MÉTODOS NUMÉRICOS PARA LA COMPUTACIÓN

**Tema 4: Librería BLAS**

**2020/21**

X de noviembre de 2020

**Grupo 03:** José María Amusquívar Poppe y Prashant Jeswani Tejwani

Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería en Informática

Índice

[Actividad práctica 2 3](#_Toc56766674)

[Ejercicio 1 3](#_Toc56766675)

[Ejercicio 2 3](#_Toc56766676)

[Ejercicio 3 3](#_Toc56766677)

[Ejercicio 4 (optativo) 3](#_Toc56766678)

[Actividad práctica 3 3](#_Toc56766679)

[Ejercicio 1 3](#_Toc56766680)

[Ejercicio 2 3](#_Toc56766681)

[Ejercicio 3 (optativo) 3](#_Toc56766682)

[Actividad práctica 4 4](#_Toc56766683)

[Ejercicio 1 4](#_Toc56766684)

[Ejercicio 2 (optativo) 4](#_Toc56766685)

[Actividad práctica 5 4](#_Toc56766686)

[Ejercicio 1 4](#_Toc56766687)

[Ejercicio 2 4](#_Toc56766688)

[Ejercicio 3 4](#_Toc56766689)

[Ejercicio 4 (optativo) 4](#_Toc56766690)

[Ejercicio 5 (optativo) 4](#_Toc56766691)

# 

# Actividad práctica 2

Se realizan los siguientes ejercicios usando la librería CBLAS nivel 1. Se emplea Matlab para verificar que el resultado es correcto.

## Ejercicio 1

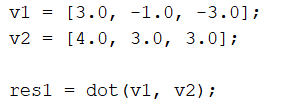
Se ha definido dos vectores 3D (Figura 1 y 2) y comprobado que su producto escalar es nulo:

Figura 3. Función de CBLAS para el producto escalar.

Figura 2. Vectores a operar (C++).

Figura 1. Vectores a operar (MATLAB).

Si se ejecuta el respectivo comando de CBLAS en C++ (Figura 3), donde el primer parámetro corresponde con el número de elementos del vector; el segundo y el cuarto con los respectivos vectores a operar; y el tercero y quinto con el valor de incremento, qué en este caso será igual a uno puesto que se desea operar los vectores recorriendo secuencialmente sus índices.

Realizando dicha operación se obtiene un resultado igual a cero, lo mismo ocurre si se realiza esta operación en MATLAB usando la función “dot()”. Entonces, se puede concluir que ambos vectores son ortogonales, por lo que tienen un punto de intersección.

## Ejercicio 2

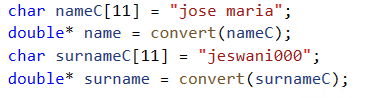
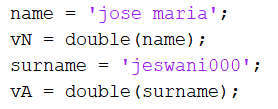
Se ha construido dos vectores conteniendo el valor ASCII de los 10 primeros caracteres de un nombre del integrante del grupo y el apellido del otro (Figura 4 y 5), rellenándolo con cero para llegar a los 10. Posteriormente, se obtiene el resultado de sumar al primer vector el triple del segundo. Y, finalmente, el resultado se mapea a caracteres ‘a.…z’.

Figura 6. Operación mult\*surname+name.

Figura 5. Vectores ASCII (C++).

Figura 4. Vectores ASCII (MATLAB).

Una vez definidos ambos vectores, se ejecuta (en C++) la función de CBLAS “daxpy()” (Figura 6), que multiplica por un escalar “mult” el vector “surname”, y a dicho resultado le suma el vector “name”. En MATLAB se obtiene este resultado haciendo explícitamente lo que se pide:

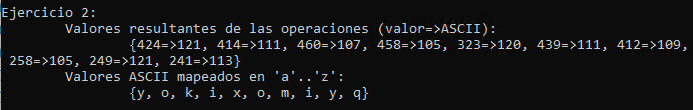
El resultado obtenido usando CBLAS se sobrescribe en el vector “name”, cuyos valores son:

Figura 7. Resultado y mapeado de los valores.

La primera lista de valores mostrados corresponden con los valores obtenidos del cálculo realizado usando “daxpy()” (Figura 6). Esta lista está compuesta por un par de valores, el primero representa el valor obtenido, su pareja corresponde con el primer valor mapeado en el rango 97-122 (a-z) en código ASCII decimal. Finalmente, la segunda lista corresponde con los caracteres mapeados obtenidos.

## Ejercicio 3

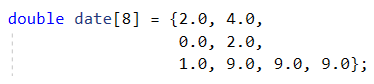
Se crea un vector conteniendo todos los dígitos de la fecha de nacimiento (Figura 8 y 9). La nota final de la asignatura de MNC es el resultado de calcular el módulo 11 de la norma2 de ese vector.

Figura 11. Cálculo de la norma y el módulo 11 (MATLAB).

Figura 10. Cálculo de la norma y el módulo 11 (C++).

Figura 9. Vector cumpleaños Prashant (MATLAB).

Figura 8. Vector cumpleaños José María (C++).

Para realizar este apartado, primero se comprobó que los resultados coincidían en ambos entornos. Una vez confirmado esto, se ha usado ambas fechas de cumpleaños para realizar los cálculos. El resultado de la figura 10 y 11 es **5.3707** y **6**, respectivamente. Podría haber ido peor.

## Ejercicio 4 (optativo)

A continuación, se estudia el efecto de los parámetros de incremento en C++ del ejercicio anterior.

El incremento especifica el espacio guardado entre índice e índice, por tanto, si se especifica un incremento de 1, el array será recorrido secuencialmente empezando por el primer elemento; si se especifica un 2, el array será recorrido dando saltos de 2 en 2, índice 0, índice 2, índice 4... Sin embargo, el incremento también modifica el valor máximo hasta el que se recorre:

**(1 + (n -1)\*abs (incx))**

donde "n" es el número de elementos del vector, e "incx" el incremento, por tanto, si éste último es 1, el máximo coincide con el número de elementos del vector, pero si éste es 2 el máximo será el doble del número de elementos menos 1. Por ello, se debe tener cuidado al especificar este valor pues puede generar cálculos erróneos accediendo a posiciones de memoria incorrectos.

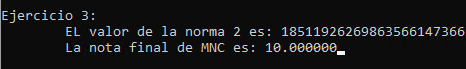
Por ejemplo, si en este ejercicio se especifica un incremento distinto a 1 sin modificar el número de elementos, se obtiene un valor muy grande que es incorrecto (Figura 12). Si se desea iterar sobre los índices pares (2), entonces el número de elementos debe reducirse a la mitad, 4 (Figura 13).

Figura 11. Incremento igual a 2, número de elementos a 4. Correcto

Figura 12. Incremento igual a 2, número de elementos igual a 8. Erróneo.

Si se especifica un incremento de 0, entonces se realiza la norma únicamente del primer elemento del vector (date[0] = 2.0).

# Actividad práctica 3

Se realizan los siguientes ejercicios usando la librería CBLAS nivel 2. Se emplea Matlab para verificar que el resultado es correcto.

## Ejercicio 1

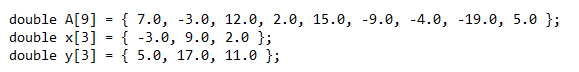
Se define una matriz (A) y dos vectores (x, y) (Figura 12) para realizar las siguientes operaciones:

Figura 12. Inicialización de vectores (C++).

Para realizar estas operaciones se emplea la función “dgemv()”, que trabaja a nivel matriz-vector. Esta función acepta parámetros que le especifican si se desea recorrer por filas o columnas, si se traspone la matriz o no, el incremento de los índices, los escalares que multiplicarán los objetos, así como sus dimensiones y los objetos en sí.

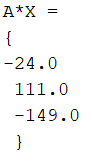
Figura 13. Función utilizada de CBLAS.

1. A\*x

La matriz “A” es de 3 por 3, así que el vector “x” será de 1 por 3, y el vector resultado será un vector columna de 3 filas. Empleando la función mostrada en la figura 13, pasando el parámetro “beta” igual a 0 para que la operación al lado derecha de la suma se elimine.



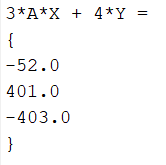
Comprobando estos resultados MATLAB de la siguiente manera:

El resultado obtenido en MATLAB de esta operación es:

1. 3\*A\*x+4\*y

La matriz “A” es de 3 por 3, así que el vector “x” será de 1 por 3, y se multiplicará por el escalar 3; el vector resultado de esta operación será un vector columna de 3 filas. Por lo que el vector “y” tendrá que ser un vector columna (se traspone), multiplicado por el escalar 4. Es resultado de esta operación será un vector columna (al igual que el anterior) de 3 filas.

Comprobando estos resultados MATLAB de la siguiente manera:

El resultado obtenido en MATLAB de esta operación es:

## Ejercicio 2

Se prueba el efecto de los parámetros de ‘layout’ y trasposición en C++ del ejercicio anterior.

En el anterior apartado, las operaciones fueron realizadas recorriendo los elementos por filas y sin trasponer la matriz. En este apartado se comprobarán los tres casos restantes.

Recorrer la matriz por columnas y trasponiéndola. Es lógico que los resultados obtenidos en esta prueba sean los mismos que si se recorre la matriz por filas y no se traspone.

Figura 14. Se recorre por columnas y matriz traspuesta.

Recorrer la matriz por columnas y sin trasponer. Esta prueba obtiene unos resultados no vistos hasta ahora, puesto que es un nuevo enfoque.

Figura 15. Se recorre por columnas y sin matriz traspuesta.

Recorrer la matriz por filas y trasponiéndola. Esta prueba obtiene unos resultados idénticos a los obtenidos en la prueba de recorrido por columnas y no traspuesta.

Figura 16. Se recorre por filas y matriz traspuesta.

Como se puede apreciar, de las 4 configuraciones posibles sólo existen dos variaciones posibles de resultados, ya que las otras dos son simplemente enfoques reflejados de éstas.

## Ejercicio 3 (optativo)

A continuación, se prueba el efecto de los parámetros de incremento y ‘lda’ en C++ del ejercicio anterior.

Al igual que en el apartado optativo 4 del ejercicio 2, la explicación de este apartado es similar, el incremento especifica el espacio guardado entre índice e índice, por lo que, si este valor se modifica, se ha de modificar también el número de elementos (el número de columnas de la matriz). En cambio, el parámetro “lda” especifica el número de filas de la matriz, por lo que representa el número de filas que tendrá el vector resultado.

# Actividad práctica 4

Se realizan los siguientes ejercicios usando la librería CBLAS nivel 3. Se emplea Matlab para verificar que el resultado es correcto.

## Ejercicio 1

Se definen tres matrices (A, B y C) de dimensión 3x3 y realizan las siguientes operaciones:

1. A\*B
2. A\*BT
3. 2\*A\*B + 3\*C

## Ejercicio 2 (optativo)

Se realiza una operación que implique matrices no cuadradas para generar como resultado una matriz de 5x5.

# Actividad práctica 5

## Ejercicio 1

Se crean tres matrices (A, B y C) de dimensión NxN y se rellenan con valores aleatorios tipo ‘double’.

## Ejercicio 2

Se calcula el número de GFLOPS para los distintos valores de N, realizando el promedio de 100 ejecuciones de la operación A\*B.

## Ejercicio 3

Se repite las pruebas utilizando el modo ‘Parallel’ para comparar los resultados.

## Ejercicio 4 (optativo)

Se realizan pruebas con aritmética de precisión simple para comparar los resultados.

## Ejercicio 5 (optativo)

Se realiza una comparativa de tiempos de ejecución con otros lenguajes como Python y Java.