C++ Project 2 Report

First Function (Non-Bonus):

* This function (computePi) takes number of intervals as a parameter. The user will be prompted from the main function to input the number of intervals.
* The function computes pi using the trapezoidal rule and returns it. This function also computes and returns the error of the computed pi using the actual pi as a reference. So, this function returns a size 2 vector containing the computed pi and error which will be printed out using the print\_vector function.
* The first element is the computed pi and the second element is the error. (in the vector)
* For printing out the computed pi, I set a fixed precision of 12 in the print\_vector function since the task did not specify a decimal place to round to.
* As the number of intervals increases, the more precise the computed pi becomes. Consequently, this also results in a smaller error.
* The larger the number of intervals, the longer it takes for the program to run, but the computed pi becomes more precise and the error decreases.

Second Function (Bonus):

* This function (computePiBonus) takes precision as a parameter. The user will be prompted from the main function to input the precision.
* The function computes pi using the trapezoidal rule and returns it. This function computes and returns the error of the computed pi using the actual pi as a reference. Lastly, it also returns the estimated number of intervals to have the computed pi to the precision given by the user.
* For printing all three values, the function returns a size 3 vector which is printed out using the print\_vector function. The first element is the computed pi, the second element is the error, and the third element is the estimated number of intervals. (in the vector)
* As the requested precision increases, the number of intervals increases substantially. The error decreases substantially. The program will take longer to run if the requested precision is high.
* One major flaw in this function is that the estimated number of intervals is overestimated. For example, the actual number of intervals to receive pi to .0001 could be 1000, but this function estimates it to 44000 intervals.
* This occurs because I am incrementing the number of intervals by multiplying by 1.2. Incrementing the intervals by 1 would make the program run too slowly. Even by multiplying by 1.2, asking for a high precision like 10^-8 would take too long for the program to finish running.

A screenshot of a computer

Description automatically generatedOutput: