

Métodos Numéricos

Instituto de Matemática y Estadística “Rafael Laguardia” (IMERL)
Facultad de Ingeniería
Universidad de la República

Última actualización: 31 de julio de 2025.

Índice general

1. Aritmética de punto flotante y errores	9
1.1. Introducción. Aproximaciones y errores	9
1.1.1. Errores	11
1.2. Errores de aproximación	12
1.2.1. Sumas de Riemann	12
1.2.2. Desarrollos de Taylor	13
1.2.3. Calcular raíces cuadradas como un Herón	14
1.3. Errores	15
1.3.1. Errores absolutos y relativos	15
1.3.2. Funciones reales: número de condición	16
1.3.3. Propagación de errores en operaciones aritméticas	17
1.4. Aritmética de punto flotante	20
1.4.1. Representación de punto flotante	20
1.4.2. Constantes de máquina y números especiales	22
1.4.3. Error de representación en punto flotante	24
1.5. Propagación de errores en punto flotante	25
1.5.1. Advertencias	26
1.5.2. Análisis de error hacia adelante	26
1.5.3. Cálculo de derivadas usando diferencias hacia adelante	30
2. Sistemas de ecuaciones lineales	35
2.1. Introducción	35
2.2. Escalerización gaussiana	37
2.2.1. Sistemas triangulares	37

2.2.2. Escalerización gaussiana (sin pivoteo)	39
2.2.3. Escalerización gaussiana con pivoteo parcial	41
2.3. Descomposición <i>LU</i>	44
2.4. Matrices dispersas y de banda	48
2.5. Estabilidad y convergencia	50
2.5.1. Normas de vectores y matrices	50
2.5.2. Errores y residuos: un ejemplo	56
2.5.3. Número de condición	58
2.5.4. Análisis de perturbaciones	60
3. Interpolación polinomial	63
3.1. Introducción. ¿Por qué interpolar?	63
3.2. Polinomio interpolante	64
3.2.1. Forma de Vandermonde	66
3.2.2. Forma de Lagrange	68
3.2.3. Forma de Newton	71
3.3. Interpolación de funciones	73
3.3.1. Error de interpolación	74
3.3.2. Fenómeno de Runge	78
3.4. Interpolación a trozos	79
3.4.1. Interpolación lineal a trozos	81
3.4.2. Interpolación cúbica a trozos	84
4. Ecuaciones no lineales	93
4.1. Introducción	93
4.2. Método de bisección	94
4.3. Método de la regla falsa	96
4.4. Método de la secante	98
4.5. Método de Newton-Raphson	102
4.6. Resumen: una muy breve comparación	106
4.7. Métodos iterativos generales	107
4.7.1. Convergencia	108
4.7.2. Orden y velocidad de convergencia	111

4.7.3. Criterios de parada	112
4.8. Sistemas de ecuaciones	114
4.8.1. Métodos iterativos generales	114
4.8.2. Método de Newton-Raphson	116
4.8.3. Variantes al método de Newton-Raphson*	118
5. Mínimos cuadrados	121
5.1. Introducción. Modelos y ajustes	121
5.2. Ecuaciones normales	125
5.3. Descomposición QR	128
5.3.1. Aplicación de QR a mínimos cuadrados	130
5.3.2. Transformaciones de Householder	132
5.4. Mínimos cuadrados no lineales	139
6. Descomposición SVD	143
6.1. Aplicaciones de SVD	145
6.1.1. Computar rangos, espacios de columnas y núcleos de matrices	145
6.1.2. Aproximaciones de rango bajo	146
6.1.3. Análisis de componentes principales	147
6.1.4. SVD en sistemas de recomendación	150
6.2. Aplicación de SVD a mínimos cuadrados	155
6.2.1. Mínimos cuadrados de rango completo	155
6.2.2. Mínimos cuadrados lineales con rango deficiente	156
6.2.3. Comparación de métodos para la resolución de problemas de mínimos cuadrados	159
7. Integración numérica	161
7.1. Motivación y reglas de Newton-Cotes básicas	161
7.2. Reglas compuestas	167
7.3. Adaptatividad	170
7.4. Cuadratura gaussiana	173
8. Ecuaciones diferenciales ordinarias	179
8.1. Introducción. Nociones generales	179

8.1.1. Problemas de valores iniciales	182
8.2. Métodos de Euler	185
8.3. Elementos de los métodos numéricos aplicados a ecuaciones diferenciales ordinarias	188
8.3.1. Un poco de nomenclatura	188
8.3.2. Consistencia	189
8.3.3. Cero-estabilidad	191
8.3.4. Convergencia	192
8.3.5. Estabilidad absoluta	194
8.4. Método del trapecio	197
8.5. Métodos de Runge-Kutta	199
8.5.1. Construcción de métodos de Runge-Kutta	200
8.5.2. <i>El</i> método de Runge-Kutta	202
8.6. Esquemas predictores-correctores	204
8.7. <i>Solvers</i>	205
8.7.1. El algoritmo BS23 y <code>ode23</code>	209
8.8. Rígidez	211
A. Preliminares y repaso	215
A.1. Desarrollo de Taylor y resto de Lagrange	215
A.1.1. En una variable	215
A.1.2. En varias variables	217
A.2. Teoremas de valores medios	218
A.3. Valores y vectores propios	219
A.4. Ortogonalidad	220
A.5. Ecuaciones diferenciales ordinarias	223
A.5.1. Ecuaciones de primer orden de variables separables	223
A.5.2. Ecuaciones lineales	224

Importante

- Este es un material preparado para el curso de Métodos Numéricos. La primera versión fue para el curso 2023, y para esta edición hemos hecho algunos ajustes: modificamos contendios de varias secciones y capítulos, agregamos un capítulo acerca de integración numérica y otro sobre descomposición en valores singulares.

Si encontrás errores o tenés comentarios o sugerencias sobre estas notas, te agradecemos escribir en el foro de [Sugerencias y erratas en las notas teóricas](#) la página del curso.

- Las notas están en su segundo semestre en uso. Varios estudiantes hicieron sugerencias y comentarios, encontraron errores de tipo o inconsistencias, y con eso ayudaron a mejorar las notas. De todos modos, es inevitable que sigan habiendo imperfectos; en la medida que hagamos correcciones sobre lo que ya está escrito, en el texto van a estar en **rojo** y vamos a dejarlas detalladas en la lista de actualizaciones.

Notación

Pautamos aquí muy brevemente alguna notación que utilizamos en las notas del curso.

- Cuando escribamos un símbolo de igualdad con dos puntos por delante, esto denota que se trata de una definición. Por ejemplo, si ponemos $a := b + c$ (o $b + c =: a$) estamos *definiendo* la variable a como la suma de b y c ; en cambio, si ponemos $a = b + c$ (o $b + c = a$) esta relación se *deduce* de las definiciones de a, b, c .
- Hacemos un uso un tanto liberal del símbolo “ \approx ”, que indica que estamos haciendo una aproximación, ya sea de un valor numérico o de una función. Por ejemplo, perfectamente podríamos escribir

$$\pi \approx 3, \quad \pi \approx 3,1416, \quad \text{o} \quad \cos(t) \approx 1 - \frac{t^2}{2} \quad \text{para } t \approx 0.$$

- Escribimos los números reales usando la fuente normal, y los vectores en negrita; en general, usaremos minúsculas para todos ellos. Así, tendremos $x \in \mathbb{R}$ y $\mathbf{x} \in \mathbb{R}^n$. En cambio, escribimos las matrices usando mayúsculas con fuente normal: $A \in \mathcal{M}_{m \times n}(\mathbb{R})$. Al conjunto de matrices cuadradas $n \times n$ lo denotamos simplemente como $\mathcal{M}_n(\mathbb{R})$.
- Cada demostración de un lema, proposición, teorema o corolario, concluye con el símbolo \square . En cambio, toda observación, definición o ejemplo concluye con el símbolo \triangle .

Definición 0.0.1 (*O* “grande” y *o* “chica”). Dadas dos funciones $f, g : D \subset \mathbb{R} \rightarrow \mathbb{R}$, un punto a interior a D , escribimos $f(x) = O(g(x))$ en a si existen $K > 0$ y un entorno de a tales que $g(x) > 0$ y

$$|f(x)| \leq K g(x)$$

para todo x en dicho entorno. Intuitivamente, $f(x) = O(g(x))$ en a quiere decir que f no crece más rápido que g cerca de a .

Por otra parte, escribimos $f(x) = o(g(x))$ en a si para todo $\epsilon > 0$, existe un entorno de a tal que $g(x) > 0$ y

$$|f(x)| \leq \epsilon g(x)$$

para todo x en dicho entorno o, equivalentemente, si

$$\lim_{x \rightarrow a} \frac{f(x)}{g(x)} = 0.$$

Intuitivamente, el hecho de que $f(x) = o(g(x))$ en a quiere decir que f es de magnitud menor que g cerca de a .

La notación *O* y *o* se extiende, de forma análoga, al caso en que $a = \pm\infty$. \triangle

Observación 0.0.1 (no simetría). De la definición se deduce que, si $f(x) = o(g(x))$ en a , entonces $f(x) = O(g(x))$ en a . Por otra parte, si $f(x) = O(g(x))$ en a entonces **no** se deduce que $g(x) = O(f(x))$ en a , incluso si ambas funciones son positivas. \triangle

Ejemplo 0.0.1 (órdenes). Sean $f(x) = x$ y $g(x) = \operatorname{sen}(x) + x^2$. Tenemos

$$f(x) = O(|g(x)|) \text{ en } 0, \quad g(x) = O(f(x)) \text{ en } 0, \quad f(x) = o(g(x)) \text{ en } +\infty.$$

\triangle

Lista de versiones y actualizaciones

- 31 de julio de 2025: tercera versión.
- 25 de julio de 2024: segunda versión.
- Julio-diciembre de 2023: primera versión.

Capítulo 1

Aritmética de punto flotante y errores

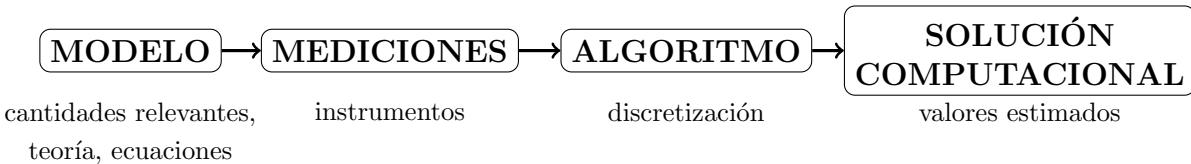
1.1. Introducción. Aproximaciones y errores.

Consideremos un proceso o sistema real del cual se desea conocer el comportamiento de determinadas cantidades a partir de cierta información de base. Podemos considerar dos estrategias para resolver problemas de este tipo:

- **Experimentación.** A través de la realización de experimentos es posible obtener valores reales del comportamiento que se desea estudiar. A pesar de esto, en muchos casos la experimentación resulta costosa (ensayos de materiales, inspección de recursos naturales), y en otros casos se tardaría demasiado (políticas económicas, sistemas biológicos).
- **Modelamiento computacional.** Utilizando herramientas de las ciencias exactas es posible formular problemas matemáticos que modelan o describen el comportamiento de dichos sistemas. A través del uso de *métodos numéricos* es posible resolver estos problemas y así predecir el comportamiento de los sistemas, aunque obteniendo un error con respecto al proceso real en todos los casos.

En este curso nos enfocamos en la segunda estrategia y en ciertos aspectos relativos a la misma. Para lograr nuestro cometido, en primer lugar requerimos **modelos** que nos permitan capturar las características que consideramos relevantes sobre el fenómeno a estudiar. El modelo que elijamos conllevará un conjunto de ecuaciones que representan cómo varían las cantidades que nos interesan al cambiar los parámetros de base. Para determinar los valores de dichos parámetros de base, que necesitaremos en nuestras ecuaciones, debemos hacer **mediciones**. Una vez que tenemos un modelo y mediciones, nos encontramos un conjunto de ecuaciones a resolver; a menos que estas ecuaciones sean muy sencillas, no podremos resolverlas en una cantidad finita de operaciones. Necesitamos una

estrategia de **aproximación** para estimar sus soluciones, lo que requiere el diseño de algoritmos. Finalmente, seguramente realicemos un programa computacional que implemente nuestro algoritmo y nos devuelva una **solución computacional** a nuestro problema. Esquemáticamente, tenemos los siguientes pasos.



Ejemplo 1.1.1 (caída de un objeto). Deseamos determinar la velocidad terminal de caída de un objeto cayendo del cielo, esto es, la velocidad v_f con la que llega al piso.

- Para el **modelado**, seguramente utilicemos las leyes que nos da la mecánica newtoniana. ¿Debemos incluir el efecto del rozamiento del aire? No hacerlo conduce a un modelo muy simple (caída libre), pero que no sería apropiado si, por ejemplo, queremos determinar la velocidad de caída de una hoja de un árbol. En algunos casos, como en la caída de una gota de lluvia, puede ser relevante que el modelo incluya que la masa del objeto pueda cambiar con el tiempo (porque la gota podría ir absorbiendo pequeñas gotitas del aire). Un buen modelo debe ser lo suficientemente rico como para capturar los fenómenos que nos interesan con la precisión que nos interesa, pero también lo suficientemente simple como para ser tratable.
- Una vez determinadas las ecuaciones modelando la caída del objeto, éstas involucran ciertos parámetros que debemos **medir o estimar**, como por ejemplo la altura y velocidad inicial de la caída, o la constante gravitatoria. En caso de incluir el rozamiento del aire, también podrían ser relevantes la forma y masa del objeto y el coeficiente de rozamiento. Para una gota de lluvia cuya masa va variando, es necesario incluir en el modelo la humedad del aire que la rodea, que también debe ser determinada.
- Nos encontramos ahora ante un sistema de ecuaciones (en este ejemplo, diferenciales) que debemos resolver. En algún caso es posible que podamos resolvérlas analíticamente, pero en general necesitaremos una estrategia para aproximar las soluciones. Esto es lo que se conoce como una **discretización**; hacia el final de este curso aprenderemos métodos para tratar con ecuaciones diferenciales ordinarias. Podemos aproximar las ecuaciones diferenciales involucradas mediante un sistema de ecuaciones algebraicas si, por ejemplo, reemplazamos las derivadas por cocientes incrementales,

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \text{ con } h > 0 \text{ "chico".}$$

- Finalmente, debemos resolver nuestro sistema de ecuaciones algebraicas. El tamaño del sistema implica que seguramente sea inviable resolverlo “a mano”, y buscaremos implementar un **programa computacional** para realizar la tarea. La salida

(output) de este programa deben ser las cantidades que deseemos estimar, como por ejemplo la velocidad terminal de caída del objeto.



Este curso se concentra en los dos últimos puntos discutidos en el Ejemplo 1.1.1: el diseño, implementación y análisis de métodos para la solución de problemas de ingeniería. Asumimos que tenemos dadas las ecuaciones o los problemas que queremos resolver. Buscamos desarrollar herramientas que nos permitan tratar computacionalmente con estos modelos, comparar diferentes métodos entre sí, entender en qué casos uno puede ser mejor que otro, estimar sus costos computacionales, e introducir nociones básicas del área de la matemática llamada *Análisis Numérico*.

1.1.1. Errores

Al resolver problemas reales mediante modelos computacionales, debemos lidiar con diversos *errores*. En cada una de las etapas de las descritas en el Ejemplo 1.1.1 se cometen errores. En referencia a los cuatro puntos mencionados en ese ejemplo, podemos tener:

- **Errores de modelo:** un [aforismo](#) muy popular en estadística dice:

Todos los modelos están mal, pero algunos son útiles.

Es inviable pretender que nuestro modelo pueda tener en cuenta toda la complejidad del problema que queremos estudiar. Es necesario hacer simplificaciones.

- **Errores de medición:** no siempre podemos recolectar toda la información que queremos, y nuestras mediciones no pueden ser infinitamente precisas.
- **Errores de aproximación:** las ecuaciones que nos da el modelo generalmente no pueden ser resueltas de forma exacta con una cantidad finita de operaciones. Debemos utilizar algoritmos que nos permitan llegar a soluciones a menos de una cierta tolerancia para el error.
- **Errores de redondeo:** las máquinas no trabajan con números reales sino con una aritmética de precisión finita, llamada *de punto flotante*, y que estudiamos en este capítulo. Al representar números reales en punto flotante, introducimos una nueva fuente de error.

Además de estos errores, no podemos descartar los errores humanos (en cualquiera de las etapas) y los errores de instrumentos (como fallas en el equipamiento para medir, defectos de hardware). En este curso no vamos a tener en cuenta estos últimos factores.

En este capítulo nos centramos en los errores asociados a la representación de punto flotante y de operaciones entre números representados en punto flotante. En el resto del

curso, nos vamos a enfocar en errores de aproximación para diversos métodos, pero en la Sección 1.2 vamos a mostrar con algunos ejemplos a qué nos referimos con la expresión “errores de aproximación”. La Sección 1.3 define los conceptos de error absoluto y relativo y nos da algunas pautas sobre cómo los errores se pueden propagar al aplicar funciones u operaciones aritméticas elementales. En la Sección 1.4 introducimos el sistema de numeración que usan las computadoras, vemos algunas constantes especiales y consideramos su capacidad de representar números reales. Finalmente, la Sección 1.5 se enfoca en cómo los distintos tipos de error pueden aparecer y propagarse en un algoritmo.

1.2. Errores de aproximación: una ventana a algunos temas del curso

El alma de este curso es el estudio de métodos para realizar **discretizaciones** (u aproximaciones) que nos permitan resolver algunos problemas. Aunque tal vez en cursos anteriores el foco no estuviese en este punto, ya tenemos alguna experiencia en este tema. A continuación mostramos algunos ejemplos, y en los capítulos siguientes consideramos y profundizamos en muchísimos más.

1.2.1. Sumas de Riemann

Del curso de Cálculo Diferencial e Integral en una Variable, sabemos que el concepto de determinar el área encerrada entre el gráfico de una función real f y el eje de las x en un cierto intervalo está dado por la *integral de f entre a y b* . A su vez, dicha integral está dada por el ínfimo de las sumas superiores o el supremo de las sumas inferiores de f en el intervalo. Por lo tanto, podemos aproximar la integral de f entre a y b mediante sumas de Riemann (inferiores o superiores) para una partición lo suficientemente fina.

Ejemplo 1.2.1 (sumas de Riemann). La integral

$$I = \int_0^1 e^{\sin x} dx$$

está bien definida (el integrando es monótono creciente en el intervalo $[0, 1]$) pero el valor de I no puede escribirse usando una cantidad finita de funciones elementales: no conocemos una primitiva de $e^{\sin x}$. Podemos considerar una partición uniforme del intervalo $[0, 1]$ con $n + 1$ puntos, $P_n = \{a_0, a_1, \dots, a_n\}$, esto es, $a_i = i/n$ con $i = 0, \dots, n$, y computar las sumas inferiores y superiores asociadas a P_n ,

$$\begin{aligned} S_*(f, P_n) &= \sum_{i=0}^{n-1} (a_{i+1} - a_i) \inf_{x \in [a_i, a_{i+1}]} e^{\sin x} = \sum_{i=0}^{n-1} \frac{1}{n} e^{\sin(\frac{i}{n})}, \\ S^*(f, P_n) &= \sum_{i=0}^{n-1} (a_{i+1} - a_i) \sup_{x \in [a_i, a_{i+1}]} e^{\sin x} = \sum_{i=0}^{n-1} \frac{1}{n} e^{\sin(\frac{i+1}{n})}. \end{aligned}$$

Por ejemplo, usando una partición P_{10} con $n = 10$ intervalos, esto nos conduce a

$$1,566\dots = S_*(f, P_{10}) \leq I \leq S^*(f, P_{10}) = 1,698\dots$$

Esto nos da cotas bastante groseras para el valor de I : haciendo el promedio entre nuestra cota superior y nuestra cota inferior, deducimos que $I \approx 1,632 \pm 0,066$. Si esta precisión fuese suficiente para los cálculos que necesitamos, podríamos quedarnos satisfechos. Si no lo fuese, podríamos tomar mayor cantidad de puntos.

n	$S_*(f, P_n)$	$S^*(f, P_n)$	Estimación de I
10	1,56609\dots	1,69807\dots	1,6320813 ± 0,06599
100	1,62527\dots	1,63847\dots	1,6318717 ± 0,00659
1000	1,63120\dots	1,63252\dots	1,6318696 ± 0,00066
10000	1,63180\dots	1,63193\dots	1,6318696 ± 0,00007

En principio, podríamos seguir aumentando n hasta tener una aproximación de I tan precisa como queramos. Llama la atención que, cada vez que aumentamos n en un factor de 10, el margen de error también se reduce en un factor de aproximadamente 10. ¿Será esto casualidad? ¿Es posible diseñar algún método que nos permita estimar I de forma más eficiente, logrando que el margen de error se reduzca más rápido? \triangle

Ejercicio 1.1. Implementar un código de Octave que permita reproducir los resultados de la tabla de arriba.

1.2.2. Desarrollos de Taylor

El Teorema de Taylor nos indica que toda función lo suficientemente regular puede ser aproximada mediante funciones polinomiales, los llamados *polinomios de Taylor*. Además, contamos con expresiones para el error cometido al hacer estas aproximaciones por polinomios: las llamadas formas de Lagrange o de Cauchy para el resto.

Si tenemos una función que no podemos evaluar de forma sencilla, los polinomios de Taylor nos ofrecen aproximaciones a los valores de dicha función.

Ejemplo 1.2.2 (polinomio de Taylor). Queremos aproximar el número real $y = \operatorname{sen}(10^{-1})$ con 4 dígitos de precisión. Utilizando los Teoremas A.1.1 y A.1.2 del capítulo de repaso tenemos, para $f(x) = \operatorname{sen}(x)$ (que es infinitamente derivable) y evaluando el polinomio de orden k alrededor de $a = 0$ en el punto $x = 10^{-1}$,

$$\operatorname{sen}(10^{-1}) = \operatorname{sen}(0) + \cos(0)10^{-1} + \dots + \frac{\operatorname{sen}^{(k)}(0)}{k!}(10^{-1})^k + r_k(10^{-1}),$$

donde

$$r_k(10^{-1}) = \frac{\operatorname{sen}^{(k+1)}(\theta_x)}{(k+1)!}(10^{-1})^{k+1}, \quad \text{y} \quad \theta_x \in [0, 10^{-1}].$$

Si usamos $k = 1$, obtenemos la aproximación $\operatorname{sen}(10^{-1}) \approx 10^{-1} = 0,1$, y podemos estimar el error cometido usando que la función seno está acotada entre -1 y 1 :

$$|r_1(10^{-1})| = \left| \frac{-\operatorname{sen}(\theta_x)}{2} 10^{-2} \right| \leq \frac{10^{-2}}{2} = 0,005.$$

Esta aproximación no es lo suficientemente buena para nuestro objetivo, por lo que intentamos con un polinomio de mayor grado. Notemos que poner $k = 2$ no nos aporta ningún otro término (porque $\operatorname{sen}^{(2)}(0) = -\operatorname{sen}(0) = 0$), por lo que tomamos $k = 3$ para obtener $\operatorname{sen}(10^{-1}) \approx 10^{-1} - \frac{10^{-3}}{6} = 0,09983$. Podemos acotar el error de esta aproximación mediante

$$|r_3(10^{-1})| = \left| \frac{\operatorname{sen}(\theta_x)}{24} 10^{-4} \right| \leq \frac{10^{-4}}{24} = 4,16 \times 10^{-6}.$$

Esto quiere decir que nuestra aproximación ya es tan precisa como queríamos.

Más aún, podemos observar que poner $k = 4$ no aportaría ningún otro término (porque $\operatorname{sen}^{(4)}(0) = \operatorname{sen}(0) = 0$), de modo que la aproximación anterior de hecho corresponde al polinomio de Taylor de grado 4 y por lo tanto podemos mejorar nuestra estimación de error:

$$|r_4(10^{-1})| = \left| \frac{\operatorname{sen}(\theta_x)}{120} 10^{-5} \right| \leq \frac{10^{-5}}{120} \approx 8,3 \times 10^{-8}.$$

△

1.2.3. Calcular raíces cuadradas como un Herón

En el siguiente ejemplo, asumiremos que tenemos a mano una calculadora que solamente nos permite realizar las operaciones básicas (suma, resta, multiplicación, división). En muchas aplicaciones es necesario calcular raíces cuadradas. Hace unos dos mil años la necesidad de tener estimaciones precisas era cada vez mayor y lamentablemente los celulares de la época no tenían calculadora, por lo que había que hacer las cuentas “a mano”. El método que describimos a continuación se debe a [Hérón de Alejandría](#).

Ejemplo 1.2.3 (aproximación de raíces cuadradas). Queremos estimar el número real $\sqrt{7}$ y lo vamos a hacer a partir de una sucesión $\{a_n\}$ definida por recurrencia. La idea es que, si $a_n < \sqrt{7}$ (respectivamente, $a_n > \sqrt{7}$) entonces $7/a_n > \sqrt{7}$ (respectivamente, $7/a_n < \sqrt{7}$), por lo que el punto medio entre a_n y $7/a_n$ puede dar una mejor aproximación de $\sqrt{7}$ que cualquiera de estos dos números. Por lo tanto, dado $a_n \in \mathbb{R}$, consideramos

$$a_{n+1} = \frac{1}{2} \left(a_n + \frac{7}{a_n} \right).$$

Sólo necesitamos un iterado inicial para tener definida nuestra sucesión. De forma sencilla, podemos estimar que $2 < \sqrt{7} < 3$. Como primera estimación, tomemos $a_0 = 3$. Esto da lugar a la sucesión cuyos primeros iterados mostramos a continuación. En negrita, marcamos las cifras correctas en a_n .

n	a_n	$a_n^2 - 7$
0	3	2
1	$\frac{8}{3} = 2,666\bar{6}$	$0.\bar{1}$
2	$\frac{127}{48} = 2,6458\bar{3}$	$\approx 4,3 \times 10^{-4}$
3	$\frac{32257}{12192} \approx 2,645751312335958$	$\approx 6,7 \times 10^{-9}$

Vemos que, en cada iteración de este procedimiento, la cantidad de dígitos correctos esencialmente se duplica. Esto se puede justificar de la siguiente forma: sea $\varepsilon_n = \frac{a_n - \sqrt{7}}{\sqrt{7}}$; en la Sección 1.3 definiremos a esta cantidad como *error relativo* de la aproximación a_n . Es fácil verificar que se cumple

$$\varepsilon_{n+1} = \frac{\varepsilon_n^2}{2(1 + \varepsilon_n)}, \quad (1.1)$$

lo que significa que el error relativo se reduce cuadráticamente y por lo tanto es esperable que la cantidad de dígitos correctos se vaya duplicando de un iterado al siguiente. \triangle

Ejercicio 1.2. Aproximar $\sqrt{7}$ usando polinomios de Taylor de grado 1 y 2 alrededor del punto $a = 3$, como en el Ejemplo 1.2.2. Notar que usando el polinomio de grado 1 se tiene la misma aproximación que a_1 arriba, pero con polinomios de grado 2 se tiene una aproximación bastante peor que a_2 .

Ejercicio 1.3. Sea $a \in \mathbb{R}^+$. Generalizar el método del Ejemplo 1.2.3 para estimar \sqrt{a} e implementarlo en Octave.

1.3. Errores

La noción de *error* se refiere a una discrepancia entre lo que computamos y lo que queremos computar. En esta sección introducimos la diferencia entre error absoluto y relativo y estimamos cómo una discrepancia en las entradas afecta a la salida al evaluar una función o al realizar operaciones aritméticas elementales (suma, resta, producto, división).

1.3.1. Errores absolutos y relativos

Para estimar el error cometido al aproximar una cierta cantidad, es importante distinguir la *magnitud absoluta* del error de su *magnitud relativa*. Por ejemplo, supongamos que A y B son dos deportistas que corren una carrera y A llega 10 segundos antes que B. Para poder decir si A le ganó a B “por mucho” o “por poco” necesitamos conocer cuánto duró la carrera: no es lo mismo si la carrera fue de 100 metros que si fue un maratón de más de 42 kilómetros.

Definición 1.3.1 (error absoluto). Sean $x \in \mathbb{R}$ un número real dado y $\bar{x} \in \mathbb{R}$ una aproximación de x . Definimos el **error absoluto** como la diferencia entre \bar{x} y x ,

$$e_x := \bar{x} - x.$$

△

Definición 1.3.2 (error relativo). Sean $x \in \mathbb{R}$ un número real dado con $x \neq 0$ y $\bar{x} \in \mathbb{R}$ una aproximación de x . Definimos el **error relativo** como el error absoluto normalizado por x ,

$$\varepsilon_x := \frac{\bar{x} - x}{x} = \frac{e_x}{x}.$$

△

Observación 1.3.1 (errores con signo). Las definiciones anteriores implican que los errores absoluto y relativo tienen signo: pueden ser positivos o negativos. Elegimos esta convención porque nos permite relacionar las cantidades x y \bar{x} mediante las sencillas identidades $\bar{x} = x + e_x$ y

$$\bar{x} = (1 + \varepsilon_x) x. \quad (1.2)$$

En general, en las aplicaciones nos interesa la magnitud de los errores y no su signo. En esos casos, tenemos que estimar o acotar $|e_x|$ o $|\varepsilon_x|$. △

La definición de error absoluto se puede generalizar a espacios vectoriales, en particular al espacio \mathbb{R}^n . En ese caso, dados un vector $\mathbf{x} \in \mathbb{R}^n$ y una aproximación $\bar{\mathbf{x}} \in \mathbb{R}^n$, definimos el error absoluto mediante

$$\mathbf{e}_{\mathbf{x}} := \bar{\mathbf{x}} - \mathbf{x} \in \mathbb{R}^n.$$

Para generalizar nuestra definición de error relativo, como no podemos dividir por el vector \mathbf{x} , necesitamos introducir una norma en nuestro espacio vectorial. Por ejemplo, si $\|\cdot\|$ denota la norma euclídea en \mathbb{R}^n , podríamos definir el error relativo mediante

$$\varepsilon_{\mathbf{x}} := \frac{\bar{\mathbf{x}} - \mathbf{x}}{\|\mathbf{x}\|} \quad \text{para } \mathbf{x} \neq \mathbf{0};$$

notar que esto no es exactamente lo mismo que definimos en \mathbb{R} , por lo que en caso de ser necesario vamos a aclarar qué definición de error relativo estamos utilizando.

1.3.2. Funciones reales: número de condición

Queremos evaluar una función derivable $f: I \rightarrow \mathbb{R}$, donde $I \subset \mathbb{R}$, y suponemos que *tenemos la capacidad de computar f exactamente*. Nos preguntamos cómo un error en la entrada puede afectar a nuestra evaluación de f . Dado un número real $x \neq 0$ tal que $f(x) \neq 0$, consideramos una aproximación $\bar{x} \in \mathbb{R}$. En virtud de (1.2), podemos relacionar de forma sencilla a $\bar{x} \in \mathbb{R}$ con el error relativo de esta aproximación.

Nos interesa computar $y := f(x)$, pero dado que tenemos acceso a \bar{x} , lo mejor que podemos hacer es computar $\bar{y} = f(\bar{x})$. Podemos estimar el error relativo en y usando un desarrollo de Taylor de primer orden para f alrededor de x :

$$\varepsilon_y = \frac{\bar{y} - y}{y} = \frac{f(\bar{x}) - f(x)}{f(x)} \approx \frac{f'(x)(\bar{x} - x)}{f(x)} = \frac{f'(x)x}{f(x)} \frac{(\bar{x} - x)}{x}.$$

Al ser $\frac{(\bar{x}-x)}{x} = \varepsilon_x$, deducimos que

$$\varepsilon_y \approx \frac{f'(x)x}{f(x)} \varepsilon_x,$$

lo que significa que el error relativo en $y = f(x)$ se relaciona con el error relativo en x mediante un cierto factor. Esto motiva la siguiente definición.

Definición 1.3.3 (número de condición). Llamamos **número de condición de f en el punto x** al número

$$\kappa_f(x) := \left| \frac{f'(x)x}{f(x)} \right|. \quad (1.3)$$

Vagamente,

- si $\kappa_f(x) \sim 1$, diremos que el problema de evaluar f en x está *bien condicionado*;
- $\kappa_f(x) \gg 1$, diremos que el problema de evaluar f en x está *mal condicionado*.

△

Remarcamos que el hecho de que el problema de evaluar f en x esté bien o mal condicionado solamente tiene que ver con la función f . No estamos asumiendo nada sobre cómo estamos computando f . En matemática, los números de condición dan una pauta de la sensibilidad en la salida de una operación (como evaluar una función) respecto a sus entradas. Al trabajar con sistemas de ecuaciones lineales, vamos a retomar este concepto y dar una definición análoga a (1.3).

1.3.3. Propagación de errores en operaciones aritméticas

Sean x e y dos números reales, para los que tenemos almacenados respectivas aproximaciones \bar{x} e \bar{y} . En esta sección analizamos lo que ocurre con la magnitud del error de aproximación al realizar las operaciones aritméticas elementales (suma, resta, producto, división). Recordamos la identidad (1.2),

$$\bar{x} = (1 + \varepsilon_x) x, \quad \bar{y} = (1 + \varepsilon_y) y. \quad (1.4)$$

Suponemos que ε_x y ε_y son ambos de magnitud pequeña, en el sentido de que su producto satisface $|\varepsilon_x \varepsilon_y| \ll \min\{|\varepsilon_x|, |\varepsilon_y|\}$.

Producto. Multiplicamos las dos igualdades en (1.4) y obtenemos

$$\bar{x}\bar{y} = xy(1 + \varepsilon_x)(1 + \varepsilon_y) \approx xy(1 + \varepsilon_x + \varepsilon_y),$$

de donde deducimos que, en el producto xy , se tiene el error relativo

$$\varepsilon_{xy} \approx \varepsilon_x + \varepsilon_y.$$

Esto implica que, en el peor de los casos, el error relativo en el producto entre dos números es comparable en magnitud al peor de los errores relativos en ellos.

División. Suponiendo que ni y ni \bar{y} son nulos, dividimos las dos igualdades en (1.4) y usamos el equivalente¹ $\frac{1}{1+t} \approx 1 - t$ para obtener

$$\frac{\bar{x}}{\bar{y}} = \frac{x}{y} \frac{(1 + \varepsilon_x)}{(1 + \varepsilon_y)} \approx \frac{x}{y} (1 + \varepsilon_x)(1 - \varepsilon_y) \approx \frac{x}{y} (1 + \varepsilon_x - \varepsilon_y).$$

Esto implica que, en la división x/y , tenemos el error relativo

$$\varepsilon_{x/y} \approx \varepsilon_x - \varepsilon_y.$$

En consecuencia, en el peor de los casos, el error relativo en la división entre dos números es comparable en magnitud al peor de los errores relativos en ellos. Recordamos que los errores relativos son tomados con signo, por lo que el signo negativo en ε_y no tiene ningún significado especial.

Suma y resta. Como restar es sumar el opuesto de un número, solamente consideraremos la suma; naturalmente, podemos suponer que $x, y \neq 0$. Escribimos directamente la definición de ε_{x+y} y usamos las dos identidades en (1.4):

$$\varepsilon_{x+y} = \frac{\bar{x} + \bar{y} - (x + y)}{x + y} = \frac{x}{x + y} \varepsilon_x + \frac{y}{x + y} \varepsilon_y.$$

Aquí, vemos que aparecen los factores $\frac{x}{x+y}$ e $\frac{y}{x+y}$ multiplicando a ε_x y ε_y , respectivamente. Distinguimos tres casos.

- Si $xy > 0$ (es decir, si x e y tienen igual signo), entonces los factores que multiplican a ε_x y ε_y son ambos positivos, suman 1, y podemos acotar

$$|\varepsilon_{x+y}| \leq \frac{x}{x+y} \max\{|\varepsilon_x|, |\varepsilon_y|\} + \frac{y}{x+y} \max\{|\varepsilon_x|, |\varepsilon_y|\} = \max\{|\varepsilon_x|, |\varepsilon_y|\}.$$

En este caso, el error relativo en la suma $x + y$ no es peor en magnitud que el peor de los errores relativos en x e y .

- Si $xy < 0$ y $|x| \gg |y|$ (o $|x| \ll |y|$), entonces los factores que multiplican a ε_x y ε_y no son mucho mayores que 1 en valor absoluto y tenemos

$$|\varepsilon_{x+y}| \leq \left| \frac{x}{x+y} \right| |\varepsilon_x| + \left| \frac{y}{x+y} \right| |\varepsilon_y| \lesssim |\varepsilon_x| + |\varepsilon_y|.$$

En este caso tenemos que, en el peor de los casos, el error relativo en la suma $x + y$ es comparable en magnitud con el peor de los errores relativos en x e y .

¹Este equivalente no es otra cosa que el polinomio de Taylor de orden 1 de la función $f(t) = \frac{1}{1+t}$ alrededor del origen.

- Si $xy < 0$ y $x \approx -y$, entonces tendremos $x + y \approx 0$. Esto nos pone en problemas: los factores que multiplican a ε_x y ε_y pueden ser muy grandes! Este fenómeno lleva el nombre de **cancelación catastrófica**.

Vamos a ilustrar por qué se usa el tan dramático término “cancelación catastrófica” con un ejemplo.

Ejemplo 1.3.1 (cancelación catastrófica). Buscamos las raíces del polinomio $x^2 - 56x + 1 = 0$. Es fácil mostrar que estas raíces son

$$r_1 = 28 + \sqrt{783} \approx 55,982; \quad r_2 = 28 - \sqrt{783} \approx 0,017863.$$

Supongamos que solamente tenemos 5 cifras de precisión. Con ellas, 28 se computa exactamente y $\sqrt{783}$ se aproxima como 27,982. Tenemos errores relativos $\varepsilon_{28} = 0$ y $\varepsilon_{\sqrt{783}} \approx -4,90 \times 10^{-6}$.

La raíz r_1 se puede calcular sin error considerable (e incluso mejor que el de $\sqrt{783}$) pues estamos sumando números positivos,

$$\bar{r}_1 = 28 + 27,982 = 55,982 \quad \varepsilon_{r_1} \approx -2,45 \times 10^{-6}.$$

Sin embargo, el cálculo de la raíz r_2 implica una resta de dos valores muy próximos,

$$\bar{r}_2 = 28 - 27,982 = 0,018, \quad \varepsilon_{r_2} \approx 7,7 \times 10^{-3}.$$

Observamos que el error relativo en r_2 aumentó en magnitud en un factor de aproximadamente 1500 respecto al error relativo en las entradas que utilizamos. Este aumento de magnitud puede causar que un error tolerable en las entradas (nuestras aproximaciones de 28 y $\sqrt{782}$) dé lugar a un error *no tolerable en la aproximación de r_2* : esto es lo que se llama una cancelación catastrófica.

Para evitar caer en una cancelación catastrófica, es importante evitar computar r_2 en la forma en que lo hicimos. Una forma de lograrlo es reescribir el polinomio original como

$$x^2 - 56x + 1 = (x - r_1)(x - r_2) = x^2 - (r_1 + r_2)x + r_1 r_2.$$

Podemos determinar r_2 igualando los coeficientes término a término y usando el valor ya calculado para r_1 . Si utilizamos el término independiente se tiene

$$r_1 r_2 = 1 \Leftrightarrow \bar{r}_2 = \frac{1}{\bar{r}_1} = \frac{1}{55,982} \approx 0,017863, \quad \varepsilon_{r_2} \approx 8,9 \times 10^{-6}.$$

Esta es una aproximación que tiene un error relativo del mismo orden de magnitud que el que tenemos en nuestra aproximación de $\sqrt{783}$. En cambio, observar que si en lugar del término independiente quisiéramos determinar r_2 usando el término en x , entonces llegaríamos al mismo mal resultado que antes. \triangle

Del ejemplo anterior, sacamos una importante moraleja: **cuando nos enfrentamos a problemas que implican restar números de igual magnitud, es importante buscar reescribir las fórmulas para evitar cancelaciones catastróficas.**

1.4. Aritmética de punto flotante

Independientemente de qué sistema de representación usemos, en general un número real puede tener infinitos dígitos. Intuitivamente, esto quiere decir que para almacenar *exactamente* un número real podemos llegar a necesitar *infinita* información. En consecuencia, las computadoras tienen que *truncar* en algún lugar las representaciones de los números. Además, la cantidad de números reales es *infinita*, por lo que tampoco es esperable que las computadoras los puedan almacenar a todos: la cantidad de números representables es finita.

Vamos a hacer pasar un poco de vergüenza a nuestra máquina. Hacemos la siguiente operación “a mano”:

$$1 + 10^{-16} - 1 = 10^{-16}.$$

Sin embargo, si ejecutamos la misma operación (y en el *mismo orden*) en Octave, nos encontramos con una sorpresa:

```
>> 1 + 1e-16 - 1      [Enter]
ans = 0
```

¿Por qué ocurre esto? La máquina realiza las operaciones en el orden en que se le presentan, por lo que primero suma $1 + 10^{-16}$ y al resultado que obtiene le resta 1. El hecho de que el resultado final sea igual a 0 quiere decir que, **para la máquina, los números $1 + 10^{-16}$ y 1 son indistinguibles**.

En general, al almacenar un número x podremos guardar algún número próximo a éste de entre el conjunto de números que puede representar exactamente la máquina. A continuación estudiamos este conjunto y algunas de sus propiedades básicas, luego analizamos qué efectos puede tener este error de *representación* (o almacenamiento), y finalmente marcamos algunos cuidados que debemos tener en consecuencia.

1.4.1. Representación de punto flotante

Un *bit* es la unidad básica de información usada en el cómputo: es un número binario (un 0 o un 1). En la aritmética de punto flotante, cada número es almacenado por un conjunto de bits. Hoy en día, el estándar más utilizado es el [formato de precisión doble de la IEEE](#), en el que cada número es representado por 64 bits, esto es, una tira de 0's o 1's de longitud 64. De los 64 bits que forman a un número x , tendremos

- 1 bit para el signo (\pm);
- 52 bits para la mantisa (f);
- 11 bits para el exponente (e).

Así, los **números de punto flotante** son aquellos que se pueden escribir de la forma

$$x = \pm(1 + f) \times 2^e. \quad (1.5)$$

Aquí, la **mantisa** es un número $0 \leq f < 1$ tal que $2^{52}f$ es un número entero, esto es,

$$f = \frac{j}{2^{52}}, \quad j \in \{0, 1, \dots, 2^{52} - 1\}.$$

Notemos que hay exactamente 2^{52} posibles valores para f , lo que se corresponde con el hecho de que f sea representado por 52 bits.

En cambio, el **exponente** e es un número entero tal que $-1022 \leq e \leq 1023$. Observemos que esta elección nos deja $2046 = 2^{11} - 2$ posibles valores para e ; los dos valores restantes para el exponente ($e = -1023$ y $e = 1024$) se reservan para ciertos *números especiales*.

Distribución de números

Analicemos con más detalle cómo se distribuyen los números de punto flotante. Para fijar ideas, consideremos los posibles números de punto flotante que se corresponden con signo positivo y $e = 0$ en (1.5). Al variar la mantisa f , tendremos un conjunto de 2^{52} números,

$$\left\{ 1, 1 + \frac{1}{2^{52}}, 1 + \frac{2}{2^{52}}, 1 + \frac{3}{2^{52}}, \dots, 1 + \frac{2^{52} - 1}{2^{52}} \right\}.$$

Este conjunto es el conjunto de números en el intervalo $[1, 2)$ que pueden ser representados *exactamente* por una máquina que use el estándar de precisión doble. Notamos que este conjunto es un conjunto de 2^{52} números equiespaciados, con una separación igual a 2^{-52} .

Ahora, fijemos un valor de e mayor en (1.5), por ejemplo $e = 4$. Al variar la mantisa f , nuevamente tendremos un conjunto de 2^{52} números equiespaciados pero ahora en el intervalo $[2^4, 2^5) = [16, 32)$,

$$\left\{ 16, \left(1 + \frac{1}{2^{52}}\right) \times 16, \left(1 + \frac{2}{2^{52}}\right) \times 16, \left(1 + \frac{3}{2^{52}}\right) \times 16, \dots, \left(1 + \frac{2^{52} - 1}{2^{52}}\right) \times 16 \right\}.$$

La separación entre elementos consecutivos de este conjunto es $\frac{16}{2^{52}} = 2^{-48}$, esto es, 16 veces mayor que la separación en el intervalo $[1, 2)$.

Observamos, pues, que **el conjunto de números de punto flotante es finito y la distancia entre números consecutivos no es globalmente uniforme**.

Redondeo y truncamiento.

Como hemos visto, los sistemas de representación pueden representar una cantidad finita de números, por lo que al desear representar un número real x que quizás no esté incluido en el mismo, la computadora deberá aproximarla con un número $\text{fl}(x)$ que sí lo esté. Para esta aproximación existen dos métodos habitualmente usados:

- **Redondeo:** aproximar el real al número representable más cercano. Si los números representables más cercanos son dos, se aproxima al menor.
- **Truncamiento:** aproximar el real al número representable menor más próximo.

A menos que especifiquemos lo contrario, en adelante asumimos que nuestra máquina hace redondeo para representar números reales que no estén incluidos en el sistema de punto flotante que use.

1.4.2. Constantes de máquina y números especiales

En esta sección definimos algunas constantes relevantes respecto a un sistema de punto flotante. Comenzamos con una que representa la capacidad relativa de aproximar números reales.

Definición 1.4.1 (épsilon de máquina). Llamamos **épsilon de máquina** (ε_M) a la separación entre el número 1 y el siguiente número representable de un sistema de punto flotante. \triangle

Observación 1.4.1 (valor numérico de ε_M). Para determinar el valor de ε_M en el formato de precisión doble, nos basta con observar que el menor número estrictamente mayor a 1 y representable en precisión doble se corresponde con las elecciones:

$$\text{signo: } +, \quad \text{exponente: } e = 0, \quad \text{mantisa: } f = 2^{-52},$$

lo que arroja el número $x = 1 + 2^{-52}$ en (1.5). Deducimos, por lo tanto, que en precisión doble

$$\varepsilon_M = 2^{-52} \approx 2,2 \times 10^{-16}.$$

En Octave y Matlab, el comando `eps` nos devuelve el valor de ε_M para precisión doble, que es el formato que nuestras máquinas usan por defecto:

```
>> eps [Enter]
ans = 2.2204e-16
```

\triangle

Como los sistemas de punto flotante incluyen una cantidad *finita* de números, necesariamente debe haber uno máximo y uno mínimo (en valor absoluto) representables.

Definición 1.4.2 (Real_{\max} y Real_{\min}). Llamamos **Real_{max}** al mayor número representable exactamente en un sistema de punto flotante y **Real_{min}** al menor número positivo representable en un sistema de punto flotante. \triangle

Observación 1.4.2. Para determinar el valor de Real_{\max} en el formato de precisión doble, basta con tomar el signo positivo, y el exponente y la mantisa mayores posibles:

$$\text{signo: } +, \quad \text{exponente: } e = 1023, \quad \text{mantisa: } f = 1 - 2^{-52},$$

lo que en (1.5) arroja

$$\text{Real}_{\max} = (2 - 2^{-52}) \times 2^{1023} \approx 1,8 \times 10^{308}.$$

De forma análoga, para determinar el valor de Real_{\min} en el formato de precisión doble tomamos el signo positivo, y el exponente y la mantisa menores posibles:

$$\text{signo: } +, \quad \text{exponente: } e = -1022, \quad \text{mantisa: } f = 0,$$

lo que da lugar a

$$\text{Real}_{\min} = 1 \times 2^{-1022} \approx 2,2 \times 10^{-308}.$$

En Octave y Matlab, los comandos `realmax` y `realmin` nos devuelven los valores de Real_{\max} y Real_{\min} para precisión doble, respectivamente:

```
>> realmax [Enter]
ans = 1.7977e+308
>> realmin [Enter]
ans = 2.2251e-308
```

△

Recordamos que los exponentes $e = -1023$ y $e = 1024$ se reservan para ciertos números especiales. Veamos algunos de estos casos.

Definición 1.4.3 (overflow e `Inf`). Si un cálculo en la máquina arroja un número mayor que Real_{\max} , decimos que se produce un **overflow**. El resultado de este cálculo es un número de punto flotante llamado `Inf`, que se almacena con el exponente $e = 1024$ y la mantisa $f = 0$. △

Definición 1.4.4 (`NaN`). Si hacemos un cálculo en la máquina que no está definido en el sistema real, como por ejemplo $0/0$ o $\text{Inf} - \text{Inf}$, el resultado de nuestro cálculo es un número de punto flotante llamado `NaN`², que se almacena con el exponente $e = 1024$ y una mantisa $f \neq 0$. △

Cero. Al número 0 obviamente no se lo puede escribir en el formato (1.5). En el estándar IEEE 754, se tienen *ceros con signo*: uno positivo y uno negativo, y que formalmente se tratan como objetos distintos. Estos números se representan con el exponente $e = -1023$, la mantisa $f = 0$ y el signo determina de qué cero se trata. Como como caso de uso, si escribimos en Octave:

²`NaN` proviene del inglés *Not-a-Number*, que significa “no es un número”.

```
>> x_pos = 1e-300;
>> x_neg = -1e-300;
>> pos_cero = x_pos * 0
    pos_cero = 0
>> neg_cero = x_neg * 0
    neg_cero = 0
```

obtenemos que ambos resultados son cero. Ahora, si usamos estas variables para calcular:

```
>> 1/pos_cero          [Enter]
    ans = Inf
>> 1/neg_cero          [Enter]
    ans= -Inf
```

tenemos dos infinitos distintos. En particular, escribiendo el cero positivo como $+0,0$ y el negativo como $-0,0$, en el estándar IEEE 754 se cumple

$$\frac{1}{+0,0} = +\text{Inf}, \quad \frac{1}{-0,0} = -\text{Inf}.$$

Por defecto, al computar o escribir 0 en Octave o Matlab, obtenemos $+0,0$.

Subnormales. Finalmente, puede ocurrir que hagamos en la máquina un cálculo cuyo resultado sea un número real menor en valor absoluto que Real_{\min} . En ese caso, se puede hablar de un *underflow*. El resultado podría ser 0 (uno de los dos!), pero optativamente el formato da lugar a una mayor representación: los números subnormales. Éstos son números a los que el estándar IEEE 754 deja como *optativos* de incluir. Se encuentran entre `eps*realmin` y `realmin` y se los representa con el exponente $e = -1023$ y mantis $f = 2^{-52}, \dots, 1 - 2^{-52}$.

1.4.3. Error de representación en punto flotante

Consideremos un número $x \in \mathbb{R} \setminus \{0\}$ y sea $\text{fl}(x)$ su representación en punto flotante utilizando *redondeo*. Una pregunta muy relevante para nosotros es estimar el error cometido al aproximar x por $\text{fl}(x)$. Como adelantamos antes de la Definición 1.4.1, el épsilon de máquina entra en juego para estimar el *error relativo* de esta aproximación (ver Definición 1.3.2). El siguiente resultado nos muestra que, independientemente de la magnitud de x , el error relativo al representarlo en punto flotante permanece uniformemente acotado.

Proposición 1.4.1 (aproximación relativa con precisión doble). *Sean $x \in \mathbb{R}$ tal que $2^{-1022} \leq |x| < 2^{1024}$ y $\text{fl}(x)$ su representación de punto flotante en formato de precisión doble. Entonces, se cumple*

$$|\varepsilon_x| = \frac{|\text{fl}(x) - x|}{|x|} \leq \frac{\varepsilon_M}{2}, \tag{1.6}$$

donde $\varepsilon_M = 2^{-52}$ es el épsilon de máquina de dicho formato.

Demostración. Sea x tal que $2^{-1022} \leq |x| < 2^{1024}$; sin perder generalidad, asumimos que x es positivo. Consideremos los intervalos de la forma $[2^e, 2^{e+1})$ con e entero y tal que $-1022 \leq e \leq 1023$. Como estos intervalos son disjuntos dos a dos y la unión de todos ellos es $[2^{-1022}, 2^{1024})$, necesariamente x tiene que estar en uno de ellos. Esto es, existe un único e entero con $-1022 \leq e \leq 1023$ y $\alpha_x \in [0, 1)$ tales que

$$x = (1 + \alpha_x) \times 2^e.$$

Si $\alpha_x \leq 1 - \frac{2^{-52}}{2}$, entonces la representación de punto flotante de x usando redondeo es un número $\text{fl}(x)$ de la forma

$$\text{fl}(x) = (1 + f_x) \times 2^e.$$

Notemos que el exponente es el mismo que en x , y la mantisa es $f_x = k2^{-52}$ para algún $k = 0, \dots, 2^{52} - 1$. Si restamos, dividimos por x y simplificamos las exponenciales, nos encontramos con que

$$\frac{\text{fl}(x) - x}{x} = \frac{f_x - \alpha_x}{1 + \alpha_x}.$$

Finalmente, tenemos que $1 + \alpha_x \geq 1$, y como estamos usando redondeo se cumple $|\alpha_x - f_x| \leq \frac{2^{-52}}{2}$; notar que como el espaciado entre mantisas es de 2^{-52} , la mantisa más cercana a α_x no puede estar a una distancia mayor que la mitad de este espaciado. Tomando valor absoluto en la igualdad anterior y acotando, se llega a (1.6).

Por otra parte, si $1 - \frac{2^{-52}}{2} < \alpha_x < 1$, entonces la representación de punto flotante de x usando redondeo es

$$\text{fl}(x) = 1 \times 2^{e+1}.$$

Dejamos como ejercicio verificar que en ese caso también se cumple (1.6). \square

Observación 1.4.3 (cota inferior para $|x|$). La condición sobre $|x|$ en la Proposición 1.4.1 es necesaria para evitar que $\text{fl}(x)$ sea Inf o 0, aunque la cota inferior puede mejorarse si se usan números subnormales. También observemos que esta proposición refiere al error relativo al aproximar con redondeo un número en punto flotante; si nos interesa estimar el error absoluto en esta aproximación, usando la Definición 1.3.1 tenemos

$$|e_x| = |x \varepsilon_x| \leq \frac{|x| \varepsilon_M}{2}.$$

Por lo tanto, **mientras nuestra mejor cota para el error relativo al aproximar un número real es uniforme, nuestra mejor cota para el error absoluto depende de la magnitud de dicho número real.** \triangle

1.5. Propagación de errores en punto flotante

En la Sección 1.4 discutimos la capacidad de la aritmética flotante de aproximar números reales y en la Sección 1.3 estudiamos cómo los errores se pueden ver afectados al evaluar funciones o realizar operaciones aritméticas en forma *exacta*. Aquí combinamos ambos temas: analizamos cómo se propagan los errores al utilizar aritmética de punto flotante.

1.5.1. Advertencias

La aritmética en punto flotante no es ni asociativa ni distributiva: las operaciones se hacen por etapas y en cada operación se aplica el redondeo correspondiente. Por lo tanto, colocar paréntesis puede alterar el resultado en punto flotante incluso cuando no lo hace en aritmética real. Recordando que en el formato de precisión doble $\varepsilon_M = 2^{-52} \approx 2,204 \times 10^{-16}$, veamos un ejemplo.

```
>> (1 + 1e-16) + 1e-16 - 1 [Enter]
ans = 0

>> 1 + (1e-16 + 1e-16) - 1 [Enter]
ans = 2.2204e-16

>> (1 + 1e-16) + (1e-16 - 1) [Enter]
ans = 1.1102e-16
```

¿Por qué obtenemos tres resultados diferentes³? En el primer caso, la operación entre paréntesis da como resultado 1 (porque $1 + 10^{-16}$ se redondea a 1), al sumarle a ese 1 el número 10^{-16} se vuelve a obtener 1, y al restar 1 se obtiene 0. En el segundo caso, primero se resuelve la suma entre paréntesis, cuyo resultado es $2.0000e-16$. Al sumar 1 más dicho número el resultado es $1+eps$ (porque estamos usando redondeo), y al restar 1 se obtiene como resultado final eps . Finalmente, en el tercer caso se resuelven las sumas entre paréntesis en primer lugar: el resultado de la primera es 1 y el de la segunda es $-1+0.5*eps$ (ver el ejercicio a continuación). Al sumar ambas, el resultado final es $0.5*eps$.

Ejercicio 1.4. Explicar por qué al computar $1e-16 - 1$ en Octave se obtiene el resultado $-1+0.5*eps$.

1.5.2. Análisis de error hacia adelante

Ahora combinamos ingredientes sobre propagación de errores (secciones 1.3.2 y 1.3.3) con lo que sabemos sobre aproximaciones de punto flotante (Sección 1.4.3). Como en la Sección 1.3.2, supongamos que queremos evaluar una función derivable $f: I \rightarrow \mathbb{R}$ en un cierto punto $x \in I$ tal que $y = f(x) \neq 0$ y que tenemos la capacidad de evaluar f exactamente en cualquier punto dado. En general, no tenemos acceso a x , sino a su representación de punto flotante $fl(x)$. Por la Proposición 1.4.1, sabemos que $\left| \frac{fl(x)-x}{x} \right| \leq \frac{\varepsilon_M}{2}$.

Lo mejor que podemos esperar es que computemos $\hat{y} = f(fl(x))$ *exactamente* y que luego approximemos \hat{y} usando punto flotante, esto es, que computemos $\bar{y} = fl(\hat{y}) = fl(f(fl(x)))$.

³Para hacer las cosas peor, los tres resultados son incorrectos en aritmética real!

De nuestra discusión sobre números de condición, sabemos que

$$\left| \frac{\hat{y} - y}{y} \right| \leq \kappa_f(x) \frac{\varepsilon_M}{2}.$$

Luego, usando la desigualdad triangular y que $y \approx \hat{y}$, deducimos

$$\left| \frac{\bar{y} - y}{y} \right| \leq \left| \frac{\text{fl}(\hat{y}) - \hat{y}}{y} \right| + \left| \frac{\hat{y} - y}{y} \right| \approx \left| \frac{\text{fl}(\hat{y}) - \hat{y}}{\hat{y}} \right| + \left| \frac{\hat{y} - y}{y} \right| \leq \frac{\varepsilon_M}{2} + \kappa_f(x) \frac{\varepsilon_M}{2}. \quad (1.7)$$

Al término $\frac{\varepsilon_M}{2} + \kappa_f(x) \frac{\varepsilon_M}{2}$ lo llamamos **error inevitable al computar $f(x)$** . Es la mejor cota que podemos tener para el error relativo en caso de que pudiésemos computar f exactamente. Para valorar la calidad de un algoritmo, más que utilizar como referencia el error en su salida, debemos comparar este error con el error inevitable. Si un algoritmo para computar $y = f(x)$ produce una salida \bar{y} con error relativo comparable con el error inevitable, diremos que el algoritmo es “bueno”. Esta no es una definición matemática, pero nos da una pauta de la calidad de un método dado. Veamos dos ejemplos.

Ejemplo 1.5.1 (algoritmo malo, algoritmo bueno). Queremos evaluar $y = 1 - \cos(x)$ en $x = 10^{-10}$. Estimamos el número de condición de $f(x) = 1 - \cos(x)$ en el punto 10^{-10} (ver (1.3)) usando que $\sin t \approx t$ y $1 - \cos t \approx t^2/2$ para $t \approx 0$,

$$\kappa_f(10^{-10}) = \left| \frac{-\sin(10^{-10}) 10^{-10}}{1 - \cos(10^{-10})} \right| \approx 2.$$

Por (1.7), esto implica que el error inevitable al computar $y = 1 - \cos(10^{-10})$ es aproximadamente $\frac{3}{2}\varepsilon_M$; en otras palabras, teóricamente debería ser posible computar y sin cometer un gran error relativo.

Para computar y , la estrategia que parece más obvia es la que haríamos si tuviésemos lápiz y papel: en primer lugar calcular $y_1 = \cos(10^{-10})$, y luego computar $y = 1 - y_1$. Esto se corresponde con componer $f_2 \circ f_1(10^{-10})$, donde

$$f_1(z) = \cos z, \quad f_2(z) = 1 - z.$$

Analicemos la mejor cota que podemos obtener para la magnitud del error relativo en este algoritmo. Razonando como hicimos para llegar a (1.7), tenemos

$$|\varepsilon_y| \leq \frac{\varepsilon_M}{2} + \kappa_{f_2}(y_1) |\varepsilon_{y_1}|,$$

y

$$|\varepsilon_{y_1}| \leq \frac{\varepsilon_M}{2} + \kappa_{f_1}(x) \frac{\varepsilon_M}{2}.$$

Aquí utilizamos que la magnitud del error relativo al aproximar x se puede acotar por $\frac{\varepsilon_M}{2}$, ya que x es la entrada de nuestro algoritmo. Combinando las dos expresiones, obtenemos

$$|\varepsilon_y| \leq \frac{\varepsilon_M}{2} + \kappa_{f_2}(y_1) \left(\frac{\varepsilon_M}{2} + \kappa_{f_1}(x) \frac{\varepsilon_M}{2} \right).$$

Por lo tanto, nos basta con estimar los números de condición $\kappa_{f_1}(x)$ y $\kappa_{f_2}(y_1)$. Usando nuevamente la definición (1.3) y las aproximaciones $\operatorname{sen}(10^{-10}) \approx 10^{-10}$ y $1 - y_1 = 1 - \cos(10^{-10}) \approx 10^{-20}/2$, tenemos

$$\begin{aligned}\kappa_{f_1}(x) &= \left| \frac{-\operatorname{sen}(10^{-10}) 10^{-10}}{\cos(10^{-10})} \right| \approx 10^{-20}, \\ \kappa_{f_2}(y_1) &= \left| \frac{y_1}{1 - y_1} \right| \approx 2 \times 10^{20}.\end{aligned}$$

Sustituyendo, obtenemos la cota

$$|\varepsilon_y| \leq \frac{\varepsilon_M}{2} + 2 \times 10^{20} \left(\frac{\varepsilon_M}{2} + 10^{-20} \frac{\varepsilon_M}{2} \right) \approx \varepsilon_M \times 10^{20}.$$

¡Esta cota es del orden de 10^{20} veces peor que el error inevitable! Esto nos da la pauta de que este es un mal algoritmo: si lo implementamos computacionalmente, estamos introduciendo cancelaciones catastróficas. Es muy sencillo verificarlo computacionalmente:

```
>> y = 1 - cos(1e-10) [Enter]
y = 0
```

Analicemos un algoritmo alternativo. Utilizamos la identidad trigonométrica

$$1 - \cos(x) = 2 \operatorname{sen}^2(x/2),$$

y computamos $y_1 = f_1(x) = x/2$, $y_2 = f_2(y_1) = \operatorname{sen}(y_1)$, $y_3 = f_3(y_2) = y_2^2$, $y_4 = f_4(y_3) = 2y_3$. Se verifica que (Ejercicio 1.5)

$$\kappa_{f_1}(x) = 1, \quad \kappa_{f_2}(y_1) \approx 1, \quad \kappa_{f_3}(y_2) = 2, \quad \kappa_{f_4}(y_3) = 1$$

y por lo tanto este algoritmo da lugar a

$$\begin{aligned}|\varepsilon_y| &\leq \frac{\varepsilon_M}{2} + \kappa_{f_4}(y_3) \left[\frac{\varepsilon_M}{2} + \kappa_{f_3}(y_2) \left(\frac{\varepsilon_M}{2} + \kappa_{f_2}(y_1) \left(\frac{\varepsilon_M}{2} + \kappa_{f_1}(x) \frac{\varepsilon_M}{2} \right) \right) \right] \\ &\approx \frac{\varepsilon_M}{2} + \left[\frac{\varepsilon_M}{2} + 2 \left(\frac{\varepsilon_M}{2} + \left(\frac{\varepsilon_M}{2} + \frac{\varepsilon_M}{2} \right) \right) \right] = 4\varepsilon_M.\end{aligned}$$

Esta cota para el error relativo es comparable con el error inevitable: este es un buen algoritmo. Notar que este segundo algoritmo evita restar números de igual magnitud. Confirmamos nuestra predicción teórica computacionalmente:

```
>> y = 2*sin(1e-10/2)^2 [Enter]
y = 5.0000e-21
```

△

Ejercicio 1.5. Completar los detalles de la estimación del error para el segundo algoritmo en el ejemplo anterior.

Ejemplo 1.5.2 (otro ejemplo de cancelaciones). Consideremos el polinomio

$$p(x) = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1. \quad (1.8)$$

Es fácil verificar que p tiene una raíz en $x = 1$. Corremos en Octave unas líneas de código que nos permitan graficarlo para $x \in (0,99, 1,01)$:

```
>> x = linspace(0.99, 1.01);
>> y = x.^7-7*x.^6+21*x.^5-35*x.^4+35*x.^3-21*x.^2+7*x-1;
>> plot(x,y)
```

El resultado se muestra a la izquierda en la Figura 1.1. El gráfico no se parece en nada al de un polinomio. Con nuestros conocimientos de cancelaciones catastróficas, podemos entender que evaluar computacionalmente la expresión (1.8) para $x \approx 1$ es una mala práctica. Como alternativa, podemos darnos cuenta de que se puede factorizar

$$p(x) = (x - 1)^7. \quad (1.9)$$

Usamos esta expresión en Octave y graficamos nuevamente

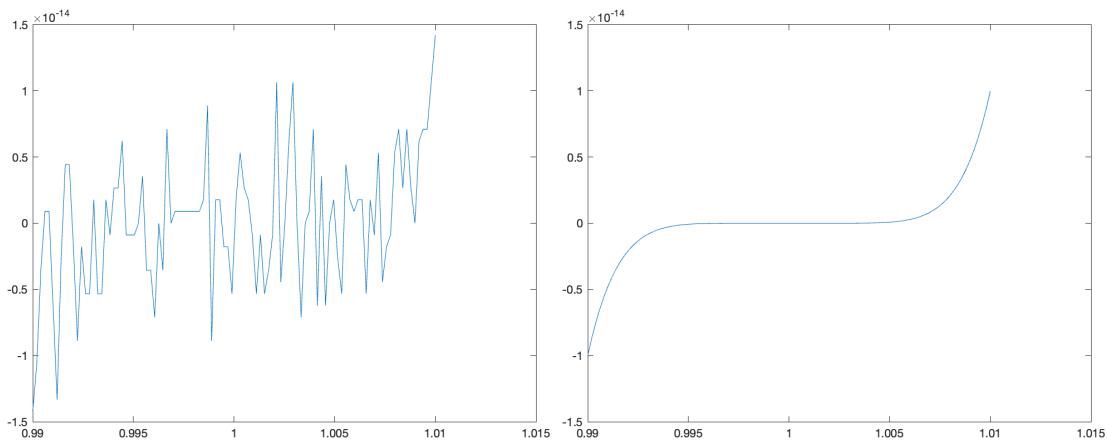
```
>> x = linspace(0.99, 1.01);
>> z = (x-1).^7;
>> plot(x,z)
```

El resultado se muestra a la derecha en la Figura 1.1, y notamos que ahora sí se parece a lo que esperamos del gráfico de un polinomio.

En teoría, las dos opciones que presentamos arriba nos permitirían graficar al polinomio p en el intervalo $(0,99, 1,01)$. Sin embargo, al evaluar $p(x)$ con x cercano a 1, la primera introduce cancelaciones que se pueden evitar si se factoriza a p . La segunda opción es computacionalmente mucho más robusta que la primera. \triangle

Ejercicio 1.6. Justificar la última afirmación del ejemplo anterior: realizar un análisis de error hacia adelante para computar $p(x)$ con x cercano a 1 usando ambos métodos.

Observación 1.5.1 (análisis de error hacia atrás). El análisis de error hacia adelante mide la discrepancia entre los valores computados y los valores exactos que queremos aproximar. Un enfoque alternativo para estimar el error consiste en, dado un algoritmo, preguntarse cuánta debería ser la discrepancia entre la entrada que usamos y la entrada exacta para que se produzca la salida aproximada que obtuvimos. Esta alternativa se suele llamar *análisis de error hacia atrás*, pero no profundizaremos en ella. Para quien esté interesado, referimos a [Hea02, Sección 1.2.5]. \triangle

Figura 1.1: Gráfico de $p(x)$ usando (1.8) (izquierda) y usando (1.9) (derecha).

1.5.3. Cálculo de derivadas usando diferencias hacia adelante

Concluimos este capítulo con una aplicación que ilustra cómo pueden interactuar dos tipos de errores distintos que discutimos en la Sección 1.1.1. Volviendo al Ejemplo 1.1.1, podemos pensar que estamos haciendo los últimos dos pasos que allí mencionamos: discretización e implementación.

Sea $f : \mathbb{R} \rightarrow \mathbb{R}$, de clase C^2 . Recordamos la definición de la derivada de f en $x \in \mathbb{R}$,

$$f'(x) = \lim_{h \rightarrow 0} \Delta_h f(x), \quad \text{donde} \quad \Delta_h f(x) := \frac{f(x + h) - f(x)}{h}.$$

Al número $\Delta_h f(x)$ se lo suele llamar *cociente incremental* (o *diferencia hacia adelante*) de f en x con paso h . Queremos evaluar $f'(x)$ numéricamente en un punto x en el que $f(x) \neq 0$, $f''(x) \neq 0$. Para simplificar nuestro análisis, vamos a suponer que podemos almacenar a x y a $x + h$ exactamente. Como queremos que nuestra máquina pueda distinguir $f(x)$ de $f(x + h)$, ya sabemos que no podemos tomar h arbitrariamente pequeño. Una estrategia razonable –y que ya propusimos en el Ejemplo 1.1.1– es aproximar

$$f'(x) \approx \Delta_h f(x), \tag{1.10}$$

para pasos $h > 0$ pequeños y fijos⁴. Si estuviésemos trabajando con aritmética real, entonces obviamente sería conveniente tomar h lo más pequeño posible. Sin embargo, si trabajamos con una máquina nos enfrentamos al riesgo de introducir cancelaciones catastróficas en el numerador: si $f(x) \neq 0$, hacer $f(x + h) - f(x)$ con h pequeño es justamente lo que en la Sección 1.3.3 dijimos que era peligroso computacionalmente. De hecho, más que trabajar con (1.10), en la práctica vamos a aproximar con

$$f'(x) \approx \frac{\text{fl}(f(x + h)) - \text{fl}(f(x))}{h}. \tag{1.11}$$

⁴Tomar $h < 0$ no reviste ninguna dificultad técnica adicional. Simplemente tomamos $h > 0$ para aliviar la notación.

Podríamos haber puesto $\text{fl}(h)$ en vez de h en el denominador, pero como no vamos a tener cancelaciones, la simplificación que hicimos no es relevante.

Estimemos la magnitud del error absoluto en la aproximación (1.11). Observamos que hay dos fuentes de error:

- El **error de truncamiento**, asociado a que en vez de tomar límite estamos dejando fijo un h pequeño. Esperamos que este error se reduzca cuanto menor sea h .
- El **error de redondeo**, asociado a que estamos trabajando con precisión finita (punto flotante). Esperamos que este error aumente cuanto menor sea h .

Una forma de cuantificar esta observación es utilizar una desigualdad triangular para separar el error absoluto como la suma de las componentes de truncamiento y de redondeo,

$$\begin{aligned} |e_{f'(x)}| &= \left| f'(x) - \frac{\text{fl}(f(x+h)) - \text{fl}(f(x))}{h} \right| \\ &\leq |f'(x) - \Delta_h f(x)| + \left| \Delta_h f(x) - \frac{\text{fl}(f(x+h)) - \text{fl}(f(x))}{h} \right| \\ &=: |\text{error}_{\text{truncamiento}}| + |\text{error}_{\text{redondeo}}|. \end{aligned} \quad (1.12)$$

Error de truncamiento

Comenzamos por acotar el primer término del lado derecho en (1.12). Este término tiene que ver exclusivamente con reemplazar la derivada por un cociente incremental. Para analizarlo, basta con utilizar un desarrollo de Taylor de primer orden en el punto x , usando la expresión de Lagrange para el resto,

$$f(x+h) = f(x) + f'(x)h + f''(c) \frac{h^2}{2} \quad \text{para algún } c \in [x, x+h].$$

Con esta expresión, podemos despejar y estimar

$$|\text{error}_{\text{truncamiento}}| = |f''(c)| \frac{h}{2}.$$

Como estamos asumiendo que f es de clase C^2 y $c \in [x, x+h]$, entonces esperamos que $f''(c) \approx f''(x)$. Obtenemos así una estimación para el error de truncamiento,

$$|\text{error}_{\text{truncamiento}}| \approx |f''(x)| \frac{h}{2}. \quad (1.13)$$

Deducimos que el error de truncamiento es de orden h : esperamos que decrezca linealmente al decrecer el paso h .

Error de punto flotante

Por la Proposición 1.4.1 y usando la expresión (1.2), sabemos que se cumple

$$\text{fl}(f(x+h)) = (1 + \varepsilon_{f(x+h)})f(x+h), \quad \text{fl}(f(x)) = (1 + \varepsilon_{f(x)})f(x),$$

con

$$|\varepsilon_{f(x+h)}| \leq \frac{\varepsilon_M}{2}, \quad |\varepsilon_{f(x)}| \leq \frac{\varepsilon_M}{2}.$$

Por lo tanto, podemos reemplazar $\text{fl}(f(x+h))$ y $\text{fl}(f(x))$ y utilizar la desigualdad triangular y las cotas para los errores relativos para obtener

$$\begin{aligned} |\text{error}_{\text{redondeo}}| &= \frac{|f(x+h)\varepsilon_{f(x+h)} - f(x)\varepsilon_{f(x)}|}{h} \\ &\leq \frac{|f(x+h)| |\varepsilon_{f(x+h)}| + |f(x)| |\varepsilon_{f(x)}|}{h} \\ &\leq \left(\frac{|f(x+h)| + |f(x)|}{h} \right) \frac{\varepsilon_M}{2}. \end{aligned}$$

Como f es continua y $f(x) \neq 0$, podemos suponer $f(x+h) \approx f(x)$, y concluimos que

$$|\text{error}_{\text{redondeo}}| \lesssim \frac{|f(x)| \varepsilon_M}{h}. \quad (1.14)$$

A diferencia del error de truncamiento, nuestra cota para este error es inversamente proporcional al paso h . Esto significa que un paso demasiado pequeño puede incrementar el error relacionado a la representación en punto flotante.

Paso óptimo

Tenemos entonces que, por un lado, se requiere un paso pequeño para minimizar el error de truncamiento, pero por otro lado un paso demasiado pequeño puede dar lugar a problemas de punto flotante. Una forma de encontrar el paso óptimo es combinar (1.13) y (1.14), sustituir en (1.12) para obtener

$$|e_{f'(x)}| \leq |f''(x)| \frac{h}{2} + \frac{|f(x)| \varepsilon_M}{h},$$

y buscar el valor $h_{\text{opt}} > 0$ en el que el lado derecho se hace lo menor posible. Esto se logra de forma sencilla derivando el lado derecho de la desigualdad respecto a h y despejando. Obtenemos

$$h_{\text{opt}} = \sqrt{\frac{2|f(x)| \varepsilon_M}{|f''(x)|}}, \quad (1.15)$$

lo que da lugar a una cota óptima para la magnitud del error absoluto,

$$|e_{f'(x),\text{opt}}| = \sqrt{2|f(x)||f''(x)| \varepsilon_M} \quad (1.16)$$

En la Figura 1.2 ilustramos cómo se comportan las dos componentes del error y su suma al variar h . Esta sencilla figura es ilustrativa del comportamiento cualitativo de los errores de truncamiento y de redondeo, pero en la práctica podemos ver algo un poco más interesante.

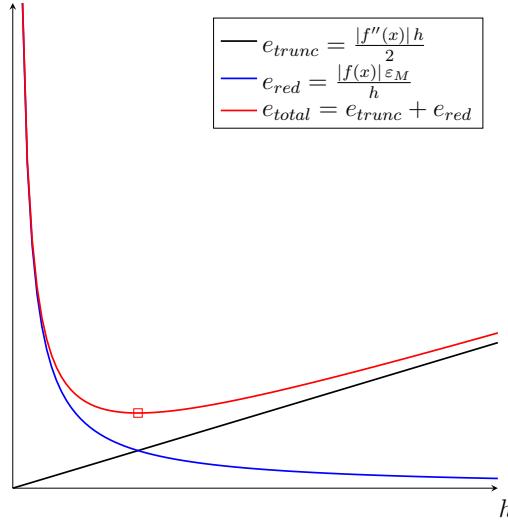


Figura 1.2: Cota superior del error total. Permite hallar h óptimo.

Ejemplo 1.5.3 (estimación del paso óptimo). Apliquemos la discusión anterior a aproximar la derivada de la función $f(x) = e^x$ en el punto $x = 1$. Como $f''(1) = f(1) = e$, las expresiones (1.15) y (1.16) nos indican que

$$h_{\text{opt}} = \sqrt{2\varepsilon_M} = 2^{-52/2} \approx 2,1 \times 10^{-8}, \quad |e_{f'(x),\text{opt}}| = h_{\text{opt}} e \approx 5,7 \times 10^{-8}.$$

Implementamos en Octave un código que tome varios h en el intervalo $[10^{-16}, 10^{-1}]$, calcule las aproximaciones (1.11) y estime el error sabiendo que $f'(x) = e$.

```

n = 1000;
derivadas = zeros(1,n);
h = 10.^(-linspace(1,16,n)); % toma n puntos entre 10^{-16} y 10^{-1}
f = @(x) exp(x);
x = 1;

for k = 1:n
    derivadas(k) = (f(x+h(k))-f(x))/(h(k));
end

errores = abs(derivadas - exp(1)*ones(1,n));
loglog(h,errores) % genera un grafico con ejes logaritmicos

```

La salida de este código se muestra en la Figura 1.3. Observamos que, para h grande, el error computado decrece linealmente al reducir h : esta es la región en la que domina el error de truncamiento. En cambio, para h pequeño el error se comporta de manera errática aunque muestra una tendencia a crecer al reducir h : en esta región domina el error de redondeo. La transición entre una región y la otra se da cuando h (y el error) son del orden de 10^{-8} , consistentemente con nuestras estimaciones de h_{opt} y $|e_{f'(x), \text{opt}}|$.

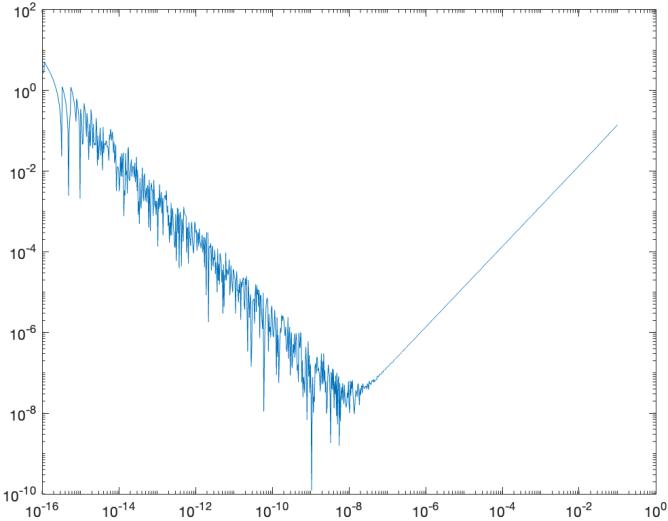


Figura 1.3: Error computacional en el Ejemplo 1.5.3.

△

Ejercicio 1.7. En la Figura 1.3 se observa que para $h \approx 10^{-16} \approx \varepsilon_M$ el error absoluto es del orden de una unidad. Estimar el error de redondeo para $h = 10^{-16}$.

Capítulo 2

Sistemas de ecuaciones lineales

2.1. Introducción

Un sistema de m ecuaciones lineales y n incógnitas consiste en un conjunto de relaciones algebraicas de la forma

$$\sum_{j=1}^n a_{ij} x_j = b_i \quad \forall i = 1 \dots m,$$

que puede ser representado en notación matricial como $A\mathbf{x} = \mathbf{b}$:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \in \mathcal{M}_{m \times n}(\mathbb{R}), \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \in \mathbb{R}^m.$$

Cuando $m = n$, decimos que el sistema es *cuadrado*; en ese caso, el sistema tiene solución única si y sólo si $\det A \neq 0$. En este capítulo trabajamos con sistemas de este tipo, a los que llamamos *compatibles determinados*.

Asumiendo que el sistema $A\mathbf{x} = \mathbf{b}$ tiene solución única, nos planteamos cómo hallarla. En general, los métodos de resolución de sistemas lineales pueden ser clasificados en las siguientes dos categorías.

- Aquellos métodos que, si tuviésemos precisión infinita, nos permitirían obtener la solución exacta luego de una cantidad finita de pasos. Estos métodos se dicen **directos**.
- Aquellos métodos que, en lugar de permitirnos obtener la solución exacta \mathbf{x} en una cantidad finita de pasos, nos arrojan una sucesión de vectores $\{\mathbf{x}^k\} \subset \mathbb{R}^n$ con $\mathbf{x}^k \rightarrow \mathbf{x}$ con $k \rightarrow \infty$. Estos métodos se dicen **indirectos**.

Comenzamos mostrando un primer ejemplo de método directo, que ya hemos encontrado en el curso de Geometría y Álgebra Lineal 1.

Ejemplo 2.1.1 (regla de Cramer). Dado el sistema $A\mathbf{x} = \mathbf{b}$, la regla de Cramer devuelve cada entrada del vector solución solamente como un cociente de determinantes que dependen de la matriz A y del vector \mathbf{b} . En efecto, se tiene

$$x_i = \frac{\det(A_i)}{\det(A)},$$

donde A_i es la matriz resultante al reemplazar la columna i -ésima de la matriz A por el vector \mathbf{b} .

Por ejemplo, para resolver el sistema $A\mathbf{x} = \mathbf{b}$ con

$$A = \begin{bmatrix} 3 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{y} \quad \mathbf{b} = \begin{bmatrix} 5 \\ -1 \end{bmatrix},$$

computamos los determinantes

$$\det(A) = -4, \quad \det(A_1) = \begin{vmatrix} 5 & 1 \\ -1 & -1 \end{vmatrix} = -4, \quad \det(A_2) = \begin{vmatrix} 3 & 5 \\ 1 & -1 \end{vmatrix} = -8,$$

y tenemos por lo tanto $\mathbf{x} = [1 \ 2]^t$. △

Al analizar el costo de la ejecución de algoritmos, muchas veces es relevante contar la cantidad de operaciones que se realizan. Con este fin, introducimos la siguiente noción.

Definición 2.1.1 (*flop*¹). Denotaremos por *flop* a una simple operación de punto flotante, como suma, resta, producto o división. △

En general, no nos interesa la cantidad exacta de *flops* que requiere un método, sino el *orden de magnitud* de esta cantidad. Retomando el Ejemplo 2.1.1, supongamos que usamos la regla de Cramer para resolver un sistema con $A \in \mathcal{M}_n(\mathbb{R})$. Calcular el determinante de una matriz $n \times n$ desarrollando por filas o columnas requiere $O(n!)$ operaciones, y la regla de Cramer requiere que calculemos $n + 1$ determinantes de matrices $n \times n$ y los dividamos n veces. Deducimos que una aplicación ingenua de la regla de Cramer para resolver un sistema de $n \times n$ tiene un costo computacional de $O((n + 1)!) \textit{flops}$. Esto es pésimo computacionalmente. Incluso aunque sea posible calcular determinantes de forma eficiente (ver Observación 2.2.2 más abajo), es poco común usar la regla de Cramer para resolver sistemas grandes de ecuaciones lineales. No vamos a profundizar en este algoritmo.

En la Sección 2.2 vamos a analizar con detalle otro método directo ya conocido de cursos anteriores: el algoritmo de escalerización gaussiana. Luego, la Sección 2.3 profundiza sobre la descomposición *LU*, que está cercanamente relacionada con dicho algoritmo. Si bien la

¹La palabra *flop* proviene del inglés *floating point operations*.

descomposición gaussiana permite tratar con sistemas $A\mathbf{x} = \mathbf{b}$ solamente asumiendo que A es invertible y se pueden probar varios resultados en general, éstos se pueden mejorar si la matriz A posee cierta estructura adicional. Comentamos brevemente sobre este tema en la Sección 2.4. La Sección 2.5 estudia la estabilidad y convergencia del algoritmo de escalerización gaussiana; esta sección también incluye material sobre normas vectoriales y matriciales, y el importante concepto de *número de condición*.

2.2. Escalerización gaussiana

El método de escalerización gaussiana es un método directo para resolución de sistemas de ecuaciones lineales. Este algoritmo nos permite llevar un sistema $A\mathbf{x} = \mathbf{b}$, con A cuadrada y no singular, a uno equivalente de tipo triangular superior. En esta sección, en primer lugar analizamos cómo resolver sistemas triangulares de forma directa. Luego, repasamos el algoritmo de escalerización gaussiana, sin y con pivoteo, y mostramos por qué computacionalmente pivotear es importante, incluso en casos en los que en aritmética real no sería necesario.

2.2.1. Sistemas triangulares

Comenzamos por abordar un caso particularmente simple de sistemas lineales. Recordamos que una matriz $A = (a_{ij}) \in \mathcal{M}_n(\mathbb{R})$ se dice **triangular superior** si

$$a_{ij} = 0 \quad \text{para todo } i, j = 1, \dots, n, \text{ con } j < i.$$

Análogamente, decimos que A es **triangular inferior** si

$$a_{ij} = 0 \quad \text{para todo } i, j = 1, \dots, n, \text{ con } j > i.$$

Consideremos un sistema $A\mathbf{x} = \mathbf{b}$ con $A \in \mathcal{M}_n(\mathbb{R})$ triangular inferior,

$$A = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

En forma de sistema de ecuaciones, tenemos

$$\left\{ \begin{array}{lcl} a_{11}x_1 & = b_1 \\ a_{12}x_1 + a_{22}x_2 & = b_2 \\ \dots & & \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n & = b_n \end{array} \right. . \quad (2.1)$$

Como estamos asumiendo que este sistema tiene solución única, necesariamente tiene que ser $\det(A) \neq 0$. Al ser A triangular inferior, es fácil verificar que

$$\det(A) = a_{11}a_{22}\dots a_{nn},$$

de modo que $a_{ii} \neq 0$ para todo $i = 1, \dots, n$. Para resolver este sistema, parece natural comenzar por la primera ecuación en (2.1):

$$a_{11}x_1 = b_1 \quad \Rightarrow \quad x_1 = \frac{b_1}{a_{11}}.$$

Luego, podemos usar el valor de x_1 que hallamos para continuar con la segunda ecuación en (2.1),

$$a_{21}x_1 + a_{22}x_2 = b_2 \quad \Rightarrow \quad x_2 = \frac{b_2 - a_{21}x_1}{a_{22}}.$$

Podemos continuar sucesivamente: si conocemos x_1, \dots, x_{i-1} , entonces podemos despejar x_i de la i -ésima ecuación en (2.1) mediante

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j}{a_{ii}}.$$

Este método para resolver sistemas triangulares inferiores se denomina **sustitución hacia adelante**. En el Algoritmo 2.1, lo presentamos en formato de pseudo-código.

Algoritmo 2.1: Pseudo-código: sustitución hacia adelante

Datos: $A = (a_{ij}) \in \mathcal{M}_n(\mathbb{R})$ triangular inferior, $\mathbf{b} \in \mathbb{R}^n$

Resultado: $\mathbf{x} \in \mathbb{R}^n$ solución de $A\mathbf{x} = \mathbf{b}$

$x_1 \leftarrow \frac{b_1}{a_{11}}$;

para $i = 2 : n$ **hacer**

$x_i \leftarrow \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij} x_j)$;

fin

Análogamente, los sistemas de la forma $A\mathbf{x} = \mathbf{b}$ con $A \in \mathcal{M}_n(\mathbb{R})$ triangular superior se pueden resolver de forma sencilla realizando una **sustitución hacia atrás**. Dicho método consiste en comenzar por la última ecuación para hallar x_n ,

$$x_n = \frac{b_n}{a_{nn}},$$

y para $i = n-1, \dots, 1$ calcular

$$x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij} x_j}{a_{ii}}.$$

Algoritmo 2.2: Pseudo-código: sustitución hacia atrás

Datos: $A = (a_{ij}) \in \mathcal{M}_n(\mathbb{R})$ triangular superior, $\mathbf{b} \in \mathbb{R}^n$

Resultado: $\mathbf{x} \in \mathbb{R}^n$ solución de $A\mathbf{x} = \mathbf{b}$

$$x_n \leftarrow \frac{b_n}{a_{nn}};$$

para $i = n - 1 : -1 : 1$ **hacer**

$$\quad | \quad x_i \leftarrow \frac{1}{a_{ii}} (b_i - \sum_{j=i+1}^n a_{ij} x_j);$$

fin

Observación 2.2.1 (costo computacional de sustitución hacia adelante o hacia atrás). Es claro que los costos computacionales de hacer sustitución hacia adelante o hacia atrás son iguales. Veamos con detalle el costo de hacer sustitución hacia adelante: para computar x_i ($i = 1, \dots, n$) necesitamos hacer $i - 1$ productos, $i - 1$ sumas y una división. Por lo tanto, en total necesitamos hacer

$$\sum_{i=1}^n (i - 1 + i - 1 + 1) = \sum_{i=1}^n (2i - 1) = n^2$$

operaciones. Deducimos que el costo computacional de resolver un sistema $n \times n$ triangular usando sustitución hacia adelante/atrás es de $O(n^2)$ flops. \triangle

2.2.2. Escalerización gaussiana (sin pivoteo)

Sean $A \in \mathcal{M}_n(\mathbb{R})$ una matriz no singular y $\mathbf{b} \in \mathbb{R}^n$. Para llevar el sistema $A\mathbf{x} = \mathbf{b}$ a uno equivalente de tipo triangular superior, el algoritmo de escalerización gaussiana consiste en realizar ciertas *operaciones elementales* en forma sucesiva. Dichas operaciones elementales consisten en intercambiar filas o reemplazar una fila dada por una combinación lineal entre filas en la que la fila reemplazada tiene un multiplicador no nulo.

Recordamos en qué consiste el método de escalerización gaussiana. Para $k = 1, \dots, n$, en el paso k se tiene una matriz semi-escalerizada y con estructura

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1k}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2k}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix}$$

Si $a_{kk}^{(k)} \neq 0$, lo llamamos *pivot* del paso k ; por el momento, asumimos que en todos los pasos de nuestro algoritmo tenemos $a_{kk}^{(k)} \neq 0$. Notamos que la matriz A tiene estructura triangular superior en sus primeras k filas, y el resto de la matriz es llena. Buscamos llevar a 0 los elementos de la columna k -ésima que están por debajo de la diagonal. Para

ese fin, para cada $i = k + 1, \dots, n$ definimos el coeficiente $l_{ik} := \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$, al que llamamos *multiplicador*, y actualizamos la fila i -ésima de la matriz $A^{(k)}$ y la entrada i -ésima del término independiente $b^{(k)}$ mediante

$$\begin{aligned} a_{ij}^{(k+1)} &:= a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}, & \forall j = k, \dots, n. \\ b_i^{(k+1)} &:= b_i^{(k)} - l_{ik} b_k^{(k)}, \end{aligned} \quad (2.2)$$

Este procedimiento puede escribirse en la forma de pseudo-código que mostramos en el Algoritmo 2.3.

Algoritmo 2.3: Pseudo-código: EG sin pivoteo

Datos: $A = (a_{ij}) \in \mathcal{M}_n(\mathbb{R})$, $\mathbf{b} \in \mathbb{R}^n$

Resultado: $A = (a_{ij}) \in \mathcal{M}_n(\mathbb{R})$ triangular superior, $L = (l_{ik}) \in \mathcal{M}_n(\mathbb{R})$ triangular inferior y que tiene los multiplicadores del algoritmo de escalerización gaussiana sin pivoteo

para $k = 1 : n - 1$ **hacer**

para $i = k + 1 : n$ **hacer**

$l(i, k) \leftarrow a(i, k) / a(k, k);$

$a(i, k) \leftarrow 0;$

para $j = k + 1 : n$ **hacer**

$a(i, j) \leftarrow a(i, j) - l(i, k) * a(k, j);$

fin

$b_i \leftarrow b_i - l(i, k) * b_k;$

fin

fin

Estimemos el costo computacional de usar escalerización gaussiana para llevar la matriz A a una triangular superior mediante operaciones elementales. Contamos los *flops* de acuerdo a los *loops* en el Algoritmo 2.3:

- *loop* en j : $2(n - k)$ *flops*,
- *loop* en i : $(2(n - k) + 3)(n - k) = 2(n - k)^2 + 3(n - k)$ *flops*,
- *loop* en k : $\sum_{k=1}^{n-1} [2(n - k)^2 + 3(n - k)]$ *flops*.

Agrupando según k y usando que

$$\sum_{k=1}^{n-1} k = \frac{(n-1)n}{2}, \quad \sum_{k=1}^{n-1} k^2 = \frac{(n-1)n(2n-1)}{6},$$

obtenemos:

$$\begin{aligned}
 flops(EG) &= \sum_{k=1}^{n-1} (2n^2 - 4nk + 2k^2 + 3n - 3k) \\
 &= (2n^2 + 3n)(n-1) - (4n+3)\sum_{k=1}^{n-1} k + 2\sum_{k=1}^{n-1} k^2 \\
 &= (2n^2 + 3n)(n-1) - (4n+3)\frac{(n-1)n}{2} + 2\frac{(n-1)n(2n-1)}{6} = O\left(\frac{2n^3}{3}\right).
 \end{aligned}$$

Por lo tanto, si resolvemos un sistema cuadrado $n \times n$ usando escalerización gaussiana y sustitución hacia atrás, requerimos

$$O\left(\frac{2n^3}{3}\right) + O(n^2) = O\left(\frac{2n^3}{3}\right) \text{ flops.}$$

Observación 2.2.2 (bajando el costo de calcular determinantes). Al hacer escalerización gaussiana sin pivoteo, el determinante de A permanece inalterado. La matriz resultante al terminar la escalerización es triangular superior, y por lo tanto se puede calcular su determinante multiplicando todos los elementos en la diagonal. Como en la diagonal hay n elementos, y por lo tanto tenemos que hacer n multiplicaciones para calcular el determinante de una matriz triangular, este procedimiento para calcular el determinante de una matriz requiere de $O\left(\frac{2n^3}{3}\right)$ flops. \triangle

2.2.3. Escalerización gaussiana con pivoteo parcial

En la sección anterior asumimos que en cada paso de la escalerización gaussiana se tuvo $a_{kk}^{(k)} \neq 0$. Esto es necesario para que esté bien definido el multiplicador $l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$, que es el factor por el que multiplicamos a la fila k -ésima en las combinaciones lineales que hacemos en el paso k . Sabemos que la condición $a_{kk}^{(k)} \neq 0$ no es necesaria para que el sistema tenga solución única: en caso de que $a_{kk}^{(k)} = 0$ podemos intercambiar la fila k -ésima por una fila que esté más abajo y seguir con nuestro algoritmo de escalerización gaussiana. Este paso de intercambiar filas se llama *pivoteo*.

¿Por qué pivotear?

Desde el punto de vista de la aritmética real, esto es, si tuviésemos precisión infinita, pivotear sólo es necesario cuando se llega a que $a_{kk}^{(k)} = 0$ para algún paso k . Cuando trabajamos con aritmética de punto flotante, pivotear se vuelve importante cuando $a_{kk}^{(k)} \approx 0$. Ilustramos esta idea con un ejemplo.

Ejemplo 2.2.1 (el peligro de no pivotear). Queremos resolver el sistema $A\mathbf{x} = \mathbf{b}$, donde

$$A = \begin{bmatrix} 10 & -7 & 0 \\ -3 & 2,099 & 6 \\ 5 & -1 & 5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 7 \\ 3,901 \\ 6 \end{bmatrix},$$

pero tenemos una máquina que solamente nos permite computar 5 dígitos significativos y usamos redondeo. Es sencillo verificar que el sistema tiene como solución única $\mathbf{x} = [0 \ -1 \ 1]^t$. Como tenemos $a_{11}^{(1)} = 10$, podemos calcular los multiplicadores $l_{21} = -\frac{3}{10} = -0,3$, $l_{31} = \frac{5}{10} = 0,5$ y reemplazamos

$$\text{fila}_2 \leftarrow \text{fila}_2 - l_{21} * \text{fila}_1, \quad \text{fila}_3 \leftarrow \text{fila}_3 - l_{31} * \text{fila}_1,$$

lo que nos da el sistema ampliado

$$[A^{(2)} \mid \mathbf{b}^{(2)}] = \left[\begin{array}{ccc|c} 10 & -7 & 0 & 7 \\ 0 & -0,001 & 6 & 6,001 \\ 0 & 2,5 & 5 & 2,5 \end{array} \right]. \quad (2.3)$$

Ahora, nos encontramos con que $a_{22}^{(2)} = -0,001$. Como es distinto de cero y no somos conscientes del peligro en el que nos estamos metiendo, usamos este elemento como pivote y computamos el multiplicador $l_{32} = -\frac{2,5}{0,001} = -2500$. Al computar

$$\text{fila}_3 \leftarrow \text{fila}_3 - l_{32} * \text{fila}_2,$$

nos encontramos con una limitación de máquina: tenemos $a_{33}^{(3)} = 5 + 2500 * 6 = 15005$ pero para calcular $b^{(3)}$ tenemos que hacer

$$2500 * 6,001 = 15002,5 \mapsto 15002 \Rightarrow b^{(3)} = 2,5 + 15002 = 15004,5 \mapsto 15004.$$

El sistema ampliado luego de este paso es

$$[A^{(3)} \mid \mathbf{b}^{(3)}] = \left[\begin{array}{ccc|c} 10 & -7 & 0 & 7 \\ 0 & -0,001 & 6 & 6,001 \\ 0 & 0 & 15005 & 15004 \end{array} \right].$$

Como ya quedó escalerizado, podemos realizar una sustitución hacia atrás:

$$\begin{aligned} 15005x_3 &= 15004 \Rightarrow x_3 = \frac{15004}{15005} \mapsto 0,99993 \text{ (nada mal)}, \\ -0,001x_2 + 6x_3 &= 6,001 \Rightarrow x_2 = \frac{6,001 - 5,9996}{-0,001} = -1,4 \text{ (bastante mal)}, \\ 10x_1 - 7x_2 &= 7 \Rightarrow x_1 = \frac{7 - 9,8}{10} = -0,28 \text{ (un desastre)}. \end{aligned}$$

Vale la pena comparar la solución exacta del problema, $[0, \ -1, \ 1]^t$, con la que obtuvimos, $[-0,28, \ -1,4, \ 0,9993]^t$: comenzamos con un muy pequeño error en x_3 , que se propagó peligrosamente al realizar la sustitución hacia atrás. \triangle

¿Dónde estuvo el problema en el ejemplo anterior? En que al tomar pivotes pequeños, se da lugar a multiplicadores grandes. A su vez, al realizar las combinaciones lineales entre filas, multiplicadores grandes dan lugar ecuaciones con coeficientes grandes respecto a las entradas de la matriz original. Esto hace que pequeños errores relativos de redondeo sean grandes en términos absolutos y abre la posibilidad de introducir cancelaciones catastróficas, tal como nos ocurrió al despejar x_2 en el Ejemplo 2.2.1.

Una forma habitual de mitigar este problema de propagación de errores es **asegurar que los multiplicadores sean menores o iguales que 1 (en valor absoluto)**. Esto se logra, por ejemplo, con la estrategia de *pivoteo parcial*.

Pivoteo parcial. En el paso k -ésimo de la escalerización, se toma como pivote al mayor elemento (en valor absoluto) de la parte no reducida de la columna k -ésima de la matriz A , esto es,

$$\arg \max_{r \in \{k, \dots, n\}} |a_{rk}^{(k)}| = p \Rightarrow \text{pivote: } a_{pk}$$

Una vez hallado el nuevo pivote, se intercambia la fila en la que está con la fila k -ésima en la matriz ampliada y se continúa con la escalerización gaussiana.

Algoritmo 2.4: Pseudo-código: EG con pivoteo parcial

Datos: $A = (a_{ij}) \in \mathcal{M}_n(\mathbb{R})$, $\mathbf{b} \in \mathbb{R}^n$
Resultado: $A = (a_{ij}) \in \mathcal{M}_n(\mathbb{R})$ triangular superior, $L = (l_{ik}) \in \mathcal{M}_n(\mathbb{R})$ triangular inferior y que tiene los multiplicadores del algoritmo de escalerización gaussiana sin pivotear

```

para  $k = 1 : n - 1$  hacer
     $p \leftarrow \arg \max_{r \in \{k, \dots, n\}} |a_{rk}|$ 
    intercambiar( $b_k, b_p$ );
    para  $j = 1 : n$  hacer
        | intercambiar( $a_{kj}, a_{pj}$ );
    fin
    para  $i = k + 1 : n$  hacer
        |  $l(i, k) \leftarrow a(i, k) / a(k, k)$ ;
        |  $a(i, k) \leftarrow 0$ ;
        | para  $j = k + 1 : n$  hacer
            | |  $a(i, j) \leftarrow a(i, j) - l(i, k) * a(k, j)$ ;
        | fin
        |  $b_i \leftarrow b_i - l(i, k) * b_k$ ;
    fin
fin
```

Es evidente que la estrategia de escalerización gaussiana con pivoteo parcial requiere más operaciones que la de escalerización gaussiana sin pivotear. En el paso k -ésimo, estamos agregando k operaciones de comparar los elementos $|a_{rk}^{(k)}|$, $r = k, \dots, n$. Por lo tanto, el pivoteo parcial agrega $O(n^2)$ operaciones, lo que es de menor orden que el costo

computacional de realizar una escalerización gaussiana. Deducimos, pues, que el costo computacional de realizar escalerización gaussiana con pivoteo parcial y sustitución hacia atrás para resolver un sistema $n \times n$ de ecuaciones lineales es de

$$O\left(\frac{2n^3}{3}\right) \textit{flops}.$$

Ejercicio 2.1. Rehacer el Ejemplo 2.2.1, pero utilizando una estrategia de pivoteo parcial: en (2.3) intercambiar la segunda y tercera filas y seguir adelante con el algoritmo. Mostrar que usando la misma máquina con 5 dígitos de precisión el resultado obtenido es exacto.

2.3. Descomposición LU

En la Sección 2.2 analizamos el algoritmo de escalerización gaussiana (sin pivoteo y con pivoteo parcial) para la resolución de sistemas de ecuaciones lineales $A\mathbf{x} = \mathbf{b}$ con $A \in \mathcal{M}_n(\mathbb{R})$. Una consecuencia directa de este algoritmo es que nos da una forma interesante de *factorizar* la matriz A como producto de una matriz triangular inferior por una triangular superior, a menos de permutaciones. En esta sección profundizamos en este hecho; concretamente, demostramos el siguiente resultado.

Teorema 2.3.1 (descomposición LU). *Sea $A \in \mathcal{M}_n(\mathbb{R})$ no singular. Entonces, existen*

- una matriz de permutación P ,
- una matriz triangular inferior L , cuya diagonal está formada por unos, y
- una matriz triangular superior U ,

tales que vale la identidad $PA = LU$.

En el teorema anterior aparece un término que no definimos aún, que es el de *matriz de permutación*.

Definición 2.3.1 (matriz de permutación). Una matriz $P \in \mathcal{M}_n(\mathbb{R})$ se dice **de permutación** si P se puede obtener intercambiando filas o columnas de la matriz identidad $I \in \mathcal{M}_n(\mathbb{R})$. \triangle

Observación 2.3.1 (intercambios de filas o columnas). Sean $P \in \mathcal{M}_n(\mathbb{R})$ una matriz de permutación y $A \in \mathcal{M}_n(\mathbb{R})$. Es fácil verificar que PA permite intercambiar filas de la matriz A , mientras que AP intercambia columnas de A . Como ejercicio, en caso de no ser claro para el lector, se puede comprobar este hecho tomando una matriz $A \in \mathcal{M}_2(\mathbb{R})$ genérica y $P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

Computacionalmente, no es eficiente almacenar una matriz de permutación como una matriz llena, ya que sabemos que sus elementos son todos 0 o 1, y que en cada fila/columna

se tiene exactamente un único 1. Es más eficiente utilizar *vectores de permutación*, que indican en qué orden se deben intercambiar las filas o columnas de la identidad. Por ejemplo, consideremos

$$P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad p = [4 \ 2 \ 1 \ 3], \quad q = [3 \ 2 \ 4 \ 1].$$

Entonces, siguiendo con la Observación 2.3.1, dada una matriz $A \in \mathcal{M}_n(\mathbb{R})$, para calcular $B = PA$ o $C = AP$ en Octave, simplemente podemos escribir

```
>> B = A(p,:);
>> C = A(:,q);
```

△

La descomposición $PA = LU$ del Teorema 2.3.1 está íntimamente relacionada con el algoritmo de escalerización gaussiana. Para demostrar el teorema, escribimos la operación elemental “reemplazar una fila dada por una combinación lineal entre filas en la que la fila reemplazada tiene un multiplicador no nulo” en forma matricial.

Definición 2.3.2 (matriz de multiplicadores). Una **matriz de multiplicadores** es una matriz $M = (m_{ij}) \in \mathcal{M}_n(\mathbb{R})$ triangular inferior, con unos en la diagonal, que debajo de la diagonal solamente tiene elementos no nulos en una única columna y que cumple $|m_{ij}| \leq 1$ para todos i, j . △

Ejemplo 2.3.1 (producto por una matriz de multiplicadores). Sean

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ -1/3 & 0 & 1 \end{bmatrix}, \quad A = \begin{bmatrix} 6 & -6 & 6 \\ -3 & 2 & 1 \\ 2 & 0 & -1 \end{bmatrix}.$$

Entonces, el producto MA transforma a A en la matriz que

$$\text{fila}_2 \leftarrow \text{fila}_2 + 1/2 * \text{fila}_1, \quad \text{fila}_3 \leftarrow \text{fila}_3 - 1/3 * \text{fila}_1,$$

esto es,

$$MA = \begin{bmatrix} 6 & -6 & 6 \\ 0 & -1 & 4 \\ 0 & 2 & -3 \end{bmatrix}.$$

△

El ejemplo anterior ilustra cómo entran las matrices multiplicadoras en el algoritmo de escalerización gaussiana: usando la notación de la Sección 2.2.2, si en el paso k -ésimo

escribimos la matriz multiplicadora M_k que se obtiene “colgando” los opuestos de los coeficientes $l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$,

$$M_k = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & \ddots & \vdots & \dots & \dots & \vdots \\ \vdots & \dots & 1 & \dots & \dots & 0 \\ 0 & \dots & -l_{(k+1),k} & 1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \ddots & \vdots \\ 0 & \dots & -l_{nk} & 0 & \dots & 1 \end{bmatrix}, \quad (2.4)$$

entonces la operación (2.2) sobre la matriz A se puede escribir en forma compacta como realizar el producto $M_k A$.

Bosquejo de demostración del Teorema 2.3.1. Dada la matriz $A \in \mathcal{M}_n(\mathbb{R})$, escribimos las operaciones elementales asociadas al algoritmo de escalerización gaussiana en el paso k -ésimo:

- permutar la fila k -ésima con la fila j -ésima ($j > k$) se corresponde con multiplicar a izquierda por una matriz de permutación P_k , que es la matriz identidad con las filas k y j intercambiadas²;
- reemplazar una fila dada por una combinación lineal se corresponde con multiplicar a izquierda por una matriz de multiplicadores M_k .

Además, sabemos que al final del algoritmo de escalerización gaussiana se llega a una matriz triangular superior, a la que llamaremos U . Así, sabemos que existen P_1, \dots, P_{n-1} y M_1, \dots, M_{n-1} tales que

$$M_{n-1}P_{n-1}M_{n-2}P_{n-2}\dots M_2P_2M_1P_1A = U.$$

Usando el Ejercicio 2.2 (ver abajo) sucesivamente, la identidad de arriba se puede reescribir como

$$M_{n-1}\tilde{M}_{n-2}\dots\tilde{M}_1P_{n-1}P_{n-2}\dots P_1A = U,$$

donde $\tilde{M}_1, \dots, \tilde{M}_{n-2}$ son matrices de multiplicadores. Finalmente, definimos

$$P := P_{n-1}P_{n-2}\dots P_1, \quad L := \left(M_{n-1}\tilde{M}_{n-2}\dots\tilde{M}_1\right)^{-1}.$$

Tenemos que P es una matriz de permutación, que L es triangular inferior con unos en la diagonal (ver el Ejercicio 2.3 abajo) y se cumple la identidad deseada,

$$L^{-1}PA = U \Rightarrow PA = LU.$$

□

²Observar que si en el paso k -ésimo no se hace pivoteo, entonces se puede tomar $P_k = Id$.

Ejercicio 2.2. Sea $P \in \mathcal{M}_n(\mathbb{R})$ de permutación que se obtiene al permutar las filas i y j de la matriz identidad y sea $M_k \in \mathcal{M}_n(\mathbb{R})$ de multiplicadores con términos no nulos debajo de la diagonal en la columna k , con $k < i, j$. Demostrar que existe $\tilde{M} \in \mathcal{M}_n(\mathbb{R})$ de multiplicadores tal que $PM = \tilde{M}P$. Concretamente, demostrar que \tilde{M} es la matriz que se obtiene al intercambiar los elementos m_{ik} y m_{jk} de la matriz M .

Ejercicio 2.3. Demostrar que el producto de matrices de permutación es de permutación. Demostrar que el producto de matrices triangulares inferiores con unos en la diagonal es triangular inferior y con unos en la diagonal. Demostrar que la inversa de una matriz triangular inferior con unos en la diagonal es triangular inferior y con unos en la diagonal.

Observación 2.3.2 (costo computacional de la factorización LU). Como realizar el algoritmo de escalerización gaussiana tiene un costo de $O\left(\frac{2n^3}{3}\right)$ flops, deducimos que el costo de computar una descomposición LU de una matriz A es del mismo orden. \triangle

Observación 2.3.3 (uso de la descomposición LU para resolver un sistema). Supongamos que queremos resolver un sistema de la forma $A\mathbf{x} = \mathbf{b}$, y ya conocemos una descomposición de la matriz A como en el Teorema 2.3.1, $PA = LU$. Multiplicando nuestro sistema a izquierda por P , obtenemos

$$LU\mathbf{x} = P\mathbf{b}.$$

Llamamos $\mathbf{b}' := P\mathbf{b}$ e $\mathbf{y} := U\mathbf{x}$. El sistema de arriba se puede pensar como la composición de dos sistemas triangulares: por un lado el sistema $L\mathbf{y} = \mathbf{b}'$, y por el otro el sistema $U\mathbf{x} = \mathbf{y}$. Uno puede primero hallar \mathbf{y} , y luego utilizar \mathbf{y} para hallar \mathbf{x} . Como los dos sistemas involucrados son triangulares, se los puede resolver usando sustitución hacia adelante y sustitución hacia atrás, respectivamente. El costo de cada una de las resoluciones es de $O(n^2)$ flops, por lo que contar con una descomposición LU de la matriz A nos permite resolver el sistema $A\mathbf{x} = \mathbf{b}$ con $O(n^2)$ flops. \triangle

Observación 2.3.4 (resolver muchos sistemas con una misma matriz). En algunas aplicaciones, podemos encontrarnos con la necesidad de resolver muchos sistemas en los que aparece la misma matriz $A \in \mathcal{M}_n(\mathbb{R})$, esto es, para vectores $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(m)} \in \mathbb{R}^n$, tenemos que hallar $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)} \in \mathbb{R}^n$ tales que

$$A\mathbf{x}^{(1)} = \mathbf{b}^{(1)}, \dots, A\mathbf{x}^{(m)} = \mathbf{b}^{(m)}.$$

Una forma de lograr esto es aplicar escalerización gaussiana m veces; el costo computacional de este procedimiento es de $O\left(\frac{2mn^3}{3}\right)$ flops. Notamos que el tratamiento sobre la matriz A en cada algoritmo de escalerización es el mismo, por lo que hacerlo todas las veces parece ser un desperdicio de recursos. Una alternativa es computar una factorización $PA = LU$ una sola vez, almacenar estas matrices, y luego usar el método descrito en la Observación 2.3.3 para resolver los m sistemas. El costo computacional de este segundo procedimiento es de

$$O\left(\frac{2n^3}{3} + mn^2\right) \text{ flops.}$$

\triangle

2.4. Matrices dispersas y de banda

El algoritmo de escalerización gaussiana se puede realizar con cualquier matriz $\mathcal{M}_n(\mathbb{R})$, y su costo es de $O\left(\frac{2n^3}{3}\right)$ flops en general. Esta estimación del costo computacional es una cota superior, y es válida incluso en el peor escenario posible. Si la matriz A posee cierta estructura adicional, entonces se la puede explotar para desarrollar métodos o bien más eficientes o bien que posean otras propiedades. Un ejemplo de esto es la *factorización de Cholesky* para matrices simétricas y definidas positivas que se discute en el práctico del curso.

Aquí nos centramos en matrices que “tienen muchos ceros”. Como mostramos a continuación, definimos esta característica de un modo un tanto vago.

Definición 2.4.1 (matrices dispersas (o ralas)). La **dispersidad** de una matriz $A \in \mathcal{M}_n(\mathbb{R})$ es la fracción de sus elementos que son cero. Diremos que una matriz $A \in \mathcal{M}_n(\mathbb{R})$ es **dispersa (o rala)**³ si su dispersidad es cercana a 1. \triangle

Las matrices dispersas aparecen en muchísimos problemas tanto de ingeniería como de otras ramas y su manipulación eficiente es un vivo tema de investigación.

Sea A una matriz dispersa con elementos no nulos. Esto quiere decir que A “tiene muchos ceros” y por lo tanto no es eficiente almacenarlos. Una técnica de almacenamiento más apropiada consiste en guardar el valor de los elementos no nulos de A junto con sus índices de fila y columna,

$$\left. \begin{array}{l} (i_1, j_1, a_{i_1,j_1}) \\ (i_2, j_2, a_{i_2,j_2}) \\ \vdots \\ (i_m, j_m, a_{i_m,j_m}) \end{array} \right\} \text{ si una entrada } a_{ij} \text{ no está en la lista, entonces es nula.}$$

Observemos que, si tratamos a una matriz $A \in \mathcal{M}_n(\mathbb{R})$ como si fuera una matriz llena, entonces requerimos almacenar n^2 números de punto flotante. En cambio, si la tratamos como una matriz dispersa y tiene m elementos no nulos, entonces requerimos almacenar $3m$ números de punto flotante. Deducimos, pues, que si $3m \ll n^2$, se logra una interesante reducción de espacio de almacenamiento al tratar a A como dispersa. En Octave, para indicar que se trate a una matriz como dispersa, se usa la función `sparse`.

Observación 2.4.1 (inversa de una matriz dispersa). En general, que una matriz invertible A sea dispersa no implica que A^{-1} también lo sea. \triangle

Una clase de matrices dispersas muy especial y que aparece en múltiples aplicaciones es la de aquellas matrices cuyos elementos no nulos se concentran alrededor de la diagonal.

³A veces, se utiliza el inexistente término *esparsa* para nombrar a estas matrices; esta palabra parece derivar del inglés *sparse*.

Definición 2.4.2 (ancho de banda). El **ancho de banda** de una matriz $A \in \mathcal{M}_n(\mathbb{R})$ es la máxima distancia entre sus elementos no nulos y la diagonal. Equivalentemente, decimos que $A \in \mathcal{M}_n(\mathbb{R})$ tiene ancho de banda k si $a_{ij} = 0$ para todo $|i - j| > k$. \triangle

Como casos particulares de la definición anterior, tenemos:

- toda matriz diagonal tiene ancho de banda igual a 0;
- a las matrices que tienen ancho de banda igual a 1 se las llama **tridiagonales**.

A las matrices tridiagonales es posible realizarles una factorización LU en forma extremadamente eficiente, computando matrices L triangular inferior y U triangular superior y ambas con un ancho de banda igual a 1. Dicho procedimiento se llama *algoritmo de Thomas* y en el práctico demostramos que tiene un costo computacional de $O(n)$ flops.

Es importante recalcar que, si $A \in \mathcal{M}_n(\mathbb{R})$ es una matriz dispersa pero sin una estructura especial (como ser tridiagonal), entonces en general el método de escalerización gaussiana requiere $O(n^3)$ flops. En diversas aplicaciones, incluyendo en el estudio de redes eléctricas, modelos en economía, procesos físicos como difusión y radiación, y elasticidad, suelen aparecer matrices de este tipo con potencialmente miles o millones de incógnitas. En estos casos, el costo computacional puede volver inviable utilizar la escalerización gaussiana.

Los comandos *backslash* y *lu*.

En Octave y Matlab, tenemos disponibles los comandos `\`, llamado *backslash*, y `lu`.

Dados una matriz $A \in \mathcal{M}_n(\mathbb{R})$ y un vector $b \in \mathbb{R}^n$, el comando

```
>> x = A \ b;
```

nos devuelve un vector x con la solución computacional al sistema $Ax = b$. En general, la función *backslash* chequea en primer lugar si la matriz A posee una cierta estructura especial: si es triangular, realiza una sustitución hacia adelante o atrás, según corresponda; si es simétrica y definida positiva, utiliza el algoritmo de Cholesky para producir una descomposición particular. En caso de que A no tenga una estructura especial, *backslash* efectúa un algoritmo de escalerización gaussiana con pivoteo parcial.

Por otra parte, el comando

```
>> [L,U,P] = lu(A);
```

nos devuelve tres matrices L , U , P como en la Sección 2.3: L es triangular inferior, U es triangular superior, P es de permutación, y vale $PA = LU$. Estas matrices son calculadas realizando un algoritmo de escalerización gaussiana como describimos en esa sección.

2.5. Estabilidad y convergencia

Por lo que hemos estudiado en el Capítulo 1 sabemos que, incluso utilizando métodos directos, en la resolución numérica de sistemas lineales cometemos *errores*. Para los métodos directos de resolución de sistemas lineales, la única fuente de error es la representación de números reales con precisión finita. En esta sección inspeccionamos cómo se puede definir el error al resolver un sistema lineal y qué se puede decir acerca del error cometido al utilizar el método de escalerización gaussiana con pivoteo parcial.

2.5.1. Normas de vectores y matrices

Para estudiar la convergencia de un método, nuestro punto de partida debe ser definir una noción de cercanía en los espacios con los que trabajamos. Aquí recordamos algunas nociones sobre normas en espacios vectoriales en general e introducimos algunas nociones sobre normas de matrices.

En primer lugar, consideraremos el espacio vectorial \mathbb{R}^n con las operaciones habituales de suma de vectores y de producto entre números reales y vectores. Recordamos que una **norma** es una función $\|\cdot\|: \mathbb{R}^n \rightarrow \mathbb{R}$ que verifica las siguientes propiedades:

1. $\|\mathbf{u}\| \geq 0 \quad \forall \mathbf{u} \in \mathbb{R}^n$, y $\|\mathbf{u}\| = 0 \Leftrightarrow \mathbf{u} = \mathbf{0}$;
2. $\|\lambda \mathbf{u}\| = |\lambda| \|\mathbf{u}\| \quad \forall \lambda \in \mathbb{R}, \quad \forall \mathbf{u} \in \mathbb{R}^n$;
3. $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\| \quad \forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^n$.

En este curso, es de interés la siguiente familia de normas en \mathbb{R}^n .

Normas ℓ^p . Sea $1 \leq p < \infty$. Dado un vector $\mathbf{x} = [x_1, \dots, x_n]^t \in \mathbb{R}^n$, la norma ℓ^p del vector \mathbf{x} está dada por

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}.$$

Algunos casos destacables de normas ℓ^p son los siguientes.

- Para $p = 1$, se obtiene la **norma de taxi (o de Manhattan)**, $\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$.
- Para $p = 2$ se obtiene la **norma Euclídea**, $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$.

Observación 2.5.1. La norma Euclídea tiene la importante propiedad de provenir del *producto interno usual* en \mathbb{R}^n . Recordamos que este producto interno es $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$, $\langle \mathbf{x}, \mathbf{y} \rangle := \mathbf{x}^t \mathbf{y}$. A veces también usaremos el punto \cdot para denotar al producto interno en \mathbb{R}^n , $\mathbf{x} \cdot \mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle$. Es sencillo verificar que $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x} \cdot \mathbf{x}}$ para todo $\mathbf{x} \in \mathbb{R}^n$. \triangle

Norma ℓ^∞ . Dado $\mathbf{x} = [x_1, \dots, x_n]^t \in \mathbb{R}^n$, se define la norma ℓ^∞ (llamada **norma infinito, del máximo, o de Chebyschev**) del vector \mathbf{x} mediante

$$\|\mathbf{x}\|_\infty := \max_{1 \leq i \leq n} |x_i|.$$

El siguiente ejercicio justifica la notación $\|\cdot\|_\infty$ para la norma del máximo.

Ejercicio 2.4. Sea $\mathbf{x} \in \mathbb{R}^n$ un vector fijo. Demostrar que

$$\lim_{p \rightarrow \infty} \|\mathbf{x}\|_p = \|\mathbf{x}\|_\infty.$$

[Sugerencia: primero, demostrar que si $\alpha_1, \dots, \alpha_n$ son tales que $|\alpha_i| \leq 1$ para $i = 1, \dots, n$ y se tiene $|\alpha_k| = 1$ para algún k , entonces $\lim_{p \rightarrow \infty} (\sum_{i=1}^n |\alpha_i|^p)^{1/p} = 1$. Luego, si $\mathbf{x} \neq \mathbf{0}$, aplicar esta propiedad con $\alpha_i = \frac{x_i}{\|\mathbf{x}\|_\infty}$. La afirmación es trivialmente cierta si $\mathbf{x} = \mathbf{0}$.]

De cursos anteriores de álgebra lineal, sabemos que el espacio de matrices $\mathcal{M}_{m \times n}(\mathbb{R})$, con las operaciones habituales de suma de matrices y de producto de matrices por números reales, tiene estructura de espacio vectorial. Para definir normas, puede ser tentador identificar matrices en $\mathcal{M}_{m \times n}(\mathbb{R})$ con vectores en \mathbb{R}^{mn} y utilizar las normas ℓ^p o ℓ^∞ de vectores que discutimos arriba⁴. Aquí nos concentraremos en normas de matrices definidas de otra forma, las llamadas *normas de operador*, que están asociadas a identificar las matrices en $\mathcal{M}_{m \times n}(\mathbb{R})$ con transformaciones lineales $\mathbb{R}^n \rightarrow \mathbb{R}^m$.

Definición 2.5.1 (norma inducida o norma operador). Sea $\|\cdot\|_v$ una norma vectorial. Definimos la **norma matricial inducida por $\|\cdot\|_v$** (o **norma operador asociada a $\|\cdot\|_v$**), $\|\cdot\| : \mathcal{M}_{m \times n}(\mathbb{R}) \rightarrow \mathbb{R}$, mediante

$$\|A\| := \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|_v}{\|\mathbf{x}\|_v}. \quad (2.5)$$

△

Ejercicio 2.5. a) Verificar que las normas inducidas son efectivamente normas. Esto es, demostrar que la norma dada por (2.5) verifica las tres condiciones que mencionamos al comienzo de esta sección.

b) Demostrar que se cumplen las igualdades

$$\|A\| = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|_v}{\|\mathbf{x}\|_v} = \max_{\|\mathbf{x}\|_v \leq 1} \|A\mathbf{x}\|_v = \max_{\|\mathbf{x}\|_v = 1} \|A\mathbf{x}\|_v.$$

Esto muestra que el valor de la norma inducida de una matriz A por una cierta norma vectorial $\|\cdot\|_v$ es igual al máximo que logra “estirar” (según la norma $\|\cdot\|_v$) la matriz A a los vectores de norma $\|\cdot\|_v$ igual a 1.

⁴El caso $p = 2$ da lugar a la llamada **norma de Frobenius**, que es de interés en varias aplicaciones.

Observación 2.5.2 (normas diferentes). En la Definición 2.5.1 aparece la norma vectorial $\|\cdot\|_v$ aplicada tanto a vectores $\mathbf{x} \in \mathbb{R}^n$ como a vectores $A\mathbf{x} \in \mathbb{R}^m$. Esta definición se puede extender al caso en que se tienen normas diferentes en \mathbb{R}^n y en \mathbb{R}^m , pero en este curso no vamos a profundizar en esa extensión. \triangle

Definición 2.5.2 (normas compatibles y submultiplicativas). Sean $\|\cdot\|_M$ una norma matricial y $\|\cdot\|_v$ una norma vectorial.

Decimos que $\|\cdot\|_M$ es **compatible con** $\|\cdot\|_v$ si se cumple que

$$\|A\mathbf{x}\|_v \leq \|A\|_M \|\mathbf{x}\|_v, \quad \forall A \in \mathcal{M}_{m \times n}(\mathbb{R}), \forall \mathbf{x} \in \mathbb{R}^n$$

Decimos que $\|\cdot\|_M$ es **submultiplicativa** si se cumple que

$$\|AB\|_M \leq \|A\|_M \|B\|_M, \quad \forall A \in \mathcal{M}_{m \times n}(\mathbb{R}), B \in \mathcal{M}_{n \times p}(\mathbb{R}).$$

\triangle

La siguiente proposición muestra que la definición anterior aplica a las normas matriciales inducidas.

Proposición 2.5.1 (compatibilidad y submultiplicatividad de las normas operador). *Sea $\|\cdot\|$ la norma matricial inducida por una norma vectorial $\|\cdot\|_v$. Se cumplen las desigualdades*

$$\|A\mathbf{x}\|_v \leq \|A\| \|\mathbf{x}\|_v \quad \forall A \in \mathcal{M}_{m \times n}(\mathbb{R}), \forall \mathbf{x} \in \mathbb{R}^n,$$

y

$$\|AB\| \leq \|A\| \|B\| \quad \forall A \in \mathcal{M}_{m \times n}(\mathbb{R}), B \in \mathcal{M}_{n \times p}(\mathbb{R}). \quad (2.6)$$

Demostración. Comenzamos por demostrar la primera desigualdad. Sean $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ y $\mathbf{x} \in \mathbb{R}^n$. Si $\mathbf{x} = \mathbf{0}$ entonces la desigualdad vale trivialmente, por lo que podemos asumir $\mathbf{x} \neq \mathbf{0}$. Utilizamos la definición (2.5) para obtener

$$\|A\mathbf{x}\|_v = \frac{\|A\mathbf{x}\|_v}{\|\mathbf{x}\|_v} \|\mathbf{x}\|_v \leq \left(\max_{\mathbf{z} \neq \mathbf{0}} \frac{\|A\mathbf{z}\|_v}{\|\mathbf{z}\|_v} \right) \|\mathbf{x}\|_v = \|A\| \|\mathbf{x}\|_v.$$

Para demostrar la segunda desigualdad, consideramos $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ y $B \in \mathcal{M}_{n \times p}(\mathbb{R})$, y suponemos que $\|\cdot\|$ es la norma operador asociada a una norma vectorial $\|\cdot\|_v$. Dado $\mathbf{x} \in \mathbb{R}^p$, usamos dos veces la compatibilidad entre estas normas y $\|\cdot\|_v$ para escribir

$$\|AB\| = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|AB\mathbf{x}\|_v}{\|\mathbf{x}\|_v} \leq \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\| \|B\mathbf{x}\|_v}{\|\mathbf{x}\|_v} \leq \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\| \|B\| \|\mathbf{x}\|_v}{\|\mathbf{x}\|_v} = \|A\| \|B\|.$$

\square

Corolario 2.5.2 (normas de potencias). *Sea $\|\cdot\|$ una norma matricial inducida y $k \in \mathbb{N}$. Entonces, para toda $A \in \mathcal{M}_n(\mathbb{R})$ vale*

$$\|A^k\| \leq \|A\|^k.$$

Demostración. Es una consecuencia directa de la propiedad de submultiplicatividad (2.6) de las normas inducidas. \square

A partir de este punto, cuando escribamos normas no haremos referencia explícita a qué tipo de norma estamos considerando (matricial o vectorial), sino que esto se deduce del argumento que tome la norma en cada caso. Asimismo, cuando escribamos una norma matricial, por defecto asumimos que es una norma operador asociada a una cierta norma vectorial.

Ejemplos de normas inducidas. A continuación profundizamos en qué formas toman las normas matriciales inducidas por algunas de las normas vectoriales ℓ^p . Específicamente, nos concentraremos en las normas operador asociadas a las normas ℓ^1 , ℓ^2 , y ℓ^∞ .

Proposición 2.5.3 (norma inducida por la norma infinito). *Sea $\|\cdot\|_\infty$ la norma matricial inducida por la norma ℓ^∞ vectorial. Se cumple que*

$$\|A\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^n |a_{ij}| \quad \forall A = (a_{ij}) \in \mathcal{M}_{m \times n}(\mathbb{R}). \quad (2.7)$$

Esto es, la norma matricial inducida por la norma ℓ^∞ es igual al máximo de las normas ℓ^1 de las filas de la matriz.

Demostración. Sea $A = (a_{ij}) \in \mathcal{M}_{m \times n}(\mathbb{R})$. Si A es la matriz nula, entonces (2.7) vale trivialmente. Asumamos A no es nula y tomemos un vector $\mathbf{x} \in \mathbb{R}^n$ con $\|\mathbf{x}\|_\infty = 1$, lo que implica que

$$|x_j| \leq \max_k |x_k| = \|\mathbf{x}\|_\infty = 1 \quad \forall j = 1, \dots, n.$$

Usamos esta propiedad junto a la desigualdad triangular, y la definición de norma ℓ^∞ vectorial para obtener

$$\|A\mathbf{x}\|_\infty = \max_{i=1,\dots,m} |(A\mathbf{x})_i| = \max_{i=1,\dots,m} \left| \sum_{j=1}^n a_{ij}x_j \right| \leq \max_{i=1,\dots,m} \sum_{j=1}^n |a_{ij}x_j| \leq \max_{i=1,\dots,m} \sum_{j=1}^n |a_{ij}|.$$

Por lo tanto, utilizando la segunda parte del Ejercicio 2.5, llegamos a la desigualdad

$$\|A\|_\infty = \max_{\|\mathbf{x}\|_\infty=1} \|A\mathbf{x}\|_\infty \leq \max_{i=1,\dots,m} \sum_{j=1}^n |a_{ij}|. \quad (2.8)$$

Para probar que se cumple (2.7), alcanza con probar que la cota hallada se alcanza. Es decir, basta con hallar un vector $\mathbf{y} \in \mathbb{R}^n$ con $\|\mathbf{y}\|_\infty = 1$ tal que la desigualdad de arriba sea de hecho una igualdad. Para este fin, consideremos un índice $i_0 \in 1, \dots, m$ tal que la fila i_0 de la matriz A tiene la mayor norma ℓ^1 entre todas las filas de A ,

$$\sum_{j=1}^n |a_{i_0 j}| = \max_{i=1,\dots,m} \sum_{j=1}^n |a_{ij}|,$$

y consideramos el vector $\mathbf{y} \in \mathbb{R}^n$ cuyas coordenadas son el signo de las entradas de la fila i_0 de A : $\mathbf{y} := [\operatorname{sgn}(a_{i_01}), \operatorname{sgn}(a_{i_02}), \dots, \operatorname{sgn}(a_{i_0n})]^t$; esto da lugar a que $a_{i_0j}y_j = |a_{i_0j}|$ para todo $j = 1, \dots, n$. Como sus entradas son todas $-1, 0, 1$ y no pueden ser todas iguales a 0 (porque A no es la matriz nula), se tiene $\|\mathbf{y}\|_\infty = 1$ y

$$\|A\|_\infty = \max_{\|\mathbf{x}\|_\infty=1} \|A\mathbf{x}\|_\infty \geq \|A\mathbf{y}\|_\infty = \max_{i=1, \dots, m} \left| \sum_{j=1}^n a_{ij}y_j \right| \geq \left| \sum_{j=1}^n a_{i_0j}y_j \right| = \sum_{j=1}^n |a_{i_0j}|.$$

Esto prueba que la igualdad en (2.8) se alcanza cuando $\mathbf{x} = \mathbf{y}$ y concluye la prueba. \square

Sea $A \in \mathcal{M}_{m \times n}(\mathbb{R})$. Así como la norma matricial de A inducida por la norma ℓ^∞ es igual al máximo de las normas ℓ^1 de las *filas* de A , la norma matricial de A inducida por la norma ℓ^1 es igual al máximo de las normas ℓ^1 de las *columnas* de A . Dejamos la demostración de esta propiedad como ejercicio.

Ejercicio 2.6. Sea $\|\cdot\|_1$ la norma matricial inducida por la norma ℓ^1 vectorial. Demostrar que vale la igualdad

$$\|A\|_1 = \max_{j=1, \dots, n} \sum_{i=1}^m |a_{ij}| \quad \forall A = (a_{ij}) \in \mathcal{M}_{m \times n}(\mathbb{R}).$$

[Sugerencia: seguir la estrategia de demostración de la Proposición 2.5.3. Primero, probar de forma directa la desigualdad $\|A\|_1 \leq \max_{j=1, \dots, n} \sum_{i=1}^m |a_{ij}|$ y luego demostrar que la cota se alcanza cuando se toma $\mathbf{y} = \mathbf{e}_{j_0}$, el j_0 -ésimo vector de la base canónica de \mathbb{R}^n , donde j_0 es el índice de la columna de A que tiene mayor norma ℓ^1 .]

La norma matricial inducida por la norma euclídea también tiene una caracterización importante. En ella aparecen los llamados *valores singulares* de la matriz. Vamos a profundizar en ellos cuando tratemos problemas de mínimos cuadrados.

Proposición 2.5.4 (norma inducida por la norma euclídea). *Sea $\|\cdot\|_2$ la norma matricial inducida por la norma ℓ^2 vectorial. Para toda $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, se cumple que*

$$\|A\|_2 = \sqrt{\lambda_1}, \tag{2.9}$$

donde λ_1 es el mayor valor propio de $A^t A$. El número $\sigma_1(A) := \sqrt{\lambda_1}$ se llama el **primer valor singular de la matriz A** .

Observación 2.5.3 (valores propios de $A^t A$). Antes de demostrar la proposición, observemos que el enunciado tiene sentido: que $A^t A$ tiene valores propios reales y son no negativos. Dada $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ es sencillo verificar que la matriz $A^t A \in \mathcal{M}_n(\mathbb{R})$ es simétrica; por lo tanto, el Teorema Espectral para matrices reales y simétricas nos asegura que $A^t A$ es diagonalizable. Además, $A^t A$ es semidefinida positiva, pues para todo $\mathbf{x} \in \mathbb{R}^n$ tenemos

$$(A^t A \mathbf{x}) \cdot \mathbf{x} = \mathbf{x}^t A^t A \mathbf{x} = (A \mathbf{x})^t (A \mathbf{x}) = \|A \mathbf{x}\|_2^2 \geq 0. \tag{2.10}$$

Esto quiere decir que para toda $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, todos los valores propios de $A^t A$ son mayores o iguales que 0. Por lo tanto, tiene sentido tomar raíz cuadrada en (2.9). \triangle

Demostración de la Proposición 2.5.4. Por la definición de norma inducida y operando como en (2.10), tenemos

$$\|A\|_2 = \max_{\|\mathbf{x}\|_2=1} \|A\mathbf{x}\|_2 = \max_{\|\mathbf{x}\|_2=1} \sqrt{\mathbf{x}^t A^t A \mathbf{x}}. \quad (2.11)$$

Por la Observación 2.5.3 la matriz $A^t A$ es simétrica, y por lo tanto es diagonalizable y existe una base ortonormal de \mathbb{R}^n , $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$, formada por vectores propios de $A^t A$. Asumimos que para cada $i = 1, \dots, n$, el vector propio \mathbf{v}_i tiene valor propio asociado λ_i (notar que $\lambda_i \geq 0$ por (2.10)) y que $\lambda_1 \geq \lambda_2 \geq \dots \lambda_n \geq 0$.

Sea $\mathbf{x} \in \mathbb{R}^n$ tal que $\|\mathbf{x}\|_2 = 1$. Podemos escribir \mathbf{x} en esta base como

$$\mathbf{x} = \sum_{i=1}^n \alpha_i \mathbf{v}_i, \quad \text{con } \alpha_i = \mathbf{x} \cdot \mathbf{v}_i = \mathbf{x}^t \mathbf{v}_i.$$

Notemos que entonces podemos usar la ortonormalidad de la base de vectores propios para escribir

$$1 = \|\mathbf{x}\|_2^2 = \mathbf{x}^t \mathbf{x} = \left(\sum_{i=1}^n \alpha_i \mathbf{v}_i \right)^t \left(\sum_{j=1}^n \alpha_j \mathbf{v}_j \right) = \sum_{i,j=1}^n \alpha_i \alpha_j \mathbf{v}_i^t \mathbf{v}_j = \sum_{i=1}^n \alpha_i^2.$$

Por lo tanto, usando propiedades elementales del producto de matriz por vector, la definición de los coeficientes α_i y el hecho de que $\lambda_1 \geq \lambda_i$ para todo i , tenemos

$$\mathbf{x}^t A^t A \mathbf{x} = \mathbf{x}^t (A^t A \mathbf{x}) = \mathbf{x}^t \sum_{i=1}^n \alpha_i A^t A \mathbf{v}_i = \mathbf{x}^t \sum_{i=1}^n \alpha_i \lambda_i \mathbf{v}_i = \sum_{i=1}^n \alpha_i^2 \lambda_i \leq \lambda_1.$$

Tomando raíz cuadrada en la desigualdad anterior y tomando máximo entre todos los vectores con $\|\mathbf{x}\|_2 = 1$, deducimos que $\|A\|_2 \leq \sqrt{\lambda_1}$.

Para probar que vale $\|A\|_2 \geq \sqrt{\lambda_1}$, basta con encontrar un vector \mathbf{y} tal que $\|\mathbf{y}\|_2 = 1$ y $\|A\mathbf{y}\|_2 = \sqrt{\lambda_1}$. Elegimos \mathbf{y} como un vector propio de $A^t A$ asociado al mayor valor propio λ_1 . Usando la definición de norma euclídea, tenemos que

$$\|A\mathbf{y}\|_2 = \sqrt{\mathbf{y}^t A^t A \mathbf{y}} = \sqrt{\mathbf{y}^t \lambda_1 \mathbf{y}} = \sqrt{\lambda_1} \sqrt{\mathbf{y}^t \mathbf{y}} = \sqrt{\lambda_1} \|\mathbf{y}\|_2 = \sqrt{\lambda_1}.$$

Esto concluye la prueba. □

Observación 2.5.4 (advertencia sobre valores singulares de matrices diagonalizables). De Geometría y Álgebra Lineal 2 sabemos que si $A \in \mathcal{M}_n(\mathbb{R})$ es diagonalizable, entonces A^t también es diagonalizable y que sus valores propios coinciden. Sin embargo, en general, *no se cumple que los subespacios propios coincidan*. Esto implica que **incluso si A es diagonalizable, los valores propios de $A^t A$ en general no son iguales a los cuadrados de los valores propios de A .** △

2.5.2. Errores y residuos: un ejemplo

Dados una matriz invertible $A \in \mathcal{M}_n(\mathbb{R})$ y un vector $\mathbf{b} \in \mathbb{R}^n$, queremos resolver el sistema $A\mathbf{x} = \mathbf{b}$ computacionalmente. Supongamos que tenemos un algoritmo –como por ejemplo el método de escalerización gaussiana o uno de los que veremos más adelante en este capítulo– que nos permite hallar una aproximación $\bar{\mathbf{x}}$. Tal como vimos en la Definición 1.3.1, consideramos el **error** (absoluto)

$$\mathbf{e} := \bar{\mathbf{x}} - \mathbf{x} \in \mathbb{R}^n.$$

Naturalmente, es deseable que este error sea pequeño, pero para poder determinarlo *necesitamos conocer la solución del sistema*. En la práctica, no parece razonable esperar conocer la solución exacta del problema que queremos resolver, por lo que es relevante contar con otra forma de medir la discrepancia entre nuestra solución aproximada y la solución del problema exacto. Definimos el **residuo** como

$$\mathbf{r} := A\bar{\mathbf{x}} - \mathbf{b} \in \mathbb{R}^n.$$

El residuo mide por cuánto “le erra” a \mathbf{b} nuestra solución aproximada cuando la multiplicamos por A . Una diferencia fundamental entre el residuo y el error es que el residuo es *computable*: para calcular \mathbf{r} no se necesita conocer cuál es la solución del sistema que queremos resolver. Además, el error y el residuo se relacionan mediante

$$\mathbf{r} := A\bar{\mathbf{x}} - \mathbf{b} = A\bar{\mathbf{x}} - A\mathbf{x} = A\mathbf{e};$$

como A es invertible, esto demuestra que $\mathbf{e} = \mathbf{0}$ si y sólo si $\mathbf{r} = \mathbf{0}$. Sabemos que no es realista esperar que $\mathbf{e} = \mathbf{0}$, pero quizás sí podamos aspirar a que \mathbf{e} sea “chico”, en el sentido de que alguna norma de \mathbf{e} sea pequeña. Si no conocemos la solución exacta a nuestro sistema, no podemos calcular \mathbf{e} exactamente, pero sí podríamos calcular el residuo \mathbf{r} y alguna norma de este vector. Una pregunta clave es entonces: si $\|\mathbf{r}\|$ es pequeña (para alguna norma vectorial $\|\cdot\|$), ¿esto implica que necesariamente $\|\mathbf{e}\|$ sea pequeña? La respuesta corta es: no. La respuesta más larga es: depende de la matriz A . A continuación profundizaremos en este tema, empezando con un ejemplo.

Ejemplo 2.5.1 (residuo vs. error). Este ejemplo está tomado de [Mol04, Sección 2.8]. Consideremos el sistema $A\mathbf{x} = \mathbf{b}$, con

$$A = \begin{bmatrix} 0,780 & 0,563 \\ 0,913 & 0,659 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0,217 \\ 0,254 \end{bmatrix}. \quad (2.12)$$

Resolvemos este sistema usando una máquina con tres dígitos de precisión y truncamiento, y usando el algoritmo de escalerización gaussiana con pivoteo parcial.

Hacemos pivoteo (intercambio de la primera y segunda fila), calculamos el multiplicador $\ell_{21} = -\frac{0,780}{0,913} \approx -0,854$, y el sistema ampliado nos queda:

$$\left[\begin{array}{cc|c} 0,913 & 0,659 & 0,254 \\ 0 & 0,001 & 0,001 \end{array} \right].$$

Haciendo sustitución hacia atrás, obtenemos la solución aproximada

$$\bar{\mathbf{x}} = \begin{bmatrix} -0,443 \\ 1 \end{bmatrix}.$$

Si no conocemos la solución exacta del sistema, podemos calcular el residuo en nuestra solución,

$$\mathbf{r} = A\bar{\mathbf{x}} - \mathbf{b} = \begin{bmatrix} -0,000460 \\ -0,000541 \end{bmatrix}.$$

Nos quedamos tranquilos porque el residuo es pequeño⁵: por ejemplo, $\|\mathbf{r}\|_\infty = 5,41 \times 10^{-4}$. Sin embargo, se puede comprobar que la solución exacta del sistema es

$$\mathbf{x} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

¡Le erramos feo! Hicimos las cosas de la mejor forma que pudimos, pivoteando⁶, y sin embargo nuestra solución tiene un error

$$\mathbf{e} = \bar{\mathbf{x}} - \mathbf{x} = \begin{bmatrix} -1,443 \\ 2 \end{bmatrix}.$$

Claramente, en este caso, tener un residuo pequeño no nos garantiza que el error también lo sea: aquí tenemos $\|\mathbf{e}\|_\infty = 2$. \triangle

¿Qué falló en el ejemplo anterior? La matriz A es invertible, por lo que el sistema es compatible determinado, pero está muy cerca de ser singular. Si entendemos las dos ecuaciones en el sistema como las ecuaciones de dos rectas en el plano, y que por lo tanto resolver el sistema es equivalente a hallar la intersección entre ambas rectas, tenemos una situación como en la Figura 2.1 (izquierda). Observamos que las dos rectas son casi paralelas, y que la solución $\bar{\mathbf{x}}$ que computamos, si bien no está cerca de \mathbf{x} , sí está cerca de ambas rectas. Esto es precisamente lo que mide el residuo: qué tan cerca están de satisfacerse todas las ecuaciones que conforman el sistema.

En general, el algoritmo de escalerización gaussiana con pivoteo parcial nos asegura que el residuo resultante es (relativamente) pequeño. Si eso implica que el error sea pequeño o no, dependerá de cómo sea la matriz del sistema. En la Sección 2.5.4 hacemos formal esta afirmación, y cuantificamos qué tan grande puede ser el error (en alguna norma dada) en función del residuo obtenido.

Supongamos, en cambio, que al representar las ecuaciones de un sistema (2×2) $A\mathbf{x} = \mathbf{b}$ tuviésemos una situación como en la Figura 2.1 (derecha). En ese caso, las rectas están lejos de ser paralelas, y por lo tanto *estar simultáneamente cerca de ambas rectas implica estar cerca del punto de intersección*. Para un sistema de esa forma, obtener una solución aproximada con residuo pequeño da lugar a un error pequeño.

⁵En este ejemplo usamos la norma ℓ^∞ simplemente por elegir una. El lector puede repetir el análisis con cualquier otra norma vectorial, y el resultado cualitativamente será el mismo.

⁶Se puede verificar que si no hubiésemos pivoteado la solución sería muchísimo peor: el sistema nos hubiera quedado indeterminado (!!).

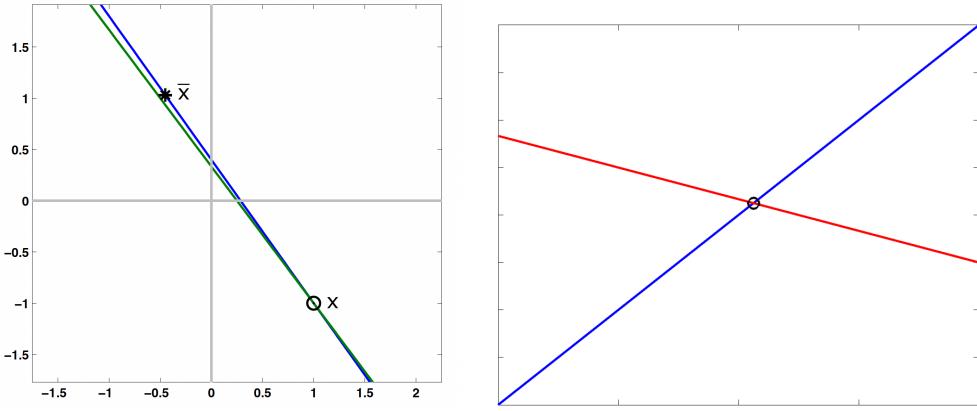


Figura 2.1: Izquierda: solución computada $\bar{\mathbf{x}}$ y solución exacta \mathbf{x} del sistema (2.12). Derecha: un sistema que, al resolverlo y obtener residuo pequeño, podemos asegurar que el error es pequeño.

2.5.3. Número de condición

El Ejemplo 2.5.1 nos muestra que obtener un residuo pequeño no nos garantiza que el error lo sea. El factor de amplificación de residuo a error se relaciona con el llamado *número de condición* de la matriz, que definimos a continuación.

Definición 2.5.3 (número de condición de una matriz). Sean $A \in \mathcal{M}_n(\mathbb{R})$ una matriz invertible y $\|\cdot\|$ una norma matricial. El **número de condición de A respecto a $\|\cdot\|$** es el número $\kappa(A, \|\cdot\|)$ definido mediante

$$\kappa(A, \|\cdot\|) := \|A\| \|A^{-1}\|.$$

Cuando no exista ambigüedad respecto a la norma matricial que estamos considerando, simplemente escribiremos $\kappa(A) = \kappa(A, \|\cdot\|)$. \triangle

Observación 2.5.5 (el número de condición depende de la norma). Sea $A \in \mathcal{M}_n(\mathbb{R})$ una matriz invertible. Es claro que el valor de $\kappa(A, \|\cdot\|)$ depende de la norma que estemos utilizando. Sin embargo, remarcamos que su *magnitud* no depende de la norma considerada. \triangle

La siguiente proposición nos indica que el número de condición de A es una medida del cociente entre lo máximo que se puede “estirar” y lo máximo que se puede “comprimir” un vector al multiplicarlo por ella.

Proposición 2.5.5 (expresión equivalente de $\kappa(A)$). Sean $A \in \mathcal{M}_n(\mathbb{R})$ una matriz invertible y $\|\cdot\|$ una norma matricial inducida por una norma vectorial $\|\cdot\|_v$. Se cumple

$$\kappa(A, \|\cdot\|) = \frac{\max_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|_v}{\|\mathbf{x}\|_v}}{\min_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|_v}{\|\mathbf{x}\|_v}}. \quad (2.13)$$

Demostración. Por el Ejercicio 2.5, ya sabemos que $\|A\| = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|_v}{\|\mathbf{x}\|_v}$, por lo que resta demostrar que

$$\|A^{-1}\| = \frac{1}{\min_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|_v}{\|\mathbf{x}\|_v}}.$$

Para probar esta igualdad, basta notar que

$$\frac{1}{\min_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|_v}{\|\mathbf{x}\|_v}} = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{x}\|_v}{\|A\mathbf{x}\|_v},$$

y como A es invertible, cualquier vector $\mathbf{y} \in \mathbb{R}^n$ se puede escribir como $\mathbf{y} = A\mathbf{x}$ con $\mathbf{x} \in \mathbb{R}^n$. Esto implica entonces que

$$\frac{1}{\min_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|_v}{\|\mathbf{x}\|_v}} = \max_{\mathbf{y} \neq \mathbf{0}} \frac{\|A^{-1}\mathbf{y}\|_v}{\|\mathbf{y}\|_v} = \|A^{-1}\|.$$

□

Una consecuencia inmediata de la proposición anterior es que el número de condición de toda matriz es mayor o igual a 1. En Octave, la función `cond` permite calcular $\kappa(A)$ respecto a varias normas vectoriales. Calcular números de condición puede ser costoso computacionalmente, pero la función `condest` nos permite estimar en forma eficiente el número de condición respecto a la norma inducida por la norma ℓ^1 vectorial.

Ejemplo 2.5.2 (número de condición en el Ejemplo 2.5.1). Fijemos la norma matricial inducida por la norma vectorial ℓ^∞ . Usando la Proposición 2.5.3, para la matriz A dada en (2.12), tenemos

$$\|A\|_\infty = 1,572.$$

Invirtiendo A , y nuevamente usando la Proposición 2.5.3 tenemos

$$A^{-1} = \begin{bmatrix} 6,59 \times 10^5 & -5,63 \times 10^5 \\ -9,13 \times 10^5 & 7,8 \times 10^5 \end{bmatrix} \Rightarrow \|A^{-1}\|_\infty = 1,693 \times 10^6.$$

En consecuencia, $\kappa(A, \|\cdot\|_\infty) \approx 2,66 \times 10^6$. Esta es una matriz *mal condicionada*. △

Error residual. El número de condición de una matriz A juega un rol central en la relación entre la magnitud de errores y residuos en soluciones aproximadas. Sea $\bar{\mathbf{x}}$ una aproximación a la solución del sistema lineal $A\mathbf{x} = \mathbf{b}$. Recordamos que el residuo se define mediante $\mathbf{r} := A\bar{\mathbf{x}} - \mathbf{b}$ y que por lo tanto el error $\mathbf{e} = \bar{\mathbf{x}} - \mathbf{x}$ y el residuo se relacionan mediante $\mathbf{r} = A\mathbf{e}$, o equivalentemente $\mathbf{e} = A^{-1}\mathbf{r}$.

Si consideramos una norma vectorial $\|\cdot\|$ en \mathbb{R}^n y su norma operador inducida (que también denotaremos mediante $\|\cdot\|$), como éstas son compatibles (Proposición 2.5.1), tenemos las desigualdades

$$\|\mathbf{r}\| = \|A\mathbf{e}\| \leq \|A\| \|\mathbf{e}\|, \quad \|\mathbf{e}\| = \|A^{-1}\mathbf{r}\| \leq \|A^{-1}\| \|\mathbf{r}\|.$$

Combinando ambas desigualdades, llegamos a

$$\frac{\|\mathbf{r}\|}{\|A\|} \leq \|\mathbf{e}\| \leq \|A^{-1}\| \|\mathbf{r}\|. \quad (2.14)$$

Por otra parte, usando forma análoga las identidades equivalentes $\mathbf{b} = A\mathbf{x}$ y $\mathbf{x} = A^{-1}\mathbf{b}$, obtenemos

$$\frac{\|\mathbf{b}\|}{\|A\|} \leq \|\mathbf{x}\| \leq \|A^{-1}\| \|\mathbf{b}\| \quad (2.15)$$

Combinando (2.14) y (2.15), deducimos

$$\frac{1}{\|A\| \|A^{-1}\|} \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} \leq \frac{\|\mathbf{e}\|}{\|\mathbf{x}\|} \leq \|A\| \|A^{-1}\| \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$$

y usando la Definición 2.5.3 concluimos

$$\frac{1}{\kappa(A)} \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} \leq \frac{\|\mathbf{e}\|}{\|\mathbf{x}\|} \leq \kappa(A) \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}. \quad (2.16)$$

Notemos que el término en el centro arriba es un *error relativo* en \mathbf{x} , mientras que el término $\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$ corresponde a un residuo relativo. La identidad (2.16) nos dice entonces que este residuo relativo y el error relativo al aproximar la solución de nuestro sistema $A\mathbf{x} = \mathbf{b}$ están relacionados mediante el número de condición de la matriz A . En otras palabras, **si $\kappa(A)$ no es muy grande, residuos relativos pequeños se corresponden con errores relativos pequeños; en cambio, si $\kappa(A)$ es grande, el hecho de que el residuo relativo sea pequeño no nos permite asegurar que el error relativo lo sea.**

2.5.4. Análisis de perturbaciones

A continuación, analizamos qué tan sensible es la solución *exacta* de un sistema $A\mathbf{x} = \mathbf{b}$ al perturbar sus parámetros. Esto es, suponiendo que tenemos un error de representación en A o en \mathbf{b} , nos preguntamos cuánto puede cambiar la solución del sistema. El número de condición de la matriz A juega un rol preponderante en la relación entre el error relativo de la solución y el error relativo en las perturbaciones. En lo que sigue, fijamos una norma vectorial y consideraremos la norma matricial inducida por ella; notaremos todas estas normas mediante $\|\cdot\|$.

Perturbación en el lado derecho ($\mathbf{b} \mapsto \mathbf{b} + \delta_{\mathbf{b}}$). Dados $A \in \mathcal{M}_n(\mathbb{R})$ invertible y $\mathbf{b} \in \mathbb{R}^n$, sea \mathbf{x} la solución del sistema $A\mathbf{x} = \mathbf{b}$. Consideramos una perturbación en el lado derecho de la igualdad, que ahora será de la forma $\mathbf{b} + \delta_{\mathbf{b}}$. Escribimos la solución de este sistema perturbado como $\mathbf{x} + \delta_{\mathbf{x}}$, y por lo tanto tenemos

$$\begin{cases} A\mathbf{x} = \mathbf{b} \\ A(\mathbf{x} + \delta_{\mathbf{x}}) = \mathbf{b} + \delta_{\mathbf{b}} \end{cases} \Rightarrow A\delta_{\mathbf{x}} = \delta_{\mathbf{b}} \Rightarrow \delta_{\mathbf{x}} = A^{-1}\delta_{\mathbf{b}}.$$

Tomando normas en ambos lados de la última igualdad y aplicando la compatibilidad de la norma inducida (Proposición 2.5.1), tenemos

$$\|\delta_{\mathbf{x}}\| = \|A^{-1}\delta_{\mathbf{b}}\| \leq \|A^{-1}\| \|\delta_{\mathbf{b}}\|.$$

Por otra parte, haciendo un razonamiento análogo, tenemos

$$\mathbf{b} = A\mathbf{x} \Rightarrow \|\mathbf{b}\| \leq \|A\| \|\mathbf{x}\|.$$

Multiplicando las dos desigualdades anteriores, deducimos entonces que vale

$$\frac{\|\delta_{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \|A\| \|A^{-1}\| \frac{\|\delta_{\mathbf{b}}\|}{\|\mathbf{b}\|}.$$

Recordando la Definición 2.5.3, concluimos

$$\boxed{\frac{\|\delta_{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \kappa(A) \frac{\|\delta_{\mathbf{b}}\|}{\|\mathbf{b}\|}}.$$

Perturbación en la matriz ($A \mapsto A + \delta_A$). Dados $A \in \mathcal{M}_n(\mathbb{R})$ invertible y $\mathbf{b} \in \mathbb{R}^n$, sea \mathbf{x} la solución del sistema $A\mathbf{x} = \mathbf{b}$. Consideramos una perturbación en la matriz del sistema, que ahora será de la forma $A + \delta_A$ y asumimos que esta nueva matriz también es invertible. Escribimos la solución de este sistema perturbado como $\mathbf{x} + \delta_{\mathbf{x}}$, y por lo tanto tenemos

$$\begin{cases} A\mathbf{x} = \mathbf{b} \\ (A + \delta_A)(\mathbf{x} + \delta_{\mathbf{x}}) = \mathbf{b} \end{cases} \Rightarrow A\delta_{\mathbf{x}} + \delta_A(\mathbf{x} + \delta_{\mathbf{x}}) = \mathbf{0} \Rightarrow \delta_{\mathbf{x}} = -A^{-1}\delta_A(\mathbf{x} + \delta_{\mathbf{x}})$$

Tomamos norma de ambos lados de la última igualdad, usamos la Proposición 2.5.1, multiplicamos y dividimos el lado derecho de la igualdad por $\|A\|$ y usamos la Definición 2.5.3 para llegar a

$$\|\delta_{\mathbf{x}}\| \leq \|A^{-1}\| \|\delta_A\| \|\mathbf{x} + \delta_{\mathbf{x}}\| = \kappa(A) \frac{\|\delta_A\|}{\|A\|} \|\mathbf{x} + \delta_{\mathbf{x}}\|,$$

y concluimos la estimación

$$\boxed{\frac{\|\delta_{\mathbf{x}}\|}{\|\mathbf{x} + \delta_{\mathbf{x}}\|} \leq \kappa(A) \frac{\|\delta_A\|}{\|A\|}}. \quad (2.17)$$

Aplicación a escalerización gaussiana con pivoteo parcial.

Ahora estamos en condiciones de interpretar cómo se comportan los errores y residuos al realizar el algoritmo de escalerización gaussiana con pivoteo parcial para resolver numéricamente un sistema de la forma $A\mathbf{x} = \mathbf{b}$. El ingrediente que nos falta es el siguiente resultado, que asumimos sin demostración.

Teorema 2.5.6 (Wilkinson, 1961). Consideremos un sistema de ecuaciones lineales de la forma $A\mathbf{x} = \mathbf{b}$, con $A \in \mathcal{M}_n(\mathbb{R})$ invertible y $\mathbf{b} \in \mathbb{R}^n$. Sea $\bar{\mathbf{x}}$ la solución computacional obtenida al realizar escalerización gaussiana con pivoteo parcial. Entonces, $\bar{\mathbf{x}}$ es la solución (exacta) a un sistema de la forma

$$(A + E)\bar{\mathbf{x}} = \mathbf{b}, \quad (2.18)$$

donde $E \in \mathcal{M}_n(\mathbb{R})$ es una matriz cuyos elementos son (en valor absoluto) del orden de los errores de redondeo al representar la matriz A .

Analicemos el enunciado del Teorema 2.5.6. El hecho de que los elementos de E sean del orden de los errores de redondeo al representar la matriz A significa que, para cualquier norma matricial, tenemos

$$\|E\| \leq \rho \varepsilon_M \|A\|$$

para alguna constante $\rho > 0$, que podemos esperar cumpla $\rho \lesssim 10$. Por lo tanto, ese teorema nos garantiza estar en una situación en la que nuestra solución computacional $\bar{\mathbf{x}}$ es la solución exacta a un sistema al que le hemos perturbado la matriz A , la perturbación es $\delta_A = E$, y tenemos un cierto control en norma sobre esta perturbación. Notamos que $\delta_{\mathbf{x}} = \bar{\mathbf{x}} - \mathbf{x} = \mathbf{e}$ y que por lo tanto la fórmula (2.17) se puede reescribir como

$$\frac{\|\mathbf{e}\|}{\|\bar{\mathbf{x}}\|} \leq \kappa(A) \frac{\|E\|}{\|A\|} \leq \kappa(A) \rho \varepsilon_M.$$

Como el lado izquierdo de la desigualdad anterior es como el error relativo en \mathbf{x} (pensando en que $\|\bar{\mathbf{x}}\| \approx \|\mathbf{x}\|$), concluimos que **al realizar escalerización gaussiana con pivoteo parcial, el error relativo que se tiene no es peor que del orden del épsilon de máquina multiplicado por el número de condición de la matriz del sistema**. En particular, si la matriz del sistema está bien condicionada, podemos estar tranquilos de que nuestra solución computacional es de buena calidad.

En cambio, la fórmula (2.18) nos dice que podemos escribir el residuo como $\mathbf{r} = A\bar{\mathbf{x}} - \mathbf{b} = -E\bar{\mathbf{x}}$ y por lo tanto tenemos

$$\frac{\|\mathbf{r}\|}{\|A\| \|\bar{\mathbf{x}}\|} \leq \frac{\|E\|}{\|A\|} \leq \rho \varepsilon_M.$$

El lado izquierdo de la desigualdad anterior es como un residuo relativo, ya que toma la norma del residuo y la normaliza por el producto $\|A\| \|\bar{\mathbf{x}}\|^7$. En conclusión, **al realizar escalerización gaussiana con pivoteo parcial, el residuo relativo que se tiene es del orden del épsilon de máquina**.

⁷Para entender por qué esta es una buena normalización del residuo, puede servir pensar en el sistema como si A , \mathbf{x} y \mathbf{b} tuvieran dimensiones (de largo, de tiempo, etc). Las dimensiones del residuo tienen que ser iguales que las del vector \mathbf{b} , que a su vez deben ser iguales al producto de las dimensiones de A por las de \mathbf{x} .

Capítulo 3

Interpolación polinomial

3.1. Introducción. ¿Por qué interpolar?

El problema de interpolación consiste en encontrar una función φ desconocida a partir de un cierto conjunto (finito) de datos, respetando los valores de los mismos. Por ejemplo, si realizamos una actividad experimental y medimos observaciones $\{(x_i, y_i)\}_{i=0, \dots, n}$, entonces podemos querer que φ cumpla $\varphi(x_i) = y_i$ para todo i . Evidentemente, si no tenemos más información sobre φ , este será un problema con *infinitas* soluciones, ya que cualquier función φ cuyo gráfico pase por los datos podría ser solución (ver Figura 3.1). Una forma natural de evitar este inconveniente es *restringir* la clase de funciones posibles en la que podemos elegir φ ; en este curso nos centramos en la llamada *interpolación polinomial* (o polinómica), que justamente consiste en requerir que φ sea un polinomio o un polinomio a trozos.

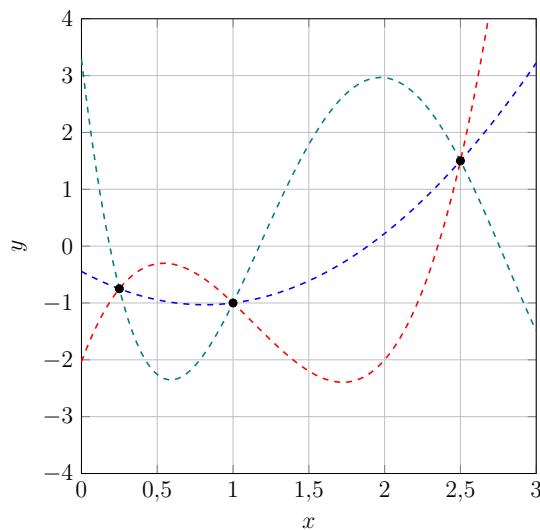


Figura 3.1: Cualquiera de las curvas dibujadas pasa por los tres puntos marcados.

Además del ejemplo que mencionamos en el párrafo anterior, el problema de interpolación tiene variados usos en la práctica, ya sea como un fin en sí mismo –buscando una función cuyo gráfico pase por ciertos puntos– o como herramienta auxiliar para llevar adelante otras técnicas. Mencionamos unos pocos ejemplos.

- Si tenemos una tabla de evaluaciones de una cierta función (desconocida) f , y nos interesa evaluar esta f en algún punto que no esté en la tabla, entonces necesitaremos “asignarle” valores a f en los puntos que no estén en nuestra tabla. Precisamente, hacer esto es definir una interpolación φ de los datos de nuestra tabla y luego evaluar φ en puntos que no estén en la tabla.
- En forma similar, dada una tabla de evaluaciones de una cierta función (desconocida) f , nos puede interesar aproximar la derivada o la integral de esta función f . Para ello, podemos nuevamente construir una interpolante de los datos de la tabla y utilizarla para derivarla o integrarla analíticamente.
- El CAD¹, o diseño asistido por computadora consiste en utilizar computadoras para ayudar en la creación, modificación, análisis u optimización de un diseño. Es posible que el lector tenga experiencia en el uso de programas como AutoCAD: uno podría preguntarse cómo hacen programas de ese tipo para definir curvas y superficies y que al abrir y cerrar los archivos éstas no cambien. ¿Tiene que almacenar todos los puntos de la curva o la superficie? ¿Puede almacenar las “fórmulas” de curvas y superficies? Un estándar en este tipo de programas es el uso de las llamadas splines, que introducimos en la Sección 3.4.2, o variantes de ellas. Como dato anecdótico, algunos de los avances algorítmicos más importantes para el tratamiento de curvas y superficies, como son los algoritmos de [De Casteljau](#) y de [De Boor](#), fueron motivados por necesidades concretas de la industria automotriz de mediados del siglo XX.

La Sección 3.2 está dedicada a la definición del polinomio interpolante a través de un conjunto dado de puntos y a distintas formas de representarlo y computarlo. Luego, en la Sección 3.3, considerando que estos puntos estén dados por la imagen de una cierta función, estimamos la discrepancia entre dicha función y la interpolante, y nos encontramos con que no siempre interpolar por más puntos implica una mejor aproximación. Finalmente, la Sección 3.4 está destinada a las llamadas interpolaciones polinomiales a trozos, en particular la interpolación lineal y tres tipos de interpolantes cúbicas distintas.

3.2. Polinomio interpolante

Consideramos el problema clásico de interpolación polimomial: dados $n + 1$ puntos en el plano $\{(x_i, y_i)\}_{i=0,\dots,n}$ con $x_i \neq x_j$ si $i \neq j$, buscamos un polinomio p tal que $p(x_i) = y_i$

¹Del inglés *Computer-Aided Design*.

para todo $i = 0, \dots, n$. La primera pregunta que surge es qué requerimientos tiene sentido pedirle a p : ¿puede ser un polinomio de cualquier grado?

Razonemos intuitivamente. Si tenemos un único punto (x_0, y_0) , entonces hay un único polinomio de grado 0 que cumple lo estipulado, la función constante $p(x) = y_0$, y obviamente hay infinitos polinomios de grado mayor o igual a 1 cuyos gráficos pasan por (x_0, y_0) . Si tenemos dos puntos $\{(x_0, y_0), (x_1, y_1)\}$, entonces en general –salvo que $y_0 = y_1$ – no podemos esperar que el gráfico de una función constante pase por esos dos puntos. En cambio, si sabemos que existe una única recta por los puntos $(x_0, y_0), (x_1, y_1)$ y podemos considerar la función lineal cuyo gráfico sea dicha recta: para $n + 1 = 2$ puntos, hay una única función lineal que cumple lo estipulado. Si consideramos polinomios de grado más alto, entonces tenemos *demasiados grados de libertad* y no tenemos una solución única.

Del razonamiento de arriba parece deducirse que, si esperamos que nuestro problema de interpolación por $n + 1$ puntos tenga solución y que ésta sea única, entonces debemos requerir que el polinomio p sea de grado n . El siguiente teorema demuestra que esta intuición es correcta.

Teorema 3.2.1 (polinomio interpolante). *Sea $n \geq 0$ un número entero. Dados $n + 1$ puntos en el plano $\{(x_i, y_i)\}_{i=0, \dots, n}$ con $x_i \neq x_j$ si $i \neq j$, existe un único polinomio p_n de grado menor o igual que n tal que $p_n(x_i) = y_i$ para todo $i = 0, \dots, n$.*

Definición 3.2.1 (polinomio interpolante). Al polinomio definido por el teorema anterior lo llamamos el **polinomio interpolante** por los puntos $\{(x_i, y_i)\}_{i=0, \dots, n}$. \triangle

Demostración del Teorema 3.2.1. Procedemos por inducción en n .

Paso base. Si $n = 0$, entonces tenemos un único punto (x_0, y_0) y, tal como razonamos antes, el único polinomio de grado 0 cuyo gráfico pasa por ese punto es $p_0(x) = y_0$.

Paso inductivo. Supongamos tenemos demostrado el teorema para n puntos, y veamos que entonces la afirmación también es verdadera para $n + 1$ puntos.

Dados $n + 1$ puntos $\{(x_i, y_i)\}_{i=0, \dots, n}$, consideremos los primeros n de ellos, $\{(x_i, y_i)\}_{i=0, \dots, n-1}$. Nuestra hipótesis inductiva dice que existe un único polinomio p_{n-1} de grado menor o igual que $n - 1$ cuyo gráfico pasa por estos n puntos. Es de esperar que el gráfico de p_{n-1} no pase por el punto (x_n, y_n) . Buscamos ahora un polinomio p_n de grado menor o igual a n , al que vamos a escribir como

$$p_n(x) := p_{n-1}(x) + q(x), \quad (3.1)$$

donde q es otro polinomio de grado menor o igual que n . Remarcamos que todo polinomio de grado n se puede escribir de esta manera.

Si evaluamos la identidad (3.1) en los primeros n puntos, como vale $p_{n-1}(x_i) = y_i$ para todo $i = 0, \dots, n - 1$ y queremos que sea $p_n(x_i) = y_i$ para todo $i = 0, \dots, n - 1$, entonces tiene que cumplirse que $q(x_i) = 0$ para todo $i = 0, \dots, n - 1$. Por lo tanto, como es de grado n , el polinomio q tiene que ser de la forma

$$q(x) = a_n(x - x_0)(x - x_1) \dots (x - x_{n-1}), \quad (3.2)$$

y $a_n \in \mathbb{R}$ es una incógnita a determinar. Para hallar a_n , nos resta imponer que $p_n(x_n) = y_n$: evaluando (3.1) en x_n y usando la expresión para q , tenemos

$$y_n = p_n(x_n) = p_{n-1}(x_n) + a_n(x_n - x_0)(x_n - x_1) \dots (x_n - x_{n-1}).$$

Como $x_n \neq x_i$ para cada $i = 0, \dots, n$, podemos despejar a_n :

$$a_n = \frac{y_n - p_{n-1}(x_n)}{(x_n - x_0)(x_n - x_1) \dots (x_n - x_{n-1})}. \quad (3.3)$$

Así, tenemos determinado un polinomio p_n de grado menor o igual a n que cumple $p_n(x_i) = y_i$ para todo $i = 0, \dots, n$. La unicidad de p_n se deduce de que todo polinomio de grado menor o igual a n se puede escribir como (3.1) y del hecho de que el a_n que hallamos es único. \square

El teorema anterior nos da condiciones para que el problema de interpolar datos con una función polinomial tenga solución única. A continuación, exploramos distintas formas de hallar este polinomio interpolante.

3.2.1. Forma de Vandermonde

Del Teorema 3.2.1, sabemos que para $n + 1$ datos en el plano $\{(x_i, y_i)\}_{i=0, \dots, n}$ con $x_i \neq x_j$ si $i \neq j$, existe un único polinomio p_n de grado menor o igual a n que interpola a esos puntos. Una primera forma de expresar a este polinomio p_n es de forma desarrollada,

$$p_n(x) = \sum_{j=0}^n c_j x^j. \quad (3.4)$$

Naturalmente, basta con hallar los coeficientes c_0, \dots, c_n para determinar a p_n . Si imponemos que $p_n(x_i) = y_i$ para todo $i = 0, \dots, n$, obtenemos un sistema de ecuaciones lineales:

$$\begin{cases} y_0 = p_n(x_0) = c_0 + c_1 x_0 + \dots + c_n x_0^n \\ y_1 = p_n(x_1) = c_0 + c_1 x_1 + \dots + c_n x_1^n \\ \vdots \quad \vdots \quad \vdots \\ y_n = p_n(x_n) = c_0 + c_1 x_n + \dots + c_n x_n^n \end{cases}$$

Remarcamos que aquí las incógnitas son los coeficientes c_0, \dots, c_n del polinomio p_n ; los puntos $(x_0, y_0), \dots, (x_n, y_n)$ son datos de nuestro problema. En forma matricial, el sistema anterior se puede escribir como

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}. \quad (3.5)$$

Definición 3.2.2 (matriz de Vandermonde). A toda matriz que presenta una progresión geométrica en cada fila, tal como la matriz que aparece en el sistema (3.5), se le llama **matriz de Vandermonde**. \triangle

Observación 3.2.1 (invertibilidad de matrices de Vandermonde). Si $x_i \neq x_j$ cuando $i \neq j$, entonces la matriz de Vandermonde asociada es invertible. Esto es una consecuencia inmediata del Teorema 3.2.1, que nos asegura la existencia y unicidad del polinomio interpolante. Si escribimos el polinomio interpolante como en (3.4), entonces el Teorema 3.2.1 puede interpretarse como que el sistema (3.5) sea compatible determinado. En consecuencia, la matriz de dicho sistema, que es la matriz de Vandermonde, es invertible. \triangle

Ejemplo 3.2.1 (forma de Vandermonde). Tenemos la siguiente tabla con tres datos:

i	x_i	y_i
0	1/4	-3/4
1	1	-1
2	5/2	3/2

Sabemos que existe un único polinomio p_2 de grado menor o igual a 2 tal que su gráfico pasa por esos tres puntos. Si escribimos p_2 como en (3.4), nuestras incógnitas son los coeficientes c_0, c_1, c_2 , y las podemos hallar resolviendo el sistema (3.5), que en este ejemplo toma la forma

$$\begin{bmatrix} 1 & 1/4 & 1/16 \\ 1 & 1 & 1 \\ 1 & 5/2 & 25/4 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -3/4 \\ -1 \\ 3/2 \end{bmatrix}.$$

Podemos resolver este sistema en forma sencilla usando cualquiera de las técnicas que discutimos el capítulo anterior, y llegamos a la solución $\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -4/9 \\ -13/9 \\ 8/9 \end{bmatrix}$. Deducimos entonces que el polinomio interpolante por los datos de la tabla es

$$p_2(x) = -\frac{4}{9} - \frac{13}{9}x + \frac{8}{9}x^2. \quad (3.6)$$

Se puede verificar que este polinomio verifica $p_2(x_i) = y_i$ para $i = 0, 1, 2$, y su gráfico es el que está representado por la curva azul en la Figura 3.1. \triangle

La forma de Vandermonde tiene la ventaja de que nos permite reducir el problema de hallar el polinomio interpolante a resolver un sistema lineal de ecuaciones compatible determinado. Sin embargo, no es un método muy usado en la práctica y es especialmente no recomendable en caso de tener que interpolar a través de muchos puntos. Esto se debe fundamentalmente a las dos limitaciones que mencionamos en las siguientes observaciones.

Observación 3.2.2 (condicionamiento de matrices de Vandermonde). Un gran problema de usar la forma de Vandermonde es que, **si bien las matrices de Vandermonde son invertibles, en general están mal condicionadas**. Por ejemplo, consideremos el siguiente código de Octave:

```
>> x = linspace(0,1,21);
>> V = vander(x);
>> condest(V)
ans = 6.7512e+16
```

En la primera línea, usamos el comando `linspace` para definir un vector fila `x` que tiene 21 elementos equiespaciados en el intervalo $[0, 1]$, esto es, $\mathbf{x} = [0, 1/20, \dots, 1]$. La segunda línea usa la función de Octave `vander` para definir la matriz de Vandermonde `V` asociada a los nodos `x(i)`, y la tercera estima el número de condición de `V` asociado a la norma ℓ^1 (recordar la Definición 2.5.3). Observamos que el resultado es bastante desalentador: nuestra estimación es que $\kappa(V)$ es del orden de 10^{16} . Esto implica, por ejemplo y teniendo en cuenta lo que hemos discutido en la Sección 2.5.4, que si quisieramos interpolar por 21 puntos cuyas abscisas sean las del vector `x` y usáramos la forma de Vandermonde resolviendo el sistema (3.5) mediante escalerización gaussiana con pivoteo parcial, nuestra mejor cota para el error relativo en los coeficientes sería del orden de 1. En otras palabras, podríamos obtener un error del 100 % en los coeficientes del polinomio!

Notar, sin embargo, que esto **no** implica que nuestra evaluación del polinomio interpolante sea mala, ya que al hacer escalerización gaussiana con pivoteo parcial el residuo es del orden de ε_M independientemente del condicionamiento de la matriz. \triangle

Observación 3.2.3 (agregar datos en la forma de Vandermonde). Otra limitación de la forma de Vandermonde surge **si queremos agregar puntos por los que interpolar**. Supongamos que tenemos la misma tabla de datos que en el Ejemplo 3.2.1, hallamos p_2 como allí se indica, pero ahora disponemos de un nuevo dato adicional (x_3, y_3) . Esto es, queremos hallar un polinomio p_3 de grado menor o igual a 3 que interpole por los puntos $(x_0, y_0), \dots, (x_3, y_3)$. Si usamos la forma de Vandermonde, entonces el trabajo que hicimos para hallar p_2 no servirá de mucho, porque **vamos a tener que rearmar el sistema** (3.5) pero ahora con la matriz de Vandermonde asociada a los 4 puntos x_0, x_1, x_2, x_3 . En general, al usar (3.4) no existe una relación directa entre la solución del sistema que nos da los coeficientes de p_2 con el que nos dé los coeficientes de p_3 , por lo que tendremos que empezar de cero nuevamente. \triangle

3.2.2. Forma de Lagrange

Las dificultades de la forma de Vandermonde que discutimos en las observaciones 3.2.2 y 3.2.3 tienen que ver esencialmente con que usamos la representación (3.4), esto es, a la elección de la base monomial $\{1, x, x^2, \dots, x^n\}$ para el espacio vectorial de los polinomios de grado menor o igual a n . Esta base monomial no tiene ninguna relación con los puntos que queremos interpolar.

La **forma de Lagrange** consiste en usar una base particular –y que sí depende de los puntos que queremos interpolar– para escribir p_n de otra forma.

Definición 3.2.3 (polinomio de base de Lagrange). Dados x_0, \dots, x_n con $x_i \neq x_j$ si $i \neq j$ y $k \in \{0, \dots, n\}$, el **polinomio de base de Lagrange** L_n^k es el único polinomio de grado menor o igual a n que cumple

$$L_n^k(x_j) = \begin{cases} 1 & \text{si } j = k \\ 0 & \text{si } j \neq k \end{cases}.$$

En otras palabras, L_n^k es el polinomio interpolante por

$$(x_0, 0), \dots, (x_{k-1}, 0), (x_k, 1), (x_{k+1}, 0), \dots, (x_n, 0).$$

△

Observación 3.2.4 (forma explícita de los polinomios de base de Lagrange). Dados x_0, \dots, x_n con $x_i \neq x_j$ si $i \neq j$ y $k \in \{0, \dots, n\}$ se tiene

$$L_n^k(x) = \frac{(x - x_0) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n)}{(x_k - x_0) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_n)}. \quad (3.7)$$

En efecto, basta con observar que el polinomio dado en (3.7) verifica la Definición 3.2.3 y usar la unicidad del polinomio interpolante dada por el Teorema 3.2.1. △

Hallar la funciones de base de Lagrange puede ser laborioso, pero una vez calculadas $\{L_n^k\}_{k=0, \dots, n}$, el polinomio interpolante se puede expresar en forma sumamente sencilla.

Proposición 3.2.2 (interpolación en forma de Lagrange). *Dados $n+1$ puntos en el plano $\{(x_i, y_i)\}_{i=0, \dots, n}$ con $x_i \neq x_j$ si $i \neq j$, el polinomio interpolante por esos puntos es*

$$p_n(x) = \sum_{k=0}^n y_k L_n^k(x), \quad (3.8)$$

donde $\{L_n^k\}_{k=0, \dots, n}$ es la base de Lagrange dada por la Definición 3.2.3.

Demostración. El lado derecho de (3.8) define un polinomio de grado menor o igual a n y se verifica, para cada $i = 0, \dots, n$,

$$\sum_{k=0}^n y_k L_n^k(x_i) = y_i.$$

Por lo tanto, el lado derecho de (3.8) coincide con el polinomio interpolante por $\{(x_i, y_i)\}_{i=0, \dots, n}$. □

Ejemplo 3.2.2 (forma de Lagrange). Volvemos al Ejemplo 3.2.1 y a la tabla de tres datos allí presentada. Usando la expresión (3.7), encontramos que la base de Lagrange asociada

a los puntos $\{1/4, 1, 5/2\}$ es

$$\begin{aligned} L_2^0(x) &= \frac{(x-1)(x-5/2)}{(1/4-1)(1/4-5/2)} = \frac{16}{27}(x-1)(x-5/2), \\ L_2^1(x) &= \frac{(x-1/4)(x-5/2)}{(1-1/4)(1-5/2)} = -\frac{8}{9}(x-1/4)(x-5/2), \\ L_2^2(x) &= \frac{(x-1/4)(x-1)}{(5/2-1)(5/2-1/4)} = \frac{8}{27}(x-1/4)(x-1). \end{aligned}$$

La Figura 3.2 muestra estas tres funciones de base. Por lo tanto, usando los valores de las coordenadas y de la tabla de datos, deducimos que el polinomio interpolante es

$$p_2(x) = \sum_{k=0}^2 y_k L_2^k(x) = -\frac{3}{4}L_2^0(x) - L_2^1(x) + \frac{3}{2}L_2^2(x).$$

Dejamos para el lector verificar que esta expresión coincide con (3.6). \triangle

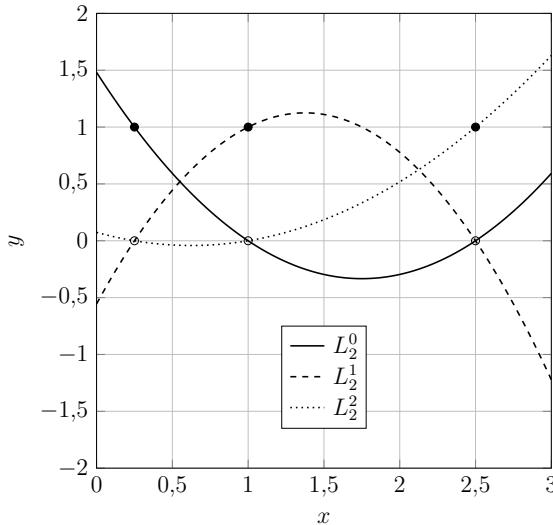


Figura 3.2: Funciones de base de Lagrange correspondientes al Ejemplo 3.2.1/3.2.2.

Observación 3.2.5 (algunas ventajas de la forma de Lagrange). Escribir el polinomio interpolante en la forma de Lagrange tiene la ventaja de que nos da una expresión compacta, y de que no tenemos problemas de condicionamiento para determinar los coeficientes en caso de que haya un número elevado de puntos. Además, como la base de Lagrange solamente depende de los $\{x_i\}$, si cambiamos el valor de un cierto y_k –pero dejando fijos los $\{x_i\}$ –, encontrar el nuevo polinomio interpolante es extremadamente sencillo: simplemente debemos cambiar el factor multiplicando a L_n^k en (3.8). \triangle

Observación 3.2.6 (algunas desventajas de la forma de Lagrange). Si queremos obtener una expresión explícita para el polinomio interpolante usando la forma de Lagrange,

entonces la evaluación de (3.7) para hallar la base correspondiente puede ser costosa. Además, si queremos evaluar p_n en algún punto que no sea x_0, \dots, x_n , entonces será más costoso computacionalmente hacerlo si lo tenemos escrito en la base de Lagrange que si lo tenemos escrito en la base monomial. En forma análoga, la forma de Lagrange puede ser más difícil de manipular a la hora de integrar o derivar el polinomio interpolante, lo que es un uso relevante de esta herramienta (ver la Sección 3.1).

Por otra parte, al igual que ocurre con la forma de Vandermonde, la forma de Lagrange no es demasiado amigable con agregar nuevos puntos para interpolar: si se agrega un nuevo punto, será necesario recalcular la nueva base. \triangle

3.2.3. Forma de Newton

Las formas de Vandermonde y de Lagrange comparten la limitación de que no son prácticas de manipular en caso de que vayamos agregando nuevos puntos por los que queremos interpolar. Esto está en contraste con el espíritu de nuestra demostración del Teorema 3.2.1: allí, hicimos una inducción en la cantidad de puntos, agregando datos secuencialmente. La **forma de Newton** consiste en construir el polinomio interpolante p_n mediante este mismo proceso inductivo. Volvamos a la demostración del Teorema 3.2.1, pero ahora con una perspectiva algorítmica.

- Si tenemos un único punto (x_0, y_0) , entonces el polinomio interpolante es $p_0(x) = y_0$.
- Si ya tenemos el polinomio interpolante p_{n-1} a través de los primeros n puntos entonces, usando (3.1), (3.2) y (3.3), tenemos

$$p_n(x) = p_{n-1}(x) + \frac{(y_n - p_{n-1}(x_n))(x - x_0)(x - x_1) \dots (x - x_{n-1})}{(x_n - x_0)(x_n - x_1) \dots (x_n - x_{n-1})}.$$

Otra forma de escribir esta última identidad es

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots + a_n(x - x_0) \dots (x - x_{n-1}). \quad (3.9)$$

Ejercicio 3.1. Sean los coeficientes $\{a_i\}_{i=0,\dots,n}$ dados por (3.3). Verificar que el lado derecho de (3.9) es efectivamente una expresión para el polinomio interpolante por los puntos $\{(x_i, y_i)\}_{i=0,\dots,n}$.

[Sugerencia: evaluar el lado derecho en x_0, \dots, x_n y usar la unicidad del polinomio interpolante.]

Observación 3.2.7 (sobre la demostración del Teorema 3.2.1). Nuestra demostración del Teorema 3.2.1 está relacionada con la forma de Newton y motivada por ella. Ese teorema se puede demostrar de varias formas alternativas. En particular, referimos a [Hea02, Sección 7.3.1] para una demostración basada en la forma de Vandermonde y a [SM03, Teorema 6.1] para una inspirada en la forma de Lagrange. \triangle

Observación 3.2.8 (algoritmo de Horner). La expresión (3.9) da lugar a un método eficiente para evaluar p_n , el llamado **algoritmo de Horner**. Éste consiste en reescribir dicha fórmula como

$$p_n(x) = a_0 + (x - x_0) \left[a_1 + (x - x_1) \left[a_2 + (x - x_2) [a_3 + \dots] \right] \right].$$

El Algoritmo 3.1 muestra cómo esta expresión puede ser implementada de forma muy sencilla.

Algoritmo 3.1: Pseudo-código: algoritmo de Horner

Datos: $a_0, \dots, a_n \in \mathbb{R}$, $x \in \mathbb{R}$

Resultado: $y = p_n(x)$

$y \leftarrow a_n;$

para $j = n - 1 : -1 : 0$ **hacer**

$| \quad y \leftarrow a_j + (x - x_j)y;$

fin

△

Observación 3.2.9 (diferencias divididas). Una vez que tenemos determinados los coeficientes $\{a_i\}_{i=0,\dots,n}$, el algoritmo de Horner nos permite evaluar p_n de forma eficiente. Para determinar estos coeficientes, bien podríamos utilizar (3.3) directamente, aunque en esta expresión también hay una “recursión escondida”.

Esta recursión se explota en el llamado **método de diferencias divididas**, que describimos a continuación. Se parte² de $f[x_i] = f(x_i) = y_i$ y se toma, para $i = 1, \dots, n$,

$$f[x_k, x_{k+1}, \dots, x_{k+i}] := \frac{f[x_{k+1}, x_{k+2}, \dots, x_{k+i}] - f[x_k, x_{k+1}, \dots, x_{k+i-1}]}{x_{k+i} - x_k}, \quad k = 0, \dots, n - i.$$

Una vez halladas estas *diferencias divididas*, simplemente se tiene

$$a_0 = f[x_0], \quad a_1 = f[x_0, x_1], \dots, \quad a_n = f[x_0, x_1, \dots, x_n].$$

△

Ejemplo 3.2.3 (forma de Newton). Volvamos al Ejemplo 3.2.1/3.2.2, y expresemos el polinomio interpolante p_2 en la forma de Newton.

En primer lugar, hallamos los coeficientes a_0, a_1, a_2 mediante el método de diferencias divididas. Recordando los valores de la tabla de datos de este ejemplo, tenemos

$$\begin{aligned} f[x_0] &= -3/4, \quad f[x_1] = -1, \quad f[x_2] = 3/2 \quad (i = 0, k = 0), \\ f[x_0, x_1] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{-1 - (-3/4)}{1 - 1/4} = -\frac{1}{3} \quad (i = 1, k = 0), \\ f[x_1, x_2] &= \frac{f[x_2] - f[x_1]}{x_2 - x_1} = \frac{3/2 - (-1)}{5/2 - 1} = \frac{5}{3} \quad (i = 1, k = 1), \\ f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{5/3 - (-1/3)}{5/2 - 1/4} = \frac{8}{9}, \quad (i = 2, k = 0). \end{aligned}$$

²A no asustarse con la expresión $f[\cdot]$: ¡es solamente notación!

Deducimos que los coeficientes de p_2 en la forma de Newton son

$$a_0 = f[x_0] = -\frac{3}{4}, \quad a_1 = f[x_0, x_1] = -\frac{1}{3}, \quad a_2 = f[x_0, x_1, x_2] = \frac{8}{9},$$

y por lo tanto³

$$p_2(x) = a_0 + (x - x_0)[a_1 + a_2(x - x_1)] = -\frac{3}{4} + \left(x - \frac{1}{4}\right) \left[-\frac{1}{3} + \frac{8}{9}(x - 1)\right].$$

Queda como ejercicio para el lector verificar que esta expresión coincide con (3.6). \triangle

3.3. Interpolación de funciones

Hasta aquí nos hemos concentrado en el problema de hallar el polinomio interpolante a través de ciertos datos. A continuación, profundizaremos en un aspecto central en muchas aplicaciones: la capacidad de *aproximar* del polinomio interpolante.

Concretamente, en esta sección asumimos que los datos están tomados al muestrear una cierta función $f: \mathbb{R} \rightarrow \mathbb{R}$ desconocida. Esto es, dados $n + 1$ puntos en el plano $\{(x_i, f(x_i))\}_{i=0, \dots, n}$ con $x_i \neq x_j$ si $i \neq j$, consideramos el único polinomio p_n de grado menor o igual a n que cumple $p(x_i) = f(x_i)$ para todo $i = 0, \dots, n$ y nos preguntamos: **¿qué tan cerca están f y p_n ?**

Una pregunta importante en este punto es cómo definir “estar cerca” o “estar lejos” para dos funciones. De nuestra experiencia de Cálculo, sabemos que las normas inducen distancias⁴; en este curso vamos a trabajar con la siguiente norma de funciones.

Definición 3.3.1 (norma del supremo). Sea $f: D \rightarrow \mathbb{R}$ una función. Definimos la **norma del supremo de f en D** , que escribimos como $\|f\|_{L^\infty(D)}$, mediante

$$\|f\|_{L^\infty(D)} := \sup_{x \in D} |f(x)|. \tag{3.10}$$

\triangle

Observación 3.3.1. La norma del supremo (3.10) es análoga a la norma ℓ^∞ en \mathbb{R}^n que consideramos en el Capítulo 2, con la obvia diferencia de que el espacio de funciones $D \rightarrow \mathbb{R}$ es de dimensión infinita. Si estamos trabajando con funciones continuas en un dominio cerrado y acotado, por el Teorema de Weierstrass el supremo es en realidad un máximo. \triangle

³Aquí usamos la expresión anidada de Horner solamente para ilustrarla, pero bien podríamos haber utilizado la fórmula (3.9).

⁴Por ejemplo: si a \mathbb{R}^n le damos una norma ℓ^p como las que introdujimos en el Capítulo 2, podríamos definir la distancia entre dos puntos $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ como $d_p(\mathbf{x}, \mathbf{y}) := \|\mathbf{x} - \mathbf{y}\|_p$. Para $p = 2$, esto se corresponde con nuestra noción habitual de distancia en el plano/espacio.

3.3.1. Error de interpolación

En esta sección, vamos a analizar el *error de interpolación polinomial*. Dadas las funciones f y p_n , la norma $\|f - p_n\|_{L^\infty([a,b])}$ consiste en tomar todos los puntos $x \in [a, b]$, evaluar la discrepancia $|f(x) - p_n(x)|$, y luego tomar la mayor de estas discrepancias. Para estimar el error de interpolación polinomial, procedemos de forma análoga: en primer lugar, damos una expresión de la discrepancia para puntos arbitrarios (Teorema 3.3.1) y luego estimamos qué tan grande puede ser esta discrepancia.

Para simplificar la notación, dadas f y p_n como en el párrafo anterior, definimos⁵ el **error de interpolación polinomial** como

$$e_n := f - p_n.$$

Notar que $e_n: [a, b] \rightarrow \mathbb{R}$ es una función: para cada punto $x \in [a, b]$ tenemos un valor $e_n(x)$. Teniendo en cuenta la Definición 3.3.1, nos interesa estimar $\|e_n\|_{L^\infty([a,b])}$.

Teorema 3.3.1 (error de interpolación polinomial). *Sean f de clase C^{n+1} en un intervalo $[a, b]$, los puntos $x_0 < x_1 < \dots < x_n$ en el intervalo $[a, b]$, y p_n el polinomio interpolante por $\{(x_i, f(x_i))\}_{i=0, \dots, n}$. Entonces, para cada $x \in [a, b]$ existe un $\gamma_x \in (a, b)$ tal que*

$$e_n(x) = f(x) - p_n(x) = \frac{f^{(n+1)}(\gamma_x)}{(n+1)!} (x - x_0)(x - x_1) \dots (x - x_n). \quad (3.11)$$

La fórmula (3.11) nos dice que el error de interpolación en un punto dado $x \in [a, b]$ depende de la derivada $(n+1)$ -ésima de f en un cierto punto (desconocido y que depende de x) del intervalo $[a, b]$ y del producto de las distancias (con signo) entre x y los nodos de interpolación x_0, \dots, x_n . Si nos interesa *acotar* el error de interpolación polinomial, entonces debemos estimar la magnitud de estos términos: volvemos a este punto en la Observación 3.3.2 más adelante. Por el momento, y para alivianar un poco la notación, introducimos la siguiente definición.

Definición 3.3.2 (polinomio nodal). Sean los puntos $x_0 < x_1 < \dots < x_n$ en el intervalo $[a, b]$. Definimos el **polinomio nodal** por x_0, \dots, x_n como $\omega_n: [a, b] \rightarrow \mathbb{R}$,

$$\omega_n(x) := \prod_{i=0}^n (x - x_i).$$

△

Demostración del Teorema 3.3.1. Fijemos un punto $x \in [a, b]$. Es claro que si $x = x_i$ para algún i entonces $f(x_i) - p_n(x_i) = 0$ y vale (3.11), por lo que podemos suponer que $x \neq x_i$ para todo $i = 0, \dots, n$.

⁵Observar que aquí estamos definiendo el error con el signo opuesto a como lo hicimos en el Capítulo 1. De todas formas, nos interesa la magnitud (valor absoluto) de este error y no su signo.

Consideramos la función auxiliar $G : [a, b] \rightarrow \mathbb{R}$ dada por

$$G(t) := e_n(t) - e_n(x) \frac{\omega_n(t)}{\omega_n(x)}. \quad (3.12)$$

Remarcamos que la variable que toma la función G es t , y que el punto x permanece fijo. Como f es de clase C^{n+1} , resulta que G también es de clase C^{n+1} . Por otra parte, observamos que $G(x_i) = 0$ para todo $i = 0, \dots, n$, y que además $G(x) = 0$. Por lo tanto, G tiene al menos $n + 2$ raíces distintas en el intervalo $[a, b]$.

Sean $z_0 < z_1 < \dots < z_{n+1}$ las $n + 2$ raíces que sabemos que G tiene en el intervalo $[a, b]$. Aplicando el Teorema de Rolle en cada subintervalo $[z_i, z_{i+1}]$ ($i = 0, \dots, n$), podemos asegurar que G' tiene al menos $n + 1$ raíces distintas en el intervalo (a, b) . Aplicamos nuevamente el Teorema de Rolle en cada subintervalo definido por las raíces de G' y hallamos que G'' tiene al menos n raíces distintas en (a, b) . Podemos seguir así hasta llegar a la derivada $(n + 1)$ -ésima de G , y deducir que existe al menos una raíz $\gamma_x \in (a, b)$ de la función $G^{(n+1)}$.

Observemos la definición (3.12) de la función G : involucra a los polinomios p_n , que es de grado n , y ω_n , que es de grado $n + 1$. Luego, la derivada de orden $n + 1$ de p_n es nula, y la de ω_n es una constante:

$$\omega_n^{(n+1)}(t) = (n + 1)!$$

Por lo tanto, tenemos

$$G^{(n+1)}(t) = f^{(n+1)}(t) - e_n(x) \frac{(n + 1)!}{\omega_n(x)},$$

y evaluando en γ_x ,

$$0 = G^{(n+1)}(\gamma_x) = f^{(n+1)}(\gamma_x) - e_n(x) \frac{(n + 1)!}{\omega_n(x)}.$$

Finalmente, despejamos $e_n(x)$ de esta última igualdad y obtenemos (3.11). \square

La utilidad principal del Teorema 3.3.1 radica en poder acotar el error de interpolación.

Corolario 3.3.2 (estimación del error de interpolación polinomial). *Bajo las mismas hipótesis del Teorema 3.3.1, tenemos las estimaciones*

$$|e_n(x)| \leq \frac{|\omega_n(x)| \|f^{(n+1)}\|_{L^\infty([a,b])}}{(n + 1)!} \quad \forall x \in [a, b], \quad (3.13)$$

y

$$\|e_n\|_{L^\infty([a,b])} \leq \frac{\|\omega_n\|_{L^\infty([a,b])} \|f^{(n+1)}\|_{L^\infty([a,b])}}{(n + 1)!}. \quad (3.14)$$

Demostración. Ambas estimaciones son consecuencia directa de (3.11). Dado $x \in [a, b]$, tomando valor absoluto en (3.11) y acotando $|f^{(n+1)}(\gamma_x)| \leq \|f^{(n+1)}\|_{L^\infty([a,b])}$, deducimos (3.13). Luego, como además vale $|\omega_n(x)| \leq \|\omega_n\|_{L^\infty([a,b])}$, tomando supremo en $x \in [a, b]$ obtenemos (3.14). \square

Ejemplo 3.3.1 (error de interpolación). Consideramos la función $f: [0, 1] \rightarrow \mathbb{R}$, $f(x) = \sin(x) \cos(x)$, y tomamos $n + 1$ puntos equiespaciados en dicho intervalo,

$$x_i := \frac{i}{n}, \quad i = 0, \dots, n.$$

Consideramos el polinomio interpolante p_n a través de $\{(x_i, f(x_i))\}_{i=0, \dots, n}$. La Figura 3.3 muestra la función f y los polinomios interpolantes p_1, p_2, p_3 , que son lineal, cuadrático y cúbico, respectivamente. Nos preguntamos cómo acotar el error de interpolación polinómica en la norma del supremo en función de n .

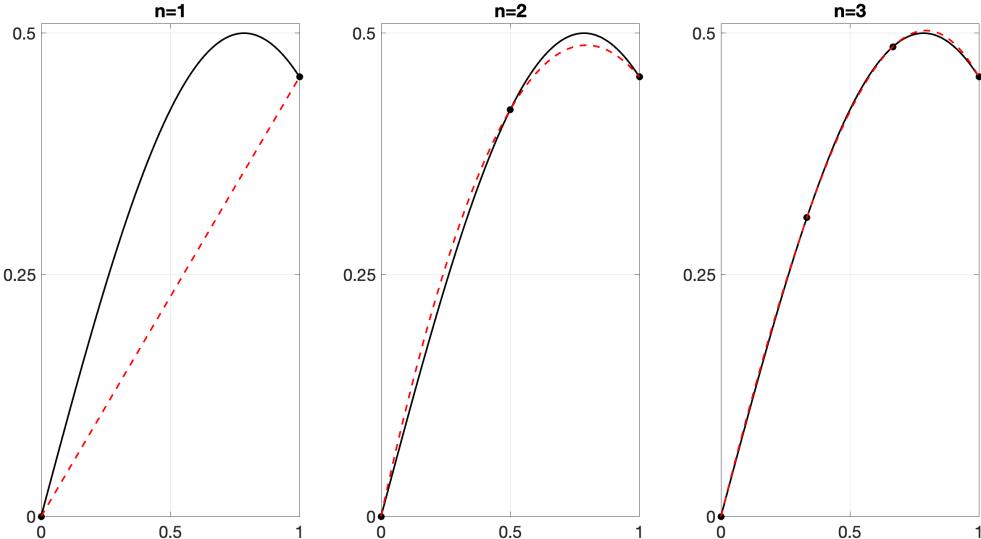


Figura 3.3: Interpolación polinomial de la función $f: [0, 1] \rightarrow \mathbb{R}$, $f(x) = \sin(x) \cos(x)$.

Es sencillo verificar que f es infinitamente derivable⁶ y que se cumplen

$$f'(x) = \cos^2(x) - \sin^2(x), \quad f''(x) = -4 \sin(x) \cos(x), \quad f'''(x) = -4(\cos^2(x) - \sin^2(x)), \dots$$

Acotando (grosamente) el valor absoluto de las funciones seno y coseno por 1, deducimos que para todo $x \in [0, 1]$ valen

$$|f'(x)| \leq 2, |f''(x)| \leq 4, |f'''(x)| \leq 8 \dots \text{ y, en general, } |f^{(n)}(x)| \leq 2^n \quad \forall n \geq 1,$$

⁶Esto se suele escribir como $f \in C^\infty$.

lo que podemos escribir como $\|f^{(n)}\|_{L^\infty([0,1])} \leq 2^n \forall n \geq 1$. Por otra parte, como estamos trabajando en el intervalo $[0, 1]$, para todo $x \in [0, 1]$ podemos acotar (nuevamente, groseramente)

$$|x - x_i| \leq 1 \quad \forall i = 0, \dots, n,$$

lo que implica

$$\|\omega_n\|_{L^\infty([0,1])} = \max_{x \in [0,1]} |\omega_n(x)| = \max_{x \in [0,1]} \prod_{i=0}^n |x - x_i| \leq 1.$$

Por lo tanto, usando (3.14), obtenemos la estimación

$$\|e_n\|_{L^\infty([0,1])} \leq \frac{2^{n+1}}{(n+1)!}.$$

Por ejemplo, si usamos polinomios de grados $n = 5$, $n = 10$, o $n = 20$, esta última desigualdad quiere decir que

$$\begin{aligned} |f(x) - p_5(x)| &\leq \frac{2^6}{6!} \approx 8,9 \times 10^{-2} \quad \forall x \in [0, 1], \\ |f(x) - p_{10}(x)| &\leq \frac{2^{11}}{11!} \approx 5,1 \times 10^{-5} \quad \forall x \in [0, 1], \\ |f(x) - p_{20}(x)| &\leq \frac{2^{21}}{21!} \approx 4,1 \times 10^{-14} \quad \forall x \in [0, 1]. \end{aligned}$$

△

Observación 3.3.2 (comportamiento del error de interpolación). En el Ejemplo 3.3.1, obtuvimos que al aumentar la cantidad de puntos por los que interpolamos f , el error de interpolación decrece. En principio esto parece muy natural, ya que al aumentar la cantidad de puntos estamos capturando “más información” sobre f . Sin embargo, esto no se deduce ni del Teorema 3.3.1 ni del Corolario 3.3.2. Notemos que en los lados derechos de las estimaciones (3.11), (3.13), (3.14) aparecen:

- un factor $1/(n+1)!$, que obviamente decrece al aumentar n ;
- el polinomio nodal ω_n , del que no sabemos mucho cómo se comporta al variar n , y al que tal vez podamos aspirar a controlar como hicimos en el Ejemplo 3.3.1;
- un término involucrando la derivada $(n+1)$ -ésima de f , que evidentemente depende de f y por lo tanto está fuera de nuestro control.

Si estamos en una situación como la del Ejemplo 3.3.1, en que podemos estimar $\|\omega_n\|_{L^\infty([0,1])}$ de forma razonable, y las normas del supremo de las derivadas de la función f no crecen “demasiado rápido” respecto a n , entonces es de esperar que al aumentar la cantidad de puntos por los que interpolamos, el error de interpolación decrezca. En estos casos, vale la pena hacer una interpolación polinomial de grado alto. △

3.3.2. Fenómeno de Runge

La Observación 3.3.2 nos deja entrever que quizás no siempre sea buena idea aumentar el grado polinomial al interpolar una función. En esta sección mostramos que, efectivamente, al aumentar el grado polinomial no necesariamente se gana en precisión. Específicamente, el llamado **fenómeno de Runge**, que describimos a continuación, tiene que ver con la presencia de oscilaciones cerca de los bordes del intervalo al realizar interpolación polinomial de grado alto.

El marco en el que trabajamos es similar al del Ejemplo 3.3.1. Consideramos la llamada *función de Runge* $f: [-1, 1] \rightarrow \mathbb{R}$,

$$f(x) = \frac{1}{1 + 25x^2},$$

y la interpolamos usando $n + 1$ puntos equiespaciados en $[-1, 1]$,

$$x_i := -1 + \frac{2i}{n}, \quad i = 0, \dots, n.$$

La Figura 3.4 nos muestra la interpolación polinomial que se obtiene al tomar $n = 5$, $n = 10$, $n = 20$. Observamos que el polinomio interpolante genera unas oscilaciones siniestras cerca de los extremos del intervalo $[-1, 1]$. En particular, notamos que no se cumple $\|e_n\|_{L^\infty([-1,1])} \rightarrow 0$ con $n \rightarrow \infty$. De hecho, se tiene $\|e_n\|_{L^\infty([-1,1])} \rightarrow \infty$.

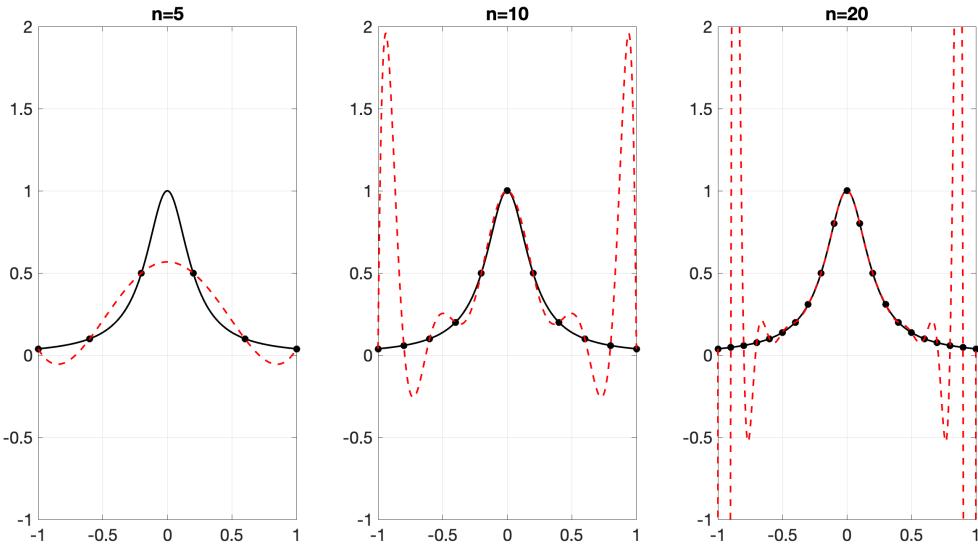


Figura 3.4: Fenómeno de Runge: interpolar la función de Runge sobre puntos equiespaciados.

¿A qué se debe tan mal comportamiento del polinomio interpolante? Recordemos los tres puntos que marcamos en la Observación 3.3.2. Se puede demostrar –es una cuenta larga y

tediosa— que las derivadas de la función de Runge *crecen rápidamente* al aumentar n . De hecho, la magnitud de estas derivadas crece más rápido que $(n + 1)!$, que era el término “bueno” que teníamos a nuestro favor. Por lo tanto, para que el error de interpolación decrezca necesitamos la ayuda del polinomio nodal. La elección de nodos equiespaciados hace que el polinomio nodal ω_n no nos dé la ayuda que necesitamos, y el resultado es desastroso.

Surge, entonces, la pregunta de **qué hacer cuando interpolamos una función cuyas derivadas de orden alto crecen rápidamente en magnitud**. Proponemos dos alternativas:

1. **Mantenerse con interpolaciones de grado bajo.** Esto logra evitar las oscilaciones que observamos en la Figura 3.4. Como una interpolación de grado bajo en un intervalo “grande” es poco precisa, la estrategia habitual consiste en dividir el intervalo en subintervalos pequeños y en cada uno de estos subintervalos aplicar una interpolación de grado bajo. Vamos a explorar en detalle esta estrategia en la Sección 3.4.
2. Elegir los nodos de interpolación de forma más astuta, de modo de lograr que $\|\omega_n\|_{L^\infty([-1,1])}$ sea lo menor posible. Comentamos muy brevemente esta alternativa. Los **nodos de Chebyschev** en el intervalo $[-1, 1]$ son

$$x_i = -\cos\left(\frac{\pi i}{n}\right), \quad i = 0, \dots, n,$$

y tienen la propiedad de *minimizar*

$$\max_{x \in [-1,1]} \prod_{i=0}^n |x - x_i| = \|\omega_n\|_{L^\infty([-1,1])}.$$

La Figura 3.5 nos muestra las interpolaciones polinomiales p_5 , p_{10} , p_{20} de la función de Runge utilizando los nodos de Chebyschev. Notemos que no se observan las oscilaciones presentes en la Figura 3.4 y que efectivamente una buena elección de los nodos por los que interpolamos nos puede permitir estabilizar estas oscilaciones. De hecho, se puede demostrar que con esta elección se tiene la convergencia $\|e_n\|_{L^\infty([-1,1])} \rightarrow 0$ con $n \rightarrow \infty$. Referimos a [QS06, Sección 3.1.2] para más detalles.

3.4. Interpolación a trozos

Como adelantamos al final de la Sección 3.3.2, una alternativa a considerar interpolaciones polinomiales de *grado alto* en un intervalo *grande* consiste en partir este intervalo en subintervalos *pequeños* y considerar interpolaciones de *grado bajo* en cada uno de estos subintervalos. Al hacer esto, se puede buscar también imponer ciertas propiedades de

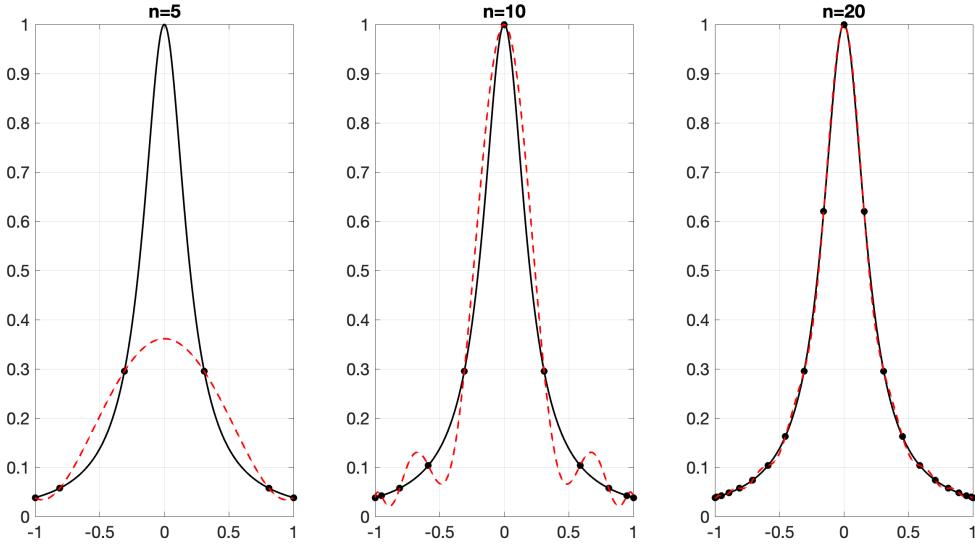


Figura 3.5: Interpolación polinomial de la función de Runge sobre nodos de Chebyschev.

suavidad o regularidad sobre la función interpolante. Esta estrategia se conoce como **interpolación a trozos**.

A la luz del fenómeno de Runge (Sección 3.3.2), una ventaja aparente de considerar interpolaciones polinomiales a trozos de grado bajo es que se mitiga la aparición de oscilaciones, asociadas a interpolaciones de grado alto de funciones con poca regularidad (recordar la Observación 3.3.2).

¿De qué grado polinomial deberíamos considerar nuestras interpolantes? Si tenemos un conjunto de datos $\{(x_i, y_i)\}_{i=0, \dots, n}$, parece natural la idea de “unir los puntos” con segmentos de recta: esto se corresponde a una interpolación lineal a trozos, que analizamos en la Sección 3.4.1. El resultado de esta interpolación es una curva que no es derivable en los puntos x_0, \dots, x_n , como la que mostramos en la Figura 3.6 más abajo.

Supongamos que deseamos que nuestra función interpolante —a la que llamaremos p — sea un polinomio a trozos y con mejores propiedades de regularidad. Para fijar ideas, supongamos que queremos que p sea una función derivable. Como p es un polinomio en cada intervalo $[x_i, x_{i+1}]$ ($i = 0, \dots, n - 1$), basta con lograr que p sea derivable en los *nodos interiores* x_1, \dots, x_{n-1} . Fijemos un $i \in \{1, \dots, n - 1\}$ y analicemos lo que ocurre en el nodo x_i . Imponer que p sea derivable en x_i es equivalente a lograr que

$$p|_{[x_{i-1}, x_i]}(x_i) = p|_{[x_i, x_{i+1}]}(x_i) \quad \text{y} \quad p'|_{[x_{i-1}, x_i]}(x_i) = p'|_{[x_i, x_{i+1}]}(x_i). \quad (3.15)$$

Por lo tanto, si queremos que el polinomio interpolante sea derivable, debemos prescribir de alguna forma el valor de $p'(x_i)$; llamemos d_i a ese valor, que por el momento asumimos conocido. Si nos restringimos a un intervalo $[x_i, x_{i+1}]$ ($i = 0, \dots, n - 1$), requerimos que p

cumpla 4 condiciones:

$$p(x_i) = y_i, \quad p(x_{i+1}) = y_{i+1}, \quad p'(x_i) = d_i, \quad p'(x_{i+1}) = d_{i+1}.$$

Como tenemos 4 restricciones, si queremos tener un problema con solución única parece natural que lo podamos lograr con 4 grados de libertad. Por lo tanto, en cada intervalo $[x_i, x_{i+1}]$ ($i = 0, \dots, n - 1$) el polinomio p debe ser un polinomio cúbico.

Más en general, **al hacer interpolaciones polinomiales a trozos, las funciones interpolantes se suelen tomar como polinomios de grado impar en cada subintervalo.** En esta sección nos restringimos a funciones lineales y cúbicas a trozos.

3.4.1. Interpolación lineal a trozos

Conceptualmente, la interpolación lineal a trozos es extremadamente simple: dados $n + 1$ puntos en el plano $\{(x_i, y_i)\}_{i=0, \dots, n}$ con $x_0 < x_1 < \dots < x_n$, se considera la función $L: [x_0, x_n] \rightarrow \mathbb{R}$ cuyo gráfico se corresponde con “unir los puntos” con segmentos de recta en orden.

Definición 3.4.1 (interpolación lineal a trozos). Dados $n + 1$ puntos en el plano $\{(x_i, y_i)\}_{i=0, \dots, n}$ con $x_0 < x_1 < \dots < x_n$, su **interpolación lineal a trozos** es la función $L: [x_0, x_n] \rightarrow \mathbb{R}$ que en cada intervalo $[x_i, x_{i+1}]$ ($i = 0, \dots, n - 1$) coincide con el polinomio interpolante por los puntos (x_i, y_i) , (x_{i+1}, y_{i+1}) . En otras palabras, L es la función dada por

$$L(x) := y_i + (x - x_i) \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right), \quad \forall x \in [x_i, x_{i+1}] \quad (i = 0, \dots, n - 1). \quad (3.16)$$

△

Observación 3.4.1 (no derivabilidad). La interpolante lineal a trozos L es una función continua, pero no es derivable en los nodos interiores x_1, \dots, x_{n-1} . △

Dados $x_0 < x_1 < \dots < x_n$, la implementación de un algoritmo que computa la interpolación lineal a trozos por $(x_0, y_0), \dots, (x_n, y_n)$, evaluada en un cierto punto $v \in [x_0, x_n]$, se puede lograr como se muestra en el Algoritmo 3.2. Allí, hacemos uso de las siguientes variables auxiliares:

$$\begin{aligned} &\text{la variable local } s := x - x_i, \quad \text{para } x \in [x_i, x_{i+1}], \\ &\text{las diferencias divididas } \delta_i := \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad \text{para } i = 0, \dots, n - 1. \end{aligned}$$

Usando s como nueva variable, podemos reescribir (3.16) compactamente como

$$L(s) = y_i + s\delta_i, \quad \forall s \in [0, x_{i+1} - x_i] \quad (i = 0, \dots, n - 1).$$

Algoritmo 3.2: Implementación de una interpolación lineal a trozos.

Datos: $\{(x_i, y_i)\}_{i=0, \dots, n}$ con $x_0 < x_1 < \dots < x_n$, y un punto $v \in [x_0, x_n]$

Resultado: $w = L(v)$ según la Definición 3.4.1

```

 $\delta \leftarrow \text{diff}(y) ./ \text{diff}(x)$  % computar las diferencias divididas;
% hallar el índice i tal que  $x(i) \leq v < x(i+1)$ ;
i = 1;
mientras  $x(i) < v$  hacer
|  $i \leftarrow i + 1$ 
fin
% computar la variable local y evaluar L;
s  $\leftarrow v - x(i)$ ;
w  $\leftarrow y(i) + s\delta(i)$  ;

```

Ejercicio 3.2. El Algoritmo 3.2 computa *todas* las diferencias divididas, lo que es innecesario si solamente nos interesa evaluar L en un punto. Sin embargo, esto puede ser útil si queremos evaluar L en varios puntos a la vez. Modificando el Algoritmo 3.2, implementar un código que permita evaluar L en un vector de puntos $v \in \mathbb{R}^m$ de forma eficiente, asumiendo que las entradas de v están ordenadas de forma creciente y pertenecen al intervalo $[x_0, x_n]$.

En forma similar a la Sección 3.3, cabe preguntarse qué tan bien podemos aproximar a una cierta función f si usamos su interpolación lineal a trozos. Concretamente, si tenemos que $y_i = f(x_i)$ para todo $i = 0, \dots, n$, y construimos la interpolante lineal a trozos L , queremos estimar la discrepancia entre f y L . Nuestra herramienta principal es el Teorema 3.3.1.

En efecto, supongamos que f es de clase C^2 , tomemos $n + 1$ puntos $x_0 < x_1 < \dots < x_n$, y L la interpolante lineal a trozos por $\{(x_i, f(x_i))\}_{i=0, \dots, n}$. Consideremos un punto $x \in [x_0, x_n]$ fijo: si $x = x_i$ para algún $i = 0, \dots, n$, entonces el error de interpolación es nulo. Si x no coincide con ninguno de los nodos, entonces $x \in (x_i, x_{i+1})$ para algún $i = 0, \dots, n - 1$. Observemos que en ese intervalo estamos haciendo una interpolación lineal ($n = 1$) de f , por lo que podemos usar (3.11): existe un $\gamma_x \in (x_i, x_{i+1})$ tal que

$$f(x) - L(x) = \frac{f''(\gamma_x)}{2} (x - x_i)(x - x_{i+1}). \quad (3.17)$$

Esta fórmula nos da una representación del error de interpolación; si queremos estimar su magnitud, entonces debemos acotar el lado derecho de la igualdad de forma análoga a como hicimos en la demostración del Corolario 3.3.2.

Teorema 3.4.1 (estimación del error de interpolación lineal a trozos). *Sean $n + 1$ puntos $x_0 < x_1 < \dots < x_n$, f una función de clase C^2 en el intervalo $[x_0, x_n]$, y L la interpolante lineal a trozos por los puntos $\{(x_i, f(x_i))\}_{i=0, \dots, n}$. Entonces, si $x \in [x_i, x_{i+1}]$ para $i \in \{0, \dots, n - 1\}$, se tiene*

$$|f(x) - L(x)| \leq \frac{\|f''\|_{L^\infty([x_i, x_{i+1}])}}{2} (x - x_i)(x_{i+1} - x). \quad (3.18)$$

Además,

$$\|f - L\|_{L^\infty([x_0, x_n])} \leq \frac{\|f''\|_{L^\infty([x_0, x_n])}}{8} \max_{j=0, \dots, n-1} |x_{j+1} - x_j|^2. \quad (3.19)$$

Demostración. Sea $x \in [x_i, x_{i+1}]$ para algún $i \in \{0, \dots, n-1\}$. Tomando valor absoluto en (3.17), y como $\gamma_x \in (x_i, x_{i+1})$ podemos acotar

$$|f''(\gamma_x)| \leq \|f''\|_{L^\infty([x_i, x_{i+1}])},$$

y se deduce (3.18). Para probar (3.19), observemos que la función nodal

$$\omega(x) := (x - x_i)(x_{i+1} - x), \quad x \in [x_i, x_{i+1}],$$

alcanza su máximo en $x = \frac{x_i + x_{i+1}}{2}$, el punto medio del intervalo $[x_i, x_{i+1}]$, y este máximo vale $\frac{|x_{i+1} - x_i|^2}{4}$. Por lo tanto, si $x \in [x_i, x_{i+1}]$, tenemos

$$|f(x) - L(x)| \leq \frac{\|f''\|_{L^\infty([x_i, x_{i+1}])}}{2} \frac{|x_{i+1} - x_i|^2}{4}.$$

Finalmente, usando que $\|f''\|_{L^\infty([x_i, x_{i+1}])} \leq \|f''\|_{L^\infty([x_0, x_n])}$ y que $|x_{i+1} - x_i|^2 \leq \max_{j=0, \dots, n-1} |x_{j+1} - x_j|^2$, obtenemos

$$|f(x) - L(x)| \leq \frac{\|f''\|_{L^\infty([x_0, x_n])}}{8} \max_{j=0, \dots, n-1} |x_{j+1} - x_j|^2.$$

Como esto vale para todo $x \in [x_0, x_n]$, deducimos (3.19). \square

Ejemplo 3.4.1 (interpolación lineal a trozos de la función de Runge). Interpolamos la función de Runge, con la que ya trabajamos en la Sección 3.3.2. Consideramos $f: [-1, 1] \rightarrow \mathbb{R}$, $f(x) = \frac{1}{1+25x^2}$, usamos $n+1$ puntos equiespaciados en $[-1, 1]$,

$$x_i := -1 + \frac{2i}{n}, \quad i = 0, \dots, n,$$

y construimos la interpolante lineal a trozos de f por estos nodos. Esto es, en la Definición 3.4.1 tomamos $y_i = f(x_i)$ para $i = 0, \dots, n$. La Figura 3.6 nos muestra el resultado tomando $n+1 = 21$ puntos, y se contrasta notablemente con el panel derecho de la Figura 3.4: la interpolación lineal a trozos evita que la función interpolante oscile.

También podemos estimar el error de interpolación usando el Teorema 3.4.1. La derivada segunda de la función de Runge es

$$f''(x) = \frac{50(75x^2 - 1)}{(1 + 25x^2)^3},$$

que en valor absoluto alcanza su máximo en $x = 0$ y es $f''(0) = -50$. Por lo tanto, $\|f''\|_{L^\infty([-1, 1])} = 50$.

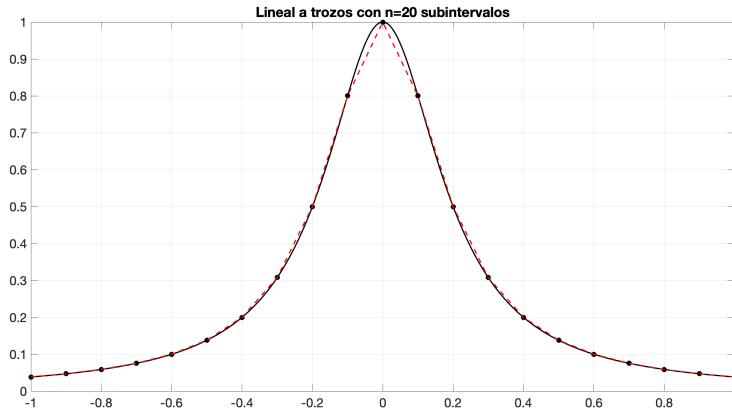


Figura 3.6: Interpolación lineal a trozos de la función de Runge.

Por otra parte, estamos tomando nodos equiespaciados, por lo que todos los subintervalos tienen la misma longitud: tenemos el intervalo $[-1, 1]$ partido en $n = 20$ subintervalos, por lo que tenemos

$$|x_{i+1} - x_i| = \frac{1}{10} \quad \forall i = 0, \dots, n-1.$$

Reemplazando en (3.19), obtenemos que, al realizar una interpolación lineal a trozos de la función de Runge con 21 nodos equiespaciados, el error en norma del supremo se puede acotar mediante

$$\|f - L\|_{L^\infty([-1,1])} \leq \frac{50}{8} \frac{1}{10^2} = \frac{1}{16}.$$

Más en general, si en lugar de 21 usáramos $n + 1$ nodos equiespaciados, entonces cada subintervalo tendría longitud

$$|x_{i+1} - x_i| = \frac{2}{n} \quad \forall i = 0, \dots, n-1.$$

y llegaríamos a la cota del error

$$\|f - L\|_{L^\infty([-1,1])} \leq \frac{50}{8} \left(\frac{2}{n}\right)^2 = \frac{25}{n^2}.$$

Notemos que $\|f - L\|_{L^\infty([-1,1])} \rightarrow 0$ cuando $n \rightarrow \infty$: a diferencia de lo que ocurría en la Figura 3.4, ahora tenemos que la sucesión de interpolantes lineales a trozos por n puntos equiespaciados converge a la función. \triangle

3.4.2. Interpolación cúbica a trozos

La interpolación lineal a trozos nos da una función que no es derivable, y quizás en ciertas aplicaciones queramos que el gráfico resultante sea más agradable a la vista. Tal como

ya analizamos (recordar la discusión alrededor de (3.15)), la siguiente opción natural es buscar interpolantes cúbicas a trozos.

A continuación discutimos varias alternativas posibles para construir tal interpolante, dependiendo del tipo de problema que tengamos y de qué propiedades deseemos que nuestra interpolante tenga. El marco general para las tres interpolantes que analizamos (de Hermite, splines, y “que preserva forma”) es el siguiente. Dado un conjunto de $n + 1$ ternas de números reales $\{(x_i, y_i, d_i)\}_{i=0,\dots,n}$, con $x_0 < \dots < x_n$, buscamos una función p que cumpla:

- para cada $i = 0, \dots, n - 1$, $p|_{[x_i, x_{i+1}]}$ sea un polinomio de grado 3,
- para cada $i = 0, \dots, n - 1$, se cumplan $p(x_i) = y_i$, $p'(x_i) = d_i$.

Es conveniente tratar este problema en cada subintervalo por separado. Para probar que existe un único polinomio cúbico en el intervalo $[x_i, x_{i+1}]$ que cumple

$$p(x_i) = y_i, \quad p(x_{i+1}) = y_{i+1}, \quad p'(x_i) = d_i, \quad p'(x_{i+1}) = d_{i+1}, \quad (3.20)$$

podemos razonar “a la Vandermonde”⁷, planteando a p en la base monomial, $p(x) = c_0 + c_1x + c_2x^2 + c_3x^3$, imponiendo estas ecuaciones y llegando a un sistema lineal 4×4 cuyas incógnitas son c_0, c_1, c_2, c_3 . Nos vamos a ahorrar el trabajo de ensamblar y resolver este sistema y simplemente vamos a dar su solución: para $x \in [x_i, x_{i+1}]$ ($i = 0, \dots, n - 1$),

$$\begin{aligned} p(x) = & \left(\frac{3h_i s^2 - 2s^3}{h_i^3} \right) y_{i+1} + \left(\frac{h_i^3 - 3h_i s^2 + 2s^3}{h_i^3} \right) y_i \\ & + \frac{s^2(s - h_i)}{h_i^2} d_{i+1} + \frac{s(s - h_i)^2}{h_i^2} d_i. \end{aligned} \quad (3.21)$$

Aquí, introdujimos la notación

$$h_i := x_{i+1} - x_i,$$

y usamos la variable local que definimos en la Sección 3.4.1,

$$s := x - x_i.$$

Ejercicio 3.3. Verificar que el polinomio dado por (3.21) cumple las condiciones (3.20).

La diferencia entre las tres alternativas que analizamos a continuación radica en cómo tomamos las pendientes d_0, \dots, d_n . Dependiendo del problema que nos interese, éstas podrían ser conocidas o no. En caso de que no lo sean, tenemos la libertad de elegirlas a nuestro criterio.

⁷Notar que, como el sistema resultante es 4×4 , no tenemos los inconvenientes que comentamos en las observaciones 3.2.2 y 3.2.3.

Interpolación de Hermite

La interpolación de Hermite parte de una base similar a la de la Sección 3.3: asumimos que nuestros datos vienen dados por muestrear una cierta función f . Específicamente, asumimos que existe $f: [x_0, x_n] \rightarrow \mathbb{R}$ suficientemente regular y tal que

$$y_i = f(x_i), \quad d_i = f'(x_i), \quad \forall i = 0, \dots, n.$$

Como las pendientes d_0, \dots, d_n son conocidas, simplemente debemos reemplazar los valores correspondientes en (3.21) y logramos construir una interpolante cúbica a trozos $p: [x_0, x_n] \rightarrow \mathbb{R}$, que cumple $p(x_i) = f(x_i)$, $p'(x_i) = f'(x_i)$ para todo $i = 0, \dots, n$.

Definición 3.4.2 (interpolante cúbica a trozos de Hermite). A la función $p: [x_0, x_n] \rightarrow \mathbb{R}$ construida como se describe arriba se le llama la **interpolante cúbica a trozos de Hermite de f por x_0, \dots, x_n** . \triangle

Observación 3.4.2. En general, la **interpolación de Hermite** refiere a interpolar una función f incorporando no solamente sus valores en los nodos (los $y_0 = f(x_0), \dots, y_n = f(x_n)$) sino que también los valores de las derivadas hasta un cierto orden de f en los nodos. Si queremos que las primeras k derivadas de nuestra interpolante de Hermite a trozos coincidan con las primeras k derivadas de f , entonces la interpolante debe ser un polinomio de grado $2k + 1$ en cada subintervalo. Por ejemplo, si queremos que nuestra interpolante tenga iguales derivada primera y derivada segunda en todos los nodos, obtendremos la llamada *interpolante quíntica de Hermite*, que es un polinomio de grado 5 a trozos. En este curso nos restringimos a la interpolación cúbica de Hermite, que es la más utilizada en la práctica. \triangle

De forma análoga a lo que hicimos en las secciones 3.3.1 y 3.4.1, podemos estimar el error de interpolación al usar interpolantes cúbicas a trozos de Hermite. Comenzamos con una fórmula de representación del error análoga a (3.11).

Teorema 3.4.2 (error de interpolación cúbica de Hermite). *Sea $f: [a, b] \rightarrow \mathbb{R}$ una función de clase C^4 , y consideremos su interpolante cúbica de Hermite p por los puntos a y b , esto es, p es el único polinomio de grado 3 que cumple $p(a) = f(a)$, $p(b) = f(b)$, $p'(a) = f'(a)$, $p'(b) = f'(b)$. Entonces, para todo $x \in [a, b]$ existe un $\gamma_x \in (a, b)$ tal que*

$$e(x) := f(x) - p(x) = \frac{(x-a)^2(x-b)^2}{24} f^{(4)}(\gamma_x). \quad (3.22)$$

Demostración. Sea $x \in [a, b]$. El resultado es trivial si $x = a$ o $x = b$, por lo que asumimos $x \in (a, b)$. Consideramos la función auxiliar $G: [a, b] \rightarrow \mathbb{R}$,

$$G(t) := e(t) - e(x) \frac{(t-a)^2(t-b)^2}{(x-a)^2(x-b)^2}.$$

Como $G(a) = G(b) = G(x) = 0$, por el Teorema de Rolle podemos asegurar que G' tiene al menos dos raíces, una en el intervalo (a, x) y otra en el intervalo (x, b) . Pero además, como

$e'(a) = f'(a) - p'(a) = 0$, es sencillo verificar que $G'(a) = 0$, y análogamente $G'(b) = 0$. Por lo tanto, deducimos que G' tiene al menos cuatro raíces distintas en el intervalo $[a, b]$.

Ahora argumentamos como en la demostración del Teorema 3.3.1: aplicando el Teorema de Rolle sucesivamente llegamos a que $G^{(4)}$ tiene al menos una raíz en el intervalo (a, b) , a la que llamamos γ_x , y tenemos

$$0 = G^{(4)}(\gamma_x) = e^{(4)}(\gamma_x) - e(x) \frac{4!}{(x-a)^2(x-b)^2}.$$

Como p es un polinomio de grado 3, $e^{(4)}(\gamma_x) = f^{(4)}(\gamma_x)$ y despejando $e(x)$ obtenemos (3.22). \square

Observación 3.4.3 (técnica de la demostración). En este punto, recomendamos comparar las demostraciones de los teoremas 3.3.1 y 3.4.2. En ambas demostraciones hacemos aparecer una función auxiliar, dependiente de una cierta variable t , que computa el error en t menos el error en x multiplicado por un cociente entre polinomios nodales. En ambas demostraciones el argumento se basa en aplicar el Teorema de Rolle sucesivamente, y para la interpolación de Hermite explotamos el hecho de que tenemos $e'(a) = e'(b) = 0$ elevando al cuadrado el cociente entre polinomios nodales, lo que nos permite “ganar” dos ceros adicionales para G' y por lo tanto llegar a un resultado más fuerte que si solamente hubiésemos hecho una interpolación por $(a, f(a)), (b, f(b))$. \triangle

Del Teorema 3.4.2 podemos deducir cotas del error de interpolación cúbica de Hermite, y aplicando estas cotas en cada subintervalo llegamos a estimaciones para el error de interpolación cúbica a trozos de Hermite.

Corolario 3.4.3 (estimación del error de interpolación cúbica a trozos de Hermite). *Sean $n + 1$ puntos $x_0 < x_1 < \dots < x_n$, f una función de clase C^4 en el intervalo $[x_0, x_n]$, y p la interpolante cúbica a trozos de Hermite por x_0, \dots, x_n . Entonces, si $x \in [x_i, x_{i+1}]$ para $i \in \{0, \dots, n-1\}$, se tiene*

$$|f(x) - p(x)| \leq \frac{\|f^{(4)}\|_{L^\infty([x_i, x_{i+1}])}}{24} (x - x_i)^2 (x_{i+1} - x)^2. \quad (3.23)$$

Además,

$$\|f - p\|_{L^\infty([x_0, x_n])} \leq \frac{\|f^{(4)}\|_{L^\infty([x_0, x_n])}}{384} \max_{j=0, \dots, n} (x_{j+1} - x_j)^4. \quad (3.24)$$

Demostración. La demostración es análoga a la del Teorema 3.4.1. Sea $x \in [x_i, x_{i+1}]$ para algún $i \in \{0, \dots, n-1\}$. Para probar (3.23), aplicamos (3.22) en el intervalo $[x_i, x_{i+1}]$, tomamos valor absoluto, y como $\gamma_x \in (x_i, x_{i+1})$ podemos acotar

$$|f^{(4)}(\gamma_x)| \leq \|f^{(4)}\|_{L^\infty([x_i, x_{i+1}])}.$$

Luego, para probar (3.24), basta con notar que

$$(x - x_i)^2 (x_{i+1} - x)^2 \leq \frac{(x_{i+1} - x_i)^4}{16} \quad \forall x \in [x_i, x_{i+1}],$$

y usar que $\|f^{(4)}\|_{L^\infty([x_i, x_{i+1}])} \leq \|f^{(4)}\|_{L^\infty([x_0, x_n])}$ y $(x_{i+1} - x_i)^4 \leq \max_{j=0, \dots, n} (x_{j+1} - x_j)^4$. \square

Ejemplo 3.4.2 (interpolación de Hermite de la función de Runge). Nuevamente interpolamos la función de Runge sobre $n + 1$ nodos equiespaciados en el intervalo $[-1, 1]$, $x_i = -1 + \frac{2i}{n}$. Como la derivada primera de la función de Runge es

$$f'(x) = \frac{-50x}{(1 + 25x^2)^2},$$

en la fórmula (3.21) y para cada $i = 0, \dots, n$, debemos imponer

$$y_i := f(x_i) = \frac{1}{1 + 25x_i^2}, \quad d_i := f'(x_i) = \frac{-50x_i}{(1 + 25x_i^2)^2}.$$

La Figura 3.7 muestra el resultado obtenido con $n + 1 = 12$ nodos. Podemos usar el

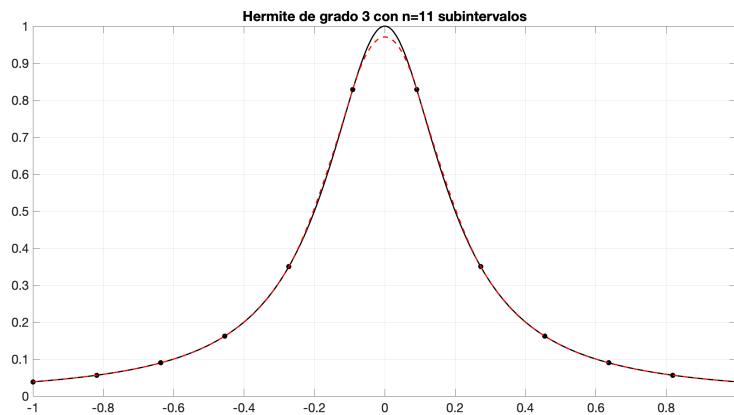


Figura 3.7: Interpolación de Hermite cúbica a trozos de la función de Runge.

Corolario 3.4.3 para estimar el error de interpolación. Al usar $n + 1$ nodos equiespaciados en el intervalo $[-1, 1]$, la longitud de cada intervalo es $x_{i+1} - x_i = \frac{2}{n}$. Con mucha paciencia (o un poco de Wolfram Alpha), se encuentra que

$$f^{(4)}(x) = \frac{15000(3125x^4 - 250x^2 + 1)}{(25x^2 + 1)^5},$$

y por lo tanto $\|f^{(4)}\|_{L^\infty([-1,1])} = f^{(4)}(0) = 15000$. Reemplazando en (3.24), obtenemos

$$\|f - p\|_{L^\infty([-1,1])} \leq \frac{15000}{384} \left(\frac{2}{n}\right)^4 = \frac{625}{n^4}.$$

Notamos que $\|f - p\|_{L^\infty([-1,1])} \rightarrow 0$ con $n \rightarrow \infty$ y, en particular para la interpolante con $n = 11$ de la Figura 3.7, hemos probado que

$$\|f - p\|_{L^\infty([-1,1])} \leq \frac{625}{11^4} \approx 4,3 \times 10^{-2}.$$

△

Splines cúbicas

Volvamos a nuestro punto de partida (3.20) y (3.21), pero ahora suponiendo que *no conocemos* los valores de d_0, \dots, d_n . Esto es, dados $n+1$ puntos en el plano $\{(x_i, y_i)\}_{i=0, \dots, n}$ con $x_0 < x_1 < \dots < x_n$, buscamos una función p cúbica a trozos, derivable en $[x_0, x_n]$, y tal que $p(x_i) = y_i$ para todo $i = 0, \dots, n$.

La idea al construir **splines cúbicas** es lograr que la función p sea lo más regular posible. En el caso de funciones cúbicas a trozos, se toman los $\{d_i\}_{i=0, \dots, n}$ de modo tal que p sea de clase C^2 , esto es, que tenga derivada segunda continua en el intervalo $[x_0, x_n]$. Derivando dos veces (3.21), obtenemos para cada $x \in [x_i, x_{i+1}]$, $i = 0, \dots, n - 1$,

$$p''(x) = \frac{(6h_i - 12s)\delta_i + (6s - 2h_i)d_{i+1} + (6s - 4h_i)d_i}{h_i^2}, \quad (3.25)$$

donde nuevamente usamos la notación

$$s = x - x_i, \quad h_i = x_{i+1} - x_i, \quad \delta_i = \frac{y_{i+1} - y_i}{h_i}.$$

Como $p|_{[x_i, x_{i+1}]}$ es un polinomio, para lograr que p sea C^2 basta con imponer que las derivadas segundas se peguen bien en los *nodos interiores* x_1, \dots, x_{n-1} . Usamos (3.25) y obtenemos, para $j = 1, \dots, n - 1$,

$$\begin{aligned} p''(x_j^+) &= \frac{6\delta_j - 2d_{j+1} - 4d_j}{h_j} && (i = j, s = 0 \text{ en (3.25)}), \\ p''(x_j^-) &= \frac{-6\delta_{j-1} + 4d_j + 2d_{j-1}}{h_{j-1}} && (i = j - 1, s = h_i \text{ en (3.25)}). \end{aligned}$$

Por lo tanto, si queremos que la interpolante p sea de clase C^2 , necesitamos que los coeficientes d_0, \dots, d_n satisfagan

$$\frac{6\delta_j - 2d_{j+1} - 4d_j}{h_j} = \frac{-6\delta_{j-1} + 4d_j + 2d_{j-1}}{h_{j-1}} \quad \forall j = 1, \dots, n - 1,$$

lo que podemos reescribir como

$$h_j d_{j-1} + 2(h_{j-1} + h_j) d_j + h_{j-1} d_{j+1} = 3h_j \delta_{j-1} + 3h_{j-1} \delta_j \quad \forall j = 1, \dots, n - 1. \quad (3.26)$$

Así, hemos obtenido un sistema (tridiagonal) de ecuaciones lineales que debemos resolver para determinar los valores de d_0, \dots, d_n . Recordemos que una vez que tenemos hallados estos valores, nos basta con utilizar la expresión (3.21) para definir la interpolante spline cúbica a trozos p .

Sin embargo, notemos que (3.26) pone requerimientos solamente sobre los nodos interiores, esto es, nos da $n - 1$ restricciones. En cambio, tenemos $n + 1$ incógnitas d_0, \dots, d_n . Aún tenemos algo de libertad para imponer dos condiciones adicionales. Aquí se abren algunas posibilidades⁸:

⁸Referimos a [Sau12, Sección 3.4.2] para más detalles y otras opciones. En ese libro, a la que nosotros llamamos spline completa se la denomina spline sujetada (*clamped*).

- Prescribir los valores de $p'(x_0)$ y $p'(x_n)$, esto es, tomar d_0 y d_n como datos, por lo que ahora (3.26) corresponde a un sistema de $n - 1$ ecuaciones con $n - 1$ incógnitas d_1, \dots, d_{n-1} . Esta elección se llama **spline completa**.
- Imponer que $p''(x_0) = 0$ y que $p''(x_n) = 0$. Esto nos agrega dos ecuaciones adicionales más a (3.26) y da lugar a un sistema $(n + 1) \times (n + 1)$. Hacer esto se conoce como tomar una **spline natural**.
- Buscar que p tenga un poco más de regularidad cerca de los bordes, esto es, que p sea de clase C^3 en los intervalos $[x_0, x_2]$ y $[x_{n-2}, x_n]$. Como p es una función cúbica a trozos, hacer esto es equivalente a usar la misma función cúbica en $[x_0, x_1]$ que en $[x_1, x_2]$ y respectivamente, en $[x_{n-2}, x_{n-1}]$ que en $[x_{n-1}, x_n]$. Esto es, estamos “eliminando” los nodos x_1 y x_{n-1} , y por esta razón a este tipo de interpolante se la conoce como **spline not-a-knot**⁹.

Ejemplo 3.4.3 (spline *not-a-knot*). Para fijar ideas, vamos a ejemplificar con la última elección, la de spline *not-a-knot*, analizando lo que ocurre en $[x_0, x_2]$. Tomando derivada tercera de p en (3.21) en los intervalos $[x_0, x_1]$ y $[x_1, x_2]$, obtenemos dos constantes. Como queremos que la interpolante tenga derivada tercera continua en x_1 , estas dos constantes deben coincidir. Así, obtenemos la identidad

$$\frac{-12\delta_0 + 6d_1 + 6d_0}{h_0^2} = \frac{-12\delta_1 + 6d_2 + 6d_1}{h_1^2}.$$

De la misma forma, imponer que p tenga derivada tercera continua en $[x_{n-2}, x_n]$ da lugar a la ecuación

$$\frac{-12\delta_{n-2} + 6d_{n-1} + 6d_{n-2}}{h_{n-2}^2} = \frac{-12\delta_{n-1} + 6d_n + 6d_{n-1}}{h_{n-1}^2}.$$

Por lo tanto, al sistema (3.26) le debemos agregar estas dos ecuaciones y obtenemos un sistema $(n + 1) \times (n + 1)$. \triangle

Observación 3.4.4 (elección de spline). En la práctica, la elección de qué tipo de spline utilizar depende de la aplicación o de qué propiedades nos parezcan deseables que la interpolante tenga. ¡No hay una opción “mejor” que otra! \triangle

Interpolación “que preserva forma”

Volvemos a nuestro punto de partida en interpolación cúbica a trozos: dados $n + 1$ puntos en el plano $\{(x_i, y_i)\}_{i=0, \dots, n}$ con $x_0 < x_1 < \dots < x_n$, buscamos una función p cúbica a trozos, derivable en $[x_0, x_n]$, y tal que $p(x_i) = y_i$ para todo $i = 0, \dots, n$. Ya sabemos, por (3.20) y (3.21), que la función p queda determinada si fijamos los valores de d_0, \dots, d_n .

⁹Que en inglés significa “no es un nodo”.

En Octave y Matlab hay una función que se llama `pchip` y que genera una interpolante “visualmente agradable”¹⁰ de los datos. La idea es que la interpolante respete los máximos y mínimos locales, esto es, que no genere oscilaciones más allá de los valores y_0, \dots, y_n .

A continuación describimos cómo opera la función `pchip` para definir los valores de d_1, \dots, d_n y, en consecuencia, construir una interpolante cúbica a trozos. Referimos a [Mol04, Sección 3.4] para más detalles. Remarcamos, una vez más, que las elecciones que tomamos aquí son arbitrarias y responden a criterios cosméticos.

- Para $i = 0, \dots, n - 1$, computamos las longitudes de los subintervalos $h_i := x_{i+1} - x_i$ y las diferencias divididas $\delta_i := \frac{y_{i+1} - y_i}{h_i}$.
- Supongamos ahora que estamos en un nodo interior, $i \in \{1, \dots, n - 1\}$.
 - Si $\delta_{i-1} \delta_i \leq 0$, estamos ante un máximo o mínimo local respecto a los valores de los datos, por lo que imponemos que $d_i = 0$.
 - Si $\delta_{i-1} \delta_i > 0$, entonces se considera una **media armónica** pesada entre las diferencias divididas, con los pesos dependiendo de la longitud de los subintervalos. Esto es, se define d_i como el valor que cumple

$$\frac{3(h_{i-1} + h_i)}{d_i} = \frac{h_{i-1} + 2h_i}{\delta_{i-1}} + \frac{2h_{i-1} + h_i}{\delta_i}.$$

- Para determinar d_0 y d_n , se utiliza lo que se llama una *fórmula no centrada*: por ejemplo, para definir d_0 se considera

$$d := \frac{(2y_0 + y_1)\delta_0 - y_0\delta_1}{y_0 + y_1}.$$

Luego,

- si $\text{sgn}(d) \neq \text{sgn}(\delta_0)$, se define $d_0 := 0$,
- en caso contrario, si $\text{sgn}(\delta_0) \neq \text{sgn}(\delta_1)$ y $|d| > 3|\delta_0|$, se define $d_0 := 3\delta_0$,
- en caso contrario, se define $d_0 := d$.

El procedimiento para definir d_n es análogo.

Ejemplo 3.4.4 (spline vs. interpolante “que preserva forma”). Consideremos los datos de la siguiente tabla:

x	0	1	3	4	5	7
y	1	1	-2	2	3	0

¹⁰Evidentemente, esta es una noción subjetiva.

Construimos interpolantes de estos datos usando una spline *not-a-knot* y la interpolante “que preserva forma” dada por `pchip`. Nuestros resultados se muestran en la Figura 3.8. La spline muestra una mayor suavidad (es dos veces derivable), pero notamos que la curva azul puede oscilar más allá de los valores de los datos. En cambio, la interpolante “que preserva forma” se ve menos suave pero no genera oscilaciones: el máximo y mínimo de la curva roja se alcanzan en nodos de interpolación. Además, en el intervalo $[0, 1]$ la curva roja es constante e igual a 1. \triangle

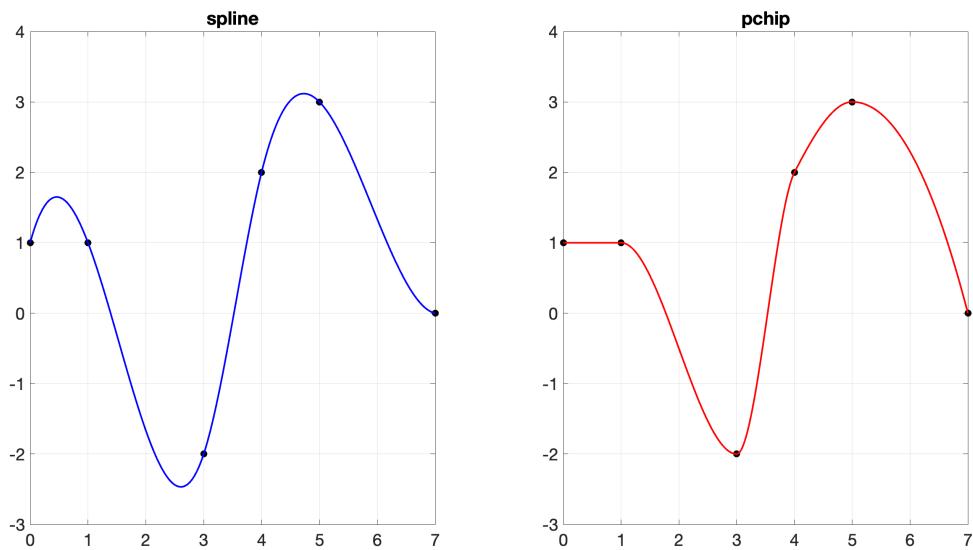


Figura 3.8: Interpolaciones cúbicas a trozos.

Capítulo 4

Ecuaciones no lineales

4.1. Introducción

En este capítulo analizamos métodos para aproximar raíces de funciones. Para fijar ideas, comenzamos nuestra discusión con funciones $f : \mathbb{R} \rightarrow \mathbb{R}$, y buscamos $x^* \in \mathbb{R}$ tal que $f(x^*) = 0$. Si f es una función lineal, o más en general si contamos con una forma de computar exactamente la preimagen de 0, entonces el problema es sencillo. Esto se corresponde con los métodos directos para resolver sistemas de ecuaciones lineales: por ejemplo, conocemos una fórmula cerrada para escribir las raíces de polinomios cuadráticos y podríamos utilizarla para –al menos en lápiz y papel– determinarlas exactamente. Los casos en que tenemos una fórmula cerrada para hallar raíces son muy particulares, y en general buscamos una aproximación de x^* mediante métodos indirectos.

Así, tendremos generada una sucesión $\{x^k\}$ que aspiramos que cumpla $x^k \rightarrow x^*$ con $f(x^*) = 0$. Como condición de parada, podemos tomar una cierta tolerancia $\varepsilon > 0$ y computar elementos de esta sucesión hasta que se satisfaga que los **errores**

$$e^k := x^k - x^*, \quad (4.1)$$

cumplen $|e^k| < \varepsilon$. Este enfoque no parece muy útil en la práctica, ya que para computar e^k debemos conocer x^* . Una alternativa computable es utilizar los **residuos**

$$r^k := f(x^k)$$

y parar cuando $|r^k| < \varepsilon$. Como ya a esta altura podemos esperar, el hecho de que un residuo “pequeño” en valor absoluto dé lugar a un error “pequeño” en valor absoluto depende de un cierto número de condición. Notemos, sin embargo, que tal número de condición no es igual al de la Definición 1.3.3, ya que nuestro objetivo aquí no es evaluar f (es decir, hallar sus imágenes), sino *hallar preimágenes del número 0*. Visualmente, la Figura 4.1 nos muestra que aquí, si $f'(x^*) \approx 0$, entonces el problema de hallar raíces de f va a estar mal condicionado. Referimos a [Sau12, Sección 1.3.3] para más detalles sobre el condicionamiento del problema de hallar raíces.

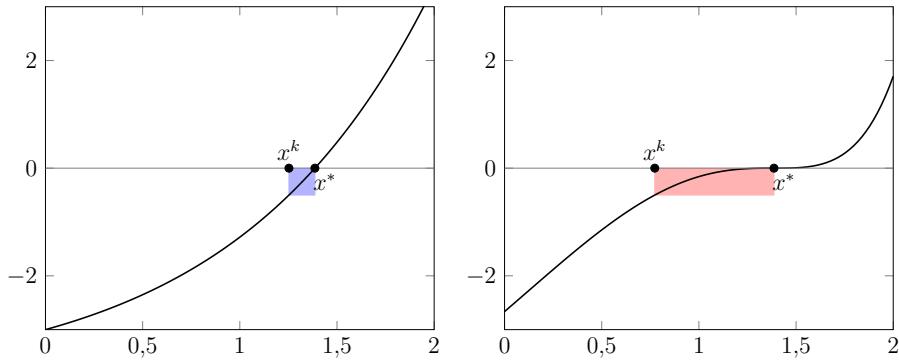


Figura 4.1: Izquierda: si $|f'(x^*)|$ está lejos de ser 0, la aproximación de x^* es un problema bien condicionado, porque si $f(x^k) \approx 0$, entonces $x^k \approx x^*$. Derecha: si $f'(x^*) \approx 0$, la aproximación de x^* es un problema mal condicionado, porque puede pasar que $f(x^k) \approx 0$ con x^k lejos de x^* .

Una pregunta fundamental para nosotros es cómo cuantificar la capacidad de un método dado para aproximar a x^* . En la siguiente definición asumimos que estamos trabajando con funciones vectoriales $\mathbf{f}: \mathbb{R}^n \rightarrow \mathbb{R}^n$, y que $\|\cdot\|$ es una norma dada en \mathbb{R}^n . En el caso escalar ($f: \mathbb{R} \rightarrow \mathbb{R}$), basta con reemplazar $\|\cdot\|$ por $|\cdot|$.

Definición 4.1.1 (orden y velocidad de un método). Sea $\mathbf{f}: \mathbb{R}^n \rightarrow \mathbb{R}^n$. Diremos que un método convergente para hallar raíces de \mathbf{f} tiene **orden** $p \geq 1$ si p es el mayor número tal que la sucesión de errores $\{\mathbf{e}^k\}$ cumple

$$\|\mathbf{e}^{k+1}\| = O(\|\mathbf{e}^k\|^p) \text{ con } k \rightarrow \infty.$$

En caso de ser de orden p , diremos que tiene **velocidad** $\beta > 0$ si

$$\lim_{k \rightarrow \infty} \frac{\|\mathbf{e}^{k+1}\|}{\|\mathbf{e}^k\|^p} = \beta.$$

△

Observación 4.1.1 (velocidad de convergencia). No todos los métodos convergentes tienen por qué tener un orden o una velocidad de convergencia. En caso de que un método lo tenga, por lo general conocer el orden nos da mucho mejor idea sobre cómo approxima a la solución que conocer su velocidad de convergencia. Cuando $p = 1$, se suele decir que la convergencia es *lineal*. En ese caso, para que sea convergente, es necesario que su velocidad de convergencia sea $\beta < 1$. En caso de que $p > 1$, no es necesario que sea $\beta < 1$ para que el método sea convergente. △

4.2. Método de biseción

A diferencia de las ecuaciones lineales, cuyo conjunto de soluciones se puede caracterizar de forma simple, las ecuaciones no lineales en general no tienen por qué tener solución única.

En general, una ecuación no lineal puede tener una cantidad arbitraria de soluciones, y el conjunto de soluciones podría ser arbitrariamente difícil de caracterizar matemáticamente. Asegurar que una ecuación no lineal tiene alguna solución puede ser una tarea difícil, pero el Teorema de Bolzano A.2.1 nos da una situación concreta en la que sí podemos lograrlo: toda función real continua que tenga un cambio de signo en un intervalo $[a, b]$ tiene al menos una raíz en dicho intervalo. El Teorema de Bolzano no nos asegura que esa raíz sea única –en general, no lo es–, pero sí podemos usarlo en forma iterada de modo de *encerrar a alguna raíz*. Esto da lugar al llamado **método de bisección**.

El método de bisección no es más que una forma sistemática de ensayo y error. Supongamos que $I \subset \mathbb{R}$ es un intervalo, y que tenemos una función $f: I \rightarrow \mathbb{R}$ continua, con $f(a) < 0, f(b) > 0$ para ciertos $a, b \in I$ ¹. Podemos considerar el punto medio entre a y b , $m := \frac{a+b}{2}$.

- Si $f(m) = 0$, conseguimos nuestro objetivo de haber hallado una raíz de f y terminamos nuestro trabajo.
- Si $f(m) > 0$, entonces como f es continua en $[a, m]$ y $f(a)f(m) < 0$, el Teorema de Bolzano nos asegura que f tiene (al menos) una raíz en $[a, m]$. Redefinimos $b := m$ y repetimos el proceso de bisección.
- Si $f(m) < 0$, entonces de la misma manera tenemos garantizado que f tiene (al menos) una raíz en $[m, b]$. Redefinimos $a := m$ y repetimos el proceso de bisección.

En forma de pseudo-código, este método se describe en el Algoritmo 4.1.

Algoritmo 4.1: Pseudo-código: bisección.

Datos: $a < b \in \mathbb{R}$, f continua en $[a, b]$ con $f(a)f(b) < 0$, $\text{tol} > 0$

Resultado: un nuevo intervalo $[a, b]$, que contiene una raíz en $[a, b]$ y de longitud

$$b - a < \text{tol}$$

mientras $b - a \geq \text{tol}$ **hacer**

```

     $m \leftarrow \frac{a+b}{2};$ 
    si  $f(m) = 0$  entonces
         $|$  break
    fin
    si no, si  $f(a)f(m) < 0$  entonces
         $|$   $b \leftarrow m$ 
    fin
    en otro caso
         $|$   $a \leftarrow m$ 
    fin
fin

```

fin

Observación 4.2.1 (orden y velocidad). Formalmente, el método de bisección no genera una sucesión $\{x^k\}$ que converge a una raíz x^* , sino una *sucesión de intervalos que van*

¹Naturalmente, el razonamiento es análogo si $f(a) > 0, f(b) < 0$.

atrapando a x^* . Vamos a llamar a este tipo de métodos como **de encierro**². Para este método no tiene sentido definir el error como en (4.1), pero la longitud del intervalo $[a, b]$ nos da una cota superior para el error. Si usamos como estimación del error la longitud del intervalo de búsqueda $[a, b]$ y observamos que en cada paso esta longitud se reduce a la mitad, deducimos que **el método de bisección es de primer orden, y tiene velocidad de convergencia igual a 1/2**. En lenguaje más práctico, como $\log_2(10) \approx 3,3$, tenemos que cada aproximadamente 3 pasos nuestro intervalo de búsqueda gana un dígito de precisión para aproximar x^* . \triangle

Observación 4.2.2 (lento, pero seguro). El método de bisección tiene dos características particulares. En primer lugar, tiene la desventaja de que usa información muy cruda de f : solamente considera su signo y no sus valores. Incluso si f fuese una función lineal el método no convergería en una sola iteración a menos que la raíz de f fuese el punto medio del intervalo inicial de búsqueda. Esta limitación se contrasta con el hecho de que el método de bisección nos *garantiza* la convergencia a una raíz. \triangle

Observación 4.2.3 (converge, pero, ¿a dónde?). Notemos que si f tiene más de una raíz en el intervalo inicial de búsqueda $[a, b]$, entonces puede no ser sencillo caracterizar hacia qué raíz converge el método de bisección. \triangle

4.3. Método de la regla falsa

Tal como comentamos en la Observación 4.2.2, el método de bisección tiene la desventaja de usar información muy cruda sobre f . La idea en el **método de la regla falsa** (también llamado **método de regula falsi**) es incorporar información sobre los valores de f y mantener la garantía de convergencia para funciones continuas con un cambio de signo.

El punto de partida de este método es el mismo que en el de la bisección: tenemos dada una función f continua y que tiene un cambio de signo en el intervalo $[a, b]$. La diferencia entre el método de la regla falsa y el método de bisección está en cómo definimos el nuevo intervalo de búsqueda: ahora, consideramos la interpolante lineal por $(a, f(a))$ y $(b, f(b))$ y tomamos m como el punto en el que esa función lineal se anula:

$$m := a - \frac{f(a)(b-a)}{f(b)-f(a)} = \frac{a f(b) - b f(a)}{f(b)-f(a)}. \quad (4.2)$$

Luego, se procede como en el método de bisección, y se determina el nuevo intervalo de búsqueda como $[a, m]$ o $[m, b]$ según cuál de los dos presente el cambio de signo al evaluar f en sus extremos. Puede probarse que el método de la regla falsa converge a una raíz de f en $[a, b]$ para funciones continuas con cambio de signo³, y que generalmente lo hace de manera más eficiente que el método de bisección al incorporar información sobre los valores de la función para el cálculo de m . Ilustramos la diferencia entre los métodos de bisección y de la regla falsa en la Figura 4.2.

²Lo más habitual es utilizar la palabra del inglés *bracketing*.

³La demostración puede encontrarse, por ejemplo, en [Ngu21].

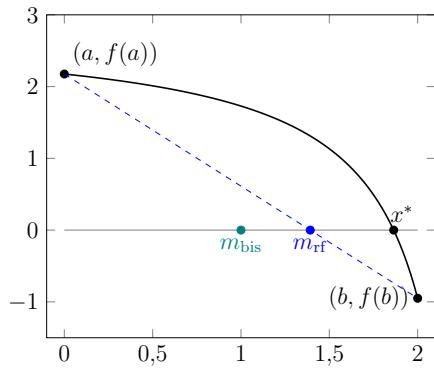


Figura 4.2: Comparación entre cómo se toma el nuevo punto m para los métodos de bisección (m_{bis}) y de la regla falsa (m_{rf}). A la recta marcada en azul se le llama la *secante* al gráfico de f por $(a, f(a)), (b, f(b))$.

Parece tentador pensar que el método de la regla falsa es de mayor orden que el de bisección, al estar incorporando información sobre cuánto vale f y no solamente su signo. Sin embargo, a diferencia de este último, la velocidad de convergencia del método de regla falsa no siempre es lineal y puede verse afectada por la forma de la función cerca de la raíz. En particular, si la función es estrictamente cóncava o convexa en el intervalo, el método mantendrá una de las aproximaciones fija mientras que la otra se acercará a la raíz buscada, lo que puede ralentizar la convergencia. Esta diferencia puede verse en el siguiente ejemplo.

Ejemplo 4.3.1 (convergencia de regla falsa). Buscamos computar $\log 3 \approx 1,0986$ a partir de hallar la raíz de $f(x) = 3e^{-x} - 1$. Inicializamos el método de la regla falsa con

$$\begin{aligned} a &= 0 \Rightarrow f(a) = 2 > 0, \\ b &= 2 \Rightarrow f(b) = 3e^{-2} - 1 \approx -0,5940 < 0, \end{aligned}$$

y computamos algunos iterados en la siguiente tabla.

n	a	b	m (ver (4.2))	$f(m)$	$e^k := m - \log 3$	$\frac{e^k}{e^{k-1}}$
1	0	2	1,5420	-0,3582	0,4434	—
2	0	1,5420	1,3078	-0,1888	0,2092	0,47
3	0	1,3078	1,1950	-0,0919	0,0964	0,46
4	0	1,1950	1,1425	-0,0430	0,0439	0,46
5	0	1,1425	1,1185	-0,0197	0,0199	0,44
6	0	1,1185	1,1076	-0,0089	0,0090	0,45

Hay dos hechos que resultan llamativos. En primer lugar, el punto a permanece fijo en la iteración, esto es, en todos los pasos tomamos $a = 0$, pues el punto m computado verifica $f(m) < 0$ y por lo tanto al iterado siguiente se toma $b \leftarrow m$. Como a está a una distancia

“grande” de la raíz $\log 3$, esto implica que si pusiéramos un criterio de parada como en el Algoritmo 4.1 con tol suficientemente pequeña la iteración no pararía nunca⁴. En este sentido, la longitud del intervalo de búsqueda no parece ser un buen indicador de qué tan cerca está el método de converger. Podría parecer más razonable tomar como indicador de error la distancia entre m y $\log 3$, como tomamos en la penúltima columna de la tabla; aquí estamos haciendo la “trampa” de que damos por conocido el valor de x^* que estamos aproximando. Allí surge el segundo hecho llamativo: los errores computados de esta forma decrecen linealmente, y la velocidad de convergencia parece ser apenas menor que $1/2$. En este ejemplo, no parece claro que el método de la regla falsa se desempeñe mejor que el de bisección. \triangle

Ejercicio 4.1. El objetivo de este ejercicio es generalizar el Ejemplo 4.3.1. Consideraremos una función $f: \mathbb{R} \rightarrow \mathbb{R}$ de clase C^2 y tal que $f'' > 0$.

1. Demostrar que f es *estrictamente convexa*, esto es, que para todos $a, b \in \mathbb{R}$, $\lambda \in (0, 1)$, se cumple

$$f((1 - \lambda)a + \lambda b) < (1 - \lambda)f(a) + \lambda f(b).$$

[Sugerencia: observar que f' es creciente y usar el Teorema de Lagrange A.2.3 en los intervalos $(a, (1 - \lambda)a + \lambda b)$ y $((1 - \lambda)a + \lambda b, b)$.]

2. Supongamos que, como en el Ejemplo 4.3.1, existen puntos $a < b$ tales que $f(a) > 0$, $f(b) < 0$ y se inicializa el método de la regla falsa con estos puntos. Demostrar que en todas las iteraciones se tiene $f(m) < 0$, y por lo tanto el punto a permanece fijo a lo largo de la iteración.
3. Análogamente, probar que si existen puntos $a < b$ tales que $f(a) < 0$, $f(b) > 0$ y se inicializa el método de la regla falsa con estos puntos, entonces en todas las iteraciones se tiene $f(m) < 0$ y por lo tanto b permanece fijo en la iteración.

4.4. Método de la secante

El método de la regla falsa corre el riesgo que uno de los extremos en el intervalo de búsqueda quede fijo a lo largo de la iteración, lo que hace que sea un poco rígido. Una alternativa simple es forzar al método a actualizar alternadamente los extremos del intervalo de búsqueda. Sin embargo, esto implica que perdamos el control sobre que haya un cambio de signo en el intervalo de búsqueda, por lo que la ganancia en flexibilidad del método tiene como contracara la pérdida de certeza de que sea convergente para funciones continuas con un cambio de signo.

Sea $f: \mathbb{R} \rightarrow \mathbb{R}$. El **método de la secante** genera una sucesión $\{x^k\}$ a partir de dos iterados iniciales x^0, x^1 , y calculando la intersección de la recta secante al gráfico de f por $(x^0, f(x^0)), (x^1, f(x^1))$ con el eje de las x . A diferencia del método de la regla falsa,

⁴En este caso, bastaría que fuera $\text{tol} < \log 3 \approx 1,0986$.

este método no revisa el signo de $f(x^{k+1})$ para definir a cuál de los puntos x^{k-1} o x^k descartar. Como este no es un método de encierro, la notación que estamos usando es un poco distinta; sin embargo, dados puntos $a = x^k$, $b = x^{k-1}$, para computar el iterado siguiente simplemente necesitamos usar (4.2):

$$x^{k+1} := x^k - \frac{f(x^k)(x^{k-1} - x^k)}{f(x^{k-1}) - f(x^k)} = \frac{x^k f(x^{k-1}) - x^{k-1} f(x^k)}{f(x^{k-1}) - f(x^k)}. \quad (4.3)$$

Para comparación futura, notamos que también podemos escribir esta definición como

$$x^{k+1} := x^k - \frac{f(x^k)}{s^k}, \quad \text{donde } s^k := \frac{f(x^k) - f(x^{k-1})}{x^k - x^{k-1}}. \quad (4.4)$$

Observación 4.4.1 (no certeza de convergencia). Es muy simple observar que el método de la secante no tiene por qué ser convergente, y ni siquiera la iteración tiene por qué estar bien definida. En efecto, si $f(x^{k-1}) = f(x^k)$, entonces la recta secante al gráfico de f por $(x^{k-1}, f(x^{k-1}))$, $(x^k, f(x^k))$ es paralela al eje de las x y por lo tanto la intersección entre ambas es vacía.

Un ejemplo sencillo de no convergencia del método de la secante es tomar la función $f(x) = |x|$, con $x^0 = 1$, $x^1 = -2$. Usando la definición (4.3) y un argumento de inducción, se deduce que $x^k = (-1)^k 2^k = (-2)^k$ para todo $n \in \mathbb{N}$ y por lo tanto el método es divergente. \triangle

En general no podemos garantizar la convergencia del método de la secante. Sin embargo, sí podemos hacerlo en caso de que nuestros iterados estén lo *suficientemente cerca* de la raíz que queremos aproximar y, bajo condiciones adecuadas en la función f , esta convergencia es superlineal.

Procedemos en tres pasos: primero, demostramos que el método es convergente; luego, deducimos una ecuación para el error; finalmente, en caso de que la convergencia sea con orden, deducimos cuál debe ser.

Lema 4.4.1 (Convergencia del método de la secante). *Sea $f: I \rightarrow \mathbb{R}$ una función de clase C^2 , y x^* un punto interior a I y tal que $f(x^*) = 0$, $f'(x^*) \neq 0$. Entonces, existe un $\delta > 0$ tal que si $|x^0 - x^*| < \delta$, $|x^1 - x^*| < \delta$, el método de la secante (4.4) converge a x^* .*

Demostración. Sea $\alpha := f'(x^*)$. Como $\alpha \neq 0$, por simplicidad en este paso asumimos $\alpha > 0$. Al ser f' continua, existe un $\delta > 0$ tal que

$$0 < \frac{3\alpha}{4} < f'(x) < \frac{5\alpha}{4} \quad \forall x \in (x^* - \delta, x^* + \delta) =: I_\delta. \quad (4.5)$$

Dados x^{k-1} , x^k , aplicamos el Teorema de Lagrange A.2.3 dos veces: por un lado, tenemos que $s^k = f'(c_k)$ para algún c_k entre x^{k-1} y x^k . Por otro lado, existe un \tilde{c}_k entre x^* y x^k tal que

$$f'(\tilde{c}_k) = \frac{f(x^k) - f(x^*)}{x^k - x^*} = \frac{f(x^k)}{x^k - x^*} \Rightarrow f(x^k) = f'(\tilde{c}_k)(x^k - x^*).$$

Usando (4.4), podemos escribir entonces

$$x^{k+1} - x^* = x^k - x^* - \frac{f(x^k)}{s^k} = x^k - x^* - \frac{f'(\tilde{c}_k)(x^k - x^*)}{f'(c_k)}.$$

Como $x^{k-1}, x^k, x^* \in I_\delta$, también tenemos que $c_k, \tilde{c}_k \in I_\delta$ y podemos usar (4.5) para acotar

$$|e^{k+1}| = |e^k| \left| 1 - \frac{f'(\tilde{c}_k)}{f'(c_k)} \right| \leq |e^k| \left| 1 - \frac{5\alpha/4}{3\alpha/4} \right| = \frac{2|e^k|}{3},$$

donde $e^k := x^k - x^*$ es el error en el paso n -ésimo. Deducimos entonces que si $x^0, x^1 \in I_\delta$ entonces el método es convergente, y lo hace al menos linealmente y con velocidad $\frac{2}{3}$. \square

Proposición 4.4.2 (Cota para el error del método de la secante). *Sea $f: I \rightarrow \mathbb{R}$ una función de clase C^2 , y x^* un punto interior a I . Entonces, el método de la secante obedece*

$$e^{k+1} = \frac{f''(\gamma_{k+1})}{2s^k} e^{k-1} e^k, \quad (4.6)$$

para algún $\gamma_{k+1} \in I$, y donde s^k está dado por (4.4).

Demostración. Comenzamos observando que, dados x^{k-1} y x^k , el siguiente iterado x^{k+1} se computa intersectando la interpolante lineal por $(x^{k-1}, f(x^{k-1}))$, $(x^k, f(x^k))$ con el eje de las x . De acuerdo a lo que discutimos en la Sección 3.2 (o a (3.16) específicamente para una interpolación lineal), esta interpolante está dada por

$$L(x) := f(x^k) + s^k(x - x^k).$$

En particular, tenemos que $L(x^{k+1}) = 0$. Además, como $f(x^*) = 0$, podemos escribir

$$L(x^*) - f(x^*) = f(x^k) + s^k(x^* - x^k),$$

y restando $L(x^{k+1}) = 0$ en el lado derecho de esta última igualdad llegamos a

$$L(x^*) - f(x^*) = f(x^k) + s^k(x^* - x^k) - [f(x^k) + s^k(x^{k+1} - x^k)] = -s^k e^{k+1}. \quad (4.7)$$

Por otra parte, como L es una interpolante lineal para f en I , podemos utilizar el Teorema 3.3.1 (o la fórmula (3.17) específicamente para una interpolación lineal) para escribir

$$L(x^*) - f(x^*) = -\frac{f''(\gamma_{x^*})}{2}(x^* - x^{k-1})(x^* - x^k) = -\frac{f''(\gamma_{k+1})}{2} e^{k-1} e^k, \quad (4.8)$$

para algún γ_{k+1} . Combinando (4.7) y (4.8), obtenemos (4.6). \square

En lo que sigue, llamamos Φ al **número áureo**

$$\Phi := \frac{1 + \sqrt{5}}{2} \approx 1,618,$$

que es la raíz positiva del polinomio $p(x) := x^2 - x - 1$.

Lema 4.4.3 (Orden de convergencia del método de la secante). *Sea $f: I \rightarrow \mathbb{R}$ una función de clase C^2 , y x^* un punto interior a I y tal que $f(x^*) = 0$, $f'(x^*) \neq 0$. Entonces, cuando el método de la secante converge con orden, éste es al menos $\Phi := \frac{1+\sqrt{5}}{2}$.*

Demostración. A partir de la proposición anterior, comenzamos controlando los términos que no involucran el error del lado derecho de (4.6). Por un lado, tenemos $|f''(\gamma_{k+1})| \leq \|f''\|_{L^\infty(I)}$. Por otro lado, como $f'(x^*) \neq 0$, usando el Teorema de Lagrange A.2.3 y el hecho de que f' es continua, se deduce que existe un $\epsilon > 0$ tal que $|s^k| > \epsilon$ si x^{k-1}, x^k están lo suficientemente cerca de x^* .

Tomando valor absoluto en (4.6) y definiendo $M := \frac{\|f''\|_{L^\infty(I)}}{2\epsilon}$, tenemos

$$|e^{k+1}| \leq M|e^k e^{k-1}|.$$

Ahora, como sabemos que el método converge con orden, podemos asumir que para k suficientemente grande se cumplen $|e^{k+1}| \approx C|e^k|^p$ y $|e^k| \approx C|e^{k-1}|^p$, donde p es el orden de convergencia y C es la velocidad de convergencia del método. Reemplazando arriba, tenemos

$$C|e^k|^p \lesssim MC|e^{k-1}|^{p+1} \Rightarrow |e^k| \lesssim M^{\frac{1}{p}}|e^{k-1}|^{\frac{p+1}{p}}.$$

Como tenemos $|e^k| \approx C|e^{k-1}|^p$ y $|e^k| \lesssim M^{\frac{1}{p}}|e^{k-1}|^{\frac{p+1}{p}}$, deducimos que el orden p debe cumplir

$$p = \frac{p+1}{p}, \quad \text{o equivalentemente} \quad p^2 - p - 1 = 0.$$

Concluimos que el orden de convergencia es $p = \Phi = \frac{1+\sqrt{5}}{2}$. \square

Combinando los Lemas 4.4.1 y 4.4.3, obtenemos el siguiente teorema.

Teorema 4.4.4 (convergencia y orden del método de la secante). *Sea $f: I \rightarrow \mathbb{R}$ una función de clase C^2 , y x^* un punto interior a I y tal que $f(x^*) = 0$, $f'(x^*) \neq 0$. Entonces, existe un $\delta > 0$ tal que si $|x^0 - x^*| < \delta$, $|x^1 - x^*| < \delta$, el método de la secante (4.4) converge a x^* , y, si la convergencia es con orden, el orden es (al menos) $\Phi = \frac{1+\sqrt{5}}{2}$.*

Observación 4.4.2 (cuenca de convergencia). El Teorema 4.4.4 nos dice que, bajo condiciones adecuadas, si el método de la secante es inicializado lo suficientemente cerca de la raíz entonces es convergente. Notemos que a partir de la demostración se deduce una forma para estimar un valor de $\delta > 0$ que garantice la convergencia: basta con tomar δ de modo que la derivada f' varíe lo suficientemente poco (en el sentido de (4.5)) en un entorno de centro x^* y radio δ . De todos modos, estimar el valor de δ es un tema delicado en general, ya que si no conocemos x^* , parece difícil que podamos conocer $f'(x^*)$ con exactitud. \triangle

Observación 4.4.3 (velocidad de convergencia). La Proposición 4.4.2, junto con la demostración del Lema 4.4.3, permiten incluso estimar la velocidad de convergencia del método.

En efecto, si ahora ya asumimos que el método converge con orden Φ y volvemos a (4.6), podemos afinar nuestra cota para M y aproximar

$$f''(\gamma_{k+1}) \approx f''(x^*), \quad s^k \approx f'(x^*) \quad \Rightarrow M \approx \left| \frac{f''(x^*)}{2f'(x^*)} \right|.$$

Como la velocidad de convergencia que estimamos en la demostración del Lema 4.4.3 era $C = M^{\frac{1}{\Phi}}$, podemos estimar la velocidad del método como

$$C \approx \left| \frac{f''(x^*)}{2f'(x^*)} \right|^{\frac{1}{\Phi}}.$$

△

Ilustramos el Teorema 4.4.4 con el siguiente ejemplo.

Ejemplo 4.4.1 (convergencia de secante). Repetimos la consigna del Ejemplo 4.3.1, pero ahora usamos el método de la secante inicializado en los mismos puntos $x^0 = 0$, $x^1 = 2$. Computamos algunos iterados en la siguiente tabla.

k	x^{k-1}	x^k	x^{k+1} (ver (4.4))	$e^{k+1} := x^{k+1} - \log 3$	$\frac{ e^{k+1} }{ e^k ^\Phi}$
1	0	2	1,5420	0,4434	0,5245
2	2	1,5420	0,8465	-0,2521	0,9398
3	1,5420	0,8465	1,1557	0,0571	0,5311
4	0,8465	1,1557	1,1056	0,0070	0,7144
5	1,1557	1,1056	1,0984	-0,0002	0,6221
6	1,1056	1,0984	1,0986	7×10^{-7}	0,6712

La última columna nos indica que el orden de convergencia observado es Φ , con una velocidad de convergencia alrededor de 0,67. Por la Observación 4.4.3, como en este ejemplo tenemos $f'(x^*) = -1$, $f''(x^*) = 1$, esperamos convergencia con velocidad $2^{-1/\Phi} \approx 0,65$. △

4.5. Método de Newton-Raphson

El método de la secante propone computar iterados a partir de la intersección del eje de las x con la recta secante al gráfico de f por los dos iterados anteriores. Esa recta secante tiene pendiente s^k (ver (4.4)); si x^k y x^{k-1} están lo suficientemente cerca, podemos considerar la aproximación

$$s^k = \frac{f(x^k) - f(x^{k-1})}{x^k - x^{k-1}} \approx f'(x^k).$$

Esto invita a considerar la siguiente iteración para hallar raíces de f ,

$$x^{k+1} := x^k - \frac{f(x^k)}{f'(x^k)}, \tag{4.9}$$

que es el llamado **método de Newton-Raphson**. Gráficamente, este método se corresponde con computar la recta tangente al gráfico de f por el punto $(x^k, f(x^k))$ e intersecarlo con el eje de las x (ver Figura 4.3). En efecto, tal recta tangente está dada por

$$y = f(x^k) + f'(x^k)(x - x^k),$$

e imponiendo $y = 0$ arriba y despejando x se llega fácilmente a (4.9).

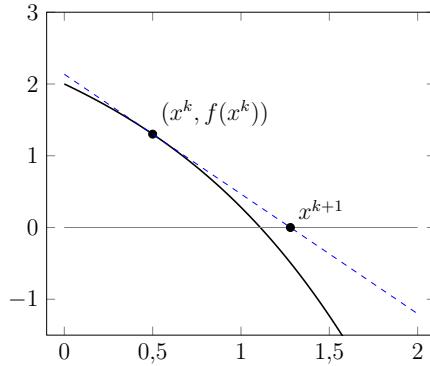


Figura 4.3: Método de Newton-Raphson.

Ejemplo 4.5.1 (el retorno de Herón). Sea $a > 0$, y consideremos $f(x) = x^2 - a$, que tiene una raíz en $x^* = \sqrt{a}$. Dado x^0 , la iteración de Newton-Raphson está dada por

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)} = x^k - \frac{(x^k)^2 - a}{2x^k} = x^k - \frac{x^k}{2} + \frac{a}{2x^k} = \frac{1}{2} \left(x^k + \frac{a}{x^k} \right).$$

Ya nos encontramos con esta iteración (con $a = 7$) en la Sección 1.2.3: el método de Herón para calcular raíces cuadradas es justamente el método de Newton-Raphson aplicado a la función $f(x) = x^2 - a$. Recordamos que en esa sección, demostramos –sin decirlo explícitamente– que el método converge con orden cuadrático; ver la fórmula (1.1). \triangle

Observación 4.5.1 (Taylor camuflado). Otra forma de pensar en el método de Newton es como la iteración que resulta al *linealizar* f alrededor de x^k , esto es, de aproximar f por su polinomio de Taylor de grado 1 en el punto x^k . En efecto, tal linearización indica

$$f(x) \approx T_1 f(x; x^k) = f(x^k) + f'(x^k)(x - x^k).$$

Como este polinomio es lineal, resulta trivial hallar su raíz y así queda definido x^{k+1} de la misma forma que en (4.9). \triangle

Observación 4.5.2 (comparación con secante). El método de Newton-Raphson no solo utiliza los valores de f , sino que también incorpora información sobre f' . Por lo tanto, es de esperar que presente una convergencia de mayor orden que el de la secante. Sin embargo, requiere que tengamos acceso a f' , lo que no siempre tiene por qué ocurrir en la práctica. \triangle

El método de Newton-Raphson en general no nos garantiza la convergencia hacia la raíz pero converge cuadráticamente bajo condiciones adecuadas, tal como ocurre en el Ejemplo 4.5.1.

Teorema 4.5.1 (convergencia y orden del método de Newton-Raphson). *Sea $f: I \rightarrow \mathbb{R}$ una función de clase C^2 , y x^* un punto interior a I y tal que $f(x^*) = 0$, $f'(x^*) \neq 0$. Entonces, existe $\delta > 0$ tal que si $|x^0 - x^*| < \delta$, el método de Newton-Raphson (4.9) converge a x^* , y la convergencia es con orden (por lo menos) cuadrático.*

Demostración. Postergamos la demostración de que la iteración de Newton-Raphson es convergente; ver el Corolario 4.7.4 abajo. Para probar que el método es de segundo orden, planteamos un desarrollo de Taylor para f alrededor de x^k , evaluado en x^* ,

$$0 = f(x^*) = f(x^k) + f'(x^k)(x^* - x^k) + \frac{f''(\theta_k)}{2}(x^* - x^k)^2,$$

para algún θ_k entre x^* y x^k . Como f' es continua (pues $f \in C^2$), $f'(x^*) \neq 0$, y el método es convergente, existe k_0 tal que $f'(x^k) \neq 0$ para todo $k \geq k_0$. Combinamos la definición (4.9) con el desarrollo de Taylor y obtenemos

$$e^{k+1} = x^{k+1} - x^* = x^k - \frac{f(x^k)}{f'(x^k)} - x^* = \frac{f''(\theta_k)}{2f'(x^k)}(x^* - x^k)^2 = \frac{f''(\theta_k)}{2f'(x^k)}(e^k)^2.$$

Por lo tanto, tomando límite en k , como $x^k \rightarrow x^*$ y $\theta_k \rightarrow x^*$ (porque está encerrado entre x^k y x^*),

$$\frac{|e^{k+1}|}{|e^k|^2} = \frac{|f''(\theta_k)|}{2|f'(x^k)|} \rightarrow \frac{|f''(x^*)|}{2|f'(x^*)|}.$$

Si $f''(x^*) \neq 0$, esto demuestra la convergencia con segundo orden y velocidad $\frac{|f''(x^*)|}{2|f'(x^*)|}$. Si $f''(x^*) = 0$ el límite de arriba es 0, por lo que la convergencia podría ser de mayor orden. \square

Observación 4.5.3 (condiciones favorables y desfavorables). El Teorema 4.5.1 nos da condiciones en las que podemos asegurar que el método de Newton-Raphson converge con (al menos) segundo orden, y nos da una pauta de que si $f''(x^*) = 0$ entonces podemos esperar que la convergencia sea aún con mayor orden. Por otra parte, cabe preguntarse qué ocurre con la iteración si $f'(x^*) = 0$ o f no es derivable en x^* , ya que ese caso no está en las hipótesis del teorema. Damos tres ejemplos sencillos para ilustrar lo que puede suceder en estos casos.

1. Sea $f: \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = x + x^3$, cuya única raíz es $x^* = 0$. Comenzamos con $x^0 > 0$ suficientemente pequeño, y la iteración de Newton-Raphson (4.9) para este problema es

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)} = x^k - \frac{x^k + (x^k)^3}{1 + 3(x^k)^2} = \frac{2(x^k)^3}{1 + 3(x^k)^2}$$

Como estamos aproximando la raíz $x^* = 0$, tenemos $e^k = x^k$ para todo k y por lo tanto obtenemos

$$|e^{k+1}| = \frac{2|e^k|^3}{1 + 3|e^k|^2} \leq 2|e^k|^3.$$

Por lo tanto, en este caso el método converge con orden $p = 3$.

2. Sean $\gamma > 1$ y $f: \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = |x|^\gamma$, cuya única raíz es $x^* = 0$. Notamos que $f'(x) = \gamma|x|^{\gamma-1}\operatorname{sgn}(x)$ y por lo tanto $f'(x^*) = 0$. Comenzamos con $x^0 > 0$, lo que da lugar a $x^k > 0$ para todo k , y la iteración para este problema queda

$$x^{k+1} = x^k - \frac{(x^k)^\gamma}{\gamma(x^k)^{\gamma-1}} = x^k \left(1 - \frac{1}{\gamma}\right).$$

Por lo tanto, la sucesión de errores aquí cumple

$$\frac{e^{k+1}}{e^k} = 1 - \frac{1}{\gamma},$$

y deducimos que el método de Newton-Raphson converge linealmente y con velocidad $1 - \frac{1}{\gamma} \in (0, 1)$.

3. Consideramos $f: \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = |x|^\gamma$, pero ahora suponiendo $0 < \gamma < 1$. Se puede realizar el mismo razonamiento del punto anterior, con la salvedad de que ahora f no es derivable en $x^* = 0$, y llegamos a

$$\frac{e^{k+1}}{e^k} = 1 - \frac{1}{\gamma}.$$

Como $0 < \gamma < 1$, tenemos $1 - \frac{1}{\gamma} < 0$. De hecho, si $1/2 < \gamma < 1$ aún tenemos $\left|1 - \frac{1}{\gamma}\right| < 1$ y podemos garantizar que el método converge linealmente. Si $0 < \gamma < 1/2$, entonces $\left|1 - \frac{1}{\gamma}\right| > 1$ y tendremos $|e^k| \rightarrow \infty$. Finalmente, en el caso límite $\gamma = 1/2$, la iteración de Newton-Raphson permanecerá “rebotando” entre x^0 y $-x^0$.

△

Ejemplo 4.5.2 (cuencas de convergencia). El comportamiento del método de Newton-Raphson comenzando lejos de las raíces puede ser un tanto errático. Consideremos $f: \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = (x + 3)(x - 1)(x - 4)$. Es evidente que esta función tiene raíces $-3, 1, 4$. Implementamos en Octave la iteración de Newton-Raphson con el criterio de que pare cuando el valor absoluto entre dos iterados sucesivos sea menor que $1\text{e-}3$, e inicializamos el método en $x^0 = 2,352$, $x^0 = 2,353$, y $x^0 = 2,354$. La siguiente tabla muestra los resultados que obtenemos.

k	$x^0 = 2,352$	$x^0 = 2,353$	$x^0 = 2,354$
1	-0,7760	-0,7843	-0,7927
2	2,3218	2,3590	2,3977
3	-0,5469	-0,8356	-1,2179
4	1,6331	2,6180	11,0674
5	0,9047	-11,1912	7,8614
6	0,9994	-7,4866	5,8484
7	1,0000	-5,1489	4,6860
8	—	-3,7926	4,1481
9	—	-3,1671	4,0094
10	—	-3,0100	4,0000
11	—	-3,0000	4,0000
12	—	-3,0000	—

¡Una perturbación del orden de 10^{-3} en el iterado inicial afecta a qué raíz converge el método! \triangle

4.6. Resumen: una muy breve comparación

Hasta aquí, hemos cubierto los siguientes métodos para hallar raíces de funciones de una variable: bisección, regla falsa, secante, y Newton-Raphson. Referimos al material [Pic] para más ejemplos y discusión sobre estos métodos.

Los métodos de bisección y de la regla falsa son de encierro: nos aseguran la convergencia si nuestra función es continua y le encontramos un cambio de signo. La contracara es que son métodos de primer orden, por lo que la convergencia puede ser lenta.

El método de la secante propone una variante sobre el de la regla falsa, con una convergencia de orden $\Phi > 1$. Sin embargo, para garantizar que este método sea convergente necesitamos inicializarlo lo suficientemente cerca de la raíz buscada. Por esta razón, puede ser conveniente –tal como hacen funciones de Octave como `fzero`– implementar la estrategia de comenzar con unos iterados con métodos de encierro (de convergencia segura, pero lenta) e ir ensayando pasos con métodos de mayor orden pero cuya convergencia no está garantizada. En la medida que los métodos de orden alto muestran un comportamiento estable, se los usa⁵.

Por otra parte, el método de Newton-Raphson es aún de mayor orden que el de la secante, ya que bajo condiciones adecuadas es de segundo orden; esto tiene sentido, ya que incorpora información no solo sobre los valores de f sino sobre los de su derivada f' . Sin embargo, para aplicar este método se requiere evaluar f' , lo que no siempre está disponible en la práctica. Al igual que el método de la secante, para que el de Newton-Raphson

⁵De hecho, `fzero` utiliza tanto el método de la secante como la llamada *Inverse Quadratic Interpolation*, que es otro método rápido pero no necesariamente convergente. Referimos a [Mol04, Sección 4.6] para más detalles.

converja se requiere inicializarlo lo suficientemente cerca de la raíz deseada. De todos modos, el método de Newton-Raphson es ampliamente utilizado en la práctica para hallar raíces de funciones reales. También es un ejemplo de un *método iterativo general*, tema que analizamos en la próxima sección.

4.7. Métodos iterativos generales

Seguimos con nuestro estudio de métodos para computar raíces de funciones $f: \mathbb{R} \rightarrow \mathbb{R}$. La iteración (4.9) para el método de Newton-Raphson sugiere que puede ser de utilidad transformar nuestro problema para f en uno equivalente de hallar puntos fijos. Esto es, buscamos una función $g: \mathbb{R} \rightarrow \mathbb{R}$ tal que

$$f(x^*) = 0 \Leftrightarrow g(x^*) = x^*, \quad (4.10)$$

y proponemos un método de la forma

$$\begin{cases} x^{k+1} = g(x^k) \\ x^0 \in \mathbb{R} \end{cases}. \quad (\text{MIG})$$

Nos referiremos a un método que se pueda escribir de esta forma como un **método iterativo general**.

Naturalmente, dada una función $f: \mathbb{R} \rightarrow \mathbb{R}$, existen infinitas formas posibles de elegir g de modo que se cumpla (4.10). Una de estas formas es $g(x) := x - \frac{f(x)}{f'(x)}$, que, como observamos, corresponde al método de Newton-Raphson y muestra que éste es un método iterativo general. Otra posibilidad para construir un MIG a partir de una función f , es tomar, por ejemplo, $g(x) := x - f(x)$.

La Observación 4.5.3 muestra que no siempre podemos esperar que la iteración (MIG) sea convergente. Sin embargo, el Teorema 4.5.1 sugiere que la derivada de la función f (y por lo tanto de la función g para nuestro MIG) debe cumplir un rol fundamental en la convergencia de estos métodos. Como muestra el Corolario 4.7.2 (ver abajo), para que (MIG) sea convergente basta con asegurar que $|g'|$ sea menor que 1 a lo largo de la iteración.

Ejemplo 4.7.1 (distintos MIGs). Buscamos aproximar la raíz $x^* = 2$ de la función $f(x) = x^2 - x - 2$. Para ello, tomamos $x^0 = 1,5$ y consideramos las siguientes funciones:

$$\begin{aligned} x^2 - x - 2 = 0 &\Leftrightarrow x = x^2 - 2 \rightsquigarrow g_1(x) := x^2 - 2, \\ x^2 - x - 2 = 0 &\Leftrightarrow x^2 = x + 2 \rightsquigarrow g_2(x) := \sqrt{x + 2}, \\ x^2 - x - 2 = 0 &\Leftrightarrow x(x - 1) = 2 \Leftrightarrow x = 1 + \frac{2}{x} \rightsquigarrow g_3(x) := 1 + \frac{2}{x}, \\ x^2 - x - 2 = 0 &\Leftrightarrow x^2 - x = 2 \Leftrightarrow x(2x - 1) = x^2 + 2 \rightsquigarrow g_4(x) := \frac{x^2 + 2}{2x - 1}. \end{aligned}$$

Observamos que, por construcción, las funciones g_1, \dots, g_4 satisfacen (4.10). Implementamos en Octave un código que compute la iteración (MIG) para estas cuatro funciones y mostramos los primeros iterados a continuación.

k	(MIG) con g_1	(MIG) con g_2	(MIG) con g_3	(MIG) con g_4
0	1,5	1,5	1,5	1,5
1	0,2500	1,8708	2,3333	2,1250
2	-1,9375	1,9674	1,8571	2,0048
3	1,7539	1,9918	2,0769	2,0000
4	1,0762	1,9980	1,9630	2,0000

El comportamiento de las cuatro iteraciones es bien diferente. La iteración con g_1 parece no estar convergiendo, mientras que las otras tres sí. La iteración con g_4 parece ser la de convergencia más rápida. De las otras dos, la iteración con g_2 se comporta monótonamente y parece estar convergiendo más rápidamente que la que usa g_3 , que oscila. Notemos también que la iteración con g_4 corresponde al método de Newton-Raphson (4.9). \triangle

4.7.1. Convergencia

La siguiente definición nos permite dar una condición compacta para asegurar la convergencia de un método iterativo general.

Definición 4.7.1 (función contractiva). Una función $g: \mathbb{R} \rightarrow \mathbb{R}$ se dice **contractiva** en $I \subset \mathbb{R}$ si existe $0 \leq m < 1$ tal que

$$|g(x) - g(y)| \leq m |x - y| \quad \forall x, y \in I. \quad (4.11)$$

\triangle

Ejercicio 4.2. a) Demostrar que toda función contractiva en un intervalo I es continua en todo punto $x \in I$. Más aún, probar que si g es contractiva en I entonces es **uniformemente continua** en I .

b) Probar que si g es una función derivable en I , y existe un $\lambda < 1$ tal que $|g'(x)| < \lambda$ para todo $x \in I$, entonces g es contractiva en I .

El siguiente es un resultado muy general, que en nuestro contexto nos permite asegurar la convergencia de (MIG).

Teorema 4.7.1 (de punto fijo). *Sea $I \subset \mathbb{R}$ un intervalo cerrado, y sea $g: I \rightarrow I$ una función contractiva. Entonces, existe un único $x^* \in I$ tal que $g(x^*) = x^*$. Más aún, para todo $x^0 \in I$, si se define la sucesión $\{x^k\}_{k \in \mathbb{N}}$ mediante (MIG), se tiene*

$$\lim_{k \rightarrow \infty} x^k = x^*.$$

Demostración. Sea $m < 1$ como en (4.11). Procedemos en tres pasos.

1. Unicidad del punto fijo. Supongamos que g tiene dos puntos fijos, esto es, que existen $x^*, y^* \in I$ tales que $g(x^*) = x^*$, $g(y^*) = y^*$. Entonces, tenemos

$$|x^* - y^*| = |g(x^*) - g(y^*)| \leq m|x^* - y^*|.$$

Por lo tanto, como $m < 1$, solamente puede ser $x^* = y^*$.

2. La sucesión dada por (MIG) es de Cauchy. Dado $x^0 \in I$, definimos $\{x^k\}_{k \in \mathbb{N}}$ mediante (MIG) y probemos que es de Cauchy. Dado $\epsilon > 0$, basta con probar que existe $k_0 > 0$ tal que si $k, j > k_0$ entonces $|x^j - x^k| < \epsilon$. Sin perder generalidad, asumimos $k < j := k + \ell$. Entonces, aplicando la desigualdad triangular sucesivamente, tenemos

$$|x^k - x^j| \leq |x^k - x^{k+1}| + |x^{k+1} - x^{k+2}| + \dots + |x^{k+\ell-1} - x^j| = \sum_{i=0}^{\ell-1} |x^{k+i} - x^{k+i+1}|.$$

Ahora, para cualquier $i = 0, \dots, \ell - 1$, usando la definición (MIG) y el hecho de que g es contractiva, tenemos

$$\begin{aligned} |x^{k+i} - x^{k+i+1}| &= |g(x^{k+i-1}) - g(x^{k+i})| \\ &\leq m|x^{k+i-1} - x^{k+i}| \leq \dots \leq m^{k+i}|x^0 - x^1|. \end{aligned}$$

Por lo tanto, obtenemos

$$\begin{aligned} |x^k - x^j| &\leq \sum_{i=0}^{\ell-1} |x^{k+i} - x^{k+i+1}| \leq |x^0 - x^1| \left(\sum_{i=0}^{\ell-1} m^{k+i} \right) \\ &= |x^1 - x^0| m^k \left(\sum_{i=0}^{\ell-1} m^i \right) = |x^1 - x^0| m^k \left(\frac{1-m^\ell}{1-m} \right), \end{aligned}$$

donde en el último paso usamos la suma geométrica $\sum_{i=0}^{\ell-1} m^i = \frac{1-m^\ell}{1-m}$. Como $1 - m^\ell < 1$ y los términos $|x^1 - x^0|$, $1 - m$ están fijos, el lado derecho arriba tiende a 0 con $k \rightarrow \infty$. Por lo tanto, tomando k_0 suficientemente grande podemos afirmar que si $j > k \geq k_0$ entonces $|x^k - x^j| < \epsilon$. Esto quiere decir que $\{x^k\}$ es de Cauchy.

3. Existencia del punto fijo. Como el intervalo I es cerrado y $\{x^k\}$ es de Cauchy, tenemos que $\{x^k\}$ tiene límite en I . Sea x^* tal límite. Solamente nos falta demostrar que x^* es un punto fijo para g . Como g es contractiva en I , es continua (Ejercicio 4.2, parte a)) y por lo tanto

$$g(x^*) = \lim_{k \rightarrow \infty} g(x^k) = \lim_{k \rightarrow \infty} x^{k+1} = x^*.$$

Observamos, además, que como por el paso 1. solamente puede haber un punto fijo, tenemos $x^k \rightarrow x^*$ independientemente de cómo elijamos x^0 . \square

Observación 4.7.1 (completitud). En el Teorema 4.7.1, es fundamental asumir que el intervalo I es cerrado; remarcamos que I incluso podría ser una semirrecta $[a, \infty)$ o $(-\infty, a]$, o incluso todo \mathbb{R} . La propiedad subyacente que estamos usando es la **completitud** de I , que nos permite afirmar que toda sucesión de Cauchy en I tiene límite en I . \triangle

Ya tenemos los ingredientes necesarios para dar condiciones que nos permitan asegurar la convergencia de (MIG).

Corolario 4.7.2 (convergencia de (MIG)). *Sea x^* un punto fijo de g y supongamos que existe un entorno $I_\delta = (x^* - \delta, x^* + \delta)$ en el que g es contractiva. Entonces, si $x^0 \in I_\delta$, la sucesión $\{x^k\}_{k \in \mathbb{N}}$ dada por (MIG) verifica*

$$x^k \in I_\delta \quad \forall k, \quad \lim_{k \rightarrow \infty} x^k = x^*.$$

Demostración. Sea $x^0 \in I_\delta$. Nos basta con hallar un intervalo cerrado $\tilde{I}_\delta \subset I_\delta$ que contenga a x^0 y a x^* . En efecto, supongamos $\tilde{I}_\delta = [x^* - \tilde{\delta}, x^* + \tilde{\delta}]$ para algún $0 < \tilde{\delta} < \delta$. Como g es contractiva en I_δ , es fácil verificar que también lo es en \tilde{I}_δ . Por lo tanto, para todo $y \in \tilde{I}_\delta$, tenemos

$$\begin{aligned} g(y) &= g(y) - g(x^*) + x^* > -m|y - x^*| + x^* > x^* - \tilde{\delta}, \\ g(y) &= g(y) - g(x^*) + x^* < m|y - x^*| + x^* < x^* + \tilde{\delta}, \end{aligned}$$

y concluimos $g(y) \in \tilde{I}_\delta$ para todo $y \in \tilde{I}_\delta$. Como $g|_{\tilde{I}_\delta}$ verifica las hipótesis del Teorema 4.7.1, concluimos que (MIG) converge al único punto fijo de g en \tilde{I}_δ , que es x^* . \square

A continuación, damos un caso particular del Corolario 4.7.2 que es de utilidad práctica.

Corolario 4.7.3 (condición suficiente de convergencia). *Supongamos que g es de clase C^1 , sea x^* un punto fijo de g y supongamos que $|g'(x^*)| < 1$. Entonces, existe $\delta > 0$ tal que si $|x^0 - x^*| < \delta$, el método (MIG) converge a x^* .*

Demostración. Al ser g' una función continua y tener $|g'(x^*)| < 1$, existen $\lambda < 1$ y $\delta > 0$ tal que si $|x - x^*| < \delta$, entonces $|g'(x)| < \lambda$. Por el Ejercicio 4.2, parte b), deducimos que g es contractiva en $(x^* - \delta, x^* + \delta)$ y podemos aplicar el Corolario 4.7.2. \square

Ejemplo 4.7.2 (distintos MIGs, revisitado). Volvemos al Ejemplo 4.7.1, en el que buscamos la raíz $x^* = 2$. Tenemos

$$\begin{aligned} g'_1(x) &= 2x - 1 & \Rightarrow g'_1(x^*) &= 3, \\ g'_2(x) &= \frac{1}{2\sqrt{x+2}} & \Rightarrow g'_2(x^*) &= \frac{1}{4}, \\ g'_3(x) &= -\frac{2}{x^2} & \Rightarrow g'_3(x^*) &= -\frac{1}{2}, \\ g'_4(x) &= \frac{2x(2x-1)-2(x^2+2)}{(2x-1)^2} & \Rightarrow g'_4(x^*) &= 0. \end{aligned}$$

Esto muestra que la iteración usando g'_1 no es convergente, mientras las otras tres sí lo son, siempre y cuando las inicialicemos lo suficientemente cerca de x^* . Además, observamos que cuanto menor sea $|g'_i(x^*)|$ más rápida parece ser la convergencia. \triangle

Un caso particular del Corolario 4.7.3 es la convergencia del método de Newton-Raphson, que tenemos pendiente del Teorema 4.5.1.

Corolario 4.7.4 (convergencia de Newton-Raphson). *En las condiciones del Teorema 4.5.1, el método de Newton-Raphson (4.9) converge a x^* .*

Demostración. Ya sabemos que (4.9) es de la forma (MIG), tomando $g(x) = x - \frac{f(x)}{f'(x)}$. Nos basta con verificar que se verifica $|g'(x^*)| < 1$ y aplicar el Corolario 4.7.3. Derivamos g , y obtenemos

$$g'(x) = 1 - \frac{f'(x)f'(x) - f''(x)f(x)}{f'(x)^2} = \frac{f''(x)f(x)}{f'(x)^2}.$$

Por lo tanto, $g'(x^*) = 0$. □

4.7.2. Orden y velocidad de convergencia

A continuación establecemos el orden y la velocidad con la que convergen los métodos iterativos generales.

Teorema 4.7.5 (orden y velocidad de convergencia). *Sea g de clase C^r , y supongamos que la sucesión $\{x^k\}_{k \in \mathbb{N}}$ generada por (MIG) converge a x^* , siendo x^* punto fijo de g . Entonces, x^k converge a x^* con orden $p \leq r$ y velocidad $\frac{|g^{(p)}(x^*)|}{p!}$, si y solo si $g^{(i)}(x^*) = 0$ para $i = 1, \dots, p-1$ y $g^{(p)}(x^*) \neq 0$.*

Demostración. (\Leftarrow) Realizamos un desarrollo de Taylor para g alrededor de x^* ,

$$g(x) = g(x^*) + \sum_{i=1}^{p-1} \frac{g^{(i)}(x^*)}{i!} (x - x^*)^i + \frac{g^{(p)}(\theta_x)}{p!} (x - x^*)^p = x^* + \frac{g^{(p)}(\theta_x)}{p!} (x - x^*)^p,$$

para algún θ_x entre x^* y x . Evaluamos este desarrollo en x^k , notando que $x^{k+1} = g(x^k)$, para obtener

$$x^{k+1} = g(x^k) = x^* + \frac{g^{(p)}(\theta_k)}{p!} (x^k - x^*)^p.$$

Reordenando, esta relación se puede escribir en función de los errores como

$$e^{k+1} = \frac{g^{(p)}(\theta_k)}{p!} (e^k)^p.$$

Como $x^k \rightarrow x^*$, $\theta_k \rightarrow x^*$ y $g^{(p)}$ es continua, tomando límite en k tenemos

$$\lim_{k \rightarrow \infty} \frac{|e^{k+1}|}{|e^k|^p} = \frac{|g^{(p)}(x^*)|}{p!} \neq 0,$$

lo que demuestra que la convergencia es con orden p y velocidad $\frac{|g^{(p)}(x^*)|}{p!}$.

(\Rightarrow) Supongamos ahora que (MIG) converge con orden p y velocidad $\frac{|g^{(p)}(x^*)|}{p!}$, lo que significa que

$$\lim_{k \rightarrow \infty} \frac{|e^{k+1}|}{|e^k|^p} = \frac{|g^{(p)}(x^*)|}{p!} \neq 0.$$

Esto implica que $g^{(p)}(x^*) \neq 0$. Veamos ahora que $g^{(i)}(x^*) = 0, \forall i = 1, \dots, p-1$. Supongamos por absurdo que alguna de estas derivadas no es nula, y sea j el orden de la primer derivada no nula en x^* , esto es,

$$g^{(i)}(x^*) = 0, \forall i = 1, \dots, j-1, g^{(j)}(x^*) \neq 0.$$

Hacemos un desarrollo de Taylor para g alrededor de x^* de orden $j-1$ y lo evaluamos en x^k ,

$$g(x^k) = g(x^*) + 0 + \dots + 0 + \frac{g^{(j)}(\theta_k)}{j!}(x^k - x^*)^j$$

para algún θ_k entre x^k y x^* . Como $x^{k+1} = g(x^k)$ y $x^* = g(x^*)$, reordenando arriba obtenemos

$$\frac{|e^{k+1}|}{|e^k|^j} = \frac{|g^{(j)}(\theta_k)|}{j!}$$

Al ser $\theta_k \rightarrow x^*$ y $g^{(j)}$ continua, el término de la derecha tiende a $\frac{|g^{(j)}(x^*)|}{j!} \neq 0$. Pero como $j < p$ y $x^k \rightarrow x^*$ con orden p , el término de la izquierda tiende a 0, lo que es una contradicción. \square

Observación 4.7.2 (derivada de la función de iteración). Del Teorema 4.7.5 se deduce que, al diseñar un método iterativo general (MIG) para hallar una raíz x^* de una cierta función f , el método tendrá orden igual a la primera derivada de g que no se anule en x^* . Como un método de primer orden es convergente si su velocidad es menor que 1 (Observación 4.1.1), notamos que el Teorema 4.7.5 nos devuelve la condición $|g'(x^*)| < 1$ para garantizar la convergencia, al igual que el Corolario 4.7.2. Además, para un método iterativo general que converja con primer orden, la convergencia será más rápida en la medida que $|g'(x^*)|$ sea lo menor posible. \triangle

4.7.3. Criterios de parada

Al considerar métodos iterativos, una pregunta relevante en la práctica es cuándo parar. Nuestra discusión aquí complementa a la de la Sección 4.1. Podemos considerar las siguientes posibilidades:

- **Usar la variación entre iterados consecutivos.** Dada una tolerancia $\varepsilon > 0$, detener el algoritmo cuando $|x^{k+1} - x^k| < \varepsilon$. Como $x^{k+1} = g(x^k)$ y $x^* = g(x^*)$, podemos aplicar el Teorema de Lagrange A.2.3 para deducir que

$$|x^{k+1} - x^*| = |g(x^k) - g(x^*)| = |g'(c_k)| |x^k - x^*|$$

para algún c_k entre x^k y x^* . Si asumimos $|g'| \leq m < 1$ en un entorno de x^* que contiene a x^k tendremos, usando la desigualdad triangular, que

$$|x^{k+1} - x^*| \leq m|x^k - x^*| \leq m(|x^k - x^{k+1}| + |x^{k+1} - x^*|).$$

Por lo tanto, como $|x^k - x^{k+1}| < \varepsilon$, deducimos que vale

$$|x^{k+1} - x^*| < \frac{m\varepsilon}{1-m}.$$

Si podemos asegurar que m no es muy cercano a 1 en el intervalo en el que estamos trabajando, usando este criterio tendremos un buen control sobre la magnitud del error al detener el algoritmo.

- **Usar los valores de f .** Dado $\varepsilon > 0$, detener el algoritmo cuando $|f(x^k)| < \varepsilon$. Es de esperar que se necesaria algún tipo de “buen condicionamiento” en juego si queremos estimar el error al detener el algoritmo usando este criterio (recordar la Figura 4.1). Supongamos $f'(x^*) \neq 0$. Si hacemos la linealización

$$f(x^k) \approx f(x^*) + f'(x^*)(x^k - x^*) = f'(x^*)(x^k - x^*),$$

llegamos a que si $|f(x^k)| < \varepsilon$ entonces

$$|x^k - x^*| \lesssim \frac{\varepsilon}{|f'(x^*)|}.$$

Esto cuantifica lo que ilustramos en la Figura 4.1: si $|f'(x^*)| \gg 0$, tendremos un buen control sobre el error al usar este criterio, mientras que si $f'(x^*) \approx 0$ este criterio no es muy significativo.

De hecho, supongamos $f'(x^*) = 0$ pero $f''(x^*) \neq 0$. Entonces, haciendo un desarrollo de Taylor de segundo orden tenemos

$$f(x^k) \approx f(x^*) + f'(x^*)(x^k - x^*) + \frac{f''(x^*)}{2}(x^k - x^*)^2 = \frac{f''(x^*)}{2}(x^k - x^*)^2.$$

Por lo tanto, si $|f(x^k)| < \varepsilon$ en este caso solamente podremos afirmar que

$$|x^k - x^*| \lesssim \sqrt{\frac{2\varepsilon}{|f''(x^*)|}}.$$

¡Nuestro control sobre el error es mucho más pobre! Naturalmente, este razonamiento se puede generalizar: si f tiene sus primeras $p-1$ derivadas nulas en x^* y $f^{(p)}(x^*) \neq 0$, entonces si $|f(x^k)| < \varepsilon$ solamente podremos afirmar que $|x^k - x^*| = O(\sqrt[p]{\varepsilon})$.

- **Usar el número de iteraciones.** Esta puede ser una importante red de seguridad en caso de que el método no esté convergiendo o lo haga muy lentamente. Podemos fijar que el algoritmo se detenga en caso de que el número de iteraciones llegue a una cantidad máxima fijada de antemano. De este modo, evitamos que el algoritmo entre en un ciclo sin fin, pero naturalmente no tenemos ningún tipo de información relevante sobre x^* en caso de que el algoritmo se detenga usando este criterio.

4.8. Sistemas de ecuaciones

En esta sección generalizamos algunos de los métodos que discutimos anteriormente en este capítulo a sistemas de ecuaciones. Consideramos sistemas de n ecuaciones no lineales en n variables, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$,

$$\left\{ \begin{array}{lcl} f_1(x_1, x_2, \dots, x_n) & = & 0 \\ f_2(x_1, x_2, \dots, x_n) & = & 0 \\ \vdots & & \vdots \\ f_n(x_1, x_2, \dots, x_n) & = & 0 \end{array} \right. . \quad (4.12)$$

Consideramos $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ tal que $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x})]^t$. En forma compacta, resolver (4.12) puede expresarse equivalentemente como

$$\text{hallar } \mathbf{x}^* = [x_1, x_2, \dots, x_n]^t \in \mathbb{R}^n \text{ tal que } \mathbf{f}(\mathbf{x}^*) = \mathbf{0}.$$

A diferencia de los sistemas de ecuaciones lineales, aquí no tenemos condiciones sencillas de verificar que nos permitan asegurar la existencia o unicidad de soluciones.

4.8.1. Métodos iterativos generales

Conceptualmente, la generalización al caso vectorial de los métodos iterativos generales que analizamos en la Sección 4.7 es bastante sencilla. En efecto, dada $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, buscamos construir una función $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ tal que

$$\mathbf{f}(\mathbf{x}^*) = \mathbf{0} \Leftrightarrow \mathbf{g}(\mathbf{x}^*) = \mathbf{x}^*, \quad (4.13)$$

y aproximamos \mathbf{x}^* mediante la sucesión

$$\left\{ \begin{array}{l} \mathbf{x}^{k+1} = \mathbf{g}(\mathbf{x}^k) \\ \mathbf{x}^0 \in \mathbb{R}^n \end{array} \right. . \quad (4.14)$$

Para simplificar la discusión, de aquí en más asumimos que \mathbf{g} es tan regular como sea necesario. No vamos a profundizar en aspectos técnicos de este tipo de métodos, pero con los conocimientos que hemos adquirido en la Sección 4.7, podemos generar algunas intuiciones. Por una parte, en el caso de métodos iterativos generales para ecuaciones no lineales en \mathbb{R} tenemos que la derivada de la función de iteración juega un rol preponderante para establecer la convergencia, asegurando la misma en caso de que la función sea contractiva. Para funciones de varias variables, la matriz jacobiana cumple el rol de la derivada, y por lo tanto es de esperar que deba asignarse una condición que asegure contractividad para la matriz jacobiana.

Por otro lado, de nuestros conocimientos de Geometría y Álgebra Lineal 2 sabemos que la intuición de “contraer direcciones” tiene que ver con la noción de vectores y valores propios. La siguiente noción nos permite tratar esta idea en forma compacta.

Definición 4.8.1 (radio espectral). Sea $A \in \mathcal{M}_n(\mathbb{R})$ una matriz cuyos valores propios (complejos) escribimos como $\lambda_1, \dots, \lambda_n$. El **radio espectral** de A , que denotamos por $\rho(A)$, se define como

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|.$$

△

Intuitivamente, la Definición 4.8.1 nos dice que el hecho de que una matriz $A \in \mathcal{M}_n(\mathbb{R})$ “contraiga” todas las direcciones es equivalente a que $\rho(A) \leq 1$ ⁶.

Combinando la definición de radio espectral con la matriz jacobiana, y en analogía con el Teorema 4.7.1, es de esperar que podamos garantizar que (4.14) converja a \mathbf{x}^* siempre y cuando la matriz jacobiana de \mathbf{g} tenga radio espectral menor que 1 en un entorno de \mathbf{x}^* , y de hecho, cuanto menor sea $\rho(\mathbb{J}_{\mathbf{g}}(\mathbf{x}^*))$ más rápida será la convergencia.

Ejemplo 4.8.1 (métodos iterativos en \mathbb{R}^2). Consideremos el sistema

$$\begin{cases} x_1 + 2x_2 - 2 = 0 \\ x_1^2 + 4x_2^2 - 4 = 0 \end{cases},$$

cuya solución $\mathbf{x}^* = [0, 1]^t$ buscamos aproximar, y analizamos dos alternativas usando (4.14). Consideramos $\mathbf{g}_1, \mathbf{g}_2: \mathbb{R}^2 \rightarrow \mathbb{R}^2$, tales que

$$\mathbf{g}_1(x_1, x_2) = \begin{bmatrix} 2x_1 + 2x_2 - 2 \\ x_1^2 + 4x_2^2 + x_2 - 4 \end{bmatrix}, \quad \mathbf{g}_2(x_1, x_2) = \begin{bmatrix} -2x_2 + 2 \\ \frac{\sqrt{4-x_1^2}}{2} \end{bmatrix}.$$

Notemos que ambas alternativas satisfacen (4.13), pero sus jacobianos son

$$\begin{aligned} \mathbb{J}_{\mathbf{g}_1}(x_1, x_2) &= \begin{bmatrix} 2 & 2 \\ 2x_1 & 8x_2 + 1 \end{bmatrix} \Rightarrow \mathbb{J}_{\mathbf{g}_1}(\mathbf{x}^*) = \begin{bmatrix} 2 & 2 \\ 0 & 9 \end{bmatrix}, \\ \mathbb{J}_{\mathbf{g}_2}(x_1, x_2) &= \begin{bmatrix} 0 & -2 \\ \frac{-x_1}{2\sqrt{4-x_1^2}} & 0 \end{bmatrix} \Rightarrow \mathbb{J}_{\mathbf{g}_2}(\mathbf{x}^*) = \begin{bmatrix} 0 & -2 \\ 0 & 0 \end{bmatrix}. \end{aligned}$$

Es sencillo verificar que los valores propios de $\mathbb{J}_{\mathbf{g}_1}(\mathbf{x}^*)$ son 9 y 2, mientras que $\mathbb{J}_{\mathbf{g}_2}(\mathbf{x}^*)$ solamente tiene valor propio 0 (que es doble). Deducimos que $\rho(\mathbb{J}_{\mathbf{g}_1}(\mathbf{x}^*)) = 9$ y que $\rho(\mathbb{J}_{\mathbf{g}_2}(\mathbf{x}^*)) = 0$, por lo que esperamos que la iteración usando \mathbf{g}_1 no converja y que la iteración usando \mathbf{g}_2 sí lo haga.

Implementamos en Octave estas iteraciones, comenzando con $\mathbf{x}^0 = [0, 5, 0, 5]^t$. La siguiente tabla muestra los primeros iterados e indica que nuestro razonamiento parece ser el correcto. Para no cargar la notación, usamos un punto y coma ; para marcar un cambio de fila.

⁶Es claro por definición que $\rho(A) \geq 0$ para toda matriz $A \in \mathcal{M}_n(\mathbb{R})$.

k	(4.14) con \mathbf{g}_1	(4.14) con \mathbf{g}_2
0	[0,5; 0,5]	[0,5; 0,5]
1	[0; -2,2500]	[1,0000; 0,9682]
2	[-6,5000; 14,0000]	[0,0635; 0,8660]
3	[13,0000; 836,2500]	[0,2679; 0,9995]
4	$[0,0017; 2,7983] \times 10^6$	[0,0010; 0,9910]

 \triangle

Observación 4.8.1 (criterios de parada). Los criterios de parada que imponemos sobre los métodos iterativos en \mathbb{R}^n son los mismos que comentamos en la Sección 4.7.3, con la única diferencia de que donde dice “valor absoluto” allí, aquí debemos usar alguna norma en \mathbb{R}^n ⁷. \triangle

4.8.2. Método de Newton-Raphson

Al igual que para ecuaciones en una variable, podemos considerar el método de Newton-Raphson para sistemas. Sea $\mathbf{f}: \mathbb{R}^n \rightarrow \mathbb{R}^n$ diferenciable, y buscamos \mathbf{x}^* tal que $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$. Dado $\mathbf{x}^k \in \mathbb{R}^n$, siguiendo el enfoque de la Observación 4.5.1 podemos linealizar \mathbf{f} alrededor de este punto, esto es, aproximar

$$\mathbf{f}(\mathbf{x}) \approx T_1 \mathbf{f}(\mathbf{x}; \mathbf{x}^k) = \mathbf{f}(\mathbf{x}^k) + \mathbb{J}_{\mathbf{f}}(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k),$$

y definir \mathbf{x}^{k+1} como el único punto en \mathbb{R}^n en el que $T_1 \mathbf{f}(\mathbf{x}^{k+1}; \mathbf{x}^k) = \mathbf{0}$. A diferencia del método de Newton-Raphson en una variable, aquí para hallar \mathbf{x}^{k+1} tendremos que resolver un sistema de ecuaciones lineal:

$$\mathbb{J}_{\mathbf{f}}(\mathbf{x}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) = -\mathbf{f}(\mathbf{x}^k). \quad (4.15)$$

De este modo, dado \mathbf{x}^k , para calcular el siguiente iterado \mathbf{x}^{k+1} podemos

- resolver el sistema lineal $\mathbb{J}_{\mathbf{f}}(\mathbf{x}^k)\mathbf{d}^{k+1} = -\mathbf{f}(\mathbf{x}^k)$,
- y luego tomar $\mathbf{x}^{k+1} := \mathbf{x}^k + \mathbf{d}^{k+1}$.

⁷Remarcamos que no es relevante cuál, ya que las normas en \mathbb{R}^n son todas equivalentes: si $\|\cdot\|$ y $\|\cdot\|$ son normas en \mathbb{R}^n , entonces existen constantes $c, c' > 0$ (que dependen de n) tales que

$$c\|\mathbf{v}\| \leq \|\mathbf{v}\| \leq c'\|\mathbf{v}\| \quad \forall \mathbf{v} \in \mathbb{R}^n.$$

El Algoritmo 4.2 describe este método en forma de pseudo-código.

Algoritmo 4.2: Pseudo-código: Newton-Raphson para sistemas.

Datos: $\mathbf{x}^0 \in \mathbb{R}^n$, $\mathbf{f}: \mathbb{R}^n \rightarrow \mathbb{R}^n$ diferenciable, $\mathbb{J}_{\mathbf{f}}: \mathbb{R}^n \rightarrow \mathcal{M}_n(\mathbb{R})$ su jacobiana,
 $\text{tol} > 0$, $\text{max_it} > 0$

Resultado: $\mathbf{x} \in \mathbb{R}^n$ que cumpla el criterio de parada que establezcamos
 $\mathbf{x} \leftarrow \mathbf{x}^0$

mientras [*criterio de parada*] $\geq \text{tol}$ & $k < \text{max_it}$ **hacer**

$\mathbf{b} \leftarrow -\mathbf{f}(\mathbf{x});$
 $A \leftarrow \mathbb{J}_{\mathbf{f}}(\mathbf{x});$
 $\mathbf{d} \leftarrow \text{solución del sistema } A\mathbf{d} = \mathbf{b};$
 $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{d};$
 $k \leftarrow k + 1;$

fin

Observación 4.8.2 (invertibilidad de $\mathbb{J}_{\mathbf{f}}$). Para que los iterados del método de Newton-Raphson estén bien definidos, es necesario que las matrices jacobianas $\mathbb{J}_{\mathbf{f}}(\mathbf{x}^k)$ sean invertibles. Esto es análogo a la versión (4.9) en \mathbb{R} , en la que necesitamos que $f'(x^k) \neq 0$. \triangle

Ejemplo 4.8.2 (Newton-Raphson en \mathbb{R}^2). Volvemos al problema del Ejemplo 4.8.1, al que ahora abordamos usando el método de Newton-Raphson en \mathbb{R}^2 . Para computar los iterados necesitamos la matriz jacobiana de \mathbf{f} , que es

$$\mathbb{J}_{\mathbf{f}}(x_1, x_2) = \begin{bmatrix} 1 & 2 \\ 2x_1 & 8x_2 \end{bmatrix}.$$

Comenzamos con $\mathbf{x}^0 = [0,5; 0,5]$. Para computar \mathbf{x}^1 primero resolvemos el sistema

$$\mathbb{J}_{\mathbf{f}}(\mathbf{x}^0)\mathbf{d}^1 = -\mathbf{f}(\mathbf{x}^0) \Rightarrow \begin{bmatrix} 1 & 2 \\ 1 & 4 \end{bmatrix} \mathbf{d}^1 = \begin{bmatrix} 1/2 \\ 11/4 \end{bmatrix} \Rightarrow \mathbf{d}^1 = \begin{bmatrix} -7/4 \\ 9/8 \end{bmatrix},$$

y por lo tanto

$$\mathbf{x}^1 = \mathbf{x}^0 + \mathbf{d}^1 = \begin{bmatrix} -5/4 \\ 13/8 \end{bmatrix}.$$

Siguiendo de este modo, implementando el Algoritmo 4.2 en Octave obtenemos los iterados

$$\mathbf{x}^1 = \begin{bmatrix} -1,2500 \\ 1,6250 \end{bmatrix}, \quad \mathbf{x}^2 = \begin{bmatrix} -0,3472 \\ 1,1736 \end{bmatrix}, \quad \mathbf{x}^3 = \begin{bmatrix} -0,0447 \\ 1,0224 \end{bmatrix}, \quad \mathbf{x}^4 = \begin{bmatrix} -0,0010 \\ 1,0005 \end{bmatrix}.$$

Observamos que la iteración converge a $\mathbf{x}^* = [0; 1]$. \triangle

Observación 4.8.3 (Newton-Raphson como método iterativo general). Sea $\mathbf{f}: \mathbb{R}^n \rightarrow \mathbb{R}^n$ una función de clase C^2 , con $\mathbf{f}(\mathbf{x}) = 0$ y $\mathbb{J}_{\mathbf{f}}(\mathbf{x}^*)$ invertible. El método de Newton-Raphson en \mathbb{R}^n es un método iterativo de la forma (4.14), en el que se toma

$$\mathbf{g}(\mathbf{x}) = \mathbf{x} - \mathbb{J}_{\mathbf{f}}(\mathbf{x})^{-1}\mathbf{f}(\mathbf{x}).$$

Es elemental (pero engoroso) verificar que las entradas de la matriz jacobiana de \mathbf{g} en \mathbf{x} están dadas por

$$(\mathbb{J}_{\mathbf{g}}(\mathbf{x}))_{ij} = \delta_{ij} - \sum_{r=1}^n \frac{\partial K_{ir}}{\partial x_j} f_r - \sum_{r=1}^n K_{ir} J_{rj}, \quad i, j = 1, \dots, n,$$

donde denotamos por δ_{ij} la delta de Kronecker ($\delta_{ij} = 1$ si $i = j$ y $\delta_{ij} = 0$ si no), $K = \mathbb{J}_{\mathbf{f}}(\mathbf{x})^{-1}$, y $J = \mathbb{J}_{\mathbf{f}}(\mathbf{x})$. Si evaluamos esta identidad en $\mathbf{x} = \mathbf{x}^*$ se obtiene que $\mathbb{J}_{\mathbf{g}}(\mathbf{x}^*) = 0$, lo que justifica que el método de Newton-Raphson en \mathbb{R}^n presente convergencia cuadrática.

△

4.8.3. Variantes al método de Newton-Raphson*

El método de Newton-Raphson tal como lo tratamos en estas notas es extremadamente popular para trabajar con problemas en una variable, pero no lo es tanto para sistemas de ecuaciones. Tiene ciertas limitaciones que hacen que muchas veces se opte por *variantes* al método.

No siempre converge. Bajo hipótesis adecuadas sobre \mathbf{f} , el método de Newton-Raphson converge a \mathbf{x}^* siempre y cuando lo inicialicemos lo suficientemente cerca de \mathbf{x}^* . Si estamos lejos de \mathbf{x}^* , el método no siempre es confiable. Una opción para mitigar una posible no convergencia del método de Newton-Raphson es **amortiguarlo**.

Concretamente, dado $\mathbf{x}^0 \in \mathbb{R}^k$, podemos considerar una sucesión $\{\alpha^k\} \subset (0, 1)$ (muchas veces se la elige simplemente como una constante) y, en cada paso,

- resolver el sistema lineal $\mathbb{J}_{\mathbf{f}}(\mathbf{x}^k)\mathbf{d}^{k+1} = -\mathbf{f}(\mathbf{x}^k)$,
- y luego tomar $\mathbf{x}^{k+1} := \mathbf{x}^k + \alpha^{k+1}\mathbf{d}^{k+1}$.

Costo computacional. El método de Newton-Raphson puede resultar inconveniente para resolver sistemas de ecuaciones, ya que requiere conocer las derivadas parciales de \mathbf{f} , y en cada paso hay que resolver un sistema de ecuaciones lineales distinto. Ensamblar las matrices jacobianas implica computar n^2 funciones escalares en cada paso, y la resolución de cada sistema –asumiendo que estas jacobianas son matrices densas– tiene un costo de $O(n^3)$ flops.

Los **métodos de quasi-Newton** apuntan a aliviar este elevado costo computacional. Una posibilidad es utilizar la misma matriz jacobiana en varias iteraciones consecutivas, lo que ahorra el trabajo de ensamblado y permite sacar ventaja de computar la factorización LU en cada actualización de la jacobiana. Naturalmente, esto afecta al orden cuadrático del método.

Otra opción consiste en reemplazar $\mathbb{J}_{\mathbf{f}}(\mathbf{x}^k)$ por una matriz $B^k \in \mathcal{M}_n(\mathbb{R})$ que la aproxime, pero que nos evite el trabajo de calcular las derivadas parciales de \mathbf{f} . Los métodos de tipo

secante son métodos de cuasi-Newton que logran obtener órdenes de convergencia mayor a 1, y toman B^k de modo que se cumpla

$$B^k(\mathbf{x}^{k+1} - \mathbf{x}^k) = \mathbf{f}(\mathbf{x}^{k+1}) - \mathbf{f}(\mathbf{x}^k).$$

Esta ecuación es análoga a la definición del método de la secante (4.4) en \mathbb{R} , con la diferencia de que aquí reemplazamos el escalar $s^k \in \mathbb{R}$ por la matriz $B^k \in \mathcal{M}_n(\mathbb{R})$. Un ejemplo popular de método de este tipo es el [método de Broyden](#). Referimos a [Hea02, Sección 5.6] para mayores detalles acerca de métodos de cuasi-Newton y otras variantes sobre el método de Newton-Raphson en \mathbb{R}^n . No profundizaremos sobre el tema en este curso.

Capítulo 5

Mínimos cuadrados

5.1. Introducción. Modelos y ajustes

El problema principal que abordamos en este capítulo tiene que ver con el *ajuste de datos*. Dados m datos u observaciones $\{(t_i, y_i)\}_{i=1,\dots,m} \subset \mathbb{R}^2$, buscamos una curva “que los ajuste”. A diferencia de lo que ocurre en el problema de interpolación que discutimos en el Capítulo 3, aquí tenemos un *modelo* de cómo deberían comportarse los datos. Ese modelo incluye una cierta cantidad n de *parámetros*, en general vamos a asumir $n < m$, y el objetivo es hallar los valores de parámetros que logren el “mejor ajuste” a los datos. La Figura 5.1 muestra dos ejemplos. En la izquierda, el modelo subyacente es que los datos siguen una relación lineal, esto es

$$y_i \approx f(t_i) := x_1 t_i + x_2, \quad i = 1, \dots, m.$$

Aquí, $x_1, x_2 \in \mathbb{R}$ son los $n = 2$ parámetros a determinar. La imagen de la derecha en la Figura 5.1 muestra un ajuste en el que se asume que los puntos (t_i, y_i) están distribuidos en una cónica,

$$x_1 t_i^2 + x_2 t_i y_i + x_3 y_i^2 + x_4 t_i + x_5 y_i + x_6 \approx 0, \quad i = 1, \dots, m,$$

por lo que tenemos $n = 6$ parámetros en nuestro modelo¹.

Estos dos ejemplos son levemente distintos en el sentido de que en uno de ellos se asume que y_i se puede escribir como una función de t_i , mientras que en el otro no. El segundo caso no reviste mayores dificultades técnicas (ver el ejercicio *Órbita planetaria* en el práctico) pero, para simplificar la discusión, vamos a asumir que existe una función $f: \mathbb{R} \rightarrow \mathbb{R}$ tal que

$$y_i \approx f(t_i), \quad \forall i = 1, \dots, m.$$

¹En este caso, si ya sabemos que los datos se distribuyen en una elipse, deberíamos imponer la restricción adicional $x_2^2 - 4x_1 x_3 < 0$. Si no imponemos esta restricción, simplemente estamos asumiendo que los puntos siguen una sección cónica (hipérbola, parábola, o elipse).

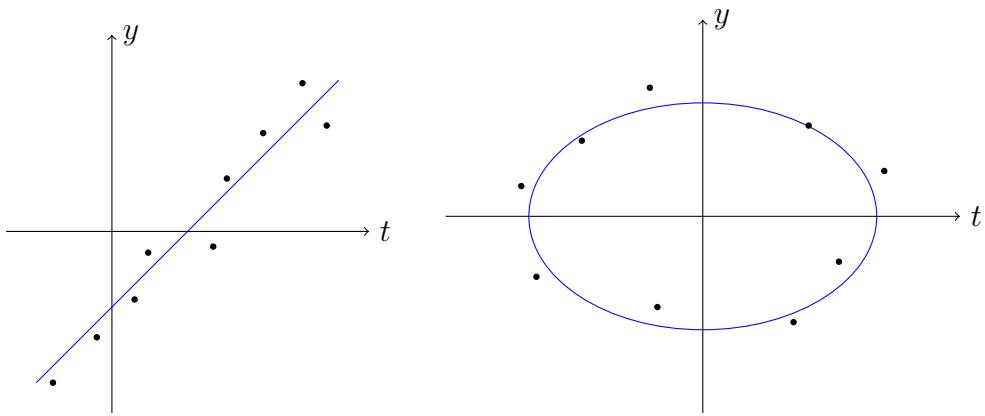


Figura 5.1: Ajustes de modelos a datos.

Queremos escribir esta función f a partir de un conjunto de funciones ϕ_1, \dots, ϕ_r conocidas (notar que podemos permitir $r \neq n$, ver el Ejemplo 5.1.1), y de un cierto conjunto de n parámetros. La dependencia en estos parámetros puede ser lineal o no. Veamos algunos ejemplos.

Ejemplo 5.1.1 (ejemplos de modelos). A continuación damos algunos ejemplos de cómo los modelos pueden depender linealmente respecto a ciertos parámetros pero no linealmente respecto a otros.

- *Polinomios:* si sabemos que $f(t)$ es un polinomio de grado $n - 1$, entonces podemos tomar el conjunto de funciones de base $\phi_1(t) = t^{n-1}, \phi_2(t) = t^{n-2}, \dots, \phi_n(t) = 1$, y podemos escribir

$$f(t) = x_1 t^{n-1} + x_2 t^{n-2} + \dots + x_n.$$

En este ejemplo tenemos $n = r$ y la dependencia en los parámetros es lineal. Es posible que el lector haya utilizado la función de Octave/Matlab `polyfit` en otros cursos: esa función hace ajustes de datos mediante modelos polinomiales en el sentido que describimos en este capítulo.

- *Exponentiales:* dados $\alpha_1, \dots, \alpha_r \in \mathbb{R}$, consideramos $\phi_j(t) = e^{-\alpha_j t}$ para $j = 1, \dots, r$, y tomamos

$$f(t) = x_1 e^{-\alpha_1 t} + x_2 e^{-\alpha_2 t} + \dots + x_r e^{-\alpha_r t}.$$

Aquí tenemos $n = 2r$ parámetros: la dependencia en x_1, \dots, x_r es lineal pero la dependencia en $\alpha_1, \dots, \alpha_r$ no lo es.

- *Funciones racionales:* dados $\alpha_1, \dots, \alpha_p \in \mathbb{R}$, consideramos $\phi_j(t) = \frac{t^{r-j}}{\alpha_1 t^{p-1} + \dots + \alpha_p}$ para $j = 1, \dots, r$, y tomamos

$$\begin{aligned} f(t) &= \frac{x_1 t^{r-1}}{\alpha_1 t^{p-1} + \dots + \alpha_p} + \frac{x_2 t^{r-2}}{\alpha_1 t^{p-1} + \dots + \alpha_p} + \dots + \frac{x_r}{\alpha_1 t^{p-1} + \dots + \alpha_p} \\ &= \frac{x_1 t^{r-1} + x_2 t^{r-2} + \dots + x_r}{\alpha_1 t^{p-1} + \dots + \alpha_p}. \end{aligned}$$

En este caso, tenemos $n = p + r$ parámetros y el modelo es lineal en x_1, \dots, x_r y no lineal respecto a $\alpha_1, \dots, \alpha_p$.

- *Gaussianas*: dados $\mu_1, \dots, \mu_r \in \mathbb{R}$ y $\sigma_1, \dots, \sigma_r > 0$, consideramos $\phi_j(t) = e^{-\left(\frac{t-\mu_j}{\sigma_j}\right)^2}$ y tomamos

$$f(t) = x_1 e^{-\left(\frac{t-\mu_1}{\sigma_1}\right)^2} + \dots + x_r e^{-\left(\frac{t-\mu_r}{\sigma_r}\right)^2}.$$

Aquí, tenemos $n = 3r$ parámetros, y la dependencia respecto a $\mu_1, \dots, \mu_r, \sigma_1, \dots, \sigma_r$ es no lineal, mientras que la dependencia respecto a x_1, \dots, x_r sí lo es.

△

En las primeras secciones del capítulo, nos vamos a concentrar en **problemas lineales**: dadas n funciones linealmente independientes $\phi_1(t), \dots, \phi_n(t)$, buscamos una función f de la forma

$$f(t) = \sum_{j=1}^n x_j \phi_j(t).$$

Un concepto útil para tratar con problemas lineales es el siguiente.

Definición 5.1.1 (matriz de diseño). Dados puntos $\{(t_i, y_i)\}_{i=1,\dots,m} \subset \mathbb{R}^2$ y funciones $\phi_1(t), \dots, \phi_n(t)$ como arriba, llamamos **matriz de diseño** asociada a ellos a $A = (a_{ij}) \in \mathcal{M}_{m \times n}(\mathbb{R})$, tal que

$$a_{ij} = \phi_j(t_i).$$

△

Notemos que la matriz de diseño almacena las evaluaciones de todas las funciones de base ϕ_1, \dots, ϕ_n en todos los puntos t_1, \dots, t_m . Por lo tanto, si tenemos un problema lineal y queremos $f(t_i) \approx y_i$ para todo $i = 1, \dots, m$, estamos buscando un vector de parámetros $\mathbf{x} = [x_1, \dots, x_n]^t$ tal que

$$\begin{cases} x_1 \phi_1(t_1) + x_2 \phi_2(t_1) + \dots + x_n \phi_n(t_1) &= y_1 \\ x_1 \phi_1(t_2) + x_2 \phi_2(t_2) + \dots + x_n \phi_n(t_2) &= y_2 \\ \vdots &\vdots \\ x_1 \phi_1(t_m) + x_2 \phi_2(t_m) + \dots + x_n \phi_n(t_m) &= y_m \end{cases}. \quad (5.1)$$

En forma compacta, este sistema se puede escribir como $A\mathbf{x} = \mathbf{y}$, donde $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ es la matriz de diseño, e $\mathbf{y} = [y_1, \dots, y_m]^t$. A diferencia de lo que estudiamos en los capítulos 2 y 3, este *no es un sistema cuadrado*; como $m > n$, en general esperamos que este sistema sea incompatible².

²Notemos que, si $m = n$, el problema (5.1) es un problema de interpolación, aunque no necesariamente de interpolación polinomial.

Dado que en general no tenemos esperanza de resolver el sistema (5.1) en forma exacta, necesitamos dar una noción de solución con la que podamos determinar valores “satisfactorios” de los parámetros. Dado un vector $\mathbf{x} \in \mathbb{R}^n$, consideremos el **residuo** $\mathbf{r} := A\mathbf{x} - \mathbf{y} \in \mathbb{R}^m$. Parece natural buscar un vector de parámetros $\mathbf{x} \in \mathbb{R}^n$ que minimice alguna norma del residuo. Para esto, puede ser útil recordar las normas ℓ^p que introdujimos en la Sección 2.5.1: apuntar a minimizar las normas ℓ^1 o ℓ^∞ del residuo tiene aplicaciones en la práctica y pueden entenderse como problemas de [programación lineal](#). En este capítulo nos vamos a concentrar en el problema de minimizar la norma euclídea (ℓ^2) del residuo.

Definición 5.1.2 (mínimos cuadrados). Dados $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, $\mathbf{y} \in \mathbb{R}^m$, decimos que un vector $\mathbf{x}^* \in \mathbb{R}^n$ es **solución del sistema** $A\mathbf{x} = \mathbf{y}$ en el sentido de **mínimos cuadrados** si

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \|A\mathbf{x} - \mathbf{y}\|_2^2.$$

Esto es, si consideramos el residuo³ $\mathbf{r} = A\mathbf{x} - \mathbf{y}$, entonces \mathbf{x}^* minimiza la función

$$\Phi: \mathbb{R}^m \rightarrow \mathbb{R}, \quad \Phi(\mathbf{x}) := \sum_{i=1}^m r_i^2 = \sum_{i=1}^m \left[\left(\sum_{j=1}^n a_{ij} x_j \right) - y_i \right]^2.$$

△

Observación 5.1.1 (cuadrado de la norma). Como seguramente el lector ya se haya encontrado en algún curso de cálculo, es fácil comprobar que el punto en el que se minimiza la norma euclídea es el mismo que en el que se minimiza el cuadrado de la norma euclídea. En la definición anterior usamos el cuadrado de la norma euclídea simplemente porque es más cómodo para trabajar. △

Observación 5.1.2 (mínimos cuadrados con pesos). A veces puede ocurrir que queramos darle más importancia al ajuste de ciertos puntos que a otros. Por ejemplo, esto puede ocurrir si los $\{(t_i, y_i)\}$ corresponden a observaciones experimentales y algunas de estas observaciones fueron tomadas en condiciones más favorables, o con mejor equipamiento. En ese caso, se puede tomar $w_1, \dots, w_m > 0$ y apuntar a minimizar la función

$$\Phi_{\mathbf{w}}(\mathbf{x}) := \sum_{i=1}^m w_i r_i^2.$$

Esto se llama un *problema de mínimos cuadrados con pesos*. Dado $\mathbf{w} := [w_1, \dots, w_m]^t$, pasar del sistema correspondiente a un problema sin pesos a uno con pesos es extremadamente simple: ensamblamos la matriz diagonal $D_{\mathbf{w}} \in \mathcal{M}_m(\mathbb{R})$ que tenga \mathbf{w} en la diagonal, y el sistema asociado al problema con pesos es

$$D_{\mathbf{w}} A \mathbf{x} = D_{\mathbf{w}} \mathbf{y},$$

donde $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ e $\mathbf{y} \in \mathbb{R}^m$ son respectivamente la matriz de diseño y el vector de datos del problema original. △

³Observar que el residuo depende de \mathbf{x} .

A continuación, en la Sección 5.2, encontramos una caracterización elegante de la solución de mínimos cuadrados, las llamadas *ecuaciones normales*. Esta caracterización conduce a resolver un sistema lineal de tamaño $n \times n$, y es una estrategia viable en algunos casos. Sin embargo, las ecuaciones normales pueden sufrir problemas de condicionamiento, por lo que no siempre son una buena estrategia computacional. En la Sección 5.3 mostramos un camino de resolución basado en una factorización particular de la matriz de diseño, llamada descomposición QR , y estudiamos cómo obtener esta factorización de forma estable. Finalmente, en la Sección 5.4 analizamos un método para la resolución de problemas de mínimos cuadrados no lineales.

5.2. Ecuaciones normales

En esta sección, hallamos y analizamos una caracterización elegante de las soluciones de mínimos cuadrados, las llamadas *ecuaciones normales*.

Teorema 5.2.1 (ecuaciones normales). *Sean $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ e $\mathbf{y} \in \mathbb{R}^m$. Entonces, $\mathbf{x}^* \in \mathbb{R}^n$ es solución del sistema $A\mathbf{x} = \mathbf{y}$ en el sentido de los mínimos cuadrados (Definición 5.1.2) si y sólo si*

$$A^t A \mathbf{x}^* = A^t \mathbf{y}. \quad (5.2)$$

A esta fórmula se la conoce como las **ecuaciones normales**.

Demostración. Consideremos la función $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}$,

$$\Phi(\mathbf{x}) := \|\mathbf{r}(\mathbf{x})\|_2^2 = \|A\mathbf{x} - \mathbf{y}\|_2^2 = (A\mathbf{x} - \mathbf{y})^t (A\mathbf{x} - \mathbf{y}),$$

que es la función que queremos minimizar. Desarrollando, encontramos

$$\Phi(\mathbf{x}) = \mathbf{x}^t A^t A \mathbf{x} - 2\mathbf{x}^t A^t \mathbf{y} + \mathbf{y}^t \mathbf{y}.$$

Es fácil verificar que Φ es de clase C^∞ en \mathbb{R}^n , ya que es una función polinomial (cuadrática) en \mathbf{x} . En caso de tener un mínimo, éste se debe alcanzar en un punto crítico, esto es, debemos hallar $\mathbf{x}^* \in \mathbb{R}^n$ tal que $\nabla \Phi(\mathbf{x}^*) = \mathbf{0}$. Dado $i \in \{1, \dots, n\}$, tenemos

$$\begin{aligned} \frac{\partial \Phi}{\partial x_i}(\mathbf{x}) &= \frac{\partial}{\partial x_i} \left(\sum_{j=1}^n \sum_{k=1}^n x_j (A^t A)_{jk} x_k \right) - 2 \frac{\partial}{\partial x_i} \left(\sum_{j=1}^n \sum_{k=1}^m x_j (A^t)_{jk} y_k \right) \\ &= 2 \sum_{k=1}^n (A^t A)_{ik} x_k - 2 \sum_{k=1}^n (A^t)_{ik} y_k = 2(A^t A \mathbf{x})_i - 2(A^t \mathbf{y})_i. \end{aligned}$$

Por lo tanto, deducimos que $\nabla \Phi(\mathbf{x}^*) = \mathbf{0}$ si y solo si $A^t A \mathbf{x}^* - A^t \mathbf{y} = \mathbf{0}$.

Nos resta verificar que tal punto crítico efectivamente es un mínimo de Φ . Sea \mathbf{x}^* tal que $A^t A \mathbf{x}^* = A^t \mathbf{y}$ y sea $\mathbf{z} \in \mathbb{R}^n$ arbitrario. Usamos la condición de punto crítico para escribir

$$\begin{aligned} \Phi(\mathbf{z}) - \Phi(\mathbf{x}^*) &= \mathbf{z}^t A^t A \mathbf{z} - 2\mathbf{z}^t A^t \mathbf{y} + \mathbf{y}^t \mathbf{y} - (\mathbf{x}^{*t} A^t A \mathbf{x}^* - 2\mathbf{x}^{*t} A^t \mathbf{y} + \mathbf{y}^t \mathbf{y}) \\ &= \mathbf{z}^t A^t A \mathbf{z} - 2\mathbf{z}^t A^t A \mathbf{x}^* + \mathbf{x}^{*t} A^t A \mathbf{x}^* \\ &= A \mathbf{z} \cdot A \mathbf{z} - 2A \mathbf{z} \cdot A \mathbf{x}^* + A \mathbf{x}^* \cdot A \mathbf{x}^* = \|A \mathbf{z} - A \mathbf{x}^*\|_2^2 \geq 0. \end{aligned} \quad (5.3)$$

Por lo tanto, \mathbf{x}^* es un mínimo de Φ . Concluimos que minimizar Φ es equivalente a que se satisfaga la condición de punto crítico $A^t A \mathbf{x}^* = A^t \mathbf{y}$. \square

Observación 5.2.1 (unicidad). Remarcamos que el Teorema 5.2.1 no indica que el problema de mínimos cuadrados tenga solución única. De hecho, esto solamente ocurre si A es de rango completo (esto es, $\text{rg}(A) = n$): en ese caso, $A^t A \in \mathcal{M}_n(\mathbb{R})$ también tendría que tener rango igual a n , lo que implica que debe ser invertible y por lo tanto las ecuaciones normales tienen solución única. En este caso tenemos que $A^t A$ es definida positiva e incluso podríamos usar el criterio de la Hessiana para clasificar el único punto crítico de Φ : basta con notar que $D^2\Phi(\mathbf{x}) = A^t A$ para todo $\mathbf{x} \in \mathbb{R}^n$.

En caso de que A sea de rango deficiente (esto es, $\text{rg}(A) < n$), entonces $A^t A$ no es invertible, y las ecuaciones normales constituyen un sistema indeterminado. En la Sección 6.2.2 del Capítulo 6 analizamos este tipo de problemas. Por el momento, solamente comentamos que en este caso tenemos que $A^t A$ es semidefinida positiva y por lo tanto el uso del criterio de la Hessiana en la demostración del Teorema 5.2.1 requeriría algo de cuidado adicional. En cambio, enfatizamos que el argumento que hicimos en (5.3) para probar que Φ alcanza un mínimo en \mathbf{x}^* es válido aún en este caso.

△

Observación 5.2.2 (ortogonalidad). La condición $A^t A \mathbf{x}^* = A^t \mathbf{y}$ es equivalente a que $A \mathbf{x}^* - \mathbf{y}$ sea ortogonal al espacio de columnas de A . En efecto, las ecuaciones normales se pueden escribir como

$$A^t(A \mathbf{x}^* - \mathbf{y}) = \mathbf{0}.$$

Esto implica que, para toda fila de A^t (esto es, para toda columna $A^{(i)}$ de A), se tiene $A^{(i)} \cdot (A \mathbf{x}^* - \mathbf{y}) = 0$. \triangle

Observación 5.2.3 (mínimos cuadrados como proyección ortogonal). Supongamos que $m > n$ y que $\text{rg}(A) = n$. El espacio de columnas de A ($\text{col}(A)$) es un subespacio de \mathbb{R}^m de dimensión n . Descomponemos \mathbb{R}^m como la suma directa entre este subespacio y su complemento ortogonal,

$$\mathbb{R}^m = \text{col}(A) \oplus \text{col}(A)^\perp.$$

La observación anterior nos asegura que $\mathbf{y} - A \mathbf{x}^* \in \text{col}(A)^\perp$, mientras que es evidente que $A \mathbf{x}^* \in \text{col}(A)$. Por lo tanto, la solución de mínimos cuadrados es el único vector $\mathbf{x}^* \in \mathbb{R}^n$ que nos permite descomponer

$$\mathbf{y} = \underbrace{A \mathbf{x}^*}_{\in \text{col}(A)} + \underbrace{\mathbf{y} - A \mathbf{x}^*}_{\in \text{col}(A)^\perp}.$$

Esto implica que $A \mathbf{x}^*$ es la proyección ortogonal (ver Definición A.4.2) del vector \mathbf{y} sobre $\text{col}(A)$. \triangle

Ejemplo 5.2.1 (ajuste lineal). Consideremos los datos de la siguiente tabla:

t	y
0	-1
1	1
2	2
3	9/2

Queremos hacer un ajuste lineal a los mismos, esto es, escribir $y(t) = x_1 t + x_2$ con x_1, x_2 a determinar. Usamos las funciones de base $\phi_1(t) = t$, $\phi_2(t) = 1$, y escribimos nuestro problema como $A\mathbf{x} = \mathbf{y}$, donde

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} -1 \\ 1 \\ 2 \\ 9/2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \quad (5.4)$$

Por lo tanto, logramos el mejor ajuste en el sentido de mínimos cuadrados si resolvemos el sistema dado por las ecuaciones normales,

$$A^t A \mathbf{x} = A^t \mathbf{y} \Rightarrow \begin{bmatrix} 14 & 6 \\ 6 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 37/2 \\ 13/2 \end{bmatrix}.$$

La solución de este último sistema es $\mathbf{x}^* = [7/4, -1]^t$. La Figura 5.2 muestra los datos y el gráfico de la función lineal $y(t) = 7t/4 - 1$ que obtuvimos. \triangle

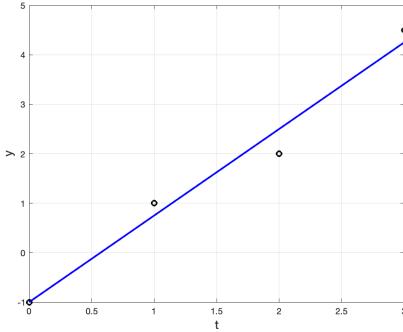


Figura 5.2: Ajuste lineal de datos.

Observación 5.2.4 (condicionamiento). Las ecuaciones normales son una herramienta poderosa para trabajar en la teoría de problemas de mínimos cuadrados. A primera vista, incluso parecen interesantes en la práctica: estamos convirtiendo el problema de resolver el sistema $A\mathbf{x} = \mathbf{y}$, con $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ en el de resolver un sistema cuadrado cuya matriz es $A^t A \in \mathcal{M}_n(\mathbb{R})$. En caso de que $m \gg n$, esto puede ser un buen beneficio respecto al tamaño del sistema resultante. Sin embargo, **las ecuaciones normales no siempre**

son una buena opción práctica, y esto se debe a que pueden presentar problemas de condicionamiento. Para sistemas pequeños, esto puede no ser demasiado problemático (a menos que tengamos un ejemplo patológico como el que mostramos a continuación).

Sin embargo, dado que los datos de entrada en general están sujetos a errores (en particular, si corresponden a observaciones empíricas o datos experimentales), en problemas de mínimos cuadrados es especialmente importante reducir la magnificación del error. Por ello, en la Sección 5.3 y el Capítulo 6 vamos a estudiar métodos para resolver problemas de mínimos cuadrados que no requieran el uso de la matriz $A^t A$. \triangle

Ejemplo 5.2.2 (condicionamiento de ecuaciones normales). Dado $\delta > 0$, consideremos

$$A = \begin{bmatrix} 1 & 1 \\ \delta & 0 \\ 0 & \delta \end{bmatrix} \in \mathcal{M}_{3 \times 2}(\mathbb{R}).$$

Es fácil verificar que

$$A^t A = \begin{bmatrix} 1 + \delta^2 & 1 \\ 1 & 1 + \delta^2 \end{bmatrix}$$

Supongamos que usamos aritmética de punto flotante con precisión doble y que $\varepsilon_M/2 \leq \delta < \sqrt{\varepsilon_M/2} \approx 10^{-8}$: en ese caso, computacionalmente A tiene rango 2 (porque sus columnas son linealmente independientes) pero $A^t A$ es singular.

Incluso si $\delta \geq \sqrt{\varepsilon_M/2}$ y el sistema no es computacionalmente singular, las ecuaciones normales pueden no ser un buen camino. Recordando la Definición 2.5.3 de número de condición y calculando

$$(A^t A)^{-1} = \frac{1}{\delta^4 + 2\delta^2} \begin{bmatrix} 1 + \delta^2 & -1 \\ -1 & 1 + \delta^2 \end{bmatrix},$$

tenemos que $\kappa(A^t A, \|\cdot\|_\infty) = \kappa(A^t A, \|\cdot\|_1) = \frac{(2+\delta^2)^2}{\delta^4+2\delta^2} \approx \frac{2}{\delta^2}$ para δ pequeño. \triangle

5.3. Descomposición QR

Aquí analizamos una alternativa para resolver problemas de mínimos cuadrados que no implique utilizar las ecuaciones normales. Consideraremos la ecuación $A\mathbf{x} = \mathbf{y}$ con $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, $m > n$. Como punto de partida, retomamos la Observación 5.2.3: hallar la solución del sistema en el sentido de los mínimos cuadrados es equivalente a computar la proyección ortogonal de \mathbf{y} sobre el espacio de columnas de A . Esto motiva a los llamados *métodos de ortogonalización* para resolver nuestro problema.

El lector debe haberse encontrado con el concepto de *matriz ortogonal* en los cursos de Geometría y Álgebra Lineal. Aquí hacemos un breve repaso de propiedades clave para nuestro análisis.

Definición 5.3.1 (matriz ortogonal). Sea $Q \in \mathcal{M}_n(\mathbb{R})$. Decimos que Q es una **matriz ortogonal** si sus columnas $\{Q^{(1)}, \dots, Q^{(n)}\}$ forman una base ortonormal de \mathbb{R}^n (recordar la Definición A.4.1). \triangle

Observación 5.3.1 (propiedades de matrices ortogonales). Mencionamos algunas propiedades que vamos a utilizar sobre matrices ortogonales.

1. Las transformaciones lineales dadas por matrices ortogonales preservan norma. Esto es, si $Q \in \mathcal{M}_n(\mathbb{R})$ es ortogonal, entonces se tiene

$$\|Q\mathbf{x}\|_2 = \|\mathbf{x}\|_2 \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

2. Si $Q \in \mathcal{M}_{m \times n}(\mathbb{R})$ con $m > n$, y las n columnas de Q forman un conjunto ortonormal en \mathbb{R}^m , entonces se pueden agregar $m - n$ vectores a las columnas de Q de modo de obtener una base ortogonal de \mathbb{R}^m . La matriz cuadrada $m \times m$ que así se obtiene es ortogonal.
3. Una matriz $Q \in \mathcal{M}_n(\mathbb{R})$ es ortogonal si y sólo si es invertible y su inversa es su traspuesta. Esto es, si y sólo si se cumple

$$Q^t Q = Q Q^t = I_n.$$

Además, si Q es ortogonal entonces Q^t también lo es.

\triangle

Teorema 5.3.1 (descomposición QR). *Dada $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ con $m \geq n$ tal que $rg(A) = n$, existen*

- una matriz $Q \in \mathcal{M}_{m \times n}(\mathbb{R})$ cuyas columnas forman un conjunto ortonormal en \mathbb{R}^m ,
- una matriz triangular superior e invertible $R \in \mathcal{M}_n(\mathbb{R})$,

tales que

$$A = QR. \tag{5.5}$$

Bosquejo de la demostración. Como A es de rango completo, las columnas de A , que denotamos por $A^{(1)}, \dots, A^{(n)}$, forman una base de $\text{col}(A)$, y podemos aplicar el Teorema A.4.1 a ellas. En particular, el proceso de ortonormalización de Gram-Schmidt produce un conjunto ortonormal $\{\mathbf{u}_1, \dots, \mathbf{u}_n\} \subset \mathbb{R}^m$ tal que $[\mathbf{u}_1, \dots, \mathbf{u}_n] = \text{col}(A)$.

Definimos $Q \in \mathcal{M}_{m \times n}(\mathbb{R})$ como la matriz que se forma “colgando” estos vectores, esto es, tal que $Q^{(j)} := \mathbf{u}_j$ para $j = 1, \dots, n$. Además, definimos $R = (r_{ij}) \in \mathcal{M}_n(\mathbb{R})$ mediante

$$r_{ij} = \begin{cases} A^{(j)} \cdot \mathbf{u}_i & \text{si } i \leq j, \\ 0 & \text{si } i > j. \end{cases}$$

Es claro que, por construcción, Q es ortogonal y R es triangular superior. Siguiendo la demostración del Teorema A.4.1 se deduce que se cumple la igualdad (5.5). \square

Observación 5.3.2 (versiones económica y completa). La factorización del Teorema 5.3.1 es la llamada *descomposición QR económica*, en la que Q es una matriz rectangular y R es cuadrada. Existe también la llamada *descomposición QR completa*, que consiste en tomar $Q \in \mathcal{M}_m(\mathbb{R})$ ortogonal y $R \in \mathcal{M}_{m \times n}(\mathbb{R})$ de modo que valga (5.5). La factorización completa se puede lograr completando las columnas de la matriz orthogonal dada por la económica de acuerdo al punto 2 en la Observación 5.3.1, y a la matriz R se le agregan ceros desde la fila $n + 1$ en adelante.

Sean Q, R las matrices dadas por una descomposición completa de una matriz $A \in \mathcal{M}_{m \times n}(\mathbb{R})$. Denotamos por $Q^{(1:n)} \in \mathcal{M}_{m \times n}(\mathbb{R})$ y $Q^{(n+1:m)} \in \mathcal{M}_{m \times (m-n)}(\mathbb{R})$ a las matrices que corresponden a tomar las primeras n y las últimas $m - n$ columnas de Q respectivamente, y por $R_{(1:n)} \in \mathcal{M}_n(\mathbb{R})$ y $R_{(n+1:m)} \in \mathcal{M}_{(m-n) \times n}(\mathbb{R})$ a las matrices que corresponden a tomar las primeras n y las últimas $m - n$ filas de R respectivamente. Como R es triangular superior, tiene que ser $R_{(n+1:m)} = \mathcal{O}_{(m-n) \times n}$, la matriz nula en $\mathcal{M}_{(m-n) \times n}(\mathbb{R})$. Por lo tanto, tenemos

$$A = QR = [Q^{(1:n)} \ Q^{(n+1:m)}] \begin{bmatrix} R_{(1:n)} \\ \mathcal{O}_{(m-n) \times n} \end{bmatrix} = Q^{(1:n)} R_{(1:n)}.$$

Deducimos, entonces, que la submatriz $Q^{(n+1:m)}$ no juega ningún papel en el producto QR, lo que muestra que la descomposición QR completa no es única. De hecho, la factorización $A = Q^{(1:n)} R_{(1:n)}$ es precisamente la descomposición QR económica de A . \triangle

En Octave, la función `qr` computa una descomposición completa de la matriz que le ingresemos como entrada. Opcionalmente, se puede ingresar el argumento ‘`econ`’, que permite computar la descomposición económica.

5.3.1. Aplicación de QR a mínimos cuadrados

Con el Teorema 5.3.1 a mano, nos preguntamos cómo se lo puede usar para resolver problemas de mínimos cuadrados. Consideramos un sistema $A\mathbf{x} = \mathbf{y}$, donde $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, $m > n$, $\text{rg}(A) = n$, y supongamos que ya tenemos una *descomposición completa* $A = QR$. En particular, como $R \in \mathcal{M}_{m \times n}(\mathbb{R})$ es triangular superior, la podemos escribir como

$$R = \begin{bmatrix} R_{(1:n)} \\ \mathcal{O}_{(m-n) \times n} \end{bmatrix},$$

donde $R_{(1:n)} \in \mathcal{M}_n(\mathbb{R})$ es triangular superior y cuadrada, y $\mathcal{O}_{(m-n) \times n}$ denota la matriz nula en $\mathcal{M}_{(m-n) \times n}(\mathbb{R})$. Dado $\mathbf{x} \in \mathbb{R}^n$, escribimos el cuadrado de la norma euclídea del residuo, usamos la factorización QR de A y el hecho de que $Q \in \mathcal{M}_m(\mathbb{R})$ es una matriz orthogonal (Observación 5.3.1),

$$\|\mathbf{r}\|_2^2 = \|A\mathbf{x} - \mathbf{y}\|_2^2 = \|QR\mathbf{x} - \mathbf{y}\|_2^2 = \|Q^t(QR\mathbf{x} - \mathbf{y})\|_2^2 = \|R\mathbf{x} - Q^t\mathbf{y}\|_2^2. \quad (5.6)$$

A continuación, dado un vector $\mathbf{z} \in \mathbb{R}^m$, lo descomponemos como

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_{(1:n)} \\ \mathbf{z}_{(n+1:m)} \end{bmatrix},$$

donde $\mathbf{z}_{(1:n)} \in \mathbb{R}^n$ y $\mathbf{z}_{(n+1:m)} \in \mathbb{R}^{m-n}$, esto es, $\mathbf{z}_{(1:n)}$ tiene las primeras n coordenadas de \mathbf{z} y $\mathbf{z}_{(n+1:m)}$ las últimas $m - n$. Es simple verificar que⁴

$$\|\mathbf{z}\|_2^2 = \|\mathbf{z}_{(1:n)}\|_2^2 + \|\mathbf{z}_{(n+1:m)}\|_2^2.$$

En particular, observemos que al aplicar esta forma de descomponer vectores nos da $(R\mathbf{x})_{(1:n)} = R_{(1:n)}\mathbf{x}$ y $(R\mathbf{x})_{(n+1:m)} = \mathcal{O}_{(m-n) \times n}\mathbf{x} = \mathbf{0}$ y por lo tanto, volviendo a (5.6), tenemos

$$\|\mathbf{r}\|_2^2 = \|R\mathbf{x} - Q^t\mathbf{y}\|_2^2 = \|R_{(1:n)}\mathbf{x} - (Q^t\mathbf{y})_{(1:n)}\|_2^2 + \|(Q^t\mathbf{y})_{(n+1:m)}\|_2^2. \quad (5.7)$$

Esto quiere decir que el cuadrado de la norma euclídea del residuo –cantidad que queremos minimizar– se puede escribir como la suma de dos términos: el primero depende del vector de parámetros \mathbf{x} , pero el segundo no. Por lo tanto, la solución del problema de mínimos cuadrados es aquel $\mathbf{x}^* \in \mathbb{R}^n$ que logre que $\|(R\mathbf{x})_{(1:n)} - (Q^t\mathbf{y})_{(1:n)}\|_2^2$ sea lo menor posible. Sin embargo, el sistema

$$R_{(1:n)}\mathbf{x} = (Q^t\mathbf{y})_{(1:n)}$$

es un sistema cuadrado y triangular superior: lo podemos resolver usando una sustitución hacia atrás (Sección 2.2.1) sin pasar por ningún problema de condicionamiento. Resolver este sistema nos da la solución $\mathbf{x}^* \in \mathbb{R}^n$.

En resumen, dada la factorización $A = QR$ de una matriz de rango completo, para hallar

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \|A\mathbf{x} - \mathbf{y}\|_2^2,$$

procedemos de la siguiente manera:

- computamos el vector $Q^t\mathbf{y}$;
- resolvemos el sistema $R\mathbf{x} = (Q^t\mathbf{y})_{(1:n)}$ usando sustitución hacia atrás para hallar la solución de mínimos cuadrados \mathbf{x}^* ;
- si nos interesa el valor mínimo de la norma euclídea del residuo, ésta es igual a $\|(Q^t\mathbf{y})_{(n+1:m)}\|_2$.

Ejemplo 5.3.1 (ajuste lineal con QR). Volvemos a hacer el Ejemplo 5.2.1, pero esta vez aplicando el método que recién describimos. Usamos el comando `[Q,R] = qr(A)` en Octave⁵ para obtener

$$Q = \begin{bmatrix} 0 & -0,8367 & -0,3858 & -0,3888 \\ -0,2673 & -0,4781 & 0,2246 & 0,8060 \\ -0,5345 & -0,1195 & 0,7082 & -0,4455 \\ -0,8018 & 0,2390 & -0,5470 & 0,0283 \end{bmatrix}, \quad R = \begin{bmatrix} -3,7417 & -1,6036 \\ 0 & -1,1952 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Por lo tanto, podemos hallar la solución de nuestro problema con los comandos

⁴Con la sutileza de que las tres normas que aparecen en la igualdad son en espacios distintos: la primera es en \mathbb{R}^m , la segunda en \mathbb{R}^n , y la tercera en \mathbb{R}^{m-n} .

⁵En la Sección 5.3.2 estudiamos cómo computar esta factorización en forma estable.

```
>> z = Q'*y;
>> x = R(1:2, 1:2) \ z(1:2);
```

La solución es $\mathbf{x} = [1.7500; -1.0000]$, que coincide con la que hallamos anteriormente. Como verificación de la fórmula (5.7), computando el residuo $\mathbf{Ax} - \mathbf{y}$ obtenemos el vector $[0,0000; -0,2500; 0,5000; -0,2500]$, cuya norma euclídea es $\sqrt{\frac{3}{8}} \approx 0,6124$. Si escribimos el comando `norm(z(3:4))`, obtenemos este mismo resultado. \triangle

Observación 5.3.3 (problemas con rango deficiente). Si $\text{rg}(A) < n$ en el Teorema 5.3.1, aún así existe la factorización QR, pero naturalmente en ese caso no podemos esperar que R sea invertible. De acuerdo a lo que comentamos en la Observación 5.2.1, un problema de mínimos cuadrados que involucre a la matriz A no tiene solución única. En la práctica, el rango de una matriz no es algo tan claro; una matriz que con aritmética real es de rango completo, no tiene por qué serlo en aritmética de punto flotante. Si un problema de mínimos cuadrados está cerca de ser de rango deficiente, en la práctica se pueden utilizar variantes al método de computar la factorización QR como realizar *pivoteo por columnas*. No vamos a tratar ese tipo de métodos en este curso, y para problemas de rango deficiente o casi de rango deficiente proponemos aplicar la factorización SVD que describiremos en el Capítulo 6. \triangle

5.3.2. Transformaciones de Householder

La sección anterior describió cómo se puede aprovechar la descomposición de QR para resolver problemas de mínimos cuadrados de rango completo en forma estable y eficiente. Una pregunta natural y que no hemos abordado aún es cómo computar la factorización QR de una matriz dada. La demostración del Teorema 5.3.1 sugiere usar el método de ortogonalización de Gram-Schmidt. Sin embargo, una aplicación directa del método tal como lo describimos en la Sección A.4 no es satisfactoria desde el punto de vista computacional: la ortogonalidad entre los vectores computados se pierde fácilmente debido a errores de redondeo. Existe una versión modificada del algoritmo que permite corregir este problema [Hea02, Sección 3.5.3], pero aquí en cambio vamos a discutir otro método de ortogonalización para resolver problemas de mínimos cuadrados y, de paso, computar factorizaciones QR. Comenzamos con una definición clave.

Definición 5.3.2 (reflexión de Householder). Dado $\mathbf{u} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$, la **reflexión (o transformación) de Householder asociada a \mathbf{u}** es una matriz $H \in \mathcal{M}_m(\mathbb{R})$ de la forma

$$H := I - \rho \mathbf{u} \mathbf{u}^t, \quad (5.8)$$

donde $\rho := 2 / \|\mathbf{u}\|_2^2$. \triangle

En forma un tanto vaga, vamos a usar la expresión *reflexión de Householder* para referirnos tanto a la transformación lineal $T : \mathbb{R}^m \rightarrow \mathbb{R}^m$ dada por $T\mathbf{x} = H\mathbf{x}$ como a su matriz

asociada H . Las reflexiones de Householder son, entonces, matrices que se pueden escribir como la resta de la matriz identidad en $\mathcal{M}_m(\mathbb{R})$ menos la matriz $2\frac{\mathbf{u}\mathbf{u}^t}{\|\mathbf{u}\|_2^2}$, que es de rango uno⁶, y que también se suele escribir como $2\frac{\mathbf{u}}{\|\mathbf{u}\|_2} \otimes \frac{\mathbf{u}}{\|\mathbf{u}\|_2}$.

Observación 5.3.4. En este punto, puede ser instructivo computar las imágenes de algunos vectores por una transformación de Householder. Sea H la transformación de Householder asociada a un cierto vector $\mathbf{u} \in \mathbb{R}^m$. Podemos calcular imágenes usando la definición (5.8). Por un lado, vectores de la forma $\mathbf{v} = \alpha\mathbf{u}$ con $\alpha \in \mathbb{R}$ se transforman en

$$H\mathbf{v} = (I - \rho\mathbf{u}\mathbf{u}^t)\mathbf{v} = \alpha\mathbf{u} - \rho\mathbf{u}\alpha\|\mathbf{u}\|_2^2 = \alpha\mathbf{u} - 2\alpha\mathbf{u} = -\mathbf{v}.$$

Por otro lado, si $\mathbf{v} \in \mathbb{R}^m$ es un vector ortogonal a \mathbf{u} , esto es, $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^t\mathbf{v} = 0$, entonces

$$H\mathbf{v} = (I - \rho\mathbf{u}\mathbf{u}^t)\mathbf{v} = \mathbf{v} - \rho\mathbf{u}\mathbf{u}^t\mathbf{v} = \mathbf{v}.$$

Por lo tanto, parece que la transformación de Householder asociada a \mathbf{u} es una simetría respecto al complemento ortogonal de \mathbf{u} : todo vector que sea colineal con \mathbf{u} es enviado a su opuesto, mientras que todo vector que sea ortogonal a \mathbf{u} permanece inafectado por H . La Figura 5.3 da una interpretación gráfica de este hecho, que a continuación demostramos con mayor rigurosidad. \triangle

Lema 5.3.2 (simetría y ortogonalidad). *Dado $\mathbf{u} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$, la matriz H definida por (5.8) es simétrica y ortogonal.*

*Demuestra*ción. Sea $H \in \mathcal{M}_n(\mathbb{R})$ dada por (5.8). Comenzamos verificando la simetría de H , y para ello notamos que $(\mathbf{u}\mathbf{u}^t)^t = (\mathbf{u}^t)^t\mathbf{u}^t = \mathbf{u}\mathbf{u}^t$ y por lo tanto

$$H^t = I^t - (\rho\mathbf{u}\mathbf{u}^t)^t = I - \rho(\mathbf{u}\mathbf{u}^t)^t = I - \rho\mathbf{u}\mathbf{u}^t = H.$$

Para probar que H es ortogonal, utilizamos la caracterización que nos da el punto 3. de la Observación 5.3.1 y computamos

$$H^t H = H H^t = H H = (I - \rho\mathbf{u}\mathbf{u}^t)(I - \rho\mathbf{u}\mathbf{u}^t) = I - 2\rho\mathbf{u}\mathbf{u}^t + \rho\mathbf{u}\mathbf{u}^t\rho\mathbf{u}\mathbf{u}^t.$$

Analicemos el último término en la fórmula anterior. Como $\rho \in \mathbb{R}$, podemos agrupar los factores en ρ , y tenemos $\rho^2 = 4/\|\mathbf{u}\|_2^4$. Además, $\mathbf{u}^t\mathbf{u} = \|\mathbf{u}\|_2^2$, por lo que este factor también es un número real,

$$\rho\mathbf{u}\mathbf{u}^t\rho\mathbf{u}\mathbf{u}^t = \rho^2\mathbf{u}(\mathbf{u}^t\mathbf{u})\mathbf{u}^t = \frac{4}{\|\mathbf{u}\|_2^4}\|\mathbf{u}\|_2^2\mathbf{u}\mathbf{u}^t = \frac{4}{\|\mathbf{u}\|_2^2}\mathbf{u}\mathbf{u}^t = 2\rho\mathbf{u}\mathbf{u}^t.$$

Concluimos entonces que H es invertible y que $H^{-1} = H^t$, por lo que es una matriz ortogonal. \square

⁶Que esta matriz es de rango uno se deduce fácilmente del hecho de que cada fila es un múltiplo de \mathbf{u}^t o de que cada columna es un múltiplo de \mathbf{u} .

Una consecuencia inmediata del lema anterior y de la Observación 5.3.1 es que si $H \in \mathcal{M}_m(\mathbb{R})$ es de Householder, entonces vale la propiedad

$$\|H\mathbf{x}\|_2 = \|\mathbf{x}\|_2 \quad \forall \mathbf{x} \in \mathbb{R}^m.$$

Dada H como en (5.8) y $\mathbf{x} \in \mathbb{R}^m$, **¿cómo calculamos $H\mathbf{x}$?** Aquí es importante recordar la noción de proyección ortogonal (Definición A.4.2); dado $\mathbf{u} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$, denotamos por $U := [\mathbf{u}]$ al subespacio de \mathbb{R}^m generado por \mathbf{u} . Entonces, tenemos

$$H\mathbf{x} = (I - \rho\mathbf{u}\mathbf{u}^t)\mathbf{x} = \mathbf{x} - 2\frac{\mathbf{u}\mathbf{u}^t\mathbf{x}}{\|\mathbf{u}\|_2^2} = \mathbf{x} - 2\frac{\langle \mathbf{u}, \mathbf{x} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle}\mathbf{u},$$

donde usamos que $\mathbf{u}^t\mathbf{x} = \langle \mathbf{u}, \mathbf{x} \rangle \in \mathbb{R}$. Luego, como $P_U(\mathbf{x}) = \frac{\langle \mathbf{u}, \mathbf{x} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle}\mathbf{u}$, encontramos la identidad

$$H\mathbf{x} = \mathbf{x} - 2P_U(\mathbf{x}).$$

La Figura 5.3 ilustra la situación: al hacer $H\mathbf{x}$ tomamos el vector \mathbf{x} , le restamos una vez $P_U(\mathbf{x})$, lo que nos llevaría a la proyección sobre U^\perp , y al resultado nuevamente le restamos $P_U(\mathbf{x})$, lo que da lugar al vector simétrico a \mathbf{x} respecto a U^\perp . Esto justifica el uso de la palabra *reflexión*: **la reflexión de Householder con vector \mathbf{u} es la simetría respecto al subespacio U^\perp .**

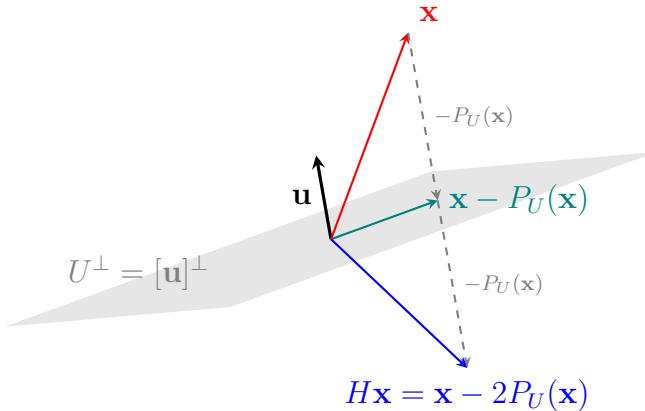


Figura 5.3: La imagen de un vector \mathbf{x} mediante una reflexión de Householder con vector \mathbf{u} es la simetría de \mathbf{x} respecto al subespacio $[\mathbf{u}]^\perp$.

Controlando la imagen de un vector dado

Las reflexiones de Householder ofrecen una alternativa computacionalmente estable para computar la factorización QR de una matriz dada A . El método de ortogonalización de Gram-Schmidt, que describimos en la demostración del Teorema A.4.1, se enfoca en ir construyendo secuencialmente las columnas de la matriz Q , y la matriz R se obtiene a partir de los coeficientes obtenidos. En cambio, al usar las transformaciones de Householder

nos vamos a enfocar en construir secuencialmente las columnas de R , y esta construcción nos permitirá obtener la matriz Q .

Comencemos con el siguiente problema: dado un vector $\mathbf{x} \in \mathbb{R}^m$, buscamos un vector $\mathbf{u} \in \mathbb{R}^m$ tal que la reflexión de Householder asociada a \mathbf{u} logre que $H\mathbf{x}$ esté en uno de los ejes coordenados. Para fijar ideas, pongamos que queremos

$$H\mathbf{x} = \alpha \mathbf{e}_k, \text{ para algún } k \in \{1, \dots, m\}, \alpha \in \mathbb{R}.$$

Como H preserva la norma euclídea, necesariamente tiene que ser $\|\mathbf{x}\|_2 = \|H\mathbf{x}\|_2 = \|\alpha \mathbf{e}_k\|_2 = |\alpha|$. Por otra parte, necesitamos que \mathbf{u} bisecte uno de los ángulos formados por $[\mathbf{x}]$ y el eje $[\mathbf{e}_k]$: veamos que basta con tomar $\mathbf{u} = \mathbf{x} + \sigma \mathbf{e}_k$, donde $\sigma = \pm \|\mathbf{x}\|_2$. En efecto, definamos $\tau := \mathbf{u}^t \mathbf{x}$, de modo que $H\mathbf{x} = \mathbf{x} - \tau \mathbf{u}$. Por un lado, calculamos

$$\|\mathbf{u}\|_2^2 = \langle \mathbf{x} + \sigma \mathbf{e}_k, \mathbf{x} + \sigma \mathbf{e}_k \rangle = \|\mathbf{x}\|_2^2 + 2\sigma \mathbf{x}_k + \sigma^2 = 2\sigma(\sigma + \mathbf{x}_k), \quad \mathbf{x}_k := \langle \mathbf{x}, \mathbf{e}_k \rangle,$$

y por lo tanto

$$\rho = \frac{2}{\|\mathbf{u}\|_2^2} = \frac{1}{\sigma(\sigma + \mathbf{x}_k)}.$$

Por otro lado,

$$\mathbf{u}^t \mathbf{x} = \langle \mathbf{x} + \sigma \mathbf{e}_k, \mathbf{x} \rangle = \|\mathbf{x}\|_2^2 + \sigma \mathbf{x}_k = \sigma(\sigma + \mathbf{x}_k).$$

Deducimos que $\tau = 1$ y, en consecuencia, $H\mathbf{x} = \mathbf{x} - \mathbf{u} = \mathbf{x} - (\mathbf{x} + \sigma \mathbf{e}_k) = -\sigma \mathbf{e}_k$.

Naturalmente, tenemos dos posibles elecciones de σ que logran que $H\mathbf{x}$ esté en el eje k -ésimo. Para reducir posibles problemas de cancelación, en la práctica se suele preferir σ de modo que $\operatorname{sgn} \sigma = \operatorname{sgn} \mathbf{x}_k$: con esto, al tomar $\mathbf{u} = \mathbf{x} + \sigma \mathbf{e}_k$, en la coordenada k -ésima tenemos dos sumandos de igual signo.

Aplicación para obtener una factorización QR

Ahora estamos en condiciones de describir cómo utilizar las reflexiones de Householder para tratar problemas de mínimos cuadrados lineales. Para fijar ideas, pensemos en que dada $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ con $m > n$, queremos hallarle una factorización QR completa. La idea es aplicar una sucesión de reflexiones de Householder para construir matrices ortogonales y simétricas $H_1, \dots, H_n \in \mathcal{M}_m(\mathbb{R})$ de forma que

$$H_n \dots H_1 A =: R \text{ sea triangular superior.} \quad (5.9)$$

Como el producto e inversa de matrices ortogonales es una matriz ortogonal, tenemos $A = QR$ con

$$Q := (H_n \dots H_1)^{-1} = H_1^{-1} \dots H_n^{-1} = H_1^t \dots H_n^t = H_1 \dots H_n.$$

Dada $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ con $m \geq n$, consideremos su primera columna $A^{(1)} \in \mathbb{R}^m$. Definimos $H_1 \in \mathcal{M}_m(\mathbb{R})$ como una reflexión de Householder que cumpla

$$H_1 A^{(1)} = \pm \|A^{(1)}\|_2 \mathbf{e}_1.$$

De acuerdo a nuestra discusión anterior, basta con que H_1 sea la transformación de Householder asociada al vector $\mathbf{u} := A^{(1)} \pm \|A^{(1)}\|_2 \mathbf{e}_1$, donde el signo del segundo sumando se suele tomar igual que el del primer elemento de $A^{(1)}$. Luego, al hacer esta primera operación, tendremos

$$A_2 := H_1 A = \left[\begin{array}{cccc} \pm\|A^{(1)}\|_2 & * & \dots & * \\ 0 & * & \dots & * \\ \vdots & * & \dots & * \\ 0 & * & \dots & * \end{array} \right] = \left[\begin{array}{c|ccc} \pm\|A^{(1)}\|_2 & * & \dots & * \\ \hline 0 & & & \\ \vdots & & & \tilde{A}_2 \\ 0 & & & \end{array} \right] \in \mathcal{M}_{m \times n}(\mathbb{R}).$$

Los * arriba denotan elementos que no necesariamente son nulos. Observemos que la primera columna de A_2 solamente tiene un elemento no nulo en su primer entrada, por lo que es la primera columna de una matriz triangular superior. Sea $\tilde{A}_2 \in \mathcal{M}_{(m-1) \times (n-1)}(\mathbb{R})$ la matriz que se obtiene al quitar la primera fila y la primera columna de A_2 .

A continuación, consideramos una transformación de Householder $\tilde{H}_2 \in \mathcal{M}_{m-1}(\mathbb{R})$ que mande la primera columna de \tilde{A}_2 en el primer vector de la base canónica de \mathbb{R}^{m-1} ,

$$\tilde{H}_2 \tilde{A}_2^{(1)} = \pm \|\tilde{A}_2^{(1)}\|_2 \mathbf{e}_1,$$

y definimos $H_2 \in \mathcal{M}_m(\mathbb{R})$ mediante

$$H_2 := \left[\begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & & & \\ \vdots & & & \tilde{H}_2 \\ 0 & & & \end{array} \right].$$

Remarcamos que \tilde{H}_2 es de Householder, pero H_2 no lo es. De todos modos, es sencillo verificar que H_2 es una matriz ortogonal y simétrica.

De este modo, logramos

$$A_3 := H_2 H_1 A = \left[\begin{array}{cc|ccc} \pm\|A^{(1)}\|_2 & * & * & \dots & * \\ 0 & \|\tilde{A}_2^{(1)}\|_2 & * & \dots & * \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & & \tilde{A}_3 \\ 0 & 0 & & & \end{array} \right],$$

con $\tilde{A}_3 \in \mathcal{M}_{(m-2) \times (n-2)}(\mathbb{R})$. Notemos que las dos primeras columnas en A_3 se corresponden a las primeras dos columnas de una matriz triangular superior. Siguiendo con este procedimiento, luego de n pasos se obtiene (5.9). Concretamente, en el paso k -ésimo,

tenemos

$$A_k := \left[\begin{array}{ccc|cc} \pm \|A^{(1)}\|_2 & \dots & * & * & \dots & * \\ 0 & \ddots & * & * & \dots & * \\ 0 & \dots & \|\tilde{A}_{k-1}^{(1)}\|_2 & * & \dots & * \\ \hline 0 & \dots & 0 & & & \\ \vdots & & \vdots & & \tilde{A}_k & \\ 0 & \dots & 0 & & & \end{array} \right],$$

y tomamos la matriz ortogonal y simétrica

$$H_k := \left[\begin{array}{c|c} I_{k-1} & \mathcal{O}_{(k-1) \times (m-k+1)} \\ \hline \mathcal{O}_{(k-1) \times (m-k+1)} & \tilde{H}_k \end{array} \right],$$

donde $\tilde{H}_k \in \mathcal{M}_{m-k+1}(\mathbb{R})$ es la transformación de Householder que manda la primera columna de \tilde{A}_k en el primer vector de la base canónica de \mathbb{R}^{m-k+1} .

Aplicación a la resolución de problemas de mínimos cuadrados lineales

En problemas de mínimos cuadrados de rango completo, podemos usar las transformaciones de Householder para resolver los sistemas en forma estable y sin necesidad de computar la factorización QR de la matriz del problema. En efecto, dada $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, si tenemos el problema $A\mathbf{x} = \mathbf{y}$ al que queremos resolver en el sentido de mínimos cuadrados, aplicamos las n reflexiones de Householder que nos llevan a (5.9) y tenemos

$$A\mathbf{x} = \mathbf{y} \Rightarrow H_n \dots H_1 A\mathbf{x} = H_n \dots H_1 \mathbf{y} =: \mathbf{z} \Rightarrow R\mathbf{x} = \mathbf{z}.$$

La matriz $R \in \mathcal{M}_{m \times n}(\mathbb{R})$ resultante es triangular superior, por lo que estamos nuevamente en una situación como en la Sección 5.3.1: para hallar \mathbf{x} , nos basta con restringirnos a las primeras n filas del sistema anterior y hacer una sustitución hacia atrás. La norma euclídea del residuo mínimo es igual a la norma euclídea del vector formado por las últimas $n - m + 1$ entradas de \mathbf{z} .

Ejemplo 5.3.2 (transformaciones de Householder). La matriz de diseño A del Ejemplo 5.2.1/5.3.1 está dada por (5.4). Consideramos su primera columna,

$$A^{(1)} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \quad \|A^{(1)}\|_2 = \sqrt{14}.$$

Por lo tanto, tomamos H_1 como la transformación de Householder asociada al vector⁷

⁷En este caso, como el primer elemento de $A^{(1)}$ es nulo, es igualmente válido tomar $\mathbf{u} := A^{(1)} - \sqrt{14}\mathbf{e}_1$.

$\mathbf{u} := A^{(1)} + \sqrt{14}\mathbf{e}_1$, esto es,

$$H_1 := I - \frac{2\mathbf{u}\mathbf{u}^t}{\|\mathbf{u}\|_2^2} = \begin{bmatrix} 0 & -0,2673 & -0,5345 & -0,8018 \\ -0,2673 & 0,9286 & -0,1429 & -0,2143 \\ -0,5345 & -0,1429 & 0,7143 & -0,4286 \\ -0,8018 & -0,2143 & -0,4286 & 0,3571 \end{bmatrix}.$$

Al hacer $H_1 A$ en Octave, obtenemos la matriz

$$A_2 = \begin{bmatrix} -3,7417 & -1,6036 \\ -0,0000 & 0,3042 \\ -0,0000 & -0,3917 \\ -0,0000 & -1,0875 \end{bmatrix}.$$

Notamos que, a menos de errores de redondeo (las entradas son del orden de 10^{-15}), la primera columna de esta matriz se corresponde a la de una matriz triangular superior.

A continuación, consideramos la matriz

$$\tilde{A}_2 := \begin{bmatrix} 0,3042 \\ -0,3917 \\ -1,0875 \end{bmatrix};$$

en este caso, esta matriz tiene una única columna. Computamos $\|\tilde{A}_2^{(1)}\|_2 = 1,1952$, consideramos la transformación de Householder \tilde{H}_2 asociada al vector $\mathbf{u} := \tilde{A}_2^{(1)} + \|\tilde{A}_2\|_2 \mathbf{e}_1$, y consideramos

$$H_2 := \left[\begin{array}{c|cccc} 1 & 0 & \dots & 0 \\ \hline 0 & & & & \\ \vdots & & \tilde{H}_2 & & \\ 0 & & & & \end{array} \right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -0,2545 & 0,3277 & 0,9099 \\ 0 & 0,3277 & 0,9144 & -0,2377 \\ 0 & 0,9099 & -0,2377 & 0,3401 \end{bmatrix}.$$

Haciendo $H_2 A_2 = H_2 H_1 A$ en Octave, obtenemos la matriz

$$R = \begin{bmatrix} -3,7417 & -1,6036 \\ -0,0000 & -1,1952 \\ -0,0000 & -0,0000 \\ -0,0000 & -0,0000 \end{bmatrix}.$$

A menos de errores de redondeo, esta es la misma matriz R que obtuvimos en el Ejemplo 5.3.1, y es un factor de la descomposición QR de A .

Por otra parte, si computamos $\mathbf{z} := H_2 H_1 \mathbf{y}$ para el vector \mathbf{y} de datos que definimos en el Ejemplo 5.2.1 y corresponde al lado derecho del sistema, tenemos

$$\mathbf{z} = \begin{bmatrix} -4,9443 \\ 1,1952 \\ -0,4347 \\ 0,4313 \end{bmatrix}.$$

Por lo tanto, para hallar la solución $\mathbf{x} \in \mathbb{R}^2$ al problema de mínimos cuadrados, nos basta con resolver el sistema triangular superior

$$\begin{bmatrix} -3,7417 & -1,6036 \\ -0,0000 & -1,1952 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -4,9443 \\ 1,1952 \end{bmatrix}.$$

Es inmediato verificar que la solución es $\mathbf{x} = \begin{bmatrix} 1,7500 \\ -1,0000 \end{bmatrix}$. Remarcamos que la obtuvimos sin necesidad de computar la factorización QR de la matriz A . \triangle

5.4. Mínimos cuadrados no lineales

Volvemos a nuestra discusión de la Sección 5.1. Como vimos en el Ejemplo 5.1.1, nos podemos encontrar con problemas de ajustar parámetros en los que la dependencia respecto a éstos es no lineal⁸. En esta sección, vamos a analizar un método para tratar este tipo de problemas.

Tenemos un conjunto de datos $\{(t_i, y_i)\}_{i=1,\dots,m} \subset \mathbb{R}^2$ y asumimos que existe una función $f: \mathbb{R} \rightarrow \mathbb{R}$, que depende de ciertos parámetros x_1, \dots, x_n , tal que

$$y_i \approx f(t_i; \mathbf{x}), \quad \text{donde } \mathbf{x} = [x_1, \dots, x_n]^t \in \mathbb{R}^n.$$

Nuestro problema consiste en elegir “el mejor” conjunto de parámetros, y nuevamente vamos a interpretar esto como un problema de minimización de la norma euclídea del vector residuo. Asumimos que la función f depende de forma tan regular de los parámetros x_1, \dots, x_n como sea necesario.

Definición 5.4.1 (residuo). Dado un vector $\mathbf{x} \in \mathbb{R}^n$, definimos el **residuo** $\mathbf{r} = \mathbf{r}(\mathbf{x}) \in \mathbb{R}^m$ como el vector tal que

$$\mathbf{r}_i := f(t_i; \mathbf{x}) - y_i, \quad i = 1, \dots, m.$$

\triangle

En analogía con la Definición 5.1.2, nos proponemos hallar

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{r}(\mathbf{x})\|_2^2. \quad (5.10)$$

Si bien en general no esperamos que $\mathbf{r}(\mathbf{x}^*) = \mathbf{0}$, razonemos como si esto pudiese ocurrir, y planteemos el método de Newton-Raphson para esta función. Comenzamos de un iterado inicial $\mathbf{x}^0 \in \mathbb{R}^n$. Luego, dado $\mathbf{x}^k \in \mathbb{R}^n$, linealizamos la función residuo en dicho vector, esto es, aproximamos

$$\mathbf{r}(\mathbf{x}) \approx \mathbf{r}(\mathbf{x}^k) + \mathbb{J}_{\mathbf{r}}(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k).$$

⁸Aquí, estamos usando la expresión “no lineal” para indicar “no necesariamente lineal”. El método que presentamos es aplicable al caso en que la dependencia respecto a algunos parámetros es lineal y respecto a otros no lo es, como en varios de los modelos que mencionamos en el Ejemplo 5.1.1.

Notemos que la función residuo $\mathbf{r}(\mathbf{x}^k)$ es la resta de un término que depende de \mathbf{x}^k y otro que no, por lo que $\mathbb{J}_{\mathbf{r}}(\mathbf{x}^k) = \mathbb{J}_{\mathbf{f}}(\mathbf{x}^k)$, donde $\mathbf{f}: \mathbb{R}^n \rightarrow \mathbb{R}^m$ es tal que

$$\mathbf{f}_i(\mathbf{x}) := f(t_i; \mathbf{x}), \quad i = 1, \dots, m.$$

Para definir los iterados, igualamos la linealización del residuo al vector nulo: tomamos \mathbf{x}^{k+1} tal que

$$\mathbf{r}(\mathbf{x}^k) + \mathbb{J}_{\mathbf{r}}(\mathbf{x}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) = \mathbf{0}.$$

En otras palabras, esto nos conduce a definir \mathbf{x}^{k+1} mediante

$$\begin{aligned} \mathbb{J}_{\mathbf{f}}(\mathbf{x}^k)\mathbf{d}^k &= -\mathbf{r}(\mathbf{x}^k), \\ \mathbf{x}^{k+1} &:= \mathbf{x}^k + \mathbf{d}^k. \end{aligned} \tag{5.11}$$

Este es el llamado **método de Gauss-Newton**. Como $\mathbf{r}: \mathbb{R}^n \rightarrow \mathbb{R}^m$, tenemos $\mathbb{J}_{\mathbf{r}}(\mathbf{x}^k) \in \mathcal{M}_{m \times n}(\mathbb{R})$ y, por lo tanto, en cada paso (5.11) implica que el incremento \mathbf{d}^k es la solución de un problema de mínimos cuadrados lineal. En otras palabras, el método de Gauss-Newton propone aproximar la solución de un problema de mínimos cuadrados no lineal mediante una sucesión de problemas de mínimos cuadrados lineales, en la que tanto las matrices como los vectores de datos se deben ir actualizando en cada paso. El Algoritmo 5.1 muestra un pseudo-código que lo implementa.

Algoritmo 5.1: Pseudo-código: Gauss-Newton.

Datos: $\{(t_i, y_i)\}_{i=1,\dots,m} \subset \mathbb{R}^2$ con $y_i \approx f(t_i; \mathbf{x})$ para $\mathbf{x} \in \mathbb{R}^n$, $\text{tol} > 0$, $\text{max_it} > 0$

Resultado: vector \mathbf{x} que satisface el criterio de parada que impongamos
 $\mathbf{x} \leftarrow \mathbf{x}^0$;

mientras [criterio de parada] $\geq \text{tol}$ & $k < \text{max_it}$ **hacer**

$\mathbf{b} \leftarrow \mathbf{y} - \mathbf{f}(\mathbf{x})$ % notar que $\mathbf{b} = -\mathbf{r}(\mathbf{x})$; $A \leftarrow \mathbb{J}_{\mathbf{f}}(\mathbf{x})$; $\mathbf{d} \leftarrow$ solución del problema de mínimos cuadrados lineal $A\mathbf{d} = \mathbf{b}$; $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{d}$; $k \leftarrow k + 1$; fin

Como en cada paso debemos resolver un problema de mínimos cuadrados lineal, podemos usar cualquiera de los métodos que analizamos en las secciones anteriores de este capítulo. Al ser un método de tipo Newton-Raphson, en caso de que el problema (5.10) tenga solución, se puede probar la convergencia si el iterado inicial está lo suficientemente cerca de dicha solución. Asimismo, se podrían considerar variantes en el espíritu de las que describimos en la Sección 4.8.3, como introducir amortiguamientos, evitar computar las jacobianas en cada paso, o ir haciendo actualizaciones de rango bajo.

Ejemplo 5.4.1 (Gauss-Newton). Consideramos los siguientes datos:

t	y
0	2
1	0.7
2	0.3
3	0.1

Queremos ajustarlos mediante un modelo de la forma

$$y \approx f(t; \mathbf{x}) = x_1 e^{tx_2},$$

con parámetros $\mathbf{x} = [x_1, x_2]^t$. Comenzamos con el iterado inicial $\mathbf{x}^0 = [1, 0]^t$, y mostramos cómo se computa \mathbf{x}^1 . Siguiendo los pasos delineados en el Algoritmo 5.1, computamos el vector

$$\mathbf{b} := -\mathbf{r}(\mathbf{x}^0) = \mathbf{y} - \mathbf{f}(\mathbf{x}) = \begin{bmatrix} 2 \\ 0,7 \\ 0,3 \\ 0,1 \end{bmatrix} - \begin{bmatrix} x_1^0 e^{t_1 x_2^0} \\ x_1^0 e^{t_2 x_2^0} \\ x_1^0 e^{t_3 x_2^0} \\ x_1^0 e^{t_4 x_2^0} \end{bmatrix} = \begin{bmatrix} 2 - 1e^0 \\ 0,7 - 1e^0 \\ 0,3 - 1e^0 \\ 0,1 - 1e^0 \end{bmatrix} = \begin{bmatrix} 1 \\ -0,3 \\ -0,7 \\ -0,9 \end{bmatrix}.$$

Por otra parte, computamos la matriz

$$A = \mathbb{J}_f(\mathbf{x}^0) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(t_1; \mathbf{x}^0) & \frac{\partial f}{\partial x_2}(t_1; \mathbf{x}^0) \\ \frac{\partial f}{\partial x_1}(t_2; \mathbf{x}^0) & \frac{\partial f}{\partial x_2}(t_2; \mathbf{x}^0) \\ \frac{\partial f}{\partial x_1}(t_3; \mathbf{x}^0) & \frac{\partial f}{\partial x_2}(t_3; \mathbf{x}^0) \\ \frac{\partial f}{\partial x_1}(t_4; \mathbf{x}^0) & \frac{\partial f}{\partial x_2}(t_4; \mathbf{x}^0) \end{bmatrix} = \begin{bmatrix} e^{t_1 x_2^0} & t_1 x_1^0 e^{t_1 x_2^0} \\ e^{t_2 x_2^0} & t_2 x_1^0 e^{t_2 x_2^0} \\ e^{t_3 x_2^0} & t_3 x_1^0 e^{t_3 x_2^0} \\ e^{t_4 x_2^0} & t_4 x_1^0 e^{t_4 x_2^0} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix},$$

y debemos resolver el problema de mínimos cuadrados $A\mathbf{d} = \mathbf{b}$. La solución de este problema es $\mathbf{d} = [0,69, -0,61]^t$, por lo que el siguiente iterado es $\mathbf{x}^1 = \mathbf{x}^0 + \mathbf{d} = [1,69, -0,61]^t$.

Implementamos en Octave el Algoritmo 5.1 con el criterio de parada $\|\mathbf{d}\|_2 < 10^{-6}$ y comenzando de $\mathbf{x}^0 = [1, 0]^t$. El algoritmo termina en 7 pasos, y computa $\mathbf{x}^k = [1,9950, -1,0095]^t$. La Figura 5.4 muestra los datos y el ajuste resultante, $f(t) = 1,9950 \cdot e^{-1,0095 t}$.

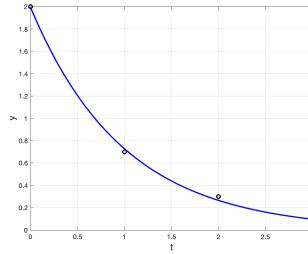


Figura 5.4: Solución mediante el método de Gauss-Newton a un problema de mínimos cuadrados no lineal.

△

Capítulo 6

Descomposición SVD

Como ya vimos en los capítulos 2 y 5, las descomposiciones matriciales juegan un papel preponderante en el álgebra lineal numérica, y permiten computar soluciones a diferentes problemas de forma eficiente y estable. En la Sección 2.3, discutimos la descomposición LU, que permite descomponer matrices no singulares, y en la Sección 5.3, la factorización QR, que usamos para descomponer matrices $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, con $m \geq n$ tales que $\text{rg}(A) = n$ (es decir, de rango completo). En caso de considerar matrices de rango deficiente (aquellas que cumplen $\text{rg}(A) < n$), debemos recurrir a otra factorización: la descomposición en valores singulares (SVD, por sus siglas en inglés).

Recordamos que el [Teorema Espectral](#) indica que cualquier matriz simétrica $S \in \mathcal{M}_n(\mathbb{R})$ puede descomponerse como $S = PDP^t$ donde D es diagonal y P es ortogonal. Por otra parte la Observación 2.5.3 afirma que dada una matriz $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, la matriz $A^t A$ es simétrica y semidefinida positiva. Por lo tanto, existen matrices $P \in \mathcal{M}_n(\mathbb{R})$ ortogonal y $D \in \mathcal{M}_n(\mathbb{R})$ diagonal tales que

$$A^t A = PDP^t.$$

Además, que $A^t A$ sea semidefinida positiva implica que sus valores propios $\lambda_1, \dots, \lambda_n$ son mayores o iguales a cero.

Definición 6.0.1 (valores singulares). Con la notación de arriba, los **valores singulares de A** son $\sigma_i := \sqrt{\lambda_i}$, $i = 1, \dots, n$. △

Observación 6.0.1 (rango y valores singulares). Si $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ tiene rango $r \leq \min\{m, n\}$, entonces como $\text{rg}(A^t A) = \text{rg}(A)$, deducimos que A tiene r valores singulares estrictamente positivos. En particular, si $A \in \mathcal{M}_n(\mathbb{R})$ es cuadrada y tiene al menos un valor singular nulo, entonces A es singular. △

El siguiente teorema, que no vamos a demostrar, indica que cualquier matriz $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ se puede descomponer como un producto entre matrices ortogonales (no necesariamente iguales) y una matriz diagonal que contiene a los valores singulares de A .

Teorema 6.0.1 (descomposición SVD). *Dada $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, con $\text{rg}(A) = r$, existen $U \in \mathcal{M}_m(\mathbb{R})$, $V \in \mathcal{M}_n(\mathbb{R})$ ortogonales y $\Sigma \in \mathcal{M}_{m \times n}(\mathbb{R})$ diagonal tales que*

$$A = U\Sigma V^t \quad (6.1)$$

y Σ es de la forma

$$\Sigma = \left[\begin{array}{c|c} \Sigma_r & \mathcal{O}_{r \times (n-r)} \\ \hline \mathcal{O}_{(m-r) \times r} & \mathcal{O}_{(m-r) \times (n-r)} \end{array} \right], \quad \text{con } \Sigma_r = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_r \end{bmatrix} \in \mathcal{M}_r(\mathbb{R}),$$

y $\sigma_1 \geq \dots \geq \sigma_r > 0$ son los valores singulares no nulos de A .

No vamos a probar la existencia de dicha factorización, ni entrar en detalles sobre cómo computarla explícitamente. El lector interesado puede consultar [GVL13, Sección 2.4] para una demostración del Teorema 6.0.1, y [QSS10, Sección 5.8.3] para una técnica que permite computar la descomposición.

Observación 6.0.2 (no unicidad). En la factorización (6.1), los elementos de Σ están únicamente determinados por A , y se los suele escribir en orden descendiente. De todos modos, aún si usamos la convención de ordenar los elementos de la diagonal de Σ , las matrices U, V en la factorización SVD no son únicas. \triangle

Interpretaremos la descomposición SVD geométricamente. Dada $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, el producto $A\mathbf{x}$ toma un vector $\mathbf{x} \in \mathbb{R}^n$ y produce un vector $\mathbf{y} = A\mathbf{x} \in \mathbb{R}^m$. Ahora, si tomamos la descomposición SVD de A , $A = U\Sigma V^t$, tenemos que $A\mathbf{x} = U\Sigma V^t \mathbf{x}$. Si nos centramos en los vectores \mathbf{x} en la esfera unitaria, esto es, con $\|\mathbf{x}\|^2 = 1$, la ortogonalidad de V implica que

$$\|V^t \mathbf{x}\|^2 = \langle V^t \mathbf{x}, V^t \mathbf{x} \rangle = (V^t \mathbf{x})^t (V^t \mathbf{x}) = \mathbf{x}^t V^t V \mathbf{x} = \mathbf{x}^t \mathbf{x} = 1.$$

Por lo tanto, para todo vector \mathbf{x} en la esfera unitaria, también se tiene que $V^t \mathbf{x}$ está en la esfera unitaria. Si llamamos $\mathbf{z} := V^t \mathbf{x}$ y le aplicamos Σ , observamos que

$$\Sigma \mathbf{z} = \begin{bmatrix} \sigma_1 & & & 0 \\ & \ddots & & \\ & & \sigma_r & 0 \\ 0 & & & \ddots & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} \sigma_1 z_1 \\ \vdots \\ \sigma_r z_r \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} x'_1 \\ \vdots \\ x'_r \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Ahora, como \mathbf{z} tiene norma 1, $\left(\frac{x'_1}{\sigma_1}\right)^2 + \dots + \left(\frac{x'_r}{\sigma_r}\right)^2 = 1$, y por lo tanto los vectores de la esfera en \mathbb{R}^n son transformados a vectores en un elipsoide r -dimensional en \mathbb{R}^m . Finalmente, como U es otra matriz ortogonal, observamos que no cambiará el tamaño

y solamente causará otro cambio de base. En resumen, si $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, entonces A aplicada a la esfera unitaria en \mathbb{R}^n resulta en un elipsoide r -dimensional rotado en \mathbb{R}^m , de semiejes $\sigma_1, \dots, \sigma_r$, y los σ_i son los valores singulares no nulos de A .

La Figura 6.1¹ ilustra esta interpretación en \mathbb{R}^2 : la matriz A transforma el círculo unitario en una elipse, y se indica cómo se transforman los vectores de la base canónica de \mathbb{R}^2 .

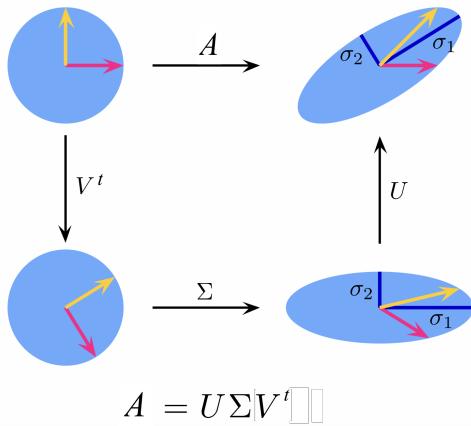


Figura 6.1: Factorización SVD.

6.1. Aplicaciones de SVD

La factorización SVD tiene numerosas aplicaciones interesantes en diversas áreas de la ingeniería, desde procesamiento de señales hasta aprendizaje automático. Aquí comentamos algunas de las más relevantes. Recomendamos el video [Mol20] para ver una perspectiva histórica y conocer otras aplicaciones.

6.1.1. Computar rangos, espacios de columnas y núcleos de matrices.

El rango de una matriz es coincide con la cantidad de valores singulares no nulos que tiene, y la factorización SVD proporciona un método excelente para computar rangos. Sin embargo, en la práctica, si queremos determinar computacionalmente el rango de una cierta matriz dada, los errores asociados a la aritmética de punto flotante hacen que esto no sea tan claro. Para calcular rangos, la función de Octave `rank` considera nulos todos aquellos valores singulares que estén por debajo de un cierto umbral.

¹Adaptada de Georg-Johann, CC BY-SA 3.0, via Wikimedia Commons.

Contar con la factorización SVD (6.1) de una cierta matriz $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ también permite determinar de forma elegante y eficiente los espacios $\text{col}(A)$ y $\ker(A)$. En efecto, si $\text{rg}(A) = r$, entonces el espacio de columnas de A (o equivalentemente, la imagen de A) está generado por las primeras r columnas de U ,

$$\text{col}(A) = [U^{(1)}, \dots, U^{(r)}],$$

e incluso ese conjunto nos da una base ortonormal de $\text{col}(A)$. Por otra parte, el núcleo de A está dado por las últimas $n - r$ columnas de V ,

$$\ker(A) = [V^{(n+1-r)}, \dots, V^{(n)}],$$

y este conjunto es una base ortonormal de $\ker(A)$. Las funciones de Octave `orth` y `null` usan la factorización SVD para devolver respectivamente estas bases ortonormales del espacio de columnas y del núcleo de A .

6.1.2. Aproximaciones de rango bajo.

Otra forma de escribir la factorización (6.1) es como una suma,

$$A = U\Sigma V^t = \sigma_1 E_1 + \dots + \sigma_n E_n, \quad (6.2)$$

donde $E_i = U^{(i)}(V^{(i)})^t \in \mathcal{M}_{m \times n}(\mathbb{R})$ es una matriz de rango 1. La expresión (6.2) permite realizar *aproximaciones de rango bajo* de la matriz A al tomar algunos de los sumandos. De hecho, teniendo en cuenta nuestra convención de ordenar los valores singulares en forma decreciente, esto produce una mejor aproximación en el siguiente sentido.

Teorema 6.1.1 (Eckhart-Young). *Dados $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ y $k < r := \text{rg}(A)$, sea $A_k := \sum_{i=1}^k \sigma_i E_i$ definida a partir de tomar los primeros k sumandos en (6.2). Entonces,*

$$\min_{B \in \mathcal{M}_{m \times n}(\mathbb{R}), \text{rg}(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1},$$

donde $\|\cdot\|_2$ denota la norma operador en $\mathcal{M}_{m \times n}(\mathbb{R})$ asociada a la norma euclídea (Definición 2.5.1).

Este teorema tiene diversas aplicaciones. Como primer ejemplo, consideremos la compresión de imágenes. Digitalmente, una imagen en blanco y negro se suele representar como una matriz de dimensiones $m \times n$. Cada entrada de dicha matriz representa un *píxel*, y suele tomar valores entre 0 (negro) y 255 (blanco). La figura 6.2 muestra un ejemplo simple de representación matricial de una imagen. El Teorema 6.1.1 nos brinda una forma de *comprimir* nuestra imagen al hacer aproximaciones de rango bajo. La Figura 6.3 nos muestra un ejemplo.

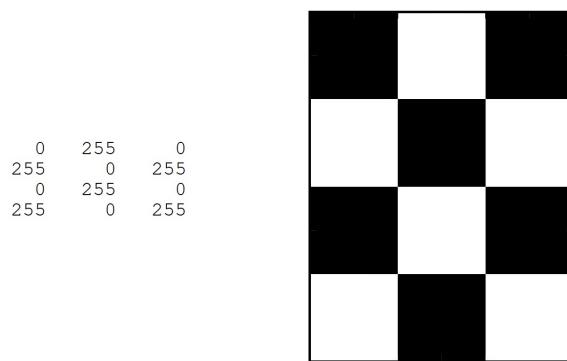
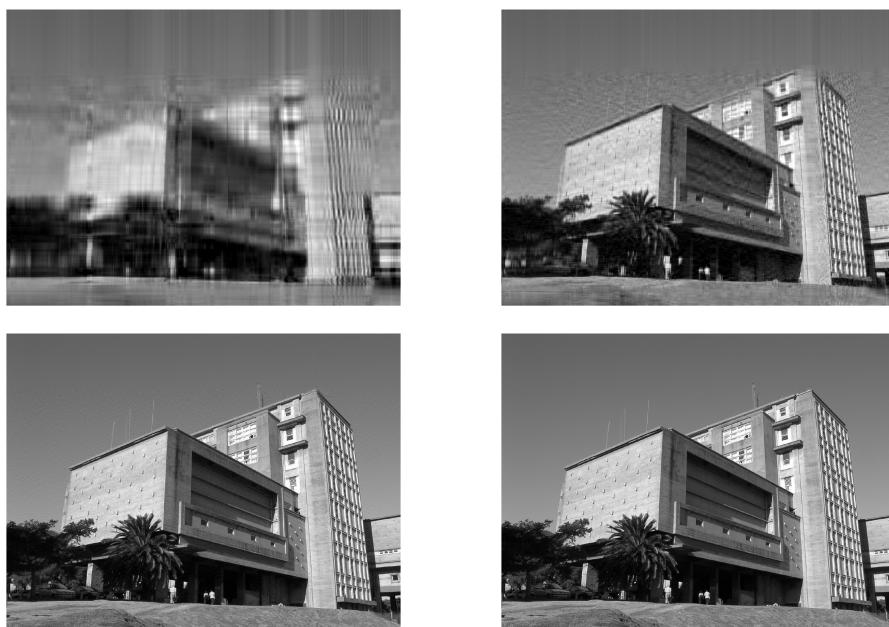


Figura 6.2: Representación matricial de una imagen en blanco y negro.

Figura 6.3: Compresión de una imagen: la original (1440×1920 píxeles) está abajo a la derecha. Mostramos las aproximaciones dadas por el Teorema 6.1.1 de rangos $k = 10$ (arriba izquierda), $k = 50$ (arriba derecha) y $k = 200$ (abajo izquierda).

6.1.3. Análisis de componentes principales.

El [Análisis de componentes principales](#) (o *Principal Component Analysis*) es una técnica estadística para reducir la dimensionalidad de un conjunto de datos; hoy en día encuentra numerosas aplicaciones en ciencia de datos y en *machine learning*, por ejemplo en visualización de datos, *clustering*, o detección de anomalías. No vamos profundizar sobre esta técnica; brevemente, indicamos que permite describir dicho conjunto de datos en función de nuevas variables no correlacionadas, llamadas *componentes principales*. Estas

componentes principales son un conjunto de k vectores unitarios, donde el vector i -ésimo se toma como el que produce el mejor ajuste a los datos entre todos los vectores que son ortogonales al espacio generado por los primeros $i - 1$ vectores. En términos de la SVD, si tenemos una matriz $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, a la que factorizamos $A = U\Sigma V^t$, sus componentes principales son las columnas de V .

El análisis de componentes principales se utiliza para crear modelos predictivos, o para reducir la dimensionalidad de los datos: esto se logra proyectando cada dato solamente en las primeras componentes principales, y permite obtener un conjunto de datos de dimensiones inferiores que “mejor aproxima” al conjunto de datos original. Ilustramos sus capacidades con un ejemplo muy sencillo.

Ejemplo 6.1.1 (análisis de componentes principales). La siguiente tabla muestra la altura, peso, y perímetro cefálico de un conjunto de 10 bebés de 1 año.

# Bebé	Peso (kg)	Altura (cm)	Per. cefálico (cm)
1	10.91	79.2	48.7
2	9.94	77.5	45.9
3	9.34	72.4	44.2
4	10.11	76.0	45.8
5	9.59	76.7	45.3
6	9.67	73.5	44.9
7	11.58	78.9	48.2
8	9.94	75.9	43.8
9	10.39	75.4	47.5
10	8.84	72.7	42.8

Visualizar estos datos como puntos en \mathbb{R}^3 no necesariamente sería de mucha ayuda para interpretarlos. Parece claro que las tres mediciones deben estar correlacionadas de alguna forma: debería haber una componente, llamémosla el *tamaño*, que permita explicar buena parte de las diferencias entre el peso, altura y perímetro cefálico de cada bebé. Colocamos nuestros datos en una matriz, la centramos y hacemos que tenga varianza 1 en cada fila² para obtener una matriz $A \in \mathcal{M}_{3 \times 10}(\mathbb{R})$, a la que le aplicamos la factorización SVD, $A = U\Sigma V^t$. Obtenemos los valores singulares

$$\sigma_1 = 4,9085, \quad \sigma_2 = 1,4182, \quad \sigma_3 = 0,9463.$$

Como σ_1 es bastante mayor que los otros dos valores singulares, parece que buena parte de la variabilidad de nuestros datos puede ser explicada por un solo factor (el *tamaño*). Tomamos esta componente principal, es decir, la primer columna de V , y, para facilitar la interpretación de los resultados, la normalizamos para que tenga norma infinito igual a 1. Obtenemos el siguiente vector:

$$\text{tamaño} = [0,90, 0,14, -0,67, 0,05, -0,09, -0,40, 1,00, -0,24, 0,27, -0,95].$$

²Este paso es para evitar las distorsiones que nos podría producir el hecho de que distintas mediciones estén tomadas en unidades distintas y son de magnitudes distintas.

Así, parece ser que los bebés número 7 y 1 son los de mayor tamaño, que el último es el que tiene menor tamaño, y si queremos ver a un bebé de tamaño “medio”, deberíamos buscar al número 4 o al 5³. Si queremos hacer un modelo explicativo usando solamente la variable tamaño, consideraremos la aproximación de rango 1 de nuestra matriz de datos,

$$E_1 = \sigma_1 U^{(1)}(V^{(1)})^t.$$

La Figura 6.4 muestra las mediciones junto a las columnas correspondientes de la matriz E_1 , que nos indica los valores predichos de acuerdo al tamaño de cada bebé. En azul se muestran las mediciones y en rojo los valores predichos por el tamaño de cada bebé. Por ejemplo, vemos que para nuestro modelo el bebé número 7 pesa más y mide menos de lo que predice su tamaño, o que el bebé 9 es un poco cabezón para su tamaño.

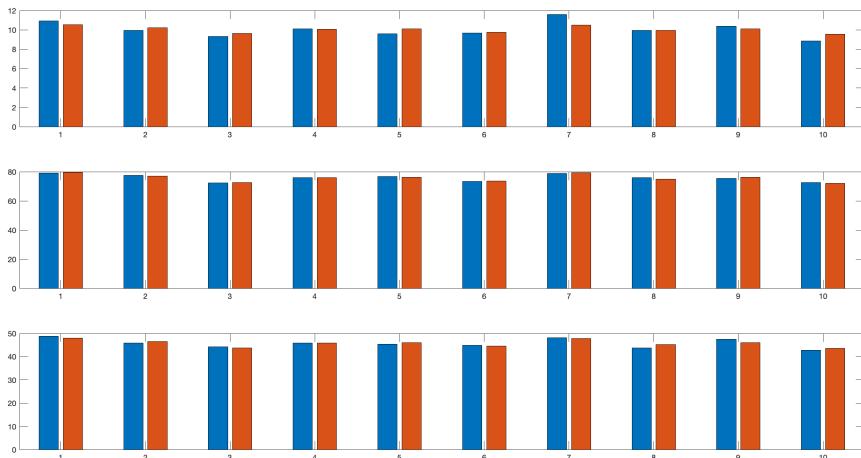


Figura 6.4: Datos del Ejemplo 6.1.1. En azul, se representan las mediciones de cada bebé (de arriba a abajo: peso, altura, perímetro cefálico), y en anaranjado los valores predichos de acuerdo al tamaño computado de cada bebé.

Además, podemos aprovechar nuestro análisis para predecir nuevos valores a partir de los datos que obtuvimos: si conocemos una medida de un bebé, por ejemplo su peso, podemos calcular su tamaño y luego predecir las otras mediciones. Por ejemplo, si tenemos un bebé que pesa 10 kg, luego de restar la media de la muestra y normalizar obtenemos que tiene un peso normalizado de $-0,0401$. Esta cantidad $-0,0401$ sería la primera entrada en los datos, y como estamos haciendo una aproximación de rango 1, conocer una entrada de una columna es suficiente para extraer las demás entradas. Así, predecimos que el tamaño de nuestro bebé es

$$\text{tamaño estimado} = -0,0401/U(1, 1) = -0,0681.$$

³Para nuestro conjunto de datos, el peso medio es de 10.03 kg, la altura media es de 75.8 cm, y el perímetro cefálico medio es de 45.7 cm.

Luego, podemos “completar” los datos faltantes (en unidades normalizadas) haciendo

$$\begin{aligned} \text{altura estimada} &= -0,0681 * U(2, 1) = -0,0384, \\ \text{per. cefálico estimado} &= -0,0681 * U(3, 1) = -0,0395, \end{aligned}$$

lo que, al deshacer la normalización nos devuelve una altura estimada de 75.73 cm y un perímetro cefálico estimado de 45.64 cm. \triangle

6.1.4. SVD en sistemas de recomendación

Desde hace algunos años, los sistemas de recomendación forman parte de nuestra vida cotidiana: los encontramos, por ejemplo, en plataformas de *streaming* o en tiendas *online*. Como caso ilustrativo, supongamos que tenemos una plataforma de películas, y almacenamos las calificaciones de nuestros m usuarios a n películas distintas. Las calificaciones pueden ser de 1 a 5, y son 0 si el usuario aún no vio una película. Una forma sencilla de modelar esto es generar una matriz $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, en la que la entrada a_{ij} nos indica qué puntaje le asigna el usuario i a la película j .

Como dueños de la plataforma tenemos dos posibilidades claras para usar estos datos, dependiendo de nuestro modelo de negocio:

- Recomendar al usuario. Dado un usuario, determinar qué película que no vio le podría gustar más.
- Vender una película (o sistemas *pay per view*). Dada una película, determinar a qué usuario que no la vio le podría gustar.

Centrémonos en el primer escenario, la recomendación a usuarios, donde la principal herramienta son los datos históricos de valoraciones. Usando éstos deseamos generar nuestras nuevas recomendaciones. Esto se conoce como **filtrado colaborativo**.

Dado que algunos usuarios son generosos con las valoraciones y otros estrictos, comenzamos ajustando el sesgo: podemos estudiar la desviación del promedio de valoraciones por usuario en vez de los puntajes asignados por éstos. La matriz normalizada que se obtiene permite entender más claramente cuánto le gustó (o disgustó) una película a un usuario, tomando en cuenta su sesgo personal. A partir de esta matriz normalizada \tilde{A} , describimos cómo implementar un filtrado colaborativo básico.

Consideremos a los valores de las filas i y j de la matriz \tilde{A} como vectores $u_i, u_j \in \mathbb{R}^n$, que corresponden a las valoraciones normalizadas de los usuarios i y j , respectivamente. Si queremos encontrar usuarios con gustos similares usando ideas geométricas, una forma sencilla sería estudiar la distancia entre vectores: cuanto más cercanos dos vectores, más similares serán las valoraciones asignadas por los usuarios. El problema con esta idea radica en que, si un usuario vio una película y el otro no, la distancia entre ellos será grande aunque sus puntuaciones en las demás películas coincidan. Una forma de mitigar

este problema sería considerar únicamente las películas que ambos puntuaron. Como queremos un sistema de recomendación, esto no parece una alternativa razonable.

Para remediar esto, podemos usar la regla del coseno en lugar de la distancia. Definimos la similaridad s entre los usuarios i y j como

$$s(i, j) := \frac{u_i^t u_j}{\|u_i\| \times \|u_j\|},$$

donde u_i y u_j son los vectores de puntuajes de los usuarios i y j respectivamente. Con esta definición, dos usuarios estarán “cerca” en gustos si los puntuajes normalizados de las películas que fueron vistas por ambos son similares. Para predecir si a un usuario i le gustará una película p , podemos encontrar los k usuarios más similares a i que valoraron la película p , y computar el promedio (ponderando con respecto a la similaridad entre ellos e i) de sus valoraciones de p . Finalmente, para ver qué película recomendar al usuario i , podemos identificar las películas que aún no ha puntuado, predecir los puntuajes de esta manera y luego recomendar aquellas con puntuajes predichos más altos.

Este filtrado colaborativo básico, si bien intuitivo, tiene la desventaja de escalar muy mal con sistemas de gran tamaño: si tenemos millones de usuarios y miles de películas, encontrar qué películas recomendar resulta demasiado costoso. Sin embargo, existe una forma sencilla y más eficiente de atacar el problema: usar la descomposición SVD. Ilustramos cómo puede usarse con una implementación concreta.

Dado un conjunto de datos con m usuarios y n películas, comenzamos creando la matriz A de valoraciones como antes, donde el lugar ij de la matriz corresponde con el puntuaje que el usuario i le otorga a la película j . Para este ejemplo, supongamos que tenemos la siguiente matriz:

```
%%Crear la matriz de valoraciones
% Filas = usuarios, Columnas = ítems
% Valoraciones de 1-5, 0 = valoración faltante
matriz_valoraciones = [
    5 4 0 1 3 2 0 1 0 3; % Usuario 1
    4 0 0 1 0 0 2 0 0 0; % Usuario 2
    1 1 0 5 1 1 0 2 0 4; % Usuario 3
    2 0 0 4 3 5 0 0 1 3; % Usuario 4
    1 1 5 0 0 2 0 0 0 0; % Usuario 5
    2 1 5 4 0 0 0 0 0 0; % Usuario 6
    0 1 3 0 1 1 3 2 4 3; % Usuario 7
    0 5 5 3 4 3 2 0 0 2; % Usuario 8
];
% Guardamos variables auxiliares
[num_usuarios, num_items] = size(matriz_valoraciones);
```

Procedemos ahora como antes, normalizando las valoraciones de acuerdo al puntaje promedio asignado por cada usuario.

```
%% PASO 2: Normalizar las valoraciones
% Calculamos el promedio de cada usuario (ignorando los 0s)
promedios_usuarios = sum(matriz_valoraciones, 2) ./...
    sum(matriz_valoraciones ~= 0, 2);

% Centramos los datos restando el promedio de cada usuario
matriz_normalizada = matriz_valoraciones - promedios_usuarios .*...
    (matriz_valoraciones ~= 0);
```

Ahora estamos en condiciones de aplicar una reducción de dimensionalidad usando la descomposición SVD. Para esto, fijamos un $k > 0$, que serán las componentes principales a preservar, y truncamos las matrices de forma que la información preservada es únicamente la relacionada a las k componentes principales como en la Subsección 6.1.2.

```
%% PASO 3: Descomposición SVD
% Realizamos la descomposición en valores singulares
[U, S, V] = svd(matriz_normalizada);

% Número de componentes principales a conservar
k = 2;

% Truncamos las matrices para quedarnos con las k componentes principales
U_k = U(:, 1:k); % Componentes latentes de usuarios
S_k = S(1:k, 1:k); % Valores singulares (matriz diagonal)
V_k = V(:, 1:k); % Componentes latentes de películas

disp('Valores singulares principales:');
disp(diag(S_k));
```

La salida en nuestro ejemplo resulta:

```
>>Valores singulares principales:
>> 6.1306 4.3081
```

Una vez aplicada la reducción de dimensionalidad, reconstruimos la matriz de valoraciones reducida. Aquí tenemos que asegurarnos que los puntajes estarán entre 1 y 5.

```

%% PASO 4: Reconstruir la matriz de valoraciones
% Reconstruimos la matriz con la aproximación de rango reducido
matriz_reconstruida = U_k * S_k * V_k';

% Volvemos a la escala original sumando los promedios de los usuarios
matriz_predicha = matriz_reconstruida + promedios_usuarios;

% Aseguramos que las predicciones estén en el rango [1,5]
matriz_predicha = max(min(matriz_predicha, 5), 1);

disp('Matriz de valoraciones predichas:');
disp(matriz_predicha);

```

Observamos que la construcción de la matriz predicha se hace en $O(1)$ operaciones, dado que k está fijo. Ahora, por ejemplo, podemos comparar la primera fila de la matriz A original con la primera fila de la matriz predicha:

```

%Usuario 1 en la matriz original
5 4 0 1 3 2 0 1 0 3;
%Predicción del modelo para el usuario 1
>> 4.6528 3.8528 2.3187 1.0000 2.9755 1.9860 2.6238 2.1256 3.4945

```

Notamos que con solamente usar dos valores principales, la primer fila se ya se reconstruye razonablemente bien en los casos donde el usuario asignó valoraciones. Ahora que tenemos nuestras predicciones, procedemos a generar recomendaciones para los usuarios. Recomendaremos las películas que predecimos puntuarán más alto y que cada usuario no miró.

```

%% PASO 5: Generar recomendaciones para cada usuario
for usuario = 1:num_usuarios
    % Encontrar ítems no valorados por este usuario
    items_no_valorados = find(matriz_valoraciones(usuario, :) == 0);

    if ~isempty(items_no_valorados)
        % Ordenar los ítems no valorados (de mayor a menor)
        valores_predichos = matriz_predicha(usuario, items_no_valorados);
        [ratings_ordenados, indices] = sort(valores_predichos, 'descend');
        items_recomendados = items_no_valorados(indices);

        %Mostrar las 3 mejores recomendaciones (o menos si no hay suficientes)
        fprintf('\nRecomendaciones para el usuario %d:\n', usuario);
        for i = 1:min(3, length(items_recomendados))
            fprintf('Ítem %d: rating predicho %.2f\n', ...
                    items_recomendados(i), ratings_ordenados(i));
        end
    end
end

```

```

    end
else
    fprintf ('\nEl usuario %d ya ha valorado todos los ítems\n', usuario);
end
end

```

Como ejemplo de salida, obtenemos

>>Recomendaciones para el usuario 1:

Ítem 9: rating predicho 3.49

Ítem 7: rating predicho 2.62

Ítem 3: rating predicho 2.32

Recomendaciones para el usuario 2:

Ítem 2: rating predicho 2.98

Ítem 9: rating predicho 2.73

Ítem 5: rating predicho 2.49

y así sucesivamente.

De esta forma, luego de obtener la factorización SVD, el resto de los pasos se pueden calcular a orden constante. Esta es la principal ventaja de del acercamiento a través de la descomposición en valores singulares: escala mejor con respecto a la cantidad de datos, y comparado al filtrado colaborativo (FC) básico que descibimos antes, resulta más eficiente si tenemos una base de usuarios y películas grande. Resumimos las diferencias entre ambos enfoques en la siguiente tabla.

	FC Básico	SVD
Complejidad de Preprocesamiento	$O(m^2n)$	$O(\min(mn^2, m^2n))$
Complejidad de Predicción	$O(m \log m)$	$O(k)$
Complejidad de Recomendación	$O(mn \log n)$	$O(mn \log n)$
Almacenamiento Requerido	$O(m^2 + mn)$	$O(k(m + n))$
Escalabilidad	Limitada (por m^2)	Alta (con k chico)
Mejor caso de uso	Pocos datos	Muchos datos

Cuadro 6.1: Comparación entre el algoritmo FC básico y el algoritmo que aplica SVD.

El problema de usar datos para sistemas de recomendación de forma eficiente es un área de investigación activa. En su momento, por hacer esto lo suficientemente bien, Netflix pagó un [premio de un millón de dólares⁴.](#)

⁴Una explicación amigable del algoritmo ganador, que explica otras ventajas prácticas de usar algoritmos de factorización matricial relacionados con la SVD, se puede encontrar en [este artículo](#).

6.2. Aplicación de SVD a mínimos cuadrados

En esta sección, revisitamos el problema estudiado en el capítulo anterior, y aplicamos la factorización SVD para resolver problemas de mínimos cuadrados. Dividimos esta discusión en dos partes: primero tratamos problemas de rango completo, y luego de rango deficiente.

6.2.1. Mínimos cuadrados de rango completo

Dado el problema

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \|A\mathbf{x} - \mathbf{y}\|_2^2,$$

con $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, $m \geq n$ y $\text{rg}(A) = n$, podemos considerar la descomposición SVD (6.1) de A , y escribir el problema como

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \|U\Sigma V^t \mathbf{x} - \mathbf{y}\|_2^2 = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \|\Sigma V^t \mathbf{x} - U^t \mathbf{y}\|_2^2.$$

Definimos el vector auxiliar $\mathbf{z} := U^t \mathbf{y}$, y comenzamos resolviendo el problema de mínimos cuadrados

$$\mathbf{w}^* = \arg \min_{\mathbf{w} \in \mathbb{R}^n} \|\Sigma \mathbf{w} - z\|_2^2.$$

Este es un problema extremadamente simple, ya que Σ es una matriz diagonal: basta con restringirnos a los primeros n elementos de \mathbf{z} y dividirlos por los valores singulares de A . Una vez que hallamos $\mathbf{w}^* := V^t \mathbf{x}^*$, la solución de nuestro problema es $\mathbf{x}^* = V \mathbf{w}^*$.

La justificación de este procedimiento es análoga a la que hicimos en la Sección 5.3.1. En efecto, usando $A = U\Sigma V^t$ podemos escribir la norma euclídea al cuadrado del residuo en un vector $\mathbf{x} \in \mathbb{R}^n$ como

$$\|\mathbf{r}\|_2^2 = \|A\mathbf{x} - \mathbf{y}\|_2^2 = \|U\Sigma V^t \mathbf{x} - \mathbf{y}\|_2^2 = \|\Sigma V^t \mathbf{x} - U^t \mathbf{y}\|_2^2,$$

donde en el último paso usamos que U es ortogonal. Como $\Sigma \in \mathcal{M}_{m \times n}(\mathbb{R})$ es diagonal, tenemos que sus últimas $m - n$ filas son todas nulas, por lo que

$$\|\mathbf{r}\|_2^2 = \|\Sigma_{(1:n)} V^t \mathbf{x} - \mathbf{z}_{(1:n)}\|_2^2 + \|\mathbf{z}_{(n+1:m)}\|_2^2. \quad (6.3)$$

Así, la solución al problema de mínimos cuadrados es el único⁵ vector $\mathbf{x}^* \in \mathbb{R}^n$ que hace que la primera norma del lado derecho se anule.

Ejemplo 6.2.1 (ajuste lineal con SVD). Volvemos al Ejemplo 5.2.1/5.3.1, al que esta vez tratamos mediante la factorización SVD de A .

⁵Como $\text{rg}(A) = n$, sus valores singulares son todos positivos y por lo tanto $\Sigma_{(1:n)}$ es invertible.

Usamos el comando `[U,S,V] = svd(A)` en Octave, y obtenemos

$$U = \begin{bmatrix} -0,1035 & -0,8302 & -0,3858 & -0,3888 \\ -0,3243 & -0,4414 & 0,2246 & 0,8060 \\ -0,5452 & -0,0525 & 0,7082 & -0,4455 \\ -0,7661 & 0,3364 & -0,5470 & 0,0283 \end{bmatrix}, \quad S = \begin{bmatrix} 4,1000 & 0 \\ 0 & 1,0908 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

$$V = \begin{bmatrix} -0,9056 & 0,4242 \\ -0,4242 & -0,9056 \end{bmatrix}.$$

Hallamos la solución al problema de mínimos cuadrados con los comandos

```
>> z = U'*y;
>> w = S(1:2, 1:2) \ z(1:2);
>> x = V*w;
```

Nuevamente encontramos la solución $\mathbf{x} = [1.7500; -1.0000]$. △

6.2.2. Mínimos cuadrados lineales con rango deficiente

A continuación, analizamos una nueva aplicación de la descomposición SVD: problemas de mínimos cuadrados lineales con rango deficiente. Concretamente, analizamos el problema de mínimos cuadrados lineal $\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \|A\mathbf{x} - \mathbf{y}\|_2^2$, donde $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ con $m > n$ y $r := \text{rg}(A) < n$. Remarcamos que, incluso en este caso, el Teorema 5.2.1 permanece válido: la dificultad está en que el problema de minimizar la norma euclídea del residuo no tiene solución única. Nuestro objetivo es encontrar un criterio significativo que nos permita *seleccionar* una de esas soluciones.

Comencemos con una perspectiva sobre casos más simples: si A fuese una matriz cuadrada e invertible, entonces la solución al problema de mínimos cuadrados sería $\mathbf{x} = A^{-1}\mathbf{y}$. En cambio, si A fuese rectangular pero de rango completo, podríamos hallar la solución al problema de mínimos cuadrados a partir de las ecuaciones normales, esto es, $\mathbf{x} = (A^t A)^{-1} A^t \mathbf{y}$. En este último caso no podemos invertir A , pero sí podemos considerar la matriz $(A^t A)^{-1} A^t$ como un reemplazo “razonable” para la inversa de A . A tal *reemplazo razonable* le vamos a llamar **pseudoinversa** de A . Notemos que si $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ con $m > n$ es de rango completo, entonces

$$[(A^t A)^{-1} A^t] A = (A^t A)^{-1} (A^t A) = I_n,$$

por lo que $(A^t A)^{-1} A^t$ es una *inversa a izquierda* de A . Sin embargo, *esta matriz en general no es una inversa a derecha*, pues

$$A[(A^t A)^{-1} A^t] \neq I_m.$$

Esto se deduce del hecho de que la matriz de la izquierda tiene rango menor o igual que n , mientras que la de la derecha tiene rango igual a $m > n$.

Para generalizar la noción de inversa de una matriz, comenzamos con el caso escalar: dado $x \in \mathbb{R} \setminus \{0\}$, su inverso respecto al producto es $\frac{1}{x}$. En caso de que $x = 0$, obviamente no tiene inverso. Vamos a llamar **pseudoinverso** de x al número

$$x^+ := \begin{cases} 1/x & \text{si } x \neq 0, \\ 0 & \text{si } x = 0. \end{cases}$$

En caso de que tengamos una matriz D diagonal (y posiblemente rectangular), vamos a llamar su **pseudoinversa** a la matriz D^+ que se obtiene al trasponer D y tomar el pseudoinverso escalar de los elementos de la diagonal. Esto es, si

$$D = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \ddots & \vdots \\ \vdots & 0 & \ddots & \vdots \\ \vdots & \vdots & 0 & d_n \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 0 \end{bmatrix} \in \mathcal{M}_{m \times n}(\mathbb{R}) \Rightarrow D^+ = \begin{bmatrix} d_1^+ & 0 & \dots & \dots & \dots & 0 \\ 0 & d_2^+ & 0 & \dots & \dots & \vdots \\ \vdots & \ddots & \ddots & 0 & \dots & \vdots \\ 0 & \dots & \dots & d_n^+ & \dots & 0 \end{bmatrix} \in \mathcal{M}_{n \times m}(\mathbb{R}).$$

Definición 6.2.1 (pseudoinversa⁶). Dada $A \in \mathcal{M}_{m \times n}(\mathbb{R})$, consideremos su descomposición SVD, $A = U\Sigma V^t$. La **pseudoinversa** (**o pseudoinversa de Moore-Penrose**) de A es la matriz $A^+ \in \mathcal{M}_{n \times m}(\mathbb{R})$ dada por

$$A^+ := V\Sigma^+U^t.$$

△

Observación 6.2.1 (no dependencia de SVD). Se puede demostrar que la Definición 6.2.1 no depende de la factorización SVD de la matriz A . △

Esta definición nos permite recuperar los casos conocidos en que la matriz es invertible o en problemas de mínimos cuadrados con rango completo.

Proposición 6.2.1 (pseudoinversa en casos conocidos). *Si $A \in \mathcal{M}_n(\mathbb{R})$ es invertible, entonces $A^+ = A^{-1}$. Más en general, si $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ con $m \geq n$ cumple $\text{rg}(A) = n$, entonces*

$$A^+ = (A^t A)^{-1} A^t.$$

Demostración. Nos basta con probar la segunda afirmación, pues la primera se deduce de ella y del hecho de que si $A \in \mathcal{M}_n(\mathbb{R})$ es invertible, entonces A^t también lo es y $(A^t)^{-1} = (A^{-1})^t$.

⁶Esta no es la definición más habitual de pseudoinversa. Se la suele definir como la única matriz que satisface las llamadas *condiciones de Moore-Penrose*; referimos a [GVL13, Sección 5.5.2] para más detalles.

Consideremos una descomposición SVD de A , $A = U\Sigma V^t$. Como $A^t = (U\Sigma V^t)^t = V\Sigma^t U^t$ y U es ortogonal, tenemos

$$A^t A = V\Sigma^t \Sigma V^t.$$

Sean $\sigma_1, \dots, \sigma_n$ los valores singulares de A (Definición 6.0.1); como $A^t A$ es invertible, estos valores singulares son estrictamente positivos. Observamos que $\Sigma_* := \Sigma^t \Sigma \in \mathcal{M}_n(\mathbb{R})$ es una matriz diagonal, y que los elementos en su diagonal son $\sigma_1^2, \dots, \sigma_n^2$. Por lo tanto,

$$(A^t A)^{-1} = (V\Sigma_* V^t)^{-1} = V\Sigma_*^{-1} V^t \Rightarrow (A^t A)^{-1} A^t = V\Sigma_*^{-1} \Sigma^t U^t.$$

Como $A^+ = V\Sigma^+ U^t$, para terminar la demostración nos basta con verificar que $\Sigma^+ = \Sigma_*^{-1} \Sigma^t$. Las dos matrices tienen el mismo tamaño ($n \times m$) y son diagonales, por lo que basta con observar que las dos tienen los mismos elementos en la diagonal. Esto es sencillo de comprobar, pues la diagonal de Σ^+ tiene los pseudoinversos escalares de los valores singulares de A , y como éstos son todos positivos, tenemos $\sigma_i^+ = \sigma_i^{-1}$ para $i = 1, \dots, n$. Por otra parte, la diagonal de $\Sigma_*^{-1} \Sigma^t$ tiene los elementos $\frac{1}{\sigma_i^2} \sigma_i = \sigma_i^{-1}$ para $i = 1, \dots, n$. \square

La proposición anterior nos indica que, si tenemos un problema de mínimos cuadrados de rango completo $\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{Ax} - \mathbf{y}\|_2^2$, entonces su solución se puede escribir formalmente como $\mathbf{x}^* = A^+ \mathbf{y}$. Notamos que esta expresión *incluso tiene sentido para problemas que no son de rango completo*, por lo que la vamos a usar como definición de solución en dicho caso. En esos problemas, en los que no tenemos unicidad de soluciones, computar $A^+ \mathbf{y}$ nos da la solución que tiene la menor norma euclídea.

Proposición 6.2.2 (mínimos cuadrados con pseudoinversa). *Sean $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ e $\mathbf{y} \in \mathbb{R}^m$. Entonces, el vector $\mathbf{x}^* := A^+ \mathbf{y}$ es la solución del sistema $\mathbf{Ax} = \mathbf{y}$ en el sentido de los mínimos cuadrados que tiene la menor norma euclídea.*

Demostración. Ya sabemos que la proposición es verdadera en caso de que A sea de rango completo, pues en dicho caso la solución al sistema es única y la Proposición 6.2.1 nos garantiza que se puede escribir como $\mathbf{x} = A^+ \mathbf{y}$.

Supongamos, entonces, que el problema es de rango deficiente: $\text{rg}(A) = r < n$. En ese caso, la matriz de valores singulares en la descomposición SVD de A es de la forma

$$\Sigma := \left[\begin{array}{c|c} \Sigma_r & \mathcal{O}_{r \times (n-r)} \\ \hline \mathcal{O}_{(m-r) \times r} & \mathcal{O}_{(m-r) \times (n-r)} \end{array} \right].$$

Arriba, $\Sigma_r \in \mathcal{M}_r(\mathbb{R})$ es una matriz diagonal e invertible, que en su diagonal tiene los r valores singulares no nulos de A , y las demás submatrices son nulas y los subíndices indican sus tamaños. El mismo razonamiento que hicimos para llegar a (6.3) nos permite llegar aquí a que en este caso la norma euclídea del residuo se puede escribir como

$$\|\mathbf{r}\|_2^2 = \|\Sigma_r \mathbf{w}_{(1:r)} - \mathbf{z}_{(1:r)}\|_2^2 + \|\mathbf{z}_{(r+1:m)}\|_2^2,$$

donde $\mathbf{w} := V^t \mathbf{x}$ y $\mathbf{z} = U^t \mathbf{y}$. Las soluciones al problema de mínimos cuadrados se obtienen resolviendo $\mathbf{w}_{(1:r)} = \Sigma_r^{-1} \mathbf{z}_{(1:r)}$, eligiendo libremente las entradas $\mathbf{w}_{(r+1:n)}$, y luego tomando $\mathbf{x} = V \mathbf{w}$.

Para demostrar la proposición, debemos probar que $\mathbf{x}^* := A^+ \mathbf{y}$ cumple esta condición y que es el vector de menor norma euclídea que lo hace. Usando la definición de pseudoinversa y cómo definimos el vector \mathbf{z} , tenemos

$$\mathbf{x}^* = A^+ \mathbf{y} = V \Sigma^+ U^t \mathbf{y} = V \Sigma^+ \mathbf{z}.$$

La pseudoinversa de Σ es

$$\Sigma^+ := \left[\begin{array}{c|c} \Sigma_r^{-1} & \mathcal{O}_{r \times (m-r)} \\ \hline \mathcal{O}_{(n-r) \times r} & \mathcal{O}_{(n-r) \times (m-r)} \end{array} \right] \in \mathcal{M}_{n \times m}(\mathbb{R}),$$

por lo que $(\Sigma^+ \mathbf{z})_{(1:r)} = \Sigma_r^{-1} \mathbf{z}_{(1:r)}$. Esto muestra que \mathbf{x}^* es solución del problema de mínimos cuadrados. Finalmente, observamos que $(\Sigma^+ \mathbf{z})_{(r+1:n)} = \mathbf{0}$, que $\|\mathbf{x}^*\|_2 = \|V \Sigma^+ \mathbf{z}\|_2 = \|\Sigma^+ \mathbf{z}\|_2$ y, para cualquier otro vector \mathbf{x} que sea solución del sistema tenemos

$$\begin{aligned} \|\mathbf{x}\|_2^2 &= \|V^t \mathbf{x}\|_2^2 = \|\mathbf{w}_{(1:r)}\|_2^2 + \|\mathbf{w}_{(r+1:n)}\|_2^2 = \|\Sigma_r^{-1} \mathbf{z}_{(1:r)}\|_2^2 + \|\mathbf{w}_{(r+1:n)}\|_2^2 \\ &= \|\Sigma^+ \mathbf{z}\|_2^2 + \|\mathbf{w}_{(r+1:n)}\|_2^2 = \|\mathbf{x}_*\|_2^2 + \|\mathbf{w}_{(r+1:n)}\|_2^2 \geq \|\mathbf{x}^*\|_2^2. \end{aligned}$$

Concluimos que \mathbf{x}^* es la solución del sistema $A\mathbf{x} = \mathbf{y}$ en el sentido de los mínimos cuadrados que tiene la menor norma euclídea. \square

6.2.3. Comparación de métodos para la resolución de problemas de mínimos cuadrados

Hemos visto varios métodos para resolver problemas de mínimos cuadrados lineales de la forma

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \|A\mathbf{x} - \mathbf{y}\|_2^2.$$

En la práctica, la elección de cuál utilizar depende de nuestros requerimientos de precisión, e implica balancearlos con el costo y la confiabilidad de cada método.

Las **ecuaciones normales** son muy sencillas de implementar. Pasamos de un sistema rectangular $m \times n$ con $m > n$ a uno cuadrado y de tamaño $n \times n$, lo que puede ser interesante cuando $m \gg n$. El sistema resultante tiene la propiedad de tener la matriz simétrica y semidefinida positiva $A^t A$ (definida positiva si $\text{rg}(A) = n$), por lo que incluso se pueden utilizar métodos como la factorización de Cholesky que tratamos en el Práctico 2. La gran desventaja que tiene el uso de las ecuaciones normales es el posible mal condicionamiento del problema resultante, lo que puede afectar a la calidad de la solución computacional que obtengamos.

Los **métodos de ortogonalización** están basados en llevar la matriz A a una forma triangular superior mediante una sucesión de transformaciones ortogonales. Esto está cercanamente relacionado con la factorización QR de la matriz A ⁷. Este tipo de métodos

⁷En la misma forma en que, como vimos en la Sección 2.3, llevar A a una forma triangular superior mediante el proceso de escalerización gaussiana está relacionado con la factorización LU .

son por lo general más costosos computacionalmente que usar las ecuaciones normales, especialmente si $m \gg n$. El método clásico de ortogonalización de Gram-Schmidt que aprendimos en Geometría y Álgebra Lineal 2 no es estable computacionalmente; el método más extendido de ortogonalización, por ser el más eficiente y preciso en general, es el de **Householder**.

Finalmente, usar la **factorización SVD** es la opción computacionalmente más cara. Su elevado costo viene a cambio de una gran precisión y confiabilidad en problemas delicados: en particular, usar SVD nos permite encontrar una solución elegante en problemas de rango deficiente.

A pesar de que no profundizamos en este punto, indicamos como referencia algunas estimaciones de costos computacionales de resolver un sistema con $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ con distintos métodos:

- ecuaciones normales + método directo (Cholesky) $\sim \frac{mn^2}{2} + \frac{n^3}{3}$ flops;
- transformaciones de Householder + sustitución hacia atrás $\sim 2mn^2 - \frac{2n^3}{3}$ flops;
- computar SVD⁸ + sistema diagonal $\sim 2m^2n + 4mn^2 + \frac{9n^3}{2}$ flops.

⁸Usando el método que se describe en [QSS10, Sección 5.8.3].

Capítulo 7

Integración numérica

7.1. Motivación y reglas de Newton-Cotes básicas

Este capítulo trata sobre el problema de integración numérica de funciones reales. Concretamente, dada una función $f: [a, b] \rightarrow \mathbb{R}$ integrable, nos preguntamos de qué forma podemos estimar

$$I(f) := \int_a^b f(x) dx.$$

Si conocemos una primitiva de f , la Regla de Barrow nos permite computar I a partir de evaluar dicha primitiva en los extremos del intervalo. Cuando no conocemos una primitiva de f , en el curso de CDIV aprendimos que se puede aproximar I mediante sumas de Riemann; recordar la Sección 1.2.1. En principio, tal como hicimos en esa sección, podríamos encerrar el valor de I entre una sucesión de sumas inferiores y otra de sumas superiores, y tomar particiones tan finas como sea necesario para nuestro requerimiento de precisión. En este capítulo vamos a proponer y analizar otros métodos para tratar con el problema de integración numérica.

Genéricamente, un **regla de integración numérica** (o una **regla de cuadratura**) es un método que, para aproximar el valor de $\int_a^b f$, utiliza combinaciones lineales de valores de f en ciertos puntos. Esto es, aproxima

$$\int_a^b f(x) dx \approx \sum_{i=0}^n w_i f(x_i), \quad (7.1)$$

donde a $w_0, \dots, w_n \subset \mathbb{R}$ se les suele llamar **pesos de la regla**, y a $x_0, \dots, x_n \subset [a, b]$ se los llama **nodos de la regla**.

Comenzamos por usar ideas de interpolación para introducir las llamadas **reglas de Newton-Cotes**.

Definición 7.1.1 (conjuntos cerrado y abierto de nodos de Newton-Cotes). Sea $[a, b] \subset \mathbb{R}$ un intervalo. Dado un número entero $n \geq 1$, un **conjunto cerrado de nodos de**

Newton-Cotes es $\{x_i\}_{i=0}^n$ tal que

$$x_i = a + \frac{i(b-a)}{n}, \quad i = 0, \dots, n.$$

Por otra parte, dado un número entero $n \geq 0$, un **conjunto abierto de nodos de Newton-Cotes** es $\{x_i\}_{i=0}^n$ tal que

$$x_j = a + \frac{(i+1)(b-a)}{n+2}, \quad i = 0, \dots, n.$$

△

En otras palabras, la Definición 7.1.1 indica que un conjunto cerrado de $n+1$ nodos de Newton-Cotes consiste en los nodos que resultan al hacer una partición uniforme del intervalo $[a, b]$ en n subintervalos. En cambio, un conjunto abierto de $n+1$ nodos de Newton-Cotes hace una partición uniforme en $n+2$ subintervalos pero le quita los extremos a y b .

Definición 7.1.2 (reglas de Newton-Cotes). Dada $f: [a, b] \rightarrow \mathbb{R}$, sea $x_0 < \dots < x_n$ un conjunto (cerrado o abierto) de nodos de Newton-Cotes. Consideramos el polinomio interpolante p_n de f por el conjunto de puntos elegidos (recordar la Definición 3.2.1), y aproximamos

$$\int_a^b f(x) dx \approx \int_a^b p_n(x) dx := Q.$$

A esta aproximación se la llama **regla de Newton-Cotes** para aproximar $\int_a^b f$. △

Observación 7.1.1 (los pesos no dependen de f). Consideremos una regla de Newton-Cotes para aproximar $\int_a^b f$. Si escribimos el polinomio interpolante p_n por $(x_0, f(x_0)), \dots, (x_n, f(x_n))$ en la forma de Lagrange (3.8), tenemos

$$p_n(x) = \sum_{i=0}^n f(x_i) L_n^i(x) \Rightarrow Q = \int_a^b p_n(x) dx = \sum_{i=0}^n f(x_i) \int_a^b L_n^i(x) dx.$$

Por lo tanto, teniendo en cuenta la expresión (7.1), observamos que los pesos de la regla de cuadratura son $w_i = \int_a^b L_n^i(x) dx$ para todo $i = 0, \dots, n$: **los pesos solamente dependen de la elección de los nodos de la regla, pero no dependen de la función cuya integral queremos aproximar.** △

A continuación, damos algunos ejemplos básicos de reglas de Newton-Cotes.

Regla del punto medio. Esta es la regla de Newton-Cotes abierta más sencilla: corresponde a tomar un único punto, el punto medio del intervalo $[a, b]$, e interpolar f por ese punto. Obviamente, el polinomio interpolante aquí es la constante

$$p_0(x) := f\left(\frac{a+b}{2}\right).$$

La **regla del punto medio** consiste en aproximar

$$I(f) \approx Q^M(f) := \int_a^b p_0(x) dx = f\left(\frac{a+b}{2}\right) (b-a). \quad (7.2)$$

Con la notación (7.1), estamos tomando $x_0 := \frac{a+b}{2}$ y $w_0 = b-a$.

Regla del trapecio. En una regla de Newton-Cotes cerrada, necesariamente los dos extremos del intervalo tienen que ser nodos. Por lo tanto, la regla de Newton-Cotes cerrada más sencilla utiliza solamente los puntos $(a, f(a))$, $(b, f(b))$ y considera la interpolante lineal por ellos,

$$p_1(x) := f(a) + \left(\frac{f(b) - f(a)}{b-a}\right)(x-a).$$

La **regla del trapecio** consiste en la aproximación

$$I(f) \approx Q^T(f) := \int_a^b p_1(x) dx = \left(\frac{f(a) + f(b)}{2}\right) (b-a). \quad (7.3)$$

Usando la notación (7.1), tenemos $x_0 := a$, $x_1 := b$, y $w_0 = w_1 := \frac{b-a}{2}$.

Regla de Simpson. Se trata de una regla de Newton-Cotes cerrada, que consiste en realizar la interpolación polinomial por tres puntos: los dos extremos del intervalo y el punto medio. Esto da lugar a una interpolante cuadrática, que llamaremos p_2 y, tal como probamos en el Práctico 3,

$$I(f) \approx Q^S(f) := \int_a^b p_2(x) dx = \left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) \frac{(b-a)}{6}. \quad (7.4)$$

Esta es la llamada **regla de Simpson**, en la que los nodos son $x_0 := a$, $x_1 := \frac{a+b}{2}$, y $x_2 := b$, y los pesos son $w_0 = w_2 := \frac{b-a}{6}$, $w_1 := \frac{2(b-a)}{3}$.

La Figura 7.1 ilustra las reglas del punto medio, del trapecio, y de Simpson.

Definición 7.1.3 (orden de una regla de cuadratura). Decimos que el **orden** de una regla de integración numérica es el menor grado del polinomio que la regla *no integra exactamente*. Esto es, una regla tiene orden m si integra exactamente a todos los polinomios de grado menor o igual a $m-1$ pero hay un polinomio de grado m tal que la regla no integra exactamente. \triangle

Ejemplo 7.1.1 (órdenes de las reglas del punto medio, del trapecio, y de Simpson). Es evidente que una regla de Newton-Cotes que use $n+1$ puntos tiene que ser de orden por lo menos igual a $n+1$, ya que si f es un polinomio de grado n , al interpolarlo por $n+1$ puntos obtenemos $p_n = f$. Por lo tanto, la regla del punto medio tiene orden al menos 1, la del trapecio tiene orden al menos 2, y la de Simpson tiene orden por lo menos 3.

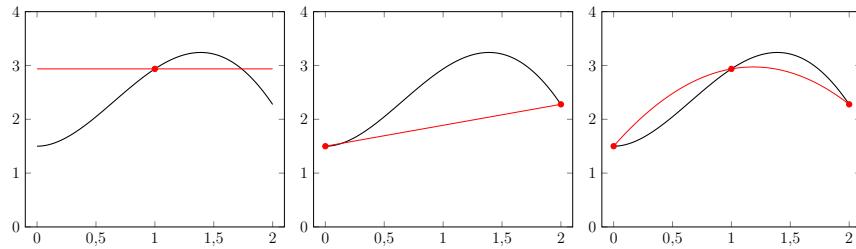


Figura 7.1: Ilustración de las reglas del punto medio (7.2) (izquierda), del trapecio (7.3) (centro), y de Simpson (7.4) (derecha): en ellas se aproxima la integral de la función graficada en negro por la del polinomio interpolante graficado en rojo.

Sin embargo, **no es necesario interpolar exactamente una función para obtener una aproximación exacta a su integral**. En efecto, supongamos que f es lineal en $[a, b]$. Como muestra la Figura 7.2, la regla del punto medio para aproximar $\int_a^b f$ es exacta, pues

$$\int_a^b f(x)dx = \frac{(f(b) + f(a))}{2} (b - a) = f\left(\frac{a+b}{2}\right) (b - a) = Q^M(f).$$

Por lo tanto, la regla del punto medio es de orden mayor o igual a 2. En forma similar, se

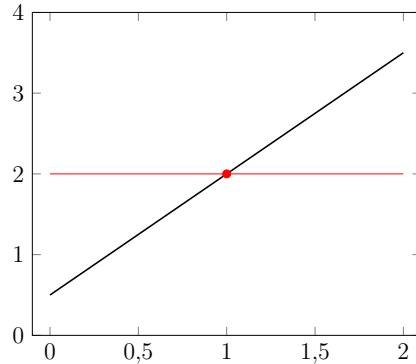


Figura 7.2: Las regla del punto medio tiene orden por lo menos igual a 2: si f es lineal en $[a, b]$, por simetría tenemos $Q^M(f) = \int_a^b f$.

puede demostrar que la regla de Simpson es capaz de integrar polinomios de grado 3 en forma exacta, por lo que tiene orden por lo menos 4. \triangle

Observación 7.1.2 (otra forma de entender la regla de Simpson). El ejemplo anterior muestra que las reglas del punto medio y del trapecio son de orden por lo menos 2. Es sencillo verificar que son de orden igual a 2: para ello, consideremos la función cuadrática $f(x) = x^2$ en el intervalo $[0, 1]$. Tenemos $I = \int_0^1 x^2 dx = \frac{1}{3}$, pero usando (7.2) y (7.3),

$$Q^M(f) = f\left(\frac{1}{2}\right) = \frac{1}{4}, \quad Q^T(f) = \left(\frac{f(0) + f(1)}{2}\right) = \frac{1}{2}.$$

Como integran exactamente a funciones lineales pero no a la función cuadrática f , deducimos que las reglas del punto medio y del trapecio son de orden igual a 2.

Más aún, observamos que $Q^M(f) < I(f) < Q^T(f)$ y que los errores son $Q^M(f) - I(f) = -\frac{1}{12}$, $Q^T(f) - I(f) = \frac{1}{6}$. En este caso, la regla del punto medio es el doble de precisa que la regla del trapecio. Como los errores tienen signos opuestos, podríamos considerar una nueva regla de cuadratura que combine a las del punto medio con la del trapecio,

$$Q^*(f) := \frac{2}{3}Q^M(f) + \frac{1}{3}Q^T(f).$$

Esta nueva regla es capaz de integrar exactamente a las funciones lineales (porque las reglas del punto medio y del trapecio lo son) y a la función $f(x) = x^2$; por lo tanto, es de orden mayor que 2. Escribiendo explícitamente los pesos y nodos de esta nueva regla, nos encontramos con

$$Q^*(f) = \frac{2}{3}f\left(\frac{1}{2}\right) + \frac{1}{3}\left(\frac{f(0) + f(1)}{2}\right) = \frac{1}{6}f(0) + \frac{2}{3}f\left(\frac{1}{2}\right) + \frac{1}{6}f(1).$$

Por lo tanto, ésta no es otra que la regla de Simpson (7.4). \triangle

Ejercicio 7.1. Demostrar que la regla de Simpson es de grado 4.

Sugerencia: para demostrar que es de orden mayor o igual a 4 sin hacer demasiadas cuentas, considerar f un polinomio de grado 3 arbitrario en $[a, b]$, e interpolarlo por los nodos $a, b, \frac{a+b}{2}$ y z , con $z \in [a, b]$ cualquiera ($z \neq a, b, \frac{a+b}{2}$). Como estamos interpolando al polinomio cúbico f por 4 puntos, la interpolante coincide con f . Escribir la interpolante en la forma de Newton (es importante que z sea el último nodo), y observar que la interpolante por los primeros 3 puntos coincide con la que se usa para definir la regla de Simpson. Por lo tanto, para demostrar que la regla de Simpson es de orden mayor o igual a 4, basta con demostrar que

$$\int_a^b (x-a)(x-b)\left(x-\frac{a+b}{2}\right) dx = 0.$$

Por otra parte, para demostrar que el orden es exactamente 4, basta con verificar que si $f(x) = x^4$ entonces

$$\frac{1}{5} = \int_0^1 f(x) dx \neq Q^S(f) = \frac{5}{24}.$$

Una pregunta esencial que nos abocamos a responder a continuación es qué podemos decir respecto a los errores cometidos por estas reglas de cuadratura básicas.

Teorema 7.1.1 (errores en reglas de cuadratura). *Sean $f: [a, b] \rightarrow \mathbb{R}$ integrable, $I(f) = \int_a^b f$ y $Q^M(f), Q^T(f), Q^S(f)$ definidas por (7.2), (7.3), y (7.4), respectivamente.*

a) Si $f \in C^2$, entonces

$$|Q^M(f) - I(f)| \leq \frac{(b-a)^3}{24} \|f''\|_{L^\infty(a,b)}, \quad (7.5)$$

y

$$|Q^T(f) - I(f)| \leq \frac{(b-a)^3}{12} \|f''\|_{L^\infty(a,b)}. \quad (7.6)$$

b) Si $f \in C^4$, entonces

$$|Q^S(f) - I(f)| \leq \frac{(b-a)^5}{2880} \|f^{(4)}\|_{L^\infty(a,b)}. \quad (7.7)$$

Demostración. Vamos a demostrar solamente (7.5) y (7.6). Escribimos $x_M := \frac{a+b}{2}$ el punto medio del intervalo $[a, b]$. Como $f \in C^2$, podemos hacer un desarrollo de Taylor de primer orden de f alrededor de x_M usando la forma de Lagrange para el resto (Teorema A.1.2): para todo $x \in [a, b]$, existe θ_x entre x y x_M (y por lo tanto $\theta_x \in (a, b)$) tal que

$$f(x) = f(x_M) + f'(x_M)(x - x_M) + \frac{f''(\theta_x)}{2}(x - x_M)^2.$$

Integramos de los dos lados de esta igualdad respecto a x , y reconocemos que $\int_a^b f(x_M) dx = f(x_M)(b-a) = Q^M(f)$, para obtener

$$I(f) = Q^M(f) + \int_a^b f'(x_M)(x - x_M) dx + \int_a^b \frac{f''(\theta_x)}{2}(x - x_M)^2 dx.$$

La primer integral del lado derecho es nula, pues $\int_a^b (x - x_M) dx = 0$; esta es justamente la simetría que identificamos en el Ejemplo 7.1.1. Por lo tanto,

$$|I(f) - Q^M(f)| = \left| \int_a^b \frac{f''(\theta_x)}{2}(x - x_M)^2 dx \right| \leq \int_a^b \left| \frac{f''(\theta_x)}{2}(x - x_M)^2 \right| dx.$$

Como la única información que tenemos sobre θ_x es que está en el intervalo (a, b) , acotamos $|f''(\theta_x)| \leq \|f''\|_{L^\infty(a,b)}$. Finalmente, es sencillo verificar que $\int_a^b (x - x_M)^2 dx = \frac{(b-a)^3}{12}$, y se concluye (7.5).

Para demostrar (7.6), notamos que la regla del trapecio resulta de integrar la interpolante lineal p_1 por $(a, f(a))$, $(b, f(b))$, por lo que podemos usar la cota (3.13):

$$|p_1(x) - f(x)| \leq \frac{|(x-a)(x-b)| \|f''\|_{L^\infty([a,b])}}{2} \quad \forall x \in [a, b].$$

Integramos esta desigualdad de los dos lados, y observamos que $\int_a^b |(x-a)(x-b)| dx = \frac{(b-a)^3}{6}$, de donde

$$|Q^T(f) - I(f)| = \left| \int_a^b (p_1(x) - f(x)) dx \right| \leq \int_a^b |p_1(x) - f(x)| dx \leq \frac{(b-a)^3 \|f''\|_{L^\infty([a,b])}}{12}.$$

□

Es claro que las reglas de Newton-Cotes en la forma en que discutimos hasta aquí son de poco uso en la práctica: si el intervalo $[a, b]$ no es muy pequeño, entonces en general no podemos esperar que las reglas del punto medio, del trapecio, o de Simpson produzcan una

buenas aproximaciones a la integral de f en $[a, b]$. Esto es análogo a usar una interpolante de grado bajo en un intervalo grande: seguramente dé una aproximación de poca calidad a la función que queremos interpolar. Una opción que puede resultar natural es aumentar el grado polinomial; sin embargo, si recordamos nuestra discusión en la Sección 3.3.2 sobre el fenómeno de Runge, rápidamente caemos en la cuenta de que hacer interpolaciones de Newton-Cotes –que usan nodos equiespaciados– no es una buena idea a menos que las funciones cuyas integrales queremos estimar sean “buenas”, como la del Ejemplo 3.3.1. Otra propiedad que vuelve no deseable a las reglas de Newton-Cotes de orden alto es que en ellas suele haber pesos con signo negativo. Esto puede conducir a sumas largas en las que hay varios términos de signos opuestos y, en consecuencia, las hacen pasibles de cancelaciones numéricas.

Podemos considerar las mismas alternativas que en la Sección 3.3.2: o elegimos mejor los nodos de cuadratura o nos mantenemos con una interpolación de grado bajo pero dividimos $[a, b]$ en subintervalos pequeños. Respecto a la primer alternativa, una opción relacionada con los nodos de Chebyschev es la llamada **cuadratura de Clenshaw-Curtis**, sobre la que no vamos a profundizar; otra opción, que exploramos en la Sección 7.4, es la llamada *cuadratura gaussiana*. La segunda alternativa da lugar a las *reglas compuestas*, que describimos a continuación.

7.2. Reglas compuestas

La idea detrás de las **reglas de cuadratura compuestas** es análoga a la que usamos en la Sección 3.4: para aproximar $\int_a^b f$, en vez de usar una única regla para todo el intervalo $[a, b]$, consideramos una partición de dicho intervalo y usamos la regla en cada uno de los subintervalos definidos. Por simplicidad, nos vamos a restringir a las tres reglas básicas que tratamos en la Sección 7.1: del punto medio, del trapecio, y de Simpson.

Concretamente, elegimos una de las tres reglas, partimos $a = x_0 < \dots < x_n = b$ y, en cada intervalo $[x_i, x_{i+1}]$, aplicamos la regla para aproximar $\int_{x_i}^{x_{i+1}} f$. Fijemos un poco de notación. Para cada $i = 0, \dots, n - 1$, llamamos

$$h_i := x_{i+1} - x_i$$

y, para $* \in \{M, T, S\}$ ¹, ponemos $Q_i^*(f)$ al valor resultante de aplicar la regla de cuadratura $*$ en el intervalo $[x_i, x_{i+1}]$. Como las integrales son aditivas respecto a los intervalos, tenemos

$$I(f) = \int_a^b f(x) dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x) dx \approx \sum_{i=0}^{n-1} Q_i^*(f) =: Q^{*,c}(f).$$

¹Esta notación solamente quiere indicar que vamos a considerar las reglas del punto medio, del trapecio, y de Simpson.

Por ejemplo, si aplicamos la regla del trapecio en cada subintervalo, tendremos

$$Q_i^T(f) = h_i \left(\frac{f(x_i) + f(x_{i+1})}{2} \right) \quad (7.8)$$

y por lo tanto la regla compuesta del trapecio es

$$Q^{T,c}(f) = \sum_{i=0}^{n-1} h_i \left(\frac{f(x_i) + f(x_{i+1})}{2} \right) = \frac{h_0}{2} f(a) + \sum_{i=1}^{n-1} \left(\frac{h_{i-1} + h_i}{2} \right) f(x_i) + \frac{h_{n-1}}{2} f(b).$$

Si usamos la notación (7.1), tenemos que los nodos de la regla son x_0, \dots, x_n , y los pesos son $w_0 = \frac{h_0}{2}$, $w_i = \frac{h_{i-1} + h_i}{2}$ para $i = 1, \dots, n - 1$, y $w_n = \frac{h_{n-1}}{2}$.

Observación 7.2.1 (notación). Nuestra notación para integrales compuestas no hace referencia a la partición que hayamos tomado, pero obviamente el valor que calculemos depende de ella. \triangle

Teorema 7.2.1 (errores de reglas compuestas). *Sean $f: [a, b] \rightarrow \mathbb{R}$ integrable, $a = x_0 < \dots < x_n = b$, $h_i = x_{i+1} - x_i$ ($i = 0, \dots, n - 1$). Llamamos $I(f) = \int_a^b f$ y $Q^{M,c}(f)$, $Q^{T,c}(f)$, $Q^{S,c}(f)$ a las reglas del punto medio, del trapecio, y de Simpson compuestas, respectivamente.*

a) Si $f \in C^2$, entonces

$$\begin{aligned} |Q^{M,c}(f) - I(f)| &\leq \sum_{i=0}^{n-1} \frac{h_i^3}{24} \|f''\|_{L^\infty(x_i, x_{i+1})}, \\ |Q^{T,c}(f) - I(f)| &\leq \sum_{i=0}^{n-1} \frac{h_i^3}{12} \|f''\|_{L^\infty(x_i, x_{i+1})}. \end{aligned} \quad (7.9)$$

b) Si $f \in C^4$, entonces

$$|Q^{S,c}(f) - I(f)| \leq \sum_{i=0}^{n-1} \frac{h_i^5}{2880} \|f''\|_{L^\infty(x_i, x_{i+1})}.$$

Demostración. Las tres desigualdades se deducen de las desigualdades correspondientes en el Teorema 7.1.1, integrando en cada subintervalo, y usando la aditividad de las integrales respecto a las particiones. \square

En caso de que las particiones sean *uniformes*, podemos escribir una cota de error en forma más compacta.

Corolario 7.2.2 (particiones uniformes). *En las mismas hipótesis que en el Teorema 7.2.1, si asumimos además que $h_i = h := \frac{(b-a)}{n}$ para todo $i = 0, \dots, n$, tenemos:*

a) Si $f \in C^2$, entonces

$$|Q^{M,c}(f) - I(f)| \leq \frac{(b-a)h^2}{24} \|f''\|_{L^\infty(a,b)},$$

y

$$|Q^{T,c}(f) - I(f)| \leq \frac{(b-a)h^2}{12} \|f''\|_{L^\infty(a,b)}.$$

b) Si $f \in C^4$, entonces

$$|Q^{S,c}(f) - I(f)| \leq \frac{(b-a)h^4}{2880} \|f^{(4)}\|_{L^\infty(a,b)}.$$

Demostración. Las desigualdades se deducen inmediatamente del teorema anterior, usando que para todo $i = 0, \dots, n-1$ y toda función $g: [a, b] \rightarrow \mathbb{R}$, se cumple $\|g\|_{L^\infty(x_i, x_{i+1})} \leq \|g\|_{L^\infty(a,b)}$, y que

$$h = \frac{(b-a)}{n} \Rightarrow \sum_{i=0}^{n-1} 1 = n = \frac{b-a}{h}.$$

□

Ejemplo 7.2.1. Sea $f: [-1, 1] \rightarrow \mathbb{R}$ la función de Runge,

$$f(x) = \frac{1}{1 + 25x^2}.$$

Como una primitiva de f es $F(x) = \frac{1}{5} \arctan(5x)$, tenemos

$$I(f) = \int_{-1}^1 f(x) dx = \frac{1}{5} \arctan(5x) \Big|_{-1}^1 = \frac{2}{5} \arctan 5 \simeq 0,5494.$$

Consideramos la regla del trapecio compuesta asociada a una partición uniforme del intervalo $[-1, 1]$ con $n+1$ nodos, esto es,

$$x_i := -1 + \frac{2i}{n}, \quad i = 0, \dots, n \Rightarrow h = \frac{2}{n}.$$

Recordemos, del Ejemplo 3.4.1, que $\|f''\|_{L^\infty(-1,1)} = 50$. Por lo tanto, para la regla compuesta del trapecio podemos acotar el error mediante

$$|I(f) - Q^{T,c}(f)| \leq \frac{2(2/n)^2}{12} \|f''\|_{L^\infty(a,b)} = \frac{100}{3n^2}.$$

Por ejemplo, si queremos garantizar que el error en nuestra aproximación sea menor que 10^{-4} , basta con tomar n tal que se cumpla

$$\frac{100}{3n^2} < 10^{-4} \Rightarrow n > \frac{10^3}{\sqrt{3}} \simeq 577,35,$$

por lo que alcanza con tomar $n = 578$ subintervalos. Al implementar la regla del trapecio compuesta en Octave, nos encontramos con que el error en ese caso es aproximadamente $1,48 \times 10^{-7}$. Si bien esto está de acuerdo con la teoría, ya que demostramos que tomar 578 subintervalos es *suficiente* para que el error de integración sea menor que 10^{-4} , da la sensación de que nuestra cota no es muy precisa. Tomando 578 subintervalos estamos refinando mucho más de lo necesario. De hecho, experimentando en Octave, uno se encuentra con que basta con solamente tomar $n = 14$ para que el error de integración satisfaga nuestro requerimiento de precisión: en ese caso, el error es aproximadamente $6,06 \times 10^{-5}$. \triangle

7.3. Adaptatividad

Sigamos interpretando los resultados del Ejemplo 7.2.1. Analizando un poco más en detalle cómo llegamos a las estimaciones del Corolario 7.2.2, observamos que acotamos la norma L^∞ de derivadas de f (en el ejemplo, las derivadas segundas) en cada subintervalo por la norma L^∞ de dicha derivada en todo el intervalo $[a, b]$. La Figura 7.3 muestra el gráfico de la derivada segunda de la función de Runge en el intervalo $[0, 1]$ (notar que f'' es una función par): la cota $|f''(x)| \leq 50$ es relevante solamente para x cercano al origen. Ya cuando $x \approx 1/2$, tenemos $|f''(x)| \approx 2$. Tomar subintervalos tan pequeños para $x \geq 1/2$ es demasiado para el requerimiento de precisión que tenemos. En este ejemplo es claro porque conocemos y podemos graficar la derivada segunda de f y sabemos que f es “mucho más lineal” cuando x está alejado de 0 que cuando está cerca.

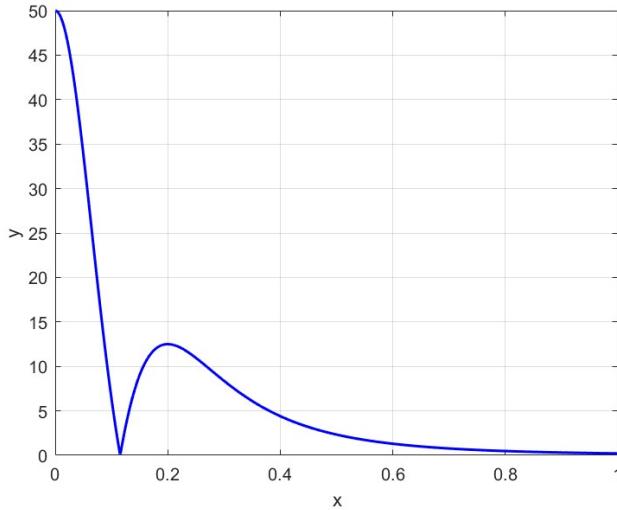


Figura 7.3: Valor absoluto de la segunda derivada de la función de Runge en el intervalo $[0, 1]$.

En esta sección, mostramos algunas ideas sobre **adaptatividad**: nuestro objetivo es hacer reglas de cuadratura compuestas que de alguna forma puedan determinar cuándo es

necesario tomar intervalos pequeños y cuándo no vale la pena hacerlo. El objetivo es, dada una cierta tolerancia, computar la integral de f sobre el intervalo $[a, b]$ con esa tolerancia y *de forma eficiente*. Para ilustrar las ideas principales, vamos a fijar una regla de cuadratura: solamente vamos a considerar que nuestra aproximación se hace mediante la regla del trapecio compuesta. El desafío está en encontrar una partición del intervalo $[a, b]$ que nos permita alcanzar nuestro objetivo.

Supongamos que implementamos una regla del trapecio compuesta sobre una partición $a = x_0 < \dots < x_n = b$. Descomponemos el error de integración numérica como la suma de los *errores locales*, asociados a la integración en cada subintervalo:

$$Q^{T,c}(f) - I(f) = \sum_{i=0}^{n-1} Q_i^T(f) - I_i(f),$$

donde $Q_i^T(f)$ está dado por (7.8) e $I_i(f) := \int_{x_i}^{x_{i+1}} f$. Sea ε nuestra tolerancia: queremos que sea $|Q^{T,c}(f) - I(f)| \leq \varepsilon$, e idealmente no mucho menor, ya que si lo fuera querría decir que seguramente nuestro algoritmo refinó al intervalo más de lo necesario.

En principio, no sabemos cómo ni en cuántas partes tenemos que partir el intervalo $[a, b]$ para que el error de integración sea menor o igual a ε . Para empezar, supongamos que podemos calcular los errores locales exactamente. Si comenzamos nuestro algoritmo con una regla del trapecio en todo el intervalo $[a, b]$, entonces podemos comparar si el error es menor a ε o no. Si lo es, terminamos; si no lo es, tenemos que seguir partiendo nuestro intervalo. Si partimos el intervalo $[a, b]$ a la mitad, ahora tendremos dos subintervalos: ¿cómo debería ser el error local en cada uno de esos subintervalos para que el algoritmo termine? Una respuesta es que en cada uno de esos subintervalos el valor absoluto del error local sea menor que $\varepsilon/2$: como tenemos dos subintervalos,

$$\begin{aligned} |Q^{T,c}(f) - I(f)| &= |(Q_0^T(f) - I_0(f)) + (Q_1^T(f) - I_1(f))| \\ &\leq |Q_0^T(f) - I_0(f)| + |Q_1^T(f) - I_1(f)| < \varepsilon. \end{aligned}$$

El factor $1/2$ multiplicando a ε en esta tolerancia local está relacionado con el hecho de que los dos subintervalos miden la mitad que el intervalo $[a, b]$. En efecto, si logramos que el error local en un subintervalo sea como ε multiplicado por la proporción del intervalo $[a, b]$ que el subintervalo cubre, entonces habremos terminado: si para cada $i = 0, \dots, n-1$, se tiene

$$|Q_i^T(f) - I_i(f)| \approx \frac{h_i}{b-a} \varepsilon, \quad (7.10)$$

entonces

$$|Q^{T,c}(f) - I(f)| = \left| \sum_{i=0}^{n-1} Q_i^T(f) - I_i(f) \right| \leq \sum_{i=0}^{n-1} |Q_i^T(f) - I_i(f)| \approx \frac{\varepsilon}{b-a} \sum_{i=0}^{n-1} h_i = \varepsilon,$$

porque $\sum_{i=0}^{n-1} h_i = b - a$. La estimación de arriba es lo mejor que podemos lograr si no tenemos un control sobre los signos de los errores, lo que es bastante difícil de lograr en la práctica.

De este modo, comenzamos a delinear cómo sería una cuadratura adaptativa usando la regla del trapecio compuesta: dadas una función $f : [a, b] \rightarrow \mathbb{R}$ integrable una tolerancia $\varepsilon > 0$, y un intervalo $I \subset [a, b]$ de longitud ℓ ,

1. se parte el intervalo en dos subintervalos de longitud $\ell/2$;
2. en cada uno de los dos subintervalos nuevos,
 - a) se aplica la regla del trapecio y se computa el error cometido,
 - b) si el error es menor que $\frac{\ell}{b-a}\varepsilon$ (esto es, si se cumple (7.10)), entonces se sale de este subintervalo y se va al siguiente; si el error es mayor que $\frac{\ell}{b-a}\varepsilon$, entonces se toma I como este nuevo subintervalo y se vuelve a comenzar.

Observación 7.3.1 (terminación). Si f es una función continua en $[a, b]$, entonces el algoritmo que describimos arriba termina. Esto se puede deducir a partir de la continuidad uniforme de f en $[a, b]$. \triangle

Un detalle importante que omitimos hasta ahora es que, en general, no podemos computar exactamente los errores de integración locales: asumir que podemos estimarlos es esencialmente asumir que podemos integrar a f “a mano”. En la práctica, se suele reemplazar el error local por una *estimación del error local*. Concretamente, en cada intervalo $[x_i, x_{i+1}]$ se considera una regla de cuadratura más precisa que la que se está usando para calcular la integral: en nuestro caso, que estamos considerando la regla del trapecio en cada subintervalo, podríamos usar la regla de Simpson o una regla del trapecio compuesta dentro del subintervalo. Para esta regla más precisa, que denotamos por $*$, esperamos que se cumpla

$$|Q_i^*(f) - I_i(f)| \ll |Q_i^T(f) - I_i(f)|.$$

Además, la cantidad $|Q_i^T(f) - Q_i^*(f)|$ es computable, y la podemos usar como una estimación del error, pues

$$\left. \begin{aligned} |Q_i^T(f) - I_i(f)| &\leq |Q_i^T(f) - Q_i^*(f)| + |Q_i^*(f) - I_i(f)| \\ |Q_i^T(f) - Q_i^*(f)| &\leq |Q_i^T(f) - I_i(f)| + |I_i(f) - Q_i^*(f)| \end{aligned} \right\} \Rightarrow |Q_i^T(f) - I_i(f)| \approx |Q_i^T(f) - Q_i^*(f)|.$$

Remarcamos que **usar la condición**

$$|Q_i^T(f) - Q_i^*(f)| \approx \frac{h_i}{b-a}\varepsilon,$$

en lugar de (7.10) implica que al final de nuestro algoritmo *esperamos* que el error de integración numérica sea menor que ε , pero no podemos garantizarlo.

Observación 7.3.2. Por lo general, las funciones de Octave/Matlab para computar integrales (por ejemplo, `quad` o `integral`), usan algoritmos adaptativos. En particular, `quad` usa la regla de Simpson para el cálculo de integrales y una regla de Newton-Cotes de 5 puntos (también conocida como regla de Simpson de 5 puntos) para la estimación de errores. \triangle

Ejemplo 7.3.1. Implementamos nuestro Algoritmo adaptativo usando la regla del trapezo compuesta a la función que consideramos en la Sección 1.2.1: queremos computar

$$I = \int_0^1 e^{\sin x} dx.$$

Ponemos una tolerancia $\varepsilon = 10^{-4}$. La Figura 7.4 (izquierda) muestra el gráfico del integrando $f(x) = e^{\sin x}$ en el intervalo y los puntos en los que el algoritmo hizo subdivisiones. En total, se utilizaron 57 nodos, y obtuvimos el valor

$$I \approx 1,6319.$$

Este valor es consistente con lo que estimamos en el Ejemplo 1.2.1 usando sumas de

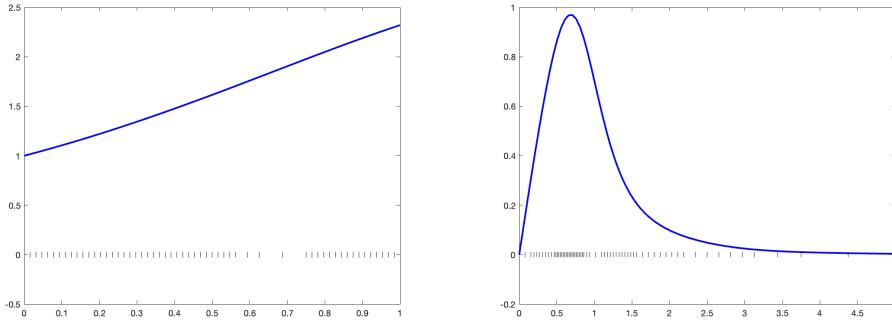


Figura 7.4: Cuadratura adaptativa. Izquierda: cálculo de $\int_0^1 e^{\sin x} dx$ con tolerancia $\varepsilon = 10^{-4}$; derecha: cálculo de $\int_0^5 \frac{2x - \sin^3 x}{1+x^5} dx$ con tolerancia $\varepsilon = 10^{-2}$.

Riemann, pero con bastante más particiones del intervalo. En este ejemplo, notamos que el gráfico de $f(x)$ parece ser “bastante lineal”² por lo que los subintervalos tomados son bastante uniformes.

El efecto *adaptativo* de nuestro algoritmo es más evidente cuando consideramos integrandos cuya derivada segunda varía mucho. La Figura 7.4 (derecha) muestra los intervalos tomados para calcular $\int_0^5 \frac{2x - \sin^3 x}{1+x^5} dx$ con una tolerancia $\varepsilon = 10^{-2}$. La partición es mucho más fina cuando x está aproximadamente entre $1/2$ y 1 que para x cercano a 5 . \triangle

7.4. Cuadratura gaussiana

Consideraremos una regla de cuadratura como (7.1). Al definir reglas de Newton-Cotes, asumimos que los nodos están equiespaciados. Si asumimos que los nodos de la cuadratura están dados, entonces el problema de diseñar reglas de cuadratura se reduce a encontrar

²Esto simplemente quiere decir que $|f''|$ es bastante pequeña en $[0, 1]$.

conjuntos satisfactorios de pesos. El problema se reduce a resolver un sistema de ecuaciones lineales, porque la expresión (7.1) es lineal en w_0, \dots, w_n .

Más precisamente, supongamos que tenemos dados $n + 1$ puntos x_0, \dots, x_n , queremos una regla como (7.1) y le pedimos que sea exacta para polinomios de grado lo más alto posible. Como tenemos $n + 1$ puntos, nos basta con hallar el polinomio interpolante p_n por esos puntos; esto es exacto siempre que el integrando sea un polinomio de grado menor o igual a n . Por lo tanto, nuestra regla de cuadratura será exacta para cualquier polinomio de grado menor o igual que n . Incluso podría ser posible que fuese exacta para algún polinomio de grado mayor, como ocurre con la regla del punto medio —que usa un interpolante de grado 0 y es exacta para funciones lineales—, pero esto dependerá de la ubicación de los nodos. En general, si tenemos dados x_0, \dots, x_n , no tiene sentido esperar que la regla (7.1) sea exacta para polinomios de grado mayor que n .

Las **reglas de cuadratura gaussiana** consisten en dejar los x_0, \dots, x_n como grados de libertad. Si tenemos $2n + 2$ grados de libertad, que se corresponden con $n + 1$ nodos y $n + 1$ pesos, en principio podríamos esperar que la regla que obtengamos sea exacta para polinomios de grado menor o igual a $2n + 1$. La dificultad radica en que el sistema de ecuaciones que se obtiene es *no lineal*. Ilustramos este punto con un ejemplo.

Ejemplo 7.4.1. Consideramos la regla de cuadratura

$$\int_{-1}^1 f(x) dx \simeq w_1 f(x_1) + w_2 f(x_2),$$

y buscamos $w_1, w_2 \in \mathbb{R}$, $x_1, x_2 \in [-1, 1]$ tales que la regla sea exacta para todos los polinomios de grado menor o igual que 3. Para este fin, basta con elegir nuestra base favorita del espacio de polinomios de grado menor o igual que 3, imponer que la regla sea exacta en cada polinomio de dicha base, y resolver el problema resultante. Tomamos la base monomial $\{1, x, x^2, x^3\}$, y queremos que se cumplan:

$$\begin{aligned} 2 &= \int_{-1}^1 1 dx = w_1 1 + w_2 1 \quad \Rightarrow w_1 + w_2 = 2, \\ 0 &= \int_{-1}^1 x dx = w_1 x_1 + w_2 x_2 \quad \Rightarrow w_1 x_1 + w_2 x_2 = 0, \\ 2/3 &= \int_{-1}^1 x^2 dx = w_1 x_1^2 + w_2 x_2^2 \quad \Rightarrow w_1 x_1^2 + w_2 x_2^2 = 2/3, \\ 0 &= \int_{-1}^1 x^3 dx = w_1 x_1^3 + w_2 x_2^3 \quad \Rightarrow w_1 x_1^3 + w_2 x_2^3 = 0. \end{aligned} \tag{7.11}$$

Tenemos que resolver el sistema (7.11). Multiplicamos la segunda ecuación por x_2^2 y se la restamos a la última para obtener

$$w_1 x_1^3 - w_1 x_1 x_2^2 = 0 \quad \Rightarrow \quad w_1 = 0, \quad \text{o } x_1 = 0, \quad \text{o } |x_1| = |x_2|.$$

Si $w_1 = 0$, entonces la primera ecuación en (7.11) implicaría $w_2 = 2$, y se llega rápidamente a una contradicción: la segunda ecuación daría que $x_2 = 0$, y sustituyendo en la tercera nos encontramos con que $0 = 2/3$. Si fuera $x_1 = 0$, se tiene una contradicción similar entre la segunda y la tercera ecuación en (7.11): tendríamos $w_2 x_2 = 0$ y $w_2 x_2^2 = 2/3$.

Por lo tanto, necesariamente tiene que ser $|x_1| = |x_2|$. Podemos descartar que $x_1 = x_2$ fácilmente usando las dos primeras fórmulas en (7.11), de modo que concluimos $x_1 = -x_2 \neq 0$ (recordemos que $x_1 = 0$ ya está descartado). Sustituimos en la segunda fórmula y tenemos $w_1 = w_2$; reemplazando esta identidad en la primera, obtenemos $w_1 = w_2 = 1$. Finalmente, para obtener los valores de x_1 y x_2 , utilizamos la tercera fórmula en (7.11) y concluimos³

$$x_1 = -\frac{1}{\sqrt{3}}, \quad w_1 = 1, \quad x_2 = \frac{1}{\sqrt{3}}, \quad w_2 = 1.$$

En resumen, la regla de cuadratura

$$\int_{-1}^1 f(x) dx \simeq f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) \quad (7.12)$$

es exacta para polinomios de grado menor o igual a 3 lo que, en términos de la Definición 7.1.3, implica que es de orden mayor o igual a 4. Notar que solamente usa dos nodos. \triangle

Ejercicio 7.2. Comprobar que la regla (7.12) es de grado igual a 4.

Observación 7.4.1. El Ejemplo 7.4.1 es un caso pequeño y que podemos tratar a mano. Para problemas más grandes, podemos utilizar métodos iterativos para resolver numéricamente los sistemas de ecuaciones resultantes, como los que discutimos en la Sección 4.8. La cuadratura gaussiana está relacionada con la teoría de polinomios ortogonales y existen algoritmos que explotan este vínculo para computar pesos y nodos en forma eficiente y estable. Al lector interesado en el tema, le referimos a [QSS10, Capítulo 10] para más detalles. \triangle

Observación 7.4.2 (otros intervalos). Las cuadraturas gaussianas se suelen expresar sobre el intervalo $[-1, 1]$. Cualquier integral sobre un intervalo $[a, b]$ se la puede llevar a una integral sobre el intervalo $[-1, 1]$ mediante un cambio de variables afín, y los pesos y nodos cambian correspondientemente, como se discutirá en el práctico. \triangle

Observación 7.4.3 (pesos son positivos). Desde el punto de vista computacional, una propiedad importante de la cuadratura gaussiana es que, en general, los pesos que se obtienen son positivos. Se puede encontrar una demostración en [SM03, Sección 10.2]. \triangle

Una variante que es relevante en algunas aplicaciones es considerar reglas de cuadratura para integrales del tipo

$$\int_{-1}^1 f(x) \rho(x) dx,$$

³Formalmente, también existe la solución $x_1 = \frac{1}{\sqrt{3}}$, $w_1 = 1$, $x_2 = -\frac{1}{\sqrt{3}}$, $w_2 = 1$, pero es claro que esta solución da lugar a la misma regla de cuadratura que la que escribimos arriba.

donde $\rho: [-1, 1] \rightarrow \mathbb{R}$ es una función no negativa y tal que $\rho(x) > 0$ en todo el intervalo $[-1, 1]$ excepto en un conjunto finito de puntos, y cumple $\int_{-1}^1 p(x) \rho(x) dx < \infty$ para todo polinomio p . En particular, la que hemos analizado hasta ahora corresponde a $\rho \equiv 1$ y se la conoce como *cuadratura de Gauss-Legendre*. Si buscamos una regla de cuadratura gaussiana con $n + 1$ puntos y escribimos el sistema de ecuaciones resultante,

$$\left\{ \begin{array}{l} I_0(\rho, f) := \int_{-1}^1 1 \rho(x) dx = \sum_{i=0}^n w_i \\ I_1(\rho, f) := \int_{-1}^1 x \rho(x) dx = \sum_{i=0}^n w_i x_i \\ \vdots \\ I_{2n+1}(\rho, f) := \int_{-1}^1 x^{2n+1} \rho(x) dx = \sum_{i=0}^n w_i x_i^{2n+1} \end{array} \right.,$$

observamos que el problema se reduce a resolver un sistema de la forma

$$\mathbf{f}(x_0, \dots, x_n, w_0, \dots, w_n) = [I_0(\rho, f), \dots, I_{2n+1}(\rho, f)]^t,$$

donde $\mathbf{f}: \mathbb{R}^{2n+2} \rightarrow \mathbb{R}^{2n+2}$ es tal que $\mathbf{f}_j(x_0, \dots, x_n, w_0, \dots, w_n) = \sum_{i=0}^n w_i x_i^{j-1}$ para $j = 1, \dots, 2n + 2$. Notamos que la función \mathbf{f} no depende de ρ , por lo que una vez que se tiene una implementación para hallar los nodos y los pesos para una cierta ρ , resulta muy simple adaptar la implementación para otra función ρ .

Ejemplo 7.4.2. Sea $a > 0$. Buscamos diseñar una regla de cuadratura gaussiana de dos puntos tal que

$$\int_0^a f(x) e^{x/a} dx \simeq w_1 f(x_1) + w_2 f(x_2) \quad (7.13)$$

sea exacta para polinomios de grado menor o igual a 3. Comenzamos asumiendo que $a = 1$, y para resolver el problema en el intervalo $[0, 1]$, procedemos como en el Ejemplo 7.4.1, imponiendo que la regla sea exacta en los monomios $\{1, x, x^2, x^3\}$. Esto nos conduce al sistema de ecuaciones

$$\begin{aligned} e - 1 &= \int_0^1 e^x dx = w_1 + w_2, \\ 1 &= \int_0^1 x e^x dx = w_1 x_1 + w_2 x_2, \\ e - 2 &= \int_0^1 x^2 e^x dx = w_1 x_1^2 + w_2 x_2^2, \\ 6 - 2e &= \int_0^1 x^3 e^x dx = w_1 x_1^3 + w_2 x_2^3. \end{aligned} \quad (7.14)$$

△

Si bien los términos del lado derecho en (7.14) son iguales a los de (7.11), la resolución de este sistema parece más compleja, y no parece claro que los nodos y pesos se deban distribuir simétricamente como en el Ejemplo 7.4.1. Implementamos en Octave el método de Newton-Raphson que describimos en la Sección 4.8.2, y obtenemos los valores numéricicos

$$\hat{x}_1 \simeq 0,2476, \quad \hat{x}_2 \simeq 0,8192, \quad \hat{w}_1 \simeq 0,7131, \quad \hat{w}_2 \simeq 1,0051.$$

Usamos el símbolo de tilde $\hat{\cdot}$ para denotar que estos nodos y pesos son los correspondientes al intervalo $[0, 1]$. Si queremos una regla para (7.13) en el intervalo $[0, a]$, simplemente hacemos el cambio de variables $\hat{x} = x/a$ y aplicamos la regla de cuadratura hallada a la función $\hat{f}(\hat{x}) := f(a\hat{x})$,

$$\begin{aligned}\int_0^a f(x) e^{x/a} dx &= a \int_0^1 f(a\hat{x}) e^{\hat{x}} d\hat{x} = a \int_0^1 \hat{f}(\hat{x}) e^{\hat{x}} d\hat{x} \\ &\simeq a \left(\hat{w}_1 \hat{f}(\hat{x}_1) + \hat{w}_2 \hat{f}(\hat{x}_2) \right) = a \hat{w}_1 f(a\hat{x}_1) + a \hat{w}_2 f(a\hat{x}_2).\end{aligned}$$

Esto muestra que los nodos y pesos en (7.13) son

$$x_1 = a\hat{x}_1, \quad x_2 = a\hat{x}_2, \quad w_1 = a\hat{w}_1, \quad w_2 = a\hat{w}_2.$$

Capítulo 8

Ecuaciones diferenciales ordinarias

8.1. Introducción. Nociones generales.

Las ecuaciones diferenciales ordinarias aparecen en modelos en diversas áreas de la ciencia, ingeniería, y economía. En concreto, las ecuaciones diferenciales ofrecen un lenguaje matemático para expresar un cambio *continuo*. Sin embargo, muchas veces ocurre que este tipo de ecuaciones tienen soluciones que no se pueden expresar en forma cerrada. Los métodos numéricos ofrecen una alternativa ampliamente usada para hallar soluciones aproximadas: algunas preguntas relevantes aquí son cómo diseñar métodos eficientes y precisos, cómo estimar qué tan lejos está la solución numérica de la solución exacta, o de qué forma lograr que el método preserve alguna propiedad de interés que pueda tener el modelo.

Una **ecuación diferencial ordinaria** es una ecuación cuya incógnita es una función $\mathbf{y}(t)$ en la que aparecen derivadas de dicha función incógnita. Por comodidad, nos vamos a referir a la variable independiente t como el *tiempo*. Una ecuación diferencial *de primer orden* es una ecuación de la forma

$$\mathbf{F}(t, \mathbf{y}(t), \mathbf{y}'(t)) = 0, \quad (8.1)$$

donde $\mathbf{y}: \mathbb{R} \rightarrow \mathbb{R}^n$ es la función incógnita, y $\mathbf{F}: \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ es un dato¹. La forma (8.1) es bastante general, y nos permitiría escribir ecuaciones muy complicadas y no lineales en \mathbf{y}' , como por ejemplo (para $n = 1$)

$$\operatorname{sen}(ty'(t)) + y(t)e^{y'(t)} = t, \quad \text{o} \quad \sqrt{1 + |y'(t)|^2} = y(t).$$

En este capítulo, vamos a trabajar con ecuaciones de la forma

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad (8.2)$$

¹Puede ocurrir que las soluciones $\mathbf{y}(t)$ que estemos buscando solamente estén definidas en un intervalo $I \subset \mathbb{R}$, y obviamente el dominio de la función F no tiene por qué ser todo $\mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n$. A los efectos de este curso, no nos vamos a detener en este aspecto, que se estudia con mayor detalle en el curso de Ecuaciones Diferenciales.

y supondremos conocida $\mathbf{f}: \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$. Observemos que para $n = 1$ esto nos da una única ecuación escalar, mientras que para $n > 1$ obtenemos un sistema de ecuaciones.

Remarcamos que asumir que nuestra ecuación diferencial es de primer orden no es algo tan restrictivo. Por ejemplo, si tenemos una ecuación diferencial de orden n de la forma

$$y^{(n)}(t) = f(t, y(t), y'(t), \dots, y^{(n-1)}(t)), \quad (8.3)$$

entonces podemos definir las n variables auxiliares $\mathbf{y}_1(t) = y(t)$, $\mathbf{y}_2(t) = y'(t) = \mathbf{y}'_1(t)$, \dots , $\mathbf{y}_n(t) = \mathbf{y}'_{n-1}(t)$, y la ecuación (8.3) puede escribirse como una ecuación de primer orden para la función vectorial $\mathbf{y}: \mathbb{R} \rightarrow \mathbb{R}^n$, de la forma

$$\mathbf{y}'(t) = \begin{bmatrix} \mathbf{y}'_1(t) \\ \mathbf{y}'_2(t) \\ \vdots \\ \mathbf{y}'_n(t) \end{bmatrix} = \begin{bmatrix} \mathbf{y}_2(t) \\ \mathbf{y}_3(t) \\ \vdots \\ f(t, \mathbf{y}_1(t), \mathbf{y}_2(t), \dots, \mathbf{y}_{(n-1)}(t)) \end{bmatrix} =: \mathbf{f}(t, \mathbf{y}(t)).$$

En el Ejemplo 8.1.3 aplicamos esta idea a una ecuación diferencial de segundo orden proveniente de aplicar las leyes de Newton.

A continuación mostramos algunos ejemplos relevantes de ecuaciones diferenciales.

Ejemplo 8.1.1 (problema test). Consideremos $f(t, y(t)) = \lambda y(t)$ en (8.2), donde $\lambda \in \mathbb{R}$ es un dato. Nuestra ecuación diferencial tiene entonces la forma $y'(t) = \lambda y(t)$. Es sencillo verificar que, para todo $C \in \mathbb{R}$, la función

$$y(t) = C e^{\lambda t}$$

satisface la ecuación (8.2). \triangle

Ejemplo 8.1.2 (ecuación logística). La llamada **ecuación logística** se utiliza en biología como un modelo simple de crecimiento de poblaciones. En ella, la incógnita es una función $y: \mathbb{R} \rightarrow \mathbb{R}$ que cumple

$$y'(t) = ay(t)(1 - y(t)), \quad \text{donde } a > 0.$$

Si pensamos en $y(t)$ como la proporción (respecto a una cierta población máxima) de un cierto animal ocupando el hábitat en un tiempo t , entonces $1 - y(t)$ representa la “capacidad ociosa” de ese hábitat. Cuando $y \approx 0$ tenemos $y' \approx ay$ y esperamos que la solución muestre un crecimiento exponencial como en el Ejemplo 8.1.1. Esto quiere decir que si hay abundancia de recursos disponibles en el hábitat, la población tiene la capacidad de crecer exponencialmente. En cambio, una vez que $y \approx 1$, la ecuación se vuelve $y' \approx 1 - y$ y la velocidad de crecimiento de la población se hace cada vez menor a medida que se va saturando la capacidad del hábitat, esto es, cuando $y \rightarrow 1^-$. \triangle

Ejemplo 8.1.3 (sistema masa-resorte). Consideremos un sistema masa-resorte sin rozamiento, como en la Figura 8.1. Sean $m > 0$ la masa, $k > 0$ la constante elástica del resorte,

y $x(t)$ el desplazamiento de la masa respecto a la posición de equilibrio en el instante t . Asumimos que la fuerza que ejerce el resorte sobre la masa es proporcional a qué tan estirado/comprimido esté², por lo que las leyes de Newton nos dicen que $x(t)$ satisface la ecuación diferencial

$$mx''(t) = -kx(t). \quad (8.4)$$

Notemos que esta no es una ecuación como (8.2), sino de *segundo orden*, porque aparecen

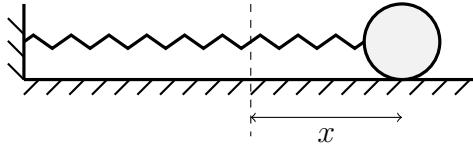


Figura 8.1: Sistema masa-resorte.

las derivadas segundas de x . Sin embargo, podemos escribirla en la forma (8.2) de un modo sencillo. Consideremos la *velocidad horizontal* de la masa, $y(t) := x'(t)$. Como $y'(t) = x''(t)$, podemos reescribir (8.4) usando ambas variables (desplazamiento y velocidad) como

$$\begin{cases} x'(t) = y(t), \\ y'(t) = -\frac{k}{m}x(t). \end{cases}$$

Luego, podemos definir una función incógnita vectorial $\mathbf{y}: \mathbb{R} \rightarrow \mathbb{R}^2$, $\mathbf{y}(t) := \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ y escribir nuestra ecuación diferencial como

$$\mathbf{y}'(t) = A\mathbf{y}(t), \quad \text{donde } A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{bmatrix}.$$

Esta ecuación sí es de la forma (8.2), con $n = 2$ y $f(t, \mathbf{y}(t)) = A\mathbf{y}(t)$.

Por otra parte, podemos incorporar una fuerza de rozamiento en nuestro sistema. Incluimos la fuerza de fricción de la masa con el piso, que suponemos de magnitud proporcional a la velocidad de la masa y sentido opuesto a ésta. Con esta adición en nuestro modelo, (8.4) toma la forma

$$mx''(t) = -kx(t) - \mu x'(t), \quad \text{con } \mu > 0.$$

Si volvemos a definir $y(t) = x'(t)$ e $\mathbf{y}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$, esta última ecuación es de la forma

$$\mathbf{y}'(t) = A\mathbf{y}(t), \quad \text{donde } A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{\mu}{m} \end{bmatrix}.$$

△

²Esta es la llamada [ley de Hooke](#).

8.1.1. Problemas de valores iniciales.

Una ecuación diferencial $\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t))$ no es suficiente para determinar una solución única, ya que la ecuación en sí solamente define los valores de las pendientes $\mathbf{y}'(t)$ de la solución. En general, hay una cantidad infinita de funciones $\mathbf{y}(t)$ que satisfacen la ecuación. Para poder distinguir una en particular de las demás, y así tener un problema bien definido, necesitamos especificar el valor de la solución en un determinado punto. Se suele llamar a ese punto t_0 y denotar por \mathbf{y}_0 al valor de la solución en ese punto. Así, se complementa a (8.2) con el dato

$$\mathbf{y}(t_0) := \mathbf{y}_0, \quad (8.5)$$

donde t_0 e \mathbf{y}_0 son asumidos conocidos. Bajo ciertas hipótesis bastante generales (ver el Teorema 8.1.1 abajo), esta condición permite asegurar la existencia de una única solución a nuestro problema localmente en el tiempo. Como la variable t muchas veces representa un tiempo, se suele pensar en t_0 como un tiempo inicial, y a (8.5) como una **condición inicial** en la solución.

Definición 8.1.1 (problema de valores iniciales). Llamamos un **problema de valores iniciales** (o **problema de Cauchy**) al problema definido por (8.2) y (8.5). Esto es, dados $\mathbf{f}: \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$, $t_0 \in \mathbb{R}$, e $\mathbf{y}_0 \in \mathbb{R}^n$, un problema de valores iniciales consiste en hallar una función $\mathbf{y} : I \rightarrow \mathbb{R}^n$, donde I es un intervalo de la forma $I = [t_0, T_{\max})$ o $I = [t_0, \infty)$, tal que

$$\begin{cases} \mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)), \\ \mathbf{y}(t_0) = \mathbf{y}_0. \end{cases} \quad (8.6)$$

△

Ejemplo 8.1.4 (problema de valores iniciales). Volvemos al Ejemplo 8.1.1. Sabemos que las funciones que cumplen la ecuación diferencial $y'(t) = ay(t)$ son de la forma

$$y(t) = Ce^{at}, \quad \text{con } C \in \mathbb{R}.$$

Si imponemos además que $y(t_0) = y_0$, con t_0 e y_0 dados, entonces debe cumplirse

$$y_0 = y(t_0) = Ce^{at_0} \Rightarrow C = y_0 e^{-at_0},$$

y encontramos la solución

$$y(t) = y_0 e^{a(t-t_0)}.$$

△

El siguiente resultado, que se demuestra en el curso de Ecuaciones Diferenciales, nos da una condición suficiente para asegurar la existencia y unicidad de soluciones localmente en el tiempo.

Teorema 8.1.1 (Picard). *Sea $D \subset \mathbb{R} \times \mathbb{R}^n$ un conjunto abierto que contiene a (t_0, \mathbf{y}_0) . Si $\mathbf{f}: D \rightarrow \mathbb{R}^n$ es una función continua y Lipschitziana con respecto a \mathbf{y} , esto es, existe $L > 0$ tal que*

$$\|\mathbf{f}(t, \mathbf{y}_1) - \mathbf{f}(t, \mathbf{y}_2)\| \leq L\|\mathbf{y}_1 - \mathbf{y}_2\| \quad \forall(t, \mathbf{y}_1), (t, \mathbf{y}_2) \in D, \quad (8.7)$$

entonces existe $\epsilon > 0$ tal que el problema (8.6) tiene una solución única en $I := [t_0, t_0 + \epsilon]$.

Ejemplo 8.1.5 (necesidad de la hipótesis de ser Lipschitziana). Consideremos el problema de valores iniciales

$$\begin{cases} y'(t) = \sqrt{|y(t)|}, \\ y(0) = 0. \end{cases}$$

Es sencillo verificar que las funciones

$$y_1(t) \equiv 0 \quad \text{e} \quad y_2(t) = \begin{cases} 0 & \text{si } t \leq 0 \\ \frac{t^2}{4} & \text{si } t > 0 \end{cases}$$

son soluciones del problema, y por lo tanto no tenemos unicidad. Esto se justifica porque la función $f(t, y) = \sqrt{|y|}$ no está en las hipótesis del Teorema de Picard. \triangle

En lo que sigue, vamos a asumir que los problemas que estamos tratando están en las hipótesis del Teorema de Picard. Vamos a asumir que nuestros problemas tienen solución única y que ésta está definida en algún intervalo I conocido.

En la teoría de las ecuaciones diferenciales, se suele usar la palabra *estabilidad* para indicar un concepto para el que en nuestro curso hemos empleado el término *condicionamiento*: informalmente, esta noción corresponde a que pequeños cambios en el valor inicial \mathbf{y}_0 induzcan pequeños cambios en los valores de $\mathbf{y}(t)$ para todo $t \geq t_0$.

Definición 8.1.2 (estabilidad de una solución de un problema de valores iniciales). La solución $\mathbf{y}(t)$ del problema de valores iniciales (8.6) se dice **estable** (en el futuro) si para todo $\epsilon > 0$ existe $\delta > 0$ tal que si $\hat{\mathbf{y}}(t)$ satisface la misma ecuación diferencial pero con condición inicial $\hat{\mathbf{y}}(t_0) = \hat{\mathbf{y}}_0$ con $\|\hat{\mathbf{y}}_0 - \mathbf{y}_0\| \leq \delta$, entonces

$$\|\hat{\mathbf{y}}(t) - \mathbf{y}(t)\| \leq \epsilon \quad \text{para todo } t > t_0.$$

\triangle

Esta definición de estabilidad refiere a la solución exacta de un problema de valores iniciales dado y nos dice que si una solución es estable y perturbamos un poco la condición inicial, la nueva solución que obtenemos permanece cerca de la solución original. En nuestro contexto, esta noción es importante porque nos da la pauta de qué podemos esperar respecto a los errores de redondeo: si la solución que queremos aproximar no es estable, ya cualquier error de representación que tengamos en el valor inicial será propagado al avanzar t , incluso si somos capaces de resolver la ecuación diferencial exactamente. A partir de

nuestra experiencia en este curso y en particular del Capítulo 1, sabemos que los errores de redondeo son una parte de la historia, y que otra parte fundamental son los errores de truncamiento, asociados a la aproximación de la ecuación diferencial mediante métodos numéricos. En la Sección 8.3 damos nociones de *estabilidad de métodos numéricos* para la resolución de problemas de valores iniciales, que contemplan el comportamiento de los errores de truncamiento.

Comencemos nuestro análisis de métodos numéricos para la aproximación de soluciones. Consideremos un problema de valores iniciales de la forma (8.6),

$$\begin{cases} \mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)) \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{cases}.$$

Nuestro objetivo es encontrar una secuencia $\{(t_k, \mathbf{y}_k)\}_{k \geq 1}$ (que puede ser finita o infinita, dependiendo de si en (8.6) es $I = [t_0, T_{\max}]$ o $I = [t_0, \infty)$) de forma tal que $\mathbf{y}_k \approx \mathbf{y}(t_k)$ para todo k . Los **pasos** $h_k := t_{k+1} - t_k$ pueden ser elegidos a priori o ser determinados a partir de ciertos requisitos de tolerancia en la solución computada.

El teorema fundamental del cálculo muestra que, si \mathbf{y} es solución al problema de valores iniciales (8.6), entonces debe satisfacer

$$\mathbf{y}(t+h) = \mathbf{y}(t) + \int_t^{t+h} f(s, \mathbf{y}(s)) ds.$$

Para el caso especial en el que f no depende de \mathbf{y} , la expresión (8.1.1) nos permite escribir

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \int_{t_n}^{t_{n+1}} f(s) ds,$$

y por lo tanto, obtenemos los valores de $\{\mathbf{y}_k\}$ resolviendo por cuadraturas mediante cualquiera de los métodos vistos en el Capítulo 7. Sin embargo, no podemos usar cuadraturas directamente para resolver este problema si la función f depende de \mathbf{y} , dado que no conocemos la función $\mathbf{y}(s)$ y, por lo tanto, no podemos evaluar el integrando. Para determinar $\{\mathbf{y}_k\}$ en el caso general, se reemplaza la ecuación diferencial por una o varias ecuaciones algebraicas. La diferencia entre los diferentes métodos de resolución de ecuaciones ordinarias radica precisamente en qué forma toman las ecuaciones algebraicas para determinar los iterados.

En la Sección 8.2 comenzamos el estudio teórico de los métodos numéricos para problemas de valores iniciales con los métodos más básicos, los llamados métodos de Euler. La Sección 8.3 introduce los conceptos fundamentales para el análisis de métodos numéricos: consistencia, estabilidad y convergencia. Aplicamos estos conceptos a un nuevo método, el del trapecio, en la Sección 8.4. Posteriormente, en la Sección 8.5 consideramos y mostramos cómo se construye una familia de métodos, los llamados de Runge-Kutta; estos métodos tienen ciertas características que hacen que sean la base de un buen número de los *solvers* de Matlab/Octave, y explicamos las ideas clave del funcionamiento de uno de ellos. La

Sección 8.6 muestra un sencillo y extendido procedimiento de combinación de dos métodos para producir uno nuevo, los llamados esquemas predictores-correctores. A continuación, la Sección 8.7 pone en práctica varios de los métodos trabajados anteriormente. En esa sección, explicamos el uso de algunos paquetes numéricos de Octave/Matlab, llamados *solvers*, y el manejo de los llamados eventos, que permiten detectar ciertos fenómenos de interés de forma automática. Finalmente, la Sección 8.8 trata con el concepto de rigidez de un problema de valores iniciales, que es de importancia a la hora de elegir con qué algoritmo se lo va a discretizar.

8.2. Métodos de Euler

En esta sección comenzamos el análisis teórico de métodos numéricos para tratar problemas de valores iniciales de la forma (8.6). Nuestro primer paso es estudiar dos métodos simples y clásicos, los llamados *métodos de Euler*.

Método de Euler hacia adelante

Supongamos que $\mathbf{y}(t_k) = \mathbf{y}_k$ y queremos determinar \mathbf{y}_{k+1} . Podemos hacer un desarrollo de Taylor de primer orden de la función \mathbf{y} en t_k para obtener

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k + h_k) = \mathbf{y}(t_k) + h_k \mathbf{y}'(t_k) + O(h_k^2). \quad (8.8)$$

Como (8.6) nos da $\mathbf{y}'(t_k) = \mathbf{f}(t_k, \mathbf{y}_k)$, tenemos que

$$\mathbf{y}(t_{k+1}) = \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k) + O(h_k^2).$$

El **método de Euler hacia adelante** (o **método de Euler explícito**, o incluso a veces llamado *método de Euler*, a secas) consiste en aproximar $\mathbf{y}(t_{k+1})$ descartando el término $O(h_k^2)$ arriba, esto es, tomar

$$\mathbf{y}_{k+1} := \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k). \quad (8.9)$$

Así, el método de Euler hacia adelante consiste en lo siguiente: dada la sucesión t_0, t_1, \dots , para aproximar la solución de (8.6) tomamos el valor de $\mathbf{y}_0 \in \mathbb{R}^n$ y luego aplicamos la iteración (8.9) para $k \geq 0$.

Método de Euler hacia atrás

En forma análoga a (8.8), consideremos ahora un desarrollo de Taylor de primer orden de la función \mathbf{y} en t_{k+1} , evaluado en t_k :

$$\mathbf{y}_k = \mathbf{y}(t_{k+1} - h_k) = \mathbf{y}(t_{k+1}) - h_k \mathbf{y}'(t_{k+1}) + O(h_k^2) = \mathbf{y}(t_{k+1}) - h_k \mathbf{f}(t_{k+1}, \mathbf{y}(t_{k+1})) + O(h_k^2).$$

El **método de Euler hacia atrás** (o **método de Euler implícito**) consiste en descartar el término $O(h_k^2)$ arriba y reemplazar $\mathbf{y}(t_{k+1})$ mediante el valor \mathbf{y}_{k+1} , que es el que queremos determinar. Concretamente, se define \mathbf{y}_{k+1} como solución de la ecuación

$$\mathbf{y}_k = \mathbf{y}_{k+1} - h_k f(t_{k+1}, \mathbf{y}_{k+1}).$$

Luego, para resolver el problema (8.6) numéricamente con el método de Euler hacia atrás, tomamos $\mathbf{y}_0 \in \mathbb{R}^n$ y luego realizamos la iteración

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}). \quad (8.10)$$

Observación 8.2.1 (una diferencia fundamental). Si bien los métodos (8.9) y (8.10) son en apariencia muy similares, tienen diferencias fundamentales. A primera vista, es claro que si conocemos \mathbf{y}_k entonces para determinar el iterado siguiente \mathbf{y}_{k+1} con el método de Euler hacia adelante, basta con reemplazar los valores conocidos de t_k, \mathbf{y}_k en el lado derecho de (8.9). Esto justifica la denominación de *explícito* para ese método. En cambio, para determinar \mathbf{y}_{k+1} con el método de Euler hacia atrás, nos encontramos con una ecuación que debemos resolver. Dependiendo de cómo sea la función f , es posible que debamos usar alguno de los métodos para ecuaciones no lineales que tratamos en el Capítulo 4. Así, en el método de Euler hacia atrás en general el iterado \mathbf{y}_{k+1} queda definido implícitamente.

En un método numérico para discretizar ecuaciones diferenciales ordinarias, la diferencia entre ser explícito o implícito tiene consecuencias tanto prácticas como analíticas. Desde el punto de vista práctico, en general es mucho más sencillo computar iterados con métodos explícitos, y el costo computacional de agregar una resolución de una ecuación no lineal en cada paso puede ser prohibitiva en varias aplicaciones. Sin embargo, los métodos implícitos por lo general gozan de mejores propiedades de *estabilidad* que los explícitos, lo que a su vez permite tomar pasos más largos y, en consecuencia, tomar menos pasos. Profundizamos sobre este último punto en la Sección 8.3.3. \triangle

Ejemplo 8.2.1 (métodos de Euler). Consideremos el problema de valores iniciales para la ecuación logística

$$\begin{cases} y'(t) = ay(t)(1 - y(t)), \\ y(t_0) = y_0 \in (0, 1). \end{cases} \quad (8.11)$$

Aquí, tenemos $f(t, y) = ay(1 - y)$. Como f no depende de t , se dice que la ecuación es *autónoma*. Esto implica que las soluciones no dependen de t_0 : si z satisface la misma ecuación diferencial pero $z(t_0 + \Delta t) = y_0$, entonces $z(t) = y(t + \Delta t)$ para todo $t \geq t_0$. Es decir, si se cambia el tiempo inicial, basta con trasladar la solución original acordemente.

Para este problema, el método de Euler hacia adelante (8.9) nos da la iteración

$$y_{k+1} = y_k + ay_k(1 - y_k)h_k = y_k [1 + a(1 - y_k)h_k].$$

Notamos que, en esta iteración, podemos tomar los pasos sin mayor dificultad: dado y_k , basta con evaluar el producto que tenemos del lado derecho para determinar y_{k+1} .

Por otra parte, si aplicamos el método de Euler hacia atrás (8.10) para este problema, obtenemos

$$y_{k+1} = y_k + ay_{k+1}(1 - y_{k+1})h_k. \quad (8.12)$$

Aquí observamos que no podemos evaluar el lado derecho directamente, sino que y_{k+1} está dado implícitamente. En este caso, como obtenemos una función cuadrática, podemos despejar

$$ah_k y_{k+1}^2 + (1 - ah_k)y_{k+1} - y_k = 0 \Rightarrow y_{k+1} = \frac{-(1 - ah_k) \pm \sqrt{(1 - ah_k)^2 + 4ah_k y_k}}{2ah_k}.$$

En principio, el paso y_{k+1} no está únicamente definido. ¿Cómo determinamos cuál de las dos soluciones tomar? Aquí debemos tener en cuenta el problema que estamos analizando: si en algún instante τ la solución exacta del problema valiera $y(\tau) = 0$, entonces tendríamos $y(t) = 0$ para todo $t \geq \tau$. Por lo tanto, las soluciones de nuestro problema no cambian de signo. A los efectos de nuestro método numérico, esto quiere decir que si y_k es positivo, entonces y_{k+1} también debe serlo. Si reescribimos la expresión que obtuvimos para y_{k+1} multiplicando y dividiendo por el conjugado del numerador y simplificamos, llegamos a

$$y_{k+1} = \frac{2y_k}{(1 - ah_k) \pm \sqrt{(1 - ah_k)^2 + 4ah_k y_k}}. \quad (8.13)$$

Luego, si $ah_k < 1$, la única forma de que y_{k+1} sea positivo es que la raíz cuadrada en el denominador esté sumando.

Si fijamos $a = 1$, $y_0 = 0,1$, $t_0 = 0$, la solución exacta a nuestro problema de valores iniciales es

$$y(t) := \frac{e^t}{e^t + 9}. \quad (8.14)$$

La Figura 8.2 compara esta solución en el intervalo $[0, 5]$ con las discretizaciones por los métodos de Euler hacia adelante y hacia atrás con paso constante $h_k = h = 0,1$. \triangle

Observación 8.2.2 (iteraciones). En el Ejemplo anterior, al considerar el método de Euler hacia atrás pudimos despejar y_{k+1} . Como ya remarcamos, en general no esperamos que esto sea posible y en cambio debemos buscar soluciones de una ecuación no lineal en cada paso.

Una opción popular es realizar el siguiente método iterativo. Dado \mathbf{y}_k , queremos hallar \mathbf{y}_{k+1} que sea solución de (8.10). Se define $\mathbf{y}_{k+1}^0 := \mathbf{y}_k$ y se usa la función de iteración $\mathbf{g}(\mathbf{y}) = \mathbf{y}_k + \mathbf{f}(t_{k+1}, \mathbf{y})h_k$, esto es,

$$\mathbf{y}_{k+1}^{j+1} := \mathbf{y}_k + \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}^j)h_k, \quad j \geq 0.$$

Típicamente se hacen solamente unas pocas iteraciones en j antes de dar el siguiente paso en k . Observamos que $\mathbb{J}_{\mathbf{g}}(\mathbf{y}) = h_k \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_{k+1}, \mathbf{y})$, donde $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ denota la matriz de derivadas parciales de \mathbf{f} respecto a la variable \mathbf{y} . Por lo tanto, si h_k es lo suficientemente pequeño,

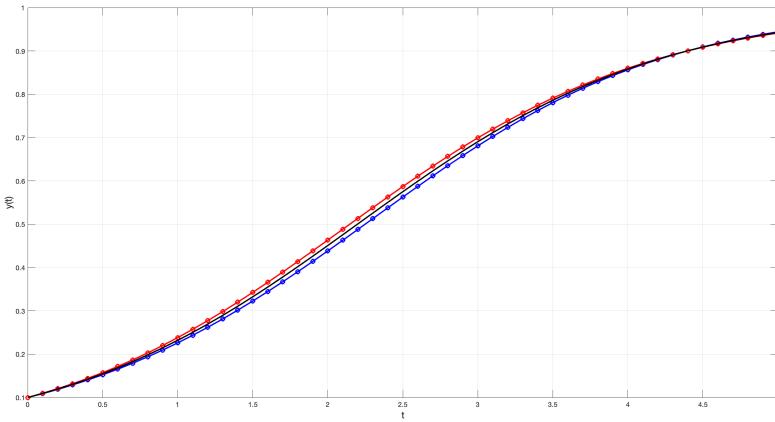


Figura 8.2: Solución a (8.11) para $a = 1$, $y_0 = 0,1$, $t_0 = 0$. En negro se representa la solución exacta, mientras que las aproximaciones por los métodos de Euler hacia adelante y hacia atrás con paso $h = 0,1$ están en azul y en rojo, respectivamente. Si bien las soluciones numéricas solamente generan puntos $\{(t_k, y_k)\}$, representamos la interpolación lineal a trozos para facilitar la visualización.

podemos garantizar que $\rho(\mathbb{J}_g(\mathbf{y})) < 1$ y tendremos que la iteración es localmente convergente. Además, a menos que \mathbf{y} varíe dramáticamente rápido, el iterado inicial \mathbf{y}_k va a estar cerca de \mathbf{y}_{k+1} .

Por ejemplo, apliquemos esta iteración a (8.12) para tomar el primer paso con los valores de parámetros $a = 1$, $y_0 = 0,1$, $t_0 = 0$, y paso constante $h = 0,1$. Nuestra iteración toma $y_1^0 = 0,1$, e

$$y_1^{j+1} = y_0 + ay_1^j(1 - y_1^j)h_1 = 0,1 + 0,1y_1^j(1 - y_1^j), \quad j \geq 0.$$

Esto da lugar a los iterados

$$y_1^1 = 0,1090, \quad y_1^2 = 0,1097119, \quad y_1^3 \approx 0,1097675, \quad y_1^4 \approx 0,1097719.$$

Para comparación, usando (8.13) obtenemos $y_1 \approx 0,1097722$. También remarcamos que el primer iterado y_1^1 es justamente el valor que toma el método de Euler hacia adelante. \triangle

8.3. Elementos de los métodos numéricos aplicados a ecuaciones diferenciales ordinarias

8.3.1. Un poco de nomenclatura

A continuación, damos dos definiciones que son útiles para distinguir ciertas características de los métodos numéricos aplicados a ecuaciones diferenciales ordinarias. Si bien hasta

aquí consideramos métodos que para generar \mathbf{y}_{k+1} solamente hacen uso del valor de \mathbf{y}_k , en principio es posible también utilizar la información respecto a iterados anteriores. Esto conduce a la primera definición.

Definición 8.3.1 (métodos de un paso y multipaso). Un método numérico para resolver el problema (8.6) se dice **de un paso** (o **de paso simple**) si, para todo k , el valor de \mathbf{y}_{k+1} solamente depende de \mathbf{y}_k y no depende de ningún otro iterado anterior. En caso de depender de (uno o más) iterados anteriores a \mathbf{y}_k , se dice que el método es **multipaso**. \triangle

La segunda definición formaliza nuestra discusión de la Observación 8.2.1.

Definición 8.3.2 (métodos implícitos y explícitos). Un método numérico para resolver el problema (8.6) se dice **explícito** si, para todo k , el valor de \mathbf{y}_{k+1} se puede computar directamente en función de (uno o varios) iterados anteriores \mathbf{y}_j , $j \leq k$. En cambio, un método se dice **implícito** si en cada paso el valor de \mathbf{y}_{k+1} queda determinado implícitamente a través de \mathbf{f} . \triangle

Observación 8.3.1 (métodos de Euler). Los métodos de Euler (8.9) y (8.10) son ambos de paso simple. El método de Euler hacia adelante es explícito y el de Euler hacia atrás es implícito, tal como adelantamos en la Observación 8.2.1. \triangle

De aquí en adelante, solamente vamos a considerar métodos de un paso. Existen varios métodos multipaso muy relevantes en la práctica, como los **métodos de Adams** (los de *Adams-Basforth* son explícitos, y los de *Adams-Moulton* son implícitos), pero su análisis requiere algunas modificaciones respecto a los métodos de paso simple. En el práctico consideramos el llamado **método leapfrog de dos pasos**,

$$\mathbf{y}_{k+1} = \mathbf{y}_{k-1} + (h_{k+1} + h_k) \mathbf{f}(t_k, \mathbf{y}_k).$$

Este es un método *multipaso* porque para determinar \mathbf{y}_{k+1} requerimos los valores de \mathbf{y}_k e \mathbf{y}_{k-1} , y es un método *explícito* porque para calcular \mathbf{y}_{k+1} basta con reemplazar con los valores de \mathbf{y}_k e \mathbf{y}_{k-1} y operar.

8.3.2. Consistencia

Al analizar qué tan cerca está la solución que obtenemos mediante un método numérico de la solución exacta de un problema de valores iniciales, hay dos nociones de error relevantes. Una ecuación diferencial ordinaria admite una *familia de soluciones* y, al fijar una condición inicial –esto es, al considerar un problema de valores iniciales–, seleccionamos una de ellas. En nuestro método numérico para un problema de valores iniciales, al ir haciendo pasos, a menos que el valor que encontramos sea exacto nos iremos cambiando de solución dentro de esa familia. La Figura 8.3 ilustra este hecho: comenzando de (t_0, y_0) , que se encuentra sobre la curva solución, al tomar el primer paso obtenemos un punto (t_1, y_1) que está sobre una curva distinta de la familia de soluciones. Al tomar otro paso,

nos vamos a mover a otra curva distinta, y así sucesivamente. En el instante t_3 , nuestro método computa un valor y_3 . Nos interesa conocer la discrepancia entre y_3 e $y(t_3)$, la distancia entre los puntos en rojo y verde hacia la derecha de la figura. Como el análisis de los métodos se suele realizar considerando lo que ocurre al tomar solamente un paso, va a ser útil considerar la discrepancia entre y_3 y la solución exacta al problema que tiene la misma ecuación diferencial pero condición inicial $y(t_2) = y_2$ en el instante t_3 : tal solución exacta es lo que llamamos $u(t_3)$ y marcamos con un punto en azul en la figura.

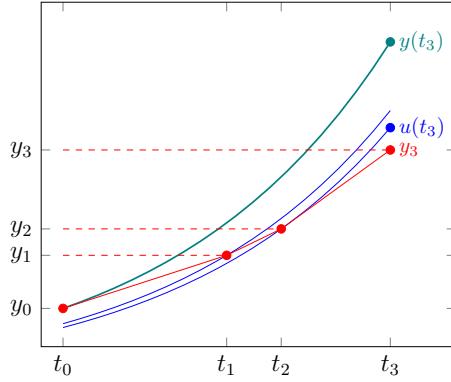


Figura 8.3: Algunos pasos en un método numérico. El error global en el paso 3 es la diferencia entre y_3 e $y(t_3)$, mientras que el error de truncamiento en dicho paso es la diferencia entre y_3 y $u(t_3)$.

Definición 8.3.3 (errores global y local). Sea $\{(t_k, \mathbf{y}_k)\}$ dado por un método numérico para aproximar el problema (8.6), cuya solución denotamos por \mathbf{y} . Los **errores globales** son $\{\mathbf{e}_k\}$ dados por

$$\mathbf{e}_k := \mathbf{y}_k - \mathbf{y}(t_k). \quad (8.15)$$

Por otra parte, los **errores locales de truncamiento** (o errores de consistencia) $\{\boldsymbol{\ell}_k\}$ se definen de la siguiente manera: $\boldsymbol{\ell}_0 = 0$ y, para todo $k \geq 0$,

$$\boldsymbol{\ell}_{k+1} := \mathbf{y}_{k+1} - \mathbf{u}(t_{k+1}), \quad (8.16)$$

donde \mathbf{u} es la solución del problema de valores iniciales

$$\begin{cases} \mathbf{u}'(t) = \mathbf{f}(t, \mathbf{u}(t)), \\ \mathbf{u}(t_k) = \mathbf{y}_k. \end{cases}$$

△

Definición 8.3.4 (consistencia). Diremos que un método numérico es **consistente** si para todo $k \geq 0$ se cumple

$$\frac{\boldsymbol{\ell}_{k+1}}{h_k} \rightarrow \mathbf{0} \quad \text{con } h_k = t_{k+1} - t_k \rightarrow 0.$$

Decimos que un método consistente es de **orden** $p \geq 1$ si, cuando la solución \mathbf{y} es de clase C^{p+1} , se tiene

$$\boldsymbol{\ell}_k = O((h_k)^{p+1}). \quad (8.17)$$

Remarcamos (recordar la Definición 0.0.1) que esto significa que existe una constante $C > 0$, que no depende de h_k , tal que

$$\|\boldsymbol{\ell}_k\| \leq C(h_k)^{p+1}.$$

La elección de qué norma usar aquí es irrelevante porque todas las normas en \mathbb{R}^n son equivalentes. \triangle

Observación 8.3.2 (no es un error de tipeo). Puede parecer llamativo que, en la definición anterior, para referirnos a un método de orden p , en la fórmula (8.17) aparece un exponente $p + 1$. La razón es que para definir el orden de un método usamos los errores *locales* de truncamiento. La cantidad que nos interesa es en realidad el error *global* y, en el pasaje de *local* a *global* tendremos un efecto de acumulación de error que, en el mejor de los casos, nos hace perder un orden (ver (8.20) más abajo). \triangle

Ejemplo 8.3.1 (los métodos de Euler son de primer orden). Analicemos los errores de consistencia para los métodos de Euler explícito e implícito. Para ello, dado \mathbf{y}_k , debemos estudiar los errores de truncamiento dados por (8.16).

Para el método de Euler hacia adelante (8.9) tenemos

$$\mathbf{y}_{k+1} = \mathbf{y}_{k+1} - \mathbf{u}(t_{k+1}) = \mathbf{y}_k + \mathbf{f}(t_k, \mathbf{y}_k)h_k - \mathbf{u}(t_{k+1}).$$

Esta diferencia es justamente lo que estimamos en (8.8) haciendo un desarrollo de Taylor: tenemos

$$\mathbf{u}(t_{k+1}) = \mathbf{y}_k + h_k \mathbf{y}'(t_k) + O(h_k^2) = \mathbf{y}_k + \mathbf{f}(t_k, \mathbf{y}_k)h_k + O(h_k^2),$$

porque $\mathbf{u}'(t_k) = \mathbf{f}(t_k, \mathbf{y}_k) = \mathbf{y}'(t_k)$. Por lo tanto, $\boldsymbol{\ell}_{k+1} = O(h_k^2)$ y deducimos que el método de Euler hacia adelante es consistente y es de primer orden.

Un razonamiento similar permite probar que el método de Euler hacia atrás es de primer orden, y queda como ejercicio en el práctico. \triangle

8.3.3. Cero-estabilidad

La noción de estabilidad de un método numérico para resolver una ecuación diferencial ordinaria es análoga a la de estabilidad de soluciones: queremos que pequeñas perturbaciones en un determinado instante no generen que la solución que obtengamos crezca indefinidamente en tiempos posteriores.

El concepto de *cero-estabilidad* garantiza que en un intervalo acotado fijo pequeñas perturbaciones de los datos producen perturbaciones acotadas de la solución numérica, cuando la longitud del paso máximo tiende a 0. El requerimiento de estabilidad de un método

aparece, en primer lugar, de la necesidad de mantener bajo control los errores introducidos al estar utilizando aritmética de punto flotante. En efecto, si un método numérico no es cero-estable, los errores de redondeo tanto al tomar \mathbf{y}_0 como en el proceso de evaluar \mathbf{f} harían que la solución computada fuera completamente inútil.

Definición 8.3.5 (cero-estabilidad). Consideremos un método numérico de un paso que discretiza la ecuación diferencial $\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t))$. Decimos que el método es **cero-estable** si, para todo $T > t_0$, si tomamos $t_0 < t_1 < \dots < t_K = T$ arbitrarios y consideramos $\{(t_k, \mathbf{y}_k)\}_{k=1,\dots,K}$, $\{(t_k, \mathbf{z}_k)\}_{k=1,\dots,K}$ las soluciones numéricas a problemas de valores iniciales con condiciones iniciales \mathbf{y}_0 y \mathbf{z}_0 respectivamente, entonces existen $C > 0$, $H > 0$ tal que si $h_k := t_{k+1} - t_k < H \forall k = 0, \dots, K$, entonces

$$\|\mathbf{y}_k - \mathbf{z}_k\| \leq C\|\mathbf{y}_0 - \mathbf{z}_0\| \quad \forall k = 0, \dots, K.$$

△

El nombre de *cero-estabilidad* proviene del hecho de que la definición anterior no depende del problema de valores iniciales considerado, y basta con verificar la definición en el sencillo problema $y'(t) = 0$. No vamos a profundizar en este concepto, pero señalamos el siguiente resultado [QSS10, Sección 11.3].

Proposición 8.3.1. *Consideremos el problema de valores iniciales (8.6) sobre un intervalo de la forma $I = [t_0, T_{\max}]$, con \mathbf{f} en las hipótesis del Teorema de Picard 8.1.1. Entonces, todo método de un paso para resolver dicho problema que sea consistente también es cero-estable.*

8.3.4. Convergencia

La pregunta de convergencia refiere a qué ocurre cuando en nuestro método numérico hacemos que la longitud de *todos* los pasos tienda a 0. Típicamente, en la práctica, incluso cuando el problema (8.6) esté definido en un intervalo no acotado $I = [t_0, \infty)$, al resolverlo numéricamente y hacer que la longitud máxima del paso tienda a 0 nos interesa controlar los errores hasta algún tiempo máximo finito. Por lo tanto, en esta sección vamos a trabajar con problemas sobre intervalos finitos $I = [t_0, T_{\max}]$.

Definición 8.3.6 (convergencia). Diremos que un método numérico para resolver el problema (8.6) es **convergente** si, para todo $T \in (t_0, T_{\max})$ fijo, si tomamos $t_0 < t_1 < \dots < t_K = T$ arbitrarios y llamamos $h := \max_{k=0,\dots,K-1} h_k = \max_{k=0,\dots,K-1} t_{k+1} - t_k$, tenemos

$$\lim_{h \rightarrow 0} \mathbf{e}_K = \lim_{h \rightarrow 0} \mathbf{y}_K - \mathbf{y}(t_K) = \mathbf{0}.$$

En palabras, la convergencia de un método quiere decir que, para todo tiempo fijo T , al hacer que los largos del paso tiendan a 0, se tiene que el valor computado por el método en tiempo T converge al valor de la solución en dicho tiempo. △

Un resultado central en la teoría de métodos numéricos para aproximar ecuaciones diferenciales es el siguiente.

Teorema 8.3.2 (de equivalencia de Lax-Richtmyer). *Todo método consistente es convergente si y sólo si es cero-estable.*

El teorema anterior tiene un significado preciso en el contexto que hemos discutido hasta ahora, pero su significado “filosófico” también es válido en otros contextos. Muchas veces, la *convergencia* de métodos numéricos (que el límite de lo que computamos sea la función buscada) es equivalente a alguna forma de *consistencia* (que las ecuaciones que se quiere aproximar se verifiquen en el límite en que los parámetros de discretización tienden a cero) más alguna forma de *estabilidad* (que el método no magnifique descontroladamente los errores de redondeo y de aproximación numérica).

Recordamos que los métodos de un paso consistentes son cero-estables, por lo que el Teorema 8.3.2 establece su convergencia. Podemos hacer más preciso el contenido de ese teorema en ese caso, vinculando los errores de truncamiento y globales.

Teorema 8.3.3 (errores locales y globales). *Si tenemos un método de un paso consistente $\{(t_k, \mathbf{y}_k)\}_{k=0, \dots, K}$ para el problema de valores iniciales (8.6) sobre un intervalo de la forma $I = [t_0, T_{\max}]$, con \mathbf{f} en las hipótesis del Teorema de Picard 8.1.1, entonces existe una constante $C > 0$ tal que para cada $k = 0, \dots, K$ se tiene*

$$\|\mathbf{e}_k\| \leq C \sum_{j=1}^k \|\boldsymbol{\ell}_j\|. \quad (8.18)$$

La constante C aquí depende de la constante L de Lipschitz de \mathbf{f} respecto a \mathbf{y} (ver (8.7)), de $\|\mathbf{f}''\|_{L^\infty(I)}$, y de T .

La fórmula (8.18) permite deducir **órdenes de convergencia**. En efecto, consideremos un tiempo $T > t_0$, en el que asumimos que la solución del problema (8.6) está bien definida, y en el que queremos estimar la discrepancia entre la solución y nuestras aproximaciones numéricas con largos de paso tendiendo a 0.

Usamos el Teorema 8.3.2 sobre el intervalo $I = [t_0, T]$ con un método que produce $\{(t_k, \mathbf{y}_k)\}_{k=0, \dots, K}$; notar que $t_K = T$. Si el método que estamos considerando es consistente de orden p , entonces tenemos $\|\boldsymbol{\ell}_j\| \leq \tilde{C} h_{j-1}^{p+1}$ para todo $j = 1, \dots, K$ para alguna constante fija $\tilde{C} > 0$. Si reemplazamos en (8.18) y combinamos las constantes que aparecen, tenemos

$$\|\mathbf{e}_K\| = \|\mathbf{y}_K - \mathbf{y}(T)\| \leq C \sum_{j=1}^K \|\boldsymbol{\ell}_j\| \leq C \sum_{j=1}^K h_{j-1}^{p+1}. \quad (8.19)$$

Tal como hicimos en la Definición 8.3.6, denotamos $h := \max_{k=0, \dots, K-1} h_k = \max_{k=0, \dots, K-1} t_{k+1} - t_k$ al paso máximo. Con esta notación, llegamos a que el error en

el tiempo T se puede acotar por

$$\|\mathbf{e}_K\| \leq C \sum_{j=1}^K h_{j-1}^{p+1} \leq C \left(\sum_{j=1}^K h_{j-1} \right) h^p = C(T - t_0) h^p. \quad (8.20)$$

Concluimos que **si el método es consistente de orden p , entonces el error en T es $O(h^p)$.**

8.3.5. Estabilidad absoluta

La cero-estabilidad trata con lo que ocurre con las soluciones numéricas en intervalos de tiempo acotados al hacer que el paso máximo $h \rightarrow 0$. Aquí consideramos una pregunta análoga pero con los roles del intervalo del tiempo y del paso temporal invertidos: ¿qué ocurre computacionalmente al resolver (8.6) sobre $I = [t_0, T_{\max}]$ cuando $T_{\max} \rightarrow \infty$ y el paso máximo h está fijo? En ese caso, fórmulas como (8.20) pierden sentido: una vez que fijamos un paso máximo h , el lado derecho en (8.20) tiene una cantidad no acotada, ya que podríamos tomar T arbitrariamente grande. Por lo tanto, nos interesan los métodos que sean capaces de aproximar la solución en intervalos de tiempo arbitrariamente largos, incluso con pasos relativamente “grandes”.

Esta forma de estabilidad, llamada *estabilidad absoluta*, nos da una pauta del comportamiento práctico de los métodos: en aplicaciones, es claro que cuanto menor sea h , más pasos debemos dar y por lo tanto mayor es el costo computacional del algoritmo que usemos. Por lo tanto, es deseable que el método no requiera que los pasos sean demasiado pequeños para producir una buena aproximación a la solución del problema de valores iniciales.

Para definir la estabilidad absoluta se utiliza el problema test que introdujimos en el Ejemplo 8.1.1. En concreto, para $\lambda < 0$, consideramos el problema de valores iniciales

$$\begin{cases} y'(t) = \lambda y(t), \\ y(0) = 1. \end{cases} \quad (8.21)$$

La solución de este problema es $y(t) = e^{\lambda t}$, que cumple $y(t) \rightarrow 0$ con $t \rightarrow \infty$ porque $\lambda < 0$. Consideremos un método numérico con paso fijo $h > 0$, que produce una sucesión $\{(t_k, y_k)\}$. Deseamos que $y_k \rightarrow 0$ con $k \rightarrow \infty$: si esto ocurre, decimos que el método es **absolutamente estable**³ para ese paso.

Analicemos la estabilidad de los métodos de Euler, comenzando por el explícito. Si aplicamos (8.9) a (8.21), nos encontramos con que, para cada $k \geq 0$,

$$y_{k+1} = y_k + h\lambda y_k = (1 + h\lambda)y_k.$$

³También se puede encontrar la expresión *estable* a secas para referirse a la estabilidad absoluta.

Como $y_0 = 1$, deducimos que

$$y_k = (1 + h\lambda)^k \quad \forall k \geq 0.$$

Para tener $y_k \rightarrow 0$ con $k \rightarrow \infty$, necesitamos entonces que sea

$$|1 + h\lambda| < 1 \Leftrightarrow -1 < 1 + h\lambda < 1.$$

Como $h > 0$ y $\lambda < 0$, la segunda desigualdad de arriba se cumple trivialmente. Sin embargo, para que valga la primera es necesario que sea

$$h < \frac{2}{|\lambda|}. \quad (8.22)$$

En conclusión, el método de Euler explícito solamente es absolutamente estable si el paso h satisface la condición (8.22): se trata de un método **condicionalmente absolutamente estable**.

Consideremos ahora el método de Euler hacia atrás. Al discretizar (8.21) con dicho método, nos encontramos con que $y_0 = 1$ e

$$y_{k+1} = y_k + h\lambda y_{k+1} \Rightarrow y_{k+1} = \frac{y_k}{1 - h\lambda}.$$

Esto implica que

$$y_k = \frac{1}{(1 - h\lambda)^k} \quad \forall k \geq 0.$$

Como $1 - h\lambda > 1$, tenemos $y_k \rightarrow 0$ *independientemente* de la longitud del paso h . Esto significa que el método de Euler hacia atrás es **incondicionalmente absolutamente estable**.

Observación 8.3.3 (por qué usamos el problema test). Remarcamos que el hecho de ser o no absolutamente estable se verifica usando el problema test (8.21). Justifiquemos brevemente el uso de dicho problema para el caso escalar ($n = 1$). Si la solución $y(t)$ que estamos obteniendo es una perturbación de otra solución $\hat{y}(t)$ a la misma ecuación diferencial $y'(t) = f(t, y(t))$, entonces debemos analizar qué ocurre con su diferencia $w := y - \hat{y}$, que verifica

$$w'(t) = y'(t) - \hat{y}'(t) = f(t, y(t)) - f(t, \hat{y}(t)).$$

Si hacemos un desarrollo de Taylor para f , tenemos

$$\begin{aligned} f(t, y(t)) &= f(t, \hat{y}(t)) + f_y(t, \hat{y}(t))(y(t) - \hat{y}(t)) + O((y(t) - \hat{y}(t))^2) \\ &= f(t, \hat{y}(t)) + f_y(t, \hat{y}(t)) w(t) + O((w(t))^2), \end{aligned}$$

por lo que podemos aproximar

$$w'(t) = f(t, y(t)) - f(t, \hat{y}(t)) \approx f_y(t, \hat{y}(t)) w(t).$$

Observamos que la perturbación w satisface una ecuación como la del problema test, pero con un factor $f_y(t, \hat{y}(t))$ en lugar de λ . Nuestro análisis de estabilidad absoluta se basa en suponer que las perturbaciones son pequeñas, lo que nos permite linealizar, y en tomar $\lambda < 0$ es arbitrario, lo que nos permite “cubrirnos” de qué valores tome la derivada parcial de f respecto a y . \triangle

Observación 8.3.4 (estabilidad en el plano complejo). En la literatura, es habitual considerar $\lambda \in \mathbb{C}$ en el problema test para definir una *región de estabilidad absoluta* sobre el plano complejo. Esto se debe a que, para sistemas de ecuaciones, debemos reemplazar f_y en la observación anterior por la matriz jacobiana de \mathbf{f} respecto a \mathbf{y} , y se llega a una ecuación para perturbaciones lineales de la forma

$$\mathbf{w}'(t) \approx \mathbb{J}_{\mathbf{f}}(t, \hat{\mathbf{y}}(t)) \mathbf{w}(t).$$

Para que $\mathbf{w} \rightarrow \mathbf{0}$ con $t \rightarrow \infty$, necesitamos que el radio espectral de esta matriz jacobiana sea menor que 1. Como los valores propios de una matriz real en general son complejos, es matemáticamente más correcto permitir $\lambda \in \mathbb{C}$ en el problema test para definir una *región* de estabilidad en lugar de un intervalo.

Formalmente, para el problema test $y' = e^{\lambda y}$, la región de estabilidad es el conjunto

$$\mathcal{A} := \{z := h\lambda \in \mathbb{C} : |y_k| \rightarrow 0\}.$$

Luego, se dice que un método es **A-estable** si $\mathcal{A} \cap \mathbb{C}^- = \mathbb{C}^-$, donde $\mathbb{C}^- := \{z \in \mathbb{C} : \operatorname{Re}(z) < 0\}$ es el conjunto de números complejos con parte real negativa. \triangle

Ejemplo 8.3.2 (métodos de Euler aplicados al problema test). Tomamos $\lambda = -2$ en el problema test (8.21), e implementamos los métodos de Euler hacia adelante y hacia atrás con distintos pasos h . Notamos que, para la estabilidad del método de Euler explícito, la condición (8.22) indica que debemos tomar $h < 1$. Computamos soluciones al problema en el intervalo $I = [0, 20]$ con 50, 30, 20 y 15 pasos uniformes, lo que corresponde a $h = 2/5, 2/3, 1, 4/3$, respectivamente. La primera fila muestra las soluciones para el método de Euler explícito, mientras que la segunda muestra las soluciones para el método implícito.

Los resultados confirman nuestro análisis anterior. Para el método de Euler hacia adelante, cuando $h < 1$, las soluciones decrecen en magnitud al aumentar k ; cuando $h = 1$, la solución numérica oscila entre $y_{2k} = 1$ e $y_{2k+1} = -1$; cuando $h > 1$ la solución numérica aumenta en magnitud al aumentar k . En cambio, para el método de Euler hacia atrás, independientemente de la longitud del paso se tiene que las soluciones numéricas decrecen al aumentar k .

\triangle

Observación 8.3.5 (estabilidad de métodos explícitos). El hecho de que el método de Euler hacia adelante sea condicionalmente estable es consecuencia de un resultado más general, que dice que **ningún método explícito es incondicionalmente absolutamente estable**. En otras palabras, si un método es incondicionalmente estable, entonces es implícito. Sin embargo, no todos los métodos implícitos son incondicionalmente estables. \triangle

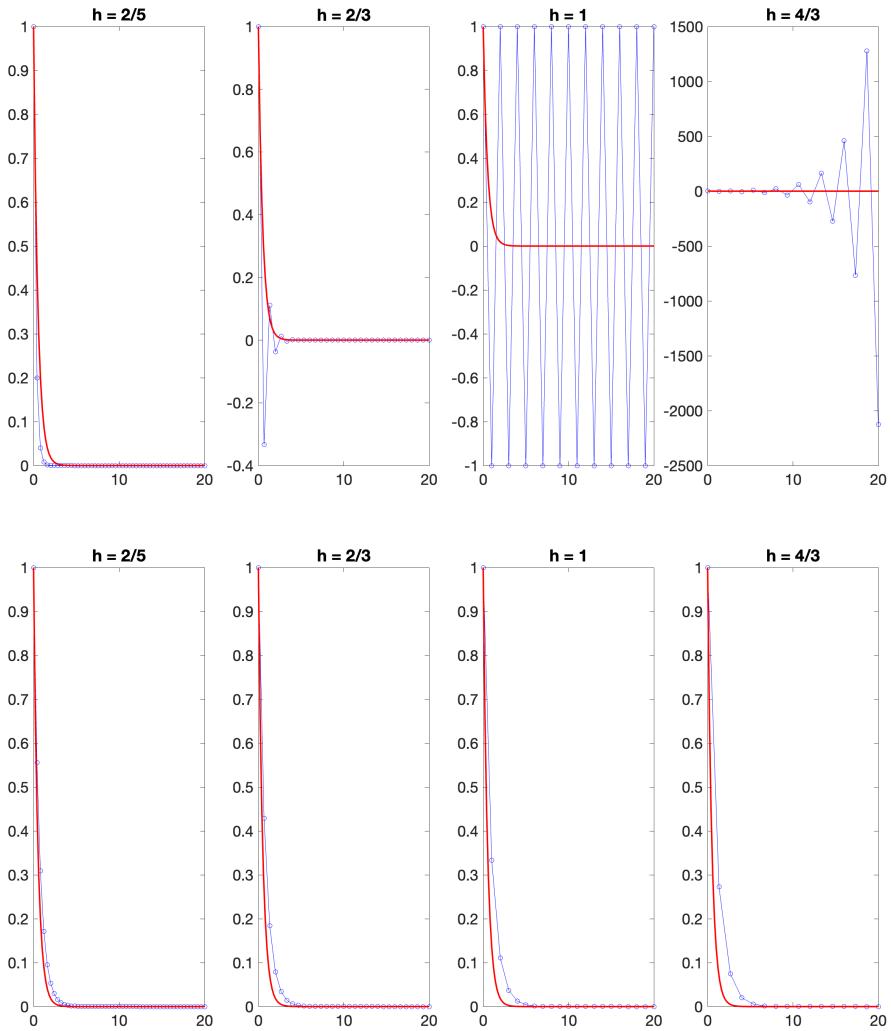


Figura 8.4: Solución a (8.21) para $\lambda = -2$ en $I = [0, 20]$. En rojo se representa la solución exacta, mientras que en azul se muestran las soluciones producidas por los métodos de Euler hacia adelante (primera fila) y de Euler hacia atrás (segunda fila). En todos los casos se tomaron pasos uniformes, con valores $h = 2/5$, $h = 2/3$, $h = 1$ y $h = 4/3$, de izquierda a derecha.

8.4. Método del trapezio

Hasta aquí solamente hemos trabajado con los métodos de Euler, que son de primer orden. En esta sección introducimos un nuevo método, el llamado **método del trapecio**⁴, que

⁴En el contexto de ecuaciones en derivadas parciales, este método también recibe el nombre de *método de Crank-Nicolson*.

resulta al combinar los métodos de Euler. En efecto, para el problema de valores iniciales (8.6), este método consiste en aproximar $\mathbf{y}(t_{k+1})$ por

$$\mathbf{y}_{k+1} := \mathbf{y}_k + h_k \left(\frac{\mathbf{f}(t_k, \mathbf{y}_k) + \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1})}{2} \right). \quad (8.23)$$

En otras palabras, los iterados del método del trapecio se calculan poniendo del lado derecho la mitad de la suma entre los lados derechos correspondientes a los iterados de los métodos de Euler hacia adelante y de Euler hacia atrás.

A partir de la definición (8.23), se infiere inmediatamente que el método del trapecio es *implícito*.

Proposición 8.4.1. *El método del trapecio es incondicionalmente absolutamente estable y es de segundo orden.*

Demostración. Comenzamos por la estabilidad. Aplicando (8.23) al problema test (8.21), tenemos $y_0 = 1$ e

$$y_{k+1} = y_k + \frac{h\lambda}{2} (\mathbf{y}_k + \mathbf{y}_{k+1}), \quad k \geq 0.$$

Esto da lugar a

$$y_k = \left(\frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} \right)^k = \left(\frac{2 + h\lambda}{2 - h\lambda} \right)^k.$$

Por lo tanto, para que el método del trapecio sea A -estable, requerimos

$$|2 + h\lambda| < |2 - h\lambda|.$$

Como $\lambda < 0$, es sencillo verificar que esta condición se satisface para todo $h > 0$.

A continuación demostramos que el método del trapecio es de segundo orden. Dado \mathbf{y}_k , sea \mathbf{u} la solución del problema

$$\begin{cases} \mathbf{u}'(t) = \mathbf{f}(t, \mathbf{u}(t)), \\ \mathbf{u}(t_k) = \mathbf{y}_k. \end{cases}$$

Queremos probar que el error de truncamiento cumple $\ell_{k+1} = \mathbf{y}_{k+1} - \mathbf{u}(t_{k+1}) = O(h_k^3)$, asumiendo que \mathbf{u} es de clase C^3 . Para ello, hacemos un desarrollo de Taylor alrededor de t_k , evaluado en t_{k+1} :

$$\begin{aligned} \mathbf{u}(t_{k+1}) &= \mathbf{u}(t_k) + h_k \mathbf{u}'(t_k) + \frac{h_k^2}{2} \mathbf{u}''(t_k) + O(h_k^3) \\ &= \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k) + \frac{h_k^2}{2} \mathbf{u}''(t_k) + O(h_k^3). \end{aligned} \quad (8.24)$$

Para tratar el término $\mathbf{u}''(t_k)$, recordamos de la Sección 1.5.3 que la aproximación por diferencias hacia adelante es de primer orden, esto es,

$$\mathbf{u}''(t_k) = \frac{\mathbf{u}'(t_{k+1}) - \mathbf{u}'(t_k)}{h_k} + O(h_k).$$

Combinando estas identidades y usando que $\mathbf{u}'(t) = \mathbf{f}(t, \mathbf{u}(t))$, obtenemos

$$\begin{aligned}\mathbf{u}(t_{k+1}) &= \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k) + \frac{h_k^2}{2} \mathbf{u}''(t_k) + O(h_k^3) \\ &= \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k) + \frac{h_k}{2} [\mathbf{f}((t_{k+1}, \mathbf{u}(t_{k+1})) - \mathbf{f}(t_k, \mathbf{y}_k)] + O(h_k^3) \\ &= \mathbf{y}_k + \frac{h_k}{2} [\mathbf{f}((t_{k+1}, \mathbf{u}(t_{k+1})) + \mathbf{f}(t_k, \mathbf{y}_k)] + O(h_k^3).\end{aligned}$$

Teniendo en cuenta la definición (8.23) de los iterados con el método del trapecio, obtenemos

$$\boldsymbol{\ell}_{k+1} = \frac{h_k}{2} [\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) - \mathbf{f}(t_{k+1}, \mathbf{u}(t_{k+1}))] + O(h_k^3). \quad (8.25)$$

Como \mathbf{f} es Lipschitziana respecto a la segunda variable, existe una constante $C > 0$ tal que

$$\|\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) - \mathbf{f}((t_{k+1}, \mathbf{u}(t_{k+1}))\| \leq C \|\mathbf{y}_{k+1} - \mathbf{u}(t_{k+1})\| = C \|\boldsymbol{\ell}_{k+1}\|.$$

Por lo tanto, tomando normas en (8.25) y usando la desigualdad triangular, deducimos

$$\|\boldsymbol{\ell}_{k+1}\| \leq C \frac{h_k}{2} \|\boldsymbol{\ell}_{k+1}\| + O(h_k^3).$$

Como queremos hacer $h_k \rightarrow 0$, podemos asumir que $h_k < 2/C$ para la constante que aparece en la desigualdad anterior, y esto implica que $\|\boldsymbol{\ell}_{k+1}\| = O(h_k^3)$. \square

8.5. Métodos de Runge-Kutta

A continuación, describimos una *familia* de métodos un poco más general que los que hemos considerado hasta el momento. Los **métodos de Runge-Kutta** son una clase de métodos de un paso que permiten alcanzar órdenes superiores al realizar *varias evaluaciones* de la función \mathbf{f} en cada intervalo $[t_k, t_{k+1}]$.

En forma general, los iterados en los métodos de Runge-Kutta se escriben como

$$\mathbf{y}_{k+1} := \mathbf{y}_k + h_k \sum_{i=1}^s b_i K_i, \quad (8.26)$$

donde

$$K_i := \mathbf{f} \left(t_k + c_i h_k, \mathbf{y}_k + h_k \sum_{j=1}^s a_{ij} K_j \right), \quad (8.27)$$

y $a_{ij}, b_i, c_i \in \mathbb{R}$ son a determinar. Al número s se le llama la cantidad de **etapas** del método.

Los números K_i corresponden a evaluaciones de \mathbf{f} en puntos “intermedios” entre (t_k, \mathbf{y}_k) y $(t_{k+1}, \mathbf{y}_{k+1})$. Observemos que si $a_{ij} = 0$ para todo $j \geq i$, entonces los valores de los K_i

se pueden determinar en forma explícita y, en consecuencia, el método de Runge-Kutta (8.26) resultante es explícito. En cambio, si hay algún valor de a_{ij} con $j \geq i$ que no sea nulo, entonces K_i queda implícitamente determinado y por lo tanto el método resultante es implícito.

Observación 8.5.1 (los métodos conocidos son de Runge-Kutta). La familia de métodos de Runge-Kutta incluye a los métodos que ya hemos discutido. Por ejemplo⁵:

- si en (8.26)–(8.27) fijamos $s = 1$, $c_1 = 0$, $a_{11} = 0$, $b_1 = 1$, obtenemos el método de Euler hacia adelante;
- si en (8.26)–(8.27) fijamos $s = 1$, $c_1 = 1$, $a_{11} = 1$, $b_1 = 1$, obtenemos el método de Euler hacia atrás;
- si en (8.26)–(8.27) fijamos $s = 2$, $c_1 = 0$, $c_2 = 1$, $a_{11} = a_{12} = 0$, $a_{21} = a_{22} = 1/2$, $b_1 = b_2 = 1/2$, obtenemos el método del trapecio.

△

8.5.1. Construcción de métodos de Runge-Kutta

Para una determinada cantidad de etapas s , se eligen valores de los coeficientes a_{ij}, b_i, c_i en (8.26) y (8.27) de modo que el error de truncamiento sea de un determinado orden; muchas veces, se busca que el método resultante sea del mayor orden posible. Para ilustrar ideas, fijemos $s = 2$ y busquemos métodos de Runge-Kutta explícitos y de segundo orden. Tales métodos se pueden escribir como

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k(b_1 K_1 + b_2 K_2),$$

donde

$$K_1 = \mathbf{f}(t_k + c_1 h_k, \mathbf{y}_k), \quad K_2 = \mathbf{f}(t_k + c_2 h_k, \mathbf{y}_k + h_k a_{21} K_1).$$

Por lo tanto, debemos hallar valores de los cinco parámetros $b_1, b_2, c_1, c_2, a_{21}$ que logren que el error de truncamiento sea $\ell_{k+1} = O(h_k^3)$. Comenzamos, como ya en este punto el lector podrá sospechar, por el desarrollo de Taylor (8.24):

$$\mathbf{u}(t_{k+1}) = \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k) + \frac{h_k^2}{2} \mathbf{u}''(t_k) + O(h_k^3).$$

Para aliviar la notación, pondremos $\mathbf{f} := \mathbf{f}(t_k, \mathbf{y}_k)$, $\mathbf{f}_t = \frac{\partial \mathbf{f}}{\partial t}(t_k, \mathbf{y}_k)$, $\mathbf{f}_y = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_k, \mathbf{y}_k)$; las dos primeras cantidades son vectoriales y la última es matricial. Como

$$\mathbf{u}''(t_k) = \frac{d}{dt} \mathbf{f}(t, \mathbf{y}(t)) \Big|_{t=t_k} = \mathbf{f}_t + \mathbf{f}_y \mathbf{y}'(t_k) = \mathbf{f}_t + \mathbf{f}_y \mathbf{f},$$

⁵El primer punto es elemental de verificar. La verificación de los dos últimos no es tan directa como parece.

podemos escribir

$$\mathbf{u}(t_{k+1}) = \mathbf{y}_k + h_k \mathbf{f} + \frac{h_k^2}{2} \mathbf{f}_t + \frac{h_k^2}{2} \mathbf{f}_y \mathbf{f} + O(h_k^3). \quad (8.28)$$

Dejamos esta fórmula en pausa y buscamos una expresión análoga para el paso del método de Runge-Kutta. Para simplificar el análisis, asumimos que $c_1 = 0$, lo que implica $K_1 = \mathbf{f}$. Además, un desarrollo de Taylor para \mathbf{f} nos da

$$K_2 = \mathbf{f}(t_k + c_2 h_k, \mathbf{y}_k + h_k a_{21} K_1) = \mathbf{f} + c_2 h_k \mathbf{f}_t + \mathbf{f}_y h_k a_{21} K_1 + O(h_k^2).$$

Con estas dos expresiones, obtenemos

$$\begin{aligned} \mathbf{y}_{k+1} &= \mathbf{y}_k + h_k b_1 K_1 + h_k b_2 K_2 \\ &= \mathbf{y}_k + h_k b_1 \mathbf{f} + h_k b_2 [\mathbf{f} + c_2 h_k \mathbf{f}_t + \mathbf{f}_y h_k a_{21} \mathbf{f} + O(h_k^2)] \\ &= \mathbf{y}_k + h_k (b_1 + b_2) \mathbf{f} + h_k^2 b_2 c_2 \mathbf{f}_t + h_k^2 a_{21} b_2 \mathbf{f}_y \mathbf{f} + O(h_k^3). \end{aligned} \quad (8.29)$$

Para lograr un método cuyo error local de truncamiento sea $O(h_k^3)$, basta con comparar los desarrollos (8.28) y (8.29) e imponer igualdades término a término en los lados derechos:

$$\begin{cases} b_1 + b_2 = 1 \\ b_2 c_2 = \frac{1}{2} \\ a_{21} b_2 = \frac{1}{2} \end{cases}. \quad (8.30)$$

Enfatizamos que *cualquier* elección de b_1, b_2, c_2, a_{21} que sea solución de este sistema da lugar a un método de Runge-Kutta explícito de segundo orden. Es sencillo observar que el sistema de arriba tiene infinitas soluciones, que se pueden parametrizar como $b_2 = \alpha \neq 0$, $b_1 = 1 - \alpha$, $c_2 = a_{21} = \frac{1}{2\alpha}$. Mencionamos a continuación dos ejemplos clásicos.

Método de Heun. Corresponde a tomar $\alpha = 1/2$, esto es, $b_1 = b_2 = 1/2$, $c_2 = a_{21} = 1$ en (8.30). Así, dado \mathbf{y}_k , el iterado \mathbf{y}_{k+1} para el método de Heun se determina calculando

$$\begin{aligned} K_1 &= \mathbf{f}(t_k, \mathbf{y}_k), \\ K_2 &= \mathbf{f}(t_k + h_k, \mathbf{y}_k + h_k K_1), \\ \mathbf{y}_{k+1} &= \mathbf{y}_k + \frac{h_k}{2}(K_1 + K_2). \end{aligned}$$

Los cálculos en cada paso se deben realizar en el orden que escribimos arriba, y es claro que el método resultante es explícito. Por diseño, se tiene que es de segundo orden. Naturalmente, también podemos escribir el iterado en forma compacta (aunque no muy agradable):

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h_k}{2}(\mathbf{f}(t_k, \mathbf{y}_k) + \mathbf{f}(t_k + h_k, \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k))). \quad (8.31)$$

El método de Heun es *condicionalmente absolutamente estable*. En efecto, si lo aplicamos al problema test (8.21) con paso constante $h > 0$, se llega a que

$$y_k = \left(1 + h\lambda + \frac{(h\lambda)^2}{2}\right)^k y_0. \quad (8.32)$$

Por lo tanto, necesitamos que valga $|1 + h\lambda + \frac{(h\lambda)^2}{2}| < 1$. Operando elementalmente, concluimos la condición de estabilidad absoluta

$$h < \frac{2}{|\lambda|}.$$

Es interesante observar que esta condición es la misma que la que obtuvimos para el método de Euler (8.22). Remarcamos, sin embargo, que si el análisis se hace sobre el plano complejo, en el espíritu de la Observación 8.3.4, entonces se obtiene una región de estabilidad que contiene estrictamente a la del método de Euler.

Ejercicio 8.1. Demostrar la identidad (8.32).

Método de Euler modificado (o del punto medio explícito). Corresponde a tomar $\alpha = 1$, esto es, $b_1 = 0$, $b_2 = 1$, $c_2 = a_{21} = 1/2$ en (8.30). Esto da lugar a un iterado

$$\begin{aligned} K_1 &= \mathbf{f}(t_k, \mathbf{y}_k), \\ K_2 &= \mathbf{f}\left(t_k + \frac{h_k}{2}, \mathbf{y}_k + \frac{h_k}{2}K_1\right), \\ \mathbf{y}_{k+1} &= \mathbf{y}_k + h_k K_2, \end{aligned}$$

o, en forma compacta,

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}\left(t_k + \frac{h_k}{2}, \mathbf{y}_k + \frac{h_k}{2} \mathbf{f}(t_k, \mathbf{y}_k)\right).$$

Por diseño, es un método de segundo orden. Además, es condicionalmente absolutamente estable, con condición de estabilidad idéntica a la del método de Heun,

$$h < \frac{2}{|\lambda|}.$$

8.5.2. El método de Runge-Kutta

La construcción de la sección anterior es generalizable a mayores cantidades de etapas y, por ende, permite obtener métodos de mayor orden. En la medida que aumentamos la cantidad de etapas, el procedimiento se vuelve más engorroso. Sin embargo, existe un método de Runge-Kutta explícito de 4 etapas que es extremadamente popular, al punto que muchas veces se lo denomina como *el* método de Runge-Kutta. Dicho método consiste

en tomar

$$\begin{aligned}
 K_1 &= \mathbf{f}(t_k, \mathbf{y}_k), \\
 K_2 &= \mathbf{f}\left(t_k + \frac{h_k}{2}, \mathbf{y}_k + \frac{h_k}{2}K_1\right), \\
 K_3 &= \mathbf{f}\left(t_k + \frac{h_k}{2}, \mathbf{y}_k + \frac{h_k}{2}K_2\right), \\
 K_4 &= \mathbf{f}(t_k + h_k, \mathbf{y}_k + h_k K_3), \\
 \mathbf{y}_{k+1} &= \mathbf{y}_k + \frac{h_k}{6} (K_1 + 2K_2 + 2K_3 + K_4).
 \end{aligned} \tag{8.33}$$

Este es un método de cuarto orden y es condicionalmente absolutamente estable.

Ejemplo 8.5.1 (métodos de Runge-Kutta). Para ilustrar las implicancias prácticas de usar métodos de alto orden, comparamos los métodos de Euler explícito (8.9), de Heun (8.31) y de Runge-Kutta de 4 pasos (8.33) para la ecuación logística (8.11). Al igual que en el Ejemplo 8.2.1, consideramos $a = 1$, $y_0 = 0,1$, $I = [0, 5]$. La Figura 8.5 muestra las soluciones obtenidas con los métodos mencionados con paso constante $h = 1$. La solución

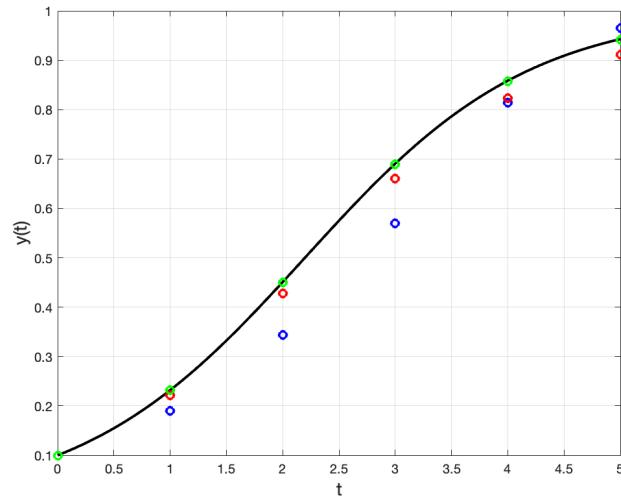


Figura 8.5: Solución a (8.11) en el intervalo $[0, 5]$, para $a = 1$, $y_0 = 0,1$, con paso constante $h = 1$. La solución exacta se representa con la curva en negro, mientras que las aproximaciones por los métodos de Euler hacia adelante (azul), de Heun (rojo) y de Runge-Kutta de 4 pasos (verde) se representan con puntos.

a este problema en tiempo $T = 5$ vale (recordar (8.14))

$$y(5) = \frac{0,1e^5}{0,1e^5 + 1 - 0,1} \approx 0,9428.$$

La siguiente tabla muestra la evolución de los errores en $T = 5$ para los tres métodos al tomar pasos de longitud constante e ir reduciendo la longitud del paso a la mitad.

h	Euler	Heun	RK4
1	$2,3 \times 10^{-2}$	$-3,2 \times 10^{-2}$	$-9,6 \times 10^{-4}$
$1/2$	$8,4 \times 10^{-3}$	$-6,3 \times 10^{-3}$	$-5,0 \times 10^{-5}$
$1/4$	$3,8 \times 10^{-3}$	$-1,5 \times 10^{-3}$	$-2,9 \times 10^{-6}$
$1/8$	$1,8 \times 10^{-3}$	$-3,5 \times 10^{-4}$	$-1,8 \times 10^{-7}$
$1/16$	$8,8 \times 10^{-4}$	$-8,8 \times 10^{-5}$	$-1,1 \times 10^{-8}$

Es interesante observar cómo evolucionan los errores: a *grosso modo*, cuando se divide h por la mitad, para el método de Euler el error se reduce a la mitad, para el de Heun en un factor de 4, y para el de Runge-Kutta de 4 etapas en un factor de 16. Esto es coherente con los hechos de que el método de Euler es de primer orden, el de Heun de es de segundo orden, y el de Runge-Kutta de 4 etapas es de cuarto orden.

En general, usar métodos de orden más alto tiene sentido en la medida en que la solución buscada sea suave y sus derivadas de orden alto no crezcan demasiado rápidamente. \triangle

8.6. Esquemas predictores-correctores

En esta sección comentamos brevemente sobre los llamados esquemas *predictores-correctores* y, en cierta forma, esta discusión complementa a la Observación 8.2.2. Esta observación trata sobre que cuando consideramos un método implícito para resolver un problema de valores iniciales, muchas veces nos enfrentamos a que debemos resolver ecuaciones algebraicas no lineales para poder computar los iterados, y se plantea como opción hacer iteraciones de punto fijo usando como iterado inicial el valor de la solución en el paso anterior.

Muchas veces, se pueden obtener mejores iterados iniciales usando un paso de un método explícito. De hecho, se puede demostrar que si se tiene \mathbf{y}_k y se quiere computar \mathbf{y}_{k+1} con un método implícito de orden p , entonces haciendo una iteración de punto fijo usando como iterado inicial un paso de un método explícito de orden (al menos) $p - 1$ se consigue que, en un paso, la aproximación \mathbf{y}_{k+1}^1 sea del mismo orden que el “iterado correcto” \mathbf{y}_{k+1} . En esto consisten los **esquemas predictores-correctores**: dado \mathbf{y}_k , se usa un método explícito para *predecir* un valor $\mathbf{y}_{k+1}^{(P)}$, y luego se utiliza dicho valor en un paso de *corrección* que aplica un método implícito. Este procedimiento da lugar a un método explícito y de orden p .

Ejemplo 8.6.1 (otra forma de entender el método de Heun). Dado el problema (8.6), hacemos un par predictor-corrector usando el método de Euler hacia adelante (8.9) como

predictor y el del trapecio (8.23) como corrector,

$$\begin{aligned}\mathbf{y}_{k+1}^{(P)} &:= \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k), \\ \mathbf{y}_{k+1} &:= \mathbf{y}_k + h_k \left(\frac{\mathbf{f}(t_k, \mathbf{y}_k) + \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}^{(P)})}{2} \right).\end{aligned}$$

Enfatizamos que la diferencia entre la segunda fórmula arriba y (8.23) está en que aquí, del lado derecho no aparece el valor desconocido \mathbf{y}_{k+1} sino el valor ya computado $\mathbf{y}_{k+1}^{(P)}$. Hemos obtenido el iterado

$$\mathbf{y}_{k+1} := \mathbf{y}_k + h_k \left(\frac{\mathbf{f}(t_k, \mathbf{y}_k) + \mathbf{f}(t_{k+1}, \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k))}{2} \right),$$

que es exactamente igual a (8.31): el método de Heun es el esquema predictor-corrector que surge al combinar los métodos de Euler hacia adelante con el del trapecio. \triangle

Ejemplo 8.6.2. Construimos el par predictor-corrector usando el método de Euler hacia adelante (8.9) como predictor y el de Euler hacia atrás (8.10) como corrector. Para cada $k \geq 0$, dado \mathbf{y}_k , procedemos en dos etapas:

$$\begin{aligned}\mathbf{y}_{k+1}^{(P)} &:= \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k) && \text{(paso predictor)}, \\ \mathbf{y}_{k+1} &:= \mathbf{y}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}^{(P)}) && \text{(paso corrector)}.\end{aligned}$$

Podemos reemplazar el valor de $\mathbf{y}_{k+1}^{(P)}$ en la segunda fórmula, lo que da lugar a

$$\mathbf{y}_{k+1} := \mathbf{y}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k)).$$

No nos hemos encontrado con este método directamente, pero podemos interpretarlo como un método de Runge-Kutta de dos etapas, en las que en (8.26)–(8.27) se toma $s = 2$, $c_1 = 0$, $c_2 = 1$, $a_{11} = a_{12} = a_{22} = 0$, $a_{21} = 1$, $b_1 = 0$, $b_2 = 1$. Como este conjunto de parámetros no satisface (8.30), este método no es de segundo orden. Dejamos al lector la tarea de verificar que es de primer orden. Este método no tiene mayor interés práctico. \triangle

8.7. *Solvers*

Tanto Matlab como Octave tienen incorporados varios *solvers*, que permiten obtener sucesiones $\{(t_k, \mathbf{y}_k)\}$ utilizando distintos métodos. Naturalmente, distintos métodos pueden tener diferentes ventajas o desventajas, ser más o menos precisos, o más o menos demandantes computacionalmente. La elección de qué método usar puede depender de nuestros requerimientos como también de algunas características del problema que queremos resolver.

Los *solvers* de ecuaciones diferenciales ordinarias son funciones de Octave/Matlab cuyo nombre es del tipo `odeNNX`. La sigla `ode` proviene del inglés *ordinary differential equation*,

NN es un par de números que indican el orden del método que el *solver* utiliza, y X es un conjunto (posiblemente vacío) de caracteres que indica alguna característica especial de dicho método. Todavía no hemos definido qué es el orden de un método (ver la Definición 8.3.4 más abajo) pero, por el momento, basta con considerar que el orden es un indicador de precisión.

Una característica importante de estos *solvers* es que utilizan pasos variables, esto es, dada una cierta tolerancia de error, intentan tomar los pasos lo mayor posible de modo que la solución computada sea aceptable para dicha tolerancia y que pueda ser computada de forma económica. La determinación de los pasos se hace usando *estimadores de error*: explicamos algunas ideas sobre el tema en la Sección 8.7.1.

A continuación, damos una lista de algunos *solvers* que podemos utilizar y una breve descripción como referencia.

- `ode45` usa un par de métodos de Runge-Kutta explícitos, uno de orden 4 y otro de orden 5: es el llamado par de [Dormand-Prince](#). El método de orden 4 es utilizado para computar los iterados, mientras que el de orden 5 se usa principalmente para estimar el error y determinar el largo del paso. De acuerdo con la [documentación de Matlab](#), éste “es el primer solver que debería probar para solucionar la mayoría de problemas.”
- `ode23` también usa un par de métodos de Runge-Kutta explícitos, pero uno de orden 2 y otro de orden 3. En forma similar a `ode45`, los iterados se computan usando el método de orden 2 y el de orden 3 se usa principalmente para estimar el error y determinar el largo del paso. Puede ser más eficiente que `ode45` para tolerancias de error grandes o en problemas moderadamente rígidos.
- `ode23s` usa un par de métodos de Runge-Kutta, pero la letra `s` al final de su nombre nos indica que es un solver diseñado para problemas *rígidos*⁶. Es apropiado para problemas de este tipo en los que tengamos requerimientos bajos de precisión.
- `ode15s` también es un *solver* para problemas rígidos, pero de mayor orden que `ode23s`.

En la lista de arriba aparecieron varios términos que aún no hemos definido. En las siguientes secciones vamos a profundizar sobre éstas y otras nociones relacionadas a la aproximación numérica de ecuaciones diferenciales ordinarias. El siguiente ejemplo muestra cómo se utilizan los *solvers* en la práctica.

Ejemplo 8.7.1 (predador-presa). El modelo de Lotka-Volterra de predador-presa se usa para describir dinámicas en ecología en las que dos especies interactúan, una como presa y otra como depredador. Consideremos un ecosistema en el que solamente hay conejos y zorros: asumimos que los conejos tienen acceso ilimitado a comida, mientras que los zorros

⁶En inglés, rigidez se dice *stiffness*.

solamente pueden comer conejos. Modelamos este ecosistema con un par de ecuaciones diferenciales no lineales acopladas,

$$\begin{cases} c'(t) = 2c(t) - \alpha c(t)z(t), \\ z'(t) = -z(t) + \alpha c(t)z(t), \\ c(t_0) = c_0, \\ z(t_0) = z_0. \end{cases} \quad (8.34)$$

Aquí, la variable t representa el tiempo, $c(t)$ (resp. $z(t)$) es la población de conejos (resp. zorros) en el instante t , y $\alpha > 0$ es una constante que indica la probabilidad de encuentro entre zorros y conejos. Se puede probar que las soluciones del problema (8.34) son periódicas, con un período que depende de las condiciones iniciales. Esto es, dados c_0, z_0 , existe un tiempo T tal que

$$c(t+T) = c(t), \quad z(t+T) = z(t), \quad \text{para todo } t \geq t_0. \quad (8.35)$$

Para resolver numéricamente el sistema (8.34), introducimos la función $\mathbf{y}(t) = [c(t); z(t)]$, y escribimos las ecuaciones compactamente,

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)) := \begin{bmatrix} 2\mathbf{y}_1(t) - \alpha\mathbf{y}_1(t)\mathbf{y}_2(t) \\ -\mathbf{y}_2(t) + \alpha\mathbf{y}_1(t)\mathbf{y}_2(t) \end{bmatrix}, \quad \mathbf{y}(t_0) = \mathbf{y}_0 := \begin{bmatrix} c_0 \\ z_0 \end{bmatrix}.$$

Buscamos resolver el sistema para $t_0 = 0$, con $c_0 = 100$, $z_0 = 10$ y $\alpha = 0,02$. En Octave, introducimos los valores de estas constantes y definimos la función \mathbf{f} ,

```
alpha = 0.02;
y0 = [100 10];
f = @(t,y) [2*y(1) - alpha*y(1)*y(2); -y(2)+alpha*y(1)*y(2)];
```

Para resolver el problema computacionalmente usando uno de los solvers que mencionamos más arriba, basta con llamarlo. Por ejemplo, para usar `ode23` para resolver el sistema en el intervalo de tiempo $[0, 10]$, escribimos

```
[tiempos, poblaciones] = ode23(f, [0 10], y0);
```

La Figura 8.6 muestra dos tipos de gráficas. En la primera, representamos c y z como funciones del tiempo. La segunda es un *diagrama de fase*, en el que c está en el eje de las x y z en el de las y . Las soluciones obtenidas con los solvers `ode23`, `ode45` y `ode23s` se ven prácticamente iguales, aunque no lo son. Por ejemplo, para este intervalo de tiempo, `ode23` hace 101 pasos, `ode45` hace 117, y `ode23s` hace 96. De todos modos, en este problema no vemos grandes diferencias entre las distintas soluciones computadas. \triangle

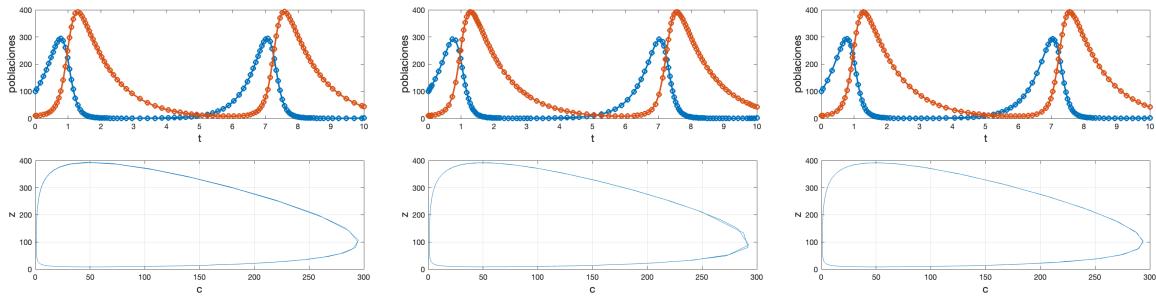


Figura 8.6: Solución al modelo de Lotka-Volterra usando los *solvers* `ode23` (izquierda), `ode45` (centro), `ode23s` (derecha). En la fila de arriba, graficamos las poblaciones de conejos (azul) y zorros (rojo) en función del tiempo. La fila de abajo representa conjuntamente las poblaciones de conejos y zorros.

Eventos

En el Ejemplo 8.7.1, las soluciones son periódicas. Si queremos determinar el *período* de estas soluciones, esto es, el menor T que cumple (8.35), podríamos intentar hacerlo visualmente a partir de la Figura 8.6: para los valores de los parámetros que tomamos, parece que T es un poco mayor que 6. Naturalmente, podríamos determinar T de forma un poco más precisa si usamos los valores de **poblaciones** obtenidos.

Una característica interesante de los *solvers* son los **eventos**. Uno puede indicar a la función que tome determinadas acciones si ocurren situaciones de interés. Como primer ejemplo, pongamos que buscamos el primer tiempo tal que la población de conejos es igual a 200. Para ello, creamos una nueva función, que en este caso llamaremos **parar**:

```
function [val,isterm,dir] = parar(t,y)
    val = y(1)-200;
    isterm = 1;
    dir = [];
```

La primera salida, **val**, es el valor que queremos que sea cero. Fijando la segunda salida **isterm** a 1 se indica que el *solver* debe terminar una vez que **val** es igual a cero. Poner que la tercera salida, **dir**, sea la matriz nula, indica que el cero que estamos buscando puede ser alcanzado desde cualquier dirección, desde valores positivos o negativos.

Con esto, modificamos nuestro código para llamar al solver como se describe a continuación.

```
alpha = 0.02;
y0 = [100 10];
F = @(t,y) [2*y(1) - alpha*y(1)*y(2); -y(2)+alpha*y(1)*y(2)];
opts = odeset('events',@parar);
```

```
[t,y,tfinal] = ode23(F,[0 10],y0,opts);
plot(t, y,'-o', 'LineWidth',2)
```

La imagen a la izquierda en la Figura 8.7 muestra la salida. Observamos que, a pesar de que al llamar a `ode23` pusimos el intervalo de tiempo $[0, 10]$, el *solver* se detuvo antes, cuando $y(1) = 200$. La variable `tfinal` tiene almacenado ese instante de tiempo, y en este caso obtuvimos $\text{tfinal} = 0.4034$.

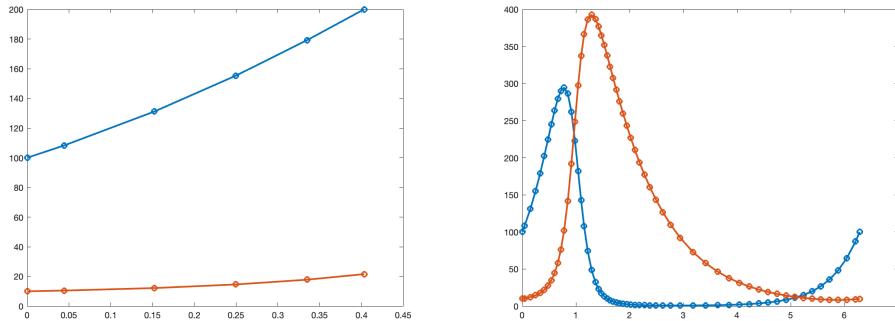


Figura 8.7: Solución al modelo de Lotka-Volterra utilizando eventos. Izquierda: el *solver* se detiene cuando $y(1) = 200$. Derecha: se detiene luego de un período.

Un ejemplo un poco más sofisticado de evento es el de determinar períodos. Consideraremos la solución computada en el Ejemplo 8.7.1. Debemos indicarle a Octave que nos interesa el instante en el que el vector `posiciones` repite las condiciones iniciales. Detectar esto puede ser bastante delicado, ya que como se puede observar haciendo zoom en la fila inferior de la Figura 8.6, si bien la solución exacta de nuestro problema es periódica, las soluciones numéricas no lo son. Si modificamos nuestra función `parar`, cambiando la condición de parada por `val = y(1)-100` y modificando el valor de la salida `dir` a 1 (lo que indica que queremos que $y(1)$ se aproxime al valor 100 por abajo), obtenemos un gráfico como se muestra en la parte derecha de la Figura 8.7, y obtenemos $\text{tfinal} = 6.2844$, lo que nos da una estimación del período de esta órbita.

8.7.1. El algoritmo BS23 y `ode23`

Una característica importante de los *solvers* de Matlab/Octave es que, dada una tolerancia (que puede ser ingresada por el usuario o la dada por defecto), *adaptan* las longitudes de los pasos de modo que la solución numérica sea aceptable de acuerdo con dicha tolerancia.

En principio, parecería que para hacer adaptatividad debería ser necesario conocer la solución exacta del problema que uno busca: en cada iteración, se podría comparar la solución obtenida con la solución computada y si el error tiene norma lo suficientemente pequeña, entonces el paso dado es aceptable. Si no lo es, entonces se debería descartar el

iterado computado y volver a intentar con un paso más corto. En la práctica, sin embargo, uno no tiene a mano la solución exacta del problema, y se suele reemplazar el error por una cantidad computable: se usa un paso con un método más preciso (típicamente de mayor orden) como *solución aproximada* y se usa la discrepancia entre las dos soluciones computadas como *estimador del error*.

Los métodos de un paso permiten adaptar las longitudes de los pasos de forma relativamente sencilla, porque uno simplemente puede ir cambiando los h_k de un paso al siguiente. Debido a que la de Runge-Kutta es una familia de métodos de un paso que pueden ser de orden arbitrario, estos métodos son la base de una buena parte de los *solvers* de Matlab/Octave. Aquí vamos a comentar algunas ideas clave relacionadas al **algoritmo BS23**⁷, que es la base del *solver* `ode23`. Para simplificar la notación, vamos a considerar un problema de valores iniciales (8.6) escalar, esto es, con $n = 1$. En el caso vectorial, se trabaja coordenada a coordenada y se debe reemplazar los valores absolutos por normas.

El algoritmo BS23 usa dos métodos de Runge-Kutta combinados, uno de orden 3 y otro de orden 2. El método de orden 3 es de 3 etapas, mientras que el de orden 2 utiliza las mismas 3 etapas y le agrega una cuarta; sin embargo, el algoritmo usa el llamado enfoque FSAL⁸, que implica que en la práctica se tengan que hacer 3 evaluaciones de f en cada paso.

Concretamente, dadas *tolerancias de error* relativo `rtol` y absoluto `atol`, el iterado y_k , y un *paso tentativo* h_k , para determinar el iterado siguiente y_{k+1} el algoritmo BS23 comienza computando

$$\begin{aligned} K_1 &:= f(t_k, y_k), \\ K_2 &:= f\left(t_k + \frac{h_k}{2}, y_k + \frac{h_k}{2}K_1\right), \\ K_3 &:= f\left(t_k + \frac{3h_k}{4}, y_k + \frac{3h_k}{4}K_2\right), \\ y_{k+1} &:= y_k + \frac{h_k}{9}(2K_1 + 3K_2 + 4K_3). \end{aligned} \tag{8.36}$$

El valor de y_{k+1} obtenido corresponde a un método de Runge-Kutta de tercer orden. A continuación, se reutilizan los valores K_1, K_2, K_3 ya computados para generar un método de segundo orden. Para ello, se agrega el valor

$$K_4 := f(t_k + h_k, y_{k+1});$$

notar que en caso de que el paso que se propone sea aceptado, este valor de K_4 es el valor de K_1 del paso siguiente. Se calcula un paso estimado

$$\tilde{y}_{k+1} := y_k + \frac{h_k}{24}(7K_1 + 6K_2 + 8K_3 + 3K_4).$$

⁷Bogacki, P. y Shampine, L. F., *A 3(2) pair of Runge–Kutta formulas*, Applied Mathematics Letters, 2 (1989), pp. 1–9.

⁸*First Same As Last*, “primero igual al último”.

Este valor \tilde{y}_{k+1} corresponde a un método de Runge-Kutta de segundo orden y de cuatro etapas. Finalmente, se compara la discrepancia entre los valores obtenidos: se computa

$$\varepsilon := y_{k+1} - \tilde{y}_{k+1} = \frac{h_k}{72} (-5K_1 + 6K_2 + 8K_3 - 9K_4).$$

En la práctica no es necesario computar \tilde{y}_{k+1} para evaluar ε , por lo que no se lo suele hacer. La cantidad ε es un *estimador del error absoluto* en el paso. A partir de este valor y las tolerancias relativas y absolutas, se determina una cantidad $\text{err} > 0$, que tiene que ver con el error relativo y que se usa para decidir si el paso es aceptable o no. Si err es suficientemente pequeña, entonces se acepta el valor de y_{k+1} computado. En cambio, si err no lo es, entonces se redefine $h_k \leftarrow \frac{h_k}{2}$, se vuelve a (8.36) y se recalculan y_{k+1} y el estimador ε para este nuevo paso de longitud menor.

Una vez que se acepta el iterado y_{k+1} , se debe elegir la longitud del paso siguiente h_{k+1} : si err es mucho menor que rtol , entonces tenemos margen para ser ambiciosos e intentar tomar un paso más largo. Si err/rtol está muy cercano a 1, entonces conviene intentar con un paso h_{k+1} similar a h_k . En la práctica, el algoritmo BS23 –y en consecuencia el solver `ode23`– utiliza una fórmula *ad hoc*,

$$h_{k+1} := h_k \min \left\{ 5, \frac{4}{5} \left(\frac{\text{rtol}}{\text{err}} \right)^{1/3} \right\}.$$

Los factores de 5 y de 4/5 arriba son para prevenir cambios excesivos de longitud entre pasos consecutivos, y la potencia 1/3 está reelecciónada con que el método de Runge-Kutta utilizado es de orden 3.

8.8. Rigidez

Esta sección trata sobre un concepto sutil e importante en la resolución numérica de ecuaciones diferenciales ordinarias: la llamada *rigidez*. Consideremos la siguiente variante sencilla del problema test (8.21):

$$\begin{cases} y'(t) = \lambda(y(t) - g(t)) + g'(t), \\ y(0) = y_0, \end{cases} \quad (8.37)$$

donde g es una función conocida y regular, y $\lambda < 0$. Se puede verificar que la solución de este problema es

$$y(t) = (y_0 - g(0))e^{\lambda t} + g(t).$$

Notemos que, si $\lambda \ll 0$, entonces el primer término del lado derecho arriba es despreciable cuando t es grande: la solución se acerca a $g(t)$ exponencialmente rápido respecto a t . Sin embargo, si usamos varios de los métodos que hemos discutido hasta ahora para resolver este problema, nos vamos a encontrar con algunas dificultades. Para que los métodos explícitos sean estables se requiere que los pasos sean extremadamente pequeños.

Para fijar ideas, tomamos $\lambda = -10$, $g(t) = t$, $y_0 = 1$, y resolvemos (8.37) sobre el intervalo $[0, 10]$. Si usamos el método de Euler hacia adelante o el de Heun con paso constante, estamos en serios problemas: para que sean estables vamos a necesitar que los pasos sean menores⁹ que $h = 2/10$. La Figura 8.8 muestra soluciones con paso constante y usando 51 (estable), 50 (límite de estabilidad) y 49 (inestable) pasos, respectivamente. El comportamiento que observamos es similar al de la primera fila en la Figura 8.4. A pesar de que la solución es prácticamente lineal para $t \gtrsim 1$, con los métodos de Euler y de Heun no podemos dar pasos muy largos sin perder el control de las soluciones numéricas.

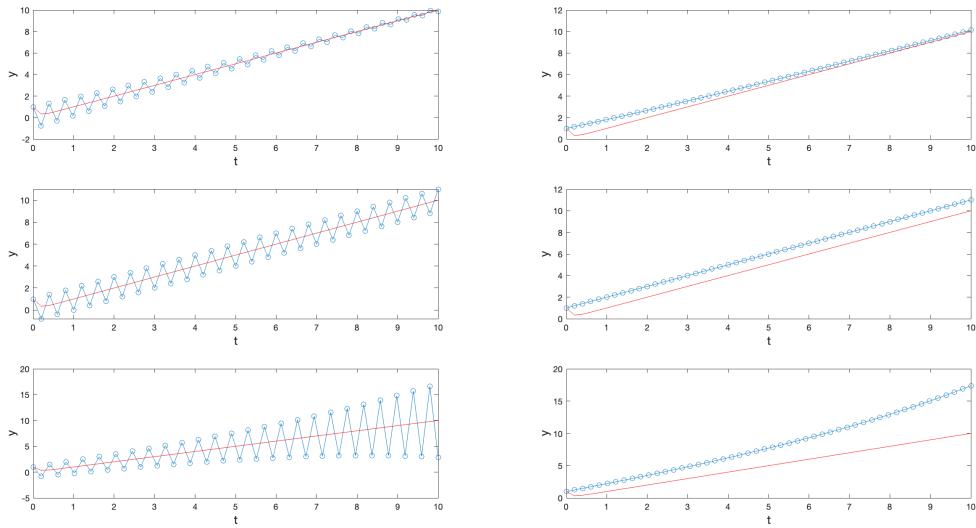


Figura 8.8: Solución a (8.37) en el intervalo $[0, 10]$, para $\lambda = -10$, $g(t) = t$, $y_0 = 1$, con paso constante. La primera columna muestra las soluciones con el método de Euler y la segunda columna con el método de Heun, tomando $h = 10/51$ (arriba), $h = 1/5$ (centro), y $h = 10/49$ (abajo). La solución exacta se representa con las curvas en rojo, mientras que las aproximaciones numéricas se muestran en azul.

Para este problema, incluso los *solvers* `ode23` y `ode45`, que usan métodos de Runge-Kutta explícitos, están en dificultades. Si bien la solución de nuestro problema es suave, no podemos usar pasos de tiempo largos. Para mayor dramatismo, cambiamos el intervalo de resolución a $[0, 100]$ y tomamos $\lambda = -100$ en (8.37). La Figura 8.9 muestra la solución computada con el solver `ode45` con los mismos parámetros pero sobre el intervalo $[0, 100]$, y un zoom en el intervalo $[95, 100]$. Para este problema, `ode45` tomó 12089 pasos para

⁹No vamos a entrar en el detalle de por qué se tiene esta condición de estabilidad, pero mencionamos que se deduce de un razonamiento como el de la Observación 8.3.3. A los efectos de la discusión sobre rigidez, basta con tener en cuenta que estos métodos son condicionalmente estables y que por lo tanto hay una longitud de paso máximo admisible.

asegurarse de estar dentro de su tolerancia de error por defecto¹⁰.

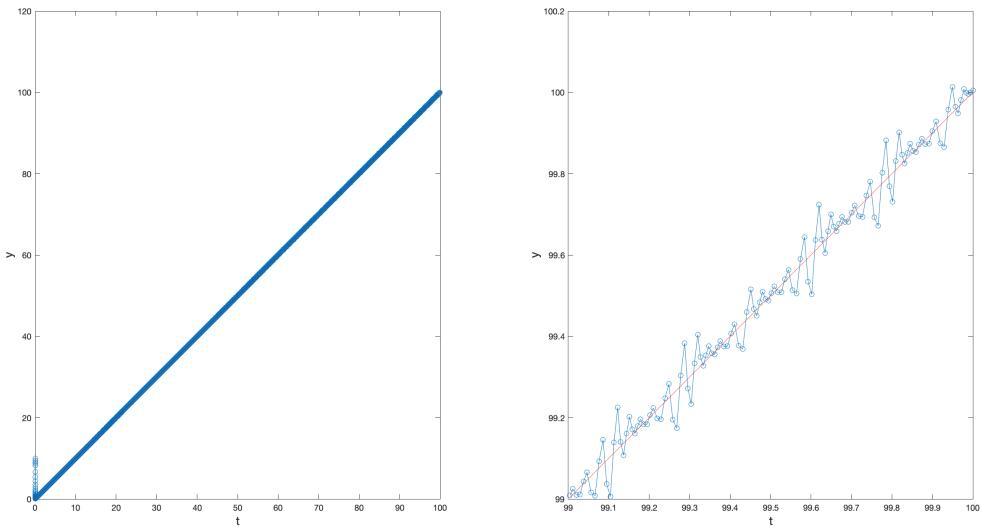


Figura 8.9: Solución a (8.37) en el intervalo $[0, 100]$, para $\lambda = -100$, $g(t) = t$, $y_0 = 1$, con el solver `ode45`. En la izquierda, vemos la solución en todo el intervalo (se ve como una curva gruesa debido a la enorme cantidad de puntos que se representan), y a la derecha el detalle para $t \in [95, 100]$.

Estamos ante un problema que parece ser sencillo pero que resulta costoso de resolver con métodos que no sean incondicionalmente estables. En particular, en virtud de la Observación 8.3.5, cualquier método explícito de paso simple va a darnos dolores de cabeza en esta situación. A problemas de este tipo se los suele llamar **rígidos**. No hay una definición rigurosa de rigidez, aunque típicamente refiere al hecho de que se busca una solución que varía lentamente (es suave), pero que está rodeada por soluciones que varían rápidamente. Esto hace que, si se le aplican métodos no absolutamente estables con control de pasos, los pasos deban ser extremadamente cortos para obtener resultados satisfactorios. La rigidez es una cuestión de *eficiencia*.

Para tratar con problemas rígidos, se suelen usar métodos implícitos. La letra **s** en los nombres de *solvers* como `ode23s` u `ode15s` hace referencia precisamente a que usan algoritmos adecuados para problemas rígidos. Para comparación, la Figura 8.10 muestra la solución obtenida con el solver `ode23s` con los mismos parámetros que usamos para producir la Figura 8.9. En este caso, el solver solamente toma 47 pasos en todo el intervalo $[0, 100]$, y de esos 47 pasos, 36 se encuentran concentrados en el intervalo $[0, 1]$.

¹⁰Por defecto, en `ode45` la tolerancia de error relativa es 10^{-3} y la absoluta es 10^{-6} .

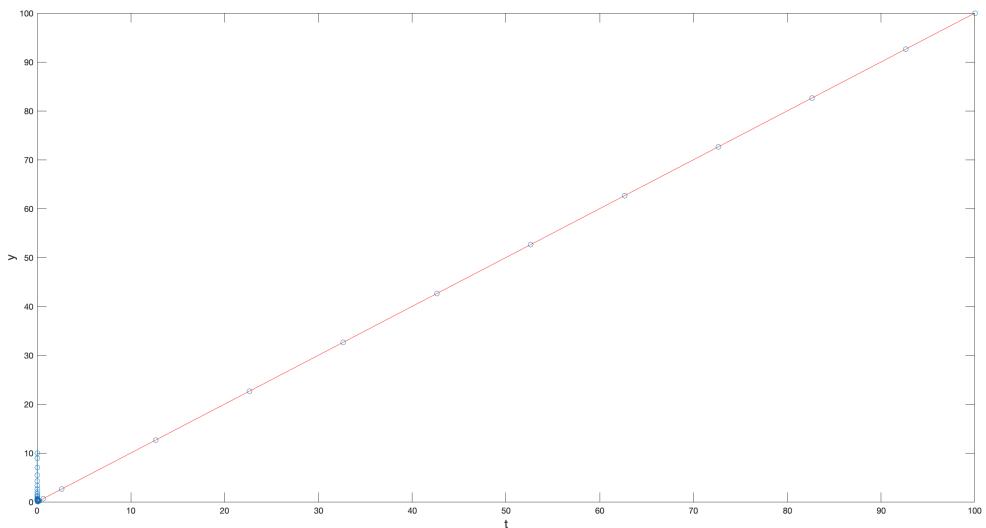


Figura 8.10: Solución a (8.37) en el intervalo $[0, 100]$, para $\lambda = -100$, $g(t) = t$, $y_0 = 1$, con el solver `ode23s`.

Apéndice A

Preliminares y repaso

Este capítulo contiene un repaso (parcial) de conceptos y herramientas fundamentales para el seguimiento del curso. Su contenido corresponde a los cursos de Geometría y Álgebra Lineal y Cálculo Diferencial e Integral en una y varias variables. En caso de tener dudas, recomendamos fuertemente consultar los materiales de dichos cursos, y en cada sección incluimos una referencia para repasar conceptos.

A.1. Desarrollo de Taylor y resto de Lagrange

A.1.1. En una variable

Lectura recomendada: T. Apostol. *Calculus vol. I*, Capítulo 7 (“Aproximación de funciones por polinomios”).

Sea $k \in \mathbb{N}$. Recordemos que una función f se dice de clase C^k (y escribimos $f \in C^k$) si es derivable k veces, y su derivada k -ésima es continua.

Teorema A.1.1 (Taylor). *Consideremos una función $f : D \subset \mathbb{R} \rightarrow \mathbb{R}$ de clase C^k . Para todo punto a interior a D existe una función r_k , denominada resto o error, tal que*

$$f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \dots + \frac{f^{(k)}(a)}{k!}(x - a)^k + r_k(x - a)$$

y

$$\lim_{x \rightarrow a} \frac{r_k(x - a)}{(x - a)^k} = 0. \quad (\text{A.1})$$

Observación A.1.1 (otra forma de expresar el desarrollo de Taylor). A veces puede ser útil escribir el desarrollo de Taylor en términos de la variación en x respecto de a , definda como $h := x - a$. Para esto, basta con reemplazar $x = a + h$ en el Teorema de Taylor:

$$f(a + h) = f(a) + f'(a)h + \frac{f''(a)}{2!}h^2 + \dots + \frac{f^{(k)}(a)}{k!}h^k + r_k(h).$$

Con esta notación, el límite (A.1) se puede reescribir como $\lim_{h \rightarrow 0} \frac{r_k(h)}{h^k} = 0$. \triangle

Definición A.1.1 (polinomio de Taylor). El polinomio involucrado en el Teorema A.1.1 se denomina **polinomio de Taylor de orden k de la función f en el punto a** y lo denotamos por $T_k f$ ¹,

$$T_k f(x) := f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \dots + \frac{f^{(k)}(a)}{k!}(x - a)^k.$$

 \triangle

Observación A.1.2. Una noción intuitiva sobre el polinomio de Taylor $T_k f(x; a)$ es que es el polinomio de grado k que “mejor aproxima” a f alrededor del punto a . Concretamente, usando el Teorema A.1.1 y en particular la fórmula (A.1), tenemos

$$|f(x) - T_k f(x; a)| = |r_k(x - a)| = o((x - a)^k).$$

Se puede demostrar, mediante un argumento de inducción, que $T_k f(x; a)$ es el único polinomio de grado k con esta propiedad. Veamos la demostración para el caso $k = 1$. Sea $\ell(x) := \alpha + \beta(x - a)$ una función lineal² tal que $f(x) - \ell(x) = o(x - a)$. Queremos probar que entonces tiene que ser $\ell = T_1 f$.

Del hecho de que $f(x) - \ell(x) = o(x - a)$ se deduce, en particular, que la diferencia entre f y ℓ tiende a 0 cuando $x \rightarrow a$,

$$0 = \lim_{x \rightarrow a} f(x) - \ell(x) = \lim_{x \rightarrow a} f(x) - (\alpha + \beta(x - a)).$$

Como $x - a \rightarrow 0$ y f es continua en a , deducimos que necesariamente tiene que ser $\alpha = f(a)$. A continuación, usamos nuevamente el hecho de que $f(x) - \ell(x) = o(x - a)$ para ahora afirmar que la diferencia entre f y ℓ tiende a 0 *más rápido que* $x - a$ cuando $x \rightarrow a$,

$$0 = \lim_{x \rightarrow a} \frac{f(x) - \ell(x)}{x - a} = \lim_{x \rightarrow a} \frac{f(x) - (f(a) + \beta(x - a))}{x - a} = \lim_{x \rightarrow a} \frac{f(x) - f(a)}{x - a} - \beta.$$

Como f es derivable en a , deducimos que necesariamente tiene que ser $\beta = f'(a)$. Por lo tanto,

$$\ell(x) = \alpha + \beta(x - a) = f(a) + f'(a)(x - a) = T_1 f(x; a).$$

 \triangle

En muchos casos, nos va a ser útil considerar polinomios de Taylor de orden uno, esto es, reemplazar una función f por la función lineal que “mejor la aproxima” alrededor de un cierto punto a .

¹La notación $T_k f$ no especifica cuál es el punto a . En caso de existir ambigüedad respecto a de qué punto se trata, en vez de $T_k f(x)$ escribiremos $T_k f(x; a)$.

²El lector podría estar considerando escribir una función lineal arbitraria como $\ell(x) := \tilde{\alpha} + \tilde{\beta}x$. Es claro que las dos formas son equivalentes, ya que tendríamos $\tilde{\beta} = \beta$ y $\tilde{\alpha} = \alpha - a\beta$, pero escribir ℓ de la forma en que lo hacemos nos permite simplificar un poco las cuentas.

Definición A.1.2 (linealización). Sea $f : D \subseteq \mathbb{R} \rightarrow \mathbb{R}$ y a un punto interior a D . **Linealizar** la función f alrededor de a consiste en reemplazar f por su polinomio de Taylor de primer orden en dicho punto. Concretamente, es reemplazar f por la función lineal

$$p_1(x) := T_1 f(x; a) = f(a) + f'(a)(x - a).$$

△

El Teorema A.1.1 nos da el polinomio que “mejor aproxima” a una cierta función f alrededor de un cierto punto a en el sentido de que se cumple (A.1). Sin embargo, ese teorema no nos indica ninguna forma de expresar el resto r_k . Concluimos esta sección con un resultado útil al respecto³.

Teorema A.1.2 (forma de Lagrange del resto). *Si $f \in C^{k+1}$, entonces, para todo $x \in \mathbb{R}$ existe un punto θ_x entre a y x (esto es, $\theta_x \in [a, x]$ si $x > a$ o $\theta_x \in [x, a]$ si $x < a$) tal que el resto de Taylor se puede expresar como*

$$r_k(x - a) = \frac{f^{(k+1)}(\theta_x)}{(k+1)!}(x - a)^{k+1}.$$

Observación A.1.3. El Teorema A.1.2 nos asegura que el resto de Taylor se puede escribir de forma análoga que los demás términos en el desarrollo de Taylor, pero con la derivada $(k+1)$ -ésima evaluada en un cierto punto θ_x entre a y x . Es importante notar que θ_x depende tanto de x como de a (y obviamente de la función f), y que el teorema anterior no nos brinda su valor exacto. △

Retomamos brevemente el tema de desarrollos de Taylor en una variable en el Capítulo 1 de las notas del curso.

A.1.2. En varias variables

Lectura recomendada: M. Fiori. *Notas del curso de CDIVV*, Capítulo 6 (“Diferenciabilidad”).

Vamos a considerar únicamente el desarrollo de Taylor de primer orden para funciones de \mathbb{R}^n en \mathbb{R}^m ($n, m \geq 1$). Éste involucra a la matriz jacobiana, que juega el papel de la derivada primera en el contexto multivariado.

Definición A.1.3 (matriz jacobiana). Sean $\mathbf{f} : D \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$, una función diferenciable y \mathbf{a} un punto interior a D . Si $\mathbf{f} = (f_1, f_2, \dots, f_m)$, la **matriz jacobiana de \mathbf{f} en \mathbf{a}** es

$$\mathbb{J}_{\mathbf{f}}(\mathbf{a}) := \begin{pmatrix} \nabla f_1(\mathbf{a}) \\ \nabla f_2(\mathbf{a}) \\ \vdots \\ \nabla f_m(\mathbf{a}) \end{pmatrix} \in \mathcal{M}_{m \times n}(\mathbb{R}),$$

³Remarcamos que existen otras formas de caracterizar el resto de Taylor.

donde ∇f_i denota el gradiente de la función coordenada f_i ,

$$\nabla f_i := \left(\frac{\partial f_i}{\partial x_1}, \frac{\partial f_i}{\partial x_2}, \dots, \frac{\partial f_i}{\partial x_n} \right), \quad i = 1, \dots, m.$$

△

El siguiente es una generalización del Teorema A.1.1 (con $k = 1$) al caso multivariado: dice que una función diferenciable se puede aproximar, en cada punto de su dominio, con una transformación lineal afín –su polinomio de Taylor de primer grado en varias variables– y que esta aproximación tiende a cero (en cada coordenada) más rápido que $\|\mathbf{x} - \mathbf{a}\|$.

Teorema A.1.3 (Taylor de primer orden). *Sean $\mathbf{f} : D \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ una función diferenciable y \mathbf{a} un punto interior a D . Entonces, existe una función \mathbf{r} , denominada resto o error, tal que*

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{a}) + J_{\mathbf{f}}(\mathbf{a})(\mathbf{x} - \mathbf{a}) + \mathbf{r}(\mathbf{x} - \mathbf{a}), \quad y \quad \lim_{\mathbf{x} \rightarrow \mathbf{a}} \frac{\|\mathbf{r}(\mathbf{x} - \mathbf{a})\|}{\|\mathbf{x} - \mathbf{a}\|} = 0.$$

Definición A.1.4 (polinomio de Taylor de primer orden). El **polinomio de Taylor de orden uno de la función diferenciable $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ en el punto $\mathbf{a} \in \mathbb{R}^n$** es $T_1\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$,

$$T_1\mathbf{f}(\mathbf{x}) := \mathbf{f}(\mathbf{a}) + J_{\mathbf{f}}(\mathbf{a})(\mathbf{x} - \mathbf{a}).$$

△

A.2. Teoremas de valores medios

Lectura recomendada: Notas de CDIV sobre *Límites y continuidad*, Sección 2.3 (“Valores intermedios”) para el Teorema de Bolzano, y T. Apostol. *Calculus vol. I*, Sección 4.14 (“Teorema del valor medio para derivadas”) para el Teorema de Rolle.

El Teorema de Bolzano nos da una condición suficiente sencilla para asegurar que una función real tenga una raíz: que sea continua y que su dominio contenga un intervalo en el que “cambia de signo”.

Teorema A.2.1 (Bolzano). *Sea $f : D \subset \mathbb{R} \rightarrow \mathbb{R}$ y $[a, b] \subset D$. Si f es continua en $[a, b]$, y $f(a)f(b) < 0$, entonces existe $c \in (a, b)$, tal que $f(c) = 0$.*

Observación A.2.1. La condición $f(a)f(b) < 0$, equivale a decir que f toma valores con signo distinto en los puntos a y b . Esto es, o bien $f(a) > 0$ y $f(b) < 0$, o bien $f(a) < 0$ y $f(b) > 0$. △

Por otra parte, el Teorema de Rolle es un resultado de valor medio para derivadas: nos da una condición para que la derivada de una función tenga una raíz.

Teorema A.2.2 (Rolle). *Sea $f : D \subset \mathbb{R} \rightarrow \mathbb{R}$ y $[a, b] \subset D$. Si f es continua en $[a, b]$, derivable en (a, b) , y $f(a) = f(b)$, entonces existe $c \in (a, b)$, tal que $f'(c) = 0$.*

Los teoremas de Bolzano y de Rolle se generalizan a valores intermedios arbitrarios. Estas generalizaciones son los llamados Teorema del Valor Intermedio (también conocido como Teorema de Darboux) y Teorema del Valor Medio (también conocido como Teorema de Lagrange), respectivamente.

Teorema A.2.3. *Sea $f : D \subset \mathbb{R} \rightarrow \mathbb{R}$, $[a, b] \subset D$ y supongamos f es continua en $[a, b]$.*

- **Darboux:** *Para todo λ entre $f(a)$ y $f(b)$, existe $c \in (a, b)$ tal que $f(c) = \lambda$.*
- **Lagrange:** *Si f es derivable en (a, b) , existe $c \in (a, b)$ tal que $f'(c) = \frac{f(b)-f(a)}{b-a}$.*

A.3. Valores y vectores propios

Lectura recomendada: *Libro rojo de GAL 2*, Capítulo 2 (“Diagonalización”).

Definición A.3.1 (vector y valor propio). Sea $A \in \mathcal{M}_n(\mathbb{R})$. Decimos que $\mathbf{v} \neq \mathbf{0}$ es un **vector propio de A , asociado al valor propio $\lambda \in \mathbb{K}$** , si se cumple

$$A\mathbf{v} = \lambda\mathbf{v}.$$

Aquí, \mathbb{K} es el cuerpo con el que estamos trabajando. En este curso utilizamos principalmente el cuerpo de los reales $\mathbb{K} = \mathbb{R}$, aunque por momentos consideraremos el cuerpo de los complejos $\mathbb{K} = \mathbb{C}$. \triangle

Una forma de hallar los valores propios de una matriz es encontrando las raíces de su polinomio característico. Este polinomio se define de la siguiente forma.

Definición A.3.2 (polinomio característico). Sea $A \in \mathcal{M}_n(\mathbb{R})$. Su **polinomio característico** es el polinomio p_A de grado n dado por

$$p_A(\lambda) := \det(A - \lambda I).$$

\triangle

Teorema A.3.1. *Los valores propios de A son las raíces del polinomio característico de A en el cuerpo \mathbb{K} . Esto es, λ es un valor propio si y sólo si se cumple $p_A(\lambda) = 0$.*

Una vez que se tienen los valores propios de la matriz A , podemos buscar los vectores propios asociados a cada valor propio.

Teorema A.3.2. Los vectores propios de A asociados al valor propio λ son todos aquellos vectores $\mathbf{v} \neq \mathbf{0}$ que son solución del sistema lineal (compatible indeterminado)

$$A\mathbf{v} = \lambda\mathbf{v} \quad (\Leftrightarrow (A - \lambda I)\mathbf{v} = \mathbf{0}).$$

En otras palabras, los vectores propios de A asociados al valor propio λ son aquellos vectores no nulos que pertenecen al núcleo de la matriz $A - \lambda I$.

Definición A.3.3 (subespacio propio). Los vectores propios de $A \in \mathcal{M}_n(\mathbb{R})$ asociados a un valor propio λ , junto con el vector nulo, forman un subespacio vectorial de \mathbb{R}^n . Éste se denomina **subespacio propio asociado a λ** , y se lo denota S_λ ,

$$S_\lambda := \{\mathbf{v} \in \mathbb{R}^n : A\mathbf{v} = \lambda\mathbf{v}\}.$$

△

Definición A.3.4 (matriz diagonalizable). Decimos que $A \in \mathcal{M}_n(\mathbb{R})$ es **diagonalizable** si existen una matriz diagonal $D \in \mathcal{M}_n(\mathbb{R})$ y una matriz invertible $P \in \mathcal{M}_n(\mathbb{R})$, tales que

$$A = PDP^{-1}.$$

△

Teorema A.3.3. Una matriz $A \in \mathcal{M}_n(\mathbb{R})$ es diagonalizable si y sólo si existe una base de \mathbb{R}^n formada por vectores propios de A . En ese caso, la matriz diagonal D tiene a los valores propios de A en su diagonal, y la matriz P tiene como columnas a la base de vectores propios.

A.4. Ortogonalidad

Lectura recomendada: Libro rojo de GAL 2, Capítulo 4 (“Producto interno y norma”).

Consideramos el **producto interno** usual en \mathbb{R}^n : dados $\mathbf{u} = (u_1, \dots, u_n)^t, \mathbf{v} = (v_1, \dots, v_n)^t$, tenemos

$$\langle \mathbf{u}, \mathbf{v} \rangle := \sum_{i=1}^n u_i v_i.$$

Este producto interno induce una **norma** en \mathbb{R}^n , la llamada norma euclídea,

$$\|\mathbf{v}\| := \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}.$$

Además de permitirnos definir normas, los productos internos nos permiten hablar de **ángulos** entre vectores. En particular, podemos generalizar la noción de perpendicularidad.

Definición A.4.1 (ortogonalidad y ortonormalidad). Decimos que dos vectores $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ son **ortogonales** si $\langle \mathbf{u}, \mathbf{v} \rangle = 0$.

Decimos que un conjunto $C \subset \mathbb{R}^n$ es **ortogonal** si sus elementos son ortogonales dos a dos, esto es, si $\langle \mathbf{u}, \mathbf{v} \rangle = 0$ para todos $\mathbf{u} \neq \mathbf{v} \in C$.

Decimos que un conjunto $C \subset \mathbb{R}^n$ es **ortonormal** si es ortogonal y todos sus elementos tienen norma igual a 1. \triangle

Dado un vector $\mathbf{v} \in \mathbb{R}^n$ y un subespacio $S \subset \mathbb{R}^n$, podemos descomponer \mathbf{v} de forma única como la suma de un vector en S y un vector ortogonal a S . Esto nos conduce a la definición de proyección ortogonal sobre S .

Definición A.4.2 (proyección ortogonal). Sea S un subespacio de \mathbb{R}^n , y sea $\{\mathbf{u}_1, \dots, \mathbf{u}_k\}$ una base ortonormal de S . Dado $\mathbf{v} \in \mathbb{R}^n$, se define su **proyección ortogonal sobre S** como el vector

$$P_S(\mathbf{v}) := \sum_{i=1}^k \langle \mathbf{v}, \mathbf{u}_i \rangle \mathbf{u}_i.$$

 \triangle

Observación A.4.1. Se puede verificar que $P_S(\mathbf{v}) \in S$, y que la definición anterior no depende de la base ortonormal de S escogida. Además, el vector $\mathbf{v} - P_S(\mathbf{v})$ es ortogonal a todo vector de S : sea $\mathbf{w} := \sum_{i=1}^k c_i \mathbf{u}_i \in S$ un vector arbitrario, entonces usando la linealidad del producto interno tenemos

$$\begin{aligned} \langle \mathbf{v} - P_S(\mathbf{v}), \mathbf{w} \rangle &= \left\langle \mathbf{v} - \sum_{i=1}^k \langle \mathbf{v}, \mathbf{u}_i \rangle \mathbf{u}_i, \sum_{j=1}^k c_j \mathbf{u}_j \right\rangle \\ &= \sum_{j=1}^k c_j \langle \mathbf{v}, \mathbf{u}_j \rangle - \sum_{i=1}^k \left(\langle \mathbf{v}, \mathbf{u}_i \rangle \sum_{j=1}^k c_j \langle \mathbf{u}_i, \mathbf{u}_j \rangle \right). \end{aligned}$$

Como $\{\mathbf{u}_1, \dots, \mathbf{u}_k\}$ es un conjunto ortonormal, tenemos $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = 0$ si $i \neq j$ y $\langle \mathbf{u}_i, \mathbf{u}_i \rangle = 1$ para todo i , y por lo tanto

$$\langle \mathbf{v} - P_S(\mathbf{v}), \mathbf{w} \rangle = \sum_{j=1}^k c_j \langle \mathbf{v}, \mathbf{u}_j \rangle - \sum_{i=1}^k c_i \langle \mathbf{v}, \mathbf{u}_i \rangle = 0.$$

 \triangle

Observación A.4.2. Sea $S \subset \mathbb{R}^n$ un subespacio vectorial. Si $\{\mathbf{w}_1, \dots, \mathbf{w}_k\}$ es una base ortogonal (pero no ortonormal) de S , entonces podemos normalizarla tomando $\mathbf{u}_i := \frac{\mathbf{w}_i}{\|\mathbf{w}_i\|} = \frac{\mathbf{w}_i}{\sqrt{\langle \mathbf{w}_i, \mathbf{w}_i \rangle}}$ y tendremos, para $\mathbf{v} \in \mathbb{R}^n$,

$$P_S(\mathbf{v}) = \sum_{i=1}^k \langle \mathbf{v}, \mathbf{u}_i \rangle \mathbf{u}_i = \sum_{i=1}^k \left\langle \mathbf{v}, \frac{\mathbf{w}_i}{\sqrt{\langle \mathbf{w}_i, \mathbf{w}_i \rangle}} \right\rangle \frac{\mathbf{w}_i}{\sqrt{\langle \mathbf{w}_i, \mathbf{w}_i \rangle}} = \sum_{i=1}^k \frac{\langle \mathbf{v}, \mathbf{w}_i \rangle}{\langle \mathbf{w}_i, \mathbf{w}_i \rangle} \mathbf{w}_i.$$

 \triangle

Definición A.4.3 (subespacio generado). Dado un conjunto de vectores $C \subset \mathbb{R}^n$, el **subespacio generado por C** , que denotaremos por $[C]$, es el conjunto⁴ de vectores que pueden ser escritos como combinaciones lineales de vectores en C . Por ejemplo, si tenemos un conjunto formado por k vectores $\mathbf{v}_1, \dots, \mathbf{v}_k$, entonces

$$[\mathbf{v}_1, \dots, \mathbf{v}_k] := \left\{ \mathbf{w} := \sum_{i=1}^k c_i \mathbf{v}_i : c_1, \dots, c_k \in \mathbb{R} \right\}.$$

△

Un resultado clave para nosotros va a ser que todo subespacio $S \subset \mathbb{R}^n$ tiene una base ortonormal.

Teorema A.4.1. *Sea $\{\mathbf{v}_1, \dots, \mathbf{v}_m\}$ una base de un subespacio $S \subset \mathbb{R}^n$. Entonces, existe $\mathcal{B} = \{\mathbf{u}_1, \dots, \mathbf{u}_m\}$ tal que \mathcal{B} es una base ortonormal de S y $[\mathbf{v}_1, \dots, \mathbf{v}_k] = [\mathbf{u}_1, \dots, \mathbf{u}_k]$ para todo $k = 1, \dots, m$.*

Demostración. Incluimos aquí la demostración de este teorema porque la haremos usando el llamado **método de ortonormalización de Gram-Schmidt**.

Comenzamos tomando $\mathbf{w}_1 = \mathbf{v}_1$, de modo que obviamente $[\mathbf{w}_1] = [\mathbf{v}_1]$. A continuación, definimos \mathbf{w}_2 como la diferencia entre \mathbf{v}_2 y su proyección ortogonal sobre $[\mathbf{w}_1]$,

$$\mathbf{w}_2 = \mathbf{v}_2 - \frac{\langle \mathbf{v}_2, \mathbf{w}_1 \rangle}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle} \mathbf{w}_1.$$

Esto da lugar a $[\mathbf{w}_1, \mathbf{w}_2] = [\mathbf{v}_1, \mathbf{v}_2]$ y además $\{\mathbf{w}_1, \mathbf{w}_2\}$ es un conjunto ortogonal, pues

$$\langle \mathbf{w}_1, \mathbf{w}_2 \rangle = \left\langle \mathbf{w}_1, \mathbf{v}_2 - \frac{\langle \mathbf{v}_2, \mathbf{w}_1 \rangle}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle} \mathbf{w}_1 \right\rangle = \langle \mathbf{w}_1, \mathbf{v}_2 \rangle - \frac{\langle \mathbf{v}_2, \mathbf{w}_1 \rangle}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle} \langle \mathbf{w}_1, \mathbf{w}_1 \rangle = 0.$$

Se sigue este proceso inductivamente: dados $\mathbf{w}_1, \dots, \mathbf{w}_{k-1}$, se define \mathbf{w}_k como la diferencia entre \mathbf{v}_k y su proyección ortogonal sobre $[\mathbf{w}_1, \dots, \mathbf{w}_{k-1}]$,

$$\mathbf{w}_k := \mathbf{v}_k - \sum_{i=1}^{k-1} \frac{\langle \mathbf{v}_k, \mathbf{w}_i \rangle}{\langle \mathbf{w}_i, \mathbf{w}_i \rangle} \mathbf{w}_i,$$

y se obtiene un conjunto ortogonal $\{\mathbf{w}_1, \dots, \mathbf{w}_k\}$ que genera el mismo subespacio de \mathbb{R}^n que $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$. Finalmente, normalizamos nuestra base ortogonal: tomamos $\mathbf{u}_i = \frac{\mathbf{w}_i}{\|\mathbf{w}_i\|}$ para cada $i = 1, \dots, m$. □

⁴Se debe probar que $[C]$ es efectivamente un subespacio de \mathbb{R}^n .

A.5. Ecuaciones diferenciales ordinarias

Lectura recomendada: E. Catsigeras. Notas de Ecuaciones Diferenciales para CDIVV.

Una **ecuación diferencial ordinaria** es una igualdad en la que:

- La incógnita es una función $y(t)$ definida y derivable hasta orden k para todo $t \in I \subset \mathbb{R}$. Nuestro objetivo es hallar esta función.
- Aparece en la ecuación alguna de las derivadas de la función y : la derivada primera $y'(t)$, y/o la derivada segunda $y''(t)$, hasta la derivada de orden k .

El **orden** de la ecuación diferencial ordinaria es el orden de la derivada de mayor orden de y que aparezca en la ecuación. Por ejemplo:

$$\begin{aligned} y''(t)y'(t) + \sin(y(t)) = 3 &\text{ es de orden 2,} \\ y'''(t) = 4e^{y'(t)} &\text{ es de orden 3.} \end{aligned}$$

En este material solamente mencionamos formas para resolver analíticamente ciertas ecuaciones de primer y segundo orden.

A.5.1. Ecuaciones de primer orden de variables separables

Las **ecuaciones de primer orden de variables separables** son aquellas que se pueden escribir de la forma

$$y'(t) = A(y)B(t), \quad (\text{A.2})$$

donde A, B son funciones conocidas y suficientemente regulares. Observemos que si existe algún y_0 tal que $A(y_0) = 0$, entonces la función constante $y(t) = y_0$ satisface (A.2). Si buscamos soluciones tales que $A(y(t)) \neq 0$, entonces podemos pasar $A(y)$ dividiendo y tomar primitivas para obtener

$$\int \frac{y'(t)}{A(y(t))} dt = \int B(t) dt + C.$$

Notar que la igualdad vale a menos de una constante aditiva $C \in \mathbb{R}$. Haciendo el cambio de variable $t \mapsto y(t)$ en la integral de la izquierda, tenemos

$$\int \frac{1}{A(y)} dy = \int B(t) dt + C. \quad (\text{A.3})$$

Si podemos calcular ambas integrales, podremos escribir y en función de t ; notemos que no siempre podremos despejar $y(t)$ explícitamente.

Ejemplo A.5.1. Consideremos la ecuación diferencial ordinaria

$$y'(t) = \frac{t}{\operatorname{sen}(y(t))}.$$

Procediendo como describimos arriba, el paso (A.3) toma la forma

$$\int \operatorname{sen}(y) dy = \int t dt + C,$$

y se puede verificar que todas las funciones $y(t)$ que cumplan

$$-\cos(y(t)) = \frac{t^2}{2} + C, \quad C \in \mathbb{R}$$

satisfacen la ecuación diferencial. \triangle

A.5.2. Ecuaciones lineales

Se llama **ecuación diferencial lineal** a aquella en que la dependencia en $y(t)$ es lineal. En general, podemos escribir las ecuaciones diferenciales ordinarias lineales de primer orden como

$$y'(t) + a(t)y(t) = f(t), \tag{A.4}$$

y las de segundo orden como

$$y''(t) + a(t)y'(t) + b(t)y(t) = f(t). \tag{A.5}$$

Aquí, las funciones a, b, f son datos. Cuando $f = 0$, estas ecuaciones se dicen **homogéneas**.

El conjunto de soluciones a ecuaciones diferenciales lineales tiene una estructura relativamente sencilla. Para problemas homogéneos, el conjunto de soluciones es un espacio vectorial de dimensión igual al orden de la ecuación. En cambio, para problemas no homogéneos, el conjunto de soluciones es un conjunto afín: en otras palabras, toda solución de (A.4) o (A.5) en el caso no homogéneo se puede escribir como una solución particular más una solución del correspondiente problema homogéneo. Esto motiva el siguiente resultado.

Teorema A.5.1 (Solución general). *Consideremos (A.4) o (A.5), y sea y_P una solución particular de esa ecuación. Entonces, toda otra solución de la ecuación se puede escribir de la forma*

$$y(t) = y_H(t) + y_P(t),$$

donde y_H es una solución de la ecuación homogénea asociada (poniendo $f = 0$ en (A.4) o (A.5), respectivamente).

Esto sugiere el siguiente procedimiento para tratar con (A.4) o (A.5): primero se busca la solución general del problema homogéneo, y luego se busca una solución particular de la ecuación no homogénea. Una forma de lograr esto último es mediante el método de variación de constantes o el método de selección.

Ecuaciones de coeficientes constantes

Cuando a es constante en (A.4), o cuando a, b lo son en (A.5), esas ecuaciones se dicen de **coeficientes constantes**. Escribimos la solución general de los problemas homogéneos asociados y marcamos algún método para hallar una solución particular de la ecuación no homogénea.

Para la **ecuación de primer orden** (A.4), el problema homogéneo toma la forma $y'(t) + ay(t) = 0$, y es sencillo verificar que la solución general es

$$y_H(t) = Ce^{-at}, \quad \text{con } C \in \mathbb{R}.$$

Para buscar una solución particular, el *método de variación de constantes* propone buscar una solución del tipo

$$y_P(t) = C(t)e^{-at},$$

donde ahora $C(t)$ es una función desconocida. Derivando e imponiendo que y_P satisface (A.4), nos encontramos con la condición

$$C'(t)e^{-at} = f(t).$$

Basta con hallar *una* función C que cumpla esta ecuación para tener nuestra solución particular.

Ejemplo A.5.2. Encontremos todas las soluciones de la ecuación diferencial

$$y'(t) + 2y(t) = e^t. \quad (\text{A.6})$$

La solución general de la ecuación homogénea es $y_H(t) = Ce^{-2t}$, y si buscamos soluciones particulares de la forma $y_P(t) = C(t)e^{-2t}$ tenemos que tiene que cumplirse

$$C'(t)e^{-2t} = e^t.$$

Por lo tanto, tiene que ser $C'(t) = e^{3t}$ e integrando encontramos la solución particular $y_P(t) = \frac{e^{3t}}{3}e^{-2t} = \frac{e^t}{3}$. Concluimos que todas las soluciones de (A.6) se pueden escribir como

$$y(t) = Ce^{-2t} + \frac{e^t}{3}, \quad \text{con } C \in \mathbb{R}.$$

△

Para hallar la solución general del problema homogéneo asociado a la **ecuación de segundo orden** (A.5), se considera la *ecuación característica*

$$p(\lambda) := \lambda^2 + a\lambda + b. \quad (\text{A.7})$$

Conociendo las raíces de este polinomio asociado, se puede hallar la solución general de la ecuación homogénea.

Teorema A.5.2 (Solución general homogénea). *Sean λ_1 y λ_2 las raíces del polinomio (A.7). Entonces, la solución general de la ecuación homogénea asociada a (A.5) es*

$$y_H(t) = \begin{cases} C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t}, & \lambda_1 \neq \lambda_2, \lambda_1, \lambda_2 \in \mathbb{R}, \\ (C_1 + C_2 t) e^{\lambda_1 t}, & \lambda_1 = \lambda_2 \in \mathbb{R}, \\ e^{\alpha t} (C_1 \cos(\beta t) + C_2 \sin(\beta t)), & \lambda_1 = \alpha + i\beta = \overline{\lambda_2} \in \mathbb{C}. \end{cases}$$

Arriba, $C_1, C_2 \in \mathbb{R}$ son arbitrarios.

La búsqueda de soluciones particulares para ecuaciones lineales de segundo orden es más engorrosa, y muchas veces se apela al *método de selección*. Referimos a las notas de Ecuaciones Diferenciales para CDIVV para detalles al respecto.

Bibliografía

- [GVL13] G. Golub and C. Van Loan. *Matrix computations*. Johns Hopkins University Press, 2013.
- [Hea02] M. Heath. *Scientific Computing*. McGraw-Hill, 2002.
- [Mol04] C. Moler. *Numerical computing with MATLAB*. SIAM, 2004. Capítulos disponibles gratuitamente en <https://la.mathworks.com/moler/chapters.html>.
- [Mol20] C. Moler. How the SVD saves the universe? Video de charla. Disponible en https://www.youtube.com/watch?v=0bUcD1TkDdY&ab_channel=Matematick%C3%A9probl%C3%A9mynematematik%C5%AF, 2020.
- [Ngu21] Trung Nguyen. The convergence of the regula falsi method. *arXiv preprint arXiv:2109.03523*, 2021.
- [Pic] J. Piccini. Complemento sobre ecuaciones no lineales. Disponible en la página del curso.
- [QS06] A. Quarteroni and F. Saleri. *Cálculo científico con MATLAB y Octave*. Springer, 2006.
- [QSS10] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*, volume 37. Springer, 2010.
- [Sau12] T. Sauer. *Numerical Analysis*. Pearson, 2012.
- [SM03] E. Süli and D. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.