# ANALISIS NUMERICO

**Luis Alvarez León y Javier Sánchez Pérez**
Departamento de Informática y Sistemas
Universidad de Las Palmas
Campus de Tafira
35017 Las Palmas, España
Tfl: 45.87.10
Email: {lalvarez/jsanchez}@dis.ulpgc.es

**Contents**

## INTRODUCCION.

El presente documento es un texto de referencia básico sobre los contenidos de la disciplina Análisis Numérico en el contexto curricular de una Ingeniería en Informática. Aunque el texto cubre los contenidos mínimos necesarios, resultará de gran interés para los alumnos complementar la información aquí suministrada con los textos de referencia básicos mencionados en la bibliografía. Muchas de las demostraciones de los resultados presentados se encuentran en este texto, en los casos en que las demostraciones no se incluyen, se suministra el libro y la página donde se encuentra tal demostración, para que el alumno interesado pueda estudiarla por su cuenta. En general, todos los temas presentados aparecen bien desarrollados en los libros de texto clásicos mencionados en la bibliografía. La única excepción es el tema de aritméticas de precisión finita, que se ha desarrollado en este texto con algo más de detalle, y con un enfoque algo más moderno, que en los libros clásicos, por considerar, que en el contexto de una ingeniería de informática, este tema es de especial relevancia.

El lenguaje de programación que se utilizará es el fortran. Se ha elegido este lenguaje por ser la plataforma donde se han desarrollado habitualmente los grandes programas de cálculo numérico, y por estar especialmente orientado al cálculo científico. En el texto se va introduciendo este lenguaje de programación a través de programas ejemplos. Estos programas ejemplos se encuentran a disposición de los alumnos en el directorio de la asignatura */users/asignaturas/ii-an* de la máquina *serdis.dis.ulpgc.es* También se encuentra a disposición de los alumnos el fichero *an.h*, donde se encuentran todas las subrutinas definidas en estos programas ejemplos.

En el texto se proponen unas prácticas de laboratorio para realizar a lo largo de la asignatura. Para establecer el orden de impartición de los contenidos presentes en este documento se ha utilizado como criterio preferente, la coordinación entre el programa de prácticas y el programa teórico de la asignatura. De tal forma que con un desarrollo normal de la docencia, los contenidos teóricos sean presentados con antelación al desarrollo de las prácticas, comenzando las prácticas de laboratorio a partir de la segunda semana de clase.

Para el buen seguimiento de la asignatura, resulta de gran interés, tener cierta soltura en el manejo de los conceptos elementales del análisis matemático, álgebra, y en la programación de algoritmos.

La materia expuesta en esta documentación, está programada para ser impartida en un cuatrimestre a razón de 3 horas/semana en el aula y 2 horas/semana en el laboratorio informático, lo que hace un total de aproximadamente 45 horas en aula (3 créditos teóricos) y 30 horas de laboratorio (2 créditos prácticos). Dado el escaso tiempo disponible se han eliminado algunos temas clásicos de un curso completo anual de Análisis Numérico como son las ecuaciones diferenciales ordinarias y las ecuaciones en derivadas parciales. Normalmente, dichos temas se verán en detalle en asignaturas posteriores. Además, en lugar de presentar de forma exhaustiva todos los métodos numéricos que se pueden encontrar en los libros de Análisis Numérico clásicos, se ha optado por reducir los contenidos e impartir una selección de los métodos numéricos más representativos.

## ARITMETICAS DE PRECISION FINITA Y FUENTES DE ERRORES NUMERICOS.

### Aritméticas de precisión finita.

Un número entero $z$ se representa en el ordenador a través de un número fijo de bits (16 bits habitualmente), donde uno de los bits se utiliza para determinar el signo, y los restantes para expresar el valor absoluto del número, de tal manera que la secuencia de bits

$$a_1 a_2 a_3 ..... a_n$$

donde $a_i = 0$ o $a_i = 1$ representa el valor absoluto del número

$$\mid z \mid = a_1 + a_2 2 + a_3 2^2 + ... + a_n 2^{n-1}$$

de tal manera que utilizando 16 bits el mayor número entero que podemos representar es

$$1 + 2 + 2^2 + ..... + 2^{14} = 2^{15} - 1 = 32767$$

es decir, que los número enteros que podemos expresar con una aritmética de 16 bits van desde $-32767$ hasta $32767$.

Para representar un número real $y$ en el ordenador nos basaremos en el siguiente resultado

**Teorema 1** *un número real positivo $y$ se puede expresar como*

$$y = 2^e \sum_{n=1}^{\infty} \frac{a_n}{2^n}$$

*donde $e$ es un número entero, $a_1 = 1$, y para $n > 1$, $a_n = 0$ ó $a_n = 1$.*

**Demostración.** Dado un número real positivo $y$, existe un entero $e$ tal que $2^{e-1} \leq y < 2^e$, y por tanto $2^{-1} \leq y2^{-e} < 1$. Por otro lado si definimos las sucesiones $S_n$ y $a_n$ de la siguiente forma: $S_1 = a_1 = 2^{-1}$, y para $n > 1$

$$\begin{array}{ll} a_n = 0 & si \quad S_{n-1} + \frac{1}{2^n} > y2^{-e} \\ a_n = 1 & si \quad S_{n-1} + \frac{1}{2^n} \leq y2^{-e} \end{array}$$

$$S_n = \sum_{k=1}^{n} \frac{a_k}{2^k}$$

entonces es claro que $\mid S_n - y2^{-e} \mid \leq \frac{1}{2^n}$, y por tanto $S_n \rightarrow y2^{-e}$ lo que concluye la demostración del teorema.

**Ejemplo 1** *Consideremos $y = 10.125$, podemos expresar este número como:*

$$10.125 = 2^4(\frac{1}{2} + \frac{1}{2^3} + \frac{1}{2^7})$$

*es decir $e = 4$, $a_1 = a_3 = a_7 = 1$, y el resto de los $a_n$ es 0. En este caso el número de elementos $a_n$ distintos de 0 es un número finito, en general ello no ocurre así.*

Evidentemente cualquier número que tenga un número finito de elementos $a_n$ distintos de 0 es un número racional y por tanto los números irracionales se representarán siempre con un número infinito de elementos $a_n$ no nulos. Sin embargo, como muestra el siguiente problema, existen además otros muchos números además de los irracionales, que no se pueden representar con un número finito de elementos $a_n$ no-nulos.

**Problema 1** ***(2 puntos)*** *Demostrar que al representar el número real 0.1 como*

$$0.1 = 2^e \sum_{n=1}^{\infty} \frac{a_n}{2^n}$$

*el número de elementos no-nulos $a_n$ es infinito.*

**Problema 2** ***(2 puntos)*** *Representar el número 0.0 703 125 como*

$$0.0\,703\,125 = 2^e \sum_{n=1}^{\infty} \frac{a_n}{2^n}$$

Para definir una aritmética de precisión finita de número reales, lo que se hace habitualmente es discretizar la fórmula de representación anterior, tomando un número finito de valores posibles $a_i$ y tomando un número finito de valores para la exponencial $e$. Como puede observarse, cada valor $a_i$ viene representado por un bit, además puesto que el valor $a_1$ es siempre igual a 1, no es necesario almacenar su valor en memoria al guardar un número real.

Por tanto, en una aritmética de precisión finita los números reales distintos de cero se representan como

$$\widetilde{y} = \pm 2^e \sum_{n=1}^{t} \frac{a_n}{2^n}$$

donde $e$ varía entre dos valores limites $e_{\min} \leq e \leq e_{\max}$. Al valor $t$ se le llama precisión de la aritmética. A la secuencia $a_1a_2a_3.....a_t$, (donde $a_i = 0, 1$) se le denomina mantisa. Hay que hacer notar aquí, que dado que hemos impuesto siempre que $a_1 = 1$, el número 0 debemos añadirlo a la aritmética, dado que 0 no se puede representar de la forma anterior.

**Problema 3** ***(1 punto)*** *Calcular los valores positivos mínimo y máximo que puede tomar un número real en una aritmética de precisión finita en función de $t$, $e_{\min}$ y $e_{\max}$.*

**Problema 4** ***(2 puntos)*** *Calcular todos los números reales que se pueden construir tomando 5 bits de la forma siguiente: 1 bit para el signo, 2 bits para la mantisa (es decir $t = 3$, puesto que $a_1 = 1$ sólo se almacenan $a_2$ y $a_3$, y 2 bits para el exponente $e$, tomando como rango de $e = -1, 0, 1, 2$. Representar dichos números sobre una recta.*

Es importante resaltar que los números reales en una aritmética de precisión finita, no están equiespaciados, es decir los números están más cercanos entre sí cerca de 0, y más alejados al alejarnos de 0.

En 1985, la sociedad I.E.E.E. presentó una serie de especificaciones standard para la definición de una aritmética de precisión finita para los números reales. En este trabajo se codifica un número real en simple precisión utilizando 32 bits de memoria, de los cuales 23 bits se utilizan para la mantisa (es decir $t = 24$ puesto que $a_1 = 1$ no se almacena), 1 bit se utiliza para el signo, y 8 bits se utilizan para el exponente $e$, lo cual da un rango de $2^8 = 256$ valores posibles para el exponente $e$. En este caso se toma $e_{\min} = -125$ y $e_{\max} = 128$. Como puede observarse el número total de exponentes posibles es 254, dos menos que los 256 posibles, ello se hace así, porque se reservan dos posiciones de memoria para tratar las denominadas excepciones como se verá más adelante.

Por tanto el valor máximo que puede tomar un número real en esta aritmética es

$$\widetilde{y}_{\max} = 2^{128} \sum_{n=1}^{24} \frac{1}{2^n} = 3.4 \times 10^{38}$$

y el valor mínimo positivo es

$$\widetilde{y}_{\min} = 2^{-125} \frac{1}{2} = 1.18 \times 10^{-38}$$

Además el número de combinaciones posibles que puede tener la mantisa es del orden de $2^{24} = 1.68 \times 10^7$. Es decir que la aritmética tiene una precisión de 7 dígitos.

También se define en este trabajo de I.E.E.E. un standard para una aritmética en doble precisión. En este caso, se utilizan 64 bits para almacenar el número real, de los cuales 52 bits se utilizan para la mantisa ($t = 53$), 1 bit para el signo, y 11 bits para el exponente, lo que da lugar a $2^{11} = 2048$ posibilidades de elección de exponente $e$. En este caso se toma $e_{\min} = -1021$ y $e_{\max} = 1024$.

Por tanto el valor máximo que puede tomar un número real en esta aritmética es

$$\widetilde{y}_{\max} = 2^{1024} \sum_{n=1}^{53} \frac{1}{2^n} = 1.78 \times 10^{308}$$

y el valor mínimo positivo es

$$\widetilde{y}_{\min} = 2^{-1021} \frac{1}{2} = 2.23 \times 10^{-308}$$

Además el número de combinaciones posibles que puede tener la mantisa es del orden de $2^{53} = 9.0 \times 10^{15}$. Es decir que la aritmética tiene una precisión de 15 dígitos.

Evidentemente estos standards no se siguen al pie de la letra por los diferentes fabricantes de ordenadores, así pues, aunque existe bastante homogeneidad en este sentido, los valores pueden cambiar ligeramente de una máquina a otra. Por otro lado, aunque en la mayoría de los ordenadores actuales la base de representación de los números es 2, todavía pueden encontrarse algunos sistemas donde la base es 10 ó 16. Nosotros no entraremos aquí a estudiar este tipo de bases, simplemente decir que el estudio es básicamente el mismo, adaptándolo a la base de representación.

**Tratamiento de las excepciones en el Standard de IEEE.**

Denominaremos excepciones a las expresiones que no se pueden expresar en una aritmética usual, como son el 0, el infinito, operaciones no válidas como $\sqrt{-1}$, etc. Estas excepciones son tratadas en el standard de IEEE de la siguiente forma: dentro de las posiciones de memoria dedicadas al exponente $e$ de un número se reservan dos que corresponden a $e_{\min} - 1$ y $e_{\max} + 1$ que se utilizan para trabajar con con las excepciones. La regla que se utiliza es la siguiente:

1. Si el valor de una variable $y$ tiene por exponente $e_{\max} + 1$ y todos los coeficientes de la mantisa valen 0, entonces $y$ se considera infinito. Por ejemplo $1/0$ debe dar infinito.
2. Si el valor de una variable $y$ tiene por exponente $e_{\max} + 1$ y algún coeficiente de la mantisa es distinto de 0, entonces $y$ se considera que no es un número (NaN (Not a Number)). Por ejemplo $\sqrt{-1}$ debe dar NaN.
3. Si el valor de una variable $y$ tiene por exponente $e_{\min} - 1$ y todos los coeficientes de la mantisa valen 0, entonces $y$ se considera igual a 0.
4. Si el valor de una variable $y$ tiene por exponente $e_{\min} - 1$ y algún coeficiente de la mantisa es distinto de 0, $y$ se considera que no está normalizado (es decir $a_1 = 0$) y el valor de $y$ sería

$$y = 2^{e_{\min}-1} \sum_{n=2}^{t} \frac{a_n}{2^n}$$

**Programa 1** *Programa en fortran 77 para calcular el número positivo más pequeño de una aritmética. El programa devuelve un entero $M$ tal que $2^{-M}$ es el número real positivo más pequeño.*

```
      A=1.
      M=0
1     A=A/2.
      IF(A.GT.0) THEN
          M=M+1
          GOTO 1
      ENDIF
      PRINT *,M
      END
```

**Nota** En fortran 77 no es necesario declarar las variables. Por defecto, las variables cuyo nombre empieza por las letras

$$I, J, K, L, M, N$$

son variables enteras, y el resto son variables reales. Las cinco primeras columnas de cada línea de un programa fortran 77 están reservadas para escribir un número de etiqueta.

La columna 6 está reservada para indicar si una línea es continuación de la anterior (en este caso basta con escribir un carácter en la columna 6.)

Un $*$ o una $C$ en la primera columna indica que la línea es de comentario.

En el fortran 77 standard no existe la instrucción *WHILE*. Sin embargo, como se muestra en el programa anterior se puede simular fácilmente un while con un *GOTO*.

Los operadores de comparación en fortran 77 son: *.GT.*, *.GE.*, *.EQ.*, *.NE.*, *.LT.* y *.LT.* Los operadores lógicos son *.AND.* y *.OR.*

**Programa 2** *Programa en fortran 77 para calcular el número positivo más grande de una aritmética. El programa devuelve un entero $M$ tal que $2^M$ es el número real positivo más grande.*

```
      A=1.
      B=1.
      M=0
1     B=2.*A
      IF(B.GT.A) THEN
            A=B
            M=M+1
      GOTO 1
      ENDIF
      PRINT *,M
      END
```

**Problema 5** ***(2 puntos)*** *Dada una aritmética de precisión finita cualquiera, calcular la distancia que hay entre el número 1 y su inmediato superior (es decir el número que va después de 1), y la distancia entre el número 1, y su inmediato inferior.*

Vamos a denotar por $A$, al conjunto de valores reales a los que da lugar una aritmética de precisión finita, es decir

$$A = \left\{ \pm 2^e \sum_{n=1}^{t} \frac{a_n}{2^n} \right\} \cup \{0\}$$

Dado un número real cualquiera $y$ al representarlo en una aritmética de precisión finita se produce un error de redondeo, denotaremos por $\widetilde{y} \in A$ al número real que mejor aproxima a $y$ dentro de $A$.

**Definición 1** *Dada una aritmética de precisión finita, se define la unidad de redondeo $u$, como*

$$u = 2^{-t}$$

Por ejemplo si $t = 24$ (reales en simple precisión) $u = 2^{-24} = 5.97 \times 10^{-8}$, y en doble precisión ($t = 53$), $u = 2^{-53} = 1.1 \times 10^{-16}$.

**Programa 3** *Programa en fortran 77 para calcular la unidad de redondeo de una aritmética. El programa devuelve un entero $M$ tal que $u = 2^{-M}$*

```
      A=1.
      M=1
1     A=A*2.
      IF((1.+1./A).GT.1) THEN
            M=M+1
      GOTO 1
      ENDIF
      PRINT *,M
      END
```

**Práctica 1 (Aritméticas finitas, 2 horas).**

La línea de comando para la compilación de un programa en fortran 77 tiene la forma:

$$> \ f77 \ \ prog1.f \ \ -o \ \ prog1$$

donde se compila el programa $prog1.f$ y se genera el ejecutable $prog1$

Para compilar un programa en fortran, necesitamos una máquina que tenga instalado el compilador. Por ejemplo la máquina *serdis.dis.ulpgc.es* tiene dicho compilador. Para utilizar el compilador desde un entorno windows, basta con conectarse a través de la utilidad ssh a *serdis.dis.ulpgc.es* y trabajar directamente sobre la terminal de conexión. Los ficheros se pueden editar y corregir en entorno windows.

Compilar y ejecutar los programas 1, 2 y 3 para comprobar cual es el número positivo más pequeño, el más grande, y la unidad de redondeo del ordenador en precisión simple. Dichos programas se encuentran en el directorio de la asignatura */users/asignaturas/ii-an* de la máquina *serdis.dis.ulpgc.es*

Si ponemos en la cabecera del programa la instrucción

$$IMPLICIT \quad DOUBLE \quad PRECISION(D)$$

cualquier variable cuyo nombre empiece por $D$ será un número real en doble precisión. Hacer las modificaciones pertinentes en los programas 1, 2 y 3, para comprobar cual es el número positivo más pequeño, el más grande, y la unidad de redondeo del ordenador en doble precisión. Además hacer operaciones del tipo $1/0$, $1/\infty$, $\infty/0$ $\infty/\infty$, $\sqrt{-1}$ e imprimir los resultados para ver como trata las excepciones el FORTRAN en la arquitectura de los ordenadores del laboratorio.

**Problema 6** ***(4 puntos)*** *Se considera una aritmética de 16 bits donde se dedican 1 bit al signo, 9 bits a la mantisa ($t = 10$) y 6 bits al exponente ( $e_{\min} = -30$ $e_{\max} = 31$). Escribir, si es posible, los siguientes números en esta aritmética:*

1. 2, y los números más cercanos a 2 por arriba y por debajo.

2. El cero, el infinito y Na.

3. Los números positivos más grande y más pequeño de la aritmética (teniendo en cuenta las excepciones)

4. $\frac{1}{9}$.

5. $2\left(\frac{1}{2}-\frac{1}{2^{10}}\right)$

**Problema 7** ***(2 puntos)*** *Sean* $A = 2\left(\frac{1}{2}+\frac{1}{2^3}+\frac{1}{2^5}\right)$ $B = 2^3\left(\frac{1}{2}+\frac{1}{2^6}+\frac{1}{2^7}\right)$. *Calcular* $B+A$ *y* $B-A$

**Problema 8** ***(2 puntos)*** *Sean* $e_{\min}$, $e_{\max}$, *los valores mínimo y máximo del exponente* $e$. *Demostrar que si* $e_{\min} < e < e_{\max}$, *entonces los números:*

$$2^e\left(\sum_{n=1}^{t}\frac{a_n}{2^n}\pm\frac{1}{2^t}\right)$$

*pertenecen al conjunto* $A$ *de números reales generados por la aritmética de precisión finita.*

A continuación, mostraremos un resultado que indica el error de redondeo máximo que se produce al aproximar un número real cualquiera en una aritmética de precisión finita.

**Teorema 2** *Sean* $\widetilde{y}_{\min}$ , $\widetilde{y}_{\max}$ *los valores positivos más grande y más pequeño de una aritmética de precisión finita. Si un número real* $z$ *verifica* $\widetilde{y}_{\min} <\mid z \mid< \widetilde{y}_{\max}$ *entonces*

$$\mid z-\widetilde{z}\mid\leq\mid z\mid u$$

**Demostración.** un número real cualquiera $z$, que tomaremos positivo sin pérdida de generalidad, se puede expresar como:

$$z = 2^e\sum_{n=1}^{\infty}\frac{a_n}{2^n}$$

donde $a_0 = 1$ y en general $a_n = 0$ o $a_n = 1$, además para un número natural $t$ cualquiera se tiene:

$$2^e\sum_{n=1}^{t}\frac{a_n}{2^n}\leq z\leq 2^e\left(\sum_{n=1}^{t}\frac{a_n}{2^n}+\frac{1}{2^t}\right)$$

además, por el problema anterior, el número que está a la derecha de la desigualdad también pertenece a la aritmética de precisión finita, y por tanto:

$$\mid z-\widetilde{z}\mid\leq\frac{2^e2^{-t}}{2}$$

ahora bien, como $a_0 = 1$ se tiene que $2^e < 2 \mid z \mid$ y por tanto

$$\mid z-\widetilde{z}\mid\leq\mid z\mid 2^{-t}=\mid z\mid u$$

con lo que queda demostrado el teorema.

**Problema 9** ***(2 puntos)*** *Dado un número* $\widetilde{z} = 2^e\sum_{n=1}^{t}\frac{a_n}{2^n}$, *en una aritmética de precisión finita. Calcular el número inmediatamente inferior a él en dicha aritmética.*

Un resultado importante para la comparación de dos números es el siguiente:

**Teorema 3** *Si* $\widetilde{z}_1, \widetilde{z}_2 \in A$ *son distintos entonces*

$$\mid \widetilde{z}_1-\widetilde{z}_2\mid\geq\max\{\mid\widetilde{z}_1\mid,\mid\widetilde{z}_2\mid\}u$$

**Demostración:** Ejercicio

En muchos algoritmos, el test de parada incluye el hecho de que dos variables estén próximas entre sí. para ello se fija un umbral o tolerancia $TOL$ que por supuesto será mayor que la unidad de redondeo $u$ y expresaremos que las variables $A$ y $B$ están cercanas entre sí con una tolerancia $TOL$ si se cumple

$$\mid A-B\mid\leq\max\{\mid A\mid,\mid B\mid\}TOL$$

este criterio es simétrico en el sentido de que trata de igual modo los números $A$ y $B$. También se puede utilizar un criterio más simple como

$$\mid A-B\mid\leq\mid A\mid TOL$$

pero en este caso le estamos dando una significación especial a $A$ con respecto a $B$.

Estos criterios de comparación de números funcionan bien salvo cuando los números $A$ y $B$ están muy próximos a 0. Por ejemplo si $B = 0$, los criterios anteriores quedan

$$\mid A\mid\leq\mid A\mid TOL$$

lo cual es imposible (Si $TOL < 1$) salvo que $A$ también sea 0.Para evitar este comportamiento se puede añadir al criterio un valor $\epsilon > 0$ de la siguiente forma:

$$\mid A-B\mid\leq(\max\{\mid A\mid,\mid B\mid\}+\epsilon)TOL$$

**Programa 4** *Programa en fortran 77 que determina si dos variables* $A, B$ *son iguales con una tolerancia TOL (tomando el máximo de* $A, B$*), y se toma* $\epsilon = 10^{-10}$

```
READ *,A,B,TOL
IF(IGUAL(A,B,TOL).EQ.0) THEN
      PRINT *,'A=B segun la tolerancia TOL'
      STOP
ELSE
      PRINT *,'A distinto de B segun la tolerancia
TOL'
      STOP
ENDIF
END

FUNCTION IGUAL(A,B,TOL)
IF(ABS(A).GT.ABS(B)) THEN
```

```
      IF(ABS(A-B).LE.(TOL*(ABS(A)+10.**(-
10.))) THEN
          IGUAL=0
          RETURN
      ELSE
          IGUAL=1
          RETURN
      ENDIF
   ELSE
      IF(ABS(A-B).LE.(TOL*(ABS(B)+10.**(-
10.))) THEN
          IGUAL=0
           RETURN
       ELSE
          IGUAL=1
          RETURN
      ENDIF
   ENDIF
   END
```

Asociado a cualquier aritmética de precisión finita de números reales, existen 4 operaciones básicas que son suma, resta, multiplicación y división de números reales dentro de la aritmética. Nosotros no vamos a entrar en este curso en como se pueden definir algorítmicamente estas operaciones, solamente mencionar, que a menudo, para minimizar el efecto de los redondeos en las operaciones, antes de realizar la operación se aumenta la precisión de los números reales (por ejemplo pasando de simple precisión a doble precisión) para a continuación realizar la operación en una aritmética de mayor precisión, y finalmente el resultado se redondea para pasarlo a la precisión inicial.

**Fuentes de errores numéricos.**

Dentro de las posibles fuentes de errores numéricos, destacaremos 3 tipos.

**Errores de redondeo**. Son los que se producen al "redondear" un número real para poder expresarlo en una aritmética de precisión finita. Como vimos en la sección anterior este error esta controlado por la denominada unidad de redondeo $u = 2^{-t}$. De tal forma que al tomar un número real $z$, y aproximarlo en la aritmética por el $\widetilde{z} \in A$ más próximo el error de redondeo tiene la expresión:

$$| z - \widetilde{z} | \leq | z | u$$

**Errores de cambio de base**. Este tipo de errores se produce al realizar un cambio de base para representar un número real. Como vimos en la sección anterior, las aritméticas standard de ordenador trabajan en base 2. Sin embargo, los humanos pensamos y razonamos en términos de números en base 10. Por ejemplo, números tan naturales para nosotros como 0.1 no pueden representarse de forma exacta en una aritmética en base 2, ello quiere decir que al representar 0.1 el ordenador va a producir un pequeño redondeo, y este pequeño error de redondeo se puede ir propagando hasta producir errores apreciables. Por ejemplo, parece razonable pensar que cuando sumamos 100 veces el número 0.01 el resultado sea exactamente 1, sin embargo ello no es así, sin embargo si sumamos $128 = 2^7$ veces el número $2^{-7}$, el resultado si es exactamente 1. Este resultado se pone de manifiesto en el siguiente programa fortran:

**Programa 5** *Programa en fortran* 77 *para determinar la diferencia entre trabajar en base* 10 *y trabajar en base* 2.

```
      A=2**(-7.)
      B=0
      C=0.01
      D=0
      DO 1 K=1,2**7
          B=B+A
1     CONTINUE
      DO 2 K=1,100
          D=D+C
2     CONTINUE
      PRINT *,(1-B)*(10**10)
      PRINT *,(1-D)*(10**10)
      END
```

Además este programa permite identificar la base de la aritmética con la que trabaja el ordenador.

Como conclusión de este apartado podemos sacar que para ser más precisos numéricamente cuando trabajamos con números más pequeños que la unidad deberíamos pensar en términos de $2^{-m}$ en lugar de $10^{-m}$, que es como solemos hacerlo.

**Errores por Cancelación**. Estos errores se producen al restar números de aproximadamente la misma magnitud. Hay que tener en cuenta que al realizar operaciones sobre una variable, los errores de redondeo se van acumulando en la parte menos significativa del número (los decimales más pequeños) dejando relativamente intacta la parte más significativa del número que corresponde a los decimales más grandes, por ello al restar dos números de magnitud parecida se cancelan las partes significativas, quedando la aportación de los pequeños decimales que es donde más error hay. Por ejemplo en el programa fortran anterior, se ha utilizado este fenómeno de cancelación para poner de manifiesto la diferencia entre trabajar con bases distintas. En los algoritmos, muchas veces se intenta evitar la posibilidad de restar 2 números que pudieran ser de magnitud parecida. Por ejemplo en la conocida fórmula del cálculo de raíces de un polinomio de grado 2, $ax^2 + bx + c = 0$ (con $a \neq 0$)

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

una forma de evitar la cancelación que se produce cuando $b \approx \sqrt{b^2-4ac}$ es calculando primero la raíz de mayor valor absoluto, es decir

$$x_1 = \frac{-\left(b + sign(b)\sqrt{b^2-4ac}\right)}{2a}$$

y calculando después la segunda raíz $x_2$ utilizando la relación $x_1 x_2 = \frac{c}{a}$.

Por lo tanto, en los algoritmos, se deberá evitar en la medida de lo posible, la resta de variables que tengan una magnitud cercana.

**Problema 10** ***(1 punto)*** *Calcular las raíces del polinomio* $P(x) = x^2 - 2x + 0.01$ *evitando los errores de cancelación.*

**Problema 11** ***(3 puntos)*** *Escribir el código en fortran 77 para implementar el cálculo de las raíces de* $ax^2 + bx + c = 0$ *evitando los errores de cancelación y teniendo en cuenta las diferentes opciones que aparecen cuando* $a \neq 0$ *y* $a = 0$.

## CALCULO DE LOS CEROS DE UNA FUNCION

En esta sección vamos a estudiar algunos métodos para calcular los ceros de una función de una variable, esto es los valores $x$ para los cuales $f(x) = 0$.

### Método de la bisección.

Se considera un intervalo $[a, b]$ donde la función $f(x)$ cambia de signo, es decir $f(a) \cdot f(b) < 0$, el método consiste en ir dividiendo el intervalo $[a, b]$ por la mitad de la siguiente forma: se toma el punto medio $\frac{a+b}{2}$, Si $f(\frac{a+b}{2}) = 0$ ya hemos encontrado la raíz $x = \frac{a+b}{2}$, en caso contrario, Si $f(\frac{a+b}{2}) \cdot f(b) < 0$ entonces hacemos $a = \frac{a+b}{2}$ y volvemos a subdividir el nuevo intervalo $[a, b]$. Si, por el contrario $f(a) \cdot f(\frac{a+b}{2}) < 0$ entonces hacemos $b = \frac{a+b}{2}$ y volvemos a empezar. Las sucesivas subdivisiones del intervalo $[a, b]$ van aproximando la raíz.

**Problema 12** ***(2 puntos)*** *Calcular* 2 *iteraciones del algoritmo de la bisección para buscar un cero de la función* $f(x) = x^2 - 2$ *en el intervalo* $[-2, 0]$

**Problema 13** ***(3 puntos)*** *Escribir el código en fortran 77 para implementar el método de la bisección*

### Método de la Regula-falsi

Este método es una variación del anterior en el sentido siguiente: En lugar de tomar el punto medio $\frac{a+b}{2}$ del intervalo, se considera el punto de intercepción de la recta que pasa por $f(a)$ y $f(b)$ y el eje $x$, es decir, en el razonamiento anterior, se sustituye el valor $x_m = \frac{a+b}{2}$ por el valor

$$x_m = a - \frac{b-a}{f(b)-f(a)} f(a)$$

**Problema 14** ***(2 puntos)*** *Calcular* 2 *iteraciones del algoritmo de la regula-falsi para buscar un cero de la función* $f(x) = x^2 - 2$ *en el intervalo* $[0, 2]$

**Problema 15** ***(3 puntos)*** *Escribir el código en fortran 77 para implementar el método de la Regula-falsi*

### Método de Newton-Raphson

Este es sin duda uno de los métodos más importantes y útiles para el cálculo de raíces. Dada una aproximación inicial de la raíz $x_0$, se busca, a partir de $x_0$ una aproximación mejor $x_1$ de la raíz de la siguiente forma: Se sustituye la función $f(x)$ por el valor de su desarrollo de Taylor centrado en $x_0$ hasta el orden 1, es decir

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

que corresponde a un polinomio de grado 1, y a continuación se calcula $x_1$ como el cero de este polinomio, es decir:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

y por tanto, de forma general, obtenemos, a partir de $x_0$ una secuencia $x_n$ de valores que van aproximando la raíz definidos por

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

A continuación veremos una aplicación de este método para calcular la raíz cuadrada de un número positivo $A$, teniendo en cuenta que decir que $x = \sqrt{A}$, es equivalente a $f(x) = x^2 - A = 0$

**Programa 6** *Programa en fortran 77 para calcular una aproximación de la raíz cuadrada de un número positivo* $A$ *con una tolerancia* $TOL$, *y un número máximo de iteraciones* $N\max$.

```
READ *,A,TOL,Nmax
IF(A.LE.0) THEN
      PRINT *,'El numero A no es positivo'
     STOP
ENDIF
X0=(1+A)/2.
```

```
      DO 1 K=1,Nmax
          X1=X0-(X0*X0-A)/(2.*X0)
          IF(IGUAL(X0,X1,TOL).EQ.0) THEN
              PRINT *,'LA RAIZ DE A ES',X0
              STOP
          ELSE
              X0=X1
          ENDIF
    1     CONTINUE
      PRINT *,'No máximo de iterac. excedido'
      END
```

### El método de la Secante

Este método es una variante del método de Newton para el caso en que no sea posible calcular la derivada de $f(x)$ de una forma analítica. En este caso se sustituye el valor $f'(x_n)$ en el algoritmo, por el valor

$$\frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

que corresponde a una aproximación de $f'(x_n)$. Para iniciar el algoritmo son necesarios dos aproximaciones iniciales $x0$ y $x1$.

**Problema 16** ***(1 punto)*** *Calcular una iteración del método de Newton-Raphson para calcular un cero de la función* $f(x) = x^3 - 3$ *partiendo de* $x_0 = 1$.

**Problema 17** ***(1 punto)*** *Calcular una iteración del método de la secante para calcular un cero de la función* $f(x) = x^3 - 3$ *partiendo de* $x_0 = 0$, $x_1 = 1$

**Problema 18** ***(3 puntos)*** *Escribir un programa en fortran 77 que implemente el método de la Secante utilizando reales de doble precisión. Los datos de entrada son las aproximaciones iniciales* $x0$, *y* $x1$, *El número máximo de iteraciones* $N\max$, *y la tolerancia* $TOL$ *para determinar la igualdad de dos números.*

### Método de Muller.

Este método es de utilidad para calcular raíces complejas de funciones, como por ejemplo polinomios, es una generalización del método de Newton-Raphson, en el sentido de que en lugar de quedarnos con la parte lineal del desarrollo de Taylor de la función, nos quedamos con los términos hasta el orden 2 de tal forma que hacemos

$$f(x) \approx f(x_{n-1}) + f'(x_{n-1})(x - x_{n-1}) + \frac{f''(x_{n-1})}{2}(x - x_{n-1})^2$$

donde $x_{n-1}$ es una aproximación de una raíz compleja de la función $f(x)$. Para obtener una aproximación $x_n$ mejor de la raíz de la raíz calculamos los ceros del polinomio de segundo grado anterior, es decir

$$x_n = x_{n-1} + \frac{-f'(x_{n-1}) \pm \sqrt{(f'(x_{n-1}))^2 - 2f(x_{n-1})f''(x_{n-1})}}{f''(x_{n-1})}$$

de las dos posibles raíces nos quedamos con aquella más cercana a $x_{n-1}$. Dicha raíz será la aproximación $x_n$ de la raíz de $f(x)$ en la etapa $n$. En el caso en que $f''(x_{n-1}) = 0$, calculamos $x_n$ por el método de Newton-Raphson. En el caso en que no conozcamos analíticamente el valor de la primera y segunda derivada de $f(x)$, podemos utilizar las siguientes aproximaciones:

$$f''(x_{n-1}) \approx 2\frac{\frac{f(x_{n-2}) - f(x_{n-3})}{x_{n-2} - x_{n-3}} - \frac{f(x_{n-1}) - f(x_{n-2})}{x_{n-1} - x_{n-2}}}{x_{n-3} - x_{n-1}}$$

$$f'(x_{n-1}) \approx \frac{f(x_{n-1}) - f(x_{n-2})}{x_{n-1} - x_{n-2}} + \frac{f''(x_{n-1})}{2}(x_{n-1} - x_{n-2})$$

como veremos posteriormente, la elección de las fórmulas anteriores equivale a aproximar $f(x)$ por la parábola que pasa por los puntos $(x_{n-3}, f(x_{n-3}))$, $(x_{n-2}, f(x_{n-2}))$ y $(x_{n-1}, f(x_{n-1}))$, y calcular posteriormente las derivadas de dicha parábola.

**Programa 7** *Programa en fortran 77 donde se muestra un ejemplo de manejo de números complejos.*

```
      IMPLICIT COMPLEX (C)
      CX=(-1,0)
      CY=CF(CX)
      PRINT *,CY
      END

      FUNCTION CF(CX)
      IMPLICIT COMPLEX(C)
      CF=SQRT(CX)
      END
```

### Práctica 2 (El método de Muller, 4 horas)

Implementar el método de Muller. Crear un programa en Fortran 77 que tenga como datos de entrada: los tres primeros valores $x_0, x_1, x_2$ de aproximación de la raíz, el Número máximo de iteraciones $N\max$, y la tolerancia $TOL$, para determinar la igualdad entre dos números. La función a la que se le calculan los ceros se define en el propio cuerpo del programa. Utilizar el método para calcular los posibles ceros de las siguientes funciones

1. $f(x) = x^2 + 1$
2. $f(x) = (x^2 + 1)x$
3. $f(x) = e^x - 1$
4. $f(x) = x - 2$

5. $f(x) = 1$

**Nota:** Utilizar como tolerancia $TOL = 0.0001$ y $N$ max $= 100$. Para el ejemplo 1 tomar como datos iniciales $x_0 = (3,0)$, $x_1 = (2,0)$ $x_2 = (1,0)$. Para el ejemplo 2 tomar como datos iniciales $x_0 = (3,0)$, $x_1 = (2,0)$, $x_2 = (1,0)$ y $x_0 = (1,0)$, $x_1 = (0.1,0)$, $x_2 = (0.01,0)$. Para el ejemplo 3 tomar como datos iniciales $x_0 = (3,0)$, $x_1 = (2,0)$, $x_2 = (1,0)$. Para el ejemplo 4 tomar como datos iniciales $x_0 = (3,0)$, $x_1 = (2,0)$, $x_2 = (1,0)$.Para el ejemplo 5 tomar como datos iniciales $x_0 = (3,0)$, $x_1 = (2,0)$, $x_2 = (1,0)$

### Cálculo de las raíces de un polinomio.

Los polinomios son un tipo particular de funciones que por su gran utilidad requieren un análisis algo más en detalle. Nos ocuparemos sólo de las raíces reales de polinomios, aunque también hay que indicar que existen algoritmos versátiles para el cálculo de las raíces complejas, como por ejemplo el método de Muller visto anteriormente.

A menudo, los alumnos pueden tener la impresión de que los algoritmos y técnicas que se aprenden en una asignatura como análisis numérico les serán de poca utilidad en el futuro. Mi experiencia como docente en esta disciplina es que con frecuencia, una vez terminada la carrera y en el desarrollo de la actividad profesional, aparecen problemas, que para su resolución requieren el uso de alguna de las técnicas presentadas en esta asignatura. El siguiente ejemplo es una buena prueba de ello.

**Ejemplo 2** *Actualmente están muy de moda los planes de pensiones. Las entidades financieras venden a sus clientes los planes de pensiones de la siguiente forma, por ejemplo, si usted aporta durante* 30 *años* 100.000 *pesetas todos los años, aportación que se va incrementando cada año en un* 10%, *es decir el* 1 *año* 100.000, *el segundo año* 110.000 *etc.. entonces le aseguramos que al final del año* 30 *tendrá a su disposición la cantidad de* 26.000.000 *de pesetas. Ahora bien, el dato más importante para el futuro pensionista (que a menudo oculta la entidad financiera) es el interés nominal anual que se está aplicando año tras año al dinero depositado?. Si llamamos i al interés nominal anual que se aplica al dinero, la ecuación que debemos resolver para obtener i es*

$$\sum_{n=0}^{29} (100.000)\,(1.1)^n\,(1.+i)^{30-n} = 26.000.000$$

*Ahora bien para calcular i, debemos calcular las raíces del polinomio en i dado por*

$$P(i) = \sum_{n=0}^{29} (100.000)\,(1.1)^n\,(1.+i)^{30-n} - 26.000.000$$

*el cálculo de las raíces de este polinomio nos lleva a* $i = 4.487\%$. *Este ejemplo muestra como un problema financiero sencillo nos lleva a la necesidad de calcular los ceros de un polinomio.*

*Algoritmo de Horner para evaluar un polinomio en un punto*

Dado un polinomio $P(x) = a_n x^n + a_{n-1}x^{n-1} + .....a_0$, dicho polinomio se puede expresar también de la forma siguiente: $P(x) = a_0 + x\,(a_1 + x\,(a_2 + x\,(a_3 + x(..... + x\,(a_{n-1} + xa_n))))).$, además, si queremos utilizar un método de cálculo de raíces como el de Newton-Raphson, necesitamos evaluar tanto el polinomio como su derivada. El siguiente resultado muestra una forma rápida y sencilla de evaluar simultáneamente un polinomio y su derivada

**Teorema 4** *(Método de Horner). Sea* $P(x) = a_n x^n + a_{n-1}x^{n-1} + .....a_0$, *si definimos* $b_k$ *como*

$$b_n = a_n$$

$$b_k = a_k + b_{k+1}x_0$$

*entonces se verifica*

$$P(x_0) = b_0$$

$$P'(x_0) = b_n x_0^{n-1} + b_{n-1}x_0^{n-2} + ..... + b_1$$

**Demostración** Sea el polinomio $Q(x) = b_n x^{n-1} + b_{n-1}x^{n-2} + ..... + b_1$. Veamos que se verifica

$$P(x) = (x - x_0)Q(x) + b_0$$

efectivamente, dado que $a_k = b_k - b_{k+1}x_0$, y $a_n = b_n$ obtenemos la igualdad anterior teniendo en cuenta

$$(x - x_0)Q(x) + b_0 =$$

$$b_n x^n + (b_{n-1} - b_n x_0)x^n + ..... + (b_0 - b_1 x_0)$$

Por último, obtenemos

$$P'(x) = (x - x_0)Q'(x) + Q(x)$$

de donde sale obviamente que $P'(x_0) = Q(x_0)$ ■

Este teorema permite calcular el polinomio y su derivada en un punto de forma muy sencilla, como muestra el siguiente programa fortran.

**Programa 8** *El siguiente programa en fortran 77 calcula la evaluación de un polinomio y su derivada en un punto* $X$, *almacenandolos en las variables* $PX$ *y* $PPX$.

```
PARAMETER(NMAX=1000)
DIMENSION A(0:NMAX)
COMMON/POL/PX,PPX
PRINT *,'Escribir Grado del Polinomio'
READ *,N
IF(N.GT.NMAX) THEN
      PRINT *, 'Grado Superior al Maximo'
STOP
ENDIF
```

```
      PRINT *,'EscriBir Coef. Polin.'
      DO 1 K=0,N
1         READ *,A(K)
      PRINT *,'Escribir valor de X'
      READ *,X
      CALL HORNER(N,A,X)
      PRINT *,'P(X)= ',PX
      PRINT *,'P '(X)= ',PPX
      END

      SUBROUTINE HORNER(N,A,X)
      DIMENSION A(0:*)
      COMMON/POL/PX,PPX
      PX=A(N)
      PPX=A(N)
      DO 1 K=N-1,1,-1
          PX=PX*X+A(K)
          PPX=PPX*X+PX
1     CONTINUE
      PX=PX*X+A(0)
      END
```

**Nota:** La declaración
$PARAMETER(NMAX = 1000)$
permite definir constantes. La declaración
$DIMENSION\ A(0 : NMAX)$
define un vector $A$, de reales en precisión simple, de tamaño $NMAX+1$, y numerados desde 0 hasta $NMAX$. La declaración
$COMMON/POL/PX, PPX$
define la zona de memoria denomina $POL$ donde se encuentran las variables globales $PX, PPX$. Para que una subrutina pueda hacer uso de esas variables debe incluir en su inicio la misma sentencia $COMMON$.

Otros resultados interesantes de utilidad para localizar en que zonas pueden estar las raíces del polinomio son:

**Teorema 5** *Sea un polinomio* $P(x) = a_n x^n + a_{n-1}x^{n-1} + .....a_0$ *con* $a_n \neq 0$, *entonces las raíces reales de* $P(x)$ *están en el intervalo*

$$\left[-1 - \frac{\max_{k=0,..,n-1} \mid a_k \mid}{\mid a_n \mid}, 1 + \frac{\max_{k=0,..,n-1} \mid a_k \mid}{\mid a_n \mid}\right]$$

**Demostración** Veamos que si $|x| > 1 + \frac{\max_{k=0,..,n-1}|a_k|}{|a_n|}$ entonces $|P(x)| > 0$. Efectivamente,

$$\begin{aligned}
|P(x)| &\geq |a_n x^n| - \max_{k=0,,,n-1} |a_k| \sum_{k=0}^{n-1} |x|^k = \\
&= |a_n|\,|x|^n - \max_{k=0,,,n-1} |a_k| \frac{1-|x|^n}{1-|x|} \geq \\
&\geq |a_n|\,|x|^n - \max_{k=0,,,n-1} |a_k| \frac{|x|^n}{|x|-1} = \\
&= \frac{|x|^n (|a_n|\,(|x|-1) - \max_{k=0,,,n-1} |a_k|)}{|x|-1} > 0
\end{aligned}$$

**Teorema 6** *Sea un polinomio* $P(x) = a_n x^n + a_{n-1}x^{n-1} + .....a_0$, *entonces el número de raíces positivas es igual al número de cambios de signos en los coeficientes* $a_n.....a_0$ *(saltando los posibles coeficientes nulos), o bien ese mismo número menos un número par.*

**Demostración** [Is-Ke] Pg. 126

Para la estimación del número de raíces reales negativas, se aplica el teorema anterior cambiando $x$ por $-x$

**Ejemplo 3** *Sea* $P(x) = 3x^4 + 10x^3 - 10x - 3$ *los signos de los coeficientes son:* $++--$ *por tanto hay un único cambio de signo y por tanto hay una raíz positiva Si cambiamos* $x$ *por* $-x$ *los signos de los coeficientes son* $+-+-$ *por tanto hay 3 cambios de signo, y por tanto hay* 3 *raíces o* 1 *raíz negativa. En este caso las raíces son* $x = 1, -1, -3, -\frac{1}{3}$..

**Problema 19** ***(1 punto)*** *Calcular una iteración del método de Muller para calcular un cero de la función* $f(x) = x^3 - 3$ *partiendo de* $x_0 = 1$ *(Calculando las derivadas de la función de forma exacta) y quedándonos con la raíz más cercana a* $x_0$.

**Problema 20** ***(2 puntos)*** *Dado el polinomio* $P(x) = 2x^3 + 3x^2 + 4x + 5$. *Evaluar el polinomio y su derivada en el punto* $x = 2$, *utilizando el algoritmo de Horner*

**Problema 21** ***(1 punto)*** *Calcular el número máximo de raíces positivas y negativas del polinomio* $x^5 - 35x^3 + 30x^2 + 124x - 120$, *y localizarlas en un intervalo.*

**Teorema 7** *Entre dos raíces de una función derivable* $f(x)$ *hay una raíz de* $f'(x)$

**Demostración** Teorema de Rolle

**Teorema 8** *La derivada* $k-$*ésima* $P^{k)}(x)$ *del polinomio* $P(x) = a_n x^n + a_{n-1}x^{n-1} + .....a_0$ *es*

$$P^{k)}(x) = \frac{a_n n!}{(n-k)!}x^{n-k} + \frac{a_{n-1}\,(n-1)\,!}{(n-k-1)!}x^{n-k-1} + ... + a_k \frac{k!}{1}$$

**Demostración** Es inmediato, derivando sucesivamente el polinomio $P(x)$

Los dos resultados anteriores permiten aislar las posibles raíces de $P(x)$ de la forma siguiente: Si denotamos por $P_{\max} = \frac{\sum_{k=0}^{n-1}|a_k|}{|a_n|}$, entonces las $m$ raíces distintas

$x_1 < x_2 < .... < x_m$ de $P(x)$ están intercaladas con las raíces $x'_1 < x'_2 < .... < x'_{m-1}$ de $P'(x)$, es decir

$$-P_{\max} \leq x_1 \leq x'_1 \leq x_2 \leq x'_2 \leq ... \leq x'_{m-1} \leq x_m \leq P_{\max}$$

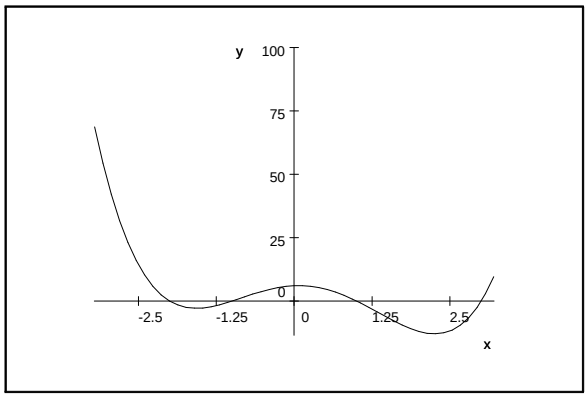
volviendo a aplicar este razonamiento sucesivamente sobre $P'(x)$, $P''(x)$,... para intercalar los ceros de una derivada con los ceros de la siguiente, podemos deducir el siguiente algoritmo para aislar todas las raíces de un Polinomio $P(x)$:

1. Se parte del intervalo $[-P_{\max}, P_{\max}]$
2. Se calcula la raíz $x_1^{n-1)}$ del Polinomio $P^{n-1)}(x)$ (que es un polinomio de grado 1)
3. Para $k = n-2, ...1$
   Se calculan las raíces de $P^k(x)$ en los intervalos

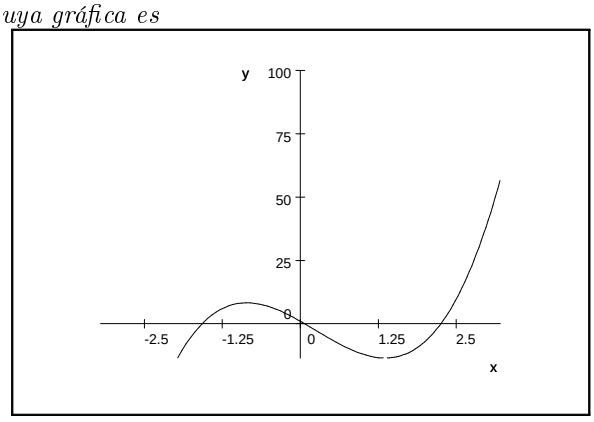
$$-P_{\max} < x_1^{k+1)} < x_2^{k+1)} < ... < x_{m_{k+1}}^{k+1)} < P_{\max}$$

Al final del procedimiento habremos aislado completamente a las raíces de $P(x)$. Este procedimiento se puede utilizar para grados relativamente pequeño ($n < 30$), puesto que su utilización requiere el cálculo de factoriales que se dispara rápidamente, por ejemplo $30! = 2.6 \times 10^{32}$. Existen métodos mejores para el cálculo de raíces de polinomios, pero que utilizan técnicas más complejas. El método presente en el siguiente programa, que combina el aislamiento de las raíces del polinomio a través de los ceros de sus derivadas con el método de Newton-Raphson funciona razonablemente bien para grados de polinomios pequeños, en el caso de raíces múltiples los resultados acumulan mayores errores de redondeo debido a que tanto el polinomio como su derivada son cero en el mismo punto.

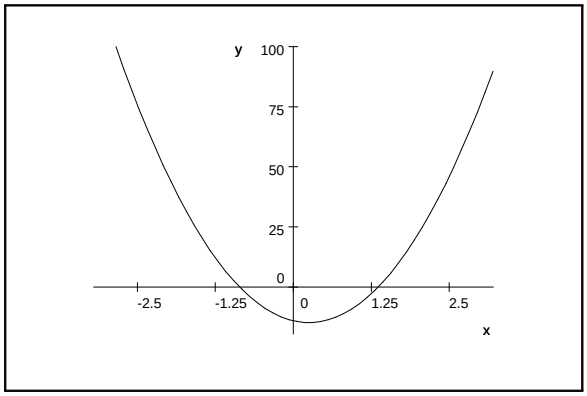
**Ejemplo 4** *Consideremos el polinomio* $P(x) = x^4 - x^3 - 7x^2 + x + 6$*.que tiene por raices* $x = 1, 3, -1, -2$ *Para este polinomio tenemos que* $P_{\max} = 15$*,por tanto las raíces están en el intervalo* $[-15, 15]$*. Por otro lado su gráfica es:*



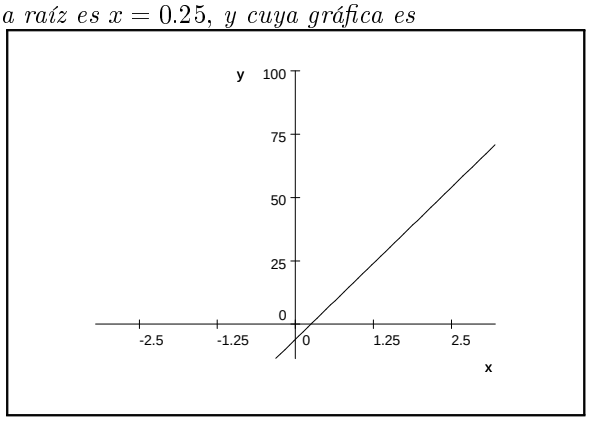

*Polinomio* $P(x) = x^4 - x^3 - 7x^2 + x + 6$
*la derivada de este polinomio es* $P'(x) = 4x^3 - 3x^2 - 14x + 1$*,cuyas raíces son* $x = -1.574, 7.05 \times 10^{-2}, 2.253$ *y cuya gráfica es*




*Polinomio* $P'(x) = 4x^3 - 3x^2 - 14x + 1$
*la derivada segunda de este polinomio es* $P''(x) = 12x^2 - 6x - 14$*,cuyas raíces son* $x = -0.858, 1.358$ *y cuya gráfica es*




*Polinomio* $P''(x) = 12x^2 - 6x - 14$
*la derivada tercera de este polinomio es* $P'''(x) = 24x - 6$*, cuya raíz es* $x = 0.25$*, y cuya gráfica es*




*Polinomio* $P'''(x) = 24x - 6$

El método funcionaría de la siguiente forma: primero calculamos el cero de $P'''(x)$, es decir $x = 0.25$, por tanto los ceros de $P''(x)$ estarían en los intervalos $[-15, 0.25]$ y $[0.25, 15]$. Puesto que hay cambio de signo de $P''(x)$ en cada uno de estos intervalos buscamos las raíces de $P''(x)$ en esos intervalos, utilizando cualquier método numérico de los vistos anterioremente, por ejemplo el método de la regula-falsi, obteniendo $-0.858$ para el intervalo $[-15, 0.25]$ y $1.358$ para el intervalo $[0.25, 15]$. Por

tanto las posibles raíces de $P'(x)$ estarán en los intervalos $[-15, -0.858]$, $[-0.858, 1.358]$ y $[1.358, 15]$. Buscamos ahora las raíces de $P'''(x)$ es esos intervalos obteniendo $x = -1.574, 7.05 \times 10^{-2}, 2.253$. Por tanto los posibles ceros de $P(x)$ estarán en los intervalos $[-15, -1.574]$, $[-1.574, 7.05 \times 10^{-2}]$, $[7.05 \times 10^{-2}, 2.253]$ y $[2.253, 15]$. Buscamos finalmente las raíces de $P(x)$ en cada un de esos intervalos y obtenemos $x = -2, -1, 1, 3$.

**Problema 22** ***(2 puntos)*** *Aislar en intervalos las raíces del polinomio* $P(x) = 20x^3 - 45x^2 + 30x - 1$.

**Programa 9** *Programa en fortran* 77 *donde se implementa la función* $ICEROPOL(A, R, TOL, N, Nmaxx)$ *que devuelve las raíces reales de un polinomio. Dicha subrutina tiene como parámetros un vector* $A()$ *donde están los coeficientes del polinomio, un vector* $R()$ *donde se guardan las raíces del polinomio una vez calculadas, la tolerancia* $TOL$ *con la que consideramos que dos números son iguales, el grado del polinomio* $N$, *y el número máximo de iteraciones* $N\max xx$ *para el proceso de Newton-Raphson. También se define la función auxiliar* $RP(N, A, X1, X2, TOL, Nmaxx, R, L)$ *que devuelve la raíz del polinomio que se obtiene aplicando el método de Newton-Raphson tomando como valor inicial el punto medio del intervalo* $[X1, X2]$.

```
      PARAMETER(NMAX=30)
      DIMENSION A(0:NMAX),R(0:NMAX-1)
      COMMON/POL/PX,PPX
      PRINT *,'Escribir Grado del Polinomio'
      READ *,N
      IF(N.GT.NMAX) THEN
        PRINT *, 'Grado Superior al Maximo'
        STOP
      ENDIF
      PRINT *,'Escribir Coef. Polin.'
      DO 1 K=0,N
1       READ *,A(K)
      PRINT *, 'Escribir Tolerancia'
      READ *,TOL
      PRINT *, 'Escribir No. iter. Max. para Newton-
Raphson'
      READ *,Nmaxx
      M=ICEROPOL(A,R,TOL,N,Nmaxx)
      PRINT *,'El Pol. tiene',M,' raices'
      DO 9 K=0,M-1
9       PRINT *,R(K)
      END



      FUNCTION ICEROPOL(A,R,TOL,N,Nmaxx)
      PARAMETER(NMAX=30)
      DIMENSION A(0:*), R(0:*), F(0:NMAX),
AP(0:NMAX), PI(0:NMAX+1)
      COMMON/POL/PX,PPX
**** Calculo de los factoriales
      F(0)=1.
      DO 2 K=1,N
2       F(K)=F(K-1)*K
*** Calculo intervalo inicial
      PMAX=ABS(A(0))
      DO 3 K=1,N-1
          IF(PMAX.LT.ABS(A(K)) THEN
            PMAX=ABS(A(K)
          ENDIF
3       CONTINUE
      PMAX=PMAX/ABS(A(N))+1.
      PI(0)=-PMAX
      PI(1)=-(A(N-1)*F(N-1))/(A(N)*F(N))
      DO 10 K=2,N
10      PI(2)=PMAX
*** Calculo de los coeficientes del
*** polinomio derivada
      DO 7 K=2,N
7       PI(K)=PMAX
      DO 4 K=N-2,0,-1
        DO 5 L=0,N-K
          AP(L)=A(L+K)*(F(K+L)/F(L))
5       CONTINUE
***CALCULAR LOS CEROS DE AP EN LOS INTER-
VALOS PI()
        DO 6 L=1,N-K
          PI(L)=RP(N-K,AP,PI(L-
1),PI(L),TOL,Nmaxx,R,L-1)
6       CONTINUE
4     CONTINUE
*** Pasamos las raices al vector R()
      M=0
      DO 8 K=1,N
        IF(R(K-1).EQ.0) THEN
          R(M)=PI(K) M=M+1
        ENDIF
8     CONTINUE
      ICEROPOL=M
      END



      FUNCTION RP(N,A,X1,X2,TOL,Nmaxx,R,L)
      DIMENSION A(0:*),R(0:*)
      COMMON/POL/PX,PPX
      R(L)=1.
      IF (X1.EQ.X2) THEN
        RP=X1
        RETURN
      ENDIF
      RP=(X1+X2)/2.
      DO 1 K=1,Nmaxx
        CALL HORNER(N,A,RP)
        IF (PPX.EQ.0.) THEN
          IF(PX.EQ.0.) THEN
            R(L)=0.
            RETURN
```

```
      ELSE
        RETURN       ENDIF
    ELSE
      RP1=RP-PX/PPX
        IF(IGUAL(RP1,RP,TOL).EQ.0) THEN
          RP=RP1
          R(L)=0.
          RETURN
        ELSE
          RP=RP1
        ENDIF
    ENDIF
1   CONTINUE
    END
```

**Problema 23** ***(2 puntos)*** *Aislar en intervalos las raíces del polinomio* $P(x) = 2x^3 + 3x^2 - 12x + 1$

## INTERPOLACION DE FUNCIONES I

El problema general de la interpolación de funciones es, a partir del conocimiento del valor de una función (y eventualmente de sus derivadas) en un conjunto finito de puntos, extrapolar el valor de la función fuera de ese conjunto finito de puntos.

### Interpolación por polinomios de Lagrange.

Sea una función $f(x)$ que conocemos en un conjunto finito de valores $\{x_i\}_{i=0,..,N}$ . Es decir, sabemos que $f(x_i) = f_i$. El polinomio interpolador de Lagrange $P_N(x)$ de $f(x)$ en los puntos $\{x_i\}_{i=0,..,N}$ , es el único polinomio de grado menor o igual que $N$ tal que

$$P_N(x_i) = f(x_i) \qquad \forall i = 0, .., N$$

$P_N(x)$ se puede expresar en término de los denominados polinomios base de Lagrange $P^i(x)$ definidos como:

$$P^i(x) = \frac{\Pi_{j\neq i}^N (x - x_j)}{\Pi_{j\neq i}^N (x_i - x_j)}$$

estos polinomios base tienen la propiedad fundamental siguiente

$$P^i(x_j) = \begin{cases} 1 & si \quad i = j \\ 0 & si \quad i \neq j \end{cases}$$

por tanto el polinomio interpolador de Lagrange puede expresarse como

$$P_N(x) = \sum_{i=0}^{N} f(x_i) P^i(x)$$

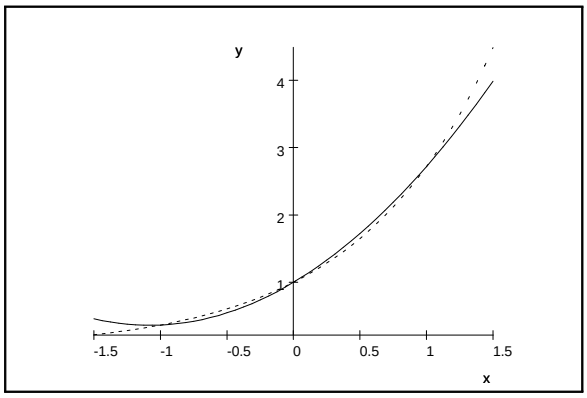
**Ejemplo 5** *Consideremos una función* $f(x) = e^x$ *, vamos a interpolarla en los puntos* $x_0 = 0$, $x_1 = -1$ *y* $x_2 = 1$. *Para calcular* $P_2(x)$ *el polinomio interpolador de Lagrange en estos puntos calcularíamos los polinomios base:*

$$P^0(x) = \frac{(x+1)(x-1)}{-1}$$
$$P^1(x) = \frac{x(x-1)}{2}$$
$$P^2(x) = \frac{x(x+1)}{2}$$

*siendo el polinomio interpolador:*

$$P_2(x) = e^0 \frac{(x+1)(x-1)}{-1} + e^{-1} \frac{x(x-1)}{2} + e \frac{x(x+1)}{2}$$

*en la siguiente figura comparamos la gráfica del polinomio* $P_2(x)$ *(trazo continuo), con la gráfica de la función* $e^x$ *(trazo discontinuo)*




**Problema 24** ***(2 puntos)*** *Calcular el polinomio interpolador de Lagrange* $P_3(x)$ *de la función* $f(x) = sen(x)$ *en los puntos* $0, \frac{\pi}{2}, \pi$ *y* $\frac{3\pi}{2}$.

**Teorema 9** *El polinomio interpolador de Lagrange es el único polinomio de grado igual o inferior a* $N$ *tal que*

$$P_N(x_i) = f(x_i) \qquad \forall i = 0, .., N$$

**Demostración** Sea $P(x)$ un polinomio de grado inferior o igual a $N$ que verifique $P(x_i) = f(x_i) \qquad \forall i = 0, .., N$, Entonces el polinomio $Q(x) = P(x) - P_N(x)$ es un polinomio de grado inferior o igual a $N$ que verifica $Q(x_i) = 0$ y por tanto posee $N + 1$ raíces, lo cual es imposible salvo que $Q(x)$ sea identicamente igual a cero. Por tanto $Q(x) \equiv 0$ y $P(x) = P_N(x)$. ■

**Error de interpolación de Lagrange y polinomios de Chebychev.**

Evidentemente, al aproximar $f(x)$ por el polinomio interpolador $P_N(x)$ en un intervalo $[a,b]$ se comete en general un error de interpolación que viene determinado por el siguiente teorema

**Teorema 10** *Sea $f(x)$ una función, y $P_N(x)$ su polinomio interpolador de Lagrange en los puntos $\{x_i\}_{i=0,..,N} \subset [a,b]$ y $x \in [a,b]$, entonces*

$$f(x) - P_N(x) = \frac{f^{N+1)}(\xi)}{(N+1)!}\Pi_{i=0}^{N}(x - x_i)$$

*donde $\xi$ es un valor intermedio perteneciente a $[a,b]$.*

**Demostración** Si $x = x_i$ el error de interpolación es cero y por tanto la fórmula anterior es válida. Consideremos ahora $x$ distinto a los $x_i$ y definamos

$$\begin{aligned} w(t) &= \Pi_{i=0}^{N}(t - x_i) \\ \lambda &= \frac{f(x) - P_N(x)}{w(x)} \\ \phi(t) &= f(t) - P_N(t) - \lambda w(t) \end{aligned}$$

la función $\phi(t)$ tiene al menos $n+1$ ceros en los puntos $x_i$ y en el punto $x$. Por tanto su función derivada $\phi'(t)$ tiene al menos $n$ ceros repartidos entre los ceros de $\phi(t)$. Análogamente $\phi''(t)$ tiene al menos $n-1$ ceros y así sucesivamente hasta llegar a $\phi^{N+1}(t)$ que tiene al menos 1 cero. Si llamamos $\xi$ a dicho cero, obtenemos

$$\phi^{N+1}(\xi) = f^{N+1)}(\xi) - \lambda(N+1)!$$

de donde despejando y sustituyendo $\lambda$ por su valor obtenemos el resultado del Teorema. ■

**Problema 25** ***(2 puntos)*** *Interpolar la función $f(x) = \frac{10}{x^2+1}$ en los puntos $x_0 = -2$, $x_1 = -1$, $x_2 = 1$, $x_3 = 2$ utilizando las diferencias de Newton y evaluar el polinomio en $x = 0$ utilizando el algoritmo de Horner.*

**Problema 26** ***(2 puntos)*** *Calcular la expresión del error interpolación al aproximar la función $f(x) = sen(x)$ en el intervalo $[0, 2\pi]$ interpolando en los puntos $0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$. y acotarlo superiormente.*

La cuestión que vamos a abordar en este apartado es, en el caso en que queramos interpolar una función en un intervalo $[a,b]$, y que nosotros podamos elegir los valores de interpolación $x_i$, como elegirlos de tal forma que el error de interpolación sea mínimo. Para ello, elegiremos los puntos $x_i$ tales que $\Pi_{i=0}^{N}(x - x_i)$ sea lo más pequeño posible en $[a,b]$.

**Teorema 11** *Sea $N \geq 0.$, y un intervalo $[a,b]$ Se consideran los puntos $x_i$ dados por*

$$x_i = a + \frac{b-a}{2}\left(1 + \cos\left(\frac{2i+1}{2N+2}\pi\right)\right) \quad i = 0, ..., N$$

*entonces*

$$\begin{aligned} \max_{x\in[a,b]} \mid \Pi_{i=0}^{N}(x - x_i) \mid = \left(\frac{b-a}{2}\right)^{N+1}\frac{1}{2^N} \leq \\ \leq \max_{x\in[a,b]} \mid \Pi_{j=0}^{N}(x - \widetilde{x}_j) \mid \end{aligned}$$

*para cualquier otra elección posible de valores de interpolación $\widetilde{x}_j$.*

**Demostración** La demostración para el intervalo $[-1,1]$ se encuentra en [Ki-Ch] Pg. 292-294. La demostración para un intervalo cualquiera $[a,b]$ se obtiene fácilmente transformando el intervalo $[-1,1]$ en $[a,b]$

Por tanto, utilizando este resultado el error de interpolación máximo viene determinado por:

$$\mid f(x) - P_N(x) \mid \leq \frac{\max_{x\in[a,b]} f^{N+1)}(\xi)}{(N+1)!2^N}\left(\frac{b-a}{2}\right)^{N+1}$$

**Ejemplo 6** *Se considera $[a,b] = [0,1]$. y $N = 5$ (es decir 6 puntos de interpolación). Los puntos de interpolación dados por el teorema anterior son:*

$$\begin{aligned} x_0 &= .\,982\,96 \\ x_1 &= .\,853\,55 \\ x_2 &= .\,629\,41 \\ x_3 &= .\,370\,59 \\ x_4 &= .\,146\,45 \\ x_5 &= 1.\,703\,7 \times 10^{-2} \end{aligned}$$

**Problema 27** ***(2 puntos)*** *Calcular el error máximo de interpolación en el intervalo $[0,1]$ al interpolar la función $\cos(x)$ en los puntos descritos en el ejemplo anterior.*

En el caso de que $[a,b] = [-1,1]$, los valores óptimos de interpolación $x_i$ dados por la fórmula anterior son las raíces de los denominados polinomios de Chebychev $T_N(x)$ construidos de la manera siguiente

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_N(x) &= 2xT_{N-1}(x) - T_{N-2}(x) \end{aligned}$$

**Método de diferencias de Newton para el cálculo del polinomio interpolador de Lagrange.**

Numéricamenta, el cálculo de $P_N(x)$ a través de los polinomios base, necesita de la evaluación de $N+1$ polinomios de grado $N$. Además, si queremos añadir un nuevo punto

de interpolación, debemos cambiar todos los polinomios base de Lagrange. Un método más directo para el cálculo de $P_N(x)$ es el denominado método de diferencias de Newton. El método consiste en ir calculando progresivamente los polinomios $P_k(x)$ que interpolan a la función en los puntos $x_0, ..., x_k$ de la siguiente forma:

$$\begin{aligned} P_0(x) &= a_0 \\ P_1(x) &= P_0(x) + a_1(x - x_0) \\ P_2(x) &= P_1(x) + a_2(x - x_0)(x - x_1) \\ &\cdot \\ P_N(x) &= P_{N-1}(x) + a_N(x - x_0)(x - x_1)...(x - x_{N-1}) \end{aligned}$$

a los coeficientes $a_k$ los denotamos por

$$a_k = f[x_0, ..., x_k]$$

**Ejemplo 7** *Vamos a interpolar la función* $f(x) = e^x$ *en los puntos* $x_0 = 0$, $x_1 = 1$, *y* $x_2 = 2$..

$$P_0(x) = 1$$
$$P_1(x) = 1 + a_1 x$$

*como* $P_1(1)$ *debe ser igual a* $e$, *despejando obtenemos*

$$a_1 = e - 1$$

*Por último*

$$P_2(x) = P_1(x) + a_2 x(x - 1)$$

*como* $P_2(2)$ *debe ser igual a* $e^2$, *despejando obtenemos*

$$a_2 = \frac{e^2 - P_1(2)}{2}$$

*por tanto el polinomio* $P_2(x)$ *lo expresamos como*

$$P_2(x) = 1 + (e - 1)x + \frac{e^2 - 2e + 1}{2} x(x - 1)$$

Como veremos en el teorema siguiente los coeficientes $f[x_0, ..., x_k]$ que se denominan diferencias divididas de Newton, verifican la siguientes propiedades:

$$\begin{aligned} f[x_i] &= f(x_i) \\ f[x_i, x_{i+1}] &= \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i} \\ &\cdot \\ f[x_i, .., x_{i+k}] &= \frac{f[x_{i+1}, .., x_{i+k}] - f[x_i, .., x_{i+k-1}]}{x_{i+k} - x_i} \end{aligned}$$

**Teorema 12** *Si denotamos por* $a_k = f[x_0, .., x_k]$ *entonces el polinomio de interpolación de Lagrange* $P_N(x)$ *viene dado por*

$$P_N(x) = \sum_{k=0}^{N} a_k \Pi_{i=0}^{k-1}(x - x_i)$$

*donde los coeficientes* $f[x_i, ..., x_k]$ *verifican*

$$f[x_i, .., x_{i+k}] = \frac{f[x_{i+1}, .., x_{i+k}] - f[x_i, .., x_{i+k-1}]}{x_{i+k} - x_i}$$

**Demostración** En primer lugar, observamos que $f[x_i, ..., x_{i+k}]$ indica para cada $P_k(x)$, el coeficiente que acompaña a la potencia $x^k$ en el polinomio interpolador $P_k(x)$ que interpola en los puntos $x_i, ..., x_{i+k}$. Como el polinomio interpolador es único, $f[x_i, ..., x_{i+k}]$ no depende del orden en que tomemos los puntos $x_i, ..., x_{i+k}$ y por tanto:

$$f[x_i, ....., x_{i+k}] = f[x_{i+k}, ....., x_i]$$

consideremos ahora el polinomio interpolador $Q_k(x)$ que interpola en los puntos $x_{i+k}, ..., x_i$, es decir cambiando el orden de los puntos. $Q_k(x)$ se puede escribir como

$$Q_k(x) = b_0 + b_1(x - x_{i+k}) + b_2(x - x_{k+i})(x - x_{k+i-1}) + ...$$

donde

$$b_j = f[x_{i+k}, .., x_{i+k-j}]$$

por la unicidad del polinomio interpolador obtenemos que $P_k(x) = Q_k(x)$, y por tanto

$$a_k = f[x_i, ....., x_{i+k}] = f[x_{i+k}, ....., x_i] = b_k$$

de nuevo por la unicidad del polinomio interpolador los coeficientes que acompañan a la potencia $x^{k-1}$ en ambos polinomios coinciden, y por tanto:

$$a_{k-1} - a_k \sum_{j=0}^{k-1} x_{i+j} = b_{k-1} - b_k \sum_{j=1}^{k} x_{i+j}$$

por tanto despejando obtenemos

$$a_k = \frac{b_{k-1} - a_{k-1}}{x_{k+i} - x_i}$$

Finalmente obtenemos el resultado del teorema teniendo en cuenta que

$$\begin{aligned} a_{k-1} &= f[x_i, ...., x_{i+k-1}] \\ b_{k-1} &= f[x_{i+k}...., x_{i+1}] = f[x_{i+1}...., x_{i+k}] \end{aligned}$$

■

**Ejemplo 8** *Sea* $f(x) = e^x$, *si interpolamos* $f(x)$ *en los puntos* $x_0 = 0$, $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, *obtenemos el*

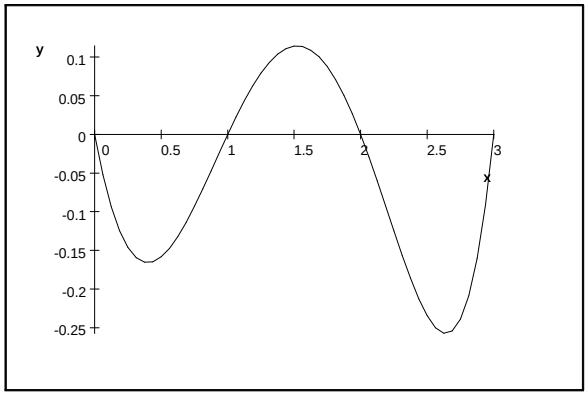
*polinomio interpolador de la siguiente forma:*

$$f[0,1] = e^1 - 1$$
$$f[1,2] = e^2 - e^1$$
$$f[2,3] = e^3 - e^2$$
$$f[0,1,2] = \frac{e^2 - 2e + 1}{2}$$
$$f[1,2,3] = \frac{e^3 - 2e^2 + e^1}{2}$$
$$f[0,1,2,3] = \frac{e^3 - 3e^2 + 3e^1 - 1}{6}$$

*Por tanto el polinomio interpolador de Lagrange es:*

$$P_3(x) = 1 + (e-1)\,x + \frac{e^2 - 2e + 1}{2}x(x-1) + \frac{e^3 - 3e^2 + 3e^1 - 1}{6}x(x-1)(x-2)$$

*en la siguiente gráfica se muestra la diferencia $e^x - P_3(x)$ en el intervalo $[0,3]$*




**Problema 28** ***(2 puntos)*** *Calcular el polinomio interpolador de Lagrange $P_3(x)$ de la función $f(x) = sen(x)$ en los puntos $0, \frac{\pi}{2}, \pi$ y $\frac{3\pi}{2}$ utilizando las diferencias divididas de Newton.*

**Problema 29** ***(3 puntos)*** *Calcular el polinomio interpolador de Lagrange $P_3(x)$ de la función $f(x) = 2^x$ en los puntos $0, 1, 3, 4$ utilizando las diferencias divididas de Newton. Expresar el polinomio tomando en primer lugar $x_0 = 0$, $x_1 = 1$, $x_2 = 3$ y $x_3 = 4$, y en segundo lugar $x_0 = 4$, $x_1 = 3$, $x_2 = 1$, y $x_3 = 0$.*

**Problema 30** ***(3 puntos)*** *Dada una función $f(x)$, y una secuencia de valores $x_n$, aproximar $f(x)$ por la parábola que pasa por los puntos $(x_{n-1}, f(x_{n-1}))$, $(x_{n-2}, f(x_{n-2}))$ y $(x_{n-3}, f(x_{n-3}))$, calcular posteriormente las derivadas del polinomio, y comprobar que coinciden con las fórmulas dadas en el método de Muller para el cálculo de las derivadas $f''(x_{n-1})$ y $f'(x_{n-1})$.*

**Programa 10** *Programa en fortran 77 donde se definen las funciones IDIFNEWTON que a partir del vector $X(0:N)$ de puntos de interpolación, y el vector $F(0:N)$ de valores de la función $f(x)$ en los puntos de interpolación, devuelve el vector $A(0:N)$ de coeficientes de diferencias divididas que definen el polinomio de Lagrange ( $A(K) = f[x_0, x_1, .., x_K]$ ), y la función EVDIFNEWTON(A,X,X0,N) que a partir de los coeficientes dados por el vector $A(0:N)$ y el conjunto de puntos de interpolación, devuelve el valor de la evaluación del polinomio de Lagrange en el punto $X0$.*

```
      PARAMETER(Nmax=1000)
      DIMENSION A(0:1000),F(0:1000),X(0:1000)
      PRINT *,'Introducir No. Ptos Interp.'
      READ *,N
      N=N-1
      PRINT *,'Introducir Ptos Interpol.'
      DO 1 K=0,N
1     READ *, X(K)
      PRINT *,'Introducir Valores de F()'
      DO 2 K=0,N
2     READ *,F(K)
      IF(IDIFNEWTON(A,X,F,N).EQ.1) THEN
        PRINT *,'Puntos de Interpolacion repetidos'
        STOP
      ENDIF
      PRINT *,'Coef. Polinomio'
      DO 3 K=0,N
3     PRINT *,A(K)
      PRINT *,'Test de Comprobacion'
      DO 4 K=0,N
4     PRINT *,X(K),F(K),EVDIFNEWTON(A,X,X(K),N)
      END

      FUNCTION IDIFNEWTON(A,X,F,N)
      Parameter(Nmax=1000)
      DIMENSION A(0:*),X(0:*),F(0:*),B(0:Nmax)
      DO 1 K=0,N
1     B(K)=F(K)
      A(0)=F(0)
      DO 2 K=1,N
        DO 3 L=0,N-K
          IF (X(K+L).EQ.X(L)) THEN
            IDIFNEWTON=1
            RETURN
          ENDIF
          B(L)=(B(L+1)-B(L))/(X(K+L)-X(L))
3       CONTINUE
        A(K)=B(0)
2     CONTINUE
      IDIFNEWTON=0
      END


      FUNCTION EVDIFNEWTON(A,X,X0,N)
      DIMENSION A(0:*),X(0:*)
      EVDIFNEWTON=A(N)
```

```
      DO 1 K=N-1,0,-1
1             EVDIFNEWTON=EVDIFNEWTON*(X0-
X(K))+A(K)
      END
```

**Implementación de funciones elementales.**

Una vez definida una aritmética en precisión finita y las 4 operaciones básicas (suma, resta, multiplicación, división), es necesario definir a partir de estas operaciones, las funciones elementales que todos usamos, como son : la raíz cuadrada $\sqrt{x}$, las funciones trigonométricas: $sen(x)$ $\cos(x)$, $\tan(x)$, la función $\ln(x)$, la función $e^x$ , la función $x^y$, etc...Las técnicas elementales para definir estas funciones son utilizar la interpolación polinómica, los desarrollos de Taylor y los algoritmos de búsqueda de ceros (como vimos anteriormente para la $\sqrt{x}$)

*Aproximación de la exponencial $e^x$.*

Un número real $x$, siempre se puede expresar como $x = m + x'$ donde $m$ es un número entero y $x' \in [0,1]$. Dado que

$$e^x = e^m e^{x'}$$

podemos descomponer el cálculo de $e^x$ en el cálculo, por un lado de $e^m$ donde al ser $m$ un entero el cálculo es inmediato a partir de multiplicaciones sucesivas de potencias naturales de $e$ ó $e^{-1}$ (si $m<0$). Por tanto, podemos concentrarnos en el cálculo de $e^{x'}$ para $x' \in [0,1]$. Utilizando como puntos de interpolación los asociados a los polinomios de Chebychev:

$$x_i = \frac{1}{2}\left(1 + \cos\left(\frac{2i+1}{2N+2}\pi\right)\right) \quad i = 0, ..., N$$

obtenemos que el error relativo verifica:

$$\frac{\mid e^{x'} - P_N(x) \mid}{e^{x'}} \leq \frac{e}{(N+1)!2^N}\left(\frac{1}{2}\right)^{N+1}$$

para $N = 6$ el error relativo es menor que $6.6 \times 10^{-8}$, y por tanto del mismo orden que la unidad de redondeo $u$ en una aritmética de 32 bits. Es decir que tomando un polinomio de grado $N = 6$, es decir 7 puntos de interpolación, obtenemos ya la mejor aproximación posible de $e^x$ en el intervalo $[0,1]$ en una aritmética de 32 bits.

***Práctica** 3 (Aproximación de $e^x$, 2 horas)*

Crear una función en fortran 77 que devuelva el valor de $e^x$ con $x \in [0,1]$ utilizando el polinomio de Lagrange $P_6(x)$ que interpola a $e^x$ en los puntos:

$$x_i = \frac{1}{2}\left(1 + \cos\left(\frac{2i+1}{14}\pi\right)\right) \quad i = 0, ..., 6$$

Comprobar que el polinomio esta bien construido, es decir que $P_6(x_i) = e^{x_i}$ para todos los $x_i$. Introducir por teclado un valor $x$, y evaluar e imprimir $P_6(x)$ y $e^x$. Utilizar $x = 0, 1, 0.5, 2$ y 3.

**Nota:** Utilizar las funciones de *an.h* IDIFNEWTON(.) que calcula el polinomio interpolador a partir de los puntos y valores de interpolación y EVDIFNEWTON(.) que evalua el polinomio interpolador en un punto.

*Aproximación de funciones trigonométricas*

Utilizaremos como modelo las funciones $f(x) = \cos(x)$. y $f(x) = sen(x)$. Puesto que estas funciones son $2\pi$ periódicas y utilizando algunas relaciones trigonométricas es suficiente, por ejemplo, definir las funciones $\cos(x)$ y $sen(x)$ en el intervalo $[0, \frac{\pi}{4}]$, y a partir de algunas relaciones trigonométricas definir las funciones para cualquier valor $x$ (en radianes). Efectivamente, denotemos por $\cos_{[0,\frac{\pi}{4}]}(x)$, y $sen_{[0,\frac{\pi}{4}]}(x)$ a las funciones trigonométricas definidas sobre el intervalo $[0, \frac{\pi}{4}]$. Podemos definir entonces las siguientes funciones

$$\cos_{[0,\frac{\pi}{2}]}(x) = \begin{cases} \cos_{[0,\frac{\pi}{4}]}(x) & si \quad x \leq \frac{\pi}{4} \\ sen_{[0,\frac{\pi}{4}]}(\frac{\pi}{2} - x) & si \quad x > \frac{\pi}{4} \end{cases}$$

$$sen_{[0,\frac{\pi}{2}]}(x) = \begin{cases} sen_{[0,\frac{\pi}{4}]}(x) & si \quad x \leq \frac{\pi}{4} \\ \cos_{[0,\frac{\pi}{4}]}(\frac{\pi}{2} - x) & si \quad x > \frac{\pi}{4} \end{cases}$$

$$\cos_{[0,\pi]}(x) = \begin{cases} \cos_{[0,\frac{\pi}{2}]}(x) & si \quad x \leq \frac{\pi}{2} \\ -\cos_{[0,\frac{\pi}{2}]}(\pi - x) & si \quad x > \frac{\pi}{2} \end{cases}$$

$$sen_{[0,\pi]}(x) = \begin{cases} sen_{[0,\frac{\pi}{2}]}(x) & si \quad x \leq \frac{\pi}{2} \\ sen_{[0,\frac{\pi}{2}]}(\pi - x) & si \quad x > \frac{\pi}{2} \end{cases}$$

$$\cos_{[0,2\pi]}(x) = \begin{cases} \cos_{[0,\pi]}(x) & si \quad x \leq \pi \\ \cos_{[0,\pi]}(2\pi - x) & si \quad x > \pi \end{cases}$$

$$sen_{[0,2\pi]}(x) = \begin{cases} sen_{[0,\pi]}(x) & si \quad x \leq \pi \\ -sen_{[0,\pi]}(2\pi - x) & si \quad x > \pi \end{cases}$$

El desarrollo en Serie de Taylor centrado en 0 del $\cos(x)$ es:

$$\cos(x) \approxeq P_n(x) = 1. - \frac{x^2}{2} + \frac{x^4}{4!} + .... + (-1)^n \frac{x^{2n}}{(2n)!}$$

y el error máximo cometido por el desarrollo de Taylor en un punto $x \in [0, \frac{\pi}{4}]$ es

$$\mid P_n(x) - \cos(x) \mid \leq sen(x)\frac{(x)^{2n+1}}{(2n+1)!}$$

La ventaja de utilizar el desarrollo de Taylor centrado en 0 es que las potencias impares de $x$ no aparecen lo que simplifica el cálculo numérico. El error relativo es

$$\frac{\mid P_n(x) - \cos(x) \mid}{\cos(x)} \leq \tan(x)\frac{(x)^{2n+1}}{(2n+1)!}$$

Además, como $\tan(x)$ es creciente en $[0, \frac{\pi}{4}]$ el valor máximo del error se encuentra en $x = \frac{\pi}{4}$. Por ejemplo para $n = 5$ obtenemos que el error relativo máximo cometido en $x = \frac{\pi}{4}$ es del orden de

$$\tan(\frac{\pi}{4})\frac{\left(\frac{\pi}{4}\right)^{2*5+1}}{(2*5+1)!} = 1.8 \times 10^{-9}$$

Por tanto, si trabajamos con una aritmética de 32 bits cuya unidad de redondeo $u$ es del orden de $10^{-8}$, tenemos que con $n = 5$ obtenemos una aproximación del $\cos(x)$ que es la mejor posible dentro de esta aritmética y por tanto no tendría sentido aumentar el valor de $n$.

**Problema 31** ***(3 puntos)*** *Aproximar la función $sen(x)$ en el intervalo $[0, \frac{\pi}{4}]$ utilizando el desarrollo de Taylor, y calcular el valor de $n$ a partir del cual la aproximación es la mejor posible dentro de una aritmética de 32 bits.*

**Problema 32** ***(2 puntos)*** *Demostrar que utilizando relaciones trigonométricas es posible calcular las funciones $sen(x)$ y $\cos(x)$ para cualquier $x$ (en radianes), utilizando únicamente su valor en el intervalo $[0, \frac{\pi}{8}]$.*

**Problema 33** ***(3 puntos)*** *Calcular los polinomios necesarios para interpolar las funciones trigonométricas $\cos(x)$ y $sen(x)$ en el intervalo $[0, \frac{\pi}{8}]$ en una aritmética de 32 bits*

*Aproximación de la función* $\ln(x)$

Como hemos visto anteriormente, un número $x$ real en una aritmética de precisión finita viene expresado habitualmente como

$$x = 2^m \left(\sum_{n=1}^{t} \frac{a_n}{2^n}\right)$$

donde $m$ es un número entero, $a_1 = 1$ y para $n > 1$ $a_n = 0$ ó $a_n = 1$. Por tanto el número $\left(\sum_{n=1}^{t} \frac{a_n}{2^n}\right)$ es mayor que $\frac{1}{2}$ y menor que 1. Aplicando las propiedades del $\ln(x)$ obtenemos que

$$\ln(x) = m\ln(2) + \ln\left(\sum_{n=1}^{t} \frac{a_n}{2^n}\right)$$

dado que el número $\ln(2)$ es una constante que supondremos calculada anteriormente, ( $\ln(2) \cong .6931471806$) podemos reducir el cálculo del $\ln(x)$ al rango de valores $\frac{1}{2} \leq x \leq 1$.

Utilizaremos los puntos de interpolación generados por los polinomios de Chebychev que para el intervalo $[\frac{1}{2}, 1]$ son :

$$x_i = \frac{1}{2} + \frac{1}{4}\left(1 + \cos\left(\frac{2i+1}{2N+2}\pi\right)\right) \quad i = 0, ..., N$$

dado que $\ln(1) = 0$ para minimizar el error relativo añadiremos como punto interpolante $x_{N+1} = 1$. El error de interpolación relativo entre $P_{N+1}(x)$ y $\ln(x)$ es:

$$\frac{|\ \ln(x) - P_{N+1}(x)\ |}{|\ \ln(x)\ |} = \frac{|\ (x-1)\Pi_{i=0}^{N}(x - x_i)\ |}{\xi^{N+1}(N+2)\ |\ \ln(x)\ |}$$

donde $\xi \in [\frac{1}{2}, 1]$. Además se tiene que en el intervalo $[\frac{1}{2}, 1]$

$$\frac{|\ x-1\ |}{|\ \ln(x)\ |} \leq 1$$

por tanto se tiene:

$$\frac{|\ \ln(x) - P_{N+1}(x)\ |}{|\ \ln(x)\ |} \leq \frac{2^{N+1}}{(N+2)}\left(\frac{1}{4}\right)^{N+1}\frac{1}{2^N}$$

para $N = 10$ el error máximo es $3.9736 \times 10^{-8}$ que es menor que la unidad de redondeo $u$, y por tanto, en una aritmética de 32 bits tendríamos la mejor aproximación posible de la función $\ln(x)$.

**Problema 34** ***(1 punto)*** *Como se puede obtener la función $y^x$, donde $x, y$ son números reales, utilizando las funciones $e^x$ y $\ln(x)$.*

## ANALISIS NUMERICO MATRICIAL I

En esta primera sección dedicada a la resolución de sistemas de ecuaciones lineales, estudiaremos los métodos directos clásicos para la resolución de un sistema de ecuaciones de la forma

$$Au = b$$

donde $A = (a_{i,j})$ es una matriz $NxN$, $b = (b_i)$ un vector de tamaño $N$ que determina los términos independientes, y $u = (u_i)$ es el vector solución buscado.

### Método de Gauss

Este método, aunque no es de los más rápidos, tiene la gran ventaja de que se puede aplicar a todo tipo de matrices, algo que como veremos en el futuro, no ocurre con otros métodos más rápidos, pero que requieren por ejemplo que la matriz sea simétrica o definida positiva. El método de Gauss se basa en transformar el sistema $Au = b$ en un sistema equivalentes $A'u = b'$ tal que la solución sea la misma, y que las matrices $A'$ sea triangular superior, es decir, que tenga valores nulos de la diagonal hacia abajo, una vez obtenidos la matriz $A'$ y el vector $b'$, el cálculo de la solución $u$ es inmediata siguiendo un remonte de las variables a través del siguiente esquema recursivo

$$u_N = \frac{b'_N}{a'_{N,N}}$$

$$u_k = \frac{b'_k - \sum_{l=k+1}^{N} a'_{k,l}u_l}{a'_{k,k}} \qquad k = N-1, .., 1$$

**Problema 35** ***(2 puntos)*** *Calcular el número de operaciones básicas (sumas, restas, multiplicaciones y divisiones) en función de la dimensión $N$ necesarias para realizar un remonte como el presentado arriba.*

Para obtener $A'$ y $b'$ se calcula en primer lugar el valor máximo en valor absoluto de la primera columna de $A$, (denominado pivote), a continuación se intercambian la primera fila de $A$ con la fila donde se encuentra el pivote, y se hace los mismo con el vector $b$ para que el sistema sea equivalente. A continuación se multiplica la fila de $A$ por el valor $\frac{-a_k}{a_1}$ y se suma a la fila $k-ésima$ de $A$ para $k = 2, ..., N$, se hace lo mismo para el vector $b$, con ello habremos obtenido un sistema equivalente tal que la primera columna es cero de la diagonal hacia abajo, volvemos ahora a hacer lo mismo para convertir la segunda columna cero de la diagonal para abajo y así sucesivamente hasta llegar a la mencionada matriz $A'$ .

**Ejemplo 9** *Ejemplo de descomposición según el método de Gauss. Se considera el sistema*

$$\begin{pmatrix} -2 & -2 & 0 \\ 6 & 18 & 12 \\ 3 & 11 & 7 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 24 \\ 8 \end{pmatrix}$$

*La descomposición de la matriz A lleva las siguientes fases:*

$$\begin{pmatrix} -2 & -2 & 0 \\ 6 & 18 & 12 \\ 3 & 11 & 7 \end{pmatrix} \xrightarrow{pivoteo} \begin{pmatrix} 6 & 18 & 12 \\ -2 & -2 & 0 \\ 3 & 11 & 7 \end{pmatrix}$$

$$\begin{pmatrix} 6 & 18 & 12 \\ -2 & -2 & 0 \\ 3 & 11 & 7 \end{pmatrix} \xrightarrow{ceros\ 1^a\ colu\,mna} \begin{pmatrix} 6 & 18 & 12 \\ 0 & 4 & 4 \\ 0 & 2 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 6 & 18 & 12 \\ 0 & 4 & 4 \\ 0 & 2 & 1 \end{pmatrix} \xrightarrow{ceros\ 2^a\ colu\,mna} \begin{pmatrix} 6 & 18 & 12 \\ 0 & 4 & 4 \\ 0 & 0 & -1 \end{pmatrix}$$

*de la misma forma, el vector b se ha transformado de la forma siguiente*

$$\begin{pmatrix} 0 \\ 24 \\ 8 \end{pmatrix} \rightarrow \begin{pmatrix} 24 \\ 0 \\ 8 \end{pmatrix} \rightarrow \begin{pmatrix} 24 \\ 8 \\ -4 \end{pmatrix} \rightarrow \begin{pmatrix} 24 \\ 8 \\ -8 \end{pmatrix}$$

*y el remonte da como solución* $u_1 = 6$, $u_2 = -6$, $u_3 = 8$.

**Problema 36** ***(2 puntos)*** *Resolver por el método de Gauss el sistema*

$$\begin{pmatrix} -1 & 2 \\ 2 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$

**Problema 37** ***(3 puntos)*** *Calcular el número de operaciones básicas necesarias para descomponer el sistema $Au = b$ en el sistema $A'u = b'$ utilizando el método de Gauss, y teniendo en cuenta la siguiente relación*

$$\sum_{k=1}^{M-1} k^2 = \frac{1}{3}M^3 - \frac{1}{2}M^2 + \frac{1}{6}M$$

**Problema 38** ***(2 puntos)*** *Implementar en FORTRAN la funcion $IDESCENSO(A, b, u,$ N, $N\max)$ que resuelve un sistema donde $A$ es una matriz triangular inferior, $b$ es el vector de términos independientes, $u$ el vector solución, $N$ es la dimensión real del sistema y $N\max$ la dimensión que se utilizo para reservar la memoria de la matriz $A$. La función devuelve 0 si termina correctamente y 1 en caso contrario.* ***Nota Importante****: Las líneas de código tienen que ir todas numeradas y no pueden superar las 15 líneas como máximo.*

**Problema 39** ***(2 puntos)*** *Resolver por el método de Gauss el siguiente sistema de ecuaciones*

$$\begin{pmatrix} 0 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

**Programa 11** *Programa en fortran 77 que implementa el método de Gauss. Se define una función IGAUSS que tiene como parámetros la matriz A, el vector independiente b, un vector auxiliar Nrow, la dimensión del sistema, y la dimensión máxima admitida. La función devuelve un valor entero M que indica si se ha terminado correctamente ($M = 0$) o incorrectamente ($M = 1, 2$), en el caso en que se ha terminado correctamente la solución se devuelve en el propio vector b.*

```
      Parameter(Nmax=1000)
      DIMENSION A(Nmax,Nmax),B(Nmax),Nrow(Nmax)
      PRINT *, 'Introducir Dimension'
      READ *,N
      PRINT *,'Introducir matriz'
      DO 2 K=1,N
        DO 2 L=1,N
2     READ *,A(K,L)
      PRINT *,'Introducir vector'
      DO 3 K=1,N
3     READ *,B(K)
      M=IGAUSS(A,B,N,Nrow,Nmax)
      IF(M.EQ.0) THEN
        PRINT *,'SOLUCION'
        DO 1 K=1,N
          PRINT *,B(K)
1       CONTINUE
      ELSE IF(M.EQ.1)THEN
```

```
      PRINT *,'Una columna es toda cero'
    ELSE
      PRINT *,'A(N,N)=0'
    ENDIF
    END

    FUNCTION IGAUSS(A,B,N,Nrow,Nmax)
    PARAMETER (NmaxGAUSS=1000)
    DIMENSION A(Nmax,*),B(*),Nrow(*)
    DIMENSION U(NmaxGAUSS)
    IF (N.GT .NmaxGAUSS) THEN
      IGAUSS=3
      PRINT *,'DIMENSION DEL SISTEMA MAYOR
DE LA PERMITIDA'
      RETURN
    ENDIF
    DO 1 K=1,N
1   Nrow(K)=K
    DO 2 K=1,N-1
      XMax=ABS(A(Nrow(K),K))
      M=K
      DO 3 L=K+1,N
        IF(ABS(A(Nrow(L),K)).GT.XMax) THEN
          XMax=ABS(A(Nrow(L),K))
          M=L
        ENDIF
3     CONTINUE
      IF(XMax.LT.(2.**(-100))) THEN
        IGAUSS=1
        RETURN
      ENDIF
      IF(K.NE.M) THEN
        MP=Nrow(K)
        Nrow(K)=Nrow(M)
        Nrow(M)=MP
        ENDIF
      DO 4 L=K+1,N
        C=A(Nrow(L),K)/A(Nrow(K),K)
         DO 5 M=K,N
          A(Nrow(L),M)=A(Nrow(L),M)-
C*A(Nrow(K),M)
5       CONTINUE
        B(Nrow(L))=B(Nrow(L))-C*B(Nrow(K))
4     CONTINUE
2   CONTINUE
    IF(ABS(A(Nrow(N),N)).LT.(2.**(-100))) THEN
      IGAUSS=2
      RETURN
    ENDIF
    U(N)=B(Nrow(N))/A(Nrow(N),N)
    DO 6 K=N-1,1,-1
      C=0
        DO 7 L=K+1,N
        C=C+A(Nrow(K),L)*U(L)
7     CONTINUE
      U(K)=(B(Nrow(K))-C)/A(Nrow(K),K)
6   CONTINUE
    DO I=1,N
      B(I)=U(I)
    ENDDO
    IGAUSS=0
    END
```

**Estimación del error de un método para resolver sistemas.**

Para estimar la fiabilidad de la solución numérica de un sistema de ecuaciones haremos lo siguiente, dada una matriz $A$, un vector de términos independientes $b$ y un vector solución $u$ calculado utilizando alguna técnica numérica, tendremos que si la solución es perfecta entonces $Au - b = 0$, ahora bien, esto no suele suceder porque los errores de redondeo y de cálculo, producen que esta estimación no sea exacta, para estimar el error cometido al resolver el sistema utilizaremos la expresión siguiente, donde $e$ es el vector $e = Au - b$

$$Error\,\mathrm{Si}\,stema = \frac{1}{N}\sum \frac{|e_i|}{|b_i|+1}$$

donde $N$ es la dimensión del sistema, $Error\,\mathrm{Si}\,stema$ representa el error relativo medio al resolver el sistema, en el denominador se añade 1 para evitar las posibles divisiones por 0. Cuanto más pequeño sea $Error\,\mathrm{Si}\,stema$, mejor aproximada estará la solución del sistema.

**Método de Cholesky**

Este método sólo se puede aplicar a matrices simétricas y definidas positivas. El siguiente teorema da 3 posibles definiciones equivalentes de una matriz definida positiva.

**Teorema 13** *Sea $A$ una matriz simétrica, las 3 siguientes afirmaciones son definiciones equivalentes de que una matriz sea definida positiva*

*(i)* $\forall\ v \in \Re^N - \{0\}$ *se cumple* ${}^tvAv > 0$

*(ii) todos los autovalores $\lambda$ de $A$ son positivos.*

*(iii) Los determinantes de todos los menores principales de $A$ son positivos.*

El método de Cholesky se basa en descomponer la matriz $A$ en la forma:

$$A = B \cdot {}^t B$$

donde $B$ es una matriz triangular inferior

$$B = \begin{pmatrix} b_{1,1} & 0 & 0 & . & 0 \\ b_{2,1} & b_{2,2} & 0 & . & . \\ b_{3,1} & b_{3,2} & b_{3,3} & . & . \\ . & . & . & . & 0 \\ b_{n,1} & b_{n,2} & b_{n,3} & . & b_{n,n} \end{pmatrix}$$

**Problema 40** ***(2 puntos)*** *Demostrar que si $A = B \cdot B^t$ ($B$ triangular inferior) y $|B| \neq 0$, entonces $A$ es simétrica y definida positiva*

**Problema 41** ***(2 puntos)*** *Descomponer la siguiente matriz $A$ por el método de Cholesky*

$$A = \begin{pmatrix} 1 & 1 & 4 \\ 1 & 5 & 6 \\ 4 & 6 & 26 \end{pmatrix}$$

De forma general, el algoritmo para calcular $B$ es el siguiente

Para $i = 1, ..., N$
  $b_{i,i} = \sqrt{\left(a_{i,i} - \sum_{k=1}^{i-1} b_{i,k}^2\right)}$
  Para $j = i+1, ..., N$
    $b_{j,i} = \frac{1}{b_{i,i}} \left(a_{j,i} - \sum_{k=1}^{i-1} b_{j,k} b_{i,k}\right)$
  Fin Para $j$
Fin Para $i$

El interés de descomponer una matriz $A$ por el método de Cholesky, es que a continuación es muy sencillo resolver el sistema de ecuaciones $Au = b$. Efectivamente, basta descomponer el sistema de la siguiente forma

$$\begin{aligned} Bz &= b \\ {}^tBu &= z \end{aligned}$$

y ambos sistemas se resuelvan rápidamente haciendo un remonte y un descenso.

**Nota:** Normalmente para evitar tener que almacenar dos matrices, una para $B$ y otra para $B^t$, se almacena todo en una única matriz $B$, simétrica, escribiendo en la parte triangular superior de $B$ la parte correspondiente a $B^t$.

**Problema 42** ***(2 puntos)*** *Calcular el número de operaciones necesarias para resolver un sistema por el método de Cholesky.*

**Problema 43** ***(2 puntos)*** *Demostrar que a partir de un método para resolver sistemas de ecuaciones se puede construir de forma inmediata un método para calcular la inversa $A^{-1}$ de una matriz $A$.*

**Práctica 4 (Método de Cholesky, 6 horas).**

Implementar en fortran 77 las siguientes funciones :

- FUNCTION ICHOLESKY_FACTORIZACION (A,DB,N,Nmax) : Calcula la descomposición de Cholesky de A y la devuelve en la matriz DB, devuelve un 0 si termina bien y $-1$ en caso contrario.
- FUNCTION IDESCENSO(DB,DZ,B,N,Nmax) : Resuelve un sistema triangular inferior donde DB es la matriz, B es el término independiente, y DZ es el vector donde devuelve la solución, devuelve un 0 si termina bien y $-1$ en caso contrario.
- FUNCTION IREMONTE(DB,DU,DZ,N,Nmax) : Resuelve un sistema triangular superior donde DB es la matriz, DZ es el término independiente, y DU es el vector donde devuelve la solución, devuelve un 0 si termina bien y $-1$ en caso contrario
- FUNCTION ERROR_SISTEMA(A,DU,B,N,Nmax) : Devuelve el error al resolver el sistema dado por la expresión ErrorSistema de la sección anterior.
- FUNCTION ICHOLESKY(A,B,DU,N,Nmax) : Resuelve un sistema por el método de Cholesky y devuelve la solución en DU, devuelve un 0 si termina bien y $-1$ en caso contrario.

Hay que hacer una versión en simple precisión, y otra versión en doble precisión donde todas las variables que empiecen por D sean dobles (La matriz A y el vector B siempre serán de simple precisión). Hay que resolver los sistemas ejemplos y calcular el ErrorSistema tanto en doble como en simple precisión.
**Nota:** El programa debe permitir introducir el sistema directamente por teclado, o desde disco duro utilizando las funciones definidas en *an.h*. Resolver los siguientes sistemas ejemplo:

1. $\begin{pmatrix} 1 & 1 & 4 \\ 1 & 5 & 6 \\ 4 & 6 & 26 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 12 \\ 36 \end{pmatrix}$
2. $\begin{pmatrix} 1 & 1 & 4 \\ 1 & 1 & 4 \\ 4 & 4 & 17 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 12 \\ 36 \end{pmatrix}$
3. $\begin{pmatrix} 1 & 1 & 4 \\ 1 & 5 & 6 \\ 4 & 6 & 17 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 12 \\ 36 \end{pmatrix}$
4. $\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -5 \end{pmatrix}$

Resolver también los sistemas que aparecen en el directorio */users/asignaturas/ii-an* de la máquina *serdis.dis.ulpgc.es*. En este directorio hay tres ejemplos de sistemas de dimensión 10, 100, y 500 respectivamente. Los 3 sistemas corresponden a matrices simétricas y definidas positivas. Estos archivos ejemplos sólo se pueden utilizar al compilar el programa en serdis bajo UNIX, si utilizamos Linux, el programa no reconocerá el formato de los archivos.

### Método de Crout para matrices tridiagonales

El caso de sistemas de ecuaciones con matrices $A$ tridiagonales posee una forma especialmente simple de factorización, vamos a descomponer $A$ en el producto de dos matrices triángulares de la forma siguiente

$$\begin{pmatrix} a_1 & b_1 & . & 0 \\ c_1 & a_2 & . & 0 \\ 0 & . & . & b_{N-1} \\ 0 & . & c_{N-1} & a_N \end{pmatrix} =$$

$$\begin{pmatrix} l_1 & 0 & . & 0 \\ m_1 & l_2 & . & 0 \\ 0 & . & . & 0 \\ 0 & . & m_{N-1} & l_N \end{pmatrix} \begin{pmatrix} 1 & u_1 & . & 0 \\ 0 & 1 & . & 0 \\ 0 & . & . & u_{N-1} \\ 0 & . & 0 & 1 \end{pmatrix}$$

los vectores $m_i$, $l_i$, y $u_i$ se calculan utilizando el esquema:

$l_1 = a_1$
$u_1 = \frac{b_1}{l_1}$
Para $i = 2, .., N-1$
  $m_{i-1} = c_{i-1}$
  $l_i = a_i - m_{i-1}u_{i-1}$
  $u_i = \frac{b_i}{l_i}$
Fin Para
$m_{N-1} = c_{N-1}$
$l_N = a_N - m_{N-1}u_{N-1}$

**Problema 44** ***(3 puntos)*** *Demostrar el algoritmo de Crout para descomponer matrices tridiagonales.*

**Problema 45** ***(2 puntos)*** *Resolver utilizando el método de Crout el siguiente sistema de ecuaciones*

$$\begin{pmatrix} 2 & 4 & 0 \\ -1 & 0 & 4 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \\ -1 \end{pmatrix}$$

**Problema 46** ***(2 puntos)*** *Calcular el número de operaciones necesarias para resolver un sistema tridiagonal por el método de Crout.*

### Subrutinas en fortran 77 para la lectura y escritura en disco de vectores y matrices.

**Programa 12** *subrutinas en fortran 77 que leen y escriben en disco vectores y matrices.*

```
****** Función que lee un vector de disco de nombre String, lo almacena
****** en la tabla Vector, y devuelve la dimension del vector
      FUNCTION LeerVector(String,Vector)
      CHARACTER * (*) String
      DIMENSION Vector(*)
      OPEN(1,FILE=String,STATUS='OLD',
form='UNFORMATTED')
      READ(1) A
      Ndimension=INT(A)
      PRINT *,'DIMENSION DEL VECTOR
',String,'=',Ndimension
      DO 1 K=1,Ndimension
      READ(1) Vector(K)
1     CONTINUE
      CLOSE(1)
      LeerVector=Ndimension
      END


***** Procedimiento que escribe en el fichero String el vector Vector
***** de dimension Ndimension
      SUBROUTINE EscribirVector(String,Vector,Ndimension)
      CHARACTER * (*) String
      DIMENSION Vector(*)
      OPEN(1,FILE=String, STATUS='NEW',
form='UNFORMATTED',ERR=1)
      GOTO 2
1     OPEN(1,FILE=String, STATUS='OLD',
form='UNFORMATTED')
2     A=Ndimension
      WRITE(1) A
      DO 3 K=1,Ndimension
        WRITE(1) Vector(K)
3     CONTINUE
      CLOSE(1)
      END


      ****** Funcion que lee una matriz cuadrada de disco de nombre String, lo almacena
****** en la tabla A, y devuelve la dimension de la matriz
      FUNCTION LeerMatriz(String,A,Nmax)
      CHARACTER * (*) String
      DIMENSION A(Nmax,*)
      OPEN(1,FILE=String, STATUS='OLD',
form='UNFORMATTED')
      READ(1) B
      Ndimension=INT(B)
      PRINT *,'DIMENSION DE LA MATRIZ ',
String,'=', Ndimension
      DO 1 K=1,Ndimension
        DO 2 L=1,Ndimension
          READ(1) A(K,L)
2       CONTINUE
1     CONTINUE
      CLOSE(1)
      LeerMatriz=Ndimension
      END
```

```
***** Procedimiento que escribe en el fichero String la matriz A
***** cuadrada de dimension Ndimension
      SUBROUTINE EscribirMatriz(String,A,Ndimension,Nmax)
      CHARACTER * (*) String
      DIMENSION A(Nmax,*)
      OPEN(1,FILE=String, STATUS='NEW', form='UNFORMATTED', ERR=1)
      GOTO 2
1     OPEN(1, FILE=String, STATUS='OLD', form='UNFORMATTED')
2 B=Ndimension
      WRITE(1) B
      DO 3 K=1,Ndimension
        DO 4 L=1,Ndimension
          WRITE(1) A(K,L)
4       CONTINUE
3     CONTINUE
      CLOSE(1)
      END
```

**Programa 13** *Programa en fortran 77 donde se describe un ejemplo de lectura/escritura de vectores y matrices.*

```
      INCLUDE 'an.h'
      PARAMETER(Nmax=100)
      CHARACTER * 10,String
      DIMENSION V(Nmax), A(Nmax,Nmax)

      String='vector.dat'
      Ndimension=LeerVector(String,V)
      PRINT *,Ndimension
      DO 1 K=1,Ndimension
        PRINT *,V(K)
        V(K)=2*V(K)
1     CONTINUE
      CALL EscribirVector('vector3.dat',V,Ndimension)
      N=LeerVector('vector3.dat',V)
      PRINT *,'N=',N
      DO 2 k=1,N
        PRINT *,V(K)
2     CONTINUE


      Ndimension=3
      DO 3
        K=1,Ndimension
        DO 4 L=1,Ndimension
          A(L,K)=L+K
          PRINT *,A(L,K)
4       CONTINUE
        PRINT *
3     CONTINUE
      CALL EscribirMatriz('matriz.dat',A,Ndimension,Nmax)
      Ndimension=LeerMatriz('matriz.dat',A,Nmax)
      PRINT *,Ndimension
      DO 5 K=1,Ndimension
        DO 6 L=1,Ndimension
          PRINT *,A(L,K)
6       CONTINUE
        PRINT *
5     CONTINUE
      END
```

**Nota** la declaración

$INCLUDE\ 'an.h'$

incluye el fichero an.h que se encuentra en el directorio de trabajo, en el cuerpo del programa. La declaración

$CHARACTER\ *\ 10, String$

define un string de caracteres de tamaño 10.

## ANALISIS NUMERICO MATRICIAL II

En esta sección veremos algunos aspectos más avanzados del análisis matricial incluyendo técnicas iterativas de resolución de sistemas de ecuaciones y cálculo de autovalores.

### Normas de vectores y matrices.

**Definición 2** *Una norma $\| \cdot \|$ es una aplicación de un espacio vectorial $E$ en $R^+ \cup \{0\}$ verificando las siguientes propiedades:*

- $\| x \| = 0$ *si sólo si* $x = 0$
- $\| \lambda x \| = | \lambda | \| x \|$ *para todo* $\lambda \in K$ *y* $x \in E$
- $\| x + y \| \leq \| x + y \|$ *para todo* $x, y \in E$.

Básicamente una norma mide la magnitud o tamaño de un vector $x$. Por ejemplo en el espacio vectorial de los números reales, la norma "natural" es el valor absoluto. Sin embargo, cuando trabajamos en varias dimensiones, esto es $x = (x_1, x_2, ...., x_N)$ existen múltiples formas de definir una norma. La definición más utilizada es la denominada norma $p$, donde $p$ es un número real positivo, definida por

$$\| x \|_p = \left( \sum_{i=1}^{N} | x_i |^p \right)^{\frac{1}{p}}$$

un caso particularmente interesante es $p = 2$ que corresponde a la norma euclídea. Otro caso interesante es cuando hacemos $p$ tender hacia infinito, lo que da lugar a la denominada norma infinito, definida por

$$\| x \|_\infty = \max_i | x_i |$$

**Problema 47** ***(4 puntos)*** *Tomar $N = 2$ , p=2, y demostrar que la norma $\| x \|_p$ verifica las propiedades de la definición de norma*

**Problema 48** ***(3 puntos)*** *Demostrar que*

$$Lim_{p\to\infty} \parallel x \parallel_p = \max_i \mid x_i \mid$$

**Problema 49** ***(2 puntos)*** *Tomar $N = 2$, y dibujar el lugar geométrico de los vectores $x = (x_1, x_2)$ que verifican*

1. $\parallel x \parallel_1 < 1$
2. $\parallel x \parallel_2 < 1$
3. $\parallel x \parallel_\infty < 1$

**Problema 50** ***(2 puntos)*** *Tomar $N = 2$ y demostrar la siguiente desigualdad*

$$\parallel x \parallel_\infty \leq \parallel x \parallel_2 \leq \parallel x \parallel_1$$

Dada una matriz $A$ de dimensión $NxN$, se podría definir su norma considerando la matriz como un vector de dimensión $NxN$. Sin embargo, resulta más útil, definir la norma de una matriz subordinándola a la norma de un vector de la siguiente manera:

**Definición 3** *Sea $A$ una matriz, y $\parallel . \parallel$ una norma vectorial. Se define la norma de $A$, subordinada a la norma vectorial $\parallel . \parallel$ como*

$$\parallel A \parallel = \sup_{x\neq 0} \frac{\parallel Ax \parallel}{\parallel x \parallel}$$

*La propiedad fundamental que verifica una norma matricial definida de esta forma es la siguiente:*

**Teorema 14** *Sea $A$ una matriz y $\parallel . \parallel$ una norma vectorial. Entonces, para cualquier vector $x$ se verifica*

$$\parallel Ax \parallel \leq \parallel A \parallel \cdot \parallel x \parallel$$

**Demostración:** Si $x = 0$ la desigualdad es trivial. Si $x \neq 0$ entonces, puesto que $\parallel x \parallel > 0$ la desigualdad anterior es equivalente a

$$\frac{\parallel Ax \parallel}{\parallel x \parallel} \leq \parallel A \parallel$$

ahora bien esta desigualdad es cierta por al propia definición de $\parallel A \parallel$

**Problema 51** ***(2 puntos)*** *Demostrar que si $A, B$ son dos matrices de dimensión $NxN$, entonces para cualquier norma de matrices subordinada a una norma vectorial se verifica*

$$\parallel AB \parallel \leq \parallel A \parallel \cdot \parallel B \parallel$$

A continuación veremos la relación que existe entre la norma de una matriz y sus autovalores. Empezaremos recordando algunos conceptos relacionados con los autovalores.

**Definición 4** *un autovalor de $A$ es un número $\lambda$ real o complejo tal que existe un vector $x$, denominado autovector tal que*

$$Ax = \lambda x$$

**Definición 5** *Se denomina polinomio característico $P(\lambda)$ de la matriz $A$, al polinomio dado por el determinante*

$$P(\lambda) = \mid A - \lambda I \mid$$

**Problema 52** ***(1 punto)*** *Demostrar que los autovalores de $A$ son los ceros del polinomio característico $P(\lambda)$.*

**Definición 6** *Se define el radio espectral de una matriz $A$ como*

$$\rho(A) = \max_i \{ \mid \lambda_i \mid : \ \lambda_i \ autovalor \ de \ A \}$$

**Teorema 15** *Sea $A$ una matriz y $\parallel . \parallel$ una norma vectorial. Entonces*

$$\parallel A \parallel \geq \rho(A)$$

**Demostración:** Si $\lambda$ es un autovalor de $A$ entonces existe un autovector $x$ tal que $Ax = \lambda x$, por tanto

$$\frac{\parallel Ax \parallel}{\parallel x \parallel} = \frac{\parallel \lambda x \parallel}{\parallel x \parallel} = |\lambda| \leq \parallel A \parallel$$

lo que demuestra el teorema

**Teorema 16** *Si los autovectores de una matriz $A$ de dimensión $NxN$ forman una base ortonormal de $R^N$ , entonces*

$$\parallel A \parallel_2 = \rho(A)$$

**Demostración:**Recordamos en primer lugar que una base ortonormal de vectores es un conjunto de vectores tales que cualquier otro vector se puede poner como combinación lineal de ellos, y además su producto escalar verifica

$$(x_i, x_j) = \sum_{k=1}^{N} (x_i)_k (x_j)_k = \begin{cases} 0 & si \quad i \neq j \\ 1 & si \quad i = j \end{cases}$$

donde $(x_i)_k$ indica la coordenada $k$ del vector $x_i$. Vamos a demostrar la desigualdad

$$\parallel A \parallel_2 \leq \rho(A)$$

dado que el teorema anterior determina la desigualdad en el otro sentido, tendríamos la igualdad, y por tanto el resultado del Teorema. Sea $x$ un vector cualquiera, puesto que $A$ posee una base ortonormal de autovectores $x_i$, el vector $x$ se podrá expresar como una combinación lineal de autovectores de la forma:

$$x = \eta_1 x_1 + \eta_2 x_2 + .. + \eta_N x_N$$

al hacer $Ax$ obtenemos, puesto que los $x_i$ son autovectores

$$Ax = \eta_1 \lambda_1 x_1 + \eta_2 \lambda_2 x_2 + .. + \eta_N \lambda_N x_N$$

como los autovectores son ortonormales obtenemos que

$$\| \quad x \|_2 = \sqrt{(\eta_1)^2 + .. + (\eta_N)^2}$$
$$\| \quad Ax \|_2 = \sqrt{(\eta_1 \lambda_1)^2 + .. + (\eta_N \lambda_N)^2}$$

y por tanto

$$\frac{\| Ax \|_2}{\| x \|_2} \leq \rho(A)$$

para cualquier vector $x$, y en consecuencia al tomar el supremo en $x$ la desigualdad se mantiene, lo que demuestra que

$$\| A \|_2 \leq \rho(A)$$

**Teorema 17** *Si una matriz $A$ de dimensión $NxN$ es simétrica, entonces todos sus autovalores son reales, y además sus autovectores forman una base ortonormal de $R^N$*

**Demostración:** [La-Th] Pg. 53

**Problema 53** ***(2 puntos)*** *Calcular los autovectores de la matriz*

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

*y determinar una base ortonormal de $R^3$ de autovectores de $A$.*

**Teorema 18** *Sea $A$ una matriz cualquiera entonces*

- $\| A \|_2 = \sqrt{\rho({}^tAA)}$
- $\| A \|_1 = \max_j \left(\sum_i \mid a_{ij} \mid\right)$
- $\| A \|_\infty = \max_i \left(\sum_j \mid a_{ij} \mid\right)$

**Demostración:** [La-Th] Pg. 73,75,

**Problema 54** ***(2 puntos)*** *Calcular las normas 2, 1 e infinito de la matriz*

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

**Problema 55** ***(2 puntos)*** *Demostrar la siguiente igualdad:*

$$\rho({}^tAA) = \rho(A \cdot {}^tA)$$

**Teorema 19** *Sea $A$ una matriz cualquiera entonces*

$$Lim_{n\to\infty} \| A^n \| = 0 \Longleftrightarrow \rho(A) < 1$$

**Demostración:**[La-Th] Pg. 80

### Condicionamiento de una matriz.

El condicionamiento de una matriz es un número que nos va a indicar la "bondad" o buen comportamiento numérico de la matriz cuando se trabaja con ella numéricamente. Para ilustrar de que estamos hablando veamos el siguiente ejemplo:

**Ejemplo 10** *Consideremos el siguiente sistema de ecuaciones*

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ v \end{pmatrix} = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$$

*cuya solución es $(1,1,1,1)$. Vamos a considerar ahora el mismo sistema, perturbando ligeramente el término independiente:*

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ v \end{pmatrix} = \begin{pmatrix} 32.1 \\ 22.9 \\ 33.1 \\ 30.9 \end{pmatrix}$$

*la solución de este sistema es $(9.2, -12.6, 4.5, -1.1)$. Como podemos observar, a pesar de que la perturbación del sistema es del orden de $0.1$, la perturbación de la solución del sistema puede llegar a ser del orden de $13.6$.*

Consideremos de forma genérica un sistema de ecuaciones de la forma

$$Au = b$$

y al mismo tiempo, el sistema de ecuaciones perturbado

$$A(u + \delta u) = b + \delta b$$

Nosotros queremos controlar el error relativo en la solución del sistema a partir del error relativo en el término independiente $b$. Es decir queremos encontrar una estimación del tipo

$$\frac{\| \delta u \|}{\| u \|} \leq \chi(A) \frac{\| \delta b \|}{\| b \|}$$

donde $\chi(A)$ es un número que llamaremos condicionamiento de la matriz. Obviamente, cuanto más pequeño sea $\chi(A)$, mejor comportamiento numérico tendrá la matriz $A$.

**Teorema 20** *Si definimos*

$$\chi(A) = \| A \| \cdot \| A^{-1} \|$$

*entonces*

$$\frac{\| \delta u \|}{\| u \|} \leq \chi(A) \frac{\| \delta b \|}{\| b \|}$$

**Demostración:** Como $A(u + \delta u) = b + \delta b$ y $Au = b$ se obtiene que $A\delta u = \delta b$, de donde $\delta u = A^{-1}\delta b$ y por tanto

$$\|\delta u\| \leq \left\|A^{-1}\right\| \|\delta b\|$$

por otro lado también se cumple:

$$\|b\| = \|Au\| \leq \|A\| \|u\|$$

de donde obtenemos

$$\frac{1}{\|u\|} \leq \frac{\|A\|}{\|b\|}$$

por tanto multiplicando esta desigualdad con la anteriormente obtenida para $\|\delta u\|$ concluimos la demostración del teorema.

**Problema 56** ***(2 puntos)*** *Demostrar que si los autovectores de una matriz $A$ de dimensión $NxN$ forman una base ortonormal de $R^N$, entonces para la norma 2 se cumple:*

$$\chi(A) = \| A \|_2 \cdot \| A^{-1} \|_2 = \frac{\max_i\{| \lambda_i |\}}{\min_i\{| \lambda_i |\}}$$

**Problema 57** ***(2 puntos)*** *Calcular el condicionamiento para la norma 2, de las siguientes matrices:*

$$A = \begin{pmatrix} 2 & 2 & -2 \\ 2 & 1 & 1 \\ -2 & 1 & 1 \end{pmatrix}$$

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

**Cálculo de autovalores y autovectores.**

En esta sección veremos algunos métodos elementales para el cálculo de autovalores y autovectores de matrices.

*Método de Jacobi.*

Este método se aplica a matrices reales y simétricas. Se basa en el hecho de que dadas dos matrices $A, R$ se verifica que los autovalores de $A$ son los mismos que los autovalores de $R^{-1}AR$. Este método intenta diagonalizar $A$ realizando transformaciones del tipo $R^{-1}AR$.

**Problema 58** ***(2 puntos)*** *Sean las matrices $A, R$. Demostrar que la matriz $A$, y la matriz $B = R^{-1}AR$ poseen los mismos autovalores*

**Problema 59** ***(2 puntos)*** *Se considera la matriz*

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

*calcular el ángulo $\alpha$ tal que la matriz*

$$R = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}$$

*verifique la matriz $B = R^{-1}AR$ sea diagonal.*

En el método de Jacobi se utilizan las denominadas matrices de rotación que tienen la forma:

$$R_{pq}(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & . & . & . & . & 0 \\ 0 & . & \cos\alpha & . & \sin\alpha & . & 0 \\ 0 & . & . & 1 & . & . & 0 \\ 0 & . & -\sin\alpha & . & \cos\alpha & . & 0 \\ 0 & . & . & . & . & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

donde los cósenos y senos están situados en las columnas y filas $p$ y $q$. Al ser una matriz de rotación se verifica $(R_{pq}(\alpha))^{-1} = {}^t R_{pq}(\alpha)$. Al realizar la operación $A' = {}^t R_{pq}(\alpha) A R_{pq}(\alpha)$ sólo se ven afectadas las filas y columnas de índices $p$ y $q$. Además la matriz $A'$ también es simétrica. Concretamente, si $A$ es una matriz simétrica, los cambios que se producen en $A'$ son:

$$\begin{aligned} a'_{pq} &= \frac{(a_{pp} - a_{qq})}{2} \sin 2\alpha + a_{pq} \cos 2\alpha \\ a'_{pp} &= a_{pp} \cos^2\alpha + a_{qq} \sin^2\alpha - a_{pq} \sin 2\alpha \\ a'_{qq} &= a_{pp} \sin^2\alpha + a_{qq} \cos^2\alpha + a_{pq} \sin 2\alpha \\ a'_{pj} &= a_{pj} \cos\alpha - a_{qj} \sin\alpha \quad j \neq p, q \\ a'_{qj} &= a_{pj} \sin\alpha + a_{qj} \cos\alpha \quad j \neq p, q \end{aligned}$$

El método de Jacobi se basa en ir modificando la matriz $A$ mediante el procedimiento anterior, haciendo 0 los elementos no diagonales más grandes en módulo. Para anular un valor $a'_{pq}$ basta elegir $\alpha$ tal que

$$\frac{(a_{pp} - a_{qq})}{2} \sin 2\alpha + a_{pq} \cos 2\alpha = 0$$

es decir

$$\cot(2\alpha) = \frac{\cos 2\alpha}{\sin 2\alpha} = \frac{(a_{qq} - a_{pp})}{2a_{pq}}$$

**Ejemplo 11** *Consideremos la matriz*

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

*Para convertir en* 0 *el elemento* $a_{12} = -1$ *de la matriz debemos elegir* $\alpha$ *tal que*

$$\cot(2\alpha) = \frac{(a_{22} - a_{11})}{2a_{12}} = 0$$

*de donde* $\alpha = -\frac{\pi}{4}$. *Por tanto la matriz* $R_{12}$ *es*

$$R_{12} = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

*y al hacer la operación* $R_{12}^t A R_{12}$ *obtenemos*

$$\begin{pmatrix} 1.0 & 0 & -.707\,107 \\ 0 & 3.0 & -.707\,107 \\ -.707\,107 & -.707\,107 & 2.0 \end{pmatrix}$$

para evitar tener que estar evaluando funciones trigonométricas que son costosas computacionalmente, y simplificar el algoritmo, podemos apoyarnos en las igualdades. trigonométricas dadas en el siguiente problema:

**Problema 60** ***(3 puntos)*** *Demostrar las siguientes igualdades trigonométricas*

$$\tan(\alpha) = -\cot(2\alpha) + sign(\cot(2\alpha))\sqrt{1 + \cot^2(2\alpha)}$$

*donde* $\alpha \in \left(-\frac{\pi}{4}, \frac{\pi}{4}\right)$, $sign(x) = 1$ *si* $x \geq 0$ *y* $sign(x) = -1$ *si* $x < 0$,

$$\begin{aligned} \cos\alpha &= \frac{1}{\sqrt{1 + \tan^2(\alpha)}} \\ \sin\alpha &= \tan(\alpha)\cos\alpha \end{aligned}$$

$$\cot(2\alpha) = \frac{-\tan(\alpha) + \sin(2\alpha)}{2\sin^2(\alpha)}$$

Utilizando las anteriores igualdades trigonométricas, la transformación de la matriz $A$ mediante el método de Jacobi se puede escribir como:

$$\begin{aligned} a'_{pq} &= 0 \\ a'_{pp} &= a_{pp} - \tan(\alpha)a_{pq} \\ a'_{qq} &= a_{qq} + \tan(\alpha)a_{pq} \\ a'_{pj} &= a_{pj}\cos\alpha - a_{qj}\sin\alpha \quad j \neq p, q \\ a'_{qj} &= a_{pj}\sin\alpha + a_{qj}\cos\alpha \quad j \neq p, q \end{aligned}$$

**Problema 61** ***(3 puntos)*** *Dentro del método de Jacobi para el cálculo de autovalores demostrar las igualdades*

$$\begin{aligned} a'_{pq} &= 0 \\ a'_{pp} &= a_{pp} - \tan(\alpha)a_{pq} \\ a'_{qq} &= a_{qq} + \tan(\alpha)a_{pq} \\ a'_{pj} &= a_{pj}\cos\alpha - a_{qj}\sin\alpha \quad j \neq p, q \\ a'_{qj} &= a_{pj}\sin\alpha + a_{qj}\cos\alpha \quad j \neq p, q \end{aligned}$$

Veamos ahora la convergencia del método de Jacobi para el cálculo de autovalores.

**Teorema 21** *Sea una matriz* $A$ *simétrica, Sea* $A^1 = A$, *y* $A^k$ *la matriz transformada de* $A^{k-1}$, *haciendo cero el elemento no diagonal más grande en módulo de la matriz* $A^{k-1}$, *entonces los elementos diagonales de la matriz* $A^k$ *convergen* $(k \to \infty)$ *hacia los autovalores de la matriz* $A$. *Además los elementos no diagonales de* $A$ *convergen hacia* 0.

**Demostración:** [La-The] Pg. 576-577.

**Algoritmo del Método de Jacobi para el Cálculo de autovalores.**

Los parámetros de entrada son la matriz simétrica $A$, su dimensión $DIM$, el número máximo de iteraciones $Nmax$ y la tolerancia $TOL$ para decidir cuando son ceros los elementos no diagonales.

PARA $n = 1, .., N$ max HACER
  $p = 2$
  $q = 1$
  $R = ABS(A(p, q))$
  PARA $i = 3, ..., DIM$ HACER
    PARA $j = 1, ..., i - 1$ HACER
      IF $ABS(A(i, j)) > R$ HACER
        $R = ABS(A(i, j))$
        $p = j$
        $q = i$
      FIN IF
    FIN PARA $j$
  FIN PARA $i$
  IF $R < TOL$ HACER
    PROCEDIMIENTO TERMINADO CORRECTAMENTE.
    LOS AUTOVALORES SE ENCUENTRAN EN LA DIAGONAL DE $A$.
    SALIR
  FIN IF
  $C = (A(q, q) - A(p, p))/(2 * A(p, q))$
  IF $C < 0$ HACER
    $T = -C - SQRT(1. + C * C)$
  ELSE
    $T = -C + SQRT(1. + C * C)$
  FIN IF

$CO = 1./SQRT(1. + T * T)$
$SI = CO * T$
PARA $j = 1, ..., DIM$ HACER
IF ( $j \neq p$ AND $j \neq q$) HACER
$D = A(p, j)$
$A(j, p) = A(p, j) = CO * D - SI * A(q, j)$
$A(j, q) = A(q, j) = SI * D + CO * A(q, j)$
FIN IF
FIN PARA $j$
$A(p, p) = A(p, p) - T * A(p, q)$
$A(q, q) = A(q, q) + T * A(p, q)$
$A(p, q) = A(q, p) = 0$
FIN PARA $n$
PROCEDIMIENTO TERMINADO INCORRECTAMENTE
NUMERO DE ITERACIONES MAXIMO EXCEDIDO

**Problema 62** ***(3 puntos)*** *Utilizar el método de Jacobi para aproximar los autovalores de la matriz:*

$$A = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

**Nota.** Para no tener que buscar en cada paso el máximo de los elementos no-diagonales de $A^k$, el algoritmo de Jacobi se puede modificar haciendo cero el primer elemento $a_{pq}$ que se encuentre que verifique $|a_{pq}| \geq TOL$.

***Práctica*** *5 (Método de Jacobi para el cálculo de autovalores, 2 horas)*

Desarrollar en fortran 77 la siguiente función :

- FUNCTION JACOBI(A,N,Nmax,TOL,Niter) que realice el cálculo de los autovalores de una matriz simétrica $A$ por el método de Jacobi. La función devuelve 0 si termina bien y -1 en caso contrario. Utilizar el método para calcular los autovalores de las siguientes matrices, tomando $TOL = 0.0001$ y $Niter = 1000$.

1. $A = \begin{pmatrix} 2 & 2 & -2 \\ 2 & 1 & 1 \\ -2 & 1 & 1 \end{pmatrix}$
2. $A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$
3. $A = \begin{pmatrix} 0 & 1 & 6 & 0 & 0 & 0 \\ 1 & 0 & 2 & 7 & 0 & 0 \\ 6 & 2 & 0 & 3 & 8 & 0 \\ 0 & 7 & 3 & 0 & 4 & 9 \\ 0 & 0 & 8 & 4 & 0 & 5 \\ 0 & 0 & 0 & 9 & 5 & 0 \end{pmatrix}$

en este caso los autovalores son:

$$\begin{pmatrix} -12.1283 \\ -10.0647 \\ -2.46536 \\ 2.109435 \\ 5.942\,94 \\ 16.6060 \end{pmatrix}$$

4. Aplicar el método para el cálculo de los autovalores de los sistemas de dimensión 10 y 100 presentes en el directorio de la asignatura.

**Nota**: Está práctica se entregará conjuntamente con la práctica 6 por estar las dos muy relacionadas.

*Método de la potencia*

**Teorema 22** *Sea una matriz $A$ que posee una base de autovectores tal que en módulo su autovalor máximo $\lambda_{\max}$ es único. Sea un vector $u^1$ no ortogonal al subespacio engendrado por los autovectores del autovalor $\lambda_{\max}$, entonces, si definimos la secuencia*

$$u^n = A\frac{u^{n-1}}{\| u^{n-1} \|}$$

*se verifica*

$$Lim_{n\to\infty} sign\left(\left(u^n, u^{n-1}\right)\right) \| u^n \| = \lambda_{\max}$$

$Lim_{n\to\infty} \left(sign\left(\left(u^n, u^{n-1}\right)\right)\right)^n \frac{u^n}{\|u^n\|}$ *es un autovector de* $\lambda_{\max}$

además dicho autovector tiene norma 1.

**Teorema 23** $sign\left(\left(u^n, u^{n-1}\right)\right)$ *es el signo del producto escalar de* $u^n$ *y* $u^{n-1}$, *es decir* $sign\left(\left(u^n, u^{n-1}\right)\right) = 1$ *si* $\left(u^n, u^{n-1}\right) \geq 0$ *y* $sign\left(\left(u^n, u^{n-1}\right)\right) = -1$ *si* $\left(u^n, u^{n-1}\right) < 0$.

**Demostración**. En primer lugar vamos a demostrar por inducción la siguiente igualdad

$$u^{n+1} = \frac{A^n u^1}{\|A^{n-1}u^1\|}$$

Para $n = 1$ la igualdad se cumple por la definición de $u^2$. Supongamos que se cumple para $n-1$, y demostrémoslo para $n$.

$$u^{n+1} = A\frac{u^n}{\|u^n\|} = A\frac{\frac{A^{n-1}u^1}{\|A^{n-2}u^1\|}}{\frac{\|A^{n-1}u^1\|}{\|A^{n-2}u^1\|}} = \frac{A^n u^1}{\|A^{n-1}u^1\|}$$

con lo que queda demostrado este primer resultado. Por otro lado como $A$ posee una base de autovectores que denotaremos por $x_i$ y $u^1$ no es ortogonal al espacio generado

por los autovectores asociados a $\lambda_{\max}$, entonces $u^1$ se puede escribir como

$$u^1 = \mu_1 x_1 + ... + \mu_N x_N$$

donde supondremos que $x_1$ es un autovector asociado a $\lambda_{\max}$ y que $\mu_1 \neq 0$. Por la igualdad anteriormente demostrada obtenemos

$$u^n = \frac{A^{n-1}u^1}{\|A^{n-2}u^1\|} = \frac{\mu_1\lambda_{\max}^{n-1}x_1+...+\mu_N\lambda_N^{n-1}x_N}{\left\|\mu_1\lambda_{\max}^{n-2}x_1+...+\mu_N\lambda_N^{n-2}x_N\right\|} =$$
$$= |\lambda_{\max}| \frac{\mu_1\left(\frac{\lambda_{\max}}{|\lambda_{\max}|}\right)^{n-1}x_1+...+\mu_N\left(\frac{\lambda_N}{|\lambda_{\max}|}\right)^{n-1}x_N}{\left\|\mu_1\left(\frac{\lambda_{\max}}{|\lambda_{\max}|}\right)^{n-2}x_1+...+\mu_N\left(\frac{\lambda_N}{|\lambda_{\max}|}\right)^{n-2}x_N\right\|}$$

Cuando hacemos $n$ tender hacia infinito todos los cocientes de la forma

$$\left(\frac{\lambda_i}{|\lambda_{\max}|}\right)^n$$

tienden hacia 0 salvo si $\lambda_i = \lambda_{\max}$. en cuyo caso, dicho cociente es $1^n$, si $\lambda_{\max}$ es positivo, o $(-1)^n$, si $\lambda_{\max}$ es negativo. Por tanto, para $n$ suficientemente grande el signo de $\lambda_{\max}$ coincide con el signo del producto escalar $(u^n, u^{n-1})$. Además

$$Lim_{n\to\infty}\|u^n\| =$$
$$= |\lambda_{\max}| \frac{\left\|\mu_1\left(\frac{\lambda_{\max}}{|\lambda_{\max}|}\right)^{n-1}x_1+...+\mu_N\left(\frac{\lambda_N}{|\lambda_{\max}|}\right)^{n-1}x_N\right\|}{\left\|\mu_1\left(\frac{\lambda_{\max}}{|\lambda_{\max}|}\right)^{n-2}x_1+...+\mu_N\left(\frac{\lambda_N}{|\lambda_{\max}|}\right)^{n-2}x_N\right\|} =$$
$$|\lambda_{\max}|$$

y por tanto

$$Lim_{n\to\infty} sign\left((u^n, u^{n-1})\right) \parallel u^n \parallel = \lambda_{\max}$$

Por otro lado, el término $\left(sign\left((u^n, u^{n-1})\right)\right)^n$ para $n$ suficientemente grande es $1^n$ si $\lambda_{\max}$ es positivo o $(-1)^n$ si $\lambda_{\max}$ es negativo. Sean $x_1, ...., x_M$ son los autovectores asociados a $\lambda_{\max}$, obtenemos que

$$Lim_{n\to\infty}\left(sign\left((u^n, u^{n-1})\right)\right)^n \frac{u^n}{\|u^n\|} =$$
$$\frac{\mu_1 x_1+...+\mu_M x_M}{\|\mu_1 x_1+...+\mu_M x_M\|}$$

que es un autovector de $\lambda_{\max}$ de norma 1.

**Problema 63** ***(3 puntos)*** *Aplicar el método de la potencia para aproximar el autovalor máximo, y el autovector asociado, de las siguientes matrices, dando 3 pasos en el método, hasta calcular $u^4$ y partiendo de $u^1 = (1, 1)$.*

$$A = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$$
$$A = \begin{pmatrix} -3 & 0 \\ 1 & 1 \end{pmatrix}$$

*Método de la potencia inversa.*

El método anterior también se puede utilizar para el cálculo del autovalor de módulo más pequeño $\lambda_{\min}$ teniendo en cuenta que

$$\lambda_{\min} = \frac{1}{\max\{\lambda_i' \text{ autovalores de } A^{-1}\}}$$

por tanto si aplicamos el método anterior a $A^{-1}$ obtenemos que la secuencia

$$u^n = A^{-1}\frac{u^{n-1}}{\parallel u^{n-1} \parallel}$$

verifica

$$Lim_{n\to\infty} sign\left((u^n, u^{n-1})\right) \parallel u^n \parallel = \frac{1}{\lambda_{\min}}$$

$$Lim_{n\to\infty} sign\left((u^n, u^{n-1})\right)\frac{u^n}{\parallel u^n \parallel} \text{ es un autovector de } \lambda_{\min}$$

En los casos prácticos se evita calcular directamente $A^{-1}$, y se obtiene $u^n$ resolviendo el sistema

$$Au^n = \frac{u^{n-1}}{\parallel u^{n-1} \parallel}$$

**Problema 64** ***(2 puntos)*** *Calcular el autovalor mayor y el autovector correspondiente de la matriz* $\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$ *utilizando el método de la potencia, dando 2 iteraciones del método a partir de $u_1 = (1, 1)$ y tomando como norma $\|u\| = \max_i |u_i|$*

**Problema 65** ***(2 puntos)*** *Utilizar el método de la potencia inversa para aproximar el autovalor más pequeño de la matriz*

$$A = \begin{pmatrix} -2 & 1 \\ 0 & 3 \end{pmatrix}$$

*llegar hasta $u^3$ partiendo de $u = (1, 1)$.*

Para autovalores que se encuentren entre $\lambda_{\min}$ y $\lambda_{\max}$, se puede proceder de la manera siguiente: Se calcula primero una aproximación $\mu$ del autovalor $\lambda$ de tal forma que $\mu$ se encuentre más cercano a $\lambda$ que a cualquier otro autovalor. Por ejemplo, utilizando el método de Jacobi, si consideramos la matriz $A' = A - \mu I$, donde $\mu$ es uno de los elementos diagonales de la matriz que resulta de aplicar el método de Jacobi, entonces se obtiene que el autovalor más pequeño de $A'$ es justamente $\lambda$, y por tanto podemos aplicar el método de potencia inversa anterior. Nótese que si el autovalor $\mu$ está calculado con mucha precisión, entonces el autovalor más pequeño de $A'$ está muy próximo a 0, y como el determinante de una matriz es el producto de sus autovalores, ello indicaría que el determinante de $A'$ estaría muy próximo a 0 y podemos tener

problemas al resolver el sistema utilizado por ejemplo el método de GAUSS a través de la función de la librería an.h IGAUSS(), para evitar esto, podemos perturbar ligeramente el valor de $\mu$ para que IGAUSS() no de problemas. Algorítmicamente quedaría como sigue. Si $\mu$ es el autovalor que estamos tratando haremos

$\epsilon = 10^{-11}$
1 $A' = A - \mu Id$
$J = IGAUSS(A', .....)$
$IF\ (J.NE.0)\ THEN$
$\epsilon = \epsilon * 10.$
$\mu = \mu(1+\epsilon)$
$GOTO\ 1$
$ENDIF$

**Problema 66** ***(3 puntos)*** *Calcular el autovalor y autovector más cercano a 2 de la matriz*

$$\begin{pmatrix} 0 & -1 & 0 \\ 0 & 3 & -1 \\ 0 & 0 & -1 \end{pmatrix}$$

*para ello calcular dos iteraciones del método de la potencia inversa partiendo de* $u^1 = (1,1,1)$.

***Práctica*** *6 (Cálculo de autovalores y autovectores de una matriz simétrica, 4 horas)*

Desarrollar en fortran 77 las siguientes funciones :

- FUNCTION ERROR_VECTORES(U,V,N) : Devuelve la diferencia entre los vectores U y V de dimensión N, utilizando la fórmula :

$$ERROR\_VECTORES = \frac{1}{N}\sum_{i=1}^{N}\frac{ABS(U(i)-V(i))}{ABS(U(i))+1.}$$

- FUNCTION AUTOVALOR_MINIMO(A,u,N,Nmax,TOL,Niter) : Calcula el autovalor y autovector mínimo de una matriz A por el método de la potencia inversa, u es el candidato inicial, y donde se devuelve el autovector al final, la función devuelve el autovalor mínimo si ha terminado bien, $2^{100}$ si ha fallado IGAUSS(.) y $2^{120}$ si falla por otro motivo. (Utilizar la función IGAUSS() de an.h para resolver los sistemas que aparezcan).

- FUNCTION IAUTOVECTORES_ (A,AUTOVECTORES,AUTOVALORES,N, Nmax,TOL,Niter) : Calcula los autovectores y autovalores de la matriz A, siguiendo las siguientes fases

  1. Se calculan los autovalores utilizando JACOBI() y se almacenan en el vector AUTOVALORES(.)
  2. Para cada autovalor $\lambda_i$ se construye la matriz $B = A - \lambda_i Id$ , se hace $\epsilon = 10^{-11}$ y se llama a la función AUTOVALOR_MINIMO para esta matriz, tomando como candidato inicial $u = (1,1,1,1.....,1)$. Si IGAUSS falla, hacer $\epsilon = 10 * \epsilon$ y perturbar el autovalor haciendo AUTOVALORES(I)= AUTOVALORES(I)$(1+\epsilon)$, y volver a construir la matriz $B$ hasta que IGAUSS no falle, los vectores se van almacenando en la matriz AUTOVECTORES, y se corrigen los autovalores haciendo AUTOVALORES(I)= AUTOVALORES(I)+ AUTOVALOR_MINIMO(). La función devuelve 0 si termina bien y -1 en caso contrario.

- FUNCTION ERROR_AUTOVECTORES (A,AUTOVECTORES,AUTOVALORES,N,Nmax) : Para comprobar que los autovalor $\lambda_i$ y su autovector $\bar{x}_i$ están bien estimados, comparar para cada autovalor $\lambda_i$ y utilizando la función ERROR_VECTORES(), los vectores $A\bar{x}_i$ y $\lambda_i\bar{x}_i$ Devolver la expresión

$$ERROR\_AUTOVECTORES =$$

$$\max_{i=1,N} ERROR\_VECTORES(A\bar{x}_i, \lambda\bar{x}_i)$$

Comprobar los resultados obtenidos con los siguientes ejemplos, tomando $TOL = 0.0001$ y $Niter = 1000$.:

1. $A = \begin{pmatrix} 2 & 2 & -2 \\ 2 & 1 & 1 \\ -2 & 1 & 1 \end{pmatrix}$
   Resultado:
   $\left\{\begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}\right\} \leftrightarrow -2, \left\{\begin{pmatrix} -2 \\ -1 \\ 1 \end{pmatrix}\right\} \leftrightarrow 4, \left\{\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}\right\} \leftrightarrow 2$
2. $A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$
   Resultado:
   $\left\{\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}\right\} \leftrightarrow 2, \left\{\begin{pmatrix} 1 \\ -\sqrt{2} \\ 1 \end{pmatrix}\right\} \leftrightarrow 2 + \sqrt{2}, \left\{\begin{pmatrix} 1 \\ \sqrt{2} \\ 1 \end{pmatrix}\right\} \leftrightarrow 2 - \sqrt{2}$
3. Las matrices de dimensión 10 y 100 del directorio de la asignatura.
4. $A = \begin{pmatrix} 0 & 1 & 6 & 0 & 0 & 0 \\ 1 & 0 & 2 & 7 & 0 & 0 \\ 6 & 2 & 0 & 3 & 8 & 0 \\ 0 & 7 & 3 & 0 & 4 & 9 \\ 0 & 0 & 8 & 4 & 0 & 5 \\ 0 & 0 & 0 & 9 & 5 & 0 \end{pmatrix}$

Resultados:

$$16.6 \leftrightarrow \begin{pmatrix} .36327 \\ .62628 \\ .901 \\ 1.1763 \\ 1.0 \\ .93854 \end{pmatrix} \quad 5.942 \leftrightarrow \begin{pmatrix} 2.9844 \\ -.79684 \\ 3.0888 \\ -1.9854 \\ 1.0 \\ -2.1653 \end{pmatrix}$$

$$2.11 \leftrightarrow \begin{pmatrix} -.5938 \\ -1.5628 \\ 0.05170 \\ -.40089 \\ 1.0 \\ .65986 \end{pmatrix} \quad -2.465 \leftrightarrow \begin{pmatrix} -1.7465 \\ .85375 \\ .57536 \\ -.21557 \\ 1.0 \\ -1.2412 \end{pmatrix}$$

$$-10.06 \leftrightarrow \begin{pmatrix} .52132 \\ .74124 \\ -.99809 \\ -.8552 \\ 1.0 \\ .26811 \end{pmatrix} \quad -12.12 \leftrightarrow \begin{pmatrix} .72918 \\ -.89618 \\ -1.3246 \\ 1.8268 \\ 1.0 \\ -1.7676 \end{pmatrix}$$

**Nota:** Observese al comparar los resultados, que los autovectores están definidos módulo la multiplicación por una constante.

## Métodos iterativos de resolución de sistemas lineales.

Estas técnicas se basan en transformar un sistema de la forma

$$Au = b$$

en una ecuación de punto fijo de la forma

$$u = Mu + c$$

de tal manera que al hacer iteraciones de la forma

$$u^n = Mu^{n-1} + c$$

se obtenga que $u^n$ converge hacia $u$, la solución del sistema original.

**Ejemplo 12** *Consideremos el sistema de ecuaciones*

$$\begin{aligned} 2x - y &= 1 \\ -x + 2y - z &= 0 \\ -y + 2z &= 1 \end{aligned}$$

*buscar la solución de este sistema es equivalente a buscar un vector $u = (x, y, z)$ que verifique*

$$\begin{aligned} x &= \frac{1+y}{2} \\ y &= \frac{x+z}{2} \\ z &= \frac{1+y}{2} \end{aligned}$$

*hacer iteraciones de esta ecuación de punto fijo consiste en partir de una aproximación inicial $(x_1, y_1, z_1)$ y hacer iteraciones de la forma*

$$\begin{aligned} x_n &= \frac{1+y_{n-1}}{2} \\ y_n &= \frac{x_{n-1}+z_{n-1}}{2} \\ z_n &= \frac{1+y_{n-1}}{2} \end{aligned}$$

*en este caso la solución exacta del sistema es $u = (1, 1, 1)$. Si hacemos iteraciones del esquema anterior a partir de la aproximación inicial $u^1 = (0, 0, 0)$ obtenemos*

$$\begin{aligned} x_2 &= \frac{1+0}{2} = \frac{1}{2} \\ y_2 &= \frac{0+0}{2} = 0 \\ z_2 &= \frac{1+0}{2} = \frac{1}{2} \end{aligned}$$

*de la misma forma obtenemos*

$$u^3 = \begin{pmatrix} 0.5 \\ 0.25 \\ 0.5 \end{pmatrix} .... \; u^8 = \begin{pmatrix} .84 \\ .73 \\ .84 \end{pmatrix} .... \; u^{17} = \begin{pmatrix} .98 \\ .96 \\ .98 \end{pmatrix}$$

*como puede observarse, las sucesivas iteraciones van aproximando a la solución $u = (1, 1, 1)$. en este caso la matriz $M$ y el vector $c$ que determinan el esquema iterativo vienen dados por*

$$M_J = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 \end{pmatrix} \quad c_J = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix}$$

**Teorema 24** *Si el esquema iterativo*

$$u^n = Mu^{n-1} + c$$

*converge hacia un vector $u$, entonces $u$ verifica*

$$u = Mu + c$$

Existen diferentes métodos para convertir un sistema de la forma $Au = b$ en una ecuación de punto fijo $u = Mu + c$, todas se basan en descomponer $A$ de la forma $A = L + D + U$, donde $D$ es la matriz diagonal que corresponde a la parte diagonal de $A$, $L$ es la matriz triangular inferior que corresponde a la parte de $A$ situada por debajo de la diagonal, y $U$ es la matriz triangular superior que corresponde a la parte de $A$ situada por encima de la diagonal de $A$.

*Método de Jacobi*

Este método consiste en tomar

$$\begin{aligned} M_J &= D^{-1}(-L - U) \\ c_J &= D^{-1}b \end{aligned}$$

Este método es el que se ha utilizado en el ejemplo anterior. El paso de una iteración a otra del método de Jacobi puede expresarse de la siguiente forma:

$$\begin{aligned} u_1^n &= \frac{-a_{12}u_2^{n-1} - ... - a_{1N}u_N^{n-1} + b_1}{a_{11}} \\ u_2^n &= \frac{-a_{21}u_1^{n-1} - a_{23}u_3^{n-1} ... - a_{2N}u_N^{n-1} + b_2}{a_{22}} \\ &. \\ u_N^n &= \frac{-a_{N1}u_1^{n-1} - a_{N2}u_2^{n-1} ... - a_{NN-1}u_{N-1}^{n-1} + b_N}{a_{NN}} \end{aligned}$$

**Problema 67** ***(3 puntos)*** *Escribir en Fortran las funciones siguientes: SIGNO_PRODUCTO_ESCALAR(uf,vf,Nf) que devuelve el signo del producto escalar de los vectores uf y vf de dimensión Nf (12 líneas de código como máximo), y la función AUTOVALOR_MAXIMO(Af,uf,Nf,Nfmax,Nfiter,Tolf) que devuelve el autovalor máximo de una matriz y su autovector por el método de la potencia. Los parámetros son la matriz Af, el vector candidato inicial uf, Nf la dimensión real, Nfmax, la dimensión para coger memoria, Nfiter número máximo de iteraciones, y Tolf la tolerancia. Esta función devuelve el valor 2.**120 si no termina correctamente. Tomar como norma* $\|u\| = \sum_i ABS(u_i)$ *(28 líneas de código como máximo)*

**Problema 68** ***(2 puntos)*** *Calcular* 3 *iteraciones del método de Jacobi para resolver el sistema*

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & -1 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -1 \\ 3 \\ 1 \end{pmatrix}$$

*partiendo de* $u^1 = (0, 0, 0)$

*Método de Gauss-Seidel*

Este método consiste en tomar

$$\begin{aligned} M_{GS} &= (D + L)^{-1}(-U) \\ c_{GS} &= (D + L)^{-1} b \end{aligned}$$

A efectos prácticos, la aplicación de este método no requiere el cálculo directo de la matriz inversa $(D + L)^{-1}$ puesto que el paso de una iteración a otra puede hacerse de forma recursiva de la siguiente forma:

$$\begin{aligned} u_1^n &= \frac{-a_{12}u_2^{n-1} - ... - a_{1N}u_N^{n-1} + b_1}{a_{11}} \\ u_2^n &= \frac{-a_{21}u_1^{n} - a_{23}u_3^{n-1} ... - a_{2N}u_N^{n-1} + b_2}{a_{22}} \\ &. \\ u_N^n &= \frac{-a_{N1}u_1^{n} - a_{N2}u_2^{n} ... - a_{NN-1}u_{N-1}^{n} + b_N}{a_{NN}} \end{aligned}$$

de tal forma que si hacemos un barrido para el cálculo de la solución de arriba hacia abajo, y vamos actualizando las componentes del vector aproximación según las vamos calculando obtenemos el método de Gauss-Seidel. Por tanto básicamente podemos decir que la diferencia entre el método de Gauss-Seidel y el método de Jacobi es que en el método de Gauss-Seidel se actualiza el vector aproximación después del cálculo de cada componente, y en el caso de Jacobi se actualiza sólo al final, después de haber calculado todas las componentes por separado.

**Ejemplo 13** *Vamos a aplicar el método de Gauss-Seidel al sistema del ejemplo anterior, es decir*

$$\begin{aligned} 2x - y &= 1 \\ -x + 2y - z &= 0 \\ -y + 2z &= 1 \end{aligned}$$

*Las iteraciones del método de Gauss-Seidel aplicado a este sistema consisten en*

$$\begin{aligned} x_n &= \frac{1 + y_{n-1}}{2} \\ y_n &= \frac{x_n + z_{n-1}}{2} \\ z_n &= \frac{1 + y_n}{2} \end{aligned}$$

Si hacemos iteraciones del esquema anterior a partir de la aproximación inicial $u^1 = (0, 0, 0)$ obtenemos

$$\begin{aligned} x_2 &= \frac{1+0}{2} = \frac{1}{2} \\ y_2 &= \frac{\frac{1}{2}+0}{2} = \frac{1}{4} \\ z_2 &= \frac{1+\frac{1}{4}}{2} = \frac{5}{8} \end{aligned}$$

de la misma forma obtenemos

$$u^3 = \begin{pmatrix} .5 \\ .25 \\ .625 \end{pmatrix} .... \; u^8 = \begin{pmatrix} .976\,56 \\ .976\,56 \\ .988\,28 \end{pmatrix}$$

**Problema 69** ***(2 puntos)*** *Calcular una base ortogonal de autovectores de la matriz* $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix}$

**Problema 70** ***(2 puntos)*** *Calcular* 3 *iteraciones del método de Gauss-Seidel para resolver el sistema*

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & -1 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -1 \\ 3 \\ 1 \end{pmatrix}$$

*partiendo de* $u^1 = (0, 0, 0)$

**Problema 71** ***(1 punto)*** *Una variante del método de Gauss-Seidel es tomar* $M = (D+U)^{-1}(-L)$, *y* $c = (D+U)^{-1}b$. *indicar en este caso que diferencias de implementación habría con respecto al caso anterior.*

*Método de relajación*

El objetivo de este método es intentar mejorar el método de Gauss-Seidel introduciendo un parámetro de relajación $w$. Se toman en este caso

$$\begin{aligned} M_w &= (D+wL)^{-1}((1-w)D - wU) \\ c_w &= w(D+wL)^{-1}b \end{aligned}$$

estas nuevas matrices se basan en realizar un promediado entre el resultado obtenido por Gauss-Seidel y el estado de la solución en la etapa anterior de la forma siguiente:

$$\begin{aligned} u_1^n &= w\frac{-a_{12}u_2^{n-1} - ... - a_{1N}u_N^{n-1} + b_1}{a_{11}} + (1-w)u_1^{n-1} \\ u_2^n &= w\frac{-a_{21}u_1^n ... - a_{2N}u_N^{n-1} + b_2}{a_{22}} + (1-w)u_2^{n-1} \\ &\cdot \\ u_N^n &= w\frac{-a_{N1}u_1^n ... - a_{NN-1}u_{N-1}^n + b_N}{a_{NN}} + (1-w)u_N^{n-1} \end{aligned}$$

La elección del parámetro $w$ es en general un problema dificil, sin embargo, en el caso de matrices tridiagonales (es decir matrices con todos los elementos nulos salvo la diagonal principal y sus codiagonales) el siguiente resultado muestra la forma de calcular el valor de $w$ óptimo.

**Teorema 25** *Si* $A$ *es una matriz tridiagonal y* $\rho(M_J) < 1$, *entonces el valor de* $w$ *que optimiza la velocidad de convergencia del método es:*

$$w_{opt} = \frac{2}{1+\sqrt{1-\rho(M_J)^2}}$$

Como puede observarse de la expresión anterior, el valor de $w_{opt}$ se encuentra siempre entre 1 y 2.
**Demostración** [La-Th]. Pg.358-362

**Ejemplo 14** *Vamos aplicar el método de relajación al sistema del ejemplo anterior, es decir*

$$\begin{aligned} 2x - y &= 1 \\ -x + 2y - z &= 0 \\ -y + 2z &= 1 \end{aligned}$$

*en este caso* $\rho(M_J) = \frac{1}{\sqrt{2}}$, *y* $w_{opt} = 1.17$

*Las iteraciones del método de relajación aplicado a este sistema consisten en*

$$\begin{aligned} x_n &= w\frac{1+y_{n-1}}{2} + (1-w)x_{n-1} \\ y_n &= w\frac{x_n + z_{n-1}}{2} + (1-w)y_{n-1} \\ z_n &= w\frac{1+y_n}{2} + (1-w)z_{n-1} \end{aligned}$$

Si hacemos iteraciones del esquema anterior a partir de la aproximación inicial $u^1 = (0,0,0)$ y tomando $w = w_{opt} = 1.17$ obtenemos

$$u^2 = \begin{pmatrix} .585 \\ .342 \\ .785 \end{pmatrix} .... \; u^3 = \begin{pmatrix} .686 \\ .802 \\ .921 \end{pmatrix} ... \; u^8 = \begin{pmatrix} .999 \\ .999 \\ .999 \end{pmatrix}$$

**Problema 72** ***(3 puntos)*** *Calcular* 3 *iteraciones del método de relajación para resolver el sistema*

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & -1 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -1 \\ 3 \\ 1 \end{pmatrix}$$

*, partiendo de* $u^1 = (0,0,0)$. *Calcular previamente el parámetro de relajación óptimo*

*Convergencia de los métodos iterativos.*

Vamos a denotar por $e^n = u^n - u$, el error relativo entre la solución del sistema $u$ y la aproximación en la etapa $n$, $u^n$.

**Teorema 26** *Se considera el esquema iterativo* $u^n = Mu^{n-1} + c$. *Entonces*

$$e^n = M^{n-1}e^1$$

**Demostración:** La solución del sistema satisface $u = Mu + c$. Restando esta igualdad de la igualdad $u^n = Mu^{n-1} + c$ obtenemos

$$u^n - u = M(u^{n-1} - u) = M^{n-1}(u^1 - u)$$

■

**Teorema 27** *El método iterativo* $u^n = Mu^{n-1} + c$ *converge para cualquier aproximación inicial si sólo si* $\rho(M) < 1$.

**Demostración:** El resultado es inmediato a partir del hecho de que una matriz $M^n$ converge hacia 0 cuando $n \to \infty$ si sólo si $\rho(M) < 1$

**Teorema 28** *Si en el método de relajación* $w \notin (0,2)$ *entonces* $\rho(M_w) \geq 1$.

**Demostración:** En primer lugar observamos que las matrices $D+Lw$ y $(1-w)D-wU$ son matrices triángulares, y por tanto su determinante es el producto de los elementos diagonales. Además utilizando que el determinante del producto de dos matrices es el producto de sus determinantes, y que el determinante de la matriz inversa es el inverso del determinante, obtenemos

$$|M_w| = \frac{|(1-w)D - wU|}{|(D+wL)|} = \frac{(1-w)^N \Pi_i a_{ii}}{\Pi_i a_{ii}}$$

por tanto, como el determinante de una matriz es el producto de sus autovalores, obtenemos que si $w \notin (0,2)$, entonces $|1-w| \geq 1$ y en consecuencia $M_w$ posee al menos un autovalor de módulo mayor o igual que uno.

**Teorema 29** *Si una matriz A verifica*

$$| a_{ii} | > \sum_{j \neq i} | a_{ij} | \quad \forall i.$$

*o*

$$| a_{jj} | > \sum_{i \neq j} | a_{ij} | \quad \forall j.$$

*entonces el método de Jacobi asociado al sistema $Au = b$ converge para cualquier aproximación inicial.*

**Demostración:** En primer lugar observamos que la matriz $M_J$ puede expresarse como:

$$\begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & -\frac{a_{13}}{a_{11}} & . & -\frac{a_{1N}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & -\frac{a_{23}}{a_{22}} & . & -\frac{a_{2N}}{a_{22}} \\ . & . & . & . & . \\ -\frac{a_{N-1,1}}{a_{N-1,N-1}} & -\frac{a_{N-1,2}}{a_{N-1,N-1}} & . & 0 & -\frac{a_{N-1,N}}{a_{N-1,N-1}} \\ -\frac{a_{N,1}}{a_{N,N}} & -\frac{a_{N,2}}{a_{N,N}} & . & -\frac{a_{N,N-1}}{a_{N,N}} & 0 \end{pmatrix}$$

teniendo en cuenta que las normas 1 e infinito de una matriz son el máximo de las sumas por filas o columnas en valor absoluto, se tiene por las condiciones del teorema que $\|M_J\| < 1$ para la norma 1 o infinito. Por tanto, el teorema se concluye teniendo en cuenta que cualquier norma de una matriz es siempre mayor o igual que su radio espectral

Este resultado se puede generalizar un poco al caso de matrices irreducibles de la siguiente forma:

**Definición 7** *Una matriz A es irreducible si un sistema de la forma $Au = b$ no puede descomponerse en dos subsistemas independientes de dimensión menor*

Dicho de otra forma, una matriz es irreducible si el cambio de cualquier valor del vector $b$ del sistema $Au = b$ afecta a todos los elementos del vector $u$.

**Teorema 30** *Si A es una matriz irreducible y se verifica*

$$| a_{ii} | \geq \sum_{j \neq i} | a_{ij} | \quad \forall i.$$

*o*

$$| a_{jj} | \geq \sum_{i \neq j} | a_{ij} | \quad \forall j.$$

*con la desigualdad estricta en al menos una fila o columna, entonces los métodos iterativos convergen.*

**Demostración**. [La-The] Pg.346-347

**Ejemplo 15** *La matriz del sistema ejemplo tratado anteriormente, esto es*

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

*satisface las hipótesis del Teorema anterior*

**Problema 73** ***(2 puntos)*** *Escribir en Fortran la función siguiente: CONDICIONAMIENTO(Af,Nf,Nfmax,TOLf,Nfiter) que devuelve el condicionamiento de una matriz utilizando el método de Jacobi para calcular los autovalores, se supondrá implementada la función JACOBI(A,N,Nmax,TOL,Niter) que devuelve 0 si termina bien y 1 si termina mal. La función CONDICIONAMIENTO devuelve 2.*120 si termina mal porque Jacobi da un error o se produce una división por cero. Los parámetros son la matriz Af, Nf la dimensión real, Nfmax, la dimensión para coger memoria, Nfiter número máximo de iteraciones, y Tolf la tolerancia (21 líneas de instrucciones como máximo)*

**Problema 74** ***(2 puntos)*** *Demostrar que si una matriz A verifica que por filas o columnas su suma es siempre igual a 0, entonces el determinante de A es cero, y por tanto el sistema asociado a A no tiene solución.*

**Problema 75** ***(3 puntos)*** *Dado un sistema iterativo*

$$u^n = Mu^{n-1} + c$$

*Demostrar que aunque el radio espectral de M sea mayor que 1, si $u^1$ y c son combinaciones lineales de autovectores de M correspondientes a autovalores de módulo menor que 1, entonces el método converge.*

***Práctica** 7 (Método de relajación, 4 horas).*

Crear una función en fortran 77 donde se implemente el método de relajación. Los parámetros de la función serán: la matriz $A$, el vector $b$, un vector $u$ donde se almacenará la solución, y que inicialmente será el vector aproximación inicial, que por defecto se tomará 0, el parámetro de relajación $w$, el número máximo de iteraciones $N\max$, y la tolerancia $TOL$ para evaluar la diferencia entre $u^n$ y $u^{n-1}$. La función devolverá el número de iteraciones necesarias para alcanzar la solución, si el método no converge devuelve $-1$. Comparar la diferencia en la velocidad de convergencia entre el método de Gauss-Seidel y el Método de relajación. Probar el método para los sistemas

1. $\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & -1 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -1 \\ 3 \\ 1 \end{pmatrix}$

2. $\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$

3. $\begin{pmatrix} 1 & 3 & 3 \\ 3 & 1 & 3 \\ 3 & 3 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 7 \\ 7 \\ 7 \end{pmatrix}$

4. Los sistemas ejemplos del directorio de la asignatura. Estos ejemplos tienen siempre como solución el vector $(1, 1, ...., 1)$.

### Método de Newton-Raphson para sistemas de ecuaciones-no-lineales

En las aplicaciones reales, muchas veces nos encontramos con sistemas no-lineales de ecuaciones. Por ejemplo, calcular las raíces, reales o complejas de un polinomio de grado 2 dado por $P_2(z) = az^2 + bz + c$, donde $z = x + yi$ es equivalente a resolver el sistema

$$\begin{aligned} ax^2 + bx - ay^2 + c &= 0 \\ 2ayx + by &= 0 \end{aligned}$$

que es un sistema no-lineal de ecuaciones. En general, un sistema no-lineal de ecuaciones de dimensión $N$, se escribe como $N$ ecuaciones del tipo

$$\begin{aligned} f_1(u_1, ...., u_N) &= 0 \\ f_2(u_1, ...., u_N) &= 0 \\ &\cdot \\ f_N(u_1, ...., u_N) &= 0 \end{aligned}$$

donde $f(u) = (f_1(u), f_2(u), ..., f_N(u))$ es una función de $\Re^N \to \Re^N$, y $u = (u_1, ...., u_N)$. El método de Newton-Raphson para sistemas de ecuaciones se basa en desarrollar por Taylor la función $f$ y truncar el desarrollo para que quede un sistema lineal, es decir

$$f(u) = f(u^0) + \nabla f(u^0)\left(u - u^0\right) + \mathcal{O}\left(\| u - u^0 \|^2\right)$$

donde $u^0$ es una aproximación de la solución de $f(u) = 0$, si truncamos el desarrollo e igualamos a 0 (para aproximar la raíz) obtenemos que la raíz del sistema lineal se obtiene resolviendo el sistema

$$\begin{aligned} \nabla f(u^0)z &= -f(u^0) \\ u^1 &= u^0 + z \end{aligned}$$

en el caso general a partir de una aproximación $u^n$ se obtiene la aproximación $u^{n+1}$ en dos etapas

$$\begin{aligned} \nabla f(u^n)z &= -f(u^n) \\ u^{n+1} &= u^n + z \end{aligned}$$

**Ejemplo 16** *Consideremos el siguiente sistema no-lineal de ecuaciones:*

$$\begin{aligned} x^2 - y^2 + 1 &= 0 \\ 2xy &= 0 \end{aligned}$$

*la matriz gradiente de esta función viene dada por*

$$\nabla f(u^n) = \begin{pmatrix} 2x & -2y \\ 2y & 2x \end{pmatrix}$$

*tomemos como aproximación inicial $u^1 = (1, 1)$. El sistema que hay que resolver para pasar de una iteración a otra es*

$$\begin{pmatrix} 2x_n & -2y_n \\ 2y_n & 2x_n \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = -\begin{pmatrix} x_n^2 - y_n^2 + 1 \\ 2y_n x_n \end{pmatrix}$$

*Si partimos de $u^1 = (1, 1)$ para obtener $u^2$ tenemos que resolver*

$$\begin{pmatrix} 2 & -2 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} -1 \\ -2 \end{pmatrix}$$

*que tiene por solución $\left(-\frac{3}{4}, -\frac{1}{4}\right)$ por tanto $u^2$ viene dado por*

$$u^2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} -\frac{3}{4} \\ -\frac{1}{4} \end{pmatrix} = \begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix}$$

*para calcular $u^3$ tenemos que resolver el sistema*

$$\begin{pmatrix} \frac{1}{2} & -\frac{3}{2} \\ \frac{3}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = -\begin{pmatrix} \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 + 1 \\ \frac{6}{16} \end{pmatrix}$$

*cuya solución es $\left(-\frac{13}{40}, -\frac{9}{40}\right)$. Por tanto $u^3$.viene dado por*

$$u^3 = \begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix} + \begin{pmatrix} -\frac{13}{40} \\ \frac{9}{40} \end{pmatrix} = \begin{pmatrix} -\frac{3}{40} \\ \frac{39}{40} \end{pmatrix}$$

*que ya es una buena aproximación de la solución exacta dada por el vector $(0, 1)$.*

**Problema 76** ***(3 puntos)*** *Calcular* 2 *iteraciones del método de Newton-Raphson no-lineal para aproximar una raíz del sistema de ecuaciones*

$$\begin{aligned} x^2+y^2-1 &= 0 \\ y-x &= 0 \end{aligned}$$

*partiendo de* $(x,y)=(1,1)$.

**Problema 77** ***(2 puntos)*** *Plantear el algoritmo necesario para calcular, utilizando el método de Newton-Raphson, las raíces complejas o reales de un polinomio de grado* 3.

**Problema 78** ***(2 puntos)*** *Se considera el sistema no-lineal*

$$\begin{aligned} (x-1)y &= 0 \\ (y-2)x &= 0 \end{aligned}$$

*A partir de* $u^1=(1,1)$, *calcular* $u^2$ *y* $u^3$ *utilizando el método de Newton-Raphson para aproximar un cero del sistema no-lineal.*

**Problema 79** ***(2 puntos)*** *Calcular* 1 *iteración del método de Newton-Raphson no-lineal para aproximar una raíz del sistema de ecuaciones*

$$\begin{aligned} e^{xyz}-1 &= 0 \\ y^2-z^3-2 &= 0 \\ (z-1)x^4-3 &= 0 \end{aligned}$$

*partiendo de* $(x,y,z)=(1,1,1)$.

## DIFERENCIACION E INTEGRACION NUMERICA

Una fórmula de diferenciación numérica es un procedimiento (o algoritmo) que permite aproximar la derivada de la función $f(x)$ en un punto $x_i$, utilizando el valor de $f(x)$ en otros puntos vecinos a $x_i$. Por otro lado una fórmula de integración numérica es un procedimiento que permite aproximar el valor de la integral en un intervalo $[a,b]$ a partir de la evaluación de $f(x)$ en algunos puntos incluidos en el intervalo $[a,b]$.

### Diferenciación Numérica

La manera habitual de aproximar la derivada de una función $f(x)$ en un punto $x_i$ es utilizando el desarrollo de Taylor centrado en $x_i$

$$f(x)=f(x_i)+\frac{f'(x_i)}{1!}(x-x_i)+...+\frac{f^{N)}(x_i)}{N!}(x-x_i)^N+...$$

si tomamos un punto $x_j \neq x_i$ y truncamos el desarrollo de Taylor obtenemos despejando

$$f'(x_i)\approx\frac{f(x_j)-f(x_i)}{x_j-x_i}+\mathcal{O}\left(|x_j-x_i|\right)$$

donde $\mathcal{O}\left(|x_j-x_i|\right)$ indica básicamente que el error cometido es una suma de potencias de $|x_j-x_i|$ donde la potencia más pequeña es 1. Se denomina orden de la aproximación a la potencia más pequeña que aparece en el término del error. Por tanto, en este caso diremos que el orden de aproximación es 1. Si el punto $x_j>x_i$ entonces la derivada se calcula hacia delante, y si $x_j<x_i$ la derivada se calcula hacia atrás.

**Ejemplo 17** *Veremos en este ejemplo como cuanto más próximo esté el punto* $x_j$ *al punto* $x_i$ *mejor será el valor aproximado de la derivada. Consideremos la función* $f(x)=x^3$. *La derivada de* $f(x)$ *en* $x=1$ *es* $f'(1)=3$ *Si tomamos* $x_i=1$ *y* $x_j=2$ *en la fórmula anterior obtenemos la aproximación*

$$f'(1)\approx\frac{2^3-1^3}{2-1}=7$$

*Si tomamos ahora* $x_j=1.1$

$$f'(1)\approx\frac{1.1^3-1^3}{1.1-1}=3.31$$

*que está mucho más próximo al valor real.*

**Problema 80** ***(2 puntos)*** *Calcular analíticamente y numéricamente la matriz gradiente en el punto* $(1,1)$ *(utilizar* $h=0.1$*) de la función:*

$$f(x,y)=\left\{\begin{array}{c} x^2+y^2-1 \\ x-y \end{array}\right.$$

**Problema 81** ***(3 puntos)*** *Dados* 3 *puntos distintos* $x_l,x_i,x_r$ *demostrar que la fórmula:*

$$f'(x_i)\approx\frac{(x_i-x_l)\frac{f(x_r)-f(x_i)}{x_r-x_i}+(x_r-x_i)\frac{f(x_i)-f(x_l)}{x_i-x_l}}{x_r-x_l}$$

*aproxima la derivada de* $f'(x_i)$ *con un orden de aproximación de* 2.

Nótese que si $x_r=x_i+h$, y $x_l=x_i-h$, entonces la fórmula anterior queda

$$f'(x_i)=\frac{f(x_i+h)-f(x_i-h)}{2h}$$

que es una conocida fórmula de diferencias centradas.

**Ejemplo 18** *Veremos en este ejemplo como utilizando la expresión anterior para aproximar la derivada de* $f(x) = x^3$. *en* $x = 1$ *la precisión es mayor que con la fórmula anterior. Por ejemplo si tomamos* $x_i = 1$ *y* $h = 1$, *la expresión anterior nos da*

$$f'(1) \approx \frac{2^3 - 0^3}{2} = 4$$

*Si tomamos ahora* $x_j = 0.1$

$$f'(1) \approx \frac{1.1^3 - 0.9^3}{0.2} = 3.01$$

*que está más próximo al valor real que utilizando la primera fórmula. En general tendremos que cuanto mayor es el orden de una fórmula de aproximación, más preciso es el valor de la derivada.*

**Nota:** Utilizar el desarrollo de Taylor para aproximar $f'(x)$ es equivalente a interpolar $f(x)$ con el polinomio de Lagrange, y posteriormente derivar el polinomio.

**Problema 82** ***(3 puntos)*** *Dados* 3 *puntos distintos* $x_l, x_i, x_r$ *calcular el polinomio de Lagrange que interpola a* $f(x)$ *en esos* 3 *puntos, calcular la derivada de ese polinomio en* $x_i$ *y comprobar que da la misma fórmula que la presentada en el problema anterior.*

**Problema 83** ***(2 puntos)*** *Calcular una aproximación de la derivada tercera* $f'''(x_i)$ *de una función* $f(x)$ *en un punto* $x_i$, *utilizando* $f(x_i)$, $f(x_i + h)$, $f(x_i - h)$, $f(x_i - 2h)$

**Problema 84** ***(3 puntos)*** *Dados* 3 *puntos. Demostrar que la fórmula*

$$f''(x_i) \approx 2\frac{\frac{f(x_r)-f(x_i)}{x_r-x_i} - \frac{f(x_i)-f(x_l)}{x_i-x_l}}{x_r - x_l}$$

*aproxima la derivada segunda de* $f(x)$ *en* $x_i$ *con un orden de aproximación de* 1.

**Problema 85** ***(2 puntos)*** *Considerar en el problema anterior que* $x_l = x_i - h$, *y* $x_r = x_i + h$. *Deducir como queda la fórmula anterior para aproximar la derivada segunda, y demostrar que en este caso el orden de aproximación es* 2.

**Problema 86** ***(3 puntos)*** *Dados* 3 *puntos* $x_l < x_i < x_r$ *calcular el polinomio de Lagrange que interpola a* $f(x)$ *en esos* 3 *puntos, calcular la derivada segunda de ese polinomio en* $x_i$ *y comprobar que da la misma fórmula que utilizando los desarrollos de Taylor.*

**Problema 87** ***(2 puntos)*** *Calcular una aproximación de la derivada primera y segunda de una función* $f(x)$ *en* $x = 0$, *teniendo en cuenta que* $f(0) = 1$, $f(1) = 0$, $f(4) = 9$

**Diferenciación numérica en dimensiones superiores**

Estudiaremos en este apartado la aproximación de las derivadas de una función de varias variables. Para simplificar la exposición, supondremos que la dimensión es 2. Para discretizar las derivadas de una función $F(x, y)$ se utilizan los desarrollos de Taylor siguientes en 2 variables. Denotaremos por $F_x = \frac{\partial F(x,y)}{\partial x}$, $F_y = \frac{\partial F(x,y)}{\partial y}$, $F_{xx} = \frac{\partial^2 F(x,y)}{\partial x^2}$, $F_{xy} = \frac{\partial^2 F(x,y)}{\partial x \partial y}$, $F_{yy} = \frac{\partial^2 F(x,y)}{\partial y^2}$

1. $F(x+h, y) = F + hF_x + \frac{h^2}{2}F_{xx} + \mathcal{O}(h^3)$
2. $F(x-h, y) = F - hF_x + \frac{h^2}{2}F_{xx} + \mathcal{O}(h^3)$
3. $F(x, y+l) = F + lF_y + \frac{l^2}{2}F_{yy} + \mathcal{O}(l^3)$
4. $F(x, y-l) = F - lF_y + \frac{l^2}{2}F_{yy} + \mathcal{O}(l^3)$
5. $F(x+h, y+l) = F + hF_x + lF_y + \frac{1}{2}(h^2F_{xx} + 2hlF_{xy} + l^2F_{yy}) + \mathcal{O}(\left(h^2 + l^2\right)^{\frac{3}{2}})$
6. $F(x-h, y-l) = F - hF_x - lF_y + \frac{1}{2}(h^2F_{xx} + 2hlF_{xy} + l^2F_{yy}) + \mathcal{O}(\left(h^2 + l^2\right)^{\frac{3}{2}})$
7. $F(x+h, y-l) = F + hF_x - lF_y + \frac{1}{2}(h^2F_{xx} - 2hlF_{xy} + l^2F_{yy}) + \mathcal{O}(\left(h^2 + l^2\right)^{\frac{3}{2}})$
8. $F(x-h, y+l) = F - hF_x + lF_y + \frac{1}{2}(h^2F_{xx} - 2hlF_{xy} + l^2F_{yy}) + \mathcal{O}(\left(h^2 + l^2\right)^{\frac{3}{2}})$

Prestaremos particular atención a dos operadores diferenciales que se utilizan con frecuencia en la práctica: El gradiente $\nabla F(x, y) = (F_x(x, y), F_y(x, y))$ que es el vector de derivadas parciales, y el Laplaciano $\Delta F(x, y) = F_{xx}(x, y) + F_{yy}(x, y)$. Utilizaremos la notación $F_{i,j} \cong F(hi, lj)$.

*Discretización del Laplaciano*

Para discretizar el operador $\Delta F$ en un entorno de $3 \times 3$ puntos pueden utilizarse diferentes esquemas. Para simplificar, supondremos que $l = h$.

**Problema 88** ***(3 puntos)*** *Demostrar, utilizando el desarrollo de Taylor, que las siguientes expresiones son discretizaciones del laplaciano:*

$$\Delta F = \frac{F_{i+1,j+1} + F_{i-1,j+1} + F_{i-1,j-1} + F_{i+1,j-1} - 4F_{i,j}}{2h^2}$$

$$\Delta F = \frac{F_{i+1,j} + F_{i-1,j} + F_{i,j+1} + F_{i,j-1} - 4F_{i,j}}{h^2}$$

El resultado del anterior problema nos proporciona 2 formas distintas de evaluar el laplaciano, por tanto, cualquier promediado de las dos expresiones también es una discretización del laplaciano, es decir:

$$\Delta F = \gamma\frac{F_{i+1,j+1}+F_{i-1,j+1}+F_{i-1,j+1}+F_{i+1,j-1}-4F_{i,j}}{2h^2}+$$
$$+(1-\gamma)\frac{F_{i+1,j}+F_{i-1,j}+F_{i,j+1}+F_{i,j-1}-4F_{i,j}}{h^2}+\mathcal{O}(h)$$

donde $\gamma$ es un parámetro libre a elegir. La elección de dicho parámetro $\gamma$ la haremos en base a que la discretización de $\Delta F$ respete lo más posible la invarianza por rotaciones de la función $F(x,y)$, para ello consideremos una función tal que en un entorno de un punto $(hi_0, hj_0)$ tiene los siguientes valores:

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

Si calculamos $\Delta F$ en el punto central a través de la anterior fórmula obtenemos:

$$\Delta F(hi_0, hj_0) = \gamma\frac{2}{2h^2} + (1-\gamma)\frac{1}{h^2}$$

ahora bien, si rotamos 45 grados la función inicial entorno al punto $(hi_0, hj_0)$ obtenemos como imagen:

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

Si calculamos de nuevo $\Delta F$ en el mismo punto obtenemos:

$$\Delta F(hi_0, hj_0) = \gamma\frac{1}{2h^2} + (1-\gamma)\frac{2}{h^2}$$

Por lo tanto si queremos que ambos valores de $\Delta F$ coincidan, debemos elegir $\gamma = \frac{2}{3}$. Hablando en términos de teoría de la señal el calculo de $\Delta F$ nos llevaría a convolucionar la imagen con la siguiente máscara:

$$\frac{1}{h^2}\begin{array}{|c|c|c|}\hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \frac{1}{3} & -\frac{8}{3} & \frac{1}{3} \\ \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline\end{array}$$

**Problema 89** ***(2 puntos)*** *Calcular una aproximación del laplaciano de una función $F(x,y)$ en el punto $(x,y) = (0,0)$. conociendo los siguientes valores: $F(0,0) = 0$, $F(\frac{1}{2},0) = \frac{1}{4}$, $F(-\frac{1}{2},0) = \frac{1}{4}$, $F(0,\frac{1}{2}) = \frac{1}{4}$, $F(0,-\frac{1}{2}) = \frac{1}{4}$, $F(\frac{1}{2},\frac{1}{2}) = \frac{1}{2}$, $F(-\frac{1}{2},-\frac{1}{2}) = \frac{1}{2}$, $F(-\frac{1}{2},\frac{1}{2}) = \frac{1}{2}$, $F(\frac{1}{2},-\frac{1}{2}) = \frac{1}{2}$*

*Discretización del gradiente.*

Siguiendo el desarrollo de Taylor mostrado anteriormente obtenemos la siguiente expresión para el gradiente:

$$(F_{i,j})_x = (1-\gamma)\frac{(F_{i+1,j}-F_{i-1,j})}{2h}+$$
$$+\gamma\frac{(F_{i+1,j+1}-F_{i-1,j+1}+F_{i+1,j-1}-F_{i-1,j-1})}{4h}$$

$$(F_{i,j})_y = (1-\gamma)\frac{(F_{i,j+1}-F_{i,j-1})}{2h}+$$
$$+\gamma\frac{(F_{i+1,j+1}-F_{i+1,j-1}+F_{i-1,j+1}-F_{i-1,j-1})}{4h}$$

donde $\gamma$ es de nuevo un parámetro a elegir. Teniendo en cuenta que la norma euclídea. del gradiente es invariante por rotaciones, en particular lo es para rotaciones de 45 grados, de donde deducimos, utilizando el mismo argumento que para el $\Delta F$ que $\gamma = 2-\sqrt{2}$. Por tanto, estamos calculando $F_x$ utilizando la máscara:

$$\frac{1}{4h}\begin{array}{|c|c|c|}\hline -(2-\sqrt{2}) & 0 & (2-\sqrt{2}) \\ \hline -2(\sqrt{2}-1) & 0 & 2(\sqrt{2}-1) \\ \hline -(2-\sqrt{2}) & 0 & (2-\sqrt{2}) \\ \hline\end{array}$$

y $F_y$ utilizando:

$$\frac{1}{4h}\begin{array}{|c|c|c|}\hline -(2-\sqrt{2}) & 2(\sqrt{2}-1) & -(2-\sqrt{2}) \\ \hline 0 & 0 & 0 \\ \hline (2-\sqrt{2}) & -2(\sqrt{2}-1) & (2-\sqrt{2}) \\ \hline\end{array}$$

**Problema 90** ***(3 puntos)*** *Demostrar que las máscaras*

$$F_x = \frac{1}{4h}\begin{array}{|c|c|c|}\hline -(2-\sqrt{2}) & 0 & (2-\sqrt{2}) \\ \hline -2(\sqrt{2}-1) & 0 & 2(\sqrt{2}-1) \\ \hline -(2-\sqrt{2}) & 0 & (2-\sqrt{2}) \\ \hline\end{array}$$

$$F_y = \frac{1}{4h}\begin{array}{|c|c|c|}\hline -(2-\sqrt{2}) & -2(\sqrt{2}-1) & -(2-\sqrt{2}) \\ \hline 0 & 0 & 0 \\ \hline (2-\sqrt{2}) & 2(\sqrt{2}-1) & (2-\sqrt{2}) \\ \hline\end{array}$$

*dan lugar a una discretización del gradiente tal que su norma euclídea es invariante por rotaciones de 45 grados.*

**Problema 91** ***(2 puntos)*** *Calcular una aproximación del gradiente de una función $F(x,y)$ en el punto $(x,y) = (0,0)$. conociendo los siguientes valores: $F(0,0) = 0$, $F(\frac{1}{2},0) = \frac{1}{2}$, $F(-\frac{1}{2},0) = -\frac{1}{2}$, $F(0,\frac{1}{2}) = -\frac{1}{2}$, $F(0,-\frac{1}{2}) = \frac{1}{2}$, $F(\frac{1}{2},\frac{1}{2}) = 0$, $F(-\frac{1}{2},-\frac{1}{2}) = 0$, $F(-\frac{1}{2},\frac{1}{2}) = -1$, $F(\frac{1}{2},-\frac{1}{2}) = 1$*

## Integración Numérica

*Métodos de Cuadratura de Gauss*

Sea $f(x)$ una función definida en un intervalo $[a,b]$. Vamos a aproximar el valor de la integral de $f(x)$ en $[a,b]$, utilizando la evaluación de $f(x)$ en ciertos puntos de $[a,b]$. Es decir, una fórmula de integración numérica se puede escribir como

$$\int_a^b f(x)dx \approx \sum_{k=1}^{N} w_k f(x_k)$$

donde $x_k$ representan los puntos de evaluación de $f(x)$ y $w_k$ el peso de cada punto de evaluación.

**Definición 8** *Una fórmula de integración numérica se denomina exacta de orden $M$ si para cualquier polinomio $P(x)$ de grado menor o igual que $M$, la fórmula es exacta, es decir*

$$\int_a^b P(x)dx = \sum_{k=1}^{N} w_k P(x_k)$$

**Definición 9** *Se denominan polinomios de Legendre $L_n(x)$ a la familia de polinomios dada por $L_0(x) = 1$, $L_1(x) = x$, y para $n = 2, 3, ....$*

$$nL_n(x) = (2n-1)xL_{n-1}(x) - (n-1)L_{n-2}(x)$$

**Teorema 31** *Sean$\{\tilde{x}_k\}_{k=1,..,N}$ los ceros del polinomio de Legendre $L_N(x)$. Si definimos*

$$\tilde{w}_k = \int_{-1}^{1} \frac{\Pi_{i\neq k}(x - \tilde{x}_{i)}}{\Pi_{i\neq k}(\tilde{x}_k - \tilde{x}_{i)}} dx$$

*entonces la fórmula de integración numérica generada por los puntos $\tilde{x}_k$ y los pesos $\tilde{w}_k$ es exacta hasta el orden $2N-1$ para el intervalo $[-1, 1]$*

**Demostración** [Hu] Pg. 205-209

**Ejemplo 19** *A continuación se exponen algunos valores de raíces $\tilde{x}_k$ y coeficientes $\tilde{w}_k$ en función del grado del polinomio $L_n(x)$*

| $n$ | $\tilde{x}_k$ | $\tilde{w}_k$ |
|---|---|---|
| 2 | 0.5773502692 | 1. |
| | $-0.5773502692$ | 1 |
| 3 | 0.7745966692 | 0.5555555556 |
| | 0. | 0.8888888889 |
| | $-$ 0.7745966692 | 0.5555555556 |
| 4 | 0.8611363116 | 0.3478548451 |
| | 0.3399810436 | 0.6251451549 |
| | $-0.3399810436$ | 0.6251451549 |
| | $-$ 0.8611363116 | 0.3478548451 |

**Problema 92** ***(2 puntos)*** *Aproximar el valor de la siguiente integral, utilizando las fórmulas de Legendre para $n = 2$ y $n = 3$*

$$\int_{-1}^{1} \left(x^3 - x^4\right) dx$$

*Cual es el valor exacto de la integral?*

**Problema 93** ***(2 puntos)*** *Se considera para el intervalo $[-1, 1]$, los puntos $x_0 = -0.5$, $x_1 = 0$ y $x_2 = 0.5$ y los pesos $w_0 = w_1 = w_2 = 2/3$. Estos puntos y estos pesos se utilizan para aproximar la integral de una función en $[-1, 1]$. Usar esta fórmula de integración para calcular númericamente la siguiente integral y compararla con el resultado análitico (exacto).*

$$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos(x)dx$$

**Problema 94** ***(2 puntos)*** *Demostrar, utilizando los ceros y pesos asociados a los polinomios de Legendre, cual sería la fórmula de integración numérica de Legendre utilizando un sólo punto de interpolación. Cual sería su exactitud?*

**Problema 95** ***(2 puntos)*** *A partir de los ceros y de los pesos asociados a los polinomios de Legendre, y dado un intervalo $[a, b]$ cualquiera, encontrar los puntos $x_k$, y los pesos $w_k$ que hacen exacta hasta el orden $2N-1$ una fórmula de integración numérica sobre el intervalo $[a, b]$*

**Problema 96** ***(2 puntos)*** *Utilizar el resultado del problema anterior para calcular de forma exacta la siguiente integral*

$$\int_0^1 \left(x^2 - x^3\right) dx$$

Cuando el intervalo $[a, b]$ es infinito es decir $a = -\infty$ ó $b = \infty$, hay que emplear otros métodos para aproximar las integrales. En el caso de $[a, b] = (-\infty, \infty)$ se utilizan los ceros de los denominados polinomios de Hermite definidos como $H_0(x) = 1$, $H_1(x) = 2x$, y

$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x)$$

para $n \geq 2$. En este caso la fórmula de integración numérica aproxima:

$$\int_{-\infty}^{\infty} f(x)e^{-x^2}dx \approx \sum_{k=0}^{N} w_k f(x_k)$$

**Teorema 32** *Si $\tilde{x}_k$ son los ceros del polinomio de Hermite y definimos*

$$\tilde{w}_k = \int_{-\infty}^{\infty} \frac{\Pi_{i\neq k}(x - \tilde{x}_{i)}}{\Pi_{i\neq k}(x_k - \tilde{x}_{i)}} e^{-x^2} dx$$

*entonces la fórmula de integración numérica generada por los puntos $\tilde{x}_k$ y los pesos $\tilde{w}_k$ es exacta hasta el orden $2N-1$ para el intervalo $(-\infty, \infty)$*

**Demostración** [Hu] Pg. 213-214

**Ejemplo 20** *A continuación se exponen algunos valores de raíces $\tilde{x}_k$ y coeficientes $\tilde{w}_k$ en función del grado del polinomio $H_n(x)$*

| $n$ | $\tilde{x}_k$ | $\tilde{w}_k$ |
|---|---|---|
| 1 | 0. | 1. 772 453 851 |
| 2 | $-0.$ 707 106 781 | 0. 886 226 925 5 |
| | 0. 707 106 781 | 0. 886 226 925 5 |

**Problema 97** ***(2 puntos)*** *Calcular de forma exacta la integral*

$$\int_{-\infty}^{\infty} \left(x^3 - x^2\right) e^{-x^2} dx$$

*utilizando los polinomios de Hermite.*

**Problema 98** ***(2 puntos)*** *Aproximar, utilizando dos puntos de aproximación, el valor de la integral:*

$$\int_{-\infty}^{\infty} \frac{1}{1+x^2} dx$$

Para el intervalo $(0, \infty)$ se utilizan los polinomios de Laguerre $L_n(x)$, definidos por $L_0(x) = 1$, $L_1(x) = 1 - x$, y

$$L_n(x) = (2n - 1 - x)L_{n-1}(x) - (n-1)^2 L_{n-2}(x).$$

para $n \geq 2$. En este caso la fórmula de integración numérica aproxima:

$$\int_0^{\infty} f(x)e^{-x} dx \approx \sum_{k=0}^{N} w_k f(x_k)$$

**Teorema 33** *Si $\tilde{x}_k$ son los ceros del polinomio de Laguerre y definimos*

$$\tilde{w}_k = \int_0^{\infty} \frac{\Pi_{i \neq k}(x - \tilde{x}_{i)}}{\Pi_{i \neq k}(x_k - \tilde{x}_{i)}} e^{-x} dx$$

*entonces la fórmula de integración numérica generada por los puntos $\tilde{x}_k$ y los pesos $\tilde{w}_k$ es exacta hasta el orden $2N - 1$. para el intervalo $(0, \infty)$*

**Demostración** [Hu] Pg. 211-213

**Ejemplo 21** *A continuación se exponen algunos valores de raíces $\tilde{x}_k$ y coeficientes $\tilde{w}_k$ en función del grado del polinomio $L_n(x)$*

| $n$ | $\tilde{x}_k$ | $\tilde{w}_k$ |
|---|---|---|
| 1 | 1. | 1. |
| 2 | 0. 585 786 438 | 0. 853 553 390 3 |
| | 3. 414 213 562 | 0. 146 446 609 3 |

**Problema 99** ***(2 puntos)*** *Calcular de forma exacta la integral*

$$\int_0^{\infty} \left(x^3 - x^2\right) e^{-x} dx$$

*utilizando los polinomios de Laguerre.*

**Problema 100** ***(2 puntos)*** *Calcular una fórmula de aproximación numérica de la integral siguiente*

$$\int_a^{\infty} f(x)e^{-x} dx$$

*donde a es un número real cualquiera*

*Fórmulas de Integración Numérica Compuestas*

Con las fórmulas que hemos visto hasta ahora, para aumentar la precisión es necesario aumentar el grado de los polinomios, lo cual resulta complejo para valores grandes de $N$. Una alternativa es dividir previamente la integral en subintegrales de la manera siguiente

$$\int_a^b f(x)dx = \sum_{k=0}^{M} \int_{x_k}^{x_{k+1}} f(x)dx$$

donde $a = x_0 < x_1 < .... < x_{M+1} = b$. A continuación se aproxima numéricamente cada una de las integrales

$$\int_{x_k}^{x_{k+1}} f(x)dx$$

para ello se puede utilizar los desarrollos a partir de los polinomios de Legendre, o bien las fórmulas más simples siguientes:

**Fórmula del rectángulo**

$$\int_{x_k}^{x_{k+1}} f(x)dx \approx f\left(\frac{x_k + x_{k+1}}{2}\right)(x_{k+1} - x_k)$$

Esta fórmula se obtiene fácilmente aproximando $f(x)$ por el polinomio interpolador en $x = \frac{x_k + x_{k+1}}{2}$, es decir

$\int_{x_k}^{x_{k+1}} f(x)dx \approx \int_{x_k}^{x_{k+1}} f\left(\frac{x_k + x_{k+1}}{2}\right) dx =$

$= f\left(\frac{x_k + x_{k+1}}{2}\right)(x_{k+1} - x_k)$

**Fórmula del trapecio**

$$\int_{x_k}^{x_{k+1}} f(x)dx \approx \frac{f(x_{k+1}) + f(x_k)}{2}(x_{k+1} - x_k)$$

Esta fórmula se deduce aproximando $f(x)$ por su polinomio interpolador en $x_k$ y $x_{k+1}$, es decir

$\int_{x_k}^{x_{k+1}} f(x)dx \approx \int_{x_k}^{x_{k+1}} \left(f(x_k)\frac{x - x_{k+1}}{x_k - x_{k+1}} + f(x_{k+1})\frac{x - x_k}{x_{k+1} - x_k}\right) dx$

$= \frac{f(x_{k+1}) + f(x_k)}{2}(x_{k+1} - x_k)$

**Fórmula de Simpson**

$$\int_{x_k}^{x_{k+1}} f(x)dx \approx \frac{f(x_{k+1}) + f(x_k) + 4f\left(\frac{x_k + x_{k+1}}{2}\right)}{6}(x_{k+1} - x_k$$

Esta fórmula se deduce aproximando $f(x)$ por su desarrollo en serie de Taylor centrado en el punto $x_m = \frac{x_k + x_{k+1}}{2}$ es decir

$\int_{x_k}^{x_{k+1}} f(x)dx \approx$

$\int_{x_k}^{x_{k+1}} \left( f(x_m) + f'(x_m)(x - x_m) + \frac{f''(x_m)}{2}(x - x_m)^2 \right) dx =$

$= f(x_m)(x_{k+1} - x_k) + \frac{f''(x_m)}{3} \left( \frac{x_{k+1} - x_k}{2} \right)^3$

ahora bien, teniendo en cuenta los resultados de la sección anterior sobre derivación numérica $f''(x_m)$ se puede aproximar como

$$f''(x_m) \approx \frac{f(x_{k+1}) - 2f(x_m) + f(x_k)}{\left( \frac{x_{k+1} - x_k}{2} \right)^2}$$

por tanto, sustituyendo este valor en la aproximación anterior obtenemos

$\int_{x_k}^{x_{k+1}} f(x)dx \approx f(x_m)(x_{k+1} - x_k) +$

$+ \frac{f(x_{k+1}) - 2f(x_m) + f(x_k)}{3} \left( \frac{x_{k+1} - x_k}{2} \right) =$

$= \frac{f(x_{k+1}) + f(x_k) + 4f\left( \frac{x_k + x_{k+1}}{2} \right)}{6} (x_{k+1} - x_k)$

Aunque estas fórmulas sean menos precisas que las deducidas a partir de los ceros de los polinomios de Legendre, tienen la ventaja de que pueden ser utilizadas cuando conocemos la función a integrar en un conjunto equiespaciado de puntos, es decir sólo conocemos $f(x)$ en un conjunto de la forma $x_k = x_0 + hk$. Nótese que en este caso la integración a partir de los ceros de los polinomios de Legendre no puede utilizarse.

**Problema 101** ***(2 puntos)*** *Aproximar, por el método de Simpson, la integral*

$$\int_{-1}^{1} \left( x^3 - x^4 \right) dx$$

*utilizando únicamente el valor de la función en los puntos:* $-1, -\frac{1}{2}, 0, \frac{1}{2}$ *y* $1$.

**Integración numérica en dimensiones superiores.**

En esta sección estudiaremos las técnicas de integración numérica sobre dominios $\Omega$ de dimensión superior a 1. Para simplificar la exposición supondremos que la dimensión es 2. Es decir, pretendemos aproximar

$$\int_{\Omega} F(x, y)dxdy$$

Aproximaremos esta integral a través de la fórmula numérica:

$$\int_{\Omega} F(x, y)dxdy \approx \sum_{i,j} w_{ij} F(x_i, y_j)$$

donde debemos elegir los puntos $(x_i, y_j)$ y los pesos $w_{ij}$. Para realizar esta elección se utilizan técnicas de cuadratura. Es decir, se exige que la fórmula sea exacta para polinomios en $x, y$ de hasta un cierto grado

$$\int_{\Omega} x^m y^n dxdy = \sum_{i,j} w_{ij} \left( x_i \right)^m \left( y_j \right)^n$$

donde $m$ y $n$ determinan el grado de los polinomios. De estas relaciones se puede deducir, en general, el valor de los puntos y los pesos. Un caso particularmente sencillo es cuando $\Omega$ es un rectángulo $[a, b]x[c, d]$. En este caso, podemos escribir:

$$\int_{\Omega} x^m y^n dxdy = \int_a^b x^m dx \int_c^d y^n dy$$

y por tanto, la exactitud en dimensión 2, la podemos deducir a partir de la exactitud en dimensión 1, que en este caso viene dado, como hemos visto anteriormente por los polinomios de Legendre.

**Problema 102** ***(3 puntos)*** *Deducir la fórmula de integración numérica sobre el rectángulo* $[-1, 1]x[-1, 1]$ *resultante de aplicar la integración numérica en una variable en los intervalos* $[-1, 1]$, *y* $[-1, 1]$.

**Problema 103** ***(2 puntos)*** *Deducir la fórmula de integración numérica sobre un rectángulo* $[a, b]x[c, d]$ *resultante de aplicar la integración numérica en una variable en los intervalos* $[a, b]$, *y* $[c, d]$.

**Problema 104** ***(2 puntos)*** *Calcular de forma exacta la integral*

$$\int_{-1}^{1} \int_{-1}^{1} x^2 y^2 dxdy$$

*utilizando integración numérica.*

Nótese que al igual que en dimensión 1, también podemos extender los resultados al caso en que los intervalos sean infinitos, de tal forma que podemos construir fácilmente fórmulas de integración numérica para las integrales

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(x, y) e^{-x^2 - y^2} dxdy$$

y

$$\int_{0}^{\infty} \int_{0}^{\infty} F(x, y) e^{-x - y} dxdy$$

**Problema 105** ***(2 puntos)*** *Calcular una aproximación numérica de la integral*

$$\int_{-\infty}^{\infty} \int_{0}^{2} \frac{x}{1 + e^{y^2}} dxdy$$

*utilizando la evaluación de* $F(x, y)$ *en* 4 *puntos.*

En el caso de que $\Omega$ sea un triángulo el cálculo es un poco más complejo. Consideremos un triángulo $T$ de vértices $(x_0, y_0)$, $(x_1, y_1)$, $(x_2, y_2)$. Denotaremos por $AREA(T)$ el área del triángulo $T$. En función de los vértices el área viene determinada por

$$AREA(T) = \frac{1}{2} ABS \left( \left| \begin{array}{ccc} 1 & 1 & 1 \\ x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \end{array} \right| \right)$$

A continuación presentaremos algunas fórmulas de integración numérica sobre triángulos utilizando diferente número de puntos

**Integración sobre triángulos utilizando un punto.**

$$\int_T F(x,y) \cong F\left( \frac{x_0+x_1+x_2}{3}, \frac{y_0+y_1+y_2}{3} \right) AREA(T)$$

**Integración sobre triángulos utilizando 3 puntos.**

$$\int_T F(x,y) \cong AREA(T) \sum_{k=1}^{3} w_k F(\widetilde{x}_k, \widetilde{y}_k)$$

donde

$$\begin{array}{rcll} w_1 & = & w_2 = w_3 = \frac{1}{3} & \\ \widetilde{x}_1 & = & \frac{x_0+x_1}{2} & \widetilde{y}_1 = \frac{y_0+y_1}{2} \\ \widetilde{x}_2 & = & \frac{x_0+x_2}{2} & \widetilde{y}_2 = \frac{y_0+y_2}{2} \\ \widetilde{x}_3 & = & \frac{x_2+x_1}{2} & \widetilde{y}_3 = \frac{y_2+y_1}{2} \end{array}$$

**Integración sobre triángulos utilizando 4 puntos.**

$$\int_T F(x,y) \cong AREA(T) \sum_{k=1}^{4} w_k F(\widetilde{x}_k, \widetilde{y}_k)$$

donde

$$\begin{array}{rcll} w_1 & = & w_2 = w_3 = \frac{25}{48} & w_4 = -\frac{27}{48} \\ \widetilde{x}_1 & = & \frac{6x_0+2x_1+2x_2}{10} & \widetilde{y}_1 = \frac{6y_0+2y_1+2y_2}{10} \\ \widetilde{x}_2 & = & \frac{2x_0+6x_1+2x_2}{10} & \widetilde{y}_2 = \frac{2y_0+6y_1+2y_2}{10} \\ \widetilde{x}_3 & = & \frac{2x_0+2x_1+6x_2}{10} & \widetilde{y}_3 = \frac{2y_0+2y_1+6y_2}{10} \\ \widetilde{x}_4 & = & \frac{x_0+x_1+x_2}{3} & \widetilde{y}_4 = \frac{y_0+y_1+y_2}{3} \end{array}$$

**Problema 106** ***(2 puntos)*** *Se considera el triángulo $T$ de vértices $(0,0)$, $(1,0)$ y $(0,1)$. Deducir cual debe ser el punto $(x_0, y_0)$ y el peso $w_0$ para que la fórmula de integración numérica:*

$$\int_T F(x,y)dxdy \approx F(x_0, y_0) w_0$$

*sea exacta para polinomios de grado 1 en $x$ e $y$. Es decir $P(x,y) = ax + by + c$*

**Problema 107** ***(2 puntos)*** *Calcular una aproximación numérica de la integral*

$$\int_\Omega x^2 y dxdy$$

*donde $\Omega$ es el triángulo de vértices $(0,0)$, $(2,0)$ y $(0,2)$ utilizando 1 punto, 3 puntos, y 4 puntos*

## INTERPOLACION DE FUNCIONES II

Esta sección es la continuación natural del tema interpolación de funciones visto anteriormente. Por motivos de coordinación entre los programas teórico y práctico de la asignatura, el tema de interpolación de funciones se dividió en dos partes, siendo ésta la segunda parte.

### Interpolación de Hermite.

En ocasiones resulta de interés interpolar no sólo el valor de la función en ciertos puntos $\{x_i\}_{i=0,..,N}$ sino también el valor de sus derivadas. Un ejemplo clásico de ello es el desarrollo de Taylor de una función en un punto $a$. En este caso aproximamos $f(x)$ por un polinomio de grado $N$, $P_N(x)$ tal que $f(x)$ y $P_N(x)$ poseen las mismas derivadas en el punto $a$ desde el orden 0 hasta el orden $N$.

$$P_N(x) = f(a) + \frac{f'(a)}{1!}(x-a) + .. + \frac{f^{N)}(a)}{N!}(x-a)^N$$

en este caso el error de interpolación viene dado por la fórmula:

$$f(x) - P_N(x) = \frac{f^{N+1)}(\xi)}{(N+1)!}(x-a)^{N+1}$$

donde $\xi$ es un valor intermedio entre $x$ y $a$. En el caso general, donde buscamos un polinomio $P(x)$ tal que él, y todas sus derivadas hasta un cierto orden $M$, coincidan con una función $f(x)$ en los puntos $\{x_i\}_{i=0,..,N}$, se utilizan los denominados polinomios base de Hermite $H_{i,j}(x)$ que son polinomios de grado menor o igual que $(N+1)(M+1)-1$ dados por las siguientes condiciones:

$$\frac{\partial^l H_{i,j}}{\partial x^l}(x_k) = \left\{ \begin{array}{ll} 1 & si \quad l = j \ y \ k = i \\ 0 & \quad\ \ l \neq j \ o \ k \neq i \end{array} \right.$$

a partir de los polinomios base de Hermite el polinomio interpolador de Hermite se define como:

$$P(x) = \sum_{i=0}^{N} \sum_{j=0}^{M} \frac{\partial^j f}{\partial x^j}(x_i) H_{i,j}(x)$$

**Problema 108** ***(3 puntos)*** *Calcular los polinomios base de Hermite que corresponden a tomar como puntos de interpolación $x_0 = -1$, $x_1 = 1$, y el orden de derivación $M = 1$.*

**Interpolación por splines cúbicos.**

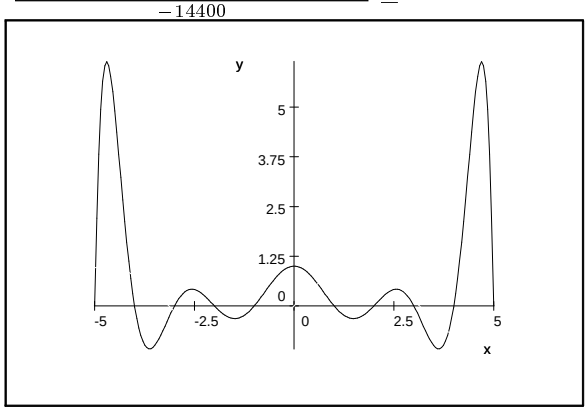
Uno de los problemas básicos del polinomio interpolador de Lagrange, es que para valores grandes de $N$, los polinomios de grado $N$ pueden tener un carácter fuertemente oscilante, y los resultados obtenidos por la interpolación pueden no ser muy satisfactorios como indica el ejemplo siguiente

**Ejemplo 22** *El polinomio base de Lagrange centrado en* 0 *sobre los puntos* $x_i = -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5$ *es*

$$P^0(x) = \frac{(x^2-1)(x^2-4)(x^2-9)(x^2-16)(x^2-25)}{-14400}$$

*tiene un marcado carácter oscilante como muestra su gráfica en el intervalo* $[-5, 5]$.

$\frac{(x^2-1)(x^2-4)(x^2-9)(x^2-16)(x^2-25)}{-14400} =$




Para evitar este problema de oscilaciones de los polinomios de Lagrange, cuando se trabaja con muchos puntos de interpolación se suele interpolar la función utilizando polinomios a trozos, definiendo un polinomio distinto para cada intervalo $[x_i, x_{i+1}]$. La técnica más conocida es la interpolación por splines cúbicos, que son polinomios de grado 3. Por tanto, tendremos un polinomio de grado 3 distinto $P_3^i(x) = d_i(x-x_i)^3 + c_i(x-x_i)^2 + b_i(x-x_i) + a_i$ para cada intervalo $[x_i, x_{i+1}]$. Por tanto si hay $N+1$ puntos el número de polinomios es $N$. Para definir estos polinomios se imponen las siguientes condiciones

$$\begin{aligned}
P_3^i(x_i) &= f(x_i) \quad i = 0, .., N-1 \\
P_3^i(x_{i+1}) &= f(x_{i+1}) \quad i = 0, ..., N-1 \\
\frac{\partial P_3^i}{\partial x}(x_{i+1}) &= \frac{\partial P_3^{i+1}}{\partial x}(x_{i+1}) \quad i = 0, .., N-2 \\
\frac{\partial^2 P_3^i}{\partial x^2}(x_{i+1}) &= \frac{\partial^2 P_3^{i+1}}{\partial x^2}(x_{i+1}) \quad i = 0, ..., N-2
\end{aligned}$$

vamos a introducir la notación $h_i = x_{i+1} - x_i$. Nótese que para definir los polinomios tenemos que buscar $4N$ valores, es decir: $a_0, ...., a_{N-1}, b_0, ..., b_{N-1}, c_0, ....., c_{N-1}, d_0, ...., d_{N-1}$. Por razones técnicas, como veremos posteriormente, vamos a utilizar también los valores $a_N$ y $c_N$.

**Teorema 34** *Si una familia de polinomios* $P_3^i(x) = d_i(x-x_i)^3 + c_i(x-x_i)^2 + b_i(x-x_i) + a_i$, $i = 0, .., N-1$, *satisface las condiciones anteriores entonces:*

$$\begin{aligned}
a_i &= f(x_i) \quad i = 0, .., N \\
d_i &= \frac{c_{i+1} - c_i}{3h_i} \quad i = 0, .., N-1 \\
b_i &= \frac{a_{i+1} - a_i}{h_i} - \frac{h_i(2c_i + c_{i+1})}{3} \quad i = 0, .., N-1
\end{aligned}$$

$$h_{i-1}c_{i-1} + 2(h_{i-1}+h_i)c_i + h_i c_{i+1} = \frac{3(a_{i+1}-a_i)}{h_i} - \frac{3(a_i - a_{i-1})}{h_{i-1}}$$

*para* $i = 1, .., N-1$.

**Demostración** De la condición $P_3^i(x_i) = f(x_i)$ se obtiene de forma inmediata que $a_i = f(x_i)$. De la condición $\frac{\partial^2 P_3^{i+1}}{\partial x^2}(x_{i+1}) = \frac{\partial^2 P_3^i}{\partial x^2}(x_{i+1})$ se obtiene

$$2c_{i+1} = 6d_i h_i + 2c_i$$

de donde despejando obtenemos

$$d_i = \frac{c_{i+1} - c_i}{3h_i}$$

De la Condición $P_3^i(x_{i+1}) = f(x_{i+1})$ se obtiene

$$d_i h_i^3 + c_i h_i^2 + b_i h_i + a_i = a_{i+1}$$

de donde despejando obtenemos:

$$\begin{aligned}
b_i &= \frac{a_{i+1} - a_i}{h_i} - d_i h_i^2 - c_i h_i = \\
&\frac{a_{i+1} - a_i}{h_i} - \frac{h_i(2c_i + c_{i+1})}{3}
\end{aligned}$$

Finalmente de la condición $\frac{\partial P_3^i}{\partial x}(x_i) = \frac{\partial P_3^{i-1}}{\partial x}(x_i)$ se obtiene

$$b_i = 3d_{i-1}h_{i-1}^2 + 2c_{i-1}h_{i-1} + b_{i-1}$$

y despejando todo en función de $c_i$ se obtiene la relación:

$$h_{i-1}c_{i-1} + 2(h_{i-1}+h_i)c_i + h_i c_{i+1} = \frac{3(a_{i+1}-a_i)}{h_i} - \frac{3(a_i - a_{i-1})}{h_{i-1}}$$

Nótese que esta última relación determina un sistema de ecuaciones donde las incógnitas son las variables $c_i$. Dicho sistema tiene $N+1$ incognitas $(c_0, ..., c_N)$ y $N-1$ ecuaciones, para completar dicho sistema hay que añadir una ecuación que involucre a $c_0$ y otra ecuación que involucre a $c_N$. Para añadir estas dos ecuaciones hay dos procedimientos standard. El primero es hacer simplemente $c_0 = c_N = 0$ que significa

$$\begin{aligned}
c_0 &= \frac{\partial^2 P_3^0}{\partial x^2}(x_0) = 0 \\
c_N &= \frac{\partial^2 P_3^{N-1}}{\partial x^2}(x_N) = 0
\end{aligned}$$

El segundo procedimiento se utiliza cuando utilizamos los valores de $f'(a)$, y $f'(b)$. En este caso imponemos:

$$\begin{aligned}\frac{\partial P_3^0}{\partial x}(x_0) &= f'(a)\\ \frac{\partial P_3^{N-1}}{\partial x}(x_N) &= f'(b)\end{aligned}$$

de donde salen las ecuaciones

$$\begin{aligned}f'(a) &= \frac{a_1-a_0}{h_0}-\frac{h_0}{3}(2c_0+c_1)\\ f'(b) &= \frac{a_N-a_{N-1}}{h_{N-1}}-\frac{h_{N-1}}{3}(2c_{N-1}+c_N)\end{aligned}$$

Por tanto, siguiendo con el resultado del teorema anterior, para calcular los splines cúbicos es necesario en primer lugar tomar $a_i = f(x_i)$, a continuación se resuelve un sistema de ecuaciones tridiagonal para el cálculo de los $c_i$, y los $b_j$, y $d_j$ se calcula diréctamente a partir de las relaciones mostradas en el teorema anterior.

**Ejemplo 23** *Vamos a calcular los polinomios interpoladores utilizando splines cúbicos al interpolar la función $f(x)$ en los puntos $x = 0, 1, 2,$ y 3 sabiendo que $f(0) = 0$, $f(1) = 1$, $f(2) = 0$, $f(3) = 2$, tomando $c_0 = c_3 = 0$. en este caso $h_i = 1$. Debemos definir 3 polinomios distintos que corresponden a los intervalos $[0,1]$, $[1,2]$, y $[2,3]$. Los términos $a_i$ vienen dados por*

$$\begin{pmatrix} a_0\\ a_1\\ a_2\\ a_3\end{pmatrix} = \begin{pmatrix} 0\\ 1\\ 0\\ 2\end{pmatrix}$$

*el sistema que debemos resolver para calcular los $c_i$ es*

$$\begin{pmatrix} 4 & 1\\ 1 & 4\end{pmatrix}\begin{pmatrix} c_1\\ c_2\end{pmatrix} = \begin{pmatrix} -6\\ 9\end{pmatrix}$$

*cuya solución es* $\begin{pmatrix} c_1\\ c_2\end{pmatrix} = \begin{pmatrix} -2.2\\ 2.8\end{pmatrix}$.

*los valores $b_i$ y $d_i$ vienen dados por*

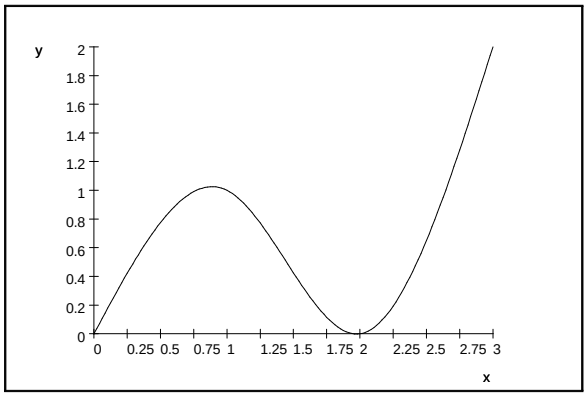
$$\begin{aligned}d_0 &= \frac{-2.2-0}{3} = -0.733\\ d_1 &= \frac{2.8+2.2}{3} = 1.667\\ d_2 &= \frac{0.-2.8}{3} = -0.933\end{aligned}$$

$$\begin{aligned}b_0 &= 1-\frac{-2.2}{3} = 1.733\\ b_1 &= -1-\frac{-4.4+2.8}{3} = -0.467\\ b_2 &= 2-\frac{5.6+0}{3} = 0.133\end{aligned}$$

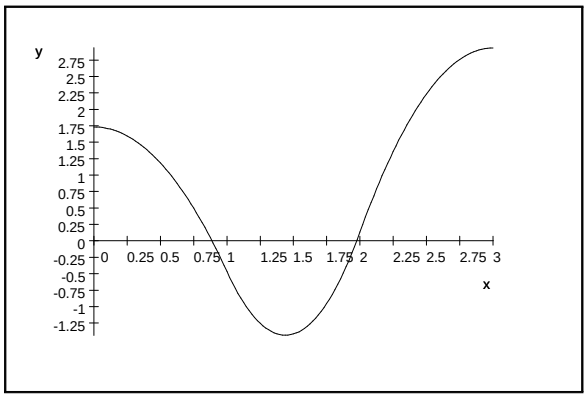
***Ejemplo 24*** *por tanto los polinomios son:*

$$\begin{aligned}P_0(x) &= -0.733x^3+1.733x\\ P_1(x) &= 1.667\,(x-1)^3-2.2\,(x-1)^2-0.467\,(x-1)+1\\ P_2(x) &= -0.933\,(x-2)^3+2.8\,(x-2)^2+0.133\,(x-2)\end{aligned}$$

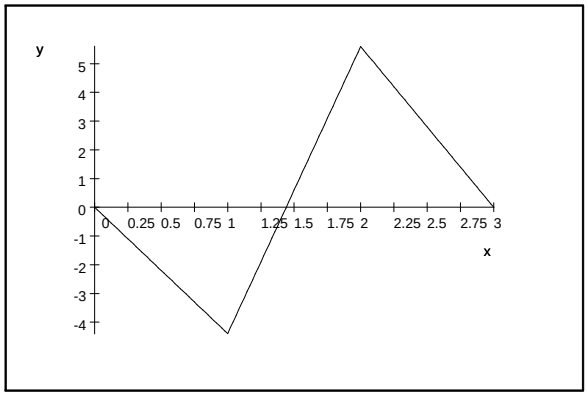
*a continuación se muestra una gráfica con los 3 polinomios concatenados en el intervalo $[0,3]$*




*como puede observarse, por las condiciones sobre las derivadas que hemos impuesto, no es posible distinguir geométricamente al trazar la curva, cuales son los puntos de unión entre los tres polinomios, es decir parece a simple vista el trazado de una única función. Veamos ahora gráficamente el perfil de la derivada de los polinomios $P_0(x)$, $P_1(x)$, y $P_2(x)$*




*como puede observarse, tampoco sobre la derivada se aprecian los puntos de unión de los polinomios. Sin embargo, sobre la gráfica de la derivada segunda los puntos de unión se detectan en los lugares donde encontramos un pico, tal y como se muestra en la gráfica de la derivada segunda siguiente:*



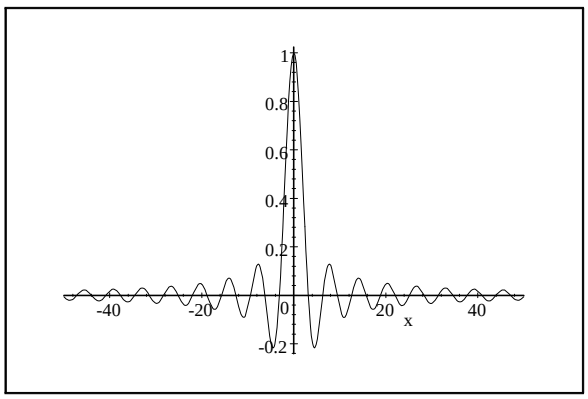

**Problema 109** ***(3 puntos)*** *Calcular los polinomios que determinan la interpolación por splines cúbicos de la función* $f(x) = \sin\left(\frac{\pi}{2}x\right)$ *para los puntos* $x = -1, 0, 1, 2$

**La interpolación a través de la función seno cardinal.**

Una base de funciones interpolantes muy utilizada en la teoría de Fourier es la base formada a partir de la función seno cardinal definida por:

$$\sin c(x) = \frac{\sin(x)}{x}$$

cuya gráfica es



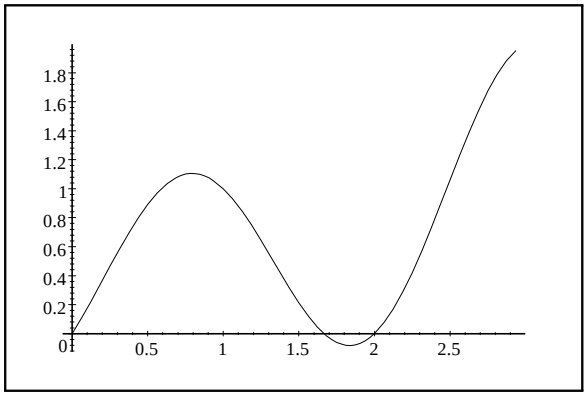

esta función tiene la propiedad de que en $x = 0$, $\sin c(0) = 1$, y para cualquier entero $i$ distinto de 0, $\sin c(\pi i) = 0$. Dada una función $f(x)$, su función interpolante en los puntos $x_i = a \cdot i$ par $i = M, ..., N$ viene dada por la función:

$$\widetilde{f}(x) = \sum_{i=M}^{N} f(x_i) \frac{\sin(\pi\left(\frac{x}{a} - i\right))}{\pi\left(\frac{x}{a} - i\right)}$$

**Ejemplo 25** *Consideremos la función* $f(x)$ *definida en los puntos* $x = 0, 1, 2,$ *y* 3 *tal que* $f(0) = 0$, $f(1) = 1$, $f(2) = 0$, $f(3) = 2$ *la interpolación de esta función utilizando la función seno cardinal viene dada por la función*

$$\widetilde{f}(x) = \frac{\sin(\pi\,(x-1))}{\pi(x-1)} + 2\frac{\sin(\pi\,(x-3))}{\pi(x-3)}$$

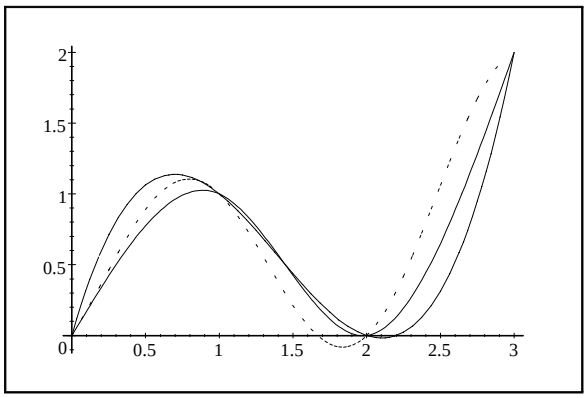
*cuya gráfica es*



**Ejemplo 26** *Vamos a comparar gráficamente el resultado de interpolar la función del ejemplo anterior utilizando la interpolación de Lagrange normal, la interpolación por splines cúbicos, y la interpolación a través de la función seno cardinal. El polinomio interpolador de Lagrange, se puede calcular fácilmente y da como resultado*

$$P(x) = x - x(x-1) + \frac{5}{6}x(x-1)(x-2)$$

*en la siguiente figura se muestran juntas las gráficas del polinomio de Lagrange (línea a trozos), los polinomios de la interpolación por splines cúbicos (línea sólida), y la interpolación utilizando la función* $\sin c(x)$ *(línea a trozos).*



como puede observarse, la interpolación por splines cúbicos es la menos oscilante. Por otro lado, cuando el número de puntos de interpolación aumenta, mayor va a hacer la diferencia entre los diferentes tipos de interpolación.

**Problema 110** ***(2 puntos)*** *Calcular la función que interpola, utilizando la función* $\sin c(x)$ *a la función* $f(x) = \sin(x)$ *en los puntos* $x = -\pi, -\frac{\pi}{2}, 0, \frac{\pi}{2}, \pi$.

**La interpolación a través de polinomios trigonométricos.**

La base de la transformada de Fourier discreta es la utilización de los polinomios trigonométricos dados por la expresión

$$P^k(x) = e^{ikx}$$

dada una función $f(x)$ definida en el intervalo $[-\pi, \pi]$, pretendemos aproximar $f(x)$ como

$$f(x) \approx \sum_{k=-N}^{N} c_k e^{ikx}$$

donde $c_k$ son coeficientes en general complejos. El siguiente resultado determina la forma de calcular dichos coeficientes $c_k$

**Teorema 35** *Los coeficientes $c_k$ que minimizan el error cuadrático medio:*

$$E(c_{-N}, ..., c_N) = \int_{-\pi}^{\pi} \left( f(x) - \sum_{k=-N}^{N} c_k e^{ikx} \right)^2 dx$$

*vienen dados por*

$$c_k = \frac{\int_{-\pi}^{\pi} f(x) e^{ikx} dx}{2\pi}$$

**Demostración** En primer lugar observamos que dada la forma cuadrática del funcional $E(c_{-N}, ..., c_N)$, este debe poseer mínimos. Por otro lado, en un mínimo, las derivadas parciales de $E(c_{-N}, ..., c_N)$ con respecto a cualquier $c_k$ son cero, y por tanto:

$$\frac{\partial E}{\partial c_k}(c_{-N}, ..., c_N) = \int_{-\pi}^{\pi} \left( f(x) - \sum_{l=-N}^{N} c_l e^{ilx} \right) e^{ikx} dx = 0$$

y el resultado del teorema sale de forma inmediata teniendo en cuenta que

$$\int_{-\pi}^{\pi} e^{ilx} e^{ikx} dx = \begin{cases} 2\pi & si \quad l = k \\ 0 & si \quad l \neq k \end{cases}$$

**Ejemplo 27** *Consideremos la función:*

$$f(x) = \begin{cases} 1 & si \quad x \in [-\frac{\pi}{2}, \frac{\pi}{2}] \\ 0 & si \quad x \notin [-\frac{\pi}{2}, \frac{\pi}{2}] \end{cases}$$

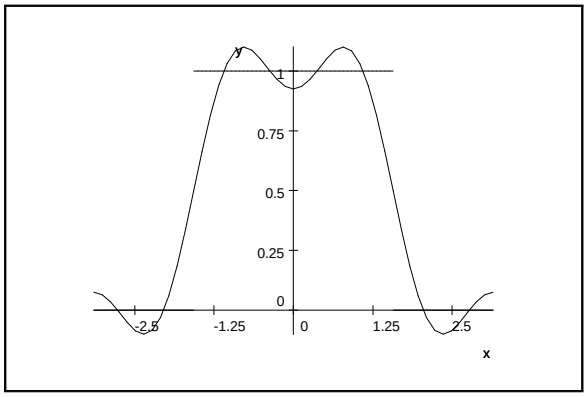
*vamos a calcular el polinomio trigonométrico interpolante para $N = 3$, los valores de $c_k$ son:*

$$c_0 = \frac{\int_{-\pi}^{\pi} f(x) dx}{2\pi} = \frac{1}{2}$$
$$c_1 = c_{-1} = \frac{\int_{-\pi}^{\pi} f(x) e^{ix} dx}{2\pi} = \frac{1}{\pi}$$
$$c_2 = c_{-2} = \frac{\int_{-\pi}^{\pi} f(x) e^{2ix} dx}{2\pi} = 0$$
$$c_3 = c_{-3} = \frac{\int_{-\pi}^{\pi} f(x) e^{3ix} dx}{2\pi} = -\frac{1}{3\pi}$$

*por tanto, el polinomio trigonométrico interpolador es*

$$P_3(x) = \frac{1}{2} + \frac{2}{\pi}\cos(x) - \frac{2}{3\pi}\cos(3x)$$

*La siguiente gráfica muestra la aproximación entre $f(x)$ y $P_3(x)$.*




**Problema 111** ***(3 puntos)*** *Calcular el polinomio trigonométrico tomando $N = 2$, que interpola a la función $f(x) = |x|$ en el intervalo $[-\pi, \pi]$.*

**Aproximación por mínimos cuadrados.**

La aproximación mínimo cuadrática aproxima, a través de una función, un conjunto de valores de forma global, sin exigir que la función aproximante pase exactamente por ese conjunto de valores.

Dado un conjunto de valores $\{(x_i, y_i)\}_{i=1,..,N}$ la aproximación mínimo cuadrática lineal consiste en buscar la recta $y = ax + b$, tal que la función de error cuadrático

$$E(a, b) = \sum_{i=1}^{N} (ax_i + b - y_i)^2$$

sea mínima.

**Teorema 36** *Los valores $a, b$ que minimizan el error cuadrático anterior son*

$$a = \frac{N \sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N \sum_{i=1}^{N} x_i^2 - \left( \sum_{i=1}^{N} x_i \right)^2}$$
$$b = \frac{\sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} x_i y_i \sum_{i=1}^{N} x_i}{N \sum_{i=1}^{N} x_i^2 - \left( \sum_{i=1}^{N} x_i \right)^2}$$

**Demostración** En primer lugar observamos que dada la forma cuadrática que tiene el funcional, debe poseer un mínimo, además, en un punto de mínimo del funcional $E(a,b)$ las derivadas parciales son cero, y por tanto

$$\begin{aligned}\frac{\partial E}{\partial a}(a,b) &= 2\sum_{i=1}^{N}(ax_i+b-y_i)\,x_i = 0\\ \frac{\partial E}{\partial b}(a,b) &= 2\sum_{i=1}^{N}(ax_i+b-y_i) = 0\end{aligned}$$

ello da lugar a un sistema lineal de ecuaciones cuyas incógnitas son $a, b$ y cuya resolución lleva al resultado establecido en el teorema

**Problema 112** ***(2 puntos)*** *Calcular la aproximación mínimo cuadrática lineal de la tabla*

| $x_i$ | $y_i$ |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 0 |
| 3 | 2 |

## BIBLIOGRAFIA BASICA


[Bu-Fa] Burden R., Faires D. "Análisis Numérico", Grupo Editorial Iberoamérica 1985. Esta obra es un clásico del Cálculo Numérico, destaca por una exposición simple y al mismo tiempo clara, con múltiples ejemplos y una descripción de los algoritmos bien diseñada.

[Bo] Borse G. "Programación en fortran 77" Anaya, 1989. En esta obra se presenta el lenguaje de programación fortran 77 con numerosas aplicaciones al análisis numérico.

[Ci] Ciarlet P.G. "Introduction à l'analyse numérique matricielle et à l'optimisation", Masson , 1990. Con un exquisito rigor se abordan los temas básicos del Análisis Numérico Matricial y métodos de optimización, incluyendo la resolución de sistemas a través de métodos directos, iterativos y métodos tipo gradiente, así como el cálculo de autovalores y vectores propios.

[Hi] Higham N. "Accuracy and Stability of Numerical Algorithms", SIAM, 1996 Esta obra, muy reciente, da una visión general sobre los últimos avances en Análisis Numérico, haciendo especial énfasis en la precisión de los algoritmos numéricos y en la propagación de errores, también resulta de interés la descripción de las aritméticas que utilizan los ordenadores más recientes como la aritmética Standard de I.E.E.E..

[Hu] Hultquist P. F. "Numerical Methods for Engineers and Computer Scientists", The Benjamin/Cummings Publishing Company, Inc. 1988. Esta obra, presenta una cuidada selección de temas básicos en Análisis Numérico, sin pretender ser tan exhaustiva como otras obras de carácter más general, la buena presentación de los temas elegidos la hacen de interés.

[Is-Ke] Isaacson E., Keller H. "Analysis of Numerical Methods". John Wiley and Sons, 1966. Uno de los libros clásicos más conocidos en Análisis Numérico. Destaca por el rigor matemático en su exposición.

[Ki-Ch] Kincaid D., Cheney W. "Análisis Numérico". Addison-Wesley Iberoamericana, 1994. Excelente libro de base para un curso de Métodos Numéricos. Contiene todos los tópicos habituales con una descripción muy completa y detallada. Los algoritmos están muy bien descritos a través de un seudocódigo. Trae una buena selección de problemas.

[La-Th] Lascaux P., Théodor R. "Analyse numérique matricielle appliquée à l'art de l'ingénieur. Vol. 1 Méthodes directes y Vol. 2 Méthodes itératives ", Masson, 1993. Esta obra, dividida en dos volúmenes, trata en profundidad todos los tópicos relacionados con el Análisis Numérico Matricial. Su mayor virtud es el rigor matemático con el que se tratan los temas y una cuidada presentación.

[St] Stewart G.W. "Afternotes on Numerical Analysis" SIAM, 1996. Esta obra, sin pretender ser exhaustiva, muestra las últimas tendencias en cuanto a la enseñanza de los conceptos básicos del Análisis Numérico.

## APENDICE A: Resumen de los comandos de UNIX

En este breve resumen seguiremos el siguiente esquema. En primer lugar aparece el comando UNIX, a continuación entre paréntesis su equivalente en MS-DOS (si existe), y finalmente un comentario y un ejemplo.

**cd** (cd) cambia el directorio activo
>cd /users/p701/fortran77
**more** (type) visualiza el contenido de un fichero
>more /users/p701/fortran77/programas/prog1.f
**ls** (dir) Visualiza contenido de un directorio
>ls /users/p701/fortran77
**cp** (copy) Copia un fichero en otro.
>cp /users/p701/fortran77/programas/prog1.f .
**rm** (del) borra un fichero
>del prog1.f
**man** (help) suministra ayuda sobre un comando
> man ls
**logout** Se termina la sesión y se sale del sistema
>logout
**ps** Visualiza los números de procesos que están abiertos que corresponden al usuario *alumno*
>ps -u *alumno*
**kill** Interrumpe la ejecución de un proceso de numero *Nproceso*
>kill -9 *Nproceso*
**mkdir** (mkdir) Crea un directorio
>mkdir practica1
**rmdir** (rmdir)Borra un directorio
>rmdir practica1
**mv** (move) Cambia de nombre o ubicación un archivo.
>mv prog1.f practica1.f
**chmod** Cambia los permisos de lectura, escritura y ejecución de un fichero. Este comando es de utilidad para salvaguardar la información de directorios y ficheros de miradas ajenas.
Hacer > *man* chmod para mirar las opciones.
**chown** Cambia el propietario de un fichero
Hacer > *man* chown para mirar las opciones.
**du** (tree) Visualiza la cadena de directorios
>du /users/p701
**find** Busca un archivo de nombre *file* en el directorio *dir*
>find *dir* -name *file* -print
**grep** Busca los ficheros que contenga la cadena de caracteres *string*
>grep *string* *

## APENDICE B: Resumen del procesador de texto *vi*.

El procesador de texto *vi*, tiene la ventaja de estar presente en cualquier máquina que trabaje sobre UNIX, y no requiere ningún entorno gráfico. Puede ejecutarse en dos modos. El modo comando (el que está por defecto al entrar en *vi*). Donde se ejecutan comandos como copy-paste, ect.. y el modo edición que es donde se escribe normalmente el texto.

**Intercambio entre modo comando y modo edición**
**ESC** Pasa de modo edición a modo comando
**i** Pasa de modo comando a modo edición
**A** Pasa a modo edición y pone el cursor al final de la línea
**O** inserta una nueva línea, pasa a modo edición y pone el cursor al principio de la nueva línea.

**Manejo de Ficheros (En modo comando)**
**:w** escribe en disco el fichero
**:wq** escribe en disco el fichero y sale del *vi*
**:e** *fichero.name* edita el fichero *fichero.name*
**:q!** sale del *vi* sin guardar cambios.
**:w** *fichero.name* escribe el fichero actual en el fichero *fichero.name* en disco
**!***comando* ejecuta el comando UNIX *comando*
**:set nu** presenta los números de linea en pantalla

**Comandos para desplazarse por el texto (En modo comando)**
**Crtl F** Página Adelante
**Crtl B** Página Atrás
**$** Pone el cursor en el final de la línea
0 Pone el cursor en el principio de línea
*/string* Busca hacia adelante el string *string*
**?***string* Busca hacia atras el string *string*
**n** Repite la última búsqueda
**G** Va al final del texto
3 **G** Va a la línea número 3.

**Comandos para borrar líneas o caractéres (En modo comando)**
**x** borra el caracter donde se encuentra el cursor
**r** *character* reemplaza el caracter donde se encuentra el cursor por el caracter *character*.
**dd** borra la línea donde se encuentra el cursor.
3 **dd** borra 3 líneas desde donde se encuentra el cursor hacia abajo.
**dw** borra la palabra donde se encuentra el cursor.

**Comandos para copiar y desplazar bloques (En modo comando)**
**yy** copia en el buffer la linea donde se encuentra el cursor
**3yy** copia en el buffer 3 líneas hacia abajo desde el cursor
**dd** copia (y borra) al buffer la linea donde se encuentra el cursor
**3dd** copia (y borra) al buffer 3 líneas hacia abajo desde el cursor
**p** copia el contenido del buffer en el texto.

## APENDICE C: Algunos fallos comunes en Fortran

1. No poner $END$ al final del programa principal o de la función
2. Escribir números como $1/2$ ó $10**20$ en precisión entera. Solución: Escribir $1./2.$ó $10.**20.$
3. Utilizar variables enteras como flotantes o al revés. Sugerencia: Aunque no sea necesario, declarar los tipos de todas las variables que se utilicen al principio del programa o función.
4. Utilizar un parámetro de una función para asignar dinámicamente memoria a un vector o matriz en el interior de la función. Solución: Poner una declaración de PARAMETER al principio de la función y con ella asignar las memorias de forma estática.
5. No poner ningún comentario en los programas
6. Anidar excesivamente los programas. Siempre hay que buscar que el número de anidamientos sea mínimo.
7. No respetar los tipos en los pasos de parámetros de las funciones.
8. Utilizar vectores sin declararlos con la sentencia DIMENSION
9. No pasar la dimensión de un vector como parámetro de una función.
10. Exceso de sentencias $GOTO$. Las sentencias $GOTO$ pueden dificultar el seguimiento del flujo del programa y sólo hay que utilizarlas cuando sean indispensables.
11. A veces los programas pueden fallar por errores de redondeo en los cálculos. Fortran da la posibilidad de cambiar el tamaño de bits utilizados para almacenar las variables en el momento de la compilacion. Por ejemplo si hacemos "f77 -r$n$ prueba.f -o prueba" donde $n$ es 8 o 16, aumentaremos la precisión de la aritmética para las variables reales, análogamente,si en lugar de utilizar la directiva -r$n$ utilizamos -d$n$ aumentaremos la precisión de las variables declaradas DOUBLE PRECISION, y si utilizamos -i$n$ las variables enteras.