25-11-2020

Métodos Numericos para la Computación

Entrega 3

4ºCurso

Grupo Prácticas 11

Alejandro Daniel Herrera Cardenes
Carlos Eduardo Pacichana Bastidas
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Indice

Practica 2

Descripción

Realizar los siguientes ejercicios usando la librería CBLAS nivel 1.
Emplear Matlab/Octave para verificar que el resultado es correcto.

1. Definir dos vectores 3D ortogonales y comprobar que su producto escalar es nulo.
2. Construir dos vectores conteniendo el valor ASCII los 10 primeros caracteres de tu nombre (vN) y de tu apellido (vA), completando con 0 si es necesario.
   Obtener el resultado de sumar al primer vector el triple del segundo y mostrar el resultado mapeado a caracteres 'a…z'.
3. Crear un vector conteniendo todos los dígitos de tu fecha de nacimiento. La nota final de la asignatura de MNC será el resultado de calcular el módulo 11 de la norma2 de ese vector.

Trabajo realizado

En la primera actividad creamos dos vectores ortogonales (v1 y v2) y comprobamos que su producto escalar es nulo mostrándolo por pantalla. Para ello usamos el método ***cblas_ddot*** que pide el número de elementos a multiplicar, el primer array, el incremento del mismo, el segundo array y su incremento.

En la actividad 2, creamos 2 vectores de 10 doubles, uno con el nombre y otro con el apellido de Carlos Pacichana y usamos el método ***cblas_daxpy*** que sigue una estructura similar al ddot, salvo que también tiene un parámetro para multiplicar al primer array), para normalizar dividimos todos los elementos del vector vN entre el máximo, lo multiplicamos por 25 para que esté entre 1 y 25 que son el número de letras del abecedario, y le sumamos 97 para que esté entre los valores ascii correspondientes a las letras minúsculas.

En la actividad 3 no hicimos más que crear un vector con la fecha, y calcular el módulo 11 de la norma 2, calculada con la función cblas_snrm2.

Implementamos las operaciones equivalentes en MatLab para comprobar que los reclutados de estas en CBlas son correctos y que mostramos en los resultados de ejecución.

Código Programa:

Los códigos de la librería ***CBlas*** de los distintos apartados serán puesto a continuación con la referencia del apartado en el pie de la foto.

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

//  PRACTICA 2.1
    double v1[3] = { 2.0, 3.0, 1.0 };
    double v2[3] = { -6.0, 4.0, 0 };
    double res1;
    const MKL_INT  n1=3, incx=1, incy=1;

    res1 = cblas_ddot(n1, v1, incx, v2, incy);

    printf("            PRACTICA 2.1\n");
    printf("-------------------------------------------------------------\n");
    printf("Resultado del producto escalar: \n%1f\n\n", res1);
```

Código principal 1

```
//  PRACTICA 2.2
    double vN[10] = {99.0,97.0,114.0,108.0,111.0,115.0,0.0,0.0,0.0,0.0};
    double vA[10] = {112.0,97.0,99.0,105.0,99.0,104.0,97.0,110.0,97.0,0.0};
    const MKL_INT n2 = 10, incn = 1, inca = 1;
    const double a = 3.0;

    cblas_daxpy(n2, a, vA, inca, vN, incn);
    double max = vN[cblas_idamax(n2, vN, incn)];

    printf("            PRACTICA 2.2\n");
    printf("-------------------------------------------------------------\n");
    printf("Resultado mapeado de caracteres:\n");
    for (int i = 0; i < n2; i++) {
        printf("| %c ", (int)((vN[i]/max)*25+97));
    }
```

Código principal 2

```
//  PRACTICA 2.3
    const int n3 = 8, incf=1;
    double res3;
    double fN[] = {1.0, 8.0, 1.0, 2.0, 1.0, 9.0, 9.0, 4.0};

    res3 = fmod(cblas_dnrm2(n3, fN, incf), 11);

    printf("\n\n            PRACTICA 2.3\n");
    printf("-------------------------------------------------------------\n");
    printf("Modulo 11 de la norma 2: \n%1f", res3);

    std::getchar();
    return 0;
}
```

Código principal 3

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
                        PRACTICA 2.1
------------------------------------------------------------
Resultado del producto escalar:
0.000000

                        PRACTICA 2.2
------------------------------------------------------------
Resultado mapeado de caracteres:
| z | w | x | y | x | y | q | s | q | a

                        PRACTICA 2.3
------------------------------------------------------------
Modulo 11 de la norma 2:
4.779734
```

Resultado de la ejecución

Matlab

A continuación los códigos de los distintos apartados pero esta vez en ***Matlab***

```
v1 = [2 3 1];
v2 = [-6 4 0];

escalar = v1*v2';
```

Código principal 1

```
vN = [99 97 114 108 111 115 0 0 0 0];
vA = [112 97 99 105 99 104 97 110 97 0];

vR = vN+3.*vA;
maximo = max(vR);
for i=1:10
    vN(i) = fix((vR(i)/maximo)*25+97);
    vMap(i) = char(vN(i));
end
```

Código principal 2

```
vF = [1 8 1 2 1 9 9 4];
mod11Norm2 = mod(norm(vF), 11);
```

Código principal 3

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
Producto Escalar: 0
Remapear Caracteres: zwxyxyqsqa
Modulo 11 de la Norma 2: 4.779734
```

Resultado de la ejecución

Practica 3

Descripción

Realizar los siguientes ejercicios usando la librería CBLAS nivel 2.
Emplear Octave para verificar que el resultado es correcto.

1. Definir una matriz (A) y dos vectores (x, y) y realizar las siguientes operaciones:
   a) A*x
   b) 3*A*x+4*y
2. Probar el efecto de los parámetros de layout y trasposición en el ejercicio anterior.

Trabajo realizado

En este ejercicio se nos pedía definir una matriz y dos vectores y realizar dos productos usando el método ***cblas_dgemv***. Este método tiene como parámetros dos constantes, la primera indica cómo está almacenada en memoria la matriz (ya que las matrices se almacenan como vectores), si primero fila o primero columna, y la segunda si la matriz está traspuesta o no, a continuación se añaden las dimensiones de la matriz (filas y columnas), el valor por el que queremos multiplicarla, la matriz en sí y finalmente el valor LDA, que significa la primera dimensión de la matriz, hay que pasarle como mínimo el valor de la dimensión de las filas, más adelante explicaremos qué sucede cuando se toca ese valor. Ahora vamos a los parámetros del primer array que admite este método, siendo simplemente el array mismo y una variable incx (incremento en X), que significa el salto que realiza la rutina para avanzar en el array, en nuestro caso lo pondremos a 1.

Finalmente entramos en los parámetros del segundo array, cuyos parámetros son beta (un valor por el que multiplicar el array, si no se quiere usar este array se pondrá un 0), el array y (que se sobrescribe para devolver resultado), y la variable incy, al igual que en el array anterior.

Para probar los valores layout y trasposición simplemente fuimos comprobando que valores resultaban y llegamos a la siguiente conclusión: Tal y como fue comentado anteriormente el valor layout indica si la matriz está almacenada por filas o columnas en memoria, y el de trasposición es para si se quiere operar con la matriz traspuesta o normal, por tanto aplicados de manera individual generan el mismo resultado.

Código Programa:

Los códigos de la librería ***CBlas*** de los distintos apartados serán puesto a continuación con la referencia del apartado en el pie de la foto.

```
//  PRACTICA 3.1

const double x[3] = {1.0, 2.0, 3.0};
double yA[3] = { 1.0, 2.0, 3.0};

const double A[9] = {1.0, 2.0, 3.0,
                     4.0, 5.0, 6.0,
                     7.0, 8.0, 9.0};

const MKL_INT m = 3, n = 3;
const double alphaA = 1.0;
const MKL_INT lda = 3;//preguntar
const MKL_INT incx = 1;
const double betaA = 0.0;
const MKL_INT incy = 1;

//  PRACTICA 3.1 Apartado A

cblas_dgemv(CblasRowMajor, CblasNoTrans, m, n, alphaA, A
    , lda, x, incx, betaA, yA, incy);

printf("        PRACTICA 3.1 Apartado (A) \n");
printf("-------------------------------------------------------\n");
printf("Resultado de A*x: \n");
for (int i = 0; i < 3; i++) {
    printf("| %.*f ",1, yA[i]);
}

//  PRACTICA 3.1 Apartado B

const double alphaB = 3.0;
const double betaB = 4.0;

double yB[3] = { 1.0, 2.0, 3.0 };

cblas_dgemv(CblasRowMajor, CblasNoTrans, m, n, alphaB, A
    , lda, x, incx, betaB, yB, incy);

printf("\n\n        PRACTICA 3.1 Apartado (B) \n");
printf("-------------------------------------------------------\n");
printf("Resultado de 3*A*x+4*y: \n");
for (int i = 0; i < 3; i++) {
    printf("| %.*f ", 1, yB[i]);
}
```

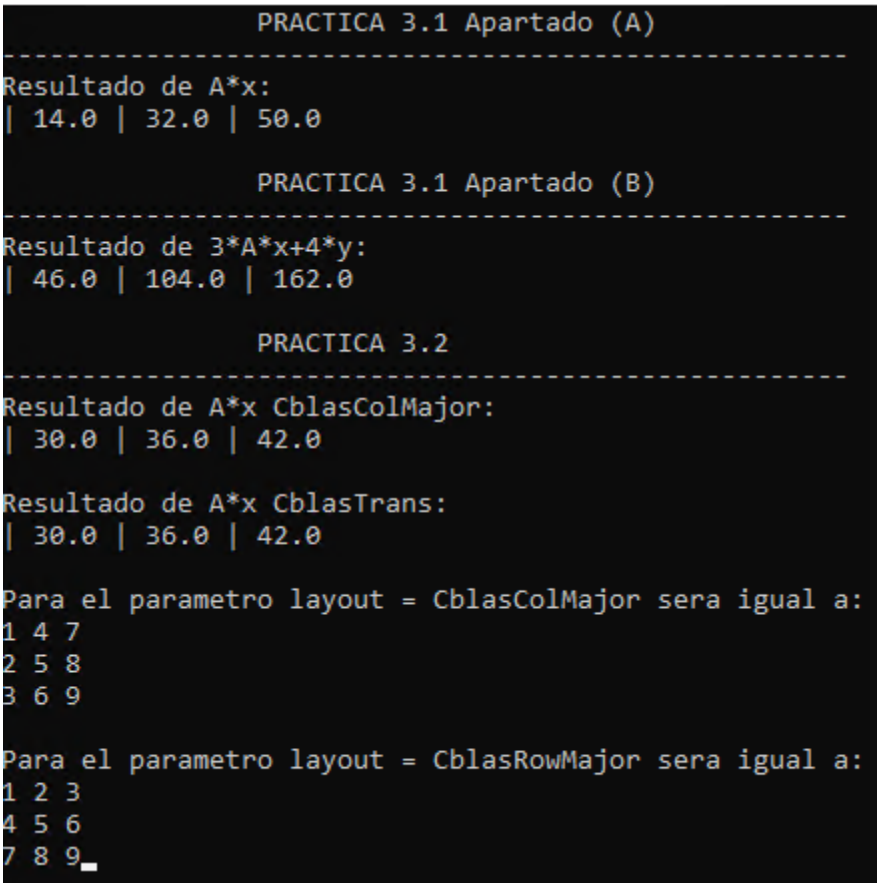
Código principal 1

```
    //  PRACTICA 3.2

    double yA1[3] = { 1.0, 2.0, 3.0 };

    cblas_dgemv(CblasColMajor, CblasNoTrans, m, n, alphaA, A
        , lda, x, incx, betaA, yA1, incy);

    printf("\n\n        PRACTICA 3.2\n");
    printf("-----------------------------------------------------\n");
    printf("Resultado de A*x CblasColMajor: \n");
    for (int i = 0; i < 3; i++) {
        printf("| %.*f ", 1, yA1[i]);
    }

    double yA2[3] = { 1.0, 2.0, 3.0 };

    cblas_dgemv(CblasRowMajor, CblasTrans, m, n, alphaA, A
        , lda, x, incx, betaA, yA2, incy);

    printf("\n\nResultado de A*x CblasTrans: \n");
    for (int i = 0; i < 3; i++) {
        printf("| %.*f ", 1, yA2[i]);
    }



    printf("\n\nPara el parametro layout = CblasColMajor sera igual a:");
    printf("\n1 4 7");
    printf("\n2 5 8");
    printf("\n3 6 9");
    printf("\n\nPara el parametro layout = CblasRowMajor sera igual a:");
    printf("\n1 2 3");
    printf("\n4 5 6");
    printf("\n7 8 9");


    std::getchar();
    return 0;
}
```

Código principal 2

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).



Resultado de la ejecución

Matlab

A continuación los códigos de los distintos apartados pero esta vez en ***Matlab***

```
x = [1 2 3];
y = [1 2 3];

A = [1 2 3; 4 5 6;7 8 9];

prac3A = A*x';
prac3B = 3.*A*x.'+4.*y.';
```

Código principal 1

```
prac3Tras = A'*x';
```

Código principal 2

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
Apartado A (A*x):
| 14 | 32 | 50

Apartado B (3*A*x+4*y):
| 46 | 104 | 162

Apartado A con Matriz Traspuesta (3*AT*x+4*y):
| 30 | 36 | 42
```

Resultado de la ejecución

Practica 4

Descripción

Realizar los siguientes ejercicios usando la librería CBLAS nivel 3.
Emplear Octave para verificar que el resultado es correcto.
Definir tres matrices (A, B y C) de dimensión 3x3 y realizar las operaciones:

a) A*B
b) A*BT
c) 2*A*B + 3*C

Trabajo realizado

Para la realización de este ejercicio, hemos creado las matrices necesarias 3x3 y utilizando la función cblas_dgemm, que es muy parecida a la usada en la actividad anterior, salvo que al tener 3 matrices, 2 que se usan para la multiplicación y suma, y una adicional para guardar el resultado realizamos las operaciones necesarias.

Código Programa:

Los códigos de la librería ***CBlas*** de los distintos apartados serán puesto a continuación con la referencia del apartado en el pie de la foto.

```
//A*B

cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, m, n, k, alphaA, A, lda, B, ldb, betaA, C, ldc);

printf("PRACTICA 4.1 Apartado (A) \n");
printf("Resultado de A*x: \n");
for (int i = 0; i < 9; i++) {
    printf("| %.*f ", 1, C[i]);
}
```

Código principal 1

```
//A* BT

cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasTrans, m, n, k, alphaA, A, lda, B, ldb, betaA, C, ldc);

printf("\n\nPRACTICA 4.1 Apartado (B) \n");
printf("Resultado de A*x: \n");
for (int i = 0; i < 9; i++) {
    printf("| %.*f ", 1, C[i]);
}
```

Código principal 2

```
//2 * A * B + 3 * C

double Cc[9] = { 10.0, 11.0, 12.0,
                 13.0, 14.0, 15.0,
                 16.0, 17.0, 18.0 };

const double alphaC = 2.0;
const double betaC = 3.0;

cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, m, n, k, alphaC, A, lda, B, ldb, betaC, Cc, ldc);

printf("\n\nPRACTICA 4.1 Apartado (C) \n");
printf("Resultado de A*x: \n");
for (int i = 0; i < 9; i++) {
    printf("| %.*f ", 1, Cc[i]);
}

std::getchar();
return 0;
```

Código principal 3

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
PRACTICA 4.1 Apartado (A)
Resultado de A*x:
| 30.0 | 24.0 | 18.0 | 84.0 | 69.0 | 54.0 | 138.0 | 114.0 | 90.0

PRACTICA 4.1 Apartado (B)
Resultado de A*x:
| 46.0 | 28.0 | 10.0 | 118.0 | 73.0 | 28.0 | 190.0 | 118.0 | 46.0

PRACTICA 4.1 Apartado (C)
Resultado de A*x:
| 90.0 | 81.0 | 72.0 | 207.0 | 180.0 | 153.0 | 324.0 | 279.0 | 234.0
```

Resultado de la ejecución

Matlab

A continuación los códigos de los distintos apartados pero esta vez en ***Matlab***

```
a = [1 2 3; 4 5 6; 7 8 9]
b = [9 8 7; 6 5 4; 3 2 1]
c = [10 11 12; 13 14 15; 16, 17, 18]


F = mtimes(a,b)
```

Código principal 1

```
D = mtimes(a,b')
```

Código principal 2

```
X = 2 * mtimes(a,b) + 3 * c
```

Código principal 3

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
>> Practica4

F =

    30    24    18
    84    69    54
   138   114    90


D =

    46    28    10
   118    73    28
   190   118    46


X =

    90    81    72
   207   180   153
   324   279   234
```

Resultado de la ejecución

Práctica 5

Descripción

1. Crear tres matrices (A, B y C) de dimensión NxN y rellenarlas con valores aleatorios tipo double.
2. Calcular el número de GFLOPS para distintos valores de N, realizando el promedio de 100 ejecuciones de la operación A*B.
3. Repetir las pruebas utilizando el modo Parallel y comparar los resultados.

Trabajo realizado

Para la realización de este ejercicio práctico reservamos en memoria espacio para 3 vectores de NxN elementos y los rellenamos con valores aleatorios. Con la ayuda de los campos de tamaño de las filas y columnas y los lda de las distintas matrices de la función cblas_dgemm, con un doble bucle fuimos calculando el tiempo promedio de la operación de multiplicar 2 matrices de MxM elementos en 100 ejecuciones. Para obtener los GFLOPS solo hay que dividir el número de operaciones hechas en la iteración entre el tiempo transcurrido y pasarlo a la unidad deseada.

Código Programa:

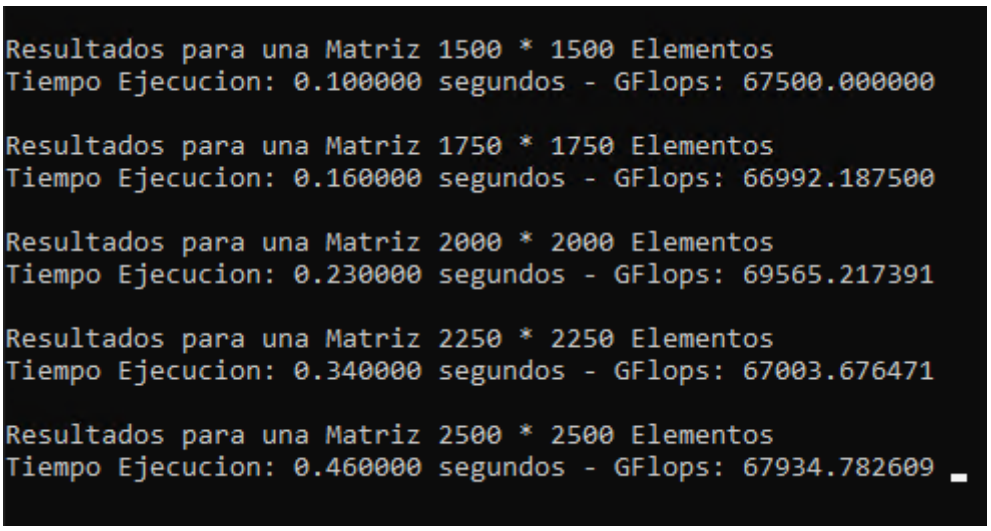
Los códigos de la librería ***CBlas*** de los distintos apartados serán puesto a continuación con la referencia del apartado en el pie de la foto.

```c
int multMatrizDouble(int lenght) {

    const int N = lenght;
    const double alphaA = 1.0;
    const double betaA = 0.0;

    const MKL_INT m = N, n = N, k = N;
    const MKL_INT lda = N, ldb = N, ldc = N;

    double* A = (double*)mkl_malloc(N * N * sizeof(double), 64);
    double* B = (double*)mkl_malloc(N * N * sizeof(double), 64);
    double* C = (double*)mkl_malloc(N * N * sizeof(double), 64);

    srand((unsigned int)time(NULL));

    for (int i = 0; i < N * N; i++) {
        A[i] = (double)rand() / (double)RAND_MAX;
        B[i] = (double)rand() / (double)RAND_MAX;
        C[i] = (double)rand() / (double)RAND_MAX;
    }

    double seconds,GFlops;
    time_t start, finish;

    time(&start);
    for (int i = 0; i < 100; i++) {
        cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, m, n, k, alphaA, A, lda, B, ldb, betaA, C, ldc);
    }
    time(&finish);

    seconds = difftime(finish,start)/100;
    GFlops = (pow(N,3) / seconds)*pow(10,-6);

    printf("\nResultados para una Matriz %i * %i Elementos\n",N,N);
    printf("Tiempo de ejecución: %1f segundos", seconds);
    printf("GFlops: %1f ", GFlops);

    mkl_free(A);
    mkl_free(B);
    mkl_free(C);

    return 0;
}
```

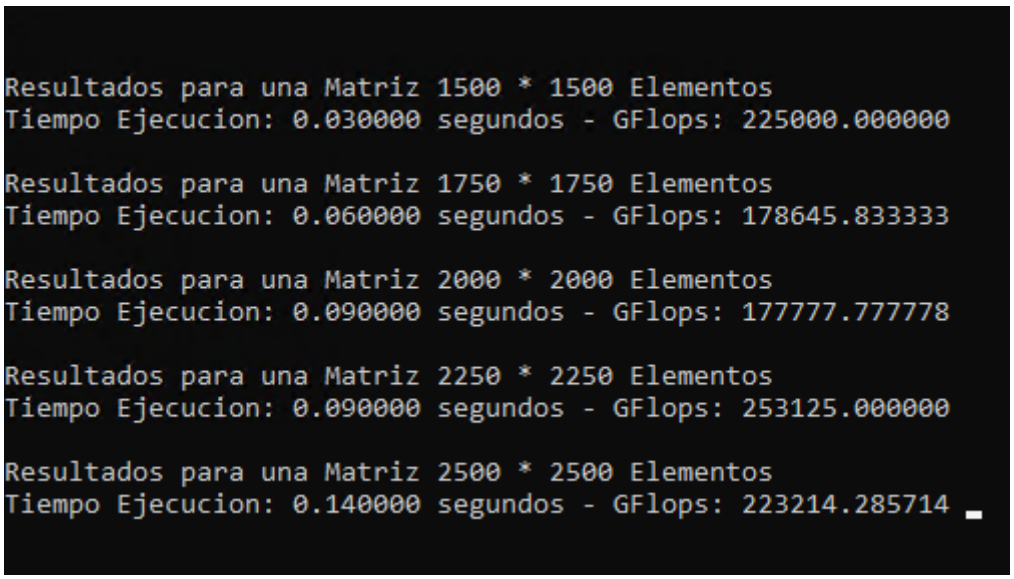
Código principal 1

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).



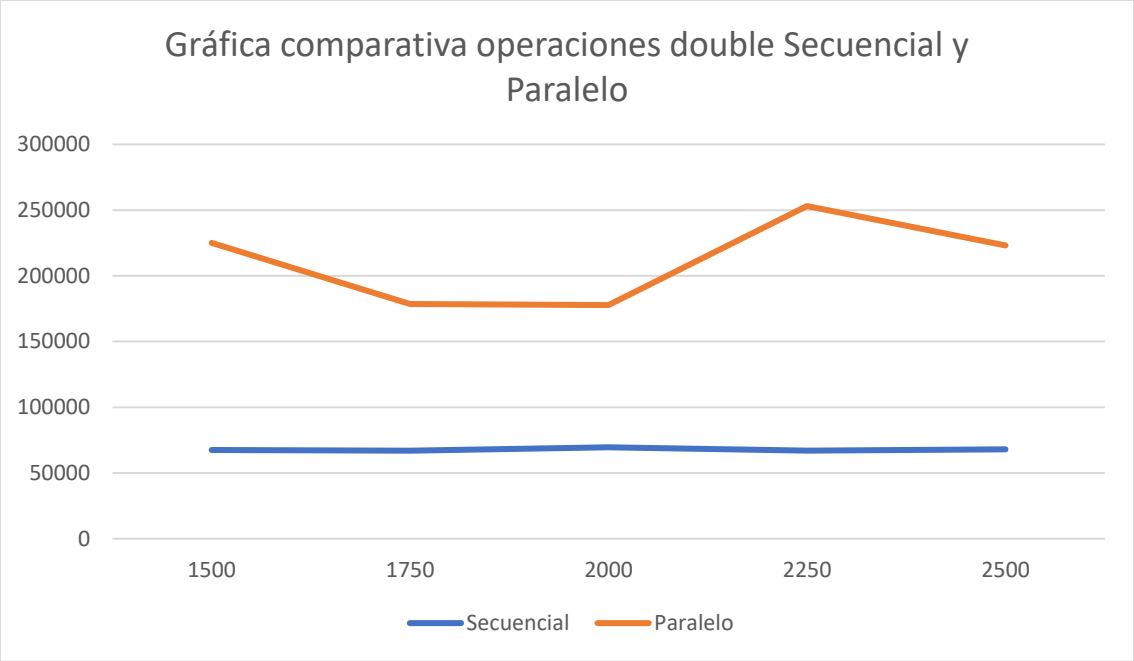
Resultado de la ejecución secuencial



Resultado de la ejecución paralela

Tamaño	Secuencial	Paralelo
1500	67500	225000
1750	66992	178645
2000	69565	177777
2250	67003	253125
2500	67934	223214

Tabla comparativa double Secuencial y Paralelo medido en GFlops



Grafica comparativa double Secuencial y Paralelo medido en GFlops

Nos fijamos que en el modo secuencial no hay casi cambio a medida que cambiamos el tamaño de la matriz sin embargo de manera paralela a mayor la matriz mayor los GFlops porque mas cantidad de operaciones realiza a la vez.

Matlab

A continuación los códigos de los distintos apartados pero esta vez en ***Matlab***

```
n = 2500;
a = rand(n);
b = rand(n);


time = 0;


for i = 1:100
    time = time + metodo3(a,b);
end

time = time / 100;
result = n ^ 3 * 2 / time / 1000000



function time = metodo3(a,b)
  tic;
    F = a*b;
 time = toc;
end
```

Código principal 1

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
>> Practica5

result =

   7.6522e+04
```

Resultado de la ejecución

Optativos

Practica 2

Descripción

[OPTATIVO] Estudiar el efecto de los parámetros de incremento en el ejercicio anterior.

Trabajo realizado

El incremento es el valor de salto entre posiciones del array, por eso cuando ponemos incremento 2, y cogemos el array saltando de 2 en 2 nos da el mismo resultado.

Código Programa:

Los códigos de la librería ***CBlas*** de los distintos apartados serán puesto a continuación con la referencia del apartado en el pie de la foto.

```
double v3[3] = { 1.0, 2.0, 3.0 };
double v4[3] = { 3.0, 2.0, 1.0 };
double v5[6] = { 3.0, 0, 2.0, 0, 1.0 ,0};
double res4, res5;

const MKL_INT  incyEx = 2;

res4 = cblas_ddot(n1, v3, incx, v4, incy);
res5 = cblas_ddot(n1, v3, incx, v5, incyEx);

printf("\n\n            EXTRA\n");
printf("-------------------------------------------------------------\n");
printf("Producto escalar sin modificar incrementos: %1f", res4);
printf("\nProducto escalar con incremento de y=2: %1f", res5);

std::getchar();
return 0;
```

Código principal 1

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
                        EXTRA
-------------------------------------------------------------
Producto escalar sin modificar incrementos: 10.000000
Producto escalar con incremento de y=2: 10.000000
```

Resultado de la ejecución

Practica 4

Descripción

[OPTATIVO] Realizar una operación que implique matrices no cuadradas, y que genere como resultado una matriz de 5x5.

Trabajo realizado

Para cumplimentar la actividad optativa creamos dos matrices de 10 y 25 elementos que representaran a dos matrices 2x5 y 5x2 que acabara en una matriz 5x5 al usar la función cblas_dgemm.

CBlas

Código Programa:

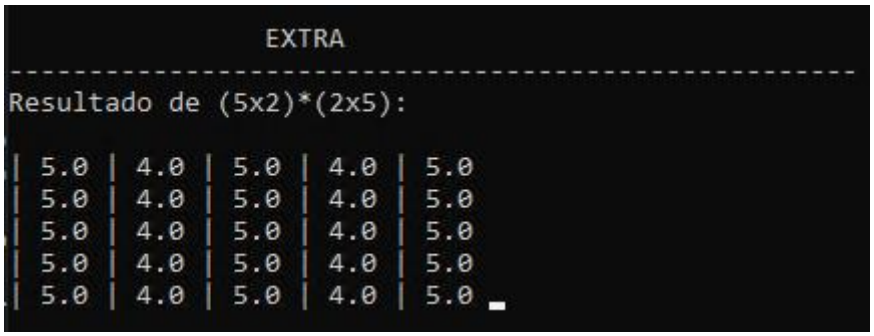
Los códigos de la librería ***CBlas*** de los distintos apartados serán puesto a continuación con la referencia del apartado en el pie de la foto.

```
double ME[10] = { 1.0, 2.0,
                  1.0, 2.0,
                  1.0, 2.0,
                  1.0, 2.0,
                  1.0, 2.0, };


double MR[25] = { 1.0, 1.0, 1.0, 1.0, 1.0,
                  1.0, 1.0, 1.0, 1.0, 1.0,
                  1.0, 1.0, 1.0, 1.0, 1.0,
                  1.0, 1.0, 1.0, 1.0, 1.0,
                  1.0, 1.0, 1.0, 1.0, 1.0,};

const MKL_INT mE = 5, nE = 5, kE = 2;
const MKL_INT ldaE = 2, ldbE = 5, ldcE = 5;

cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, mE, nE, kE, alphaA, ME, ldaE, ME, ldbE, betaA, MR, ldcE);

printf("\n\n        EXTRA \n");
printf("-------------------------------------------------------\n");
printf("Resultado de (5x2)*(2x5): \n");
for (int i = 0; i < 25; i++) {
    if (i % 5 != 0) {
        printf("| %.*f ", 1, MR[i]);
    }else{
        printf("\n");
        printf("| %.*f ", 1, MR[i]);
    }
}

std::getchar();
return 0;
```

Código principal 1

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).



Resultado de la ejecución

Matlab

A continuación los códigos de los distintos apartados pero esta vez en ***Matlab***

```
a = [1 2 1 2 1; 2 1 2 1 2]
b = [1 2; 1 2; 1 2; 1 2; 1 2]

F = b*a
```

Código principal 1

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
F =

     5     4     5     4     5
     5     4     5     4     5
     5     4     5     4     5
     5     4     5     4     5
     5     4     5     4     5
```

Resultado de la ejecución

Practica 5

Descripción

4. [OPTATIVO] Realizar pruebas con aritmética de precisión simple y comparar los resultados.

Trabajo realizado

Cambiamos la creación de las matrices por float y realizamos la misma operación que usamos en el anterior apartado.

CBlas

Código Programa:

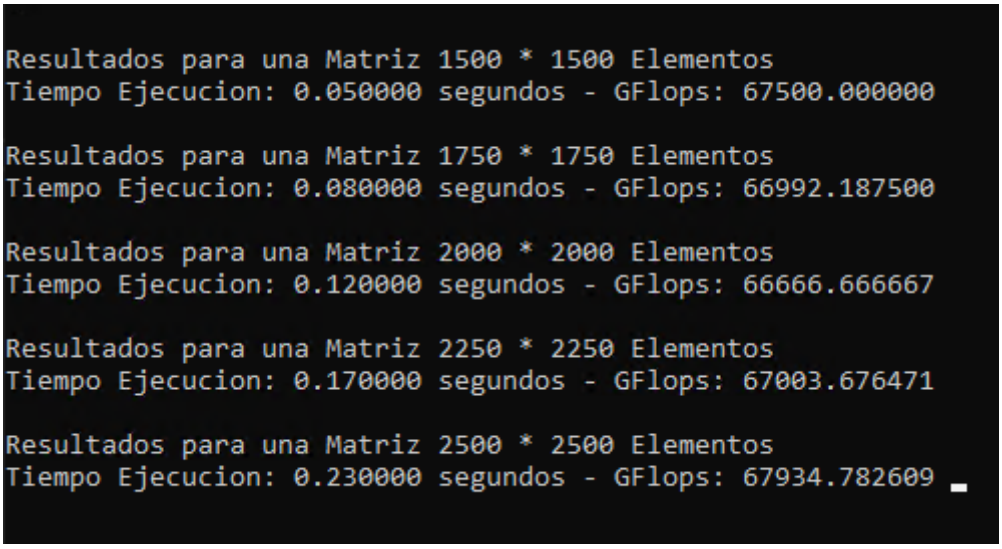
Los códigos de la librería ***CBlas*** de los distintos apartados serán puesto a continuación con la referencia del apartado en el pie de la foto.

```
int multMatrizFloat(int lenght) {

    const int N = lenght;
    const float alphaA = 1.0;
    const float betaA = 0.0;

    const MKL_INT m = N, n = N, k = N;
    const MKL_INT lda = N, ldb = N, ldc = N;

    float* A = (float*)mkl_malloc(N * N * sizeof(float), 32);
    float* B = (float*)mkl_malloc(N * N * sizeof(float), 32);
    float* C = (float*)mkl_malloc(N * N * sizeof(float), 32);

    srand((unsigned int)time(NULL));

    for (int i = 0; i < N * N; i++) {
        A[i] = (float)rand() / (float)RAND_MAX;
        B[i] = (float)rand() / (float)RAND_MAX;
        C[i] = (float)rand() / (float)RAND_MAX;
    }

    double seconds, GFlops;
    time_t start, finish;

    time(&start);
    for (int i = 0; i < 100; i++) {
        cblas_sgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, m, n, k, alphaA, A, lda, B, ldb, betaA, C, ldc);
    }
    time(&finish);

    seconds = difftime(finish, start) / 100;
    GFlops = (pow(N, 3) / seconds) * pow(10, -6);

    printf("\nResultados para una Matriz %i * %i de elementos de presicion simple\n", N, N);
    printf("Tiempo de ejecución: %1f segundos", seconds);
    printf("GFlops: %1f ", GFlops);

    mkl_free(A);
    mkl_free(B);
    mkl_free(C);

    return 0;
}
```

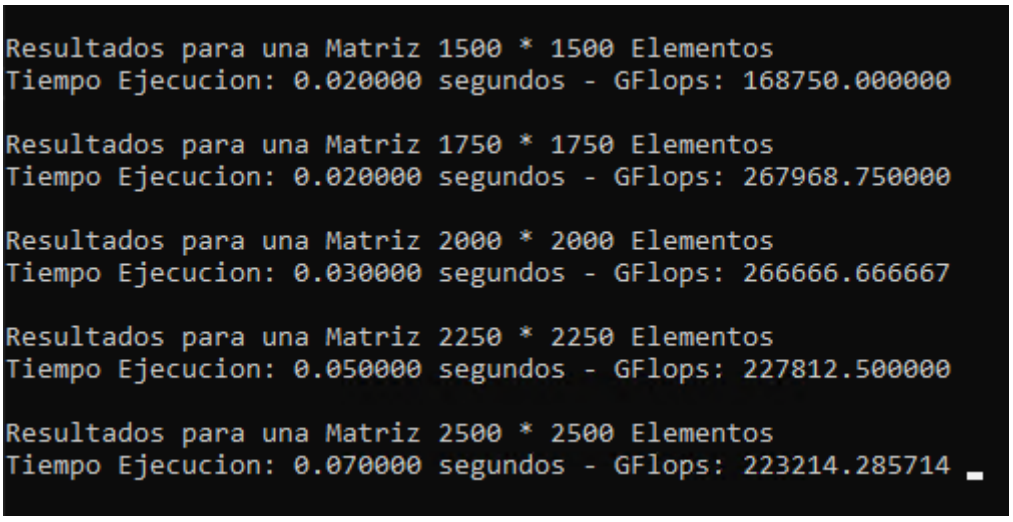
Código principal 1

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).



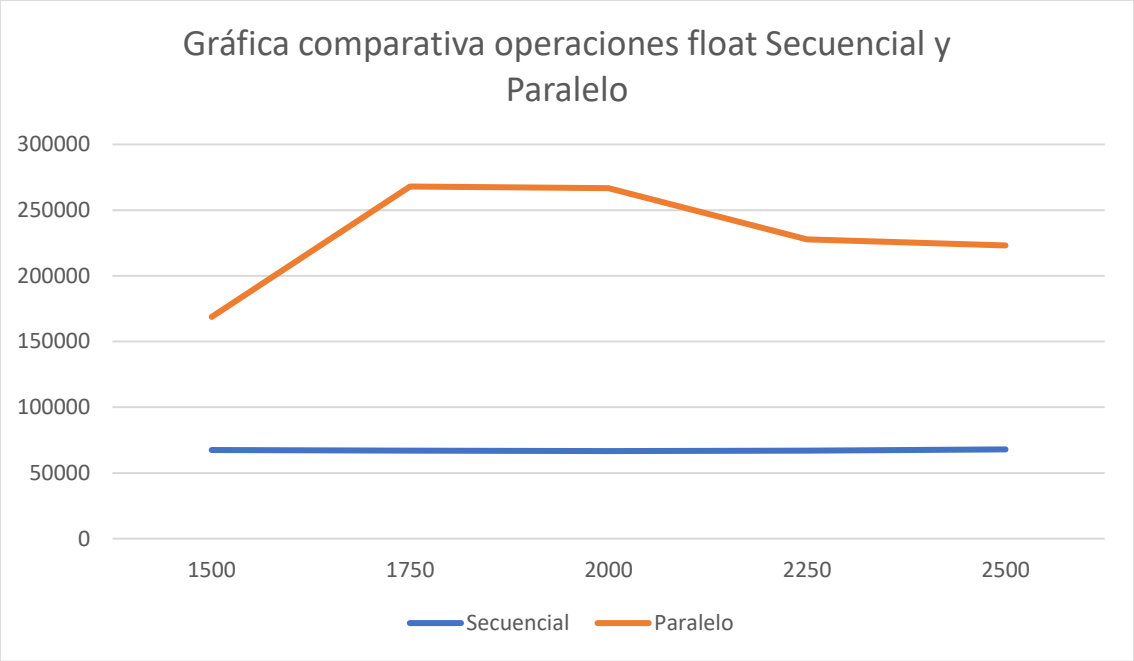
Resultado de la ejecución secuencial



Resultado de la ejecución paralelo

Tamaño	Secuencial	Paralelo
1500	67500	168750
1750	66992	267968
2000	66666	266666
2250	67003	227812
2500	67934	223214

Tabla comparativa float Secuencial y Paralelo medido en GFlops



Grafica comparativa float Secuencial y Paralelo medido en GFlops

Como comentamos anteriormente, en secuencial los GFlops no se aprecia tanta diferencia al aumentar la matriz, pero en paralelo si porque básicamente aumentan las operaciones que se realizan al mismo tiempo.

5. [OPTATIVO] Comparativa de tiempos de ejecución con otros lenguajes.

Lo realizamos en Java.

Código Programa:

```
    Random rand = new Random();
    int n = 100;
    double[][] matrizA = new double[n][n];
    double[][] matrizB = new double[n][n];
    double[][] producto = new double[matrizA.length][matrizB[0].length];
    //Filling matrix A
    for (int i = 0; i < matrizA.length; i++) {
        for (int j = 0; j < matrizA[i].length; j++) {
            matrizA[i][j] = rand.nextDouble();
        }
    }
    //Fillin matrix B
    for (int i = 0; i < matrizB.length; i++) {
        for (int j = 0; j < matrizB[i].length; j++) {
            matrizB[i][j] = rand.nextDouble();
        }
    }
    long start = System.currentTimeMillis();
    for (int e = 0; e < 100; e++) {
        for (int a = 0; a < matrizB[0].length; a++) {
            for (int i = 0; i < matrizA.length; i++) {
                double suma = 0.0;
                for (int j = 0; j < matrizA[0].length; j++) {
                    suma += matrizA[i][j] * matrizB[j][a];
                }
                producto[i][a] = suma;
            }
        }
    }
    long elapsedTimeMillis = System.currentTimeMillis() - start;
    float elapsedTimeSec = elapsedTimeMillis / 1000F;
    elapsedTimeSec = elapsedTimeSec/100;
    //Gflops
    System.out.println(Math.pow(n,3)*2/elapsedTimeSec/1000000);
}
```

Código principal 1

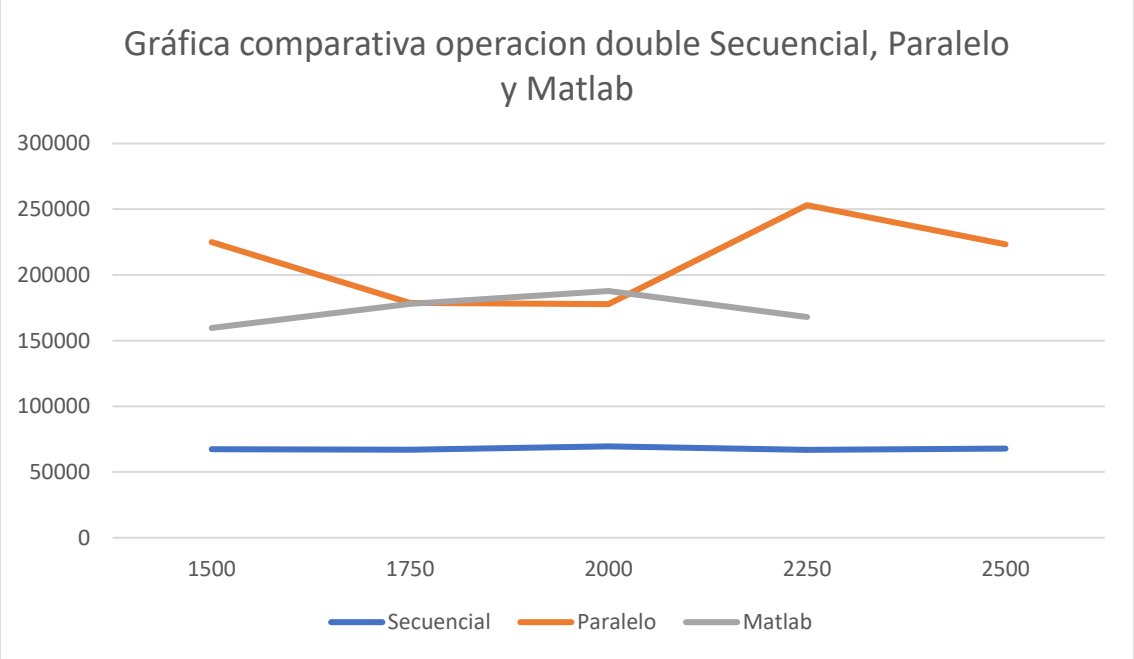
Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

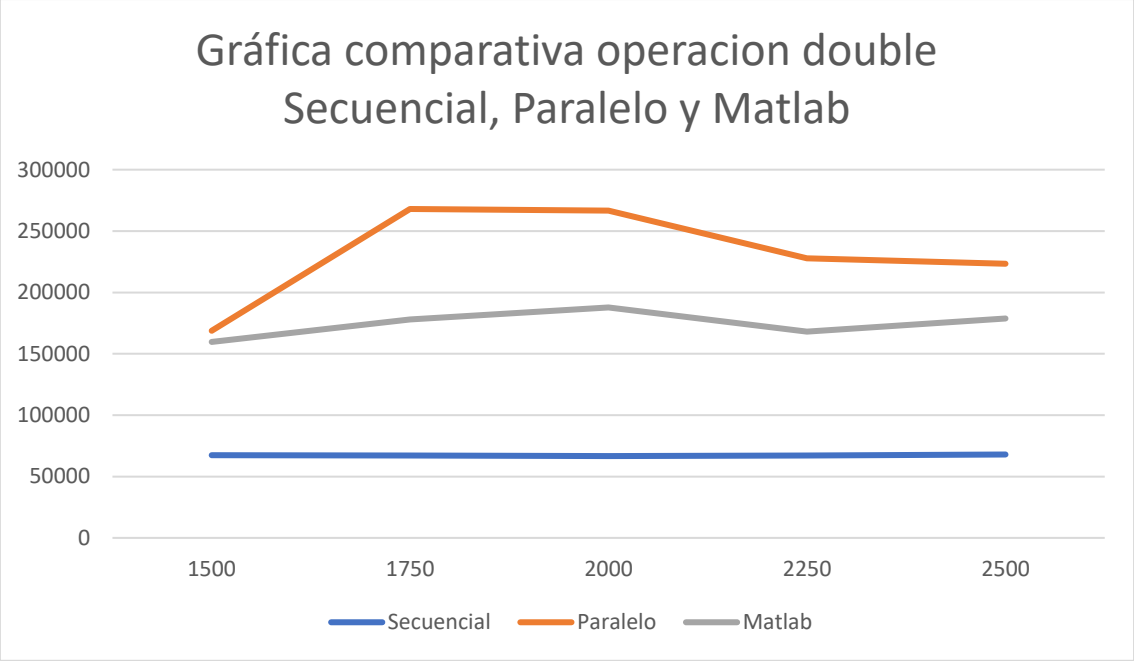
```
run:
Tiempo de ejecucion en GFlops 1600.0000357627875
```

Resultado de la ejecución

Al intentar comparar los GFlops que realizamos en Blas con Java nos dimos cuenta de que era imposible hacer una medición proporcional ya que en Java tardaba muchísimo más en calcular los GFlops con el mismo tamaño de matrices que usábamos para en Blas. Sin embargo, en Matlab daba algo proporcional entonces decidimos hacer la gráfica comparativa en Matlab.



Grafica comparativa double Secuencial, Paralelo y Matlab medido en GFlops



Grafica comparativa float Secuencial, Paralelo y Matlab medido en GFlops