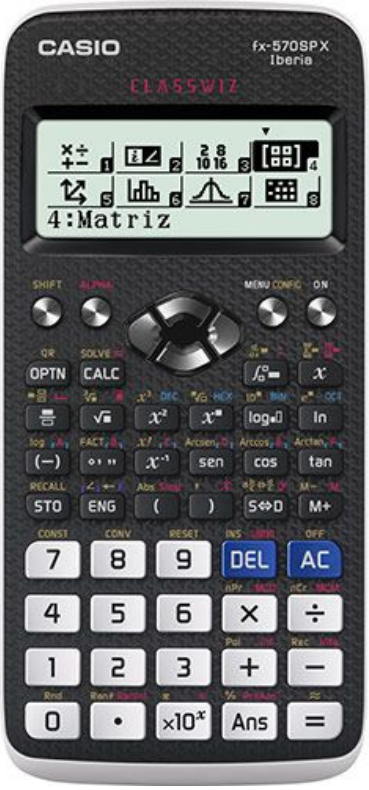
COMPUTACIÓN NUMÉRICA

COMA FLOTANTE Y ÁLGEBRA LINEAL EN MATLAB®



Edgar Perez Blanco

DNI: 46483191F

Grupo: 11 (Opción B)

CN - GEI FIB

Primavera 2019-2020

Contenidos

1. Norma IEEE - 754

Introducción

En cualquier ámbito desde llevar la contabilidad de un pequeño comercio hasta predecir la órbita de un satélite, se deben llevar a cabo cálculos que requieren de una precisión superior a la que los números enteros nos proporcionan, lo que nosotros entendemos como números reales. Los computadores tienen dificultades para tratar con ellos debido a la discretización inherente de estos. Resulta imposible comprender cómo con una tira de 64 ceros y unos podemos representar el número Pi, el cual como sabemos, cuenta con infinitos decimales.

El diseño inicial de los computadores, como todo gran proyecto, empezó con versiones simplificadas que, con el paso del tiempo, han ido mejorando poco a poco añadiendo a demanda de las necesidades del mismo ser humano, nuevas funcionalidades. Dado que en los inicios de la industria tecnológica las diferentes empresas privadas luchaban fabricar el mejor producto, los diseños de estos no eran públicos y por lo tanto, tampoco estandarizados. Las compañías presentaban sus propuestas y los usuarios decidían con cual de ellas se quedaban. Esta resultó ser caótica por una sencilla razón: todo aquellos programas que hiciesen funcionar sobre una máquina, funcionaban exclusivamente en esa máquina y ninguna otra más, lo cual ralentizaba la evolución de las tecnologías y dificultaba las tareas a los consumidores.

Coma Flotante

El Instituto de Ingeniería Eléctrica y Electrónica (IEEE) en 1985 decidió imponer el orden este caos creando una norma estándar para la representación de los números reales. El IEEE decidió utilizar la representación en coma flotante. Este se basa en dividir una tira de bits en dos partes (dos números):

- **Mantisa (o significado):** Número que contiene los dígitos del número. Si el número de la mantisa es negativo, entonces el número que representa el conjunto es negativo.
- **Exponente:** Indica en qué posición se lo coloca la coma o punto decimal en relación al inicio de la mantisa. Si el número del exponente es negativo, representa que el número es menor que uno.

A continuación un ejemplo que clarifica la explicación (en base decimal):

Mantisa	Exponente	Notación científica	Valor en punto fijo
1.5	4	$1.5 \cdot 10^4$	15000
-2.001	2	$-2.001 \cdot 10^2$	-200.1
5	-3	$5 \cdot 10^{-3}$	0.005
6.667	-11	6.667e-11	0.0000000000667

El estándar IEEE - 754

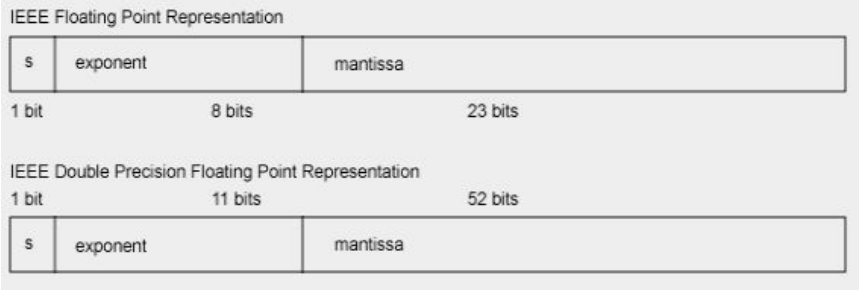
El sistema de representación en coma flotante se puede aplicar de diferentes formas, ya sea variando el número de bits reservado para cada parte, la representación del exponente, la base, etc. Centrándonos en el estándar, contamos con diferentes variantes dependiendo de la cantidad de bits usados (comúnmente llamado “nivel de precisión”). Los parámetros establecidos son los siguientes:

Formato	Bits totales	Bits significativos	Bits del exponente	Número más pequeño	Número más grande
Precisión sencilla	32	23 + 1 signo	8	$\sim 1.2 \cdot 10^{-38}$	$\sim 3.4 \cdot 10^{38}$
Precisión doble	64	52 + 1 signo	11	$\sim 5.0 \cdot 10^{-324}$	$\sim 1.8 \cdot 10^{308}$

Dada una tira de bits, para el caso de precision sencilla contamos com:

- (1 Bit) **Signo**: 0 positivo y 1 negativo
- (8 Bit) **Exponente** normalizado: Se normaliza el exponente entre -128 y +127
- (23 bit) **Mantisa**: En binario es el numero 1.xxxxxxx … donde el primer 1 se asume por no ser fraccionario y no se representa.

Para doble precisión todo funciona de forma análoga con la diferencia de que, debido al aumento de rango, el exponente se debe normalizar entre -1024 y +1023.



Representación gráfica de las tiras de bits en simple y doble precisión.

Una cantidad importante asociada al sistema de coma flotante es el conocido como epsilon de la máquina. Esta cantidad representa la distancia entre 1 y el siguiente número en coma flotante representable. Como hemos explicado anteriormente, la aritmética en coma flotante cuenta con precisión finita, y por lo tanto entre números sucesivos existen huecos de números no representables.

Números Normales

Los números representables son denominados "**Números de coma flotante normalizados**" y son aquellos que se pueden representar mediante la siguiente expresión:

$$x = \pm(1 + f) \cdot 2^e$$

Donde ***f*** es la mantissa (parte fraccionaria entre 0 y 1) y ***e*** es el exponente (dentro del rango de exponentes representados según el nivel de precisión).

Números Denormales

Los números reales cuyo módulo sea más pequeño que el número normal más pequeño (anteriormente descritos), pertenecen al grupo de los "**Números de coma flotante denormales**". Por ejemplo, el mismo cero es un número denormal, ya que este no cuenta con la parte entera que asumimos implícita (1 + ***f***) en la mantisa. Estos números son guardados en memoria sin el primer bit implícito y son reconocibles por utilizar el exponente más pequeño que se puede utilizar:

$$x = \pm f \cdot 2^{-emax+1}$$

MATLAB y la Coma Flotante

El lenguaje de programación MATLAB hace uso del sistema de representación en coma flotante de doble precisión (64 bit). Este es cuenta con los siguientes límites:

Name	Binary	Decimal
eps	2^(-52)	2.2204e-16
realmin	2^(-1022)	2.2251e-308
realmax	(2-eps)*2^1023	1.7977e+308

El software es capaz de detectar cuando un número va a dar un resultado por encima del máximo representable, overflow; o por debajo del mínimo representable, underflow. En caso de overflow, el resultado del programa es expresado como *Inf* (Infinito), y en caso de underflow, dependiendo de la máquina, o bien se expresa como un número denormal (si estos están soportados) o de lo contrario se redondea a 0. Este también trata resultados especiales como *0/0* o *Inf - Inf* como valores "no numéricos", representados como *NaN* (Not A Number).

Un dato importante es que MATLAB utiliza el sistema de coma flotante para representar los enteros; por lo tanto, a diferencia de lenguajes de programación, para este los valores a asociados a *3* y a *3.0* son iguales.

2. Representación de números en coma flotante

L'algoritme següent de MATLAB® talla la representació a t xifres d'un nombre en coma flotant x.

```
function y=tallar(x,t)
   if (x==0), y=0; return; end
   e=fix(log10(abs(x)));
   if e>=0, e=e+1; end
   y=fix(x*10^(t-e))*10^(e-t);
return
```

1. Comprova el funcionament amb un joc de proves. Per exemple
   (a) $t = 5$, $x = 123.041$ i $x = 123.046$.
   (b) $t = 5$, $x = -123.041$ i $x = -123.046$.
   (c) $t = 2$, $x = 0.00123$ i $x = 0.00128$.
   (d) $t = 2$, $x = -0.00123$ i $x = -0.00128$.
   (e) $t = 4$, $x = \pi$ i $x = -\pi$.

Para realizar las comprobaciones simplemente he realizado una serie de llamas a la función *tallar* con los parámetros indicados, y he corroborado visualmente que a partir del número de dígitos indicado, todo son ceros. Este chequeo visual lo he realizado con los siguientes resultados:

t	x1	x2	t1	t2
5	1.230410000000000e+02	1.230460000000000e+02	1.230400000000000e+02	1.230400000000000e+02
5	-1.230410000000000e+02	-1.230460000000000e+02	-1.230400000000000e+02	-1.230400000000000e+02
2	0.001230000000000	0.001280000000000	0.001200000000000	0.001200000000000
2	-0.001230000000000	-0.001280000000000	-0.001200000000000	-0.001200000000000
4	3.141592653589793	-3.141592653589793	3.141000000000000	-3.141000000000000

Los diferentes valores proporcionados son denominados como x_i y los resultados del corte para cada valor son denominados t_i. A simple vista podemos ver como para cada entrada de las columnas t_1 y t_2, sólo se mantienen t cifras significativas sin aplicar ningún tipo de redondeo ni truncamiento.

2. Escriu una rutina semblant per arrodonir a t xifres un nombre en coma flotant x

Para realizar este apartado simplemente he hecho uso de una función integrada en el propio MATLAB. La función *round* ofrece posibilidad de redondear manteniendo *t* dígitos significativos pasándole como parámetro el número de dígitos significativos y la palabra clave *'significant'*.

```
function y = myRound(x,t)

    y = round(x,t,'significant');

end
```

3. Comprova el funcionament amb un joc de proves.

   (a) $t = 5$, $x = 123.041$ i $x = 123.046$.
   (b) $t = 5$, $x = -123.041$ i $x = -123.046$.
   (c) $t = 2$, $x = 0.00123$ i $x = 0.00128$.
   (d) $t = 2$, $x = -0.00123$ i $x = -0.00128$.
   (e) $t = 4$, $x = \pi$ i $x = -\pi$.

Con el fin de comprobar el correcto funcionamiento de la función anteriormente descrita he llevado a cabo la misma metodología que en el primer apartado. Con simples llamadas a la función con los parámetros dados he obtenido los siguientes resultados:

	t	x1	x2	t1	t2
1	5	1.230410000000000e+02	1.230460000000000e+02	1.230400000000000e+02	1.230500000000000e+02
2	5	-1.230410000000000e+02	-1.230460000000000e+02	-1.230400000000000e+02	-1.230500000000000e+02
3	2	0.001230000000000	0.001280000000000	0.001200000000000	0.001300000000000
4	2	-0.001230000000000	-0.001280000000000	-0.001200000000000	-0.001300000000000
5	4	3.141592653589793	-3.141592653589793	3.142000000000000	-3.142000000000000

De nuevo, a simple vista podemos comprobar que se mantienen *t* dígitos significantes pero en este caso, en vez de haberse realizado un corte, podemos ver que se realiza un redondeo según el criterio estándar. En todas las filas, a excepción de la última, en la columna t_1 obtenemos un valor inferior respecto al análogo en la columna t_2 debido al redondeo. En la última fila podemos observar como funciona correctamente tanto con números positivos como negativos.

3. Evaluación de expresiones

Les dues expressions següents:

$$f(x) = 1.01e^{4x} - 4.62e^{3x} - 3.11e^{2x} + 12.2e^{x} - 1.99\,,$$
$$F(x) = (((1.01z - 4.62)z - 3.11)z + 12.2)z - 1.99\,, \quad z = e^{x}\,.$$

són dues fórmules diferents per avaluar la mateixa funció.

Para la realización de este ejercicio he creado dos funciones de MATLAB que encapsulan cada una de las expresiones. Estas funciones reciben como parámetro el valor *x* y un entero *n* para evaluar la expresión en *x* con una aritmética limitada a *n* cifras por redondeo.

Cada vez que se realiza una operación (suma, resta, multiplicación, división) se aplica el redondeo, tanto a los diferentes factores/sumandos como al resultado de la operación. Las potencias, las he calculado factorizadas, por ejemplo: para calcular X^3 he realizado *redondeo(redondeo(redondeo(x) * redondeo(x)))*; y así para todas ellas, a excepción de aquellas en las que el exponente no era un número entero, en las que he utilizado la función integrada de elevar a un número en el lenguaje para posteriormente aplicar el redondeo.

1. Fent ús de l'aritmètica de **tres xifres** arrodonint calculeu el valor de les dues expressions per a $x = 1.53$. Per què donen diferent $f(1.53)$ i $F(1.53)$?

En este apartado he utilizado la metodología explicada previamente (función encapsulada) pasando como parámetro el valor de *x = 1.53* y *n = 3* para realizar el las operaciones en aritmética de tres cifras. A continuación se muestran los resultados obtenidos:

f(x,n)	-6.79
F(x,n)	-7.07

Los resultados obtenidos son diferentes debido a que la primera expresión *f*, realiza un número total de operaciones (13) superior al de la segunda expresión *F* (10). Esto sería irrelevante si no tuviésemos en cuenta que, por cada operación realizada, se produce un redondeo a 3 cifras y por lo tanto una pérdida de información. Al haber realizado mas redondeos con una expresion que con otra, es totalmente coherente que se hayan obtenido resultados diferentes.

2. Fent ús de l'aritmètica de **quatre xifres** arrodonint calculeu el valor de les tres expressions per a $x = 0.925$. Per què donen diferent $f(0.925)$ i $F(0.925)$?

Para responder a este apartado he seguido exactamente la misma metodología que en el anterior pero con parámetros distintos: *x = 0.925* y *n = 4*. Estos son los resultados obtenidos:

f(x,n)	-24.25
F(x,n)	-24.26

De nuevo, como he explicado en el apartado anterior, debido a la diferente cantidad de redondeos realizada en cada una de las evaluaciones, surgen diferencias entre los resultados por pérdida de información.

3. Calculeu en cada cas l'error relatiu percentual. Quina expressió dona una millor aproximació?

En este apartado he aplicado las fórmulas estándar del error tomando como valor exacto o real el proporcionado por la segunda expresión *F* en aritmética de MATLAB (coma flotante de doble precisión). He utilizado *F* porque esta se corresponde con la expresión obtenida mediante la aplicación del Algoritmo de Horner (a pesar de esta no ser una función polinomial al cien por cien, ya que tienes partes exponenciales), el cual permite evaluar la función con mayor estabilidad gracias a reducir el número de multiplicaciones realizadas.

Teniendo en cuenta el primer apartado, el error porcentual de *F* es de alrededor del 7% mientras que el de *f* se acerca al 11% (exactamente 10.45%). Sin embargo en el segundo apartado he obtenido un mejor resultado evaluando con *f* (0.004%) que con *F* (0.04%).

	Ejercicio 1	Ejercicio 2
1 x	1.53	0.925
2 n	3	4
3 f(x)_matlab	-7.6079	-24.249
4 f(x,n)	-6.79	-24.25
5 F(x,n)	-7.07	-24.26
6 ep_f	10.75	0.0044321
7 ep_F	7.0699	0.045671

Considero que sería precipitado escoger una de las dos expresiones como clara vencedora sin realizar un número mayor de pruebas. Dependiendo el punto de x en el que evaluemos podemos obtener resultados más o menos cercanos al exacto, pero si tuviese que escoger basándome únicamente en estos datos, escogería la segunda expresión *F*, por realizar un número menor de redondeos y por que la superioridad de esta en el primer apartado es mayor que la inferioridad en el segundo, es decir: en el primer apartado *F* supera a *f* por un 3%, mientras que en el segundo, solo es peor en un 0.04%, por lo tanto, podemos considerar que hemos topado con un buen punto para f. Pero como he dicho, remarcó que

sería precipitado descartar *f* con tan pocos datos, aunque todo apunta a que así debería ser.

3. Resolución de sistemas de ecuaciones lineales

Siguin $A(N) = (a_{ij})_{N\times N}$ i $B(N) = (b_{i1})_{N\times 1}$ la matriu i el vector d'ordre N definits per

$$a_{ij} = \begin{cases} -1 & |i-j| \le 2\,, \\ 5 & i = j\,, \\ 0 & |i-j| > 2\,, \end{cases} \qquad \text{i} \qquad b_{i1} = \begin{cases} 3 & i = 1, N\,, \\ 2 & i = 2, N-1\,, \\ 1 & i \ne 1, 2, N-1, N\,, \end{cases}$$

$$\text{per a } i = 1, \ldots, N\,, \quad j = 1, \ldots, N\,.$$

Per a tots els ordres **N** tals que $6 \le N \le 30$ es demana:

1. Calculeu el determinant i el nombre de condició de les matrius A. Comproveu la simetria d'aquestes matrius. Comproveu que $X = (1, 1, \ldots, 1)$ és solució exacte per a qualsevol N (sense fer ús de MATLAB®)

Para resolver este primer apartado he utilizado las funciones predefinidas de MATLAB para calcular el determinante, el número de condición, y comprobar si es el sistema es simétrico o no.

N	Determinante	NumeroCondicion	Simetrica
6	9.3130e+03	3.7413	1
7	4.0920e+04	4.1779	1
8	1.7971e+05	4.5483	1
9	7.8910e+05	4.8915	1
10	3.4645e+06	5.1419	1
11	1.5211e+07	5.4048	1
12	6.6779e+07	5.5974	1
13	2.9318e+08	5.7765	1
14	1.2871e+09	5.9272	1
15	5.6509e+09	6.0506	1
16	2.4809e+10	6.1701	1
17	1.0892e+11	6.2604	1
18	4.7817e+11	6.3527	1
19	2.0993e+12	6.4260	1
20	9.2165e+12	6.4929	1
21	4.0463e+13	6.5537	1
22	1.7764e+14	6.6028	1
23	7.7989e+14	6.6537	1
24	3.4239e+15	6.6933	1
25	1.5032e+16	6.7332	1
26	6.5994e+16	6.7673	1
27	2.8973e+17	6.7976	1
28	1.2720e+18	6.8272	1
29	5.5844e+18	6.8507	1
30	2.4517e+19	6.8763	1

Como podemos ver el en la tabla de resultados, para toda N entre 6 y 30, la matriz del sistema es simétrica. Conforme N aumenta (tamano del sistema) tanto el determinante como el número de condición aumenta.

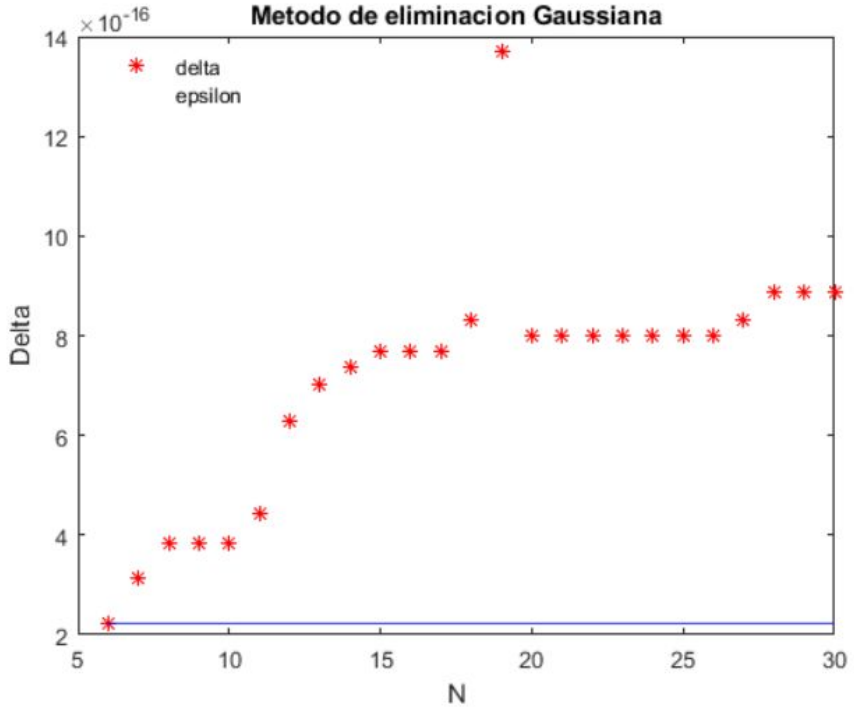
Comprobar que X es solución exacta del sistema es muy simple. Este sistema tiene 3 tipos de filas dependiendo de la cantidad de coeficientes igual a -1 que tengan. La primera y última fila cuentan con dos -1, un 5 y 0s (caso A), la segunda y penúltima fila cuentan con tres -1, un 5 y 0s (caso B) y el resto filas tienen todas cuatro -1, un 5 y un 0 (caso C). Si la solución es X = (1, 1, …, 1), entonces el valor B debe ser igual a la suma de los coeficientes de cada fila. Por lo tanto, para el caso A el término independiente debe ser

igual a 3, para el caso B debe ser igual a 2 y para el caso C igual a 1; y estos valores, son exactamente los de *b* (por definición del enunciado).

2. Fent ús de MATLAB® i d'un mètode d'eliminació gaussiana, determineu la solució x del sistema $Ax = b$. Expliqueu els avantages i inconvenients del mètode per aquest cas concret, expliqueu les desviacions de la solució que s'obtenen.

Como método de eliminación gaussiana he escogido el método de descomposición LU en triangulares, el método utilizado por MATLAB internamente para resolver sistemas. Para ello he realizo una factorización en dos triangulares L y U (incluyendo también P, con información de la diagonal), resuelvo el sistema Ly=Pb y finalmente se resuelve el sistema Ux=y, donde x es la solución final.

Dicha solución final cuenta con una serie de desviaciones. He calculado la norma vector de diferencias entre la solución obtenida y la solución real. A continuación muestro un gráfico con los valores para cada N. (La línea inferior azul representa el epsilon de la máquina.)



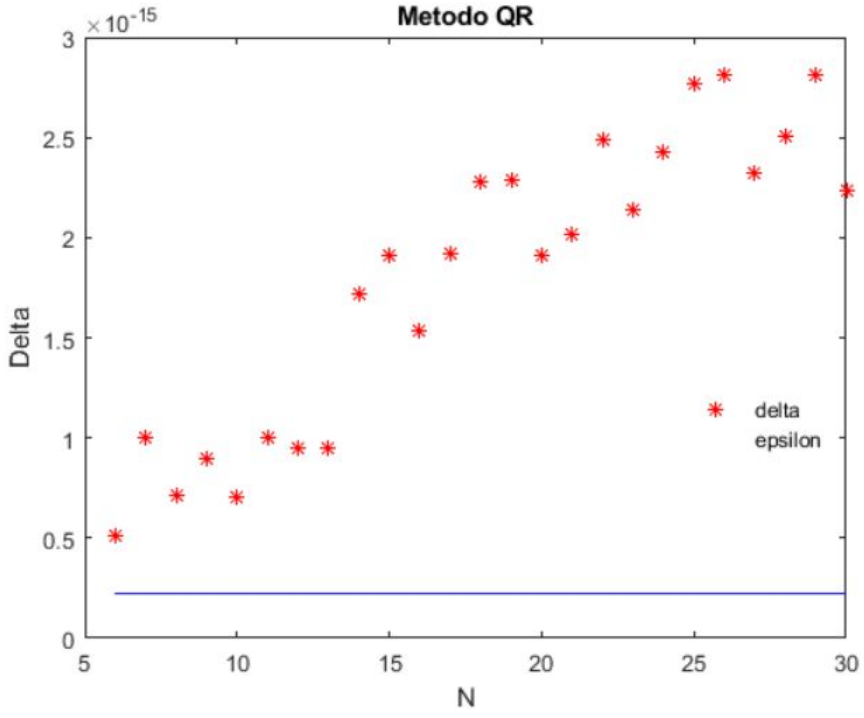
Podemos observar como conforme aumenta el tamaño del sistema, aumenta la diferencia con la solución real. Esto se debe a que como hemos visto en el primer apartado, conforme aumenta el tamaño, aumenta su número de condición, y por lo tanto es más sensible a pequeñas variaciones en los coeficientes provocadas por el redondeo y la aritmética de coma flotante durante las operaciones llevadas a cabo.

Dado que el condicionamiento del sistema es relativamente bajo, y no tratamos con muchas ecuaciones, podemos considerar este método como aceptable, ya que es bastante preciso y sobretodo rápido.

3. Fent ús de MATLAB® i del mètode QR de resolució de sistemes d'equacions lineals, determineu la solució x del sistema $Ax = b$. Expliqueu els avantages i inconvenients del mètode per aquest cas concret, expliqueu les desviacions de la solució que s'obtenen.

Para llevar a cabo el método QR he utilizado la función predefinida en MATLAB para calcular la factorización QR para posteriormente aplicar sustitución hacia atrás.

Del mismo modo que en el anterior apartado, he calculado la norma del vector de distancias entre la solución obtenida y la solución real para graficarlo respecto a N:

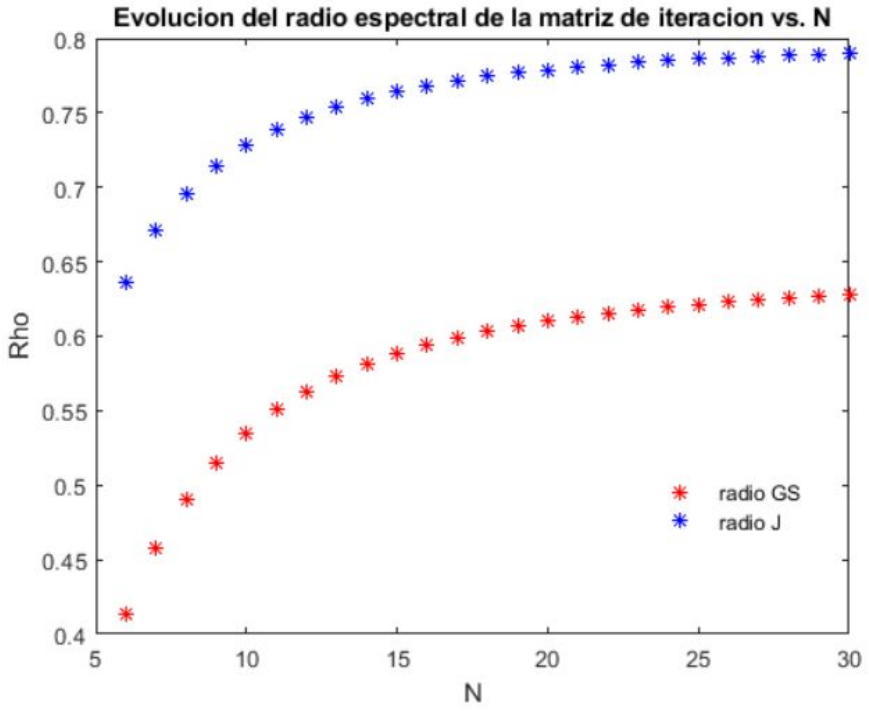


Podemos observar que como en el apartado anterior, conforme aumenta el tamaño del sistema, con todo lo que esto conlleva, obtenemos soluciones con un error mayor. Remarco también que en general el error es superior al del método de eliminación gaussiana, además de ser más costoso computacionalmente. Visto el método anterior, soy partidario de, en este caso concreto, ponerlo en segundo plano.

4. Estudieu la convergència dels mètodes de Jacobi i Gauss-Seidel per a la resolució del sistema d'equacions lineals. Prèviament feu un gràfic d'evolució del radi espectral de la matriu d'iteració de cadascun dels mètodes estudiats en funció de N. Expliqueu els avantages i inconvenients del mètodes per aquest cas concret.

Para implementar estos métodos iterativos, he calculado las matrices de iteración de cada método según su definición y con las funciones estándar de MATLAB para invertir, obtener la diagonal, etc.

Para todos los valores de N he calculado la convergencia a priori, que no es otra cosa que comprobar que el radio espectral de las matrices de iteración de cada método esté por debajo de 1. El radio espectral lo he calculado según su fórmula: el máximo del valor absoluto de los eigenvalues. A continuación el gráfico de la evolución del radio espectral respecto a N:

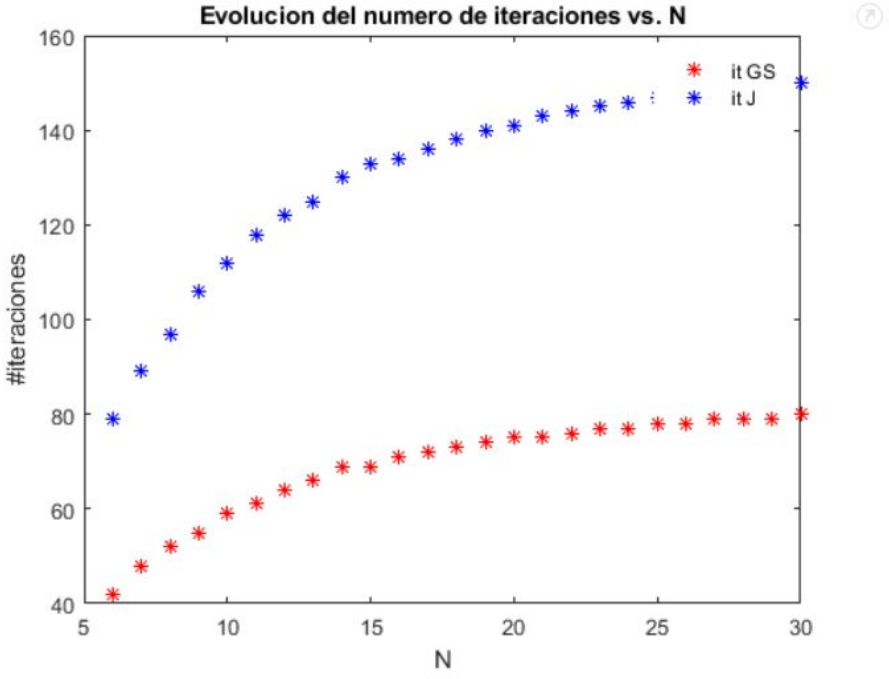


Como podemos perfectamente corroborar, el radio espectral está en todo caso por debajo de uno, así que a priori podemos asegurar que con esta matriz, nuestra solución va a converger.

Como ventaja tenemos que para este sistema los métodos son convergentes, pero como desventaja, no sabemos cuántas iteraciones nos va a llevar, y a pesar de saber que llegaremos a una solución aproximada, no sabemos cuanto nos demorara.

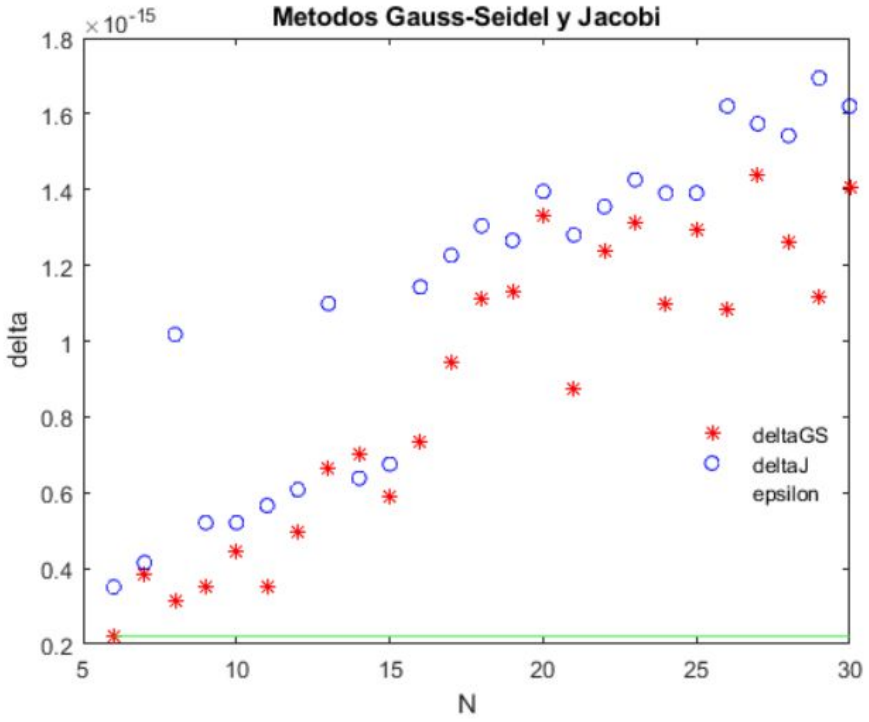
5. Compareu la solució X del sistema $Ax = b$ pels diferents mètodes experimentatss en aquest exercici. Quantes iteracions calen en cada cas? Expliqueu els avantages i inconvenients dels mètodes per aquest cas concret, expliqueu les desviacions de la solució que s'obtenen.

Contabilizar las iteraciones es una tarea muy sencilla, mediante un contador dentro del bucle es suficiente. Como condición de parada de este bucle he usado la convergencia a posteriori. He forzado que la diferencia entre la solución actual y la siguiente sea el epsilon de la máquina. Si aumentamos esta tolerancia, los métodos realizarán un número menor de iteraciones, pero yo he preferido apurar al máximo en ambos caso. La evolución de las iteraciones respecto a N se muestran en el siguiente gráfico:



Como podemos ver, el número de iteraciones aumenta conforme aumenta N, pero no de forma demasiado rápida. En este caso el método de Gauss-Seidel realiza un número menor de iteraciones para todas las N testadas, así que podemos considerar este como superior para este sistema.

A continuación el gráfico de distancias entre la solución obtenida y la solución real para los diferentes valores de N y los dos métodos iterativos:



En cuanto a la precisión de las soluciones podemos ver como de nuevo el error tiende a aumentar con el tamaño del sistema para ambos métodos, y en general, Gauss-Seidel sale de nuevo mejor parado, aunque en este caso, no por tanta diferencia. Los resultados son sutilmente más cercanos al exacto para la mayoría de tamaños, pero no es algo muy significativo.

Bibliografía

Bibliografía consultada para la realización del apartado 1 del documento:

- Borgwardt, M. B. (s.f.). **La Guía del Punto Flotante** - Números de punto flotante. Recuperado 13 marzo, 2020, de Números de punto flotante

- Colaboradores de Wikipedia. (2020a, 5 marzo). **Estándar del IEEE para aritmética en coma flotante**. Recuperado 13 marzo, 2020, de IEEE coma flotante

- Colaboradores de Wikipedia. (2020b, 10 marzo). **Formato en coma flotante de simple precisión**. Recuperado 13 marzo, 2020, de Formato en coma flotante de simple precisión

- Pérez Martínez, J. (2013, 8 julio). **Estándar IEEE 754 para la representación en coma flotante.** Recuperado 13 marzo, 2020, de Estándar IEEE 754 para la representación en coma flotante | Blog de José Pérez Martínez

- Wikipedia contributors. (2020, 6 marzo). **Denormal Numbers**. Recuperado 13 marzo, 2020, de Denormal number

- Cleve's Corner: **Cleve Moler on Mathematics and Computing** Floating Point Arithmetic Before IEEE 754

- Cleve's Corner: **Cleve Moler on Mathematics and Computing** Floating Point Numbers (I)

- Cleve's Corner: **Cleve Moler on Mathematics and Computing** Floating Point Numbers (II)

- Moler, C. "**Floating points: IEEE Standard unifies arithmetic model**," Cleve's Corner. The MathWorks,nc.,1996.

- Charles Severance, C. S. (1998, 20 febrero). **An Interview with the Old Man of Floating-Point.** Recuperado 13 marzo, 2020, de Link