Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Matemáticas

El refinamiento iterativo en la resolución de sistemas lineales cuando se combinan varias precisiones

Autor: Inés de Castro Heras
Tutora: María Paz Calvo Cabrero
2024-2025

Agradecimientos

Quisiera agradecer profundamente a todas las personas que me han acompañado no sólo en la realización de este Trabajo de Fin de Grado sino también durante estos últimos cinco años. Sin vosotros nunca habría llegado hasta aquí.

En primer lugar, agradezco a mi tutora, Mari Paz Calvo, por su constante apoyo, orientación y paciencia durante todo el proceso. Su dedicación y esfuerzo han sido fundamentales en este proyecto.

A mi familia, en especial a mi madre y a mi hermano, por su amor incondicional y por motivarme a seguir adelante incluso en los momentos más difíciles. Siempre les estaré agradecida por ser el pilar fundamental de mi vida.

A todos mis amigos, en particular los que me ha dado esta carrera tan bonita, por compartir este camino, por su compañía, consejos y por los buenos momentos que me han ayudado a sobrellevar el esfuerzo diario. Me habéis hecho muy feliz estos años y espero que siga siendo así toda la vida.

Por último, a Nico por haber sido y seguir siendo mi mayor apoyo, mi lugar seguro y mi casa cuando estaba muy lejos de ella.

A todos, gracias de corazón.

Resumen

En esta memoria se abordará en profundidad el método de refinamiento iterativo, una estrategia orientada a mejorar la precisión de soluciones aproximadas de sistemas lineales. Se comenzará con un análisis detallado del efecto de los errores de redondeo cuando los residuos se calculan con una precisión doble respecto a la utilizada en la resolución de los sistemas triangulares implicados. Además, se explorarán otros resultados que respaldan el empleo de esta técnica utilizando hasta tres niveles distintos de precisión, así como su aplicación en sistemas lineales mal acondicionados. Finalmente, se estudiarán diferentes familias de matrices que se utilizarán para ilustrar el comportamiento de algunas de las metodologías implementadas, utilizando la simulación de aritmética en coma flotante de múltiples precisiones que ofrece `MATLAB`, y se mostrarán los resultados numéricos obtenidos.

Palabras clave: refinamiento iterativo, sistemas lineales, aritmética de punto flotante.

Abstract

This report will take an in-depth look at the iterative refinement method, a strategy aimed at improving the accuracy of approximate solutions to linear systems. It will begin with a detailed analysis of the effect of rounding errors when the residuals are calculated with twice the precision used in solving the triangular systems involved. In addition, other results supporting the use of this technique will be explored, using up to three different levels of precision, as well as its application in ill-conditioned linear systems. Finally, different families of matrices will be studied to illustrate the behaviour of some of the implemented methodologies, using the multi-precision floating-point arithmetic simulation offered by `MATLAB`, and the numerical results obtained will be shown.

Keywords: iterative refinement, linear systems, floating-point arithmetic.

Introducción

En el ámbito de la resolución numérica de sistemas lineales, el refinamiento iterativo ha sido durante décadas una herramienta fundamental para mejorar la precisión de las soluciones obtenidas mediante factorizaciones directas. A pesar de su simplicidad conceptual, esta técnica encierra una sorprendente profundidad teórica y ha evolucionado en paralelo al desarrollo de la computación moderna. El objetivo de este trabajo es estudiar de manera rigurosa y práctica distintas variantes del refinamiento iterativo, utilizando diferentes configuraciones de precisiones, comparando su comportamiento, tanto teórico como práctico.

El refinamiento iterativo clásico tiene sus raíces en el año 1948 [2], cuando James H. Wilkinson implementó una versión primitiva del método utilizando factorizaciones LU y cálculos de residuos en una precisión superior a la precisión de trabajo. Estos cálculos se realizaban en máquinas mecánicas de escritorio y con tarjetas perforadas Hollerith, lo que subraya la notable eficiencia y utilidad del método incluso con los recursos computacionales limitados de la época. Durante las décadas siguientes, el método se mantuvo en uso debido a que muchas arquitecturas computacionales permitían acumular productos internos con el doble de precisión de forma eficiente.

Durante los años 70 [2], comenzó a considerarse el refinamiento iterativo en precisión fija, en el que todos los cálculos se realizan en la misma precisión. Este enfoque fue analizado por autores como Jankowski y Woźniakowski para métodos generales de resolución, y por Skeel específicamente para la factorización LU. Nicholas Higham, figura clave en el análisis moderno de errores numéricos, extendió estos estudios a partir de los años 90, incluyendo variantes con residuos en precisión extra. Esta evolución metodológica está reflejada en bibliotecas de *software* ampliamente utilizadas como LAPACK, que implementan versiones de refinamiento en precisión fija.

En los años 2000 [2], el refinamiento iterativo volvió a despertar un gran interés debido a que en muchos procesadores modernos, la aritmética en precisión simple es significativamente más rápida que en precisión doble. Esto motivó el desarrollo de esquemas mixtos en los que las partes más costosas del algoritmo se realizan en menor precisión, mientras que los residuos se calculan en precisiones más altas. Esta estrategia permite una mejora en el rendimiento computacional

sin comprometer la precisión final, siempre que la matriz original esté suficientemente bien acondicionada. Trabajos como los de Langou et al. y Dongarra y sus colaboradores contribuyeron a formalizar y difundir esta aproximación.

Más recientemente, en 2017 [2], Carson y Higham desarrollaron un análisis que permite aplicar el refinamiento iterativo incluso en casos donde la matriz tiene un número de condición tan alto como el inverso de la unidad de redondeo. Este resultado amplía de forma notable la aplicabilidad del método, al demostrar que, bajo ciertas condiciones, es posible obtener soluciones numéricamente estables incluso en escenarios adversos.

Este trabajo se estructura de la siguiente manera: en el Capítulo 1 se presentan los preliminares teóricos, incluyendo el análisis de errores y la aritmética de punto flotante; en el Capítulo 2 se aborda el refinamiento iterativo clásico, mientras que el Capítulo 3 está dedicado al caso de tres precisiones. En el Capítulo 4 se describen varias familias de matrices de prueba, diseñadas para evaluar el comportamiento del algoritmo en contextos diversos. Finalmente, en el Capítulo 5 se exponen los experimentos realizados, analizando el impacto del número de condición y de las distintas configuraciones de precisión en la convergencia del método.

Con este estudio se pretende aportar una visión completa, tanto teórica como práctica, sobre el refinamiento iterativo, destacando sus variantes más actuales.

Índice general

Índice general **7**

1. Preliminares **9**
1.1. Errores regresivos y progresivos 9
1.2. Análisis por componentes y análisis en norma 11
1.3. Número de condición de una matriz 13
1.4. Aritmética de punto flotante 14
1.4.1. Sistema numérico de punto flotante 15
1.4.2. Modelo estandarizado de aritmética 16
1.4.3. Tipos de precisiones que se utilizarán 16
1.4.4. Precisiones en `MATLAB` 18
1.5. Matrices y aritmética de punto flotante 19
1.5.1. Producto escalar 19
1.5.2. Multiplicación de un vector por una matriz 22
1.5.3. Factorización LU de una matriz 23

2. Refinamiento iterativo clásico **25**
2.1. Análisis del error progresivo 25
2.2. Análisis del error regresivo 30

3. Refinamiento iterativo: tres precisiones **39**
3.1. Análisis progresivo 40
3.2. Análisis regresivo 41
3.2.1. Análisis regresivo en norma 42
3.2.2. Análisis regresivo componente a componente 43

4. Familias de matrices de prueba **47**
4.1. Matrices ortogonales 47
4.2. Matrices de Hilbert 49
4.3. Matrices con número de condición predefinido 50
4.3.1. Construcción de la familia de matrices 50

4.3.2. El efecto de los errores de redondeo 52
4.3.3. Matrices de la familia con un número de condición predefinido 55

5. Experimentos numéricos **59**
5.1. Importancia del número de condición en la convergencia del refinamiento iterativo 59
5.2. Comparación utilizando distintas precisiones 61
5.2.1. Refinamiento tradicional con residuos en precisión extra . 61
5.2.2. Refinamiento con precisión fija 63
5.2.3. Refinamiento de precisión mixta con soluciones en menor precisión 65
5.2.4. Comparativa 69

Bibliografía **71**

A. Códigos **73**

Capítulo 1

Preliminares

En este capítulo. se presentan algunos conceptos y resultados que serán necesarios en los desarrollos posteriores. Se introducen las nociones de error regresivo y progresivo, distintos enfoques para el análisis del error, el número de condición de una matriz y las bases de la aritmética de punto flotante, así como algunos resultados básicos sobre las operaciones matriciales más frecuentes. Estos preliminares servirán de base para la comprensión rigurosa de los métodos y del análisis que se tratarán en los siguientes capítulos.

1.1. Errores regresivos y progresivos

En primer lugar, empezaremos explicando los conceptos de error progresivo y error regresivo [6].
Sea $\hat{y}$ una aproximación a $y = f(x)$ calculada con aritmética finita y una precisión u, donde f es una función escalar real de variable escalar real. Tenemos varias formas de medir la calidad de $\hat{y}$.

En la mayoría de los casos, compararemos $\hat{y}$ con y, buscando que el error relativo sea del tamaño de la unidad de redondeo u. Pero como esto no siempre es posible, optaremos por hacernos otra pregunta: ¿Para qué valor de $\hat{x} = x+\Delta x$ tenemos que $\hat{y} = f(x + \Delta x)$? En general, habrá muchos Δx que verifiquen esto y buscaremos que el tamaño de Δx sea pequeño. Para precisar estas ideas, introducimos las siguientes definiciones.

Definición 1.1.1. Definimos el error regresivo (en inglés, *backward error*) como el menor $|\Delta x|$ que verifica que $\hat{y} = f(x + \Delta x)$. En ocasiones, podemos encontrarnos también este valor $|\Delta x|$ dividido por $|x|$, es decir, $|\Delta x|/|x|$ (error regresivo relativo).

Llamamos análisis regresivo del error al proceso de acotar el error regresivo de una solución calculada. En el error regresivo interpretamos la solución

aproximada como la solución exacta para un dato perturbado.

Definición 1.1.2. Los errores progresivos (en inglés, *forward errors*) hacen referencia a la diferencia $y - \hat{y}$ y pueden ser errores absolutos

$$E_{\text{abs}}(y) = |y - \hat{y}|$$

y relativos

$$E_{\text{rel}}(y) = \frac{|y - \hat{y}|}{|y|}.$$

Para ilustrar las Definiciones 1.1.1 y 1.1.2 observemos la Figura 1.1.

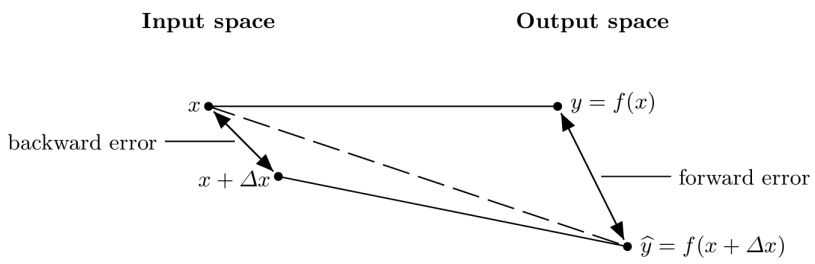

Figura 1.1: Representación gráfica del error regresivo y progresivo (tomada de [6]).

Normalmente, los datos contienen cierta incertidumbre debido a errores en mediciones, cálculos previos, o errores cometidos al almacenar los números en el ordenador. Luego, si el error regresivo es menor o igual que estas incertidumbres, entonces la solución calculada a priori podría ser la solución que estamos buscando.

Definición 1.1.3. Se dice que un método para calcular $y = f(x)$ es estable regresivamente si para cualquier x calcula la aproximación $\hat{y}$ con un error regresivo pequeño, es decir, existe Δx pequeño tal que $\hat{y} = f(x + \Delta x)$. El significado de "pequeño" dependerá de cada contexto.

Los conceptos definidos en esta sección serán necesarios para comprender correctamente conceptos de los que se hablará posteriormente. En esta sección se ha considerado únicamente el caso escalar pero posteriormente trabajaremos también con el caso vectorial.

1.2. Análisis componente a componente y análisis en norma

A lo largo de esta memoria haremos principalmente dos tipos diferentes de análisis de error que explicaremos durante esta sección [6].

Definición 1.2.1. Sea $\mathbf{b} \in \mathbb{R}^m$. Denotamos por $|\mathbf{b}|$ al vector cuyas componentes son el valor absoluto de las componentes de $\mathbf{b}$. Es decir, si $\mathbf{b} = (b_i)_{1\leq i\leq m}$ tenemos que $|\mathbf{b}| = (|b_i|)_{1\leq i\leq m}$.
Sea $A \in \mathbb{R}^{m\times n}$. Denotamos por $|A|$ a la matriz cuyos coeficientes son el valor absoluto de los coeficientes de A. Es decir, si $A = (a_{i,j})_{1\leq i\leq m,1\leq j\leq n}$ tenemos que $|A| = (|a_{i,j}|)_{1\leq i\leq m,1\leq j\leq n}$.

Con estas definiciones, es fácil probar el siguiente resultado.

Teorema 1.2.1. Sean $A \in \mathbb{R}^{m\times n}$ y $B \in \mathbb{R}^{n\times p}$. Se verifica la siguiente desigualdad

$$|AB| \leq |A||B|.$$

Demostración:
Llamaremos $C \in \mathbb{R}^{m\times p}$ al resultado de multiplicar A y B y queremos ver que

$$|C| \leq |A||B|.$$

Escribimos $A = (a_{i,j})_{1\leq i\leq m,1\leq j\leq n}$ y $B = (b_{i,j})_{1\leq i\leq n,1\leq j\leq p}$. Si llamamos $D \in \mathbb{R}^{m\times p}$ a la matriz $D = |A||B| = (d_{i,j})_{1\leq i\leq m,\,1\leq j\leq p}$, tenemos que

$$d_{i,j} = \sum_{k=1}^{n} |a_{i,k}|\,|b_{k,j}|, \quad 1 \leq i \leq m,\ 1 \leq j \leq p. \tag{1.1}$$

Por otro lado, si escribimos $C = (c_{i,j})_{1\leq i\leq m,\,1\leq j\leq p}$ tendremos que

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k}b_{k,j}, \quad 1 \leq i \leq m,\ 1 \leq j \leq p.$$

Tomando valores absolutos y aplicando la desigualdad triangular,

$$\begin{aligned} |c_{i,j}| &= \left|\sum_{k=1}^{n} a_{i,k}b_{k,j}\right| \\ &\leq \sum_{k=1}^{n} |a_{i,k}||b_{k,j}| \\ &= d_{i,j}. \end{aligned}$$

Como todos los elementos de la matriz $|C|$ son menores o iguales que los elementos de la matriz D, concluimos entonces que $|C| \leq D$, o lo que es lo mismo,

$$|AB| \leq |A||B|,$$

y queda terminada la demostración. □

Utilizando esta notación hablaremos de análisis de error componente a componente como el análisis de error en el que se quieren demostrar cotas en términos de $|A|$ y $|\mathbf{b}|$.
Otra forma de analizar el error es mediante la utilización de una norma vectorial y de la norma matricial derivada de ella. A este análisis le llamaremos análisis del error en norma. La norma que más utilizaremos en este texto será la norma infinito, cuya definición recordamos a continuación.

Definición 1.2.2. Sea $\mathbf{b} \in \mathbb{R}^m$. La norma infinito de $\mathbf{b}$ viene dada por

$$\|\mathbf{b}\|_\infty = \max_{1 \leq i \leq m} |b_i|. \tag{1.2}$$

Sea $A \in \mathbb{R}^{m \times n}$. Su norma infinito viene dada por

$$\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{m} |a_{ij}|, \tag{1.3}$$

es decir, es el máximo sobre las filas de las sumas de los valores absolutos de los elementos de cada fila.

Veremos que el análisis componente a componente es más fino que el análisis en norma.

Definición 1.2.3. Sea $A \in \mathbb{R}^{n \times n}$ y $\mathbf{b} \in \mathbb{R}^n$. El error regresivo componente a componente para una solución aproximada $\mathbf{y}$ de un sistema lineal $A\mathbf{x} = \mathbf{b}$ se define como

$$\omega_{E,\mathbf{f}}(\mathbf{y}) = \min\{\epsilon : (A + \Delta A)\mathbf{y} = \mathbf{b} + \Delta\mathbf{b}, \quad |\Delta A| \leq \epsilon E, \quad |\Delta\mathbf{b}| \leq \epsilon\mathbf{f}\},$$

donde E y $\mathbf{f}$ son matrices de tolerancias no negativas.

Existe una fórmula computacionalmente simple para $\omega_{E,\mathbf{f}}(\mathbf{y})$, como se muestra en el siguiente resultado cuya demostración puede encontrarse en [6]. Adoptamos la convención de que $\xi/0$ se interpreta como cero si $\xi = 0$ e infinito en caso contrario.

Teorema 1.2.2. El error regresivo componente a componente definido en la Definición 1.2.3 está dado por

$$\omega_{E,\mathbf{f}}(\mathbf{y}) = \max_{1\leq i\leq n} \frac{|r_i|}{(E|\mathbf{y}| + \mathbf{f})_i},$$

donde $\mathbf{r} = \mathbf{b} - A\mathbf{y}$ es el correspondiente residuo.
En el caso particular de $E = |A|$ y $\mathbf{f} = |\mathbf{b}|$ se tiene

$$\omega_{|A|,|\mathbf{b}|}(\mathbf{y}) = \max_{1\leq i\leq n} \frac{|r_i|}{(|A||\mathbf{y}| + |\mathbf{b}|)_i},$$

y $\omega_{|A|,|\mathbf{b}|}(\mathbf{y})$ controla las variaciones relativas $|\Delta A|/|A|$ y $|\Delta \mathbf{b}|/|\mathbf{b}|$ en los datos.

Definición 1.2.4. Sea $A \in \mathbb{R}^{n\times n}$ y $\mathbf{b} \in \mathbb{R}^n$. El error regresivo en norma para una solución aproximada $\mathbf{y}$ de un sistema lineal $A\mathbf{x} = \mathbf{b}$ se define como

$$\eta(\mathbf{y}) := \min \{\epsilon : (A + \Delta A)\mathbf{y} = \mathbf{b} + \Delta \mathbf{b}, \quad \|\Delta A\| \leq \epsilon\|A\|, \quad \|\Delta \mathbf{b}\| \leq \epsilon\|\mathbf{b}\|\}.$$

En [6] se puede ver la demostración del siguiente resultado.

Teorema 1.2.3. El error regresivo en norma definido en la Definición 1.2.4 está dado por

$$\eta(\mathbf{y}) = \frac{\|\mathbf{r}\|}{\|A\|\|\mathbf{y}\| + \|\mathbf{b}\|}.$$

1.3. Número de condición de una matriz

Definición 1.3.1. Sea $A \in \mathbb{R}^{n\times n}$. Para cada $\mathbf{x} \in \mathbb{R}^n$, con $\mathbf{x} \neq \mathbf{0}$, se define en [6]

$$\text{cond}(A, \mathbf{x}) = \frac{\||A^{-1}||A||\mathbf{x}|\|_\infty}{\|\mathbf{x}\|_\infty}. \tag{1.4}$$

El número de condición componente a componente de una matriz se define como

$$\text{cond}(A) = \text{cond}(A, \mathbf{e}), \tag{1.5}$$

donde $\mathbf{e} = [1, 1, \ldots, 1]^T \in \mathbb{R}^n$.

Se verifica que $\text{cond}(A) = \||A^{-1}||A|\|_\infty$ como se detalla a continuación. En primer lugar, se tiene que $\|\mathbf{e}\|_\infty = \text{máx}_i\,|e_i| = 1$ luego

$$\text{cond}(A) = \big\|\,|A^{-1}|\,|A|\,\mathbf{e}\big\|_\infty.$$

Ahora bien, el i-ésimo componente del vector $|A^{-1}||A|\mathbf{e}$ es

$$\big(|A^{-1}||A|\mathbf{e}\big)_i = \sum_{j=1}^{n}\big(|A^{-1}||A|\big)_{ij}\,e_j = \sum_{j=1}^{n}\big(|A^{-1}||A|\big)_{ij},$$

pues $e_j = 1$. Por tanto,

$$\big\|\,|A^{-1}|\,|A|\,\mathbf{e}\big\|_\infty = \max_{1\le i\le n}\sum_{j=1}^{n}\big(|A^{-1}||A|\big)_{ij} = \big\|\,|A^{-1}|\,|A|\big\|_\infty,$$

donde la última igualdad es la definición de la norma-∞ de una matriz. $\|\mathbf{e}\|_\infty = \text{máx}_i\,|e_i| = 1$, se tiene

$$\text{cond}(A) = \big\|\,|A^{-1}|\,|A|\,\mathbf{e}\big\|_\infty.$$

Por otro lado, el número de condición en norma infinito de la matriz A viene dado por

$$\kappa_\infty(A) = \|A^{-1}\|_\infty\|A\|_\infty. \tag{1.6}$$

Se muestra a continuación la relación entre ambos números de condición.

Lema 1.3.1. Sea $A \in \mathbb{R}^{n\times n}$. Se verifica que

$$\text{cond}(A) \le \kappa_\infty(A). \tag{1.7}$$

Demostración:

$$\text{cond}(A) = \||A^{-1}||A|\|_\infty \le \||A^{-1}|\|_\infty\||A|\|_\infty = \|A^{-1}\|_\infty\|A\|_\infty = \kappa_\infty(A).$$

□

1.4. Aritmética de punto flotante

La aritmética de punto flotante constituye la base de la representación numérica en la mayoría de los sistemas informáticos modernos. A diferencia de la aritmética entera, permite trabajar con una amplia gama de valores, desde cantidades extremadamente pequeñas hasta números muy grandes, aunque siempre con una precisión limitada por el formato utilizado. Esta sección introduce los fundamentos de los sistemas númericos de punto flotante, describe el modelo estandarizado de aritmética, explora las principales precisiones empleadas actualmente y finalmente trata una forma de trabajar con diferentes precisiones en `MATLAB`.

1.4.1. Sistema numérico de punto flotante

Para comprender la aritmética de punto flotante [11] debemos presentar el concepto de sistema numérico de punto flotante [7].

Definición 1.4.1. Un sistema numérico de punto flotante es un conjunto finito $F = F(\beta, t, e_{\text{mín}}, e_{\text{máx}})$ de $\mathbb{R}$ cuyos elementos son de la forma

$$x = \pm m \times \beta^{e-t+1}. \tag{1.8}$$

Denominaremos base a β, que es 2 en prácticamente todos los ordenadores actuales. El entero t es la precisión y el entero e es el exponente, que se encuentra en el rango $e_{\text{mín}} \leq e \leq e_{\text{máx}}$, y el estándar IEEE requiere que $e_{\text{mín}} = 1 - e_{\text{máx}}$. La mantisa m es un entero que satisface $0 \leq m \leq \beta^t - 1$. Para asegurar una representación única de cada $x \in F$ diferente de cero, se asume que $m \geq \beta^{t-1}$ si $x \neq 0$, de manera que el sistema esté *normalizado*.

Los números positivos más grandes y más pequeños en el sistema son $x_{\text{máx}} = \beta^{e_{\text{máx}}}(\beta - \beta^{1-t})$ y $x_{\text{mín}} = \beta^{e_{\text{mín}}}$, respectivamente. Otras dos cantidades importantes son $u = \frac{1}{2}\beta^{1-t}$, la *unidad de redondeo*, y $\epsilon = \beta^{1-t}$, el *épsilon de máquina*, que es la distancia desde 1 hasta el siguiente número de punto flotante más grande.

Los números con $e = e_{\text{mín}}$ y $0 < m < \beta^{t-1}$ se denominan *números subnormales*, tienen el exponente mínimo pero menos de t dígitos de precisión y se utilizan para representar números tan próximos a 0 que no puedan ser representados en forma *normal*, es decir, con la forma de (1.8). El conjunto de estos números forma una cuadrícula equiespaciada entre 0 y el número normalizado más pequeño.

En la Tabla 1.1 podemos observar los números positivos más pequeños, en la segunda columna, y más grandes, en la tercera columna, para algunos de los formatos de punto flotante más utilizados. Profundizaremos más en cada uno de estos formatos en la Sección 1.4.3.

Precisión	$x_{\text{mín}}$	$x_{\text{máx}}$
bfloat16	$11{,}8 \times 10^{-39}$	339×10^{36}
fp16	$61{,}0 \times 10^{-6}$	$65{,}5 \times 10^{3}$
fp32	$11{,}8 \times 10^{-39}$	340×10^{36}
fp64	$22{,}2 \times 10^{-309}$	180×10^{306}

Tabla 1.1: Tabla de formatos de punto flotante indicando el número mínimo $x_{\text{mín}}$ y máximo $x_{\text{máx}}$ que pueden representar.

1.4.2. Modelo estandarizado de aritmética

Haciendo operaciones aritméticas básicas en un ordenador también se cometen errores luego para realizar un análisis de redondeo debemos hacer algunas suposiciones. Las operaciones más comunes y con las que trabajaremos en este texto son las que incluyen uno de los siguientes operadores $+, -, *$ o $/$. En el modelo estandarizado de aritmética [7] con el que vamos a trabajar supondremos que:

$$fl(x\ op\ y) = (x\ op\ y)(1+\delta), \qquad |\delta| \leq u, \quad op = +, -, *, /. \tag{1.9}$$

1.4.3. Tipos de precisiones que se utilizarán

En esta sección explicaremos el formato y el tamaño de algunas de las precisiones más utilizadas actualmente.

- En primer lugar, hablaremos sobre la aritmética de **media precisión** (*half precision artihmetic*) [10], a la que nos referiremos en los experimentos como 'h'. Fue publicada en el estándar IEEE 754 y también es conocida de forma más técnica como fp16. Este nombre se debe a que las palabras tienen 16 bits, el primer bit está dedicado al signo (será 1 si el número que queremos representar es negativo y 0 si es positivo), los 5 bits siguientes están dedicados al exponente y los últimos 10 a la mantisa. En este caso tendremos que la base β será igual a 2.

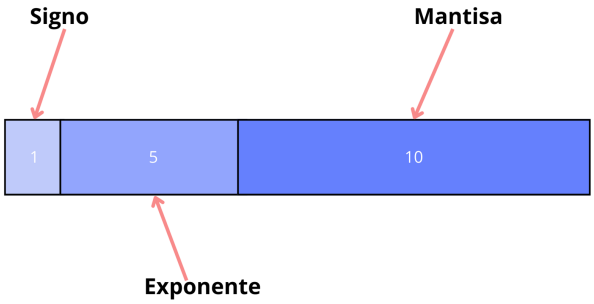

Figura 1.2: Estructura y organización de los bits en la precisión media.

- Otra de las precisiones más conocidas es la **precisión simple** (*single precision arithmetics*) [1], a la que nos referiremos en los experimentos como 's'. También fue publicada en el estándar IEEE 754 y se conoce de forma técnica como f32, puesto que utiliza 32 bits. Al igual que para la fp16, el primer bit es para indicar el signo, los 8 siguientes para el

exponente, y finalmente los últimos 23 bits se utilizarán para la mantisa. En este caso también tendremos que la base β será igual a 2.

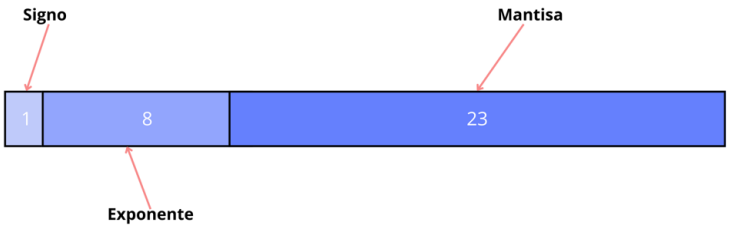

Figura 1.3: Estructura y organización de los bits en la precisión simple.

- La última precisión publicada en el estándar IEEE 754 de la que hablaremos será la **precisión doble** (*double precision arithmetics*) [1], a la que nos referiremos en los experimentos como 'd'. Esta precisión también es conocida como fp64. En este caso se necesitan 64 bits para guardar un valor numérico, el primer bit, al igual que en los dos casos anteriores, es para indicar el signo, los 11 siguientes para el exponente, y los 52 bits restantes se utilizarán para la mantisa. En este caso también tendremos que la base β será igual a 2.

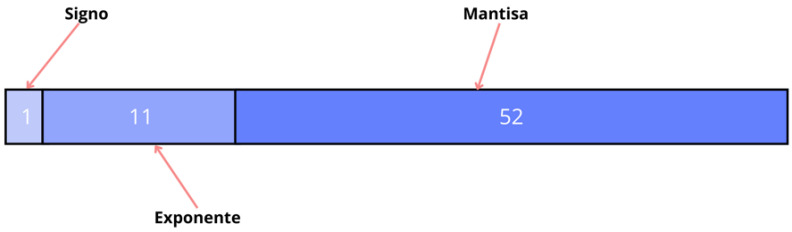

Figura 1.4: Estructura y organización de los bits en la precisión doble.

- Por último, hablaremos sobre la precisión **bfloat16** [5]. Esta es la única precisión de las que hemos comentado que no fue publicada en el estándar IEEE 754 sino que fue creada por el equipo de investigación de inteligencia artificial de Google Brain. Al igual que la fp16 necesita 16 bits para guardar valores numéricos pero estos se distribuyen de forma diferente. El primer bit guardará el signo, los 8 bits siguientes el exponente y finalmente los 7 bits del final la mantisa.

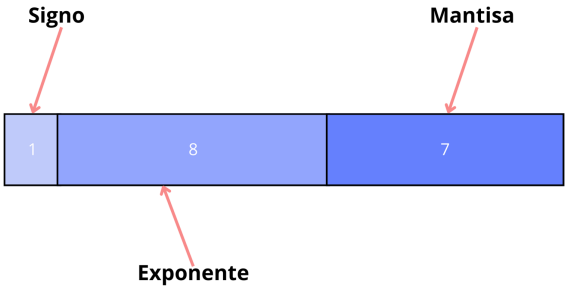

Figura 1.5: Estructura y organización de los bits en la precisión bfloat16.

Una vez explicadas las principales precisiones con las que trabajaremos podríamos preguntarnos el por qué de la existencia de dos precisiones que ocupen lo mismo, 16 bits, la fp16 y la bfloat16. El motivo es sencillo, la fp16 dedica más bits a la mantisa lo que nos permite tener más precisión a la hora de representar fracciones mientras que la bfloat16 dedica más dígitos al exponente lo que nos permite representar un rango mayor de valores. En conclusión, cada una aporta beneficios diferentes y debemos emplear en cada caso la que más nos convenga.
En la Tabla 1.2 encontramos de forma organizada la información sobre los bits dedicados a la mantisa y el exponente, en la segunda columna, y el valor de la unidad de redondeo, en la tercera columna, de los casos particulares de los formatos de punto flotante explicados durante esta sección para así tener una visión más completa de la "precisión" de cada uno de estos formatos.

Precisión	**(mantisa, exp)**	u
bfloat16	(7, 8)	$3{,}91 \times 10^{-3}$
fp16	(10, 5)	$4{,}88 \times 10^{-4}$
fp32	(23, 8)	$5{,}96 \times 10^{-8}$
fp64	(52, 11)	$1{,}11 \times 10^{-16}$

Tabla 1.2: Tabla de formatos de punto flotante indicando los bits dedicados a la mantisa, los bits dedicados al exponente y el valor de la unidad de redondeo u.

1.4.4. Precisiones en `MATLAB`

Para trabajar con las diferentes precisiones en `MATLAB` utilizaremos la función `chop`. Esta función es un simulador de precisiones [8] que nos permitirá simular

las precisiones presentadas anteriormente para poder trabajar con una o varias de ellas en nuestros experimentos. Esta función nos permitirá trabajar con la precisión que queramos, como podemos observar en la Tabla 1.3, y también elegir el tipo de redondeo que queremos que se realice en las operaciones, entre otros ajustes posibles, como podemos ver en la Tabla 1.4.

Alias Matlab	**Descripción**
'b', 'bfloat16'	bfloat16
'h', 'half', 'fp16'	IEEE half precision (por defecto)
's', 'single', 'fp32'	IEEE single precision
'd', 'double', 'fp64'	IEEE double precision
'c', 'custom'	formato personalizado

Tabla 1.3: Formatos de precisión para la función `chop`

Parámetro `Matlab`	**Descripción del redondeo**
1	Redondea al número entero más próximo utilizando el redondeo par en caso de empate
2	Redondeo hacia arriba
3	Redondeo hacia abajo
4	Redondeo hacia cero
5	Redondeo estocástico con probabilidad proporcional a 1 menos la distancia a esos números de coma flotante
6	Redondeo estocástico con la misma probabilidad

Tabla 1.4: Tipos de redondeo para la función `chop`

1.5. Operaciones matriciales en aritmética de punto flotante

Revisamos en esta sección algunos resultados sobre los errores regresivos en las operaciones más sencillas (y también más utilizadas) entre vectores y matrices.

1.5.1. Producto escalar

Consideramos el producto escalar de dos vectores [6] $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,

$$s_n = \mathbf{x}^T\mathbf{y} = x_1y_1 + \cdots + x_ny_n, \tag{1.10}$$

y denotamos por s_i a la suma parcial i-ésima,

$$s_i = x_1y_1 + \cdots + x_iy_i. \tag{1.11}$$

Usando el modelo estandarizado explicado en la Sección 1.4.2 tenemos que

$$\begin{aligned}\hat{s}_1 &= fl(x_1y_1) = x_1y_1(1+\delta_1),\\ \hat{s}_2 &= fl(\hat{s}_1 + x_2y_2) = (\hat{s}_1 + x_2y_2(1+\delta_2))(1+\delta_3)\\ &= (x_1y_1(1+\delta_1) + x_2y_2(1+\delta_2))(1+\delta_3)\\ &= x_1y_1(1+\delta_1)(1+\delta_3) + x_2y_2(1+\delta_2)(1+\delta_3),\end{aligned}$$

con $|\delta_i| \leq u, i = 1, \ldots, 3$.

En este caso no necesitamos distinguir entre los diferentes δ_i así que sustituiremos las expresiones $\delta_i + 1$ por $\delta \pm 1$ con $|\delta| \leq u$. Continuando con el proceso llegamos a que

$$\hat{s}_n = x_1y_1(1\pm\delta)^n + x_2y_2(1\pm\delta)^n + x_3y_3(1\pm\delta)^{n-1} + \cdots + x_ny_n(1\pm\delta)^2. \tag{1.12}$$

Con el siguiente Lema introduciremos una notación que nos ayudará a simplificar la expresión de (1.12).

Lema 1.5.1. Si $|\delta_i| \leq u$ y $\rho_i = \pm 1$ para $i = 1, \ldots, n$, y suponemos que $nu < 1$, entonces

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1 + \theta_n,$$

donde

$$|\theta_n| \leq \frac{nu}{1-nu} =: \gamma_n.$$

Demostración:

La demostración se hace por inducción y separaremos los casos de $\rho_n = +1$ y $\rho_n = -1$.

1. Para $n = 1$

   - Si $\rho_n = \rho_1 = 1$, debemos ver que $(1+\delta_1) = 1 + \theta_1$, con $|\theta_1| \leq u/(1-u)$. Como $\delta_1 = \theta_1$, basta con ver que $|\delta_1| \leq u/(1-u)$. Como sabemos que $|\delta_1| \leq u$, veremos que $u \leq u/(1-u)$. Por las condiciones del Lema sabemos que

     $$0 \leq u = 1 \cdot u < 1$$

     y, por tanto, que $1 - u > 0$, así que dividiendo a ambos lados por u y multiplicando por $1 - u$ tenemos que

     $$u \leq \frac{u}{(1-u)} \Leftrightarrow 1 - u \leq 1 \Leftrightarrow u \geq 0.$$

     Por tanto, como $u \geq 0$, el Lema se cumple para $n = 1$ y $\rho_n = +1$.

- Si $\rho_n = \rho_1 = -1$, debemos ver que $\dfrac{1}{1+\delta_1} = 1+\theta_1$, con $|\theta_1| \leq u/(1-u)$. Como $\dfrac{1}{1+\delta_1} = 1 - \dfrac{\delta_1}{1+\delta_1}$, llegamos a que $|\theta_1| = |\dfrac{\delta_1}{1+\delta_1}| \leq \dfrac{|\delta_1|}{1+|\delta_1|} \leq \dfrac{u}{1+|\delta_1|}$. Como $1-u \leq 1+|\delta_1|$ pues $u > 0$, concluimos que $|\theta_1| \leq u/(1-u)$.

2. Suponemos que se cumple para $n-1$, veamos, de nuevo separando $\rho_n = 1$ y $\rho_n = -1$, que se cumple para n.

   - Comencemos por el caso en el que $\rho_n = +1$. Por un lado tenemos que
   $$\prod_{i=1}^{n}(1+\delta_i) = (1+\delta_n)(1+\theta_{n-1}) = 1+\theta_n,$$
   luego operando llegamos a que,
   $$\theta_n = \delta_n + (1+\delta_n)\theta_{n-1}.$$
   Aplicando que $|\delta_n| \leq u$ y la hipótesis de inducción para $n-1$ llegamos a que
   $$\begin{aligned} |\theta_n| &\leq u + (1+u)\frac{(n-1)u}{1-(n-1)u} \\ &= \frac{u(1-(n-1)u) + (1+u)(n-1)u}{1-(n-1)u} \\ &= \frac{nu}{1-(n-1)u} \leq \gamma_n. \end{aligned} \tag{1.13}$$
   Y por tanto, queda probado por inducción que el Lema se cumple para el caso $\rho_n = +1$.
   - Para el caso en el que $\rho_n = -1$, encontramos, de manera similar, que
   $$|\theta_n| \leq \frac{nu - (n-1)u^2}{1-nu+(n-1)u^2} \leq \gamma_n.$$

□

La notación presentada en el lema anterior se utilizará a lo largo del texto en numerosas ocasiones y siempre que se utilice se supondrá que $nu < 1$, lo cual es verdad en prácticamente cualquier circunstancia que nos encontremos trabajando con aritmética IEEE de doble o simple precisión. Por tanto, aplicando el Lema 1.5.1 a la expresión de (1.12), llegamos a que

$$\hat{s}_n = x_1y_1(1+\theta_n) + x_2y_2(1+\theta'_n) + x_3y_3(1+\theta_{n-1}) + \cdots + x_ny_n(1+\theta_2).$$

Este es un resultado de error regresivo y podemos interpretarlo como que el producto interno calculado es el exacto para un conjunto perturbado de datos $x_1, \ldots, x_n$, $y_1(1+\theta_n)$, $y_2(1+\theta'_n), \ldots, y_n(1+\theta_2)$ (alternativamente, podríamos perturbar los x_i y dejar los y_i sin cambios). Cada perturbación relativa está acotada por $\gamma_n = nu/(1-nu)$, por lo que las perturbaciones son pequeñas.

Este resultado aplica a un orden concreto de evaluación y es fácil ver que para cualquier orden de evaluación tenemos, usando notación vectorial,

$$fl\left(\mathbf{x}^T\mathbf{y}\right) = (\mathbf{x}+\Delta\mathbf{x})^T\mathbf{y} = \mathbf{x}^T(\mathbf{y}+\Delta\mathbf{y}), \quad |\Delta\mathbf{x}| \leq \gamma_n|\mathbf{x}|, \quad |\Delta\mathbf{y}| \leq \gamma_n|\mathbf{y}|, \tag{1.14}$$

donde $|\mathbf{x}|$ denota el vector con elementos $|x_i|$. Por tanto, deducimos la siguiente cota de error de forma inmediata

$$|\mathbf{x}^T\mathbf{y} - fl(\mathbf{x}^T\mathbf{y})| = |\mathbf{x}^T\mathbf{y} - (\mathbf{x}+\Delta\mathbf{x})^T\mathbf{y}| \leq |\mathbf{x}^T\mathbf{y} - (1+\gamma_n)\mathbf{x}^T\mathbf{y}| = \gamma_n|\mathbf{x}^T\mathbf{y}|. \tag{1.15}$$

También podemos escribir esta expresión como

$$|\mathbf{x}^T\mathbf{y} - fl(\mathbf{x}^T\mathbf{y})| \leq nu|\mathbf{x}^T\mathbf{y}| + O(u^2). \tag{1.16}$$

Ambas expresiones son válidas y la elección de trabajar con una o con otra dependerá del contexto. Por ejemplo, usar cotas de primer orden como en (1.16) puede facilitar el álgebra del análisis pero también puede generar algunas dudas sobre el tamaño del término constante oculto por la O de Landau.

Si $\mathbf{y} = \mathbf{x}$, de manera que estamos formando una suma de cuadrados $\mathbf{x}^T\mathbf{x}$, esto muestra que se obtiene una alta precisión relativa. Sin embargo, en general, una alta precisión no está garantizada si $|\mathbf{x}^T\mathbf{y}| \ll |\mathbf{x}|^T|\mathbf{y}|$.

1.5.2. Multiplicación de un vector por una matriz

Sea $A \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$, e $\mathbf{y} = A\mathbf{x} \in \mathbb{R}^m$ [6]. Observamos que podemos construir el vector $\mathbf{y}$ como m productos internos de la forma, $y_i = \mathbf{a}_i^T\mathbf{x}$, $i = 1, \ldots, m$, siendo $\mathbf{a}_i^T$ la i-ésima fila de la matriz A. De (1.14) tenemos que la componente i-ésima calculada en aritmética de punto flotante $\hat{y}_i$ satisface

$$\hat{y}_i = fl(\mathbf{a}_i)^T\mathbf{x} = (\mathbf{a}_i + \Delta a_i)^T\mathbf{x}, \qquad |\Delta\mathbf{a}_i| \leq \gamma_n|\mathbf{a}_i|$$

lo que nos permite concluir con este resultado sobre el error regresivo

$$\hat{\mathbf{y}} = (A + \Delta A)\mathbf{x}, \qquad |\Delta A| \leq \gamma_n|A| \tag{1.17}$$

y con este otro para el error progresivo

$$|\mathbf{y} - \hat{\mathbf{y}}| \leq \gamma_n|A||\mathbf{x}|.$$

1.5.3. Factorización LU de una matriz

Sea $A \in \mathbb{R}^{n\times n}$ y suponemos que podemos descomponerla en el producto de dos matrices $A = LU$ siendo $L \in \mathbb{R}^{n\times n}$ una matriz triangular inferior con unos en la diagonal y $U \in \mathbb{R}^{n\times n}$ una matriz triangular superior invertible [6]. Esta factorización es especialmente útil para resolver sistemas de ecuaciones de la forma $A\mathbf{x} = \mathbf{b}$, ya que nos bastaría con

1. Resolver $L\mathbf{y} = \mathbf{b}$.

2. Después, resolver $U\mathbf{x} = \mathbf{y}$.

Aunque, a priori, resolver un sistema utilizando la factorización LU pueda parecer más costoso, en realidad, es notablemente menos costoso en general debido a la forma concreta de las matrices L y U que nos permite resolver los sistemas de forma más barata.

En general, cuando hablemos de calcular la factorización LU de una matriz A nos referiremos a calcularla utilizando el algoritmo de eliminación gaussiana, con o sin pivotaje.
Aunque la factorización LU no es uno de los objetivos principales de este texto, presentaremos ahora unos resultados cuya demostración puede encontrarse en [6] y que serán utilizados posteriormente.

Teorema 1.5.1. Si aplicamos eliminación gaussiana a $A \in \mathbb{R}^{n\times n}$ para hallar su factorización LU, los factores calculados en aritmética finita $\widehat{L} \in \mathbb{R}^{n\times n}$ y $\widehat{U} \in \mathbb{R}^{n\times n}$ satisfacen

$$\widehat{L}\widehat{U} = A + \Delta A, \quad \text{con} \quad |\Delta A| \leq \gamma_n |\widehat{L}||\widehat{U}|.$$

Teorema 1.5.2. Sea $A \in \mathbb{R}^{n\times n}$ y suponemos que calculamos la factorización LU con eliminación gaussiana. Sean $\widehat{L}$ y $\widehat{U}$ los factores calculados, y $\hat{\mathbf{x}}$ una solución calculada para $A\mathbf{x} = \mathbf{b}$. Entonces

$$(A + \Delta A)\hat{\mathbf{x}} = \mathbf{b}, \quad \text{con} \quad |\Delta A| \leq \gamma_{3n} |\widehat{L}||\widehat{U}|.$$

Ahora presentaremos el concepto de factor de crecimiento que medirá el posible aumento en la magnitud de los elementos de la matriz a lo largo del proceso de eliminación gaussiana sin pivotaje.

Definición 1.5.1. El factor de crecimiento en la eliminación gaussiana se define como

$$\rho_n = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|}, \tag{1.18}$$

donde $a_{ij}^{(k)}$ representa el elemento i, j de la matriz en la etapa k del proceso de eliminación (justo antes de eliminar la incógnita x_k). Este factor cuantifica el crecimiento de los elementos de la matriz durante la factorización LU sin pivotaje.

Utilizando la cota

$$|u_{ij}| = |a_{ij}^{(n-1)}| \leq \rho_n \max_{i,j} |a_{ij}|, \tag{1.19}$$

llegamos al siguiente teorema clásico cuya demostración se encuentra en [6].

Teorema 1.5.3. Sea $A \in \mathbb{R}^{n\times n}$ y suponemos que la eliminación gaussiana con pivotaje parcial produce una solución calculada $\hat{\mathbf{x}}$ para $A\mathbf{x} = \mathbf{b}$. Entonces,

$$(A + \Delta A)\hat{\mathbf{x}} = \mathbf{b}, \quad \text{con} \quad \|\Delta A\|_\infty \leq n^2 \gamma_{3n} \rho_n \|A\|_\infty. \tag{1.20}$$

Capítulo 2

Refinamiento iterativo clásico

El refinamiento iterativo es una técnica utilizada para mejorar una aproximación $\hat{\mathbf{x}}$ a la solución de un sistema lineal

$$A\mathbf{x} = \mathbf{b}, \tag{2.1}$$

siendo $A \in \mathbb{R}^{n\times n}$ una matriz cuadrada no singular y $\mathbf{b} \in \mathbb{R}^n$.
El proceso consiste en tres pasos:

1. Calcular el residuo $\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}}$
2. Resolver $A\mathbf{d} = \mathbf{r}$
3. Actualizar $\mathbf{y} = \hat{\mathbf{x}} + \mathbf{d}$

Este proceso se repite si fuera necesario tomando $\hat{\mathbf{x}} = \mathbf{y}$.

La idea de esta técnica se basa en que si $\mathbf{r}$ y $\mathbf{d}$ se calculan con suficiente precisión obtendremos mejoras en la aproximación a la solución de (2.1). Además, la economía del refinamiento iterativo es favorable ya que basta con calcular una única vez una factorización de la matriz (habitualmente la factorización LU) ya que en las iteraciones posteriores la podremos reutilizar.

2.1. Análisis del error progresivo para el refinamiento iterativo clásico

Sea $A \in \mathbb{R}^{n\times n}$ una matriz no singular y $\hat{\mathbf{x}}$ una solución aproximada de $A\mathbf{x} = \mathbf{b}$.
Definimos $\mathbf{x}_1 = \hat{\mathbf{x}}$ y consideramos el siguiente algoritmo de refinamiento iterativo: Para $i = 1, 2, \ldots$

1. Calcular $\mathbf{r}_i = \mathbf{b} - A\mathbf{x}_i$, con precisión $\overline{u}$

2. Resolver $A\mathbf{d}_i = \mathbf{r}_i$, con precisión u

3. Actualizar $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{d}_i$, con precisión u

El refinamiento iterativo clásico utiliza $\overline{u} = u^2$, es decir, el residuo $\mathbf{r}_i$ se calcula con precisión doble que la utilizada para el resto de los cálculos.
De ahora en adelante, para simplificar la notación llamaremos $\mathbf{r}_i, \mathbf{d}_i$ y $\mathbf{x}_i$ a las cantidades calculadas. Lo único que debe cumplir el método con el que se van a resolver los sistemas lineales con matriz A es que la solución calculada $\hat{\mathbf{x}}$ para un sistema $A\mathbf{x} = \mathbf{b}$ sea solución exacta de un sistema lineal perturbado, es decir, que exista una matriz ΔA tal que

$$(A + \Delta A)\hat{\mathbf{x}} = \mathbf{b}, \quad \text{con} \quad |\Delta A| \leq uW, \tag{2.2}$$

donde W es una matriz no negativa dependiente de A, n y u, pero independiente de $\mathbf{b}$. En virtud del Teorema 1.5.2 esto se cumple si los sistemas lineales se resuelven con eliminación gaussiana, como se comentará más adelante. Vamos a analizar ahora el error que se produce en cada uno de los tres pasos de este algoritmo.

- Primero, analizamos el cálculo de $\mathbf{r}_i$, que se divide en dos etapas. En la primera calculamos, $\mathbf{s}_i = fl(\mathbf{b} - A\mathbf{x}_i) = \mathbf{b} - A\mathbf{x}_i + \Delta\mathbf{s}_i$ en la precisión (posiblemente) extendida $\bar{u}$. Por tanto, aplicando la cota (1.17) tenemos que

$$|\Delta\mathbf{s}_i| \leq \bar{\gamma}_{n+1}(|\mathbf{b}| + |A||\mathbf{x}_i|), \tag{2.3}$$

siendo $\bar{\gamma}_{n+1} \leq \dfrac{(n+1)\,\bar{u}}{1-(n+1)\,\bar{u}}$.
En la segunda etapa, redondeamos el residuo $\mathbf{r}_i$ a la precisión en la que estamos trabajando, $\mathbf{r}_i = fl(\mathbf{s}_i) = \mathbf{s}_i + \mathbf{f}_i$, donde $|\mathbf{f}_i| \leq u|\mathbf{s}_i|$.
Por lo tanto, uniendo los resultados obtenidos en el análisis de estas dos etapas, llegamos a que

$$\mathbf{r}_i = \mathbf{b} - A\mathbf{x}_i + \Delta\mathbf{r}_i, \qquad |\Delta\mathbf{r}_i| \leq u|\mathbf{b} - A\mathbf{x}_i| + (1+u)\bar{\gamma}_{n+1}(|\mathbf{b}| + |A||\mathbf{x}_i|).$$

Si escribimos $\mathbf{x}_i = \mathbf{x} + (\mathbf{x}_i - \mathbf{x})$ y sustituimos en esta expresión, obtenemos la cota

$$|\Delta\mathbf{r}_i| \leq [u + (1+u)\bar{\gamma}_{n+1}]|A||\mathbf{x} - \mathbf{x}_i| + 2(1+u)\bar{\gamma}_{n+1}|A||\mathbf{x}|. \tag{2.4}$$

- Para el segundo paso del algoritmo y utilizando (2.2) tenemos que $(A + \Delta A_i)\mathbf{d}_i = \mathbf{r}_i$, donde $|\Delta A_i| \leq uW$. Escribimos

$$(A + \Delta A_i)^{-1} = (A(I + A^{-1}\Delta A_i))^{-1} = (I + A^{-1}\Delta A_i)^{-1}A^{-1}.$$

Suponiendo que en alguna norma matricial derivada de una norma vectorial $\|A\Delta A_i\| < 1$ y desarrollando en serie de potencias la inversa de $I + A_i^{-1}\Delta A_i$ tenemos que

$$(I + A^{-1}\Delta A_i)^{-1} = \sum_{k=0}^{\infty}(-A^{-1}\Delta A_i)^k = I - A^{-1}\Delta A_i + \sum_{k=2}^{\infty}(-A^{-1}\Delta A_i)^k$$

y, podemos escribir

$$(A + \Delta A_i)^{-1} = (I + F_i)A^{-1}$$

con

$$F_i = -A^{-1}\Delta A_i + \sum_{k=2}^{\infty}(-A^{-1}\Delta A_i)^k.$$

Tomando valores absolutos

$$|F_i| \leq |A_i^{-1}\Delta A_i| + \sum_{k=2}^{\infty}\left(|A_i^{-1}||\Delta A_i|\right)^k.$$

Como por (2.2) sabemos que $|\Delta A_i| \leq uW$, podemos escribir el sumatorio como $O(u^2)$ y entonces, usando el Teorema 1.2.1

$$|F_i| \leq |A^{-1}||\Delta A_i| + O(u^2) \leq u|A^{-1}|W + O(u^2). \tag{2.5}$$

Debemos ver la matriz F_i como una matriz que describe la perturbación inducida por ΔA_i en la inversa de la matriz A. Por lo tanto,

$$\mathbf{d}_i = (I + F_i)A^{-1}\mathbf{r}_i = (I + F_i)(\mathbf{x} - \mathbf{x}_i + A^{-1}\Delta\mathbf{r}_i). \tag{2.6}$$

- Finalmente, para el tercer paso tenemos que

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{d}_i + \Delta\mathbf{x}_i, \quad |\Delta\mathbf{x}_i| \leq u|\mathbf{x}_i + \mathbf{d}_i| \leq u(|\mathbf{x} - \mathbf{x}_i| + |\mathbf{x}| + |\mathbf{d}_i|). \tag{2.7}$$

Aplicando la expresión obtenida en (2.6) para $\mathbf{d}_i$ tenemos que

$$\begin{aligned}\mathbf{x}_{i+1} &= \mathbf{x}_i + (I + F_i)(\mathbf{x} - \mathbf{x}_i + A^{-1}\Delta\mathbf{r}_i) + \Delta\mathbf{x}_i \\ &= \mathbf{x} + A^{-1}\Delta\mathbf{r}_i + F_i(\mathbf{x} - \mathbf{x}_i) + F_iA^{-1}\Delta\mathbf{r}_i + \Delta\mathbf{x}_i\end{aligned}$$

y llegamos a

$$\mathbf{x}_{i+1} - \mathbf{x} = F_i(\mathbf{x} - \mathbf{x}_i) + (I + F_i)A^{-1}\Delta\mathbf{r}_i + \Delta\mathbf{x}_i.$$

Acotando componente a componente, usando (2.7) y de nuevo (2.6),

$$\begin{aligned}
|\mathbf{x}_{i+1}-\mathbf{x}| &\leq |F_i||\mathbf{x}-\mathbf{x}_i| + (I+|F_i|)|A^{-1}||\Delta\mathbf{r}_i| + u|\mathbf{x}-\mathbf{x}_i| + u|\mathbf{x}| + u|\mathbf{d}_i| \\
&\leq |F_i||\mathbf{x}-\mathbf{x}_i| + (I+|F_i|)|A^{-1}||\Delta\mathbf{r}_i| \\
&\quad +u|\mathbf{x}-\mathbf{x}_i| + u|\mathbf{x}| + u(I+|F_i|)(|\mathbf{x}-\mathbf{x}_i| + |A^{-1}||\Delta\mathbf{r}_i|) \\
&= ((1+u)|F_i| + 2uI)\,|\mathbf{x}-\mathbf{x}_i| + (1+u)(I+|F_i|)|A^{-1}||\Delta\mathbf{r}_i| + u|\mathbf{x}|.
\end{aligned}$$

Teniendo en cuenta la cota para $|\Delta\mathbf{r}_i|$ que hallamos en (2.4) y sustituyendo en la cota anterior observamos que

$$\begin{aligned}
|\mathbf{x}_{i+1}-\mathbf{x}| &\leq ((1+u)|F_i| + 2uI)\,|\mathbf{x}-\mathbf{x}_i| \\
&\quad +(1+u)\,(u+(1+u)\bar{\gamma}_{n+1})\,(I+|F_i|)|A^{-1}||A||\mathbf{x}-\mathbf{x}_i| \\
&\quad +2(1+u)^2\bar{\gamma}_{n+1}(I+|F_i|)A^{-1}||A||\mathbf{x}| + u|\mathbf{x}| \\
&\quad = G_i|\mathbf{x}-\mathbf{x}_i| + \mathbf{g}_i,
\end{aligned}$$

con

$$G_i = (1+u)(|F_i| + 2uI + (1+u)(u+(1+u)\bar{\gamma}_{n+1})(I+|F_i|)|A^{-1}||A|)$$

y

$$\mathbf{g}_i = 2(1+u)^2\bar{\gamma}_{n+1}(I+|F_i|)A^{-1}||A||\mathbf{x}| + u|\mathbf{x}|.$$

Usando la cota obtenida para F_i en (2.5), podemos estimar

$$\begin{aligned}
G_i &\approx |F_i| + (u+\bar{\gamma}_{n+1})(I+|F_i|)|A^{-1}||A| \\
&\lesssim u|A^{-1}|W + (u+\bar{\gamma}_{n+1})(I+u|A^{-1}|W)|A|,
\end{aligned} \tag{2.8}$$

$$\begin{aligned}
\mathbf{g}_i &\approx 2\bar{\gamma}_{n+1}(I+|F_i|)|A^{-1}||A||\mathbf{x}| + u|\mathbf{x}| \\
&\lesssim 2\bar{\gamma}_{n+1}(I+u|A^{-1}|W)|A^{-1}||A||\mathbf{x}| + u|\mathbf{x}|
\end{aligned} \tag{2.9}$$

En general, tendremos que $||G_i||_\infty < 1$ a no ser que A esté muy mal acondicionada y que el método que utilicemos para resolver los sistemas lineales sea muy inestable. Esto significa que el error $|\mathbf{x}-\mathbf{x}_i|$ se va contrayendo hasta llegar a un punto en el que el término $\mathbf{g}_i$ se vuelve significativo.

Definición 2.1.1. Definimos la precisión límite en norma infinito del refinamiento iterativo como el valor mínimo de

$$\|\mathbf{x} - \mathbf{x}_i\|_\infty / \|\mathbf{x}\|_\infty \tag{2.10}$$

La precisión límite en norma, es aproximadamente $||\mathbf{g}_i||_\infty / ||\mathbf{x}||_\infty$. Esto se debe a que como $||G_i||_\infty < 1$ entonces el método converge y por tanto $\|\mathbf{x} - \mathbf{x}_i\|_\infty$ tiende a 0, como estamos trabajando con el mínimo de $\|\mathbf{x} - \mathbf{x}_i\|_\infty$ tenemos que

$$\min_{0 \le i \le n} \|\mathbf{x}_i - \mathbf{x}\|_\infty \le \|G_i\|_\infty \|\mathbf{x} - \mathbf{x}_{i-1}\|_\infty + \|\mathbf{g}_i\|_\infty \approx \|\mathbf{g}_i\|_\infty. \tag{2.11}$$

Si nos fijamos en la aproximación (2.9) y recordamos la definición de número de condición (1.4) podemos hacer la aproximación siguiente:

$$\frac{\|\mathbf{g}_i\|_\infty}{\|\mathbf{x}\|_\infty} \approx 2n\bar{u}\ \mathrm{cond}(A, \mathbf{x}) + u. \tag{2.12}$$

De hecho, si existe cierto μ que verifique $2n\bar{u}(I + u|A^{-1}|W)|A^{-1}||A||\mathbf{x}| \le u\mu|\mathbf{x}|$, entonces el error relativo obtenido componente a componente será de orden μu, o lo que es lo mismo,

$$\min_{0 \le i \le n} |\mathbf{x} - \mathbf{x}_i| \lesssim \mu u |\mathbf{x}|. \tag{2.13}$$

Mirando la expresión (2.8) observamos que G_i es esencialmente independiente de $\bar{u}$. Por tanto, las tasas de convergencia del refinamiento iterativo utilizando precisión fija (trabajando solamente con u) o precisión mixta (trabajando con las dos precisiones u y $\bar{u}$) son muy similares. La diferencia de usar una u otra técnica se verá reflejada en la precisión límite.

Ya se ha comentado que habitualmente en el uso del refinamiento iterativo se utiliza $\bar{u} = u^2$ y, por tanto, podemos resumir nuestros hallazgos con el siguiente teorema.

Teorema 2.1.1. (refinamiento iterativo con precisión mixta) Apliquemos refinamiento iterativo en precisión fija a un sistema lineal $A\mathbf{x} = \mathbf{b}$ de tamaño $n \times n$ con matriz no singular usando un método que satisfaga (2.2) y calculando los residuos en precisión doble. Sea $\eta = u|| \, |A^{-1}|(|A| + W)||_\infty$. Entonces, suponiendo que η es suficientemente menor que 1, el refinamiento iterativo reduce el error directo por un factor de aproximadamente η en cada iteración, hasta que

$$||\mathbf{x} - \hat{\mathbf{x}}_i||_\infty / ||\mathbf{x}||_\infty \approx u.$$

Para la factorización LU, por el Teorema 1.5.2 tenemos que siendo $\widehat{L}$ y $\widehat{U}$ los factores calculados, se verifica que $|\Delta A| \leq \gamma_{3n}|\widehat{L}||\widehat{U}|$. Por tanto, podemos concluir que tomando

$$uW \equiv \gamma_{3n}|\widehat{L}||\widehat{U}|,$$

se verifica la condición (2.2) que debe cumplir el método empleado para resolver los sistemas lineales de matriz A.

Consideremos ahora el caso en que $\bar{u} = u$, lo que se denomina refinamiento iterativo en precisión fija. Tenemos un análogo del teorema previo.

Teorema 2.1.2. (Refinamiento iterativo en precisión fija) Apliquemos refinamiento iterativo en precisión fija a un sistema lineal $A\mathbf{x} = \mathbf{b}$ de tamaño $n \times n$ con matriz no singular usando un método que satisfaga (2.2) y sea $\eta = u|||A^{-1}|(|A| + W)||_\infty$. Entonces, suponiendo que η es suficientemente menor que 1, el refinamiento iterativo reduce el error directo por un factor aproximadamente η en cada iteración, hasta que

$$||\mathbf{x} - \hat{\mathbf{x}}_i||_\infty / ||\mathbf{x}||_\infty \lesssim 2n\text{cond}(A, \mathbf{x})u.$$

La principal diferencia entre utilizar precisión mixta o precisión fija es que en el caso de la precisión fija no podemos asegurar un error relativo del orden de u, como con la precisión mixta, sino que tan solo podemos asegurar un error relativo del orden de $\text{cond}(A, \mathbf{x})u$. Se trata de una cota más fuerte que la que se obtiene para la solución original calculada $\hat{\mathbf{x}}$, de la que sólo podemos decir que

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_\infty}{\|\mathbf{x}\|_\infty} \lesssim u \frac{\||A^{-1}|W|\mathbf{x}|\|_\infty}{\|\mathbf{x}\|_\infty}.$$

observando la expresión de η en el teorema anterior.

De hecho, un error relativo límite del orden de $\text{cond}(A, \mathbf{x})u$ es el mejor que podemos esperar si no usamos una precisión mayor, porque corresponde a la incertidumbre introducida al realizar perturbaciones relativas por componentes en A de tamaño u; este nivel de incertidumbre está usualmente presente, debido a errores en el cálculo de A o en el redondeo de sus elementos al representarlos en el sistema de coma flotante.

Como conclusión final de esta subsección podemos resaltar que, por lo que se refiere al error, el refinamiento iterativo merece la pena, incluso si los residuos se calculan únicamente con la precisión de trabajo u.

2.2. Análisis del error regresivo para el refinamiento iterativo en precisión fija

En esta sección haremos un análisis del error regresivo del refinamiento iterativo en precisión fija, que será aplicable a una amplia clase de métodos para

resolver sistemas de ecuaciones lineales.

Suponemos que la solución calculada $\hat{\mathbf{x}}$ de $A\mathbf{x} = \mathbf{b}$ satisface

$$|\mathbf{b} - A\hat{\mathbf{x}}| \leq u\,(g(A,\mathbf{b})|\hat{\mathbf{x}}| + \mathbf{h}(A,\mathbf{b}))\,, \tag{2.14}$$

donde $g : \mathbb{R}^{n\times(n+1)} \to \mathbb{R}^{n\times n}$ y $\mathbf{h} : \mathbb{R}^{n\times(n+1)} \to \mathbb{R}^n$ tienen entradas no negativas. Las funciones g y $\mathbf{h}$ pueden depender de n y u, así como de los datos A y $\mathbf{b}$. También suponemos que el residuo $\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}}$ se calcula de tal manera que si $\hat{\mathbf{r}}$ denota el residuo calculado

$$|\hat{\mathbf{r}} - \mathbf{r}| \leq u\,\mathbf{t}(A,\mathbf{b},\hat{\mathbf{x}}), \tag{2.15}$$

donde $\mathbf{t} : \mathbb{R}^{n\times(n+2)} \to \mathbb{R}^n$ es no negativo. Si $\mathbf{r}$ se calcula de la manera convencional con precisión u, entonces podemos tomar

$$\mathbf{t}(A,\mathbf{b},\hat{\mathbf{x}}) = \frac{\gamma_{n+1}}{u}(|A||\hat{\mathbf{x}}| + |\mathbf{b}|), \tag{2.16}$$

con $\gamma_{n+1} = (n+1)u/(1-(n+1)u)$. Primero damos un resultado asintótico que no hace ninguna suposición adicional sobre el método que se utiliza para resolver el sistema de ecuaciones lineales.

Teorema 2.2.1. Sea $A \in \mathbb{R}^{n\times n}$ no singular. Supongamos que el sistema lineal $A\mathbf{x} = \mathbf{b}$ se resuelve en aritmética de punto flotante y se obtiene una solución $\hat{\mathbf{x}}$, y que se aplica un paso de refinamiento iterativo que produce la solución corregida $\hat{\mathbf{y}}$. Si la solución calculada $\hat{\mathbf{x}}$ satisface (2.14) y el residuo calculado $\hat{\mathbf{r}}$ satisface (2.15), entonces la solución corregida $\hat{\mathbf{y}}$ satisface

$$|\mathbf{b} - A\hat{\mathbf{y}}| \leq u\,(h(A,\hat{\mathbf{r}}) + t(A,\mathbf{b},\hat{\mathbf{y}}) + |A||\hat{\mathbf{y}}|) + u\mathbf{q}, \tag{2.17}$$

donde $\mathbf{q} = O(u)$ si $t(A,\mathbf{b},\hat{\mathbf{x}}) - t(A,\mathbf{b},\hat{\mathbf{y}}) = O(\|\hat{\mathbf{x}} - \hat{\mathbf{y}}\|_\infty)$.

Demostración:

Para realizar esta demostración, iremos siguiendo por orden los pasos del algoritmo de refinamiento iterativo:

- En primer lugar, se considera el residuo exacto $\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}}$ de la solución inicialmente calculada $\hat{\mathbf{x}}$ que por (2.14), satisface

$$|\mathbf{r}| = |\mathbf{b} - A\hat{\mathbf{x}}| \leq u\big(g(A,\ \mathbf{b})\big|\hat{\mathbf{x}}| + \mathbf{h}(A,\mathbf{b})\big). \tag{2.18}$$

  El residuo calculado en aritmética de punto flotante es de la forma $\hat{\mathbf{r}} = \mathbf{r} + \Delta\mathbf{r}$, y por (2.15)

$$|\Delta\mathbf{r}| = |\hat{\mathbf{r}} - \ \mathbf{r}| \leq u\mathbf{t}(A,\ \mathbf{b},\hat{\mathbf{x}}). \tag{2.19}$$

- En el siguiente paso del algoritmo, resolvemos $A\mathbf{d} = \hat{\mathbf{r}}$. Por tanto, si aplicamos (2.14) al sistema $A\mathbf{d} = \hat{\mathbf{r}}$ tendremos que la corrección calculada $\hat{\mathbf{d}}$ satisface

$$A\hat{\mathbf{d}} = \hat{\mathbf{r}} + \mathbf{f}_1, \qquad \text{con } |\mathbf{f}_1| \leq u(g(A,\hat{\mathbf{r}})|\hat{\mathbf{d}}| + \mathbf{h}(A,\hat{\mathbf{r}})), \tag{2.20}$$

  ya que por (2.14) se tendría que

$$||\hat{\mathbf{r}}| - |A\hat{\mathbf{d}}|| \leq |\hat{\mathbf{r}} - A\hat{\mathbf{d}}| \leq u\big(g(A,\hat{\mathbf{r}})\big|\hat{\mathbf{d}}| + \mathbf{h}(A,\hat{\mathbf{r}})).$$

- Finalmente, realizaremos la suma $\mathbf{y} = \hat{\mathbf{x}} + \hat{\mathbf{d}}$. Como estamos utilizando el modelo estandarizado de aritmética explicado en la Sección 1.4.2 tenemos que:

$$fl(\hat{\mathbf{x}} + \hat{\mathbf{d}}) = (\hat{\mathbf{x}} + \hat{\mathbf{d}})(1+\delta),$$

  con $|\delta| \leq u$. Entonces, operando llegamos a que

$$(\hat{\mathbf{x}} + \hat{\mathbf{d}})(1+\delta) = \hat{\mathbf{x}} + \hat{\mathbf{d}} + \delta(\hat{\mathbf{x}} + \hat{\mathbf{d}})$$

  y como

$$|\delta(\hat{\mathbf{x}} + \hat{\mathbf{d}})| = |\delta||(\hat{\mathbf{x}} + \hat{\mathbf{d}})| \leq u(|\hat{\mathbf{x}}| + |\hat{\mathbf{d}}|)$$

  se puede escribir

$$\hat{\mathbf{y}} = fl(\hat{\mathbf{x}} + \hat{\mathbf{d}}) = \hat{\mathbf{x}} + \hat{\mathbf{d}} + \mathbf{f}_2, \quad \text{con } |\mathbf{f}_2| \leq u(|\hat{\mathbf{x}}| + |\hat{\mathbf{d}}|). \tag{2.21}$$

  Reuniendo los resultados anteriores obtenemos

$$\mathbf{b} - A\hat{\mathbf{y}} = \mathbf{b} - A\hat{\mathbf{x}} - A\hat{\mathbf{d}} - A\mathbf{f}_2 = \hat{\mathbf{r}} - \Delta\mathbf{r} - A\hat{\mathbf{d}} - A\mathbf{f}_2 = -\mathbf{f}_1 - \Delta\mathbf{r} - A\mathbf{f}_2.$$

Por lo tanto, combinando todas las acotaciones que hemos ido calculando llegamos a que

$$\begin{aligned} |\mathbf{b} - A\hat{\mathbf{y}}| &\leq u\big(g(A,\hat{\mathbf{r}})|\hat{\mathbf{d}}| + \mathbf{h}(A,\hat{\mathbf{r}})\big) + u\,\mathbf{t}(A,\mathbf{b},\hat{\mathbf{x}}) + u\,|A|\big(|\hat{\mathbf{x}}| + |\hat{\mathbf{d}}|\big) \\ &= u\big(\mathbf{h}(A,\hat{\mathbf{r}}) + \mathbf{t}(A,\mathbf{b},\hat{\mathbf{y}}) + |A|\,|\hat{\mathbf{y}}|\big) + u\,\mathbf{q} \end{aligned} \tag{2.22}$$

donde

$$\mathbf{q} = \mathbf{t}(A,\mathbf{b},\hat{\mathbf{x}}) - \mathbf{t}(A,\mathbf{b},\hat{\mathbf{y}}) + g(A,\hat{\mathbf{r}})|\hat{\mathbf{d}}| + |A|(|\hat{\mathbf{x}}| - |\hat{\mathbf{y}}| + |\hat{\mathbf{d}}|).$$

Como tenemos que $\hat{\mathbf{x}} - \hat{\mathbf{y}}, |\hat{\mathbf{x}}| - |\hat{\mathbf{y}}|$ y $\hat{\mathbf{d}}$ son de orden u podemos finalmente concluir $\mathbf{q} = O(u)$ si $t(A,\mathbf{b},\hat{\mathbf{x}}) - t(A,\mathbf{b},\hat{\mathbf{y}}) = O(\|\hat{\mathbf{x}} - \hat{\mathbf{y}}\|_\infty)$. □

El Teorema 2.2.1 muestra que, a primer orden, el error regresivo componente a componente $\omega_{|A|,|\mathbf{b}|}(\mathbf{y})$ será pequeño después del primer paso de refinamiento iterativo, siempre y cuando $\mathbf{h}(A, \hat{\mathbf{r}})$ y $\mathbf{t}(A, b, \hat{\mathbf{y}})$ estén acotados por un múltiplo modesto de $|A||\hat{\mathbf{y}}| + |\mathbf{b}|$. Para comprender dicha observación basta con fijarse en la expresión de $\omega_{|A|,|\mathbf{b}|}(\mathbf{y})$ dada en el Teorema 1.2.2. Esto es cierto para $\mathbf{t}$ si el residuo se calcula de la manera convencional como vimos en (2.16), y en algunos casos podemos tomar $\mathbf{h} \equiv \mathbf{0}$, como se muestra abajo. Nótese que la función g de (2.14) no aparece en el término de primer orden de (2.17). Esta es la principal razón por la cual el refinamiento iterativo mejora la estabilidad ya que la posible inestabilidad manifestada en g es suprimida por la etapa de refinamiento.

Una debilidad del teorema anterior es que la cota (2.17) es asintótica. Dado que no se da una cota estricta para $\mathbf{q}$, es difícil sacar conclusiones firmes sobre el tamaño de $\omega_{|A|,|\mathbf{b}|}(\mathbf{y})$. El siguiente resultado resuelve este problema, a cambio de imponer más condiciones sobre las funciones g y $\mathbf{h}$ que aparecen en (2.14).

Para ello, comenzaremos introduciendo

$$\sigma(B,\ \mathbf{x}) = \frac{\max_i(|B||\mathbf{x}|)_i}{\min_i(|B||\mathbf{x}|)_i},$$

una medida de cómo de bien o mal escalado está el vector $|B||\mathbf{x}|$, que será una forma de medir la uniformidad o disparidad del tamaño de las distintas componentes del vector $|B||\mathbf{x}|$.

Teorema 2.2.2. Bajo las condiciones del Teorema 2.2.1, suponemos además que $g(A, \mathbf{b}) = G|A|$ y $\mathbf{h}(A, \mathbf{b}) = H|\mathbf{b}|$, donde $G, H \in \mathbb{R}^{n\times n}$ tienen entradas no negativas, y suponemos también que el residuo se calcula de la manera convencional. Entonces, existe una función

$$f(t_1, t_2) \approx \frac{t_2(t_1 + n + 1)}{\mathrm{cond}(A^{-1})} + \frac{2(t_1 + n + 2)^2(1 + ut_2)^2}{(n + 1)}$$

tal que si

$$\mathrm{cond}(A^{-1})\sigma(A, \hat{\mathbf{y}}) \leq \left(f\left(\|G\|_\infty, \|H\|_\infty\right) u\right)^{-1}$$

entonces

$$|\mathbf{b} - A\hat{\mathbf{y}}| \leq 2\gamma_{n+1}|A||\hat{\mathbf{y}}|.$$

Demostración:

En esta demostración, continuaremos el razonamiento de la demostración del teorema anterior desde (2.22) y usando la fórmula (2.16) para $\mathbf{t}$ puesto que

calculamos el residuo de forma tradicional. Con las nuevas hipótesis sobre g y $\mathbf{h}$ tenemos

$$|\mathbf{b} - A\hat{\mathbf{y}}| \leq u\big(g(A,\hat{\mathbf{r}})\big|\hat{\mathbf{d}}| + \mathbf{h}(A,\hat{\mathbf{r}})\big) + u\mathbf{t}(A,\mathbf{b},\hat{\mathbf{x}}) + u|A|(|\hat{\mathbf{x}}| + |\hat{\mathbf{d}}|) \tag{2.23}$$

$$= u(G|A||\hat{\mathbf{d}}| + H|\hat{\mathbf{r}}|) + u\frac{\gamma_{n+1}}{u}(|A||\hat{\mathbf{x}}| + |\mathbf{b}|) + u|A|(|\hat{\mathbf{x}}| + |\hat{\mathbf{d}}|) \tag{2.24}$$

$$\leq uH|\hat{\mathbf{r}}| + \gamma_{n+1}|\mathbf{b}| + (\gamma_{n+1} + u)|A||\hat{\mathbf{x}}| + u(I+G)|A||\hat{\mathbf{d}}|. \tag{2.25}$$

Por otro lado, la desigualdad (2.18) implica

$$|\mathbf{b}| - |A||\hat{\mathbf{x}}| \leq |\mathbf{b} - A\hat{\mathbf{x}}| = |\mathbf{r}| \leq u\big(g(A,\ \mathbf{b})\big|\hat{\mathbf{x}}| + \mathbf{h}(A,\mathbf{b})\big) = u\left(G|A||\hat{\mathbf{x}}| + H|\mathbf{b}|\right),$$

o lo que es lo mismo

$$(I - uH)|\mathbf{b}| \leq (I + uG)|A||\hat{\mathbf{x}}|.$$

Si $u\|H\|_\infty < 1/2$ (por ejemplo) entonces $I - uH$ es no singular con una inversa no negativa. Para ver que es invertible basta ver que $\ker(I - uH) = \{\mathbf{0}\}$. Razonaremos por reducción al absurdo suponiendo que existe $\mathbf{x}$ distinto de $\mathbf{0}$ que cumple que $\mathbf{x} \in \ker(I - uH)$. Entonces tendríamos que

$$(I - uH)\mathbf{x} = \mathbf{0} \Rightarrow \mathbf{x} - uH\mathbf{x} = \mathbf{0} \Rightarrow uH\mathbf{x} = \mathbf{x}.$$

Como $\|uH\|_\infty < 1$, tomando normas y usando la igualdad anterior se llega a

$$\|\mathbf{x}\|_\infty = \|uH\mathbf{x}\|_\infty \leq \|uH\|_\infty\|\mathbf{x}\|_\infty < \|\mathbf{x}\|_\infty,$$

lo cual es absurdo. Por tanto, queda probado que $I - uH$ es no singular.
Además, es conocido que $(I - uH)^{-1} = \sum_{k=0}^{\infty} u^k H^k$ y por tanto, como H es una matriz con entradas no negativas tenemos que $(I - uH)^{-1}$ también lo es por ser suma de matrices con entradas no negativas ya que $u > 0$.
También tenemos que

$$\left\|\sum_{k=0}^{\infty}(uH)^k\right\|_\infty \leq \sum_{k=0}^{\infty}\|(uH)^k\|_\infty \leq \sum_{k=0}^{\infty}\|uH\|_\infty^k = \frac{1}{1 - \|uH\|_\infty},$$

y como habíamos supuesto que $u\|H\|_\infty < 1/2$ tenemos que

$$\|(I - uH)^{-1}\|_\infty \leq 2.$$

Multiplicando por $(I - uH)^{-1}$ en (2.2) podemos acotar $|\mathbf{b}|$ de forma que

$$|\mathbf{b}| \leq (I - uH)^{-1}(I + uG)|A||\hat{\mathbf{x}}|.$$

Una vez hemos acotado $|\mathbf{b}|$ podemos suponer durante el resto de la demostración que $\mathbf{b} = \mathbf{0}$ para simplificarla sin que esto suponga perturbaciones importantes. Haciendo este reemplazo en (2.25) y aproximando $\gamma_{n+1} + u \approx \gamma_{n+1}$, tenemos

$$|\mathbf{b} - A\hat{\mathbf{y}}| \leq uH|\hat{\mathbf{r}}| + \gamma_{n+1}|A||\hat{\mathbf{x}}| + u(I+G)|A||\hat{\mathbf{d}}|. \tag{2.26}$$

Nuestra tarea ahora es acotar $|A||\hat{\mathbf{x}}|$, $|\hat{\mathbf{r}}|$ y $|A||\hat{\mathbf{d}}|$ en términos de $|\hat{\mathbf{y}}|$. Manipulando (2.21) obtenemos la desigualdad

$$|\hat{\mathbf{x}}| \leq (1-u)^{-1}\left(|\hat{\mathbf{y}}| + (1+u)|\hat{\mathbf{d}}|\right) \approx |\hat{\mathbf{y}}| + |\hat{\mathbf{d}}| \tag{2.27}$$

ya que por (2.21)

$$|\hat{\mathbf{x}}| = |\hat{\mathbf{y}} - \hat{\mathbf{d}} - \mathbf{f}_2| \leq |\hat{\mathbf{y}}| + |\hat{\mathbf{d}}| + |\mathbf{f}_2| \leq |\hat{\mathbf{y}}| + |\hat{\mathbf{d}}| + u(|\hat{\mathbf{x}}| + |\hat{\mathbf{d}}|).$$

Además, podemos acotar $|\hat{\mathbf{r}}|$ utilizando la cota para $|\mathbf{r}|$ de (2.18) y la cota para $|\Delta\mathbf{r}|$ dada por (2.19) para el caso particular en el que el residuo se calcula de la manera convencional

$$|\hat{\mathbf{r}}| = |\mathbf{r} + \Delta\mathbf{r}| \leq |\mathbf{r}| + |\Delta\mathbf{r}| \leq u\left(G|A||\hat{\mathbf{x}}| + H|\mathbf{b}|\right) + \gamma_{n+1}\left(|A||\hat{\mathbf{x}}| + |\mathbf{b}|\right),$$

y eliminando los términos $|\mathbf{b}|$ y utilizando (2.27) se obtiene

$$|\hat{\mathbf{r}}| \leq (uG + \gamma_{n+1}I)|A||\hat{\mathbf{x}}| \leq (uG + \gamma_{n+1}I)|A|(|\hat{\mathbf{y}}| + |\hat{\mathbf{d}}|). \tag{2.28}$$

Sustituyendo las cotas calculadas en (2.27) y (2.28) en (2.26) obtenemos que

$$\begin{aligned} |\mathbf{b} - A\hat{\mathbf{y}}| &\leq \left(\gamma_{n+1}I + uH(uG + \gamma_{n+1}I)\right)|A|\,|\hat{\mathbf{y}}| \\ &\quad + \left(\gamma_{n+1}I + u(I+G) + uH(uG + \gamma_{n+1}I)\right)|A|\,|\hat{\mathbf{d}}| \\ &=: \left(\gamma_{n+1}I + M_1\right)|A|\,|\hat{\mathbf{y}}| + M_2\,|A|\,|\hat{\mathbf{d}}|. \end{aligned} \tag{2.29}$$

donde

$$\begin{aligned} \|M_1\|_\infty &\leq u\|H\|_\infty(u\|G\|_\infty + \gamma_{n+1}), \\ \|M_2\|_\infty &\leq \gamma_{n+2} + u\|G\|_\infty + u\|H\|_\infty(u\|G\|_\infty + \gamma_{n+1}). \end{aligned}$$

A partir de (2.20), tenemos

$$\hat{\mathbf{d}} = A^{-1}(\hat{\mathbf{r}} + \mathbf{f}_1)$$

con

$$|\mathbf{f}_1| \leq u\big(g(A,\hat{\mathbf{r}})\big|\hat{\mathbf{d}}| + h(A,\hat{\mathbf{r}})\big),$$

y haciendo uso de (2.28) llegamos a la siguiente cota para $\hat{\mathbf{d}}$

$$\begin{aligned} |\hat{\mathbf{d}}| &\leq |A^{-1}|\left(|\hat{\mathbf{r}}| + uG|A||\hat{\mathbf{d}}| + uH|\hat{\mathbf{r}}|\right) \\ &\leq |A^{-1}|\left((I+uH)(uG+\gamma_{n+1}I)|A|(|\hat{\mathbf{y}}|+|\hat{\mathbf{d}}|) + uG|A||\hat{\mathbf{d}}|\right). \end{aligned}$$

Después de pre-multiplicar por $|A|$, esto se puede reorganizar como

$$(I - uM_3)|A||\hat{\mathbf{d}}| \leq u|A||A^{-1}|M_4|A||\hat{\mathbf{y}}|, \tag{2.30}$$

donde

$$\begin{aligned} M_3 &= |A||A^{-1}|((I+uH)(G+(\gamma_{n+1}/u)I)+G), \\ M_4 &= (I+uH)(G+(\gamma_{n+1}/u)I). \end{aligned}$$

Usando que

$$\frac{\gamma_{n+1}}{u} = \frac{nu}{u(1-nu)} = \frac{n}{1-nu} \leq \frac{n+1}{1-(n+1)u} \approx n+1,$$

tenemos las cotas

$$\begin{aligned} \|M_3\|_\infty &\leq \mathrm{cond}(A^{-1})(\|G\|_\infty + n + 1)(2 + u\|H\|_\infty), \\ \|M_4\|_\infty &\leq (\|G\|_\infty + n + 1)(1 + u\|H\|_\infty). \end{aligned}$$

Si, por ejemplo, $u\|M_3\|_\infty < 1/2$ entonces $I - uM_3$ es no singular con $\|(I - uM_3)^{-1}\|_\infty \leq 2$ (utilizando el mismo razonamiento que hicimos para probar que $I - uH$ era no singular y acotar su inversa) y podemos reescribir (2.30) como

$$|A||\hat{\mathbf{d}}| \leq u(I - uM_3)^{-1}|A||A^{-1}|M_4|A||\hat{\mathbf{y}}|. \tag{2.31}$$

Sustituyendo esta cota en (2.29) obtenemos

$$\begin{aligned} |\mathbf{b} - A\hat{\mathbf{y}}| &\leq (\gamma_{n+1}I + M_1 + uM_2(I - uM_3)^{-1}|A||A^{-1}|M_4)|A||\hat{\mathbf{y}}| \\ &= (\gamma_{n+1}I + M_5)|A||\hat{\mathbf{y}}| \leq (\gamma_{n+1} + \|M_5\|_\infty \sigma(A, \hat{\mathbf{y}}))|A||\hat{\mathbf{y}}|. \end{aligned} \tag{2.32}$$

Finalmente, acotamos $\|M_5\|_\infty$. Escribiendo $g = \|G\|_\infty, h = \|H\|_\infty$, tenemos

$$\begin{aligned} \|M_5\|_\infty \leq u^2gh + uh\gamma_{n+1} + 2u\big(\gamma_{n+2} + ug + u^2gh + uh\gamma_{n+1}\big) \\ \times\ \mathrm{cond}(A^{-1})\ (g + n + 1)(1 + uh). \end{aligned} \tag{2.33}$$

Esta expresión se aproxima a estar acotada por $u^2(h(g+n+1) + 2(g+n+2)^2(1+uh)^2)$. Por tanto, si $\|M_5\|_\infty \sigma(A, \hat{\mathbf{y}})$ es menor o igual que γ_{n+1}, llegamos al resultado que queríamos probar.

□

El teorema anterior dice que, mientras A no esté demasiado mal acondicionada, $|A||\hat{\mathbf{y}}|$ no esté demasiado mal escalado, es decir, $\mathrm{cond}(A^{-1})\sigma(A,\hat{\mathbf{y}})$ no sea demasiado grande, y el método utilizado para resolver los sistemas lineales con matriz A no sea demasiado inestable, es decir, $f(\|G\|_\infty, \|H\|_\infty)$ no sea demasiado grande, entonces

$$\omega_{|A|,|\mathbf{b}|} \leq 2\gamma_{n+1}$$

después de un paso de refinamiento iterativo. Nótese que el término $\gamma_{n+1}|A||\hat{\mathbf{y}}|$ en (2.32) proviene de la cota de error para la evaluación del residuo, por lo que esta cota para ω es la menor que podríamos esperar probar.

Apliquemos el Teorema 2.2.2 a la eliminación Gaussiana con o sin pivotaje. Si hay pivotaje, asumimos sin pérdida de generalidad, que no se requieren intercambios. El Teorema 1.5.2 muestra que podemos tomar

$$g(A,\ \mathbf{b}) \approx 3n|\widehat{L}||\widehat{U}|, \quad \mathbf{h}(A,\mathbf{b}) = \mathbf{0}.$$

donde $\widehat{L}, \widehat{U}$ son los factores LU calculados de A. Para aplicar el Teorema 2.2.2 usamos $A \approx \widehat{L}\widehat{U}$ y como entonces $|\widehat{U}| \approx |\widehat{L}^{-1}A|$ escribimos

$$g(A,b) \approx 3n|\widehat{L}||\widehat{L}^{-1}A| \leq 3n|\widehat{L}||\widehat{L}^{-1}||A|,$$

lo cual muestra que podemos tomar

$$G = 3n|\widehat{L}||\widehat{L}^{-1}|, \quad f(\|G\|_\infty, \|H\|_\infty) \approx 18n\||\widehat{L}||\widehat{L}^{-1}|\|_\infty^2.$$

Sin pivotaje, el factor de crecimiento $\||\widehat{L}||\widehat{L}^{-1}|\|_\infty$ no está acotado, pero con pivotaje parcial no puede exceder 2^n y típicamente es $O(n)$.

Podemos concluir que, para la eliminación Gaussiana con pivotaje parcial, un paso de refinamiento iterativo con precisión fija generalmente será suficiente para producir un error relativo pequeño por componentes mientras A no esté demasiado mal acondicionada y $|A||\hat{\mathbf{y}}|$ no esté demasiado mal escalado. Sin pivotaje, esto también es cierto con la condición adicional de que el cálculo de $\hat{\mathbf{x}}$ original no sea demasiado inestable.

Capítulo 3

Refinamiento iterativo con tres precisiones

Como ya hemos visto, el refinamiento iterativo es un método para mejorar una solución aproximada $\hat{\mathbf{x}}$ de un sistema lineal $A\mathbf{x} = \mathbf{b}$ mediante el cálculo del residuo $\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}}$, la resolución de la ecuación de corrección $A\mathbf{d} = \mathbf{r}$, la formación de la actualización $\mathbf{y} = \hat{\mathbf{x}} + \mathbf{d}$ y la repetición de estos pasos según sea necesario. En capítulos anteriores, hemos analizado paso a paso el error de este algoritmo utilizando 2 precisiones diferentes o precisión fija para realizar estos pasos y durante este capítulo plantearemos resultados para cuando se consideran tres niveles de precisión:

- u, es la precisión con la que se almacenan los datos $A, \mathbf{b}$ y la solución $\mathbf{x}$ (la precisión de trabajo),
- u_f, es la precisión con la que se calcula la factorización de A,
- u_r, es la precisión con la que se calculan los residuos.

Se supone que las precisiones satisfacen

$$u_r \leq u \leq u_f. \tag{3.1}$$

El algoritmo también considera una cuarta precisión u_s con la que se resuelve (efectivamente) la ecuación de corrección, y se supone que $u \leq u_s \leq u_f$.

Mientras que u, u_f y u_r pueden ser diferentes dependiendo del entorno de cálculo (idealmente determinado por el hardware), u_s, en los casos que nos interesan, será igual a u_f o u.

Siguiendo [2], vamos a ir exponiendo algunos resultados sin demostración sobre el análisis regresivo y el análisis progresivo para el algoritmo de refinamiento iterativo con 3 precisiones descrito en el Algoritmo 1.

Algoritmo 1 Refinamiento Iterativo

Sea la matriz no singular $A \in \mathbb{R}^{n\times n}$ y $\mathbf{b} \in \mathbb{R}^n$ dados en precisión u. Este algoritmo utiliza refinamiento iterativo para generar una secuencia de aproximaciones $\mathbf{x}_i$, todas almacenadas en precisión u, para la solución de $A\mathbf{x} = \mathbf{b}$.

1: Resolver $A\mathbf{x}_0 = \mathbf{b}$ en precisión u_f y almacenar $\mathbf{x}_0$ en precisión u.
2: **for** $i = 0 : \infty$ **do**
3: Calcular $\mathbf{r}_i = \mathbf{b} - A\mathbf{x}_i$ en precisión u_r y redondear $\mathbf{r}_i$ a precisión u_s.
4: Resolver $A\mathbf{d}_i = \mathbf{r}_i$ en precisión u_s y almacenar $\mathbf{d}_i$ en precisión u.
5: $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{d}_i$ en precisión u.
6: **end for**

Para realizar el análisis progresivo y regresivo supondremos que el método utilizado para resolver el sistema en el paso 4 del Algoritmo 1 produce una solución calculada $\hat{\mathbf{d}}_i$ para $A\mathbf{d}_i = \hat{\mathbf{r}}_i$ que satisface tres condiciones

$$\hat{\mathbf{d}}_i = (I + u_s E_i)\mathbf{d}_i, \quad u_s\|E_i\|_\infty < 1, \tag{3.2}$$

$$\|\hat{\mathbf{r}}_i - A\hat{\mathbf{d}}_i\|_\infty \leq u_s(c_1\|A\|_\infty\|\hat{\mathbf{d}}_i\|_\infty + c_2\|\hat{\mathbf{r}}_i\|_\infty), \tag{3.3}$$

$$|\hat{\mathbf{r}}_i - A\hat{\mathbf{d}}_i| \leq u_s G_i|\hat{\mathbf{d}}_i|, \tag{3.4}$$

donde c_1, c_2 son constantes no negativas y E_i, G_i son matrices con entradas no negativas. Todas ellas pueden depender de n, A, $\hat{\mathbf{r}}_i$ y u_s

3.1. Análisis progresivo

Vamos a resumir los resultados que se obtienen en [2] tras un exhaustivo análisis progresivo con el siguiente teorema y el siguiente corolario.

Teorema 3.1.1. Si aplicamos el Algoritmo 1 a un sistema lineal $A\mathbf{x} = \mathbf{b}$, donde $A \in \mathbb{R}^{n\times n}$ es no singular, y suponemos que el método que utilizamos para resolver el sistema en el paso 4 produce una solución $\hat{\mathbf{d}}_i$ que satisface (3.2), entonces para todo $i \geq 0$, la solución calculada $\hat{\mathbf{x}}_{i+1}$ tras la iteración i-ésima satisface

$$|\mathbf{x} - \hat{\mathbf{x}}_{i+1}| \leq F_i|\mathbf{x} - \hat{\mathbf{x}}_i| + \mathbf{f}_i,$$

donde

$$F_i = u_s(I + u_s|E_i|)|A^{-1}||C_i| + u_s|E_i|,$$

$$\mathbf{f}_i = (1 + u_s)\gamma_p^r(I + u_s|E_i|)|A^{-1}|(|\mathbf{b}| + |A||\hat{\mathbf{x}}_i|) + u|\hat{\mathbf{x}}_{i+1}|$$

y

$$\|F_i\|_\infty \leq 2u_s \,\text{mín}(\text{cond}(A), \kappa_\infty(A)\mu_i) + u_s\|E_i\|_\infty, \tag{3.5}$$

$$\|\mathbf{f}_i\|_\infty \leq 2(1+u_s)\gamma_p^r \|A^{-1}\||A|(|\mathbf{x}|+|\hat{\mathbf{x}}_i|)_\infty + u\|\hat{\mathbf{x}}_{i+1}\|_\infty, \tag{3.6}$$

siendo

$$\mu_i = \frac{\|A(\mathbf{x}-\hat{\mathbf{x}}_i)\|_\infty}{\|A\|_\infty \|\mathbf{x}-\hat{\mathbf{x}}_i\|_\infty}, \quad \text{con } \|A\|_\infty \neq 0,\ \|\mathbf{x}-\hat{\mathbf{x}}_i\|_\infty \neq 0 \tag{3.7}$$

El siguiente resultado nos da información sobre la convergencia del Algoritmo 1 y la precisión alcanzable con el mismo.

Corolario 3.1.1. Bajo las condiciones del Teorema 3.1.1, siempre que

$$\phi_i = 2u_s \,\mín(\text{cond}(A), \kappa_\infty(A)\mu_i) + u_s\|E_i\|_\infty \tag{3.8}$$

sea suficientemente menor que 1, el error progresivo se reduce en la i-ésima iteración por un factor aproximadamente ϕ_i hasta que se produce un iterante $\hat{\mathbf{x}}$ para el cual

$$\frac{\|\mathbf{x}-\hat{\mathbf{x}}\|_\infty}{\|\mathbf{x}\|_\infty} \lesssim 4pu_r\text{cond}(A,\mathbf{x}) + u, \tag{3.9}$$

siendo p el número máximo de componentes distintas de cero en cualquier fila de la matriz ampliada $[A\ b]$ ($n+1$, en general, si A no es dispersa).

Obsérvese que ϕ_i en (3.8) depende únicamente de u_s. Esto significa que la tasa de convergencia del refinamiento iterativo depende solamente de la precisión efectiva de las soluciones y no se ve afectada por la precisión con la que se almacenan los datos o se calcula el residuo. Por el contrario, la precisión límite (3.9) depende de u_r y de u, pero no de u_s.

También debemos tener en cuenta que la precisión u_f utilizada para la solución inicial en el paso 1 no aparece explícitamente en el Teorema 3.1.1 ni en el Corolario 3.1.1. Sin embargo, independientemente del método que utilicemos para resolver los sistemas en el Algoritmo 1, u_f afecta a la precisión de $\hat{\mathbf{x}}_0$ y, por lo tanto, al número de iteraciones requeridas para alcanzar el nivel de tolerancia fijado.

3.2. Análisis regresivo

Resumimos a continuación los resultados que se obtienen en [2] para el análisis regresivo, dividiéndolos en resultados obtenidos trabajando con la norma, en la Subsección 3.2.1, y resultados obtenidos trabajando componente a componente, en la Subsección 3.2.2.

3.2.1. Análisis regresivo en norma

Resumiremos los resultados sobre el análisis regresivo en norma con el siguiente teorema y el siguiente corolario, cuyas demostraciones se encuentran en [2].

Teorema 3.2.1. Si aplicamos el Algoritmo 1 a un sistema lineal $A\mathbf{x} = \mathbf{b}$ con una matriz no singular $A \in \mathbb{R}^{n\times n}$ que satisface $c_1\kappa_\infty(A)u_s < 1$, y suponemos que el método que utilizamos para resolver el sistema en el paso 4 produce una solución $\hat{\mathbf{d}}_i$ que satisface (3.2), entonces para todo $i \geq 0$, la solución calculada $\hat{\mathbf{x}}_{i+1}$ tras la iteración i-ésima satisface

$$\|\mathbf{b} - A\hat{\mathbf{x}}_{i+1}\|_\infty \leq \alpha_i\|\mathbf{b} - A\hat{\mathbf{x}}_i\|_\infty + \beta_i,$$

donde

$$\alpha_i = u_s\left(1 + (1+u_s)\frac{c_1\kappa_\infty(A) + c_2}{1 - c_1\kappa_\infty(A)u_s}\right)$$

y

$$\beta_i = \left(1 + \frac{u_s(c_1\kappa_\infty(A) + c_2)}{1 - c_1\kappa_\infty(A)u_s}\right)(1+u_s)\gamma_p^r(\|\mathbf{b}\|_\infty + \|A\|_\infty\|\hat{\mathbf{x}}_i\|_\infty) + u\|A\|_\infty\|\hat{\mathbf{x}}_{i+1}\|_\infty.$$

Corolario 3.2.1. Bajo las condiciones del Teorema 3.2.1, si

$$\phi = (c_1\kappa_\infty(A) + c_2)u_s \tag{3.10}$$

es suficientemente menor que 1, entonces el residuo se reduce en cada iteración por un factor aproximado de ϕ hasta que

$$\|\mathbf{b} - A\hat{\mathbf{x}}_i\|_\infty \lesssim \gamma_p^r(\|\mathbf{b}\|_\infty + \|A\|_\infty\|\hat{\mathbf{x}}_{i-1}\|_\infty) + u\|A\|_\infty\|\hat{\mathbf{x}}_i\|_\infty.$$

Bajo las condiciones del corolario anterior, suponiendo que $\|\hat{\mathbf{x}}_{i-1}\|_\infty \approx \|\hat{\mathbf{x}}_i\|_\infty$, lo cual es bastante razonable, y usando que $u_r \leq u$, se obtiene, finalmente que

$$\|\mathbf{b} - A\hat{\mathbf{x}}_i\|_\infty \lesssim pu(\|\mathbf{b}\|_\infty + \|A\|_\infty\|\hat{\mathbf{x}}_i\|_\infty). \tag{3.11}$$

Si recordamos la caracterización de $\eta(y)$ que encontramos en el Teorema 1.2.3 vemos que (3.11) equivale a

$$\eta(\hat{\mathbf{x}}_i) \lesssim pu$$

es decir, $\hat{\mathbf{x}}_i$ es una solución estable regresivamente, con respecto a la precisión de trabajo u. El tamaño de c_1 y c_2 en (3.3) da cuenta de la mayor o menor

estabilidad del método con el que se resuelven los sistemas lineales con matriz A, y mientras $(c_1\kappa_\infty(A) + c_2)u_s$ sea suficientemente menor que 1, el Corolario 3.2.1 garantiza que el refinamiento iterativo generará un error regresivo pequeño.

Cabe destacar que hay poca o ninguna ventaja en los errores regresivos en norma al calcular los residuos con una precisión extra, ya que observamos que α_i en el Teorema 3.2.1 es independiente de u_r y el residuo límite no es menor cuando $u_r < u$.

3.2.2. Análisis regresivo componente a componente

Resumiremos los resultados sobre el análisis regresivo componente a componente con el siguiente teorema y el siguiente corolario cuyas demostraciones se encuentran en [2].

Teorema 3.2.2. Si aplicamos el Algoritmo 1 a un sistema lineal $A\mathbf{x} = \mathbf{b}$ con una matriz no singular $A \in \mathbb{R}^{n\times n}$ y suponemos que el método que utilizamos para resolver el sistema en el paso 4 produce una solución $\hat{\mathbf{d}}_i$ que satisface (3.2), y que para todo i

$$u_s\||G_i|A^{-1}|\|_\infty + (1+u_s)\gamma_p^r \operatorname{cond}(A^{-1}) \leq \frac{1}{2}, \tag{3.12}$$

entonces, para $i \geq 0$, el iterante calculado $\hat{\mathbf{x}}_{i+1}$ satisface

$$|\mathbf{b} - A\hat{\mathbf{x}}_{i+1}| \leq W_i|\mathbf{b} - A\hat{\mathbf{x}}_i| + \mathbf{y}_i,$$

donde

$$W_i = u_s I + (1+u_s)M_1 Z_1|A^{-1}|,$$

$$\mathbf{y}_i = (1+u_s)(1+u)\gamma_p^r(I + M_1 Z_1|A^{-1}|)(|\mathbf{b}| + |A||\hat{\mathbf{x}}_{i+1}|) + u|A||\hat{\mathbf{x}}_{i+1}|,$$

y donde

$$Z_1 = u_s G_i + (1+u_s)\gamma_r^p|A|,$$

con G_i definida en (3.4) y

$$M_1 = (I - Z_1|A^{-1}|)^{-1}.$$

Para interpretar el teorema, observamos primero que $M_1 = I + O(u_s)$, pues

$$M_1 = (I - Z_1|A^{-1}|)^{-1} = \sum_{k=0}^{\infty}(Z_1|A^{-1}|)^k = I + \sum_{k=1}^{\infty}(Z_1|A^{-1}|)^k,$$

por lo que

$$W_i \lesssim u_s I + (u_s G_i + \gamma_p^r |A|)|A^{-1}|,$$

lo que implica que

$$\|W_i\|_\infty \lesssim u_s + u_s \||G_i|A^{-1}|\|_\infty + \gamma_p^r \operatorname{cond}(A^{-1}), \tag{3.13}$$

y además $\|W_i\|_\infty < 1$ por (3.12). En la práctica podemos esperar que $G_i \geq |A|$, por lo que el término dominante en esta cota será $u_s \||G_i|A^{-1}|\|_\infty$. Finalmente, concluimos que $\|W_i\|_\infty \ll 1$ si el método que utilizamos para resolver sistemas en el Algoritmo 1 no es demasiado inestable y A no está demasiado mal acondicionada en relación con la precisión u_s.

El lema siguiente nos dará una cota para el error regresivo componente a componente, $\omega_{|A||\mathbf{b}|}(\hat{\mathbf{x}}_{i+1})$.

Lema 3.2.1. Para $A \in \mathbb{R}^{n\times n}$ y $\mathbf{x} \in \mathbb{R}^n$ tenemos

$$|A||\mathbf{x}| \leq \|A\|_\infty \xi(\mathbf{x})|\mathbf{x}|,$$

donde $\xi(\mathbf{x}) = \max_j |x_j| / \min_j |x_j|$, con x_j denotando la j-ésima componente de $\mathbf{x}$.

Usando el lema anterior, llegamos a que

$$\begin{aligned} |y_i| \lesssim \gamma_p^r \Big(1 + \big\| \left(u_s G_i + \gamma_p^r |A|\right) |A^{-1}| \big\|_\infty \Big) \\ \times\ \xi\big(|\mathbf{b}| + |A||\hat{\mathbf{x}}_{i+1}|\big)\big(|\mathbf{b}| + |A||\hat{\mathbf{x}}_{i+1}|\big) + u|A||\hat{\mathbf{x}}_{i+1}|. \end{aligned} \tag{3.14}$$

Por lo tanto, el error regresivo relativo componente a componente de $\hat{\mathbf{x}}_{i+1}$ satisface

$$\omega(\hat{\mathbf{x}}_{i+1}) \lesssim \gamma_p^r(1 + u_s \||G_i|A^{-1}|\|_\infty + \gamma_p^r \operatorname{cond}(A^{-1}))\xi(|\mathbf{b}| + |A||\hat{\mathbf{x}}_{i+1}|) + u. \tag{3.15}$$

Esta cota será del orden de u siempre que el método que utilicemos para resolver sistemas en el Algoritmo 1 no sea demasiado inestable, A no esté demasiado mal acondicionada y $\xi(|\mathbf{b}|+|A||\hat{\mathbf{x}}_{i+1}|)$ no sea demasiado grande. La última condición esencialmente requiere que el vector $|\mathbf{b}| + |A||\mathbf{x}|$ no esté demasiado mal escalado.

En la Tabla 3.1 encontramos un resumen de las condiciones suficientes para la convergencia del refinamiento iterativo con tres precisiones y las cotas del valor límite para los errores progresivos, regresivos en norma y regresivos componente a componente que hemos desarrollado en la Sección 3.1 y la Sección 3.2.

Error	**Condición de convergencia**	**Cota del valor límite**
Progresivo	$2u_s \min(\text{cond}(A), \kappa_\infty(A)\mu_i)$ $+u_s\|E_i\|_\infty < 1$	$4pu_r \,\text{cond}(A, \mathbf{x}) + u$
Regresivo en norma	$(c_1\kappa_\infty(A) + c_2)u_s < 1$	pu
Regresivo componente a componente	$u_s\|\|G_i\|A^{-1}\|\|\|_\infty + (1 + u_s)$ $\times\gamma_p^r \,\text{cond}(A^{-1}) \leq 1/2$	$\gamma_p^r(1 + u_s\|\|G_i\|A^{-1}\|\|\|_\infty +$ $\gamma_p^r \,\text{cond}(A^{-1})) \times \xi(\|\mathbf{b}\| + \|A\|\|\mathbf{x}\|) + u$

Tabla 3.1: Resumen de los resultados obtenidos para el análisis progresivo y los análisis regresivos en norma y componente a componente.

Capítulo 4

Familias de matrices de prueba

Para ilustrar los resultados teóricos incluidos en los dos capítulos anteriores, hemos implementado el refinamiento iterativo con varias precisiones, utilizando para ello distintas familias de matrices para las que es posible conocer el comportamiento de su número de condición. En este capítulo se presentan muy brevemente las matrices ortogonales (matrices muy bien acondicionadas) y las matrices de Hilbert (matrices muy mal acondicionadas), y se hace un estudio más detallado de una familia de matrices construida en [3], para las que se conoce su número de condición y cuya factorización LU es conocida.

4.1. Matrices ortogonales

En esta subsección nos centraremos en hablar sobre las matrices ortogonales [4] y su importancia en nuestros experimentos. A continuación se proporciona una de las definiciones más extendidas para las matrices ortogonales.

Definición 4.1.1. Se dice que una matriz cuadrada $Q \in \mathbb{R}^{n\times n}$ es ortogonal si $Q^TQ = I$, o lo que es lo mismo $Q^{-1} = Q^T$.

Otra definición es la siguiente.

Definición 4.1.2. Se dice que una matriz cuadrada $Q \in \mathbb{R}^{n\times n}$ es ortogonal si sus columnas (y filas) forman un sistema ortonormal, es decir, sus columnas son vectores ortogonales y con norma 1 para el producto interno habitual.

En el siguiente teorema proporcionaremos algunas propiedades importantes sobre las matrices ortogonales.

Teorema 4.1.1. Sea $Q \in \mathbb{R}^{n\times n}$ una matriz ortogonal

1. Para todo $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, $\langle Q\mathbf{u}, Q\mathbf{v}\rangle = \langle \mathbf{u}, \mathbf{v}\rangle$ y $\|Q\mathbf{u}\|_2 = \|\mathbf{u}\|_2$.
2. Si λ es un valor propio real de Q, entonces $\lambda = \pm 1$.

Demostración:

1. Dados $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$:

$$\langle Q\mathbf{u}, Q\mathbf{v}\rangle = (Q\mathbf{u})^T(Q\mathbf{v}) = \mathbf{u}^T(Q^TQ)\mathbf{v} = \mathbf{u}^T I_n\mathbf{v} = \mathbf{u}^T\ \mathbf{v} = \langle \mathbf{u}, \mathbf{v}\rangle.$$

   En particular,

$$\|Q\mathbf{u}\|_2 = \sqrt{\langle Q\mathbf{u}, Q\mathbf{u}\rangle} = \sqrt{\langle \mathbf{u}, \mathbf{u}\rangle} = \|\mathbf{u}\|_2.$$

2. Sea $\lambda \in \mathbb{R}$ un valor propio de la matriz ortogonal $Q \in \mathbb{R}^{n\times n}$, existe $\mathbf{x} \in \mathbb{R}^n$ con $\mathbf{x} \neq 0$ tal que $Q\mathbf{x} = \lambda\mathbf{x}$.

   Esto implica que $\langle Q\mathbf{x}, Q\mathbf{x}\rangle = \langle \lambda\mathbf{x}, \lambda\mathbf{x}\rangle$, desarrollaremos ambos términos:

   - $\langle Q\mathbf{x}, Q\mathbf{x}\rangle = \|\mathbf{x}\|_2^2$.
   - $\langle \lambda\mathbf{x}, \lambda\mathbf{x}\rangle = \lambda^2\langle \mathbf{x}, \mathbf{x}\rangle = \lambda^2\|\mathbf{x}\|_2^2$

   Finalmente, como $\mathbf{x} \neq \mathbf{0}$, concluimos que $\|\mathbf{x}\|_2^2 \neq 0$ y por tanto $\lambda = \pm 1$.

□

El motivo por el cual se dice que las matrices ortogonales están bien acondicionadas es que su número de condición para la norma euclídea es 1, pues el número de condición de una matriz A, con respecto a la norma euclídia, resulta ser

$$\kappa_2(A) = \frac{\sigma_{\text{máx}}}{\sigma_{\text{mín}}}$$

donde $\sigma_{\text{máx}}$ y $\sigma_{\text{mín}}$ son el mayor y el menor valor singular de A, respectivamente. Para una matriz ortogonal, todos sus valores singulares son iguales a 1, ya que se calculan como las raíces de los autovalores de la matriz Q^TQ que es igual a la identidad, lo que implica que

$$\kappa_2(Q) = 1$$

Estas matrices nos serán muy útiles durante los experimentos al estar tan bien acondicionadas ya que las utilizaremos para comparar los resultados obtenidos con los obtenidos con otros tipos de matrices que, en principio, darán resultados peores.

4.2. Matrices de Hilbert

En esta subsección nos centraremos en hablar sobre las matrices de Hilbert [9] y su importancia en nuestros experimentos.

La matriz de Hilbert de dimensión n es una matriz cuadrada cuyos elementos son de la forma siguiente

$$H_{ij} = \frac{1}{i+j-1}.$$

De forma general, la matriz de Hilbert de dimensión n es de la forma

$$H_n = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{pmatrix}.$$

El motivo por el cual estas matrices serán tan importantes en nuestros experimentos es que están muy mal acondicionadas. En la Tabla 4.1 podemos observar cómo el número de condición crece de forma muy rápida con la dimensión de la matriz, llegando a ser del orden de 10^{16} para $n = 12$.

Dimensión	**Número de condición**
1	1
2	19.281
3	524.06
4	15514
5	$4{,}7661 \times 10^5$
6	$1{,}4951 \times 10^7$
7	$4{,}7537 \times 10^8$
8	$1{,}5258 \times 10^{10}$
9	$4{,}9315 \times 10^{11}$
10	$1{,}6025 \times 10^{13}$
11	$5{,}2218 \times 10^{14}$
12	$1{,}6455 \times 10^{16}$

Tabla 4.1: Número de condición de matrices de Hilbert hasta dimensión 12.

4.3. Familia de matrices con número de condición predefinido

En esta sección hablaremos sobre una familia de matrices propuestas en [3], con una característica fundamental, y es que podemos elegir los parámetros con los que se contruyen de forma que la matriz tenga el número de condición en la norma infinito que a nosotros nos interese.

4.3.1. Construcción de la familia de matrices

Para comenzar a exponer cómo se construyen las matrices de esta familia consideramos en primer lugar la matriz triangular superior $T(\theta) \in \mathbb{R}^{n\times n}$, cuyo elemento (i, j) viene dado por

$$(T(\theta))_{ij} = \begin{cases} 0, & i > j, \\ 1, & i = j, \\ -\theta, & i < j, \end{cases} \tag{4.1}$$

donde θ es un parámetro real no negativo. Es decir,

$$T(\theta) = \begin{pmatrix} 1 & -\theta & -\theta & \cdots & -\theta \\ 0 & 1 & -\theta & \cdots & -\theta \\ 0 & 0 & 1 & \cdots & -\theta \\ \vdots & \vdots & \vdots & \ddots & -\theta \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

Es fácil comprobar que los coeficientes de la inversa de $T(\theta)$ vienen dados por

$$(T(\theta)^{-1})_{ij} = \begin{cases} 0, & i > j, \\ 1, & i = j, \\ \theta(1+\theta)^{j-i-1}, & i < j. \end{cases} \tag{4.2}$$

Utilizando la matriz $T(\theta) \in \mathbb{R}^{n\times n}$ que acabamos de definir en (4.1), vamos a definir una familia biparamétrica de matrices $A(\alpha, \beta) \in \mathbb{R}^{n\times n}$ como

$$A(\alpha, \beta) \;=\; L(\alpha)\, U(\beta), \tag{4.3}$$

siendo

$$L \;=\; T(\alpha)^T, \quad U \;=\; T(\beta), \tag{4.4}$$

con

$$0 \leq \alpha \leq 1, \quad \beta \geq 0. \tag{4.5}$$

Dado que los elementos subdiagonales de L tienen módulo como máximo 1, la eliminación gaussiana con pivotaje parcial de A no requiere intercambios de filas cuando se aplica para hallar la factorización LU de la matriz $A = A(\alpha, \beta)$ en aritmética exacta.
En general, generar la matriz A multiplicando las matrices L y U requeriría $2n^3/3$ operaciones en coma flotante, lo cual es demasiado costoso en un entorno de computación a gran escala. Pero aprovechando la estructura de los factores L y U que encontramos en (4.4) podemos dar una expresión explícita de los elementos de A para cada n. Para facilitar este proceso, escribiremos de forma genérica la estructura de las matrices L y U.

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -\alpha & 1 & 0 & \cdots & 0 \\ -\alpha & -\alpha & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -\alpha & -\alpha & \cdots & -\alpha & 1 \end{pmatrix} \quad \text{y} \quad U = \begin{pmatrix} 1 & -\beta & -\beta & \cdots & -\beta \\ 0 & 1 & -\beta & \cdots & -\beta \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & -\beta \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Dado que L y U son matrices triangulares inferior y superior respectivamente, tendremos en general que

$$a_{ij} = \sum_{k=1}^{min(i,j)} l_{ik}\, u_{kj}.$$

En particular, para los elementos de la diagonal de A (en los cuales $i = j$), se tiene que

$$a_{ii} = \sum_{k=1}^{i} \ell_{ik} u_{ki} = \sum_{k=1}^{i-1} \alpha\beta + 1 \cdot 1 = 1 + (i-1)\alpha\beta.$$

Para aquellos elementos que se encuentren por encima de la diagonal ($i < j$) tenemos que

$$a_{ij} = \sum_{k=1}^{i} \ell_{ik} u_{kj} = \sum_{k=1}^{i-1} \alpha\beta - 1 \cdot \beta = -\beta + (i-1)\alpha\beta,$$

y para los elementos que se encuentren por debajo de la diagonal ($i > j$) tendremos que

$$a_{ij} = \sum_{k=1}^{j} \ell_{ik} u_{kj} = \sum_{k=1}^{j-1} \alpha\beta - \alpha \cdot 1 = -\alpha + (j-1)\alpha\beta.$$

Por lo tanto, el coeficiente (i, j) de la matriz $A(\alpha, \beta) \in \mathbb{R}^{n\times n}$ será de la forma

$$a_{ij} = \begin{cases} -\alpha + (j-1)\alpha\beta, & i > j, \\ 1 + (i-1)\alpha\beta, & i = j, \\ -\beta + (i-1)\alpha\beta, & i < j, \end{cases} \tag{4.6}$$

y podemos escribirla de la siguiente manera para comprender de forma más visual su estructura

$$A(\alpha,\beta) = \begin{pmatrix} 1 & -\beta & -\beta & \cdots & -\beta \\ -\alpha & 1+\alpha\beta & -\beta+\alpha\beta & \cdots & -\beta+\alpha\beta \\ -\alpha & -\alpha+\alpha\beta & 1+2\alpha\beta & \cdots & -\beta+2\alpha\beta \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\alpha & -\alpha+\alpha\beta & -\alpha+2\alpha\beta & \cdots & 1+(n-1)\alpha\beta \end{pmatrix}.$$

4.3.2. El efecto de los errores de redondeo

Ahora que sabemos cómo podemos construir las matrices de esta familia, hablaremos de los errores de redondeo que surgen en la construcción y factorización de las matrices $A(\alpha, \beta)$ en aritmética de punto flotante.

Primero, consideramos los errores en el cálculo de $A(\alpha, \beta)$ a partir de (4.6). Usando el modelo estándar de aritmética de punto flotante desarrollado en la Sección 1.4.2, encontramos que los coeficientes de la matriz calculada $\widehat{A}(\alpha, \beta)$ satisfacen

$$|\widehat{a}_{ij} - a_{ij}| \leq \begin{cases} \gamma_3(\alpha + (j-1)\alpha\beta), & i > j, \\ \gamma_3(1 + (i-1)\alpha\beta), & i = j, \\ \gamma_3(\beta + (i-1)\alpha\beta), & i < j, \end{cases}$$

donde recordemos que $\gamma_k = ku/(1-ku)$, siendo u la unidad de redondeo. Utilizamos γ_k, con $k = 3$ ya que para construir cada uno de los $\widehat{a}_{ij}$ hacen falta 4 operaciones, como podemos ver en (4.6). Por ejemplo, para el caso en el que $i > j$, para calcular $\widehat{a}_{ij}$, en primer lugar hallamos $j - 1$, en segundo lugar, $\alpha\beta$, en tercer lugar $(j-1)\alpha\beta$ y en cuarto lugar, $-\alpha + (j-1)\alpha\beta$.

Observando la estructura de $A(\alpha, \beta)$ en (4.6), es fácil darse cuenta de que

$$\max_{i,j} |a_{ij}| \;\geq\; \max\big(\beta,\, 1 + (n-1)\alpha\beta\big).$$

y por tanto, se tiene que

$$|\widehat{a}_{ij} - a_{ij}| \leq 2\gamma_3 \max_{i,j} |a_{ij}|,$$

y entonces,

$$\|A - \widehat{A}\|_\infty \leq 2\gamma_3 n \|A\|_\infty, \tag{4.7}$$

donde el factor n es pesimista. De hecho, también se cumple que $|A-\widehat{A}| \leq c\gamma_3|A|$ siempre que $|-\alpha+(j-1)\alpha\beta|$ y $|-\beta+(i-1)\alpha\beta|$ no excedan $\alpha+(j-1)\alpha\beta$ y $\beta+(i-1)\alpha\beta$, respectivamente, en más de un factor c para todos los $i \neq j$.

Esta última condición se cumple con una constante modesta c siempre que ni α ni β estén demasiado cerca de un elemento del conjunto $\{1/k : k = 1, 2, \ldots, n-1\}$. Concluimos que $A(\alpha, \beta)$ se construye con la precisión esperada.

También necesitamos mostrar que la factorización LU sin pivotaje es numéricamente estable para A. Por lo tanto, necesitamos acotar los elementos de todos los complementos de Schur que surgen en la factorización.

Definición 4.3.1. Sea $A \in \mathbb{R}^{n\times n}$ una matriz invertible que se factoriza mediante eliminación gaussiana sin pivotaje, es decir, $A = LU$, donde L es una matriz triangular inferior con unos en la diagonal y U es una matriz triangular superior, ambas invertibles. Durante el proceso de eliminación, en cada paso se produce una partición de la matriz y se define un *complemento de Schur* correspondiente.

Concretamente, si en un paso se particiona la matriz A como

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

donde A_{11} es invertible, el complemento de Schur se define como

$$S = A_{22} - A_{21}A_{11}^{-1}A_{12}.$$

La estabilidad numérica de la factorización LU sin pivotaje para A se garantiza si se pueden acotar adecuadamente los elementos de todos los complementos de Schur que surgen durante el proceso.

Durante la factorización, a partir de la estructura de $T_n(\theta)$, donde el subíndice n denota la dimensión de la matriz, tenemos que

$$A(\alpha,\beta) = T(\alpha)^T T(\beta) = \begin{bmatrix} 1 & \mathbf{0}^T \\ -\alpha\mathbf{e} & T_{n-1}(\alpha)^T \end{bmatrix} \begin{bmatrix} 1 & -\beta\mathbf{e}^T \\ \mathbf{0} & T_{n-1}(\beta) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -\beta\mathbf{e}^T \\ -\alpha\mathbf{e} & T_{n-1}(\alpha)^T T_{n-1}(\beta) + \alpha\beta\mathbf{e}\mathbf{e}^T \end{bmatrix},$$

donde $\mathbf{e}$ es el vector de unos de tamaño adecuado. Así, después del primer paso de eliminación Gaussiana, tenemos la matriz

$$\begin{bmatrix} \mathbf{1} & -\beta\mathbf{e}^T \\ \mathbf{0} & T_{n-1}(\alpha)^T T_{n-1}(\beta) \end{bmatrix},$$

en la cual $T_{n-1}(\alpha)^T T_{n-1}(\beta)$ es la submatriz principal de orden $n-1$ de $A(\alpha,\beta)$. Por inducción, se sigue que todos los elementos de todos los complementos de Schur son elementos de $A(\alpha,\beta)$, y por lo tanto, el factor de crecimiento es

$$\rho_n(A) = \frac{\text{máx}_{i,j,k}\,|a_{ij}^{(k)}|}{\text{máx}_{i,j}\,|a_{ij}|} = 1, \tag{4.8}$$

que es un resultado muy conveniente.
Recordemos que por el Teorema 1.5.1 sabemos que los factores L y U calculados $\widehat{L}$ y $\widehat{U}$ satisfacen

$$\widehat{L}\widehat{U} = A + \Delta A, \quad |\Delta A| \leq \gamma_n |\widehat{L}||\widehat{U}|. \tag{4.9}$$

Ahora, dado que $\widehat{\ell}_{ij} = \ell_{ij} + O(u)$ y $\widehat{u}_{ij} = u_{ij} + O(u)$, se tiene que $|\widehat{L}||\widehat{U}| = |L||U| + O(u)$, y también

$$|L||U| = (-L + 2I)(-U + 2I) = LU - 2(L + U) + 4I = A + G,$$

con

$$G = -2(L + U) + 4I. \tag{4.10}$$

Recordando que $\|A\|_\infty = \text{máx}_i \sum_j |a_{ij}|$, que $\|A\|_1 = \|A^T\|_\infty$ y la forma de las matrices $T(\theta)$ dadas por (4.1) tenemos

$$\begin{array}{rcccl} \|L\|_\infty & = & \|T(\alpha)^T\|_\infty & = & 1 + (n-1)\alpha \\ \|U\|_\infty & = & \|T(\beta)\|_\infty & = & 1 + (n-1)\beta \end{array} \tag{4.11}$$

y por tanto que

$$\|L\|_\infty \leq \|A\|_\infty \quad \text{y} \quad \|U\|_\infty \leq \|A\|_\infty. \tag{4.12}$$

De aquí en adelante trabajaremos con la condición $\alpha \leq \beta$, y usando (4.12) y la definición de G dada en (4.10), obtenemos

$$\|G\|_\infty \leq 2(\|L\|_\infty + \|U\|_\infty) + 4 \leq 2(\|A\|_\infty + \|A\|_\infty) + 4\|A\|_\infty = 8\|A\|_\infty.$$

Por lo tanto,

$$\|\Delta A\|_\infty \leq 9\gamma_n \|A\|_\infty + O(u) = 9nu\|A\|_\infty + O(u), \tag{4.13}$$

y concluimos que la factorización es numéricamente estable.

4.3.3. Matrices de la familia con un número de condición predefinido

En esta subsección hablaremos de cómo podemos elegir los parámetros α y β para que $A(\alpha, \beta)$ tenga el número de condición en norma infinito deseado κ_∞, lo cual es la ventaja principal de esta familia de matrices. Esta tarea se puede lograr eficientemente combinando una técnica económica para evaluar $\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$ con un algoritmo de búsqueda de raíces que se utiliza para resolver la ecuación escalar $\kappa_\infty(A(\alpha, \beta)) = \kappa$. Además, proporcionaremos fórmulas explícitas para $\|A\|_\infty$ y $\|A^{-1}\|_\infty$ que pueden evaluarse en un número de operaciones que apenas depende de la dimensión, y finalmente, discutiremos cómo pueden aprovecharse para encontrar un valor adecuado de α y β. Durante toda la subsección supondremos que $0 < \alpha \leq 1$ y que $\alpha \leq \beta$.

Para poder dar una expresión para $\kappa_\infty(A(\alpha, \beta))$, vamos a comenzar dando una expresión tanto para $\|A(\alpha, \beta)\|_\infty$ como para $\|A(\alpha, \beta)^{-1}\|_\infty$.

Para calcular $\|A(\alpha, \beta)\|_\infty$ debemos calcular la suma de los valores absolutos de todas las filas de la matriz $A(\alpha, \beta)$, y esto lo podemos hacer de forma sencilla (4.6). Si llamamos λ_i a la suma de los valores absolutos de los elementos de la fila i-ésima de $A(\alpha, \beta)$ tenemos

$$\begin{aligned}
\lambda_i &= \sum_{j=1}^{n} |a_{ij}| \\
&= \left(\sum_{j=1}^{i-1} |(j-1)\alpha\beta - \alpha| \right) + 1 + (i-1)\alpha\beta + \sum_{j=i+1}^{n} |(i-1)\alpha\beta - \beta| \qquad (4.14) \\
&= \alpha \left(\sum_{j=1}^{i-1} |1 - (j-1)\beta| \right) + 1 + (i-1)\alpha\beta + (n-i)\beta|1 - (i-1)\alpha|.
\end{aligned}$$

Para calcular el máximo de los λ_i, y por tanto, la norma infinito de $A(\alpha, \beta)$, analizaremos la diferencia $\Lambda_i = \lambda_{i+1} - \lambda_i$ entre la suma de los valores absolutos de dos filas consecutivas. Restando la fórmula de λ_i de la de λ_{i+1}, obtenemos

$$\begin{aligned}
\Lambda_i = {} & \alpha|1 - (i-1)\beta| + \alpha\beta \\
& + (n-i-1)\beta|1 - i\alpha| - (n-i)\beta|1 - (i-1)\alpha|. \qquad (4.15)
\end{aligned}$$

El valor de Λ_i depende del signo de las tres cantidades que se encuentran dentro de los valores absolutos en (4.15). En general, esto daría lugar a $2^3 = 8$ combinaciones posibles, pero si $1 - (i-1)\alpha \leq 0$ entonces $1 - i\alpha < 0$, y $\alpha \leq \beta$ implica que también $1 - (i-1)\beta \leq 0$. Por lo tanto, solo hay que analizar cinco casos de los ocho posibles.

En [3] tras un razonamiento largo pero no demasiado complejo en el que se analizan los cinco casos, llegan a una expresión cerrada para $\|A(\alpha,\beta)\|_\infty$ de la que hablaremos en el Teorema 4.3.1 y además observaremos que calcular $\|A(\alpha,\beta)\|_\infty$ requiere solo un número moderado de operaciones.

Para calcular $\|A(\alpha,\beta)^{-1}\|_\infty$, primero debemos calcular la suma de los valores absolutos de los elementos de cada fila de la matriz $A(\alpha,\beta)^{-1}$. Denotamos por δ_i a la suma correspondiente a la fila i-ésima. Teniendo en cuenta que

$$A(\alpha,\beta)^{-1} = \left[T(\alpha)^T \cdot T(\beta)\right]^{-1} = T(\beta)^{-1} \cdot \left[T(\alpha)^T\right]^{-1}$$

y que se puede utilizar (4.2) para obtener explícitamente los coeficientes de $A(\alpha,\beta)^{-1}$, se llega a que

$$\delta_i = (1+\alpha)^i \left((1+\alpha)^{-1} + \frac{\beta(1-r^{n-i})}{1-r} \right), \tag{4.16}$$

donde $r = (1+\alpha)(1+\beta)$.
Introduciendo de nuevo la diferencia entre la suma de los valores absolutos de los elementos de dos filas consecutivas, $\Delta_i = \delta_{i+1} - \delta_i$, con i entre 1 y $n-1$, se tiene que

(a) si $\Delta_i \leq 0$, entonces $\Delta_k \leq 0$ para $1 \leq k < i$;
(b) si $\Delta_i \geq 0$, entonces $\Delta_k \geq 0$ para $i < k < n$.

Tenemos

$$\begin{aligned}
\Delta_i &= (1+\alpha)^{i+1}\left(\frac{1}{1+\alpha} + \frac{\beta(1-r^{n-i-1})}{1-r}\right) - (1+\alpha)^i\left(\frac{1}{\alpha+1} + \frac{\beta(1-r^{n-i})}{1-r}\right) \\
&= \frac{(1+\alpha)^{i-1}\alpha(1-r) + (1+\alpha)^2\beta(1-r^{-i+n-1}) - (1+\alpha)\beta(1-r^{n-i})}{1-r} \\
&= \frac{(1+\alpha)^{i-1}\left(\alpha(1-r) + (1+\alpha)\alpha\beta + (1+\alpha)\beta r^{n-i-1}(r-1-\alpha)\right)}{1-r} \\
&= \frac{(1+\alpha)^{i-1} r^{-i-1}\left(-\alpha^2 r^{i+1} + (1+\alpha)^2\beta^2 r^n\right)}{1-r} \\
&= \frac{(1+\alpha)^{-2}(1+\beta)^{-i-1}\left(\alpha^2 r^{i+1} - (1+\alpha)^2\beta^2 r^n\right)}{r-1}.
\end{aligned}$$

Por lo tanto, las cantidades Δ_i y $\zeta_i = \alpha^2 r^{i+1} - (1+\alpha)^2\beta^2 r^n$ tienen el mismo signo, ya que los otros factores son necesariamente positivos siempre que α y β lo sean, y teniendo en cuenta que $0 \leq \alpha \leq 1$ y $\beta \geq 0$.

De manera similar, los signos de Δ_{i-1} y Δ_{i+1} dependen de los signos de

$$\zeta_{i-1} = \alpha^2 r^i - (1+\alpha)^2 \beta^2 r^n$$

y

$$\zeta_{i+1} = \alpha^2 r^{i+2} - (1+\alpha)^2 \beta^2 r^n,$$

respectivamente. Como $r = (1+\alpha)(1+\beta) > 1$, $\Delta_i \leq 0$ implica $\Delta_{i-1} < 0$ y $\Delta_i \geq 0$ implica $\Delta_{i+1} > 0$. Si continuamos con el razonamiento por inducción, como

$$\Delta_i = \delta_{i+1} - \delta_i, \quad 1 \leq i \leq n$$

podemos concluir que

$$\|A(\alpha,\beta)^{-1}\|_\infty = \text{máx}(\delta_1, \delta_n).$$

Con los resultados que acabamos de obtener, enunciaremos el siguiente teorema

Teorema 4.3.1. Sea $A(\alpha,\beta) = T(\alpha)^T T(\beta) \in \mathbb{R}^{n\times n}$, donde T está definido en (4.1). Para $0 < \alpha \leq 1$ y $\alpha \leq \beta$, tenemos

$$\|A(\alpha,\beta)\|_\infty = \text{máx}(\lambda_1, \lambda_{i'}, \lambda_n), \quad \|A(\alpha,\beta)^{-1}\|_\infty = \text{máx}(\delta_1, \delta_n), \tag{4.17}$$

donde $i' = \text{mín}\,(\lfloor 1/\alpha \rfloor, n)$, y λ_i y δ_i están definidos en (4.14) y (4.16), respectivamente. La notación $\lfloor 1/\alpha \rfloor$ hace referencia a la parte entera del número real $1/\alpha$.

Por tanto, deducimos que

$$\kappa_\infty(A(\alpha,\beta)) = \|A(\alpha,\beta)\|_\infty \|A(\alpha,\beta)^{-1}\|_\infty = \text{máx}(\lambda_1, \lambda_{i'}, \lambda_n)\,\text{máx}(\delta_1, \delta_n). \tag{4.18}$$

Calcular el número de condición de $A(\alpha,\beta)$ requiere solo un número de operaciones lineal en n, pues tanto calcular $\|A(\alpha,\beta)\|_\infty$, como calcular $\|A(\alpha,\beta)^{-1}\|_\infty$ también requieren de un número de operaciones lineal en n.

Vamos a analizar el efecto que tienen los errores de redondeo en el número de condición de la matriz calculada. Sabemos por (4.7) que la matriz calculada $\widehat{A}(\alpha,\beta)$ tiene un error relativo componente a componente pequeño. Además, en [3] se prueba que

$$\frac{|\kappa_\infty(\widehat{A}(\alpha,\beta)) - \kappa_\infty(A(\alpha,\beta))|}{\kappa_\infty(A(\alpha,\beta))} \lesssim 6nu\kappa_\infty(A(\alpha,\beta)).$$

Por lo tanto, si elegimos los parámetros α y β de forma que el número de condición $\kappa_\infty(A(\alpha,\beta)) \ll u^{-1}$, entonces $\kappa_\infty(\widehat{A}(\alpha,\beta))$ será del mismo orden de magnitud que $\kappa_\infty(A(\alpha,\beta))$, lo cual es un resultado muy bueno.

Para generar una matriz de prueba con un número de condición dado κ, es necesario determinar los posibles valores de α y β tales que

$$\kappa_\infty(A(\alpha,\beta)) = \kappa.$$

Fijándonos en la expresión para el número de condición dada en (4.18), observamos que esto equivale a encontrar un cero de la siguiente función

$$\begin{aligned} f(\alpha,\beta;n,\kappa) &= \|A(\alpha,\beta)\|_\infty \|A(\alpha,\beta)^{-1}\|_\infty - \kappa \\ &= \text{máx}(\lambda_1,\lambda_i',\lambda_n)\,\text{máx}(\delta_1,\delta_n) - \kappa, \end{aligned} \tag{4.19}$$

para $\alpha \in (0,1]$ y $\alpha \leq \beta$.
La solución para un valor dado de κ no es única. La función f es continua pero no diferenciable, y las raíces pueden encontrarse convirtiendo $f(\alpha,\beta;n,\kappa) = 0$ en una ecuación de un solo parámetro, definiendo, por ejemplo, α como múltiplo fijo de β. En ese caso, podemos escribir la función f como

$$f_\rho(\beta;n,\kappa) = f(\rho\beta,\beta;n,\kappa), \quad \rho \in (0,1], \tag{4.20}$$

Es importante asegurar que para una solución β_* se cumpla $\alpha = \rho\beta_* < 1$; si esta desigualdad no se satisface, entonces se deberá usar un valor diferente de ρ. Aunque hay numerosas formas de escoger α y β para obtener el número de condición deseado, una de las más comunes es escogiendo $\alpha = \beta/2$, es decir, escoger $\rho = 1/2$.

En la Tabla 4.2 reproducimos la Tabla 4.2 de [3] que corresponde a $\rho = \frac{1}{2}$ y donde se fija el parámetro β para construir matrices $A(\beta/2,\beta)$ de dimensiones entre $n = 10^2$, y $n = 10^{10}$, y con números de condición en norma infinito entre 10^2 y 10^4. En nuestros experimentos numéricos hemos utilizado únicamente matrices de dimensión $n = 100$ y $n = 1000$, y números de condición entre 10^2 y 10^8.

	$\kappa = 10^2$	$\kappa = 10^4$	$\kappa = 10^6$	$\kappa = 10^8$	$\kappa = 10^{10}$
$n = 10^2$	$2{,}54 \times 10^{-2}$	$5{,}35 \times 10^{-2}$	$8{,}07 \times 10^{-2}$	$1{,}09 \times 10^{-1}$	$1{,}40 \times 10^{-1}$
$n = 10^3$	$2{,}50 \times 10^{-3}$	$5{,}21 \times 10^{-3}$	$7{,}81 \times 10^{-3}$	$1{,}05 \times 10^{-2}$	$1{,}33 \times 10^{-2}$
$n = 10^4$	$2{,}50 \times 10^{-4}$	$5{,}20 \times 10^{-4}$	$7{,}79 \times 10^{-4}$	$1{,}04 \times 10^{-3}$	$1{,}32 \times 10^{-3}$

Tabla 4.2: Valores de β que generan una matriz $A(\beta/2,\beta)$ de tamaño $n \times n$ con número de condición en norma infinito igual a κ.

Capítulo 5

Experimentos numéricos

En esta sección se presentarán experimentos en `MATLAB` que ayudarán a comprender mejor los resultados presentados en la parte teórica y a darnos una visión más tangible de las mejoras que supone utilizar precisión mixta y tres precisiones en el refinamiento iterativo.

5.1. Importancia del número de condición en la convergencia del refinamiento iterativo

En esta sección vamos a poner en valor la importancia del número de condición de la matriz, mostrando a través del siguiente experimento que cuanto mayor sea el número de condición, mayor será el tiempo de CPU que requiere la ejecución del algoritmo para alcanzar una precisión determinada.

Vamos a trabajar únicamente con precisión doble pues las matrices de Hilbert están muy mal acondicionadas. Además, el refinamiento iterativo se detendrá si $\frac{\|\mathbf{x}_i - \mathbf{x}_{i-1}\|_\infty}{\|\mathbf{x}_i\|_\infty} \leq 10^{-4}$. En caso de no llegar a satisfacerse dicha condición, se parará ... el refinamiento iterativo en un máximo de 10 iteraciones.

En la Figura 5.1 se ha representado el tiempo de CPU necesario para alcanzar la precisión fijada (o el número máximo de iteraciones permitido) frente a la dimensión de la matriz, para tres tipos de matrices cuadradas de dimensiones $n = 500, 1000, \cdots, 2500$. Se han considerado las matrices de Hilbert (como ejemplo de matriz muy mal acondicionada), matrices ortogonales obtenidas mediante la ortonormalización de las columnas de matrices aleatorias (como ejemplo de matrices con número de condición unidad en norma euclídea) y matrices aleatorias, como un caso intermedio.

Si observamos la Figura 5.1, nos damos cuenta de que el tiempo de CPU aumenta notablemente con la dimensión de la matriz y además, para la misma

dimensión el tiempo de CPU es mayor que el resto para las matrices de Hilbert y menor que el resto para las matrices ortogonales, algo que cabía esperar por lo visto en capítulos anteriores.

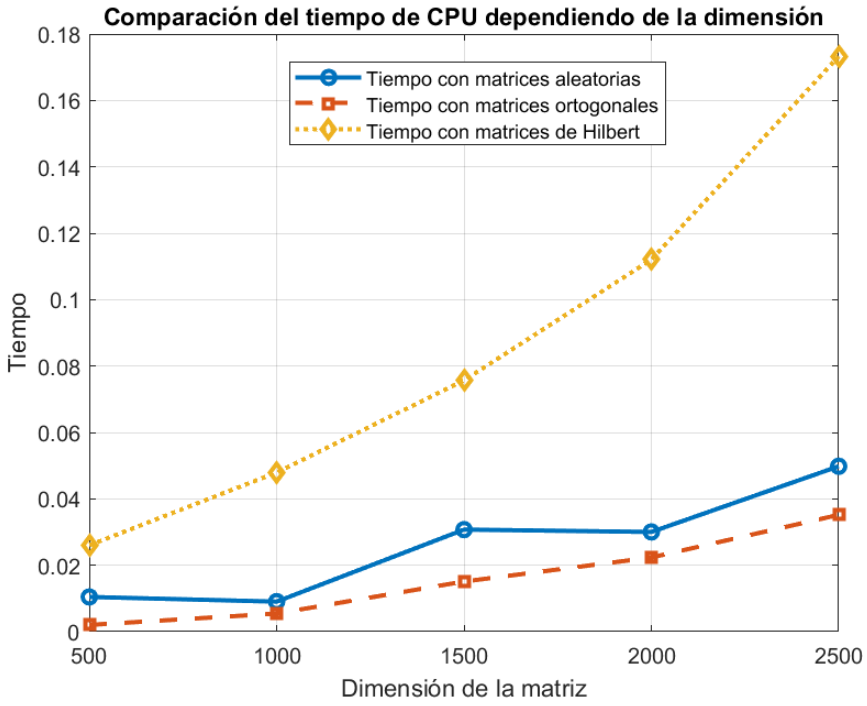

Figura 5.1: Gráfica que mide el tiempo de CPU al ejecutar el algoritmo de refinamiento iterativo con matrices aleatorias, ortogonales y de Hilbert de diferentes dimensiones.

Efectivamente, si miramos la Tabla 5.1 podemos ver los números de condición en norma infinito de las matrices utilizadas y comprobar que cuanto mayor es el número de condición, más tarda el algoritmo de refinamiento iterativo en completarse.

Dimensión	**Aleatorias**	**Ortogonales**	**Hilbert**
500	69375	411,15	$6{,}0791 \times 10^{20}$
1000	$1{,}0932 \times 10^{5}$	818,81	$1{,}1484 \times 10^{22}$
1500	$1{,}0694 \times 10^{8}$	1223,9	$2{,}2498 \times 10^{24}$
2000	$1{,}3065 \times 10^{6}$	1625,9	$2{,}1688 \times 10^{23}$
2500	$2{,}1482 \times 10^{6}$	2036	$9{,}4915 \times 10^{21}$

Tabla 5.1: Números de condición en norma infinito de las matrices de la Figura 5.1.

5.2. Comparación cuando se utilizan distintas precisiones

Durante esta sección analizamos cómo afectan las precisiones escogidas a la convergencia del refinamiento iterativo. Para ello trabajaremos con la familia de matrices de la Sección 4.3 y utilizaremos diferentes números de condición. Generalmente, y a no ser que se indique lo contrario, se trabajará con matrices de dimensión $n = 100$. Además, en todos los casos se tomará que la solución $\mathbf{x}$ del sistema lineal $A\mathbf{x} = \mathbf{b}$ será el vector de unos, luego $\mathbf{b}$ será el producto de $A\mathbf{1}$ calculado con la precisión de trabajo u.

En cada subsección, se presenta la combinación de precisiones con las que vamos a trabajar, algunos resultados teóricos sin demostración sobre cotas para la constante ϕ_i definida en (3.8) y para la precisión límite, y los resultados de los experimentos. Además, para cada experimento se incluirán dos tipos de gráficas para ilustrar los resultados:

- Una gráfica que muestra el factor ϕ_i por el que se reduce el error progresivo en la iteración i-ésima, definido en (3.8), y los distintos términos que contribuyen en la definición de ϕ_i: $2u_s\text{cond}(A)$, $2u_s\kappa_\infty(A)\mu_i$ y $u_s\|E_i\|_\infty$.

- Otra gráfica con el valor del error progresivo, del error regresivo en norma y del error regresivo componente a componente en cada iteración del refinamiento iterativo. Además, en estas gráficas se marcará con línea discontinua el valor de la precisión de trabajo u.

A lo largo de esta sección tomamos $u_s = u_f$. Además, como en la Sección 3.1, llamaremos p al número máximo de componentes distintas de cero en cualquier fila de la matriz ampliada $[A\ b]$.

En todas las gráficas se observa que se han dado 10 iteraciones. Esto se ha decidido de esta forma porque así se aprecia mucho mejor la convergencia del método, además de que visualmente es más sencillo comparar gráficas correspondientes a distintas configuraciones de precisiones.

5.2.1. Refinamiento tradicional con residuos en precisión extra

Consideremos el caso donde $u_f = u$ y $u_r = u^2$, lo que corresponde al refinamiento iterativo tradicional con residuos calculados al doble de la precisión de trabajo. Según el Corolario 3.1.1 (ver también Tabla 3.1), la máxima precisión que puede alcanzarse es

$$4pu^2\,\mathrm{cond}(A,\mathbf{x}) + u \leq 4pu\,\mathrm{cond}(A)\cdot u + u \leq \left(\frac{4}{3}\phi_i + 1\right) u, \tag{5.1}$$

por lo que mientras ϕ_i sea suficientemente menor que 1, se logrará una solución con error relativo en norma del orden de u.

Ilustraremos este resultado, en primer lugar, tratando el caso en el que $(u_f, u, u_r) = (h, h, s)$ y $\kappa_\infty(A) = 10^2$. En la gráfica izquierda de la Figura 5.2 observamos que ϕ_i es del orden de 10^{-1} en todas las iteraciones, y por tanto, suficientemente menor que 1. Esto se refleja en la gráfica derecha de la Figura 5.2 donde observamos que tras 3 iteraciones el error progresivo es del orden de u pues $u = 2^{-11} \approx 10^{-3}$ ya que la precisión de trabajo es la precisión media. Los errores regresivos son algo inferiores, siendo menor el medido componente a componente.

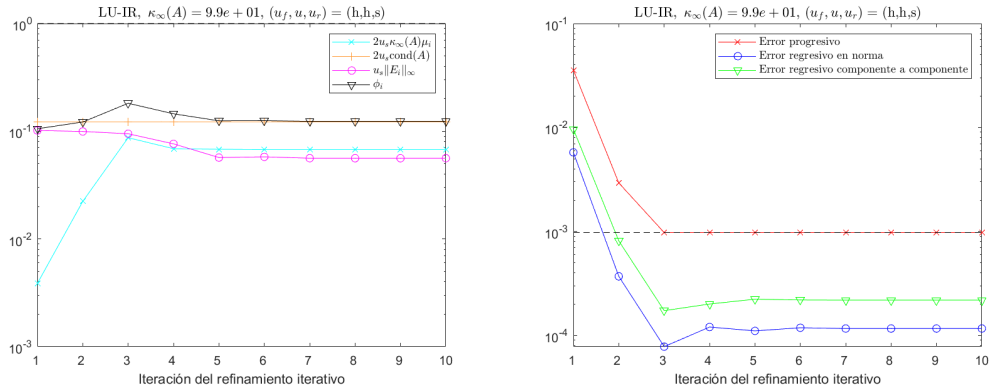

Figura 5.2: Gráfica con cotas (izquierda) y gráfica de errores (derecha) del refinamiento iterativo con precisiones $(u_f, u, u_r) = (h, h, s)$ y $\kappa_\infty(A) = 10^2$.

Podemos comparar los resultados anteriores con los que obtenemos utilizando precisiones $(u_f, u, u_r) = (s, s, d)$ y $\kappa_\infty(A) = 10^6$. En primer lugar cabe destacar que en el caso anterior hablábamos de una matriz con número de condición $\kappa_\infty(A) = 10^2$, muy inferior al número de condición con el que trabajamos ahora. La matriz ahora está mucho peor acondicionada, pero trabajamos con mayor precisión. Observando la gráfica derecha de la Figura 5.3 vemos que tras apenas 2 iteraciones ya obtenemos unos errores regresivos por debajo de u, que ahora es aproximadamente 10^{-7}, ya que estamos trabajando con precisión simple. El valor del error progresivo, aunque grande, es totalmente esperable si observamos la expresión del valor límite que aparece en la Tabla 3.1, pues $\kappa_\infty(A) = 10^6$. Los errores son significativamente inferiores para este caso que para el anterior (en ambos casos del orden de la unidad de redondeo, como se

esperaba) incluso trabajando con un número de condicion mucho mayor y en una iteración menos.

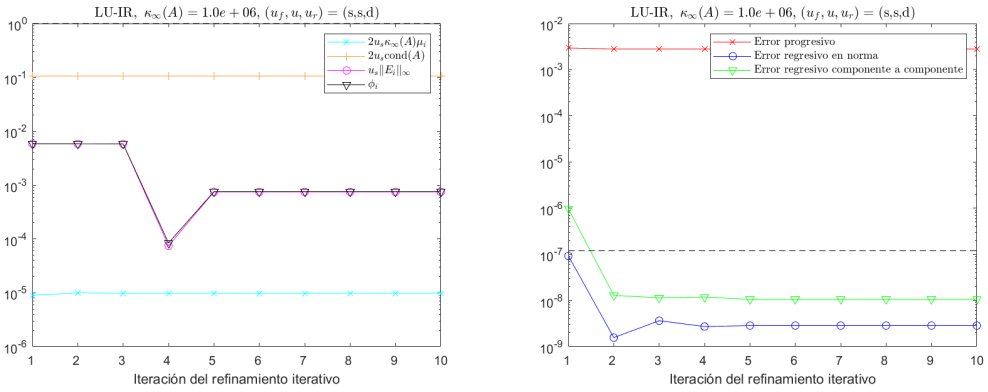

Figura 5.3: Gráfica con cotas (izquierda) y gráfica de errores (derecha) del refinamiento iterativo con precisiones $(u_f, u, u_r) = (s, s, d)$ y $\kappa_\infty(A) = 10^6$.

5.2.2. Refinamiento con precisión fija

Si tomamos $u_f = u = u_r$, tenemos refinamiento iterativo con precisión fija, y ϕ_i permanece sin cambios con respecto al caso anterior. La diferencia es que la precisión límite ahora es [2]

$$4pu\,\mathrm{cond}(A, \mathbf{x}) + u \approx 4pu\,\mathrm{cond}(A, \mathbf{x}). \tag{5.2}$$

puesto que $u_r = u$.

El beneficio del refinamiento iterativo con precisión fija para el error directo es que proporciona una precisión límite del orden de $\mathrm{cond}(A, \mathbf{x})u$ en lugar de $\kappa_\infty(A)u$ que es la precisión de la solución calculada original, y esto es independiente de cualquier inestabilidad en la factorización siempre que $\phi_i \ll 1$.

Para ilustrar este resultado comenzaremos con el caso en el que todos los cálculos se hacen en precisión simple, es decir, $(u_f, u, u_r) = (s, s, s)$ y $\kappa_\infty(A) = 10^2$. Observamos en la gráfica izquierda de la Figura 5.4 que en este caso ϕ_i es del orden de 10^{-6}. Además, se ha calculado que $\mathrm{cond}(A, \mathbf{x}) \approx 6{,}4 \times 10^{-3}$ y u es del orden de 10^{-7} porque estamos trabajando con precisión simple. Por tanto, la precisión límite será del orden de 10^{-7}. Si observamos la gráfica derecha de la Figura 5.4 vemos que solo llegamos a valores para el valor del error regresivo en norma del orden de 10^{-8}.

Se continua comentando el caso en el que todos los cálculos se hacen en precisión doble, es decir, $(u_f, u, u_r) = (d, d, d)$ y se trabaja con una matriz con número de condición en norma infinito de $\kappa_\infty(A) = 10^8$. En primer lugar, observamos en la gráfica izquierda de la Figura 5.5 que en este caso obtenemos valores de ϕ_i

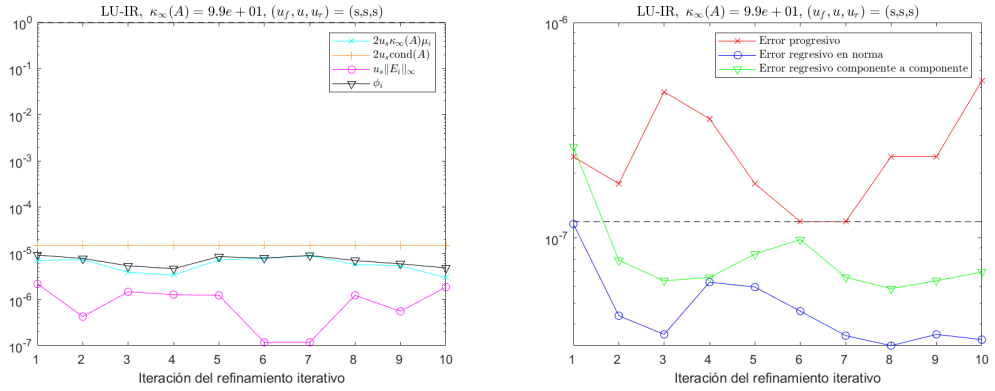

Figura 5.4: Gráfica con cotas (izquierda) y gráfica de errores (derecha) del refinamiento iterativo con precisiones $(u_f, u, u_r) = (s, s, s)$ y $\kappa_\infty(A) = 10^2$.

notablemente inferiores a los obtenidos en los experimentos anteriores, siendo del orden de 10^{-10}. Estos valores tan bajos a priori indicarían una convergencia muy rápida. Se ha calculado que en este caso, $\mathrm{cond}(A, \mathbf{x}) \approx 3\times10^7$ y u es aproximadamente 2×10^{-16}, pues estamos trabajando con precisión doble. Según (5.2) la precisión límite debería ser del orden de $\mathrm{cond}(A, \mathbf{x})u \approx 6\times6\times10^{-9}$. La gráfica derecha de la Figura 5.5 proporciona información sobre los errores y vemos que efectivamente los valores del error progresivo son aproximadamente del orden de 10^{-9} lo que encaja perfectamente con el resultado teórico proporcionado en esta sección.

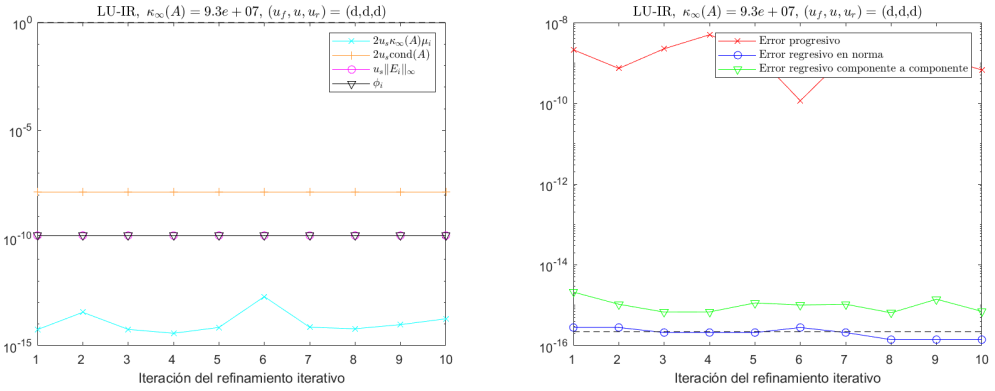

Figura 5.5: Gráfica con cotas (izquierda) y gráfica de errores (derecha) del refinamiento iterativo con precisiones $(u_f, u, u_r) = (d, d, d)$ y $\kappa_\infty(A) = 10^8$.

Cabe destacar que el refinamiento iterativo con precisión fija doble ha proporcionado errores muy inferiores al refinamiento iterativo con precisión fija simple sobre todo teniendo en cuenta que en el experimento con precisión doble se

ha utilizado una matriz con número de condición en norma infinito $\kappa_\infty(A) = 10^8$ mientras que para el experimento con precisión simple se ha trabajado con una matriz con número de condición en norma infinito aproximado $\kappa_\infty(A) = 10^2$, es decir, hacer todos los cálculos en precisión doble proporciona resultados más precisos que hacerlos con precisión simple, incluso trabajando con matrices mal acondicionadas.
Además, en ambos experimentos utilizando precisión fija, observamos que el refinamiento iterativo no ha reducido significativamente los errores.

5.2.3. Refinamiento de precisión mixta con soluciones en menor precisión

El tercer escenario de interés es aquel en el que calculamos la factorización LU y realizamos las sustituciones progresiva y regresiva con una precisión menor que la precisión de trabajo. Consideramos tres casos particulares, que generan nuevos resultados.

- *Caso 1:* $u = u_r = u_f^2$. En este caso, la convergencia está asegurada si [2]

$$\phi_i = 3n\||A^{-1}||\widehat{L}||\widehat{U}|\|_\infty u^{1/2} < 1 \tag{5.3}$$

y, suponiendo que esta condición se cumple, la precisión límite es [2],

$$4pu\,\mathrm{cond}(A,\mathbf{x}) + u \leq u^{1/2}\cdot 4pu^{1/2}\,\mathrm{cond}(A) \lesssim \frac{4}{3}\phi_i u^{1/2} < \frac{4}{3}u^{1/2}. \tag{5.4}$$

Comparado con el refinamiento en precisión fija, tenemos un requisito de convergencia más estricto y la misma precisión límite, pero ahora hay un ahorro computacional significativo.

Los errores regresivos en norma y por componentes permanecen del orden de u bajo las suposiciones de que A no está demasiado mal acondicionada o de que la factorización no es demasiado inestable.

Comenzamos hablando sobre el caso en el que $(u_f, u, u_r) = (h, s, s)$ para ilustrar estos resultados. En la primera gráfica de la Figura 5.6 observamos que ϕ_i es del orden de 10^{-1} y por tanto es menor que 1 para todos los valores de i y efectivamente en la segunda gráfica de la Figura 5.6 observamos que el refinamiento iterativo converge, aunque son necesarias 6 iteraciones para alcanzar la precisión límite esperada. De hecho, como hemos comentado anteriormente el error relativo en norma es del orden

de u pues en este caso u es la unidad de redondeo para la precisión simple que, como ya se ha indicado anteriormente, es del orden de 10^{-7}.

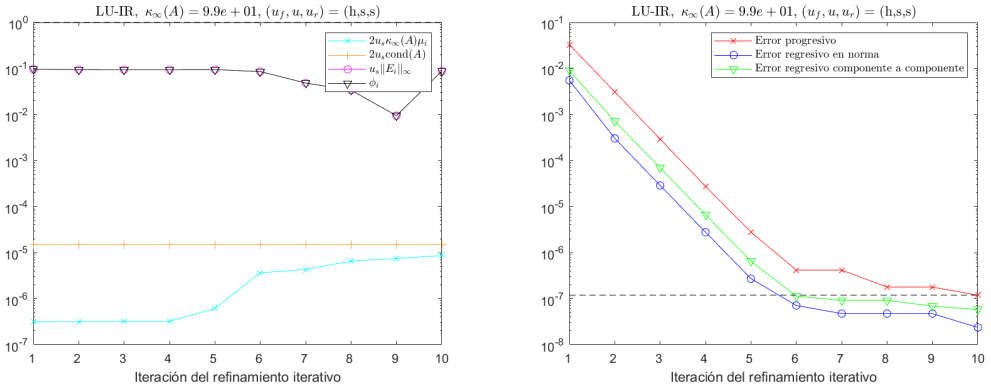

Figura 5.6: Gráfica con cotas (izquierda) y gráfica de errores (derecha) del refinamiento iterativo con precisiones $(u_f, u, u_r) = (h, s, s)$ y $\kappa_\infty(A) = 10^2$.

A continuación, consideramos el caso en el que $(u_f, u, u_r) = (s, d, d)$ y trabajaremos con una matriz con número de condición en norma infinito $\kappa_\infty = 10^2$ y otra con $\kappa_\infty = 10^4$. En la primera gráfica de las Figuras 5.7 y 5.8 observamos que ϕ_i es del orden de 10^{-6} cuando $\kappa_\infty = 10^2$ y del orden de 10^{-5} cuando $\kappa_\infty = 10^4$ y, por tanto, en ambos casos es bastante menor que 1 para todos los valores de i.

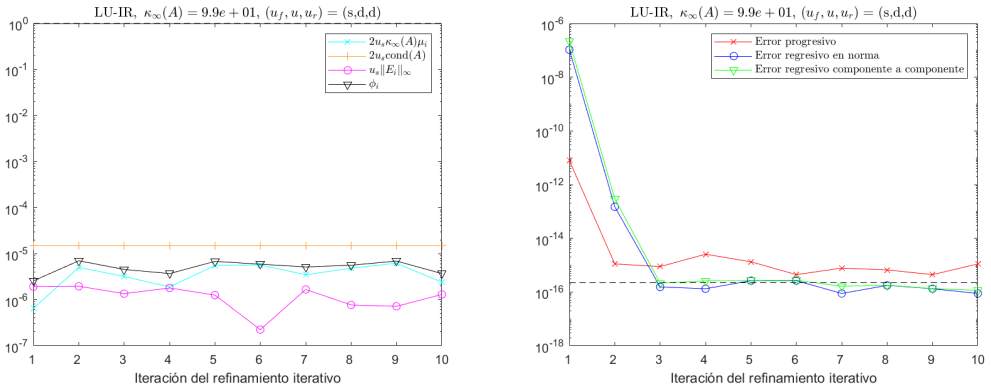

Figura 5.7: Gráfica con cotas (izquierda) y gráfica de errores (derecha) del refinamiento iterativo con precisiones $(u_f, u, u_r) = (s, d, d)$ y $\kappa_\infty(A) = 10^2$.

Obsérvese que estos valores para ϕ_i son notablemente inferiores que los obtenidos anteriormente trabajando con $(u_f, u, u_r) = (h, s, s)$, lo que pro-

porciona una convergencia más rápida según el Corolario 3.2.1, y es evidente al comparar la gráfica de la derecha de las Figuras 5.7 con la de la Figura 5.8. Además, para ambos números de condición volvemos a obtener errores relativos en norma del orden de u pues en este caso, u es la unidad de redondeo para la precisión doble que es del orden de 10^{-16}. Cuando el número de condición de la matriz es 10^4 observamos, sin embargo, que el error progresivo es bastante mayor que los errores regresivos y que se mantiene por encima de 10^{-13}.

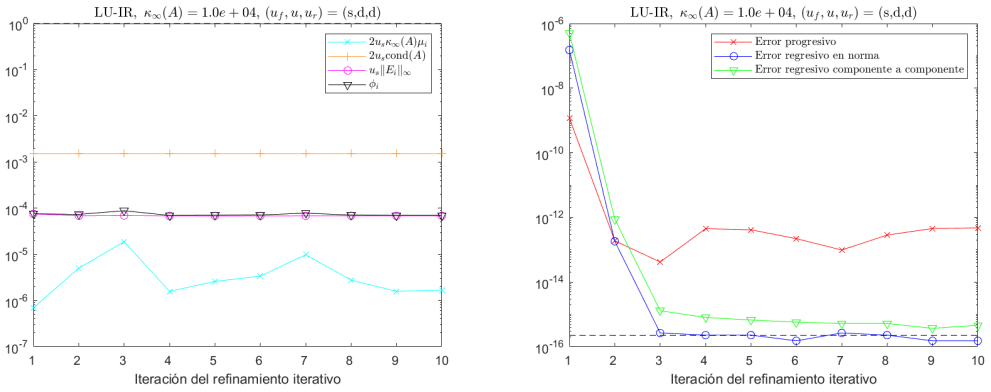

Figura 5.8: Gráfica con cotas (izquierda) y gráfica de errores (derecha) del refinamiento iterativo con precisiones $(u_f, u, u_r) = (s, d, d)$ y $\kappa_\infty(A) = 10^4$.

- *Caso 2*: $u_r = u^2$, $u = u_f^2$. En este caso, si se logra la convergencia el error relativo en norma viene dado por [2]

$$4pu^2\text{cond}(A, \mathbf{x}) + u \leq u^{3/2}4p\text{cond}(A, \mathbf{x})u^{1/2} + u \leq \frac{4}{3}u^{3/2} + u \approx u.$$

Ahora logramos la precisión total u, aunque solo para problemas con $\kappa_\infty(A)$ no mayores que $u^{-1/2}$. No obstante, esto es una ganancia significativa sobre el Caso 1, a cambio de calcular unos pocos residuos con precisión u^2. La cota del error regresivo es del orden de u, como en el caso anterior.

Si se observa la gráfica derecha de la Figura 5.9 se aprecia que, como se había anunciado en el resultado teórico, los errores regresivos son del orden de u pues en este caso u es la unidad de redondeo para la precisión simple, que es del orden de 10^{-7}. Además, se observa que esta gráfica es muy similar a la obtenida en la gráfica derecha de la Figura 5.6 en la que se trabajaba con precisiones $(u_f, u, u_r) = (h, s, s)$ y el mismo número de condición en norma infinito $\kappa_\infty(A) = 10^2$, pero sin recurrir a la precisión doble para el cálculo de los residuos.

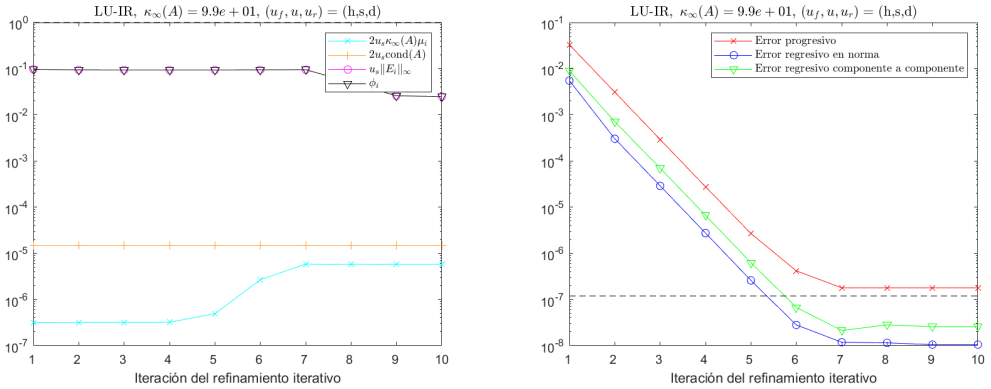

Figura 5.9: Gráfica con cotas (izquierda) y gráfica de errores (derecha) del refinamiento iterativo con precisiones $(u_f, u, u_r) = (h, s, d)$ y $\kappa_\infty(A) = 10^2$.

- *Caso 3*: $u = u_r = u_f^4$. En este caso más extremo, la factorización se realiza a un cuarto de la precisión de trabajo. La condición de convergencia es [2],

$$\phi_i = 3n\|A^{-1}\|\|\widehat{L}\|\|\widehat{U}\|_\infty u^{1/4} < 1,$$

y la precisión límite es

$$4pu\text{cond}(A, \mathbf{x}) + u \leq u^{3/4}, 4pu^{1/4}\text{cond}(A) + u \leq \frac{4}{3}\phi_i u^{3/4} + u \lesssim u^{3/4}.$$

De nuevo, la cota del error regresivo es todavía del orden de u.

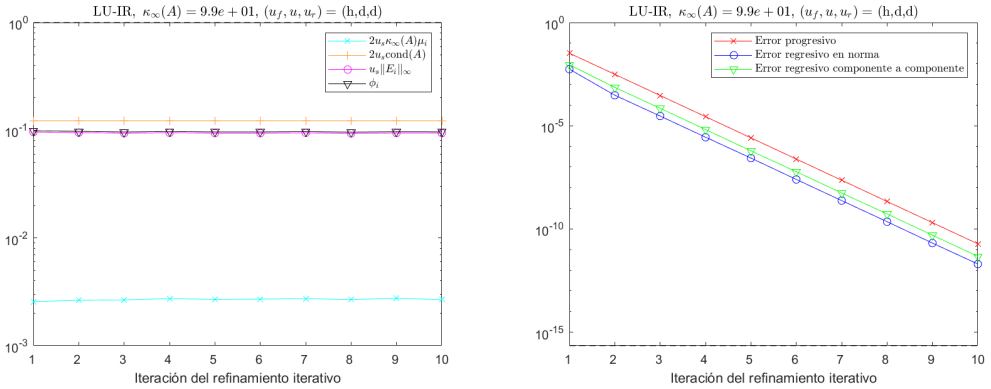

Figura 5.10: Gráfica con cotas (izquierda) y gráfica de errores (derecha) del refinamiento iterativo con precisiones $(u_f, u, u_r) = (h, d, d)$ y $\kappa_\infty(A) = 10^2$.

Para ilustrar este caso consideramos $(u_f, u, u_r) = (h, d, d)$ y $\kappa_\infty(A) = 10^2$. Como se puede apreciar en la gráfica derecha de la Figura 5.10, el valor

de los errores regresivos en la décima y última iteración es del orden de 10^{-12}. Según el resultado teórico la cota del error regresivo es u, es decir, aproximadamente 2×10^{-16} y, por lo tanto, 10 iteraciones no han sido suficientes para conseguir la precisión esperada.

5.2.4. Comparativa

En esta sección se comparan los resultados de las distintas configuraciones de precisión utilizadas en los experimentos de las subsecciones anteriores. Se observa que, en general, todas las combinaciones válidas consiguen reducir el error, pero algunas lo hacen de forma más eficiente que otras.

En particular, cuando se usa mayor precisión en la resolución del sistema (u) y en el cálculo de residuos (u_r), como en los casos (s, d, d) o (d, d, d), la convergencia suele ser más rápida y se alcanzan errores finales mucho más pequeños. Esto se nota en el valor del factor ϕ_i, que es más bajo y permite llegar antes al límite teórico de precisión.

Por otro lado, configuraciones computacionalmente más baratas, como la utilizada en la Figura 5.6 (h, s, s), también funcionan bien en algunos casos, especialmente si el número de condición de la matriz no es muy alto. En estos casos, aunque la precisión final es menor (por ejemplo, errores del orden de $u = 10^{-7}$), el coste computacional es mucho menor, lo que puede ser interesante en ciertas aplicaciones.

Sin embargo, observamos que aunque (h, s, s) y la configuración tradicional (s, s, d) alcanzan errores del mismo orden, el refinamiento tradicional lo hace de forma más rápida.

También se ha visto que en las configuraciones con precisión fija, como (s, s, s) o (d, d, d), el refinamiento no siempre consigue mejorar notablemente el error, y ello depende principalmente de cómo de bien acondicionada esté la matriz.

En resumen, dependiendo de cuánta precisión se necesite en el resultado final, puede ser mejor usar una configuración u otra:

- Para obtener una solución muy precisa, es mejor usar (s, d, d) o (d, d, d), aunque sean más costosos.
- Si se busca un equilibrio entre coste y precisión, una opción como (h, s, s) puede ser suficiente aunque (s, s, d) proporcionará errores del tamaño de la cota teórica en menos iteraciones, lo cual puede suponer menos coste total aunque el coste por iteración sea mayor.
- Si la precisión no es tan importante, configuraciones como (s, s, s) pueden valer, sabiendo que el error final será mayor.

Estas observaciones pueden servir como guía a la hora de elegir qué configuración de precisiones usar para el refinamiento iterativo según el problema y los recursos disponibles.

Además de comparar los distintos esquemas de refinamiento por la precisión final alcanzada, resulta interesante estudiar su robustez frente al número de condición $\kappa_\infty(A)$ de la matriz. Como se mostró en la Sección 5.1, el coste en tiempo de CPU depende fuertemente de $\kappa_\infty(A)$ especialmente para dimensiones altas de la matriz.

Rango de $\kappa_\infty(A)$	Configuraciones recomendadas
$\kappa_\infty(A) \leq 10^2$	(h,s,d), (h,h,s), (h,s,s), (s,s,s)
$10^2 < \kappa_\infty(A) \leq 10^4$	(s,d,d), (s,s,d)
$10^4 < \kappa_\infty(A) \leq 10^6$	(s,s,d)
$\kappa_\infty(A) > 10^6$	(d,d,d)

Tabla 5.2: Tabla que muestra las configuraciones de precisiones más recomendadas según el número de condición de la matriz.

Basándonos en los experimentos planteados, para cada configuración de precisiones se puede identificar un rango práctico del valor del número de condición dentro del cual el refinamiento converge en menos de diez iteraciones y alcanza la cota teórica indicada en la subsección correspondiente. Para ilustrar esto se ha creado la Tabla 5.2 que nos muestra recomendaciones generales sobre que configuración de precisiones elegir si conocemos aproximadamente el número de condición de la matriz con la que vamos a trabajar.

Bibliografía

[1] AMD. Single precision vs. double precision: Main differences. AMD, 2024. Último acceso: 20 de noviembre de 2024.

[2] Erin Carson and Nicholas J. Higham. Accelerating the solution of linear systems by iterative refinement in three precisions. *SIAM Journal on Scientific Computing*, 40(2):A817–A847, 2018.

[3] Massimiliano Fasi and Nicholas J. Higham. Matrices with tunable infinity-norm condition number and no need for pivoting in LU factorization. *SIAM Journal on Matrix Analysis and Applications*, 42:417–435, 2021.

[4] Gene H Golub and Charles F Van Loan. *Matrix computations*. Johns Hopkins Univerity Press, 2013.

[5] Furkan Gözükara. What is the difference between fp16 and bf16? Here a good explanation for you, 2024. Último acceso: 20 de noviembre de 2024.

[6] Nicholas J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[7] Nicholas J. Higham and Theo Mary. Mixed precision algorithms in numerical linear algebra. *Acta Numerica*, 31:347–414, 2022.

[8] Nicholas J. Higham and Srikara Pranesh. Chop - MATLAB code for rounding matrix elements to lower precision. https://www.github.com/higham/chop. GitHub repository.

[9] Andrew N. Jackson. Demonstration of the Ill-Conditioned Nature of Hilbert Matrices. Technical report, University of Edinburgh, 1996.

[10] MathWorks. What Is Half Precision? - MATLAB & Simulink - MathWorks España, 2024. Último acceso: 20 de noviembre de 2024.

[11] Microsoft Corporation. Representación de punto flotante de IEEE — Microsoft Learn, 2024. Último acceso: 13 de mayo de 2025.

Apéndice A

Códigos en MATLAB utilizados para crear las gráficas de los experimentos numéricos

A continuación, se presentan los códigos con los que se han generado las gráficas del Capítulo 5 referente a los experimentos númericos.

```
function [x, iter, residuos, error_rel, error_abs,
   error_exacto_rel, error_exacto_abs,tiempoCPU] =
   refinamiento_iterativo_con_tres_precisiones_familia(
   alfa, beta, n, sol, b, uf, u,ur,us, maxIter)
    %Esta funcion tiene como objetivo hacer el
       refinamiento iterativo para
    %un sistema lineal de la forma Ax=b utilizando 3
       precisiones diferentes
    %gracias a la funcion chop de Matlab, siendo A una
       matriz de la forma
    %A=LU=T(alfa)'T(beta).
    %
    % Entradas:
    %   alfa: parametro para construir la  matriz L
    %   beta: parametro para construir la  matriz U
    %   n: dimension del sistema
    %   sol: solucion exacta del sistema (para despues
       calcular los errores exactos)
    %   b: Vector de terminos independientes.
    %   ur: Precision para el calculo del residuo (paso 1)
       .
    %   uf: Precision para la resolucion del sistema Ad=r
       (paso 2).
```

```
15      %   u: Precision para la actualizacion de la solucion
           (calculo de d)(paso 3).
16      %   us: precision con la que se resuelve la ecuacion
           de correccion
17      %   maxIter: Maximo numero de iteraciones permitidas.
18      %
19      % Salidas:
20      %   x: Matriz con las aproximaciones a la solucion en
           cada iteracion.
21      %   iter: Numero de iteraciones realizadas.
22      %   residuos: Vector con los residuales en cada
           iteracion.
23      %   error_rel: error relativo
24      %   error_abs: error absoluto
25      %   error_exacto_rel: error relativo respecto a la
           solucion exacta
26      %   error_exacto_abs: error absoluto respecto a la
           solucion exacta
27      %   tiempoCPU: tiempo que tarda la CPU en realizar el
           refinamiento iterativo
28
29
30      % Inicializamos las variables:
31
32      % Contador de iteraciones
33      iter = 0;
34      x=zeros(n,maxIter);
35      error_rel=zeros(maxIter);
36      error_abs=zeros(maxIter);
37      error_exacto_rel=zeros(maxIter);
38      error_exacto_abs=zeros(maxIter);
39      errrc=zeros(maxIter,1);
40      temp=zeros(maxIter,1);
41
42      % Configuramos chop a precision doble para calcular la
            factorizacion LU
43      % L=generar_matriz_T(alfa,n)' ;
44      % U=generar_matriz_T(beta,n) ;
45      options.format=uf;
46      options.round=1;
47      chop([], options);
48
49      A=generarAfam(alfa,beta,n,u);
50      [L,U]= lutx_chop(A,uf);
```

```
51      x(:,1)=resolver_sistemaLU_fam(L, U, b, us);
52
53      options.format=u;
54      options.round=1;
55      chop([], options);
56      x(:,1)=chop(x(:,1));
57
58      %Calcular el numero de condicion de A, cond(A) y cond(
           A,x) para la
59      %solucion exacta
60      kA = cond(A,'inf')
61      condAx = norm(abs(inv(A))*abs(A)*abs(sol),'inf')/norm(
           sol,'inf');
62      condA = norm(abs(inv(A))*abs(A),'inf');
63
64      tic
65      %Comenzamos con las iteraciones del refinamiento
           iterativo
66      while iter < maxIter
67
68          % Paso 1: Calculamos el residuo r = b - A*x con la
               precision1
69          % Configurar chop a precision1
70          options.format=ur;
71          options.round=1;
72          chop([], options);
73
74          r=chop(b-multiplicarLUfam(alfa, beta, x(:,iter+1),
               ur));
75          residuos(iter + 1) = norm(r, inf);
76
77          % Paso 2: Resolvemos el sistema Ad = r con la
               precision2
78          % Configurar chop a precision2
79          options.format=us;
80          options.round=1;
81          chop([], options);
82          d=resolver_sistemaLU_fam(L,U,r/residuos(iter+1),us
               );
83
84          % Paso 3: Actualizamos la solucion y = x + d con
               la precision3
85          % Configurar chop a precision3
86          options.format=u;
```

```
87          options.round=1;
88          chop([], options);
89          x(:,iter+2) = chop(x(:,iter+1) + residuos(iter+1)*
               d);
90
91          %Calculamos los errores pra la figura 3
92          normx=norm(x(:,iter+2),inf);
93          error_exacto_abs(iter+1)= norm((x(:,iter+2)-sol),
               inf);
94          error_exacto_rel(iter+1)= error_exacto_abs(iter+1)
               /norm(sol,'inf'); %mp
95          error_abs(iter+1)= norm((x(:,iter+2)-x(:,iter+1)),
                inf);
96          error_rel(iter+1)= error_abs(iter+1)/normx;
97
98          options.format='d';
99          options.round=1;
100         chop([], options);
101
102         %Calcular el error progresivo (errp), mu_i(que la
               utilizaremos
103         %despues para calcular lim, que es la primera cota
                que calculamos para la segunda figura),
104         % despues calculamos el error regresivo en norma y
                finalmente el
105         % error regresivo componente a componente
106         % CALCULAR LO QUE DIBUJAREMOS EN LA FIGURA 1
107         errp(iter+1) = chop(norm(x(:,iter+2)-sol,'inf')/
               norm(sol,'inf'));
108         mu(iter+1) = chop(norm(chop(A*chop(x(:,iter+2)-sol
               )),'inf')/chop(norm(A,'inf')*norm(chop(x(:,iter
               +2)-sol),'inf')));
109         errrn(iter+1) = chop(norm(r,'inf')./chop(chop(norm
               (A,'inf')*norm(x(:,iter+2),'inf'))+ norm(b,'inf
               ')));
110         temp = chop(abs(r) ./ chop(chop(abs(A)*abs(x(:,
               iter+2))) + abs(b))) ;
111         temp(isnan(temp)) = 0; % Hace que 0/0 sea 0.
112         errrc(iter+1) = max(temp);
113
114         % Incrementamos el contador de iteraciones
115         iter = iter + 1;
116
117         %CALCULAR LO QUE DIBUJAREMOS EN LA FIGURA 2
```

```
118         lim(iter) = chop(chop(chop(2*calcularu(us))*kA)*mu
                (iter));
119         lim2(iter) = chop(chop(2*calcularu(us))*condA);
120         dact = chop(A\chop(r./norm(r,'inf')));
121         etai(iter) = chop(norm(chop(d-dact),'inf')./norm(
                dact,'inf'));
122         phi(iter) = min(lim(iter),lim2(iter))+etai(iter);
123
124     end
125     tiempoCPU=toc;
126
127     residuos=residuos(1:length(residuos)-1);
128
129
130     %ajustamos los vectores de errores
131     error_rel=error_rel(1:iter);
132     error_abs=error_abs(1:iter);
133     error_exacto_rel=error_exacto_rel(1:iter);
134     error_exacto_abs=error_exacto_abs(1:iter);
135
136
137     %en caso de que alguno de los valores sea 0 para la
            maquina (aunque en
138     %realidad no sea 0) se le da el valor de la precision
            de trabajo para
139     %que se dibuje en la grafica
140
141     errrc=errrc(1:length(errp));
142     errp(errp == 0) = calcularu(u);
143     errrn(errrn == 0) = calcularu(u);
144     errrc(errrc == 0) = calcularu(u);
145     errp(~isfinite(errp)) = calcularu(u);
146     errrn(~isfinite(errrn)) = calcularu(u);
147     errrc(~isfinite(errrc)) = calcularu(u);
148     lim(~isfinite(lim)) = calcularu(u);
149     lim2(~isfinite(lim2)) = calcularu(u);
150     etai(~isfinite(etai)) = calcularu(u);
151     phi(~isfinite(phi)) = calcularu(u);
152
153
154     % PRIMERA GRAFICA
155     %Dibujamos la grafica con los errores progresivos y
            regresivos en norma y componente a componente resp.
             errp, errrn, errrc
```

```
156     fig1 = figure();
157     semilogy(1:iter, errp, '-rx');
158     hold on
159     semilogy(1:iter, errrn, '-bo');
160     hold on
161     semilogy(1:iter, errrc, '-gv');
162     hold on
163     semilogy(1:iter, calcularu(u)*ones(iter,1), '--k');
164
165
166     atm = get(gca,'xticklabels');
167     m = str2double(atm);
168     xlab = [];
169     num = 1;
170     for i = 1:numel(m)
171         if ceil(m(i)) == m(i)
172             xlab(num) = m(i);
173             num = num + 1;
174         end
175     end
176     set(gca,'xticklabels',xlab);
177     set(gca,'xtick',xlab);
178     xlabel({'Iteracion del refinamiento iterativo'},'
        Interpreter','latex');
179
180     str_e = sprintf('%0.1e',kA);
181     tt = strcat('LU-IR,  $$\, \kappa_{\infty}(A) = ',str_e
        ,', \, (u_f,u,u_r) = $$ (',uf,',',u,',',ur,')');
182     title(tt,'Interpreter','latex');
183
184     h = legend('Error progresivo','Error regresivo en
        norma','Error regresivo componente a componente');
185     set(h,'Interpreter','latex');
186
187
188
189
190
191     %SEGUNDA GRAFICA
192
193     lim(lim==0)=eps;
194     lim2(lim2==0)=eps;
195     etai(etai==0)=eps;
196     phi(phi==0)=eps;
```

```
197
198
199     fig2 = figure();
200     semilogy(1:iter, lim, '-cx');
201     hold on
202     semilogy(1:iter, lim2, '-+','Color',[1
           0.600000023841858 0.200000002980232]);
203     hold on
204     semilogy(1:iter, etai, '-mo');
205     hold on
206     semilogy(1:iter, phi, '-kv');
207     hold on
208     semilogy(1:iter, ones(iter,1)', '--k');
209
210
211     set(gca,'xticklabels',xlab);
212     set(gca,'xtick',xlab);
213     xlabel({'Iteracion del refinamiento iterativo'},'
           Interpreter','latex');
214
215     title(tt,'Interpreter','latex');
216
217     h = legend('$2u_s \kappa_{\infty}(A)\mu_i$','
           $2u_s$cond$(A)$', '$u_s \Vert E_i \Vert_\infty$','$
           \phi_i$');
218     set(h,'Interpreter','latex');
219
220 end
```

Código A.1: refinamiento_iterativo_con_tres_precisiones_familia.m

```
1  function u = calcularu(precision)
2      switch precision
3          case 'h'
4              u = 2^(-10);
5          case 'b'
6              u = 2^(-7);
7          case 's'
8              u = eps('single');
9          case 'd'
10             u = eps('double');
11         otherwise
12             error('La precisi n introducida no es v lida
                   .');
```

```
    end
end
```

Código A.2: calcularu.m

```
function A = generarAfam(alfa, beta,n, precision)
% CREARMATRIZCHOP Crea la matriz A de tama o n x n con:
%    a_{ij} = -alfa + (j-1)*alfa*beta, si i > j
%    a_{ij} = 1 + (i-1)*alfa*beta,     si i = j
%    a_{ij} = -beta + (i-1)*alfa*beta, si i < j
% aplicando chop en cada operaci n aritm tica que
   involucre
% a alfa y beta.

    options.format=precision;
    options.round=1;
    chop([], options);

    % Convertimos alfa y beta a la precisi n dada
    alfa = chop(alfa);
    beta = chop(beta);

    % Inicializa la matriz A
    A = zeros(n);

    for i = 1:n
        for j = 1:n
            if i > j
                % a_{ij} = -alfa + (j-1)*alfa*beta
                % (j-1) no necesita chop porque (j-1) es
                   un entero exacto
                product = chop(alfa * beta);      % alfa*
                   beta
                product = chop((j-1) * product);  % (j-1)*
                   alfa*beta
                A(i,j) = chop(-alfa + product);   % -alfa
                   + ...

            elseif i < j
                % a_{ij} = -beta + (i-1)*alfa*beta
                product = chop(alfa * beta);      % alfa*
                   beta
                product = chop((i-1) * product);  % (i-1)*
                   alfa*beta
```

```
33                  A(i,j) = chop(-beta + product);    % -beta
                        + ...
34
35              else
36                  % a_{ij} = 1 + (i-1)*alfa*beta
37                  product = chop(alfa * beta);       % alfa*
                        beta
38                  product = chop((i-1) * product);   % (i-1)*
                        alfa*beta
39                  A(i,j) = chop(1 + product);        % 1 +
                        ...
40              end
41          end
42      end
43  end
```

Código A.3: generarAfam.m

```
1   function y = multiplicarLUfam(alfa, beta, v, precision)
2   % Multiplica dos matrices L=T(alfa)'*U=T(beta) por un
        vector v en la
3   % precision indicada
4   %   Detailed explanation goes here
5       n=length(v);
6       w=zeros(n,1);
7       y=zeros(n,1); %mp
8
9       options.format=precision;
10      options.round=1;
11      chop([], options);
12
13      %Calculamos el producto de U*v
14      for i=1:n-1
15
16          w(i) = chop(v(i)- chop(beta*chop(sum(v(i+1:n)))));
17      end
18      w(n)=v(n);
19
20      %Calculamos el producto de L*w=LUv
21      y(1)=w(1);
22      for i=2:n
23          y(i)= chop(w(i)- chop(alfa*chop(sum(w(1:i-1)))));
24
25      end
```

```
end
```

Código A.4: multiplicarLUfam.m

```
function[]= numerocondicionparamatricesdehilbert(n)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
Numero_de_condicion=zeros(n,1);
for m=1:n
    H=hilb(m)
    Numero_de_condicion(m,1)=cond(H)
end

% Graficamos los resultados obtenidos del numero de
   condicion de las
% matrices de hilbert

    x=1:n;
    figure;
    plot(x, Numero_de_condicion, '-o', 'LineWidth', 2 );
    xlabel('Dimensi n de la matriz');
    ylabel('N mero de condici n');

    title('N mero de condici n de matrices de Hilbert de
        diferentes dimensiones');
    grid on

    Dimension= x';
    table(Dimension, Numero_de_condicion)
end
```

Código A.5: numerocondicionparamatricesdehilbert.m

```
function x = resolver_sistemaLU_fam(L,U,b, precision)

    options.format=precision;
    options.round=1;
    chop([], options);

    L = chop(L);
    U = chop(U);
    b = chop(b);

    % Resolver Ly = b
    n = length(b);
```

```
    y = zeros(n, 1);
    y(1)=b(1);
    for i = 2:n
        y(i) = chop(b(i) - chop(L(i, 1:i-1) * y(1:i-1)));
    end

    % Resolver Ux = y
    x = zeros(n, 1);
    x(n)=y(n)/U(n,n);
    for i = n-1:-1:1
        x(i) = chop((y(i) - chop(U(i, i+1:n) * x(i+1:n)))
           / U(i, i));
    end
end
```

Código A.6: resolver_sistemaLU_fam.m

```
% Definir las precisiones
precisiones = {'h', 'b', 's', 'd'};

% Calcular los valores de u uno por uno
u_h = calcularu('h');
u_b = calcularu('b');
u_s = calcularu('s');
u_d = calcularu('d');

% Crear un array con los valores
valores_u = [u_h; u_b; u_s; u_d];

% Crear la tabla
tabla_u = table(precisiones', valores_u, 'VariableNames',
   {'Precision', 'UnidadRedondeo'});

% Mostrar la tabla
disp(tabla_u);
```

Código A.7: tablaconu.m

```

function [x, iter, residuos] =
   refinamiento_iterativo_tres_precisiones(A, b,  uf, u,ur
   ,us, tol, maxIter)
    %Esta funci n tiene como objetivo hacer el
       refinamiento iterativo para
```

```
4       %un sistema lineal de la forma Ax=b utilizando 3
            precisiones diferentes gracias a la funci n chop
            de Matlab
5       %
6       % Entradas:
7       %   A: Matriz del sistema
8       %   b: Vector de t rminos independientes.
9       %   tol: Tolerancia para la norma infinito del residuo
            .
10      %   maxIter: M ximo n mero de iteraciones permitidas
            .
11      %
12      % Salidas:
13      %   x: Aproximaci n final a la soluci n.
14      %   iter: N mero de iteraciones realizadas.
15      %   residuos: Vector con los residuales en cada
            iteraci n.
16
17
18      % Inicializamos las variables:
19
20      % Soluci n inicial
21      x = zeros(size(b));
22      % Contador de iteraciones
23      iter = 0;
24
25      % Configuramos chop a precisi n doble para calcular
            la factorizaci n LU
26      % de forma precisa
27      options.format=uf;
28      options.round=1;
29      chop([], options);
30      [L, U, P] = lu(A);
31
32
33      tic
34      %Comenzamos con las iteraciones del refinamiento
            iterativo
35      while iter < maxIter
36          % Paso 1: Calculamos el residuo r = b - A*x con la
                precisi n1
37          % Configurar chop a ur
38          options.format=ur;
39          options.round=1;
```

```
40          chop([], options);
41          r = chop(b - A * x);
42          residuos(iter + 1) = norm(r, inf);
43
44          % Verificamos si la norma infinito del residuo es
               menor que la
45          % tolerancia, si lo es, acabamos con el
               refinamiento iterativo
46          if norm(r, inf) <= tol
47              break;
48          end
49
50          % Paso 2: Resolvemos el sistema Ad = r con la
               precision2
51          % Configurar chop a  us
52          options.format=us;
53          options.round=1;
54          chop([], options);
55          d = chop(U \ (L \ (P * r)));
56
57          % Paso 3: Actualizamos la soluci n y = x + d con
               la u
58          % Configurar chop a u
59          options.format=u;
60          options.round=1;
61          chop([], options);
62          x = chop(x + d);
63
64          % Incrementamos el contador de iteraciones
65          iter = iter + 1;
66      end
67
68      if iter == maxIter && norm(r, inf) > tol
69          warning('El refinamiento iterativo no convergi
               despu s de %d iteraciones.', maxIter);
70      end
71      tiempo=toc;
72  end
```

Código A.8: refinamiento_iterativo_tres_precisiones.m

```
1  function [tiemporand, tiempoort, tiempohilbert] =
       medir_tiempos_tamano(tamanomax,  uf, u,ur,us, tol,
       maxIter)
```

```
2   %Esta funci n tiene como objetivo medir el tiempo que
        se tarda en
3   %realizar el proceso de refinamiento iterativo con 3
        precisiones fijas
4   %y variando el tama o de las matrices, trabaja con
        matrices aleatorias, ortogonales y de hilbert,
        despues dibuja una gr fica que
5   %representa estos resultados
6   %
7   % Entradas:
8   %   tamanomax: se trabajar  con matrices de 500,
        500*2, ...,
9   %   500*tamanomax asique esta variable indica el
        tama o m ximo de las
10  %   matrices que se quieren analizar
11  %   tol: Tolerancia para la norma infinito del residuo
        .
12  %   maxIter: M ximo n mero de iteraciones permitidas
        .
13  %
14  % Salidas:
15  %   tiemporand: tiempo que se tarda en realizar el
        refinamiento iterativo con la
16  %               funci n
        refinamiento_iterativo_tres_precisiones utilizando
        matrices
17  %               aleatorias
18  %   tiempoort: tiempo que se tarda en realizar el
        refinamiento
19  %              iterativo con la funci n
        refinamiento_iterativo_tres_precisiones
20  %              utilizando matrices ortogonales
21  %   tiempohilbert: tiempo que se tarda en realizar el
        refinamiento
22  %                  iterativo con la funci n
        refinamiento_iterativo_tres_precisiones
23  %                  utilizando matrices de Hilbert
24
25  % Inicializamos las variables:
26
27  % Tiempos iniciales
28  tiemporand = zeros(1,tamanomax);
29  tiempoort = zeros(1,tamanomax);
30  tiempohilbert = zeros(1,tamanomax);
```

```
31      con1= zeros(1,tamanomax);
32      con2= zeros(1,tamanomax);
33      con3= zeros(1,tamanomax);
34
35      for n=1:tamanomax
36
37          % Definimos la matriz A y el vector independiente
                b
38          % aleatoriamente
39          A = rand(500*n,500*n);
40          b = rand(500*n,1);
41
42          % Comenzamos a medir el tiempo
43          tic
44          refinamiento_iterativo_tres_precisiones(A, b, uf,
                u,ur,us, tol, maxIter);
45          % Paramos el temporizador y guardamos el tiempo
46          tiemporand(1,n)=toc;
47          con1(1,n)=cond(A,"inf");
48
49          % Definimos la matriz A de forma que sea ortogonal
50          A = orth(rand(500*n,500*n));
51          con2(1,n)=cond(A,"inf");
52          % Comenzamos a medir el tiempo
53          tic
54          refinamiento_iterativo_tres_precisiones(A, b,  uf,
                u,ur,us, tol, maxIter);
55          % Paramos el temporizador y guardamos el tiempo
56          tiempoort(1,n)=toc;
57
58          % Definimos la matriz A de forma que sea de
                Hilbert
59          A = hilb(500*n);
60          con3(1,n)=cond(A,"inf");
61          % Comenzamos a medir el tiempo
62          tic
63          refinamiento_iterativo_tres_precisiones(A, b, uf,
                u,ur,us, tol, maxIter);
64          % Paramos el temporizador y guardamos el tiempo
65          tiempohilbert(1,n)=toc;
66
67
68      end
69
```

```
% Graficamos los resultados obtenidos del tiempo por cada
   dimensi n (vector x)
    x=(1:tamanomax)*500;

    figure;
    plot(x, tiemporand, '-o', 'LineWidth', 2);
    hold on;
    plot(x, tiempoort, '--s', 'LineWidth', 2);
    plot(x, tiempohilbert, ':d', 'LineWidth', 2);
    hold off;

    xlabel('Dimensi n de la matriz');
    ylabel('Tiempo');
    legend('Tiempo con matrices aleatorias', 'Tiempo con
       matrices ortogonales', 'Tiempo con matrices de
       Hilbert')
    title('Comparaci n del tiempo de CPU dependiendo de
       la dimensi n');
    grid on

   %tabla con los n meros de condicion de las matrices
   table(con1', con2', con3')

end
```

Código A.9: medir_tiempos_tamano.m