential_distribution<double> distribution(lambda);
```

The program then generates N_{exp} random decay times:

```
double results[Nexp];
for (int i = 0; i < Nexp; ++i) {
    double random_number = distribution(generator);
    results[i] = random_number;
    //std::cout << results[i] << std::endl;
}
```

Following the generation of decay times, the code writes them to a file named "results.txt":

```
std::ofstream outFile("results.txt");
if(outFile.is_open()) {
    for(int i = 0; i < Nexp; i++) {
        outFile << results[i] << "\n";
    }
    outFile.close();
    //std::cout << "Array exported to file successfully.\n";
} else {
    std::cerr << "Unable to open output file.\n";
    return 1;
}
```

3 Decay Analysis Function

The decay_analysis() function reads decay times from a file, calculates the average decay time, and prints the results. Here is the C++ code for the decay_analysis() function:

```
#include <iostream>
#include <fstream>
```

```
#include <vector>

const std::string FILE_NAME = "results.txt";

double decay_analysis() {
    std::ifstream inFile(FILE_NAME);
    std::vector<double> dataVector;

    // ...
}
```

The function starts by including the necessary libraries and defining a constant string, FILE_NAME, to store the name of the input file. It then opens the input file and initializes a vector to store the decay times.

Next, the function reads the decay times from the input file into the vector:

```
if (inFile.is_open()) {
    double value;
    while (inFile >> value) {
        dataVector.push_back(value);
    }
    inFile.close();
} else {
    throw std::runtime_error("Unable to open results file!");
}
```

If the file is open, the function reads each decay time from the file and stores it in the vector. After reading all the decay times, the function closes the file. If the file cannot be opened, the function throws a runtime error.

After reading the decay times, the function calculates the average decay time:

```
double total = 0;
for (std::size_t i = 0; i < dataVector.size(); ++i) {
    total = total + dataVector[i];
}
double average_lifetime = total / dataVector.size();
```

The function initializes a variable, ‘total‘, to store the sum of all decay times. It then iterates through the vector, adding each decay time to the total. After the loop, the function calculates the average decay time by dividing the total by the number of decay times in the vector.

Finally, the function returns the average decay time:

```
//std::cout << average_lifetime << std::endl;
return average_lifetime;
```

The average value should be approximately 26.033 nanoseconds for pi+, and this is expected because thats how I chose the value of lambda. For exponential decays, $\lambda = \frac{1}{mean}$.