

# Práctica 3. Librería BLAS

Métodos Numéricos para la Computación

Grado en Ingeniería Informática  
Escuela de Ingeniería Informática  
Universidad de Las Palmas de Gran Canaria

Curso 2015/2016

# Actividades de la Práctica

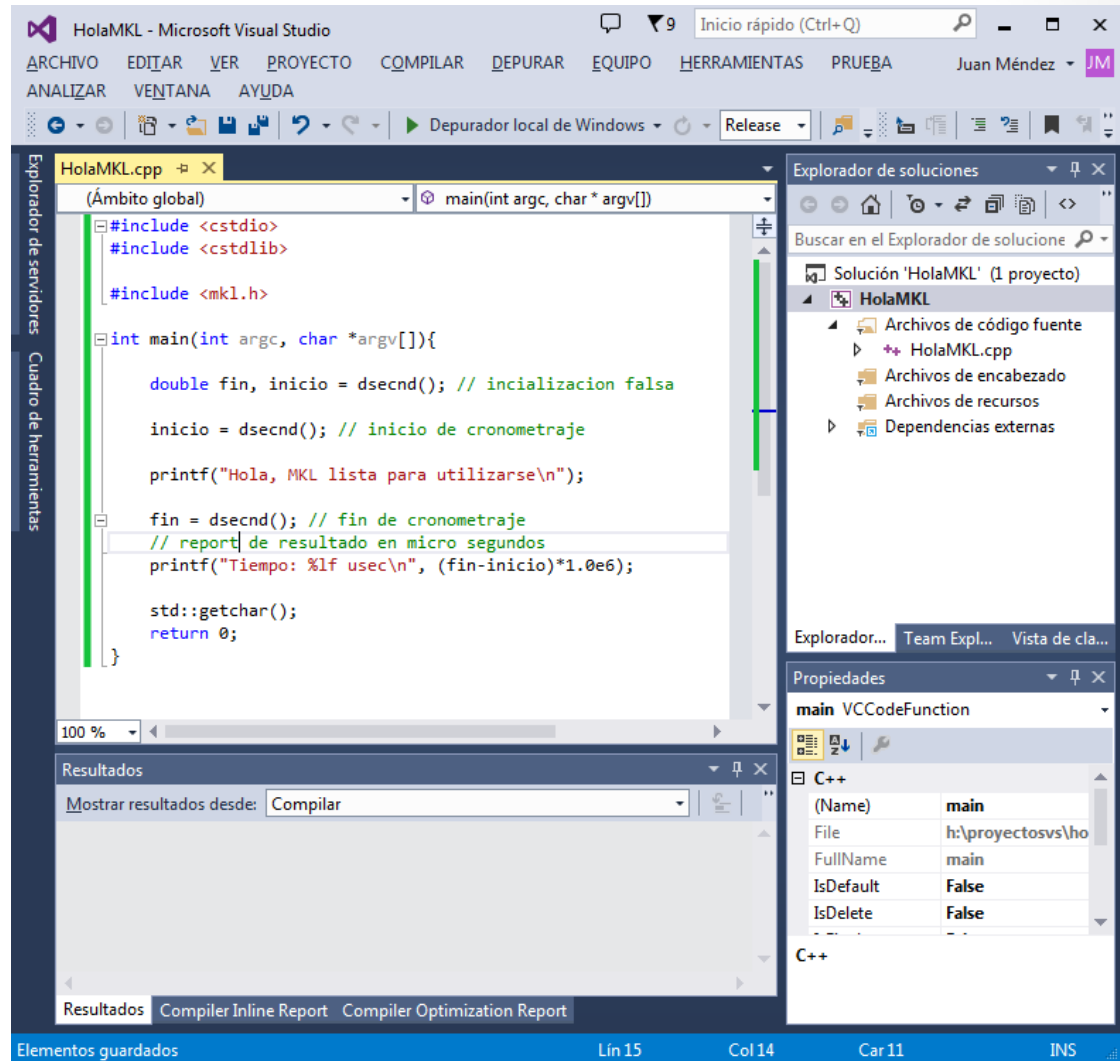
- Utilizar Visual Studio para crear proyectos que utilizan la librería MKL
- Adquirir experiencia en el uso de la librería BLAS, por ejemplo en operaciones Matriz\*Vector y Matriz\*Matriz
- Valorar el coste computacional en función del tamaño del problema
- Practicar con las versiones sequential y parallel de MKL.
- Elaborar la memoria incluyendo tablas y gráficos de los resultados

# Introducción a MKL

- Utilizaremos la librería BLAS en C a partir de la distribución incluida en la librería MKL (Math Kernel Library) de Intel que sigue un convenio de uso de CBLAS.
- MKL incluye las funciones CBLAS con prefijo `cblas_`xxx, además de funciones auxiliares específicas con prefijo `mkl_`xxx
- Se proporciona un manual de esta librería y el uso está facilitado por la integración de MKL en Visual Studio. Ahorro de tiempo en la creación de scripts de compilación y enlace.

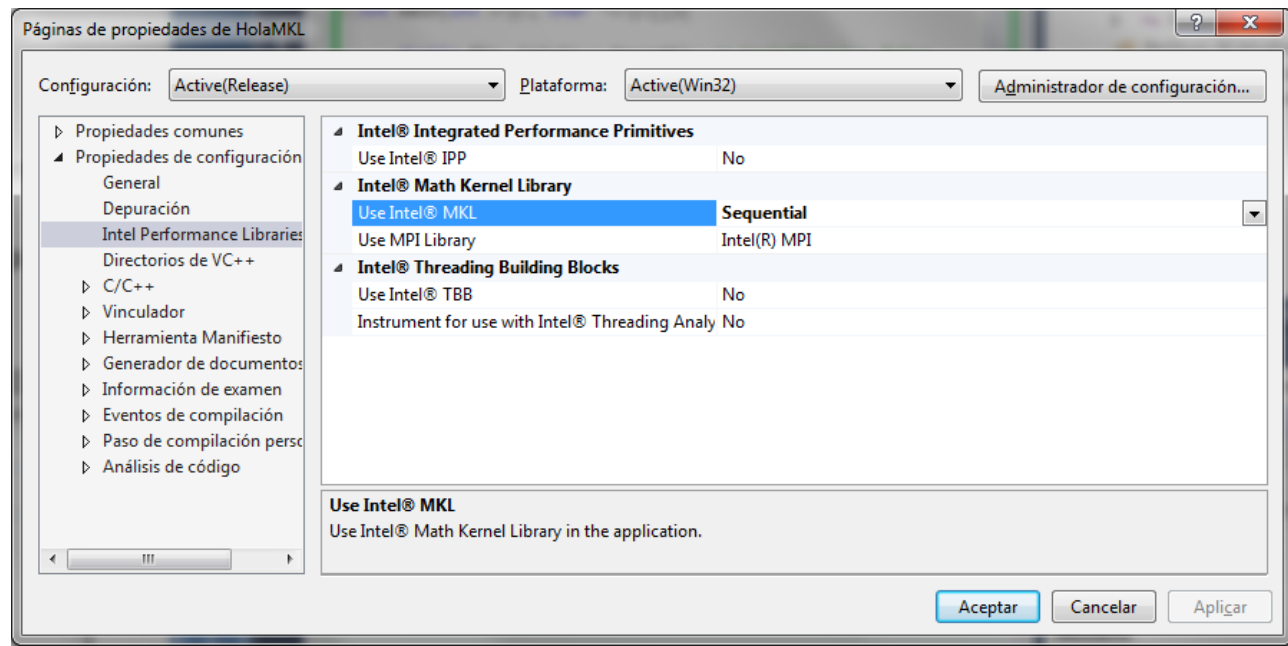
# Tarea 1. Hola MKL

- Crear un nuevo proyecto Visual Studio de tipo General y vacío, por ejemplo llamado HolaMKL.
- En Proyecto agregar un nuevo elemento, por ejemplo HolaMKL.cpp editando la función main() más o menos similar a la incluida
- El ejemplo puede ser un patrón general, con cronometraje incluido.



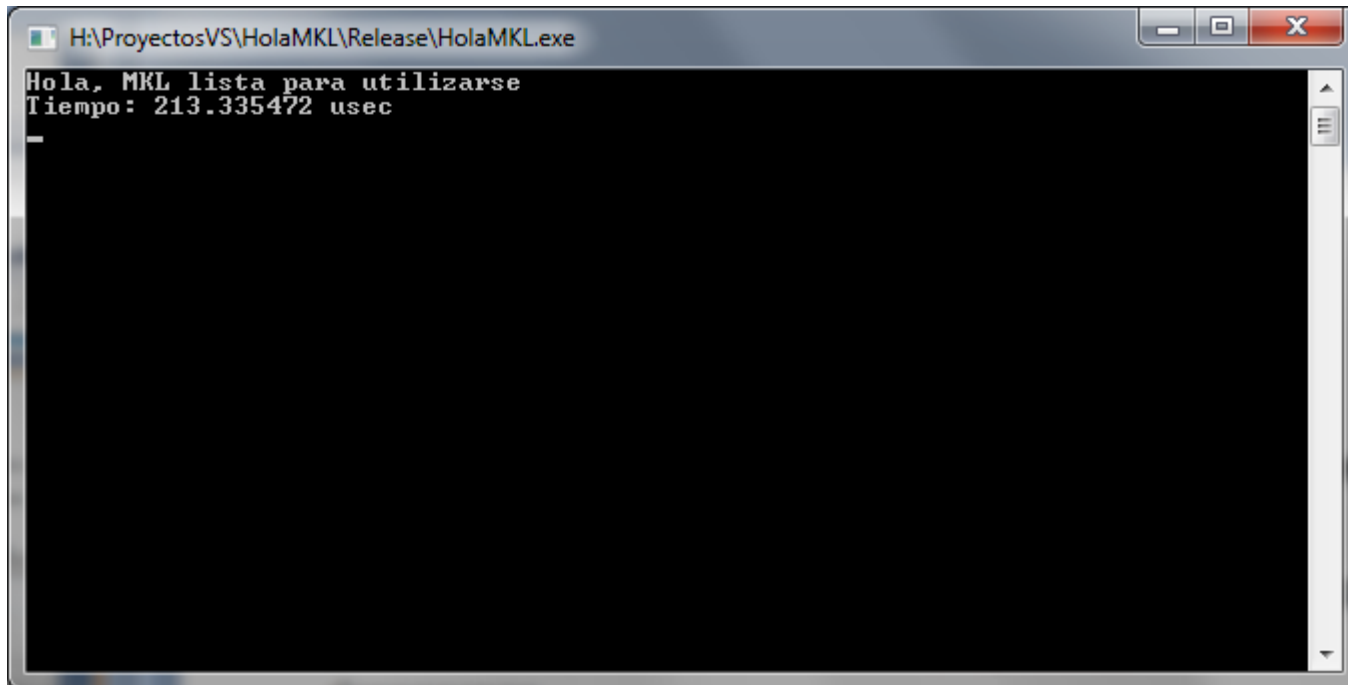
# HolaMKL

- El include de mkl.h tendrá señalado un error. Procederemos a incluir el entorno de soporte de MKL
- En las propiedades del proyecto HolaMKL, activar el soporte para el uso de MKL en la versión Sequential. El error desaparece
- Compilar y ejecutar la aplicación, mejor en modo Release



# HolaMKL

- La inclusión del soporte se puede extender incluyéndola para las versiones Debug y Release
- Return finaliza la aplicación



A screenshot of a Windows command prompt window. The title bar shows the file path: H:\ProyectosVS\HolaMKL\Release\HolaMKL.exe. The window contains the following text:

```
Hola, MKL lista para utilizarse  
Tiempo: 213.335472 usec
```

# Tarea 2. Uso de BLAS Nivel 1

- Las operaciones son del tipo vector\*vector
- Producto escalar: dot
- Norma: nrm2
- Suma ponderada: daxpy
- Crear un nuevo proyecto, por ejemplo vectorvector

(Ámbito global)

```

#include <stdio>
#include <stdlib.h>

#include <mk1.h>

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

    double A[4] = { 1.0, 2.0, 3.0, 4.0 };
    double B[4] = { 5.0, 6.0, 7.0, 8.0 };

    // cálculo de la norma o módulo
    double norma = cblas_dnrm2(4, A, 1);
    printf("Norma de A: %lf\n", norma);
    norma = cblas_dnrm2(4, B, 1);
    printf("Norma de B: %lf\n", norma);

    // cálculo del producto escalar
    double pescalar = cblas_ddot(4, A, 1, B, 1);
    printf("Producto escalar de A y B: %lf\n", pescalar);

    // suma ponderada: B = B + 2*A
    cblas_daxpy(4, 2.0, A, 1, B, 1);
    printf("Suma ponderada:\n");
    for (int i = 0; i < 4; i++) printf("%lf\n", B[i]);

    std::getchar();
    return 0;
}

```

Explicación en la pizarra de los argumentos, en especial inc

```

H:\ProyectosVS\vectorvector\Release\vectorvector.exe
Norma de A: 5.477226
Norma de B: 13.190906
Producto escalar de A y B: 70.000000
Suma ponderada:
7.000000
10.000000
13.000000
16.000000

```



# Tarea 3. Uso de BLAS Nivel 2

- Operación de producto  $\text{matrix} \times \text{vector}$
- $\text{d(double)ge(general)mv(matrix vector)}$
- La función será: `cblas_dgemv()`
- Crear un nuevo proyecto por ejemplo llamado `matrizvector`
- Las matrices deben almacenarse siguiendo un criterio.
- Las matrices pueden utilizarse con/sin trasposición

matrizvector - Microsoft Visual Studio

ARCHIVO EDITAR VER PROYECTO COMPILAR DEPURAR EQUIPO HERRAMIENTAS PRUEBA ANALIZAR VENTANA Juan Méndez JM AYUDA

Depurador local de Windows Release

HolaMKL.cpp matrizvector.cpp

(Ámbito global) main(int argc, char \*argv[])

```
#include <stdio>
#include <stdlib>

#include <mk1.h>

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

    double fin, inicio = dsecnd(); // inicializacion falsa

    double A[9] = {1.0,2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0};
    double B[3] = {10.0, 11.0, 12.0};
    double C[3]; // C = A*B

    inicio = dsecnd(); // inicio de cronometraje

    cblas_dgemv(CblasRowMajor, CblasNoTrans, 3, 3, 1.0, A, 3, B, 1, 0.0, C, 1);

    fin = dsecnd(); // fin de cronometraje

    for (int i = 0; i < 3; i++) printf("%lf\n",C[i]);

    // report de resultado en micro segundos
    printf("Tiempo: %lf msec\n", (fin - inicio)*1.0e3);

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

100 %

Resultados

Mostrar resultados desde: Compilar

Resultados Compiler Inline Report Compiler Optimization Report

Explorador de soluciones

Buscar en el Explorador de soluciones

Solución 'matrizvector' (1 proyecto)

- matrizvector
  - Archivos de código fuente
    - matrizvector.cpp
  - Archivos de encabezado
  - Archivos de recursos
  - Dependencias externas

Explorador... Team Expl... Vista de cla...

Propiedades

main VCCodeFunction

C++

(Name)	main
File	h:\proyectosvs\ma
FullName	main
IsDefault	False
IsDelete	False
IsFinal	False
IsInjected	False

C++

## No traspuesta: CblasNoTrans

```
H:\ProyectosVS\matrizvector\Release\matrizvector.exe
68.000000
167.000000
266.000000
Tiempo: 9.011933 msec
-
```

## Traspuesta: CblasTrans

```
H:\ProyectosVS\matrizvector\Release\matrizvector.exe
138.000000
171.000000
204.000000
Tiempo: 8.991265 msec
-
```

```
Toolbox Path Cache read in 0.34 seconds.
MATLAB Path initialized in 0.53 seconds.
>> A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =

     1     2     3
     4     5     6
     7     8     9
```

```
>> B = [10;11;12]
```

```
B =

    10
    11
    12
```

```
>> A*B
```

```
ans =

    68
   167
   266
```

```
>> A' * B
```

```
ans =

   138
   171
   204
```

```
>> |
```

# Tarea 3. Uso de BLAS Nivel 3

- Construir un programa C que utilizando MKL realice el producto matricial de grandes dimensiones utilizando: `cblas_dgemm()`
- Recordar la importancia del alineamiento de los datos, usar `mkl_malloc()` y `mkl_free()`
- Flops = numero de operaciones en punto flotante por segundo.

# Inicialización

```
matrizmatriz.cpp ×
(Ámbito global) main(int argc, char * ar

#include <stdio>
#include <stdlib>
#include <math>
#include <time>

#include <mkl.h>

#define N 1000
#define NTEST 100

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

    double fin, inicio = dsecnd(); // incializacion falsa
    double *A = (double*)mkl_malloc(N*N*sizeof(double), 64); // alineado a double
    if (A == (double*)NULL){perror("Error Malloc");exit(1);}

    double *B = (double*)mkl_malloc(N*N*sizeof(double), 64);
    if (B == (double*)NULL){perror("Error Malloc");exit(1);}

    double *C = (double*)mkl_malloc(N*N*sizeof(double), 64);
    if (C == (double*)NULL){perror("Error Malloc");exit(1);}

    // distribución aleatoria de las matrices A y B
    srand((unsigned int)time(NULL));
    for (int i = 0; i < N; i++){
        for (int j = 0; j < N; j++){
            A[i*N + j] = (double)rand() / (double)RAND_MAX;
            B[i*N + j] = (double)rand() / (double)RAND_MAX;
        }
    }
}
```

mkl\_malloc con alineamiento para datos del tipo double. Está relacionado con el uso de los sets de instrucciones vectoriales del tipo SSE.

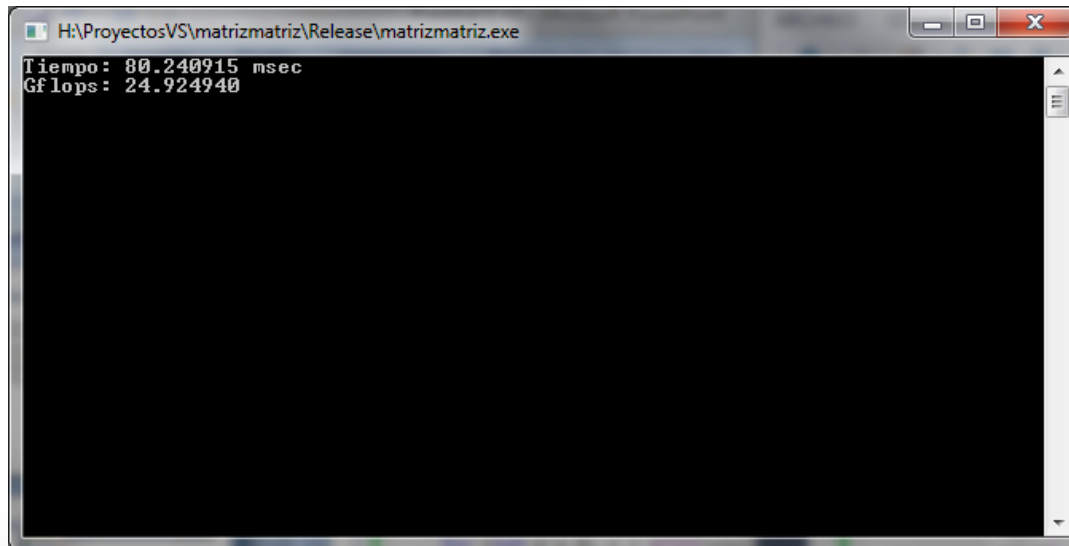
# Computo y prestaciones

```
inicio = dsecnd(); // inicio de cronometraje
for (int c = 0; c < NTEST;c++) // repetir el calculo NTEST veces
    cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, N, N, N, 1.0, A, N, B, N, 0.0, C, N);
fin = dsecnd(); // fin de cronometraje

double tiempo = (fin - inicio) / (double)NTEST; // tiempo medio
// report de resultado en milisegundos
printf("Tiempo: %lf msec\n", tiempo*1.0e3);
printf("Gflops: %lf\n", 2.0*pow((double)N,3.0)*1.0e-9/tiempo); // Gigafllops de potencia

mkl_free(A); // liberar espacio
mkl_free(B);
mkl_free(C);

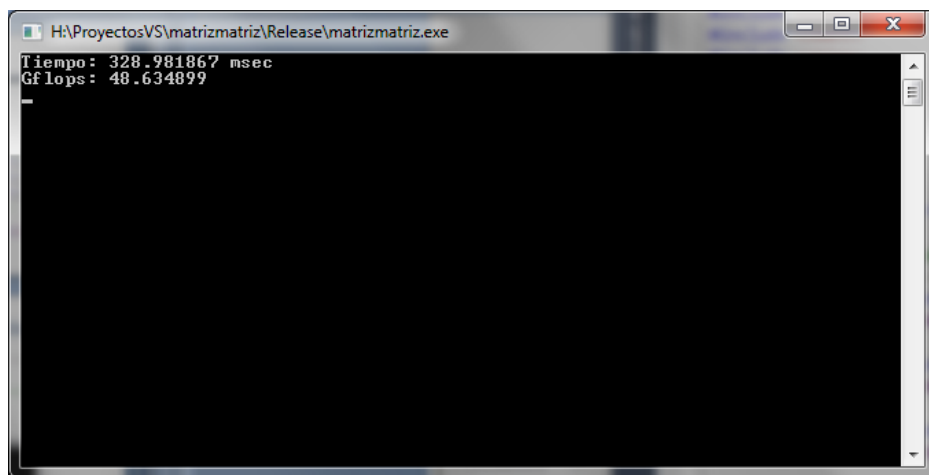
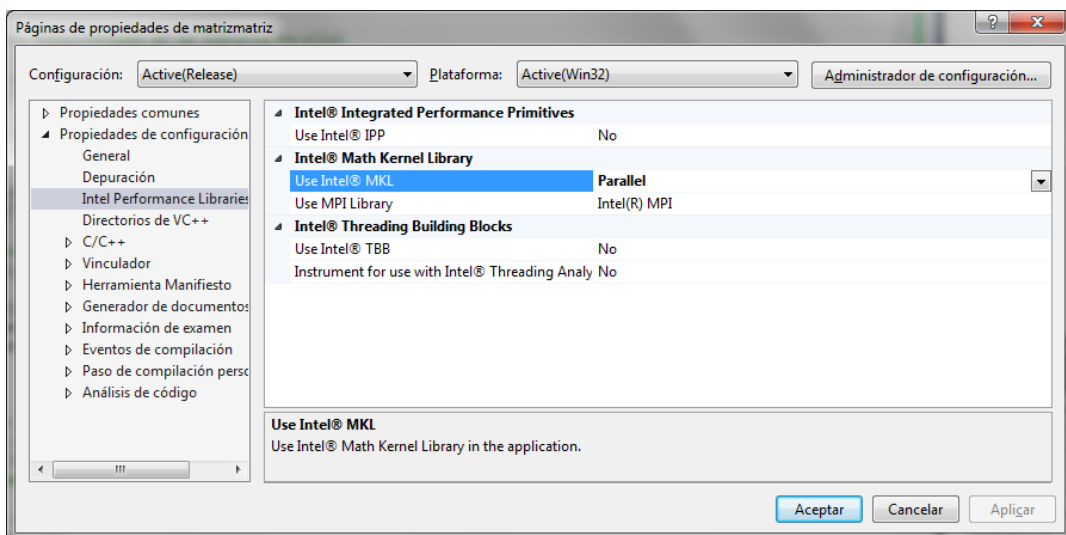
std::getchar();
return 0;
}
```



```
H:\ProyectosVS\matrizmatriz\Release\matrizmatriz.exe
Tiempo: 80.240915 msec
Gflops: 24.924940
```

# Librería versión Parallel

La versión Parallel utiliza múltiples núcleos para realizar el cálculo matricial. El control sobre el grado la forma de paralelismo es casi nulo. Pero se observa un gran aumento de la potencia. El control del paralelismo a nivel algorítmico será mayor cuando utilicemos, por ejemplo, OpenMP.



Producto matricial de  
dimensión: 2000x2000

# Tareas específicas

- Obtener una tabla de potencia en Gigaflops en función del tamaño  $N$  de las matrices. Incluir la tabla en la memoria.
- Construir en MATLAB una representación gráfica de la potencia en función de  $N$  e incluirla en la memoria.
- Realizar una comparativa entre la versión Sequential y la versión Parallel para diferentes tamaños del problema.
- En todas las gráficas indicar en el título el tipo de CPU empleado en las pruebas.
- Comentar el comportamiento observado en los datos recogidos y la posible razón del mismo.



# Que debe entregar el alumno?

- Cada alumno entregará en el Campus Virtual una memoria en PDF o Word en la que estará contenida una descripción del trabajo realizado, incluyendo descripción, el listado MATLAB o C de la actividad realizada y la captura de pantalla de las gráficas o imágenes generadas. Para autentificar las imágenes cuando sea posible el alumno incluirá su nombre en cada imagen mediante la función title()).
- En principio la tarea quedará abierta para su entrega hasta cierta fecha que se indicará.
- Se puede trabajar en grupo en el Laboratorio, pero la memoria elaborada y entregado será individual.

# Bibliografía