EJERCICIO 7 : Multiplicación matricial

Autor : Jesús Ruiz Castellano, 76439001-L

1.- Código fuente : multiplicacion.cpp

```cpp
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <climits>
#include <cassert>

using namespace std;

/**
 * @brief Multiplica una matriz
 * @details [long description]
 *
 * @param vr: vector de resultado. Debe tener n elementos. Es modificado
 * @param v1: vector de elementos. Debe tener n elementos. No es modificado
 * @param v2: vector de elementos. Debe tener n elementos. No es modificado
 * @param n: numero de elementos. num_elem > 0
 */
inline void multiplicacion(int **vr, int **v1, int **v2, int n) {

        for (int i = 0; i<n; i++)
                for (int j=0 ; j<n; j++)
                        for (int k = 0; k < n; k++)
                                vr[i][j]+=v1[i][k]*v2[k][j];


}
```

```cpp
int main(int argc, char * argv[])
{

        if (argc != 2)
        {
                cerr << "Formato " << argv[0] << " <num_elem>" << endl;
                return -1;
        }

        int n = atoi(argv[1]);

        int ** T1 = new int * [n]; // Reservamos memoria para las 3 matrices
        int ** T2 = new int * [n];
        int ** TR = new int * [n];

        for (int i=0; i<n; i++){
                T1[i]=new int [n];
                T2[i]=new int [n];
                TR[i]=new int [n];
        }

        srand(time(0));
        // Se rellenan dos matrices con números aleatorios, y una tercera con todo ceros.
        for (int i=0; i<n; i++)
                for (int j=0; j<n; j++){
                        T1[i][j] = rand();
                        T2[i][j] = rand();
                        TR[i][j] = 0;           // Se rellena con 0
        }


        clock_t tini = clock();  // Iniciamos el reloj

        for ( int i=0 ; i<1000 ; i++ )
                multiplicacion(TR, T1, T2, n);

        clock_t tfin = clock();  // Detenemos el reloj

        cout << n << "  " << ( ((double)(tfin - tini)) / CLOCKS_PER_SEC )/ 1000.0 << endl;

        delete [] T1;
        delete [] T2;
        delete [] TR;

        return 0;
};
```

EJERCICIO 7 : Multiplicación matricial

2.- Hardware usado:

2.1- CPU

vendor_id : GenuineIntel
model name : Intel(R) Core(TM) i3 CPU M 330 @ 2.13GHz
cpu MHz : 933.000

2.2- Velocidad de Reloj

Versión : hwclock de util-linux 2.20.1
jue 13 oct 2016 10:45:11 CEST -0.563198 segundos

2.3- Memoria RAM

MemTotal : 3907668 kB
SwapTotal : 4049916 kB

3.- Sistema Operativo

Ubuntu 14.04.3 LTS
Arquitectura : x86_64 (64 bits)

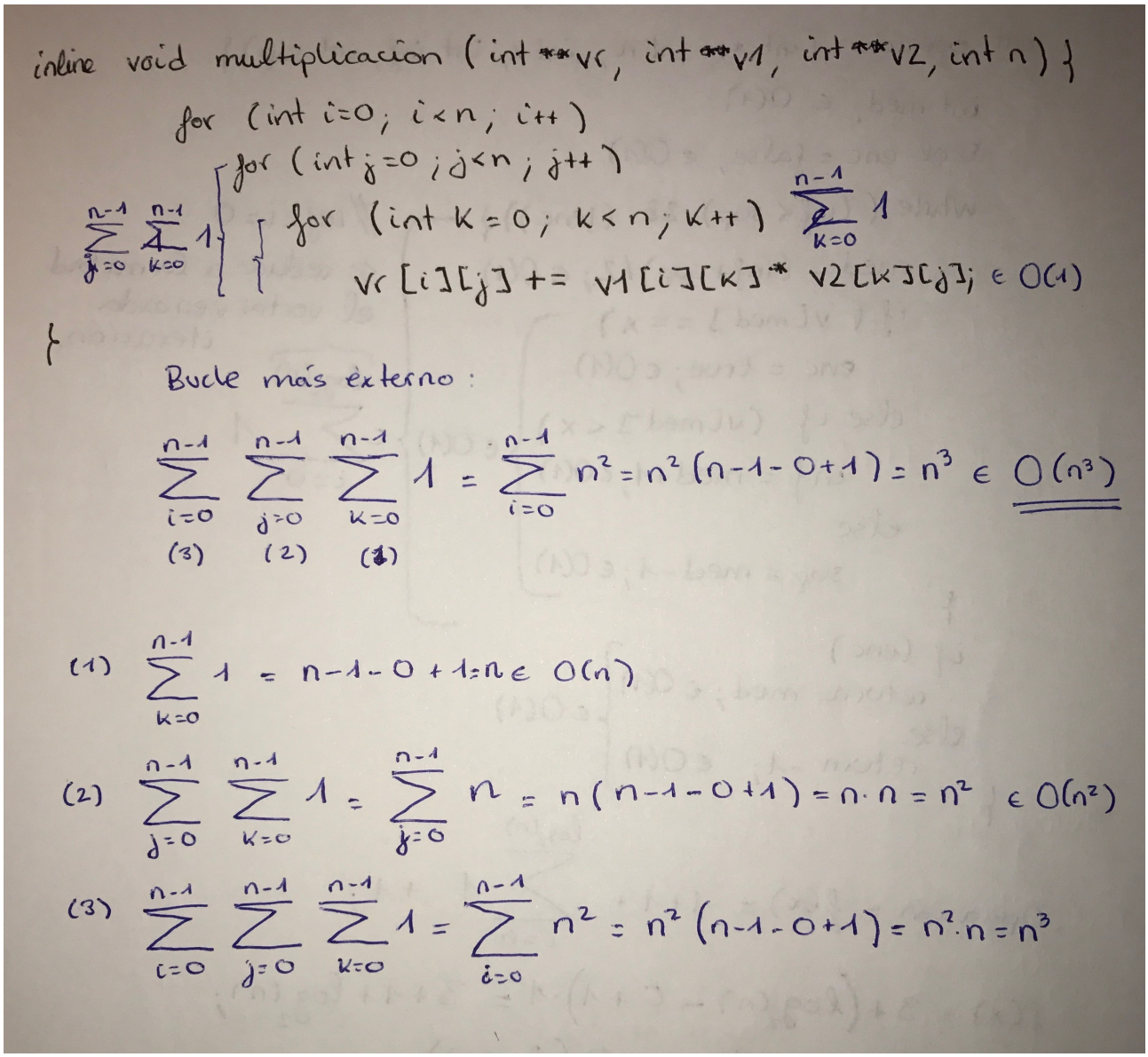
4.- Compilador usado y opciones de compilación

gcc - GNU project C and C++ compiler
Opción de compilación : g++ -o <nombre_ejecutable> <ejecutable.cpp>
g++ -o multiplicacion multiplicacion.cpp

EJERCICIO 7 : Multiplicación matricial

5.- Desarrollo completo del cálculo de la Eficiencia teórica

EFICIENCIA TEÓRICA

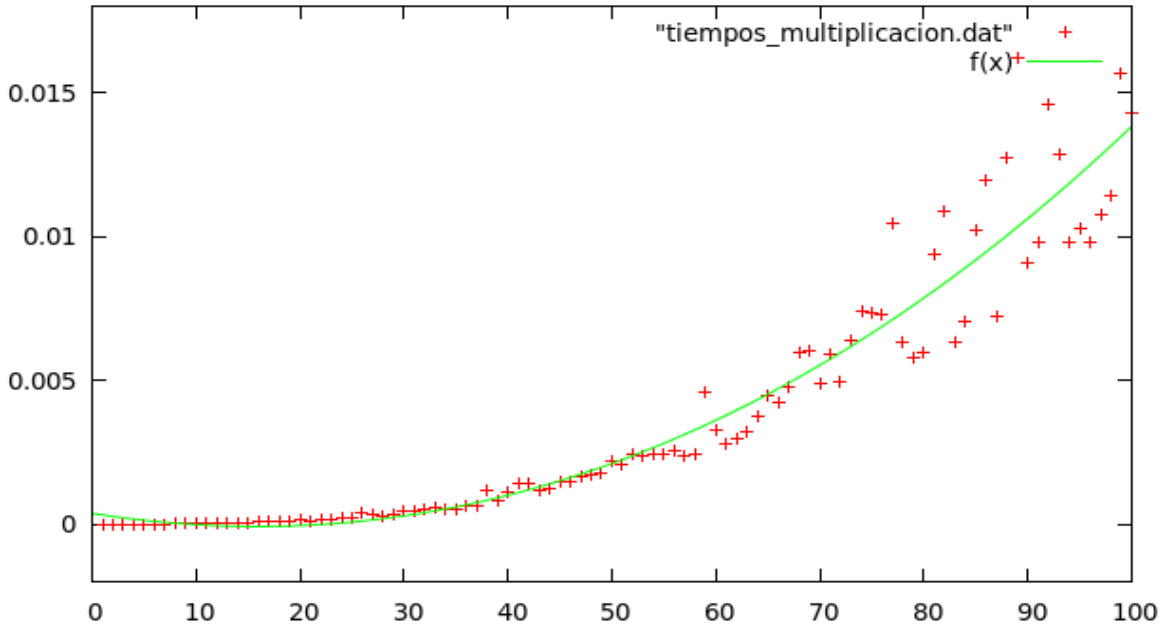


EJERCICIO 7 : Multiplicación matricial

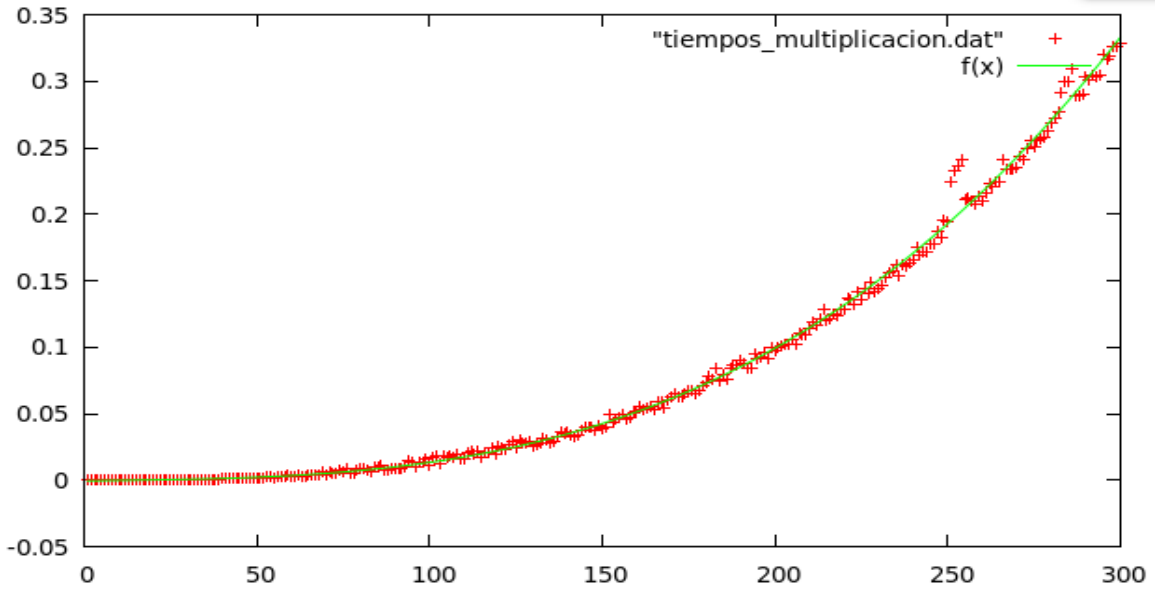
6.- Parámetros usados para el cálculo de la eficiencia empírica y gráfica

Para ésta parte he ejecutado el programa con los siguientes valores para tamaño máximo de las matrices A y B : 100, 300, 600, 1000, 1500, 3000, 6000 y 10000.

6.1- Gráfica de tiempos para la ejecución del programa con num. de elementos = 100

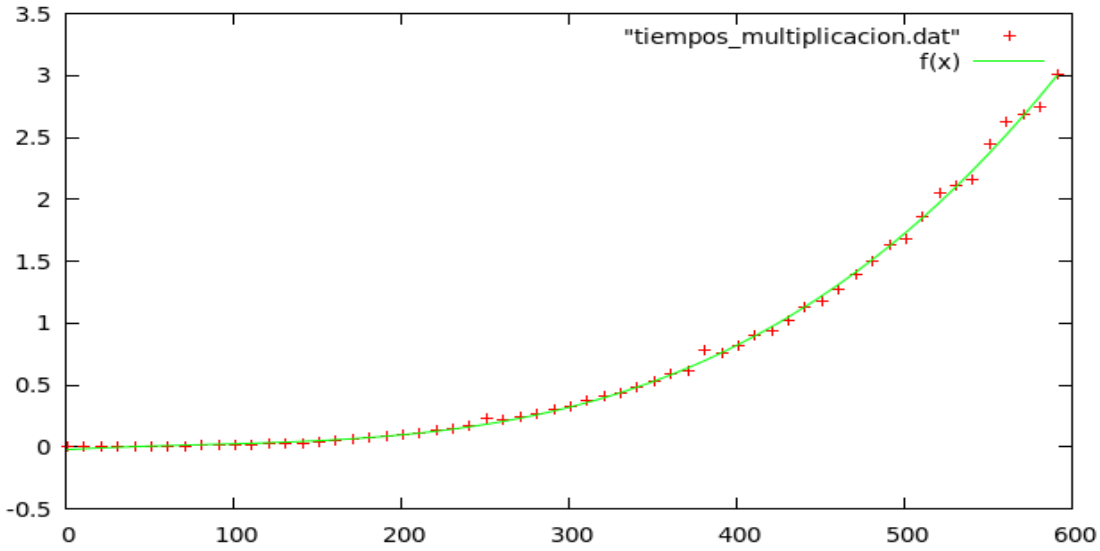


6.2- Gráfica de tiempos para la ejecución del programa con num. de elementos = 300

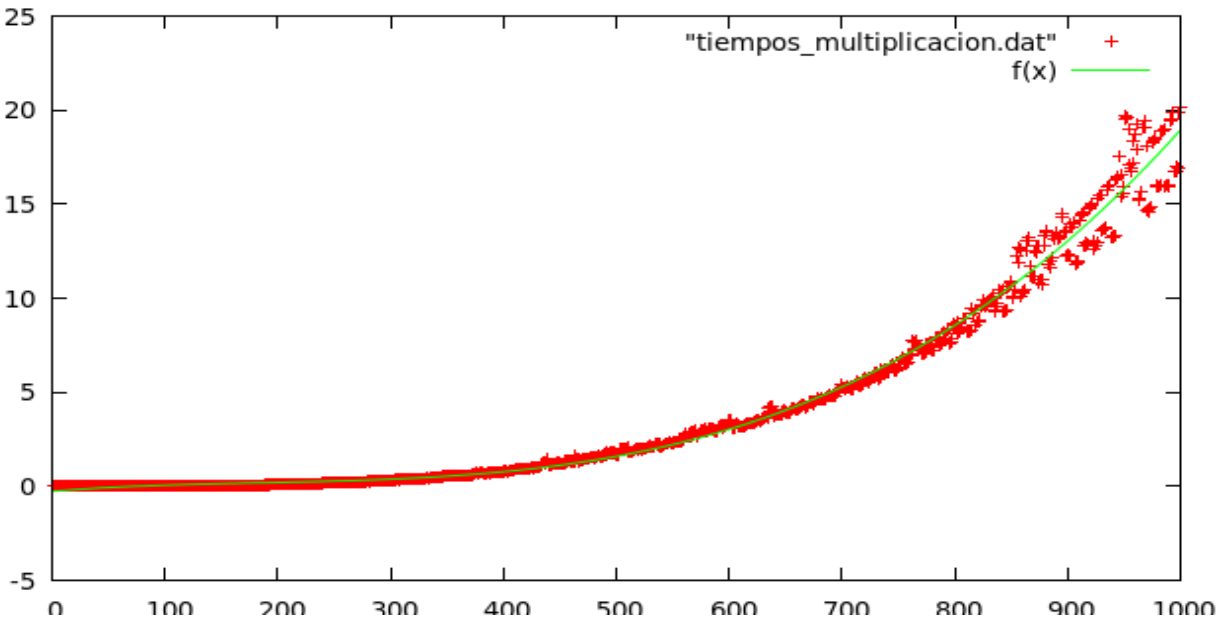


EJERCICIO 7 : Multiplicación matricial

6.3- Gráfica de tiempos para la ejecución del programa con num. de elementos = 600



6.4- Gráfica de tiempos para la ejecución del programa con num. de elementos = 1000



***Con numero de elementos = 1000, en adelante, mi ordenador tarda demasiado en ejecutarlo. Por eso he probado hasta ahí.**

***Como se ve en el caso en el que hay 1000 elementos, la gráfica se ajusta más a la forma que tiene que tener, dada por su valor de eficiencia teórica ($O(n^3)$)**