How to learn Applied Mathematics through modern FORTRAN

Department of Applied Mathematics
School of Aeronautical and Space Engineering
Technical University of Madrid (UPM)

Part I

User Manual

CHAPTER

ONE

FIRST EXAMPLES: CALCULUS AND ALGEBRA

1.1 My first program:Hello world

```
program TestSystemsofEquations


    use Linear_systems
    use Jacobian_module
    use Non_Linear_Systems


    implicit none
```

1.2 sum of a numeric series

Dar el resultado de la suma de los 100 primeros trminos de las siguientes series:

1. Serie de nmeros naturales.
2. Serie de nmeros naturales impares.
3. Serie numrica donde el trmino general de la serie es: $a_n = 1/n^2$ desde $n = 1$.
4. Serie numrica donde el trmino general de la serie es $a_n = 1/n!$ desde $n = 1$.
5. Serie numrica donde el trmino general de la serie es $a_n = (-1)^{n+1}/(2n-1)$ desde $n = 1$.

1.3 Operaciones con matrices y vectores

Considerar los vectores $V, W \in \mathbb{R}^N$ de componentes:

$$\{v_i = \frac{1}{i^2}, \ \ i = 1 \dots N\},$$

$$\{w_i = \frac{(-1)^{i+1}}{2i-1}, \ \ i = 1 \dots N\}.$$

Considerar la matriz $A \in \mathcal{M}_{N\times N}(\mathbb{R})$ donde su trmino genrico vale $a_{ij} = (i/N)^j$. Escribir un programa para calcular las operaciones siguientes con $N = 100$:

1. Suma de todas las componentes del vector V y del vector W.

2. Suma de todas las componentes de la matriz A.
3. Suma de las componentes del vector W mayores que cero.
4. Producto escalar de los vectores V y W.
5. Producto escalar del vector V y la columna N de la matriz A.
6. Suma de las componentes de vector que resulta de multiplicar la matriz A por el vector V.
7. Traza de la matriz A.

1.4 dynamic allocation of memory

Dada la matriz $A \in \mathcal{M}_{M\times M}(\mathbb{R})$ de trmino genrico

$$\{a_{ij} = (i/M)^j, \;\; i = 0, \dots M-1, \;\; j = 0, \dots M-1\}.$$

calcular las siguientes operaciones:

1. Calcular
$$\sum_{M=1}^{10} traza(A)$$
2. Calcular
$$\sum_{M=1}^{5} traza(A^2)$$
3. Calcular con $M = 4$
$$traza\left(\sum_{k=1}^{5} A^k\right)$$

1.5 Piecewise functions

Sean los vectores $X, F \in \mathbb{R}^{N+1}$. Las componentes de X almacenan los valores discretos del dominio de definicin y F las imgenes correspondientes de la funcin $F : \mathbb{R} \to \mathbb{R}$ continua a trozos siguiente:

$$F(x) = \begin{cases} 1, & a \le x \le -\dfrac{\pi}{2}, \\ \cos(\pi x), & -\dfrac{\pi}{2} < x < \dfrac{\pi}{2}, \\ 0, & \dfrac{\pi}{2} \le x \le b. \end{cases}$$

Considerar una particin equiespaciada de la forma:

$$\{x_i = a + i\Delta x, \;\; i = 0 \dots N\}, \qquad \Delta x = \frac{b-a}{N}, \qquad a < -\frac{\pi}{2}, \qquad b > \frac{\pi}{2}.$$

Se pide calcular la suma;

$$S_N = \sum_{i=0}^{N} F_i \Delta x$$

1. con $N = 10$
2. con $N = 20$
3. con $N = 100$

1.6 Series of functions

Aproximar mediante un desarrollo en serie de potencias de la forma

$$f(x) = \sum_{k=0}^{M} a_k \, x^k, \qquad a_k = \frac{f^{(k)}(0)}{k!},$$

las funciones $F : \mathbb{R} \to \mathbb{R}$, siguientes:

1. $f(x) = e^x$ y calcular el valor $f(1)$ con $M = 5$.
2. $f(x) = \sin(x)$ y calcular el valor $f(\pi/2)$ con $M = 8$.
3. $f(x) = \cosh(x)$ y calcular el valor $f(1)$ con $M = 10$.
4. $f(x) = \dfrac{1}{1-x}$ y calcular el valor $f(0.9)$ con $M = 20$.
5. $f(x) = e^x$ y calcular el valor ms preciso de $f(1)$ con doble precisin.
6. $f(x) = \sin(x)$ y calcular el valor ms preciso $f(\pi/2)$ con doble precisin.
7. $f(x) = \cosh(x)$ y calcular el valor ms preciso de $f(1)$ con doble precisin.
8. $f(x) = \dfrac{1}{1-x}$ y calcular el valor ms preciso de $f(0.9)$ con doble precisin.

1.7 Reading and writing data files

Crear los ficheros de datos ForTran con nombres `input_1.dat` e `input_2.dat` con la información siguiente:

Contenido del fichero de entrada `input_1.dat` :

```
Datos de entrada 1

1.2      3.4      6.2     -14.0    0.1
-25.2    -8.6     5.1      9.9    17.0
-1.0     -2.0     -5.4     -8.6    0.0
3.14     -11.9    -7.0     -12.1   9.2
6.66     5.32     0.001    0.2     0.001
```

Contenido del fichero de entrada `input_2.dat` :

```
Datos de entrada 2

1.2      3.4      6.2     -14.0    0.1      4.89    7.54
-25.2    -8.6     5.1     12.0     9.9      12.24   17.0
0.0      34.5     -1.0    -2.0     -43.04   -8.6    0.0
3.14     -11.9    71.0    7.0      17.0     -12.1   9.2
6.66     5.32     0.001   0.2      0.001    0.008   -0.027
54.0     77.1     -9.002  -13.2    0.017    65.53   -0.021
23.04    -51.98   -34.2   9.99     5.34     8.87    3.22
```

Escribir un programa que gestione los datos de los ficheros anteriores siguiendo los pasos siguientes: Declarar las matrices $A \in \mathcal{M}_{N\times N}(\mathbb{R})$, $B \in \mathcal{M}_{N\times 3}(\mathbb{R})$, $C \in \mathcal{M}_{N\times 2}(\mathbb{R})$ y los vectores $U, V, W, T \in \mathbb{R}^N$.
Leer el fichero de entrada (`input_1.dat` o `input_2.dat`) de la forma siguiente:

1. Cargar el fichero completo en la matriz A.
2. Cargar las cuatro primeras columnas del fichero en los vectores U, V , W y T.
3. Cargar la primera columna en el vector T y las tres últimas columnas en la matriz B.
4. Cargar la segunda columna en el vector U y las dos últimas columnas en la matriz C.
5. Cargar las columnas 1, 2 y 4 en la matriz B.

Además, el programa debe crear el fichero de salida (`output_1.dat` o `output_2.dat`), donde se irán escribiendo las matrices y vectores de los apartados anteriores. El formato de escritura debe ser el de números reales con cinco decimales.

Para el enunciado anterior, escribir los programas siguientes:

1. Programa 1 : Asignación estática de memoria.
   Ejecutar el programa por separado para los ficheros `input_1.dat` e `input_2.dat`. Para ello modificar las dimensiones y en nombre de los ficheros en el programa fuente.
2. Programa 2 : Asignación dinámica de memoria.
   Ejecutar el programa una única vez para gestionar los datos de los ficheros de entrada `input_1.dat` e `input_2.dat`.

1.8 Systems of linear equations

Implementar un mdulo para la resolucin de sistemas lineales de ecuaciones algebraicas. Los mtodos de resolucin propuestos son el de eliminacin Gaussiana, factorizacin LU, factorizacin LU de la biblioteca *Numerical Recipes* y Jacobi.

Para cada mtodo se pide:

- Validar los resultados con varios casos de prueba con dimensiones distintas.
- Evaluar tiempos de ejecucin.
- Comparar resultados con los mtodos restantes.

Aplicacin : Estudiar el condicionamiento de sistemas lineales de ecuaciones para matrices aleatorias y de Vandermonde.

1.9 Systems of nonlinear equations

Implementar un mdulo para la resolucin numrica de ecuaciones no lineales. Para funciones $F : \mathbb{R} \to \mathbb{R}$, los mtodos de resolucin propuestos son el de la biseccin y el de Newton-Raphson. Para funciones $F : \mathbb{R}^N \to \mathbb{R}^N$, se proponen el mtodo de Newton-Raphson con matriz Jacobiana analtica y el mtodo de Newton-Raphson con matriz Jacobiana numrica. Para la validacin de los mtodos propuestos, se pide implementar un mdulo con al menos tres funciones $F : \mathbb{R} \to \mathbb{R}$ y al menos tres funciones $F : \mathbb{R}^N \to \mathbb{R}^N$. Este mdulo debe contener las derivadas y matrices Jacobianas correspondientes de las funciones propuestas.

En el informe correspondiente, presentar tablas de soluciones numricas en cada paso de iteracin para las funciones de prueba propuestas.

1.10 Eigenvalues and eigenvectors

Implementar un mdulo para el clculo de autovalores y autovectores de una matriz. Los mtodos de resolucin propuestos son el mtodo de la potencia y el mtodo de la potencia inversa. Implementar el mtodo de la potencia inversa a partir de la matriz inversa y resolviendo el sistema lineal correspondiente.

Para cada mtodo se pide:

- Validar los resultados con varios casos de prueba con dimensiones distintas.
- Evaluar tiempos de ejecucin. Comparar tiempos de ejecucin del mtodo de la potencia inversa mediante los dos algoritmos propuestos : matriz inversa y solucin del sistema lineal.

Aplicacin : Estudiar el condicionamiento de sistemas lineales de ecuaciones para matrices aleatorias y de Vandermonde. Calcular la relacin $\lambda_{max}/\lambda_{min}$ de los casos de prueba presentados en el hito 1 y relacionar y discutir los resultados.

```
subroutine   power_method

integer :: i, j, k
integer, parameter :: PI = 4 * atan(1d0)
integer, parameter :: N = 20
real :: x(0:N), Vandermonde(0:N, 0:N), sigma
real :: a=-1, b=1
real V(0:N), V0(0:N)


x = [ ( a + (b-a)*i/N, i=0, N) ]

forall(i=0:N, j=0:N) Vandermonde(i,j) = x(i)**j

V = 1
V0 = 0
do while( abs(norm2(V)-norm2(V0)) > 1d-5 )
V0 = V
V = matmul( Vandermonde, V ) / norm2(V)
write(*,*) maxval(V)
end do
sigma = dot_product( V, V )
write(*,*) "sigma = ", sigma


end subroutine
```

1.11 Finite differences

1. Obtener las frmulas de las derivadas numricas primeras descentradas, con tres puntos equiespaciados a una distancia Δx.
2. A partir de la funcin $f(x) = e^x$ en el punto $x = 0$, representar grficamente el error total de las derivadas numricas frente al valor de Δx en precisin simple y doble. En particular, representar grficamente las derivadas primeras adelantada (definicin de derivada), centrada y descentradas y la derivada segunda, con tres puntos equiespaciados a una distancia Δx. Discutir los resultados obtenidos.

3. Resolver los problemas de contorno en ecuaciones diferenciales ordinarias siguientes:

   - **Problema 1:**
   $$u'' + u = 0, \quad x \in [-1,1], \qquad u(-1) = 1, \quad u(1) = 0,$$
   - **Problema 2:**
   $$u'' + u' - u = \sin(2\pi x), \quad x \in [-1,1], \qquad u(-1) = 0, \quad u'(1) = 0.$$

   Para los problemas citados anteriormente se pide:

   (a) A partir de las derivadas numricas con tres puntos equiespaciados escribir el sistema de ecuaciones resultante.

   (b) Obtener la solucin numrica mediante la resolucin de un sistema lineal de ecuaciones, con $N = 10$ y $N = 100$.

   (c) Representar grficamente los resultados obtenidos.

1.12 Numerical integration

Implementar un mdulo para la resolucin numrica de integrales definidas de funciones $F : \mathbb{R} \to \mathbb{R}$. Los mtodos de resolucin propuestos son las reglas del rectngulo, punto medio, trapecio y Simpson. Implementar un mdulo de funciones $F : \mathbb{R} \to \mathbb{R}$ de prueba para validar los mtodos numricos propuestos. Este mdulo debe contener al menos tres funciones con funciones primitivas conocidas y una funcin cuya funcin primitiva sea desconocida.

Evaluar el error de las soluciones numricas para cada mtodo propuesto y para distintos valores del incremento de la particin.

1.13 to be included

elemental advance = no dummy versus actual assumed shape explicit shape

global, local and scope in modules

1 versus 1.

tab instead of blanks

brackets

mask in intrinsic functions

! and &

;

camel case versus underscore

overloading

forall parallel

enter matrices by row or columns

lower bound: upper bound

array operations C = A + B

public versus private

encapsulamiento y ocultaci

FORALL (I=1:N-1, J=1:N, J¿I) A(I,J) = A(J,I)

```
x   = 1.23456789123456789123456789Q00
xd  = 1.23456789123456789123456789Q00
xdd = 1.23456789123456789123456789Q00
write(*, '(ES)' ) x  ! scientific notation 1.234 (first digit should be greater or equal than o
write(*, '(E)' ) x   ! exponential normalized  notation 0.1234 (first digit is zero )

write(*,*) " Single, double and quadruple precision "
```

```
write(*, '(E)' ) x
write(*, '(E)' ) xd
write(*, '(E)' ) xdd





!***********************************************************************
!*
!***********************************************************************
subroutine type_element




interface operator (+)
module procedure element
end interface

character(len=20) :: name
real (kind=4) :: x
real (kind=8) :: xd
real (kind=16) :: xdd

real :: a, b, c;
!  real :: a1, a2, a3;


type (person) :: father = person( "juan"), mother = person("cris")

associate ( name1 => father%name, name2=>mother%name )
name = trim(name1) // trim(name2)
end associate

a = 1; b = 1 ; c = 1;
associate ( a1 =>a, a2 =>b, a3=>c )
a3 = a1 + a2
write(*,*) " a3 = ", a3
end associate
write(*,*) " c = ", c
!    write(*,*) " a3 = ", a3

!   write(*,*) " father =", father % name
write(*,*) " element =", father + mother
write(*,*) " name =", name

x   = 1.234567891234567891234567890Q00
xd  = 1.234567891234567891234567890Q00
xdd = 1.234567891234567891234567890Q00
write(*, '(ES)' ) x  ! scientific notation 1.234 (first digit should be greater or equal than or
write(*, '(E)' ) x   ! exponential normalized  notation 0.1234 (first digit is zero )

write(*,*) " Single, double and quadruple precision "
write(*, '(E)' ) x
```

```
write(*, '(E)' ) xd
write(*, '(E)' ) xdd

write(*, '(3(E, :, ","))' ) x, xd, xdd

end subroutine



!***********************************************************************
!*
!***********************************************************************
subroutine Hello_world


write(*,'(a)', advance='no') " Hello world..... "
write(*,'(a)', advance='no') " press enter"
read(*,*)

end subroutine



!***********************************************************************
!*
!***********************************************************************
subroutine  commandline

character(len=256) :: line,  enval
integer :: i, iarg, stat, clen, len
integer :: estat, cstat

iarg = command_argument_count()
write(*,*) " iarg = ", iarg
do i=1,iarg
call get_command_argument(i,line,clen,stat)
write (*,'(I0,A,A)') i,': ',line(1:clen)
end do
call get_command(line,clen,stat)
write (*,'(A)') line(1:clen)

call get_environment_variable('HOSTNAME',enval,len,stat)
if (stat == 0) write (*,'(A,A)') 'Host=', enval(1:len)
call get_environment_variable('USER',enval,len,stat)
if (stat == 0) write (*,'(A,A)') 'User=', enval(1:len)


! call execute_command_line('ls -al', .TRUE., estat, cstat)
call execute_command_line('dir', .TRUE., estat, cstat)
if (estat==0) write (*,'(A)') "Command completed successfully"

end subroutine

!***********************************************************************
!*
```

```
!************************************************************************
subroutine  allocate_characteristics

real, allocatable :: V(:), A(:, :)
character(:), allocatable :: S
integer :: i, j, N

real, pointer ::  B(:,:), Diagonal(:)
real, pointer :: memory(:)

real :: x, y, z
class(*), pointer :: p1(:)


N = 10
V = [ ( i/real(N), i=1, N ) ]  ! automatic allocation allocate( V(N) )
write(*,* ) " V = ", V

N = 2
A = reshape( [ (  ( (i/real(N))**j ,i=1, N ), j=1, N ) ], [N, N] )
do i=1, N
write(*,* ) " A = ", A(i,:)
end do

N = 4
A = reshape( [ (  ( (i/real(N))**j ,i=1, N ), j=1, N ) ], [N, N] )
do i=1, N
write(*,'(A, 100f6.2)' ) " A = ", A(i,:)
end do

S = "Hello world"
write(*,*) " S = ", S, len(S)

S = "Hello"
write(*,*) " S = ", S, len(S)


allocate( memory(1:N*N) )
B(1:N,1:N) => memory
memory = 0
forall(i=1:N) B(i,i) = 10.
do i=1, N
write(*,'(A, *(f6.2) )' ) " B = ", B(i,:)
end do
diagonal => memory(::N+1)
write(*,'(A, 100f6.2)' ) " diagonal = ", diagonal
write(*,'(A, 100f6.2)' ) " trace = ", sum(diagonal)
write(*,'(A, 100f6.2)' ) " trace = ", sum(memory(::N+1))

x = 1; y = 2; z = 3;

write(*,'("i=",I0,", REALs=",*(G0,1X),"....")') i, x, y, z  ! C++ style


allocate( integer :: p1(5) ) ! p1 is an array of integers
```

```
select type (p1)
type is (integer)
p1 = [( i, i=1, size(p1) ) ]
write(*,'(" p1 = ", *(I0, 1x) )')  p1

class default
stop 'Error in type selection'
end select

deallocate(p1)
allocate( real :: p1(3) ) ! now p1 is an array of reals

select type (p1)
type is (real)
p1 = [( i, i=1, size(p1) )  ]
write(*,'(" p1 = ", *(G0, 1x) )')  p1

class default
stop 'Error in type selection'
end select



end subroutine
```

CHAPTER

TWO

FIRST EXAMPLES: CALCULUS AND ALGEBRA

2.1 My first program:Hello world

```
program TestLagrangeInterpolation

   use Lagrange_interpolation


   implicit none
```

2.2 sum of a numeric series

Dar el resultado de la suma de los 100 primeros trminos de las siguientes series:

1. Serie de nmeros naturales.
2. Serie de nmeros naturales impares.
3. Serie numrica donde el trmino general de la serie es: $a_n = 1/n^2$ desde $n = 1$.
4. Serie numrica donde el trmino general de la serie es $a_n = 1/n!$ desde $n = 1$.
5. Serie numrica donde el trmino general de la serie es $a_n = (-1)^{n+1}/(2n-1)$ desde $n = 1$.

2.3 Operaciones con matrices y vectores

Considerar los vectores $V, W \in \mathbb{R}^N$ de componentes:

$$\{v_i = \frac{1}{i^2}, \ \ i = 1 \ldots N\},$$

$$\{w_i = \frac{(-1)^{i+1}}{2i-1}, \ \ i = 1 \ldots N\}.$$

Considerar la matriz $A \in \mathcal{M}_{N\times N}(\mathbb{R})$ donde su trmino genrico vale $a_{ij} = (i/N)^j$. Escribir un programa para calcular las operaciones siguientes con $N = 100$:

1. Suma de todas las componentes del vector V y del vector W.
2. Suma de todas las componentes de la matriz A.
3. Suma de las componentes del vector W mayores que cero.

4. Producto escalar de los vectores V y W.
5. Producto escalar del vector V y la columna N de la matriz A.
6. Suma de las componentes de vector que resulta de multiplicar la matriz A por el vector V.
7. Traza de la matriz A.

2.4 dynamic allocation of memory

Dada la matriz $A \in \mathcal{M}_{M\times M}(\mathbb{R})$ de trmino genrico

$$\{a_{ij} = (i/M)^j,\ \ i = 0,\dots M-1,\ \ j = 0,\dots M-1\}.$$

calcular las siguientes operaciones:

1. Calcular
$$\sum_{M=1}^{10} traza(A)$$
2. Calcular
$$\sum_{M=1}^{5} traza(A^2)$$
3. Calcular con $M = 4$
$$traza\left(\sum_{k=1}^{5} A^k\right)$$

2.5 Piecewise functions

Sean los vectores $X, F \in \mathbb{R}^{N+1}$. Las componentes de X almacenan los valores discretos del dominio de definicin y F las imgenes correspondientes de la funcin $F : \mathbb{R} \to \mathbb{R}$ continua a trozos siguiente:

$$F(x) = \begin{cases} 1, & a \le x \le -\dfrac{\pi}{2}, \\ \cos(\pi x), & -\dfrac{\pi}{2} < x < \dfrac{\pi}{2}, \\ 0, & \dfrac{\pi}{2} \le x \le b. \end{cases}$$

Considerar una particin equiespaciada de la forma:

$$\{x_i = a + i\Delta x,\ \ i = 0\dots N\}, \qquad \Delta x = \frac{b-a}{N}, \qquad a < -\frac{\pi}{2}, \qquad b > \frac{\pi}{2}.$$

Se pide calcular la suma;

$$S_N = \sum_{i=0}^{N} F_i \Delta x$$

1. con $N = 10$
2. con $N = 20$
3. con $N = 100$

2.6 Series of functions

Aproximar mediante un desarrollo en serie de potencias de la forma

$$f(x) = \sum_{k=0}^{M} a_k \, x^k, \qquad\qquad a_k = \frac{f^{(k)}(0)}{k!},$$

las funciones $F : \mathbb{R} \rightarrow \mathbb{R}$, siguientes:

1. $f(x) = e^x$ y calcular el valor $f(1)$ con $M = 5$.
2. $f(x) = \sin(x)$ y calcular el valor $f(\pi/2)$ con $M = 8$.
3. $f(x) = \cosh(x)$ y calcular el valor $f(1)$ con $M = 10$.
4. $f(x) = \dfrac{1}{1-x}$ y calcular el valor $f(0.9)$ con $M = 20$.
5. $f(x) = e^x$ y calcular el valor ms preciso de $f(1)$ con doble precisin.
6. $f(x) = \sin(x)$ y calcular el valor ms preciso $f(\pi/2)$ con doble precisin.
7. $f(x) = \cosh(x)$ y calcular el valor ms preciso de $f(1)$ con doble precisin.
8. $f(x) = \dfrac{1}{1-x}$ y calcular el valor ms preciso de $f(0.9)$ con doble precisin.

2.7 Reading and writing data files

Crear los ficheros de datos ForTran con nombres `input_1.dat` e `input_2.dat` con la información siguiente:

Contenido del fichero de entrada `input_1.dat` :

```
Datos de entrada 1

1.2     3.4     6.2    -14.0    0.1
-25.2   -8.6    5.1     9.9    17.0
-1.0    -2.0    -5.4    -8.6    0.0
3.14    -11.9   -7.0    -12.1   9.2
6.66    5.32    0.001   0.2     0.001
```

Contenido del fichero de entrada `input_2.dat` :

```
Datos de entrada 2

1.2     3.4     6.2    -14.0    0.1      4.89   7.54
-25.2   -8.6    5.1    12.0     9.9      12.24  17.0
0.0     34.5    -1.0   -2.0     -43.04   -8.6    0.0
3.14    -11.9   71.0   7.0     17.0      -12.1   9.2
6.66    5.32    0.001   0.2     0.001    0.008  -0.027
54.0    77.1    -9.002  -13.2   0.017    65.53  -0.021
23.04   -51.98  -34.2   9.99    5.34     8.87    3.22
```

Escribir un programa que gestione los datos de los ficheros anteriores siguiendo los pasos siguientes: Declarar las matrices $A \in \mathcal{M}_{N\times N}(\mathbb{R})$, $B \in \mathcal{M}_{N\times 3}(\mathbb{R})$, $C \in \mathcal{M}_{N\times 2}(\mathbb{R})$ y los vectores $U, V, W, T \in \mathbb{R}^N$. Leer el fichero de entrada (`input_1.dat` o `input_2.dat`) de la forma siguiente:

1. Cargar el fichero completo en la matriz A.
2. Cargar las cuatro primeras columnas del fichero en los vectores U, V , W y T.
3. Cargar la primera columna en el vector T y las tres últimas columnas en la matriz B.
4. Cargar la segunda columna en el vector U y las dos últimas columnas en la matriz C.
5. Cargar las columnas 1, 2 y 4 en la matriz B.

Además, el programa debe crear el fichero de salida (`output_1.dat` o `output_2.dat`), donde se irán escribiendo las matrices y vectores de los apartados anteriores. El formato de escritura debe ser el de números reales con cinco decimales.

Para el enunciado anterior, escribir los programas siguientes:

1. Programa 1 : Asignación estática de memoria.
   Ejecutar el programa por separado para los ficheros `input_1.dat` e `input_2.dat`. Para ello modificar las dimensiones y en nombre de los ficheros en el programa fuente.
2. Programa 2 : Asignación dinámica de memoria.
   Ejecutar el programa una única vez para gestionar los datos de los ficheros de entrada `input_1.dat` e `input_2.dat`.

2.8 Systems of linear equations

Implementar un mdulo para la resolucin de sistemas lineales de ecuaciones algebraicas. Los mtodos de resolucin propuestos son el de eliminacin Gaussiana, factorizacin LU, factorizacin LU de la biblioteca *Numerical Recipes* y Jacobi.

Para cada mtodo se pide:

- Validar los resultados con varios casos de prueba con dimensiones distintas.
- Evaluar tiempos de ejecucin.
- Comparar resultados con los mtodos restantes.

Aplicacin : Estudiar el condicionamiento de sistemas lineales de ecuaciones para matrices aleatorias y de Vandermonde.

2.9 Systems of nonlinear equations

Implementar un mdulo para la resolucin numrica de ecuaciones no lineales. Para funciones $F : \mathbb{R} \to \mathbb{R}$, los mtodos de resolucin propuestos son el de la biseccin y el de Newton-Raphson. Para funciones $F : \mathbb{R}^N \to \mathbb{R}^N$, se proponen el mtodo de Newton-Raphson con matriz Jacobiana analtica y el mtodo de Newton-Raphson con matriz Jacobiana numrica. Para la validacin de los mtodos propuestos, se pide implementar un mdulo con al menos tres funciones $F : \mathbb{R} \to \mathbb{R}$ y al menos tres funciones $F : \mathbb{R}^N \to \mathbb{R}^N$. Este mdulo debe contener las derivadas y matrices Jacobianas correspondientes de las funciones propuestas.

En el informe correspondiente, presentar tablas de soluciones numricas en cada paso de iteracin para las funciones de prueba propuestas.

2.10 Eigenvalues and eigenvectors

Implementar un mdulo para el clculo de autovalores y autovectores de una matriz. Los mtodos de resolucin propuestos son el mtodo de la potencia y el mtodo de la potencia inversa. Implementar el mtodo de la potencia inversa a partir de la matriz inversa y resolviendo el sistema lineal correspondiente.

Para cada mtodo se pide:

- Validar los resultados con varios casos de prueba con dimensiones distintas.
- Evaluar tiempos de ejecucin. Comparar tiempos de ejecucin del mtodo de la potencia inversa mediante los dos algoritmos propuestos : matriz inversa y solucin del sistema lineal.

Aplicacin : Estudiar el condicionamiento de sistemas lineales de ecuaciones para matrices aleatorias y de Vandermonde. Calcular la relacin $\lambda_{max}/\lambda_{min}$ de los casos de prueba presentados en el hito 1 y relacionar y discutir los resultados.

```
subroutine   power_method

integer :: i, j, k
integer, parameter :: PI = 4 * atan(1d0)
integer, parameter :: N = 20
real :: x(0:N), Vandermonde(0:N, 0:N), sigma
real :: a=-1, b=1
real V(0:N), V0(0:N)


x = [ ( a + (b-a)*i/N, i=0, N) ]

forall(i=0:N, j=0:N) Vandermonde(i,j) = x(i)**j

V = 1
V0 = 0
do while( abs(norm2(V)-norm2(V0)) > 1d-5 )
V0 = V
V = matmul( Vandermonde, V ) / norm2(V)
write(*,*) maxval(V)
end do
sigma = dot_product( V, V )
write(*,*) "sigma = ", sigma


end subroutine
```

2.11 Finite differences

1. Obtener las frmulas de las derivadas numricas primeras descentradas, con tres puntos equiespaciados a una distancia Δx.
2. A partir de la funcin $f(x) = e^x$ en el punto $x = 0$, representar grficamente el error total de las derivadas numricas frente al valor de Δx en precisin simple y doble. En particular, representar grficamente las derivadas primeras adelantada (definicin de derivada), centrada y descentradas y la derivada segunda, con tres puntos equiespaciados a una distancia Δx. Discutir los resultados obtenidos.

3. Resolver los problemas de contorno en ecuaciones diferenciales ordinarias siguientes:

   - **Problema 1:**
   $$u'' + u = 0, \quad x \in [-1, 1], \qquad u(-1) = 1, \quad u(1) = 0,$$
   - **Problema 2:**
   $$u'' + u' - u = \sin(2\pi x), \quad x \in [-1, 1], \qquad u(-1) = 0, \quad u'(1) = 0.$$

   Para los problemas citados anteriormente se pide:

   (a) A partir de las derivadas numricas con tres puntos equiespaciados escribir el sistema de ecuaciones resultante.

   (b) Obtener la solucin numrica mediante la resolucin de un sistema lineal de ecuaciones, con $N = 10$ y $N = 100$.

   (c) Representar grficamente los resultados obtenidos.

2.12 Numerical integration

Implementar un mdulo para la resolucin numrica de integrales definidas de funciones $F : \mathbb{R} \to \mathbb{R}$. Los mtodos de resolucin propuestos son las reglas del rectngulo, punto medio, trapecio y Simpson. Implementar un mdulo de funciones $F : \mathbb{R} \to \mathbb{R}$ de prueba para validar los mtodos numricos propuestos. Este mdulo debe contener al menos tres funciones con funciones primitivas conocidas y una funcin cuya funcin primitiva sea desconocida.

Evaluar el error de las soluciones numricas para cada mtodo propuesto y para distintos valores del incremento de la particin.

2.13 to be included

elemental advance = no dummy versus actual assumed shape explicit shape
global, local and scope in modules
1 versus 1.
tab instead of blanks
brackets
mask in intrinsic functions
! and &
;
camel case versus underscore
overloading
forall parallel
enter matrices by row or columns
lower bound: upper bound
array operations C = A + B
public versus private
encapsulamiento y ocultaci
FORALL (I=1:N-1, J=1:N, J¿I) A(I,J) = A(J,I)

```
x   = 1.234567891234567891234567890Q00
xd  = 1.234567891234567891234567890Q00
xdd = 1.234567891234567891234567890Q00
write(*, '(ES)' ) x  ! scientific notation 1.234 (first digit should be greater or equal than o
write(*, '(E)' ) x   ! exponential normalized  notation 0.1234 (first digit is zero )

write(*,*) " Single, double and quadruple precision "
```

```
write(*, '(E)' ) x
write(*, '(E)' ) xd
write(*, '(E)' ) xdd





!***********************************************************************
!*
!***********************************************************************
subroutine type_element




interface operator (+)
module procedure element
end interface

character(len=20) :: name
real (kind=4) :: x
real (kind=8) :: xd
real (kind=16) :: xdd

real :: a, b, c;
!  real :: a1, a2, a3;


type (person) :: father = person( "juan"), mother = person("cris")

associate ( name1 => father%name, name2=>mother%name )
name = trim(name1) // trim(name2)
end associate

a = 1; b = 1 ; c = 1;
associate ( a1 =>a, a2 =>b, a3=>c )
a3 = a1 + a2
write(*,*) " a3 = ", a3
end associate
write(*,*) " c = ", c
!    write(*,*) " a3 = ", a3

!   write(*,*) " father =", father % name
write(*,*) " element =", father + mother
write(*,*) " name =", name

x   = 1.234567891234567891234567890Q00
xd  = 1.234567891234567891234567890Q00
xdd = 1.234567891234567891234567890Q00
write(*, '(ES)' ) x  ! scientific notation 1.234 (first digit should be greater or equal than on
write(*, '(E)' ) x   ! exponential normalized  notation 0.1234 (first digit is zero )

write(*,*) " Single, double and quadruple precision "
write(*, '(E)' ) x
```

```
write(*, '(E)' ) xd
write(*, '(E)' ) xdd

write(*, '(3(E, :, ","))' ) x, xd, xdd

end subroutine



!***************************************************************************
!*
!***************************************************************************
subroutine Hello_world


write(*,'(a)', advance='no') " Hello world..... "
write(*,'(a)', advance='no') " press enter"
read(*,*)

end subroutine



!***************************************************************************
!*
!***************************************************************************
subroutine  commandline

character(len=256) :: line,  enval
integer :: i, iarg, stat, clen, len
integer :: estat, cstat

iarg = command_argument_count()
write(*,*) " iarg = ", iarg
do i=1,iarg
call get_command_argument(i,line,clen,stat)
write (*,'(I0,A,A)') i,': ',line(1:clen)
end do
call get_command(line,clen,stat)
write (*,'(A)') line(1:clen)

call get_environment_variable('HOSTNAME',enval,len,stat)
if (stat == 0) write (*,'(A,A)') 'Host=', enval(1:len)
call get_environment_variable('USER',enval,len,stat)
if (stat == 0) write (*,'(A,A)') 'User=', enval(1:len)


! call execute_command_line('ls -al', .TRUE., estat, cstat)
call execute_command_line('dir', .TRUE., estat, cstat)
if (estat==0) write (*,'(A)') "Command completed successfully"

end subroutine

!***************************************************************************
!*
```

```
!***********************************************************************
subroutine  allocate_characteristics

real, allocatable :: V(:), A(:, :)
character(:), allocatable :: S
integer :: i, j, N

real, pointer ::  B(:,:), Diagonal(:)
real, pointer :: memory(:)

real :: x, y, z
class(*), pointer :: p1(:)


N = 10
V = [ ( i/real(N), i=1, N ) ]  ! automatic allocation allocate( V(N) )
write(*,* ) " V = ", V

N = 2
A = reshape( [ (  ( (i/real(N))**j ,i=1, N ), j=1, N ) ], [N, N] )
do i=1, N
write(*,* ) " A = ", A(i,:)
end do

N = 4
A = reshape( [ (  ( (i/real(N))**j ,i=1, N ), j=1, N ) ], [N, N] )
do i=1, N
write(*,'(A, 100f6.2)' ) " A = ", A(i,:)
end do

S = "Hello world"
write(*,*) " S = ", S, len(S)

S = "Hello"
write(*,*) " S = ", S, len(S)


allocate( memory(1:N*N) )
B(1:N,1:N) => memory
memory = 0
forall(i=1:N) B(i,i) = 10.
do i=1, N
write(*,'(A, *(f6.2) )' ) " B = ", B(i,:)
end do
diagonal => memory(::N+1)
write(*,'(A, 100f6.2)' ) " diagonal = ", diagonal
write(*,'(A, 100f6.2)' ) " trace = ", sum(diagonal)
write(*,'(A, 100f6.2)' ) " trace = ", sum(memory(::N+1))

x = 1; y = 2; z = 3;

write(*,'("i=",I0,", REALs=",*(G0,1X),"....")') i, x, y, z  ! C++ style


allocate( integer :: p1(5) ) ! p1 is an array of integers
```

```
select type (p1)
type is (integer)
p1 = [( i, i=1, size(p1) ) ]
write(*,'(" p1 = ", *(I0, 1x) )')  p1

class default
stop 'Error in type selection'
end select

deallocate(p1)
allocate( real :: p1(3) ) ! now p1 is an array of reals

select type (p1)
type is (real)
p1 = [( i, i=1, size(p1) )  ]
write(*,'(" p1 = ", *(G0, 1x) )')  p1

class default
stop 'Error in type selection'
end select



end subroutine
```

CHAPTER

THREE

FIRST EXAMPLES: CALCULUS AND ALGEBRA

3.1 My first program:Hello world

```
program TestSystemsofEquations


   use Linear_systems
   use Jacobian_module
   use Non_Linear_Systems


   implicit none
```

3.2 sum of a numeric series

Dar el resultado de la suma de los 100 primeros trminos de las siguientes series:

1. Serie de nmeros naturales.
2. Serie de nmeros naturales impares.
3. Serie numrica donde el trmino general de la serie es: $a_n = 1/n^2$ desde $n = 1$.
4. Serie numrica donde el trmino general de la serie es $a_n = 1/n!$ desde $n = 1$.
5. Serie numrica donde el trmino general de la serie es $a_n = (-1)^{n+1}/(2n-1)$ desde $n = 1$.

3.3 Operaciones con matrices y vectores

Considerar los vectores $V, W \in \mathbb{R}^N$ de componentes:

$$\{v_i = \frac{1}{i^2},\ \ i = 1 \dots N\},$$

$$\{w_i = \frac{(-1)^{i+1}}{2i-1},\ \ i = 1 \dots N\}.$$

Considerar la matriz $A \in \mathcal{M}_{N\times N}(\mathbb{R})$ donde su trmino genrico vale $a_{ij} = (i/N)^j$. Escribir un programa para calcular las operaciones siguientes con $N = 100$:

1. Suma de todas las componentes del vector V y del vector W.

2. Suma de todas las componentes de la matriz A.
3. Suma de las componentes del vector W mayores que cero.
4. Producto escalar de los vectores V y W.
5. Producto escalar del vector V y la columna N de la matriz A.
6. Suma de las componentes de vector que resulta de multiplicar la matriz A por el vector V.
7. Traza de la matriz A.

3.4 dynamic allocation of memory

Dada la matriz $A \in \mathcal{M}_{M\times M}(\mathbb{R})$ de trmino genrico

$$\{a_{ij} = (i/M)^j, \;\; i = 0,\dots M-1, \;\; j = 0,\dots M-1\}.$$

calcular las siguientes operaciones:

1. Calcular
$$\sum_{M=1}^{10} traza(A)$$
2. Calcular
$$\sum_{M=1}^{5} traza(A^2)$$
3. Calcular con $M = 4$
$$traza\left(\sum_{k=1}^{5} A^k\right)$$

3.5 Piecewise functions

Sean los vectores $X, F \in \mathbb{R}^{N+1}$. Las componentes de X almacenan los valores discretos del dominio de definicin y F las imgenes correspondientes de la funcin $F : \mathbb{R} \to \mathbb{R}$ continua a trozos siguiente:

$$F(x) = \begin{cases} 1, & a \le x \le -\dfrac{\pi}{2}, \\ \cos(\pi x), & -\dfrac{\pi}{2} < x < \dfrac{\pi}{2}, \\ 0, & \dfrac{\pi}{2} \le x \le b. \end{cases}$$

Considerar una particin equiespaciada de la forma:

$$\{x_i = a + i\Delta x, \;\; i = 0\dots N\}, \qquad \Delta x = \frac{b-a}{N}, \qquad a < -\frac{\pi}{2}, \qquad b > \frac{\pi}{2}.$$

Se pide calcular la suma;

$$S_N = \sum_{i=0}^{N} F_i \Delta x$$

1. con $N = 10$
2. con $N = 20$
3. con $N = 100$

3.6 Series of functions

Aproximar mediante un desarrollo en serie de potencias de la forma

$$f(x) = \sum_{k=0}^{M} a_k\, x^k, \qquad a_k = \frac{f^{(k)}(0)}{k!},$$

las funciones $F : \mathbb{R} \to \mathbb{R}$, siguientes:

1. $f(x) = e^x$ y calcular el valor $f(1)$ con $M = 5$.
2. $f(x) = \sin(x)$ y calcular el valor $f(\pi/2)$ con $M = 8$.
3. $f(x) = \cosh(x)$ y calcular el valor $f(1)$ con $M = 10$.
4. $f(x) = \dfrac{1}{1-x}$ y calcular el valor $f(0.9)$ con $M = 20$.
5. $f(x) = e^x$ y calcular el valor ms preciso de $f(1)$ con doble precisin.
6. $f(x) = \sin(x)$ y calcular el valor ms preciso $f(\pi/2)$ con doble precisin.
7. $f(x) = \cosh(x)$ y calcular el valor ms preciso de $f(1)$ con doble precisin.
8. $f(x) = \dfrac{1}{1-x}$ y calcular el valor ms preciso de $f(0.9)$ con doble precisin.

3.7 Reading and writing data files

Crear los ficheros de datos ForTran con nombres `input_1.dat` e `input_2.dat` con la información siguiente:

Contenido del fichero de entrada `input_1.dat` :

```
Datos de entrada 1

1.2      3.4     6.2     -14.0    0.1
-25.2    -8.6    5.1      9.9    17.0
-1.0     -2.0    -5.4     -8.6    0.0
3.14     -11.9   -7.0     -12.1   9.2
6.66     5.32    0.001   0.2      0.001
```

Contenido del fichero de entrada `input_2.dat` :

```
Datos de entrada 2

1.2      3.4     6.2     -14.0    0.1       4.89    7.54
-25.2    -8.6    5.1     12.0     9.9       12.24   17.0
0.0      34.5    -1.0    -2.0     -43.04    -8.6     0.0
3.14     -11.9   71.0    7.0     17.0       -12.1    9.2
6.66     5.32    0.001    0.2     0.001     0.008    -0.027
54.0     77.1    -9.002   -13.2   0.017     65.53    -0.021
23.04    -51.98  -34.2    9.99    5.34      8.87     3.22
```

Escribir un programa que gestione los datos de los ficheros anteriores siguiendo los pasos siguientes: Declarar las matrices $A \in \mathcal{M}_{N\times N}(\mathbb{R})$, $B \in \mathcal{M}_{N\times 3}(\mathbb{R})$, $C \in \mathcal{M}_{N\times 2}(\mathbb{R})$ y los vectores $U, V, W, T \in \mathbb{R}^N$. Leer el fichero de entrada (`input_1.dat` o `input_2.dat`) de la forma siguiente:

1. Cargar el fichero completo en la matriz A.
2. Cargar las cuatro primeras columnas del fichero en los vectores U, V , W y T.
3. Cargar la primera columna en el vector T y las tres últimas columnas en la matriz B.
4. Cargar la segunda columna en el vector U y las dos últimas columnas en la matriz C.
5. Cargar las columnas 1, 2 y 4 en la matriz B.

Además, el programa debe crear el fichero de salida (`output_1.dat` o `output_2.dat`), donde se irán escribiendo las matrices y vectores de los apartados anteriores. El formato de escritura debe ser el de números reales con cinco decimales.

Para el enunciado anterior, escribir los programas siguientes:

1. Programa 1 : Asignación estática de memoria.
   Ejecutar el programa por separado para los ficheros `input_1.dat` e `input_2.dat`. Para ello modificar las dimensiones y en nombre de los ficheros en el programa fuente.
2. Programa 2 : Asignación dinámica de memoria.
   Ejecutar el programa una única vez para gestionar los datos de los ficheros de entrada `input_1.dat` e `input_2.dat`.

3.8 Systems of linear equations

Implementar un mdulo para la resolucin de sistemas lineales de ecuaciones algebraicas. Los mtodos de resolucin propuestos son el de eliminacin Gaussiana, factorizacin LU, factorizacin LU de la biblioteca *Numerical Recipes* y Jacobi.

Para cada mtodo se pide:

- Validar los resultados con varios casos de prueba con dimensiones distintas.
- Evaluar tiempos de ejecucin.
- Comparar resultados con los mtodos restantes.

Aplicacin : Estudiar el condicionamiento de sistemas lineales de ecuaciones para matrices aleatorias y de Vandermonde.

3.9 Systems of nonlinear equations

Implementar un mdulo para la resolucin numrica de ecuaciones no lineales. Para funciones $F : \mathbb{R} \to \mathbb{R}$, los mtodos de resolucin propuestos son el de la biseccin y el de Newton-Raphson. Para funciones $F : \mathbb{R}^N \to \mathbb{R}^N$, se proponen el mtodo de Newton-Raphson con matriz Jacobiana analtica y el mtodo de Newton-Raphson con matriz Jacobiana numrica. Para la validacin de los mtodos propuestos, se pide implementar un mdulo con al menos tres funciones $F : \mathbb{R} \to \mathbb{R}$ y al menos tres funciones $F : \mathbb{R}^N \to \mathbb{R}^N$. Este mdulo debe contener las derivadas y matrices Jacobianas correspondientes de las funciones propuestas.

En el informe correspondiente, presentar tablas de soluciones numricas en cada paso de iteracin para las funciones de prueba propuestas.

3.10 Eigenvalues and eigenvectors

Implementar un mdulo para el clculo de autovalores y autovectores de una matriz. Los mtodos de resolucin propuestos son el mtodo de la potencia y el mtodo de la potencia inversa. Implementar el mtodo de la potencia inversa a partir de la matriz inversa y resolviendo el sistema lineal correspondiente.

Para cada mtodo se pide:

- Validar los resultados con varios casos de prueba con dimensiones distintas.
- Evaluar tiempos de ejecucin. Comparar tiempos de ejecucin del mtodo de la potencia inversa mediante los dos algoritmos propuestos : matriz inversa y solucin del sistema lineal.

Aplicacin : Estudiar el condicionamiento de sistemas lineales de ecuaciones para matrices aleatorias y de Vandermonde. Calcular la relacin $\lambda_{max}/\lambda_{min}$ de los casos de prueba presentados en el hito 1 y relacionar y discutir los resultados.

```
subroutine   power_method

integer :: i, j, k
integer, parameter :: PI = 4 * atan(1d0)
integer, parameter :: N = 20
real :: x(0:N), Vandermonde(0:N, 0:N), sigma
real :: a=-1, b=1
real V(0:N), V0(0:N)


x = [ ( a + (b-a)*i/N, i=0, N) ]

forall(i=0:N, j=0:N) Vandermonde(i,j) = x(i)**j

V = 1
V0 = 0
do while( abs(norm2(V)-norm2(V0)) > 1d-5 )
V0 = V
V = matmul( Vandermonde, V ) / norm2(V)
write(*,*) maxval(V)
end do
sigma = dot_product( V, V )
write(*,*) "sigma = ", sigma


end subroutine
```

3.11 Finite differences

1. Obtener las frmulas de las derivadas numricas primeras descentradas, con tres puntos equiespaciados a una distancia Δx.
2. A partir de la funcin $f(x) = e^x$ en el punto $x = 0$, representar grficamente el error total de las derivadas numricas frente al valor de Δx en precisin simple y doble. En particular, representar grficamente las derivadas primeras adelantada (definicin de derivada), centrada y descentradas y la derivada segunda, con tres puntos equiespaciados a una distancia Δx. Discutir los resultados obtenidos.

3. Resolver los problemas de contorno en ecuaciones diferenciales ordinarias siguientes:

   - **Problema 1:**
   $$u'' + u = 0, \quad x \in [-1,1], \qquad u(-1) = 1, \quad u(1) = 0,$$
   - **Problema 2:**
   $$u'' + u' - u = \sin(2\pi x), \quad x \in [-1,1], \qquad u(-1) = 0, \quad u'(1) = 0.$$

   Para los problemas citados anteriormente se pide:

   (a) A partir de las derivadas numricas con tres puntos equiespaciados escribir el sistema de ecuaciones resultante.

   (b) Obtener la solucin numrica mediante la resolucin de un sistema lineal de ecuaciones, con $N = 10$ y $N = 100$.

   (c) Representar grficamente los resultados obtenidos.

3.12 Numerical integration

Implementar un mdulo para la resolucin numrica de integrales definidas de funciones $F : \mathbb{R} \to \mathbb{R}$. Los mtodos de resolucin propuestos son las reglas del rectngulo, punto medio, trapecio y Simpson. Implementar un mdulo de funciones $F : \mathbb{R} \to \mathbb{R}$ de prueba para validar los mtodos numricos propuestos. Este mdulo debe contener al menos tres funciones con funciones primitivas conocidas y una funcin cuya funcin primitiva sea desconocida.

Evaluar el error de las soluciones numricas para cada mtodo propuesto y para distintos valores del incremento de la particin.

3.13 to be included

elemental advance = no dummy versus actual assumed shape explicit shape

global, local and scope in modules

1 versus 1.

tab instead of blanks

brackets

mask in intrinsic functions

! and &

;

camel case versus underscore

overloading

forall parallel

enter matrices by row or columns

lower bound: upper bound

array operations C = A + B

public versus private

encapsulamiento y ocultaci

FORALL (I=1:N-1, J=1:N, J¿I) A(I,J) = A(J,I)

```
x   = 1.2345678912345678912345678900Q0
xd  = 1.2345678912345678912345678900Q0
xdd = 1.2345678912345678912345678900Q0
write(*, '(ES)' ) x  ! scientific notation 1.234 (first digit should be greater or equal than on
write(*, '(E)' ) x   ! exponential normalized  notation 0.1234 (first digit is zero )

write(*,*) " Single, double and quadruple precision "
```

```
write(*, '(E)' ) x
write(*, '(E)' ) xd
write(*, '(E)' ) xdd





!*************************************************************************
!*
!*************************************************************************
subroutine type_element




interface operator (+)
module procedure element
end interface

character(len=20) :: name
real (kind=4) :: x
real (kind=8) :: xd
real (kind=16) :: xdd

real :: a, b, c;
!  real :: a1, a2, a3;


type (person) :: father = person( "juan"), mother = person("cris")

associate ( name1 => father%name, name2=>mother%name )
name = trim(name1) // trim(name2)
end associate

a = 1; b = 1 ; c = 1;
associate ( a1 =>a, a2 =>b, a3=>c )
a3 = a1 + a2
write(*,*) " a3 = ", a3
end associate
write(*,*) " c = ", c
!    write(*,*) " a3 = ", a3

!   write(*,*) " father =", father % name
write(*,*) " element =", father + mother
write(*,*) " name =", name

x   = 1.234567891234567891234567890Q00
xd  = 1.234567891234567891234567890Q00
xdd = 1.234567891234567891234567890Q00
write(*, '(ES)' ) x  ! scientific notation 1.234 (first digit should be greater or equal than o
write(*, '(E)' ) x   ! exponential normalized  notation 0.1234 (first digit is zero )

write(*,*) " Single, double and quadruple precision "
write(*, '(E)' ) x
```

```
write(*, '(E)' ) xd
write(*, '(E)' ) xdd

write(*, '(3(E, :, ","))' ) x, xd, xdd

end subroutine



!***************************************************************************
!*
!***************************************************************************
subroutine Hello_world


write(*,'(a)', advance='no') " Hello world..... "
write(*,'(a)', advance='no') " press enter"
read(*,*)

end subroutine



!***************************************************************************
!*
!***************************************************************************
subroutine  commandline

character(len=256) :: line,  enval
integer :: i, iarg, stat, clen, len
integer :: estat, cstat

iarg = command_argument_count()
write(*,*) " iarg = ", iarg
do i=1,iarg
call get_command_argument(i,line,clen,stat)
write (*,'(I0,A,A)') i,': ',line(1:clen)
end do
call get_command(line,clen,stat)
write (*,'(A)') line(1:clen)

call get_environment_variable('HOSTNAME',enval,len,stat)
if (stat == 0) write (*,'(A,A)') 'Host=', enval(1:len)
call get_environment_variable('USER',enval,len,stat)
if (stat == 0) write (*,'(A,A)') 'User=', enval(1:len)


! call execute_command_line('ls -al', .TRUE., estat, cstat)
call execute_command_line('dir', .TRUE., estat, cstat)
if (estat==0) write (*,'(A)') "Command completed successfully"

end subroutine

!***************************************************************************
!*
```

```
!*************************************************************************
subroutine  allocate_characteristics

real, allocatable :: V(:), A(:, :)
character(:), allocatable :: S
integer :: i, j, N

real, pointer ::  B(:,:), Diagonal(:)
real, pointer :: memory(:)

real :: x, y, z
class(*), pointer :: p1(:)


N = 10
V = [ ( i/real(N), i=1, N ) ]  ! automatic allocation allocate( V(N) )
write(*,* ) " V = ", V

N = 2
A = reshape( [ (  ( (i/real(N))**j ,i=1, N ), j=1, N ) ], [N, N] )
do i=1, N
write(*,* ) " A = ", A(i,:)
end do

N = 4
A = reshape( [ (  ( (i/real(N))**j ,i=1, N ), j=1, N ) ], [N, N] )
do i=1, N
write(*,'(A, 100f6.2)' ) " A = ", A(i,:)
end do

S = "Hello world"
write(*,*) " S = ", S, len(S)

S = "Hello"
write(*,*) " S = ", S, len(S)


allocate( memory(1:N*N) )
B(1:N,1:N) => memory
memory = 0
forall(i=1:N) B(i,i) = 10.
do i=1, N
write(*,'(A, *(f6.2) )' ) " B = ", B(i,:)
end do
diagonal => memory(::N+1)
write(*,'(A, 100f6.2)' ) " diagonal = ", diagonal
write(*,'(A, 100f6.2)' ) " trace = ", sum(diagonal)
write(*,'(A, 100f6.2)' ) " trace = ", sum(memory(::N+1))

x = 1; y = 2; z = 3;

write(*,'("i=",I0,", REALs=",*(G0,1X),"....")') i, x, y, z  ! C++ style


allocate( integer :: p1(5) ) ! p1 is an array of integers
```

```
select type (p1)
type is (integer)
p1 = [( i, i=1, size(p1) ) ]
write(*,'(" p1 = ", *(I0, 1x) )')  p1

class default
stop 'Error in type selection'
end select

deallocate(p1)
allocate( real :: p1(3) ) ! now p1 is an array of reals

select type (p1)
type is (real)
p1 = [( i, i=1, size(p1) )  ]
write(*,'(" p1 = ", *(G0, 1x) )')  p1

class default
stop 'Error in type selection'
end select



end subroutine
```

CHAPTER

FOUR

FIRST EXAMPLES: CALCULUS AND ALGEBRA

4.1 My first program:Hello world

```
program TestSystemsofEquations


    use Linear_systems
    use Jacobian_module
    use Non_Linear_Systems


    implicit none
```

4.2 sum of a numeric series

Dar el resultado de la suma de los 100 primeros trminos de las siguientes series:

1. Serie de nmeros naturales.
2. Serie de nmeros naturales impares.
3. Serie numrica donde el trmino general de la serie es: $a_n = 1/n^2$ desde $n = 1$.
4. Serie numrica donde el trmino general de la serie es $a_n = 1/n!$ desde $n = 1$.
5. Serie numrica donde el trmino general de la serie es $a_n = (-1)^{n+1}/(2n-1)$ desde $n = 1$.

4.3 Operaciones con matrices y vectores

Considerar los vectores $V, W \in \mathbb{R}^N$ de componentes:

$$\{v_i = \frac{1}{i^2}, \;\; i = 1 \dots N\},$$

$$\{w_i = \frac{(-1)^{i+1}}{2i-1}, \;\; i = 1 \dots N\}.$$

Considerar la matriz $A \in \mathcal{M}_{N\times N}(\mathbb{R})$ donde su trmino genrico vale $a_{ij} = (i/N)^j$. Escribir un programa para calcular las operaciones siguientes con $N = 100$:

1. Suma de todas las componentes del vector V y del vector W.

2. Suma de todas las componentes de la matriz A.
3. Suma de las componentes del vector W mayores que cero.
4. Producto escalar de los vectores V y W.
5. Producto escalar del vector V y la columna N de la matriz A.
6. Suma de las componentes de vector que resulta de multiplicar la matriz A por el vector V.
7. Traza de la matriz A.

4.4 dynamic allocation of memory

Dada la matriz $A \in \mathcal{M}_{M\times M}(\mathbb{R})$ de trmino genrico

$$\{a_{ij} = (i/M)^j, \;\; i = 0,\ldots M-1, \;\; j = 0,\ldots M-1\}.$$

calcular las siguientes operaciones:

1. Calcular
$$\sum_{M=1}^{10} traza(A)$$
2. Calcular
$$\sum_{M=1}^{5} traza(A^2)$$
3. Calcular con $M = 4$
$$traza\left(\sum_{k=1}^{5} A^k\right)$$

4.5 Piecewise functions

Sean los vectores $X, F \in \mathbb{R}^{N+1}$. Las componentes de X almacenan los valores discretos del dominio de definicin y F las imgenes correspondientes de la funcin $F : \mathbb{R} \to \mathbb{R}$ continua a trozos siguiente:

$$F(x) = \begin{cases} 1, & a \le x \le -\dfrac{\pi}{2}, \\ \cos(\pi x), & -\dfrac{\pi}{2} < x < \dfrac{\pi}{2}, \\ 0, & \dfrac{\pi}{2} \le x \le b. \end{cases}$$

Considerar una particin equiespaciada de la forma:

$$\{x_i = a + i\Delta x, \;\; i = 0\ldots N\}, \qquad \Delta x = \frac{b-a}{N}, \qquad a < -\frac{\pi}{2}, \qquad b > \frac{\pi}{2}.$$

Se pide calcular la suma;

$$S_N = \sum_{i=0}^{N} F_i \Delta x$$

1. con $N = 10$
2. con $N = 20$
3. con $N = 100$

4.6 Series of functions

Aproximar mediante un desarrollo en serie de potencias de la forma

$$f(x) = \sum_{k=0}^{M} a_k \, x^k, \qquad\qquad a_k = \frac{f^{(k)}(0)}{k!},$$

las funciones $F : \mathbb{R} \to \mathbb{R}$, siguientes:

1. $f(x) = e^x$ y calcular el valor $f(1)$ con $M = 5$.
2. $f(x) = \sin(x)$ y calcular el valor $f(\pi/2)$ con $M = 8$.
3. $f(x) = \cosh(x)$ y calcular el valor $f(1)$ con $M = 10$.
4. $f(x) = \dfrac{1}{1-x}$ y calcular el valor $f(0.9)$ con $M = 20$.
5. $f(x) = e^x$ y calcular el valor ms preciso de $f(1)$ con doble precisin.
6. $f(x) = \sin(x)$ y calcular el valor ms preciso $f(\pi/2)$ con doble precisin.
7. $f(x) = \cosh(x)$ y calcular el valor ms preciso de $f(1)$ con doble precisin.
8. $f(x) = \dfrac{1}{1-x}$ y calcular el valor ms preciso de $f(0.9)$ con doble precisin.

4.7 Reading and writing data files

Crear los ficheros de datos ForTran con nombres `input_1.dat` e `input_2.dat` con la información siguiente:

Contenido del fichero de entrada `input_1.dat` :

```
Datos de entrada 1

1.2      3.4     6.2     -14.0    0.1
-25.2    -8.6    5.1      9.9    17.0
-1.0     -2.0    -5.4     -8.6    0.0
3.14     -11.9   -7.0     -12.1   9.2
6.66     5.32    0.001   0.2      0.001
```

Contenido del fichero de entrada `input_2.dat` :

```
Datos de entrada 2

1.2      3.4     6.2     -14.0    0.1       4.89   7.54
-25.2    -8.6    5.1     12.0     9.9       12.24  17.0
0.0      34.5    -1.0    -2.0     -43.04    -8.6    0.0
3.14     -11.9   71.0    7.0     17.0       -12.1   9.2
6.66     5.32    0.001    0.2     0.001     0.008   -0.027
54.0     77.1    -9.002  -13.2    0.017     65.53   -0.021
23.04    -51.98  -34.2    9.99    5.34      8.87    3.22
```

Escribir un programa que gestione los datos de los ficheros anteriores siguiendo los pasos siguientes: Declarar las matrices $A \in \mathcal{M}_{N\times N}(\mathbb{R})$, $B \in \mathcal{M}_{N\times 3}(\mathbb{R})$, $C \in \mathcal{M}_{N\times 2}(\mathbb{R})$ y los vectores $U, V, W, T \in \mathbb{R}^N$. Leer el fichero de entrada (`input_1.dat` o `input_2.dat`) de la forma siguiente:

1. Cargar el fichero completo en la matriz A.
2. Cargar las cuatro primeras columnas del fichero en los vectores U, V , W y T.
3. Cargar la primera columna en el vector T y las tres últimas columnas en la matriz B.
4. Cargar la segunda columna en el vector U y las dos últimas columnas en la matriz C.
5. Cargar las columnas 1, 2 y 4 en la matriz B.

Además, el programa debe crear el fichero de salida (`output_1.dat` o `output_2.dat`), donde se irán escribiendo las matrices y vectores de los apartados anteriores. El formato de escritura debe ser el de números reales con cinco decimales.

Para el enunciado anterior, escribir los programas siguientes:

1. Programa 1 : Asignación estática de memoria.
   Ejecutar el programa por separado para los ficheros `input_1.dat` e `input_2.dat`. Para ello modificar las dimensiones y en nombre de los ficheros en el programa fuente.
2. Programa 2 : Asignación dinámica de memoria.
   Ejecutar el programa una única vez para gestionar los datos de los ficheros de entrada `input_1.dat` e `input_2.dat`.

4.8 Systems of linear equations

Implementar un mdulo para la resolucin de sistemas lineales de ecuaciones algebraicas. Los mtodos de resolucin propuestos son el de eliminacin Gaussiana, factorizacin LU, factorizacin LU de la biblioteca *Numerical Recipes* y Jacobi.

Para cada mtodo se pide:

- Validar los resultados con varios casos de prueba con dimensiones distintas.
- Evaluar tiempos de ejecucin.
- Comparar resultados con los mtodos restantes.

Aplicacin : Estudiar el condicionamiento de sistemas lineales de ecuaciones para matrices aleatorias y de Vandermonde.

4.9 Systems of nonlinear equations

Implementar un mdulo para la resolucin numrica de ecuaciones no lineales. Para funciones $F : \mathbb{R} \to \mathbb{R}$, los mtodos de resolucin propuestos son el de la biseccin y el de Newton-Raphson. Para funciones $F : \mathbb{R}^N \to \mathbb{R}^N$, se proponen el mtodo de Newton-Raphson con matriz Jacobiana analtica y el mtodo de Newton-Raphson con matriz Jacobiana numrica. Para la validacin de los mtodos propuestos, se pide implementar un mdulo con al menos tres funciones $F : \mathbb{R} \to \mathbb{R}$ y al menos tres funciones $F : \mathbb{R}^N \to \mathbb{R}^N$. Este mdulo debe contener las derivadas y matrices Jacobianas correspondientes de las funciones propuestas.

En el informe correspondiente, presentar tablas de soluciones numricas en cada paso de iteracin para las funciones de prueba propuestas.

4.10 Eigenvalues and eigenvectors

Implementar un mdulo para el clculo de autovalores y autovectores de una matriz. Los mtodos de resolucin propuestos son el mtodo de la potencia y el mtodo de la potencia inversa. Implementar el mtodo de la potencia inversa a partir de la matriz inversa y resolviendo el sistema lineal correspondiente.

Para cada mtodo se pide:

- Validar los resultados con varios casos de prueba con dimensiones distintas.
- Evaluar tiempos de ejecucin. Comparar tiempos de ejecucin del mtodo de la potencia inversa mediante los dos algoritmos propuestos : matriz inversa y solucin del sistema lineal.

Aplicacin : Estudiar el condicionamiento de sistemas lineales de ecuaciones para matrices aleatorias y de Vandermonde. Calcular la relacin $\lambda_{max}/\lambda_{min}$ de los casos de prueba presentados en el hito 1 y relacionar y discutir los resultados.

```
subroutine   power_method

integer :: i, j, k
integer, parameter :: PI = 4 * atan(1d0)
integer, parameter :: N = 20
real :: x(0:N), Vandermonde(0:N, 0:N), sigma
real :: a=-1, b=1
real V(0:N), V0(0:N)


x = [ ( a + (b-a)*i/N, i=0, N) ]

forall(i=0:N, j=0:N) Vandermonde(i,j) = x(i)**j

V = 1
V0 = 0
do while( abs(norm2(V)-norm2(V0)) > 1d-5 )
V0 = V
V = matmul( Vandermonde, V ) / norm2(V)
write(*,*) maxval(V)
end do
sigma = dot_product( V, V )
write(*,*) "sigma = ", sigma


end subroutine
```

4.11 Finite differences

1. Obtener las frmulas de las derivadas numricas primeras descentradas, con tres puntos equiespaciados a una distancia Δx.
2. A partir de la funcin $f(x) = e^x$ en el punto $x = 0$, representar grficamente el error total de las derivadas numricas frente al valor de Δx en precisin simple y doble. En particular, representar grficamente las derivadas primeras adelantada (definicin de derivada), centrada y descentradas y la derivada segunda, con tres puntos equiespaciados a una distancia Δx. Discutir los resultados obtenidos.

3. Resolver los problemas de contorno en ecuaciones diferenciales ordinarias siguientes:

   - **Problema 1:**
   $$u'' + u = 0, \quad x \in [-1,1], \qquad u(-1) = 1, \quad u(1) = 0,$$
   - **Problema 2:**
   $$u'' + u' - u = \sin(2\pi x), \quad x \in [-1,1], \qquad u(-1) = 0, \quad u'(1) = 0.$$

   Para los problemas citados anteriormente se pide:

   (a) A partir de las derivadas numricas con tres puntos equiespaciados escribir el sistema de ecuaciones resultante.

   (b) Obtener la solucin numrica mediante la resolucin de un sistema lineal de ecuaciones, con $N = 10$ y $N = 100$.

   (c) Representar grficamente los resultados obtenidos.

4.12 Numerical integration

Implementar un mdulo para la resolucin numrica de integrales definidas de funciones $F : \mathbb{R} \to \mathbb{R}$. Los mtodos de resolucin propuestos son las reglas del rectngulo, punto medio, trapecio y Simpson. Implementar un mdulo de funciones $F : \mathbb{R} \to \mathbb{R}$ de prueba para validar los mtodos numricos propuestos. Este mdulo debe contener al menos tres funciones con funciones primitivas conocidas y una funcin cuya funcin primitiva sea desconocida.

Evaluar el error de las soluciones numricas para cada mtodo propuesto y para distintos valores del incremento de la particin.

4.13 to be included

elemental advance = no dummy versus actual assumed shape explicit shape

global, local and scope in modules

1 versus 1.

tab instead of blanks

brackets

mask in intrinsic functions

! and &

;

camel case versus underscore

overloading

forall parallel

enter matrices by row or columns

lower bound: upper bound

array operations C = A + B

public versus private

encapsulamiento y ocultaci

FORALL (I=1:N-1, J=1:N, J¿I) A(I,J) = A(J,I)

```
x   = 1.234567891234567891234567890Q00
xd  = 1.234567891234567891234567890Q00
xdd = 1.234567891234567891234567890Q00
write(*, '(ES)' ) x  ! scientific notation 1.234 (first digit should be greater or equal than o
write(*, '(E)' ) x   ! exponential normalized  notation 0.1234 (first digit is zero )

write(*,*) " Single, double and quadruple precision "
```

```
write(*, '(E)' ) x
write(*, '(E)' ) xd
write(*, '(E)' ) xdd





!**************************************************************************
!*
!**************************************************************************
subroutine type_element




interface operator (+)
module procedure element
end interface

character(len=20) :: name
real (kind=4) :: x
real (kind=8) :: xd
real (kind=16) :: xdd

real :: a, b, c;
!  real :: a1, a2, a3;


type (person) :: father = person( "juan"), mother = person("cris")

associate ( name1 => father%name, name2=>mother%name )
name = trim(name1) // trim(name2)
end associate

a = 1; b = 1 ; c = 1;
associate ( a1 =>a, a2 =>b, a3=>c )
a3 = a1 + a2
write(*,*) " a3 = ", a3
end associate
write(*,*) " c = ", c
!    write(*,*) " a3 = ", a3

!   write(*,*) " father =", father % name
write(*,*) " element =", father + mother
write(*,*) " name =", name

x   = 1.234567891234567891234567890Q00
xd  = 1.234567891234567891234567890Q00
xdd = 1.234567891234567891234567890Q00
write(*, '(ES)' ) x  ! scientific notation 1.234 (first digit should be greater or equal than on
write(*, '(E)' ) x   ! exponential normalized  notation 0.1234 (first digit is zero )

write(*,*) " Single, double and quadruple precision "
write(*, '(E)' ) x
```

```
write(*, '(E)' ) xd
write(*, '(E)' ) xdd

write(*, '(3(E, :, ","))' ) x, xd, xdd

end subroutine



!***************************************************************************
!*
!***************************************************************************
subroutine Hello_world


write(*,'(a)', advance='no') " Hello world..... "
write(*,'(a)', advance='no') " press enter"
read(*,*)

end subroutine



!***************************************************************************
!*
!***************************************************************************
subroutine  commandline

character(len=256) :: line,  enval
integer :: i, iarg, stat, clen, len
integer :: estat, cstat

iarg = command_argument_count()
write(*,*) " iarg = ", iarg
do i=1,iarg
call get_command_argument(i,line,clen,stat)
write (*,'(I0,A,A)') i,': ',line(1:clen)
end do
call get_command(line,clen,stat)
write (*,'(A)') line(1:clen)

call get_environment_variable('HOSTNAME',enval,len,stat)
if (stat == 0) write (*,'(A,A)') 'Host=', enval(1:len)
call get_environment_variable('USER',enval,len,stat)
if (stat == 0) write (*,'(A,A)') 'User=', enval(1:len)


! call execute_command_line('ls -al', .TRUE., estat, cstat)
call execute_command_line('dir', .TRUE., estat, cstat)
if (estat==0) write (*,'(A)') "Command completed successfully"

end subroutine

!***************************************************************************
!*
```

```
!*****************************************************************************
subroutine  allocate_characteristics

real, allocatable :: V(:), A(:, :)
character(:), allocatable :: S
integer :: i, j, N

real, pointer ::  B(:,:), Diagonal(:)
real, pointer :: memory(:)

real :: x, y, z
class(*), pointer :: p1(:)


N = 10
V = [ ( i/real(N), i=1, N ) ]  ! automatic allocation allocate( V(N) )
write(*,* ) " V = ", V

N = 2
A = reshape( [ (  ( (i/real(N))**j ,i=1, N ), j=1, N ) ], [N, N] )
do i=1, N
write(*,* ) " A = ", A(i,:)
end do

N = 4
A = reshape( [ (  ( (i/real(N))**j ,i=1, N ), j=1, N ) ], [N, N] )
do i=1, N
write(*,'(A, 100f6.2)' ) " A = ", A(i,:)
end do

S = "Hello world"
write(*,*) " S = ", S, len(S)

S = "Hello"
write(*,*) " S = ", S, len(S)


allocate( memory(1:N*N) )
B(1:N,1:N) => memory
memory = 0
forall(i=1:N) B(i,i) = 10.
do i=1, N
write(*,'(A, *(f6.2) )' ) " B = ", B(i,:)
end do
diagonal => memory(::N+1)
write(*,'(A, 100f6.2)' ) " diagonal = ", diagonal
write(*,'(A, 100f6.2)' ) " trace = ", sum(diagonal)
write(*,'(A, 100f6.2)' ) " trace = ", sum(memory(::N+1))

x = 1; y = 2; z = 3;

write(*,'("i=",I0,", REALs=",*(G0,1X),"....")') i, x, y, z  ! C++ style


allocate( integer :: p1(5) ) ! p1 is an array of integers
```

```
select type (p1)
type is (integer)
p1 = [( i, i=1, size(p1) ) ]
write(*,'(" p1 = ", *(I0, 1x) )')  p1

class default
stop 'Error in type selection'
end select

deallocate(p1)
allocate( real :: p1(3) ) ! now p1 is an array of reals

select type (p1)
type is (real)
p1 = [( i, i=1, size(p1) )  ]
write(*,'(" p1 = ", *(G0, 1x) )')  p1

class default
stop 'Error in type selection'
end select



end subroutine
```

Part II

Developer guidelines

Part III

Application Program Interface (API Manual)