



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

Universidad Técnica Federico Santa María

Departamento de Informática

Draft

**Apuntes de
Computación Científica**

Versión: 2024-v0.613 - 11 de marzo de 2024

Profesor: Claudio E. Torres L., Ph.D.
y un gran equipo, ver capítulo [10](#).

Índice general

Prefacio	7
1. Breve Introducción a Álgebra Lineal	9
1.1. Producto matriz-vector y matriz-matriz	10
1.2. Espacios Vectoriales y aplicaciones lineales	11
1.2.1. Espacio Vectorial	11
1.2.2. Subespacio Vectorial	12
1.2.3. ¿Qué significa hacer un cambio de base?	12
1.2.4. Transformación Lineal	14
1.2.5. Combinación lineal	14
1.2.6. Dependencia lineal	14
1.2.7. Independencia lineal	15
1.2.8. Ejemplo Integrador	15
1.3. Propiedades de las matrices y algunos operadores importantes	18
1.3.1. Rango	18
1.3.2. Nullspace	18
1.3.3. Transpuesta	19
1.3.4. Column Space, Row Space y Rank	19
1.3.5. Full Rank	19
1.3.6. Inversa	19
1.3.7. Adjoint and Adjugate (or classical Adjunct matrix) of a Matrix	20
1.3.7.1. Adjoint of a Matrix	20
1.3.7.2. Adjugate of a Matrix	20
1.3.8. Determinante	21
1.3.9. Traza	21
1.3.10. Valores y vectores propios de una matriz	22
1.3.11. Resumen de propiedades de una matriz invertible	22
1.4. Producto interno, producto externo y algunas matrices particulares	23
1.4.1. Producto Interno	23
1.4.2. Producto externo	24
1.4.3. Algunas matrices particulares	24
1.4.3.1. Matriz diagonal	24
1.4.3.2. Matrices triangulares	25
1.4.3.3. Matrices unitarias	26
1.5. Normas	27
1.5.1. Normas Vectoriales	27
1.5.1.1. Las normas más utilizadas:	28

1.5.1.2. Norma lp con pesos	28
1.5.2. Normas Matriciales	29
1.5.2.1. Normas Matriciales inducidas por normas vectoriales	29
1.5.2.2. Norma de Frobenius	33
1.6. Ejercicios Propuestos	35
2. Estándar de punto flotante y pérdida de importancia	36
2.1. Números Binarios	38
2.2. Estándar de “Punto flotante” en base 2 de Números Reales (R):	39
2.2.1. Formatos IEEE 754	40
2.2.2. El número 1 y 2 en “double precision”	40
2.3. Machine Epsilon	41
2.4. Regla IEEE de redondeo al más cercano	42
2.5. Importancia de “Machine Epsilon”	43
2.6. Representación de Máquina	44
2.7. Caso especial del exponente 1111111111	45
2.8. Caso especial del exponente 0000000000	46
2.9. Pérdida de Importancia	47
2.10. Ejercicios Propuestos	51
3. Raíces en 1D	52
3.1. Método de la Bisección	53
3.1.1. Algoritmo	54
3.1.2. ¿Cuál es el error y qué tan rápido converge el método de la bisección?	55
3.2. Iteración de Punto Fijo	56
3.2.1. Algoritmo	56
3.2.2. Ejemplos de $g(x)$	57
3.2.3. Geometría de la iteración de Punto Fijo – Diagrama Cobweb	58
3.2.4. Convergencia lineal de iteración de punto fijo	60
3.3. Teoremas y definiciones útiles	62
3.4. Método de Newton-Raphson	64
3.4.1. Introducción	64
3.4.2. Algoritmo y ejemplo numérico	65
3.4.3. Convergencia Cuadrática del Método de Newton	65
3.4.4. Convergencia Lineal del Método de Newton	67
3.5. Método de la Secante	69
3.6. Criterios de detención de los algoritmos discutidos	70
3.7. Sensibilidad en la búsqueda de raíces	73
4. Sistemas de Ecuaciones Lineales	75
4.1. Métodos conocidos de resolución	76
4.1.1. Por sustitución	76
4.1.2. Usando Regla de Cramer	76
4.1.3. ¿Qué aprendimos del método por sustitución y de la regla de Cramer?	77
4.2. Eliminación Gaussiana Simple y su complejidad computacional	77
4.3. Backward Substitution	80
4.3.1. Número de operaciones elementales	81
4.4. Factorización LU de A	82
4.4.1. Utilización de la factorización LU	84
4.4.2. ¿Cuándo es útil la factorización LU?	87

4.4.3. Número de condición kappa de un sistema de ecuaciones lineales	88
4.5. Factorización PA = LU	90
4.6. Método de Newton en R elevado a n	93
4.7. Métodos Iterativos	95
4.7.1. Método de Jacobi	97
4.7.2. Método de Gauss-Seidel	99
4.7.3. EXTRA: Succesive Over-Relaxation (SOR(w))	100
4.7.4. Convergencia de métodos iterativos para sistemas de ecuaciones lineales	100
4.7.5. Radio espectral y su uso en convergencia de métodos iterativos	101
4.7.6. Matrices diagonal dominante y su convergencia en métodos iterativos clásicos	102
4.7.7. Teorema de los círculos de Gershgorin	102
4.7.8. Comentarios varios	103
4.8. ¿Existen algoritmos especializados para matrices particulares?	104
5. Interpolación Polinomial en 1D	105
5.1. Matriz de Vandermonde	107
5.2. Interpolación de Lagrange	108
5.3. Interpolación Baricéntrica	111
5.4. EXTRA: Diferencias divididas de Newton	113
5.4.1. Ejemplo	114
5.4.2. Algoritmicamente	114
5.5. Ejemplo numérico de interpolación polinomial	114
5.6. De interpolación polinomial de puntos arbitrarios a interpolación polinomial de funciones	116
5.7. Error de Interpolación y Fenómeno de Runge	117
5.8. Puntos de Chebyshev	118
5.9. Ejemplo (3.10) del libro guía	123
5.10. Cambio de Intervalo	123
5.11. ¿Existen otros algoritmos de interpolación?	124
6. Mínimos Cuadrados	125
6.1. Interpolación vs Aproximación de Mínimos Cuadrados	125
6.2. Primera Alternativa: Mínimos cuadrados por minimización	127
6.3. Segunda Alternativa: Mínimos Cuadrados desde el Álgebra Lineal y las Ecuaciones Normales.	129
6.3.1. Ejemplo numérico de mínimos cuadrados	133
6.4. Tipos de modelos	133
6.4.1. Modelo lineal	134
6.4.2. Modelo cuadrático	134
6.4.3. Modelo periódico	134
6.4.4. Modelo exponencial	135
6.4.5. Ley de potencia	137
6.5. ¿Qué sabemos de TransposeConjugate(A)A?	138
6.6. Factorización QR	140
6.6.1. Estructura de Q y R	140
6.6.2. Ortonormalización de Gram-Schmidt	141
6.6.3. Ortonormalización de Gram-Schmidt modificada	147
6.6.4. ¿Por qué y cómo usamos la factorización QR para encontrar la solución que minimiza el error cuadrático?	147
6.6.5. Ejemplo numérico	149

7. GMRes: Método del residuo mínimo generalizado	151
7.1. Motivación	151
7.2. Derivación de GMRes	153
7.3. ¿Son linealmente independientes b, Ab, \dots ?	161
7.4. ¿Realmente se puede solo usar Aq_k y no $Ak b$?	163
7.5. GMRes visita a Sylvester	163
7.6. GMRes visita a Newton	166
8. Integración Numérica (Cuadratura)	169
8.1. Suma de Riemann	169
8.2. Comentario previo a la presentación de los próximos algoritmos	171
8.3. Regla del Punto Medio	171
8.3.1. Pregunta sobre la Regla del Punto Medio	172
8.4. Regla del Trapecio	173
8.4.1. Pregunta sobre la Regla del Trapecio	174
8.5. Regla de Simpson	174
8.5.1. Pregunta sobre la Regla de las Parábolas de Simpson	176
8.6. Cuadratura Gaussiana - versión reducida	176
8.6.1. Preguntas sobre la Cuadratura Gaussiana	178
8.7. Cuadratura Gaussiana - versión extendida	178
9. Introducción a resolución numérica de ODE	184
9.1. Problemas de Valor Inicial - IVP	184
9.1.1. Notación para IVP	186
9.1.2. Método de Euler	186
9.1.3. Backward Euler	189
9.1.4. Análisis del Error inducido por el Método de Euler	189
9.1.5. RK2: Runge-Kutta de segundo orden	191
9.1.6. RK4: Runge-Kutta de cuarto orden	192
9.1.7. EXTRA: Derivación alternativa de RK2 y más	193
9.1.8. Sistemas Dinámicos	196
9.1.8.1. El péndulo: Un ejemplo simple de un sistema dinámico	197
9.1.9. EXTRA: Análisis de Estabilidad Lineal	198
9.2. Problemas de Valor de Frontera - BVP	201
9.2.1. Método del Disparo	201
9.2.2. Diferencias Finitas para ODE	204
9.2.3. EXTRA: Otra forma de obtener aproximaciones de diferencias finitas	206
10. Agradecimientos y registro de cambios	208
A. SVD: Descomposición en Valores Singulares	221
A.1. Introducción	221
A.2. Interpretación Geométrica	221
A.3. Computación de la SVD	223
A.4. SVD Reducida y SVD Completa	225
A.4.1. Descomposición de Valores Propios	227
A.5. Algunas propiedades importantes de la SVD	228
A.6. Ejemplo numérico de la computación de la SVD	230

B. PCA: Análisis de Componentes Principales	232
B.1. Introducción	232
B.2. Algunos conceptos preliminares	235
B.3. PCA en 5 pasos, paso 1: centrar los datos	236
B.4. PCA en 5 pasos, paso 2: entender qué significa un cambio de base	237
B.5. PCA en 5 pasos, paso 3: varianza y covarianza	239
B.6. PCA en 5 pasos, paso 4: construyendo las componentes principales	241
B.6.1. La conexión secreta de PCA con la SVD	242
B.7. PCA en 5 pasos, paso 5: escribiendo los datos originales con la nueva base	243
B.8. ¿Por qué queremos re-escribir la matriz de datos utilizando las componentes principales?	244
B.9. EXTRA: Compresión de imágenes con la SVD	245
C. Splines Cúbicas	247
C.1. Propiedades de Spline Cúbica	248
C.1.1. Propiedad 1 (Continuidad)	248
C.1.2. Propiedad 2 (Diferenciabilidad)	249
C.1.3. Propiedad 3 (Continuidad en la segunda derivada)	249
C.2. Interpretación de las 3 propiedades	249
C.3. Cantidad de ecuaciones a satisfacer	249
C.4. Tipos de condiciones de borde para splines cúbicas	250
C.4.1. Spline Natural	250
C.4.2. Spline con curvatura ajustada	250
C.4.2.1. Spline cúbica fijada (<u>Clamped cubic spline</u>)	250
C.4.3. Spline Terminada parabólicamente	250
C.4.4. Spline cúbica sin nodo (<u>Not-a-knot cubic spline</u>)	251
C.5. Unicidad de la Spline Cúbica	251
C.5.1. Ejemplo	251
C.6. Ejercicios Propuestos	252
D. Algoritmos para matrices simétricas y definidas positivas	253
D.1. La Factorización de Cholesky	253
D.1.1. Submatriz	253
D.1.2. Propiedades de una matriz definida positiva y simétrica	253
D.1.2.1. Caso base	254
D.1.3. Descomposición de matrices simétricas y definida positiva	254
D.1.3.1. Caso Base	254
D.1.3.2. Caso General	255
D.2. Método del Gradiente Descendente	257
D.2.1. Minimizar una función cuadrática convexa	257
D.2.2. ¿Qué es y cómo se obtiene $\alpha_{\text{sub } k}$	257
D.2.2.1. ¿Cómo se obtiene el óptimo de $f(\alpha)$?	258
D.2.2.2. Algoritmo	258
D.2.2.3. Gráficamente	258
D.2.3. Convergencia en matrices simétricas y definida positiva	258
D.3. Método del Gradiente Conjugado	259
D.3.1. Derivación del Método del Gradiente Conjugado	260
D.3.2. Análisis de la A-ortogonalidad	263
D.3.3. Gradiente Conjugado - Interpretación Gráfica	266
D.3.4. Convergencia del Algoritmo del Gradiente Conjugado para una matriz simétrica y definida positiva	266

D.3.5. Gradiente Conjugado - Versión 2	266
D.3.5.1. Ejemplo	267
E. Computación de Pesos y Nodos de la Cuadratura Gaussiana	268
E.1. Definiciones y teoremas bases	268
E.2. Nodos de la Cuadratura Gaussiana	268
E.3. Pesos de la Cuadratura Gaussiana	269

Prefacio

¡Bienvenid@s a los apuntes del curso de Computación Científica!

Estos apuntes nos acompañarán durante todo el curso. Los apuntes han sido desarrollados y mejorados a lo largo de varios años de trabajo y much@s voluntari@s que han corregido y mejorado diversas explicaciones y contenidos. Se agradece a cada un@ de los y las voluntari@s por sus valiosos aportes en el capítulo 10 y se invita a utilizar el formato de sugerencias y/o comentarios que se mencionará en clases. *Las sugerencias se van incluyendo durante el semestre por lo que se invita a no imprimir los apuntes completos a comienzo de semestre y estar atentos al cambio de versión de los apuntes.* Notar que cada vez que se suba una nueva versión, se detalla el registro de cambio en el capítulo 10, en la última tabla incluida.

El curso de Computación Científica en el Departamento de Informática de la Universidad Técnica Federico Santa María tiene como objetivo principal diseñar, analizar e implementar algoritmos eficientes para la solución de problemas continuos en Matemáticas e Ingeniería. El desarrollo de **algoritmos** en el curso se basa fuertemente en Matemática Aplicada. Las habilidades a desarrollar incluyen el análisis, síntesis, resolución computacional de los problemas discutidos, además de la **evaluación crítica** de los **resultados computacionales obtenidos**. Cada una de estas habilidades se aplicarán a cada uno de los contenidos temáticos del curso.

Los resultados de aprendizaje están fuertemente ligados al objetivo del curso indicado anteriormente, los cuales se detallan a continuación:

- Analiza problemas de ingeniería, seleccionando la estructura matemática adecuada.
- Propone un algoritmo de resolución para el problema, resolviéndolo mediante las técnicas o métodos numéricos correspondientes.
- Analiza los resultados de un método numérico, verificando sus propiedades teóricas y numéricas.
- Analiza métodos computacionales.

Para el exitoso cumplimiento de los resultados de aprendizaje planteados, se utilizará una metodología de trabajo interactiva, donde *se necesita una participación activa de tod@s*. Las clases consideran una componente de trabajo previo y una componente expositiva de parte del profesor, además de diversas actividades.

Es decir, **se sugiere encarecidamente estudiar las secciones de los apuntes indicadas en el syllabus antes de cada clase para aprovechar al máximo la clase**. Por ejemplo se puede aprovechar el proyecto [The elixir of Scientific Computing](#) para estudio previo.

Los contenidos a discutir incluyen los siguientes temas:

- Breve introducción de Álgebra Lineal.
- Estándar de punto flotante y pérdida de importancia.

- Búsqueda de ceros en ecuaciones no lineales en 1D.
- Sistemas de ecuaciones lineales y no lineales, métodos clásicos y avanzados.
- Interpolación en 1D.
- Mínimos cuadrados.
- Integración numérica.
- Ecuaciones diferenciales ordinarias.

Es importante destacar en este punto que los contenidos son *integradores*, es decir, lo aprendido en los temas iniciales se aplica a los temas posteriores. **Por lo que es muy importante ir entendiendo cada tema e ir reconociendo cuando aplicarlo.**¹ Por ejemplo, el primer tema es Álgebra Lineal y este se aplica directamente en casi todos los contenidos posteriores. Lo mismo ocurre con el Estándar de Punto Flotante, y así sucesivamente. En particular, en el Estándar de Punto Flotante se discute la pérdida de importancia numérica, el cuál es un tema crucial en la construcción de algoritmos numéricos efectivos.

Una sugerencia para este curso es balancear adecuadamente las habilidades de mecanización y de análisis crítico, en este curso se trabaja activamente con ambas. Por ejemplo, para cada actividad se sugiere primeramente realizar un análisis crítico de lo presentado para luego aplicar lo aprendido y ejecutar las acciones necesarias para resolver lo solicitado². Esto es muy importante dado que los contenidos van incrementando la cantidad de herramientas disponibles, por lo que es muy importante determinar el tipo de problema presentado antes de poder resolverlo. Lo otro importante es diferenciar el contexto de la problemática. Por ejemplo, considere el siguiente problema: *Determine el valor de “a” tal que se cumpla la siguiente ecuación:* $\int_0^a \exp(x) dx = 1$. En este caso el contexto del problema es integración en una variable, sin embargo la problemática es búsqueda de ceros! Por lo cual es muy importante, balancear la mecanización con el análisis crítico para que al finalizar el curso se puedan cumplir exitosamente todos los resultados de aprendizaje!

En los apuntes se incluyen algunas secciones que incluyen la palabra EXTRA en su título, estos contenidos se han incluido como material adicional para el o la estudiante curios@. En algunos casos corresponden a contenidos adicionales y en otros se presentan explicaciones más profundas de contenidos propios del curso. Los apuntes se basan principalmente pero no completamente en el texto guía del curso, el cual es Timothy Sauer. (2018). Numerical Analysis. United States: Pearson [8]. Referencias adicionales importantes incluyen [9] y [7].

Sin más que agregar, espero que el curso les muestre una pincelada de algoritmos y aplicaciones avanzadas para un impacto positivo en la sociedad!

Saludos,

Prof. Claudio Torres, Ph.D.
ctorres@inf.utfsm.cl
 DI-UTFSM
 Marzo 2024

¹Este es el núcleo del curso, reconocer cuando aplicar lo aprendido! Ver la vista del grafo en [The elixir of Scientific Computing](#) para una visualización de la interacción de los temas discutidos.

²Es decir, aplicar la metodología IDEA!

Capítulo 1

Breve Introducción a Álgebra Lineal

En este capítulo revisaremos brevemente algunos elementos matemáticos y definiciones varias que se utilizarán en el curso. Para más detalles se recomienda ver el apéndice A del texto guía: Timothy Sauer. (2017). Numerical Analysis. United States: Pearson. También una muy buena referencia son los textos complementarios:

- Numerical Linear Algebra, Lloyd N. Trefethen and David Bau, III, SIAM, 1997, ISBN: 0898713617.
- Numerical Mathematics, Alfio Quarteroni, Riccardo Sacco, Fausto Saleri, Springer, Text in Applied Mathematics 37, 2000, ISBN: 0387989595.
- Applied Numerical Linear Algebra, James W. Demmel, SIAM, 1997, SIAM, ISBN: 0898713897.

El correcto entendimiento¹ le ayudará a discernir adecuadamente sobre las ventajas y desventajas de los algoritmos que discutiremos en el curso. Esto es crucial, ya que uno de los desafíos es poder determinar cual es el algoritmo adecuado para el problema que se enfrenta.

Las estructuras matemáticas básicas con las que se trabaja en álgebra lineal son vectores y matrices. Un vector de dimensión finita² se puede entender como un arreglo unidimensional de números. Tradicionalmente se consideran que son vectores columnas de la siguiente forma:

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \in \mathbb{R}^n.$$

El cual también se puede transformar matemáticamente a un vector fila aplicándole el operador transpuesta T , es decir $\mathbf{u}^T = [u_1, u_2, \dots, u_n]$. La importancia de utilizar vectores en su formato de columna o fila recae en la correctitud matemática de la expresión matemática asociada. Por ejemplo, si uno quiere obtener el producto interno, también conocido como producto punto para vectores de dimensión finita, se requiere que ambos vectores tengan la misma dimensión. Lo mismo ocurre si uno multiplica una matriz por un vector o un vector por una matriz, se requiere que exista compatibilidad de sus dimensiones. En este caso podemos considerar que un vector columna de dimensión n se puede interpretar como una matriz de dimensión $n \times 1$ y un vector fila de dimensión n se interpretaría como una matriz de dimensión $1 \times n$.

¹Este es un punto muy importante! Ya que si uno entiende la mitad de algún tema y queda conforme, eso puede implicar que uno entienda la mitad de la mitad del siguiente tema, es decir el 25%, y así sucesivamente. Luego de 7 iteraciones solo entenderá correctamente menos del 1% de lo discutido. Moraleja: Entender correctamente!

²¡Hay vectores de dimensión infinita también!

Este breve análisis nos da paso para definir una matriz, en nuestro caso consideramos una matriz $A \in \mathbb{R}^{m \times n}$ como un arreglo bidimensional de números, con m filas y n columnas de la siguiente forma,

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

donde denotamos a A_{ij} el elemento de la i -ésima fila y la j -ésima columna de la matriz A , es decir a_{ij} .

1.1. Producto matriz-vector y matriz-matriz

El producto matriz-vector y matriz-matriz están en el día a día de este curso. No solo es importante entender sus definiciones y fórmulas, sino que también es necesario poder entender las distintas formas de interpretarlo y manipularlos. Por ejemplo considere el siguiente producto matriz-vector entre la matriz $A \in \mathbb{R}^{m \times n}$ y el vector $\mathbf{u} \in \mathbb{R}^n$,

$$A\mathbf{u} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} a_{11} \cdot u_1 + a_{12} \cdot u_2 + \cdots + a_{1n} \cdot u_n \\ \vdots \\ a_{m1} \cdot u_1 + a_{m2} \cdot u_2 + \cdots + a_{mn} \cdot u_n \end{bmatrix}.$$

Del cual podemos notar de inmediato que para que el producto tenga sentido matemático se requiere que la cantidad de columnas de la matriz A sea igual a la dimensión del vector \mathbf{u} , de otra forma, ¡nos sobran o faltan coeficientes! También notamos que la cantidad de filas m parece no afectar el producto. Solo necesitamos que m sea mayor o igual a 1. Un caso particular que veremos prontamente es el caso cuando $m = 1$, ¿qué ocurre en ese caso?³.

El rol que juega la cantidad de filas de A es indicar la dimensión del vector resultante luego de hacer el producto, es decir el vector $\mathbf{w} = A\mathbf{u} \in \mathbb{R}^m$.

Por otro lado, ¿Qué tal si vemos A como una colección de vectores columnas? Es decir,

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = [\mathbf{a}_1 \mid \mathbf{a}_2 \mid \cdots \mid \mathbf{a}_n],$$

donde $\mathbf{a}_k = \begin{bmatrix} a_{1k} \\ \vdots \\ a_{mk} \end{bmatrix}$. Entonces el producto matriz vector puede interpretarse como:

$$A\mathbf{u} = u_1 \mathbf{a}_1 + u_2 \mathbf{a}_2 + \cdots + u_n \mathbf{a}_n = \sum_{k=1}^n u_k \mathbf{a}_k$$

Es decir, el producto matriz-vector se puede entender como una combinación lineal⁴ de las columnas de A ponderadas por los coeficientes de \mathbf{u} . Esta interpretación es muy interesante, dado que nos permite realizar una manipu-

³Se puede interpretar como el producto interno entre el vector $[a_{11}, a_{12}, \dots, a_{1n}]$, es decir la primera fila de A , y \mathbf{u} . Esto lo profundizaremos en las próximas secciones de este capítulo!

⁴Se desarrollará este concepto en las siguientes secciones de este capítulo.

lación algebraica vectorial del producto. Esta manipulación algebraica vectorial es muy útil dado que es fácilmente algoritmizable⁵ con el uso de librerías computacionales adecuadas.

La generalización natural del producto matriz-vector es el producto matriz-matriz. Por ejemplo considere las siguientes matrices: $A \in \mathbb{R}^{m \times n}$ y $B \in \mathbb{R}^{n \times l}$. Entonces el producto de ellas se denota de la siguiente forma,

$$AB = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_i & \cdots & \mathbf{b}_l \end{bmatrix} = C \in \mathbb{R}^{m \times l},$$

es decir, el producto de las matrices A y B nos da la matriz C de dimensiones $m \times l$. En particular, se puede construir la matriz C computando cada uno de sus vectores columna \mathbf{c}_i de la siguiente forma: $\mathbf{c}_i = \sum_{k=1}^n b_{k,i} \mathbf{a}_k = A\mathbf{b}_i$, para $i \in \{1, 2, 3, \dots, l\}$ y donde $b_{k,i}$ es el k -ésimo elemento del vector \mathbf{b}_i . En resumen, $\mathbf{c}_i = A\mathbf{b}_i$, es decir, la matriz C se construye obteniendo el producto matriz-vector entre la matriz A y cada columna de B . Es decir, es solo una colección de productos matriz-vector!

1.2. Espacios Vectoriales y aplicaciones lineales

1.2.1. Espacio Vectorial

Un espacio vectorial es un conjunto no vacío V de objetos, llamados vectores, en el cual hay 2 operaciones definidas, llamadas adición y multiplicación por escalar (números reales o complejos), restringido a los siguientes axiomas. Los axiomas deben ser válidos para todos los vectores $\mathbf{u}, \mathbf{v}, \mathbf{w}$ de V y para todos los escalares c y d .

1. La suma de \mathbf{u} y \mathbf{v} , denotada como $\mathbf{u} + \mathbf{v}$, está en V
2. $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$ (conmutatividad)
3. $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$ (asociatividad de adición)
4. Hay un vector nulo $\mathbf{0}$ en V de tal forma $\mathbf{u} + \mathbf{0} = \mathbf{u}$
5. Para cada \mathbf{u} en V , hay un vector $-\mathbf{u}$ en V de tal forma que $\mathbf{u} + (-\mathbf{u}) = \mathbf{0}$
6. El múltiplo escalar de \mathbf{u} por c , denotado por $c \cdot \mathbf{u}$, está en V
7. $c(\mathbf{u} + \mathbf{v}) = c\mathbf{u} + c\mathbf{v}$
8. $(c + d)\mathbf{u} = c\mathbf{u} + d\mathbf{u}$
9. $c(d\mathbf{u}) = (cd)\mathbf{u}$
10. $1\mathbf{u} = \mathbf{u}$

Por ejemplo: El espacio vectorial representado por \mathbb{R}^2 consiste en todos los vectores columnas con 2 componentes.

⁵Aquí nos referimos a que en Python podemos usar NumPy (<https://numpy.org>) y/o SciPy (<https://www.scipy.org>) que tienen vectores y matrices como estructuras de datos, con las cuales podemos implementar rápidamente algoritmos vectoriales! Por ejemplo multiplicar un escalar por un vector se realiza directamente y no es necesario utilizar un loop para realizar la tarea. Esto disminuye la posibilidad de un bug y reduce el tiempo de desarrollo. Una maravilla!

1.2.2. Subespacio Vectorial

Un subespacio vectorial H de un espacio vectorial V es un sub-conjunto de V que tiene 3 propiedades:

- El vector nulo de V está en H .
- H es cerrado bajo la adición. Esto es, para cada \mathbf{u} y \mathbf{v} en H , la suma $\mathbf{u} + \mathbf{v}$ está en H .
- H es cerrado bajo la multiplicación escalar. Esto es, para cada \mathbf{u} en H y cada escalar c , el vector $c\mathbf{u}$ está en H .

Por ejemplo, considere el siguiente subespacio vectorial H de \mathbb{R}^3 en la Figura 1.1. El subespacio vectorial H (o

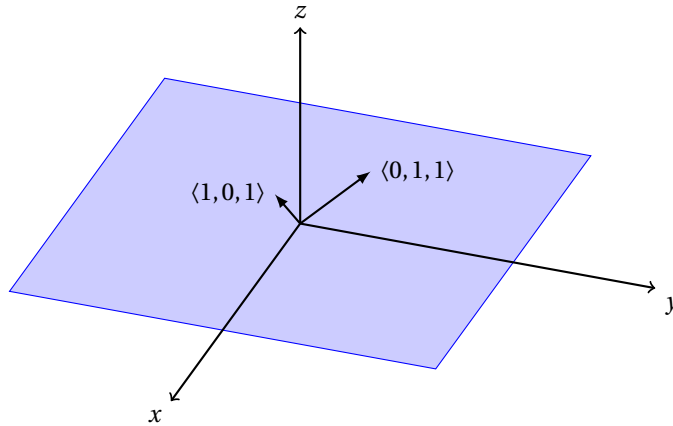


Figura 1.1: Sketch de subespacio vectorial en \mathbb{R}^3 .

el plano en este caso) se define como el conjunto de todos los vectores que se pueden generar por la combinación lineal de los vectores $\langle 0, 1, 1 \rangle$ y $\langle 1, 0, 1 \rangle$, es decir, $a\langle 0, 1, 1 \rangle + b\langle 1, 0, 1 \rangle$, para $a \in \mathbb{R}$ y $b \in \mathbb{R}$. Lo cual corresponde a la definición del span:

$$H = \text{span}(\langle 0, 1, 1 \rangle, \langle 1, 0, 1 \rangle)$$

Por ejemplo, ¿Pertenece $\langle 1, 1, 0 \rangle$ a H ?⁶

1.2.3. ¿Qué significa hacer un cambio de base?

Hacer un cambio de base es simplemente determinar las coordenadas de un vector como la combinación lineal de un conjunto de vectores, idealmente que sean linealmente independiente. Por ejemplo, considere las siguientes alternativas de escribir el vector $\begin{bmatrix} 4 \\ 3 \end{bmatrix}$, ver Figura 1.2. En rojo se observa que para escribir el vector $\langle 4, 3 \rangle$ en la base canónica se requiere multiplicar por 4 el vector canónico \mathbf{i} y por 3 el vector canónico \mathbf{j} , y luego sumarlos, es decir: $4\hat{i} + 3\hat{j} = 4 \cdot \langle 1, 0 \rangle + 3 \cdot \langle 0, 1 \rangle = \langle 4, 3 \rangle$

En la Figura 1.2 se puede visualizar además la descomposición de $\langle 4, 3 \rangle$ en otras bases, identificados con los vectores en verde y azul, respectivamente. El análisis realizado anteriormente con los vectores canónicos puede hacerse con las otras bases. Supongamos que los 2 vectores azules o los 2 vectores verdes pueden escribirse de la siguiente forma: $\mathbf{b}_1 = \langle b_{1,1}, b_{1,2} \rangle$ y $\mathbf{b}_2 = \langle b_{2,1}, b_{2,2} \rangle$, respectivamente⁷. En ese caso, sin embargo, no conocemos los

⁶Para determinar si $\langle 1, 1, 0 \rangle$ pertenece al subespacio vectorial H debemos determinar si existen algún valor particular de a y b en $a\langle 0, 1, 1 \rangle + b\langle 1, 0, 1 \rangle$ tal que puedan representar al vector $\langle 1, 1, 0 \rangle$ exactamente. ¿Existen?

⁷Notar que si bien hemos definido los vectores \mathbf{b}_1 y \mathbf{b}_2 de forma arbitraria, estos se consideran conocidos para este análisis.

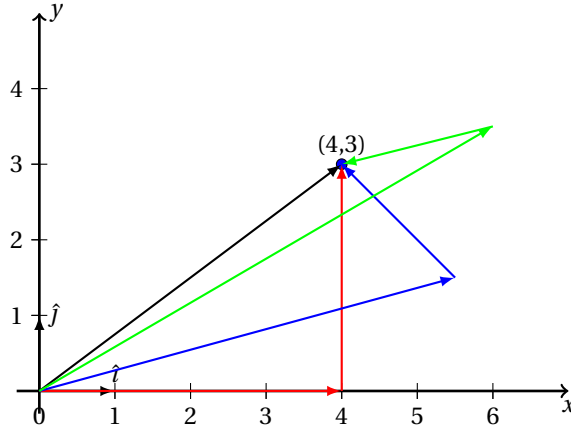


Figura 1.2: Ejemplo de uso de diferentes bases para representar un mismo punto en \mathbb{R}^2 .

escalamientos que debemos hacer a cada vector para poder obtener el vector $\langle 4, 3 \rangle$. Esto lo podemos escribir en la siguiente ecuación vectorial:

$$\alpha \mathbf{b}_1 + \beta \mathbf{b}_2 = \alpha \begin{bmatrix} b_{1,1} \\ b_{1,2} \end{bmatrix} + \beta \begin{bmatrix} b_{2,1} \\ b_{2,2} \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}.$$

Lo cual puede interpretarse como un sistema de ecuaciones lineales de la siguiente forma:

$$\underbrace{\begin{bmatrix} b_{1,1} & b_{2,1} \\ b_{1,2} & b_{2,2} \end{bmatrix}}_A \underbrace{\begin{bmatrix} \alpha \\ \beta \end{bmatrix}}_{\mathbf{b}} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}.$$

Entonces, en este caso el problema se convirtió en resolver un sistema de ecuaciones lineales! Desde el punto de vista Matemático, esto se resuelve de la siguiente forma:

$$A\mathbf{x} = \mathbf{b} \Rightarrow \mathbf{x} = A^{-1}\mathbf{b},$$

donde A^{-1} es denominada la inversa de A . Lo cual puede inducir a pensar que es necesario obtener la matriz inversa de A (i.e. A^{-1}), sin embargo veremos en el curso que eso no es realmente necesario!⁸

⁸Se recomienda encarecidamente evitar obtener la matriz inversa en la mayoría de los casos! Existen excepciones, por ejemplo cuando la matriz es diagonal o ortonormal.

Comentario al margen

Un sistema de ecuaciones lineales muy particular y muy simple es el siguiente: $I\mathbf{x} = \mathbf{c} \Rightarrow \mathbf{x} = \mathbf{c}$, donde la matriz A del sistema de ecuaciones lineales es la matriz identidad I . La cual se define como:

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

En ese caso resolver el sistema de ecuaciones lineales es muy simple! La solución es el mismo lado derecho. En general esto no ocurre frecuentemente, pero si ocurriera, es importante reconocer en este caso que la solución se puede obtener fácilmente.

1.2.4. Transformación Lineal

Se denomina Transformación Lineal a una aplicación tal que su dominio y codominio sean espacios vectoriales y que cumplan lo siguiente,

$$\text{Sea } A \in \mathbb{R}^{n \times n}, \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \text{ y } \alpha \in \mathbb{R} \Rightarrow A(\mathbf{x} + \mathbf{y}) = A\mathbf{x} + A\mathbf{y} \text{ y } A(\alpha\mathbf{x}) = \alpha A\mathbf{x}$$

¡MUY IMPORTANTE!

Revisar Jupyter Notebook asociado a este punto!

1.2.5. Combinación lineal

Una Combinación Lineal de dos o más vectores es el vector que se obtiene al sumar estos vectores multiplicados por escalares.

Nota

Por ejemplo: Sea $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$, $\alpha \neq 0$ y $\mathbf{x} \neq \beta\mathbf{y}$, y considere las siguientes relaciones: $\mathbf{v}_1 = \mathbf{x} + \mathbf{y}$, $\mathbf{v}_2 = \alpha\mathbf{x}$, y $\mathbf{v}_3 = \frac{1}{\alpha}\mathbf{y}$. Preguntas:

1. ¿Son \mathbf{v}_2 y \mathbf{x} linealmente independientes?^a
2. ¿Son \mathbf{v}_1 , \mathbf{x} e \mathbf{y} linealmente independientes?^b
3. ¿Son \mathbf{v}_2 y \mathbf{v}_3 linealmente independientes?^c

^aNo, dado que la definición de $\mathbf{v}_2 = \alpha\mathbf{x}$, es decir, es solo un escalamiento de \mathbf{x} , por lo cual hay una dependencia lineal.

^bNo, dado que \mathbf{v}_1 es efectivamente la suma de \mathbf{x} y \mathbf{y} .

^cSí, dado que $\mathbf{x} \neq \beta\mathbf{y}$, no hay forma de escribir \mathbf{v}_2 como una combinación lineal de \mathbf{v}_3 . Otra forma de verlo es la siguiente: La única solución a la siguiente ecuación $\lambda_0\mathbf{v}_2 + \lambda_1\mathbf{v}_3 = \mathbf{0}$ son las constantes $\lambda_0 = \lambda_1 = 0$, es decir, no hay **Dependencia Lineal**.

1.2.6. Dependencia lineal

Se dice que ciertos vectores \mathbf{x}_i son Linealmente Dependientes si hay una combinación lineal de ellos que es igual al vector cero, sin que todos coeficiente de la combinación lineal sea cero, es decir, existe una solución no

nula para los coeficientes λ_i . Matemáticamente se expresa de la siguiente forma:

$$\sum_{i=1}^m \lambda_i \mathbf{x}_i = \mathbf{0} \Rightarrow \exists \lambda_i \neq 0.$$

Nota

En este caso solo sabemos que existe dependencia lineal, pero no sabemos cómo es la dependencia lineal.

1.2.7. Independencia lineal

Se dice que ciertos vectores \mathbf{x}_i son Linealmente Independientes si ninguno de ellos puede ser escrito con una combinación lineal de los restantes. Esto puede ser escrito Matemáticamente de la siguiente forma:

$$\sum_{i=1}^m \lambda_i \mathbf{x}_i = \mathbf{0} \Rightarrow \lambda_i = 0, \quad \forall i \in \{1, \dots, m\}.$$

1.2.8. Ejemplo Integrador

Considere el siguiente problema: $x + y = 1$, $ax + y = b$. Aquí se considera que a y b son “conocidos” y se debe buscar x y y .

Lo primero que notamos es que el problema en cuestión es un sistema de ecuaciones lineales de 2×2 . En este caso, como son solo 2 variables, podemos despejar y y obtener las gráficas de cada función lineal considerando el caso cuando $a \leq 0$ y $a > 0$. Por simplicidad se considerará que $b = 2$, luego se analizará en más detalle esto. Entonces,

$$\begin{aligned} ax + y = b &\Rightarrow y = b - ax \\ x + y = 1 &\Rightarrow y = 1 - x \end{aligned}$$

La Figura 1.3 muestra un sketch del ejemplo.

Al realizar un análisis preliminar, notamos que tenemos 3 casos:

1. $a = 1$ y $b = 1$.
2. $a \neq 1$.
3. $a = 1$ y $b \neq 1$.

Las implicaciones se resumen en la Tabla 1.1.

Adicionalmente, y no menos importante, podemos interpretar el mismo resultado anterior de la siguiente forma. Las 2 ecuaciones, $x + y = 1$ y $ax + y = b$, pueden escribirse matricialmente de la siguiente forma:

$$\begin{bmatrix} 1 & 1 \\ a & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ b \end{bmatrix}. \quad (1.1)$$

Comentario al margen

El lado derecho de la ecuación anterior, también puede escribirse como una combinación lineal de 2 vectores de la siguiente forma:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ b \end{bmatrix} = 1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + b \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Sin embargo, en general, se omiten, pero es importante saber que sí es posible. Note que los vectores que aparecen son los vectores canónicos $\langle 1, 0 \rangle$ y $\langle 0, 1 \rangle$, respectivamente.

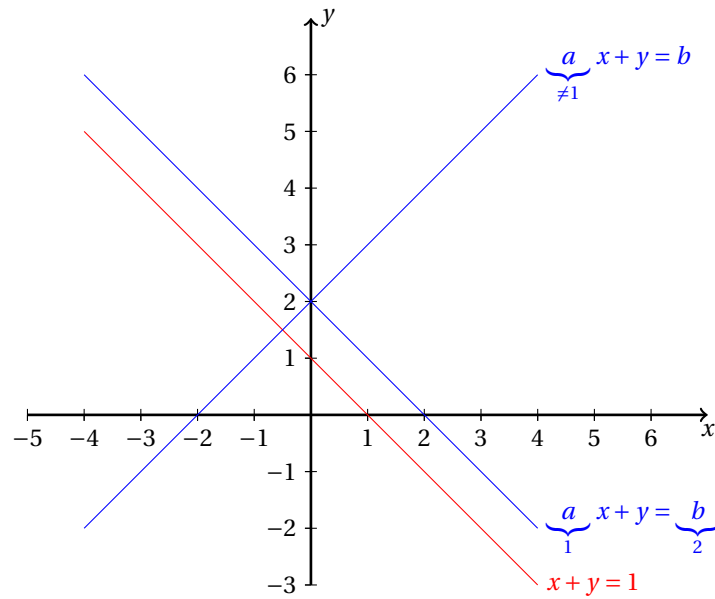


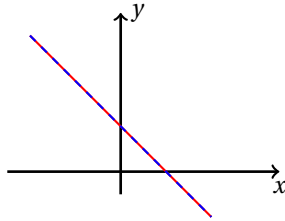
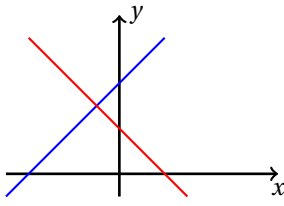
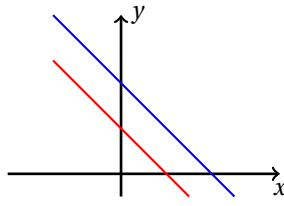
Figura 1.3: Interpretación de casos especiales para el sistema 2×2 en estudio. Note que para el caso $ax + y = b$, con $a = b = 1$ se obtiene que la recta en rojo y azul serían las mismas, lo que implica que hay ∞ soluciones! El caso actual con $a = 1$ y $b \neq 1$ implica que no hay solución porque las rectas son paralelas pero con diferente intercepto, y por último para $a \neq 1$ hay una única solución, que es justo donde se interceptan las rectas.

Ahora, el producto matriz-vector en el lado izquierdo de la ecuación (1.1) se puede representar de la siguiente forma:

$$x \underbrace{\begin{bmatrix} 1 \\ a \end{bmatrix}}_{\mathbf{v}_1} + y \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{\mathbf{v}_2} = \underbrace{\begin{bmatrix} 1 \\ b \end{bmatrix}}_{\mathbf{b}},$$

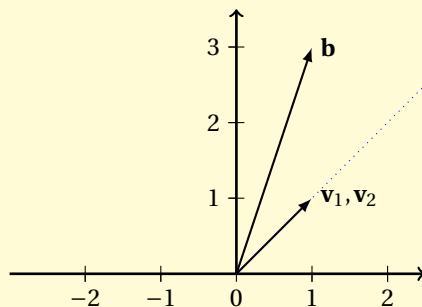
donde x y y se interpretan como los escalamientos necesarios de los vectores \mathbf{v}_1 y \mathbf{v}_2 para representar el vector \mathbf{b} . En la Figura 1.4 podemos ver otra interpretación gráfica del mismo sistema de ecuaciones lineales de 2×2 en estudio.

Tabla 1.1: Tres casos en un sistema de ecuaciones de 2×2 .

Caso 1	Caso 2	Caso 3
$a = 1$	$a \neq 1$	$a = 1$
$b = 1$		$b \neq 1$
$y = 1 - x$	$x = \frac{b-1}{a-1} \wedge y = \frac{a-b}{a-1}$	
∞ soluciones	1 solución	No existe solución
		
Existe sobreposición de rectas	Las rectas intersectan en un solo punto	Las rectas son paralelas

¡MUY IMPORTANTE!

Considerando el caso de $a = 1$, independiente del valor de b , obtenemos la siguiente interpretación gráfica:



De la cual surgen las siguientes preguntas:

- ¿Cuándo no hay solución del sistema de ecuaciones lineales en estudio?^a
- ¿Cuándo hay solución del sistema de ecuaciones lineales en estudio?^b
- ¿Cuál es la interpretación gráfica del caso cuando sí hay solución del sistema de ecuaciones lineales?

^aEn este caso **no** hay solución si **b** **no** pertenece al span de v_1 o el span de v_2 .

^bEn este caso hay solución si **b** pertenece al span de v_1 o el span de v_2 .

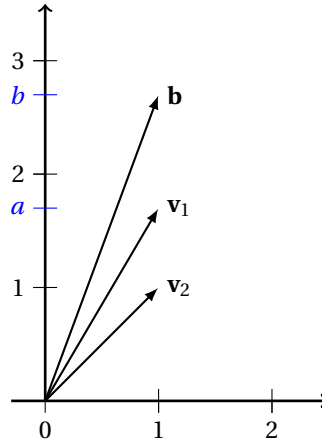


Figura 1.4: Segunda interpretación de la solución del sistema de ecuaciones lineales de 2×2 en estudio.

1.3. Propiedades de las matrices y algunos operadores importantes

1.3.1. Rango

El “Rango” de una matriz A , denotado como $\text{Range}(A)$, es el espacio vectorial “spanned” (generado)⁹ por las columnas de A .

Por ejemplo, considere la siguiente matriz $A = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \dots \quad \mathbf{a}_n] \in \mathbb{R}^{n \times n}$, entonces el rango de A es el espacio vectorial generado por los vectores columnas de A , es decir, \mathbf{a}_i , para $i \in \{1, 2, \dots, n\}$. Notar que la dimensión del espacio vectorial generado puede ser hasta n . En el caso de que existieran columnas linealmente dependientes, la dimensión del espacio generado será menor que n .

1.3.2. Nullspace

Se define el Nullspace de una matriz $A \in \mathbb{R}^{m \times n}$ como:

$$\text{Null}(A) = \{\mathbf{x} : \mathbf{x} \in \mathbb{R}^n \text{ y } A\mathbf{x} = \mathbf{0}\}$$

Una primera mirada a esta definición da a entender que la solución nula, es decir $\mathbf{x} = \mathbf{0}$, satisface la definición. Si bien esta percepción es correcta, uno busca la existencia de soluciones no nulas y eventualmente la dimensionalidad del sub-espacio generado.

Por ejemplo, considere $\mathbf{a}_1 \neq \mathbf{0}$, $A\mathbf{x} = [\mathbf{a}_1 \quad | \quad \mathbf{0}] \cdot \begin{bmatrix} 0 \\ 1 \\ c \end{bmatrix} = \mathbf{0}$, en este caso $\text{Null}(A) = \text{span}(\langle 0, 1 \rangle)$. Es decir, son todos los vectores que se pueden generar multiplicando el vector $\langle 0, 1 \rangle$ por un escalar, dado que al multiplicar cualquiera de estos vectores por la matriz A el resultado seguirá siendo el vector nulo $\mathbf{0}$. En este ejemplo la dimensionalidad del $\text{Null}(A)$ es 1, dado con solo 1 vector no-nulo se genera todo el nullspace.

⁹El concepto de span es importante y será utilizado regularmente, por lo cual es importante recordarlo.

1.3.3. Transpuesta

La Transpuesta de una matriz A se denota como la matriz A^T y corresponde a alternar el orden de los sub-índices, es decir $A_{i,j} = A_{j,i}^T$. Por ejemplo,

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \end{bmatrix}$$

Una propiedad importante de la transpuesta de una matriz es la siguiente: $(AB)^T = B^T A^T$. Note que el orden se alterna al aplicar la transpuesta.

1.3.4. Column Space, Row Space y Rank

- Column Space: Es el espacio generado por los vectores columnas de una matriz. Es lo mismo que el $\text{Range}(A)$.
- Column Rank: es la dimensión del column space, es decir, el número de columnas linealmente independientes.
- Row Space: Es el espacio generado por los vectores **filas** de la matriz A . Equivalentemente, corresponde al $\text{Range}(A^T)$.
- Row Rank: es la dimensión del row space, es decir, el número de filas linealmente independientes.

Es importante destacar que el “Column rank” y el “Row rank” son siempre iguales, por lo tanto es suficiente indicarlo como “rank”. Esto se puede obtener desde la descomposición de valores singulares (SVD), más detalles en el apéndice A.

1.3.5. Full Rank

Una matriz $A \in \mathbb{R}^{m \times n}$ es Full Rank si tiene el máximo rank posible, el cual corresponde al **mínimo** entre sus dimensiones m y n .

Por ejemplo, considere $A \in \mathbb{R}^{3 \times 2}$, entonces la matriz sería Full Rank si y solo si su rank es $\min(3, 2) = 2$.

1.3.6. Inversa

Una matriz tiene inversa si y solo si es cuadrada y es Full Rank. Esto significa que existe una matriz Z que satisface la siguiente ecuación:

$$AZ = ZA = \underbrace{I}_{\text{Matriz Identidad}} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Es decir, al multiplicar¹⁰ la matriz A por la matriz Z o al multiplicar la matriz Z por la matriz A se obtiene la matriz identidad I . En general uno denota la matriz inversa A como A^{-1} , es decir la inversa de A .

¹⁰Recuerde que la multiplicación de matrices no es conmutativa, por lo cual los casos que se indican no son equivalentes, pero sí se cumple la identidad!

Una propiedad importante de la matriz inversa es la siguiente: $(AB)^{-1} = B^{-1} A^{-1}$, siempre y cuando existan las inversas de la matrices A y B .

Comentario al margen

Una primera aplicación de la inversa de una matriz es la resolución de sistemas de ecuaciones lineales, $A\mathbf{x} = \mathbf{b}$, donde $\mathbf{x} = A^{-1}\mathbf{b}$. Sin embargo en el curso veremos diversas formas de obtener \mathbf{x} sin necesidad de obtener la matriz inversa de A , es decir A^{-1} !

En las siguientes equivalencias se presentan 4 formas de representar un sistema de ecuaciones lineales. Lo importante a destacar es que un sistema de ecuaciones lineales se puede interpretar como un cambio de base, es decir, uno al escribir $A\mathbf{x}$ está diciendo que, dada una matriz A fija, puedo elegir las coordenadas \mathbf{x} para construir otro vector con las columnas de la matriz A . En particular, uno busca las coordenadas exactas para poder escribir el vector \mathbf{b} . Desde el otro punto de vista, uno al multiplicar $A^{-1}\mathbf{b}$ está obteniendo el vector \mathbf{x} dada la combinación lineal de las columnas de la matriz A^{-1} multiplicada por las coordenadas definidas por \mathbf{b} . Entonces todo se resumen a un cambio de coordenadas!

$$A \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \Leftrightarrow A\mathbf{x} = \mathbf{b} \Leftrightarrow A \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \Leftrightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = A^{-1} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

1.3.7. Adjoint and Adjugate (or classical Adjunct matrix) of a Matrix

¡MUY IMPORTANTE!

El momento de traducir del inglés algunas definiciones, como lo es el caso de la matriz adjunta, se producen algunas colisiones de nombre. Por lo cual se recomienda en general verificar las definiciones asociadas a cada concepto. Acá se explicarán dos definiciones que se considerarán en el curso: Adjoint and Adjugate, particularmente la primera será bastante utilizada. Por lo tanto es muy importante entender la diferencia.

1.3.7.1. Adjoint of a Matrix

La matriz traspuesta conjugada, o adjoint de una matriz A , es la matriz A^* que cumple lo siguiente:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \Rightarrow \overline{A^T} = A^* = \begin{bmatrix} \overline{a_{11}} & \overline{a_{21}} & \overline{a_{31}} \\ \overline{a_{12}} & \overline{a_{22}} & \overline{a_{32}} \end{bmatrix},$$

donde el operador conjugado cambia el signo de la parte imaginaria de un número complejo, por ejemplo: $z = a + \mathbf{i}b \Rightarrow \bar{z} = a - \mathbf{i}b$. Notar que si en general trabajaremos con matrices reales, no hay diferencia entre la traspuesta de una matriz y su matriz adjoint, pero si la matriz tiene número complejos, sí habrá una diferencia.

Una propiedad importante de la Adjoint de una matriz es la siguiente: $(AB)^* = B^* A^*$. De similar forma a lo que ocurre con la traspuesta de una matriz.

1.3.7.2. Adjugate of a Matrix

Una muy buena explicación se encuentra en el siguiente [link¹¹](https://nhigham.com/2020/06/16/what-is-the-adjugate-of-a-matrix/), y es en el cual se basa el siguiente desarrollo. Considere la matriz $A \in \mathbb{R}^{n \times n}$ para la explicación. Sin embargo, antes de continuar, debemos recordar la definición

¹¹<https://nhigham.com/2020/06/16/what-is-the-adjugate-of-a-matrix/>

de la matrices de cofactores de A , la matriz de cofactores de A se define de la siguiente forma:

$$C_{i,j} = (-1)^{i+j} \det(A_{[i],[j]})$$

donde $C_{i,j}$ indica el valor de la entrada en la fila i y columna j de la matriz C , $A_{[i],[j]}$ corresponde a la matriz resultante de dimensión $(n-1) \times (n-1)$ luego de eliminar la fila i y columna j de la matriz A , y $\det(\cdot)$ corresponde al determinante. Por lo tanto la matriz Adjugate de la matriz A , denotada como $\text{adj}(A)$ corresponde a la matriz C transpuesta, es decir $\text{adj}(A) = C^T$. En particular $\text{adj}(A)$ satisface la siguiente identidad,

$$\text{adj}(A) = \det(A) A^{-1},$$

o también se puede escribir de la siguiente forma:

$$A \text{adj}(A) = \det(A) I,$$

donde I es la matriz identidad. Más detalles, ver el [link](#) antes mencionado.

1.3.8. Determinante

El determinante de una matriz se puede obtener recursivamente de la siguiente forma para una matriz $A \in \mathbb{R}^{n \times n}$:

$$\det(A) = \sum_{j=1}^n (-1)^{i+j} A_{i,j} \det(A_{[i],[j]}).$$

Algunas propiedades importantes del determinante son las siguiente:

1. $\det(A^{-1}) = \frac{1}{\det(A)}$.
2. $\det(AB) = \det(A) \det(B)$, donde $B \in \mathbb{R}^{n \times n}$.
3. $\det(\alpha A) = \alpha^n \det(A)$, donde α es un escalar.
4. $\det(A^T) = \det(A)$, donde T es el operador transpuesta.
5. $\det(A) = \prod_{i=1}^n \lambda_i$, donde λ_i para $i \in \{1, 2, \dots, n\}$ son los valores propios de A .

1.3.9. Traza

La traza de una matriz $A \in \mathbb{R}^{n \times n}$ se define de la siguiente forma:

$$\text{tr}(A) = \sum_{i=1}^n A_{i,i},$$

lo que significa que es básicamente la suma de los elementos de la diagonal. Algunas propiedades de la traza son las siguientes:

1. $\text{tr}(A+B) = \text{tr}(A) + \text{tr}(B)$.
2. $\text{tr}(\alpha A) = \alpha \text{tr}(A)$.
3. $\text{tr}(A^T) = \text{tr}(A)$.
4. $\text{tr}(A) = \sum_{i=1}^n \lambda_i$, donde λ_i para $i \in \{1, 2, \dots, n\}$ son los valores propios de A .

Respecto a esta última propiedad, notar que si una matriz tiene traza nula, i.e. $\text{tr}(A) = 0$, esto no significa que la matriz sea necesariamente singular ni tampoco que alguno de sus valores propios sea necesariamente 0. Para más detalles ver el siguiente [link](#).

1.3.10. Valores y vectores propios de una matriz

Sea $A \in \mathbb{R}^{n \times n}$ y $\mathbf{v} \in \mathbb{R}^n$, con $\mathbf{v} \neq \mathbf{0}$. Entonces si A, \mathbf{v} y λ , satisfacen la ecuación $A\mathbf{v} = \lambda\mathbf{v}$, entonces λ es denominado un valor propio de A y \mathbf{v} es denominado un vector propio de A .

Una característica importante de los valores propios de A es que los valores propios son raíces de la ecuación característica, es decir,

$$p(\lambda) = \det(A - \lambda I) = 0,$$

donde I es la matriz identidad. Algunas propiedades de los valores y vectores propios de A :

1. Si la matriz A es invertible y tiene valores propios λ_i y vectores propios \mathbf{v}_i , entonces los valores propios de A^{-1} son $\frac{1}{\lambda_i}$ y los vectores propios son los mismos.
2. Si \mathbf{v} es un vector propio de A , entonces $\alpha\mathbf{v}$ con $\alpha \neq 0$ también es un vector propio de A . Esto no ocurre con los valores propios, pero si podemos escalar un vector propios y sigue siendo un vector propio asociado al mismo valor propio. En general en Numpy o Scipy los vectores propios están normalizados, para estandarizar las salidas. Sin embargo, siempre es recomendable revisar la documentación en caso de que alguna librería en particular cambie su documentación¹².

1.3.11. Resumen de propiedades de una matriz invertible

Para $A \in \mathbb{R}^{n \times n}$, las siguientes condiciones son equivalentes:

1. A tiene inversa A^{-1}
2. $\text{Rank}(A)=n$
3. $\text{Range}(A)=\mathbb{R}^n$
4. $\text{Null}(A)=\emptyset$
5. 0 no es un valor propio de A
6. 0 no es un valor singular (singular value) de A
7. $\det(A) \neq 0$

Nota

Se sugiere al lector curioso^a que haga la conexión entre el resumen de propiedades de una matriz invertible y los contenidos presentados anteriormente.

^aY los no curiosos también!

¹²Se volverá a revisar este tema en la tarea de SVD!

1.4. Producto interno, producto externo y algunas matrices particulares

1.4.1. Producto Interno

El Producto Interno de dos vectores columnas, $\mathbf{x} \in \mathbb{C}^m$ y $\mathbf{y} \in \mathbb{C}^m$, se define de la siguiente forma:

$$\mathbf{x}^* \mathbf{y} = \sum_{i=1}^m \overline{x_i} y_i,$$

donde $\overline{x_i}$ corresponde al operador de conjugación del posible número complejo x_i . Otra forma de denotarlo es con paréntesis angulares¹³, por ejemplo $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^m \overline{x_i} y_i$.

Un caso del producto interno es cuando se aplica a un vector consigo mismo, es decir:

$$\mathbf{x}^* \mathbf{x} = \sum_{i=1}^m \overline{x_i} x_i = \sum_{i=1}^m |x_i|^2,$$

lo cual nos entrega la norma 2 al cuadrado del vector \mathbf{x} , también conocida como la norma Euclidiana, es decir,

$$\mathbf{x}^* \mathbf{x} = \langle \mathbf{x}, \mathbf{x} \rangle = \sum_{i=1}^m |x_i|^2 = \|\mathbf{x}\|_2^2.$$

En la sección 1.5 veremos en más detalle normas, tanto vectoriales como matriciales.

Otra interpretación importante del producto interno es la siguiente:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos(\alpha)$$

donde α es el ángulo entre los vectores \mathbf{x} y \mathbf{y} . En la siguiente Figura 1.5 se presenta un sketch. Respecto al ángulo, hay 2 interpretaciones importantes, considerando que $\|\mathbf{x}\| \neq 0$ y $\|\mathbf{y}\| \neq 0$:

- Si $\alpha = 0$, lo que implica que $\cos(\alpha) = 1$, entonces hay dependencia lineal entre los vectores.
- Si $\alpha = \pi/2$, lo que implica que $\cos(\alpha) = 0$, entonces los vectores son ortogonales¹⁴. Esto es válido tanto para \mathbb{R}^2 como para dimensiones superiores!

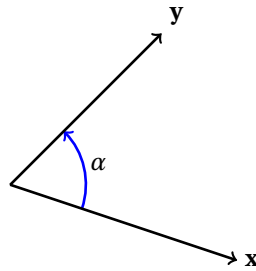


Figura 1.5: Sketch del ángulo entre 2 vectores.

¹³En algunas ocasiones también se usa esta misma notación para identificar explícitamente un vector. Pero en el caso que aparecen 2 vectores se debe entender como el producto interno a menos que se indique lo contrario.

¹⁴En la práctica los problemas numéricos surgen cuando 2 vectores que debieran ser ortogonales, no lo son debido a aproximaciones algorítmicas. Esto lo veremos más adelante!

1.4.2. Producto externo

El producto interno, anteriormente descrito, generaba un escalar a partir de 2 vectores. En este caso el producto externo genera una matriz a partir de 2 vectores columna. La definición es la siguiente, considere $\mathbf{x} \in \mathbb{R}^n$ y $\mathbf{y} \in \mathbb{R}^n$, entonces:

$$\mathbf{xy}^T = \begin{bmatrix} x_1 y_1 & x_1 y_2 & x_1 y_3 & \dots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & x_2 y_3 & \dots & x_2 y_n \\ x_3 y_1 & x_3 y_2 & x_3 y_3 & \dots & x_3 y_n \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_n y_1 & x_n y_2 & x_n y_3 & \dots & x_n y_n \end{bmatrix},$$

donde $\mathbf{x}^T = [x_1, x_2, x_3, \dots, x_n]$ y $\mathbf{y}^T = [y_1, y_2, y_3, \dots, y_n]$. Una propiedad importante del producto externo es que la matriz resultante tiene Rank igual a 1, por lo tanto no es invertible. Esto se puede observar debido a que todas las filas o columnas son linealmente dependientes entre si.

Comentario al margen

Una fórmula muy interesante relacionada con el producto externo es la Fórmula de Sherman-Morrison, la cual se define de la siguiente forma. Sea $A \in \mathbb{R}^{n \times n}$ e invertible, $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{v} \in \mathbb{R}^n$, y $\mathbf{v}^T A \mathbf{u} \neq -1$, entonces $A + \mathbf{uv}^T$ es invertible y se puede escribir de la siguiente forma:

$$(A + \mathbf{uv}^T)^{-1} = A^{-1} - \frac{A^{-1} \mathbf{uv}^T A^{-1}}{1 + \mathbf{v} A^{-1} \mathbf{u}}.$$

Lo interesante de esta fórmula es que si conociéramos la inversa de una matriz A , podemos fácilmente obtener la inversa de una perturbación de esta, es decir $(A + \mathbf{uv}^T)^{-1}$. Notar que esta perturbación se denomina perturbación de rank 1 dado que \mathbf{uv}^T tiene rank 1.

1.4.3. Algunas matrices particulares

En las siguientes secciones se presentará una breve lista de matrices con estructuras bien definidas. La razón principal de poder organizar las matrices en función de su estructura es que, desde el punto de vista computacional y matemático, uno puede construir algoritmos específicos para resolver problemas donde aparecen esas matrices. O, y no menos importante, poder transformar problemas generales en problemas donde aparecen las matrices en cuestión. Por ejemplo, en el curso veremos la descomposición LU y QR que transforman un problema general en una secuencia de problemas con matrices con propiedades particulares.

1.4.3.1. Matriz diagonal

Una matriz diagonal es una matriz que solo contiene elementos no nulos en su diagonal, en general se denotan por D , pero en realidad pueden tener cualquier nombre. La estructura de una matriz diagonal es la siguiente:

$$D = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_n \end{bmatrix} = \text{diag}(d_1, d_2, d_3, \dots, d_n).$$

Dado que tienen muchos elementos nulos, se prefiere en algunas ocasiones solo representar la matriz D de la forma compacta $\text{diag}(d_1, d_2, d_3, \dots, d_n)$. Un caso particular de una matriz diagonal es la matriz identidad I , que es la matriz diagonal donde los elementos de la diagonal son el número 1. Una propiedad importante de las matrices

diagonales es que es bastante simple poder resolver un sistema de ecuaciones lineales cuando la matriz asociada es una matriz diagonal, por ejemplo considere $D\mathbf{x} = \mathbf{b}$, de forma explícita obtenemos:

$$\begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

al realizar el producto matriz D con el vector \mathbf{x} obtenemos $d_i x_i = b_i$ para cada $i \in \{1, 2, \dots, n\}$. Lo que nos permite concluir que la solución es simplemente $x_i = \frac{b_i}{d_i}$. Matricialmente podríamos incluso obtener la inversa, que no es recomendada, pero en este caso es bastante simple:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \frac{1}{d_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{d_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \frac{1}{d_n} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} \frac{b_1}{d_1} \\ \frac{b_2}{d_2} \\ \vdots \\ \frac{b_n}{d_n} \end{bmatrix},$$

del cual observamos la misma conclusión de antes: $x_i = \frac{b_i}{d_i}$.

1.4.3.2. Matrices triangulares

Otra matriz particular e importante son las matrices triangulares, las cuales pueden ser triangular superior o inferior. Aquí presentaremos solamente la estructura de una matriz triangular superior:

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ 0 & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{n,n} \end{bmatrix}.$$

Es decir, una matriz triangular superior es una matriz con valores 0 en las componentes bajo la diagonal principal. Una propiedad importante de una matriz diagonal es que si no tienen ningún valor igual a 0 en su diagonal, sabemos que la matriz es invertible. Sobre la diagonal puede tener valores nulos, pero no en la diagonal para ser invertible.

Al igual que en el caso anterior, si quisiéramos resolver un sistema de ecuaciones lineales $U\mathbf{x} = \mathbf{b}$, obtenemos lo siguiente:

$$\begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & \cdots & u_{1,n} \\ 0 & u_{2,2} & \cdots & \cdots & u_{2,n} \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & \cdots & u_{n-1,n-1} & u_{n-1,n} \\ 0 & 0 & \cdots & \cdots & u_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix},$$

En este caso no obtenemos un despeje tan elegante como en el caso anterior, dado que no podemos escribir tan fácilmente U^{-1} . Sin embargo, no es un problema. En este caso la solución va en la dirección de construir un algoritmo para obtener la solución, sin recurrir a obtener la inversa! ¡Esto es el corazón de este curso! En este momento se explicará brevemente el algoritmo, y luego lo estudiaremos con más detalle. El algoritmo consiste en notar que en

la última fila de la matriz existe un solo término distinto de cero, lo que significa que cuando uno hace el producto matriz vector obtiene la ecuación:

$$u_{n,n} x_n = b_n,$$

lo que significa que podemos obtener $x_n = \frac{b_n}{u_{n,n}}$. Con este resultado, podemos analizar la penúltima ecuación ahora, la cual es la siguiente:

$$u_{n-1,n-1} x_{n-1} + u_{n-1,n} x_n = b_{n-1},$$

de la cual conocemos todos los valores, excepto x_{n-1} . El cual podemos despejar. Así hemos obtenido un segundo valor del vector de incógnitas, siguiendo el mismo patrón, podemos rápidamente construir un algoritmo para obtener \mathbf{x} . El nombre del algoritmo es Backward Substitution y lo estudiaremos en el capítulo 4.

1.4.3.3. Matrices unitarias

El último tipo de matriz que veremos en esta introducción son las matrices unitarias. Sea Q una matriz cuadrada $\in \mathbb{C}^{m \times m}$, entonces Q es una matriz unitaria si $Q^* = Q^{-1}$, tal que $Q^* Q = I$. Esto indica que si Q es unitaria, su inversa se obtiene obteniendo su matriz Adjoint, es decir, la transpuesta conjugada de Q . Esta es una propiedad muy particular, pero ¿Cómo es posible que una matriz logre eso? La forma de obtenerlo es la siguiente, considere la siguiente estructura de Q :

$$Q = [\mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \quad \mathbf{q}_n],$$

donde \mathbf{q}_i , para $i \in \{1, 2, 3, \dots, n\}$, son vectores unitarios, es decir $\|\mathbf{q}_i\| = \sqrt{\langle \mathbf{q}_i, \mathbf{q}_i \rangle} = 1$, y además son ortogonales¹⁵ entre sí, es decir su producto interno es nulo:

$$\langle \mathbf{q}_i, \mathbf{q}_j \rangle = \mathbf{q}_i^* \mathbf{q}_j = 0, \quad \forall i \neq j.$$

Entonces, con estas condiciones satisfechas, podemos evaluar $Q^* Q$,

$$Q^* Q = \underbrace{\begin{bmatrix} \mathbf{q}_1^* \\ \mathbf{q}_2^* \\ \vdots \\ \mathbf{q}_n^* \end{bmatrix}}_{Q^*} \underbrace{[\mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \quad \mathbf{q}_n]}_Q = \begin{bmatrix} \mathbf{q}_1^* \mathbf{q}_1 & \mathbf{q}_1^* \mathbf{q}_2 & \cdots & \mathbf{q}_1^* \mathbf{q}_n \\ \mathbf{q}_2^* \mathbf{q}_1 & \mathbf{q}_2^* \mathbf{q}_2 & \cdots & \mathbf{q}_2^* \mathbf{q}_n \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q}_n^* \mathbf{q}_1 & \mathbf{q}_n^* \mathbf{q}_2 & \cdots & \mathbf{q}_n^* \mathbf{q}_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix},$$

Claramente notamos que solo los términos de la diagonal serán 1, debido a que los vectores son unitarios, y los términos fuera de la diagonal son 0, debido a la ortogonalidad entre ellos.

Ahora, ¿qué ocurre si queremos resolver un sistema de ecuaciones lineales donde la matriz es una matriz unitaria? En ese caso obtendríamos el siguiente sistema de ecuaciones lineales:

$$\begin{aligned} Q\mathbf{x} &= \mathbf{b} \\ Q^{-1}Q\mathbf{x} &= Q^{-1}\mathbf{b} \\ I\mathbf{x} &= Q^{-1}\mathbf{b} \\ \mathbf{x} &= Q^{-1}\mathbf{b} \\ \mathbf{x} &= Q^*\mathbf{b} \end{aligned}$$

Lo cual es bastante más simple que calcular A^{-1} luego multiplicar por \mathbf{b} .

¹⁵Cuando un par de vectores son ortogonales y además tienen norma 1 se dice que son ortonormales.

Comentario al margen

Considerando el uso de vectores ortonormales: “El teorema de Pitágoras” asegura que para un conjunto de “ n ” vectores ortogonales \mathbf{x}_i se satisface la siguiente identidad:

$$\left\| \sum_{i=1}^n \mathbf{x}_i \right\|_2^2 = \sum_{i=1}^n \|\mathbf{x}_i\|_2^2$$

- ¿Podría demostrar la identidad con lo aprendido anteriormente?
- ¿Podría implementar en Numpy un código que verifique la identidad anterior?

¡MUY IMPORTANTE!

En general no es recomendado calcular A^{-1} explícitamente para obtener $A^{-1}\mathbf{b}$. Como se vio en las secciones anteriores, en algunos casos es más “fácil” y más “rápido” obtener el vector $A^{-1}\mathbf{b}$ directamente. Trabajaremos en esto en los siguientes capítulos del curso.

1.5. Normas

Esta sección se basa en el libro “Numerical linear Algebra” de Lloyd N. Trefethen and David Bau, III, SIAM.

1.5.1. Normas Vectoriales

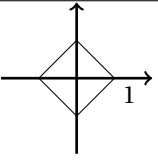
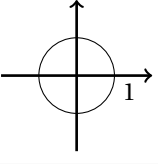
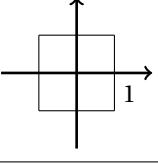
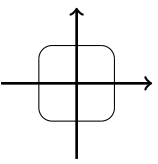
Las normas son la noción del tamaño o distancia en un espacio vectorial. Una norma es una función $\|\cdot\| : \mathbb{C}^m \rightarrow \mathbb{R}_0^+$ que asigna un valor real positivo o igual a cero a cada vector. Formalmente, una norma debe satisfacer las siguientes 3 condiciones, para todo vector \mathbf{x} y \mathbf{y} , y escalar $\alpha \in \mathbb{C}$,

- (1) $\|\mathbf{x}\| \geq 0$, y $\|\mathbf{x}\| = 0 \Rightarrow \mathbf{x} = \mathbf{0}$
- (2) $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$, Desigualdad Triangular
- (3) $\|\alpha\mathbf{x}\| = |\alpha| \cdot \|\mathbf{x}\|$

1.5.1.1. Las normas más utilizadas:

Tabla 1.2 presentan las definiciones de varias normas y se incluyen los isocontornos igual a 1 a modo referencial para el caso de $n = 2$.

Tabla 1.2: Normas vectoriales tradicionales y sus isocontornos unitarios

$l_1: \ \mathbf{x}\ _1 = \sum_{i=1}^n x_i $		$ x_1 + x_2 = 1$	Norma l_1 : Esta norma genera un isocontorno romboidal.
$l_2: \ \mathbf{x}\ _2 = \left(\sum_{i=1}^n x_i ^2 \right)^{\frac{1}{2}} = \sqrt{\mathbf{x}^* \mathbf{x}}$		$\sqrt{x_1^2 + x_2^2} = 1$	Norma l_2 : Esta norma genera un isocontorno circular.
$l_\infty: \ \mathbf{x}\ _\infty = \max_{i=1}^n x_i $		$\max\{ x_1 , x_2 \} = 1$	Norma l_∞ : Esta norma genera un isocontorno cuadrado.
$l_p: \ \mathbf{x}\ _p = \left(\sum_{i=1}^n x_i ^p \right)^{\frac{1}{p}}, (1 \leq p \leq \infty)$		$(x_1 ^p + x_2 ^p)^{\frac{1}{p}} = 1$	Norma l_p : Esta norma genera un isocontorno cuadrado con esquinas redondeadas.

1.5.1.2. Norma l_p con pesos w_i

Una posible extensión a las normas vectoriales ya discutidas es modificar el peso de cada una de sus componentes. Por ejemplo, si le asignamos un peso distinto w_i a cada componente x_i del vector \mathbf{x} , obtendríamos la siguiente definición para la norma l_p :

$$\|\mathbf{x}\|_{W,p} = \|W\mathbf{x}\|_p = \left(\sum_{i=1}^n |w_i \cdot x_i|^p \right)^{\frac{1}{p}},$$

donde $W = \text{diag}(w_1, w_2, w_3, \dots, w_n)$. La Figura 1.6 muestra el isocontorno en este caso, el cual corresponde a una forma elipsoidal.

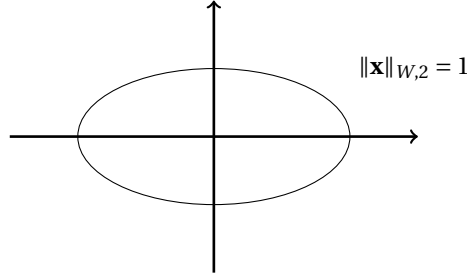


Figura 1.6: Sketch de isocontorno unitario de una norma l_p con pesos.

Comentario al margen

Desigualdad de Hölder y de Cauchy-Schwarz: Sea p y q dos números reales que se satisfacen la siguiente ecuación $\frac{1}{p} + \frac{1}{q} = 1$, donde $1 \leq p, q < \infty$, entonces la desigualdad de Hölder para cualquier par de vectores \mathbf{x} y \mathbf{y} es válida,

$$|\mathbf{x}^* \mathbf{y}| \leq \|\mathbf{x}\|_p \|\mathbf{y}\|_q.$$

Ahora, considerando el caso particular donde $p = q = 2$, obtenemos la desigualdad de Cauchy-Schwarz,

$$|\mathbf{x}^* \mathbf{y}| \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2.$$

¡MUY IMPORTANTE!

En la literatura es tradicional observar la definición de normas sin sub-índice, por ejemplo $\|\mathbf{x}\|$. Es este caso, para normas vectoriales, se considera que se refiere a la norma-2, a menos que sea indicado de otra forma.

1.5.2. Normas Matriciales

Similar al caso de las normas vectoriales, existen las normas matriciales. En este caso, las normas matriciales deben cumplir con las siguientes 3 condiciones:

- (1) $\|A\| \geq 0$, y $\|A\| = 0 \Rightarrow A = 0$,
- (2) $\|A + B\| \leq \|A\| + \|B\|$,
- (3) $\|\alpha A\| = |\alpha| \|A\|$.

Estas condiciones son análogas a las requeridas para las normas vectoriales.

1.5.2.1. Normas Matriciales inducidas por normas vectoriales

Una alternativa para obtener normas matriciales, es utilizar las normas vectoriales ya presentadas. Por ejemplo, considere las siguientes normas vectoriales $\|\cdot\|_{(n)}$ y $\|\cdot\|_{(m)}$ en el dominio y rango de $A \in \mathbb{R}^{m \times n}$, respectivamente, ver Sección 1.5.1.1. Entonces la norma matricial inducida $\|A\|_{(m,n)}$ es el número C más pequeño posible que hace válida la siguiente desigualdad para todo $\mathbf{x} \in \mathbb{R}^n$:

$$\|A\mathbf{x}\|_{(m)} \leq C \|\mathbf{x}\|_{(n)}.$$

Considerando que $\|\mathbf{x}\|_{(n)} \neq 0$ podemos re-escribir la ecuación anterior de la siguiente forma,

$$\frac{\|A\mathbf{x}\|_{(m)}}{\|\mathbf{x}\|_{(n)}} \leq C = \|A\|_{(m,n)}$$

Ahora, obteniendo el máximo, obtenemos formalmente la definición de una norma matricial inducida por las normas vectoriales $\|\cdot\|_{(n)}$ y $\|\cdot\|_{(m)}$ de la siguiente forma:

$$\max_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \|\mathbf{x}\|_{(n)} \neq 0}} \frac{\|A\mathbf{x}\|_{(m)}}{\|\mathbf{x}\|_{(n)}} = C = \|A\|_{(m,n)}$$

O incluso se podría simplificar de la siguiente forma al considerar que $\|\mathbf{x}\|_{(n)} = 1$,

$$\max_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \|\mathbf{x}\|_{(n)} = 1}} \|A\mathbf{x}\|_{(m)} = \|A\|_{(m,n)}.$$

De la definición anterior, se obtiene la siguiente desigualdad:

$$\|A\mathbf{x}\|_{(m)} \leq \|A\|_{(m,n)} \|\mathbf{x}\|_{(n)}.$$

Para los casos donde la matriz es cuadrada, es decir $n = m$, uno solo indica el tipo de norma vectorial utilizada. Por ejemplo,

$$\|A\mathbf{x}\|_2 \leq \|A\|_2 \|\mathbf{x}\|_2.$$

En este caso corresponde a la norma vectorial l_2 para obtener la norma matricial $\|A\|_2$.

La obtención de las normas matriciales inducidas por normas vectoriales en general no es tan directo comparado en como se obtienen las normas vectoriales. A continuación, revisaremos 3 casos particulares pero importantes: norma-1 matricial, norma- ∞ matricial y norma-2 matricial¹⁶. Para hacer la explicación más concreta, utilizaremos una matriz particular, es decir, la siguiente matriz A ,

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

En el primer caso, i.e. norma-1, por definición, obtenemos la siguiente desigualdad,

$$\left\| \begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\|_1 \leq C \left\| \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\|_1.$$

Ahora, considerando la definición de la norma donde $\|\mathbf{x}\|_1 = 1$, obtenemos, la siguiente desigualdad simplificada pero incluyendo el operador máx, antes omitido,

$$\max_{[x_1, x_2] \in \mathbb{R}^2} \left\| \begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\|_1 \leq C.$$

Esto nos da ahora la oportunidad de poder interpretar gráficamente. La interpretación contiene 2 componentes, (i) graficar la el “círculo”¹⁷ unitario en \mathbb{R}^2 considerando la norma-1 vectorial, ver Figura 1.7a, y (ii) graficar la Transformación Lineal realizada por la matriz A al conjunto de vectores (o puntos) en el “círculo” unitario definido por la norma-1 vectorial, ver Figura 1.7a. En resumen, la norma-1 de una matriz de dimensión se obtienen al calcular la Máxima suma absoluta de columnas, es decir $\|A\|_1 = 4$ para este ejemplo. La definición formal es la siguiente,

$$\|A\|_1 = \max_j \|\mathbf{a}_j\|_1,$$

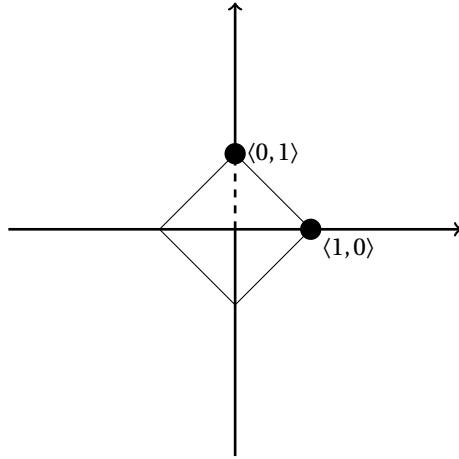
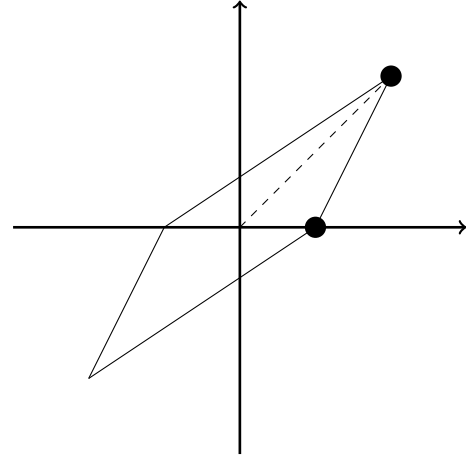
(a) Isocontorno $|x_1| + |x_2| = 1$.(b) Elementos del isocontorno transformados. Se aprecia que el punto de referencia $\langle 1, 0 \rangle$ queda igual al ser multiplicado por A , en cambio el punto $\langle 0, 1 \rangle$ se transforma en $\langle 2, 2 \rangle$ al ser multiplicado por A . En este caso se obtiene $\|A\|_1 = 4$.

Figura 1.7: Ejemplo de norma-1 matricial. La norma-1 de una matriz se obtienen al calcular la Máxima suma absoluta de columnas. La línea punteada indica el vector en el espacio de partida que al ser transformado obtiene máximo valor de la norma-1, el cual corresponde al vector $[0, 1]$.

lo cual también se puede interpretar como la máxima norma-1 vectorial entre todos sus vectores columnas \mathbf{a}_j .

En el caso de la norma- ∞ matricial se puede hacer el mismo análisis al realizado con la norma-1 matricial. En la Figura 1.8a se observa el isocontorno unitario y en la Figura 1.8b se observa la transformación del isocontorno unitario. En esta caso la norma matricial es $\|A\|_\infty = 3$. La definición de esta norma es la siguiente,

$$\|A\|_\infty = \max_i \|\mathbf{r}_i\|_1,$$

es decir, es la máxima norma-1 vectorial entre todos sus vectores filas \mathbf{r}_i . Notar que no es un error que se indique que se utiliza la norma-1, la diferencia importante es que se aplica a las filas \mathbf{r}_i de la matriz A .

Finalmente llegamos a la norma-2 matricial. Al igual que en los casos anteriores, se muestra en la Figura 1.9a el isocontorno unitario asociado, y en la Figura 1.9b se muestra la transformación del isocontorno unitario. En esta caso la norma matricial es $\|A\|_2 \approx 2,9208$. La definición de esta norma es la siguiente,

$$\|A\|_2 = \sigma_1,$$

donde σ_1 es el primer y mayor valor singular de A . Los valores singulares se mencionaron anteriormente y se estudiarán durante el curso. Ver el comentario al margen asociado y el apéndice A para aprender más de los valores singulares.

¹⁶Se deja la norma-2 matricial para el final debido a que requiere un análisis distinto.

¹⁷Se denomina círculo unitario en la norma-1 aunque en realidad se forme un rombo!

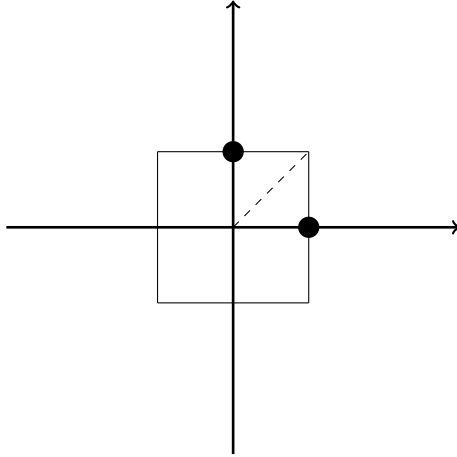
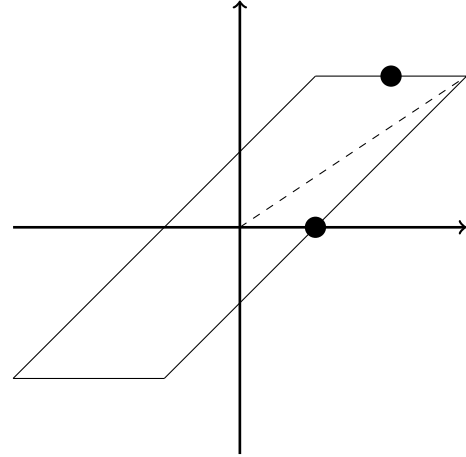
(a) Isocontorno máx $\{|x_1|, |x_2|\} = 1$.(b) Elementos del isocontorno transformados. Se aprecia que los puntos de referencia $\langle 1, 0 \rangle$ y $\langle 0, 1 \rangle$ no obtienen el máximo en este caso, sino el punto $\langle 1, 1 \rangle$. En este caso se obtiene $\|A\|_\infty = 3$.

Figura 1.8: Ejemplo de norma- ∞ matricial. La norma- ∞ de una matriz se obtienen al calcular la Máxima suma absoluta de las filas. La línea punteada indica el vector en el espacio de partida que al ser transformado obtiene máximo valor de la norma- ∞ , el cual corresponde al vector $\langle 1, 1 \rangle$.

Comentario al margen

Un comentarios breve sobre los valores singulares de una matriz: Los valores singulares de una matriz vienen de la descomposición de valores singulares, la cual indica que cualquier matriz A puede ser escrita como el producto de 3 matrices, es decir $A = U \Sigma V^*$, donde U y V son matrices unitarias, $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots)$ es una matriz diagonal con valores reales y mayores o igual a 0 y * es el operador adjoint. Lo interesante de esta descomposición es que uno puede obtener el producto $A^* A$, o simplemente $A^T A$ en caso de que la matriz A sea real y uno obtiene,

$$\begin{aligned} A^T A &= (U \Sigma V^T)^T U \Sigma V^T \\ &= V \Sigma^T U^T U \Sigma V^T \\ &= V \Sigma^T I \Sigma V^T \\ &= V \Sigma^2 V^T, \end{aligned}$$

donde $\Sigma^2 = \text{diag}(\sigma_1^2, \sigma_2^2, \dots)$. Esto nos indica que los valores singulares de A son la raíz cuadrada de los valores propios de la matriz $A^T A$! Esto es debido a que $V \Sigma^2 V^T$ es la descomposición de valores propios de $A^T A$. Más detalles en apéndice A y se sugiere también ver apéndice B.

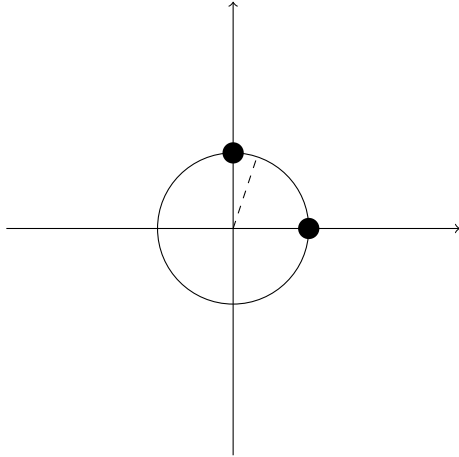
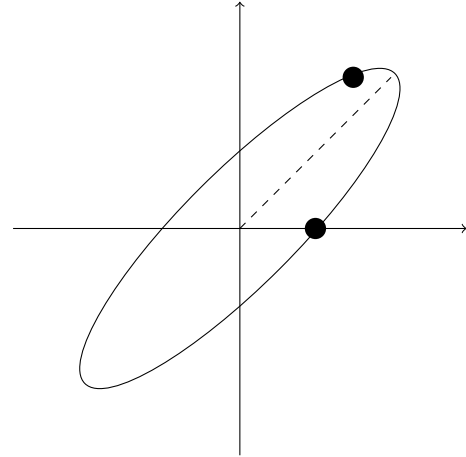
(a) Isocontorno $x_1^2 + x_2^2 = 1$.(b) Elementos del isocontorno transformados. Se aprecia que los puntos de referencia $[1, 0]$ y $[0, 1]$ quedan a lo largo de la elipse obtenida. En este caso se obtiene $\|A\|_2 \approx 2,9208$.

Figura 1.9: Ejemplo de norma-2 matricial. La norma-2 de una matriz se obtienen al obtener el mayor valor singular, σ_1 , de la matriz A . La línea punteada indica el vector en el espacio de partida que al ser transformado obtiene máximo valor de la norma-2.

1.5.2.2. Norma de Frobenius

Finalmente llegamos a una norma matricial que no es inducida por una norma vectorial. Esta corresponde a la norma de Frobenius, la cual se puede obtener de las siguientes formas,

$$\begin{aligned} \|A\|_F &= \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right)^{\frac{1}{2}} \\ &= \left(\sum_{j=1}^n \|a_j\|_2^2 \right)^{\frac{1}{2}} \\ &= \sqrt{\text{tr}(A^* A)} = \sqrt{\text{tr}(A A^*)} \end{aligned}$$

donde $\text{tr}(B)$ denota la traza definida en la sección 1.3.9, y corresponde a la suma de los elementos de la diagonal de la matriz B .

¡MUY IMPORTANTE!

Thm 1. Para cualquier matriz $A \in \mathbb{C}^{m \times n}$ y matriz $Q \in \mathbb{C}^{m \times m}$ unitaria, tenemos:

$$\|QA\|_2 = \|A\|_2, \quad \|QA\|_F = \|A\|_F$$

Demostración. Primero demostraremos que $\|QA\|_2 = \|A\|_2$. Por la definición de una norma matricial inducida por una norma vectorial tenemos para la norma-2 matricial lo siguiente:

$$\max_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \|\mathbf{x}\|_2=1}} \|QAx\|_2 = \|QA\|_2.$$

Ahora, recordando que la norma vectorial $\|QAx\|_2$ puede obtenerse como la raíz cuadrada del producto interno entre el vector QAx y si mismo, es decir,

$$\begin{aligned} \|QAx\|_2 &= \sqrt{(QAx)^* (QAx)} \\ &= \sqrt{\mathbf{x}^* A^* Q^* Q A \mathbf{x}} \end{aligned}$$

Considerando que Q es unitaria, sabemos que $Q^* = Q^{-1}$ por lo estudiado en la Sección 1.4.3.3, entonces,

$$\begin{aligned} \|QAx\|_2 &= \sqrt{\mathbf{x}^* A^* Q^* Q A \mathbf{x}} \\ &= \sqrt{\mathbf{x}^* A^* A \mathbf{x}} \\ &= \sqrt{(A\mathbf{x})^* (A\mathbf{x})} \\ &= \|A\mathbf{x}\|_2. \end{aligned}$$

Por lo tanto,

$$\|QA\|_2 = \max_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \|\mathbf{x}\|_2=1}} \|QAx\|_2 = \max_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \|\mathbf{x}\|_2=1}} \|A\mathbf{x}\|_2 = \|A\|_2.$$

De la misma forma para $\|QA\|_F$, y considerando que,

$$\begin{aligned} \|QA\|_F &= \text{tr}((QA)^* (QA)) \\ &= \text{tr}(A^* Q^* Q A) \\ &= \text{tr}(A^* A) \\ &= \|A\|_F. \end{aligned}$$

□

En resumen, las matrices unitarias no modifican la norma-2 vectorial, ni las norma-2 matricial, ni la norma de Frobenius matricial!

1.6. Ejercicios Propuestos

“El teorema de Pitágoras” asegura que para un conjunto de n vectores ortogonales \mathbf{x}_i se obtiene lo siguiente,

$$\left\| \sum_{i=1}^n \mathbf{x}_i \right\|_2^2 = \sum_{i=1}^n \|\mathbf{x}_i\|_2^2$$

1. Demuestre esto para el caso $n = 2$ por computación explícita de $\|\mathbf{x}_1 + \mathbf{x}_2\|$.
2. Demuestre lo anterior para el caso general
3. Sea $A \in \mathbb{C}^{n \times n}$ una matriz Hermitiana, es decir $A = A^*$.
 - Demuestre que todos los valores propios de A son reales.
 - Demuestre que si \mathbf{v} y \mathbf{w} son vectores propios asociados a valores propios diferentes, entonces \mathbf{v} y \mathbf{w} son ortogonales.

Recuerde que un vector propio A es un vector no-nulo $\mathbf{v} \in \mathbb{C}$ de tal forma que $A\mathbf{v} = \lambda\mathbf{v}$ para algún $\lambda \in \mathbb{C}$, donde λ se denomina el valor propio asociado a vector propio \mathbf{v} .

Capítulo 2

Estándar de punto flotante y pérdida de importancia

En este capítulo empezaremos a conectar la computación con la teoría más profundamente. Aquí es, lo que podríamos denominar, el origen de la Computación Científica. Se revisará en detalle el estándar de punto flotante IEEE 754, aritmética de punto flotante y pérdida de importancia.

Comentario al margen

Ideas Útiles:

¿Cuál es la mejor forma de evaluar el siguiente polinomio $P(x)$?

$$P(x) = c_5 x^4 + c_4 x^3 + c_3 x^2 + c_2 x + c_1$$

a) **Forma directa:** Simplemente multiplicamos y sumamos:

$$\begin{aligned} c_5 x^4 &\rightarrow \text{requiere 4 multiplicaciones,} \\ + c_4 x^3 &\rightarrow \text{requiere 3 multiplicaciones y 1 suma,} \\ + c_3 x^2 &\rightarrow \text{requiere 2 multiplicaciones y 1 suma,} \\ + c_2 x &\rightarrow \text{requiere 1 multiplicación y 1 suma,} \\ + c_1 &\rightarrow \text{requiere 1 suma.} \end{aligned}$$

Se necesitan 10 multiplicaciones y 4 sumas, un total de 14 operaciones elementales.

b) **Forma directa pre-calculada:** Precalculamos y guardamos:

$$\begin{aligned} a &= x^2 \rightarrow \text{requiere 1 multiplicación,} \\ b &= x a \rightarrow \text{requiere 1 multiplicación y representa } b = x^3, \\ d &= x b \rightarrow \text{requiere 1 multiplicación y representa } d = x^4. \end{aligned}$$

Al momento de evaluar el polinomio reemplazamos los términos precalculados:

$$\begin{aligned} c_5 x^4 &= c_5 d \rightarrow \text{requiere 1 multiplicación,} \\ + c_4 x^3 &= c_4 b \rightarrow \text{requiere 1 multiplicación y 1 suma,} \\ + c_3 x^2 &= c_3 a \rightarrow \text{requiere 1 multiplicación y 1 suma,} \\ + c_2 x &\rightarrow \text{requiere 1 multiplicación y 1 suma,} \\ + c_1 &\rightarrow \text{requiere 1 suma,} \end{aligned}$$

Se necesitan 7 multiplicaciones y 4 sumas, un total de 11 operaciones elementales. Esto ya es una mejora, se pasó de un algoritmo que requería 14 operaciones elementales a uno que requería 11 pero almacenando 3 variables temporales.

c) **Multiplicación anidada o Método de Horner:**

El método de Horner considera la siguiente re-escritura del polinomio original,

$$P(x) = c_1 + x(c_2 + x(c_3 + x(c_4 + c_5 x)))$$

La evaluación requiere ir evaluando términos de forma anidada,

$$\begin{aligned} f_1 &= (c_4 + c_5 x) \rightarrow \text{requiere 1 multiplicación y 1 suma,} \\ f_2 &= (c_3 + x f_1) \rightarrow \text{requiere 1 multiplicación y 1 suma,} \\ f_3 &= (c_2 + x f_2) \rightarrow \text{requiere 1 multiplicación y 1 suma,} \\ P(x) &= (c_1 + x f_3) \rightarrow \text{requiere 1 multiplicación y 1 suma.} \end{aligned}$$

En este caso, se necesitan 4 multiplicaciones y 4 sumas, un total de 8 operaciones elementales! Lo cual hace al método de Horner el método que reduce la cantidad de operaciones elementales significativamente!

2.1. Números Binarios

Un número binario con decimales se puede representar de la siguiente forma,

$$(B)_2 = \dots b_2 b_1 b_0 . b_{-1} b_{-2} b_{-3} \dots$$

donde cada dígito binario, o bit, es 0 o 1. Para denotar un número binario se utilizará el paréntesis en conjunto con el sub-índice 2, de forma similar se considerará si el sub-índice es 10, entonces se interpreta como un número decimal, i.e. base 10. Notar que la separación entre la parte entera y fraccionaria sigue siendo el punto (o coma, según corresponda!).

Ejemplo 1 Convierta el siguiente número binario en su forma decimal,

$$(1001.01)_2$$

Solución:

$$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 8 + 0 + 0 + 1 + 0 + \frac{1}{4} = (9.25)_{10}$$

Ejemplo 2 Convierta el siguiente número decimal a binario,

$$(28.7)_{10} = (28)_{10} + (0.7)_{10}.$$

Comenzaremos con la parte entera:

$$28 \div 2 = 14 \text{ con resto } 0$$

$$14 \div 2 = 7 \text{ con resto } 0$$

$$7 \div 2 = 3 \text{ con resto } 1$$

$$3 \div 2 = 1 \text{ con resto } 1$$

$$1 \div 2 = 0 \text{ con resto } 1$$

Por lo tanto la parte entera se expresa como $(11100)_2$, esto se logra al recuperar los restos de forma inversa como aparecieron. Esto se obtiene así debido a que al dividir por 2, se van obteniendo efectivamente en el resto el dígito binario correspondiente. Ahora su parte decimal, en este caso el procedimiento es inverso, es decir, se debe ir multiplicando por 2, y la parte entera que se obtiene corresponde al dígito binario requerido:

$$,7 \times 2 = ,4 + 1$$

$$,4 \times 2 = ,8 + 0$$

$$,8 \times 2 = ,6 + 1$$

$$,6 \times 2 = ,2 + 1$$

$$,2 \times 2 = ,4 + 0$$

$$,4 \times 2 = ,8 + 0$$

$$\vdots$$

Por lo tanto la parte decimal se expresa como $(.1011001100110\dots)_2 = (.1\overline{01110})_2$.

$$\therefore (28,7)_{10} = (11100.1\overline{01110})_2.$$

Notar que para determinar que un número es periódico debemos obtener varios términos más, acá se muestran unos pocos solamente, sin embargo vemos que el patrón se obtiene al haber obtenido un término antes ya obtenido, es decir $,4 \times 2 = ,8 + 0$.

Ejemplo 3

¿Qué número representa $(0.\overline{10})_2$ en base 10?

Lo primero que podemos hacer es escribir algunos dígitos en su forma expandida,

$$(0.101010\dots)_2,$$

del cual podemos notar el patrón y escribirlo como una serie de la siguiente forma,

$$(0.\underbrace{101010\dots}_{2^{-1}+2^{-3}+2^{-5}\dots})_2 = \sum_{i=0}^{\infty} 2^{-(2i+1)} = \frac{1}{2} \cdot \sum_{i=0}^{\infty} (2^{-2})^i = \frac{1}{2} \cdot \frac{1}{1-\frac{1}{4}} = \frac{1}{2} \cdot \frac{4}{3} = \frac{2}{3}$$

Comentario al margen

Para la computación del Ejemplo 3 utilizamos la famosa, útil e inolvidable serie geométrica, la cual la recordaremos a continuación solo por la felicidad de verla nuevamente:

$$1 + r + r^2 + r^3 + \dots = \sum_{i=0}^{\infty} r^i = \frac{1}{1-r},$$

para $|r| < 1$. También es muy interesante recordar la forma cerrada de la sumatoria de los primeros $n + 1$ términos de la serie geométrica,

$$1 + r + r^2 + r^3 + \dots + r^n = \sum_{i=0}^n r^i = \begin{cases} \frac{1-r^{n+1}}{1-r}, & \text{si } r \neq 1 \\ n+1, & \text{si } r = 1 \end{cases}.$$

2.2. Estándar de “Punto flotante” en base 2 de Números Reales (R):

En esta sección profundizaremos en la estructura de los números que utilizaremos en la implementación computacional de los algoritmos que discutiremos. Este es un tema muy importante en el sentido de que la mayoría de la computación que realizamos en computadores modernos, en donde se requiere operar con números reales, utilizará el **estándar de punto flotante IEEE 754**¹. La forma coloquial de entenderlo corresponde a la analogía de utilizar la notación científica pero en base 2. A continuación presentamos una *interpretación conveniente* de las componentes de la “palabra”² que utilizamos para almacenar un número en el estándar de punto flotante.

$$\underbrace{\pm}_{\text{signo siempre 1}} \underbrace{1}_{\text{signo siempre 1}} \cdot \underbrace{bbbbbb\dots}_{\text{mantisa}} \cdot 2^p \rightarrow \text{exponente}$$

En rigor, uno puede escribir cualquier número real en el estándar de punto flotante, pero dependiendo del número en cuestión se podría requerir un mantisa finita o infinita!

Ejemplo 1 : A continuación se presentan varios números en el estándar de punto flotante:

$$\begin{aligned} 9 &= (1001)_2 = +1.001 \cdot 2^3 \Rightarrow 1 \cdot 2^3 + 1 \cdot 2^0 = (1 + 1 \cdot 2^{-3}) \cdot 2^3 = +1.001 \cdot 2^3, \\ 2 &= (10)_2 = +1.00 \cdot 2^1, \\ 0.5 &= (0.1)_2 = +1.00 \cdot 2^{-1}, \\ 0.75 &= \frac{1}{2} + \frac{1}{4} = (0.11)_2 = +1.10 \cdot 2^{-1}. \end{aligned}$$

¹IEEE 754 Floating point standard, google it!

²En la sección 2.6 se presenta la forma efectiva de cómo se almacena computacionalmente.

Acá se observa que al ser el número $(9,4)_{10}$ periódico en base 2 es necesario decidir si almacenarlo como el número que se obtiene al truncar la parte de la mantisa que no se puede almacenar o aproximar al siguiente número representable en este formato. La respuesta corresponde a aproximar al más cercano de los 2 números anteriormente mencionados, sin embargo hay que considerar el caso de empate, es decir, si está exactamente al medio. Este lo discutiremos en la siguiente sección.

2.4. Regla IEEE de redondeo al más cercano

A continuación se presenta la regla de redondeo para **double precision**³, la cual se puede extender fácilmente a los otros formatos.

Def 2. *Regla de redondeo IEEE: Para double precision, si el 53vo bit a la derecha del punto binario es 0, redondear hacia abajo (truncar después del bit 52). Si el 53vo bit es 1, entonces redondear hacia arriba (agregar 1 al bit 52), a menos que todos los bits “conocidos” a la derecha de 1 son 0, en este caso 1 es agregado al bit 52 si y solo si el bit 52 es 1.*

Para un mejor entendimiento de la definición anterior se presenta el siguiente esquema, el cual muestra el límite entre lo que sí se puede almacenar y lo que no, es decir, el umbral existente entre el bit 52 y 53 de la mantisa.

$$+1. \text{-----} \boxed{} \boxed{} \boxed{} \boxed{} \\ \text{52 53}$$

La línea vertical incluida separa el bit 52 del 53 de la mantisa. La anterior definición nos da paso a la siguiente definición:

Def 3. *Se denota $fl(x)$ al número realmente almacenado utilizando el formato de punto flotante elegida y la “regla IEEE de redondeo al más cercano”, cuando se quiere almacenar “x”.*

Esto significa que, desde el punto de vista de notación, indicaremos que cuando queremos almacenar el número x , lo que realmente quedará almacenado es $fl(x)$. Por ejemplo, supongamos que queremos almacenar el siguiente número:

$$+1,00101100110\dots011001100|110\dots \cdot 2^3,$$

donde la barra vertical denota la separación entre el bit 52 y 53 de la mantisa, ya mencionado anteriormente. Entonces el número que realmente se almacena es el siguiente:

$$\underbrace{fl(+1,00101100110\dots011001100|110\dots \cdot 2^3)}_{\text{Número original}} = \underbrace{+1,00101100110\dots011001101}_{\text{Número almacenado}} \cdot 2^3.$$

Notar que acá se aplicó la regla de redondeo, en este caso se sumó 1 al bit 52 dado los bits conocidos a la derecha del bit 53 eran distintos de 0.

Aplique la regla a los siguiente casos, siguiendo el mismo formato ya presentado:

$$\begin{aligned} fl(+1.0001\dots0001|1001\dots \cdot 2^8) &= \boxed{+1.0001\dots \text{-----}} \cdot 2^8, \\ fl(+1.0001\dots0001|1000\dots \cdot 2^5) &= \boxed{+1.0001\dots \text{-----}} \cdot 2^5, \\ fl(+1.0001\dots0100|1000\dots \cdot 2^3) &= \boxed{+1.0001\dots \text{-----}} \cdot 2^3, \end{aligned}$$

³Se súper destaca que esta instancia de la regla de redondeo es para *double precision*, para los otros formatos se debe adaptar.

2.5. Importancia de “Machine Epsilon”

Antes de discutir la importancia de ϵ_{mach} , necesitamos presentar las siguientes definiciones de error.

Def 4. Se define como error absoluto entre una cantidad exacta x y su aproximación o cantidad computada x_c de la siguiente forma:

$$\text{Error absoluto} = |x_c - x|.$$

Def 5. Se define como error relativo entre una cantidad exacta x y su aproximación o cantidad computada x_c de la siguiente forma:

$$\text{Error relativo} = \frac{\text{Error absoluto}}{|x|} = \frac{|x_c - x|}{|x|}, \quad |x| > 0.$$

En el caso de utilizar cantidades vectoriales o matriciales se reemplaza el valor absoluto $|\cdot|$ por alguna norma vectorial o matricial $\|\cdot\|$, respectivamente.

Ahora, se presenta un resultado muy importante.

¡MUY IMPORTANTE!

En el modelo IEEE de aritmética de punto flotante, el error relativo de redondeo de $\text{fl}(x)$ no es más que la mitad de ϵ_{mach} :

$$\frac{|\text{fl}(x) - x|}{|x|} \leq \frac{1}{2} \cdot \epsilon_{mach}.$$

O mejor,

$$|\text{fl}(x) - x| \leq \frac{1}{2} \cdot \epsilon_{mach} |x|.$$

Es decir, el error de representar un número en double precision (y otros formatos) es proporcional al tamaño del número original.

Ahora considere la siguiente muestra de números del estándar de punto flotante donde se utiliza 2 bits para la mantisa, no se indica la cantidad de bits para signo (que debería ser 1!) ni para el exponente para focalizar el análisis en la mantisa. En la Figura 2.2 se presenta un esquema de los números de punto flotante que pueden representarse con 2 bits en la mantisa.

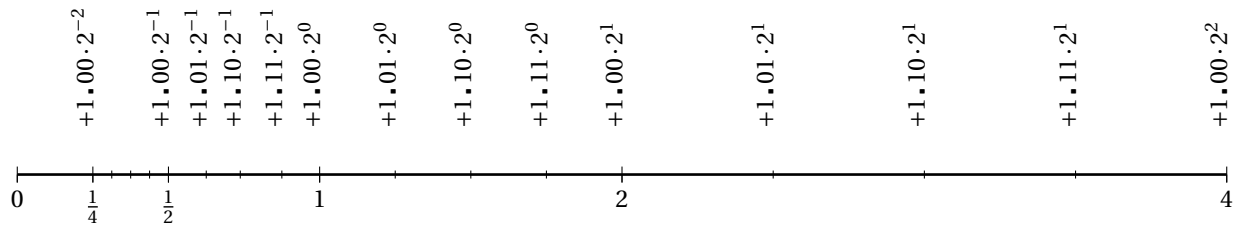


Figura 2.2: Recta numérica representando números del estándar de punto flotante con 2 bits disponibles para la mantisa.

De la Figura 2.2 podemos enfatizar los siguientes puntos,

1. La cantidad de números representables entre potencias de 2 es constante, es decir, en los intervalos incluidos: $[2^{-2}, 2^{-1}]$, $[2^{-1}, 2^0]$, $[2^0, 2^1]$ y $[2^1, 2^2]$, en este caso solo se pueden representar 3 números en cada intervalo.
2. El gap o espaciado entre números representables aumenta al aumentar la magnitud del número. Esto está precisamente relacionado con la importancia de ϵ_{mach} antes mencionada, es decir, al almacenar x , el error relativo es proporcional al tamaño de x .

Continuando con el ejemplo anterior, donde tiene disponible solo 2 bits en la mantisa ¿cuál es el primero entero no representable?⁴

2.6. Representación de Máquina

Hasta el momento hemos estudiado principalmente el comportamiento asociado a la mantisa, ahora nos queda revisar la forma de manejo del signo y el exponente para entender completamente que es lo que ocurre al almacenar un “número real” x en el computador. Como ya hemos discutido en secciones anteriores, lo que se almacena es efectivamente la representación de punto flotante $\text{fl}(x)$. En esta sección continuamos con el estudio de double precision, es decir, debemos entender como se utilizan los 64 bits disponibles. Tabla 2.1 muestra la distribución de bits y como se almacenan en una palabra de 64 bits, esto claramente dependerá de cada arquitectura utilizada pero debe tener esas 3 componentes.

Tabla 2.1: Almacenamiento **computacional** de un número en double precision.

s	$e_1 e_2 \dots e_{10} e_{11}$	$b_1 b_2 \dots b_{51} b_{52}$
1- bit	11 bits	52 bits

¡MUY IMPORTANTE!

Es importante re-destacar que de los 64 bits de la representación de punto flotante el primer bit almacenado corresponde al signo, luego los 11 siguientes corresponden al exponente y los últimos 52 a las mantisa. En las explicaciones anteriores se presentaba en un orden distinto para enfatizar la interpretación de estos, pero el almacenamiento computacional sigue el orden antes descrito.

El signo s se representa de la siguiente forma,

$$s: \begin{cases} 0, \text{ si el número es positivo} \\ 1, \text{ si es negativo} \end{cases}$$

El cual no requiere mucho más detalle. Sin embargo el exponente requiere una explicación detallada que se presenta a continuación:

- Al tener a nuestra disposición 11 bits en $e_1 e_2 \dots e_{10} e_{11}$ podemos almacenar $2^{11} = 2048$ números.
- Estos 2048 números se interpretan como si almacenáramos los números enteros $\{0, 1, 2, 3, \dots, 2046, 2047\}$.
- Dado que necesitamos exponentes positivos y negativos se introduce un exponent bias o simplemente shift, que corresponde en este caso a $2^{11-1} - 1 = 1023$.

⁴Hay que pensar en que el gap en este caso sería por lo menos 2 para no poder representar el primer entero

- El exponent bias simplemente traslada⁵ los números enteros representables por los 11 bits del exponente de la siguiente forma: $\{1 - 1023, 2 - 1023, 3 - 1023, \dots, 2046 - 1023\} = \{-1022, -1021, -1020, \dots, 1023\}$. Figura 2.3 presenta de forma gráfica la relación y a continuación se presenta un diagrama que resume conceptualmente lo explicado:

$$e_1 \dots e_{11} : \quad 0 - 2047 \Leftrightarrow \begin{array}{ccc} -1022 & a & 1023 \\ \downarrow & & \downarrow \\ 1 & - & 2046 \end{array}$$

- En el punto anterior se omitieron 2 casos: (i) cuando el exponente $e_1 e_2 \dots e_{10} e_{11} = 0000000000$, y (ii) $e_1 e_2 \dots e_{10} e_{11} = 1111111111$. Estos casos se explicarán en la siguientes secciones.

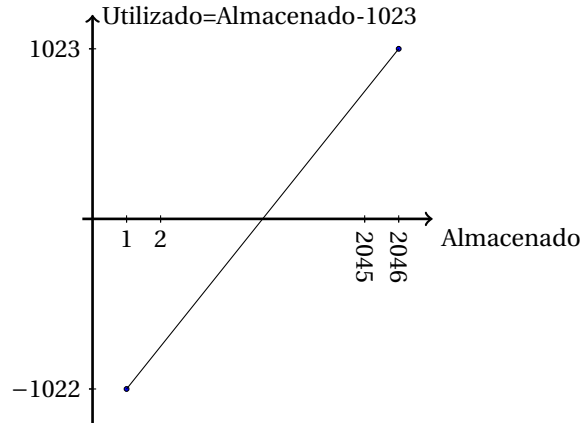


Figura 2.3: Relación entre el valor almacenado en el exponente y el valor que realmente representa.

Matemáticamente podemos interpretar un número almacenado en base 10 de la siguiente forma:

$$\begin{aligned} \boxed{\pm 1.b_1 b_2 \dots b_{52} \cdot 2^p} &= \underbrace{\pm}_{\text{signo}} \left(1 + \underbrace{\sum_{i=1}^{52} b_i \cdot 2^{-i}}_{\text{mantisa}} \right) \cdot 2^{p-\text{exponente}} \\ &= \pm (2^0 + b_1 \cdot 2^{-1} + b_2 \cdot 2^{-2} + \dots + b_{52} \cdot 2^{-52}) \cdot 2^p, \end{aligned}$$

donde,

$$\begin{aligned} p &= e_1 \cdot 2^{10} + e_2 \cdot 2^9 + e_3 \cdot 2^8 + \dots + e_{11} \cdot 2^0 - 1023, \\ &= \left(\sum_{i=1}^{11} e_{12-i} \cdot 2^{i-1} \right) - 1023. \end{aligned}$$

2.7. Caso especial del exponente 1111111111

Este caso corresponde a la situación donde los bits del exponente son todos 1, es decir $e_1 \dots e_{11} = 1111111111$. En esta situación tenemos 3 situaciones:

⁵Notar que se exceptúa del traslado el caso 0 y 2047 ya que con casos especiales que se explican en las secciones 2.8 y 2.7, respectivamente.

- Si el bit del signo es 0 y todos los bits de la mantisa son 0, entonces representa a $+\infty$. Esto se obtiene, por ejemplo, al dividir $1/0$.
- Si el bit del signo es 1 y todos los bits de la mantisa son 0, entonces representa a $-\infty$. Es decir, el signo del infinito lo determina efectivamente el bit de signo.
- Si alguno de los bits de la mantisa, digamos b_k para $k \in \{1, 2, \dots, 52\}$, es distinto de 0, entonces se interpreta como NaN, es decir *not-a-number*. Esto se obtiene por ejemplo al dividir $0/0$.

En la Tabla 2.2 se presenta un resumen de los casos.

Tabla 2.2: Resumen de caso cuando se tienen exponente $p = 2047$

S	e_1	e_2	e_3	...	e_{11}	b_1	b_2	...	b_{52}	lo que representa	
0	1	1	1	...	1	0	0	...	0	$+\infty$	1/0
1	1	1	1	...	1	0	0	...	0	$-\infty$	-1/0
1	1	1	1	...	1	x	x	...	x	<u>NaN</u>	0/0

2.8. Caso especial del exponente 0000000000

Este caso corresponde a la situación donde los bits del exponente son todos 0, es decir $e_1 \dots e_{11} = 0000000000$. Este caso especial considera números no normalizados para representar números sub-normales de la siguiente forma,

$$\boxed{\pm 0.b_1 b_2 b_3 \dots b_{52} \cdot 2^{-1022}}.$$

Notar los siguientes puntos claves,

- El exponente es fijo en -1022 .
- El número al costado derecho del signo ya no es 1, sino 0. Es decir no se está justificando a la izquierda, por lo tanto no es número normalizado.
- Los únicos bits que se pueden modificar son los bits de la mantisa.

Lo interesante de esta representación es que permite representar números positivos mucho más pequeños que lo que permite la representación normalizada.

Ahora, ya que entendemos todos los casos especiales y los no especiales, podemos hacernos la pregunta, ¿cuál es el número más pequeño representable en double precision? La solución requiere utilizar representación sub-normal de la siguiente forma:

$$\begin{array}{c|cccc|cccc|c} 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 1 \\ \hline s & e_1 & \dots & e_{11} & b_1 & \dots & \dots & b_{51} & b_{52} \end{array}$$

El cual puede obtenerse de la siguiente forma:

$$\begin{aligned} \boxed{+(0.0\dots 01) \cdot 2^{-1022}} &= (b_1 \cdot 2^{-1} + b_2 \cdot 2^{-2} + \dots + b_{52} \cdot 2^{-52}) \cdot 2^{-1022} \\ &= 1 \cdot 2^{-52} \cdot 2^{-1022} \\ &= 2^{-1074} \approx 4,94 \cdot 10^{-324}. \end{aligned}$$

Comentario al margen

Otro ejemplo de pérdida de importancia: Considere que tiene a su disposición un computador que opera con aritmética de 3-dígitos-decimales-significantes, es decir, solo almacena los primeros 3 dígitos partiendo desde el primer dígito distinto a 0. Considerando lo anterior, se le solicita implementar un algoritmo que obtenga el resultado de la siguiente operación:

$$\sqrt{9.01} - 3.$$

El computador también dispone de una librería que puede calcular raíces cuadradas, en particular conocemos $\sqrt{9.01} = 3.0016$, pero como es un computador con aritmética de solo 3 dígitos significantes podemos solo almacenar hasta antes de la barra vertical, es decir 3.00 como el resultado de la raíz cuadrada $\sqrt{9.01}$. Entonces, el primer algoritmo ingenuo que se propone es el siguiente:

```

1      a=9.01
2      b=3
3      c=sqrt(a) % se obtiene c=3.00
4      out = c-b % el cual da 0.00

```

Lamentablemente el valor de out entregado por esta implementación es 0. La pregunta que nos podemos hacer ahora es, ¿podemos construir otro algoritmo que entregue un resultado de mejor calidad? La respuesta es sí, considere el siguiente desarrollo,

$$\begin{aligned}\sqrt{a} - b &= (\sqrt{a} - b) \frac{(\sqrt{a} + b)}{(\sqrt{a} + b)} \\ &= \frac{(a - b^2)}{(\sqrt{a} + b)},\end{aligned}$$

donde lo único que se hizo fue multiplicar por un *uno* conveniente, en esta caso $\frac{(\sqrt{a} + b)}{(\sqrt{a} + b)}$. Esto nos dio una expresión distinta pero matemáticamente equivalente, entonces nuestro algoritmo nuevo sería el siguiente,

```

1      a=9.01
2      b=3
3      c=sqrt(a) % se sigue obteniendo c=3.00
4      d=b**2
5      out = (a-d)/(c+b) % el cual da 0.01/6=0.00167 \approx
      1.6*10^{-3} :-)

```

En este caso sí fuimos capaces de obtener el resultado correcto en los 3 primeros dígitos sin necesidad de recurrir a modificar el hardware disponible o utilizando más memoria que la disponible originalmente, solo necesitábamos modificar el algoritmo adecuadamente!

¡MUY IMPORTANTE!

Para finalizar esta sección es importante destacar que uno puede apalea algunos problemas de la pérdida de importancia al aumentar la cantidad de memoria disponible para almacenar números, por ejemplo utilizar `long double`. Pero esto significará una computación más lenta (dado que se tienen que procesar más bits por número) y eventualmente pueden surgir los mismos problemas que se quisieron evitar. Lo recomendado, y para lo cual no hay una receta, es analizar el problema en cuestión, entenderlo y luego proponer un algoritmo que evite pasar por lo cuellos de botella de pérdida de importancia! Este es un tema crucial y fundamental en Computación Científica!

2.10. Ejercicios Propuestos

1. Evalúe computacionalmente las siguientes funciones a medida que x tienda a 0^+ .

$$E_1(x) = \frac{1 - \cos(x)}{\sin^2(x)},$$
$$E_2(x) = \frac{1}{1 + \cos(x)}.$$

¿Son $E_1(x)$ y $E_2(x)$ idénticas?

2. Proponga un algoritmo que encuentre las raíces de $x^2 + 9^{12} \cdot x - 3 = 0$
3. ¿Cuándo habría problemas numéricos al evaluar $f(x)$ y $g(x)$?

$$f(x) = \frac{1 - (1 - x)^3}{x},$$
$$g(x) = \frac{1}{1 + x} - \frac{1}{1 - x}.$$

4. Diseñe un algoritmo para calcular las raíces de, $x^2 + b \cdot x - 10^{-12} = 0, b \geq 100$.
5. Se sugiere encarecidamente revisar el Jupyter Notebook en el siguiente [link](#).

Capítulo 3

Raíces en 1D

En este capítulo revisaremos distintos algoritmos para la búsqueda de ceros o, también denominadas, raíces de una función. En particular veremos el caso 1D, es decir analizaremos funciones $f(x) : \mathbb{R} \rightarrow \mathbb{R}$. El análisis para funciones de más dimensiones se estudiará en la sección 4.6.

Nuestra primera definición en este capítulo corresponde a lo que significa matemáticamente una raíz:

Def 6. La función $f(x)$ tiene una raíz en $x = r$ si $f(r) = 0$.

En la definición anterior hay 3 elementos claves:

1. La función $f(x)$.
2. El dominio de la variable x , que al no ser especificado se asume que es \mathbb{R} .
3. La raíz r .

Las 3 componentes son importantes y al mismo tiempo necesarias para poder aplicar cualquier algoritmo de los que estudiaremos. Primero, consideremos el caso del dominio. En general hay 2 opciones para este caso, (i) corresponde a lo ya indicado, el dominio son los números reales \mathbb{R} , sin embargo de todos modos se requerirá alguna estimación “gruesa” de donde puede estar la raíz, (ii) el dominio es acotado, por ejemplo un intervalo $[a, b]$, con $b > a$. En ambos casos, se sabe a priori un rango de valores en donde buscar la raíz.

Segundo, la función $f(x)$. Si bien los algoritmos que se estudiarán en el curso requieren la definición de la función, el desafío es en algunas ocasiones definir la función $f(x)$. Por ejemplo si tenemos una pseudo-parábola $p(x) = ax^2 - bx - c \sin(x)$, para valores conocidos de $a < 0$, b , y c , podemos hacernos la pregunta, ¿para que valor de x la función $p(x)$ tiene su máximo? Aquí, aún no tenemos definida la función $f(x)$ a la cual queremos encontrar la raíz, ¿cuál sería $f(x)$ entonces? En este caso $f(x) = p'(x) = 2ax - b - c \cos(x)$. En este caso queda claro que la aplicación de búsqueda de raíces estaría mal aplicada si es que buscamos una raíz de $p(x)$, ya que no es lo solicitado. Lo solicitado es buscar un máximo de $p(x)$, lo cual requiere un análisis preliminar del problema para transformarlo a la forma en donde podemos aplicar la teoría de búsqueda de raíces! La palabra destacada y muy importante es transformar, ya que ese es el valor agregado. Una vez transformado el problema, viene la componente un poco más mecánica de ejecutar un algoritmo que reciba $f(x)$ para encontrar la raíz.

Por último y no menos importante, aparece el tercer punto, la raíz r . Es importante no olvidar este punto ya que parece innecesario, pero es crucial. Por ejemplo, que ocurre si se le solicita encontrar algún valor de x que satisfaga la siguiente ecuación,

$$\sin(x) = 3.$$

Lo primero que podemos hacer es construir $f(x)$, en esta caso tenemos 2 opciones: $f_1(x) = \sin(x) - 3$ o $f_2(x) = 3 - \sin(x)$. Ambas definen las mismas raíces, si es que existen. La pregunta que falta hacerse acá es: ¿existe realmente

un valor de x tal que la función $\sin(x)$ sea 3? La respuesta es no, debido a que la función $\sin(x)$ satisface la siguiente desigualdad,

$$-1 \leq \sin(x) \leq 1.$$

Entonces no existe en este caso una raíz.

En resumen, al enfrentar un problema de búsqueda de ceros uno debe preguntarse:

- ¿Qué es $f(x)$?
- ¿Cuál es el dominio de x ?
- ¿Existe una raíz r de $f(x)$?

Comentario al margen

Antes de entrar en detalles técnicos, es importante presentar un problema inicial en donde podemos destacar la utilidad de este capítulo y su posible importancia en actividades cotidianas. Por ejemplo, en una entrevista de trabajo se le solicita a usted que construya un algoritmo que le entregue la raíz cuadrada de 2 con la mayor cantidad de decimales posibles. ¿Qué algoritmo propondría?^a

^aImplémtelo y vea cuanto se demora en obtener 12 decimales correctos de $\sqrt{2}$!

3.1. Método de la Bisección

Es importante mencionar en este punto que la construcción de todos los algoritmos en este capítulo dependen fuertemente de diversos teoremas que iremos recordando según sea necesario. El primer teorema a recordar es el siguiente,

Thm 2. Sea f una función continua en $[a, b]$, satisfaciendo $f(a)f(b) < 0$. Entonces f tiene una raíz entre a y b ; es decir, existe un número r que satisface $a < r < b$ y $f(r) = 0$.

Aquí hay 2 componentes claves a destacar del teorema, (i) que requiere que la función $f(x)$ sea continua¹ y que el producto $f(a)f(b)$ sea menor que 0. La Figura 3.1 muestra un sketch de las componentes del Teorema 2. Esto significa que si la función no es continua o no cumple que $f(a)f(b)$ sea menor que 0, no podemos concluir que existe un r en el intervalo $[a, b]$ tal que $f(r) = 0$. Estos detalles no son meramente cosas técnicas sin importancia, son cruciales para evitar realizar computación innecesaria.

Por ejemplo, considere $f(x) = 1$. En este caso la función es continua pero no alterna signo para ningún punto, por lo cual no existe un r que sea una raíz de $f(x)$. Considere ahora la siguiente función,

$$f(x) = x^3 + x - 1$$

¿Cómo se puede obtener r tal que $f(r) = 0$ dado el Teorema 2? Considere $a = 0$ y $b = 1$. ¿Qué podemos concluir? Primero, sabemos que $f(x)$ es continua. Ahora debemos evaluar $f(x)$ en a y en b :

$$f(a) = f(0) = -1,$$

$$f(b) = f(1) = 1.$$

Por lo tanto obtenemos,

$$f(a)f(b) = (-1)(1) = -1 < 0.$$

Por lo tanto, dado el Teorema 2, podemos concluir que existe una raíz r en el intervalo $[a, b]$. No conocemos el valor de r , pero sabemos que sí existe!

¿Puede imaginarse el algoritmo de la bisección?²

¹No necesariamente diferenciable.

²Le sugiero darse unos minutos para analizar como transformar el análisis anterior en un algoritmo!

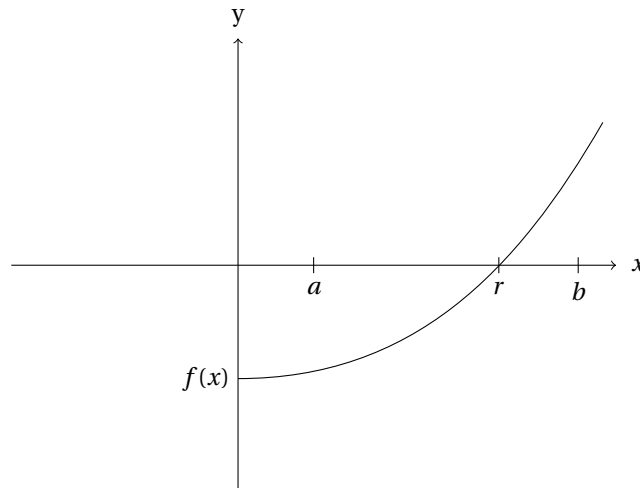


Figura 3.1: Sketch de componentes de Teorema 2.

3.1.1. Algoritmo

El algoritmo considera como input, como mínimo, 3 parámetros: $f(x)$, a y b . Además en esta implementación se espera un cuarto parámetro: TOL, el cual define la tolerancia o error que se aceptará para detener el algoritmo³. Dependiendo de la implementación, uno puede asumir que se cumple $f(a)f(b) < 0$ al comienzo o verificarlo. Ahora, suponiendo que sí se cumple al comienzo. Uno puede dividir el intervalo en 2 segmentos: $[a, c]$ y $[c, b]$, para $c = \frac{a+b}{2}$. Considerando que solo hay una raíz en el intervalo⁴ $[a, b]$, entonces la raíz debe estar en alguno de los 2 intervalos. ¿Qué podemos hacer para saber en que intervalo quedó la raíz? Ya sabemos que $f(x)$ es continua, por lo que lo único que nos queda por hacer es evaluar si $f(a)f(c)$ o $f(c)f(b)$ es menor que 0. En rigor solo uno de esos 2 términos será < 0 , por lo que no es necesario evaluar ambos. Al determinar en cual de los 2 intervalos está la raíz, podemos actualizar nuestro intervalo de búsqueda y seguir haciendo lo mismo iterativamente! Hasta que se cumpla la condición de término, asociada a la cuarta variable de input TOL.

Entonces, dado un intervalo inicial $[a, b]$ y $f(a)f(b) < 0$, el algoritmo para el método de la bisección es:

```

1  def biseccion(a, b, f, TOL) :
2      while (  $\frac{b-a}{2} > \text{TOL}$  ) :
3           $c = \frac{a+b}{2}$ 
4          if (  $f(c) = 0$  ) :
5              break
6          if (  $f(a)f(c) < 0$  ) :
7               $b = c$ 
8          else :
9               $a = c$ 
10     return  $\frac{a+b}{2}$ 

```

Hay que considerar un par de excepciones,

- Si la raíz es a o b , no es necesario ejecutar nada y ya se tiene la raíz!

³Más detalles en la Sección 3.6.

⁴Esto es en realidad un supuesto fuerte y conveniente. ¿Qué pasa si no se cumple?

- Si $f(c) = 0$, no es necesario continuar y la raíz sería c .

3.1.2. ¿Cuál es el error y qué tan rápido converge el método de la bisección?

Un comportamiento interesante del método de la bisección es que sabemos por construcción que el intervalo inicial $[a, b]$ se irá reduciendo a la mitad en cada iteración dado como se define c . Esto significa lo siguiente:

- Largo intervalo inicial $(b - a)$.
- Largo intervalo 1era iteración $\frac{(b-a)}{2}$.
- \vdots
- Largo intervalo tras n -ésima iteración $\frac{(b-a)}{2^n}$.

Si consideramos que la aproximación inicial obtenida por el método de la bisección la denominamos $x_c^{(0)} = \frac{a+b}{2}$ y además como sabemos que la raíz está en el intervalo $[a, b]$, podemos concluir que la diferencia entre la aproximación x_c y r satisface la siguiente desigualdad,

$$|x_c^{(0)} - r| \leq \frac{(b-a)}{2}.$$

Entonces, el error en la n -ésima iteración es,

$$\text{Error Absoluto} = |x_c^{(n)} - r| \leq \frac{(b-a)}{2^{n+1}}.$$

La cual requiere $n + 2$ evaluaciones de $f(x)$. Por lo tanto sabemos que el **Error Absoluto** de la aproximación se va reduciendo a la mitad en cada iteración!

Comentario al margen

En la siguiente sección volveremos a estudiar en más detalle este punto. La razón es que veremos algoritmos que hacen que decaiga el error mucho más rápido, pero también existe la posibilidad que el error decaiga más lento, por lo tanto no debe olvidar el método de la bisección!

Una buena forma para evaluar la eficiencia del método, es preguntarse cuanto puede decaer el error por cada evaluación de la función $f(x)$. Consideremos la siguiente definición,

Def 7. Se definirá que una solución es correcta en p decimales si el error absoluto es menor que $0,5 \cdot 10^{-p}$.

Entonces al definir p , imponemos una cota superior para el error aceptado. Por ejemplo, determine cuantas evaluaciones se necesitan con el método de la bisección para encontrar una raíz de $f(x) = \cos x - x$ en el intervalo $[0, 1]$ correcta en 6 decimales. Solución: Sabemos,

$$|x_c - r| < \frac{1-0}{2^{n+1}} < 0,5 \cdot 10^{-6}.$$

Despejando n obtenemos,

$$n > \frac{6}{\log_{10} 2} \approx 19,9.$$

Por lo tanto necesitamos $20 + 2 = 22$ evaluaciones para obtener una aproximación de la raíz correcta en 6 decimales. Lo interesante del desarrollo anterior es que podemos determinar el número de evaluaciones requeridas para obtener cierto error sin realizar ninguna evaluación de $f(x)$!

3.2. Iteración de Punto Fijo

Este tema está en el núcleo de la Computación Científica, no solo por su simplicidad de descripción, sino por su flexibilidad de aplicación a diversos problemas.

Nota

Para entender lo fantástico que es una iteración de punto fijo, iniciemos esta sección con un ejemplo. Busque una calculadora o use la calculadora en su teléfono móvil o tablet o simplemente ejecute Python en la terminal. Ahora elija algún número real cualquiera, luego aplique la función “cos” a ese número (¡asegúrese de que esté en radianes eso sí!) repetidas veces, es decir aplicar la función cos al número elegido inicialmente, luego al resultado y luego al nuevo resultado y así sucesivamente. Después de cierto número de iteraciones notará que el número no cambia, es decir converge a un número en específico, en este caso debería ser aproximadamente 0,73908513321516.... Entonces, la pregunta que surge es ¿Por qué ocurre esto?! En esta sección se dará una explicación a este fenómeno. Lo que usted acaba de ejecutar son varias iteraciones de una iteración de punto fijo!

Para formalizar lo que se acaba de explicar, necesitamos definir que es un punto fijo.

Def 8 (Punto fijo). *El número real r es un punto fijo de la función $g(x)$ si $g(r) = r$.*

¡MUY IMPORTANTE!

En la definición 8 se utiliza la notación de que r es un punto fijo, notar que un punto fijo no es lo mismo que la raíz de una función. Por lo tanto no confundirse con esta importante sutileza. Por ejemplo, una función $f(x)$ o una función $g(x)$ pueden tener una raíz e independientemente también pueden tener algún punto fijo. Ahora revisaremos la teoría asociada a la iteración de punto fijo, para luego explicar como la podemos utilizar para encontrar una raíz! ☺

3.2.1. Algoritmo

La componente mecánica de la computación asociada a una iteración de punto fijo es bastante directa. A continuación se presenta una breve ejecución genérica de una iteración de punto fijo con la función $g(x)$ partiendo del initial guess x_0 , acá se denomina el punto inicial como la iteración $i = 0$. La primera iteración obtiene x_1 como el resultado de evaluar $g(x)$ en x_0 , y así sucesivamente. Entonces se obtiene la siguiente ejecución,

$$\begin{aligned} i = 0 : x_0 &= \text{“initial guess”} \\ i = 1 : x_1 &= g(x_0) \\ i = 2 : x_2 &= g(x_1) \\ &\vdots \\ i = n : x_n &= g(x_{n-1}) \end{aligned}$$

Por lo tanto, el algoritmo queda expresado de la siguiente forma,

```
1   $x_0$  = Initial Guess
2  for i in range(1,n):
3       $x_i$  =  $g(x_{i-1})$ 
```

Para entender la convergencia de una iteración de punto fijo, considere el siguiente teorema.

Thm 3 (Teorema de los límites continuos). Sea f una función continua en un vecindario de x_0 , y suponga que $\lim_{n \rightarrow \infty} x_n = x_0$. Entonces:

$$\lim_{n \rightarrow \infty} f(x_n) = f\left(\lim_{n \rightarrow \infty} x_n\right) = f(x_0).$$

En otras palabras los límites pueden trasladarse al interior de funciones continuas.

En el caso de una iteración de punto fijo considera que tiene una secuencia x_i que puede converger o no a medida que el número de pasos tiende a infinito. Sin embargo si g es continua y las x_i convergen a un número r , entonces r es un punto fijo de g . Entonces si conocemos $g(x)$, que $\lim_{i \rightarrow \infty} x_i = r$ y el Teorema 3, obtenemos,

$$g(r) = g\left(\lim_{i \rightarrow \infty} x_i\right) = \lim_{i \rightarrow \infty} g(x_i) = \lim_{i \rightarrow \infty} x_{i+1} = r,$$

Es decir, r es un punto fijo de $g(r)$.

Comentario al margen

Para nuestro ejemplo inicial con la función coseno, tenemos que la iteración de punto fijo quedara definida como:

$$x_{i+1} = \cos(x_i),$$

donde $g(x) = \cos(x)$.

¡MUY IMPORTANTE!

Equivalentemente, podemos interpretar esta iteración de punto fijo como si estuviéramos encontrando una raíz de $f(x)$ donde $f(x) = x - \cos(x)$. Por lo que si r es un punto fijo de $g(x)$, lo que significa que $g(r) = r$, entonces implica que r es una raíz de $f(x) = x - \cos(x)$, dado que $f(r) = r - \cos(r) = 0$.

3.2.2. Ejemplos de $g(x)$

Considere que interesa buscar una raíz de $f(x) = x^3 + x - 1$ utilizando una iteración de punto fijo. La pregunta que surge es ¿dónde está $g(x)$? La respuesta es que no hay una única función $g(x)$ que tienen como punto fijo una raíz de $f(x)$! La diferencias entre las diferentes opciones que existen es que algunas iteraciones convergerán más rápidos que otras e incluso otras podrían divergir!⁵ Considere las siguientes 3 opciones:

- 1.- Despejando x del segundo término de $f(x)$ obtenemos:

$$x = 1 - x^3 = g_1(x)$$

- 2.- Despejando x del término cúbico de $f(x)$ obtenemos:

$$x = \sqrt[3]{1 - x} = g_2(x)$$

⁵Aquí es importante destacar que las iteraciones de punto fijo que convergen son útiles para encontrar el punto fijo y las que divergen pueden servir para generar números pseudo-aleatorios! Ver el capítulo 9 del libro guía.

3.- Inspirándose profundamente y despejando x :

$$\begin{aligned}
 x^3 + x - 1 &= 0 & / + 1 \\
 x^3 + x &= 1 & / + 2x^3 \\
 3x^3 + x &= 1 + 2x^3 \\
 x(3x^2 + 1) &= 1 + 2x^3 \\
 x &= \frac{1 + 2x^3}{1 + 3x^2} = g_3(x)
 \end{aligned}$$

Estás 3 funciones, $g_1(x)$, $g_2(x)$, y $g_3(x)$, son iteraciones de punto fijo válidas para obtener alguna raíz de $f(x)$. Y en realidad podemos obtener infinitas funciones de iteración de punto fijo! El desafío está en poder construir alguna que requiere pocas iteraciones. Esta pregunta la responderemos en las siguientes secciones. Ahora, ¿podría usted proponer alguna otra iteración de punto fijo como las anteriores?

3.2.3. Geometría de la iteración de Punto Fijo – Diagrama Cobweb

En esta sección veremos 2 ejemplos numéricos de un par de iteraciones de punto fijo. En el primer caso se presentará una iteración de punto fijo que diverge y el segundo caso una que converge. Note que ambas funciones a analizar, $g_1(x)$ y $g_2(x)$, tiene un punto fijo, sin embargo solo para la segunda función se podrá obtener el punto fijo iterando la función respectiva. La primera función que estudiaremos es la siguiente,

$$g_1(x) = -\frac{3}{2}x + \frac{5}{2}.$$

Para la cual definimos su initial guess en $x_0 = 1,1$ y si iteramos 20 veces obtenemos el siguiente resultado:

$x_0 = 1,1,$	$x_7 = g_1(x_6) = -0,708594,$	$x_{14} = g_1(x_{13}) = 30,1929,$
$x_1 = g_1(x_0) = 0,85,$	$x_8 = g_1(x_7) = 3,56289,$	$x_{15} = g_1(x_{14}) = -42,7894,$
$x_2 = g_1(x_1) = 1,225,$	$x_9 = g_1(x_8) = -2,84434,$	$x_{16} = g_1(x_{15}) = 66,6841,$
$x_3 = g_1(x_2) = 0,6625,$	$x_{10} = g_1(x_9) = 6,7665,$	$x_{17} = g_1(x_{16}) = -97,5261,$
$x_4 = g_1(x_3) = 1,50625,$	$x_{11} = g_1(x_{10}) = -7,64976,$	$x_{18} = g_1(x_{17}) = 148,789,$
$x_5 = g_1(x_4) = 0,240625,$	$x_{12} = g_1(x_{11}) = 13,9746,$	$x_{19} = g_1(x_{18}) = -220,684,$
$x_6 = g_1(x_5) = 2,13906,$	$x_{13} = g_1(x_{12}) = -18,462,$	$x_{20} = g_1(x_{19}) = 333,526.$

Claramente se observa que los resultados numéricos divergen. Una alternativa para analizar el comportamiento de la iteración de punto fijo es el diagrama de Cobweb. Este diagrama grafica la evolución de la iteración de punto fijo en un plano cartesiano uniendo los puntos obtenidos. La visualización consiste en graficar la función $y = g(x)$ y $y = x$ en un dominio adecuado. Luego, en cada iteración i , se definen 2 pasos, el primero consiste en dibujar una línea entre las coordenadas $(x_i, x_{i+1} = g(x_i))$ y (x_{i+1}, x_{i+1}) , y en el segundo paso se dibuja una línea entre las coordenadas (x_{i+1}, x_{i+1}) y $(x_{i+1}, x_{i+2} = g(x_{i+1}))$. Este sucesión de líneas conectando puntos genera un espiral que se expande o se contrae cuando diverge o converge, respectivamente. En la Figura 3.2 se observa la evolución del diagrama Cobweb para las primeras 6 iteraciones del ejemplo numérico.

La Figura 3.2 muestra en azul la gráfica de la función $y = x$ y en rojo la gráfica de $y = g_1(x)$, y se observa claramente que el espiral se expande. Además de la evolución de la iteración de punto fijo, el diagrama Cobweb nos

muestra explícitamente el punto fijo de $g_1(x)$, que es cuando $g_1(r) = r$, es decir es cuando se intersectan las funciones $y = g_1(x)$ y $y = x$. Esto nos muestra que $g_1(x)$ tiene un punto fijo, pero no lo podemos obtener iterando con $g_1(x)$. La pregunta que surge ahora es: ¿Cuál es el punto fijo?⁶

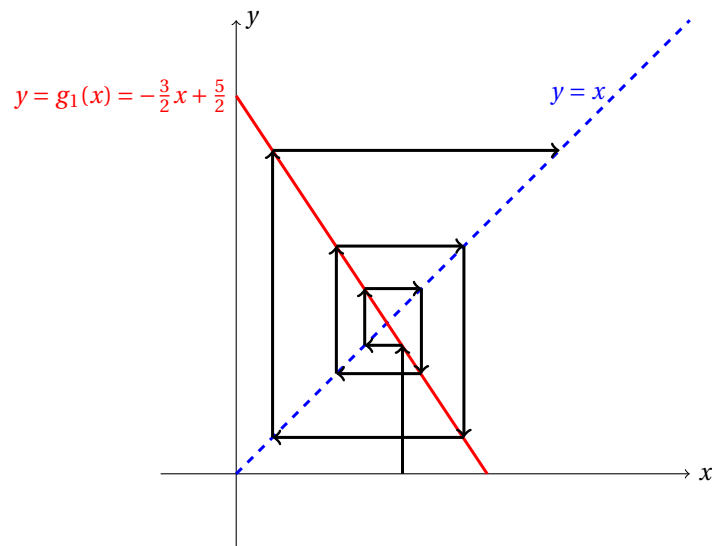


Figura 3.2: Ejemplo de divergencia de una iteración de punto fijo en un diagrama Cobweb.

Ahora, ya vimos una iteración de punto fijo divergente, pero acá nos interesa hacer que las iteraciones de punto fijo converjan. Considere la siguiente función,

$$g_2(x) = -\frac{1}{2}x + \frac{3}{2}.$$

Donde el resultado de realizar 20 iteraciones de punto fijo es el siguiente,

$x_0 = 2,7,$	$x_7 = g_2(x_6) = 0,986719,$	$x_{14} = g_2(x_{13}) = 1,0001,$
$x_1 = g_2(x_0) = 0,15,$	$x_8 = g_2(x_7) = 1,00664,$	$x_{15} = g_2(x_{14}) = 0,999948,$
$x_2 = g_2(x_1) = 1,425,$	$x_9 = g_2(x_8) = 0,99668,$	$x_{16} = g_2(x_{15}) = 1,00003,$
$x_3 = g_2(x_2) = 0,7875,$	$x_{10} = g_2(x_9) = 1,00166,$	$x_{17} = g_2(x_{16}) = 0,999987,$
$x_4 = g_2(x_3) = 1,10625,$	$x_{11} = g_2(x_{10}) = 0,99917,$	$x_{18} = g_2(x_{17}) = 1,00001,$
$x_5 = g_2(x_4) = 0,946875,$	$x_{12} = g_2(x_{11}) = 1,00042,$	$x_{19} = g_2(x_{18}) = 0,999997,$
$x_6 = g_2(x_5) = 1,02656,$	$x_{13} = g_2(x_{12}) = 0,999792,$	$x_{20} = g_2(x_{19}) = 1,0000.$

En este caso podemos concluir que sí existe clara convergencia! Y notamos de inmediato que el punto fijo también es 1. La Figura 3.3 muestra las primeras 6 iteraciones de la evolución, no se incluyen más iteraciones porque no serían distinguible en el diagrama. En este caso obtenemos un espiral que se contrae

⁶ $r = 1.$

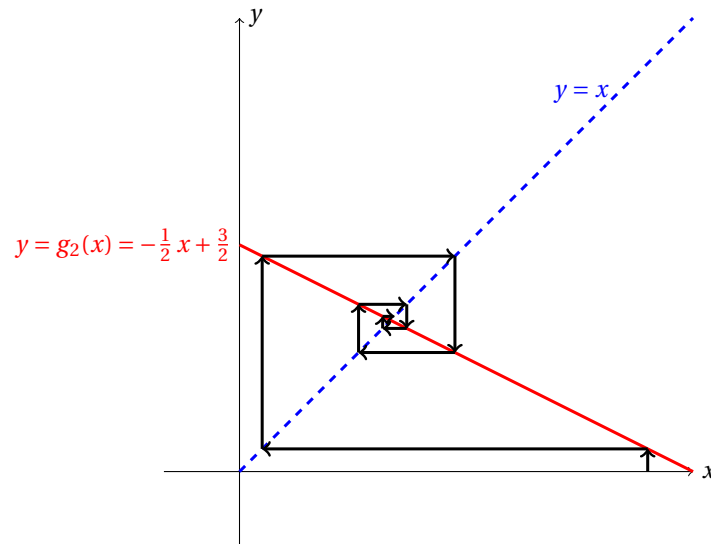


Figura 3.3: Ejemplo de convergencia de una iteración de punto fijo en un diagrama Cobweb.

En ambos casos notamos lo interesante que es el comportamiento de la evolución de ambas iteraciones de punto fijo. Sin embargo se omitió un caso, que es cuando no hay divergencia ni convergencia, ¿podría usted construir algebraicamente este caso?⁷

Ahora, en la siguiente sección estudiaremos algebraicamente las condiciones de convergencia y divergencia de una iteración de punto fijo.

3.2.4. Convergencia lineal de iteración de punto fijo

El análisis de la convergencia de una iteración de punto fijo requiere la utilización del Teorema del valor medio, el cual se presenta a continuación,

Thm 4 (Teorema del valor medio). Sea f una función continua en el intervalo $[a, b]$ y diferenciable en el intervalo $]a, b[$.

Entonces existe un número “ c ”, entre a y b de tal forma que $f'(c) = \frac{f(b) - f(a)}{(b - a)}$.

En la Figura 3.4 se observa un sketch de lo que significa el Teorema del valor medio gráficamente. En particular se muestra un ejemplo de una función $f(x)$ y se define el intervalo $[a, b]$, entonces al obtener el valor de la secante $\frac{f(b) - f(a)}{(b - a)}$ notamos que existe un valor $c \in [a, b]$ tal que $f'(c) = \frac{f(b) - f(a)}{(b - a)}$.

Ahora, antes de estudiar las condiciones de convergencia de una iteración de punto fijo, debemos definir la noción de convergencia lineal,

Def 9 (Convergencia lineal). Sea $e_i = |x_i - r|$ el error absoluto del paso i de un método iterativo. Si se cumple que:

$$\lim_{i \rightarrow \infty} \frac{e_{i+1}}{e_i} = S < 1.$$

Se dice entonces que el método obedece “convergencia lineal” con tasa S .

⁷Hay que pensar un poco acá!

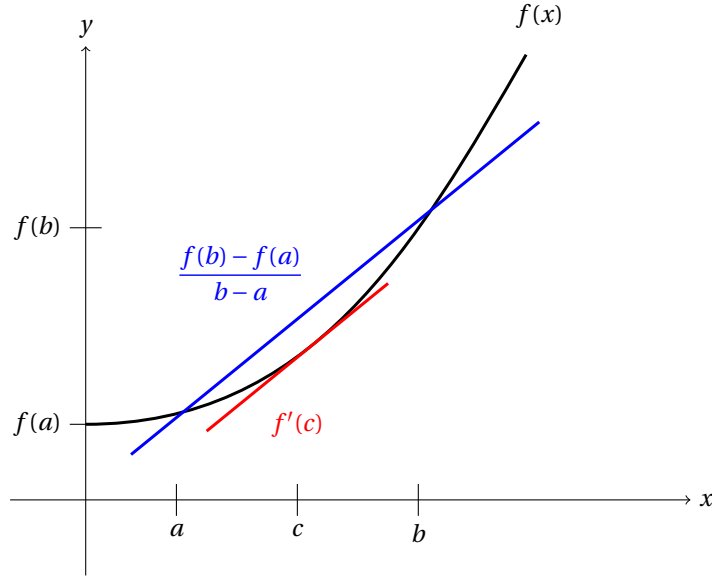


Figura 3.4: Sketch asociado del Teorema del valor medio. En negro se presenta la gráfica de la función $f(x)$, en este caso con forma parabólica. En azul se muestra una recta que une los puntos $(a, f(a))$ y $(b, f(b))$ que tiene pendiente $\frac{f(b)-f(a)}{b-a}$. En rojo se muestra la recta tangente a la función $f(x)$ en el punto c que tiene pendiente $f'(c) = \frac{f(b)-f(a)}{b-a}$.

Comentario al margen

Dada la definición de “convergencia lineal” recién presentada, ¿cuál sería la tasa de convergencia del método de la bisección?

Ahora ya tenemos lo necesario para presentar y demostrar cuando converge una iteración de punto fijo,

Thm 5 (Convergencia de una iteración de punto fijo). *Asuma que una función $g(x)$ es continua y diferenciable, que tiene un punto fijo $g(r) = r$, y que $S = |g'(r)| < 1$. Entonces la iteración de Punto Fijo converge linealmente con tasa S al punto fijo r para “initial guesses” lo suficientemente cerca de r .*

Demostración. Sea x_i el i -ésimo término de la iteración de punto fijo $x_i = g(x_{i-1})$ con “initial guess” x_0 . De acuerdo al teorema del valor medio $\left(f'(c) = \frac{(f(b)-f(a))}{(b-a)}\right)$, podemos indicar lo siguiente,

$$g'(c_i) = \frac{(g(x_i) - g(r))}{x_i - r},$$

donde se interpreta $g(x)$ como $f(x)$, $a = r$, $b = x_i$ y $c = c_i$. Recordando $x_{i+1} = g(x_i)$ y $r = g(r)$ podemos hacer el siguiente desarrollo,

$$\begin{aligned} g'(c_i) &= \frac{(g(x_i) - g(r))}{(x_i - r)} \\ &= \frac{(x_{i+1} - r)}{(x_i - r)}. \end{aligned}$$

Ahora, moviendo al lado izquierdo el denominador del lado derecho obtenemos,

$$g'(c_i)(x_i - r) = (x_{i+1} - r).$$

Recordando la definición de error $e_i = |x_i - r|$, aplicando valor absoluto y alternado los lados de la ecuación obtenemos,

$$e_{i+1} = |g'(c_i)| e_i. \quad (3.1)$$

Por otro lado, si $S = |g'(r)| < 1$, entonces existe un pequeño vecindario alrededor de r , digamos $[r - \epsilon, r + \epsilon]$, tal que para todo x en ese vecindario se tiene que $|g'(x)| \leq \frac{S+1}{2}$, donde $\frac{S+1}{2}$ es simplemente el promedio entre 1 y $S < 1$, lo cual es claramente menor a 1. Entonces para x_i lo suficientemente cerca de r , implica que c_i pertenece al vecindario $[r - \epsilon, r + \epsilon]$ entonces,

$$e_{i+1} = |g'(c_i)| e_i \leq \frac{S+1}{2} e_i.$$

Lo cual significa que el error decrece, por lo menos, con el factor $\frac{S+1}{2} < 1$. Ahora, obteniendo el cociente entre e_{i+1} y e_i en la ecuación (3.1), y tomando el límite a medida que i tiende a infinito se obtiene,

$$\lim_{i \rightarrow \infty} \frac{e_{i+1}}{e_i} = \lim_{i \rightarrow \infty} |g'(c_i)| = |g'(r)| = S.$$

□

Por lo tanto, podemos concluir que una iteración de punto fijo convergente converge linealmente con tasa $S = |g'(r)|$. Es decir, para saber si una iteración de punto fijo convergerá se debe obtener $|g'(r)|$, sin embargo en general uno no conoce r a priori! Dado que es lo que anda buscando, sin embargo sí puede conocer una estimación de S al ejecutar un par de iteraciones de punto fijo. ¿Cómo en realidad se puede obtener una estimación de S ? La teoría dice que S es aproximadamente $\frac{e_{i+1}}{e_i}$, y $e_i = |x_i - r|$, es decir, de todos modos necesitamos r ! Una alternativa es estimar el error e_i como $|x_i - x_{i+1}|$, donde en este caso x_{i+1} actúa como una aproximación de r . Entonces lo que sí se puede obtener es $S = \frac{e_{i+1}}{e_i} \approx \frac{|x_{i+1} - x_{i+2}|}{|x_i - x_{i+1}|}$.

Comentario al margen

Ahora es momento de enfrentarse a un problema interesante. Considere que usted tiene a su disposición un hardware que puede computar muy rápidamente una función $g(x)$ y además conoce un “initial guess” x_0 . Lamentablemente al aplicar el algoritmo de iteración de punto fijo no se obtiene una sucesión de términos convergentes, pero sí se sabe que existe un punto fijo $r = g(r)$. ¿Es posible proponer una nueva iteración de punto fijo convergente basada en $g(x)$? Digamos $r = G(r)$, donde al iterar con $G(x)$ si se obtenga una sucesión de términos convergente a r . Proponga por lo menos una. Hint: Podría utilizar $g_1(x)$ de la sección 3.2.3 para verificar su propuesta.

3.3. Teoremas y definiciones útiles

En esta sección se recordarán un serie de teoremas que serán utilizados en las siguientes secciones.

Primero, debemos precisar el concepto de “localmente convergente”, al cual nos hemos referido anteriormente.

Def 10 (Método localmente convergente). *Un método iterativo es llamado localmente convergente a r si el método converge a r para “initial guesses” suficientemente cercanos a r .*

En otras palabras, un método converge localmente a la raíz r , si existe una vecindad $(r - \epsilon, r + \epsilon)$, para $\epsilon > 0$, de tal manera que se de la convergencia a r a partir de “initial guesses” que estén en el intervalo $(r - \epsilon, r + \epsilon)$, i.e. $x_0 \in (r - \epsilon, r + \epsilon)$. Esta definición propone el concepto de vecindario, sin embargo en la práctica no es trivial construirlo y se requiere una combinación de estudio particular de la función en cuestión y experimentos numéricos.

El siguiente teorema se parece mucho al teorema del valor medio, Teorema 4, sin embargo es distinto.

Thm 6 (Teorema del valor intermedio). *Sea f una función continua en un intervalo $[a, b]$, entonces f recorre cada valor entre $f(a)$ y $f(b)$. Es decir, si y es un número entre $f(a)$ y $f(b)$, entonces existe un número c en el intervalo $[a, b]$ tal que $f(c) = y$.*

Nota

En el teorema del valor intermedio, Teorema 6, nuevamente aparece la referencia un valor c en un intervalo, digamos $[a, b]$, al igual que apareció en teorema del valor medio, Teorema 4. Por lo tanto, es muy importante poner atención a esos detalles.

El teorema de Rolle puede interpretarse como un caso particular del teorema de valor medio, Teorema 4, considerando $f(a) = f(b)$. **Le invito a verificarlo, ¡no crea todo lo que lee o le cuenten!**

Thm 7 (Teorema de Rolle). *Sea f una función continua y diferenciable en un intervalo $[a, b]$ y asuma que $f(a) = f(b)$. Entonces existe un número c , entre a y b de tal forma que $f'(c) = 0$.*

Finalmente llegamos a un teorema $\left[\lim_{n \rightarrow \infty} \sum_{i=1}^n (\text{muy})^n \right]$ importante, el gran teorema de Taylor!⁸

Thm 8 (Teorema de Taylor con residuo). *Sea x y x_0 números reales, y $f(x)$ $k+1$ -veces continuamente diferenciable en el intervalo entre x y x_0 , entonces existe un número c entre x y x_0 tal que:*

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + \frac{f^{(k+1)}(c)}{(k+1)!}(x - x_0)^{k+1}$$

Notar que la constante c aparece evaluando la $(k+1)$ -ésima derivada de $f(x)$, es decir el último término de la expansión de Taylor. Otra observación importante es que esta relación es válida para la tríada (x, x_0, c) , es decir x , x_0 y c son valores específicos, si se modifica uno de ellos, los otros no son necesariamente válidos, por lo que la identidad no tiene porque mantenerse válida.

En la práctica, uno aproxima la función $f(x)$ en un vecindario entorno a x_0 , y, aunque no sería válida la identidad, uno puede acotar el error por medio del primer término omitido. En general uno utiliza solo los primeros términos de la expansión para analizar algunos comportamientos locales de una función. Por ejemplo, si uno quiere estudiar la función $f(x)$ entorno al punto x_0 , uno puede ir tomando términos de la expansión de Taylor y analizar como se comportan la función, por ejemplo considere los siguientes casos,

Considerando 1 término: $f(x) \approx f(x_0)$, donde el error es orden $\mathcal{O}((x - x_0))$.

Considerando 2 términos: $f(x) \approx f(x_0) + f'(x_0)(x - x_0)$, donde el error es orden $\mathcal{O}((x - x_0)^2)$.

En el primer caso la aproximación es una constante, luego una recta, después vendría una parábola y así sucesivamente. Claramente notamos que la aproximación es exacta justo cuando $x = x_0$ y el error o diferencia respecto al valor real empieza a aumentar a medida que uno se aleja de x_0 . Por lo tanto Taylor es útil, en principio, para comportamientos locales, ¡que son precisamente los que nos interesan!

⁸Ver más detalles en [Taylor Theorem en Wikipedia](#) y [Prof. Brook Taylor](#) para conocer más del mismo Taylor! El Teorema fue publicado hace poco más de 300 años y hasta el día de hoy perdura su importancia!

3.4. Método de Newton-Raphson

Aproximadamente en 1670 Newton propone un método basado en aproximaciones lineales, para lo cual Newton debía obtener nuevas ecuaciones en cada iteración, y luego en 1690 Raphson notó que no era necesario y presentó algunos ejemplos. Luego se obtuvo la interpretación en función de la recta tangente y se convierte en lo que conocemos hoy como el método de Newton-Raphson o simplemente en método de Newton⁹.

3.4.1. Introducción

El Teorema de Taylor 8 nos indica que tenemos la siguiente identidad al considerar 3 términos de la expansión para una función $f(x)$,

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(c)}{2!}(x - x_0)^2.$$

Consideremos ahora que estamos interesados en encontrar una raíz de la función $f(x)$, es decir queremos encontrar r tal que $f(r) = 0$. Reemplazando x por r en la expansión de Taylor de 3 términos obtenemos,

$$f(r) = f(x_0) + f'(x_0)(r - x_0) + \frac{f''(c)}{2!}(r - x_0)^2.$$

De la cual podemos simplificar el lado izquierdo dado que sabemos que $f(r) = 0$, entonces obtenemos,

$$0 = f(x_0) + f'(x_0)(r - x_0) + \frac{f''(c)}{2!}(r - x_0)^2.$$

Despejando r del término lineal obtenemos,

$$\begin{aligned} -f'(x_0)(r - x_0) &= f(x_0) + \frac{f''(c)}{2!}(r - x_0)^2 \\ r - x_0 &= -(f'(x_0))^{-1} f(x_0) - (f'(x_0))^{-1} \frac{f''(c)}{2}(r - x_0)^2 \\ r &= x_0 - \frac{f(x_0)}{f'(x_0)} - \frac{\frac{f''(c)}{2}(r - x_0)^2}{f'(x_0)}. \end{aligned}$$

Este despeje algebraico es exacto, pero requiere conocer el valor de c en $f''(c)$ para poder determinar c .

Nota

De forma coloquial podemos indicar que para encontrar una raíz $f(r) = 0$, el método de Newton inicia de una estimación inicial x_0 y se traza la recta tangente a la función $f(x)$ en x_0 . La recta tangente seguirá en forma aproximada a la función hasta que intersecte el eje x . El punto de intersección de la recta tangente con el eje x es la nueva aproximación de la raíz, y así sucesivamente.

Por lo tanto para obtener una aproximación de la raíz r , digamos x_1 a partir de un valor conocido x_0 , se omite el término cuadrático¹⁰, es decir no se considera el último término de la ecuación anterior. Por lo tanto se obtiene,

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

Lo cual da origen al método de Newton, o también se puede denominar como la iteración de punto fijo de Newton,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} =: g_N(x_i).$$

⁹Ver más detalle en The Princeton Companion to Mathematics, 2008, sección II.4.2.3

¹⁰Esto se hace bajo el argumento que si $|r - x_0|$ es pequeño por lo tanto $|r - x_0|^2$ es aún más pequeño, lo cual es válido si $|r - x_0| < 1$.

3.4.2. Algoritmo y ejemplo numérico

En general, el método de Newton converge mucho más rápido que el método de la Bisección o alguna iteración de punto fijo con convergencia lineal. En particular, el método de Newton puede converger cuadráticamente, lo cual lo veremos en la siguiente sección. Algorítmicamente se traduce en la siguiente implementación:

```

1   $x_0 = \text{Initial Guess}$ 
2  for  $i$  in  $\text{range}(n)$  :
3       $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ 

```

Para ejemplificar su uso, utilicemos la siguiente función $f(x) = x^3 + x - 1$, donde $f'(x) = 3x^2 + 1$. Esta función es la misma que se utilizó en los ejemplos de iteraciones de punto fijo en la Sección 3.2.2. Entonces, al construir la iteración de punto fijo de Newton obtenemos lo siguiente,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$x_{i+1} = x_i - \frac{x_i^3 + x_i - 1}{3x_i^2 + 1}$$

$$x_{i+1} = \frac{2x_i^3 + 1}{3x_i^2 + 1} =: g_N(x_i).$$

En particular notamos que la iteración de punto fijo de Newton obtenida en este caso es idéntica al tercer ejemplo de la Sección 3.2.2, es decir $g_3(x)$ de la Sección 3.2.2 es igual a $g_N(x)$ aquí. En la Tabla 3.1 presentamos el resultado numérico de aplicar el método de Newton a la función anteriormente presentada considerando como initial guess $x_0 = -0.7$.

Tabla 3.1: Ejemplo del método de Newton.

i	x_i	e_i	$\frac{e_i}{e_{i-1}^2}$
0	-0.7	1.38	-
1	0.12	0.55	0.2906
2	0.95	0.27	0.8933
3	0.73	0.052	0.6924
4	0.6845	0.0022	0.8214
5	0.682332	0.00000437	0.8527
6	0.68232780	0.00000000	0.8541
7	0.68232780	0.00000000	- - -

3.4.3. Convergencia Cuadrática del Método de Newton

En la definición 9 se introdujo la primera definición de convergencia que indicaba que si el cociente de errores $\frac{e_{i+1}}{e_i}$ era constante y menor a 1 entonces se obtenía convergencia lineal. Además en la definición 10 se formalizó el concepto de convergencia local. Ahora, se presenta la definición de Convergencia Cuadrática, que implica una convergencia más rápida que una convergencia lineal, es decir, en un menor número de iteraciones,

Def 11 (Convergencia Cuadrática). Sea e_i el error de la iteración i de un método iterativo. La iteración converge cuadráticamente si:

$$\lim_{i \rightarrow \infty} \frac{e_{i+1}}{e_i^2} = M < \infty.$$

Thm 9. Sea $f(x)$ una función dos veces continuamente diferenciable y $f(r) = 0$. Si $f'(r) \neq 0$, entonces el método de Newton es local y cuadráticamente convergente a r . El error e_i en el paso i satisface lo siguiente,

$$\lim_{i \rightarrow \infty} \frac{e_{i+1}}{e_i^2} = M,$$

donde,

$$M = \frac{|f''(r)|}{2|f'(r)|}.$$

Demostración. La demostración contienen 2 componentes: (i) Demostrar que converge localmente y (ii) demostrar que converge cuadráticamente. (i) Para demostrar que converge localmente, consideraremos el Teorema 5, denominado Convergencia de un iteración de punto fijo y la definición 10, sobre la convergencia local de un método. Para nuestro objetivo, analizaremos el método de Newton como una iteración de punto fijo, la función a utilizar es la siguiente,

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

La cual claramente tiene como punto fijo r , dado que $g(r) = r - \frac{f(r)}{f'(r)} = r - \frac{0}{f'(r)} = r$, y $f'(r) \neq 0$. Ahora necesitamos demostrar, según lo requerido por el Teorema 5, que $|g'(r)| < 1$, entonces obtenemos,

$$\begin{aligned} g'(x) &= 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} \\ &= \frac{f(x)f''(x)}{f'(x)^2}. \end{aligned}$$

Evaluando en $x = r$ y recordando que $f(r) = 0$ y $f'(r) \neq 0$, obtenemos,

$$g'(r) = \frac{f(r)f''(r)}{(f'(r))^2} = 0.$$

Por lo tanto el método de Newton converge localmente.

Ahora se requiere demostrar (ii), es decir que converge cuadráticamente. Para tal objetivo, utilizaremos nuevamente la expansión de Taylor con 3 términos de $f(x)$ entorno a $x = r$,

$$\begin{aligned} f(r) &= f(x_i) + f'(x_i)(r - x_i) + \frac{f''(c_i)}{2}(r - x_i)^2 \\ 0 &= f(x_i) + (r - x_i)f'(x_i) + (r - x_i)^2 \frac{1}{2}f''(c_i). \end{aligned}$$

Moviendo al lado izquierdo $f(x_i)$, dividiendo por $f'(x_i)$ y luego sumando x_i en ambos lados obtenemos,

$$\begin{aligned} -f(x_i) &= (r - x_i)f'(x_i) + (r - x_i)^2 \frac{1}{2}f''(c_i) \\ \frac{-f(x_i)}{f'(x_i)} &= r - x_i + (r - x_i)^2 \frac{1}{2} \frac{f''(c_i)}{f'(x_i)} \\ x_i - \frac{f(x_i)}{f'(x_i)} &= r + (r - x_i)^2 \frac{1}{2} \frac{f''(c_i)}{f'(x_i)}. \end{aligned}$$

Notamos que al lado izquierdo se obtiene una iteración de método de Newton, el cual da exactamente x_{i+1} .

$$x_{i+1} = r + (r - x_i)^2 \frac{1}{2} \frac{f''(c_i)}{f'(x_i)}.$$

Ahora, al pasar r al lado izquierdo, obtenemos la e_{i+1} al lado izquierdo y e_i al lado derecho luego de aplicar valor absoluto,

$$\begin{aligned} x_{i+1} - r &= (r - x_i)^2 \frac{1}{2} \frac{f''(c_i)}{f'(x_i)} \\ |x_{i+1} - r| &= |r - x_i|^2 \frac{1}{2} \frac{|f''(c_i)|}{|f'(x_i)|} \\ e_{i+1} &= e_i^2 \frac{1}{2} \frac{|f''(c_i)|}{|f'(x_i)|}. \end{aligned}$$

Dividiendo por e_i^2 y luego tomando el límite a medida que i tiende a infinito obtenemos,

$$\begin{aligned} \lim_{i \rightarrow \infty} \frac{e_{i+1}}{e_i^2} &= \lim_{i \rightarrow \infty} \frac{1}{2} \frac{|f''(c_i)|}{|f'(x_i)|} \\ &= \frac{1}{2} \left| \frac{f''(r)}{f'(r)} \right|. \end{aligned}$$

Por lo tanto se obtiene la constante M incluida en la definición de convergencia cuadrática 11. Lo que finaliza la demostración de convergencia local y cuadrática del método de Newton. \square

¡MUY IMPORTANTE!

En el análisis anterior se utilizó por lo menos 2 veces el hecho de que $f'(r) \neq 0$, ¿Qué pasaría si $f'(r) = 0$? ¿Se puede imaginar una función que tenga una raíz, i.e. $f(r) = 0$ y además su derivada sea 0 en el mismo punto, i.e. $f'(r) = 0$?^a

^aUna forma de obtener una función que satisfaga la restricción $f(r) = f'(r) = 0$ es cuando la función tenga una raíz justo en un mínimo o máximo o punto de inflexión, que en realidad no es necesariamente un caso fortuito y puede ocurrir regularmente.

3.4.4. Convergencia Lineal del Método de Newton

En la sección anterior demostramos que el método de Newton convergía local y cuadráticamente, pero que dependía fuertemente que $f'(r) \neq 0$. Entonces ahora analizaremos que ocurre con el método de Newton cuando esa restricción no se cumple y que se puede hacer en este caso.

Asuma que r es una raíz de f y que f es diferenciable. Entonces si $0 = f(r) = f'(r) = \dots = f^{(m-1)}(r)$, pero $f^{(m)}(r) \neq 0$, decimos que la raíz r de f tiene una multiplicidad m . Decimos que f tiene una raíz múltiple si $m > 1$, en otro caso ($m = 1$), la raíz se dice que es simple.

Para ejemplificar esta situación considere la siguiente función $f(x) = x^2$, de la cual sabemos que tiene una raíz en $r = 0$ y además $f'(x) = 2x$, que al evaluarla en $r = 0$ obtenemos que $f'(r) = 0$. Es importante destacar que $f'(x) \neq 0$ para valores de x distintos a 0, por lo que podemos aplicar el Método de Newton sin problemas, pero

queremos ver que ocurre con el algoritmo. Entonces la iteración de Newton queda de la siguiente forma,

$$\begin{aligned} x_{i+1} &= x_i - \frac{f(x_i)}{f'(x_i)} \\ &= x_i - \frac{x_i^2}{2x_i} \\ &= \frac{x_i}{2}. \end{aligned}$$

Aplicando valor absoluto, recordando que $r = 0$ y manipulando algebraicamente la ecuación podemos obtener el cociente de los errores de la siguiente forma,

$$\begin{aligned} |x_{i+1} - r| &= \frac{|x_i - r|}{2} \\ e_{i+1} &= \frac{e_i}{2} \\ \frac{e_{i+1}}{e_i} &= \frac{1}{2}. \end{aligned}$$

Lo que confirma que la convergencia obtenida es lineal con tasa $S = \frac{1}{2}$, por lo tanto se pudo ejecutar el método de Newton pero la convergencia obtenida no fue cuadrática. En la Tabla 3.2 se muestra lo obtenido numéricamente al aplicar el método de Newton a $f(x) = x^2$.

Tabla 3.2: Ejemplo numérico sobre convergencia lineal del Método de Newton para la función $f(x) = x^2$.

i	x_i	e_i	$\frac{e_i}{e_{i-1}}$
0	1	1	-
1	$\frac{1}{2}$	0.5	0.5
2	0.25	0.25	0.5
3	0.125	0.125	0.5

Comentario al margen

El mismo resultado algebraico obtenido para $f(x) = x^2$ se obtiene para la función $f(x) = x^m$, para $m > 1$. En este caso $f'(x) = m x^{m-1}$, y al aplicar el método de Newton la iteración de punto fijo queda de la siguiente forma,

$$\begin{aligned} x_{i+1} &= x_i - \frac{x_i^m}{m x_i^{m-1}} \\ &= \frac{m-1}{m} x_i. \end{aligned}$$

Donde se obtiene el resultado general de convergencia lineal.

Del análisis anterior se obtiene el siguiente Teorema,

Thm 10 (Convergencia Lineal del Método de Newton). *Asuma que f es una función $(m+1)$ - veces continua y diferenciable en $[a, b]$ y tiene una multiplicidad m en la raíz r . Entonces el método de Newton es linealmente convergente a r con tasa S ,*

$$\lim_{i \rightarrow \infty} \frac{e_{i+1}}{e_i} = \frac{m-1}{m} = S \neq 0.$$

Del Teorema anterior podemos extraer una conclusión bien interesante, que es que a partir de su tasa de convergencia lineal podemos estimar la multiplicidad de la raíz! La utilidad de esto se observa en el siguiente Teorema, que actúa como antídoto del Teorema anterior,

Thm 11 (Método de Newton Modificado). *Si f es $(m + 1)$ - veces continua y diferenciable en $[a, b]$, donde hay una raíz r de multiplicidad $m > 1$, entonces el método de Newton modificado,*

$$x_{i+1} = x_i - m \frac{f(x_i)}{f'(x_i)}.$$

Converge local y cuadráticamente a r .

3.5. Método de la Secante

En la secciones anteriores estudiamos en profundidad las propiedades de convergencia local del algoritmo de iteración de punto fijo y el método de Newton. En particular, se demostró que el método de Newton puede converger cuadráticamente, sin embargo requiere el uso de la derivada, i.e. $f'(x)$, de la función a la cual se le busca una raíz. Lamentablemente en algunas ocasiones no se tiene acceso a la derivada. Esto en principio nos deja limitado a que solo podríamos utilizar el algoritmo de la bisección o alguna iteración de punto fijo que pudiéramos construir. Por lo cual surge la pregunta, ¿Existe alguna forma de adaptar el método de Newton para que no requiera la derivada exacta? La respuesta es sí, y aquí surge el método de la Secante. Para hacer efectiva esta modificación requerimos de la siguiente aproximación de diferencias finitas, en particular se utilizará backward difference¹¹,

$$\begin{aligned} f'(x) &= \frac{f(x) - f(x - \Delta x)}{x - (x - \Delta x)} + \mathcal{O}(\Delta x) \\ &= \frac{f(x) - f(x - \Delta x)}{\Delta x} + \mathcal{O}(\Delta x). \end{aligned}$$

Esta aproximación nos indica que la derivada de una función se puede obtener como el cociente de la diferencia entre 2 valores de la función $f(x)$ y su variación en x , i.e. Δx . Que es precisamente el origen de la definición de derivada! Entonces, partiendo del método de Newton podemos utilizar la aproximación de la derivada de la siguiente forma,

$$\begin{aligned} x_{i+1} &= x_i - \frac{f(x_i)}{f'(x_i)}, \text{ método de Newton,} \\ &\approx x_i - \frac{f(x_i)}{\frac{f(x_i) - f(x_i - \Delta x)}{\Delta x}}. \end{aligned}$$

Reemplazando $f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$, donde se consideró que: $x = x_i$ y $x - \Delta x = x_{i-1}$. Entonces se obtiene,

$$\begin{aligned} x_{i+1} &\approx x_i - \frac{f(x_i)}{\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}} \\ &\approx x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}. \end{aligned}$$

Lo cual induce el método de la Secante,

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}.$$

¹¹Se verá en más detalles en la Sección 9.2.2. También se puede obtener por medio de una expansión de Taylor! En particular de la función $f(x - \Delta x)$, considerando Δx pequeño.

Note que ahora se reemplazó el signo de aproximación \approx por la igualdad $=$, dado que se está definiendo un nuevo método, i.e. el método de la Secante. Es importante notar sin embargo que el método de la Secante requiere 2 initial guesses para iniciarse, x_0 y x_1 , a diferencias de una iteración de punto fijo o el método de Newton. Esto no es en general un problema dado que al considerar ya x_0 uno podría perturbarlo un poco y obtener un segundo initial guess, es decir podría definir $x_1 = x_0 + \delta$, para un valor de δ pequeño. El algoritmo queda de la siguiente forma,

```

1   $x_0 = \text{Initial Guess 1}$ 
2   $x_1 = \text{Initial Guess 2}$ 
3  for  $i$  in range( $n$ ):
4       $x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$ 

```

Para el método de la Secante, bajo el supuesto de que converge a r y que $f'(r) \neq 0$, se cumple la siguiente relación aproximada del error:

$$e_{i+1} \approx \left| \frac{f''(r)}{2f'(r)} \right| e_i e_{i-1}$$

$$, \approx \left| \frac{f''(r)}{2f'(r)} \right|^{\alpha-1} e_i^\alpha.$$

Donde $\alpha = \frac{1+\sqrt{5}}{2} \approx 1,62$. La convergencia de este método se denomina convergencia super-lineal, lo cual significa que está entre la convergencia de los métodos linealmente convergentes y los cuadráticamente convergentes. En particular se espera en este que el siguiente cociente se estabilice,

$$\frac{e_{i+1}}{e_i^\alpha} \approx L.$$

3.6. Criterios de detención de los algoritmos discutidos

Hasta el momento hemos analizado en extenso la teoría de convergencia local de diversos algoritmos, partiendo por el método de la bisección, luego estudiamos el algoritmo de iteración de punto fijo, el método de Newton y finalmente del algoritmo de la Secante. Lo cuales muestran comportamientos y requerimientos diversos. Ahora es el momento de analizar cuando detener la ejecución de los algoritmos descritos anteriormente.

La detención de un algoritmo puede realizarse, principalmente por 2 situaciones: (i) Se ejecuta una cantidad definida de iteraciones, (ii) se cumple algún criterio de error definido. Respecto al primero, (i), este caso se utiliza como un seguro para evitar que los algoritmos queden en un ciclo infinito. Lamentablemente no podemos fijar a priori un número muy pequeño en este caso, a menos que tengamos más información del contexto del problema. Por ejemplo como se estudió para el caso del método de la bisección¹².

Antes de entrar en más detalle respecto al punto (ii), debemos recordar que en todos los análisis anteriores consideramos la definición de error e_i como $|x_i - r|$. Esta definición es poco práctica dado que requiere la raíz r , por lo tanto no es útil asociarla a un criterio de detención. Sin embargo, como se presentó brevemente al final de la Sección 3.2.4, podemos aproximar este valor. Lo que nos indica la definición de error $e_i = |x_i - r|$, es que se necesita obtener una estimación del error en la iteración i , por lo cual se computa la diferencia entre la estimación en la i -ésima aproximación y r . En este caso, podemos reemplazar r por una aproximación que sea mejor que x_i , por lo cual un candidato sería x_{i+1} , o cualquiera de las posteriores estimaciones de r , considerando que se observa convergencia. Esto nos resuelve el problema de estimar e_i , por lo cual podemos definir, por lo menos, los siguientes criterios de detención:

1. Error absoluto: $|x_i - x_{i+1}| < \text{TOL}$.

¹²En el ejemplo de encontrar una raíz de $f(x) = \cos x - x$ con 6 decimales, en donde se determinó que se requerían 22 evaluaciones.

2. Error relativo: $\frac{|x_i - x_{i+1}|}{|x_{i+1}|} < \text{TOL}$.

3. Error relativo acotado: $\frac{|x_i - x_{i+1}|}{\max(|x_{i+1}|, \Theta)} < \text{TOL}$, por ejemplo $\Theta = 10^{-10}$.

El error absoluto, $|x_{i+1} - x_i| < \text{TOL}$, tiene la ventaja que ayudarnos a determinar cuántos decimales correctos se obtienen, sin embargo no funciona bien si es que la raíz que se anda buscando es muy cercana a 0. El error relativo, $\frac{|x_i - x_{i+1}|}{|x_{i+1}|} < \text{TOL}$, entrega una mejor interpretación del error sin embargo podría sufrir por división por 0 si la raíz que se busca es efectivamente 0. Por último, el error relativo acotado, $\frac{|x_i - x_{i+1}|}{\max(|x_{i+1}|, \Theta)} < \text{TOL}$, elimina la posibilidad de división por 0 pero agrega un nuevo parámetro, Θ , que requiere definirse. Nótese que el error relativo acotado se comporta exactamente igual que el error relativo mientras $|x_{i+1}| \geq \Theta$.

La elección de cual criterio de detención se debe utilizar dependerá del problema a estudiar y es aquí donde usted debe tomar una decisión.

Comentario al margen

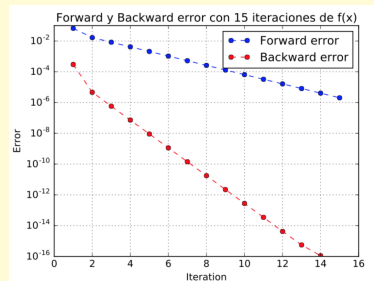
Un breve comentario para entender la relación entre Backward-error y Forward-error Asuma que f es una función y que r es una de sus raíces, i.e. $f(r) = 0$, y que x_a es una aproximación a r . Para el problema de encontrar una raíz, el backward-error de la aproximación x_a es $|f(x_a) - 0|$ y el forward error es $|x_a - r|$. La relación entre el backward-error y forward-error relación puede ser mejor ilustrada considerando 2 términos de la expansión de Taylor de $f(x)$ entorno a r ,

$$f(r) = f(x_a) + f'(c)(r - x_a).$$

Reordenando los términos y aplicando valor absoluto obtenemos,

$$\underbrace{|(f(r) - f(x_a))|}_{\text{Backward - error}} = |f'(c)| \underbrace{|r - x_a|}_{\text{Forward - error}}$$

Por ejemplo, si aplicamos el método de la bisección y graficamos el backward-error y forward-error en función del número de iteraciones, obtenemos lo siguiente,



Nos damos cuenta en el gráfico que el backward-error disminuye de forma mucho mas rápida que el forward-error, es decir, si el backward-error es pequeño, no implica que necesariamente el forward-error es pequeño!.

Lo que se quiere en búsqueda de ceros es que el forward-error sea pequeño pero para poder medirlo necesitamos conocer la solución^a por lo cual lo que se hace es obtener al backward-error y concluir sobre el forward-error.

Es **¡¡¡MUY IMPORTANTE!!!** que se entienda este punto, **backward-error pequeño no necesariamente implica forward-error pequeño**, solo es válido cuando el problema es “bien condicionado”.

Este comportamiento se puede generalizar si es que $f'(r) = 0$,

$$f(x_a) = f(r) + f'(r)(r - x_a) + \dots + \frac{f^{(m+1)}(c)}{(m+1)!} (r - x_a)^{m+1}.$$

Reordenando los términos y aplicando valor absoluto obtenemos,

$$|f(r) - f(x_a)| = \left| \frac{f^{(m+1)}(c)}{(m+1)!} \right| |r - x_a|^m.$$

^a;que es precisamente lo que andamos buscando!

3.7. Sensibilidad en la búsqueda de raíces

Para entender la sensibilidad en la búsqueda de raíces consideremos el polinomio que propuso Wilkinson en 1994,

$$W(x) = \prod_{i=1}^{20} (x - i),$$

del cual conocemos exactamente sus raíces, las cuales son $r_i = i$ para $i \in \{1, 2, \dots, 20\}$. Sin embargo cuando expandimos el polinomio obtenemos,

$$W(x) = x^{20} - 210x^{19} + \dots + 1206647803780373360x^6 + \dots + 243 \dots 000.$$

Ahora, si buscamos una raíz de ese polinomio expandido con double precision utilizando como initial guess $x_0 = 16$ obtenemos el siguiente resultado,

$$x_a = 16,002294682053055.$$

Sin embargo sabemos que la solución exacta es “16”!! Lo que nos da el siguiente error absoluto,

$$|x - x_a| \approx 10^{-3}.$$

¿Qué ocurrió entonces? Para responder este pregunta debemos primero analizar lo que efectivamente está ocurriendo. Este problema se puede definir de la siguiente forma: “Encontrar r , dado $f(r) = 0$, para pequeñas perturbaciones de los coeficientes del polinomio de la forma $\epsilon g(x)$, $\epsilon \ll 1$, agregadas a la función original”. Esto se puede escribir matemáticamente de la siguiente forma,

$$f(r + \Delta r) + \epsilon g(r + \Delta r) = 0.$$

Donde notamos que al agregar $\epsilon g(x)$ claramente perturbamos la raíz original, por lo cual necesitamos agregar Δr para cuantificar ese efecto. Nuevamente acá necesitamos pedirle ayuda a la útil expansión de Taylor, por lo cual obtenemos,

$$f(r) + \Delta r f'(r) + \epsilon g(r) + \epsilon \Delta r g'(r) + \mathcal{O}(\Delta r^2) = 0.$$

Considerando $g'(r) \ll f'(r)$ y omitiendo términos de orden superior, podemos despejar Δr ,

$$\Delta r \approx \frac{-\epsilon g(r)}{f'(r)}.$$

Ahora, volviendo al problema inicial de Wilkinson, podemos cuantificar el efecto sobre Δr . En este caso particular consideremos que $g(x)$ es efectivamente el monomio $c_{15}x^{15}$, es decir, perturbaremos un poquito la función $W(x)$. Por completitud explicitaremos el coeficiente c_{15} del monomio x^{15} ,

$$W(x) = \prod_{i=1}^{20} (x - i) = x^{20} + \dots - 1672280820x^{15} + \dots.$$

Entonces la función perturbada corresponde a,

$$W_\epsilon(x) = W(x) + \epsilon g(x).$$

Calculando Δr obtenemos,

$$\begin{aligned} \Delta r &\approx \frac{-\epsilon g(r)}{f'(r)} \\ &\approx \frac{16^{15} 1672280820}{15!4!} \epsilon. \end{aligned}$$

Considerando que $\epsilon = \epsilon_{\text{mach}}$ en double precision obtenemos,

$$\begin{aligned}\Delta r &\approx \frac{16^{15} 1672280820}{15!4!} \epsilon_{\text{mach}} \\ &\approx 6,15 \cdot 10^{13} \cdot 2,22 \cdot 10^{-16} \\ &\approx 0,0136.\end{aligned}$$

Lo cual es consistente con el resultado anterior, donde el error obtenido fue aproximadamente 10^{-3} . Por lo tanto, en double precision solo podemos obtener la 16-ava raíz de $W(x)$ a los más con 1 o 2 decimales de exactitud.

Un concepto que aparece de forma involuntariamente oculta en el análisis anterior corresponde al “condicionamiento”, en particular el número de condición, denominado tradicionalmente como κ . El condicionamiento está relacionado al problema matemático que se está resolviendo y determina si un problema es bien o mal condicionado. Lamentablemente no hay una definición precisa en este aspecto, sin embargo podemos indicar que un problema bien condicionado tiene un número de condición “pequeño”, digamos κ puede variar entre 1 y 10^4 , y mal condicionado para valores de κ sobre 10^{10} . En el ejemplo anterior el número de condición obtenido es,

$$\kappa = \frac{16^{15} 1672280820}{15!4!} = 6,15 \cdot 10^{13}.$$

Por lo tanto el problema es mal condicionado, y la rule-of-thumb indica que para un problema mal condicionado se pierden la siguiente cantidad de decimales en la computación,

$$\text{Número de decimales que se pierden} \approx \log_{10}(\kappa).$$

Lo cual es consistente con lo obtenido anteriormente.

Comentario al margen

Otro concepto matemático relevante es si un problema está bien propuesto (well-posed) en el sentido de Hadamard. El cual debe cumplir lo siguiente para ser denominado bien propuesto,

1. Existe una solución.
2. La solución es única.
3. El comportamiento de la solución cambia continuamente en función de los datos o condiciones iniciales entregadas.

La utilidad de que un problema sea bien propuesto es que entrega el conocimiento de que tiene sentido buscar la solución numéricamente dado que sabes que existe! Si no sabemos si una solución existe, estamos en la incertidumbre de buscar una aguja en un pajar donde no hay ninguna aguja! Es importante señalar que si un problema es bien propuesto, no es el fin de la historia. Dado que el problema aún puede ser bien o mal condicionado!

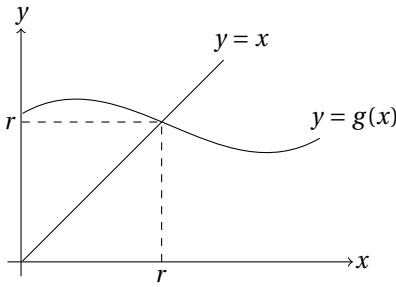
Un ejemplo de problema mal propuesta sería buscar una raíz de una ecuación cuadrática, en ese caso sabemos que tiene 2 soluciones, sin embargo podría convertirse en un problema bien propuesto si acotamos el intervalo donde solo exista una única solución. En resumen, debemos estar atento a las 3 condiciones definidas por Hadamard antes de enfrentarse a un problema numéricamente.

Capítulo 4

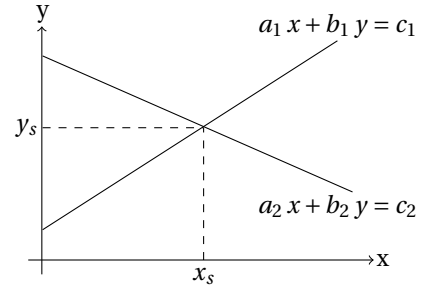
Sistemas de Ecuaciones Lineales

En este capítulo estudiaremos en profundidad algunos algoritmos clásicos para la resolución de sistemas de ecuaciones lineales.

En el capítulo anterior estudiamos en detalle la búsqueda de raíces en 1D, la cual podía ser interpretada, desde el punto de vista de una iteración de punto fijo, como la intersección de una recta y una función no-lineal, ver Figura 4.1a. Ahora, podemos decir, que estudiaremos una variante más simple, es decir, encontrar la intersección de 2 rectas, ver Figura 4.1b. Sin embargo, esta aparente simplificación del problema abre una secuencia interesante de desafíos que iremos estudiando. En particular, los temas en los que pondremos atención incluyen: calidad de la solución (i.e. error), tiempo de computación (i.e. cuanto se demora) y almacenamiento requerido (i.e. memoria RAM).



(a) Iteración de punto fijo



(b) Sistema de ecuaciones lineales de 2×2 .

Figura 4.1: Comparación entre una iteración de punto fijo y un sistema de ecuaciones lineales.

Por ejemplo, en la Figura 4.1b, se propone la búsqueda del punto donde intersectan 2 rectas, el cual se denota como (x_s, y_s) . Este sistema lo podemos escribir de la siguiente forma:

$$a_1 x + b_1 y = c_1, \quad (4.1)$$

$$a_2 x + b_2 y = c_2. \quad (4.2)$$

Aún más, podemos escribir el sistema de ecuaciones lineales anterior en su forma vectorial:

$$\underbrace{\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}}_A \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}}_{\mathbf{c}}.$$

Es decir obtenemos $A\mathbf{x} = \mathbf{c}$, el cual es un sistema de ecuaciones lineales de 2 ecuaciones y 2 incógnitas. La solución matemática es simplemente $\mathbf{x} = A^{-1}\mathbf{c}$. Sin embargo en este capítulo estudiaremos distintas alternativas de obtener \mathbf{x} sin recurrir a construir A^{-1} !

4.1. Métodos conocidos de resolución

4.1.1. Por sustitución

A partir de la ecuación (4.1) despejamos x y obtenemos:

$$x = \frac{c_1 - b_1 y}{a_1}. \quad (4.3)$$

Reemplazando la ecuación (4.3) en la ecuación (4.2) se obtiene,

$$a_2 \left(\frac{c_1 - b_1 y}{a_1} \right) + b_2 y = c_2.$$

La cual es 1 ecuación y 1 incógnita, por lo tanto fácilmente se obtiene y ,

$$y = \frac{c_2 - a_2 \frac{c_1}{a_1}}{b_2 - a_2 \frac{b_1}{a_1}} = \frac{a_1 c_2 - a_2 c_1}{a_1 b_2 - a_2 b_1}. \quad (4.4)$$

Para obtener x , solo debemos reemplazar el valor de y obtenido en la ecuación (4.4) en la ecuación (4.3) y obtenemos,

$$x = \frac{c_1 b_2 - b_1 c_2}{a_1 b_2 - a_2 b_1}. \quad (4.5)$$

4.1.2. Usando Regla de Cramer

Para nuestro sistema de ecuaciones lineales de 2×2 obtenemos los valores de x e y calculando el cociente de los siguientes determinantes,

$$x = \frac{\begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}},$$

y,

$$y = \frac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}}$$

Para más detalles sobre la regla de Cramer, se sugiere ver el siguiente [link](https://en.wikipedia.org/wiki/Cramer%27s_rule)¹.

¹https://en.wikipedia.org/wiki/Cramer%27s_rule

4.1.3. ¿Qué aprendimos del método por sustitución y de la regla de Cramer?

- Del método por sustitución: Simple de explicar pero quizás un poco complicado de implementar.
- Muy simple de explicar pero MUY costoso computacionalmente. ¿Cuál es el costo computacional de obtener un determinante de una matriz de $n \times n$?²

Entonces, ¿cómo deberíamos proceder para resolver un sistema de ecuaciones lineales? Antes de ver el caso general, analicemos un caso más simple. Por ejemplo, el método por sustitución es engorroso porque la estructura de la matriz es densa³, pero ¿Qué ocurriría si la matriz tuviera un patrón más adecuado pero no trivial? Por ejemplo un patrón trivial⁴ sería el caso de una matriz diagonal. En ese caso, la solución se puede obtener directamente dividiendo los coeficientes del lado derecho por los elementos de la diagonal de la matriz. Entonces, ¿Cuál sería un patrón adecuado? En realidad la respuesta tradicional es *depende*, sin embargo aquí podemos decir que un patrón adecuado es cuando la matriz es triangular superior (o inferior!)⁵. Por ejemplo, considere el siguiente sistema de ecuaciones lineales,

$$ax + by = d, \quad (4.6)$$

$$0x + cy = e. \quad (4.7)$$

En este caso, la solución no es trivial, pero podemos obtenerla rápidamente. Lo primero que se puede hacer es obtener y desde la ecuación 4.7 de la siguiente forma,

$$y = \frac{e}{c}.$$

Ahora que conocemos y , podemos reemplazarla en la ecuación 4.6 y podemos obtener x ,

$$x = \frac{d - by}{a}.$$

Por lo tanto, vemos que es muy conveniente resolver sistemas de ecuaciones lineales donde la matriz es triangular superior. Entonces surge la siguiente pregunta que quizás se hizo Gauss y muchas otras personas: ¿Podremos transformar todos los sistemas de ecuaciones lineales a un sistema de ecuaciones lineales **equivalente** donde la matriz sea triangular superior?⁶

4.2. Eliminación Gaussiana Simple y su complejidad computacional

Consideremos nuevamente un sistema de ecuaciones lineales con 2 ecuaciones y 2 incógnitas:

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}. \quad (4.8)$$

El cual podemos re-escribir en el siguiente tableau, donde se acopla los coeficientes de la matriz asociada y el lado derecho del sistema de ecuaciones lineales:

$$\left[\begin{array}{cc|c} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{array} \right].$$

²En el peor caso $\mathcal{O}(n!)$ (n factorial!), por lo tanto hay que evitarlo a toda costa!

³Es decir, todos los coeficientes son no nulos.

⁴Trivial no significa necesariamente irrelevante en este caso, solo se está usando para ejemplificar un caso donde uno puede describir la solución explícita directamente.

⁵En la sección 1.4.3.2 presentamos este tipo de matrices inicialmente. Ahora veremos su importancia!

⁶Esto lo responderemos en la siguiente sección.

Este tableau nos permite ahora realizar una operación fila que hará el coeficiente en la segunda fila y primera columna igual a 0. La operación fila es la siguiente:

$$R_2 = R_2 - \frac{a_2}{a_1} R_1,$$

la cual indica que debemos restarle a la fila 2 la fila 1 multiplicada por el $\frac{a_2}{a_1}$. Por lo que obtenemos:

$$\left[\begin{array}{cc|c} a_1 & b_1 & c_1 \\ 0 & \left(b_2 - \frac{a_2}{a_1} b_1\right) & c_2 - \frac{a_2}{a_1} c_1 \end{array} \right].$$

De este tableau modificado podemos escribir un sistema de ecuaciones lineales equivalente que tiene la misma solución que el sistema de ecuaciones lineales original indicado en la ecuación (4.8). El sistema de ecuaciones lineales queda de la siguiente forma:

$$\begin{aligned} a_1 x + b_1 y &= c_1, \\ 0 x + \left(b_2 - \frac{a_2}{a_1} b_1\right) y &= c_2 - \frac{a_2}{a_1} c_1. \end{aligned}$$

Notar que incluimos $0x$ solo por completitud, pero en realidad la segunda ecuación tiene solo 1 incógnita ahora, la cual es y . El valor de y entonces lo podemos obtener directamente:

$$y = \frac{c_2 - \frac{a_2}{a_1} c_1}{b_2 - \frac{a_2}{a_1} b_1} = \frac{a_1 c_2 - a_2 c_1}{a_1 b_2 - a_2 b_1}.$$

Que es exactamente lo mismo que obtuvimos antes, por ejemplo en la ecuación (4.4). Ahora, con la primera ecuación podemos obtener x reemplazando el valor de y encontrado.

$$x = \frac{c_1 - b_1 y}{a_1} = \frac{c_1 b_2 - b_1 c_2}{a_1 b_2 - a_2 b_1}.$$

Este procedimiento induce el siguiente algoritmo:

1. Construir tableau a partir del sistema de ecuaciones lineales $A\mathbf{x} = \mathbf{b}$.
2. Aplicar operaciones fila para modificar el sistema de ecuaciones lineales original y obtener un sistema de ecuaciones lineales equivalente pero con la matriz asociada que sea triangular superior y un lado derecho modificado acorde: $U\mathbf{x} = \hat{\mathbf{c}}$.
3. Resolver el sistema de ecuaciones equivalente $U\mathbf{x} = \hat{\mathbf{c}}$ con backward substitution.

En el caso general el tableau tiene la siguiente forma:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & & a_{2n} & b_2 \\ \vdots & & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array} \right] = \left[\begin{array}{c|c} A & \mathbf{b} \end{array} \right]$$

Lo cual se traduce en el siguiente algoritmo para convertir el tableau en su forma triangular superior:

```

1  for j in range(n):
2      for i in range(j+1, n):
3          mult = Bij / Bjj
4          Bi,j+1:n+1 = Bi,j+1:n+1 - mult · Bj,j+1:n+1

```

donde la matriz B es donde se almacena el tableau. La inquietud que surge ahora está asociada al tiempo de computación requerido para obtener el tableau en su forma triangular superior. Es decir, ¿Cuántas operaciones elementales se necesitan en G.E.?

Comentario al margen

Antes de proseguir, recordaremos las siguientes identidades:

$$1 + 1 + 1 + \cdots + 1 = \sum_{i=1}^n 1 = n,$$

$$1 + 2 + 3 + \cdots + n = \sum_{i=1}^n i = \frac{n(n+1)}{2},$$

$$1^2 + 2^2 + 3^2 + \cdots + n^2 = \sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}.$$

Estas identidades nos serán muy útiles al contar el número de operaciones elementales de los algoritmos!

Para responder la pregunta anterior, y para un mejor análisis, se repetirá el algoritmo de eliminación Gaussiana antes incluido:

```

1  for j in range(n-1):
2      for i in range(j+1,n):
3          mult = B[i,j]/B[j,j]
4          B[i,j+1:] = B[i,j+1:] - mult*B[j,j+1:]

```

Entonces, para contar las operaciones elementales, debemos analizar las líneas del código donde se realizan operaciones elementales y las contamos, ver Tabla 4.1. Ahora que contamos las operaciones elementales en las

Tabla 4.1: Número de operaciones elementales en la eliminación Gaussiana

línea	# operaciones	operación
(3)	1	División
(4)	2 ("n + 1 - j"-veces)	Multiplicación escalar por vector y resta

líneas (3) y (4), debemos contabilizar las veces que se repiten estas en función de los ciclos incluidos, por lo tanto

se obtiene lo siguiente,

$$\begin{aligned}
 \# \text{ Total de Operaciones Elementales} &= \sum_{j=1}^{n-1} \underbrace{\sum_{i=j+1}^n}_{(2)} \left(\underbrace{1}_{(3)} + \underbrace{2(n-j+1)}_{(4)} \right) \\
 &= \sum_{j=1}^{n-1} \sum_{k=1}^{n-j} (1 + 2(n-j+1)) \\
 &= \sum_{j=1}^{n-1} (1 + 2(n-j+1))(n-j) \\
 &= \sum_{j=1}^{n-1} (3n - 3j + 2n^2 - 4nj + 2j^2) \\
 &= (3n + 2n^2) \left(\sum_{j=1}^{n-1} 1 \right) - 3 \left(\sum_{j=1}^{n-1} j \right) - 4n \left(\sum_{j=1}^{n-1} j \right) + 2 \sum_{j=1}^{n-1} j^2 \\
 &= (3n + 2n^2)(n-1) - 3 \frac{(n-1)n}{2} - 4n \frac{(n-1)n}{2} + 2 \frac{(n-1)n(2n-1)}{6} \\
 &= \frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n \\
 &\sim \frac{2}{3}n^3
 \end{aligned}$$

Por lo tanto la eliminación Gaussiana de un sistema de ecuaciones lineales de $n \times n$ requiere la ejecución de aproximadamente $\sim \frac{2}{3}n^3$ operaciones elementales. Para entender este resultado se proponen las siguientes preguntas:

- ¿Cuál es el tamaño del sistema de ecuaciones lineales que se puede resolver en su computador personal en un segundo?
- ¿Cuál es el tamaño del sistema de ecuaciones lineales que se puede resolver en su computador personal en una hora?
- ¿Cuál es el tamaño del sistema de ecuaciones lineales que se puede resolver en su computador personal en un día?
- ¿Cuál es el tamaño del sistema de ecuaciones lineales que se puede resolver en su computador personal en un año?⁷
- Si es que para algunos de los tamaños de los sistemas de ecuaciones lineales obtenido no le hubiera alcanzado en la RAM disponible, lo natural es que el sistema operativo utilice la memoria swap (o similar). ¿Cómo afecta esto en los tiempos de computo?

4.3. Backward Substitution

En la descripción de la eliminación Gaussiana y en la sección 1.4.3.2 se mencionó la importancia del algoritmo de backward substitution. En esta sección lo estudiaremos con mayor detalle. El algoritmo de backward substitution es un algoritmo diseñado específicamente para encontrar la solución a sistemas de ecuaciones lineales de la forma $U\mathbf{x} = \mathbf{b}$, donde U es una matriz triangular superior y se usa \mathbf{b} para denotar el vector del lado derecho.

⁷¿Alcanzará en la memoria RAM disponible?

El tilde incluido en $\tilde{\mathbf{b}}$ es para dar énfasis en que no es el mismo \mathbf{b} del sistema de ecuaciones lineales original $A\mathbf{x} = \mathbf{b}$ indicado en la sección anterior. La estructura explícita del sistema es la siguiente:

$$\begin{bmatrix} u_{11} & u_{12} & \dots & \dots & u_{1n} \\ 0 & u_{22} & \dots & \dots & u_{2n} \\ \vdots & 0 & \ddots & & \vdots \\ \vdots & & \ddots & u_{n-1,n-1} & u_{n-1,n} \\ 0 & \dots & & 0 & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_{n-1} \\ \tilde{b}_n \end{bmatrix}$$

El patrón regular de la matriz, permite deducir rápidamente el algoritmo asociado. Por ejemplo, en la última fila de la matriz solo existe 1 término no-nulo, lo que indica que podemos escribir explícitamente la última ecuación, la cual queda de la siguiente forma:

$$u_{n,n} x_n = \tilde{b}_n,$$

de la cual se puede obtener explícitamente el valor de x_n , el cual es,

$$x_n = \frac{\tilde{b}_n}{u_{n,n}}.$$

Ahora, podemos escribir de forma explícita la $(n-1)$ -ésima ecuación,

$$u_{n-1,n-1} x_{n-1} + u_{n-1,n} x_n = \tilde{b}_{n-1}.$$

Nuevamente en este caso solo tenemos una incógnita, i.e. x_{n-1} , dado que x_n lo obtuvimos con la ecuación anterior. Por lo tanto podemos despejar x_{n-1} explícitamente también,

$$x_{n-1} = \frac{\tilde{b}_{n-1} - u_{n-1,n} x_n}{u_{n-1,n-1}}.$$

Notar que uno podría reemplazar x_n en la ecuación anterior por $\frac{\tilde{b}_n}{u_{n,n}}$, sin embargo no es necesario, ya que ya se almacenó en x_n el cociente $\frac{\tilde{b}_n}{u_{n,n}}$. Ahora, siguiendo el mismo procedimiento, podemos escribir explícitamente la i -ésima ecuación,

$$u_{i,i} x_i + \sum_{j=i+1}^n u_{i,j} x_j = \tilde{b}_i.$$

En este punto, ya conocemos los valores de x_j para $j \in \{n, n-1, \dots, i+1\}$. Por lo tanto solo nos queda despejar x_i ,

$$x_i = \frac{\tilde{b}_i - \sum_{j=i+1}^n u_{i,j} x_j}{u_{i,i}}.$$

Repetimos este procedimiento hasta i igual a 1. Recuerde que el proceso se ejecuta desde la última ecuación hasta la primera ecuación. Este algoritmo se puede formalizar en la siguiente implementación:

4.3.1. Número de operaciones elementales

El siguiente paso natural es obtener el número de operaciones elementales para el algoritmo de Backward Substitution definido en el Algoritmo 1. De la implementación propuesta, podemos concluir que se realizan 2 operaciones elementales en la línea 3 del algoritmo y una operación elemental en la línea 4, por lo tanto podemos

```

1 for i in range(n, 1, -1):
2     for j in range(i+1, n):
3          $b_i = b_i - u_{i,j} \cdot x_j$ 
4      $x_i = \frac{b_i}{u_{i,i}}$ 

```

Algoritmo 1: Backward Substitution

escribir la cantidad total de operaciones elementales de la siguiente forma, solo recordar que las sumatorias toman en consideración la repetición de las operaciones en los ciclos incluidos. Entonces,

$$\begin{aligned}
 \sum_{i=n}^1 \left[\left(\sum_{j=i+1}^n 2 \right) + 1 \right] &= \sum_{i=1}^n \left[2 \left(\sum_{j=i+1}^n 1 \right) + 1 \right] \\
 &= \sum_{i=1}^n [2(n - i + 1 - 1) + 1] \\
 &= 2n \left(\sum_{i=1}^n 1 \right) - 2 \left(\sum_{i=1}^n i \right) + \sum_{i=1}^n 1 \\
 &= 2nn - 2 \frac{n(n+1)}{2} + n \\
 &= 2n^2 - n^2 - n + n \\
 &= n^2.
 \end{aligned}$$

Donde $\sum_{i=n}^1$ denota la sumatoria de términos en orden inverso, es decir que suma desde $i = n$ hasta $i = 1$ restando de 1 en 1. Se propone esta notación para que la analogía entre la implementación y el conteo del números operaciones elementales sea directa, pero no se vaya a confundir, es solo una sumatoria! En resumen, resolver un sistema de ecuaciones lineales donde la matriz es triangular superior solo requiere n^2 operaciones! Lo cual es muy bueno! ¿Es posible obtener un algoritmo análogo pero para sistema de ecuaciones lineales cuando la matriz es triangular inferior?⁸

4.4. Factorización LU de A

En la sección anterior se estudió la eliminación Gaussiana para resolver un sistema de ecuaciones lineales $A\mathbf{x} = \mathbf{b}$, el cual se transformaba a un sistema análogo de la forma $U\mathbf{x} = \tilde{\mathbf{b}}$. La ventaja de este algoritmo es que podemos obtener \mathbf{x} realizando $\sim \frac{2}{3}n^3$ operaciones elementales, no se considera en general el costo de aplicar Backward Substitution dado que es solo n^2 , lo cual es un orden menor que el costo más significativo. La desventaja es que si quisiéramos resolver otro sistema de ecuaciones lineales donde solo cambia el lado derecho, es decir $A\mathbf{x} = \mathbf{d}$, tenemos que construir nuevamente el tableau y aplicar las operaciones filas correspondiente. Naturalmente obtendremos la misma matriz U , es decir el sistema de ecuaciones lineales equivalente será $U\mathbf{x} = \tilde{\mathbf{d}}$. Entonces surge la pregunta, ¿existirá una algoritmo en que no se necesite ejecutar la eliminación Gaussiana nuevamente para resolver un sistema de ecuaciones lineales donde solo cambia el lado derecho? La respuesta es sí, y corresponde a la factorización LU de A! La idea es simple y muy efectiva, esta corresponde a escribir la matriz A como el producto de las matrices L y U , es decir $A = LU$, donde L es una matriz triangular inferior y U es una matriz triangular superior. En realidad puede parecer como una solicitud muy exigente imponer una estructura tan rígida sobre la matriz L ,

⁸Sí!, se llama Forward Substitution y sigue la misma filosofía de Backward Substitution pero se inicia desde la primera ecuación hasta la última.

pero lo interesante es que sí se puede construir! La matriz U indicada es la misma que ya obtuvimos, por lo tanto ya estamos a medio camino. Ahora solo nos queda determinar L , que en realidad está “escondido” en la misma eliminación Gaussiana, solo que no lo habíamos notado!

Antes de encontrar L , procederemos a definir explícitamente que es una matriz triangular inferior y triangular superior:

- Una matriz $L \in \mathbb{R}^{m \times n}$ es **triangular inferior** si sus coeficientes satisfacen $l_{ij} = 0$ para $i < j$.
- Una matriz $U \in \mathbb{R}^{m \times n}$ es **triangular superior** si sus coeficientes satisfacen $u_{ij} = 0$ para $i > j$.

Ahora, para construir L nos basaremos en las siguientes aseveraciones,

1. Sea $L_{ij}(-c)$ una matriz triangular inferior la cual tiene coeficientes no-cero en su diagonal y en la posición (i, j) . En particular la diagonal son puros 1's y en la posición (i, j) está el valor “ $-c$ ”. Entonces $L_{ij}(-c) A$ representa la operación fila “restar la fila j multiplicada por c a la fila i ”.

Ejemplo: $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, $L_{21}(-c) \cdot A = \begin{bmatrix} 1 & 0 \\ -c & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} - c a_{11} & a_{22} - c a_{12} \end{bmatrix}$

2. $L_{ij}^{-1}(-c) = L_{ij}(c)$, es decir, la matriz inversa de $L_{ij}(-c)$ es $L_{ij}(c)$. Esto significa que $L_{ij}(-c) L_{ij}(c) = I$, donde I es la matriz identidad.

Ejemplo: $\begin{bmatrix} 1 & 0 \\ -c & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ -c & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -c & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ c - c & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

3. La siguiente ecuación matricial es cierta:

$$\begin{bmatrix} 1 & 0 & 0 \\ c_1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ c_2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & c_3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ c_1 & 1 & 0 \\ c_2 & c_3 & 1 \end{bmatrix}$$

Las 3 aseveraciones anteriores son cruciales para poder obtener la matriz L de la factorización LU de A . A modo de ejemplo, considere la siguiente matriz A ,

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

Dado que A es una matriz de 3×3 sabemos que solo necesitamos hacer 3 operaciones fila para convertirla en una matriz triangular superior, en este caso, así obtendríamos U . Considerando la aseveración 1, que indica que podemos escribir la operaciones filas como una transformación lineal, es decir, como el producto por una matriz, podemos construir U de la siguiente forma:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{7}{3} & 1 \end{bmatrix}}_{L_{3,2}(7/3)} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix}}_{L_{3,1}(3)} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{L_{2,1}(-2)} \underbrace{\begin{bmatrix} 1 & 2 & -1 \\ 2 & 1 & -2 \\ -3 & 1 & 1 \end{bmatrix}}_A = \underbrace{\begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 0 \\ 0 & 0 & -2 \end{bmatrix}}_U.$$

Claramente se observa que la matriz $L_{2,1}(-2)$ representa la operación fila adecuada para eliminar el coeficiente $A_{2,1} = 2$. Lo mismo ocurre con $L_{3,1}(3)$. En este caso el coeficiente de la operación fila es positivo porque el coeficiente $A_{3,1} = -3$ es negativo. La última operación fila requiere el coeficiente $\frac{7}{3}$ porque con la segunda operación fila ejecutada se modificó el coeficiente $A_{3,2}$. Es importante destacar aquí que las operaciones filas ahora no se están realizando sobre el tableau indicado anteriormente, por lo tanto no estamos incluyendo el lado derecho del sistema de ecuaciones lineales asociado. En resumen, tenemos la siguiente identidad:

$$L_{3,2}(7/3) L_{3,1}(3) L_{2,1}(-2) A = U.$$

Despejando la matriz A , obtenemos:

$$\begin{aligned} L_{3,2} (7/3) L_{3,1} (3) L_{2,1} (-2) A &= U, \\ L_{3,1} (3) L_{2,1} (-2) A &= L_{3,2}^{-1} (7/3) U, \\ L_{2,1} (-2) A &= L_{3,1}^{-1} (3) L_{3,2}^{-1} (7/3) U, \\ A &= L_{2,1}^{-1} (-2) L_{3,1}^{-1} (3) L_{3,2}^{-1} (7/3) U. \end{aligned}$$

Notar que al despejar A el orden de las matrices $L_{i,j}$'s se invierte con respecto al orden que tenía al lado izquierdo del signo igual. Ahora, si consideramos la aseveración 2, que indica que la inversa de una matriz que representa una operación fila es $L_{ij}^{-1}(-c) = L_{ij}(c)$. Por lo tanto podemos simplificar aún más el despeje⁹ de A realizado anteriormente,

$$\begin{aligned} A &= L_{2,1}^{-1} (-2) L_{3,1}^{-1} (3) L_{3,2}^{-1} (7/3) U \\ &= L_{2,1} (2) L_{3,1} (-3) L_{3,2} (-7/3) U. \end{aligned}$$

Finalmente, utilizando la aseveración 3, obtenemos:

$$\begin{aligned} A &= L_{2,1} (2) L_{3,1} (-3) L_{3,2} (-7/3) U \\ &= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{L_{2,1} (2)} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}}_{L_{3,1} (-3)} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -7/3 & 1 \end{bmatrix}}_{L_{3,2} (-7/3)} \underbrace{\begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 0 \\ 0 & 0 & -2 \end{bmatrix}}_U \\ &= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -7/3 & 1 \end{bmatrix}}_L \underbrace{\begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 0 \\ 0 & 0 & -2 \end{bmatrix}}_U \end{aligned}$$

Por lo tanto fuimos capaces de obtener L ! En resumen, L almacena los coeficientes necesarios para hacer las operaciones filas en la eliminación Gaussiana. Entonces lo único requerido para obtener la factorización LU de la matriz A es almacenar convenientemente los coeficientes utilizados en las operaciones filas respectivas.

Entonces ahora surge la pregunta, ¿cómo se utiliza la factorización LU de A para resolver $A\mathbf{x} = \mathbf{b}$? Esta pregunta la responderemos en la siguientes sección.

4.4.1. Utilización de la factorización LU

En la sección anterior discutimos como obtener la factorización LU de una matriz A . Esta es en realidad solo la primera factorización matricial que veremos, pero hay muchas más! La ventaja crucial de esta factorización es que obtenemos 2 matrices con una estructura muy particular, es decir una estructura triangular. Entonces, ¿podemos tomar ventaja de esa estructura? La respuesta es sí! Por ejemplo, considere el siguiente sistema de ecuaciones lineales:

$$A\mathbf{x} = \mathbf{b}.$$

⁹Notar que esta manipulación algebraica de las matrices $L_{ij}(c)$ es válida por la estructura que presenta, es decir, no es válido en general para matrices triangulares inferiores que no cumplan con ese patrón. ¿Es posible manipular algebraicamente matrices triangulares inferiores generales para llevarlas a la forma que permita invertirlas fácilmente? ¿Es válida la aseveración 3 si se cambia el orden de las matrices? ¿Se puede adaptar la aseveración 2 para matrices triangulares superiores?

Suponga ahora que obtuvimos la factorización LU de A , es decir tenemos la siguiente identidad $A = LU$. Reemplazando en el sistema de ecuaciones lineales anterior obtenemos,

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ LU\mathbf{x} &= \mathbf{b}. \end{aligned}$$

Ahora viene el primer paso clave, considere el vector incógnita $\mathbf{c} = U\mathbf{x}$. Note que se puede hacer esto dado que no conocemos \mathbf{x} , por lo tanto al multiplicar una matriz conocida, es decir U , por un vector desconocido, el resultado sigue siendo desconocido. Realizando este reemplazo obtenemos,

$$\begin{aligned} L \underbrace{U\mathbf{x}}_{\mathbf{c}} &= \mathbf{b} \\ L\mathbf{c} &= \mathbf{b}. \end{aligned}$$

¿Qué podemos hacer con $L\mathbf{c} = \mathbf{b}$? Podemos resolverlo! Para ser más explícito, podemos escribir el sistema de ecuaciones anterior en su forma vectorial,

$$\begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ l_{2,1} & 1 & \ddots & \ddots & 0 \\ l_{3,1} & l_{3,2} & 1 & \ddots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ l_{n,1} & \cdots & \cdots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}.$$

De la estructura anterior podemos concluir que podemos obtener los coeficientes del vector \mathbf{c} aplicando Forward Substitution, es decir, obtener c_1 primero, luego c_2 , y así sucesivamente. La cantidad de operaciones elementales requeridas son las mismas que para Backward Substitution, es decir n^2 . ¿Qué hemos logrado? Hemos sido capaces de resolver el primer paso, es decir, \mathbf{c} ya no es un vector desconocido, ahora conocemos sus coeficientes! ¿Qué nos falta entonces? Recordemos que anteriormente hicimos el cambio de variable $\mathbf{c} = U\mathbf{x}$, donde se argumentó que como no conocíamos \mathbf{x} , tampoco conocíamos \mathbf{c} . Sin embargo, ahora sí conocemos \mathbf{c} . Esto implica que en la ecuación $\mathbf{c} = U\mathbf{x}$ solo tenemos 1 vector incógnita, es decir el vector \mathbf{x} , re-escribiendolo como un sistema de ecuaciones obtenemos,

$$U\mathbf{x} = \mathbf{c}.$$

Nuevamente, escribiremos matricialmente el sistema de ecuaciones lineales obtenido,

$$\begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ 0 & u_{2,2} & \cdots & \\ 0 & 0 & \ddots & \vdots \\ \vdots & & & \\ 0 & 0 & \cdots & u_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}.$$

En este caso, solo re-destacar que el lado derecho, es decir el vector \mathbf{c} es conocido, por lo tanto podemos aplicar directamente el algoritmo de Backward Substitution y obtener nuestro anhelado \mathbf{x} ¹⁰.

En resumen, el procedimiento inducido de la explicación se puede expresar en el siguiente Algoritmo 2. Ahora



¹⁰Esto nos alegra mucho!

. Este osito feliz se incluyó hace algunos años para expresar la alegría de haber obtenido \mathbf{x} !

```

1 L, U = computeLU(A)
2 c = ForwardSubstitution(L, b)
3 x = BackwardSubstitution(U, c)
4 return x

```

Algoritmo 2: Resolución de un sistema de ecuaciones lineales con la factorización $A=LU$.

que entendemos la factorización LU destacaremos algunas interpretaciones interesantes. Por ejemplo, al reemplazar la factorización LU en el sistema de ecuaciones $A\mathbf{x} = \mathbf{b}$ obtenemos $LU\mathbf{x} = \mathbf{b}$, de aquí podemos despejar también \mathbf{x} de una forma más tradicional, es decir,

$$\begin{aligned}
 LU\mathbf{x} &= \mathbf{b}, \\
 U\mathbf{x} &= L^{-1}\mathbf{b}, \\
 \mathbf{x} &= U^{-1}L^{-1}\mathbf{b}.
 \end{aligned}$$

En este punto es muy importante destacar que si bien aparecen las matrices inversas de L y de U en el despeje de \mathbf{x} , no necesitamos en realidad obtener las inversas! Por ejemplo, para obtener $L^{-1}\mathbf{b}$, debemos interpretar que es lo que realmente se está solicitando. Por ejemplo, surgen, por lo menos, las siguientes 2 interpretaciones:

1. Me están solicitando obtener la matriz inversa de L , es decir L^{-1} , y luego multiplicarla por el vector \mathbf{b} , para obtener el vector resultante de ese producto.
2. Me está solicitando obtener el vector resultante del producto entre L^{-1} y \mathbf{b} , pero no es requerido explícitamente obtener la matriz inversa de L .

Si bien ambas interpretaciones nos llevan a la misma respuesta, la segunda nos ahorra un montón de computación. Entonces, ¿cómo realmente debemos entender la segunda interpretación? La palabra clave en ambas interpretaciones es que se está solicitando el vector resultante que se obtiene luego de multiplicar la matriz inversa de L y \mathbf{b} , por ejemplo llamemos a este vector \mathbf{c} . Esto significa que \mathbf{c} es igual a $L^{-1}\mathbf{b}$, o mejor dicho,

$$\mathbf{c} = L^{-1}\mathbf{b}.$$

Multiplicando por L por la izquierda obtenemos,

$$\begin{aligned}
 L\mathbf{c} &= LL^{-1}\mathbf{b} \\
 L\mathbf{c} &= I\mathbf{b} \\
 L\mathbf{c} &= \mathbf{b},
 \end{aligned}$$

donde I es la matriz identidad. Por lo tanto obtuvimos un sistema de ecuaciones lineales donde la matriz asociada es triangular inferior, por lo que podemos aplicar Forward Substitution! Que es exactamente lo mismo que ya hicimos. La moraleja acá es que fuimos capaces de obtener el producto entre la inversa de una matriz y un vector sin calcular la inversa! Este mismo procedimiento se puede aplicar con $U^{-1}(L^{-1}\mathbf{b})$, donde primero se obtiene $L^{-1}\mathbf{b}$ y luego, con Backward Substitution, se obtiene $U^{-1}(L^{-1}\mathbf{b})$. En general, cuando se solicita obtener la inversa de una matriz multiplicada por un vector, uno puede resolver el sistema de ecuaciones asociado.

Comentario al margen

La relación entre la eliminación Gaussiana y la factorización LU se obtiene al multiplicar por L^{-1} por la izquierda al sistema de ecuaciones lineales original, es decir,

$$\begin{aligned} L^{-1} A \mathbf{x} &= L^{-1} \mathbf{b}, \\ L^{-1} L U \mathbf{x} &= L^{-1} \mathbf{b}, \\ U \mathbf{x} &= \underbrace{L^{-1} \mathbf{b}}_{\mathbf{\tilde{b}}}. \end{aligned}$$

Es decir, el resultado obtenido en el tableau de la eliminación Gaussiana es lo mismo que multiplicar L^{-1} por la izquierda!

Para finalizar, debemos notar que el número de operaciones elementales requeridas para obtener las matrices L y U sigue siendo $\sim \frac{2}{3} n^3$ dado que no se requieran más operaciones para obtener L , solo se requiere almacenar los coeficientes utilizados en las operaciones filas. Los costos de realizar Backward Substitution y Forward Substitution no modifican la complejidad, por lo cual se omiten.

4.4.2. ¿Cuándo es útil la factorización LU?

En realidad es muy útil en muchas situaciones, acá solo mencionaremos algunos casos. Por ejemplo, si tenemos una secuencia de sistemas de ecuaciones lineales $n \times n$ de la siguiente forma:

$$\begin{aligned} A \mathbf{x}_1 &= \mathbf{b}_1, \\ A \mathbf{x}_2 &= \mathbf{b}_2, \\ &\vdots \\ A \mathbf{x}_m &= \mathbf{b}_m. \end{aligned}$$

En este caso la tarea es dado $A, \mathbf{b}_1, \dots, \mathbf{b}_m$, encontrar $\mathbf{x}_1, \dots, \mathbf{x}_m$. Una aproximación natural sería resolver cada sistema de ecuaciones lineales de forma independiente, lo que significa obtener m factorizaciones LU de A , m Backward Substitutions y m Forward Substitutions. Entonces la cantidad de operaciones elementales sería:

$$\# \text{ Op. Elem. Alg. 1} \sim \frac{2}{3} m n^3 + 2 m n^2.$$

En cambio, si tomamos ventaja de que la matriz A es la misma en todos los sistemas de ecuaciones lineales solo necesitamos obtener una sola vez la factorización de LU de A . Entonces, el número de operaciones elementales total de este algoritmo modificado es,

$$\# \text{ Op. Elem. Alg. 2} \sim \frac{2}{3} n^3 + 2 m n^2.$$

En este caso, al tomar ventaja de que la matriz es la misma, reducimos significativamente el costo computacional al eliminar el coeficiente m en el primer término del número de operaciones elementales. En el segundo término aún corresponde mantener m . Este problema puede considerarse como si estuviéramos resolviendo la siguiente ecuación matricial,

$$\underbrace{A [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m]}_X = \underbrace{[\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m]}_B$$

$$A X = B.$$

Un resultado interesante de este análisis es el computo de la inversa de la matriz A por medio de la factorización LU. Por ejemplo, si consideramos que $m = n$ en el ejemplo anterior y que la matriz B es la matriz identidad I , entonces obtenemos la siguiente ecuación matricial,

$$AX = I,$$

$$A \underbrace{[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]}_X = \underbrace{[\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n]}_I,$$

donde \mathbf{e}_i es el i -ésimo vector canónico. Al resolver estos n sistemas de ecuaciones lineales lo que obtendremos al final es la matriz X que al ser multiplicada por A da la matriz identidad, por lo tanto X es la matriz inversa de A ! Por lo tanto podemos obtener la inversa de una matriz solo realizando $\sim \frac{8}{3}n^3$ operaciones elementales! Aún así, no se recomienda obtener la inversa para resolver un sistema de ecuaciones lineales!

Otra situación en donde es útil la factorización LU es cuando se necesita resolver el siguiente sistema de ecuaciones lineales,

$$A^l \mathbf{x} = \mathbf{b},$$

para $l \in \mathbb{N}$ y $l > 0$. Para el cual podemos construir, por lo menos, 2 algoritmos:

1. Obtener la matriz $B = A^l$ como el producto de la matriz A consigo misma $l - 1$ veces. Luego obtener la factorización LU de B , y finalmente obtener \mathbf{x} luego de aplicar Forward y Backward Substitution.
2. La segunda opción es obtener la factorización LU de A y resolver la siguiente secuencia de sistemas de ecuaciones lineales para \mathbf{c}_{l-1} hasta \mathbf{c}_1 ,

$$\begin{aligned} A \underbrace{(A^{l-1} \mathbf{x})}_{\mathbf{c}_{l-1}} &= \mathbf{b}, \\ A \underbrace{(A^{l-2} \mathbf{x})}_{\mathbf{c}_{l-2}} &= \mathbf{c}_{l-1}, \\ A \underbrace{(A^{l-3} \mathbf{x})}_{\mathbf{c}_{l-3}} &= \mathbf{c}_{l-2}, \\ &\vdots \\ A \underbrace{(A \mathbf{x})}_{\mathbf{c}_1} &= \mathbf{c}_2, \\ A \mathbf{x} &= \mathbf{c}_1. \end{aligned}$$

¿Cuál de los 2 algoritmos requiere un menor número de operaciones elementales?¹¹

4.4.3. Número de condición κ de un sistema de ecuaciones lineales

En esta sección re-visitaremos el concepto del número de condición y como afecta la computación al resolver sistemas de ecuaciones lineales. Por ejemplo, sea \mathbf{x}_a una solución aproximada al sistema de ecuaciones lineales $A\mathbf{x} = \mathbf{b}$. Considere que trabajaremos con la siguiente instancia,

$$\begin{bmatrix} 1 & 1 \\ 1+\epsilon & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2+\epsilon \end{bmatrix}.$$

¹¹En el primer caso debe obtener el número de operaciones elementales requerido para hacer el producto de todas las matrices y luego la factorización LU. En el segundo caso se debe incluir el costo de obtener la factorización LU de A y luego aplicar Forward y Backward Substitutions l veces. Obtenga los valores!

Donde sabemos que la solución es $\mathbf{x} = [1, 1]^T$. Considere que se ha encontrado la siguiente aproximación de la solución $\mathbf{x}_a = [-1, 3+\epsilon]^T$, que claramente es una no muy buena aproximación! Sin embargo, analicemos qué ocurre con el Backward-Error¹², que se define como $\|\mathbf{b} - A\mathbf{x}_a\|$ y el Forward-Error, que se define como $\|\mathbf{x} - \mathbf{x}_a\|$. Entonces, el Backward-Error es el siguiente,

$$\begin{aligned}\|\mathbf{b} - A\mathbf{x}_a\|_\infty &= \left\| \begin{bmatrix} 2 \\ 2+\epsilon \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 1+\epsilon & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 3+\epsilon \end{bmatrix} \right\|_\infty \\ &= \left\| \begin{bmatrix} 2 \\ 2+\epsilon \end{bmatrix} - \begin{bmatrix} 2+\epsilon \\ 2 \end{bmatrix} \right\|_\infty \\ &= \left\| \begin{bmatrix} -\epsilon \\ \epsilon \end{bmatrix} \right\|_\infty \\ &= \epsilon.\end{aligned}$$

Y el Forward-Error,

$$\begin{aligned}\|\mathbf{x} - \mathbf{x}_a\|_\infty &= \left\| \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} -1 \\ 3+\epsilon \end{bmatrix} \right\|_\infty \\ &= \left\| \begin{bmatrix} 2 \\ -2-\epsilon \end{bmatrix} \right\|_\infty \\ &= 2+\epsilon.\end{aligned}$$

Lo interesante de este análisis es que nos demuestra que el tener un Backward-Error pequeño, no necesariamente implica tener un Forward-Error pequeño. Recuerde que ϵ lo estamos considerando como un valor mayor que 0 y pequeño. Este ejemplo es muy importante, ya que si bien uno quiere que el Forward-Error sea pequeño, en la práctica no podemos obtenerlo. Solo podemos obtener el Backward-Error.

La pregunta que surge ahora es ¿Cómo está relacionado el Forward-Error con el Backward-Error? La respuesta es que están conectado por medio del número de condición $\kappa(A)$ de la matriz A . El número de condición $\kappa(A)$ se define de la siguiente forma,

$$\kappa_p(A) = \|A\|_p \|A^{-1}\|_p \geq 1,$$

para alguna norma matricial $\|\cdot\|_p$. Considerando que el radio espectral nos provee una cota inferior para todas las norma matriciales, ver Sección 4.7.5, podemos obtener una cota inferior de $\kappa(A)$ de la siguiente forma,

$$\kappa_p(A) = \frac{\max_i(|\lambda_i|)}{\min_i(|\lambda_i|)},$$

donde λ_i son los valores propios de A . Pero, ¿cómo están relacionados el Forward error y el Backward error?

$$\frac{1}{\kappa(A)} \cdot \frac{\|\mathbf{b} - A \cdot \mathbf{x}_a\|}{\|\mathbf{b}\|} \leq \frac{\|\mathbf{x}_a - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \kappa(A) \cdot \frac{\|\mathbf{b} - A \cdot \mathbf{x}_a\|}{\|\mathbf{b}\|} \quad (4.9)$$

¿Qué pasa en nuestro ejemplo previo entonces? Para responder este pregunta, necesitamos obtener todas las componentes indicadas.

$$\begin{aligned}A &= \begin{bmatrix} 1 & 1 \\ 1+\epsilon & 1 \end{bmatrix}, \\ \|A\|_\infty &= 2+\epsilon, \\ A^{-1} &= \begin{bmatrix} -1 & 1 \\ 1+\epsilon & -1 \end{bmatrix} \cdot \frac{1}{\epsilon}, \\ \|A^{-1}\|_\infty &= \frac{2}{\epsilon} + 1.\end{aligned}$$

¹²Este error también se denota como la norma del vector residual $\mathbf{r} = \mathbf{b} - A\mathbf{x}_a$, es decir $\|\mathbf{r}\|$. O simplemente como el residuo.

Por lo tanto, el número de condición de A considerando la norma infinito es:

$$\kappa_{\infty}(A) = \|A\|_{\infty} \cdot \|A^{-1}\|_{\infty} = \frac{4}{\epsilon} + 4 + \epsilon.$$

Lo cual no es una muy buena noticia, ya que a medida que ϵ tiende a 0, el número de condición aumenta! Por ejemplo, si analizamos la desigualdad presentada en la ecuación (4.9). En particular, para la cota superior, es decir,

$$\begin{aligned} \frac{\|\mathbf{x} - \mathbf{x}_a\|_{\infty}}{\|\mathbf{x}\|_{\infty}} &\leq \kappa(A) \cdot \frac{\|\mathbf{b} - A \cdot \mathbf{x}_a\|_{\infty}}{\|\mathbf{b}\|_{\infty}} \\ &\leq \left(\frac{4}{\epsilon} + 4 + \epsilon\right) \cdot \frac{\epsilon}{2 + \epsilon} \\ &\leq 2 + \epsilon. \end{aligned}$$

Por lo tanto, ahora entendemos la relación entre el Forward-Error y el Backward-Error! Se sugiere ver el ejemplo numérico con la matriz de Hilbert en Jupyter notebook.

¡MUY IMPORTANTE!

Rule of Thumb: En general $\log_{10} \kappa(A)$ indica cuántos dígitos se perderán en la computación de \mathbf{x} ! Por lo cual hay que evitar matrices mal condicionadas!^a

^aUna alternativa de trabajar con matrices mal condicionadas es utilizar un preconditionador!

4.5. Factorización $PA = LU$

Hasta este momento hemos supuesto implícitamente que siempre existe la factorización LU de una matriz, o incluso que va a funcionar bien. Sin embargo no es el caso, por ejemplo considere la siguiente matriz $\begin{bmatrix} 0 & 2 \\ 3 & 4 \end{bmatrix}$. Esta matriz no tiene una factorización LU dado que tiene un 0 en el primer pivote, y no hay forma de multiplicar 0 por un número tal que pueda hacer cero el primer coeficiente de la segunda fila. Otra desventaja de la factorización LU es que puede inducir grandes errores en la computación. Un ejemplo de esto último se presenta a continuación. Considerando $\delta = 10^{-20}$,

$$\begin{bmatrix} \delta & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}.$$

Para esta matriz obtenemos la siguiente factorización LU obtenida usando aritmética de double precision,

$$\begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix} \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix}.$$

Notar que al hacer el producto en double precision obtenemos:

$$\begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix}. \quad (4.10)$$

La cual claramente no es la matriz original, pero es lo que se obtuvo numéricamente. Resolviendo el sistema de ecuaciones lineales con la factorización LU encontrada y considerando que utilizamos double precision obtenemos la siguiente aproximación a la solución,

$$\mathbf{x}_a = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

La cual claramente está bastante alejada de la solución. A modo referencial, podemos obtener la factorización LU de la matriz A de forma simbólica para entender que es lo que está ocurriendo, en este caso se obtendría:

$$A = \begin{bmatrix} 1 & 0 \\ \delta^{-1} & 1 \end{bmatrix} \begin{bmatrix} \delta & 1 \\ 0 & 2 - \delta^{-1} \end{bmatrix}.$$

Claramente se observa que el coeficiente $\text{fl}(2 - \delta^{-1}) = \text{fl}(2 - 10^{20}) = -10^{20}$, es decir, por pérdida de importancia el valor 2 no afecta al valor 10^{20} en double precision, por lo cual solo se almacena 10^{20} . Esta situación es precisamente lo que provoca que el producto numérico entre L y U obtenido en la ecuación (4.10) no sea exactamente la A original.

Ahora, considere que alternamos las filas de la matriz, o equivalentemente multiplicamos el sistema de ecuaciones lineales por $P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, es decir,

$$\begin{aligned} A\mathbf{x} &= \mathbf{b}, \\ PA\mathbf{x} &= P\mathbf{b}. \end{aligned}$$

En este caso obtenemos el siguiente sistema de ecuaciones lineales,

$$\begin{bmatrix} 1 & 2 \\ \delta & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}.$$

Si bien el cambio puede parecer poco significativo, realmente hace una gran diferencia. Ahora la factorización LU de la matriz PA es la siguiente,

$$PA = LU = \begin{bmatrix} 1 & 0 \\ 10^{-20} & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}.$$

Y al resolver al sistema de ecuaciones asociados se obtiene la siguiente aproximación,

$$\mathbf{x}_a = \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

La cual es mucho mejor que la obtenida anteriormente! Como vemos, la matriz P , al multiplicarla ambos lados, lo único que hizo es un cambio de filas. Sin embargo, este cambio de filas redujo significativamente el error numérico. Esta es la idea de la factorización $PA = LU$, es decir, obtener la factorización LU de una matriz con un distinto orden de filas.

La realización de permutaciones de filas se obtienen por medio de una **matriz de permutación**, la cual es una matriz P de tamaño $n \times n$ donde todos sus valores son ceros, excepto por un uno en cada fila y cada columna. De forma equivalente, una matriz de permutación se obtiene al efectuar cambios de fila (o de columna) a la matriz identidad I .

La idea que está detrás de la factorización PALU es dejar en la diagonal de la matriz el máximo coeficiente, en valor absoluto, de la columna bajo la diagonal principal, es decir dejar en la diagonal el coeficiente $\max_{i>j} |a_{ij}|$. Esto es llamado **partial pivoting**. Antes de efectuar las operaciones fila, debemos revisar que el elemento “pivote” a usar es mayor que los elementos hacia abajo en la columna. Si no es así, hacemos la permutación de filas necesaria para corregir esto.

De ésta forma, los “factores” que usaremos en cada operación fila nunca serán mayores a 1 en valor absoluto. Afortunadamente, esto no modifica el algoritmo básico de la factorización LU, solo agrega la componente de la permutación. La cual no agrega ninguna operación elemental dado que la matriz de permutación se puede almacenar convenientemente. El siguiente ejemplo muestra como quedaría la factorización PALU.

$$\begin{array}{ccccc}
 \underbrace{\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}}_{P_1} & \underbrace{\begin{bmatrix} 2 & 1 & 5 \\ 4 & 4 & -4 \\ 1 & 3 & 1 \end{bmatrix}}_A & = & \begin{bmatrix} 2 & 1 & 5 \\ 4 & 4 & -4 \\ 1 & 3 & 1 \end{bmatrix}, & \underbrace{\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}}_P, & \underbrace{\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}}_L, \\
 \underbrace{\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}}_{P_1} & \underbrace{\begin{bmatrix} 2 & 1 & 5 \\ 4 & 4 & -4 \\ 1 & 3 & 1 \end{bmatrix}}_A & = & \begin{bmatrix} 4 & 4 & -4 \\ 2 & 1 & 5 \\ 1 & 3 & 1 \end{bmatrix}, & \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} = P_1 P, & \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}, \\
 \underbrace{\begin{bmatrix} 1 & & \\ -\frac{1}{2} & 1 & \\ & & 1 \end{bmatrix}}_{L_1} & \begin{bmatrix} 4 & 4 & -4 \\ 2 & 1 & 5 \\ 1 & 3 & 1 \end{bmatrix} & = & \begin{bmatrix} 4 & 4 & -4 \\ 0 & -1 & 7 \\ 1 & 3 & 1 \end{bmatrix}, & \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}, & \begin{bmatrix} 1 & & \\ \frac{1}{2} & 1 & \\ & & 1 \end{bmatrix} \\
 \underbrace{\begin{bmatrix} 1 & & \\ & 1 & \\ -\frac{1}{4} & & 1 \end{bmatrix}}_{L_2} & \begin{bmatrix} 4 & 4 & -4 \\ 0 & -1 & 7 \\ 1 & 3 & 1 \end{bmatrix} & = & \begin{bmatrix} 4 & 4 & -4 \\ 0 & -1 & 7 \\ 0 & 2 & 2 \end{bmatrix} & \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}, & \begin{bmatrix} 1 & & \\ \frac{1}{2} & 1 & \\ \frac{1}{4} & & 1 \end{bmatrix}, \\
 \underbrace{\begin{bmatrix} 1 & & \\ & & 1 \\ & 1 & \end{bmatrix}}_{P_2} & \begin{bmatrix} 4 & 4 & -4 \\ 0 & -1 & 7 \\ 0 & 2 & 2 \end{bmatrix} & = & \begin{bmatrix} 4 & 4 & -4 \\ 0 & 2 & 2 \\ 0 & -1 & 7 \end{bmatrix}, & \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} = P_2 P, & \underbrace{\begin{bmatrix} 1 & & \\ \frac{1}{4} & 1 & \\ \frac{1}{2} & & 1 \end{bmatrix}}_{L_3}, \\
 & & & & & \text{Notar el cambio de los} \\
 & & & & & \text{coeficientes respecto} \\
 & & & & & \text{al } L \text{ anterior.}
 \end{array}$$

Por lo tanto la factorización PALU es la siguiente:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 & 5 \\ 4 & 4 & -4 \\ 1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 4 & 4 & -4 \\ 0 & 2 & 2 \\ 0 & 0 & 8 \end{bmatrix}$$

En general P no es conocido a priori y se obtiene durante la ejecución del algoritmo, el procedimiento para obtener U se describe de la siguiente forma,

$$\begin{aligned} A &= U_0 \\ P_1 A &= U_{\frac{1}{2}} \\ L_1 P_1 A &= U_1 \\ P_2 L_1 P_1 A &= U_{1+\frac{1}{2}} \\ L_2 P_2 L_1 P_1 A &= U_2 \\ P_3 L_2 P_2 L_1 P_1 A &= U_{2+\frac{1}{2}} \\ L^{-1} P A &:= L_3 P_3 L_2 P_2 L_1 P_1 A = U_3 =: U \end{aligned}$$

De la anterior aún podemos obtener PA por medio de la siguiente definición de una matriz similar a las L_i originales, esta corresponden a $L'_i = P_{n-1} \dots P_{i+1} L_i P_{i+1}^{-1} \dots P_{n-1}^{-1}$. Notar que $P_k^{-1} = P_k$, por lo que lo único que hace esta variante es que alterna las filas de L_i al multiplicar por la izquierda por los P_k , y luego alterna las columnas del

producto resultante al multiplicar por P_k^{-1} por la derecha. Al finalizar la matriz L'_i sigue teniendo el mismo patrón de la matriz L_i original. Este cambio nos permite re-escribir el producto $L_3 P_3 L_2 P_2 L_1 P_1$ de la siguiente forma:

$$L_3 P_3 L_2 P_2 L_1 P_1 = L'_3 L'_2 L'_1 P_3 P_2 P_1,$$

reemplazando en por la definición de L'_i obtenemos,

$$\begin{aligned} L'_3 L'_2 L'_1 P_3 P_2 P_1 &= L_3 (P_3 L_2 P_3^{-1}) (P_3 P_2 L_1 P_2^{-1} P_3^{-1}) P_3 P_2 P_1 \\ &= L_3 P_3 L_2 P_2 L_1 P_1. \end{aligned}$$

Entonces al finalizar PALU obtenemos la siguiente identidad,

$$L'_3 L'_2 L'_1 P_3 P_2 P_1 A = U.$$

Considerando $L^{-1} = L'_3 L'_2 L'_1$ y $P = P_3 P_2 P_1$, obtenemos finalmente, $PA = LU$. Es importante notar que U también fue alternando filas según avanzaba la computación, por lo que no es igual al U de la factorización de ALU. Se sugiere revisar la implementación de PALU en el Jupyter Notebook correspondiente, en el notebook se implementan alternando las filas de U parcial que se tiene. Notar que el U parcial que se tiene no se le agregan los 1s en la diagonal para que al permutar filas no se desordenen los elementos de la diagonal, solo se agregan al final.

Una vez que obtenemos las matrices P , L y U , podemos resolver el sistema de ecuaciones asociado $A\mathbf{x} = \mathbf{b}$. El procedimiento es el siguiente:

- Multiplicar por P por la izquierda a ambos lados del sistema de ecuaciones lineales:

$$PA\mathbf{x} = P\mathbf{b},$$

luego reemplazamos $PA = LU$,

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

- Resolvemos $L\mathbf{c} = P\mathbf{b}$ para \mathbf{c} , usando Forward Substitution.
- Resolvemos $U\mathbf{x} = \mathbf{c}$ para \mathbf{x} , usando Backward Substitution.

En resumen, es lo mismo que hicimos antes pero solo se requiere permutar el vector \mathbf{b} , pero no es necesario permutar las fila de A !

4.6. Método de Newton en \mathbb{R}^n

Hasta este punto hemos estudiado algunos algoritmos para resolver sistemas de ecuaciones lineales. En esta sección haremos una pausa antes de estudiar más algoritmos de resolución de sistemas de ecuaciones lineales. Por lo cual estudiaremos el método de Newton en alta dimensión, es decir \mathbb{R}^2 o superior. Al igual que en 1D, el método de Newton se utiliza para resolver sistemas de ecuaciones no-lineales. A modo referencial, recordemos que en 1D para encontrar una raíz de $f(x)$ se propuso la siguiente iteración de punto fijo,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

la cual comienza de un initial guess x_0 . Estudiamos que, en el caso base, se observa convergencia cuadrática y, cuando la raíz es además un punto crítico, la convergencia observada es lineal. Considere el siguiente ejemplo,

$$\begin{aligned} x^2 + y^2 &= 1, \\ y &= x^2. \end{aligned}$$

¿Cómo podemos encontrar la intersección de las curvas? Tradicionalmente se podría utilizar el método de la sustitución, es decir,

$$y + y^2 = 1.$$

Lo que nos da $y_{\pm} = \frac{-1 \pm \sqrt{5}}{2}$, es decir,

$$\begin{aligned} y_1 &\approx 0,61, \\ y_2 &\approx -1,61. \end{aligned}$$

¿Cómo obtenemos el valor o valores de x ?¹³

El análisis anterior puede re-escribirse de la siguiente forma,

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} x^2 + y^2 - 1 \\ y - x^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Es decir, nos interesa buscar una raíz de $\mathbf{F}(\mathbf{x})$. Utilizando un análisis similar al utilizado para derivar el método de Newton en 1D proponemos la siguiente linealización de $\mathbf{F}(\mathbf{x})$,

$$\begin{aligned} \underbrace{\mathbf{F}(\mathbf{x}_{i+1})}_0 &= \mathbf{F}(\mathbf{x}_i) + J(\mathbf{x}_i) (\mathbf{x}_{i+1} - \mathbf{x}_i) + O(\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2), \\ -\mathbf{F}(\mathbf{x}_i) &= J(\mathbf{x}_i) (\mathbf{x}_{i+1} - \mathbf{x}_i), \\ -J^{-1}(\mathbf{x}_i) \mathbf{F}(\mathbf{x}_i) &= \mathbf{x}_{i+1} - \mathbf{x}_i, \\ \mathbf{x}_{i+1} &= \mathbf{x}_i - J^{-1}(\mathbf{x}_i) \mathbf{F}(\mathbf{x}_i), \end{aligned}$$

donde $J(\mathbf{x}_i)$ es la matriz Jacobiana de \mathbf{F} evaluada en el punto \mathbf{x}_i . Desde el punto de vista notacional, se presentan las siguientes formas equivalentes de escribir $\mathbf{F}(\mathbf{x})$,

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}\right) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) \end{bmatrix}.$$

Lo mismo ocurre con la forma de escribir la matriz Jacobiana de \mathbf{F} evaluada en \mathbf{x}_i . A continuación presentamos algunas formas equivalentes,

$$J(\mathbf{x}) = J_{\mathbf{F}}(\mathbf{x}_i) = J(F)|_{\mathbf{x}=\mathbf{x}_i} = \left[\frac{\partial f_i}{\partial x_j} \right] \bigg|_{\mathbf{x}=\mathbf{x}_i} = \begin{bmatrix} \nabla f_1 \\ \nabla f_2 \\ \vdots \\ \nabla f_n \end{bmatrix} \bigg|_{\mathbf{x}=\mathbf{x}_i}$$

Volviendo al ejemplo inicial, podemos entonces aplicar el método de Newton. La función es entonces,

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} x^2 + y^2 - 1 \\ y - x^2 \end{bmatrix} = \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix}.$$

¹³Utilizando la segunda ecuación con y_1 .

Lo primero que debemos hacer es obtener la matriz Jacobiana de \mathbf{F} , entonces procederemos a obtener los gradientes de f_1 y f_2 ,

$$\begin{aligned}\nabla f_1(x, y) &= \left\langle \frac{\partial f_1}{\partial x}, \frac{\partial f_1}{\partial y} \right\rangle = \langle 2x, 2y \rangle, \\ \nabla f_2(x, y) &= \left\langle \frac{\partial f_2}{\partial x}, \frac{\partial f_2}{\partial y} \right\rangle = \langle -2x, 1 \rangle.\end{aligned}$$

Por lo tanto la matriz Jacobiana es,

$$J(\mathbf{x}) = \begin{bmatrix} 2x & 2y \\ -2x & 1 \end{bmatrix}.$$

Entonces el método de Newton genera la siguiente iteración de punto fijo de alta dimensión,

$$\begin{aligned}\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} &= \begin{bmatrix} x_i \\ y_i \end{bmatrix} - J^{-1}(\mathbf{x}_i) \begin{bmatrix} f_1(x_i, y_i) \\ f_2(x_i, y_i) \end{bmatrix} \\ \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} &= \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} 2x_i & 2y_i \\ -2x_i & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_i^2 + y_i^2 - 1 \\ y_i - x_i^2 \end{bmatrix}\end{aligned}$$

A continuación se presentan 2 implementaciones del método de Newton en los Algoritmos 3 y 4. El Algoritmo 3

```

1  $\mathbf{x}_0$  = "Initial guess"
2 for  $i$  in  $\text{range}(n)$  :
3      $\mathbf{x}_{i+1} = \mathbf{x}_i - (J(\mathbf{x}_i))^{-1} \mathbf{F}(\mathbf{x}_i)$ 
```

Algoritmo 3: Método de Newton

se puede re-escribir para que no quede expresado dependiendo de la inversa de la matriz Jacobiana. Esta modificación se presenta en el Algoritmo 4. Al igual que en caso 1D, la solución encontrada por el método de Newton en

```

1  $\mathbf{x}_0$  = "Initial guess"
2 for  $i$  in  $\text{range}(n)$  :
3     solve  $J(\mathbf{x}_i) \mathbf{w} = -\mathbf{F}(\mathbf{x}_i)$  for  $\mathbf{w}$ 
4      $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{w}$ 
```

Algoritmo 4: Método de Newton, versión 2

alta dimensión depende de la estimación inicial. Se sugiere encarecidamente ver el Jupyter Notebook asociado!

4.7. Métodos Iterativos

La eliminación Gaussiana, las factorizaciones LU y $PA = LU$, y la factorización de Cholesky¹⁴ se conocen como Métodos Directos, es decir, terminan luego de una cantidad finita de operaciones elementales. El desafío con estos métodos surge cuando el tamaño o dimensionalidad del problema empieza a crecer. Es decir, resolver problemas

¹⁴Que también es una factorización matricial de la forma $A = R^T R$, donde R es una matriz triangular superior. Sin embargo se requiere que la matriz A sea simétrica, i.e. $A = A^T$, y definida positiva, i.e. todos sus valores propios deben ser positivos. Este método logra disminuir el tiempo de computación respecto a PALU en un 50%, es decir tiene una complejidad computacional $\sim \frac{1}{3} n^3$, lo cual es una mejora muy interesante! Ver apéndice D.1

con un $n \gg 1$. Por ejemplo, si una factorización LU demora $T_1 \approx \frac{2}{3} n^3 \tau_{oe}$ unidades de tiempo, donde τ_{oe} es el tiempo aproximado requerido por operación elemental. Entonces resolver un problema del doble de tamaño demorará 8 veces T_1 ! Esto se debe a que,

$$\begin{aligned} T_2 &\approx \frac{2}{3} (2n)^3 \tau_{oe} \\ &\approx \frac{2}{3} 2^3 n^3 \tau_{oe} \\ &\approx 2^3 \frac{2}{3} n^3 \tau_{oe} \\ &\approx 2^3 T_1 \\ &\approx 8 T_1. \end{aligned}$$

Es decir, la relación entre la dimensionalidad del problema y el tiempo de computación no es lineal, de hecho es cúbica. Por lo tanto, rápidamente nos damos cuenta que para valores de n muy grandes los tiempos son prohibitivos. Por ejemplo si consideramos que $\tau_{oe} = 5 [ns]$, en la Tabla 4.2 se presentan los tiempos de computación requeridos por la factorización LU para distintos valores de n . Los tiempos presentados en la Tabla 4.2 aumentan en 1000 veces

Tabla 4.2: Tiempo de computación requerido por factorización LU para varios valores de n

n	Tiempo de computación
10^3	≈ 3 segundos.
10^4	≈ 55 minutos y 33 segundos.
10^5	≈ 38 días, 13 horas, 55 minutos y 33 segundos.

cada vez que el tamaño aumenta solo 10 veces, esto es debido a la complejidad computacional cúbica indicada anteriormente.

Por otro lado, los Métodos Iterativos aparecen como una alternativa a los métodos directos a costa de entregar una solución aproximada que se puede ir mejorando al ir iterando. Por ejemplo, una iteración de punto fijo en 1D, para la búsqueda de un punto fijo, es un método iterativo, que se aproxima al punto fijo a medida que uno itera. Siempre y cuando la iteración de punto fijo sea convergente! En esta sección veremos métodos iterativos para encontrar una aproximación a la solución de un sistema de ecuaciones lineales. Estos métodos pueden entenderse como una iteración de punto fijo de alta dimensión, lo cual nos da la posibilidad de construir nuevas variantes! Recuerde que la creatividad es crucial para obtener iteraciones de punto fijo convergentes.

4.7.1. Método de Jacobi

El primer algoritmo que estudiaremos es el método de Jacobi. Para explicar este método, primero debemos considerar la siguiente descomposición matricial de la matriz A :

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & \ddots & a_{2,n-1} & a_{2,n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,1} & a_{n,2} & \dots & a_{n,n-1} & a_{n,n} \end{bmatrix} = L + D + U$$

$$= \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ a_{2,1} & 0 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & 0 & 0 \\ a_{n,1} & a_{n,2} & \dots & a_{n,n-1} & 0 \end{bmatrix}}_L + \underbrace{\begin{bmatrix} a_{1,1} & 0 & \dots & 0 & 0 \\ 0 & a_{2,2} & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & a_{n-1,n-1} & 0 \\ 0 & 0 & \dots & 0 & a_{n,n} \end{bmatrix}}_D + \underbrace{\begin{bmatrix} 0 & a_{1,2} & \dots & a_{1,n-1} & a_{1,n} \\ 0 & 0 & \ddots & a_{2,n-1} & a_{2,n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & a_{n-1,n} \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}}_U.$$

Es decir, la matriz A se descompone en la suma de 3 matrices: L , D y U . La definición de cada una es la siguiente:

- L : Contiene los elementos bajo la diagonal de A .
- D : Contiene los elementos de la diagonal de A .
- U : Contiene los elementos sobre la diagonal de A .

¡MUY IMPORTANTE!

¡Estos L y U son diferentes a los L y U de $PA = LU$ o $A = LU$! Acá obtener L , D , y U no requiere ningún algoritmo sofisticado, solo requiere acceder a las componentes respectivas de la matriz A .

Nota

Anteriormente usamos r para denotar a la raíz o un punto fijo de una función en \mathbb{R} , sin embargo en este y los próximos capítulos \mathbf{r} (vector) está asociado a otro concepto. Acá $\mathbf{r} = \mathbf{b} - A\mathbf{x}_a$ es el residuo, donde \mathbf{x}_a es la solución aproximada obtenida. Por lo tanto no se usará \mathbf{r} , en la medida de lo posible, para indicar soluciones de sistemas de ecuaciones lineales o no lineales.

Para obtener el método de Jacobi podemos partir del problema base, es decir un sistema de ecuaciones lineales:

$$A\mathbf{x} = \mathbf{b}.$$

Ahora, el paso natural es reemplazar A por la descomposición introducida anteriormente y expandiendo los términos:

$$\begin{aligned} A\mathbf{x} &= \mathbf{b}, \\ (L + D + U)\mathbf{x} &= \mathbf{b}, \\ L\mathbf{x} + D\mathbf{x} + U\mathbf{x} &= \mathbf{b}. \end{aligned}$$

Dejando solo el término $D\mathbf{x}$ al lado izquierdo de la ecuación obtenemos:

$$\begin{aligned} D\mathbf{x} &= \mathbf{b} - L\mathbf{x} - U\mathbf{x}, \\ D\mathbf{x} &= \mathbf{b} - (L + U)\mathbf{x}. \end{aligned}$$

Luego, multiplicando por la inversa de D por la izquierda podemos despejar \mathbf{x} ,

$$\begin{aligned} D^{-1} D\mathbf{x} &= D^{-1} (\mathbf{b} - (L + U)\mathbf{x}), \\ \mathbf{x} &= D^{-1} (\mathbf{b} - (L + U)\mathbf{x}). \end{aligned}$$

Lo cual nos da una iteración de punto fijo de alta dimensión para resolver un sistema de ecuaciones lineales. La iteración de punto fijo en este caso se denota como el método de Jacobi y se describe de la siguiente forma:

$$\begin{aligned} \mathbf{x}_0 &= \text{"dato inicial"}, \\ \mathbf{x}_{n+1} &= D^{-1} (\mathbf{b} - (L + U)\mathbf{x}_n), \end{aligned}$$

donde la segunda ecuación se puede interpretar como $\mathbf{x}_{n+1} = \mathbf{G}(\mathbf{x}_n)$, es decir, una iteración de punto fijo en alta dimensión! Las distintas definiciones de " \mathbf{G} " definirán distintos algoritmos.

Nota

En este capítulo y en los próximos se usará la notación de sub-índice asociado a un vector para indicar iteraciones o distintos vectores en general, por ejemplo en la derivación anterior del método de Jacobi se utiliza \mathbf{x}_n para denotar la aproximación de la solución obtenida por el método de Jacobi en la n -ésima iteración. Esta nota se incluye porque en algunas explicaciones, libros, apuntes, etc, se utiliza el sub-índice para denotar el i -ésimo elemento del vector en cuestión, por lo cual usted debe estar atento a esta dualidad de interpretación y no cometer errores en la interpretación. En estos apuntes se dejará, en la medida de lo posible, claro a que interpretación se hace alusión, pero si encuentra alguna explicación que puede mejorarse, por favor enviar su sugerencia.

Existen 2 formas alternativas de definir el método de Jacobi, y cada una tiene su utilidad propia. A continuación se presenta la primera alternativa.

$$\begin{aligned} \mathbf{x}_{n+1} &= D^{-1} (\mathbf{b} - (L + U)\mathbf{x}_n), \\ &= D^{-1} (\mathbf{b} - (L + U + D - D)\mathbf{x}_n), \\ &= D^{-1} (\mathbf{b} - (A - D)\mathbf{x}_n), \\ &= D^{-1} (\mathbf{b} - A\mathbf{x}_n + D\mathbf{x}_n), \\ &= D^{-1} (\mathbf{b} - A\mathbf{x}_n) + D^{-1} D\mathbf{x}_n, \\ &= D^{-1} (\mathbf{b} - A\mathbf{x}_n) + \mathbf{x}_n, \\ &= \mathbf{x}_n + D^{-1} (\mathbf{b} - A\mathbf{x}_n), \\ &= \mathbf{x}_n + D^{-1} \mathbf{r}_n. \end{aligned}$$

donde $\mathbf{r}_n = \mathbf{b} - A\mathbf{x}_n$ es el vector residual de la n -ésima iteración. Lo interesante de esta interpretación es que nos indica que estamos encontrando una iteración de punto fijo a partir del vector residual, donde la matriz D^{-1} "ajusta" el residuo. Note que si el residuo es $\mathbf{0}$, entonces llegamos al anhelado punto fijo!¹⁵

La segunda forma alternativa del método de Jacobi se basa en escribirlo en la siguiente forma $\mathbf{x}_{n+1} = M\mathbf{x}_n + \hat{\mathbf{b}}$. Es decir, la iteración de punto fijo se re-escribe como el producto de la matriz M con el vector \mathbf{x}_n más un vector constante $\hat{\mathbf{b}}$. Siguiendo el mismo desarrollo anterior, obtenemos:

$$\begin{aligned} \mathbf{x}_{n+1} &= D^{-1} (\mathbf{b} - (L + U)\mathbf{x}_n), \\ &= D^{-1} \mathbf{b} - D^{-1} (L + U)\mathbf{x}_n, \\ &= -D^{-1} (L + U)\mathbf{x}_n + D^{-1} \mathbf{b}. \end{aligned}$$

¹⁵Siempre y cuando la matriz A no sea mal condicionada!

Entonces, la matriz M para el método de Jacobi corresponde a $M = -D^{-1}(L+U)$ y el vector constante a $\hat{\mathbf{b}} = D^{-1}\mathbf{b}$. La utilidad de esta representación la veremos cuando estudiemos la convergencia de métodos iterativos para sistemas de ecuaciones lineales en la sección 4.7.4. Lo importante en este punto es conocer que existen por lo menos 3 formas alternativas de considerar el método de Jacobi.

4.7.2. Método de Gauss-Seidel

El siguiente algoritmo que estudiaremos es el método de Gauss-Seidel. Este algoritmo es una pequeña pero significativa variante del método de Jacobi. En el método de Jacobi se construyó una iteración de punto fijo en donde se dejaba la matriz diagonal D al lado izquierdo porque es una matriz “fácil” de invertir. En este caso se dejará al lado izquierdo otra matriz “fácil” de invertir. A continuación se presenta el despeje conveniente para la construcción del método de Gauss-Seidel.

$$\begin{aligned} A\mathbf{x} &= \mathbf{b}, \\ (L + D + U)\mathbf{x} &= \mathbf{b}, \\ (L + D)\mathbf{x} + U\mathbf{x} &= \mathbf{b}, \\ (L + D)\mathbf{x} &= \mathbf{b} - U\mathbf{x}, \\ \mathbf{x} &= (L + D)^{-1}(\mathbf{b} - U\mathbf{x}). \end{aligned}$$

En este caso $(L + D)$ es una matriz triangular inferior, por lo que se puede utilizar Forward substitution para resolver el sistema de ecuaciones lineales asociado en cada iteración. Claramente se podría haber dejado al lado izquierdo la matriz $(U + D)$ y luego haber obtenido una matriz triangular superior, en ese caso se debería usar Backward substitution. Entonces, el método de Gauss-Seidel se presenta a continuación:

$$\begin{aligned} \mathbf{x}_0 &= \text{“dato inicial”}, \\ \mathbf{x}_{n+1} &= (L + D)^{-1}(\mathbf{b} - U\mathbf{x}_n). \end{aligned}$$

Al igual que en el método de Jacobi, podemos escribir en las 2 formas alternativas mencionadas anteriormente. La primera, en función del vector residual, se obtiene de la siguiente forma:

$$\begin{aligned} \mathbf{x}_{n+1} &= (L + D)^{-1}(\mathbf{b} - U\mathbf{x}_n), \\ &= (L + D)^{-1}(\mathbf{b} - (U + D + L - D - L)\mathbf{x}_n), \\ &= (L + D)^{-1}(\mathbf{b} - (L + U + D)\mathbf{x}_n + (L + D)\mathbf{x}_n), \\ &= (L + D)^{-1}(\mathbf{b} - A\mathbf{x}_n + (L + D)\mathbf{x}_n), \\ &= (L + D)^{-1}(\mathbf{b} - A\mathbf{x}_n) + (L + D)^{-1}(L + D)\mathbf{x}_n, \\ &= (L + D)^{-1}\mathbf{r}_n + \mathbf{x}_n, \\ &= \mathbf{x}_n + (L + D)^{-1}\mathbf{r}_n. \end{aligned}$$

La diferencia en este caso, al re-escribirlo en función del vector residual, es solamente la matriz que multiplica al vector residual. Por último, la segunda forma alternativa de escribir el método de Gauss-Seidel, para dejarlo en la forma $\mathbf{x}_{n+1} = M\mathbf{x}_n + \hat{\mathbf{b}}$, es la siguiente,

$$\begin{aligned} \mathbf{x}_{n+1} &= (L + D)^{-1}(\mathbf{b} - U\mathbf{x}_n), \\ &= -(L + D)^{-1}U\mathbf{x}_n + (L + D)^{-1}\mathbf{b}. \end{aligned}$$

Nuevamente recordar que esta segunda forma alternativa será útil al momento de estudiar la convergencia del método.

4.7.3. EXTRA: Succesive Over-Relaxation (SOR(ω))

Se define $\omega \in \mathbb{R}$, al cual llamamos el parámetro de relajación. Si $\omega > 1$, se refiere como sobre-relajación. Luego, multiplicamos por ω la igualdad $(L + U + D)\mathbf{x} = \mathbf{b}$, y despejamos como:

$$\begin{aligned}(L + D + U)\mathbf{x} &= \mathbf{b} & / \cdot \omega, \omega \neq 0 \\ \omega(L + D + U)\mathbf{x} &= \omega\mathbf{b} & / + D\mathbf{x} \\ \omega L\mathbf{x} + \omega D\mathbf{x} + \omega U\mathbf{x} + D\mathbf{x} &= \omega\mathbf{b} + D\mathbf{x} \\ (\omega L + D)\mathbf{x} &= \omega\mathbf{b} + (1 - \omega)D\mathbf{x} - \omega U\mathbf{x}\end{aligned}$$

Finalmente, el método iterativo nos quedaría como:

$$\begin{aligned}\Rightarrow \quad \mathbf{x}_0 &= \text{"dato inicial"} \\ \mathbf{x}_{n+1} &= (\omega L + D)^{-1} (\omega\mathbf{b} + [(1 - \omega)D - \omega U]\mathbf{x}_n) \\ &= (\omega L + D)^{-1} \omega\mathbf{b} + (\omega L + D)^{-1} [(1 - \omega)D - \omega U]\mathbf{x}_n\end{aligned}$$

Y, nuevamente, también podemos ver este método en términos de su residuo como:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \left(L + \frac{D}{\omega}\right)^{-1} \cdot (\mathbf{b} - A \cdot \mathbf{x}_n) = \mathbf{x}_n + \left(L + \frac{D}{\omega}\right)^{-1} \mathbf{r}_n$$

4.7.4. Convergencia de métodos iterativos para sistemas de ecuaciones lineales

En esta sección estudiaremos las condiciones para que exista convergencia o divergencia de método iterativos. En la Tabla 4.3 presentamos un resumen de las alternativas de representación de los algoritmos anteriormente descritos. En particular, en esta sección nos interesa la última columna, es decir la representación $\mathbf{x}_{n+1} = M\mathbf{x}_n + \hat{\mathbf{b}}$.

Tabla 4.3: Tabla resumen de alternativas de representación de métodos iterativos

	$\mathbf{x}_{n+1} = G(\mathbf{x}_n)$	$\mathbf{x}_{n+1} = \mathbf{x}_n + N\mathbf{r}_n$	$\mathbf{x}_{n+1} = M\mathbf{x}_n + \hat{\mathbf{b}}$
Jacobi	$\mathbf{x}_{n+1} = D^{-1}(\mathbf{b} - (L + U)\mathbf{x}_n)$	$\mathbf{x}_{n+1} = \mathbf{x}_n + D^{-1}\mathbf{r}_n$	$\mathbf{x}_{n+1} = -D^{-1}(L + U)\mathbf{x}_n + D^{-1}\mathbf{b}$
Gauss-Seidel	$\mathbf{x}_{n+1} = (L + D)^{-1}(\mathbf{b} - U\mathbf{x}_n)$	$\mathbf{x}_{n+1} = \mathbf{x}_n + (L + D)^{-1}\mathbf{r}_n$	$\mathbf{x}_{n+1} = -(L + D)^{-1}U\mathbf{x}_n + (L + D)^{-1}\mathbf{b}$

Considere que ejecutamos 2 iteraciones de cualquiera de los algoritmos anteriormente descritos, es decir,

$$\begin{aligned}\mathbf{x}_{n+1} &= M\mathbf{x}_n + \hat{\mathbf{b}}, \\ \mathbf{x}_{n+2} &= M\mathbf{x}_{n+1} + \hat{\mathbf{b}}.\end{aligned}$$

Restando las 2 ecuaciones obtenemos,

$$\begin{aligned}\mathbf{x}_{n+2} - \mathbf{x}_{n+1} &= M\mathbf{x}_{n+1} + \hat{\mathbf{b}} - M\mathbf{x}_n - \hat{\mathbf{b}}, \\ \mathbf{x}_{n+2} - \mathbf{x}_{n+1} &= M\mathbf{x}_{n+1} - M\mathbf{x}_n, \\ \mathbf{x}_{n+2} - \mathbf{x}_{n+1} &= M(\mathbf{x}_{n+1} - \mathbf{x}_n).\end{aligned}$$

Aplicando alguna¹⁶ norma matricial basada en alguna norma vectorial obtenemos,

$$\begin{aligned}\|\mathbf{x}_{n+2} - \mathbf{x}_{n+1}\| &= \|M(\mathbf{x}_{n+1} - \mathbf{x}_n)\|, \\ \|\mathbf{x}_{n+2} - \mathbf{x}_{n+1}\| &\leq \|M\| \|\mathbf{x}_{n+1} - \mathbf{x}_n\|.\end{aligned}$$

¹⁶Aquí hay libertad en la elección de la norma matricial basada en una norma vectorial. En la sección 4.7.5 revisaremos en más detalle este punto.

Si definimos el error como $e_n = \|\mathbf{x}_{n+1} - \mathbf{x}_n\|$, obtenemos la siguiente relación,

$$e_{n+1} \leq \|M\| e_n. \quad (4.11)$$

De la desigualdad anterior podemos concluir que si $\|M\| < 1$, entonces el método reducirá el error, por lo tanto convergerá. Una posible norma a utilizar es $\|M\|_\infty$, pero se puede utilizar cualquier norma matricial. En realidad, si se puede demostrar que para alguna norma matricial $\|M\| < 1$, entonces los métodos iterativos anteriormente descritos convergerán. Hay que tener en consideración eso sí que se debe utilizar la matriz M correspondiente para cada método. En la siguiente sección se analiza en más detalle este punto para el caso donde se use una norma matricial inducida por una norma vectorial.

4.7.5. Radio espectral y su uso en convergencia de métodos iterativos

Def 12. Sea $A \in \mathbb{C}^{n \times n}$ con valores propios $\lambda_1, \lambda_2, \dots, \lambda_n$, se define el radio espectral de A como:

$$\rho(A) = \max_{k \in \{1, 2, \dots, n\}} |\lambda_k|$$

Recordando la sección 1.5.2.1 donde se presentan las normas matriciales inducidas por normas vectoriales, podemos definir una norma matricial inducida por la p -norma vectorial de la siguiente forma:

$$\|A\|_p = \sup_{\|\mathbf{w}\|_p \neq 0} \frac{\|A\mathbf{w}\|_p}{\|\mathbf{w}\|_p} = \sup_{\|\mathbf{w}\|_p = 1} \|A\mathbf{w}\|_p.$$

De la cual podemos concluir que para un vector arbitrario \mathbf{w} con $\|\mathbf{w}\|_p \neq 0$, obtenemos la siguiente desigualdad:

$$\frac{\|A\mathbf{w}\|_p}{\|\mathbf{w}\|_p} \leq \|A\|_p.$$

Multiplicando por $\|\mathbf{w}\|_p$ obtenemos:

$$\|A\mathbf{w}\|_p \leq \|A\|_p \|\mathbf{w}\|_p.$$

Ahora consideremos que el vector \mathbf{w} es uno de los vectores propios de A , por ejemplo \mathbf{v}_k , el cual tiene asociado al valor propio λ_k , entonces obtenemos:

$$\begin{aligned} \|A\mathbf{v}_k\|_p &\leq \|A\|_p \|\mathbf{v}_k\|_p \\ |\lambda_k| \|\mathbf{v}_k\|_p &\leq \|A\|_p \|\mathbf{v}_k\|_p, \end{aligned}$$

cancelando $\|\mathbf{v}_k\|_p$ dado que $\|\mathbf{v}_k\|_p \neq 0$,

$$|\lambda_k| \leq \|A\|_p.$$

Aplicando el operador \max_k en ambos lados de la ecuación obtenemos exactamente la definición del radio espectral al lado izquierdo de la ecuación,

$$\rho(A) = \max_k |\lambda_k| \leq \max_k \|A\|_p = \|A\|_p.$$

Por lo tanto podemos concluir que el radio espectral $\rho(A)$ es una cota inferior a todas las normas matriciales inducidas por normas vectoriales. En particular para el análisis de la convergencia de los métodos iterativos clásicos presentados en esta sección, podemos utilizar este resultado para estudiar la norma matricial de la matriz M en la ecuación (4.11), en este caso si el radio espectral es menor a 1 se observará convergencia del método iterativo analizado. Una posible desventaja de este procedimiento para estudiar la convergencia es el costo computacional asociado a obtener los valores propios. Para lidiar con esta desventaja, se analizan 2 casos aproximaciones interesantes en las próximas 2 secciones.

4.7.6. Matrices diagonal dominante y su convergencia en métodos iterativos clásicos

Def 13. La matriz $A = (a_{ij})$ cuadrada de dimensión $n \times n$ es estrictamente diagonal dominante si para cada $1 \leq i \leq n$, $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$. En otras palabras, es cuando en cada una de sus filas el valor absoluto de su diagonal es mayor estricto que la suma de los valores absolutos de los demás componentes de dicha fila.

Thm 12 (Convergencia de una matriz diagonal dominante). Si una matriz A de dimensión $n \times n$ es estrictamente diagonal dominante se cumple que:

- (1) A es una matriz no singular.
- (2) Los métodos de Jacobi y Gauss-Seidel convergen a una solución única de $A\mathbf{x} = \mathbf{b}$, $\forall \mathbf{b}$ y $\forall \mathbf{x}_0$.

4.7.7. Teorema de los círculos de Gershgorin

Thm 13. Sea A una matriz de $n \times n$. Cada valor propio λ de A pertenece por lo menos a uno de los discos $|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|$.

Ejemplo:

$$A = \begin{bmatrix} 8 & 1 & 0 \\ 1 & 4 & \epsilon \\ 0 & \epsilon & 1 \end{bmatrix}$$

donde $|\epsilon| < 0,1$. Por el Teorema de Gershgorin se obtiene la siguiente relación para los valores propios:

$$\begin{aligned} |\lambda - 8| &\leq 1, \\ |\lambda - 4| &\leq 1 + |\epsilon|, \\ |\lambda - 1| &\leq |\epsilon|, \end{aligned}$$

lo que es consistente con los valores propios: $\lambda_1 \approx 8,23607$, $\lambda_2 \approx 3,76397$, y $\lambda_3 \approx 0,99965$. En la Figura 4.2 se muestra la versión gráfica de este ejemplo.

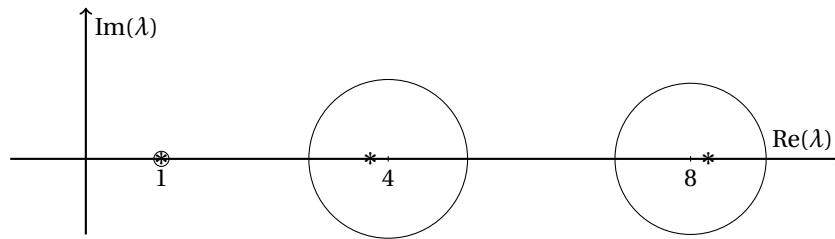


Figura 4.2: Gráfico de los valores propios y los círculos de Gershgorin. En este caso se observa que efectivamente los valores propios se encuentran dentro de los círculos de Gershgorin.

4.7.8. Comentarios varios

Para finalizar esta sección, debemos evaluar la gran pregunta que nos surge: ¿Por qué preferir utilizar métodos iterativos a métodos directos? Como ya lo dejamos entrever anteriormente, la razón está en el costo computacional. Recuerde que para métodos directos discutidos tenemos los siguientes costos:

- Eliminación Gaussiana: $\sim \frac{2}{3}n^3$.
- PALU: $\sim \frac{2}{3}n^3$.
- Cholesky¹⁷: $\sim \frac{1}{3}n^3$.

Sin embargo, los métodos iterativos tienen un costo computacional de: $I \cdot n^2$, donde I corresponde al número de iteraciones a realizar¹⁸. Donde la componente cuadrática se obtiene principalmente por el producto matriz-vector involucrado en cada iteración, además del posible backward o forward substitution requerido. Para el caso donde $I \ll n$, podemos concluir que los métodos iterativos son mucho más rápidos que los directos. Además, para el caso cuando tenemos secuencias de sistemas de ecuaciones lineales donde A y/o \mathbf{b} van cambiando, podemos utilizar la aproximación obtenida en una iteración anterior como dato inicial para la siguiente iteración, es decir como \mathbf{x}_0 .

Hasta el momento hemos considerado que las matrices con las cuales estamos trabajando son *matrices densas*, es decir, la mayoría de sus coeficientes son no nulos. En ese caso determinamos anteriormente que el número de operaciones elementales requeridas para multiplicar una matriz con un vector es $2n^2$. Sin embargo también existen *matrices dispersas (sparse)*¹⁹. Este tipo de matrices se caracteriza porque tiene muy pocos coeficientes distintos a 0 por fila o columna. Este significa un cambio muy importante, principalmente en el número de operaciones elementales requeridas para obtener el producto matriz vector. En este caso el producto matriz vector requiere solo a lo más kn operaciones elementales, donde k es la cota superior de coeficientes no nulos por fila. Notar que no solo el número de operaciones elementales baja al trabajar con matrices dispersas, también la memoria requerida. Ya que se pueden diseñar estructuras de datos adecuadas para solo almacenar a lo más k coeficientes por fila, lo que implica que los requerimientos de memoria también son $\mathcal{O}(n)$. Esto da paso a la posibilidad de obtener algoritmos muy eficientes para resolver sistemas de ecuaciones lineales con matrices dispersas! Una observación interesante es que en algunas ocasiones los métodos directos como PALU también pueden tomar ventaja de los patrones de las matrices dispersas, sin embargo también puede ocurrir que aunque la matriz A sea dispersa las matrices L y U de la factorización PALU no mantengan ese patrón, es decir sean matrices densas²⁰.

A continuación se presenta un ejemplo numérico para los algoritmos discutidos en esta sección.

$$\begin{aligned} 1u + 3v &= -1, \\ 5u + 4v &= 6. \end{aligned}$$

El cual se puede re-escribir matricialmente de la siguiente forma,

$$\begin{bmatrix} 5 & 4 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 6 \\ -1 \end{bmatrix}.$$

Notar que se alternaron las filas de la matriz y el lado derecho para que la matriz sea estrictamente diagonal dominante, esto no es siempre posible, pero si fuera posible, es una muy buena idea hacerlo! De la representación

¹⁷Se incluye por completitud

¹⁸No confundir con la matriz identidad!

¹⁹También denotadas como matrices ralas.

²⁰Las matrices triangulares superiores o triangulares inferiores tiene la mitad de los coeficientes aproximadamente igual a cero, pero no se consideran matrices dispersas ya que de todos modos necesitan almacenar aproximadamente $\frac{n^2}{2}$ coeficientes, lo cual sigue siendo $\mathcal{O}(n^2)$.

obtenemos L , D y U ,

$$L = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix},$$

$$D = \begin{bmatrix} 5 & 0 \\ 0 & 3 \end{bmatrix},$$

$$U = \begin{bmatrix} 0 & 4 \\ 0 & 0 \end{bmatrix}.$$

Por simplicidad consideraremos $\mathbf{x}_0 = [0 \ 0]^T$ en los 3 algoritmos²¹. Solo se presentarán las expresiones matriciales de las iteraciones pero se invita a ejecutarlas en el jupyter notebook respectivo²².

- Método de Jacobi:

$$\begin{aligned} \mathbf{x}_{n+1} &= D^{-1} (\mathbf{b} - (L + U) \cdot \mathbf{x}_n) \\ &= \begin{bmatrix} \frac{1}{5} & 0 \\ 0 & \frac{1}{3} \end{bmatrix} \left(\begin{bmatrix} 6 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 4 & 0 \end{bmatrix} \mathbf{x}_n \right). \end{aligned}$$

- Método de Gauss-Seidel:

$$\begin{aligned} \mathbf{x}_{n+1} &= (L + D)^{-1} (\mathbf{b} - U \mathbf{x}_n) \\ &= \begin{bmatrix} 5 & 0 \\ 1 & 3 \end{bmatrix}^{-1} \left(\begin{bmatrix} 6 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 & 4 \\ 0 & 0 \end{bmatrix} \mathbf{x}_n \right). \end{aligned}$$

- SOR(ω)²³:

$$\begin{aligned} \mathbf{x}_{n+1} &= (\omega L + D)^{-1} (\omega \mathbf{b} + [(1 - \omega) D - \omega U] \mathbf{x}_n) \\ &= \begin{bmatrix} 5 & 0 \\ \omega & 3 \end{bmatrix}^{-1} \left(\begin{bmatrix} 6\omega \\ -\omega \end{bmatrix} + \begin{bmatrix} (1 - \omega)5 & -4\omega \\ 0 & (1 - \omega)3 \end{bmatrix} \mathbf{x}_n \right) \end{aligned}$$

4.8. ¿Existen algoritmos especializados para matrices particulares?

Para finalizar este capítulo es relevante destacar que los algoritmos discutidos anteriormente no toman ventajas de las propiedades de la matriz asociada al sistema de ecuaciones lineales a resolver. Por ejemplo, un caso particular pero muy recurrente es cuando la matriz asociada al sistema de ecuaciones lineales $A\mathbf{x} = \mathbf{b}$, es decir $A \in \mathbb{R}^{n \times n}$ es simétrica ($A = A^T$) y definida positiva (todos sus valores propios son reales y positivos). Este tipo de matrices surgen, por ejemplo, en problemas de mínimos cuadrados, los cuales veremos en el capítulo 6. Para este subconjunto de sistemas de ecuaciones lineales uno puede diseñar algoritmos especializados que pueden obtener la solución del sistema de ecuaciones lineales utilizando menos operaciones elementales y/o utilizando menos memoria. Para el lector interesado en este tema se sugiere ver el apéndice D donde se discute la Factorización de Cholesky, el método del Gradiente Descendente (muy utilizado en Redes Neuronales y Ciencia de Datos) y el majestuoso Gradiente Conjugado.

Antes de finalizar, queremos señalar que aún no hemos terminado de discutir sobre algoritmos para resolver sistemas de ecuaciones lineales. En el capítulo 7 aprenderemos de GMRes. La razón de no haberlo incluido en este capítulo es porque primero requiere que aprendamos de problemas de Mínimos Cuadrados, lo cual haremos en capítulo 6. GMRes es un pariente del majestuoso Gradiente Conjugado, pero no requiere que la matriz asociada al sistema de ecuaciones lineales tenga propiedades especiales!

²¹Se incluirá SOR(ω) por completitud.

²²¿Cuál algoritmo cree usted que requiere menos iteraciones?

²³Se incluye por completitud del análisis.

Capítulo 5

Interpolación Polinomial en 1D

En este capítulo estudiaremos distintos algoritmos y resultados teóricos, con grandes implicancias prácticas, asociados a interpolación polinomial en una dimensión. Lo primero que haremos es formalizar la noción de interpolación con la siguiente definición:

Def 14 (Interpolación en 1D). *Una función $y = p(x)$ interpola los datos $(x_1, y_1), \dots, (x_n, y_n)$ si $p(x_i) = y_i$ para cada $i \in \{1, \dots, n\}$.*

En general, la función $p(x)$ con la cual trabajaremos serán polinomios, pero podrían tener otra estructura. Dependiendo de la definición de la estructura de la función $p(x)$, uno puede construir algoritmos especializados para su construcción y uso.

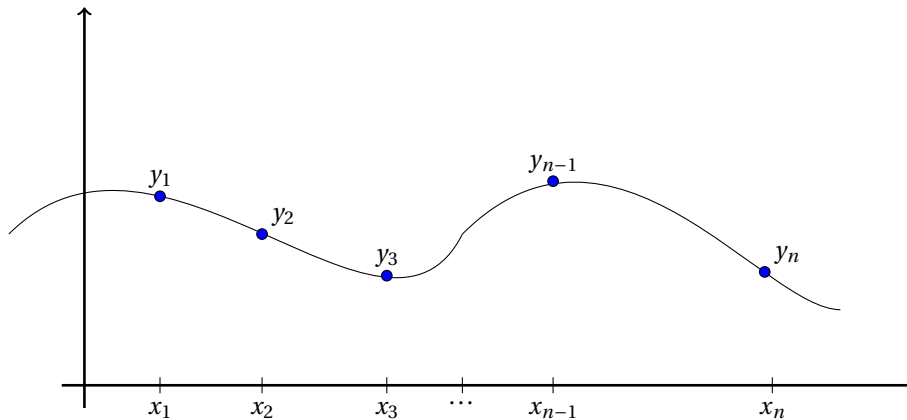


Figura 5.1: Sketch de una función interpoladora que asegura que al evaluar el interpolador $p(x)$ en los puntos de colocación x_i se obtiene $p(x_i) = y_i$.

Antes de llegar a interpolar n puntos, partamos con un ejemplo con 2 puntos primero. Entonces la pregunta es: ¿Cuál es el polinomio de *grado mínimo* que interpola (x_1, y_1) y (x_2, y_2) ? Aquí se ha destacado un término clave, el cual es “grado mínimo”. La razón de destacar ese término es debido a que uno en realidad puede construir infinitos polinomios que interpolen esos puntos, sin embargo existe solo uno que es de grado mínimo. En la Figura 5.2 se presentan los 2 puntos antes mencionados de forma ilustrativa. En las siguientes secciones veremos como

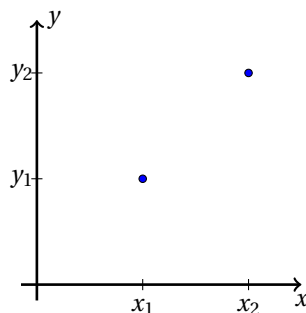


Figura 5.2: Sketch de 2 puntos a interpolar.

efectivamente construir el polinomio de grado mínimo que interpole (x_1, y_1) y (x_2, y_2) .

¡MUY IMPORTANTE!

Inicialmente uno podría considerar que podría ser un problema desafiante encontrar el polinomio de grado mínimo, aunque no es trivial, es en realidad muy conveniente que solo exista uno. La conveniencia es que su construcción no dependerá del algoritmo elegido! La expresión algebraica podría ser distinta, pero lo que representa es lo mismo.

Antes de avanzar a las siguientes secciones se invita a que usted busque la forma de construir un polinomio de grado 1 y un polinomio de grado 2 que interpole la data anterior.

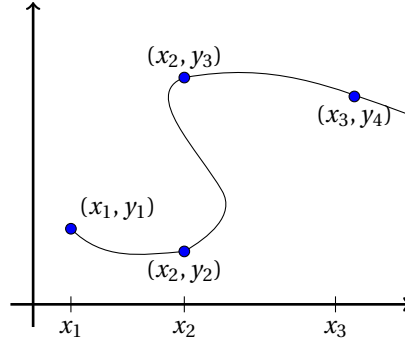
Para finalizar esta introducción, presentamos el Teorema principal de interpolación polinomial:

Thm 14 (Unicidad de la interpolación polinomial). *Sea $(x_1, y_1), \dots, (x_n, y_n)$ n puntos en el plano \mathbb{R}^2 con distintos x_i , entonces existe uno y sólo un polinomio $p(x)$ de grado $(n - 1)$ o menor que satisface la siguiente ecuación: $p(x_i) = y_i$ para $i \in \{1, 2, \dots, n\}$.*

Este teorema tiene sutilezas muy interesantes que iremos analizando en las siguientes secciones, preliminarmente se destacan los siguientes puntos:

- ...distintos x_i ...
- ...un polinomio $p(x)$ de grado $(n - 1)$ o **menor**...

Veremos la importancia algebraica del primer punto al analizar el determinante de la matriz de Vandermonde y la importancia del segundo punto la veremos de forma práctica en el ejemplo que discutiremos. Respecto al primer punto, se puede hacer un gráfico conveniente para destacar si es que existieran puntos x_i repetidos. En la Figura 5.3 se muestra un diagrama de que es lo que ocurriría si un punto de interpolación estuviera repetido. En este caso se observa que la curva construida ya deja de ser una función ya que es multivaluada, es decir que para x_2 tiene por lo menos 3 posibles valores. La teoría de interpolación polinomial no puede utilizarse directamente en esta situación, sin embargo aún sería posible interpolar la data si se considera una definición paramétrica para cada variable. Es decir se define una interpolación polinomial para la variable $x(s)$ y otra para la variable $y(s)$, donde s sería la variable paramétrica independiente.

Figura 5.3: Sketch de una curva con puntos x_i repetidos.

5.1. Matriz de Vandermonde

El primer algoritmo que uno naturalmente construye está relacionado a la matriz de Vandermonde. Para explicar mejor este procedimiento procederemos a analizar el caso inicial cuando uno quiere encontrar el polinomio de grado mínimo para interpolar los puntos: (x_1, y_1) y (x_2, y_2) . En este caso uno propone un polinomio de grado 1 de la siguiente forma:

$$p(x) = a_0 + a_1 x.$$

La razón principal de por que se elige un polinomio de grado 1 es porque tiene 2 grados de libertad y conocemos que hay 2 condiciones que debemos satisfacer¹. Entonces si imponemos las 2 condiciones a la estructura polinomial propuesta obtenemos,

$$p(x_1) = a_0 + a_1 x_1 = y_1,$$

$$p(x_2) = a_0 + a_1 x_2 = y_2.$$

Re-escribiéndolo matricialmente obtenemos,

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.$$

Para resolver este sistema de ecuaciones podemos utilizar alguno de los algoritmos anteriormente discutidos. Siguiendo el mismo análisis podemos construir el mismo sistema de ecuaciones lineales para el caso de interpolar 3 puntos, es decir obtendríamos lo siguiente,

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}.$$

De lo cual podemos notar el patrón que sigue la matriz asociada al aumentar la cantidad de puntos, la matriz generada es la *matriz de Vandermonde* y tiene la siguiente forma general para cuando se interpolan n puntos,

$$V = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-2} & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-2} & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-2} & x_{n-1}^{n-1} \\ 1 & x_n & x_n^2 & \dots & x_n^{n-2} & x_n^{n-1} \end{bmatrix}.$$

¹Las condiciones son $p(x_1) = y_1$ y $p(x_2) = y_2$.

Es decir la fila asociada al punto x_i de la matriz V tiene la siguiente forma: $[1, x_i^1, x_i^2, \dots, x_i^{n-1}]$, para algún $i \in \{1, 2, \dots, n\}$. Notar que no es necesario que los x_i estén en algún orden, sin embargo sí es importante que sean todos distintos, de otra forma existirían 2 filas iguales y la matriz sería singular². Para destacar la importancia de que no existan valores x_i repetidos en la construcción de la matriz de Vandermonde se conoce la forma explícita del determinante, el cual es el siguiente,

$$\det(V) = \prod_{1 \leq i < j \leq n} (x_j - x_i).$$

Del cual notamos que si existiera algún x_i repetido, es decir $x_i = x_j$, obtendríamos un determinante igual a 0. Lo cual es algo que debemos evitar si queremos tener una solución única del sistema de ecuaciones lineales asociado.

Por último, y no menos importante, es que, en general, la matriz de Vandermonde es muy mal condicionada, es decir $\kappa(V) \gg 1$. Por lo cual no es un camino recomendado cuando la cantidad de puntos a interpolar crece mucho. Entonces, ¿Cómo se puede construir un polinomio interpolador con muchos puntos? Para responder esta pregunta, aparece el método de Lagrange que se presentará en la próxima sección.

5.2. Interpolación de Lagrange

En esta sección veremos el método de interpolación polinomial de Lagrange. La gran ventaja de este método respecto a la utilización de la matriz de Vandermonde es que no hay que resolver ningún sistema de ecuaciones lineales. Esto se logra con una definición bien particular pero muy conveniente de la estructura del polinomio interpolador. La estructura del polinomio interpolador para n puntos de interpolación tiene la siguiente forma:

$$\begin{aligned} p(x) &= y_1 L_1(x) + y_2 L_2(x) + \dots + y_n L_n(x) \\ &= \sum_{i=1}^n y_i L_i(x), \end{aligned}$$

donde $L_i(x_i) = 1$ y $L_i(x_j) = 0$ para $i \neq j$ e $i, j \in \{1, 2, \dots, n\}$. Esta definición conveniente nos permite obtener efectivamente y_k cuando uno evalúa $p(x_k)$, es decir,

$$\begin{aligned} p(x_k) &= y_1 \underbrace{L_1(x_k)}_0 + y_2 \underbrace{L_2(x_k)}_0 + \dots + y_k \underbrace{L_k(x_k)}_1 + \dots + y_n \underbrace{L_n(x_k)}_0 \\ &= y_k. \end{aligned}$$

Hasta este punto se podrían considerar *muy conveniente* las propiedades de $L_i(x)$ pero en realidad no es tan complicado construir una expresión polinomial que logre lo solicitado. Para la construcción $L_i(x)$ primero definiremos $l_i(x)$ de la siguiente forma,

$$l_i(x) = \prod_{k=1, k \neq i}^n (x - x_k) = (x - x_1)(x - x_2) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n).$$

Lo que indica esta expresión es simplemente el producto entre todos los términos $(x - x_k)$ exceptuando el término $(x - x_i)$. Así se logra que $l_i(x)$ sea 0 al evaluarlo en cualquier $x_k \neq x_i$. Sin embargo, al evaluarlo en x_i se obtiene $l_i(x_i) = \prod_{k=1, k \neq i}^n (x_i - x_k)$, el cual no es necesariamente 1 y sabemos que no es 0 por construcción. Por lo tanto, aparece naturalmente la definición de $L_i(x)$ como el cociente entre $l_i(x)$ y $l_i(x_i)$ de la siguiente forma,

$$\begin{aligned} L_i(x) &= \frac{l_i(x)}{l_i(x_i)} \\ &= \frac{(x - x_1)(x - x_2) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_1)(x_i - x_2) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}. \end{aligned}$$

²Recuerde que si hay 2 filas iguales significa que son linealmente dependientes, por lo que implica que la matriz es singular, es decir, no existe la inversa. Esto implica que si el sistema de ecuaciones lineales tuviera solución, no sería única.

Con la construcción de $L_i(x)$ finalizamos la descripción de la interpolación de Lagrange. Desde el punto de vista práctico, uno puede pre-calcular, por ejemplo, el denominador de cada $L_i(x)$, ya que es solo un producto de números. Sin embargo el numerador de cada $L_i(x)$ se debe evaluar cada vez que se evalúe el polinomio interpolador $p(x)$, es decir, cada vez que se conozca x . De este análisis surgen las siguientes preguntas³,

- ¿Cuántas operaciones elementales se requieren para pre-calcular los denominadores todos los $L_i(x)$ involucrados al interpolar n puntos?
- ¿Cuántas operaciones elementales se requieren para evaluar el polinomio interpolador de Lagrange con n puntos considerando que ya se pre-calaron los denominadores de cada coeficientes $L_i(x)$?

Retomando el ejemplo de interpolar los puntos (x_1, y_1) y (x_2, y_2) , para el cual se generó la Figura 5.2, podemos ahora construir explícitamente el polinomio de interpolación de Lagrange de la siguiente forma,

$$p(x) = y_1 L_1(x) + y_2 L_2(x),$$

donde $L_1(x) = \frac{(x-x_2)}{(x_1-x_2)}$ y $L_2(x) = \frac{(x-x_1)}{(x_2-x_1)}$. La Figura 5.4 muestra la recta interpolada y cada término de la interpolación de Lagrange.

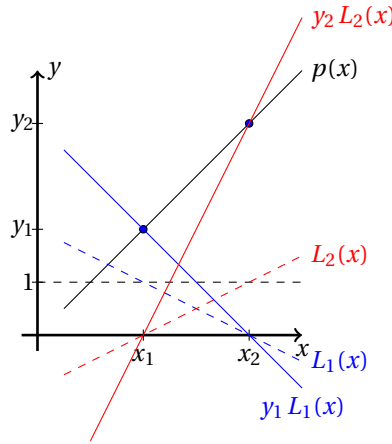


Figura 5.4: Interpolación de 2 puntos y cada uno de los términos de la interpolación de Lagrange.

Por otro lado, a partir de la representación algebraica de la interpolación de Lagrange podemos establecer la conexión con los coeficientes que uno obtendría si es que utiliza la matriz de Vandermonde, es decir los coeficientes de la representación $p(x) = a_0 + a_1 x$. Recuerde que $p(x)$ es único, por lo cual la siguiente identidad es válida,

$$\begin{aligned} a_0 + a_1 x &= y_1 \frac{(x-x_2)}{(x_1-x_2)} + y_2 \frac{(x-x_1)}{(x_2-x_1)} \\ &= \left(\frac{y_1(-x_2)}{(x_1-x_2)} + \frac{y_2(-x_1)}{(x_2-x_1)} \right) + \left(\frac{y_1}{(x_1-x_2)} + \frac{y_2}{(x_2-x_1)} \right) x, \end{aligned}$$

de lo cual podemos concluir que,

$$\begin{aligned} a_0 &= \frac{y_1(-x_2)}{(x_1-x_2)} + \frac{y_2(-x_1)}{(x_2-x_1)}, \\ a_1 &= \frac{y_1}{(x_1-x_2)} + \frac{y_2}{(x_2-x_1)}. \end{aligned}$$

³Aquí lo importante es determinar si se requieren $\mathcal{O}(n)$, $\mathcal{O}(n^2)$, o $\mathcal{O}(n^3)$ operaciones elementales.

Este ejemplo destaca el concepto de unicidad en interpolación polinomial, es decir, en aritmética exacta, todos los algoritmos llegan al mismo polinomio. La diferencia está en la componente computacional asociada!

Otro caso interesante de analizar corresponde al concepto de polinomio minimal. En la Figura 5.5 se presenta

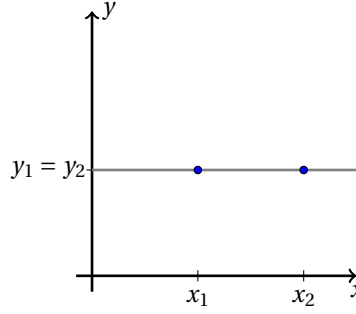


Figura 5.5: Interpolación de 2 puntos donde $y_1 = y_2$.

gráficamente que es lo que significa que $y_1 = y_2$. Lo que se obtiene es básicamente una constante igual a y_1 o y_2 , dado que son iguales, no hay diferencia. Este resultado podemos analizar desde el punto de vista de método de Lagrange y de la matriz de Vandermonde. Primeramente consideremos el método de Lagrange de la siguiente forma,

$$\begin{aligned} p(x) &= y_1 \frac{(x - x_2)}{(x_1 - x_2)} + y_1 \frac{(x - x_1)}{(x_2 - x_1)} \\ &= y_1 \left(\frac{x - x_2}{x_1 - x_2} - \frac{x - x_1}{x_1 - x_2} \right) \\ &= y_1 \left(\frac{x_1 - x_2}{x_1 - x_2} \right) \\ &= y_1. \end{aligned}$$

Entonces efectivamente la interpolación de Lagrange nos entrega el polinomio minimal o de grado mínimo, que en este caso es una constante! Sin embargo se requirió un manejo algebraico para demostrar que efectivamente se obtiene un constante. En la práctica uno no tiene que hacer esta simplificación necesariamente, lo que en principio uno hace es solo evaluar el polinomio de Lagrange directamente, el que, en aritmética exacta, entregará el valor constante.

Ahora, desde el punto de vista de la matriz de Vandermonde uno tendría que resolver el siguiente sistema de ecuaciones lineales,

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_1 \end{bmatrix}.$$

Re-escribiendo el sistema de ecuaciones lineales considerando que el producto matriz-vector es la combinación lineal de las columnas de la matriz de Vandermonde obtenemos,

$$a_0 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + a_1 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = y_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Del cual podemos notar que $a_0 = y_1$ y $a_1 = 0$ resuelven la ecuación vectorial. Notar que esta ecuación vectorial tiene solución única dado que los vectores $[1, 1]^T$ y $[x_1, x_2]^T$ son linealmente independientes. En caso de que x_1 y

x_2 fueran iguales, la solución que se obtendría no sería única y se puede expresar de la siguiente forma,

$$\begin{aligned} a_0 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + a_1 x_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} &= y_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ (a_0 + a_1 x_1) \begin{bmatrix} 1 \\ 1 \end{bmatrix} &= y_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \end{aligned}$$

por lo tanto tenemos infinitas soluciones para a_0 y a_1 que satisfacen la ecuación $a_0 + a_1 x_1 = y_1$. Nuevamente se destaca la importancia de que $x_1 \neq x_2$, es decir, que los nodos de interpolación sean distintos.

5.3. Interpolación Baricéntrica

La [interpolación baricéntrica](#) es otro algoritmo para realizar interpolación polinomial. En este caso, es posible construir el algoritmo de interpolación baricéntrica desde la interpolación de Lagrange. Lo interesante de este algoritmo es que reduce significativamente el número de operaciones elementales requeridas para evaluar el polinomio construido y tiene la posibilidad de reducir significativamente también el número de operaciones elementales al construir el polinomio interpolador, si es que se hace sobre un conjunto de nodos regulares.

Para construir el algoritmo de interpolación polinomial, primero definiremos $l(x)$ de la siguiente forma,

$$l(x) = \prod_{i=1}^n (x - x_i).$$

Notar que es distinto a $l_i(x) = \prod_{k=1, k \neq i}^n (x - x_k)$ utilizado anteriormente en la interpolación de Lagrange. La diferencia es que $l(x)$ incluye todos los términos en el producto. Esta definición nos permite re-escribir $l_i(x)$ de la siguiente forma,

$$l_i(x) = \frac{l(x)}{(x - x_i)}.$$

Adicionalmente, podemos obtener el termino asociado al denominador de $L_i(x)$ de la siguiente forma,

$$w_i = \frac{1}{l_i(x_i)} = \frac{1}{l'(x_i)}.$$

Esto nos permite obtener $L_i(x)$ de la siguiente forma,

$$L_i(x) = \frac{l(x)}{(x - x_i)} w_i.$$

Por lo tanto, podemos re-escribir la interpolación de Lagrange de la siguiente forma:

$$\begin{aligned} p(x) &= \sum_{i=1}^n y_i L_i(x) \\ &= \sum_{i=1}^n y_i \frac{l(x)}{(x - x_i)} w_i \\ &= l(x) \sum_{i=1}^n y_i \frac{w_i}{(x - x_i)}. \end{aligned} \tag{5.1}$$

Lo cual ya es un éxito, sin embargo se puede mejorar el resultado anterior. Para llegar a la interpolación baricéntrica necesitamos re-escribir el termino $l(x)$. Lo interesante es que podemos utilizar el resultado recién obtenido, es decir la ecuación (5.1), para interpolar la constante 1. Al interpolar una constante con un polinomio sabemos que

se podrá reproducir exactamente por el teorema de unicidad de interpolación polinomial, es decir obtenemos la siguiente identidad,

$$\begin{aligned} p(x) &= l(x) \sum_{i=1}^n \underbrace{y_i}_{1} \frac{w_i}{(x-x_i)} \\ &= l(x) \sum_{i=1}^n \frac{w_i}{(x-x_i)} \\ &= 1, \end{aligned}$$

lo que nos entrega la siguiente identidad,

$$1 = l(x) \sum_{i=1}^n \frac{w_i}{(x-x_i)}.$$

De la cual podemos despejar $l(x)$,

$$l(x) = \frac{1}{\sum_{i=1}^n \frac{w_i}{(x-x_i)}}.$$

Reemplazando esta definición de $l(x)$ en la ecuación (5.1) obtenemos,

$$\begin{aligned} p(x) &= l(x) \sum_{i=1}^n y_i \frac{w_i}{(x-x_i)} \\ &= \frac{\sum_{i=1}^n y_i \frac{w_i}{(x-x_i)}}{\sum_{i=1}^n \frac{w_i}{(x-x_i)}}. \end{aligned} \tag{5.2}$$

¡Esta última expresión corresponde a la interpolación baricéntrica! Nuevamente es muy importante destacar que la interpolación baricéntrica es solamente otro algoritmo que obtiene el mismo polinomio que uno obtendría con la matriz de Vandermonde o la interpolación de Lagrange en **aritmética exacta**, la gran diferencia nuevamente está asociada a la computación requerida!

¡MUY IMPORTANTE!

Aquí hay aclarar que el cociente de sumatorias no se puede simplificar en una sola sumatoria, es decir,

$$\frac{\sum_{i=1}^n y_i \frac{w_i}{(x-x_i)}}{\sum_{i=1}^n \frac{w_i}{(x-x_i)}} \neq \sum_{i=1}^n \frac{y_i \frac{w_i}{(x-x_i)}}{\frac{w_i}{(x-x_i)}}.$$

¡Así que por favor no cometer ese error!

Comentario al margen

Para valorar lo interesante que es la interpolación baricéntrica considere las siguientes preguntas:

- ¿Es realmente una interpolación polinomial la ecuación (5.2)?^a
- ¿Cuál es el costo de evaluar $p(x)$ en ecuación (5.2) si conocemos los coeficientes w_i ?^b
- ¿Cuál es el costo de obtener los coeficientes w_i ?^c
- ¿Qué ocurre con la evaluación justo en un punto de interpolación x_k ?^d

^aSí.

^b $\mathcal{O}(n)$, lo cual es maravilloso!

^c $\mathcal{O}(n^2)$, lo que implica que en principio el algoritmo es $\mathcal{O}(n^2)$, sin embargo cuando los puntos de interpolación siguen un patrón regular, como en el caso de los puntos de Chebyshev que veremos pronto, los conocemos algebraicamente! Es decir, hay un algoritmo $\mathcal{O}(n)$ que los obtiene.

^dNo se evalúa la expresión porque se sabe que es un punto de interpolación y debe dar y_k . Matemáticamente lo que uno debe obtener sería el límite en ese caso, pero como ya sabemos el resultado del límite, no es necesario obtenerlo!

5.4. EXTRA: Diferencias divididas de Newton

Recuerde:

$$\begin{aligned} p(x) &= 1 + 2x + 3x^2 + 4x^3 \sim \mathcal{O}(n^2) \\ &= 1 + x(2 + x(3 + 4x)) \sim \mathcal{O}(n) \end{aligned}$$

↑

Mín número de operaciones

$$P_{n-1}(x) = \sum_{i=0}^{n-1} a_i x^i \iff \text{Matriz de Vandermonde}$$

$$P_{n-1}(x) = \sum_{i=1}^n y_i L_i(x)$$

Def 15. Denotemos que $\underbrace{f[x_1, x_2 \dots x_{n-1}, x_n]}_{Y_i}$ es el coeficiente del término x^{n-1} en el único polinomio que interpola $(x_i, \underbrace{f(x_i)}_{Y_i})$.

$$P_{n-1}(x) = \sum_{i=0}^{n-1} a_i x^i = a_0 + a_1 x + \dots + \underline{a_{n-1}} x^{n-1}.$$

En el caso de diferencias divididas de Newton, el polinomio interpolador se define como:

$$\begin{aligned} \Rightarrow P_{n-1}(x) = & f[x_1] + f[x_1 \ x_2](x - x_1) \\ & + f[x_1 \ x_2 \ x_3](x - x_1)(x - x_2) \\ & + f[x_1 \ x_2 \ x_3 \ x_4](x - x_1)(x - x_2)(x - x_3) \\ & \vdots \\ & + f[x_1 \dots x_n](x - x_1) \dots (x - x_{n-1}). \end{aligned}$$

5.4.1. Ejemplo

Use las diferencias divididas para encontrar el polinomio de interpolación que pasa a través de los puntos: $(x_1, y_1), (x_2, y_2), (x_3, y_3)$.

Recuerde: $f[x_i] = y_i$

x_1	y_1	$= f[x_1]$	$f[x_1 \ x_2] = \frac{f[x_2] - f[x_1]}{x_2 - x_1}$	$f[x_1 \ x_2 \ x_3] = \frac{f[x_2 \ x_3] - f[x_1 \ x_2]}{x_3 - x_1}$
x_2	y_2	$= f[x_2]$	$f[x_2 \ x_3] = \frac{f[x_3] - f[x_2]}{x_3 - x_2}$	
x_3	y_3	$= f[x_3]$		

$$\Rightarrow P_2(x) = f[x_1] + f[x_1 \ x_2](x - x_1) + f[x_1 \ x_2 \ x_3](x - x_1)(x - x_2)$$

5.4.2. Algoritmicamente

```

1 for j in (n):
2     f[xj] = yj
3 for i in (2, n):
4     for j in (n+1-i):
5         f[xj...x(j+1-i)] =  $\frac{f[x_{(j+1)}...x_{(j+1-i)}] - f[x_j...x_{(j+1-2)}]}{x_{(j+1-i)} - x_j}$ 

```

El polinomio de Newton es de la forma:

$$P_{n-1}(x) = \sum_{i=1}^n f[x_1 \dots x_i] \prod_{j=1}^{i-1} (x - x_j)$$

5.5. Ejemplo numérico de interpolación polinomial

Use la matriz de Vandermonde, la interpolación de Lagrange, interpolación Baricéntrica y Diferencias Divididas de Newton⁴ para encontrar el polinomio de interpolación que pasa a través de los puntos: (0, 1), (2, 3), y (3, 0).

⁴Se incluye por completitud.

La construcción del polinomio de interpolación por medio de la matriz de Vandermonde requiere que resolvamos el siguiente sistema de ecuaciones lineales:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix},$$

donde la solución es:

$$\begin{aligned} a_0 &= 1, \\ a_1 &= \frac{11}{3}, \\ a_2 &= -\frac{4}{3}. \end{aligned}$$

Por lo tanto el polinomio interpolador obtenido por medio de la matriz de Vandermonde es:

$$p(x) = 1 + \frac{11}{3}x - \frac{4}{3}x^2.$$

Ahora, para construir el polinomio de interpolación por medio de la interpolación de Lagrange primero obtendremos $L_1(x)$, $L_2(x)$, y $L_3(x)$:

$$\begin{aligned} L_1(x) &= \frac{(x-2)(x-3)}{(0-2)(0-3)}, \\ L_2(x) &= \frac{(x-0)(x-3)}{(2-0)(2-3)}, \\ L_3(x) &= \frac{(x-0)(x-2)}{(3-0)(3-2)}. \end{aligned}$$

Por lo que el polinomio de interpolación queda definido de la siguiente forma,

$$\begin{aligned} p(x) &= y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x) \\ &= 1 \frac{(x-2)(x-3)}{6} + 3 \frac{(x-0)(x-3)}{-2} + 0 L_3(x) \\ &= 1 \frac{(x-2)(x-3)}{6} + 3 \frac{(x-0)(x-3)}{-2}. \end{aligned}$$

Note que no incluimos la expresión explícita de $L_3(x)$ en la expresión anterior porque al estar multiplicado por $y_3 = 0$ se anulará. Recuerde que estamos utilizando $p(x)$ para ambos polinomios porque son efectivamente los mismos! Se sugiere expandir el polinomio obtenido por la interpolación de Lagrange para comprobar que se obtiene la misma expresión.

Para el caso de la interpolación Baricéntrica debemos primero obtener w_1 , w_2 , y w_3 ,

$$\begin{aligned} w_1 &= \frac{1}{(0-2)(0-3)} = \frac{1}{6}, \\ w_2 &= \frac{1}{(2-0)(2-3)} = -\frac{1}{2}, \\ w_3 &= \frac{1}{(3-0)(3-2)} = \frac{1}{3}. \end{aligned}$$

Por lo tanto el polinomio interpolador es:

$$\begin{aligned}
 p(x) &= \frac{y_1 \frac{w_1}{x-x_1} + y_2 \frac{w_2}{x-x_2} + y_3 \frac{w_3}{x-x_3}}{\frac{w_1}{x-x_1} + \frac{w_2}{x-x_2} + \frac{w_3}{x-x_3}} \\
 &= \frac{1 \frac{w_1}{x-0} + 3 \frac{w_2}{x-2} + 0 \frac{w_3}{x-3}}{\frac{w_1}{x-0} + \frac{w_2}{x-2} + \frac{w_3}{x-3}} \\
 &= \frac{\frac{1/6}{x-0} - \frac{3/2}{x-2}}{\frac{1/6}{x-0} - \frac{1/2}{x-2} + \frac{1/3}{x-3}}.
 \end{aligned}$$

Por completitud, se incluye el polinomio de interpolación que se obtendría con el algoritmo de las diferencias divididas de Newton. Para lo cual primero debemos obtener la tabla de diferencias divididas:

$x_0 = 0$	$f[x_0] = y_0 = 1$		
$x_1 = 2$	$f[x_1] = y_1 = 3$	$f[x_0, x_1] = 1$	
$x_2 = 3$	$f[x_2] = y_2 = 0$	$f[x_1, x_2] = -3$	$f[x_0, x_1, x_2] = -\frac{4}{3}$

$$\begin{aligned}
 p(x) &= f[x_0] + f[x_0, x_1](x-x_0) + f[x_0, x_1, x_2](x-x_0)(x-x_1) \\
 &= 1 + 1(x-x_0) + \left(-\frac{4}{3}\right)(x-x_0)(x-x_1) \\
 &= 1 + 1(x-0) - \frac{4}{3}(x-0)(x-2).
 \end{aligned}$$

5.6. De interpolación polinomial de puntos arbitrarios a interpolación polinomial de funciones

Hasta ahora hemos aprendido a interpolar polinomialmente un conjunto de puntos $(x_1, y_1), \dots, (x_n, y_n)$, donde sabemos que se necesita que todos los x_i sean distintos⁵. Para construir el polinomio hemos discutido tres algoritmos principalmente: Matriz de Vandermonde, Interpolación de Lagrange e Interpolación Baricéntrica, los cuales seguiremos usando. La diferencia ahora es que nos enfocaremos en interpolar funciones conocidas, $f(x)$, para obtener una representación de estas por medio de un polinomio, $p(x)$, principalmente en un intervalo finito, por ejemplo $[a, b] \in \mathbb{R}$. Entonces uno podría hacerse las siguientes preguntas:

- ¿Por qué será necesario encontrar un polinomio $p(x)$ que aproxime $f(x)$ en un intervalo finito $[a, b]$?
- ¿Por qué no ocupo directamente la función $f(x)$ cuando necesite evaluarla en un punto $x \in [a, b]$?
- ¿Qué se gana al aproximar la función por un polinomio?
- ¿Qué se pierde al aproximar la función por un polinomio?

⁵Si es que hubiera un valor de x_i repetido, digamos $x_k = x_l$, sabemos que no podemos utilizar ambos puntos en la interpolación polinomial porque generará una matriz de Vandermonde singular. Sin embargo, si sabemos adicionalmente que $y_k = y_l$, entonces podríamos solo usar uno de ellos y el otro eliminarlo de la lista de puntos! De esta forma no obtendríamos que el determinante de la matriz de Vandermonde asociada sea 0.

Para responder inicialmente estas preguntas se presenta a continuación una breve lista de funciones especiales que uno podría considerar “costosas” de evaluar computacionalmente, sin embargo existen muchas más:

$$\text{Función Gamma}^6: \quad \Gamma(x) = \int_0^\infty y^{x-1} \exp(-y) dy, \quad \operatorname{Re}(x) > 0.$$

$$\text{Función de Airy}^7: \quad \operatorname{Ai}(x) = \frac{1}{\pi} \int_0^\infty \cos\left(\frac{t^3}{3} + xt\right) dt.$$

$$\text{Función de Bessel del primer tipo}^8: \quad J_n(x) = \frac{1}{\pi} \int_0^\pi \cos(n\tau - x \sin(\tau)) d\tau, \quad n \in \mathbb{N}.$$

Entonces, si quisiéramos evaluar cada una de estas funciones un valor de $x \in [a, b]$ tendríamos que calcular una integral⁹. Más aún, si necesitáramos evaluar en muchas veces alguna de estas funciones, rápidamente nos enfrentaríamos al problema del tiempo de computación requerido para realizar la tarea. Por lo que naturalmente surge la pregunta, ¿Se podría acelerar la computación?, la respuesta es sí, y lo podríamos acelerar con una “adecuada”¹⁰ interpolación polinomial.

Para finalizar, procederemos a responder las preguntas antes planteadas:

- ¿Por qué será necesario encontrar un polinomio $p(x)$ que aproxime $f(x)$ en un intervalo finito $[a, b]$? Porque puede ayudarnos a reducir los tiempos de computación requeridos de evaluar $f(x)$ en varios puntos,
- ¿Por qué no ocupo directamente la función $f(x)$ cuando necesite evaluarla en un punto $x \in [a, b]$? Porque sería muy costoso computacionalmente, se podría hacer, pero habría que esperar el tiempo que sea necesario.
- ¿Qué se gana al aproximar la función por un polinomio? Tiene el potencial de entregar el mismo resultado numérico que la evaluación de la función original pero en menor tiempo.
- ¿Qué se pierde al aproximar la función por un polinomio? Se pierde el valor “exacto” sin embargo recuerde que en general trabajamos en double precision por lo que no habrá diferencia práctica entre la función original y una muy buena aproximación numérica.

5.7. Error de Interpolación y Fenómeno de Runge

A continuación presentamos un teorema crucial en interpolación polinomial, el teorema de error de interpolación:

Thm 15 (Error de interpolación). *Asuma que $p(x)$ es el polinomio interpolador (de grado $n - 1$ o menor) que ajusta n puntos $(x_1, y_1), \dots, (x_n, y_n)$. El error de interpolación es,*

$$f(x) - p(x) = \frac{(x - x_1)(x - x_2) \dots (x - x_n)}{n!} f^{(n)}(c),$$

donde c está entre el menor y el mayor de los números x, x_1, \dots, x_n .

El error de interpolación¹¹ tiene 3 componentes claves:

⁶En este caso se conoce que $\Gamma(n) = (n - 1)!$ cuando $n \in \mathbb{N}$, por ejemplo $\Gamma(4) = 3! = 3 \cdot 2 \cdot 1 = 6$. Ver https://en.wikipedia.org/wiki/Gamma_function.

⁷Ver https://en.wikipedia.org/wiki/Airy_function

⁸Ver https://en.wikipedia.org/wiki/Bessel_function

⁹En capítulo 8 veremos como aproximar integrales numéricamente!

¹⁰Este se responderá en la siguiente sección!

¹¹Se recomienda comparar este teorema con el teorema de Taylor con residuo 8. En el caso de la aproximación de Taylor el error se reduce localmente en el punto de la expansión, en este caso, el objetivo es reducir el error en un intervalo.

- $n!$: Este término nos induce preliminarmente en la tentativa dirección que el error de interpolación, es decir la diferencia entre $f(x)$ y $p(x)$ se reduce a medida que n aumentan. Esto es parcialmente cierto dado que para que esto ocurra, las otras componentes deben estar acotadas o crecer más lento que $n!$. El único grado de libertad que nos entrega este término es aumentar n .
- $f^{(n)}(c)$: Este término requiere la computación de la n -ésima derivada de $f(x)$ y evaluarla en un punto c desconocido. Solo conocemos que c está entre el menor y el mayor de los números x, x_1, \dots, x_n . Desde el punto de vista de obtener una cota para $f^{(n)}(c)$, nos interesaría obtener el mayor valor de $|f^{(n)}(x)|$ en el intervalo definido por $[\min(x, x_1, \dots, x_n), \max(x, x_1, \dots, x_n)]$, de esta forma se destaca que no es crítico obtener c , sino un valor para acotar $|f^{(n)}(x)|$. Este término no nos entrega ningún grado de libertad, solo dependemos de la función que estamos interpolando.
- $(x-x_1)(x-x_2)\dots(x-x_n)$: Finalmente llegamos al término crítico que estudiaremos profundamente. La importancia de este término recae en la oportunidad que nos entrega, es decir, destaca que el error de interpolación depende de los puntos utilizados para interpolar. Entonces, surgen las siguientes preguntas ¿es conveniente utilizar puntos equiespaciados para interpolar? ¿existirán otros puntos para reducir el impacto de este término en el error de interpolación polinomial? Este último término es el que nos entrega más grados de libertad, ya que nos permite definir, posiblemente, mejores puntos para interpolar.

En resumen, de los tres términos analizados anteriormente, el único que nos entrega la posibilidad de *mejorar* la interpolación es el tercero.

Por completitud, escribiremos la expresión que nos entrega el peor caso del error de interpolación en el intervalo $\Omega = [\min_i x_i, \max_i x_i]$, donde x_i , para $i \in \{1, 2, \dots, n\}$, son los puntos de interpolación,

$$\begin{aligned} E_{\max} &= \max_{x \in \Omega} |f(x) - p(x)| \\ &= \max_{x \in \Omega} \frac{|(x-x_1)(x-x_2)\dots(x-x_n)|}{n!} |f^{(n)}(c)| \\ &\leq \frac{\max_{x \in \Omega} |(x-x_1)(x-x_2)\dots(x-x_n)|}{n!} \max_{x \in \Omega} |f^{(n)}(x)|. \end{aligned}$$

Desde el punto de vista computacional, nos interesa obtener el mínimo valor de E_{\max} para el menor n posible. E_{\max} pequeño implica que los valores de $f(x)$ son muy cercanos a $p(x)$, y un valor de n pequeño implica que el costo computacional debiera ser bajo.

Antes de proceder en analizar el término $(x-x_1)(x-x_2)\dots(x-x_n)$, consideremos el caso en que queremos construir un polinomio interpolador en 9 puntos equiespaciados en el intervalo $[0, 1]$. Es decir queremos interpolar los siguientes puntos: $\{(0, 0), (1/8, 0), (2/8, 0), (3/8, 0), (4/8, 1), (5/8, 0), (6/8, 0), (7/8, 0), (8/8, 0)\}$ ¹². En la Figura 5.6 se presenta en rojo los puntos de interpolación y en azul lo que obtiene el polinomio interpolador. Se muestra en azul como oscila la evaluación del polinomio entre puntos de interpolación en los extremos del intervalo, esto se hace más significativo al aumentar la cantidad de puntos a interpolar. Notar que las oscilaciones no dependen de la data interpolada, sino que depende del uso de puntos equiespaciados en la interpolación polinomial. Para reducir este fenómeno y lograr reducir el error de interpolación significativamente se presenta en la siguiente sección los puntos de Chebyshev. Es importante destacar que tanto para realizar interpolación con puntos equiespaciados o puntos de Chebyshev u otros puntos, los algoritmos de construcción del polinomio interpolador son los mismos. Lo único que cambia es la data con la cual se usan.

5.8. Puntos de Chebyshev

En esta sección responderemos la pregunta propuesta anteriormente sobre si existen puntos de interpolación más convenientes que los puntos equiespaciados para interpolación polinomial en un intervalo finito. En la sec-

¹²Se omitió simplificar los términos para destacar como fueron obtenido, es decir son múltiplos de $1/8$.

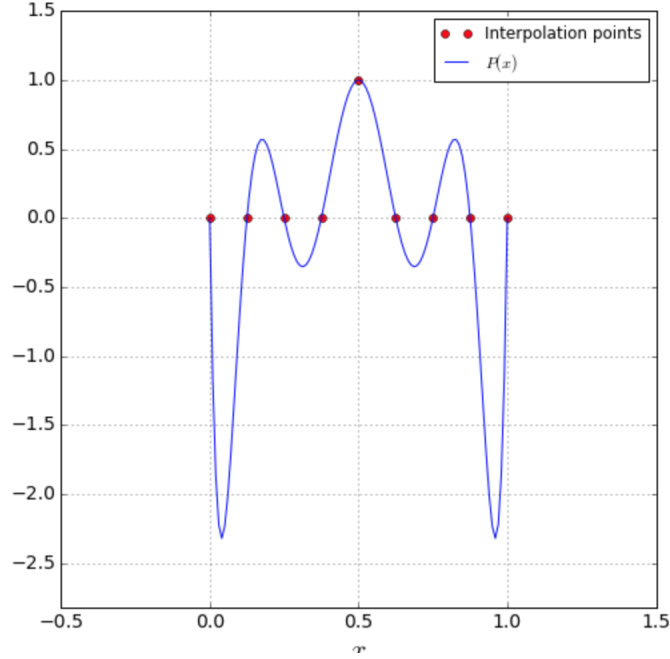


Figura 5.6: Gráfico que presenta el fenómeno de Runge. Este fenómeno se presenta al construir un interpolador polinomial con puntos equiespaciados. El fenómeno consiste en generar oscilaciones significativas entre puntos de interpolación en los extremos del intervalo de interpolación.

ción anterior se analizó que la componente del error más prometedora para la reducción del error correspondía al término $(x - x_1)(x - x_2) \dots (x - x_n)$, en particular a los puntos de interpolación x_i para $i \in \{1, 2, \dots, n\}$.

¡MUY IMPORTANTE!

Nuevamente destacar que lo que se analizará en esta sección es la sección de puntos de interpolación solamente, los algoritmos discutidos anteriormente seguirán aplicándose de la misma manera.

Por simplicidad de análisis y sin pérdida de generalidad, se utilizará de aquí en adelante la siguiente restricción¹³,

$$-1 \leq x_1, x_2, x_3, \dots, x_n \leq 1.$$

Además también se considerará que $x \in [-1, 1]$, es decir el punto donde se evalúa el interpolador también estará en el intervalo indicado.

Para ejemplificar el problema que se enfrenta, procederemos a analizar el caso en que se quiere encontrar solo 2 puntos, es decir x_1 y x_2 . Para este caso, la función a minimizar es la siguiente:

$$w(\hat{x}_1, \hat{x}_2) = \max_{x \in [-1, 1]} |(x - \hat{x}_1)(x - \hat{x}_2)|.$$

Para ser más específico, el procedimiento considera que la minimización se aplique sobre el cuadrado unitario

¹³Esta restricción es conveniente pero puede modificarse luego simplemente con un cambio de variable del intervalo $[-1, 1]$ a un intervalo general $[a, b]$, con $b > a$.

$[0, 1]^2$ y entregue los valores de x_1 y x_2 . Esto se traduce en la siguiente expresión,

$$\begin{aligned} [x_1, x_2] &= \operatorname{argmin}_{\hat{x}_1, \hat{x}_2 \in [-1, 1]} \max_{x \in [-1, 1]} |(x - \hat{x}_1)(x - \hat{x}_2)| \\ &= \operatorname{argmin}_{\hat{x}_1, \hat{x}_2 \in [-1, 1]} w(\hat{x}_1, \hat{x}_2). \end{aligned}$$

El cual se entiende como minimizar el peor caso. La función argmin indica que retorna los valores x_1 y x_2 que minimizan la expresión $w(\hat{x}_1, \hat{x}_2)$, la cual a su vez entrega el máximo valor de $|(x - \hat{x}_1)(x - \hat{x}_2)|$ considerando que \hat{x}_1 y \hat{x}_2 son valores fijos. La Figura 5.7 muestra la evaluación de $w(\hat{x}_1, \hat{x}_2)$ sobre una grilla discreta del dominio $[-1, 1]^2$. Esta representación gráfica solo ilustra convenientemente el caso donde se buscan 2 puntos, en particular los mínimos se alcanzan en 2 puntos, estos son aproximadamente $(0,75, -0,75)$ y $(-0,75, 0,75)$. Afortunadamente

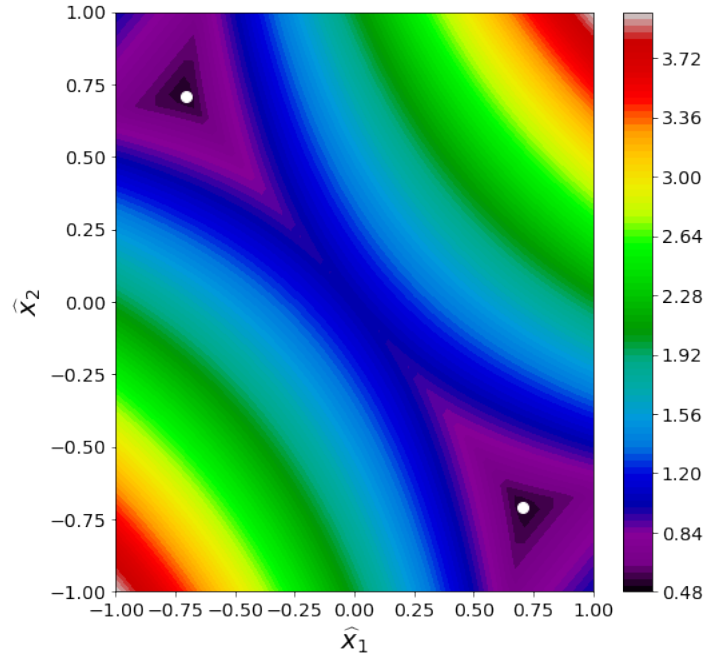


Figura 5.7: Mapa de calor de la función $w(\hat{x}_1, \hat{x}_2)$ sobre una grilla discreta del dominio $[-1, 1]^2$. La barra de colores indica el valor de la función en el punto correspondiente. En color negro se observa donde la función alcanza el mínimo. Los puntos blancos cercano a las coordenadas $(0,75, -0,75)$ y $(-0,75, 0,75)$ son donde exactamente se obtiene el mínimo.

conocemos numéricamente estos valores, los cuales corresponden a:

$$[x_1, x_2] = [0,707106781186547, -0,707106781186547],$$

o

$$[x_1, x_2] = [-0,707106781186547, 0,707106781186547].$$

Estos valores fueron extraídos del Jupyter Notebook *Bonus - 05 - Finding 2 Chebyshev Points Graphically.ipynb*, el cual se sugiere revisar para entender los detalles de como fue generada la imagen asociada. La respuesta obtenida

consiste en dos pares de soluciones, pero en realidad son simétricas, por lo que en realidad es suficiente solo considerar un par. Desafortunadamente resolver este problema de forma numérica cada vez que se necesite requerirá una cantidad de operaciones elementales significativa. Afortunadamente esto ya fue resuelto algebraicamente por Chebyshev! Para el caso de 2 puntos la solución algebraica corresponde a:

$$x_1 = \cos\left(\frac{\pi}{4}\right) = \frac{\sqrt{2}}{2} \approx 0,70710678118654752440084436210484903928483593768847,$$

$$x_2 = \cos\left(\frac{3\pi}{4}\right) = -\frac{\sqrt{2}}{2} \approx -0,70710678118654752440084436210484903928483593768847.$$

Otra forma visual de entender la importancia de los puntos de Chebyshev es por la construcción de la gráfica de la función $|(x - \hat{x}_1)(x - \hat{x}_2)|$ para distintos valores de x_1 y x_2 en el intervalo $[-1, 1]$. En la figura 5.8 se ejemplifica que al elegir los puntos de Chebyshev se obtiene el menor error máximo de los 4 pares de valores de x_1 y x_2 utilizados.

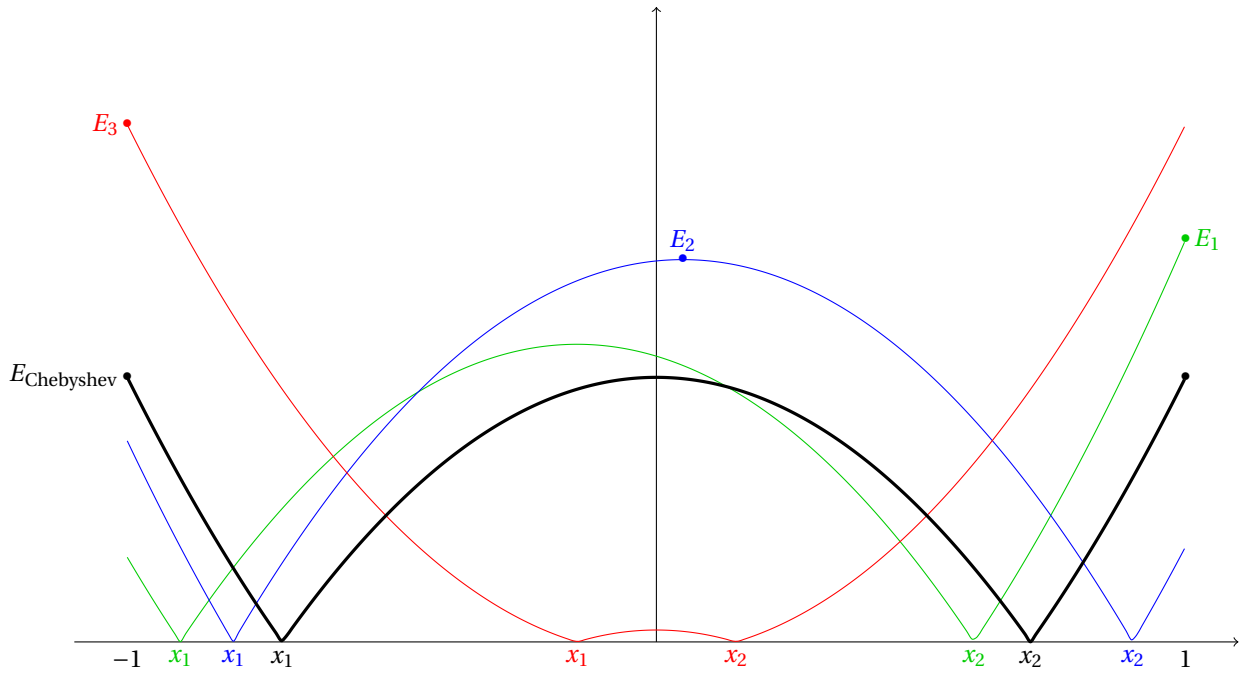


Figura 5.8: Gráfica de la función $|(x - \hat{x}_1)(x - \hat{x}_2)|$ para 4 casos. En color negro se destacan los puntos de Chebyshev, los cuales minimizan el error máximo. **Créditos:** Bastián Soto Iturra, 2022-2.

El siguiente teorema formaliza el resultado recién obtenido de forma numérica.

Thm 16 (Teorema de Chebyshev). *La elección de los números reales $-1 \leq x_1, x_2, \dots, x_n \leq 1$ que hace el valor de*

$$\max_{-1 \leq x \leq 1} |(x - x_1) \dots (x - x_n)|$$

lo más pequeño posible es

$$x_i = \cos\left(\frac{(2i-1)\pi}{2n}\right), \quad i \in \{1, 2, \dots, n\},$$

y el valor mínimo es $\frac{1}{2^{n-1}}$. De hecho, el mínimo es alcanzado por:

$$(x - x_1) \dots (x - x_n) = \frac{1}{2^{n-1}} T_n(x),$$

donde $T_n(x) = \cos(n \arccos(x))$ es polinomio de Chebyshev de grado n .

A partir del teorema, se llega a la conclusión de que el error de interpolación se minimiza si los n puntos de interpolación en $[-1, 1]$ se eligen como las raíces del polinomio de interpolación de Chebyshev $T_n(x)$ de grado n . Este resultado tiene importantes implicancias computacionales, ya que nos permite interpolar una función con los puntos de interpolación óptimos! En los experimentos computacionales veremos que el error se reduce mucho más rápido que con puntos equiespaciados y además necesita menos puntos de interpolación, por lo que reduce el número de operaciones elementales tanto para la construcción como evaluación de un polinomio por requerir un n menor.

La definición de los polinomios de Chebyshev es bien interesante, ya que depende de las funciones coseno y su inversa, pero son en realidad polinomios. A continuación se presentan algunos polinomios de Chebyshev en su forma polinomial:

$$T_0(x) = \cos(0 \arccos(x)) = 1,$$

$$T_1(x) = \cos(1 \arccos(x)) = x,$$

$$T_2(x) = \cos(2 \arccos(x)) = 2x^2 - 1.$$

Y así sucesivamente. Lo otro interesante de los polinomios de Chebyshev es que también cumplen la siguiente relación de recurrencia,

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

La definición de los polinomios de Chebyshev en función de la función coseno y su inversa también permite concluir que podemos acotar los polinomios por 1, es decir $|T_n(x)| \leq 1$. Esto no lo podemos hacer con puntos de interpolación equiespaciados, por lo cual es una ventaja importante de los puntos de Chebyshev respecto a los puntos equiespaciados.

Por otro lado, los puntos de Chebyshev pueden ser interpretados geométricamente como la proyección de puntos equiespaciados sobre un semicírculo descrito por la curva paramétrica $\langle \cos(\theta), \sin(\theta) \rangle$ sobre el eje x . Por ejemplo si consideramos la siguiente discretización equiespaciada de n ángulos θ ,

$$\theta_i = \frac{(2i-1)\pi}{2n}.$$

Por lo que el semicírculo quedaría descrito por la siguiente parametrización,

$$\langle \cos(\theta_i), \sin(\theta_i) \rangle, \quad i \in \{1, 2, \dots, n\}.$$

La Figura 5.9 muestra el caso de $n = 5$. Esta interpretación gráfica es solo ilustrativa, pero de todos modos destaca que al tener ángulos equiespaciados θ_i en el semicírculo, produce naturalmente puntos aglomerados en los bordes del intervalo. Los ángulos en este caso corresponden a:

$$\theta_1 = \frac{(2 \cdot 1 - 1)\pi}{10} = \frac{\pi}{10},$$

$$\theta_2 = \frac{(2 \cdot 2 - 1)\pi}{10} = \frac{3\pi}{10},$$

$$\theta_3 = \frac{(2 \cdot 3 - 1)\pi}{10} = \frac{5\pi}{10},$$

$$\theta_4 = \frac{(2 \cdot 4 - 1)\pi}{10} = \frac{7\pi}{10},$$

$$\theta_5 = \frac{(2 \cdot 5 - 1)\pi}{10} = \frac{9\pi}{10}.$$

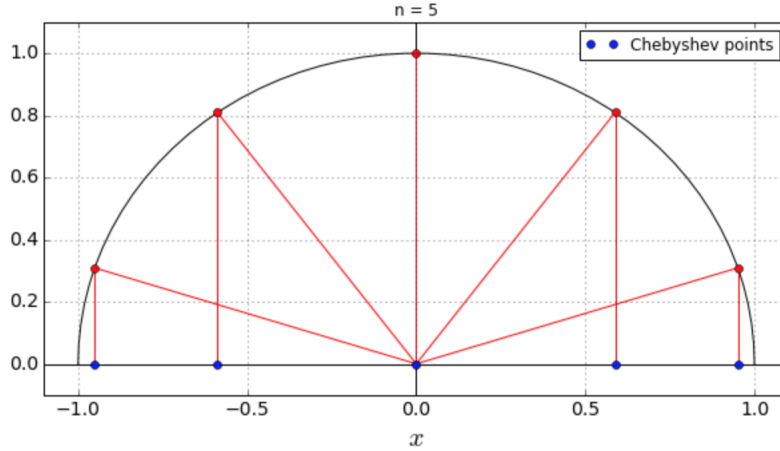


Figura 5.9: Interpretación gráfica de 5 puntos de Chebyshev. Se presenta en el semicírculo los 5 puntos de la parametrización $\langle \cos(\theta_i), \sin(\theta_i) \rangle$ considerando ángulos equiespaciados. En el eje x , en azul, se presentan los 5 puntos de Chebyshev $\cos(\theta_i)$ asociados.

5.9. Ejemplo (3.10) del libro guía

Encuentre el peor caso del error para la diferencia de $\exp(x)$ y el polinomio de Chebyshev de grado 4 en el intervalo $x = [-1, 1]$.

Considerando que se utilizan 5 puntos de Chebyshev, la siguiente desigualdad es válida,

$$\begin{aligned} |\exp(x) - p(x)| &= \frac{|(x - x_1) \dots (x - x_5)|}{5!} |f^{(5)}(c)| \\ &\leq \frac{1}{2^{5-1}} \frac{1}{5!} |f^{(5)}(c)|. \end{aligned}$$

Para efectivamente poder encontrar una cota superior para el error necesitamos acotar el término desconocido, el cual es $|f^{(5)}(c)|$. Lo interesante en este caso es que podemos obtener la quinta derivada de $\exp(x)$ fácilmente,

$$f^{(5)}(c) = \exp(c),$$

es decir, la derivada de la función exponencial sigue siendo la función exponencial! Ahora, el máximo de la función exponencial en el intervalo $[-1, 1]$ se obtiene en $x = 1$, por lo tanto,

$$|\exp(x) - p(x)| \leq \frac{1}{2^{5-1}} \frac{1}{5!} \exp(1) \approx 0,001415771785656.$$

Por lo tanto el peor caso del error es aproximadamente 0,001415771785656.

5.10. Cambio de Intervalo

El análisis de la construcción de los puntos Chebyshev se ha restringido al intervalo $[-1, 1]$, sin embargo con un cambio de variable podemos llevarlo al intervalo $[a, b]$. Por completitud, recordamos que los puntos de Chebyshev en el intervalo $[-1, 1]$ son los siguientes,

$$x_i = \cos\left(\frac{(2i-1)\pi}{2n}\right).$$

Entonces, si consideramos el siguiente cambio de variable,

$$\tilde{x}_i = \frac{b-a}{2} x_i + \frac{b+a}{2},$$

podemos fácilmente mover los puntos de Chebyshev al intervalo $[a, b]$. En la Figura 5.10 se muestra la interpretación gráfica del cambio de variable.

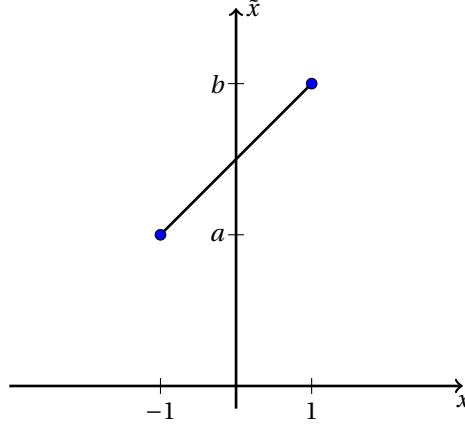


Figura 5.10: Interpretación gráfica del cambio de variable $\tilde{x} = \frac{b-a}{2} x + \frac{b+a}{2}$. En la figura se aprecia que cuando $x = -1$ se obtiene $\tilde{x} = a$, y cuando $x = 1$ se obtiene $\tilde{x} = b$.

Este simple y útil cambio de variable implica el siguiente cambio en la cota de un término del error de interpolación, es decir,

$$|(x - \tilde{x}_1) \dots (x - \tilde{x}_n)| \leq \frac{\left(\frac{b-a}{2}\right)^n}{2^{n-1}}.$$

En el caso que $a = -1$ y $b = 1$ se recupera el resultado original. Por lo tanto el error de interpolación en el intervalo $[a, b]$ considerando puntos de Chebyshev modificados es el siguiente,

$$\begin{aligned} |f(x) - p(x)| &= \frac{|(x - \tilde{x}_1)(x - \tilde{x}_2) \dots (x - \tilde{x}_n)|}{n!} |f^{(n)}(c)| \\ &\leq \frac{\left(\frac{b-a}{2}\right)^n}{2^{n-1} n!} |f^{(n)}(c)|. \end{aligned}$$

5.11. ¿Existen otros algoritmos de interpolación?

Para finalizar este capítulo, es importante destacar que los algoritmos discutidos en esta sección corresponde específicamente a algoritmos para interpolación polinomial en 1D. Esto es posible extenderlo a más dimensiones, pero antes de eso es conveniente mencionar otros algoritmos para interpolación polinomial en 1D. En este caso nos referimos a *Splines Cúbicas*, ver apéndice C. La ventaja de las *splines* cúbicas es que no sufren del fenómeno de Runge pero requieren resolver un sistema de ecuaciones lineales, similar a la utilización de la matriz de Vandermonde pero para una matriz mejor condicionada. Una aplicación interesante de las *splines* cúbicas es cuando se conectan con problemas de Mínimos Cuadrados, los cuales veremos en el siguiente capítulo!

Capítulo 6

Mínimos Cuadrados

En este capítulo analizaremos una nueva categoría de problemas, la cual corresponde a sistemas de ecuaciones lineales con más ecuaciones que incógnitas. La versión que estudiaremos es el problema de mínimos cuadrados, pueden haber variantes, pero es esencial primero entender en profundidad esta versión. La Figura 6.1 muestra un *sketch* de un problema de mínimos cuadrados donde se obtiene una aproximación lineal que aproxima los datos (x_i, y_i) para $i \in \{1, 2, \dots, m\}$, entregados. En la siguiente sección se discutirán más detalles sobre lo que significa y la necesidad de construir una aproximación de mínimos cuadrados y no una interpolación.

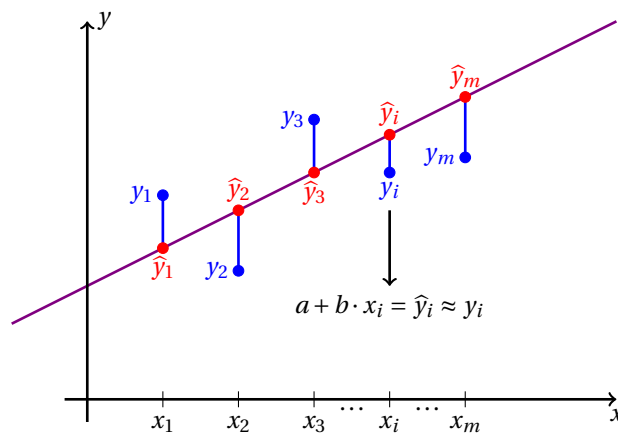


Figura 6.1: Sketch de una aproximación de mínimos cuadrados. Los pares ordenados (x_i, y_i) se interpretan como los datos y los pares ordenados $(x_i, \hat{y}_i = a + b x_i)$ se interpretan como la aproximación lineal que reduce el error cuadrático. El error cuadrático se define como: $\sum_{i=1}^m (y_i - a - b x_i)^2$.

6.1. Interpolación vs Aproximación de Mínimos Cuadrados

La primera pregunta que podría surgir asociada a mínimos cuadrados es la siguiente: ¿Para qué necesitamos una aproximación de mínimos cuadrados si es que podemos construir una interpolación que pase exactamente por los datos (x_i, y_i) entregados? La respuesta está asociado al origen de los datos con lo que trabajaremos en este

capítulo. En el Capítulo 5, de Interpolación Polinomial en 1D, se consideró que la data con la cual se estaba trabajando era exacta. Más aún, cuando interpolábamos funciones queríamos que el error fuera lo más bajo posible. Incluso, los puntos de Chebyshev nos ayudaron a reducir el error significativamente. Ahora, en este capítulo la consideración de exactitud de los datos obtenidos cambia. En particular, se considerará que los datos vienen con un error aditivo en y_i ¹. Esto significa que podemos escribir la siguiente relación algebraica para el error:

$$y_i = y_i^{(e)} + \varepsilon_i, \quad \text{para } i \in \{1, 2, \dots, m\}, \quad (6.1)$$

donde y_i es el dato conocido al cual tenemos acceso, $y_i^{(e)}$ es el dato *exacto* que nos gustaría recuperar y al cual no tenemos acceso, y ε_i es el error en la obtención del dato y al cual tampoco tenemos acceso.

¡MUY IMPORTANTE!

¿Cuál es la relación entre y_i , $y_i^{(e)}$, e \hat{y}_i ? La respuesta es simple:

- y_i : Es el dato al cual tenemos acceso, ver ecuación (6.1).
- $y_i^{(e)}$: Es el dato *exacto* al cual nos gustaría tener acceso, ver ecuación (6.1).
- \hat{y}_i : Es lo que recuperaremos con la aproximación de mínimos cuadrados, ver figura 6.1.

Estos tres términos satisfacen la siguiente identidad:

$$y_i = y_i^{(e)} + \varepsilon_i = \hat{y}_i + r_i,$$

de lo cual el único término que nos falta por definir es r_i , el cual se define como $r_i = y_i - \hat{y}_i$ y corresponde al **residuo** obtenido al aproximar la **data** y_i con el modelo elegido.

Por otro lado, es tentador pasar al lado izquierdo de la ecuación ε_i y despejar $y_i^{(e)}$, y luego aplicar algún algoritmo de interpolación polinomial que estudiamos en el Capítulo 5, sin embargo esto no es posible ya que no conocemos $y_i^{(e)}$ ni ε_i . Solo sabemos que esos valores existen. Lo que uno puede considerar es que $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, es decir es una variable aleatoria que sigue una distribución normal² con media 0 y varianza σ^2 .

Entonces, no podemos aplicar directamente los algoritmos de interpolación estudiados anteriormente dado que la data viene con un error asociado que es desconocido. Por lo tanto debemos analizar este problema de una manera distinta a lo ya estudiado.

¡MUY IMPORTANTE!

En principio, si es que no tuviéramos datos con x_i repetidos, podríamos efectivamente interpolar la data presentada en la Figura 6.1 con un polinomio. Esto generaría un polinomio de grado $n - 1$ o menor. El problema es que al realizar esto se produciría *over-fitting*^a, es decir se está incluyendo en el modelo el error en la data. Dependiendo de la aplicación o situación en particular donde se quiera aplicar el modelo obtenido, podría traer consigo resultados inesperados. Un ejemplo simple sería el caso en que uno quisiera estimar la derivada en cierta parte del dominio. Dado que la interpolación podría tener muchas oscilaciones, no serán confiables los resultados para estimar la derivada. En caso contrario, aplicando primero una aproximación que minimice los errores primero y luego aproxime la derivada, se espera que los resultados sean más razonables.

^aVer [Overfitting](#) en Wikipedia.

¹Por simplicidad consideraremos que los datos x_i no vienen con error.

²Esto es solo un ejemplo, los datos pueden venir con otros errores, pero el más común es el error Gaussiano.

6.2. Primera Alternativa: Mínimos cuadrados por minimización

En esta sección estudiaremos una primera alternativa para obtener los coeficientes que minimizan el error cuadrático, es decir, encontrar los coeficientes por medio de una minimización en alta dimensión sin restricciones³.

Antes de poder aplicar un algoritmo de minimización, debemos definir la **estructura** de la aproximación que utilizaremos. En este caso particular, y basándonos en el ejemplo presentado en la Figura 6.1, se trabajará con una estructura lineal respecto a la data, es decir:

$$\hat{y} = a + b x.$$

Nota

En realidad se pueden proponer diversas estructuras, por ejemplo considere estos 2 casos:

$$y = a + b x + c x^2,$$

$$y = a + b x + c \sin(x).$$

En ambos casos, aunque la relación entre y y x no sea lineal, el problema es lineal respecto a los coeficientes a , b , y c . No hay límite en la estructura que se puede utilizar! Pero sí es muy importante que la estructura propuesta esté relacionada con el contexto del problema bajo estudio.

Entonces, una vez definida la estructura algebraica con la cual se buscará minimizar el error cuadrático, podemos construir la función de error cuadrático. La función de error cuadrático⁴ $E(\cdot)$ se define de la siguiente forma:

$$\begin{aligned} E(a, b) &= \sum_{i=1}^m r_i^2, \\ &= \sum_{i=1}^m (y_i - \hat{y}_i)^2, \\ &= \sum_{i=1}^m (y_i - (a + b x_i))^2, \\ &= \sum_{i=1}^m (y_i - a - b x_i)^2. \end{aligned} \tag{6.2}$$

Esta función tiene 2 grados de libertad, es decir, solo podemos modificar las variables a y b , ya que todas las otras componentes son datos. Por lo tanto surge la siguiente pregunta: ¿Cómo encontramos los coeficientes a y b tal que se minimice el $E(a, b)$, es decir, el error cuadrático? Dado que no tenemos restricciones impuestas sobre a y/o b , esto se convierte en un problema de minimización en alta dimensión. En realidad alta dimensión acá corresponde solo a dimensión 2, pero igual se considera que es alta dimensión ya que el procedimiento es el mismo si se consideran dimensiones superiores. El procedimiento es simple y directo, encontrar el punto crítico por medio de resolver $\nabla E = \mathbf{0}$, es decir, igualar el gradiente de $E(a, b)$ a 0⁵. Es importante recordar que el gradiente en este caso corresponde a obtener,

$$\nabla E(a, b) = \left\langle \frac{\partial E}{\partial a}, \frac{\partial E}{\partial b} \right\rangle.$$

³Se podrían agregar restricciones si el problema analizado lo requiere. Sin embargo se requeriría trabajar con [Multiplicadores de Lagrange!](#) por ejemplo

⁴También es posible utilizar el error cuadrático medio, la única diferencia es considerar el coeficiente $1/m$ frente a la sumatoria, sin embargo esto no modifica la solución por lo cual se omite.

⁵En 1D el procedimiento es igualar la derivada a 0, pero en alta dimensión la derivada se *convierte* en el gradiente!

Por lo que en este caso corresponde a,

$$\frac{\partial E}{\partial a} = \frac{\partial}{\partial a} \sum_{i=1}^m (y_i - a - b x_i)^2 = \sum_{i=1}^m 2 (y_i - a - b x_i) (-1), \quad (6.3)$$

$$\frac{\partial E}{\partial b} = \frac{\partial}{\partial b} \sum_{i=1}^m (y_i - a - b x_i)^2 = \sum_{i=1}^m 2 (y_i - a - b x_i) (-x_i). \quad (6.4)$$

Igualando $\frac{\partial E}{\partial a}$ a 0, i.e. igualando ecuación (6.3) a 0, y manipulando algebraicamente obtenemos,

$$\begin{aligned} \sum_{i=1}^m 2 (y_i - \bar{a} - \bar{b} x_i) (-1) &= 0, \\ \sum_{i=1}^m (y_i - \bar{a} - \bar{b} x_i) &= 0, \\ \left(\sum_{i=1}^m y_i \right) - \bar{a} m - \bar{b} \left(\sum_{i=1}^m x_i \right) &= 0, \\ m \bar{a} + \left(\sum_{i=1}^m x_i \right) \bar{b} &= \left(\sum_{i=1}^m y_i \right). \end{aligned} \quad (6.5)$$

Notar que utilizamos \bar{a} y \bar{b} en la ecuación anterior porque esta ecuación es la que efectivamente debe ser satisfecha por los coeficientes que minimizan el error, donde en las ecuaciones (6.3) y (6.4) no era el caso. De igual manera para $\frac{\partial E}{\partial b}$, i.e. ecuación (6.4), obtenemos,

$$\begin{aligned} \sum_{i=1}^m 2 (y_i - \bar{a} - \bar{b} x_i) (-x_i) &= 0, \\ \sum_{i=1}^m (y_i x_i - \bar{a} x_i - \bar{b} x_i^2) &= 0, \\ \left(\sum_{i=1}^m y_i x_i \right) - \bar{a} \left(\sum_{i=1}^m x_i \right) - \bar{b} \left(\sum_{i=1}^m x_i^2 \right) &= 0, \\ \left(\sum_{i=1}^m x_i \right) \bar{a} + \left(\sum_{i=1}^m x_i^2 \right) \bar{b} &= \left(\sum_{i=1}^m y_i x_i \right). \end{aligned} \quad (6.6)$$

Ahora, ecuación (6.5) en conjunto con la ecuación (6.6) generan el siguiente sistema de ecuaciones lineales cuadrado,

$$\begin{bmatrix} m & \sum_{i=1}^m x_i \\ \sum_{i=1}^m x_i & \sum_{i=1}^m x_i^2 \end{bmatrix} \begin{bmatrix} \bar{a} \\ \bar{b} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^m y_i \\ \sum_{i=1}^m y_i x_i \end{bmatrix}. \quad (6.7)$$

El cual tiene la siguiente solución,

$$\begin{aligned} \bar{a} &= \frac{(\sum_{i=1}^m x_i^2) (\sum_{i=1}^m y_i) - (\sum_{i=1}^m x_i) (\sum_{i=1}^m y_i x_i)}{m (\sum_{i=1}^m x_i^2) - (\sum_{i=1}^m x_i)^2}, \\ \bar{b} &= \frac{m (\sum_{i=1}^m y_i x_i) - (\sum_{i=1}^m x_i) (\sum_{i=1}^m y_i)}{m (\sum_{i=1}^m x_i^2) - (\sum_{i=1}^m x_i)^2}. \end{aligned}$$

Notar nuevamente que se ha utilizado una línea sobre las variables a y b , es decir \bar{a} y \bar{b} , para diferenciar entre las variables no instanciadas, originalmente definidas en $E(a, b)$, y las instancias que minimizan el error cuadrático, es decir \bar{a} y \bar{b} .

Entonces, hemos exitosamente encontrado los valores para a y b que minimizan el error cuadrático definido en la ecuación (6.2) por medio de un proceso de minimización en alta dimensión sin restricciones.

6.3. Segunda Alternativa: Mínimos Cuadrados desde el Álgebra Lineal! y las Ecuaciones Normales.

La segunda alternativa que estudiaremos corresponde a una interpretación desde el Álgebra Lineal 😊. Esta alternativa es nuestro primer encuentro⁶ con los fundamentos utilizados en métodos avanzados de Álgebra Lineal, en particular, para resolver sistemas de ecuaciones lineales.

Entonces, considerando la notación presentada anteriormente, es decir que $y_i \approx \hat{y}_i = a + b x_i$, podemos escribir nuestro problema como un sistema de ecuaciones lineales sobre-determinado, es decir, con más ecuaciones que incógnitas, de la siguiente forma,

$$\begin{aligned} a + b x_1 &\approx y_1, \\ a + b x_2 &\approx y_2, \\ a + b x_3 &\approx y_3, \\ &\vdots \\ a + b x_m &\approx y_m. \end{aligned}$$

Y para evitar el uso de símbolo de aproximación \approx , podemos considerar el residuo r_i , introducido anteriormente, de la siguiente forma,

$$\begin{aligned} a + b x_1 + r_1 &= y_1, \\ a + b x_2 + r_2 &= y_2, \\ a + b x_3 + r_3 &= y_3, \\ &\vdots \\ a + b x_m + r_m &= y_m. \end{aligned}$$

Utilizando notación matricial obtenemos,

$$\underbrace{\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_m \end{bmatrix}}_{\mathbf{r}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{b}} - \underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix}}_A \begin{bmatrix} a \\ b \end{bmatrix}. \quad (6.8)$$

De esta formulación, podemos extraer algunos puntos bien importantes:

- La matriz tiene m filas y solo 2 columnas. En general, se utiliza la variable m para denotar la cantidad de filas y n para la cantidad de columnas.

⁶En realidad podría ser nuestro segundo encuentro si que antes estudió el método de Gradiente Conjugado en la sección D.3

- Si $m > n$, y el rank es n , entonces tenemos un sistema de ecuaciones lineales sobre-determinado y *full rank*.
- Si $m = n$, entonces tenemos un sistema de ecuaciones lineales cuadrado y ya sabemos como resolverlo.
- En caso de que tengamos $m < n$, tenemos un sistema de ecuaciones lineales bajo-determinado. En este caso la solución no es única, por ejemplo considere $x + y = 1$. Una alternativa para encontrar una solución única es imponer restricciones, por ejemplo, resolver $x + y = 1$ tal que se minimice $x^2 + y^2$.
- Por otro lado, se puede notar que la estructura de la matriz A obtenida es similar a la de la matriz de Vandermonde utilizada en interpolación polinomial, sin embargo en este caso está truncada la cantidad de columnas.

En resumen, dado que en el caso de análisis tenemos más filas que columnas, podemos concluir que estamos analizando un sistema de ecuaciones lineales sobre-determinado⁷.

Del desarrollo anterior, podemos re-escribir la ecuación (6.8) de la siguiente forma,

$$\begin{aligned}\mathbf{r} &= \mathbf{b} - A \begin{bmatrix} a \\ b \end{bmatrix}, \\ \mathbf{r} &= \mathbf{b} - [\mathbf{v}_1, \quad \mathbf{v}_2] \begin{bmatrix} a \\ b \end{bmatrix}, \\ \mathbf{r} &= \mathbf{b} - a\mathbf{v}_1 - b\mathbf{v}_2.\end{aligned}$$

Es decir, estamos buscando la *mejor* combinación lineal de los vectores \mathbf{v}_1 y \mathbf{v}_2 para aproximar \mathbf{b} , o equivalentemente, que se *minimice*⁸ el residuo \mathbf{r} . Donde consideramos como la *mejor* combinación lineal, la combinación lineal que minimice el error cuadrático, el cual puede ser re-escrito de la siguiente forma,

$$E(a, b) = \|\mathbf{r}\|_2^2 = \|\mathbf{b} - a\mathbf{v}_1 - b\mathbf{v}_2\|_2^2. \quad (6.9)$$

Notar que esta nueva forma de escribir el error cuadrático es exactamente igual a la definición anteriormente presentada en la ecuación (6.2)⁹. Lo interesante de esta nueva definición es que nos permite introducir la relación con Álgebra Lineal que andamos buscando. En la Figura 6.2 se presenta una interpretación gráfica de que significa reducir el error cuadrático desde el punto de vista del Álgebra Lineal.

En la Figura 6.2 se presenta que $\bar{\mathbf{x}}$ es el vector que minimiza el error cuadrático. Para nuestro ejemplo esto corresponde a $\bar{\mathbf{x}} = [\bar{a}, \bar{b}]^T$. Sin pérdida de generalidad y por simplicidad en la explicación, trabajaremos de ahora en adelante con el caso general $\mathbf{r} = \mathbf{b} - A\mathbf{x}$, con $A \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, y $\mathbf{r} \in \mathbb{R}^m$. Pero, antes de continuar, procederemos a re-hacer la figura antes presentada en una forma más compacta y al mismo tiempo más general. Solo notar que de todos modos es muy importante primero entender la Figura 6.2 antes de entender la Figura 6.3.

Ahora, en la Figura 6.3 se presentan las componentes claves para poder construir una expresión que nos permita encontrar $\bar{\mathbf{x}}$, es decir, el vector que minimiza $\|\mathbf{b} - A\mathbf{x}\|_2^2$. En particular, el punto clave en la Figura 6.3 es la ortogonalidad¹⁰ entre $\mathbf{r} = \mathbf{b} - A\bar{\mathbf{x}}$ y $A\mathbf{x}$. Entonces, si \mathbf{r} y $A\mathbf{x}$ son ortogonales, se debe cumplir que su producto interno debe ser nulo. Esto se reduce a la siguiente expresión,

$$\begin{aligned}(A\mathbf{x})^* \mathbf{r} &= 0 \\ (A\mathbf{x})^* (\mathbf{b} - A\bar{\mathbf{x}}) &= 0 \\ \mathbf{x}^* A^* (\mathbf{b} - A\bar{\mathbf{x}}) &= 0.\end{aligned}$$

⁷Si todos los x_i son iguales, entonces no existirá una solución única. La solución que minimiza el error cuadrático será única si A es full rank, ver sección 1.3.5. En este caso particular, considerando que $m > 2$, entonces es full rank si el rank es 2. Es decir, si la cantidad de filas linealmente independientes es 2, o, equivalentemente, si la cantidad de columnas linealmente independientes es 2. Notar que si todos los x_i son iguales, la segunda columna será linealmente dependiente de la primera, lo que implica que el rank será 1.

⁸En este caso nos referimos a que se minimice la norma 2 del residuo, es decir $\|\mathbf{r}\|_2$.

⁹Se sugiere encarecidamente comparar ambas definiciones!

¹⁰Recuerde que dos vectores son ortogonales si su producto interno es nulo. Por ejemplo, \mathbf{v} y \mathbf{w} son ortogonales si $\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^* \mathbf{w} = 0$, donde $*$ corresponde al operador transpuesta conjugada.

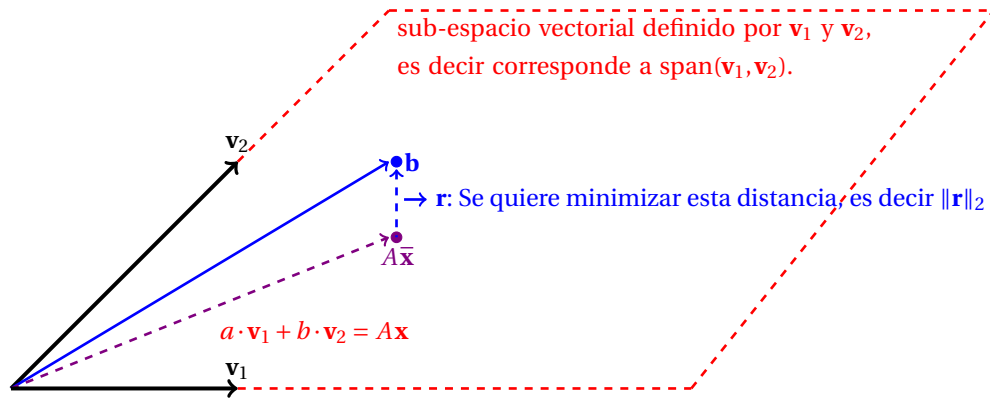


Figura 6.2: Diagrama donde se presenta el sub-espacio vectorial generado por los vectores \mathbf{v}_1 y \mathbf{v}_2 en rojo. En azul se presenta el vector \mathbf{b} , que no necesariamente pertenece al sub-espacio vectorial generado por los vectores \mathbf{v}_1 y \mathbf{v}_2 . La línea punteada en azul muestra el vector residual $\mathbf{b} - A\bar{\mathbf{x}}$, que tradicionalmente denotamos por \mathbf{r} . El punto $A\bar{\mathbf{x}}$ en morado corresponde a la *mejor* combinación lineal de los vectores \mathbf{v}_1 y \mathbf{v}_2 que minimizan el error cuadrático, es decir, que minimizan $\|\mathbf{r}\|_2^2 = \|\mathbf{b} - A\mathbf{x}\|_2^2$. En rojo también se presenta $A\mathbf{x}$, que se interpreta como el sub-espacio vectorial generado por \mathbf{v}_1 y \mathbf{v}_2 . Notar que no hay línea sobre \mathbf{x} en $A\mathbf{x}$, esto es porque corresponde al sub-espacio vectorial, no al punto $A\bar{\mathbf{x}}$ que minimiza el error cuadrático. Ahora que entiende el significado de los colores, se le invita a re-revisar la primera figura de este capítulo, es decir Figura 6.1.

Considerando que $\mathbf{x} \neq \mathbf{0}$, surge la pregunta, **¿Cómo podemos asegurar que el producto interno sea cero?** Para poder responder la pregunta, primeros haremos un arreglo conveniente de la ecuación anterior,

$$\begin{aligned}\mathbf{x}^* A^* (\mathbf{b} - A\bar{\mathbf{x}}) &= 0, \\ \mathbf{x}^* (A^* (\mathbf{b} - A\bar{\mathbf{x}})) &= 0.\end{aligned}$$

Entonces, como $\mathbf{x} \neq \mathbf{0}$, solo nos queda concluir que $A^* (\mathbf{b} - A\bar{\mathbf{x}})$ debe ser $\mathbf{0}$, lo que nos da,

$$\begin{aligned}A^* (\mathbf{b} - A\bar{\mathbf{x}}) &= \mathbf{0}, \\ A^* \mathbf{b} - A^* A\bar{\mathbf{x}} &= \mathbf{0}.\end{aligned}$$

Lo cual se reduce a,

$$A^* A\bar{\mathbf{x}} = A^* \mathbf{b}.$$

La cual se conoce como las Ecuaciones Normales. Las Ecuaciones Normales nos permite obtener una expresión explícita para $\bar{\mathbf{x}}$, es decir,

$$\bar{\mathbf{x}} = (A^* A)^{-1} A^* \mathbf{b}.$$

¡MUY IMPORTANTE!

La solución de las Ecuaciones Normales nos entrega la expresión $(A^* A)^{-1} A^*$, la cual es conocida como la **pseudo-inversa de Moore-Penrose**. La cual se denota como A^\dagger o A^+ , dependiendo del texto que se use. En estos apuntes se utilizará $A^\dagger = (A^* A)^{-1} A^*$.

Recuerde que en general no es necesario obtener la inversa de $A^* A$, solo es necesario resolver el sistema de ecuaciones lineales asociado, en este caso $A^* A\bar{\mathbf{x}} = A^* \mathbf{b}$. En principio se puede utilizar cualquiera de los algorit-

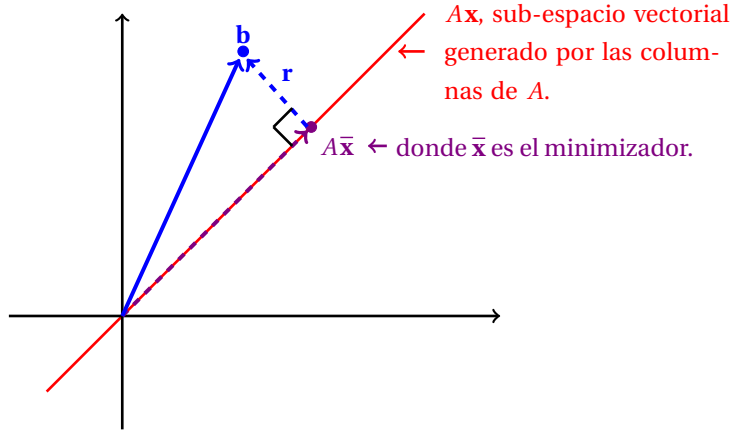


Figura 6.3: Esta diagrama es una generalización de la Figura 6.2. La misma notación es utilizada, sin embargo acá se está considerando que $A \in \mathbb{R}^{m \times n}$ y que $m \gg n$. Por completitud se explica nuevamente los elementos incluidos. En rojo se presenta el sub-espacio vectorial generado por los vectores columnas de la matriz A , es decir $A\mathbf{x} \in \mathbb{R}^m$. En azul se incluye el vector $\mathbf{b} \in \mathbb{R}^m$ y el vector residual $\mathbf{r} = \mathbf{b} - A\bar{\mathbf{x}} \in \mathbb{R}^m$. En morado se incluye el vector $\bar{\mathbf{x}} \in \mathbb{R}^n$, que es el vector que minimiza el error cuadrático $\|\mathbf{r}\|_2^2 = \|\mathbf{b} - A\bar{\mathbf{x}}\|_2^2$. Es importante destacar que el vector $\bar{\mathbf{x}} \in \mathbb{R}^n$ y no a \mathbb{R}^m . El cuadrado incluido indica que el vector $\mathbf{r} = \mathbf{b} - A\bar{\mathbf{x}}$ es ortogonal a todo el sub-espacio generado por $A\mathbf{x}$.

mos estudiados anteriormente ya que la matriz A^*A es cuadrada de dimensión $n \times n$. Es importante destacar que sabemos que la matriz A^*A es simétrica¹¹ y si es full rank, sabemos que es no singular.

Ahora, si analizamos nuevamente el problema de mínimos cuadrados presentado anteriormente en la ecuación (6.8), es decir,

$$\underbrace{\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_m \end{bmatrix}}_{\mathbf{r}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{b}} - \underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix}}_A \begin{bmatrix} a \\ b \end{bmatrix}. \quad (6.10)$$

Lo primero que debemos hacer es construir las Ecuaciones Normales, es decir $A^*A\bar{\mathbf{x}} = A^*\mathbf{b}$, entonces,

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_1 & x_2 & x_3 & \dots & x_m \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix} \begin{bmatrix} \bar{a} \\ \bar{b} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_1 & x_2 & x_3 & \dots & x_m \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix}.$$

Multiplicando las matrices, obtenemos,

$$\begin{bmatrix} m & \sum_{i=1}^m x_i \\ \sum_{i=1}^m x_i & \sum_{i=1}^m x_i^2 \end{bmatrix} \begin{bmatrix} \bar{a} \\ \bar{b} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^m y_i \\ \sum_{i=1}^m y_i x_i \end{bmatrix}. \quad (6.11)$$

¹¹En general es Hermitiana si consideramos $A \in \mathbb{C}^{m \times n}$, pero como se está considerando que $A \in \mathbb{R}^{m \times n}$, entonces A^*A es simétrica.

Lo cual es exactamente igual a lo que obtuvimos antes en la ecuación (6.7)! La gran diferencia es que en este caso no se calculó ninguna derivada y de todas maneras se obtuvo la solución que minimiza el error cuadrático.

Entonces, hemos construido 2 procedimientos distintos que resuelven el problema de mínimos cuadrados. El primero por medio de un procedimiento de minimización obteniendo el mínimo por medio del gradiente de la función de error, y el segundo con las Ecuaciones Normales. Si bien ambos procedimientos son equivalentes, en las siguientes secciones veremos que hay una ventaja en encontrar el mínimo por medio de la teoría entregada por el segundo procedimiento.

6.3.1. Ejemplo numérico de mínimos cuadrados

Considere el siguiente problema de mínimos cuadrados, el cual corresponde al ejemplo (4.2) del libro guía,

$$\underbrace{\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}}_{\mathbf{r}} = \underbrace{\begin{bmatrix} -3 \\ 15 \\ 9 \end{bmatrix}}_{\mathbf{b}} - \underbrace{\begin{bmatrix} 1 & -4 \\ 2 & 3 \\ 2 & 2 \end{bmatrix}}_A \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\mathbf{x}}. \quad (6.12)$$

Construyendo las ecuaciones normales $A^* A \bar{\mathbf{x}} = A^* \mathbf{b}$ obtenemos,

$$\begin{bmatrix} 1 & 2 & 2 \\ -4 & 3 & 2 \end{bmatrix} \begin{bmatrix} 1 & -4 \\ 2 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 \\ -4 & 3 & 2 \end{bmatrix} \begin{bmatrix} -3 \\ 15 \\ 9 \end{bmatrix}.$$

Multiplicando, obtenemos,

$$\begin{bmatrix} 9 & 6 \\ 6 & 29 \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} = \begin{bmatrix} 45 \\ 75 \end{bmatrix}.$$

Resolviendo el sistema de ecuaciones lineales cuadrado, se obtiene,

$$\bar{x}_1 = 3,8, \quad (6.13)$$

$$\bar{x}_2 = 1,8. \quad (6.14)$$

En particular, podemos obtener el vector residual,

$$\mathbf{r} = \mathbf{b} - A \bar{\mathbf{x}} = \begin{bmatrix} -3 \\ 15 \\ 9 \end{bmatrix} - \begin{bmatrix} 1 & -4 \\ 2 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 3,8 \\ 1,8 \end{bmatrix} = \begin{bmatrix} 0,4 \\ 2 \\ -2,2 \end{bmatrix},$$

donde $\|\mathbf{r}\|_2^2 = (0,4)^2 + (2)^2 + (-2,2)^2 = 9$, es decir el mínimo valor para el error cuadrático es 9.

6.4. Tipos de modelos

En esta sección se presenta una breve lista de ejemplos en donde es posible aplicar mínimos cuadrados. **Por simplicidad se presentarán los modelos en la forma $A\mathbf{x} \approx \mathbf{b}$, omitiendo la inclusión del vector residual $\mathbf{r} = \mathbf{b} - A\mathbf{x}$.** Algunos modelos propuestos son lineales con respecto a los coeficientes que se buscan, por lo cual se puede aplicar directamente la teoría antes presentada. Sin embargo también se presentan modelos no-lineales respecto a los coeficientes que se buscan, para los cuales se explica qué cambio de variables conveniente se sugiere.

En caso de que no fuera posible realizar un cambio de variables conveniente, uno puede aplicar [mínimos cuadrados no-lineales](#) directamente, en particular el algoritmo de [Levenberg-Marquardt](#).

6.4.1. Modelo lineal

Este modelo es el mismo que vimos originalmente, la única diferencia en este caso es que la data se está considerando que viene en distintos tiempo t_i . Esto puede utilizarse para aproximar data temporal, por ejemplo una serie de tiempo (t_i, y_i) , ver Figura 6.4.

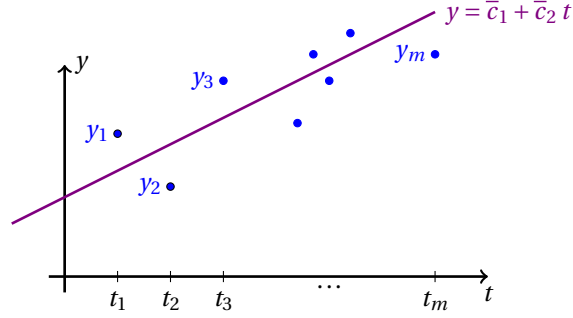


Figura 6.4: Aproximación de mínimos cuadrados con $y = c_1 + c_2 t$.

El modelo queda expresado en el siguiente sistema de ecuaciones lineales sobre-determinado,

$$\underbrace{\begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \\ \vdots & \vdots \\ 1 & t_m \end{bmatrix}}_A \underbrace{\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}}_{\mathbf{x}} \approx \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{b}}.$$

Entonces se pueden aplicar directamente las Ecuaciones Normales: $A^* A \bar{\mathbf{x}} = A^* \mathbf{b}$.

6.4.2. Modelo cuadrático

El siguiente modelo es la tradicional extensión de un modelo lineal, es decir se propone utilizar una aproximación cuadrática de la data (t_i, y_i) , ver Figura 6.5.

El modelo queda expresado en el siguiente sistema de ecuaciones lineales sobre-determinado,

$$\underbrace{\begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \\ \vdots & \vdots & \vdots \\ 1 & t_m & t_m^2 \end{bmatrix}}_A \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}}_{\mathbf{x}} \approx \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{b}}.$$

Entonces se pueden aplicar directamente las Ecuaciones Normales: $A^* A \bar{\mathbf{x}} = A^* \mathbf{b}$.

6.4.3. Modelo periódico

El siguiente modelo que se presenta es un modelo adecuado en el caso que la data muestre un comportamiento oscilatorio o periódico, ver Figura 6.6.

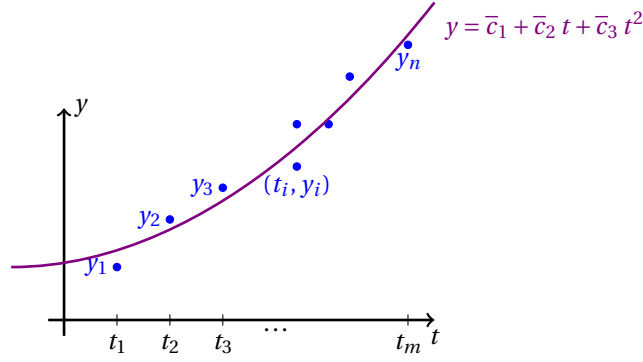


Figura 6.5: Aproximación de mínimos cuadrados con $y = c_1 + c_2 t + c_3 t^2$.

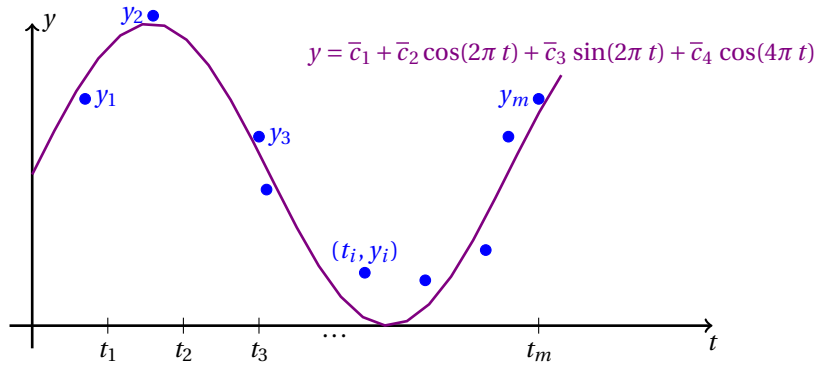


Figura 6.6: Aproximación de mínimos cuadrados con $y = c_1 + c_2 \cos(2\pi t) + c_3 \sin(2\pi t) + c_4 \cos(4\pi t)$.

El modelo queda expresado en el siguiente sistema de ecuaciones lineales sobre-determinado,

$$\underbrace{\begin{bmatrix} 1 & \cos(2\pi t_1) & \sin(2\pi t_1) & \cos(4\pi t_1) \\ 1 & \cos(2\pi t_2) & \sin(2\pi t_2) & \cos(4\pi t_2) \\ 1 & \cos(2\pi t_3) & \sin(2\pi t_3) & \cos(4\pi t_3) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \cos(2\pi t_m) & \sin(2\pi t_m) & \cos(4\pi t_m) \end{bmatrix}}_A \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}}_{\mathbf{x}} \approx \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{b}}.$$

Notar que si bien el modelo no es lineal respecto a la variable independiente t , si es lineal respecto a los parámetros c_1 , c_2 , c_3 , y c_4 , entonces aún se puede aplicar las Ecuaciones normales de forma tradicional, es decir, $A^* A \bar{\mathbf{x}} = A^* \mathbf{b}$.

6.4.4. Modelo exponencial

En este caso, si la data crece más rápido de lo que un modelo lineal o cuadrático pueden capturar, se propone un modelo exponencial. En la Figura 6.7 se presenta en naranja una propuesta de modelo exponencial, es decir $y = c_1 \exp(c_2 t)$.

Lamentablemente en este caso no es posible aplicar directamente el método de mínimos cuadrados lineal dado que el modelo no es lineal respecto a los coeficientes c_1 y c_2 . Una opción sería aplicar el método de mínimos cua-

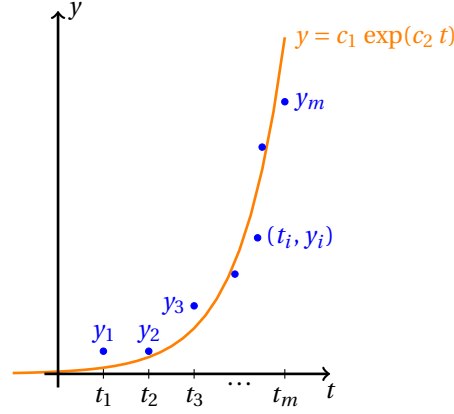


Figura 6.7: Aproximación con $y = c_1 \exp(c_2 t)$. Se presentan la propuesta de modelo en color naranja porque no es la función a la cual se le minimizará el error cuadrático. En la Figura 6.8 se muestra la función a la cual se le minimizará el error cuadrático.

drados no-lineales. Otra opción es transformar el problema no-lineal en c_1 y c_2 en uno lineal respecto a las nuevas variables. En este caso la transformación es bastante directa, es decir, dado que tenemos un modelo exponencial podemos aplicar la función logaritmo¹², entonces, aplicando la función logaritmo en ambos lados de la ecuación obtenemos,

$$\begin{aligned}\log(y) &= \log(c_1 \exp(c_2 t)), \\ &= \log(c_1) + c_2 t.\end{aligned}$$

Sin embargo aún este modelo transformado no es lineal respecto a los coeficientes c_1 y c_2 , pero notando que $\log(c_1)$ es en realidad un coeficiente, lo podemos re-escribir como $K = \log(c_1)$, entonces el modelo queda de la siguiente forma,

$$\log(y) = K + c_2 t.$$

Ahora sí es lineal respecto a los coeficientes K y c_2 ! Luego de obtener \bar{K} , uno puede recuperar c_1 aplicando la función inversa, es decir $c_1 = \exp(\bar{K})$. Esta transformación significa que en realidad nosotros estamos aplicando mínimos cuadrados en el espacio transformado. El espacio transformado se presenta en la Figura 6.8. Más aún, en este caso se aprecia que los datos transformados tienen una dependencia lineal respecto a la variable independiente t .

El modelo queda expresado en el siguiente sistema de ecuaciones lineales sobre-determinado,

$$\underbrace{\begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \\ \vdots & \vdots \\ 1 & t_m \end{bmatrix}}_A \underbrace{\begin{bmatrix} K \\ c_2 \end{bmatrix}}_{\mathbf{x}} \approx \underbrace{\begin{bmatrix} \log(y_1) \\ \log(y_2) \\ \log(y_3) \\ \vdots \\ \log(y_m) \end{bmatrix}}_{\mathbf{b}}.$$

Entonces se pueden aplicar las Ecuaciones Normales, $A^* A \bar{\mathbf{x}} = A^* \mathbf{b}$, al modelo transformado. Luego se obtiene $\bar{c}_1 = \exp(\bar{K})$ y \bar{c}_2 se obtiene directamente. Notar que se ha definido \bar{c}_1 como la transformada inversa de \bar{K} , la razón

¹²Recuerde que cuando se utiliza la función logaritmo $\log(\cdot)$ se refiere al logaritmo natural. En el caso que se refiera al logaritmo en base 10 se utiliza $\log_{10}(\cdot)$.

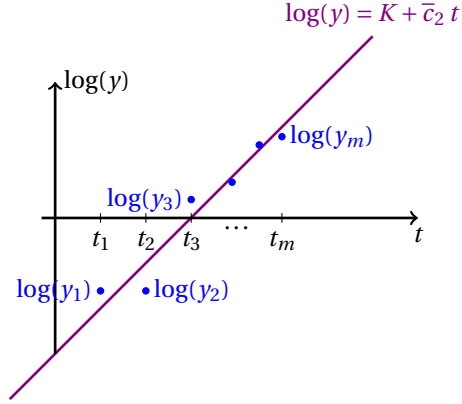


Figura 6.8: Aproximación de mínimos cuadrados con $\log(y) = K + c_2 t$.

de esto es para destacar que \tilde{c}_1 no es la solución del problema de mínimos cuadrados no-lineal original, sino la solución del problema de mínimos cuadrados modificado.

6.4.5. Ley de potencia

Este caso es una variante del modelo exponencial anterior, el modelo propuesto es $y = c_1 t^{c_2}$. Al igual que en el caso anterior, el modelo no es lineal con respecto a los coeficientes c_1 y c_2 , ver Figura 6.9.

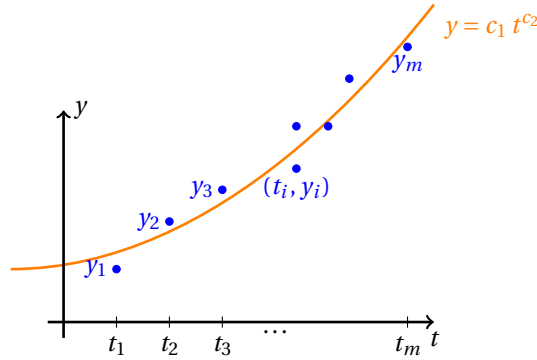


Figura 6.9: Aproximación con $y = c_1 t^{c_2}$. Al igual que en la Figura 6.7, se presenta en color naranja porque no es la función a la cual se le minimizará el error cuadrático.

Siguiendo la misma idea aplicada en el caso anterior, se propone aplicar la función logaritmo al modelo propuesto $y = c_1 t^{c_2}$, es decir,

$$\begin{aligned}\log(y) &= \log(c_1 t^{c_2}), \\ &= \log(c_1) + c_2 \log(t).\end{aligned}$$

Al igual que en caso anterior, podemos definir $K = \log(c_1)$. El cual sí es lineal respecto a las variables K y c_2 , entonces

el modelo queda expresado en el siguiente sistema de ecuaciones lineales sobre-determinado,

$$\underbrace{\begin{bmatrix} 1 & \log(t_1) \\ 1 & \log(t_2) \\ 1 & \log(t_2) \\ \vdots & \vdots \\ 1 & \log(t_m) \end{bmatrix}}_A \underbrace{\begin{bmatrix} K \\ c_2 \end{bmatrix}}_{\mathbf{x}} \approx \underbrace{\begin{bmatrix} \log(y_1) \\ \log(y_2) \\ \log(y_3) \\ \vdots \\ \log(y_m) \end{bmatrix}}_{\mathbf{b}}.$$

Entonces se pueden aplicar las Ecuaciones Normales, $A^* A \bar{\mathbf{x}} = A^* \mathbf{b}$, al modelo transformado. Luego se obtiene $\tilde{c}_1 = \exp(\bar{K})$ y \tilde{c}_2 se obtiene directamente. Al igual que el caso anterior, se utiliza la notación \tilde{c}_1 para destacar que el coeficiente obtenido es sobre el problema transformado, no el problema de mínimos cuadrados no lineales original.

Comentario al margen

Los 3 primeros modelos presentados eran modelos lineales respecto a los coeficientes asociados, sin embargo para los últimos 2 modelos tuvimos que recurrir a un cambio de variable conveniente. En particular se aplicó la función logaritmo porque uno de los coeficientes que estábamos buscando estaba en el exponente de la función exponencial o de t , respectivamente. Lo cual nos simplificó significativamente el problema, sin embargo una transformación de los datos de esa forma no siempre es posible, pero si es importante considerarla. Ahora, si el modelo propuesto fuera de la forma $y = c_1 \exp(c_2 t) + c_3$ o $y = c_1 t^{c_2} + c_3$, el problema ya no puede simplificarse tan fácilmente. Aquí es donde uno debe aplicar la creatividad para poder resolver el problema! O aplicar directamente mínimos cuadrados no-lineales!

6.5. ¿Qué sabemos de $A^* A$?

En la primera parte de este capítulo se construyeron las ecuaciones normales, las cuales son,

$$A^* A \bar{\mathbf{x}} = A^* \mathbf{b}.$$

Si definimos $B = A^* A$, y sabemos que $B^* = (A^* A)^* = A^* A^{**} = A^* A = B$, entonces podemos decir que B es una matriz Hermitiana, de la cual sabemos que los valores propios son reales y los vectores propios son ortogonales¹³.

Por otro lado, en la sección D.1 se presentó la definición de cuando una matriz es definida positiva, ver Definición 20, y se analizan varios algoritmos especializados para este tipo de matrices. En este caso, presentamos una pequeña extensión, es decir, la definición de una matriz semi-definida positiva:

Def 16 (Matriz semi-definida positiva). *La matriz A de $n \times n$ es definida positiva si $\mathbf{x}^* A \mathbf{x} \geq 0$ para todo vector no nulo \mathbf{x} y existe por lo menos un vector no nulo \mathbf{y} tal que $\mathbf{y}^* A \mathbf{y} = 0$.*

Lo interesante de esta definición es que podemos analizar la matriz B , es decir,

$$\begin{aligned} \mathbf{x}^* B \mathbf{x} &= \mathbf{x}^* A^* A \mathbf{x} \\ &= (A \mathbf{x})^* (A \mathbf{x}) \\ &= \|A \mathbf{x}\|_2^2 \geq 0. \end{aligned}$$

¹³Ver ejercicio 3 en la sección 1.6.

En este caso, sin embargo, solo podemos concluir que la matriz B es semi-definida positiva. Esto se debe a que si la matriz A (original¹⁴) no es full-rank, entonces no se puede asegurar que B es definida positiva. Otro punto importante de las matrices semi-definidas positivas es que sus valores propios son mayores o iguales a 0.

En particular, si consideramos que $A \in \mathbb{R}^{m \times n}$, entonces las ecuaciones normales se reducen a,

$$A^T A \bar{\mathbf{x}} = A^T \mathbf{b},$$

donde T es el operador transpuesta. La misma teoría explicada anteriormente aplica en este caso, es decir los vectores propios de $A^T A$ son ortogonales y reales, y los valores propios son reales y mayores o igual a 0.

El hecho de que B sea semi-definido positivo genera un problema cuando uno quiere resolver las ecuaciones normales, ya que la matriz inversa no existirá¹⁵. Lamentablemente esto no es algo que se pueda considerar como casos poco probables. Por ejemplo, un caso particular es el siguiente, considere que se quiere obtener con mínimos los coeficientes a y b de $y = a + bx$ donde la data es $(x_1, y_1), (x_1, y_2), \dots, (x_1, y_m)$, es decir, la variable x_1 es exactamente la misma para todos los datos. Esto implica que $A^T A$ sea singular¹⁶. Para resolver el problema, uno tiene por lo menos 2 alternativas:

1. Modificar el modelo, es decir reemplazar $y = a + bx$ por $y = a$. De esta forma el problema de mínimos cuadrados será otro pero no se obtendrá una matriz $A^T A$ singular.
2. Agregar un regularizador¹⁷ Esto significa convertir el problema de mínimos cuadrados original en el siguiente:

$$\underbrace{(A^T A + \delta^2 I)}_C \bar{\mathbf{x}}_\delta = A^T \mathbf{b},$$

donde $\delta \geq 0$ y I es la matriz identidad. Este problema modificado tiene las siguientes características,

- a) La matriz C es definida positiva, por lo cual ya no tenemos problema con la singularidad de $A^T A$.
- b) Agregarle $\delta^2 I$ significa que los valores propios de C son $\lambda_i + \delta^2$. Es decir, si $A^T A$ era singular, C ya no lo es. Recuerde que los valores propios de $A^T A$ son mayores o igual a 0, por lo que agregarle una constante, los aleja del 0.
- c) La solución obtenida, $\bar{\mathbf{x}}_\delta$, ya no es la solución de mínimos cuadrados original, $\bar{\mathbf{x}}$. Sin embargo se puede considerar una buena alternativa, más aún, se puede estudiar el comportamiento a medida que $\delta \rightarrow 0$.
- d) Recordando que el problema de mínimos cuadrados original puede escribirse como,

$$\bar{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{b} - A\mathbf{x}\|_2^2,$$

entonces la versión regularizada se puede expresar de la siguiente forma,

$$\bar{\mathbf{x}}_\delta = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{b} - A\mathbf{x}\|_2^2 + \delta \|\mathbf{x}\|_2^2 = \underset{\mathbf{x}}{\operatorname{argmin}} \left\| \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} A \\ \delta I \end{bmatrix} \mathbf{x} \right\|_2^2.$$

En resumen, los problemas de mínimos cuadrados son muy interesantes!

¹⁴No confundirse con la matriz A que aparece en la definición!

¹⁵Recuerde que no es necesario obtener la matriz inversa para resolver un sistema de ecuaciones lineales. La solución se puede obtener, por ejemplo usando PALU, sin embargo antes de usar PALU o otro *solver*, debemos estar seguros que el sistema de ecuaciones tiene solución. Por esta razón es importante saber que la inversa exista o mejor, saber que la matriz no es singular.

¹⁶Se sugiere hacer el ejercicio y construir la matriz $A^T A$ y determinar si es singular o no!

¹⁷Por ejemplo, un regularizador del tipo [Tikhonov](#).

6.6. Factorización QR

En esta sección estudiaremos la factorización matricial QR. Esta factorización es muy útil para resolver problemas de mínimos cuadrados ya que permite obtener la solución que minimiza el error cuadrático pero sin recurrir explícitamente a la computación de las ecuaciones normales. Esto no significa que las ecuaciones normales no sean importantes! Solo significa que existe un algoritmo especializado que fue diseñado para esa tarea. Por otro lado, la factorización QR también puede utilizarse para resolver sistemas de ecuaciones lineales cuadrados, por lo cual es conveniente tenerla en nuestra lista de herramientas de Computación Científica!

6.6.1. Estructura de Q y R

A diferencia de la factorización LU o PALU, la factorización QR impone la siguiente estructura sobre las matrices Q y R,

$$A = QR,$$

donde Q es una matriz unitaria, ver sección 1.4.3.3, y R es una matriz triangular superior, ver sección 1.4.3.2. La matriz Q que se requiere construir no es una matriz unitaria cualquiera, es en realidad la matriz Q tal que las columnas de Q, es decir, \mathbf{q}_i , generen el mismo sub-espacio que las columnas \mathbf{a}_i de A, es decir, que $\text{Range}(A) = \text{Range}(Q)$, ver sección 1.3.1 para más detalles.

Además es muy importante recordar el Teorema 1, por completitud se repite el teorema a continuación,

Thm 17. Para cualquier matriz $A \in \mathbb{C}^{m \times n}$ y matriz $Q \in \mathbb{C}^{m \times m}$ unitaria, tenemos:

$$\|QA\|_2 = \|A\|_2, \quad \|QA\|_F = \|A\|_F.$$

El cual nos indica que el producto por una matriz unitaria Q no modifica la norma 2 matricial ni la norma de Frobenius de la matriz resultante¹⁸. En particular, utilizaremos en esta sección la siguiente implicancia,

$$\|Q\mathbf{x}\|_2 = \|\mathbf{x}\|_2,$$

La cual indica que las matrices unitarias no modifican la norma 2 vectorial. Un corolario de este resultado es el siguiente teorema,

Thm 18. Para cualquier matriz $Q \in \mathbb{C}^{n \times n}$ unitaria se tiene que $|\lambda| = 1$, donde λ es un valor propio de Q.

Demostración. Sea λ un valor propio de Q y \mathbf{v} el vector propio asociado al valor propio λ , entonces,

$$\begin{aligned} Q\mathbf{v} &= \lambda\mathbf{v}, \\ \|Q\mathbf{v}\|_2 &= \|\lambda\mathbf{v}\|_2, \\ \|\mathbf{v}\|_2 &= |\lambda| \|\mathbf{v}\|_2. \end{aligned}$$

Considerando que un vector propio no puede ser el vector nulo, es decir $\|\mathbf{v}\|_2 \neq 0$, entonces,

$$\begin{aligned} \|\mathbf{v}\|_2 &= |\lambda| \|\mathbf{v}\|_2, \\ 1 &= |\lambda|. \end{aligned}$$

Lo cual es exactamente lo que buscábamos. □

¹⁸Es muy importante destacar que esto es válido solo para la norma 2, matricial y vectorial, y la norma matricial de Frobenius, para otras normas ya no se cumple.

Finalmente, pero no menos importante, debemos recordar una propiedad crucial de las matrices unitarias cuadradas descrita en la sección 1.4.3.3. Sea $Q \in \mathbb{C}^{m \times m}$, entonces,

$$Q^{-1} = Q^*.$$

Si $Q \in \mathbb{R}^{m \times m}$, entonces, $Q^{-1} = Q^T$. Esto significa que si quisiéramos resolver un sistema de ecuaciones lineales de la forma $Q\mathbf{x} = \mathbf{b}$, entonces la solución sería simplemente $\mathbf{x} = Q^* \mathbf{b}$. Es decir, no es necesario aplicar PALU y otro *solver*!

6.6.2. Ortonormalización de Gram-Schmidt

La ortonormalización de Gram-Schmidt es uno de varios¹⁹ algoritmos que nos entrega la factorización QR de una matriz A ²⁰, y será el procedimiento el cual estudiaremos en detalle. En particular, a partir de las columnas de la matriz A va construyendo columna a columna la matriz Q y coeficiente a coeficiente la matriz triangular superior R .

¡MUY IMPORTANTE!

En este punto es importante destacar que existen 2 variantes de la factorización QR para una matriz $A \in \mathbb{R}^{m \times n}$, con $m > n$. Una es la factorización QR-reducida, es decir $\hat{Q}\hat{R}$, y la otra es la factorización QR-*full*, es decir QR . Lo interesante es que ambas describen exactamente a la matriz A , es decir,

$$A = \hat{Q}\hat{R} = QR.$$

La diferencia es que $\hat{Q} \in \mathbb{R}^{m \times n}$ y $\hat{R} \in \mathbb{R}^{n \times n}$, donde $Q \in \mathbb{R}^{m \times m}$ y $R \in \mathbb{R}^{m \times n}$. En particular se cumple la siguiente relación,

$$Q = [\hat{Q}, \quad \check{Q}], \quad (6.15)$$

$$R = \begin{bmatrix} \hat{R} \\ \underline{0} \end{bmatrix}, \quad (6.16)$$

donde $\check{Q} = [\mathbf{q}_{n+1}, \dots, \mathbf{q}_m] \in \mathbb{R}^{m \times (m-n)}$ y $\underline{0}$ es la matriz nula de dimensión $(m-n) \times n$. Es decir, las primera n columnas de la matriz Q corresponden a la matriz \hat{Q} y las siguientes $(m-n)$ columnas, corresponde a la matriz \check{Q} y contiene vectores ortonormales respecto a las columnas de \hat{Q} y entre ellos mismos. Por otro lado, las primeras n filas de la matriz R corresponden a la matriz \hat{R} y luego las restantes filas son vectores nulos. Al hacer el producto QR podemos verificar que se obtiene lo mismo que entrega $\hat{Q}\hat{R}$, es decir A , entonces,

$$\begin{aligned} QR &= [\hat{Q}, \quad \check{Q}] \begin{bmatrix} \hat{R} \\ \underline{0} \end{bmatrix} \\ &= \hat{Q}\hat{R} + \check{Q}\underline{0} \\ &= \hat{Q}\hat{R} \\ &= A. \end{aligned}$$

Por lo cual primero construiremos la factorización QR-reducida, es decir $\hat{Q}\hat{R}$.

Para describir el algoritmo de Gram-Schmidt, considere la matriz $A \in \mathbb{R}^{m \times n}$, con $m > n$. La cual la podemos

¹⁹Ver https://en.wikipedia.org/wiki/QR_decomposition#Using_Householder_reflections para entender el uso de los reflectores de Householder para construir la factorización QR .

²⁰Por simplicidad trabajaremos con matrices reales.

describir convenientemente de la siguiente manera,

$$A = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n],$$

donde $\mathbf{a}_i \in \mathbb{R}^m$ para $i \in \{1, 2, \dots, n\}$, son los vectores columnas de la matriz A . Entonces, si escribimos de manera conveniente la identidad $A = \hat{Q}\hat{R}$, obtenemos,

$$[\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] = \underbrace{[\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n]}_{\hat{Q}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1n} \\ 0 & r_{22} & r_{23} & \dots & r_{2n} \\ 0 & 0 & r_{33} & \dots & r_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & r_{nn} \end{bmatrix}}_{\hat{R}}. \quad (6.17)$$

Recordando que Q es ortonormal, esto significa que sus columnas son ortogonales entre si y que la norma-2 de cada una de sus columnas es 1, tenemos las siguientes condiciones para las columnas de Q ,

$$\mathbf{q}_i^T \mathbf{q}_j = 0, \quad \text{para } i \neq j, \quad (6.18)$$

$$\|\mathbf{q}_i\|_2^2 = 1. \quad (6.19)$$

Por lo que surge la pregunta, ¿cómo efectivamente obtenemos \hat{Q} y \hat{R} ? Antes de llegar a responder esta pregunta, procedemos a multiplicar \hat{Q} por \hat{R} , entonces obtenemos,

$$[\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k, \dots, \mathbf{a}_n] = \begin{bmatrix} r_{11}\mathbf{q}_1, & r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2, & \dots, & \sum_{i=1}^k r_{ik}\mathbf{q}_i, & \dots, & \sum_{i=1}^n r_{in}\mathbf{q}_i \end{bmatrix}. \quad (6.20)$$

De la igualdad anterior, podemos utilizar el hecho que de cada columna del lado izquierdo debe ser igual a su correspondiente columna al lado derecho²¹, es decir tenemos las siguientes ecuaciones vectoriales,

$$\begin{aligned} \text{1ra columna} & : \mathbf{a}_1 = r_{11}\mathbf{q}_1, \\ \text{2da columna} & : \mathbf{a}_2 = r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2, \\ & \vdots \\ \text{k-ésima columna} & : \mathbf{a}_k = \sum_{i=1}^k r_{ik}\mathbf{q}_i, \\ & \vdots \\ \text{n-ésima columna} & : \mathbf{a}_n = \sum_{i=1}^n r_{in}\mathbf{q}_i. \end{aligned}$$

De la igualdad obtenida para la primera columna, tenemos lo siguiente,

$$\overbrace{\mathbf{a}_1}^{\text{Conocido}} = \underbrace{r_{11}}_{\text{Desconocido}} \underbrace{\mathbf{q}_1}_{\text{Desconocido}}. \quad (6.21)$$

¡Es decir tenemos 1 ecuación y 2 incógnitas! ¿Qué podemos hacer? Bueno, en realidad tenemos más ecuaciones, recuerde las ecuaciones (6.18) y (6.19). Entonces la pregunta ahora es: ¿Cómo usamos la ortonormalidad de los vectores \mathbf{q}_i para encontrar r_{11} y \mathbf{q}_1 ? En realidad tenemos 2 alternativas:

1. Multiplicar por la izquierda por \mathbf{q}_1^* , es decir obtener el producto interno con respecto a \mathbf{q}_1 .

²¹El mismo argumento es válido para las filas, sin embargo en este caso es más conveniente analizarlo columna a columna.

2. Obtener la norma 2 en ambos lados de la ecuación (6.21).

Ambas alternativas parecen razonable, por lo cual las debemos analizar en detalle. En el caso de la primera alternativa tenemos,

$$\begin{aligned} \mathbf{a}_1 &= r_{11} \mathbf{q}_1, \\ \mathbf{q}_1^* \mathbf{a}_1 &= r_{11} \underbrace{\mathbf{q}_1^T \mathbf{q}_1}_1, \\ \mathbf{q}_1^T \mathbf{a}_1 &= r_{11}, \end{aligned} \quad (6.22)$$

donde se utilizó el hecho que $\mathbf{q}_1^T \mathbf{q}_1 = \|\mathbf{q}_1\|_2^2 = 1$, es decir ecuación (6.19). Desafortunadamente esta alternativa no nos entrega algo nuevo, ya que tanto el lado izquierdo como el lado derecho de la ecuación tienen componentes desconocidas. Entonces debemos analizar la segunda alternativa,

$$\begin{aligned} \mathbf{a}_1 &= r_{11} \mathbf{q}_1, \\ \|\mathbf{a}_1\|_2 &= \|r_{11} \mathbf{q}_1\|_2, \\ \|\mathbf{a}_1\|_2 &= |r_{11}| \underbrace{\|\mathbf{q}_1\|_2}_1, \\ \|\mathbf{a}_1\|_2 &= |r_{11}|, \end{aligned}$$

donde utilizamos ahora el hecho que los vectores \mathbf{q}_i son unitarios nuevamente. En este caso sin embargo, sí podemos calcular el lado izquierdo de la ecuación, es decir $\|\mathbf{a}_1\|_2$. Por otro lado, el lado derecho nos indica que r_{11} podría ser $\|\mathbf{a}_1\|_2$ o $-\|\mathbf{a}_1\|_2$. Aquí es donde tenemos que tomar una decisión, la cual es elegir la alternativa positiva²², es decir,

$$r_{11} = \|\mathbf{a}_1\|_2$$

De aquí en adelante consideraremos que todos los coeficientes de la diagonal de la matriz \hat{R} son positivos, es decir $r_{ii} > 0$. Entonces ahora tenemos lo siguiente,

$$\underbrace{\mathbf{a}_1}_{\text{Conocido}} = \underbrace{r_{11}}_{\text{Conocido}} \underbrace{\mathbf{q}_1}_{\text{Desconocido}}.$$

Ahora solo nos queda determinar \mathbf{q}_1 . ¿Cómo encontramos \mathbf{q}_1 ? Simplemente despejándolo,

$$\mathbf{q}_1 = \frac{\mathbf{a}_1}{r_{11}}. \quad (6.23)$$

Entonces hemos exitosamente logrado determinar los 2 términos desconocidos de la ecuación (6.21), por lo cual tenemos,

$$\underbrace{\mathbf{a}_1}_{\text{Conocido}} = \underbrace{r_{11}}_{\text{Conocido}} \underbrace{\mathbf{q}_1}_{\text{Conocido}}.$$

Este es un gran logro! Y lo debemos festejar como tal! 😊. Ahora solo nos quedan $(n - 1)$ ecuaciones vectoriales que resolver.

Entonces, debemos resolver la segunda ecuación vectorial en la ecuación (6.20). Nuevamente podemos destacar las componentes conocidas y desconocidas,

$$\underbrace{\mathbf{a}_2}_{\text{Conocido}} = \underbrace{r_{12}}_{\text{Desconocido}} \underbrace{\mathbf{q}_1}_{\text{Conocido}} + \underbrace{r_{22}}_{\text{Desconocido}} \underbrace{\mathbf{q}_2}_{\text{Desconocido}} \quad (6.24)$$

²²La opción de elegir $r_{11} = -\|\mathbf{a}_1\|_2$ también es posible, pero consideraremos la opción positiva. Este es el punto preciso que demuestra que la factorización QR no es única, sin embargo si definimos que los $r_{ii} > 0$, entonces recuperamos la unicidad.

Si seguimos el mismo procedimiento anterior, podemos calcular la norma 2 en ambos lados²³, es decir,

$$\begin{aligned}
 \mathbf{a}_2 &= r_{12} \mathbf{q}_1 + r_{22} \mathbf{q}_2, \\
 \|\mathbf{a}_2\|_2 &= \sqrt{(r_{12} \mathbf{q}_1 + r_{22} \mathbf{q}_2)^T (r_{12} \mathbf{q}_1 + r_{22} \mathbf{q}_2)}, \\
 &= \sqrt{(r_{12} \mathbf{q}_1^T + r_{22} \mathbf{q}_2^T) (r_{12} \mathbf{q}_1 + r_{22} \mathbf{q}_2)}, \\
 &= \sqrt{\underbrace{r_{12}^2 \mathbf{q}_1^T \mathbf{q}_1}_1 + \underbrace{r_{22} r_{12} \mathbf{q}_2^T \mathbf{q}_1}_0 + \underbrace{r_{12} r_{22} \mathbf{q}_1^T \mathbf{q}_2}_0 + \underbrace{r_{22}^2 \mathbf{q}_2^T \mathbf{q}_2}_1}, \\
 &= \sqrt{r_{12}^2 + r_{22}^2}.
 \end{aligned}$$

Desafortunadamente llegamos a un problema similar al que llegamos antes, es decir, tenemos una ecuación y 2 incógnitas en $\|\mathbf{a}_2\|_2 = \sqrt{r_{12}^2 + r_{22}^2}$. ¿Qué alternativa nos queda? La alternativa que nos queda es usar la ortonormalidad de los vectores \mathbf{q}_1 y \mathbf{q}_2 a nuestro favor. Similar al caso anterior, tenemos 2 opciones: hacer el producto interno con respecto a \mathbf{q}_1 o con respecto a \mathbf{q}_2 . Similar la alternativa utilizada en la ecuación (6.22) y recordando que en ese momento no conocíamos \mathbf{q}_1 , sabemos que no es útil hacer el producto interno con respecto a un vector desconocido. Sin embargo, por completitud, haremos el ejercicio en este caso de hacer el producto interno con respecto a \mathbf{q}_2 , esto es,

$$\begin{aligned}
 \mathbf{a}_2 &= r_{12} \mathbf{q}_1 + r_{22} \mathbf{q}_2, \\
 \mathbf{q}_2^T \mathbf{a}_2 &= r_{12} \underbrace{\mathbf{q}_2^T \mathbf{q}_1}_0 + r_{22} \underbrace{\mathbf{q}_2^T \mathbf{q}_2}_1, \\
 \mathbf{q}_2^T \mathbf{a}_2 &= r_{22}.
 \end{aligned}$$

Como sospechábamos, obtenemos una ecuación que no nos ayuda a obtener ninguna información nueva, esto dado que tanto al lado izquierdo como al lado derecho tenemos términos desconocidos, i.e. \mathbf{q}_2 y r_{22} , respectivamente. Entonces, el camino natural es hacer el producto interno con respecto a \mathbf{q}_1 , dado que ya lo conocemos, entonces,

$$\begin{aligned}
 \mathbf{q}_1^T \mathbf{a}_2 &= r_{12} \underbrace{\mathbf{q}_1^T \mathbf{q}_1}_1 + r_{22} \underbrace{\mathbf{q}_1^T \mathbf{q}_2}_0, \\
 \mathbf{q}_1^T \mathbf{a}_2 &= r_{12}.
 \end{aligned}$$

Este camino sí nos entregó un resultado nuevo! Es decir nos entregó r_{12} . Entonces ahora podemos marcar r_{12} como conocido!

$$\underbrace{\mathbf{a}_2}_{\text{Conocido}} = \underbrace{r_{12}}_{\text{Conocido}} \underbrace{\mathbf{q}_1}_{\text{Conocido}} + \underbrace{r_{22}}_{\text{Desconocido}} \underbrace{\mathbf{q}_2}_{\text{Desconocido}}.$$

Lo interesante de este resultado es que ahora tenemos al lado derecho un término completamente conocido²⁴, es decir, ya conocemos el vector $r_{12} \mathbf{q}_1$. Entonces esto nos permite dejar al lado izquierdo de la ecuación todos los términos conocidos, es decir,

$$\underbrace{\mathbf{a}_2}_{\text{Conocido}} - \underbrace{r_{12}}_{\text{Conocido}} \underbrace{\mathbf{q}_1}_{\text{Conocido}} = \underbrace{r_{22}}_{\text{Desconocido}} \underbrace{\mathbf{q}_2}_{\text{Desconocido}}. \quad (6.25)$$

Lo interesante de este resultado es que tiene la misma estructura algebraica de un problema que ya resolvimos anteriormente. Nos referimos a la ecuación (6.21), que corresponde a la primera ecuación vectorial que resolvimos.

²³Recuerde que la norma 2 de un vector puede obtenerse como $\|\mathbf{x}\| = \sqrt{\mathbf{x}^* \mathbf{x}}$, y en caso que $\mathbf{x} \in \mathbb{R}^m$, entonces se usa T en vez de * .

²⁴No solo parcialmente conocido como antes, es decir, antes solo conocíamos \mathbf{q}_1 pero no conocíamos r_{12} , ahora conocemos ambos!

Recuerde que en ese caso teníamos un vector conocido al lado izquierdo y un vector desconocido al lado derecho, en particular, el lado derecho estaba formado por el coeficiente r_{11} y el vector \mathbf{q}_1 . Esta estructura es la misma que encontramos ahora en la ecuación (6.25). Entonces, podemos aplicar directamente el conocimiento adquirido anteriormente, es decir, obtener la norma 2 en ambos costados de la ecuación y considerar, en este caso, que $r_{22} > 0$, entonces,

$$\begin{aligned} \mathbf{a}_2 - r_{12} \mathbf{q}_1 &= r_{22} \mathbf{q}_2, \\ \|\mathbf{a}_2 - r_{12} \mathbf{q}_1\|_2 &= \|r_{22} \mathbf{q}_2\|_2, \\ &= |r_{22}| \|\mathbf{q}_2\|_2, \\ &= r_{22} \underbrace{\|\mathbf{q}_2\|_2}_1, \\ &= r_{22}. \end{aligned}$$

Por lo tanto $r_{22} = \|\mathbf{a}_2 - r_{12} \mathbf{q}_1\|_2$. Lo que nos permite actualizar la lista de términos conocidos y desconocidos de la ecuación (6.25),

$$\underbrace{\mathbf{a}_2}_{\text{Conocido}} - \underbrace{r_{12}}_{\text{Conocido}} \underbrace{\mathbf{q}_1}_{\text{Conocido}} = \underbrace{r_{22}}_{\text{Conocido}} \underbrace{\mathbf{q}_2}_{\text{Desconocido}}.$$

Entonces, para obtener \mathbf{q}_2 podemos aplicar el mismo procedimiento utilizado en la ecuación (6.23), es decir, despejar \mathbf{q}_2 ,

$$\mathbf{q}_2 = \frac{\mathbf{a}_2 - r_{12} \mathbf{q}_1}{r_{22}}.$$

Entonces hemos sido nuevamente capaces de encontrar todos los términos desconocidos! Es decir,

$$\underbrace{\mathbf{a}_2}_{\text{Conocido}} = \underbrace{r_{12}}_{\text{Conocido}} \underbrace{\mathbf{q}_1}_{\text{Conocido}} + \underbrace{r_{22}}_{\text{Conocido}} \underbrace{\mathbf{q}_2}_{\text{Conocido}}.$$

Nuevamente tenemos que celebrar! 😊. Continuando, ahora solo nos quedan $(n - 2)$ ecuaciones. Lo bueno, es que ya podemos aplicar directamente el conocimiento adquirido en la tercera ecuación. La tercera ecuación vectorial corresponde a,

$$\underbrace{\mathbf{a}_3}_{\text{Conocido}} = \underbrace{r_{13}}_{\text{Desconocido}} \underbrace{\mathbf{q}_1}_{\text{Conocido}} + \underbrace{r_{23}}_{\text{Desconocido}} \underbrace{\mathbf{q}_2}_{\text{Conocido}} + \underbrace{r_{33}}_{\text{Desconocido}} \underbrace{\mathbf{q}_3}_{\text{Desconocido}}. \quad (6.26)$$

De lo aprendido anteriormente podemos rápidamente obtener r_{13} de la siguiente forma,

$$\begin{aligned} \mathbf{a}_3 &= r_{13} \mathbf{q}_1 + r_{23} \mathbf{q}_2 + r_{33} \mathbf{q}_3, \\ \mathbf{q}_1^T \mathbf{a}_3 &= r_{13} \mathbf{q}_1^T \mathbf{q}_1 + r_{23} \mathbf{q}_1^T \mathbf{q}_2 + r_{33} \mathbf{q}_1^T \mathbf{q}_3, \\ &= r_{13} \underbrace{\mathbf{q}_1^T \mathbf{q}_1}_1 + r_{23} \underbrace{\mathbf{q}_1^T \mathbf{q}_2}_0 + r_{33} \underbrace{\mathbf{q}_1^T \mathbf{q}_3}_0, \\ &= r_{13}. \end{aligned}$$

Por lo tanto $r_{13} = \mathbf{q}_1^T \mathbf{a}_3$. De la misma manera podemos obtener r_{23} , es decir,

$$\begin{aligned} \mathbf{a}_3 &= r_{13} \mathbf{q}_1 + r_{23} \mathbf{q}_2 + r_{33} \mathbf{q}_3, \\ \mathbf{q}_2^T \mathbf{a}_3 &= r_{13} \mathbf{q}_2^T \mathbf{q}_1 + r_{23} \mathbf{q}_2^T \mathbf{q}_2 + r_{33} \mathbf{q}_2^T \mathbf{q}_3, \\ &= r_{13} \underbrace{\mathbf{q}_2^T \mathbf{q}_1}_0 + r_{23} \underbrace{\mathbf{q}_2^T \mathbf{q}_2}_1 + r_{33} \underbrace{\mathbf{q}_2^T \mathbf{q}_3}_0, \\ &= r_{23}. \end{aligned} \quad (6.27)$$

Por lo tanto $r_{23} = \mathbf{q}_2^T \mathbf{a}_3$. Luego podemos obtener r_{33} moviendo todos los términos conocidos al lado izquierdo y calculando la norma 2 en ambos lados,

$$\begin{aligned} \mathbf{a}_3 - r_{13} \mathbf{q}_1 - r_{23} \mathbf{q}_2 &= r_{33} \mathbf{q}_3, \\ \|\mathbf{a}_3 - r_{13} \mathbf{q}_1 - r_{23} \mathbf{q}_2\|_2 &= r_{33} \|\mathbf{q}_3\|_2, \\ &= r_{33}. \end{aligned}$$

Por lo tanto $r_{33} = \|\mathbf{a}_3 - r_{13} \mathbf{q}_1 - r_{23} \mathbf{q}_2\|_2$. Finalmente obtenemos \mathbf{q}_3 despejando, es decir,

$$\mathbf{q}_3 = \frac{\mathbf{a}_3 - r_{13} \mathbf{q}_1 - r_{23} \mathbf{q}_2}{r_{33}}.$$

Por lo tanto, una vez más, hemos obtenido todos los términos desconocidos! 😊. Entonces,

$$\underbrace{\mathbf{a}_3}_{\text{Conocido}} = \underbrace{r_{13}}_{\text{Conocido}} \underbrace{\mathbf{q}_1}_{\text{Conocido}} + \underbrace{r_{23}}_{\text{Conocido}} \underbrace{\mathbf{q}_2}_{\text{Conocido}} + \underbrace{r_{33}}_{\text{Conocido}} \underbrace{\mathbf{q}_3}_{\text{Conocido}}.$$

De forma general, en el k -ésimo caso obtenemos la siguiente ecuación,

$$\underbrace{\mathbf{a}_k}_{\text{Conocido}} = \underbrace{r_{1,k}}_{\text{Desconocido}} \underbrace{\mathbf{q}_1}_{\text{Conocido}} + \underbrace{r_{2,k}}_{\text{Desconocido}} \underbrace{\mathbf{q}_2}_{\text{Conocido}} + \underbrace{r_{3,k}}_{\text{Desconocido}} \underbrace{\mathbf{q}_3}_{\text{Conocido}} + \cdots + \underbrace{r_{k-1,k}}_{\text{Desconocido}} \underbrace{\mathbf{q}_{k-1}}_{\text{Conocido}} + \underbrace{r_{k,k}}_{\text{Desconocido}} \underbrace{\mathbf{q}_k}_{\text{Desconocido}}.$$

Donde podemos determinar la expresión para cada término desconocido de la siguiente forma,

$$\begin{aligned} r_{i,k} &= \mathbf{q}_i^T \mathbf{a}_k, \quad \text{para } i \in \{1, 2, \dots, k-1\}, \\ r_{k,k} &= \left\| \mathbf{a}_k - \sum_{i=1}^{k-1} r_{i,k} \mathbf{q}_i \right\|_2, \\ \mathbf{q}_k &= \frac{\mathbf{a}_k - \sum_{i=1}^{k-1} r_{i,k} \mathbf{q}_i}{r_{k,k}}. \end{aligned}$$

Lo cual es válido para $k \in \{1, 2, \dots, n\}$. El procedimiento construido corresponde a la *Ortonormalización Clásica de Gram-Schmidt*, el cual se formaliza en el Algoritmo 5. En el Algoritmo 5 se considera que uno entrega una colección de n vectores \mathbf{a}_j linealmente independiente. Una pregunta natural que surge entonces es: ¿Qué pasaría si hubieran vectores linealmente dependientes?²⁵

```

1 for k in range(1, n+1):
2     y =  $\mathbf{a}_k$ 
3     for i in range(1, k):
4          $r_{ik} = \mathbf{q}_i^T \mathbf{a}_k$ 
5          $\mathbf{y} = \mathbf{y} - r_{ik} \mathbf{q}_i$ 
6      $r_{k,k} = \|\mathbf{y}\|_2$ 
7      $\mathbf{q}_k = \frac{\mathbf{y}}{r_{k,k}}$ 

```

Algoritmo 5: Ortonormalización Clásica de Gram-Schmidt.

²⁵La respuesta es simple, se obtendrá que algún coeficiente $r_{k,k}$ será nulo y el procedimiento no podrá continuar. Esto sirve entonces también para determinar si un conjunto de vectores son linealmente independientes!

6.6.3. Ortonormalización de Gram-Schmidt modificada

La factorización QR por medio de la ortonormalización de Gram-Schmidt clásica considerando aritmética exacta obtiene efectivamente lo que se necesita, sin embargo al utilizar aritmética de punto flotante, por ejemplo *double precision*, hay que tener cuidado. En particular, la ortonormalización de Gram-Schmidt clásica en *double precision* no asegura la ortogonalidad de los vectores \mathbf{q}_i a medida que el algoritmo procede. Esto en realidad es un gran problema porque ya no serían válido los supuestos considerados en la construcción del algoritmo mismo. Si bien los problemas que la aritmética de punto flotante trae consigo no se pueden evitar completamente²⁶, si podemos controlarlos y aminorarlos por medio de entender su origen.

En esta sección presentaremos la ortonormalización de Gram-Schmidt **modificada**, que es una pequeña variación de la versión clásica pero es muy significativa. Antes de proceder es importante destacar que la versión clásica y modifica de la ortonormalización de Gram-Schmidt hacen exactamente lo mismo para resolver las primeras 2 ecuaciones vectoriales, es decir ecuaciones (6.21) y (6.24), por lo cual se analizará la tercera ecuación vectorial, es decir ecuación (6.26), en particular el paso realizado en la ecuación (6.27), y luego se generalizará. Entonces, por completitud, la tercera ecuación es,

$$\underbrace{\mathbf{a}_3}_{\text{Conocido}} = \underbrace{r_{13}}_{\text{Desconocido}} \underbrace{\mathbf{q}_1}_{\text{Conocido}} + \underbrace{r_{23}}_{\text{Desconocido}} \underbrace{\mathbf{q}_2}_{\text{Conocido}} + \underbrace{r_{33}}_{\text{Desconocido}} \underbrace{\mathbf{q}_3}_{\text{Desconocido}}.$$

Al igual que en el caso anterior, acá también se obtiene $r_{13} = \mathbf{q}_1^T \mathbf{a}_3$. La gran diferencia ocurre al obtener r_{23} , y de aquí en adelante. En el caso anterior se consideró que por ortogonalidad $\mathbf{q}_2^T \mathbf{q}_1 = 0$. Ahora, dado que operaremos en *double precision*, no haremos tal consideración. Lo cual significa la siguiente modificación,

$$\begin{aligned} \mathbf{a}_3 &= r_{13} \mathbf{q}_1 + r_{23} \mathbf{q}_2 + r_{33} \mathbf{q}_3, \\ \mathbf{a}_3 - r_{13} \mathbf{q}_1 &= r_{23} \mathbf{q}_2 + r_{33} \mathbf{q}_3, \\ \mathbf{q}_2^T (\mathbf{a}_3 - r_{13} \mathbf{q}_1) &= r_{23} \mathbf{q}_2^T \mathbf{q}_2 + r_{33} \mathbf{q}_2^T \mathbf{q}_3, \\ &= r_{23} \underbrace{\mathbf{q}_2^T \mathbf{q}_2}_1 + r_{33} \underbrace{\mathbf{q}_2^T \mathbf{q}_3}_0, \\ &= r_{23}. \end{aligned}$$

Lo que nos da que $r_{23} = \mathbf{q}_2^T (\mathbf{a}_3 - r_{13} \mathbf{q}_1)$, lo cual es exactamente igual, desde el punto de vista Matemático, a la expresión antes obtenida. Sin embargo desde el punto de vista computacional la diferencia es muy importante! Es importante destacar los siguientes puntos sobre esta modificación,

- Aún se considera la ortogonalidad con respecto a los vectores \mathbf{q}_j desconocidos, en este caso se consideró en $\mathbf{q}_2^T \mathbf{q}_3 = 0$.
- Una forma de explicar en palabras simples lo que está ocurriendo con la modificación es que al considerar $r_{13} \mathbf{q}_1$ en $(\mathbf{a}_3 - r_{13} \mathbf{q}_1)$ y luego multiplicar por la izquierda por \mathbf{q}_2^T , el coeficiente r_{23} y posteriormente el coeficiente r_{33} y vector \mathbf{q}_3 van capturando y autocorrigiendo la inexactitud de la computación.

En resumen, la modificación es bastante pequeña, pero muy importante. Esto se reduce en el Algoritmo 6, la diferencia ocurren en la línea 4 del algoritmo.

6.6.4. ¿Por qué y cómo usamos la factorización QR para encontrar la solución que minimiza el error cuadrático?

La primera pregunta que debemos responder entonces es la siguiente: ¿Por qué usar la factorización QR si podemos usar las ecuaciones normales directamente? La respuesta es nuevamente desde el aspecto computacional.

²⁶Quizás pueden haber ejemplos particulares donde no afecten, pero no es el caso general.

```

1 for k in range(1, n+1):
2     y = ak
3     for i in range(1, k):
4          $r_{ik} = \mathbf{q}_i^T \mathbf{y}$ 
5         y = y -  $r_{ik} \mathbf{q}_i$ 
6      $r_{k,k} = \|\mathbf{y}\|_2$ 
7      $\mathbf{q}_k = \frac{\mathbf{y}}{r_{k,k}}$ 

```

Algoritmo 6: Ortonormalización **Modificada** de Gram-Schmidt.

Si bien las ecuaciones normales nos entregan el sistema de ecuaciones lineales asociados a la solución que minimiza el error cuadrático, estas requiere resolver $A^T A \bar{\mathbf{x}} = A^T \mathbf{b}$. Entonces, si generalizamos la noción de número de condición $\kappa(A)$ a matrices rectangulares por medio del uso de los valores singulares, tenemos,

$$\kappa_2(A) = \|A\|_2 \left\| A^\dagger \right\|_2 = \frac{\sigma_1}{\sigma_n}.$$

Entonces, si trabajamos con las ecuaciones normales, estamos elevando al cuadrado el número de condición!

$$\kappa_2(A^T A) = \|A^T A\|_2 \|(A^T A)^{-1}\|_2 = \frac{\sigma_1^2}{\sigma_n^2}.$$

¡Lo cual no es bueno para la computación! Entonces, es muy adecuado trabajar con algoritmos que no requieran explícitamente de las ecuaciones normales pero que de todos modos encuentren la solución que minimiza el error cuadrático! Esto no es una contradicción, es en realidad la importancia de los resultados teóricos vs los prácticos. Por ejemplo podemos recordar la importancia teórica de la matriz de Vandermonde en interpolación polinomial, sin embargo en la práctica usamos otros algoritmos para efectivamente hacer interpolación polinomial.

Entonces, ahora es el turno de responder la segunda pregunta: ¿cómo usamos la factorización QR para encontrar la solución que minimiza el error cuadrático? Para responder esta pregunta, debemos retomar el problema original, es decir,

$$A\mathbf{x} = \mathbf{b}, \quad (6.28)$$

donde $A \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$, y $\mathbf{b} \in \mathbb{R}^m$. Ahora, consideremos que construimos la factorización QR-full de A y la reemplazamos en la ecuación (6.28), entonces,

$$\begin{aligned} A\mathbf{x} &= \mathbf{b}, \\ QR\mathbf{x} &= \mathbf{b}. \end{aligned}$$

Entonces, el camino natural a seguir es construir las ecuaciones normales. Notar que esto no contradice lo mencionado anteriormente, solo estamos indicando que aplicaremos las ecuaciones normales para demostrar que aún estamos obteniendo la misma solución obtenida por las ecuaciones normales. Más aún, la utilización de las ecuaciones normales nos entregará el algoritmo final donde ya no será necesario recurrir a las ecuaciones normales. Entonces, se obtiene la siguiente identidad,

$$\begin{aligned} A^T A \bar{\mathbf{x}} &= A^T \mathbf{b}, \\ (QR)^T QR \bar{\mathbf{x}} &= (QR)^T \mathbf{b}, \\ R^T Q^T QR \bar{\mathbf{x}} &= R^T Q^T \mathbf{b}, \\ R^T \underbrace{Q^T Q}_I R \bar{\mathbf{x}} &= R^T Q^T \mathbf{b}, \\ R^T R \bar{\mathbf{x}} &= R^T Q^T \mathbf{b}, \end{aligned}$$

donde se utilizó la condición de que Q es unitaria, por lo que $Q^T Q = I$, es decir la matriz identidad. Ahora, recordando la relación entre las versiones *full* y *reducidas* de las matrices Q y R en las ecuaciones (6.15) y (6.16), respectivamente, podemos utilizarlas para simplificar aún más las ecuaciones obtenidas,

$$\begin{aligned} \begin{bmatrix} \hat{R}^T & \underline{0}^T \end{bmatrix} \begin{bmatrix} \hat{R} \\ \underline{0} \end{bmatrix} \bar{\mathbf{x}} &= \begin{bmatrix} \hat{R}^T & \underline{0}^T \end{bmatrix} \begin{bmatrix} \hat{Q}^T \\ \check{Q}^T \end{bmatrix} \mathbf{b} \\ \left(\hat{R}^T \hat{R} + \underline{0}^T \underline{0} \right) \bar{\mathbf{x}} &= \begin{bmatrix} \hat{R}^T & \underline{0}^T \end{bmatrix} \begin{bmatrix} \hat{Q}^T \mathbf{b} \\ \check{Q}^T \mathbf{b} \end{bmatrix} \\ \hat{R}^T \hat{R} \bar{\mathbf{x}} &= \hat{R}^T \hat{Q}^T \mathbf{b} + \underline{0}^T \check{Q}^T \mathbf{b} \\ \hat{R}^T \hat{R} \bar{\mathbf{x}} &= \hat{R}^T \hat{Q}^T \mathbf{b}. \end{aligned}$$

Recordando que ahora el sistema de ecuaciones lineales es cuadrado y que \hat{R}^T es cuadrada también, podemos multiplicar por la izquierda por \hat{R}^{-T} y finalmente obtenemos,

$$\hat{R} \bar{\mathbf{x}} = \hat{Q}^T \mathbf{b}.$$

Incluso podemos expresar la solución de la siguiente forma,

$$\bar{\mathbf{x}} = \hat{R}^{-1} \hat{Q}^T \mathbf{b}.$$

Es importante recordar que \hat{R} es una matriz triangular superior, por lo que no es necesario obtener su inversa, solo es necesario utilizar *backward substitution*!

Por último, se presenta un resultado muy interesante, que corresponde al valor explícito del mínimo obtenido, es decir,

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{b} - A\mathbf{x}\|_2^2 &= \|\mathbf{b} - A\bar{\mathbf{x}}\|_2^2 \\ &= \|\check{Q}^T \mathbf{b}\|_2^2. \end{aligned}$$

El cual se puede obtener reemplazando²⁷ en $\|\mathbf{b} - A\bar{\mathbf{x}}\|_2^2$ por $\bar{\mathbf{x}} = \hat{R}^{-1} \hat{Q}^T \mathbf{b}$.

6.6.5. Ejemplo numérico

En esta sección se desarrollará el mismo ejemplo numérico desarrollado en la sección 6.3.1, utilizando las ecuaciones normales, el cual corresponde a la ecuación (6.12), y también corresponde al ejercicio (4.14) del libro guía. Por completitud se presenta nuevamente el sistema de ecuaciones lineales sobre-determinado,

$$\underbrace{\begin{bmatrix} 1 & -4 \\ 2 & 3 \\ 2 & 2 \end{bmatrix}}_A \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\mathbf{x}} \approx \underbrace{\begin{bmatrix} -3 \\ 15 \\ 9 \end{bmatrix}}_{\mathbf{b}}.$$

En este caso particular no habrá diferencia si utilizamos la ortonormalización de Gram-Schmidt clásica o modificada porque solo tenemos 2 vectores columnas. De todos modos se presentan los resultados parciales para obtener

²⁷Es un buen ejercicio algebraico obtener este resultado!

la factorización QR reducida de A ,

$$\begin{aligned}
 r_{11} &= \|\mathbf{a}_1\|_2 = 3, \\
 \mathbf{q}_1 &= \frac{\mathbf{a}_1}{r_{11}} = [1/3, 2/3, 2/3]^T, \\
 r_{12} &= \mathbf{q}_1^T \mathbf{a}_2 = 2, \\
 \mathbf{y} &= \mathbf{a}_2 - r_{12} \mathbf{q}_1, \\
 r_{22} &= \|\mathbf{y}\|_2 = 5, \\
 \mathbf{q}_2 &= \frac{\mathbf{y}}{r_{22}} = [-14/15, 1/3, 2/15]^T.
 \end{aligned}$$

Lo que nos da la factorización QR-reducida de A ,

$$A = \begin{bmatrix} 1 & -4 \\ 2 & 3 \\ 2 & 2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1/3 & -14/15 \\ 2/3 & 1/3 \\ 2/3 & 2/15 \end{bmatrix}}_{\hat{Q}} \underbrace{\begin{bmatrix} 3 & 2 \\ 0 & 5 \end{bmatrix}}_{\hat{R}}.$$

Entonces ahora necesitamos obtener,

$$\begin{aligned}
 \hat{Q}^T \mathbf{b} &= \begin{bmatrix} 1/3 & 2/3 & 2/3 \\ -14/15 & 1/3 & 2/15 \end{bmatrix} \begin{bmatrix} -3 \\ 15 \\ 9 \end{bmatrix} \\
 &= \begin{bmatrix} 15 \\ 9 \end{bmatrix}.
 \end{aligned}$$

Finalmente debemos resolver el sistema de ecuaciones lineales triangular,

$$\begin{aligned}
 \hat{R} \bar{\mathbf{x}} &= \hat{Q}^T \mathbf{b}, \\
 \bar{\mathbf{x}} &= \hat{R}^{-1} \hat{Q}^T \mathbf{b} \\
 &= \begin{bmatrix} 3 & 2 \\ 0 & 5 \end{bmatrix}^{-1} \begin{bmatrix} 15 \\ 9 \end{bmatrix} \\
 &= \begin{bmatrix} 19/5 \\ 9/5 \end{bmatrix} \\
 &= \begin{bmatrix} 3,8 \\ 1,8 \end{bmatrix}.
 \end{aligned}$$

Lo cual es exactamente lo mismo que se obtuvo anteriormente en las ecuaciones (6.13) y (6.14), respectivamente.

Capítulo 7

GMRes: Método del residuo mínimo generalizado

En este capítulo estudiaremos [GMRes](#), que viene del inglés *Generalized Minimal Residual method*. Este algoritmo se une a los algoritmos ya estudiados en el Capítulo 4 para resolver sistemas de ecuaciones lineales cuadrados y es un pariente del método del Gradiente Conjugado, ver sección D.3. La gran diferencia respecto al método del Gradiente Conjugado recae en que GMRes no requiere que la matriz A sea definida positiva, sin embargo, ambos métodos son muy interesantes.

En particular, GMRes pertenece a la categoría de algoritmos que utilizan sub-espacios de [Krylov](#) para encontrar la solución a un sistema de ecuaciones lineales no singular. Una característica importante de GMRes, es que es un algoritmo iterativo/directo, es decir entrega soluciones parciales a medida que itera, sin embargo asegura entregar la solución única de un sistema de ecuaciones lineales no singular cuadrado de $n \times n$ a lo más en n iteraciones! Es decir, une el mundo de los algoritmos directos, como PALU, con el mundo de los métodos iterativos, como el método de Jacobi y Gauss-Seidel.

7.1. Motivación

Antes de revisar los detalles específicos sobre GMRes, es conveniente revisar el teorema de Cayley-Hamilton.

Thm 19 (Teorema de Cayley–Hamilton). *El teorema de Cayley-Hamilton dice que una matriz de dimensiones $n \times n$ es aniquilada por su [polinomio característico](#) $p(\lambda) = \det(\lambda I - A)$, el cual es mónico de grado n . Referencias: [Wolfram](#) y [Wikipedia](#).*

Preliminarmente, no se ve una relación directa con GMRes. Sin embargo, debemos analizar paso a paso sus implicancias. Por ejemplo, el polinomio característico de una matriz $A \in \mathbb{R}^{n \times n}$ se puede expresar de la siguiente forma:

$$p(\lambda) = \det(\lambda I - A) = \lambda^n + \check{c}_{n-1} \lambda^{n-1} + \cdots + \check{c}_1 \lambda + (-1)^n \det(A),$$

donde $\lambda \in \mathbb{C}$, “det” es el determinante e I es la matriz identidad de dimensiones $n \times n$. Ahora, si evaluamos el polinomio característico $p(\lambda)$ con la matriz A obtenemos:

$$p(A) = A^n + \check{c}_{n-1} A^{n-1} + \cdots + \check{c}_1 A + (-1)^n \det(A) I.$$

Lo que indica el teorema de Cayley-Hamilton es que el resultado de la evaluación es la matriz nula¹, es decir:

$$p(A) = A^n + \check{c}_{n-1} A^{n-1} + \cdots + \check{c}_1 A + (-1)^n \det(A) I = \underline{\underline{0}}, \quad (7.1)$$

donde $\underline{\underline{0}}$ es la matriz de 0 de dimensiones $n \times n$. Ahora, La ecuación (7.1) nos permite obtener una expresión para la matriz inversa de A de la siguiente forma,

$$\begin{aligned} A^n + \check{c}_{n-1} A^{n-1} + \cdots + \check{c}_1 A + (-1)^n \det(A) I &= \underline{\underline{0}}, \\ A^n + \check{c}_{n-1} A^{n-1} + \cdots + \check{c}_1 A &= (-1)^{n-1} \det(A) I, \\ A^{n-1} + \check{c}_{n-1} A^{n-2} + \cdots + \check{c}_1 I &= (-1)^{n-1} \det(A) A^{-1}, \end{aligned}$$

entonces,

$$A^{-1} = \frac{(-1)^{n-1}}{\det(A)} (A^{n-1} + \check{c}_{n-1} A^{n-2} + \cdots + \check{c}_2 A + \check{c}_1 I). \quad (7.2)$$

Lo interesante de este resultado es que hemos obtenido una relación explícita entre la matriz inversa de A , es decir A^{-1} , y una combinación lineal de las potencias de A . Es importante notar que es poco práctico obtener la inversa de esta forma, pero ya veremos su utilidad.

Consideremos ahora que queremos resolver nuestro clásico sistema de ecuaciones lineales,

$$A\mathbf{x} = \mathbf{b}, \quad (7.3)$$

donde $\mathbf{x} \in \mathbb{R}^n$ y $\mathbf{b} \in \mathbb{R}^n$. Entonces, considerando que A es no singular, sabemos que podemos expresar la solución \mathbf{x} de la siguiente forma,

$$\mathbf{x} = A^{-1} \mathbf{b}. \quad (7.4)$$

Ahora, si multiplicamos la ecuación (7.2) por \mathbf{b} por la derecha obtenemos,

$$\begin{aligned} A^{-1} \mathbf{b} &= \frac{(-1)^{n-1}}{\det(A)} (A^{n-1} + \check{c}_{n-1} A^{n-2} + \cdots + \check{c}_2 A + \check{c}_1 I) \mathbf{b} \\ &= \frac{(-1)^{n-1}}{\det(A)} (A^{n-1} \mathbf{b} + \check{c}_{n-1} A^{n-2} \mathbf{b} + \cdots + \check{c}_2 A \mathbf{b} + \check{c}_1 \mathbf{b}). \end{aligned}$$

Este resultado nos entrega una relación muy interesante, es decir, recordando la ecuación (7.4) que indica $\mathbf{x} = A^{-1} \mathbf{b}$, entonces podemos concluir que,

$$\begin{aligned} \mathbf{x} = A^{-1} \mathbf{b} &= \frac{(-1)^{n-1}}{\det(A)} (A^{n-1} \mathbf{b} + \check{c}_{n-1} A^{n-2} \mathbf{b} + \cdots + \check{c}_2 A \mathbf{b} + \check{c}_1 \mathbf{b}) \\ &= \sum_{i=1}^n \check{c}_i A^{i-1} \mathbf{b}. \end{aligned} \quad (7.5)$$

Lo cual nos demuestra que podemos representar la solución del sistema de ecuaciones lineales $A\mathbf{x} = \mathbf{b}$ por medio de una combinación lineal de los vectores: \mathbf{b} , $A\mathbf{b}$, $A^2\mathbf{b}$, ..., $A^{n-1}\mathbf{b}$. Notar que en la primera parte de la expresión anterior se utilizó la notación \check{c}_i y en la segunda parte \hat{c}_i , esto no es un error, es solo que \hat{c}_i incluye el factor $\frac{(-1)^{n-1}}{\det(A)}$. Además se considera que, como A es no singular, $A^0 = I$, es decir la matriz identidad. Ahora, si en vez de considerar los n vectores antes descritos, consideramos el sub-espacio vectorial generado por los primeros k , es decir,

$$\mathcal{K}_k = \text{span}(\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, A^3\mathbf{b}, \dots, A^{k-1}\mathbf{b}), \quad (7.6)$$

¹Una forma de obtener este resultado es considerar que uno tiene la descomposición de valores propios $A = V \Lambda V^{-1}$, donde las columnas de V son los vectores propios de A y $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$, es decir es la matriz diagonal con los valores propios de A . Recuerde que cada vector propio y valor propio de A satisfacen la siguiente ecuación $A\mathbf{v}_i = \lambda_i \mathbf{v}_i$. Entonces al reemplazar la descomposición de valores propios uno llega a la siguiente representación $p(A) = V \text{diag}(p(\lambda_1), p(\lambda_2), \dots, p(\lambda_n)) V^{-1}$, la que efectivamente es la matriz nula por definición del polinomio característico.

donde \mathcal{K}_k se conoce como el sub-espacio de Krylov. Entonces podemos introducir GMRes como el algoritmo que busca una aproximación de \mathbf{x} restringiéndolo al sub-espacio de Krylov \mathcal{K}_k , es decir, busca construir una aproximación \mathbf{x}_k tal que minimice el error cuadrático $\|\mathbf{b} - A\mathbf{x}_k\|_2^2$.

Lo fantástico de este algoritmo, que se mencionó al comienzo de este capítulo y que ahora podemos entender, es que si $k = n$ se asegura que seremos capaces de obtener efectivamente la solución exacta! 😊

7.2. Derivación de GMRes

En la introducción a este capítulo y en la sección anterior se presentó que GMRes es un algoritmo que resuelve un sistema de ecuaciones lineales, $A\mathbf{x} = \mathbf{b}$ de forma iterativa pero que, en aritmética exacta, llega a la solución en a lo más n iteraciones. La forma en que logra esto es restringiendo el espacio de búsqueda de la solución y resolviendo un problema de mínimos cuadrados equivalente. Si bien la restricción del espacio de búsqueda puede ser arbitraria, es decir, en principio se podría elegir cualquier conjunto de vectores linealmente independientes y luego resolver el problema de mínimos cuadrados asociado, en este caso se utilizará el sub-espacio de Krylov \mathcal{K}_k . Por lo tanto, la derivación que se presentará considera que en la k -ésima iteración del algoritmo $\mathbf{x}_k \in \mathcal{K}_k$. Al final de esta sección se verá la importancia de esta elección y se entenderá porque es una elección muy conveniente.

Antes de empezar con la derivación de método de GMRes debemos recordar las dimensiones de los elementos con que estaremos trabajando:

$$\begin{aligned} A &\in \mathbb{R}^{n \times n}, \\ \mathbf{x} &\in \mathbb{R}^n, \\ \mathbf{b} &\in \mathbb{R}^n, \\ A^j \mathbf{b} &\in \mathbb{R}^n, \quad j \in \{0, 1, 2, \dots, k-1\}. \end{aligned}$$

Por completitud presentamos nuevamente el sub-espacio de Krylov \mathcal{K}_k ya definido en la ecuación (7.6),

$$\mathcal{K}_k = \text{span}(\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, A^3\mathbf{b}, \dots, A^{k-1}\mathbf{b}),$$

del cual podemos concluir que es un sub-espacio vectorial de \mathbb{R}^n de dimensión k . Es decir, cada elemento de $\mathcal{K}_k \in \mathbb{R}^n$ sin embargo al solo tener k vectores linealmente independientes, no se puede representar todos los elementos en \mathbb{R}^n . Sí los podrá representar todos cuando $k = n$.

¡MUY IMPORTANTE!

Un punto clave de GMRes, es decir del sub-espacio de Krylov \mathcal{K}_k , es que en realidad es suficiente que $\mathbf{x} \in \mathcal{K}_k$. Esto significa, que solo requerimos que la solución^a del sistema de ecuaciones lineales asociado esté en \mathcal{K}_k ! Si efectivamente esto ocurre, entonces GMRes terminará solo en k iteraciones y entregará la solución del sistema de ecuaciones lineales, no una aproximación! 😊

^aque no conocemos y que andamos buscando

Ahora, que ya hemos destacado la importancia del sub-espacio de Krylov, debemos entender como utilizarlo. Por lo tanto, la pregunta que debemos responder ahora es: ¿Qué significa que $\mathbf{x}_k \in \mathcal{K}_k$? La respuesta es simple, es decir, significa que el vector \mathbf{x}_k debe ser representado como una combinación lineal de un conjunto de vectores basales que generen el sub-espacio \mathcal{K}_k . Notar que se indica “un” conjunto de vectores, porque en realidad no existe un único conjunto. Más aún, veremos que los primeros vectores que utilizamos para definir \mathcal{K}_k no son muy buenos desde el punto de vista numérico. Sin embargo, son importantes desde el punto de vista teórico, en resumen, \mathbf{x}_k se puede representar de la siguiente forma,

$$\mathbf{x}_k = \tilde{c}_1 \mathbf{b} + \tilde{c}_2 A\mathbf{b} + \dots + \tilde{c}_k A^{k-1} \mathbf{b}.$$

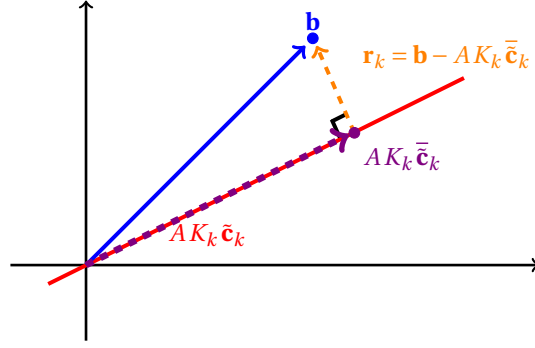


Figura 7.1: Esta Figura preserva la misma definición de colores incluida en las Figuras 6.2 y 6.3. La única diferencia es que se utiliza color naranja para destacar el vector residual. Es importante destacar además que en este caso se considera que \mathbf{x}_k está restringido al sub-espacio de Krylov \mathcal{K}_k , es decir $\mathbf{x}_k = K_k \tilde{\mathbf{c}}_k$.

La cual también puede ser escrita matricialmente como,

$$\mathbf{x}_k = \underbrace{[\mathbf{b}, \quad A\mathbf{b}, \quad \dots, \quad A^{k-1}\mathbf{b}]}_{K_k} \underbrace{\begin{bmatrix} \tilde{c}_1 \\ \tilde{c}_2 \\ \vdots \\ \tilde{c}_k \end{bmatrix}}_{\tilde{\mathbf{c}}_k} = K_k \tilde{\mathbf{c}}_k.$$

Entonces, ya entendemos lo que significa que $\mathbf{x}_k \in \mathcal{K}_k$. Ahora la pregunta que surge es la siguiente, ¿Cómo utilizamos esta representación para encontrar \mathbf{x}_k con mínimos cuadrados? Para responder esta pregunta debemos recordar que $\mathbf{x} \approx \mathbf{x}_k$, entonces reemplazando en el sistema de ecuaciones lineales original obtenemos,

$$\begin{aligned} A\mathbf{x}_k &\approx \mathbf{b} \\ AK_k \tilde{\mathbf{c}}_k &\approx \mathbf{b}. \end{aligned}$$

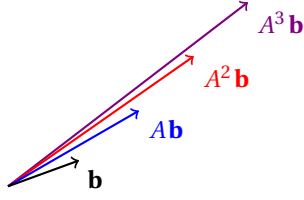
Del cual notamos un par de cosas,

- AK_k es una matriz de dimensiones $n \times k$.
- $\tilde{\mathbf{c}}_k \in \mathbb{R}^k$.

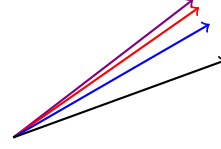
Es decir, tenemos un sistema de ecuaciones lineales sobre-determinado considerando que $n > k$. Más conocido como *un problema de mínimos cuadrados*! Esto significa que podemos obtener la solución que minimiza el error cuadrático de la siguiente forma,

$$\tilde{\mathbf{c}}_k = \underset{\tilde{\mathbf{c}}_k \in \mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{b} - AK_k \tilde{\mathbf{c}}_k\|_2^2. \quad (7.7)$$

El cual podemos resolver con las ecuaciones normales o con la descomposición QR de la matriz AK_k , donde luego se puede obtener \mathbf{x}_k como $\mathbf{x}_k = K_k \tilde{\mathbf{c}}_k$. La Figura 7.1 destaca el rol de las componentes involucradas en el problema de mínimos cuadrados definido en la ecuación (7.7). Esto, si bien responde la pregunta antes realizada sobre como utilizar el sub-espacio de Krylov \mathcal{K}_k , es solo un paso intermedio antes de construir GMRes. El desafío surge debido a que el problema de mínimos cuadrados definido en la ecuación (7.7) utiliza una base mal condicionada del sub-espacio de Krylov, ver Figura 7.2. Esto se debe a que los vectores resultantes generan una secuencia de vectores que



(a) Ejemplo de 4 vectores del sub-espacio de Krylov, es decir \mathbf{b} , $A\mathbf{b}$, $A^2\mathbf{b}$, y $A^3\mathbf{b}$.



(b) Normalización de los vectores presentados en la Figura izquierda, es decir, $\frac{\mathbf{b}}{\|\mathbf{b}\|}$, $\frac{A\mathbf{b}}{\|A\mathbf{b}\|}$, $\frac{A^2\mathbf{b}}{\|A^2\mathbf{b}\|}$, y $\frac{A^3\mathbf{b}}{\|A^3\mathbf{b}\|}$.

Figura 7.2: Esta Figura presenta a modo de ejemplo un bosquejo de los vectores \mathbf{b} , $A\mathbf{b}$, $A^2\mathbf{b}$, y $A^3\mathbf{b}$. En este caso se aprecia que a medida que se tiene un exponente mayor de j en $A^j\mathbf{b}$, el ángulo entre los vectores $A^j\mathbf{b}$ y $A^{j-1}\mathbf{b}$ es menor.

converge al vector propio asociado al valor propio dominante, lo que se traduce en el [método de la potencia](#)². Es decir, va generando vectores que cada vez se *parecen* más entre sí. En particular, este procedimiento genera vectores que podrían ser numéricamente linealmente dependientes, en resumen, esta forma de construir los vectores generadores del sub-espacio de Krylov produce una base mal condicionada.

Ahora, surge la pregunta ¿Cómo podemos construir una nueva base del sub-espacio de Krylov \mathcal{K}_k ? Es decir, como encontramos un conjunto de vectores basales *bien* condicionados y linealmente independientes³ que también generen el mismo sub-espacio, es decir \mathcal{K}_k . Esto significa lo siguiente,

$$\mathcal{K}_k = \text{span}(\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, A^3\mathbf{b}, \dots, A^{k-1}\mathbf{b}), \quad (7.8)$$

$$= \text{span}(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4, \dots, \mathbf{q}_k), \quad (7.9)$$

donde por conveniencia podemos definir que los \mathbf{q}_i sean ortonormales, es decir, tienen norma-2 igual a 1 y son ortogonales entre sí. Entonces, dada la relación anterior, y para poder responder la pregunta anterior sobre como construir una base *bien* condicionada, naturalmente uno puede reconocer que podemos aplicar la ortonormalización de Gram-Schmidt! En particular la variante modificada, ver sección [6.6.3](#).

²Es importante destacar que el método de la potencia está en el corazón del algoritmo básico de Google, es decir [PageRank](#).

³En la sección [7.3](#) se analizará con más detalle la independencia lineal de los vectores basales del sub-espacio de Krylov \mathcal{K}_k .

¡MUY IMPORTANTE!

Es importante destacar que la aplicación de la ortonormalización de Gram-Schmidt modificada en el contexto de GMRes se conoce como la iteración de Arnoldi. Sin embargo, requiere un poco de explicación esta observación.

La ortonormalización de Gram-Schmidt consiste en que a partir de una secuencia de vectores indexados, digamos \mathbf{a}_i , que tradicionalmente se interpreta como las columnas de una matriz A , pero podría ser simplemente un conjunto de vectores, se obtiene otro conjunto de vectores \mathbf{q}_i tal que sean ortogonales, es decir, $\mathbf{q}_i^* \mathbf{q}_j = 0$ para $i \neq j$, y unitarios, es decir, $\mathbf{q}_i^* \mathbf{q}_i = \|\mathbf{q}_i\| = 1$, en resumen, que sean ortonormales. Se puede considerar que se tiene a disposición la secuencia de vectores *a priori* sin embargo esto no es necesario, recuerde que el proceso de ortonormalización de Gram-Schmidt procesa de a un vector a la vez.

Por otro lado, la iteración de Arnoldi, se puede interpretar como la ortonormalización de Gram-Schmidt aplicada a una secuencia de vectores, que no se conocen *a priori*, pero se van obteniendo durante la ejecución de la ortonormalización. La base del algoritmo consiste en ortonormalizar el sub-espacio de Krylov \mathcal{K}_k ,

$$\mathcal{K}_k = \text{span}(\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, A^3\mathbf{b}, \dots, A^{k-1}\mathbf{b}),$$

es decir, se quiere construir la siguiente base ortonormal,

$$\mathcal{K}_k = \text{span}(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4, \dots, \mathbf{q}_k).$$

La conexión se logra por medio de considerar la siguiente secuencia de vectores que también generan el sub-espacio de Krylov \mathcal{K}_k ,

$$\mathcal{K}_k = \text{span}(\mathbf{q}_1, A\mathbf{q}_1, A\mathbf{q}_2, \dots, A\mathbf{q}_{k-1}). \quad (7.10)$$

Entonces, si conectamos la generación dinámica de los vectores antes presentados con la ortonormalización de Gram-Schmidt *modificada*, obtenemos la iteración de Arnoldi! Ver adicionalmente el apartado 7.4 para profundizar la razón de porque se puede reemplazar $A^k \mathbf{b}$ por $A\mathbf{q}_k$.

Por lo tanto, de las ecuaciones (7.8) y (7.9) podemos concluir que,

$$\text{span}(\mathbf{b}) = \text{span}(\mathbf{q}_1), \quad (7.11)$$

$$\text{span}(\mathbf{b}, A\mathbf{b}) = \text{span}(\mathbf{q}_1, \mathbf{q}_2), \quad (7.12)$$

$$\vdots$$

$$\text{span}(\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, A^3\mathbf{b}, \dots, A^{k-1}\mathbf{b}) = \text{span}(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4, \dots, \mathbf{q}_k). \quad (7.13)$$

Lo cual implica que $\mathbf{q}_1 = \frac{\mathbf{b}}{\|\mathbf{b}\|}$, y

$$A\mathbf{b} = \hat{h}_{11}\mathbf{q}_1 + \hat{h}_{21}\mathbf{q}_2, \quad (7.14)$$

$$A^2\mathbf{b} = \hat{h}_{12}\mathbf{q}_1 + \hat{h}_{22}\mathbf{q}_2 + \hat{h}_{32}\mathbf{q}_3, \quad (7.15)$$

$$\vdots$$

$$A^{k-1}\mathbf{b} = \sum_{j=1}^k \hat{h}_{j,k-1} \mathbf{q}_j.$$

Por otro lado, si consideramos la siguiente relación, podemos obtener una versión aún más conveniente,

$$\begin{aligned}\mathcal{K}_k &= \text{span}(\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, A^3\mathbf{b}, \dots, A^{k-1}\mathbf{b}), \\ &= \text{span}(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4, \dots, \mathbf{q}_k), \\ &= \text{span}(\mathbf{q}_1, A\mathbf{q}_1, A\mathbf{q}_2, A\mathbf{q}_3, \dots, A\mathbf{q}_{k-1}),\end{aligned}\tag{7.16}$$

donde básicamente se agrego la ecuación (7.16) a las ecuaciones (7.8) y (7.9). En el apartado 7.4 se explica la razón para utilizar la ecuación (7.16). La ventaja de esta relación es que produce las siguientes ecuaciones,

$$A\mathbf{q}_1 = h_{11}\mathbf{q}_1 + h_{21}\mathbf{q}_2, \tag{7.17}$$

$$A\mathbf{q}_2 = h_{12}\mathbf{q}_1 + h_{22}\mathbf{q}_2 + h_{32}\mathbf{q}_3, \tag{7.18}$$

$$\vdots$$

$$A\mathbf{q}_k = \sum_{j=1}^{k+1} h_{j,k}\mathbf{q}_j. \tag{7.19}$$

Notar que en este caso se incluyó una ecuación extra, es decir la ecuación asociada a $A\mathbf{q}_k$ que implica la inclusión del vector \mathbf{q}_{k+1} al lado derecho. Lo interesante de las ecuaciones (7.17)-(7.19) es que podemos re-escribirlas de la siguiente manera,

$$[A\mathbf{q}_1, A\mathbf{q}_2, \dots, A\mathbf{q}_k] = \begin{bmatrix} h_{11}\mathbf{q}_1 + h_{21}\mathbf{q}_2, & h_{12}\mathbf{q}_1 + h_{22}\mathbf{q}_2 + h_{32}\mathbf{q}_3, & \dots, & \sum_{j=1}^{k+1} h_{j,k}\mathbf{q}_j \end{bmatrix}.$$

Notando que en realidad tenemos el producto de matrices en ambos costados de la ecuación, obtenemos la siguiente identidad,

$$A \underbrace{[\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k]}_{Q_k} = \underbrace{[\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k, \mathbf{q}_{k+1}]}_{Q_{k+1}} \underbrace{\begin{bmatrix} h_{11} & h_{12} & \dots & h_{1,k-1} & h_{1,k} \\ h_{21} & h_{22} & \dots & h_{2,k-1} & h_{2,k} \\ 0 & h_{32} & \ddots & h_{3,k-1} & h_{3,k} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & h_{k,k-1} & h_{k,k} \\ 0 & \dots & \dots & 0 & h_{k+1,k} \end{bmatrix}}_{\tilde{H}_k}.$$

La cual corresponde a la reducción parcial de A a una forma de Hessenberg, es decir,

$$AQ_k = Q_{k+1}\tilde{H}_k, \tag{7.20}$$

donde Q_k es la matriz con k columnas ortonormales, Q_{k+1} es la matriz con $k+1$ columnas ortonormales, y $\tilde{H}_k \in \mathbb{R}^{(k+1) \times k}$ es una matriz upper Hessenberg. Las matrices upper Hessenberg son matrices parecidas a las matrices triangulares superiores pero también tienen coeficientes en la primera sub-diagonal. En resumen, fuimos capaces de construir una nueva base del sub-espacio de Krylov \mathcal{K}_k !

Nota

Antes de continuar con la construcción de GMRes, es importante destacar como la iteración de Arnoldi construye los $h_{j,k}$ y los vectores \mathbf{q}_j . En particular nos referimos a las ecuaciones (7.17)-(7.19), que repetimos acá por completitud,

$$\begin{aligned} A\mathbf{q}_1 &= h_{11}\mathbf{q}_1 + h_{21}\mathbf{q}_2, \\ A\mathbf{q}_2 &= h_{12}\mathbf{q}_1 + h_{22}\mathbf{q}_2 + h_{32}\mathbf{q}_3, \\ &\vdots \\ A\mathbf{q}_k &= \sum_{j=1}^{k+1} h_{j,k}\mathbf{q}_j. \end{aligned}$$

Por ejemplo, recordando que $\mathbf{q}_1 = \frac{\mathbf{b}}{\|\mathbf{b}\|}$, podemos obtener h_{11} , h_{21} , \mathbf{q}_2 , h_{12} , h_{22} , h_{32} , y \mathbf{q}_3 :

$$\begin{aligned} h_{11} &= \mathbf{q}_1^T A\mathbf{q}_1, \\ h_{21} &= \|A\mathbf{q}_1 - h_{11}\mathbf{q}_1\|, \\ \mathbf{q}_2 &= \frac{A\mathbf{q}_1 - h_{11}\mathbf{q}_1}{h_{21}}, \\ h_{12} &= \mathbf{q}_1^T A\mathbf{q}_2, \\ h_{22} &= \mathbf{q}_2^T (A\mathbf{q}_2 - h_{12}\mathbf{q}_1), \\ h_{32} &= \|A\mathbf{q}_2 - h_{12}\mathbf{q}_1 - h_{22}\mathbf{q}_2\|, \\ \mathbf{q}_3 &= \frac{A\mathbf{q}_2 - h_{12}\mathbf{q}_1 - h_{22}\mathbf{q}_2}{h_{32}}. \end{aligned}$$

Siguiendo el mismo procedimiento, podemos obtener los siguientes coeficientes y vectores. Recuerde que la iteración de Arnoldi no es más que la ortonormalización de Gram-Schmidt modificada!

Finalmente, podemos decir que ya tenemos todas las componentes necesarias para construir GMRes. Entonces, retomando la idea inicial de resolver el sistema de ecuaciones lineales $A\mathbf{x} = \mathbf{b}$ restringiendo \mathbf{x} al sub-espacio de Krylov \mathcal{K}_k y minimizando el error cuadrático, podemos construir nuevamente el problema de minimización cuadrática asociado presentado en la ecuación (7.7). Por completitud se repite acá,

$$\bar{\mathbf{c}}_k = \underset{\mathbf{c}_k \in \mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{b} - AK_k \bar{\mathbf{c}}_k\|_2^2.$$

Ahora, ya que tenemos otra y *mejor* base para \mathcal{K}_k tenemos la siguiente identidad,

$$K_k \bar{\mathbf{c}}_k = Q_k \mathbf{c}_k.$$

Esto quiere decir que tanto los vectores columnas de K_k como Q_k generan \mathcal{K}_k . Notar que para cualquier vector en \mathcal{K}_k es posible encontrar las coordenadas $\tilde{\mathbf{c}}_k$ si utilizamos K_k , y lo mismo es válido si utilizamos Q_k con \mathbf{c}_k ⁴, sin embargo las coordenadas no son necesariamente las mismas. Entonces, el problema cuadrático puede transformarse a la siguiente forma,

$$\bar{\mathbf{c}}_k = \underset{\mathbf{c}_k \in \mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{b} - AQ_k \mathbf{c}_k\|_2^2.$$

⁴Notar que este vector no tiene tilde!

Lo cual es claramente una mejor! Sin embargo, podemos mejorarlo aún más. En este caso note que es un problema de mínimos cuadrados con n ecuaciones y k incógnitas. Ahora, recordando la ecuación (7.20), es decir, la reducción parcial a la forma de Hessenberg $AQ_k = Q_{k+1}\tilde{H}_k$, obtenemos,

$$\begin{aligned}\bar{\mathbf{c}}_k &= \operatorname{argmin}_{\mathbf{c}_k \in \mathbb{R}^k} \|\mathbf{b} - AQ_k \mathbf{c}_k\|_2^2 \\ &= \operatorname{argmin}_{\mathbf{c}_k \in \mathbb{R}^k} \|\mathbf{b} - Q_{k+1} \tilde{H}_k \mathbf{c}_k\|_2^2.\end{aligned}$$

El cual aún sigue siendo un problema de mínimos cuadrados de dimensión $n \times k$. Aún nos faltan un par de pasos claves, el primero es re-escribir el vector \mathbf{b} convenientemente, recuerde que $\mathbf{q}_1 = \frac{\mathbf{b}}{\|\mathbf{b}\|}$, por lo que podemos escribirlo de la siguiente forma,

$$\begin{aligned}\mathbf{b} &= \|\mathbf{b}\| \mathbf{q}_1 \\ &= \|\mathbf{b}\| Q_{k+1} \mathbf{e}_1,\end{aligned}$$

donde \mathbf{e}_1 es el primer vector canónico de dimensión $k+1$ que contiene un 1 en su primera componente y 0 en todas las otras componentes. Esto nos asegura que al multiplicar \mathbf{e}_1 por Q_{k+1} obtengamos efectivamente \mathbf{q}_1 ! Reemplazándolo en la definición anterior obtenemos,

$$\begin{aligned}\bar{\mathbf{c}}_k &= \operatorname{argmin}_{\mathbf{c}_k \in \mathbb{R}^k} \|\mathbf{b} - Q_{k+1} \tilde{H}_k \mathbf{c}_k\|_2^2 \\ &= \operatorname{argmin}_{\mathbf{c}_k \in \mathbb{R}^k} \|\|\mathbf{b}\| Q_{k+1} \mathbf{e}_1 - Q_{k+1} \tilde{H}_k \mathbf{c}_k\|_2^2 \\ &= \operatorname{argmin}_{\mathbf{c}_k \in \mathbb{R}^k} \|Q_{k+1} (\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k)\|_2^2,\end{aligned}$$

donde tenemos la siguiente identidad,

¡MUY IMPORTANTE!

$$\begin{aligned}\|Q_{k+1} (\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k)\|_2^2 &= (Q_{k+1} (\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k))^T (Q_{k+1} (\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k)) \\ &= (\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k)^T \underbrace{Q_{k+1}^T Q_{k+1}}_{I_{k+1}} (\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k) \\ &= (\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k)^T (\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k) \\ &= \|\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k\|_2^2.\end{aligned}$$

El desarrollo anterior es en realidad el corazón de GMRes y lo que justifica todo el desarrollo teórico anterior! Entonces, tenemos la grandiosa identidad,

$$\begin{aligned}\bar{\mathbf{c}}_k &= \operatorname{argmin}_{\mathbf{c}_k \in \mathbb{R}^k} \|\mathbf{b} - AQ_k \mathbf{c}_k\|_2^2 \\ &= \operatorname{argmin}_{\mathbf{c}_k \in \mathbb{R}^k} \|\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k\|_2^2.\end{aligned}\tag{7.21}$$

La cual nos dice que para obtener $\bar{\mathbf{c}}_k$ podemos resolver el problema de mínimos cuadrados $\|\mathbf{b} - AQ_k \mathbf{c}_k\|_2^2$ de dimensión $n \times k$, o el problema de mínimos cuadrados equivalente $\|\|\mathbf{b}\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k\|_2^2$ de dimensión $(k+1) \times k$. Sin

duda alguna, la conclusión es que es mucho más conveniente resolver el sistema de ecuaciones lineales sobre-determinado de dimensión $(k+1) \times k$. Por completitud, presentamos la estructura del sistema de ecuaciones lineales de dimensión $(k+1) \times k$,

$$\bar{\mathbf{c}}_k = \underset{\mathbf{c}_k \in \mathbb{R}^k}{\operatorname{argmin}} \left\| \begin{bmatrix} \|\mathbf{b}\| \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1,k-1} & h_{1,k} \\ h_{21} & h_{22} & \dots & h_{2,k-1} & h_{2,k} \\ 0 & h_{32} & \ddots & h_{3,k-1} & h_{3,k} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & h_{k,k-1} & h_{k,k} \\ 0 & \dots & \dots & 0 & h_{k+1,k} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{k-1} \\ c_k \end{bmatrix} \right\|_2. \quad (7.22)$$

El desarrollo anterior se reduce al siguiente algoritmo,

```

1  $\mathbf{x}_0 = \text{"initial guess"}$ 
2  $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ 
3  $\mathbf{q}_1 = \frac{\mathbf{r}_0}{\|\mathbf{r}_0\|_2}$ 
4 for  $k$  in  $\text{range}(1, m+1)$  :
5    $\mathbf{y} = A\mathbf{q}_k$ 
6   for  $j$  in  $\text{range}(1, k+1)$  :
7      $h_{jk} = \mathbf{q}_j^T \cdot \mathbf{y}$ 
8      $\mathbf{y} = \mathbf{y} - h_{jk}\mathbf{q}_j$ 
9    $h_{k+1,k} = \|\mathbf{y}\|_2$ 
10  if  $h_{k+1,k} > 0$  :
11     $\mathbf{q}_{k+1} = \frac{\mathbf{y}}{h_{k+1,k}}$ 
12   $\bar{\mathbf{c}}_k = \underset{\mathbf{c}_k \in \mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{r}_0\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k\|_2$ 
13   $\mathbf{x}_k = Q_k \bar{\mathbf{c}}_k + \mathbf{x}_0$ 

```

Algoritmo 7: GMRes

En el Algoritmo 7 se incluyen algunos cambios menores respecto a la formulación original y también hay algunos puntos que destacar:

- En la línea 1 del algoritmo se incluyó un initial guess, el cual no fue incluido en la formulación original, sin embargo el cambio es simple, es decir, si uno quiere resolver el sistema de ecuaciones lineales $A\mathbf{x} = \mathbf{b}$, donde ya conoce algo de la solución, es decir un initial guess \mathbf{x}_0 , entonces puede re-escribir el sistema de ecuaciones lineales de la siguiente forma:

$$\begin{aligned}
 A\mathbf{x} &= \mathbf{b}, \\
 A(\mathbf{y} + \mathbf{x}_0) &= \mathbf{b}, \\
 A\mathbf{y} &= \mathbf{b} - A\mathbf{x}_0, \\
 A\mathbf{y} &= \tilde{\mathbf{b}}.
 \end{aligned}$$

Es decir, solo se modifica el lado derecho de forma conveniente. Esto es bien útil en GMRes por que permite reiniciarlo (restart), es decir, luego de ejecutarlo, por ejemplo por 100 iteraciones, uno puede utilizar

la aproximación obtenida como `initial guess` y empezar nuevamente GMRes. Hay ventajas y desventajas de esta alternativa que son importante conocer para aplicaciones avanzadas.

- GMRes necesita almacenar las matrices Q_{k+1} y \tilde{H}_k , las cuales aumentan de tamaño en función del número de iteraciones k , por lo que es importante saber cuantificar el número de iteraciones a utilizar en GMRes para no quedar sin memoria. Aquí, una alternativa es reiniciar GMRes por ejemplo, como se explicó en el primer punto.
- En las únicas líneas del algoritmo en que aparece la matriz A son en las líneas 2 y 5. En particular aparecen multiplicando un vector, es decir $A\mathbf{x}_0$ y $A\mathbf{q}_k$, respectivamente. Esto significa que no se requiere acceder a los coeficientes de la matriz directamente, solo se necesita la computación del producto de la matriz A por un vector. Por ejemplo, si uno definiera la función $\text{afun}(\mathbf{v}) = A\mathbf{v}$, es decir una función a la que se le entregue un vector \mathbf{v} y luego retorne el producto de ese vector \mathbf{v} por la matriz A , es decir $A\mathbf{v}$, entonces GMRes podría ejecutarse sin ningún problema! La importancia de esto recae en que se puede tener un hardware especializado que implemente $\text{afun}(\mathbf{v})$, tomando ventaja de algún almacenamiento conveniente por ejemplo, y aún en esas condiciones podría resolver el sistema de ecuaciones lineales asociado. Esta ventaja no la entrega PALU ni los métodos iterativos tradicionales como Jacobi o Gauss-Seidel. Ver ejemplo de como resolver la [ecuación de Sylvester](#) $AX + XB = C$ para A, B , y C matrices conocidas en $\mathbb{R}^{n \times n}$, la matriz incógnita $X \in \mathbb{R}^{n \times n}$, en el Jupyter Notebook adicional respectivo. Ver también apartado 7.5.
- En la línea 12 del algoritmo uno tiene que resolver un problema de mínimos cuadrados una y otra vez, lo cual implica obtener la descomposición QR reducida de \tilde{H}_k en cada iteración. Lo interesante es que la \tilde{H}_k solo aumenta en una fila y una columna por iteración, por lo que uno puede tomar ventaja de la factorización QR reducida de \tilde{H}_{k-1} para obtener la factorización QR reducida de \tilde{H}_k , y así sucesivamente! **Warning: No confundir los vectores ortonormales de la factorización QR reducida, con los vectores ortonormales de la iteración de Arnoldi que usa GMRes!**
- Por último, en la línea 9 del algoritmo se obtiene el coeficiente $h_{k+1,k}$ y luego en la línea 10 se verifica que sea mayor a 0. En realidad lo que se está verificando es que sea distinto a 0 ya que como el coeficiente viene de una norma vectorial sabemos que va a ser mayor o igual a 0. En principio se podría pensar que en el caso de que $h_{k+1,k} = 0$ tengamos un problema, pero en realidad es una muy buena noticia y se conoce como GMRes breakdown⁵. Esto se debe a que al tener ese coeficiente igual 0 implica que el problema de minimización en la línea 12 se puede resolver exactamente, ya que se convierte de un problema de mínimos cuadrados de dimensión $(k+1) \times k$ a un sistema de ecuaciones lineales cuadrado de dimensión $k \times k$, es decir, se podrá resolver exactamente! Esto significa que efectivamente se podrá encontrar la solución exacta del sistema de ecuaciones lineales cuadrado original definido en la ecuación (7.3). Más aún, la ecuación (7.21) indica que el poder hacer 0 el error cuadrático del problema de mínimos cuadrados reducido, también hacemos 0 el error cuadrático del problema original.

En resumen, hemos sido capaces de construir GMRes y ahora tenemos a nuestra disposición una nueva familia de algoritmos para resolver sistemas de ecuaciones lineales cuadrados! 😊

7.3. ¿Son linealmente independientes $\{\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, A^3\mathbf{b}, \dots, A^{k-1}\mathbf{b}\}$?

Un punto crucial en GMRes es poder construir una base ortonormal del sub-espacio de Krylov \mathcal{K}_k . La cual se logra, según lo explicado en la sección anterior, con la iteración de Arnoldi. Otro punto crítico es que efectivamente el sub-espacio de Krylov genere vectores linealmente independientes a medida que aumenta k , sin embargo, hay que hacer una observación importante. En realidad es solo necesario generar vectores linealmente independientes

⁵Recuerde que estamos considerando que la matriz A es no singular, si A fuera **singular**, podrían haber breakdown anticipados que no implican que se obtuvo una solución.

para poder representar la solución $\mathbf{x} = A^{-1}\mathbf{b}$ pero no \mathbb{R}^n . Esta observación no es contradictoria con la ecuación (7.5), que repetimos acá por completitud,

$$\mathbf{x} = A^{-1}\mathbf{b} = \frac{(-1)^{n-1}}{\det(A)} (A^{n-1}\mathbf{b} + \check{c}_{n-1}A^{n-2}\mathbf{b} + \dots + \check{c}_2A\mathbf{b} + \check{c}_1\mathbf{b}).$$

Esta ecuación no se exige que los vectores $A^j\mathbf{b}$ sean linealmente independientes, solo se requiere que representen la solución.

Ahora, para entender un poco más lo que está ocurriendo debemos recordar la ecuación (7.13), es decir,

$$\text{span}(\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, A^3\mathbf{b}, \dots, A^{k-1}\mathbf{b}) = \text{span}(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4, \dots, \mathbf{q}_k).$$

Esta relación nos permite re-escribir nuestra representación de la solución,

$$\mathbf{x}_k = \sum_{i=1}^k c_i \mathbf{q}_i,$$

de la siguiente forma,

$$\begin{aligned} \mathbf{x}_k &= \sum_{i=1}^k \hat{c}_i A^{i-1} \mathbf{b}, \\ &= \left(\sum_{i=1}^k \hat{c}_i A^{i-1} \right) \mathbf{b}, \\ &= q_k(A) \mathbf{b}, \end{aligned}$$

donde $q_k(z)$ es un polinomio de grado $k-1$. Ahora, podemos re-escribir el vector residual original de la siguiente forma,

$$\begin{aligned} \|\mathbf{b} - A\mathbf{x}_k\|_2^2 &= \|\mathbf{b} - Aq_k(A)\mathbf{b}\|_2^2, \\ &= \|(I - Aq_k(A))\mathbf{b}\|_2^2, \\ &= \|p_k(A)\mathbf{b}\|_2^2. \end{aligned}$$

Este resultado nos indica que si se logra hacer 0 el vector residual en la k -ésima iteración, efectivamente encontramos la solución, y la podemos representar como $\mathbf{x} = q_k(A)\mathbf{b}$, donde solo utilizamos k vectores y no los n vectores como se obtuvo en la ecuación (7.5). Además se presenta el polinomio $p_k(z)$ de grado k y que $p_k(0) = 1$. Esta representación nos demuestra que GMRes se puede ver como una minimización polinomial del residuo, es decir, $\|\mathbf{b} - A\mathbf{x}_k\|_2^2 = \|p_k(A)\mathbf{b}\|_2^2$.

Por otro lado, aún no hemos explicado si efectivamente la secuencia $A^j\mathbf{b}$ puede asegurar la generación de vectores linealmente independientes. Para entender esto, debemos analizar la ecuación (7.19), es decir,

$$A\mathbf{q}_k = h_{1,k}\mathbf{q}_1 + h_{2,k}\mathbf{q}_2 + \dots + h_{k,k}\mathbf{q}_k + h_{k+1,k}\mathbf{q}_{k+1}.$$

En particular hay que poner atención a la construcción del coeficiente $h_{k+1,k}$, que corresponde a,

$$h_{k+1,k} = \|A\mathbf{q}_k - h_{1,k}\mathbf{q}_1 - h_{2,k}\mathbf{q}_2 - \dots - h_{k,k}\mathbf{q}_k\|.$$

Como se indicó al final de la sección anterior, este coeficiente puede ser 0, lo que significa que se obtiene un breakdown positivo, porque el problema de mínimos cuadrados pequeño resultante, ver ecuaciones (7.21) y (7.22), en realidad se convierte en un sistema de ecuaciones lineales cuadrado y se puede resolver exactamente. Sin embargo también significa que el vector \mathbf{q}_{k+1} no puede ser generado por $A\mathbf{q}_k$ ya que $A\mathbf{q}_k$ no genera un vector **linealmente independiente** al obtener $h_{k+1,k} = 0$! Lo cual responde nuestra pregunta inicial⁶. Recuerde que de todos modos este breakdown es beneficioso ya que se puede concluir que se obtuvo la solución exacta al sistema de ecuaciones lineales cuadrado original en la ecuación (7.3). Entonces, como se indicó al principio, la secuencia $A^j\mathbf{b}$ solo nos entrega los vectores linealmente independiente que necesitamos para encontrar \mathbf{x} y nada más.

⁶Es decir, los vectores generados son linealmente independientes hasta que se genera uno linealmente dependiente, pero en ese caso se obtiene la solución!

7.4. ¿Realmente se puede solo usar $A\mathbf{q}_k$ y no $A^k\mathbf{b}$?

En la identidad utilizada para relacionar los distintos sub-espacios de Krylov, en particular en la ecuación (7.16), aparece la relación entre $A^k\mathbf{b}$ y $A\mathbf{q}_k$. Para explicitar esta relación considere la siguiente identidad,

$$\|\mathbf{b}\| \mathbf{q}_1 = \mathbf{b}.$$

Lo cual asegura que $\text{span}(\mathbf{b}) = \text{span}(\mathbf{q}_1)$. Ahora, si multiplicamos por la izquierda por A , obtenemos,

$$\begin{aligned} \|\mathbf{b}\| A\mathbf{q}_1 &= A\mathbf{b} \\ \hat{h}_{1,1}\mathbf{q}_1 + \hat{h}_{2,1}\mathbf{q}_2 &= A\mathbf{b}. \end{aligned} \quad (7.23)$$

En la lado izquierdo de la ecuación se está mencionando que el vector $\|\mathbf{b}\| A\mathbf{q}_1$ se puede re-escribir como la combinación lineal de \mathbf{q}_1 y \mathbf{q}_2 , este es un punto importante. Note que los coeficientes $\hat{h}_{1,1}$ y $\hat{h}_{2,1}$ se obtienen utilizando el algoritmo de Gram-Schmidt modificado. En particular, si el coeficiente $\hat{h}_{2,1}$ es 0, entonces la conclusión es que $\text{span}(A\mathbf{q}_1) = \text{span}(\mathbf{q}_1)$ y \mathbf{q}_2 no puede ser determinado siguiendo este procedimiento. Por otro lado, si el coeficiente $\hat{h}_{2,1}$ es distinto de 0 entonces se puede efectivamente encontrar el vector \mathbf{q}_2 . Por lo que en este caso, se logra efectivamente que $\text{span}(\mathbf{b}, A\mathbf{b}) = \text{span}(\mathbf{q}_1, \mathbf{q}_2)$. Ahora, para continuar la explicación, considere que multiplicamos la ecuación (7.23) por A por la izquierda, entonces,

$$\hat{h}_{1,1} A\mathbf{q}_1 + \hat{h}_{2,1} A\mathbf{q}_2 = A^2\mathbf{b}. \quad (7.24)$$

Ya sabemos que el primer término, es decir $A\mathbf{q}_1$ puede escribirse como una combinación lineal de \mathbf{q}_1 y \mathbf{q}_2 , además naturalmente podemos proponer que $A\mathbf{q}_2$ se pueda re-escribir como la combinación lineal \mathbf{q}_1 , \mathbf{q}_2 , y \mathbf{q}_3 , lo que nos da como resultado lo siguiente,

$$\hat{h}_{1,2}\mathbf{q}_1 + \hat{h}_{2,2}\mathbf{q}_2 + \hat{h}_{3,2}\mathbf{q}_3 = A^2\mathbf{b}, \quad (7.25)$$

donde nuevamente podemos determinar los coeficientes $\hat{h}_{i,2}$ para $i \in \{1, 2, 3\}$ con el algoritmo de Gram-Schmidt modificado. Además, se puede llegar a la misma conclusión si es que, en este caso, el coeficiente $\hat{h}_{3,2} = 0$. Pero si consideramos que $\hat{h}_{3,2} \neq 0$, entonces hemos exitosamente encontrado \mathbf{q}_3 . Este mismo procedimiento se puede generalizar para $A^k\mathbf{b}$. Sin embargo, aún no hemos respondido la duda inicial de forma explícita, pero si se enunció en la ecuación (7.24). Justo en el paso donde se indica que $A\mathbf{q}_2$ puede escribirse como una combinación lineal de \mathbf{q}_1 , \mathbf{q}_2 , y \mathbf{q}_3 , es decir, podemos escribir la siguiente ecuación,

$$A\mathbf{q}_2 = h_{1,2}\mathbf{q}_1 + h_{2,2}\mathbf{q}_2 + h_{3,2}\mathbf{q}_3. \quad (7.26)$$

Que es justamente la relación que estábamos buscando! Notar que ahora no utilizamos los coeficientes \hat{h} 's porque la identidad es distinta, en la ecuación (7.25) se igualaba a $A^2\mathbf{b}$ y en la (7.26), sin embargo esto no cambia el resultado. El desarrollo anterior demuestra que es equivalente utilizar $A^2\mathbf{b}$ o $A\mathbf{q}_2$ para obtener \mathbf{q}_3 . Se deja como ejercicio extender el resultado para $A\mathbf{q}_k$!

7.5. GMRes visita a Sylvester

Al finalizar la derivación de GMRes en apartado 7.2 se mencionó la [ecuación de Sylvester](#), la cual consiste en la siguiente ecuación matricial:

$$\hat{A}X + X\hat{B} = \hat{C}, \quad (7.27)$$

donde \hat{A} , \hat{B} , y \hat{C} pertenecen a $\mathbb{R}^{n \times n}$ y son conocidas⁷. La tarea es determinar la matriz incógnita $X \in \mathbb{R}^{n \times n}$. La posible primera idea que surge es tratar de despejar la matriz X , por ejemplo de la siguiente forma,

$$X + \hat{A}^{-1}X\hat{B} = \hat{A}^{-1}\hat{C}, \quad (7.28)$$

⁷La inclusión de un techo en \hat{A} es para destacar, en particular, que la matriz \hat{A} es distinta a la matriz A que se presentará luego.

considerando que \hat{A} es no-singular. Rápidamente nos damos cuenta que no es tan simple despejar la matriz X ya que no podemos factorizarla en la manipulación algebraica anterior⁸. Viendo el vaso medio lleno, se podría construir una iteración de punto fijo **matricial** con ecuación (7.28) de la siguiente forma,

$$X^{(0)} = \text{"initial guess matricial"} \\ X^{(n+1)} = \hat{A}^{-1} (\hat{C} - X^{(n)} \hat{B}), \text{ para } n \in \{1, 2, 3, \dots\}.$$

Es importante destacar que no es necesario obtener la inversa de la matriz \hat{A} , se puede resolver el sistema de ecuaciones lineales correspondiente! Sí sería una buena idea obtener la factorización PALU de A una sola vez y luego utilizarlas en las siguientes iteraciones. Las preguntas que surgen de inmediato son las siguientes:

- ¿Cómo podemos asegurar la convergencia de la iteración de punto fijo?
- ¿Podríamos construir otro tipo de iteración de punto fijo?
- Si convergiera muy lento, ¿Cómo podríamos acelerar la convergencia?

Las cuales son muy válidas y pertinentes, y pueden producir algoritmos muy efectivos para casos particulares de matrices \hat{A} , \hat{B} , y \hat{C} , por lo cual hay que siempre mantener presente este tipo de alternativas. Por otro lado, el objetivo de esta sección es estudiar la utilización de GMRes para resolver la ecuación de Sylvester. A primera vista, parece ser que GMRes no se puede utilizar, por lo menos en su forma tradicional, porque no estamos resolviendo un sistema de ecuaciones lineales escrito en su forma tradicional, es decir,

$$Ax = b.$$

Esta forma tradicional es la forma requerida también para utilizar todos los algoritmos presentados anteriormente, es decir: LU, PALU, Jacobi, Gauss-Seidel, etc. Entonces, ¿es realmente un sistema de ecuaciones lineales la ecuación (7.27)? Si es así, ¿cómo podemos obtener A , x , y b ? Para responder estas preguntas, consideremos el siguiente ejemplo,

$$\underbrace{\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}}_{\hat{A}} \underbrace{\begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{bmatrix}}_X + \underbrace{\begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{bmatrix}}_X \underbrace{\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix}}_{\hat{B}} = \underbrace{\begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}}_{\hat{C}}.$$

Multiplicando,

$$\begin{bmatrix} a_{1,1} x_{1,1} + a_{1,2} x_{2,1} & a_{1,1} x_{1,2} + a_{1,2} x_{2,2} \\ a_{2,1} x_{1,1} + a_{2,2} x_{2,1} & a_{2,1} x_{1,2} + a_{2,2} x_{2,2} \end{bmatrix} + \begin{bmatrix} b_{1,1} x_{1,1} + b_{2,1} x_{1,2} & b_{1,2} x_{1,1} + b_{2,2} x_{1,2} \\ b_{1,1} x_{2,1} + b_{2,1} x_{2,2} & b_{1,2} x_{2,1} + b_{2,2} x_{2,2} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}.$$

Sumando términos al lado izquierdo,

$$\begin{bmatrix} (a_{1,1} + b_{1,1}) x_{1,1} + a_{1,2} x_{2,1} + b_{2,1} x_{1,2} & b_{1,2} x_{1,1} + (a_{1,1} + b_{2,2}) x_{1,2} + a_{1,2} x_{2,2} \\ a_{2,1} x_{1,1} + (a_{2,2} + b_{1,1}) x_{2,1} + b_{2,1} x_{2,2} & a_{2,1} x_{1,2} + b_{1,2} x_{2,1} + (a_{2,2} + b_{2,2}) x_{2,2} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}. \quad (7.29)$$

Igualando término a término tenemos las siguientes 4 ecuaciones,

$$(a_{1,1} + b_{1,1}) x_{1,1} + a_{1,2} x_{2,1} + b_{2,1} x_{1,2} = c_{1,1} \quad (7.30a)$$

$$a_{2,1} x_{1,1} + (a_{2,2} + b_{1,1}) x_{2,1} + b_{2,1} x_{2,2} = c_{2,1} \quad (7.30b)$$

$$b_{1,2} x_{1,1} + (a_{1,1} + b_{2,2}) x_{1,2} + a_{1,2} x_{2,2} = c_{1,2} \quad (7.30c)$$

$$a_{2,1} x_{1,2} + b_{1,2} x_{2,1} + (a_{2,2} + b_{2,2}) x_{2,2} = c_{2,2}. \quad (7.30d)$$

⁸Es un buen ejercicio seguir tratando de despejar X !

Re-escribiendolo ahora como un sistema de ecuaciones lineales obtenemos lo que andábamos buscando!

$$\underbrace{\begin{bmatrix} (a_{1,1} + b_{1,1}) & a_{1,2} & b_{2,1} & 0 \\ a_{2,1} & (a_{2,2} + b_{1,1}) & 0 & b_{2,1} \\ b_{1,2} & 0 & (a_{1,1} + b_{2,2}) & a_{1,2} \\ 0 & b_{1,2} & a_{2,1} & (a_{2,2} + b_{2,2}) \end{bmatrix}}_A \underbrace{\begin{bmatrix} x_{1,1} \\ x_{2,1} \\ x_{1,2} \\ x_{2,2} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} c_{1,1} \\ c_{2,1} \\ c_{1,2} \\ c_{2,2} \end{bmatrix}}_{\mathbf{b}} \quad (7.31)$$

Nota

Una forma alternativa de obtener el sistema de ecuaciones lineales anterior es la siguientes,

$$A = (I \otimes \hat{A}) + (\hat{B}^T \otimes I),$$

$$\mathbf{x} = \text{vec}(X),$$

$$\mathbf{b} = \text{vec}(\hat{C}),$$

donde \otimes es el producto de Kronecker, I es la identidad y vec se define de la siguiente forma,

$$\text{vec} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a \\ c \\ b \\ d \end{bmatrix}.$$

Para más detalles de este desarrollo ver el Jupyter Notebook adicional en el repositorio en GitHub.

¡MUY IMPORTANTE!

Es necesario aclarar que si las matrices \hat{A} y \hat{B} son de dimensión $n \times n$, entonces la matriz A es de dimensión $n^2 \times n^2$. ¡Por lo que requiere mucha más memoria para ser almacenada! Es decir, la idea es evitar su construcción!

Ahora, dado que construimos el sistema de ecuaciones lineales en su forma tradicional en la ecuación (7.31), podemos utilizar sin problemas GMRes, es decir, podemos utilizar el algoritmo 7. Sin embargo el desafío es poder utilizar GMRes directamente con la ecuación (7.27), sin requerir la construcción de A en ecuación (7.31).

Antes de analizar el cómo utilizar solamente la ecuación (7.27) debemos hacer una mínima modificación al algoritmo 7, la cual se traduce en la siguiente variante, ver algoritmo 8. Específicamente se ha destacado en rojo los 2 cambios realizados, es decir, la utilización de la función **afun** en las líneas 2 y 5. La tarea de esta función es recibir un vector y retornar el productor matriz-vector asociado. La primera interpretación que tradicionalmente se hace de la función **afun** es la presentada en algoritmo 9. La primera desventaja que notamos en el algoritmo 9 es que requerimos tener a nuestra disposición la matriz A de la ecuación (7.31), lo que es justamente lo que queremos evitar. Entonces, para tomar ventaja del algoritmo 8 se propone la siguiente alternativa de implementación de **afun**, la cual se basa en la computación realizada al lado izquierdo de la ecuación (7.29), ver algoritmo 10. Notar que la computación del lado izquierdo de ecuación (7.29) da paso para construir explícitamente la matriz A en ecuación (7.31), sin embargo, no requiere la construcción explícita de A . Más aún, lo que se hace en la ecuación (7.29) es obtener el lado izquierdo de las ecuaciones en (7.30). Recuerde que la computación del lado izquierdo es simplemente un producto matriz-vector dado que se conocen tanto los coeficientes de las matrices A y B como el vector “incógnitas” X .

```

1  $\mathbf{x}_0 = \text{"initial guess"}$ 
2  $\mathbf{r}_0 = \mathbf{b} - \text{afun}(\mathbf{x}_0)$ 
3  $\mathbf{q}_1 = \frac{\mathbf{r}_0}{\|\mathbf{r}_0\|_2}$ 
4 for  $k$  in  $\text{range}(1, m+1)$  :
5    $\mathbf{y} = \text{afun}(\mathbf{q}_k)$ 
6   for  $j$  in  $\text{range}(1, k+1)$  :
7      $h_{jk} = \mathbf{q}_j^T \cdot \mathbf{y}$ 
8      $\mathbf{y} = \mathbf{y} - h_{jk} \mathbf{q}_j$ 
9    $h_{k+1,k} = \|\mathbf{y}\|_2$ 
10  if  $h_{k+1,k} > 0$  :
11     $\mathbf{q}_{k+1} = \frac{\mathbf{y}}{h_{k+1,k}}$ 
12   $\bar{\mathbf{c}}_k = \underset{\mathbf{c}_k \in \mathbb{R}^k}{\text{argmin}} \|\mathbf{r}_0\| \mathbf{e}_1 - \tilde{H}_k \mathbf{c}_k\|_2$ 
13   $\mathbf{x}_k = Q_k \bar{\mathbf{c}}_k + \mathbf{x}_0$ 

```

Algoritmo 8: GMRes - Matrix-free**Nota**

Es importante destacar que cuando uno ejecuta GMRes se van obteniendo varios productos matriz-vector, por ejemplo, el primer productor matriz vector es $A\mathbf{x}_0$ y luego para cada \mathbf{q}_k se obtiene $A\mathbf{q}_k$, sin embargo lo crucial no es construir la matriz A explícitamente para realizar el producto matriz-vector, sino la construcción de un procedimiento que haga la tarea, esa es la importancia de construir adecuadamente **afun**.

En resumen, se ha logrado mostrar el gran potencial que tiene GMRes para resolver sistemas de ecuaciones lineales sin la necesidad que estos estén escritos en su forma tradicional o estándar, lo que trae consigo el beneficio de reducir los requerimientos de memoria asociados de llevarlos a su forma estándar. ¡Lo cual puede ser un gran cuello de botella! Para ver GMRes en acción le sugerimos revisar el Jupyter Notebook asociado en [link](#).

7.6. GMRes visita a Newton

La gran flexibilidad de GMRes no termina en la sección anterior, otra aplicación es cuando uno quiere resolver un *sistema de ecuaciones no lineales cuadrado*, es decir, cuando tenemos la misma cantidad de ecuaciones que incógnitas. Esta variante se le denomina Newton-Krylov, ver más detalles [6] y [1].

En esta sección se realizará una descripción abstracta del método para luego aplicarlo en el apartado 9.2.2.

Considere que quiere aplicar el método de Newton, ver apartado 4.6, para encontrar una raíz de $\mathbf{F}(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^n$, es decir, queremos encontrar \mathbf{x} tal que,

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}.$$

Esto se logra con la siguiente iteración de punto fijo en alta dimensión,

$$\begin{aligned}
 \mathbf{x}_0 &= \text{"Initial Guess"}, \\
 J(\mathbf{x}_i) \mathbf{w} &= -\mathbf{F}(\mathbf{x}_i), \quad \text{resolver } \mathbf{w} \text{ para } i \in \{1, 2, 3, \dots\} \\
 \mathbf{x}_{i+1} &= \mathbf{x}_i + \mathbf{w}.
 \end{aligned} \tag{7.32}$$

```

1 def afun_0(v):
2     # "A" corresponde a la matriz asociada a la forma tradicional
      de escribir un sistema de ecuaciones lineales, es decir,
      Ax=b.
3     # "v" corresponde el vector de entrada, este ser'a
      inicialmente "x0" y luego "qk" en GMRes.
4     # Notar que la arroba es simplemente la notaci'on para el
      producto matriz vector.
5     out = A @ v
6     return out

```

Algoritmo 9: afun - versi3n preliminar

```

1 def afun_1(v):
2     # Primero se hace un "reshape" del vector "v" de dimensi'on n
      ^2 a una matriz X dimensi'on n x n
3     X = np.reshape(v, (n,n), order='F')
4     # Recordando la ecuaci'on de Sylvester: A_h*X+X*B_h=C_h, se
      obtiene el producto matriz-vector asociado.
5     out = A_h @ X + X @ B_h
6     # Se aplica el operador "vec" a la matriz "out" y se retorna
7     return out.flatten('F')

```

Algoritmo 10: afun - versi3n *matrix-free*

Aqu' queremos destacar que la ejecuci3n del M3todo de Newton en alta dimensi3n requiere resolver un sistema de ecuaciones lineales distinto, ver ecuaci3n (7.32), en cada iteraci3n. En particular notamos que $J(\mathbf{x}_i)$ corresponde a la matriz Jacobiana de $\mathbf{F}(\mathbf{x})$ evaluada en \mathbf{x}_i . Entonces, lo primero que uno podr'ia pensar que se requiere para aplicar el M3todo de Newton es obtener la matriz Jacobiana $J(\mathbf{x}_i)$, la cual, por completitud, corresponde a la siguiente matriz,

$$J(\mathbf{x}_i) = \left[\begin{array}{cccc} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{array} \right]_{\mathbf{x} = \mathbf{x}_i}.$$

Lo que claramente indica que uno tiene que obtener muchas derivadas! Sin embargo, antes de empezar a obtener derivadas, considere la siguiente expresi3n, la cual corresponde a la linealizaci3n de la funci3n $\mathbf{F}(\mathbf{x}_0 + \varepsilon \mathbf{v})$ entorno a $\varepsilon = 0$ para vectores \mathbf{x}_0 y \mathbf{v} dados,

$$\mathbf{F}(\mathbf{x}_0 + \varepsilon \mathbf{v}) = \mathbf{F}(\mathbf{x}_0) + \varepsilon J(\mathbf{x}_i) \mathbf{v} + \mathcal{O}(\varepsilon^2).$$

Despejando $J(\mathbf{x}_i)\mathbf{v}$ obtenemos,

$$J(\mathbf{x}_i)\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{x}_0 + \varepsilon \mathbf{v}) - \mathbf{F}(\mathbf{x}_0)}{\varepsilon}, \quad (7.33)$$

donde la aproximación “mejora” a medida que ε tiende a 0⁹. Entonces, ¿cómo podemos conectar la ecuación (7.32) y ecuación (7.33)?

La respuesta es muy simple a esta altura, ¡claramente con GMRes!

Entonces, si definimos $\text{afun}(\mathbf{v}) = (\mathbf{F}(\mathbf{x}_0 + \text{eps} * \mathbf{v}) - \mathbf{F}(\mathbf{x}_0)) / \text{eps}$, podemos simplemente llamar a GMRes para encontrar una aproximación a la solución numérica de la ecuación (7.32), es decir podemos re-escribir el Método de Newton de la siguiente forma,

$\mathbf{x}_0 = \text{“Initial Guess”}$,

$\mathbf{w} = \text{GMRes}(\text{afun}, \mathbf{b} = -\mathbf{F}(\mathbf{x}_i), \text{threshold} = 10^{-10})$, resolver \mathbf{w} para $i \in \{1, 2, 3, \dots\}$

$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{w}$.

En resumen, lo que hace el método denominado “Newton-Krylov” es aplicar el clásico método de Newton pero utilizando GMRes para resolver el sistema de ecuaciones asociado a cada iteración. En particular, se toma ventaja de que GMRes no requiere operar sobre la matriz $J(\mathbf{x}_i)$, sino lo que necesita es una función ($\text{afun}(\mathbf{v})$) que dado un vector \mathbf{v} retorne el producto de $J(\mathbf{x}_i)$ con \mathbf{v} , lo cual se logra con la ecuación (7.33). **¡La gran ventaja de esta variante es que no se necesita obtener ninguna derivada!** Solo se necesita tener acceso a $\mathbf{F}(\mathbf{x})$ para construir todo el algoritmo! Esto es muy interesante desde el punto de vista teórico y computacional, porque nos permite utilizar GMRes para resolver problemas desafiantes, como lo son los sistemas de ecuaciones no lineales, solo definiendo adecuadamente $\text{afun}(\mathbf{v})$.

⁹(a) Pero recuerde que si ε es muy pequeño, puede haber errores por pérdida de importancia! (b) Existen otras aproximaciones pero para presentar el método es suficiente considerar esta alternativa.

Capítulo 8

Integración Numérica (Cuadratura)

En este capítulo estudiaremos algoritmos para obtener computacionalmente valores para, en principio, integrales definidas que uno pudiera no resolver algebraicamente. Por ejemplo un caso clásico sería $\int_a^b \frac{\sin(x)}{x} dx$. Entonces, en este capítulo trabajaremos con la siguiente definición:

Def 17 (Integración Numérica o Cuadratura). *Es la obtención de la constante $c \in \mathbb{R}$, o una aproximación de c , de una integral definida $c = \int_a^b f(x) dx$ utilizando algún método numérico.*

¡MUY IMPORTANTE!

En este capítulo utilizaremos el siguiente ejemplo:

$$\int_{-1}^1 \exp(x) dx = \exp(1) - \exp(-1) = 2,3504023872876029137647637011912016303114359626682...$$

En general, existen un gran número de métodos que podríamos utilizar. En particular, en este capítulo estudiaremos solo algunos de ellos. Como es natural de esperar, no existe un método que resuelva todos los problemas. Por lo tanto la elección del método apropiado es muy importante, y primeramente depende, entre otras cosas, de los siguientes factores:

- Si la función $f(x)$ ha sido evaluada previamente en puntos típicamente equiespaciados...en estos casos resulta conveniente aplicar regla de Simpson, Trapecio o Punto Medio, por nombrar algunos.
- La función $f(x)$ puede evaluarse en puntos elegidos arbitrariamente ... en estos casos resulta conveniente aplicar la Cuadratura Gaussiana.

Antes de analizar los algoritmos antes mencionados, partiremos por las clásicas Sumas de Riemann! O en realidad, la versión numérica de ellas.

8.1. Suma de Riemann

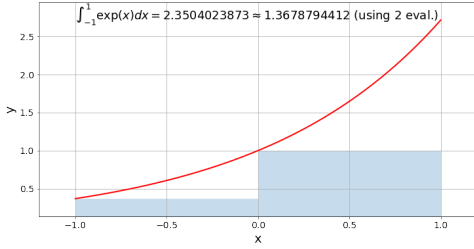
Recordemos que se dice que una función es integrable cuando las sumas izquierdas y derechas de Riemann convergen al mismo valor c , es decir,

$$c = \int_a^b f(x) dx.$$

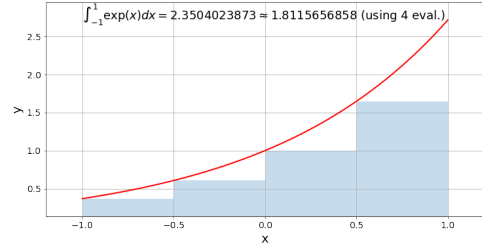
La Suma de Riemann por la izquierda se entiende por la siguiente formulación:

$$c = \int_a^b f(x) dx = \sum_{k=0}^{m-1} f(x_k)(x_{k+1} - x_k) + E_L,$$

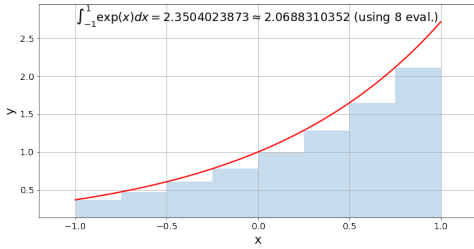
donde los puntos x_k pertenecen a la partición o discretización $P = \{x_0, x_1, \dots, x_m\}$ del intervalo de integración $[a, b]$ y E_L al error de la aproximación numérica. En la Figure 8.1 se presenta un ejemplo de la aproximación de la integral por una suma de Riemann por la izquierda con una discretización equiespaciada del intervalo $[-1, 1]$ de problema de ejemplo.



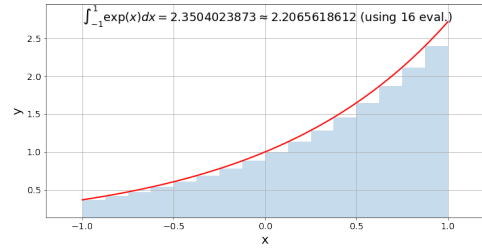
(a) $m = 2$, $c \approx 1,36788$.



(b) $m = 4$, $c \approx 1,81157$.



(c) $m = 8$, $c \approx 2,06883$.



(d) $m = 16$, $c \approx 2,20656$.

Figura 8.1: Ejemplo de suma de Riemann por la izquierda, donde el valor exacto es $c = \exp(1) - \exp(-1) \approx 2,35040 \dots$

Por otro lado, la suma de Riemann por la derecha se define como,

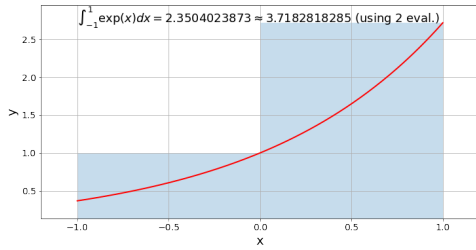
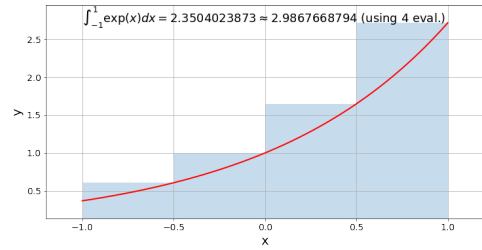
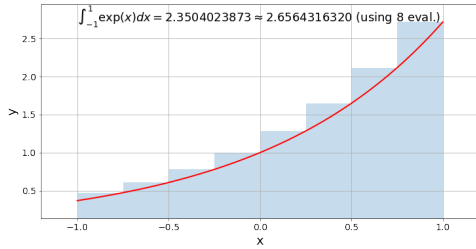
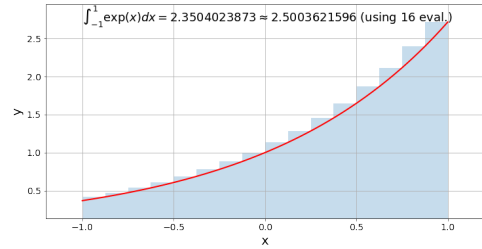
$$c = \int_a^b f(x) dx = \sum_{k=0}^{m-1} f(x_{k+1})(x_{k+1} - x_k) + E_R,$$

donde los puntos x_k pertenecen a la partición o discretización $P = \{x_0, x_1, \dots, x_m\}$ del intervalo de integración $[a, b]$ y E_R al error de la aproximación numérica. En la Figura 8.2 se muestra una interpretación de la suma de Riemann por la derecha.

Por lo tanto, podríamos utilizar tanto la suma de Riemann por la derecha como la suma de Riemann por la izquierda sobre una discretización del intervalo de integración $[a, b]$. Es decir,

$$\begin{aligned} c = \int_a^b f(x) dx &\approx \sum_{k=0}^{m-1} f(x_k)(x_{k+1} - x_k), \\ &\approx \sum_{k=0}^{m-1} f(x_{k+1})(x_{k+1} - x_k), \end{aligned}$$

donde el intervalo $[a, b]$ fue discretizado en $m + 1$ puntos que producen m segmentos $[x_k, x_{k+1}]$, produciendo la siguiente relación $a = x_0 < x_1 < \dots < x_{m-1} < x_m = b$. En particular si los puntos x_i son equiespaciados, entonces $x_{k+1} - x_k$ se denota como h o Δx , particularmente utilizaremos h en estos apuntes.

(a) $m = 2$, $c \approx 3,71828$.(b) $m = 4$, $c \approx 2,98677$.(c) $m = 8$, $c \approx 2,65643$.(d) $m = 16$, $c \approx 2,50036$.Figura 8.2: Ejemplo de suma de Riemann por la derecha, donde el valor exacto es $c = \exp(1) - \exp(-1) \approx 2,35040\dots$

8.2. Comentario previo a la presentación de los próximos algoritmos

En la sección anterior acabamos de construir implícitamente dos algoritmos para obtener una aproximación de $c = \int_a^b f(x)dx$. En esos algoritmos determinamos que al dividir el intervalo de integración, digamos el intervalo $[a, b]$, en varios intervalos más pequeños podemos mejorar la calidad de la aproximación, es decir, reducir el error. En las siguientes secciones se dará más énfasis a esta aproximación, construyendo varios algoritmos que realicen una integración en cada sub-intervalo¹ más pequeño para luego unir los resultados para obtener c , es decir $\int_a^b f(x)dx$, o una “buena” aproximación de c .

8.3. Regla del Punto Medio

El problema a resolver es el mismo, estimar $c = \int_a^b f(x)dx$, sin embargo ahora definiremos que el intervalo $[a, b]$ se dividirá en m sub-intervalos, es decir $[a = x_0, x_1, \dots, x_i, x_{i+1}, \dots, b = x_m]$ ². Por simplicidad de análisis trabajaremos con el primer sub-intervalo $[x_0, x_1]$, luego de haber definido el algoritmo de integración, podemos propagarlo para los otros intervalos.

El algoritmo o regla del punto medio se puede construir como el algoritmo que integra exactamente una constante³, digamos K , es decir:

$$\int_{x_0}^{x_1} K dx = K(x_1 - x_0),$$

por lo tanto, si consideramos que en un intervalo $[x_0, x_1]$ pequeño nuestra función de integración es aproximada-

¹Se utilizará el concepto de sub-intervalo para diferenciarlo del intervalo de integración $[a, b]$.

²Notar que se necesitan $m + 1$ puntos para generar m intervalos.

³En realidad es un algoritmo más astuto que solo integrar una constante!

mente constante obtenemos:

$$\int_{x_0}^{x_1} f(x) dx \approx f(x_*) (x_1 - x_0),$$

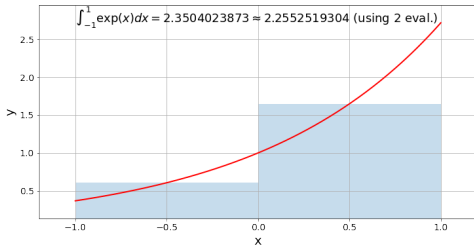
donde $x_* = \frac{x_0 + x_1}{2}$ ⁴. El error directo en el intervalo $[x_0, x_1]$ inducido por esta aproximación es explícitamente el siguiente:

$$\int_{x_0}^{x_1} f(x) dx = f(x_*) h + \frac{h^3}{24} f''(\hat{c}),$$

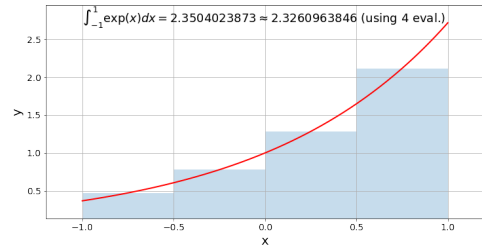
donde $h = x_1 - x_0$, $x_* = \frac{x_0 + x_1}{2}$, y $\hat{c} \in [x_0, x_1]$. El cálculo del error directo está relacionado con la expansión de Taylor de $f(x)$ en un punto adecuado. El error compuesto en el intervalo $[a = x_0, \dots, b = x_m]$ es el siguiente:

$$\begin{aligned} \int_a^b f(x) dx &= \sum_{i=1}^m \int_{x_{i-1}}^{x_i} f(x) dx \\ &= \sum_{i=1}^m h f(x_{*,i}) + \frac{(b-a)}{24} h^2 f''(\hat{c}), \end{aligned}$$

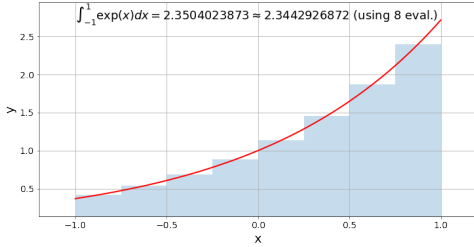
donde $h = (b-a)/m$, $x_{*,i} = \frac{1}{2}(x_{i-1} + x_i)$ y $\hat{c} \in [a, b]$.



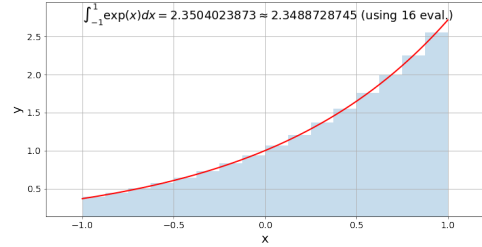
(a) $m = 2$, $c \approx 2,25525$.



(b) $m = 4$, $c \approx 2,32609$.



(c) $m = 8$, $c \approx 2,34429$.



(d) $m = 16$, $c \approx 2,34887$.

Figura 8.3: Ejemplo de la regla de Punto Medio, donde el valor exacto es $c = \exp(1) - \exp(-1) \approx 2,35040 \dots$

El siguiente código presenta una implementación de método del punto medio:

8.3.1. Pregunta sobre la Regla del Punto Medio

- ¿Será posible escribir una integración numérica como un producto punto?

⁴¿Qué ocurre con el algoritmo si se utiliza un punto distinto al punto medio desde el punto de vista del error directo que se explica a continuación?

```

1 def midpoint(myfun, m, a, b)
2     # Vectorization of function 'myfun' so it can be apply it
3     # to arrays element-wise.
4     f = np.vectorize(myfun)
5     # We want m bins, so we need m+1 points.
6     x = np.linspace(a, b, m+1)
7     h = (b-a)/m
8     xhat = x[:-1] + h/2
9     w = np.full(m,h)
10    return np.dot(f(xhat),w)

```

Algoritmo 11: Método del Punto Medio

- Si fuera posible, ¿Cuales serían los 2 vectores?
- ¿Qué tan rápido converge el algoritmo?
- ¿Cuál es el orden de convergencia?
- ¿Qué significa el orden de convergencia?
- ¿Cómo se observa numéricamente el orden de convergencia?
- ¿Qué implica computacionalmente el orden de convergencia?

8.4. Regla del Trapecio

El algoritmo o regla del trapecio propone aproximar la integral por medio de trapecios en cada sub-intervalo de integración, es decir para $[x_0, x_1]$ se obtiene:

$$\int_{x_0}^{x_1} f(x) dx \approx (x_1 - x_0) \frac{1}{2} (f(x_0) + f(x_1)).$$

El error directo en el intervalo $[x_0, x_1]$ inducido por esta aproximación es explícitamente el siguiente:

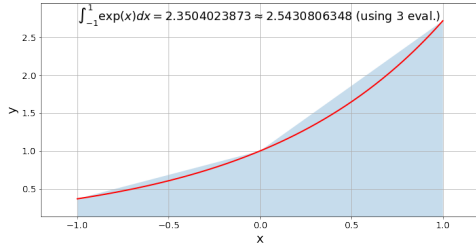
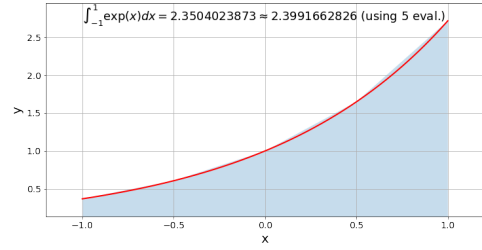
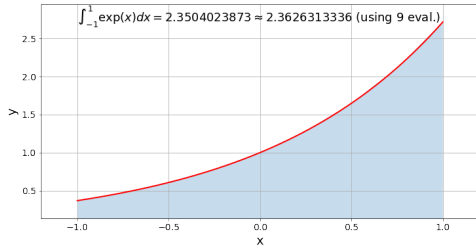
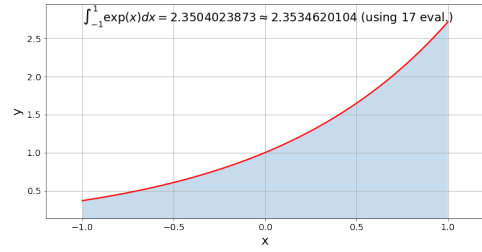
$$\int_{x_0}^{x_1} f(x) dx = \frac{h}{2} (f(x_0) + f(x_1)) - \frac{h^3}{12} f''(\hat{c}),$$

donde $h = x_1 - x_0$ y $\hat{c} \in [x_0, x_1]$. El error compuesto en el intervalo $[a = x_0, \dots, b = x_m]$ es el siguiente:

$$\begin{aligned} \int_a^b f(x) dx &= \sum_{i=1}^m \int_{x_{i-1}}^{x_i} f(x) dx \\ &= \frac{h}{2} \left[f(a) + f(b) + 2 \sum_{i=1}^{m-1} f(x_i) \right] - (b-a) \frac{h^2}{12} f''(\hat{c}), \end{aligned}$$

donde $h = (b-a)/m$ y $\hat{c} \in [a, b]$.

El siguiente código presenta una implementación de método del trapecio:

(a) $m = 2$, $c \approx 2,54308$.(b) $m = 4$, $c \approx 2,39916$.(c) $m = 8$, $c \approx 2,36263$.(d) $m = 16$, $c \approx 2,35346$.Figura 8.4: Ejemplo de la regla del Trapecio, donde el valor exacto es $c = \exp(1) - \exp(-1) \approx 2,35040 \dots$

8.4.1. Pregunta sobre la Regla del Trapecio

- ¿Cuál es el orden de convergencia?

8.5. Regla de Simpson

El algoritmo o regla del Simpson propone aproximar la integral por medio de parábolas, sin embargo en este caso se necesitarían 3 puntos o 2 sub-intervalos para la derivación de este, es decir para $[x_0, x_2]$. En particular si integramos una parábola obtenemos lo siguiente:

$$\int_{x_0}^{x_2} ax^2 + bx + c dx = (x_1 - x_0) \frac{1}{3} [(ax_0^2 + bx_0 + c) + 4(ax_1^2 + bx_1 + c) + (ax_2^2 + bx_2 + c)],$$

la cual induce la siguiente aproximación:

$$\int_{x_0}^{x_2} f(x) dx \approx h \frac{1}{3} [f(x_0) + 4f(x_1) + f(x_2)].$$

El error directo en el intervalo $[x_0, x_2]$ inducido por esta aproximación es explícitamente el siguiente:

$$\int_{x_0}^{x_2} f(x) dx = \frac{h}{3} (f(x_0) + 4f(x_1) + f(x_2)) - \frac{h^5}{90} f^{(4)}(\bar{c}),$$

donde $h = x_1 - x_0 = x_2 - x_1$ y $\bar{c} \in [x_0, x_2]$. Antes de definir la formulación compuesta debemos destacar que en este caso necesitamos tener una cantidad par de sub-intervalos, es decir m debe satisfacer la siguiente relación $m = 2n$,

```

1 def trapezoid(myfun, m, a, b):
2     f = np.vectorize(myfun)
3     x = np.linspace(a, b, m+1)
4     h = (b-a)/m
5     w = np.full(m+1, h)
6     # Why do we need the next two lines?
7     w[0]/=2
8     w[-1]/=2
9     return np.dot(f(x), w)

```

Algoritmo 12: Regla del Trapecio

donde $n \in \mathbb{N}$. Entonces la formulación compuesta en el intervalo $[a = x_0, \dots, b = x_m]$ es la siguiente:

$$\int_a^b f(x) dx = \frac{h}{3} \left(f(x_0) + 4 \sum_{i=1}^n f(x_{2i-1}) + 2 \sum_{i=1}^{n-1} f(x_{2i}) + f(x_m) \right) - (b-a) \frac{h^4}{180} f^{(4)}(\hat{c})$$

donde $h = (b-a)/m$, $\hat{c} \in [a, b]$ y recordar que m debe ser par.

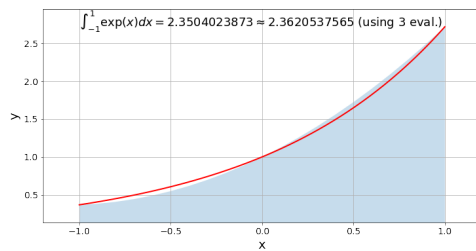
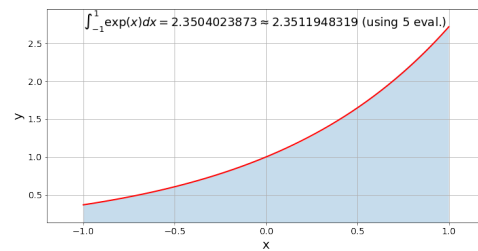
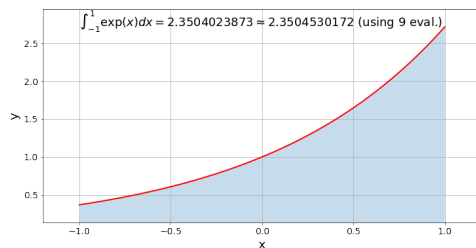
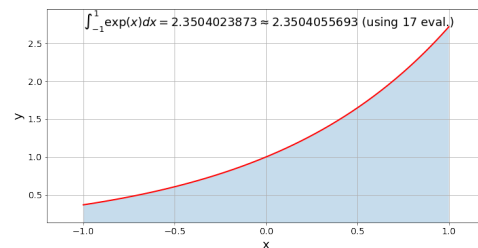
(a) $m = 2$, $c \approx 2,36205$.(b) $m = 4$, $c \approx 2,35119$.(c) $m = 8$, $c \approx 2,35045$.(d) $m = 16$, $c \approx 2,35040$.

Figura 8.5: Ejemplo de la regla de las parábolas de Simpson, donde el valor exacto es $c = \exp(1) - \exp(-1) \approx 2,35040\dots$

El siguiente código presenta una implementación de método de las parábolas de Simpson:

```

1 def simpsons(myfun, m, a, b):
2     f = np.vectorize(myfun)
3     x = np.linspace(a, b, m+1)
4     h = (b-a)/m
5     w = np.full(m+1, h/3)
6     w[1:-1:2] *= 4
7     w[2:-1:2] *= 2
8     return np.dot(f(x), w)

```

Algoritmo 13: Regla de Simpson

8.5.1. Pregunta sobre la Regla de las Parábolas de Simpson

- ¿Cuál es el orden de convergencia?
- ¿Qué significa el orden de convergencia en este algoritmo?
- ¿Cómo se observa numéricamente el orden de convergencia en este caso?
- ¿Qué implica computacionalmente el orden de convergencia?

8.6. Cuadratura Gaussiana - versión reducida

En los algoritmos anteriores observamos que la integración numérica se podía escribir como el producto punto entre un vector de pesos \mathbf{w} y unos nodos \mathbf{x} , es decir,

$$\mathbf{w} \cdot f(\mathbf{x}) = \sum_{i=1}^n w_i f(x_i).$$

Entonces aparece la pregunta natural,

¿Cuál sería la mejor forma de elegir los pesos w_i y nodos x_i para obtener $\int_a^b f(x) dx$? o ¿Podríamos elegir nodos x_i y pesos w_i para $i \in \{1, 2, \dots, n\}$ tal que integren exactamente polinomios de grado $2n-1$?

Esas son exactamente las pregunta que responde el algoritmo de la Cuadratura Gaussiana, ver apéndice E. En particular, para más detalles de la computación práctica utilizada para obtener los pesos y nodos de la cuadratura Gaussiana. A continuación se detalla una “breve” introducción a la computación de los nodos y pesos de la cuadratura Gaussiana.

En particular, considere que se quiere aproximar la función $f(x)$ por el polinomio $Q(x) = \sum_{i=1}^n L_i(x) f(x_i)$, es decir, por medio de una interpolación polinomial, en específico utilizando la interpolación de Lagrange. Notar que la dependencia en x se encuentra solamente en los términos $L_i(x)$ y que si bien los x_i no dependen de x , sabemos que son constantes pero desconocidas aún. Al realizar la integración⁵ obtenemos lo siguiente:

$$\begin{aligned} \int_{-1}^1 f(x) dx &\approx \int_{-1}^1 Q(x) dx = \int_{-1}^1 \sum_{i=1}^n L_i(x) f(x_i) dx \\ &= \sum_{i=1}^n f(x_i) \underbrace{\int_{-1}^1 L_i(x) dx}_{w_i}, \end{aligned}$$

⁵Notar que se está integrando en este caso en el intervalo $[-1, 1]$, esto simplifica el análisis pero no lo limita. Para cambiar de intervalo solo es necesario un cambio de variables.

donde $L_i(x) = \frac{l_i(x)}{l_i(x_i)}$, $l_i(x) = (x - x_1)(x - x_2) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)$, y tiene la propiedad de que $L_i(x_i) = 1$ y $L_i(x_j) = 0$ para $j \neq i$. En particular podemos decir que si uno elige los n nodos x_i , la cuadratura será exacta⁶ para polinomio de grado $n - 1$. En resumen, la construcción de $L_i(x)$ y consecuentemente la obtención de los pesos w_i dependen exclusivamente por la definición de los nodos x_i . En este caso los nodos x_i “adecuados”⁷ se definen como las raíces del n -ésimo polinomio de Legendre $p_n(x)$:

$$p_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} [(x^2 - 1)^n].^8$$

A continuación se listan algunos de ellos:

- $p_0(x) = 1$
- $p_1(x) = x$
- $p_2(x) = \frac{1}{2}(3x^2 - 1)$
- $p_3(x) = \frac{1}{2}(5x^3 - 3x)$
- $p_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3)$

Lo anterior se resume en la siguiente tabla:

n	$p_n(x)$	x_i	w_i
1	x	0	+2,000
2	$\frac{1}{2}(3x^2 - 1)$	$-\sqrt{\frac{1}{3}} \approx -0,577$ $+\sqrt{\frac{1}{3}} \approx 0,577$	+1,000 +1,000
3	$\frac{1}{2}(5x^3 - 3x)$	$-\sqrt{\frac{3}{5}} \approx -0,774$ 0,000 $+\sqrt{\frac{3}{5}} \approx +0,774$	$\frac{5}{9} \approx +0,555$ $\frac{8}{9} \approx +0,888$ $\frac{5}{9} \approx +0,555$
4	$\frac{1}{8}(35x^4 - 30x^2 + 3)$	$-\sqrt{\frac{15+2\sqrt{30}}{35}} \approx -0,861$ $-\sqrt{\frac{15-2\sqrt{30}}{35}} \approx -0,339$ $+\sqrt{\frac{15-2\sqrt{30}}{35}} \approx +0,339$ $+\sqrt{\frac{15+2\sqrt{30}}{35}} \approx +0,861$	$\frac{90-5\sqrt{30}}{180} \approx +0,347$ $\frac{90+5\sqrt{30}}{180} \approx +0,652$ $\frac{90+5\sqrt{30}}{180} \approx +0,652$ $\frac{90-5\sqrt{30}}{180} \approx +0,347$

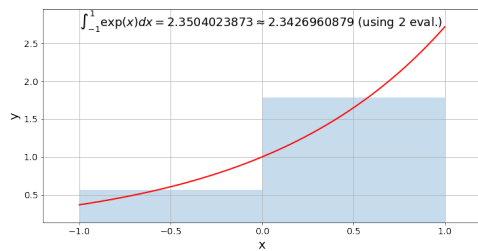
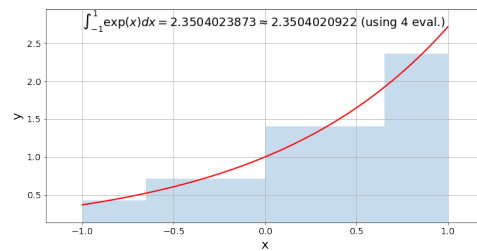
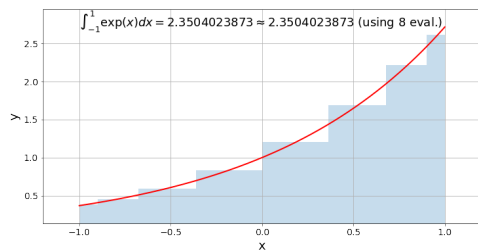
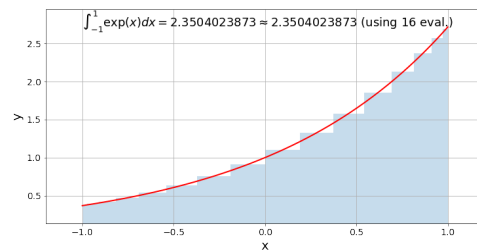
El siguiente código presenta una implementación de la Cuadratura Gaussiana, ver algoritmo 14.

Es importante señalar que la ventaja de la utilización de las raíces de los polinomio de Legendre como nodos de interpolación de la función $Q(x)$ para la aproximación de la función $f(x)$ recae en que integran exactamente hasta polinomios de grado $2n - 1$, esto significa que si la función es un polinomio de grado $2n - 1$ o menor, lo integra exactamente! Un buen ejercicio numérico es verificar esto en el jupyter notebook respectivo!

⁶Notar que en esta caso se está indicando que si uno elige n puntos arbitrarios, en particular que no sean necesariamente los nodos de la cuadratura Gaussiana, se puede concluir que la cuadratura será por lo menos exacta para polinomio de grado $n - 1$.

⁷Ver el apartado 8.7 para entender porque son los nodos adecuados.

⁸Notar que para hacer unitarios los polinomio en función de la norma $\int_{-1}^1 p_n^2(x) dx$ se debe agregar el coeficiente $\sqrt{\frac{2}{n+1}}$ a la expresión anterior. Esto se usa en el apartado 8.7.

(a) $m = 2$, $c \approx 2,34269$.(b) $m = 4$, $c \approx 2,35040$.(c) $m = 8$, $c \approx 2,35040$.(d) $m = 16$, $c \approx 2,35040$.Figura 8.6: Ejemplo de la Cuadratura Gaussiana, donde el valor exacto es $c = \exp(1) - \exp(-1) \approx 2,35040 \dots$

8.6.1. Preguntas sobre la Cuadratura Gaussiana

- ¿Cuál es el orden de convergencia?
- ¿Cómo se observa numéricamente el orden de convergencia en este caso?
- ¿Qué implica computacionalmente el orden de convergencia?
- ¿Es útil para todo tipo de funciones?

8.7. Cuadratura Gaussiana - versión extendida

En los algoritmos anteriores observamos que la integración numérica se podía escribir como el producto punto entre un vector de pesos \mathbf{w} y unos nodos \mathbf{x} , es decir,

$$\int_{-1}^1 f(x) dx \approx \sum_{i=1}^n w_i f(x_i) = \mathbf{w} \cdot f(\mathbf{x}).$$

Notar que hemos elegido el intervalo de integración de -1 a 1 por simplicidad en el análisis, sin embargo con un cambio de variable se puede transformar de un intervalo $[a, b]$ a $[-1, 1]$ y la conclusión se mantendrá válida.

¡MUY IMPORTANTE!

Antes de continuar, es importante señalar que esta sección se basa en el libro “Numerical Methods for Special Functions”, ver [3].

```

1 def gaussianquad(f, m, a, b):
2     x, w = gaussian_nodes_and_weights(m)
3     return np.dot(w, f(x))
4 def gaussian_nodes_and_weights(m):
5     if m==1:
6         return np.array([1]), np.array([2])
7     # Why an eigenvalue problems is related to the roots of the
8     # Legendre polynomial mentioned before? (This is a tricky
9     # question!)
10    beta = .5 / np.sqrt( 1. - (2.*np.arange(1.,m))**(-2) )
11    T = np.diag(beta,1) + np.diag(beta,-1)
12    D, V = np.linalg.eigh(T)
13    x = D
14    w = 2*V[0,:]**2
15    return x, w

```

Algoritmo 14: Cuadratura Gaussiana

La primera alternativa que uno pudiera visualizar para aproximar la integral $\int_{-1}^1 f(x) dx$ es aproximar $f(x)$ por un polinomio interpolador, digamos $P_{n-1}(x)$, en particular podemos definir el polinomio interpolador de la siguiente manera:

$$P_{n-1}(x) = \sum_{i=1}^n f(x_i) L_i(x),$$

donde hemos utilizado el método de interpolación de Lagrange, ver apartado 5.2. Ahora, el paso natural es integrar el polinomio interpolador para encontrar una aproximación de $\int_{-1}^1 f(x) dx$, es decir, $\int_{-1}^1 f(x) dx \approx \int_{-1}^1 P_{n-1}(x) dx$, lo cual nos da lo siguiente

$$\begin{aligned}
 \int_{-1}^1 P_{n-1}(x) dx &= \int_{-1}^1 \sum_{i=1}^n f(x_i) L_i(x) dx \\
 &= \sum_{i=1}^n f(x_i) \underbrace{\int_{-1}^1 L_i(x) dx}_{w_i}.
 \end{aligned} \tag{8.1}$$

De lo cual podemos concluir que la cuadratura será exacta si la función $f(x)$ es un polinomio de grado $n-1$ y se utilizan los nodos x_i definidos y pesos w_i obtenidos anteriormente. Lo que nos lleva a la siguiente definición:

Def 18 (Grado de exactitud). *Una regla de cuadratura $\sum_{i=1}^n w_i f(x_i)$ tiene grado de exactitud “m” si entrega el resultado exacto cuando la función $f(x)$ es cualquier polinomio de grado no mayor que “m” y no es exacta para todos los polinomios de grado “m+1”.*

Esta definición nos ayuda a comparar distintos algoritmos de integración numérica. Por ejemplo, en nuestro caso anterior podemos asegurar que integrará exactamente polinomios de grado $n-1$, es decir, el grado de exactitud será $n-1$.

Ahora, la pregunta natural que surge es la siguiente,

¿Cuál sería la mejor forma de elegir los pesos w_i y nodos x_i tal que se pueda maximizar el grado de exactitud al obtener $\int_a^b f(x) dx$?

Notar que tenemos la libertad de definir $2n$ coeficientes, es decir, $\{x_1, x_2, \dots, x_n\}$ y $\{w_1, w_2, \dots, w_n\}$. Entonces ahora la pregunta ambiciosa que tenemos es la siguiente,

¿Podríamos elegir nodos x_i y pesos w_i para $i \in \{1, 2, \dots, n\}$ tal que integren exactamente hasta polinomios de grado $2n-1$?

Para responder esto, considere el siguiente polinomio de grado $2n-1$,

$$\hat{p}_{2n-1}(x) = \sum_{j=0}^{2n-1} a_j x^j. \quad (8.2)$$

Del cual podemos notar que tenemos una combinación lineal de los monomios x^j para $j \in \{0, 1, 2, \dots, 2n-1\}$. Entonces, para integrar exactamente ese polinomio debemos integrar exactamente cada uno de los monomios x^j , lo cual se traduce en la siguiente expresión,

$$\int_{-1}^1 x^j dx = \sum_{i=1}^n w_i x_i^j,$$

donde

$$\int_{-1}^1 x^j dx = \begin{cases} 2, & \text{si } j = 0, \\ \frac{1 + (-1)^j}{1 + j}, & \text{e.t.o.c.} \end{cases}.$$

Lo cual genera el siguiente sistema de ecuaciones no-lineales:

$$\begin{aligned} \sum_{i=1}^n w_i &= 2, \\ \sum_{i=1}^n w_i x_i &= 0, \\ \sum_{i=1}^n w_i x_i^2 &= \frac{2}{3}, \\ &\vdots \\ \sum_{i=1}^n w_i x_i^{2n-1} &= 0. \end{aligned}$$

El cual se podría resolver utilizando el método de Newton en alta dimensión, ver apartado 4.6. Lamentablemente este camino genera un sistema de ecuaciones no-lineales mal condicionado, por lo que no es recomendado. Sin embargo el análisis anterior entrega los antecedentes necesarios para concluir de que sí pueden existir nodos x_i y pesos w_i que maximicen el grado de exactitud!⁹

Ahora, antes de explicar cómo podemos obtener los nodos y pesos que maximizan el grado de exactitud, debemos introducir la idea de producto interno de funciones, el cual se define de la siguiente forma,

$$\langle p, q \rangle = \int_{-1}^1 p(x) q(x) dx. \quad (8.3)$$

El cual nos ayuda a generar 2 resultados importantes,

⁹Igual sería interesante que usted obtenga los nodos y pesos para un valor de n pequeño, por ejemplo 4. ¿Cuales serían los valores?

¹⁰También es posible incluir una función de peso $w(x)$ en el producto interno, es decir, $\int_{-1}^1 p(x) q(x) w(x) dx$ pero por simplicidad se omite.

- $\|p\| = \sqrt{\langle p, p \rangle}$, es decir la “norma-2” de la función $p(x)$ definida en el intervalo $[-1, 1]$ ¹¹
- $\langle p, q \rangle = 0$, es decir, podemos introducir la noción de ortogonalidad e incluso ortonormalidad entre polinomios!

Respecto al segundo punto, podemos considerar el siguiente conjunto de polinomios $\mathbb{P}_N = \{p_i(x)\}_{i=0}^N$ tal que satisfacen la siguiente identidad,

$$\langle p_i, p_j \rangle = \delta_{i,j}, \quad i, j \in \{0, 1, 2, \dots, N\}.$$

Es decir, el conjunto de polinomios es ortonormal! Recuerde que $\delta_{i,j}$ corresponde la función Kronecker delta, que es 1 si $i = j$ y 0 en todo otro caso. La pregunta que puede surgir ahora es, ¿Cómo puedo construir polinomios ortonormales? La respuesta es simple, se puede utilizar la ortonormalización de Gram-Schmidt! Ver apartado 6.6.2. La única salvedad es que ahora trabajaremos con polinomio pero no vectores¹², por lo que se debe reemplazar el producto interno vectorial por el producto interno en la ecuación (8.3). El otro punto importante es notar que uno debe ortonormalizar un conjunto de vectores, en este caso los “vectores” serán los monomios $\{1, x, x^2, x^3, \dots\}$. Note que en este caso, considerando que se utilizará álgebra exacta, es suficiente utilizar el algoritmo clásico de Gram-Schmidt¹³ y no el algoritmo modificado. En este caso particular, los polinomios ortogonales que se obtienen¹⁴ son los polinomios de Legendre¹⁵.

Este resultado nos permite, al igual cómo se hace con vectores, re-escribir cualquier vector, es decir un polinomio en este caso, como una combinación lineal de la nueva base ortonormal de la siguiente forma, por ejemplo para un polinomio de grado m se tiene lo siguiente,

$$h_m(x) = \sum_{k=0}^m \langle p_k, h_m \rangle p_k(x).$$

Luego de introducir algunas componentes importantes, podemos demostrar porque la utilización de las raíces de los polinomios de Legendre son muy útiles como nodos al integrar numéricamente una función. Primero, considere que $f(x) \in \mathbb{P}_{2n-1}$, es decir, $f(x)$ es un polinomio de grado no mayor a $2n-1$, por lo cual se puede representar de la siguiente forma,

$$f(x) = r(x) p_n(x) + g(x), \quad \text{donde } r(x), g(x) \in \mathbb{P}_{n-1},$$

es decir, al dividir $f(x)$ por $p_n(x)$ obtenemos $r(x)$ con un resto $g(x)$. En particular, recuerde que podemos representar $p_n(x) = c_n (x - x_1)(x - x_2) \dots (x - x_n)$, donde c_n es solo una constante distinta de 0 que asegura que el polinomio $p_n(x)$ tiene norma 1. Entonces, integrando en el intervalo $[-1, 1]$ obtenemos,

$$\int_{-1}^1 f(x) dx = \int_{-1}^1 r(x) p_n(x) dx + \int_{-1}^1 g(x) dx. \quad (8.4)$$

Ahora procederemos a determinar ambas integrales del lado derecho de forma independiente. El primer paso es representar $r(x)$ como una combinación lineal de los polinomios de Legendre $p_k(x)$, en particular, solo necesitamos n polinomios¹⁶, es decir,

$$r(x) = \sum_{k=0}^{n-1} \langle r, p_k \rangle p_k(x).$$

¹¹Notar que se puede extender la definición a otros intervalos, pero por simplicidad la mantendremos en $[-1, 1]$.

¹²Uno puede considerar los polinomios como vectores de dimensión infinita!

¹³Es un buen ejercicio algebraico obtener por ejemplo los primeros 4 polinomios ortonormales!

¹⁴Recuerde que la única diferencia entre polinomios ortonormales y ortogonales es un factor conveniente para asegurar tener norma igual a uno.

¹⁵https://en.wikipedia.org/wiki/Legendre_polynomials

¹⁶Es decir va del polinomio $p_0(x)$ hasta el polinomio $p_{n-1}(x)$

Reemplazando la representación anterior en la primera integral del lado derecho de la ecuación (8.4) obtenemos,

$$\begin{aligned}\int_{-1}^1 r(x) p_n(x) dx &= \int_{-1}^1 \left(\sum_{k=0}^{n-1} \langle r, p_k \rangle p_k(x) \right) p_n(x) dx \\ &= \sum_{k=0}^{n-1} \langle r, p_k \rangle \int_{-1}^1 p_k(x) p_n(x) dx.\end{aligned}$$

De lo cual podemos notar que por ortonormalidad de los polinomios de Legendre sabemos que $\int_{-1}^1 p_k(x) p_n(x) dx = 0$, por lo tanto,

$$\int_{-1}^1 r(x) p_n(x) dx = 0.$$

Ahora, la segunda integral se puede obtener de la siguiente forma,

$$\int_{-1}^1 g(x) dx = \sum_{i=1}^n w_i g(x_i),$$

esto se debe a que $g(x)$ es un polinomio de grado $n-1$ o menor, es decir, en la ecuación (8.1) se demostró que la integral se puede obtener de forma exacta una vez definidos los nodos x_i . Entonces hemos llegado a la siguiente conclusión preliminar,

$$\int_{-1}^1 f(x) dx = \sum_{i=1}^n w_i g(x_i).$$

La pregunta que surge ahora es la siguiente,

¡MUY IMPORTANTE!

¿Debemos determinar el resto $g(x)$ de la división de $f(x)$ por $p_n(x)$ para poder integrar exactamente $f(x)$?^a

^aSugiero pensar unos minutos en esta pregunta antes de seguir, quizás llega de forma independiente a la respuesta!

Bueno, luego de pensar unos minutos, notamos lo siguiente: lo que realmente necesitamos evaluar es $g(x)$ en las raíces del polinomio de Legendre de grado n , no necesitamos evaluar $g(x)$ en puntos arbitrarios. Ahora, qué ocurre si evaluamos $f(x)$ en las raíces de $p_n(x)$, es decir en x_i ,

$$\begin{aligned}f(x_i) &= r(x_i) p_n(x_i) + g(x_i) \\ &= g(x_i).\end{aligned}$$

Es decir, como elegimos los nodos como las raíces del polinomio de Legendre de grado n , es decir, x_i , tenemos la siguiente identidad,

$$p_n(x_i) = 0, \quad \forall x_i.$$

En resumen, nuestro resultado final se expresa de la siguiente forma,

$$\int_{-1}^1 f(x) dx = \sum_{i=1}^n w_i f(x_i).$$

Lo cual es justamente lo que queríamos lograr! Aún falta la segunda parte de la demostración para cumplir con la definición 18, la cual corresponde a demostrar que la cuadratura anterior no integra exactamente todos los polinomios de grado $2n$. La segunda parte de la demostración se encuentra en [3], sección 5.3.1. Todo la explicación anterior se resume en el siguiente teorema, el cual corresponde a una versión simplificada al teorema 5.9 en [3].

Thm 20. Sea $p_n(x)$ el polinomio mónico de grado n tal que,

$$\int_{-1}^1 x^k p_n(x) dx = 0, \quad k \in \{0, 1, 2, \dots, n-1\}.$$

Sean x_1, \dots, x_n las raíces de $p_n(x)$ y w_i definido de la siguiente forma,

$$w_i = \int_{-1}^1 L_i(x) dx, \quad L_i(x) = \prod_{k=1, k \neq i}^n \frac{x - x_k}{x_i - x_k},$$

donde $i \in \{1, 2, \dots, n\}$. Entonces, la cuadratura,

$$\int_{-1}^1 f(x) dx \approx \sum_{i=1}^n w_i f(x_i)$$

es exacta para polinomio de grado $2n-1$ o menor.

Para el cálculo explícito de los nodos y pesos referimos al lector a revisar el apéndice [E](#).

Capítulo 9

Introducción a resolución numérica de Ecuaciones Diferenciales Ordinarias

En todos los problemas estudiados hasta este momento, es decir, búsqueda de ceros en 1D, resolución de sistemas de ecuaciones lineales y no-lineales, y mínimos cuadrados, hemos considerado que la incógnita es una variable, digamos x , o un vector \mathbf{x} . En este capítulo, sin embargo, consideraremos que la incógnita es una función, por ejemplo $y(t)$ o $y(x)$, dependiendo de una variable temporal o espacial, respectivamente.

Para el caso de que la función incógnita dependa de una variable temporal, es decir $y(t)$, se les denota *Problemas de Valor Inicial*¹ o IVP del inglés Initial Value Problem; y si depende de una variable espacial, es decir $y(x)$, se les denota *Problemas de Valor de Frontera* o BVP del inglés Boundary Value Problem. Como se podrá sospechar, también existen los IBVP, es decir Initial Boundary Value Problem, que dependen al mismo tiempo tanto del tiempo t como de una componente espacial x , es decir la incógnita es una función de, por lo menos, 2 variables, por ejemplo $u(x, t)$.

En este capítulo solo consideraremos ecuaciones diferenciales ordinarias, es decir, ecuaciones diferenciales que tiene como incógnita una función en una variable. Las funciones que dependen de más de una variable, digamos $u(x, t)$, se obtienen al resolver ecuaciones diferenciales parciales, esto sin embargo va más allá de este curso. Solo como ejemplo de una clásica ecuación diferencial parcial tenemos la clásica ecuación de calor, $u_t(x, t) = \frac{\partial^2 u}{\partial x^2}(x, t)$ para $x \in [0, L]$ y $t \in [0, T]$, con $u(x, 0) = f(x)$, $u(0, t) = l(t)$ y $u(L, t) = r(t)$. En la Figura 9.1 se muestran unos sketches de ejemplos de soluciones de un IVP, BVP e IBVP.

9.1. Problemas de Valor Inicial - IVP

Un IVP clásico es el siguiente: $y'(t) = y(t)$ con $y(0) = 1$ para $t \in [0, T]$. Esta ecuación diferencial pregunta si existe alguna función que al derivarla uno recupere la misma función. Aunque uno podría ya conocer la solución, el punto a destacar acá son las componentes del problema, las cuales son:

1. Tenemos una ODE²: $y'(t) = y(t)$.
2. Tenemos una condición inicial: $y(0) = 1$.
3. Y finalmente, tenemos un intervalo de tiempo donde se quiere resolver la ODE: $t \in [0, T]$.

¹También existen los problemas de valor final!

²ODE del inglés Ordinary Differential Equation

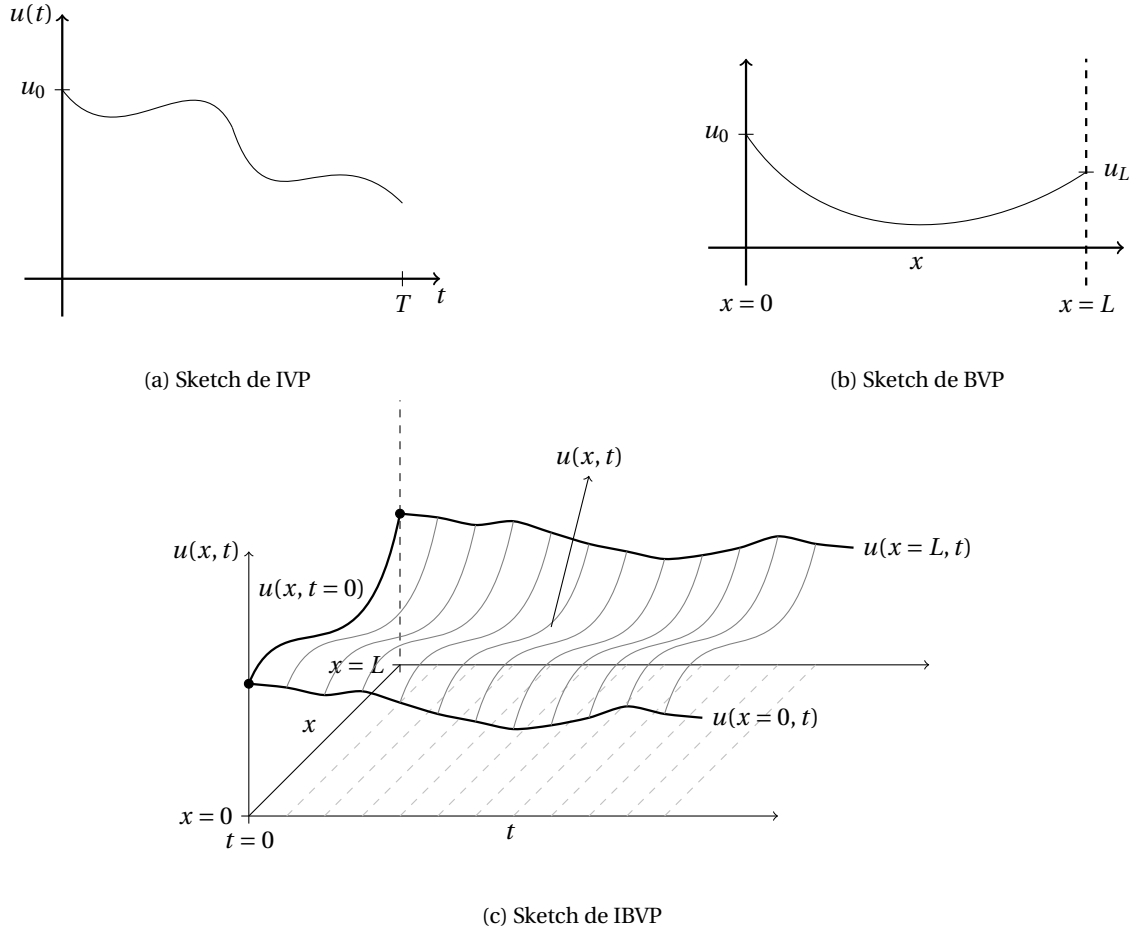


Figura 9.1: Sketches de IVP, BVP y IBVP.

Créditos imagen original: Profesor Cristopher Arenas. Updated in 2021.

La ODE es la que define la ecuación que la solución $y(t)$ debe satisfacer. Por ejemplo, podríamos proponer que la solución candidata del problema es la siguiente función: $\hat{y}(t) = 1$. Para la cual podemos determinar que la función $\hat{y}(t) = 1$ es continua y diferenciable en $t \in [0, T]$, además satisface la condición inicial $\hat{y}(0) = 1$. Pareciera entonces que fuéramos en buen camino, solo nos falta determinar si efectivamente satisface la ODE. Para lo cual debemos obtener su derivada, $\hat{y}'(t) = 0$, con lo cual llegamos a una contradicción ya que $\hat{y}'(t) = 0 \neq \hat{y}(t) = 1$, es decir no satisface la ODE $y'(t) = y(t)$. Lo que acabamos de hacer es proponer una solución y luego verificar si efectivamente resolvía el problema, en este caso no lo resolvía completamente, solo cumplía 2 de las 3 componentes indicadas. *Lo cual no es suficiente.*

La solución de la ODE anteriormente indicada es $y(t) = \exp(t)$, uno rápidamente puede determinar que la derivada de la función exponencial es la misma función, es decir, satisface la ODE. Al evaluar la función exponencial en $t = 0$ obtenemos $\exp(0) = 1$, así que también satisface la condición inicial y finalmente es continua y diferenciable en todo el intervalo $t \in [0, T]$, por lo tanto cumple con las 3 componentes indicadas. Para este caso particular fuimos capaces de resolver el problema de forma algebraica y explícita. Sin embargo, existen problemas para los cuales no se puede encontrar una solución de forma algebraica y debemos recurrir a métodos numéricos para encontrar aproximaciones de la función incógnita.

9.1.1. Notación para IVP

La forma general de un problema IVP que utilizaremos es la siguiente:

$$\dot{y} = f(t, y(t)), \quad (9.1)$$

$$y(0) = y_0, \quad (9.2)$$

$$t \in [0, T],$$

donde \dot{y} denota la derivada con respecto a t de la función $y(t)$, en el caso de que aparezcan dos puntos, corresponde a la segunda derivada con respecto a t . $f(\cdot, \cdot)$ es una función de 2 variables, que recibe como primer argumento el tiempo t y como segundo argumento la función $y(t)$, la cual es conocida. En el caso del ejemplo anterior con la ODE $\dot{y} = y$ y utilizando la notación de IVP, la función $f(t, y(t))$ corresponde a $f(t, y(t)) = y(t)$, es decir, solo dependía del segundo argumento. y_0 es la condición inicial entregada, para el problema de ejemplo ese valor era 1. Por último, T es el tiempo máximo donde uno quiere resolver la ODE.

Los IVP se clasifican en 2 categorías:

- Sistemas No-Autónomos: $\dot{y} = f(t, y)$, es decir la función f depende explícitamente de t .
- Sistemas Autónomos: $\dot{y} = f(y)$, es decir la función f solo depende del segundo argumento, i.e. la función $y(t)$.

Un ejemplo de un sistema no-autónomo es el siguiente:

$$\dot{y} = t y + t^3,$$

$$y(0) = 1,$$

$$t \in [0, 1].$$

El cual tiene solución conocida $y(t) = 3 \exp(t^2/2) - t^2 - 2$. Y un ejemplo de un sistema autónomo es el siguiente:

$$\dot{y} = c y (1 - y),$$

$$y(0) = y_0,$$

$$t \in [0, 1].$$

El cual tiene solución conocida:

$$y(t) = \begin{cases} 1, & \text{para } y_0 = 1. \\ 1 - \frac{1}{1 + \frac{y_0}{1 - y_0} \exp(c t)}, & \text{para } y_0 \neq 1. \end{cases}$$

En la siguiente sección veremos como obtendremos numéricamente una aproximación de la solución algebraica de una ODE. La idea principal de utilizar algoritmos numéricos para encontrar una aproximación de la solución recae en que para algunas ODE no existe una solución que se pueda obtener algebraicamente, por lo tanto los métodos numéricos nos permiten obtener soluciones a problemas matemáticos que de otra forma no podríamos!

9.1.2. Derivación inicial de distintos algoritmos para IVP - Método de Euler

La forma general de un IVP se presentó en las ecuaciones (9.1) y (9.2), lo interesante de esa ecuación es que conocemos una estructura general de la IVP, es decir: $\dot{y} = f(t, y(t))$ y podemos trabajar con ella. Nuestro primer paso será integrarla entre $t = 0$ y $t = t_1$:

$$\int_0^{t_1} \dot{y}(s) ds = \int_0^{t_1} f(s, y(s)) ds,$$

donde notamos que podemos utilizar el Teorema Fundamental del Cálculo en el lado izquierdo de la ecuación y obtenemos

$$y(t_1) - y(0) = \int_0^{t_1} f(s, y(s)) ds.$$

Despejando $y(t_1)$ al lado izquierdo y utilizando la condición inicial $y(0) = y_0$ obtenemos:

$$y(t_1) = y_0 + \int_0^{t_1} f(s, y(s)) ds. \quad (9.3)$$

Lo que acabamos de derivar es una expresión algebraica para obtener $y(t_1)$, sin embargo requiere integrar la función $g(s) = f(s, y(s))$ ³. En el capítulo anterior estudiamos varios algoritmos de integración numérica (o también llamados cuadraturas numéricas), ahora es el momento donde podemos aplicarlos!

Antes de aplicar algún algoritmo numérico debemos cuestionarnos la necesidad de este⁴, es decir, ¿Realmente necesitamos un algoritmo numérico si conocemos $f(t, y(t))$? ¡La respuesta es depende! Efectivamente nosotros conocemos $f(t, y(t))$, lo que no conocemos es $y(t)$ para $t = [0, t_1]$. Lo único que conocemos es $y(0) = y_0$.

Ahora, la pregunta que nos compete es la siguiente: ¿Cuál algoritmo numérico de integración podemos utilizar para obtener una aproximación de la integral $\int_0^{t_1} g(s) ds$? Para resolver esta pregunta, listemos y apliquemos los algoritmos numéricos a la integral en cuestión:

- Suma de Riemann por la izquierda: $\int_0^{t_1} g(s) ds \approx g(0) (t_1 - 0) = f(0, y(0)) t_1 = f(0, y_0) t_1$.
- Suma de Riemann por la derecha: $\int_0^{t_1} g(s) ds \approx g(t_1) (t_1 - 0) = f(t_1, y(t_1)) t_1$.
- Regla del punto medio: $\int_0^{t_1} g(s) ds \approx g\left(\frac{t_1+0}{2}\right) (t_1 - 0) = f(t_1/2, y(t_1/2)) t_1$.
- Regla del trapecio: $\int_0^{t_1} g(s) ds \approx (g(0) + g(t_1)) \frac{(t_1 - 0)}{2} = (f(0, y_0) + f(t_1, y(t_1))) t_1 / 2$.
- Regla del Simpson: $\int_0^{t_1} g(s) ds \approx (g(0) + 4g(t_1/2) + g(t_1)) \frac{(t_1 - 0)/2}{3} = (f(0, y_0) + 4f(t_1/2, y(t_1/2)) + f(t_1, y(t_1))) t_1 / 3$.
- Cuadratura Gaussiana⁵: $\int_0^{t_1} g(s) ds \approx \sum_{i=1}^n \hat{w}_i g(\hat{x}_i) = \sum_{i=1}^n \hat{w}_i f(\hat{x}_i, y(\hat{x}_i))$.

Entonces, ¿Cuáles algoritmos podemos aplicar? Para poder decidir que algoritmo podemos aplicar, debemos observar el término a la derecha del último signo igual en cada línea. Notamos que en general depende de $y(t)$ evaluado en diversos tiempos, es decir, necesitamos los valores de $\{y(0), y(t_1/2), y(t_1)\}$ y para todos los \hat{x}_i de la cuadratura Gaussiana. Sin embargo, solo conocemos $y(0) = y_0$. Por lo tanto, ¡Sólo podemos utilizar el algoritmo de la suma de Riemann por la izquierda! Pero ya aprenderemos algunas alternativas!

En resumen, hemos construido un algoritmo para encontrar una aproximación de $y(t_1)$ de la ecuación (9.3). Es decir,

$$y(t_1) = y_0 + \int_0^{t_1} f(s, y(s)) ds \approx y_0 + f(0, y_0) t_1. \quad (9.4)$$

De la cual obtenemos nuestra aproximación de $y(t_1)$, la cual llamaremos y_1 , es decir,

$$y_1 = y_0 + f(0, y_0) t_1.$$

De modo general obtenemos,

$$y_{i+1} = y_i + f(t_i, y_i) (t_{i+1} - t_i),$$

para $i \in \{0, 1, 2, \dots\}$ y recordar que y_0 es conocido. Este algoritmo es conocido como el Método de Euler, ver algoritmo 15 y figura 9.2.

El input del algoritmo consiste en lo siguiente:

³Por simplicidad de notación usaremos $g(s)$ como el integrando.

⁴¡Esto es válido para cualquier algoritmo numérico! Si se puede resolver explícitamente, para que resolverlo numéricamente. Pero si no se puede resolver algebraicamente, los métodos numéricos vienen al rescate!

⁵Notar que acá los nodos y pesos no están considerados en el intervalo $[-1, 1]$, por esta razón de agregado un “widehat” a los nodos y pesos.

```

1 def eulerMethod(t0,T,N,y0,f):
2     t = np.linspace(t0,T,N+1)
3     h = (T-t0)/N
4     y = np.zeros(N+1)
5     y[0] = y0
6     for i in np.arange(N):
7         y[i+1] = y[i]+f(t[i],y[i])*h
8     return t, y

```

Algoritmo 15: Método de Euler

- t_0 : Tiempo de la condición inicial $y(t_0) = y_0$.
- T : Tiempo final del intervalo a obtener la aproximación numérica, es decir $t = [t_0, T]$, por lo tanto $T > t_0$.
- N : Número de puntos de la discretización temporal. El número de puntos totales serán $N + 1$ dado que se incluye la condición inicial como un punto adicional. Esto significa que $h = (T - t_0)/N$.
- y_0 : Condición inicial, es decir $y(t_0) = y_0$.
- f : Función $f(t, y)$ que recibe 2 parámetros: t y y . Es la función que define el IVP: $\dot{y} = f(t, y)$.

El output corresponde a los vectores t y y de dimensiones $N + 1$ y representan la data (t_i, y_i) para $i \in \{0, 1, 2, \dots, N\}$.

En la siguientes secciones estudiaremos distintos algoritmos para resolver numéricamente ODEs y además IVPs de orden superior, o mejor llamados sistemas dinámicos.

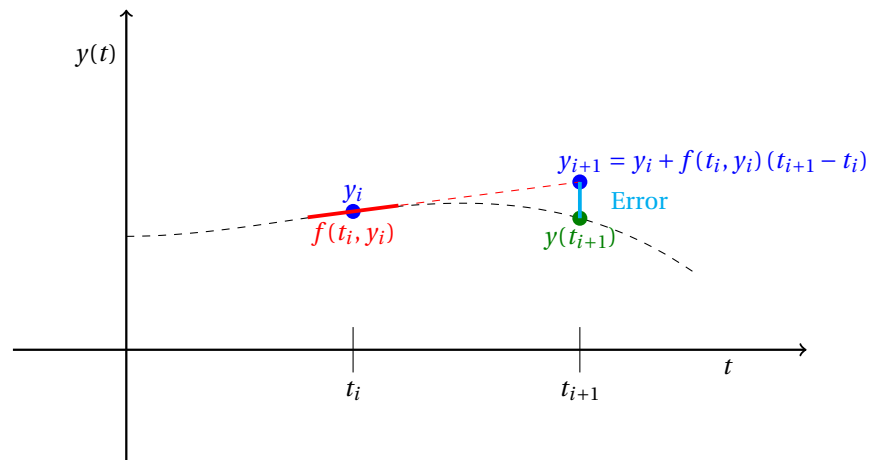


Figura 9.2: Se presenta diagrama generalizado del método de Euler, para conectarlo con la ecuación (9.4) considere que $i = 0$, $t_0 = 0$ y $t_1 = h$. En color azul se presentan las aproximaciones numéricas obtenidas, en línea continua roja y texto en rojo representa la pendiente obtenida, en línea segmentada roja representa una *extrapolación* para construir la estimación y_{i+1} , en verde se presenta el valor exacto de la función en cuestión, y en calipso se muestra el error.

9.1.3. Segunda derivación inicial - Backward Euler

En la derivación del método de Euler en la sección anterior analizamos todas las técnicas de integración numérica discutidas en el capítulo anterior, donde concluimos que solo podíamos utilizar la suma de Riemann por la izquierda en la ecuación (9.4), donde la aproximación era la siguiente:

$$y(t_1) = y_0 + \int_0^{t_1} f(s, y(s)) ds \approx y_0 + f(0, y_0) t_1.$$

Sin embargo, dándole una segunda mirada a la suma de Riemann por la derecha notamos que obtenemos la siguiente aproximación:

$$y(t_1) = y_0 + \int_0^{t_1} f(s, y(s)) ds \approx y_0 + f(t_1, y(t_1)) t_1.$$

De la misma forma que se realizó anteriormente, podemos obtener una ecuación para y_1 , la cual sería la siguiente:

$$y_1 = y_0 + f(t_1, y_1) t_1.$$

Si bien el desarrollo pareciera ser similar, hay una diferencia muy importante. En este caso nuestra incógnita y_1 está en ambos lados de la ecuación! La pregunta natural que surge sería: ¿Cómo podemos despejar y_1 ? Uno puede intentar dejar los términos que dependen de y_1 al lado izquierdo y los que no al lado derecho de la siguiente forma:

$$y_1 - f(t_1, y_1) t_1 = y_0.$$

Aún realizando esta manipulación algebraica no queda claro como despejar y_1 . ¿Qué podemos hacer entonces? Intentemos mover todos los términos al lado izquierdo:

$$y_1 - f(t_1, y_1) t_1 - y_0 = 0.$$

Dado que t_1 , y_0 y $f(t, y)$ son conocidos, podemos denotar el lado izquierdo como una función de y_1 , es decir:

$$\hat{f}(y_1) = y_1 - f(t_1, y_1) t_1 - y_0 = 0.$$

Ahora que se pudo expresar el problema de otra forma, observamos claramente que nos enfrentamos a un problema de búsqueda de ceros! Aquí podemos aplicar los algoritmos discutidos en el capítulo 3! Notar que el problema es equivalente a encontrar una raíz de $\hat{f}(x)$, esto significa que cuando uno obtenga el x^* tal que $\hat{f}(x^*) = 0$, podemos concluir que $x^* = y_1$.

De modo general debemos resolver un problema de búsqueda de ceros para cada time-step o paso de tiempo,

$$\hat{f}_i(y_{i+1}) = y_{i+1} - y_i - f(t_{i+1}, y_{i+1}) (t_{i+1} - t_i) = 0,$$

para $i \in \{0, 1, 2, \dots, N-1\}$. El Algoritmo 16 presenta una implementación del Método de Backward Euler.

Claramente podemos concluir que este algoritmo es más costoso computacionalmente que el método de Euler (Forward Euler) dado que se necesita realizar una búsqueda de ceros en cada paso de tiempo. La pregunta que surge entonces es la siguiente ¿Cuándo sería recomendable utilizar Backward Euler versus Forward Euler?⁶

9.1.4. Análisis del Error inducido por el Método de Euler

Hasta este momento hemos aprendido a aplicar 2 algoritmos para resolver numéricamente IVP. Sin embargo, no hemos discutido sobre el error que se comete al resolver un IVP con los algoritmos en cuestión.

A continuación presentamos un breve análisis para entender la idea detrás del análisis e instamos al lector a revisar más detalles en el libro guía y las referencias de este.

⁶Ver sección EXTRA 9.1.9!

```

1 def backwardEulerMethod(t0,T,N,y0,f):
2     t = np.linspace(t0,T,N+1)
3     h = (T-t0)/N
4     y = np.zeros(N+1)
5     y[0] = y0
6     for i in np.arange(N):
7         f_hat= lambda x: x - y[i]-f(t[i+1],x)*h
8         # You must select a solver, "find_root" is just a generic
           name. The input used are the anonymous function "
           f_hat" and the initial guess y_i.
9         x = find_root(f_hat,y[i])
10        y[i+1] = x
11    return t, y

```

Algoritmo 16: Método de Backward Euler

Considere que se aplicará el método de Euler (Forward Euler) a $\dot{y} = f(t, y(t))$ y $y(0) = y_0$ en el tiempo $t = t_i$, $y(t_i) = y_i$ y con paso h , i.e.,

$$y_{i+1} = y_i + h f(t_i, y_i). \quad (9.5)$$

Por otro lado, si consideramos el valor exacto $y(t_{i+1}) = y(t_i + h)$, y expandimos en Serie de Taylor la función considerando que h es pequeño obtenemos,

$$y(t_{i+1}) = y(t_i + h) = y(t_i) + \dot{y}(t_i) h + \ddot{y}(c) \frac{h^2}{2}.$$

Recordando que $\dot{y} = f(t, y(t))$, podemos evaluarlo en $t = t_i$ y obtenemos $\dot{y}(t_i) = f(t_i, y(t_i))$, lo simplifica la ecuación anterior a lo siguiente,

$$y(t_{i+1}) = y(t_i) + f(t_i, y(t_i)) h + \ddot{y}(c) \frac{h^2}{2}. \quad (9.6)$$

Ahora, obteniendo la diferencia entre la aproximación y_{i+1} y el valor exacto $y(t_{i+1})$ se obtiene el error $e_{i+1} = y_{i+1} - y(t_{i+1})$ de la siguiente forma,

$$\begin{aligned} y_{i+1} - y(t_{i+1}) &= y_i + h f(t_i, y_i) - y(t_i) - f(t_i, y(t_i)) h + \mathcal{O}(h^2) \\ &= (y_i - y(t_i)) + (f(t_i, y_i) - f(t_i, y(t_i))) h + \mathcal{O}(h^2) \\ e_{i+1} &= e_i + (f(t_i, y_i) - f(t_i, y(t_i))) h + \mathcal{O}(h^2), \end{aligned}$$

donde podemos considerar $y_i = y(t_i) + y_i - y(t_i) = y(t_i) + e_i$,

$$e_{i+1} = e_i + (f(t_i, y(t_i) + e_i) - f(t_i, y(t_i))) h + \mathcal{O}(h^2).$$

Para continuar, se necesita introducir la definición de que una función sea Lipschitz continua:

Def 19. Una función $f(t, y)$ es Lipschitz continua en la variable y en el rectángulo $S = [a, b] \times [\alpha, \beta]$ si existe una constante L (llamada la constante de Lipschitz) que satisface:

$$|f(t, y_1) - f(t, y_2)| \leq L |y_1 - y_2|$$

para cada (t, y_1) y (t, y_2) en S .

Ahora que conocemos la definición de que una función sea Lipschitz continua y considerando que $f(t, y)$ lo es con constante de Lipschitz L , y aplicando el valor absoluto al error, obtenemos lo siguiente,

$$\begin{aligned} |e_{i+1}| &\leq |e_i| + |(f(t_i, y(t_i) + e_i) - f(t_i, y(t_i)))| h + c h^2 \\ &\leq |e_i| + L |e_i| h + c h^2 \\ &\leq (1 + L h) |e_i| + c h^2 \\ &\dots \text{omitiendo algunos pasos...} \\ &\leq \mathcal{O}(h). \end{aligned}$$

Por lo tanto podemos concluir que el error del Método de Euler (Forward Euler) es $|y(t_i) - y_i| = \mathcal{O}(h)$ ⁷. Esto significa que el método de Euler es de orden 1, lo que implica que cuando se reduce h a la mitad, el error también se reduce a la mitad.

En resumen, lo que acabamos de analizar es el error cometido por la aproximación numérica del Método de Euler. La pregunta natural que aparece ahora es la siguiente: ¿Existen algoritmos de orden superior? Similar a lo que vimos en integración numérica. La respuesta es sí!

9.1.5. RK2: Runge-Kutta de segundo orden

En la sección anterior aprendimos que el Método de Euler era de orden 1, es decir $\mathcal{O}(h)$. Ahora, para obtener algoritmos de orden superior, podemos volver a revisar la aproximación de la integral original, es decir:

$$y(t_1) = y_0 + \int_0^{t_1} f(s, y(s)) ds.$$

En el caso del Método de Euler aplicamos la suma de Riemann por la izquierda y para el Método Backward Euler aplicamos la suma de Riemann por la derecha⁸. En ambos casos construimos los algoritmos asociados para efectivamente determinar y_{i+1} desde un valor conocido y_i . El siguiente paso natural es ver como aplicar otro algoritmo de integración numérica. En este caso, aplicaremos el Método del Punto Medio. Solo notar que ya lo aplicamos antes y determinamos que no se podía utilizar directamente, ahora sin embargo, explicaremos que se puede hacer para efectivamente utilizarlo. Entonces, aplicando el Método del Punto Medio obtenemos:

$$y(t_1) = y_0 + \int_0^{t_1} f(s, y(s)) ds \approx y_0 + f(t_1/2, y(t_1/2)) (t_1 - 0).$$

Recordamos de inmediato que el argumento utilizado antes fue que no conocíamos $y(t_1/2)$, el cual sigue siendo válido. ¿Qué podemos hacer entonces? La alternativa es estimarlo con el Método de Euler! Esto sería $k_1 = y_0 + \frac{t_1}{2} f(0, y_0)$, entonces obtenemos el siguiente algoritmo para el Método de Runge-Kutta de segundo orden⁹:

$$k_1 = f(t_i, y_i), \tag{9.7}$$

$$y_{i+1} = y_i + h f\left(t_i + h/2, y_i + \frac{h}{2} k_1\right). \tag{9.8}$$

El cual se traduce en la siguiente implementación en Python en el Algoritmo 17, adicionalmente se sugiere ver figura 9.3 y apartado 9.1.7.

Lo interesante de este algoritmo es que es un método de orden 2, es decir el error es $\mathcal{O}(h^2)$. Esto significa que si h se reduce a la mitad el error se reduce 4 veces! Lo que significa que podríamos obtener una aproximación numérica sin la necesidad de utilizar un h tan pequeño, lo que significa un menor tiempo de computación para un mismo error de aproximación numérico!

⁷Respecto a los pasos omitido en la derivación anterior, ver Corolario 6.5 en la página 296 del libro guía: Numerical Analysis, Timothy Sauer, Second Edition.

⁸A costa de resolver un problema de búsqueda de ceros en cada paso de tiempo o time-step!

⁹Hay diversas formas de escribirlo pero todas son equivalentes. Una segunda alternativa es considerar $k_1 = f(t_i, y_i)$, $k_2 = f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} k_1\right)$, y luego $y_{i+1} = y_i + h k_2$.

```

1 def RK2(t0, T, N, y0, f):
2     t = np.linspace(t0, T, N+1)
3     h = (T-t0)/N
4     y = np.zeros(N+1)
5     y[0] = y0
6     for i in np.arange(N):
7         k1 = f(t[i], y[i])
8         y[i+1] = y[i] + f(t[i] + h/2, y[i] + k1 * h/2) * h
9     return t, y

```

Algoritmo 17: RK2

9.1.6. RK4: Runge-Kutta de cuarto orden

Finalmente llegamos al famoso algoritmo de Runge-Kutta de 4to orden, o mejor conocido como RK4¹⁰. El algoritmo es el siguiente:

$$\begin{aligned}
 k_1 &= f(t_i, y_i) \\
 k_2 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} k_1\right) \\
 k_3 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} k_2\right) \\
 k_4 &= f(t_i + h, y_i + h k_3) \\
 y_{i+1} &= y_i + \frac{h}{6} (k_1 + 2 k_2 + 2 k_3 + k_4).
 \end{aligned}$$

El cual se traduce en la siguiente implementación en Python en el Algoritmo 18.

```

1 def RK4(t0, T, N, y0, f):
2     t = np.linspace(t0, T, N+1)
3     h = (T-t0)/N
4     y = np.zeros(N+1)
5     y[0] = y0
6     for i in np.arange(N):
7         k1 = f(t[i], y[i])
8         k2 = f(t[i] + h/2, y[i] + k1 * h/2)
9         k3 = f(t[i] + h/2, y[i] + k2 * h/2)
10        k4 = f(t[i] + h, y[i] + k3 * h)
11        y[i+1] = y[i] + (k1 + 2 * k2 + 2 * k3 + k4) * h/6
12    return t, y

```

Algoritmo 18: RK4

¹⁰En este caso se omitirá la derivación pero recomendamos al lector revisar el libro guía y las referencias incluidas en el mismo.

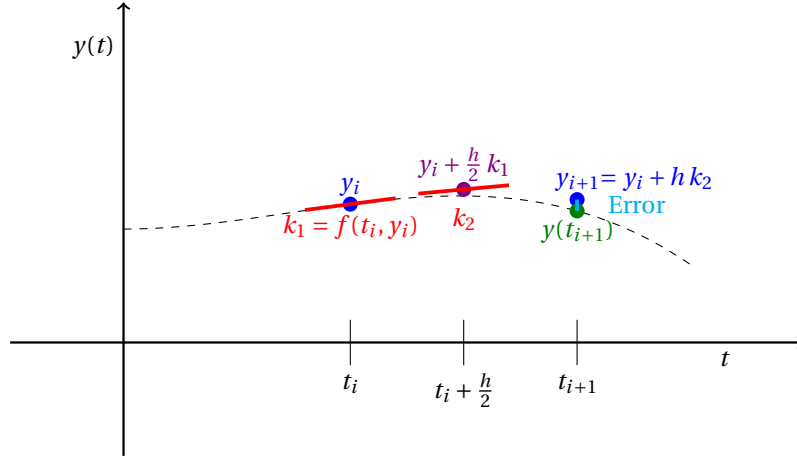


Figura 9.3: Se presenta diagrama del método RK2. En este diagrama se está considerando la representación $k_1 = f(t_i, y_i)$, $k_2 = f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} k_1\right)$, y luego $y_{i+1} = y_i + h k_2$. Esta representación es compatible con las ecuaciones (9.7) y (9.8), la relación se obtiene por medio de reemplazar k_2 en $y_{i+1} = y_i + h k_2$. En color azul se presentan las aproximaciones numéricas obtenidas, en línea continua roja y texto en rojo representa la pendiente obtenida, en morado se muestran estados intermedios obtenidos, en verde se presenta el valor exacto de la función en cuestión, y en calpso se muestra el error.

Claramente notamos que el costo computacional del método ha aumentado, requiriendo 4 llamadas a la implementación de la función $f(t, y)$, donde anteriormente se requerían solo 1 llamada para el Método de Euler y 2 llamadas para RK2¹¹. La gran ventaja de este algoritmo es que es un método de 4to orden, lo que significa que el error se comporta $\mathcal{O}(h^4)$. Esto significa que si h se reduce a la mitad, el error disminuye 16 veces! Se recomienda ver la figura 9.4 con el diagrama correspondiente propuesto.

9.1.7. EXTRA: Derivación alternativa de RK2 y más

En el apartado 9.1.5 se presentó la derivación del método de Runge-Kutta de 2do orden como una combinación del método de integración numérica del Punto Medio y el método de Euler, sin embargo esto no muestra todo el potencial de todas las variantes del método explícito de Runge-Kutta de segundo orden. Para poder apreciar al potencial por completo, reiniciaremos nuestro análisis desde una variante de la ecuación (9.3), es decir,

$$y(t_{i+1}) = y_i + \int_{t_i}^{t_{i+1}} f(s, y(s)) ds,$$

la diferencia es que ahora estamos avanzando del estado (t_i, y_i) al estado $(t_{i+1}, y(t_{i+1}))$. En este punto recordamos que nuestro procedimiento para determinar una aproximación de $y(t_{i+1})$ recae en poder aproximar lo mejor posible la integral $\int_{t_i}^{t_{i+1}} f(s, y(s)) ds$. Según lo ya discutido anteriormente, no se puede utilizar, en principio, la cuadratura Gaussiana debido a que no se conoce explícitamente $y(s)$ para valores mayores de $s \in]t_i, t_{i+1}]$. Sin embargo, podemos describir de la siguiente forma la aproximación:

$$y(t_{i+1}) \approx y_i + h \sum_{j=1}^m b_j f(t_i + c_j h, y(t_i + c_j h)).$$

¹¹En el método de Backward Euler se requieren probablemente más llamadas, pero tiene otros beneficios! Ver sección 9.1.9 para más detalles.

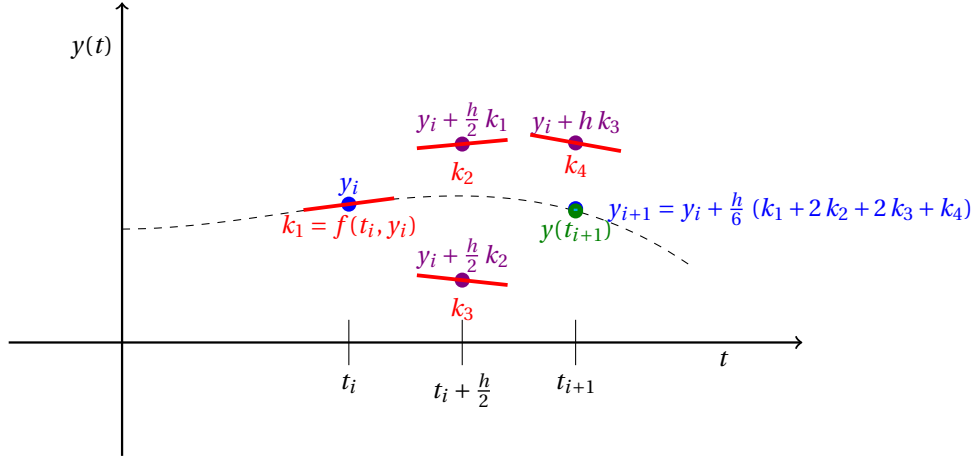


Figura 9.4: Se presenta diagrama del método RK4. **En este diagrama se trasladan verticalmente de forma significativa los estados intermedios k_2 , k_3 y k_4 para poder representarlos adecuadamente, pero recuerde que esto es solo algo de presentación en este diagrama.** En color azul se presentan las aproximaciones numéricas obtenidas, en línea continua roja y texto en rojo representa la pendiente obtenida, en morado se muestran estados intermedios obtenidos, en verde se presenta el valor exacto de la función en cuestión, y en calpso se muestra el error. En este último caso se omitió la palabra “Error” debido a la falta de espacio disponible.

Entonces, la alternativa disponible es simplemente aproximar estos valores utilizando la variables intermedias k_l de la siguiente forma¹²:

$$\begin{aligned}
 k_1 &= f(t_i, y_i), \\
 k_2 &= f(t_i + c_2 h, y_i + h a_{2,1} k_1), \\
 k_3 &= f(t_i + c_3 h, y_i + h (a_{3,1} k_1 + a_{3,2} k_2)), \\
 &\vdots \\
 k_m &= f\left(t_i + c_m h, y_i + h \left(\sum_{l=1}^{m-1} a_{m,l} k_l\right)\right).
 \end{aligned}$$

Lo que finalmente nos permite obtener una expresión para y_{i+1} de la siguiente forma,

$$y_{i+1} = y_i + h \sum_{j=1}^m b_j k_j.$$

La cual por conveniencia se puede compactar en la siguiente expresión utilizando el tableau de Butcher,

c_1					
c_2	$a_{2,1}$				
c_3	$a_{3,1}$	$a_{3,2}$			
\vdots	\vdots		\ddots		
c_m	$a_{m,1}$	$a_{m,2}$	\dots	$a_{m,m-1}$	
	b_1	b_2	\dots	b_{m-1}	b_m

¹²Existen variantes adicionales, por ejemplo acá mencionaremos la variante de métodos explícitos, sin embargo también existen métodos implícitos!

El cual organiza convenientemente los coeficientes antes utilizados. Notar por ejemplo que en la explicación previa se impone $c_1 = 0$, sin embargo se incluye por claridad.

Ahora, volviendo al caso particular en estudio, es decir RK2, podemos construir una versión reducida del desarrollo anterior, es decir:

$$\begin{array}{c|cc} 0 & & \\ c_2 & a_{2,1} & \\ \hline & b_1 & b_2 \end{array},$$

y

$$y_{i+1} = y_i + h (b_1 k_1 + b_2 k_2).$$

Más explícitamente tenemos,

$$y_{i+1} = y_i + h (b_1 k_1 + b_2 f(t_i + c_2 h, y_i + h a_{2,1} k_1)). \quad (9.9)$$

Notar que se mantiene k_1 sin reemplazarlo por su definición $f(t_i, y_i)$ simplemente porque puede considerarse una constante en este punto.

Entonces, la pregunta que emerge rápidamente es, ¿Cómo obtenemos los coeficientes desconocidos? Para responder esta pregunta se realizará un análisis similar al realizado en el apartado 9.1.4, es decir se comparará la expansión de Taylor de la expresión $y(t_i + h)$ y lo obtenido en y_{i+1} a medida que h tiende a 0. Lo primero es obtener la expansión en serie de Taylor de $y(t_i + h)$,

$$y(t_i + h) = y(t_i) + y'(t_i) h + \frac{y''(t_i)}{2} h^2 + \mathcal{O}(h^3).$$

Pero antes de continuar se recordará que conocemos explícitamente algunos términos en función de la información que conocemos de nuestro IVP, es decir de nuestro problema de valor inicial $\dot{y}(t) = f(t, y(t))$ y $y(t_i) = y_i$,

$$y'(t_i) = \dot{y}(t_i) = f(t_i, y(t_i)) = f(t_i, y_i),$$

y $y''(t)$ se obtiene derivando con respecto a t la ecuación $\dot{y}(t) = f(t, y(t))$, lo cual entrega $\ddot{y}(t) = f_t(t, y) + f_y(t, y) \dot{y} = f_t(t, y) + f_y(t, y) f(t, y)$, entonces,

$$y''(t_i) = \ddot{y}(t_i) = f_t(t_i, y_i) + f_y(t_i, y_i) f(t_i, y_i).$$

Esto significa que la expansión de Taylor de $y(t_i + h)$ queda expresada de la siguiente forma,

$$y(t_i + h) = y_i + f(t_i, y_i) h + \frac{f_t(t_i, y_i) + f_y(t_i, y_i) f(t_i, y_i)}{2} h^2 + \mathcal{O}(h^3). \quad (9.10)$$

Ahora, necesitamos re-escribir la ecuación (9.9) en función de potencias de h para poder comparar término a término con la expresión anterior, entonces también se realizará una expansión en serie de Taylor ahora de y_{i+1}

$$\begin{aligned} y_{i+1} &= y_i + h (b_1 k_1 + b_2 f(t_i + c_2 h, y_i + h a_{2,1} k_1)) \\ &= y_i + (b_1 + b_2) k_1 h + ((b_2 c_2) f_t(t_i, y_i) + (b_2 a_{2,1}) k_1 f_y(t_i, y_i)) h^2 + \mathcal{O}(h^3). \end{aligned} \quad (9.11)$$

Si comparamos término a término las ecuaciones ecuación (9.11) y ecuación (9.10) podemos obtener,

$$y_{i+1} - y(t_{i+1}) = \mathcal{O}(h^3),$$

siempre y cuando se cumplan las siguientes ecuaciones,

$$\begin{aligned} b_1 + b_2 &= 1, \\ a_{2,1} b_2 &= \frac{1}{2}, \\ c_2 b_2 &= \frac{1}{2}. \end{aligned}$$

El cual tiene la siguiente solución si definimos $c_2 = \alpha$,

$$\begin{aligned} a_{2,1} &= \alpha, \\ b_1 &= \frac{2\alpha - 1}{2\alpha}, \\ b_2 &= \frac{1}{2\alpha}. \end{aligned}$$

Entonces se obtiene la siguiente versión generalizada,

$$\begin{aligned} k_1 &= f(t_i, y_i), \\ k_2 &= f(t_i + \alpha h, y_i + \alpha h k_1), \\ y_{i+1} &= y_i + h \left(\frac{2\alpha - 1}{2\alpha} k_1 + \frac{1}{2\alpha} k_2 \right). \end{aligned}$$

Si se evalúa en $\alpha = \frac{1}{2}$ entonces se obtiene RK2! ¿Qué otras alternativas de α existen?

9.1.8. Sistemas Dinámicos

En esta sección presentaremos una extensión natural de los IVP, lo cual corresponde a los sistemas dinámicos. Por ejemplo considere el siguiente problema de valor inicial o IVP en 2 variables:

$$\begin{aligned} \dot{x} &= f_1(t, x, y) \\ \dot{y} &= f_2(t, x, y) \\ x(0) &= x_0 \\ y(0) &= y_0, \end{aligned}$$

donde $x(t)$ y $y(t)$ son funciones que dependen del tiempo y f_1 y f_2 son funciones en 3 variables. La pregunta que surge ahora es: ¿Cómo se aplica la teoría y métodos numéricos ya discutidos en este caso? Para responder la pregunta, debemos primero re-escribir el problema utilizando notación vectorial, lo que implica que nuestro problema de valor inicial en 2 variables queda de la siguiente forma:

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{F}(t, \mathbf{y}) \\ \mathbf{y}(0) &= \mathbf{y}_0, \end{aligned}$$

donde $\mathbf{y}(t) = \langle x(t), y(t) \rangle^T$, $\mathbf{F}(t) = \langle f_1(t, \mathbf{y}), f_2(t, \mathbf{y}) \rangle^T$, y $\mathbf{y}(0) = \langle x_0, y_0 \rangle^T$. Entonces básicamente hemos construido un IVP vectorial, lo que desde ahora llamaremos un Sistema Dinámico.

A continuación presentamos un paso de las versiones vectoriales de los algoritmos descritos anteriormente:

- Método de Euler:

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \mathbf{F}(t_i, \mathbf{y}_i) h$$

- Backward Euler

$$\begin{aligned} \mathbf{G}(\mathbf{x}) &= \mathbf{x} - \mathbf{y}_i - \mathbf{F}(t_i, \mathbf{x}) h \\ \mathbf{y}_{i+1} &= \text{findRoot}(\mathbf{G}(\mathbf{x}), \mathbf{y}_i) \end{aligned}$$

- RK2

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{F}(t_i, \mathbf{y}_i) \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + h \mathbf{F}(t_i + \frac{h}{2}, \mathbf{y}_i + \frac{h}{2} \mathbf{k}_1) \end{aligned}$$

■ RK4¹³

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{F}(t_i, \mathbf{y}_i) \\
\mathbf{k}_2 &= \mathbf{F}\left(t_i + \frac{h}{2}, \mathbf{y}_i + \frac{h}{2} \mathbf{k}_1\right) \\
\mathbf{k}_3 &= \mathbf{F}\left(t_i + \frac{h}{2}, \mathbf{y}_i + \frac{h}{2} \mathbf{k}_2\right) \\
\mathbf{k}_4 &= \mathbf{F}(t_i + h, \mathbf{y}_i + h \mathbf{k}_3) \\
\mathbf{y}_{i+1} &= \mathbf{y}_i + \frac{h}{6} (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)
\end{aligned}$$

En realidad la descripción anterior es válida para sistemas dinámicos de cualquier dimensión finita, es decir, pueden ser de 3, 4 o hasta n dimensiones! Los algoritmos se aplican igual.

9.1.8.1. El péndulo: Un ejemplo simple de un sistema dinámico

Para dar un ejemplo concreto de un sistema dinámico y al mismo tiempo entender como manejar IVP con derivadas de orden superior, es decir con términos que involucren desde la segunda derivada en el tiempo en adelante, analizaremos las ecuaciones de movimiento de un péndulo simple.

Consideremos que tenemos un péndulo simple¹⁴ con su pivote en un punto \mathbf{P} y con una barra rígida de largo l con una masa despreciable en comparación con la partícula con masa m en el extremo opuesto al pivote. Considerando la segunda Ley de Newton, obtenemos la siguiente ecuación de evolución para el ángulo θ formado entre la vertical bajo el pivote y la barra rígida,

$$\begin{aligned}
m l \ddot{\theta} &= -m g \sin(\theta), \\
\theta(0) &= \theta_0, \\
\dot{\theta}(0) &= \omega_0,
\end{aligned} \tag{9.12}$$

donde g es la constante de aceleración de gravedad, θ_0 es el ángulo inicial y ω_0 es la velocidad inicial del ángulo. Antes de continuar, procederemos a re-escribir la ecuación (9.12) cancelando la masa m y pasando l al lado derecho de la ecuación de la siguiente forma:

$$\ddot{\theta} = -\frac{g}{l} \sin(\theta). \tag{9.13}$$

La forma de la ecuación anterior aún no es compatible con las definiciones previas de un IVP, $\dot{y} = f(t, y)$, o un sistema dinámico, $\dot{\mathbf{y}} = \mathbf{F}(t, \mathbf{y})$. Entonces la pregunta que surge ahora es, ¿Cómo podemos re-escribir la ecuación (9.13) para que tenga la estructura de un IVP o un sistema dinámico? La respuesta es simple, con un cambio de variable conveniente¹⁵. El cambio de variables es el siguiente:

$$\begin{aligned}
y_1(t) &= \theta(t), \\
y_2(t) &= \dot{\theta}(t).
\end{aligned}$$

Aplicando la derivada con respecto a t obtenemos lo siguiente:

$$\begin{aligned}
\dot{y}_1(t) &= \dot{\theta}(t), \\
\dot{y}_2(t) &= \ddot{\theta}(t).
\end{aligned} \tag{9.14}$$

¹³Notar que la definición de \mathbf{k}_1 para RK2 es distinta a la de RK4!

¹⁴Ver esquema en https://es.wikipedia.org/wiki/P ndulo_simple y la p gina 305 del libro gu a: Numerical Analysis, Timothy Sauer, Second Edition.

¹⁵En realidad el cambio de variable es el mismo para problemas con derivadas de orden superior! Por lo que es muy importante que se entienda!

Donde notamos que el lado derecho de la primera ecuación es efectivamente la definición de $y_2(t)$ y el lado derecho de la segunda ecuación es la ecuación (9.13), en particular, $\ddot{\theta} = -\frac{g}{l} \sin(\theta)$. Realizando los reemplazos correspondientes, dejando todo en función de las nuevas variables $y_1(t)$ y $y_2(t)$ y eliminando la dependencia de t por simplicidad, obtenemos,

$$\begin{aligned} \dot{y}_1 &= y_2, \\ \dot{y}_2 &= -\frac{g}{l} \sin(y_1). \end{aligned}$$

¡El cual está escrito como un sistema dinámico! Donde $\mathbf{y} = \langle y_1(t), y_2(t) \rangle^T$ y $\mathbf{F}(t, \mathbf{y}) = \langle y_2, -\frac{g}{l} \sin(y_1) \rangle^T$. Finalmente surge la pregunta: ¿Cómo se implementa esto en Python¹⁶? Ver los Jupyter Notebooks del curso!

9.1.9. EXTRA: Análisis de Estabilidad Lineal

En esta sección analizaremos una pregunta que surgió anteriormente al final de la sección 9.1.3, ¿Cuándo sería recomendable utilizar Backward Euler versus Forward Euler? Recordemos que el método de Backward Euler requiere buscar la raíz de una función escalar para un IVP o encontrar la raíz de un sistema de ecuaciones no-lineales (o lineal) para un sistema dinámico. A primera impresión, Backward Euler parece ser mucho más costoso que Forward Euler, sin embargo, aclaremos esa duda en esta sección.

Para resolver esta duda, consideremos el siguiente IVP¹⁷:

$$\begin{aligned} \dot{y} &= \lambda y, \\ y(0) &= y_0, \end{aligned} \tag{9.15}$$

donde $\lambda \in \mathbb{C}$, $y_0 \in \mathbb{R}$ y $y_0 \neq 0$. Lo interesante de este problema es que conocemos su solución algebraica, la cual es:

$$y(t) = y_0 \exp(\lambda t).$$

Donde podemos descomponer λ es su parte real ($\lambda_{\text{Re}} = \text{Re}(\lambda)$) e imaginaria ($\lambda_{\text{Im}} = \text{Im}(\lambda)$) de la siguiente forma: $\lambda = \lambda_{\text{Re}} + i\lambda_{\text{Im}}$, donde $i^2 = -1$. Esta descomposición permite obtener la parte real e imaginaria ahora de la solución $y(t)$ de la siguiente forma:

$$\begin{aligned} y(t) &= y_0 \exp((\lambda_{\text{Re}} + i\lambda_{\text{Im}}) t) \\ &= y_0 \exp(\lambda_{\text{Re}} t + i\lambda_{\text{Im}} t) \\ &= y_0 \exp(\lambda_{\text{Re}} t) \exp(i\lambda_{\text{Im}} t), \end{aligned}$$

aplicando la formula de Euler¹⁸ obtenemos,

$$y(t) = y_0 \exp(\lambda_{\text{Re}} t) (\cos(\lambda_{\text{Im}} t) + i \sin(\lambda_{\text{Im}} t)).$$

Solo recordar que la función exponencial es siempre positiva, por lo tanto la función puede tender a infinito, mantenerse constante o decaer a 0. De la consideración anterior, podemos extraer tres comportamientos importantes a medida que $t \rightarrow \infty$:

- Si $\lambda_{\text{Re}} > 0$: La “envolvente”¹⁹ crece.

¹⁶Obviamente con NumPy y SciPy!

¹⁷Si bien es un caso particular, es el IVP que se analiza para realizar el análisis de estabilidad lineal. No es necesario realizar otro IVP o sistema dinámico.

¹⁸ $\exp(ix) = \cos(x) + i \sin(x)$

¹⁹ $\exp(\lambda_{\text{Re}} t)$

- Si $\lambda_{\text{Re}} = 0$: La “envolvente” es constante.
- Si $\lambda_{\text{Re}} < 0$: La “envolvente” decrece.

Estos casos son independientes del valor de la parte imaginaria de λ , es decir λ_{Im} . Antes de conectar el análisis con los métodos numéricos, destacamos el caso particular cuando $\lambda_{\text{Re}} < 0$. En este caso podemos concluir que la solución algebraica decrece, probablemente de manera oscilatoria si la parte imaginaria de λ es distinta de 0, pero lo importante es que debe decrecer. Este es un punto clave que lo retomaremos prontamente.

Ahora, que conocemos la solución algebraica de nuestro IVP (9.15), podemos aplicar algún algoritmo numérico y estudiar su comportamiento. Por ejemplo, si aplicamos el Método de Euler²⁰ obtenemos lo siguiente:

$$\begin{aligned} y_{i+1} &= y_i + \lambda h y_i \\ &= (1 + \lambda h) y_i. \end{aligned}$$

Y, aplicándolo recursivamente hasta llegar al tiempo $t = 0$, es decir hasta llegar a la condición inicial, obtenemos:

$$y_{i+1} = (1 + \lambda h) y_i = (1 + \lambda h)^2 y_{i-1} = \dots = (1 + \lambda h)^{i+1} y_0$$

Entonces, considerando que la parte real de λ es menor que cero, es decir $\text{Re}(\lambda) < 0$, y que h es mayor que cero.

¿Qué esperamos que ocurra con y_{i+1} a medida que i tiende a infinito? o en realidad ¿Qué **debería** ocurrir con y_i a medida que i tienda a infinito?

Recapitulando, tenemos los siguientes antecedentes:

Solución algebraica:	$y(t) = y_0 \exp(\lambda_{\text{Re}} t) (\cos(\lambda_{\text{Im}} t) + i \sin(\lambda_{\text{Im}} t))$
Aproximación con el método de Euler:	$y_i = y_0 (1 + \lambda h)^i$

Por simplicidad de análisis²¹, consideremos que $\lambda_{\text{Im}} = 0$, $\lambda_{\text{Re}} < 0$ y $y_0 > 0$, entonces tenemos:

Solución algebraica:	$y(t) = y_0 \exp(\lambda_{\text{Re}} t)$
Aproximación con el método de Euler:	$y_i = y_0 (1 + \lambda_{\text{Re}} h)^i$

Ahora entonces podemos responder claramente la pregunta “¿Qué **debería** ocurrir con y_i a medida que i tienda a infinito?” Debería ocurrir que y_i decaiga a 0 tal como lo haría $y(t) = y_0 \exp(\lambda_{\text{Re}} t)$, dado que $\lambda_{\text{Re}} < 0$. Sin embargo, la solución numérica podría no decrecer. Esto ocurriría en el caso de que $|1 + \lambda_{\text{Re}} h| \geq 0$. Para analizar el rango de valores adecuados que debe tomar h tal que decaiga la aproximación numérica debemos asegurar lo siguiente:

$$|1 + \lambda_{\text{Re}} h| < 1.$$

Despejando el rango de h obtenemos,

$$\begin{aligned} -1 &< 1 + \lambda_{\text{Re}} h < 1 \\ -2 &< \lambda_{\text{Re}} h < 0 \\ 0 &< h < \frac{2}{-\lambda_{\text{Re}}}. \end{aligned}$$

Recordando que $\lambda_{\text{Re}} < 0$, concluimos entonces el rango de h que hace que decaiga y_i a medida que i tiende a infinito es $\left] 0, \frac{2}{-\lambda_{\text{Re}}} \right[$. Este simple pero importante resultado nos indica que el método numérico solo tendrá sentido, o mejor dicho, será útil para un rango específico de h . De otra forma, solo se obtendrán resultados que no se pueden interpretar como que y_i es una aproximación de $y(t_i)$, es decir, son simplemente artefactos numéricos.

²⁰En realidad solo analizaremos el Método de Euler de forma explícita por ahora.

²¹Solo usaremos esta simplificación en este paso, pero en general debemos considerar que $\lambda \in \mathbb{C}$ y y_0 puede ser positivo o negativo.

Ahora, para el caso general, debemos considerar $\lambda \in \mathbb{C}$. Esto significa estudiar nuevamente la siguiente desigualdad:

$$|1 + \lambda h| < 1.$$

Para simplicidad de análisis, consideremos la siguiente representación del producto $\lambda h = x + iy$. Aplicando esta representación en la desigualdad obtenemos:

$$\begin{aligned} |1 + \lambda h| &< 1 \\ |1 + x + iy| &< 1 \\ |(1+x) + iy| &< 1 \\ (1+x)^2 + y^2 &< 1. \end{aligned}$$

Este resultado nos indica que si λ es un número complejo con su parte real negativa, debemos elegir h tal que se asegure que el producto λh esté dentro de la región de estabilidad. En este caso particular corresponde a un círculo de radio 1 centrado en punto $(-1, 0)$ del plano complejo. Es importante señalar que este resultado es compatible con el obtenido anteriormente, si en este caso se define $y = 0$, se obtiene lo mismo que se obtuvo antes! Ver Figura 9.5 donde se presentan las regiones de estabilidad de el método de Euler, RK2 y RK4.

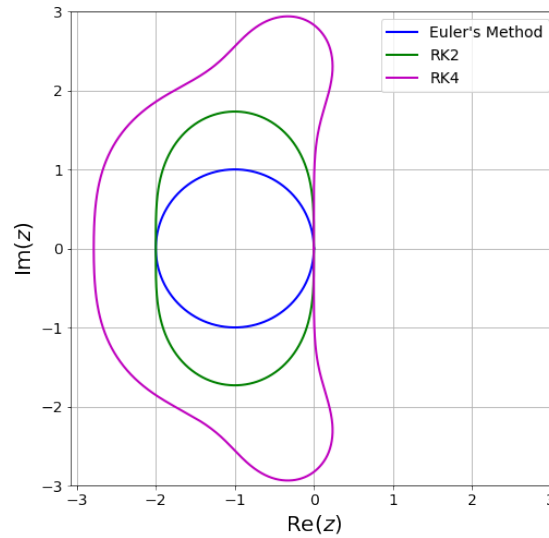


Figura 9.5: Regiones de estabilidad. La región de estabilidad del método de Euler es el interior del círculo de radio uno centrado en $(-1, 0)$ en el plano complejo. La región de estabilidad de RK2 es el interior de aproximadamente una elipse centrada en $(-1, 0)$, con semi-eje menor igual a uno en el eje real y semi-eje mayor igual a 1.75 aproximadamente. La región de estabilidad de RK4 es el interior de la región más grande mostrada. La cual se aproxima a un trapecio con coordenadas $(0, -2.8)$, $(0, 2.8)$, $(-2.65, -1)$ y $(-2.65, 1)$. Ver <https://www.chebfun.org/examples/ode-linear/Regions.html> para más detalles.

Una observación interesante es que si la parte real de λ es positiva, esto significa que $|1 + \lambda h|^i$ crecerá, y eso está correcto según lo que indica el análisis presentado anteriormente. En ese caso no hay problema con h !

Para finalizar esta sección, se presentan 2 preguntas:

1. ¿Cómo se aplican las regiones de estabilidad para sistema dinámicos?²²
2. ¿Cuál es la relación entre las regiones de estabilidad y la computación requerida para obtener la aproximación numérica?²³

9.2. Problemas de Valor de Frontera - BVP

Los problemas de valor de frontera o BVP se caracterizan principalmente porque no dependen del tiempo t sino de una variable espacial, por ejemplo x . Por ejemplo un posible BVP es el siguiente: $y''(x) = 0$ para $x \in]0, 1[$, $y(0) = c_1$ y $y(1) = c_2$. La pregunta que se nos hace aquí es la siguiente: ¿Cuál es la función en una variable que su segunda derivada es igual a 0 en el dominio abierto $]0, 1[$, y al evaluarla en $x = 0$ y en $x = 1$ nos da c_1 y c_2 , respectivamente?

Al igual que en el caso anterior de IVP, tenemos 3 componentes para el BVP también:

1. Tenemos una ODE: $y''(x) = 0$.
2. Tenemos condiciones de borde²⁴: $y(0) = c_1$ y $y(1) = c_2$.
3. Y finalmente, tenemos un intervalo espacial donde se quiere obtener la solución de la ODE: $x \in [0, 1]$ ²⁵.

En este caso es rápido verificar que la solución es $y(x) = c_1 + (c_2 - c_1)x$, la cual se sugiere verificar que efectivamente cumple con la ODE y las condiciones de borde al mismo tiempo. Es importante destacar que tanto la ODE como las condiciones de borde deben satisfacerse al mismo tiempo para que se denomine como la *solución*. Por ejemplo, $\hat{y}(x) = x$ satisface la ODE, es decir $\hat{y}''(x) = 0$, sin embargo no cumple con las condiciones de borde, por lo tanto no es una solución del BVP que estamos estudiando.

A continuación, estudiaremos dos alternativas para manejar problemas de valor de frontera o BVP. El primero es el *Método del Disparo*, que utiliza lo aprendido sobre IVP para resolver un BVP; y el segundo es el *Método de Diferencias Finitas*²⁶, en este caso se introduce un paradigma distinto para manejar las ODE, en el cual se resuelven en conjunto la ODE con las condiciones de borde, produciendo un sistema de ecuaciones lineal o no lineal, dependiendo del BVP.

9.2.1. Método del Disparo

La idea base detrás del método del disparo es *considerar* un BVP como un IVP, sin embargo tenemos que buscar la forma de manejar la condición de borde adicional que tenemos. Recuerde que:

IVP Una ODE, 1 condición inicial y el dominio o intervalo de tiempo.

BVP Una ODE, 2 condiciones de borde y el dominio o intervalo espacial.

La única diferencia sería la condición de borde adicional.

Por ejemplo, consideremos el mismo BVP presentado anteriormente: $y''(x) = 0$ para $x \in]0, 1[$, $y(0) = c_1$ y $y(1) = c_2$. Pero ahora realizando la siguiente interpretación: $x \rightarrow t$. De esta forma obtendríamos el siguiente IVP si es que

²²En este caso se debe aplicar todos a los valores propios de la matriz Jacobiana de \mathbf{F} evaluada en cada punto. Esto es muy costoso computacionalmente, por lo cual se sugiere estudiar el comportamiento de los valores propios del sistema dinámico preliminarmente y luego elegir convenientemente h . En principio, el valor propio con la parte real más negativa será el que restringirá h .

²³La relación es que dada una región de estabilidad debemos asegurarnos de elegir el Δt adecuado, lo que puede significar mucha computación si se necesita un Δt muy pequeño!

²⁴Antes se llamó condición inicial porque era efectivamente una condición inicial, acá sin embargo, tenemos la información de ambos extremos del dominio!

²⁵Notar que si bien la ODE está definida en $]0, 1[$ nos interesa encontrar la solución de la ODE que es válida en todo el intervalo, incluido los bordes!

²⁶No confundir este método con el método de las Diferencias Divididas de Newton!

renombramos la condición de borde $y(0) = c_1$ como condición inicial y agregar la condición inicial para la derivada, es decir:

$$\begin{aligned} y''(t) &= 0 \\ y(0) &= c_1 \\ y'(0) &= \alpha. \end{aligned}$$

Este problema se convierte en realidad en un sistema dinámico, similar al Péndulo discutido en la Sección 9.1.8.1, en particular en la ecuación (9.14). Del desarrollo anterior, surgen las siguientes dos preguntas:

- ¿Qué le ocurrió a la condición de borde $y(1) = c_2$?²⁷
- ¿Qué es α ?²⁸

Para resolver ambas preguntas, debemos continuar con la transformación de la ODE de segundo orden en el sistema dinámico asociado. Esto significa hacer siguiente cambio de variables:

$$\begin{aligned} y_1(t) &= y(t), \\ y_2(t) &= y'(t). \end{aligned}$$

Por lo cual obtenemos:

$$\begin{aligned} \dot{y}_1 &= y_2, \\ \dot{y}_2(t) &= 0, \\ y_1(0) &= c_1, \\ y_2(0) &= \alpha. \end{aligned} \tag{9.16}$$

Ahora destacamos la necesidad de introducir $y_2(0) = \alpha$, ya que para convertir exitosamente en BVP en un IVP necesitamos *conocer*²⁹ toda la data en el tiempo $t = 0$. En este punto, luego de definir algún valor para α , podríamos utilizar cualquier de los algoritmos discutidos para aproximar numéricamente el IVP³⁰ (9.16). Aparentemente hemos construido un algoritmo basado en los solvers de IVP para resolver un BVP! Pero en realidad nos falta un poquito.

Antes de continuar, debemos introducir la siguiente notación: $y_{1,k}^{[\alpha]}$, donde el sub-índice k denota que es una aproximación de $y_1(t_k)$ considerando que $y_2(0) = \alpha$ y que $t_N = 1$ ³¹. Es decir, la solución obtenida depende de la inicialización de $y_2(0)$ y de la discretización utilizada en el tiempo. No se entrará en mayor detalle acá sobre la discretización en el tiempo dado que ya se discutió en la Sección anterior, sin embargo el rol del α es importante.

Supongamos ahora que resolvemos el IVP de la ecuación (9.16) con algunos de los solvers considerando $\alpha = 0$, es decir tendríamos una versión discreta de $y_1(t)$ para $\alpha = 0$, es decir:

$$\{y_{1,0}^{[0]}, y_{1,1}^{[0]}, y_{1,2}^{[0]}, \dots, y_{1,N-1}^{[0]}, y_{1,N}^{[0]}\}.$$

De esta aproximación numérica podemos extraer el último elemento, el cual es $y_{1,N}^{[0]}$. Recordado que $y_{1,N}^{[0]}$ es una aproximación de $y_1(1)$ con $\alpha = 0$, podemos verificar si la aproximación obtenida satisface la condición de borde en $t = 1$, es decir $y(1) = c_2$. En general, es muy poco probable que se cumpla de inmediato, entonces, ¿Qué podemos hacer?

²⁷ Aún no lo hemos explicado!

²⁸ Este coeficiente nos ayudará a satisfacer la segunda condición de borde

²⁹ Aunque no lo conocemos aún, pero lo definiremos pronto!

³⁰ Por ejemplo: el método de Euler, Backward Euler, RK2 y RK4!

³¹ Tiempo *final* del intervalo/dominio en estudio

La respuesta es simple, resolver nuevamente el IVP ahora con otro valor de α , digamos $\alpha = \alpha_1$. El cual nos entregaría otra aproximación de $y_1(t)$:

$$\{y_{1,0}^{[\alpha_1]}, y_{1,1}^{[\alpha_1]}, y_{1,2}^{[\alpha_1]}, \dots, y_{1,N-1}^{[\alpha_1]}, y_{1,N}^{[\alpha_1]}\}.$$

Nuevamente podemos aplicar el mismo análisis, ¿Es $y_{1,N}^{[\alpha_1]}$ igual o cercano a $y_1(1) = c_2$?

En principio podemos iterar infinitas veces, pero en realidad podemos reconocer que estamos resolviendo un problema de búsqueda de ceros! Como los ya discutidos en el capítulo 3! Sin embargo acá la función a resolver es un poco más intrincada. Por ejemplo, consideremos el siguiente pseudo-código en Python:

```

1 # We consider "c1" and "c2" are already known.
2 def ErrorSecondBoundaryCondition(alpha, N=1000):
3     y0 = np.zeros(2)
4     y0[0] = c1
5     y0[1] = alpha
6     # Following notation introduced in previous chapter.
7     def f_IVP(t, y):
8         y1, y2 = y
9         dydt = [y2, 0]
10        return dydt
11    # Following notation introduced in previous chapter.
12    # We consider the output variable has dimensions (N+1)x2
13    y = SolverIVP(0, 1, N, y0, f_IVP)
14    return y[-1, 0] - c_2

```

El código presentado resuelve el IVP definido en la ecuación 9.16 con algún solver denominado SolverIVP, y luego retorna la diferencia de la aproximación entre el valor obtenido de la aproximación $y_1(1)$ dado α con c_2 . Lo interesante de esta función es que cuando se retorne el valor de 0 o algún valor muy cercano a 0 podemos encontrar el correcto valor de α que debemos usar para encontrar la solución de nuestro BVP! Es decir, debemos encontrar un cero de ErrorSecondBoundaryCondition. Aquí por ejemplo sería adecuado utilizar el método de la Bisección presentado en la sección 3.1. Luego de haber encontrado la raíz de ErrorSecondBoundaryCondition, resolvemos el IVP en la ecuación 9.16 y la solución obtenida para $y_1(t)$ será nuestra aproximación de la solución del BVP donde se satisface la ODE y ambas condiciones de borde al mismo tiempo!

Una posible interpretación del método del disparo es el clásico problema de lanzamiento parabólico de un objeto. En este caso podríamos suponer que queremos hacer llegar el objeto a un lugar específico lanzándolo con un ángulo pre-definido pero controlando la rapidez de salida. En el primer intento se lanza con rapidez v_0 , pero determinamos que no alcanza a llegar al lugar específico. Continuamos con una segunda rapidez, v_1 , y así continuamos hasta que encontramos la rapidez correcta, digamos v_c . Este procedimiento es en realidad una búsqueda de ceros, por lo cual lo importante es reconocer que efectivamente es una búsqueda de ceros para luego aplicar los algoritmos ya conocidos! Eso nos ahorrará un montón de tiempo!

En resumen, el método del disparo convierte un BVP a un IVP a costa de resolver un problema de búsqueda de ceros. En el caso particular discutido correspondía a una búsqueda de ceros en 1D, pero podría ser también una búsqueda de ceros en alta dimensión. En ese caso es bueno recordar que uno puede construir sus propias iteraciones de puntos fijo de alta dimensión, o aplicar métodos iterativos para resolver sistemas de ecuaciones lineales o no lineales. O también es posible ver esos problemas como problemas de minimización! Hay varias alternativas, algunas serán más eficientes que otras, usted deberá elegir cual es la adecuada para el problema que enfrente. Ese es el valor agregado que usted le puede dar!

9.2.2. Diferencias Finitas para ODE

En esta última sección de los apuntes, revisaremos el método de diferencias finitas para ODEs. Este método realiza una aproximación directa de la ODE e incluye de inmediato las condiciones de borde, generando un sistema de ecuaciones lineales o no lineales, dependiendo del BVP.

Antes de proceder a aproximar la ODE, debemos ir paso a paso y definir como podemos aproximar la primera y segunda derivada de una función³². Primero, recordemos la definición de la derivada:

$$y'(x) = \lim_{h \rightarrow 0} \frac{y(x+h) - y(x)}{h}.$$

Esta definición se puede interpretar como el cociente entre la variación de la función $y(x)$ en los puntos $x+h$ y x , y el incremento h . Ahora, dado que estamos trabajando en obtener una aproximación sobre una grilla discreta, considere que tenemos los siguientes pares ordenados:

$$\{(x_0, y_0), (x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\},$$

donde y_i representa la aproximación, hasta ahora desconocida, de $y(x_i)$, para $x_i = i h$, con $i \in \{0, 1, 2, \dots, N\}$ y $h = \frac{b-a}{N}$. En este caso consideramos que estamos discretizando el dominio $\Omega = [a, b]$, donde $x \in \Omega$. Para el caso particular del BVP estudiado anteriormente $a = 0$ y $b = 1$.

Ahora, continuando con el análisis, necesitamos construir un algoritmo que nos permita determinar los valores y_i desconocidos. Para poder construir el algoritmo, primero necesitamos definir la noción de derivada sobre esta grilla discreta, es decir responder la siguiente pregunta: ¿Cómo podríamos obtener una aproximación de la derivada si solo tenemos los valores discretos x_i y y_i ?

La respuesta es simple, no podemos asegurar que obtendremos el valor exacto de la derivada, pero si podemos obtener una aproximación de esta, por ejemplo:

$$y'(x_i) = \lim_{h \rightarrow 0} \frac{y(x_i+h) - y(x_i)}{h} = \lim_{h \rightarrow 0} \frac{y(x_{i+1}) - y(x_i)}{h} \approx \frac{y_{i+1} - y_i}{h}.$$

Esta aproximación es conocida como Forward Difference. De inmediato podemos introducir 2 aproximaciones adicionales y el orden³³ de aproximación de cada una:

$$\begin{aligned} y'(x_i) &= \frac{y(x_i+h) - y(x_i)}{h} + \mathcal{O}(h) \approx \frac{y_{i+1} - y_i}{h}, \text{ Forward Difference.} \\ y'(x_i) &= \frac{y(x_i) - y(x_i-h)}{h} + \mathcal{O}(h) \approx \frac{y_i - y_{i-1}}{h}, \text{ Backward Difference.} \\ y'(x_i) &= \frac{y(x_i+h) - y(x_i-h)}{2h} + \mathcal{O}(h^2) \approx \frac{y_{i+1} - y_{i-1}}{2h}, \text{ Central Difference.} \end{aligned}$$

Por lo tanto tenemos 3 formas de aproximar la primera derivada! Pero, ¿Cómo podemos aproximar la segunda

³²En general solo las primeras y segundas derivadas son suficientes para una gran cantidad de problemas, sin embargo, también se puede estimar derivadas de orden superior.

³³El orden de las aproximaciones se puede obtener de las expansiones de Taylor de cada una de ellas. También se puede utilizar las expansiones de Taylor para obtener diferentes aproximaciones de diferencias finitas, pero requiere una gimnasia aritmética importante!

derivada?³⁴ En este caso obtenemos:

$$\begin{aligned} y''(x_i) &= \frac{\text{Forward Difference} - \text{Backward Difference}}{h} + \mathcal{O}(h^2) \\ &= \frac{\frac{y(x_i+h) - y(x_i)}{h} - \frac{y(x_i) - y(x_i-h)}{h}}{h} + \mathcal{O}(h^2) \\ &= \frac{y(x_i+h) - 2y(x_i) + y(x_i-h)}{h^2} + \mathcal{O}(h^2) \\ &\approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} \end{aligned}$$

La pregunta natural que surge ahora es: ¿Cómo se utilizan estas aproximaciones para obtener un algoritmo que nos entregue una aproximación numérica de la solución del BVP?

Para responder la pregunta, apliquemos esta idea a nuestro caso de estudio, es decir al BVP: $y''(x) = 0$ para $x \in]0, 1[$, $y(0) = c_1$ y $y(1) = c_2$. En este caso necesitamos definir la cantidad de puntos que tendrá la discretización, por simplicidad elegiremos 5, es decir:

$$\{(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}.$$

De los cuales conocemos: $x_i = i h$, donde $h = 1/4$ e $i \in \{0, 1, 2, 3, 4\}$, y por las condiciones de borde $y_0 = c_1$ y $y_4 = c_2$. Entonces solo nos faltan 3 incógnitas: y_1 , y_2 y y_3 . Para determinar estas incógnitas necesitamos 3 ecuaciones, las cuales las podemos obtener de la discretización de la ODE, es decir de $y''(x) = 0$. Por lo cual necesitamos aproximar $y''(x) = 0$ en los puntos x_1 , x_2 y x_3 ³⁵, por lo cual obtenemos:

$$\begin{aligned} x = x_1, y''(x_1) = 0 &\Rightarrow \frac{y_2 - 2y_1 + y_0}{h^2} = 0, \\ x = x_2, y''(x_2) = 0 &\Rightarrow \frac{y_3 - 2y_2 + y_1}{h^2} = 0, \\ x = x_3, y''(x_3) = 0 &\Rightarrow \frac{y_4 - 2y_3 + y_2}{h^2} = 0. \end{aligned}$$

Reemplazando los valores conocidos de $y_0 = c_1$ y $y_4 = c_2$, y moviendo al RHS los valores conocidos, obtenemos el siguiente sistema de ecuaciones lineales:

$$\begin{aligned} y_2 - 2y_1 &= -c_1, \\ y_3 - 2y_2 + y_1 &= 0, \\ -2y_3 + y_2 &= -c_2. \end{aligned}$$

O escrito matricialmente,

$$\begin{pmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} -c_1 \\ 0 \\ -c_2 \end{pmatrix}$$

El cual podemos resolver con algunos de los algoritmos discutidos en los capítulos 4 y 7!

Para finalizar esta sección, se presentan las siguientes preguntas:

1. ¿Cuál método es más rápido para resolver BVP? El método del disparo o diferencias finitas.³⁶

³⁴Es la derivada de la derivada!

³⁵Notar que la ODE no es válida en los bordes del dominio, solo dentro del dominio, por eso solo se aproxima en el dominio y en los bordes se utilizan las condiciones de borde.

³⁶Depende! Ver el Jupyter Notebook asociado en GitHub.

2. ¿Podríamos aplicar el Método del Disparo o Diferencias Finitas a un BVP implícito de la forma $F(x, y(x), y'(x), y''(x)) = 0$ con $y(0) = c_1$ y $y(1) = c_1$?³⁷
3. ¿Cómo afecta la región de estabilidad al método de Diferencias Finitas para resolver BVP?³⁸
4. Aplique el método del disparo al siguiente problema: $y''(x) = 1$ para $x \in]0, 1[$, $y(0) = 5$ y $y(1) = -2$.
5. Aplique el método de diferencias finitas al siguiente problema: $y''(x) = 1$ para $x \in]0, 1[$, $y(0) = 5$ y $y(1) = -2$.

9.2.3. EXTRA: Otra forma de obtener aproximaciones de diferencias finitas

Anteriormente se presentaron 3 formas de aproximar las primeras derivada de una función sobre una grilla. Otra forma posible de decirlo es obtener la *derivada discreta* sobre una grilla, la cual claramente no es única, ya conocemos 3 alternativas!

En esta sección presentaremos brevemente un algoritmo para construir aproximaciones de diferencias finitas, sin embargo usted también podría consultar Wikipedia para ver las tablas de diferencias finitas de diversos ordenes, ver https://en.wikipedia.org/wiki/Finite_difference_coefficient. El algoritmo se basa en construir un polinomio interpolador $p(x)$, luego derivar el polinomio $p'(x)$ y finalmente evaluar el polinomio en el punto donde se quiere evaluar la derivada $p'(x_i)$. Por ejemplo, para obtener la aproximación de **Forward Difference** podemos interpolar los siguientes puntos: $\{(x_i, y_i), (x_i + h, y_{i+1})\}$. Lo cual nos da el siguiente polinomio interpolador³⁹,

$$p_{FD}(x) = y_i \frac{x - (x_i + h)}{x_i - (x_i + h)} + y_{i+1} \frac{x - x_i}{(x_i + h) - x_i}.$$

Al derivarlo obtenemos:

$$p'_{FD}(x) = y_i \frac{1}{x_i - (x_i + h)} + y_{i+1} \frac{1}{(x_i + h) - x_i}.$$

Y al evaluarlo en x_i se reduce a⁴⁰:

$$p'_{FD}(x_i) = \frac{y_{i+1} - y_i}{h}.$$

La cual es exactamente la aproximación de Forward Difference presentada anteriormente.

Ahora, consideremos que tenemos 3 puntos: $\{(x_i - h, y_{i-1}), (x_i, y_i), (x_i + h, y_{i+1})\}$. Primero debemos construir el polinomio interpolador:

$$\begin{aligned} p_{CD}(x) &= y_{i-1} \frac{(x - x_i)(x - (x_i + h))}{((x_i - h) - x_i)((x_i - h) - (x_i + h))} + y_i \frac{(x - (x_i - h))(x - (x_i + h))}{(x_i - (x_i - h))(x_i - (x_i + h))} + y_{i+1} \frac{(x - (x_i - h))(x - x_i)}{(x_i + h - (x_i - h))(x_i + h - x_i)} \\ &= y_{i-1} \frac{(x - x_i)(x - (x_i + h))}{2h^2} + y_i \frac{(x - (x_i - h))(x - (x_i + h))}{h^2} + y_{i+1} \frac{(x - (x_i - h))(x - x_i)}{2h^2}. \end{aligned}$$

Ahora se deriva el polinomio,

$$p'_{CD}(x) = y_{i-1} \frac{-h + 2x - 2x_i}{2h^2} + y_i \frac{2x - 2x_i}{h^2} + y_{i+1} \frac{h + 2x - 2x_i}{2h^2}.$$

Ahora solo nos queda evaluarlo en $x = x_i$,

$$\begin{aligned} p'_{CD}(x_i) &= y_{i-1} \frac{-h + 2x_i - 2x_i}{2h^2} + y_i \frac{2x_i - 2x_i}{h^2} + y_{i+1} \frac{h + 2x_i - 2x_i}{2h^2} \\ &= y_{i-1} \frac{-1}{2h} + y_{i+1} \frac{1}{2h} \\ &= \frac{y_{i+1} - y_{i-1}}{2h}. \end{aligned}$$

³⁷La respuesta es sí, pero deberá reconocer que tipo de problema está resolviendo ahora!

³⁸No lo afecta!

³⁹Por simplicidad utilizaremos el algoritmo de interpolación de Lagrange.

⁴⁰En este caso no se aprecia mucho la diferencia, pero se presentará luego un caso donde sí se aprecia la diferencia.

La cual es exactamente la aproximación de diferencias finitas centradas o **Central Difference** presentada anteriormente! Lo interesante de este procedimiento es que es muy algoritmizable, se podría implementar en SymPy!⁴¹, y además podríamos evaluar el polinomio derivado en algún otro punto. Por ejemplo si se evalúa $p'_{CD}(x)$ en $x = x_i - h$ se obtiene otra aproximación de diferencias finitas de segundo orden. La cual se puede utilizar al aproximar condiciones de borde que dependen de las derivadas!

⁴¹Es muy importante utilizar aritmética exacta para obtener esos coeficientes!, de otra forma se puede perder el orden de la aproximación.

Capítulo 10

Agradecimientos y registro de cambios

Esta sección lista tod@s l@s voluntari@s que han trabajado en diferentes versiones de estos apuntes. En el [Prefacio](#) se presenta una breve presentación de los apuntes. Originalmente estos apuntes están basados en el libro **NUMERICAL ANALYSIS** Second Edition - Timothy Sauer y apuntes de clases del profesor Claudio Torres, sin embargo estos han evolucionado significativamente en el tiempo gracias a los aportes de muchos estudiantes del curso de Computación Científica.

Desde la primera versión en \LaTeX de los apuntes se ha hecho un gran esfuerzo de ir entregando los créditos correspondientes a cada persona que ha hecho correcciones, sugerencias y/o comentarios, y se agradece cada uno de ellos.

En las siguientes Tablas se presentan los nombres de las personas que han realizado diversos aportes a los apuntes y, cuando es posible, se indican el detalles particular sugerido, especialmente en las Tablas finales. Debido a la evolución de los apuntes, las referencias a los cambios pueden haber dejado de ser válidas, por ejemplo, la enumeración de secciones inicial ya no es la mismas. Sin embargo, se mantendrán los nombres los detalles para referencia.

Las secciones fueron traspasadas a \LaTeX por:

Sección/es	Autor/a
1	Patricio Ramirez
2	Ian Zamorano
3	Felipe Osses
4	Francisco Salazar
5	Laura Bermeo
6 y 7	Yerson Escobar
8	Javier Ortiz
9	Germán Marcelo Treimun
10	Marco Rojas
11	Diego Reyes
12 y 13	Alfredo Gallardo
14	Renata Mella

Tabla 10.1: Autores traspaso a LaTeX de Apuntes CC1 - 2015.

Las secciones fueron corregidas por:

Sección/es	Autor/a
2 y 3	María José Astudillo
4 y 5	Daniel Quinteros
1,7,8,9 y 10	Fernando Iturbe

Tabla 10.2: Correctores Apuntes CC1 - 2016.

Las gráficas fueron corregidos por:

Autor/a
Roberto Fuentes

Tabla 10.3: Corrector gráficos Apuntes CC1 - 2016.

Las secciones fueron mejoradas por:

Sección	Autor/a
1 y 2	Daniel San Martín
3	Pablo Ibarra
4	Patricio Horth Medina
5 y 6	Victor Zuñiga Millán
7	Diego Villegas Aravena
8 y 10	Fabián Da Silva
9	Iván Lazo R
11, 12 y 14	Germán Treimun

Tabla 10.4: CC1-2016

Los anexos fueron creados por:

Sección	Autor/a
Review de calculo	Katherine Barros
Tutorial de instalación iPython	Diego Reyes

Tabla 10.5: CC1-2016

Apunte corregido y re-estructurado por:

Autor/a
Fernando Iturbe

Tabla 10.6: Correctores Apuntes CC1 - 2017.

Apunte corregido:

Autor/a
Rodrigo Hernández

Tabla 10.7: Correctores Apuntes CC1 - 2018.

Sugerencias:

Autor/a
Prof. Cristopher Arenas
Prof. Ariel Sanhueza

Tabla 10.8: Apuntes CC1 - 2019.

Apunte corregido:

Autor/a
Belén González (Sección 1.1.4)
Héctor Labraña (Typo pag. 23 versión 2020-v0.553)
Patricio Calderón (Typo sección 4.2.2. Cantidad de operaciones en G.E, versión 2020-v0.558)
Kevin Reyes (Typo inicio sección 4.9. Sub-índice α_0 y α_1 , versión 2020-v0.559)
C.Torres: Algoritmo clásico de Gram-Schmidt, sección 8.1.3, indentación extra eliminada en líneas 6 y 7.
C.Torres: Se agrega sección 9.3 Bonus Track: Teorema de Cayley–Hamilton y GMRes, versión 2020-v0.561

Tabla 10.9: Correctores Apuntes INF-285 - 2020.

Apunte actualizado:

Autor/a
Prof. Claudio Torres : Actualización importante (2021-v0.562) en capítulos 1, 2 y 3. Se agregó además las sección 5.3 y capítulos 9, 10 y 11, y se incluyeron Anexos A, B y C.
Prof. Claudio Torres: 2021-v0.562 - Typo en sección 2.2.2. Corregido en versión 2021-v0.563.
Ayudante Sebastián Sanchez: 2021-v0.562 - Typo en sección 1.3.6. Corregido en versión 2021-v0.563.
Rodrigo Cayazaya Marín: 2021-v0.562 - Errores y typos en secciones 1.2.5 y 1.3.2. Corregido en versión 2021-v0.564.
Harold Caballero: 2021-v0.562 - Typos en introducción de la sección 2.1. Corregido en versión 2021-v0.564.
Rodrigo Cayazaya Marín: 2021-v0.564 - Typos en secciones 1.4.2, 1.4.3.2, 1.5.1.2, y 1.5.2.1. Corregido en versión 2021-v0.565.
Jorge Sanhueza: 2021-v0.564 - Typo en sección 1.3.10. Corregido en versión 2021-v0.565.
Lukas Gutiérrez: 2021-v0.564 - Typo en sección 2.5. Corregido en versión 2021-v0.565.
Prof. Claudio Torres: 2021-v0.565 - Typo en sección 2.1 en la definición inicial de un número binario. Corregido en versión 2021-v0.566.
Comentario en clase 20210413: 2021-v0.566 - Corrección de signo en sección 2.4. Corregido en versión 2021-v0.567.
Yi Zhou: 2021-v0.566 - Aclaración en sección 2.6. Corregido en versión 2021-v0.567.
Prof. Claudio Torres: 2021-v0.566 - Aclaración en introducción de sección 2.2 y agregada referencia a sección 2.6. Corregido en versión 2021-v0.567.
Prof. Claudio Torres: 2021-v0.567 - Corregido typos y aclaración de “Comentario al margen” al final de la sección 2.9. Corregido en versión 2021-v0.568.
Fabián Levicán: 2021-v0.567 - Aclaración significativas en secciones 1.3.2 Nullspace, 1.3.4, y 1.3.5 Full rank. Corregido en versión 2021-v0.568.
Diego Gutiérrez: 2021-v0.567 - Typo inicio sección 2.2. Corregido en versión 2021-v0.568.
Diego Norambuena: 2021-v0.567 - Typos en secciones 2.2.2. Corregido en versión 2021-v0.568.
Prof. Claudio Torres: 2021-v0.567 - Se agrega comentario “¡MUY IMPORTANTE!” al principio de la sección 2.6 para aclarar el cómo se almacenan computacionalmente los números de “double precision”. Corregido en versión 2021-v0.568.
Rodrigo Cayazaya Marín: 2021-v0.568 - Typos en secciones 3.1.2, 3.2.2, y 3.2.3. Corregido en versión 2021-v0.569.
Diego Norambuena: 2021-v0.568 - Typos en secciones 2.9, 2.10, 3.2.1. Corregido en versión 2021-v0.569.
Tamara Letelier: 2021-v0.568 - Typos 3.2.4 Thm 5, 3.4.2 (2 typos), 3.5. Corregido en versión 2021-v0.569.
Pedro Yañez: 2021-v0.568 - Typos 3.4. Corregido en versión 2021-v0.569.
Prof. Claudio Torres: 2021-v0.568 - Cambio de bmatrix a pmatrix. Corregido en versión 2021-v0.569.
Prof. Claudio Torres: 2021-v0.568 - Actualización mayor en la primera parte del Capítulo 4. Corregido en versión 2021-v0.569.
Gonzalo Fernández: 2021-v0.569 - tildes en capítulo 2. Corregido en versión 2021-v0.570.
Isidora Ubilla Z.: 2021-v0.570 - typo 3.4.1. Corregido en versión 2021-v0.571.

Tabla 10.10: Correctores Apuntes INF-285 - 2021.

Autor/a
Bárbara Uribe Cataldo: 2021-v0.571 - typo footnote sección 2.2, typo del typo en sección 3.6. Corregido en versión 2021-v0.572.
Pedro Yañez: 2021-v0.571 - typo sección 2.2, sugerencia en sección 2.3, typo sección 2.4, typo sección 2.5, intro capítulo 4, sección 4.1.3. Corregido en versión 2021-v0.572.
Natalia Sanchez: 2021-v0.571 - typo Figura 3.4. Corregido en versión 2021-v0.572.
Nicolás Puente: 2021-v0.571 - typo sección 3.6, sección 4.1.3. Corregido en versión 2021-v0.572.
Jesus Oyanedel: 2021-v0.571 - sugerencia en sección 4.2. Corregido en versión 2021-v0.572.
Ignacio Jorquera: 2021-v0.571 - typo sección 4.3.1. Corregido en versión 2021-v0.572.
José Quezada: 2021-v0.571 - typo 4.4.3. Corregido en versión 2021-v0.572.
Sofía Carrasco Caimanque: 2021-v0.571 - typo sección 4.5 en el ejemplo. Corregido en versión 2021-v0.572.
Rodrigo Cayazaya Marín: 2021-v0.571 - typo sección 4.4.3, typo sección 4.5. Corregido en versión 2021-v0.572.
Prof. Claudio Torres: 2021-v0.571 - Actualización Capítulo 4 - Métodos Iterativos. Corregido en versión 2021-v0.572.
Prof. Claudio Torres: 2021-v0.572 - typo tabla 4.2, saltos de línea varios en sección 4.8.1, typo Def. 14, cambio en 4.8.7. Corregido en versión 2021-v0.573.
Axel Reyes: 2021-v0.573 - typo sección 4.4.1. Corregido en versión 2021-v0.574.
Rodrigo Cayazaya Marín: 2021-v0.574 - typos en sección 4.6. Corregido en versión 2021-v0.575.
Branco Catalán: 2021-v0.574 - typo secciones 4.2 y 4.3. Corregido en versión 2021-v0.575.
Diego Norambuena: 2021-v0.574 - typo secciones 3.1.1 y 3.2. Corregido en versión 2021-v0.575.
Prof. Claudio Torres: 2021-v0.574 - update and typos secciones 4.8.1, 4.8.4, 4.8.8 (importante), modificada explicación en ejemplo introductorio de factorización PALU en sección 4.5 basado en comentario de Rodrigo Cayazaya Marín. Actualización significativa en Capítulo 5: Introducción, 5.1, 5.2, 5.3, 5.5. Corregido en versión 2021-v0.575.
Cristóbal Abarca: 2021-v0.575 - typo sección 4.8.4. Corregido en versión 2021-v0.576.
Nicolás Puente: 2021-v0.575 - typo sección 5.2. Corregido en versión 2021-v0.576.
Prof. Claudio Torres: 2021-v0.575 - update significativo segunda parte de Capítulo 5. Corregido en versión 2021-v0.576.
Branco Catalán: 2021-v0.576 - typo sección B2 y B3. Corregido en versión 2021-v0.577.
Prof. Claudio Torres: 2021-v0.576 - update significativo en la primera parte de Capítulo 7, hasta la sección 7.4. Corregido en versión 2021-v0.577.
Ángela Martínez: 2021-v0.577 - typo sección 7.2. Corregido en versión 2021-v0.578.
Prof. Claudio Torres: 2021-v0.577 - typo en la versión del apunte. Se modificó el uso de \bar{a} y \bar{b} en la sección 7.2, antes de definir el sistema de ecuaciones lineales. Se eliminan las etiquetas en la Figura 7.3, se actualizan Figuras 7.2 y 7.3. Corregida referencia faltante a Figura 7.9 y color de y_i . Corregido en versión 2021-v0.578.
Prof. Claudio Torres: 2021-v0.578 - Se agregó sección (7.5). Actualizado en versión 2021-v0.579.
Prof. Claudio Torres: 2021-v0.578 - Se actualizó la segunda parte del capítulo 7, lo que corresponde la sección (7.6). Actualizado en versión 2021-v0.579.
Diego Norambuena: 2021-v0.578 - typo corregido en sección 5.6. Actualizado en versión 2021-v0.579.
Rodrigo Cayazaya Marín: 2021-v0.578 - typo corregido en sección 5.6, 5.7, introducción capítulo 5,. Actualizado en versión 2021-v0.579.
Tamara Letelier: 2021-v0.578 - typo corregido en sección 7.2 (2 typos), 7.4 . Actualizado en versión 2021-v0.579.
Marcos Maldonado: 2021-v0.578 - typo corregido en sección 5.2. Actualizado en versión 2021-v0.579.
Claudio Vergara: 2021-v0.578 - typo corregido en sección 7.3. Actualizado en versión 2021-v0.579.

Tabla 10.11: Correctores Apuntes INF-285 - 2021.

Autor/a
Prof. Claudio Torres: 2021-v0.579 - Se invierte la dirección del vector \mathbf{r} en Figuras 6.2 y 6.3, typo en caption de Figura 6.2, actualización importante en el Capítulo 8 de GMRes, sección 7.4.3, sección 7.4.1. Actualizado en versión 2021-v0.580.
Felipe Samur: 2021-v0.579 - typo corregido en sección 7.6.2. Actualizado en versión 2021-v0.580.
Diego Norambuena: 2021-v0.579 - typo corregido en sección 5.1. Actualizado en versión 2021-v0.580.
Felipe Lepin: 2021-v0.579 - typo corregido en sección 7.4.3, 7.4.2. Actualizado en versión 2021-v0.580.
Pedro Donoso: 2021-v0.579 - typo corregido en sección 7.4.2. Actualizado en versión 2021-v0.580.
Rodrigo Cayazaya Marín: 2021-v0.579 - typo corregido en sección 7.3, 7.5. Actualizado en versión 2021-v0.580.
Claudio Vergara: 2021-v0.579 - typo corregido en intro Capítulo 5, sección 5.1. Actualizado en versión 2021-v0.580.
Marcos Maldonado: 2021-v0.579 - typo corregido en Figura 7.8. Actualizado en versión 2021-v0.580.
Tamara Letelier: 2021-v0.579 - typo corregido en sección 4.8.4, 4.8.8, 5.6, 7.4.4. Actualizado en versión 2021-v0.580.
Ignacio Rojas: 2021-v0.579 - typo corregido en sección 7.6.2. Actualizado en versión 2021-v0.580.
Tamara Letelier: 2021-v0.580 - typo corregido en sección 7.6.2 (3 typos!). Actualizado en versión 2021-v0.581.
Rodrigo Cayazaya Marín: 2021-v0.580 - typo corregido en sección 7.6.2 (2 typos). Actualizado en versión 2021-v0.581.
Branco Catalán: 2021-v0.580 - typo corregido en sección 4.8.1, 4.8.8, intro Capítulo 5, 5.3. Actualizado en versión 2021-v0.581.
Diego Quezada: 2021-v0.580 - typo corregido en sección 4.4.1. Actualizado en versión 2021-v0.581.
Raimundo Gross: 2021-v0.581 - typo corregido en sección 7.1, 7.6.2. Actualizado en versión 2021-v0.582.
Iñaki Oyarzún: 2021-v0.581 - typo corregido en sección 4.8. Actualizado en versión 2021-v0.582.
Nicolás Cerpa: 2021-v0.581 - typo corregido en sección 7.6.3. Actualizado en versión 2021-v0.582.
Cristian Jara: 2021-v0.581 - typo corregido en sección 5.7. Actualizado en versión 2021-v0.582.
Jaime Pablo Rodríguez: 2021-v0.581 - typo corregido en sección 7.4.4. Actualizado en versión 2021-v0.582.
Branco Catalán: 2021-v0.581 - typo corregido en sección 4.9.3, 5.7. Actualizado en versión 2021-v0.582.
Prof. Claudio Torres: 2021-v0.581 - extensión de “Nota” en sección 8.2, typo en ecuación (8.20), update puntos finales sección 8.2, actualización Capítulo 9. Actualizado en versión 2021-v0.582.
Prof. Claudio Torres: 2021-v0.582 - fixing Riemann sum number of evaluations considered in plots, fixing some captions in chapter 9. Actualización Capítulo 10. Actualizado en versión 2021-v0.583.
Prof. Claudio Torres: 2021-v0.583 - typo corregido en Algoritmo 15. Actualizado en versión 2021-v0.584.
Raimundo Gross: 2021-v0.584 - typo corregido en sección 7.1, 8.2, 8.3. Actualizado en versión 2021-v0.585.
Nicolás Cerpa: 2021-v0.584 - typo corregido en sección 8.1. Actualizado en versión 2021-v0.585.
Branco Catalán: 2021-v0.584 - typo corregido en sección 2.2.2, 2.3, 9.4. Actualizado en versión 2021-v0.585.
Rodrigo Cayazaya: 2021-v0.584 - typo corregido en sección 8.2, sugerencia en la sección 9.1, typo en sección 9.6. Actualizado en versión 2021-v0.585.
Prof. Claudio Torres: 2021-v0.584 - Actualización de definiciones de sumas de Riemann por la izquierda y la derecha. Actualizado en versión 2021-v0.585.
Nicolás Puente: 2021-v0.584 - typo corregido en sección 9.4, 9.6, 10.1.3. Actualizado en versión 2021-v0.585.
Tamara Letelier: 2021-v0.584 - typo corregido en sección 10.1.2, 10.2. Actualizado en versión 2021-v0.585.
Prof. Claudio Torres: 2021-v0.585 - typo corregido en sección 9.3, 9.4 y 9.5. Actualizado en versión 2021-v0.586.
Raimundo Gross: 2021-v0.585 - typo corregido en sección 10.1.2. Actualizado en versión 2021-v0.586.

Tabla 10.12: Correctores Apuntes INF-285 - 2021.

Autor/a
Ignacio Rojas: 2021-v0.586 - typo corregido en sección 10.1.2. Actualizado en versión 2021-v0.587.
Cristian Jara: 2021-v0.586 - typo corregido en secciones 9.3 y 3 en sección 10.2.1. Actualizado en versión 2021-v0.587.
Claudio Vergara: 2021-v0.586 - typo corregido en sección 10.1.2. Actualizado en versión 2021-v0.587.
Esteban Carrillo: 2021-v0.586 - typo corregido en sección 8.1. Actualizado en versión 2021-v0.587.
Tomás Guttman: 2021-v0.586 - typo corregido en sección 1.1 y sugerencia incluida en la sección 1.1. Actualizado en versión 2021-v0.587.
Prof. Claudio Torres: 2021-v0.587 - cambio en los títulos de las secciones de BONUS a EXTRA para evitar confusión con lo BONUS en los Jupyter Notebooks, y actualización de tablas de agradecimientos antiguas. Actualizado en versión 2021-v0.588.

Tabla 10.13: Correctores Apuntes INF-285 - 2021.

Autor/a
Prof. Claudio Torres: 2021-v0.588 - se agrega caption a Figura 1.1 en sección 1.2.2, se agrega caption a Figura 1.2 en sección 1.2.3, Figura 1.3 actualizada y caption, sketches en Tabla 1.1 actualizados. Actualizado en versión 2021-v0.589.
Prof. Claudio Torres: 2021-v0.589 - typo corregido en sección 2.9, comentario al margen ->Otro ejemplo de pérdida de importancia. Actualizado en versión 2021-v0.590.
Pedro Arriagada: 2021-v0.590 - typo corregido en sección 2.4, hay una interpretación distinta en LaTeX de <code>dots }</code> dependiendo del entorno, se reemplazo por <code>{verb ... }</code> en la sección en cuestión. Actualizado en versión 2021-v0.592.
Tomás Barros: 2021-v0.592 - typo corregido en sección 4.1.1 y se agrega sugerencia en Tabla 4.1. Actualizado en versión 2021-v0.593.
Prof. Claudio Torres: 2021-v0.592 - actualización significativa en el Apéndice A y B sobre SVD y PCA. Actualizado en versión 2021-v0.593.
Tomás Barros: 2021-v0.593 - typo corregido en sección 4.1.3, 4.4, B.3 (4 sugerencias), B.4 (2 sugerencias), B.5 (2 sugerencias), B.6, 1.3.10, A.3, A.4, 4.3.1, 4.4, y 4.4.1 (2 sugerencias). Actualizado en versión 2021-v0.594.
Tomás Barros: 2021-v0.594 - 4 typos corregidos en sección 7.6.2, 2 typos corregidos en sección 5.7, 2 typos corregidos en sección 4.8.1. Actualizado en versión 2021-v0.595.

Tabla 10.14: Correctores Apuntes INF-285 - 2021-2.

Autor/a
Prof. Claudio Torres: 2021-v0.595 - actualización del Prefacio y el capítulo 10: Agradecimientos y registro de cambios . Actualizado en versión 2022-v0.596.
Prof. Claudio Torres: 2022-v0.596 - actualización sección 8.3, linealmente dependiente a linealmente independiente en el último párrafo. Actualización de figura 5.4. Actualizado en versión 2022-v0.597.
Franco Cabezas: 2022-v0.596 - 2 typos corregidos en sección 1.5.2.1. Actualizado en versión 2022-v0.597.
Ariane Carvajal: 2022-v0.596 - 5 typos corregidos en el Prefacio. Actualizado en versión 2022-v0.597.
Víctor Morales: 2022-v0.596 - typo corregido en sección 1.3.2. Actualizado en versión 2022-v0.597.
Sofía Palet: 2022-v0.596 - typo corregido en sección 2.2. Actualizado en versión 2021-v0.597.
Vicente Perelli: 2021-v0.596 - 2 typos corregidos en comentario al margen de la intro al Capítulo 2. Typo en sección 2.2. Typo corregido en la sección 2.2.1. Se modificó texto en la sugerencia en sección 2.2.2. Typo en sección 3.7. Actualizado en versión 2022-v0.597.
Diego Varela: 2022-v0.596 - typo corregido en sección 3.5. Actualizado en versión 2022-v0.597.
Rodrigo Munita: 2022-v0.596 - typo corregido en sección 3.2. Typo en teorema 4. Actualizado en versión 2022-v0.597.
Prof. Claudio Torres: 2022-v0.597 - Se agrega apartado 5.6 , se corrige tipo en el año en esta sección de agradecimientos. Actualizado en versión 2022-v0.598.
Víctor Morales: 2022-v0.597 - typo corregido en sección A.6. Actualizado en versión 2022-v0.598.
Etienne Rojas: 2022-v0.597 - typo corregido en sección B.3. Actualizado en versión 2022-v0.598.
Felipe Quintanilla: 2022-v0.597 - typos corregidos en sección B.3, 4.6. Actualizado en versión 2022-v0.598.
Prof. Claudio Torres: 2022-v0.598 - se agrega explicación adicional en apartado 7.2 . Actualizado en versión 2022-v0.599.
Ruth Vicuña: 2022-v0.598 - 2 typos corregidos en sección 4.8.8, typo en Introducción de Capítulo 5, typo corregido en sección 5.1. Actualizado en versión 2022-v0.599.
Etienne Rojas: 2022-v0.598 - typo corregido en sección 7.4.4, Figura 7.8. Actualizado en versión 2022-v0.599.
Bryan Alvarado: 2022-v0.598 - typo corregido en sección 7.2. Actualizado en versión 2022-v0.599.
Felipe Quintanilla: 2022-v0.598 - typo corregido en sección 1.5.2.1, sugerencia en sección 5.2. Actualizado en versión 2022-v0.599.
Pablo Betancourt: 2022-v0.598 - typo corregido en sección 4.6, typo en sección 4.8.4. Actualizado en versión 2022-v0.599.
Etienne Rojas: 2022-v0.598 - varios typos corregido en sección 7.3 asociados a 0 y 0 en las ecuaciones normales, typo en sección 7.6.1, typo en 7.6.4, typo en sección 7.6.2. Actualizado en versión 2022-v0.599.
Diego Varela: 2022-v0.598 - Indica que link de primera referencia no funciona, se debe verificar primero cual será el link actualizado antes de corregir. Hace sugerencia al orden de las figuras en sección 7.4, la idea es que aparezcan al lado del modelo asociado. Actualizado en versión 2022-v0.599.
Felipe Quintanilla: 2022-v0.598 - 4 typos corregidos en sección 5.6, 2 typos en sección 5.7, typo en sección 5.8. Actualizado en versión 2022-v0.599.
Diana Gil: 2022-v0.598 - typo antiguo pero ahora en sección 5.7, ahora se entendió la sugerencia. Actualizado en versión 2022-v0.599.
Felipe Quintanilla: 2022-v0.598 - typo corregido en sección 7.5. Actualizado en versión 2022-v0.599.
Cristóbal Abarca: 2022-v0.598 - typo corregido en sección 4.6. Actualizado en versión 2022-v0.599.
Etienne Rojas: 2022-v0.598 - Notación corregida en sección 8.1. Actualizado en versión 2022-v0.599.

Tabla 10.15: Correctores Apuntes INF-285 - 2022-1.

Autor/a
Renato Burgos, 2022-v0.599 - typo corregido en sección 1.6. Actualizado en versión 2022-v0.600.
Prof. Claudio Torres: 2022-v0.599 - actualización de apartado 8.6 y se agrega apartado 8.7. Se actualizan link de referencia [2]. Se actualiza apéndice E. Actualizado en versión 2022-v0.600.
Prof. Claudio Torres: 2022-v0.600 - typo corregido en sección apartado 9.1.8, en la definición de RK2, se re-escribió para ser consistente con apartado 9.1.5. Typo en apartado 4.2. Se extiende apéndice E. Actualizado en versión 2022-v0.601.
Alan Zapata: 2022-v0.601 - typo corregido en sección 4.5. Actualizado en versión 2022-v0.602.
Ruth Vicuña: 2022-v0.601 - typo corregido en las secciones 9.1 y 9.2. Actualizado en versión 2022-v0.602.
Felipe Quintanilla: 2022-v0.601 - typo corregido en las secciones 8.1 (2 typos), 9.1, 5.6, 4.5 (2 typos). Actualizado en versión 2022-v0.602.
Ignacio Quintana: 2022-v0.601 - typo corregido en sección 8.2. Actualizado en versión 2022-v0.602.

Tabla 10.16: Correctores Apuntes INF-285 - 2022-1.

Autor/a
Prof. Claudio Torres: 2022-v0.601 - actualización significativa en el orden de los capítulos, varios temas se pasaron al apéndice. Se agrega apartado 4.8 y se actualiza el prefacio. Actualización apartado 1.4, apartado 1.5.2.1, apartado 2.4 y apartado 5.11. Actualizado en versión 2022-v0.602.
Etienne Rojas: 2022-v0.602 - typo corregido en la intro del capítulo 3, apartado 3.1.2, apartado 3.2.4, typo en link apartado 6.4. Actualizado en versión 2022-v0.603.
Nicolás Román: 2022-v0.602 - typo corregido en apartado 3.1. Actualizado en versión 2022-v0.603.
Rodrigo Munita: 2022-v0.602 - typo corregido en capítulo 3. Actualizado en versión 2022-v0.603.
Ariane Carvajal: 2022-v0.602 - typo corregido en apartado 2.4, apartado 2.6, 4 typos en apartado 2.9, apéndice A.2, figura A.4, apartado 4.3, apartado 4.4.1, apartado 4.4.2, 2 typos en apartado 4.7.2, apartado 3.4.4, y apartado 6.4.5. Actualizado en versión 2022-v0.603.
Diego Varela: 2022-v0.602 - typo corregido en apéndice A.1. Actualizado en versión 2022-v0.603.
Ricardo Lorca: 2022-v0.602 - typo corregido en apéndice A.4, apartado 5.2, apartado 5.6, apartado 7.1, capítulo 9, apartado 9.2.2. Actualizado en versión 2022-v0.603.
Victor Morales: 2022-v0.602 - typo corregido en apartado 4.7.6. Actualizado en versión 2022-v0.603.
Bastián Soto Iturra: 2022-v0.602 - construcción en tikz de la figura 5.8. Actualizado en versión 2022-v0.603.
Prof. Claudio Torres: 2022-v0.602 - se agrega apartado 7.5, se cambia título de apéndice A.2. Actualizado en versión 2022-v0.603.
Rodrigo Munita: 2022-v0.603 - typo corregido en capítulo 10. Actualizado en versión 2022-v0.604.
Etienne Rojas: 2022-v0.604 - typo corregido en apartado 7.5, algoritmo 10; apartado 8.6; apartado 8.7. Actualizado en versión 2022-v0.605.
Víctor Morales: 2022-v0.604 - typo corregido en sección apartado 8.7. Actualizado en versión 2022-v0.605.
Pablo Estobar: 2022-v0.604 - typo corregido en apartado 5.2. Actualizado en versión 2022-v0.605.
Ariane Carvajal: 2022-v0.604 - typo corregido en Intro de capítulo 5. Actualizado en versión 2022-v0.605.
Carlos Verdugo: 2022-v0.604 - typo corregido en apartado 7.5. Actualizado en versión 2022-v0.605.
Prof. Claudio Torres: 2022-v0.604 - modificada etiqueta de algoritmo 10 y se agrega apartado 7.6. Actualizado en versión 2022-v0.605.

Tabla 10.17: Correctores Apuntes INF-285 - 2022-2.

Autor/a
Prof. Claudio Torres: 2022-v0.605 - corrigiendo uso de \verb. Actualización del Prefacio . Modificación pequeña en apartado 1.3.7 y apartado 2.9 . Actualizado en versión 2023-v0.606. Agregando ítem en apartado 2.10 . Mejorando visualización de fórmula en apartado 1.5.1.1 . Se corrige Prefacio en el índice general. Se corrige typo en [1] . Se actualiza portada.
Prof. Roberto León: 2022-v0.605 - typo corregido en Prefacio . Actualizado en versión 2023-v0.606.
Camilo Vera: 2023-v0.606 - typo corregido en apartado 1.3.6 . Actualizado en versión 2022-v0.607.
Nicolás Tapia: 2023-v0.606 - typo corregido en apartado 2.5 . Actualizado en versión 2023-v0.607.
Ariane Carvajal: 2023-v0.606 - tres typos corregidos en Introducción de capítulo 3 . Dos typos corregido en apartado 3.1 . Typo corregido en apartado 3.1.1 . Sugerencia incluida en apartado 5.2 . Typo corregido en Aseveración 1, apartado 4.4 . Typo corregido en apartado 5.1 . Typo en introducción de capítulo 6 . Typo en apartado 6.3 . Actualizado en versión 2023-v0.607.
Raúl Cuello: 2023-v0.606 - typo corregido en apartado 2.2.1 . Typo corregido en apartado 6.6.2 . Actualizado en versión 2023-v0.607.
Sofía Riquelme: 2023-v0.606 - typo corregido en apartado 3.2.1 . Actualizado en versión 2023-v0.607.
Javier Pérez: 2023-v0.606 - sugerencia incluida en apartado 4.4 . Actualizado en versión 2023-v0.607.
Francisco Manríquez (Ayudante): 2023-v0.606 - typo corregido en apéndice B.7 . Typo corregido en apartado 6.6.2 , ahora se incluye una referencias a los reflectores de Householder y se aclara que la factorización de Gram-Schmidt no es el único algoritmo disponible para construir la factorización QR de una matriz. Se sugiere profundizar la relación entre la iteracion de Arnoldi y Gram-Schmidt modificado, se incluye en segundo cuadro "¡MUY IMPORTANTE!".en apartado 7.2 . Actualizado en versión 2023-v0.607.
Rodrigo Munita: 2023-v0.606 - typo/sugerencia corregido/incluido en apartado 4.1.2 , apartado 4.5 . Actualizado en versión 2023-v0.607.
Martín Ruiz Hernández: 2023-v0.606 - variante de la sugerencia incluida en figura 5.7 . Actualizado en versión 2023-v0.607.
Prof. Claudio Torres: 2023-v0.606 - actualización de apartado 6.3 . Se modifica segundo cuadro "¡MUY IMPORTANTE!".en apartado 7.2 . Actualización de diferenciales usados en integrales definidas, antes se usaba ds ahora ds en apartado 9.1 . Se agrega apartado 9.1.7 . Se agregaron figura 9.2 , figura 9.3 , y figura 9.4 . Actualizado en versión 2023-v0.607.
Prof. Claudio Torres: 2023-v0.607 - versión correcta publicada. Actualizado en versión 2023-v0.608.
Javier Pérez: 2023-v0.608 - sugerencia agregada en apartado 8.6 . Actualizado en versión 2023-v0.609.
Ariane Carvajal: 2023-v0.608 - Dos typos corregidos en apartado 9.2.1 , dos typos corregidos en apartado 7.5 y un typo corregido en apartado 7.2 . Actualizado en versión 2023-v0.609.

Tabla 10.18: Correctores Apuntes INF-285 - 2023-1.

Autor/a
Prof. Claudio Torres: 2022-v0.609 - se actualiza presentación en capítulo 6. Actualizado en versión 2023-v0.610.
Prof. Claudio Torres: 2022-v0.610 - se actualiza apartado 7.2, figura 7.1 y ecuación (7.22) para consistencia con nueva presentación en capítulo 6. Actualizado en versión 2023-v0.611.
Prof. Claudio Torres: 2023-v0.611 - typo corregido en apartado 8.3, se modificó \hat{c} por \widehat{c} en apartado 8.3, typo corregido en apartado 8.6, se reemplaza <i>hat</i> por <i>widehat</i> en apartado 9.1.3, se corrige typo en apartado 9.1.6. Actualizado en versión 2023-v0.612.
Diego Aliaga: 2023-v0.611 - 2 typos corregidos en apartado 9.1.7. Actualizado en versión 2023-v0.612.

Tabla 10.19: Correctores Apuntes INF-285 - 2023-2.

Autor/a
Prof. Claudio Torres: 2023-v0.612 - Actualización del Prefacio y typo en apartado 2.1. Actualizado en versión 2024-v0.613.

Tabla 10.20: Correctores Apuntes INF-285 - 2024-1.

Apéndice A

SVD: Descomposición en Valores Singulares

La **descomposición en valores singulares** (del inglés SVD: Singular Value Decomposition) es una factorización matricial que siempre existe. La SVD factoriza cualquier matriz A en el producto $U\Sigma V^*$, donde U y V son matrices unitarias y Σ es una matriz diagonal con coeficientes mayores o iguales a 0. Una aplicación importante de la SVD es que nos permite caracterizar de forma explícita distintas características de una matriz, como son el *column-space*(A), *row-space*(A), *null-space*(A), *null-space*(A^*), y sus valores singulares. En las siguientes secciones analizaremos en más detalle estos puntos.

A.1. Introducción

Antes de analizar la descomposición en valores singulares, SVD, se presenta un ejemplo para recordar que es lo que significa multiplicar por una matriz y como se puede analizar. En las Figuras A.1 y A.2 muestra lo que le ocurre a los vectores en el círculo unitario, es decir $\|\mathbf{x}\| = 1$ para $\mathbf{x} \in \mathbb{R}^2$. Como se observa en las Figuras respectivas, se obtienen elipses en ambos casos. Este patrón no es algo particular de estas matrices, es en realidad un comportamiento esperado al multiplicar un conjunto de vectores por una matriz. Entonces la SVD tiene por objetivo caracterizar este comportamiento con la construcción de 3 matrices, U , Σ , y V .

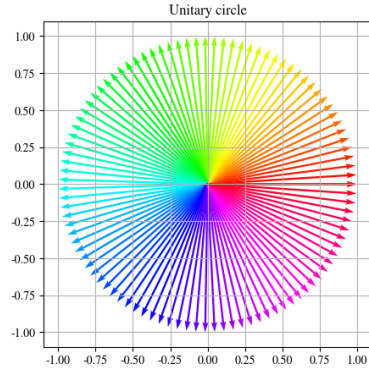
Para el caso donde $\mathbf{x} \in \mathbb{R}^n$, ya no se obtiene una elipse, lo que se obtiene es una “hiperelipse”! Esto se analizará en la siguiente sección. Se invita también a re-revisar el jupyter notebook “Bonus - 01 - Linear Transformation.ipynb” disponible en el repositorio de GitHub asociado.

A.2. Interpretación Geométrica

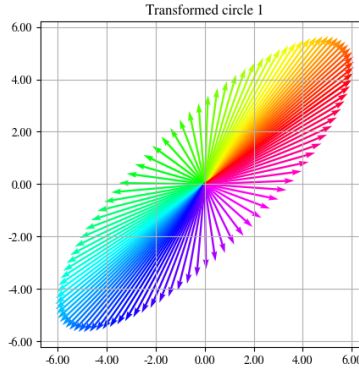
En la sección anterior se presentó que el conjunto de vectores unitarios¹ generan una elipse en \mathbb{R}^2 . Ahora en alta dimensión, por ejemplo si $A \in \mathbb{R}^{n \times n}$ entonces lo que se obtiene es una “hiperelipse”. Se puede definir una hiperelipse en \mathbb{R}^n como la superficie obtenida, “alargando” la esfera unitaria en \mathbb{R}^n por factores $\sigma_1, \sigma_2, \dots, \sigma_n$ (posiblemente nulos) en direcciones ortogonales $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n \in \mathbb{R}^n$. Por conveniencia, consideraremos los vectores \mathbf{u}_i unitarios, i.e., $\|\mathbf{u}_i\|_2 = 1$. Los vectores $\{\sigma_i \mathbf{u}_i\}$ son los *semiejes principales* de la hiperelipse, con longitud $\sigma_1, \sigma_2, \dots, \sigma_n$. Si A tiene rango r , exactamente r de las n longitudes son no nulas.

Esta interpretación geométrica de la SVD puede entenderse mejor en \mathbb{R}^2 , por ejemplo, si denominamos S al círculo unitario en \mathbb{R}^2 , entonces para la matriz A , podemos denotar como AS al conjunto de vectores unitarios

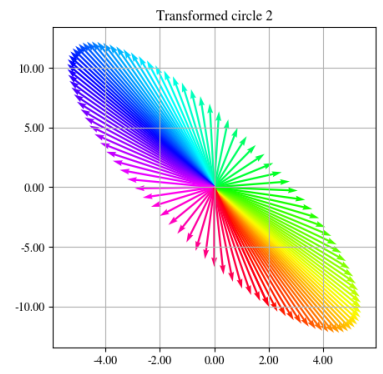
¹Considerando la norma 2



(a) Vectores en el círculo unitario: $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$.

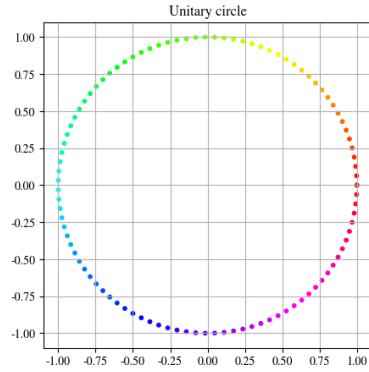


(b) Vectores del círculo unitario transformados por la matriz A_1 . La transformación lineal se expresa de la siguiente forma: $A_1 \mathbf{x} = \begin{bmatrix} 6 & 1 \\ 4 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

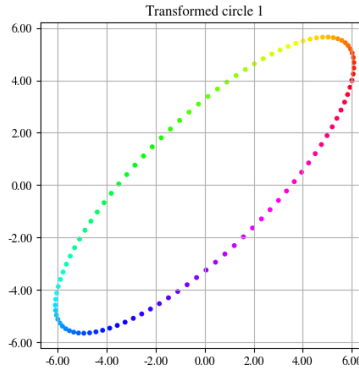


(c) Vectores del círculo unitario transformados por la matriz A_2 . La transformación lineal se expresa de la siguiente forma: $A_2 \mathbf{x} = \begin{bmatrix} 2 & 5 \\ -10 & -7 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

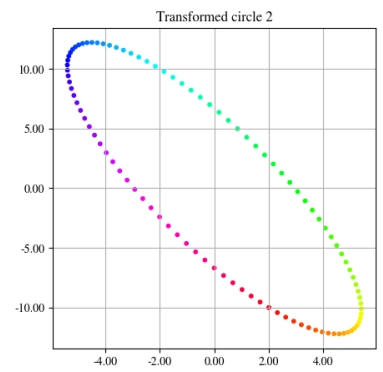
Figura A.1: Este ejemplo muestra que le ocurre a los vectores del círculo unitario al ser transformados por la matrices A_1 y A_2 . Se utilizan “vectores” de diferentes colores para notar que le ocurre a cada uno de ellos al ser transformados, es decir, el vector verde en las coordenadas transformadas corresponde al vector verde en la coordenada de la primera figura.



(a) Vectores en el círculo unitario: $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$.



(b) Vectores del círculo unitario transformados por la matriz A_1 . La transformación lineal se expresa de la siguiente forma: $A_1 \mathbf{x} = \begin{bmatrix} 6 & 1 \\ 4 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$



(c) Vectores del círculo unitario transformados por la matriz A_2 . La transformación lineal se expresa de la siguiente forma: $A_2 \mathbf{x} = \begin{bmatrix} 2 & 5 \\ -10 & -7 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

Figura A.2: Este ejemplo muestra lo mismo que en la Figura A.1, sin embargo los vectores por simplicidad se representan como puntos.

“transformados” por la matriz A . Para ser más específicos la siguiente identidad es válida:

$$AV = U\Sigma$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}.$$

En la Figura A.3 se presenta la interpretación geométrica en 2-D.

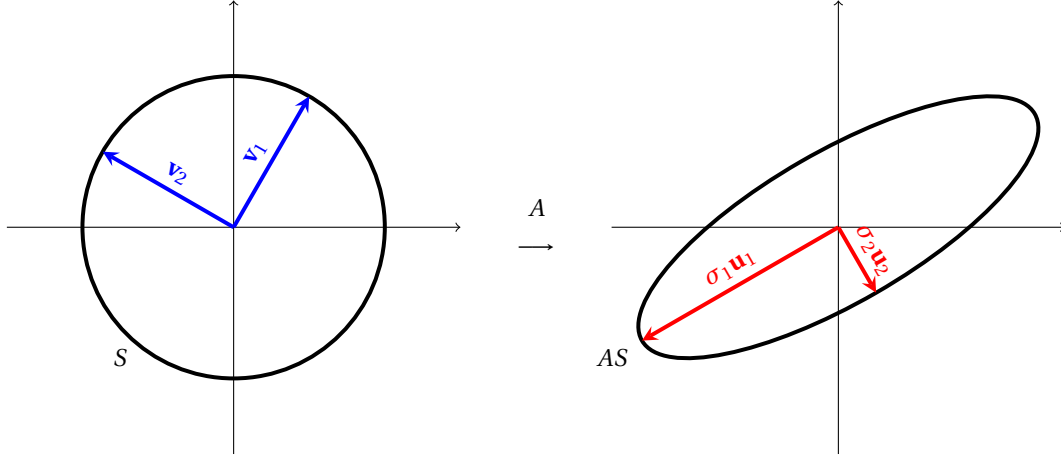


Figura A.3: Interpretación geométrica de la SVD de una matriz 2×2

A continuación se definen algunas propiedades de A en términos de la forma de AS , suponer que $A \in \mathbb{R}^{n \times n}$ y A **full rank**².

1. Los n **valores singulares** de A , son las longitudes de los n semiejes principales de AS , denominados $\sigma_1, \sigma_2, \dots, \sigma_n$. Se sigue la convención que los valores singulares son enumerados descendientemente: $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n > 0$.
2. Los n **vectores singulares izquierdos** de A , son los vectores unitarios $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n\}$ orientados en la dirección de los semiejes principales de AS .
3. Los n **vectores singulares derechos** de A , son los vectores unitarios $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\} \in S$, que son las pre-imágenes de los semiejes principales de AS , luego $A\mathbf{v}_j = \sigma_j \mathbf{u}_j$.

A.3. Computación de la SVD

La SVD de una matriz no requiere que la matriz sea cuadrada, como así lo requiere la descomposición de valores propios, por lo cual podemos considerar que $A \in \mathbb{R}^{m \times n}$. Sin embargo, la factorización de valores propios sí nos ayuda a obtener la SVD.

En la sección anterior se mencionó implícitamente la estructura de la SVD, por lo que ahora es necesario mencionarlo explícitamente,

$$A = \hat{U} \hat{\Sigma} V^*, \quad (\text{A.1})$$

en este caso hemos introducido la notación \hat{U} y $\hat{\Sigma}$ en vez de U y Σ simplemente, esto se explicará con más detalle en el apéndice A.4. Esta estructura nos permite obtener fácilmente A^* de la siguiente forma,

$$A^* = V \hat{\Sigma} \hat{U}^*.$$

²Es decir $\text{rank} = n$.

Por lo que si obtenemos $A^* A$ y por ortogonalidad de \mathbf{u}_i , el producto se reduce a,

$$\begin{aligned} A^* A &= V \hat{\Sigma} \hat{U}^* \hat{U} \hat{\Sigma} V^* \\ &= V \hat{\Sigma}^2 V^* \\ &= V \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2) V^*. \end{aligned}$$

El cual no depende de U y además nos entrega exactamente la factorización de valores propios de $A^* A$. Notar que $A^* A$ es simétrica y, por lo menos, semi-definida-positiva. Recuerde que la factorización de valores propios de una matriz cuadrada, digamos M , tiene la siguiente forma,

$$M = W \Lambda W^{-1},$$

o,

$$M W = W \Lambda.$$

En particular, en este caso sabemos que los vectores propios son ortonormales y los valores propios son mayores o igual a 0, se obtiene la siguiente relación considerando $M = A^* A$,

$$\begin{aligned} \mathbf{v}_i &= \mathbf{w}_i, \\ \sigma_i &= \sqrt{\lambda_i}. \end{aligned}$$

Entonces hemos sido capaces de obtener V y Σ , por lo tanto para obtener U solo necesitamos despejar U de la siguiente forma,

$$\begin{aligned} A &= \hat{U} \hat{\Sigma} V^*, \\ A V &= \hat{U} \hat{\Sigma}, \\ A V \hat{\Sigma}^{-1} &= \hat{U}, \\ A V \text{diag}(\sigma_1^{-1}, \sigma_2^{-1}, \dots, \sigma_n^{-1}) &= \hat{U}. \end{aligned}$$

Notar que en este último paso se requiere $\sigma_i > 0$, de otra forma se podría estar dividiendo por 0 y como saben, eso no es bueno!

Alternativamente uno puede obtener la SVD siguiendo el siguiente algoritmo³. Considere la siguiente matriz B ,

$$B = \begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix}$$

Esta matriz B es una matriz simétrica de $(m+n)$ filas y $(m+n)$ columnas, por lo cual podemos obtener sus valores propios y vectores propios sin nunca calcular $A^* A$. Además sabemos que los valores propios son reales y los vectores propios ortogonales!

Entonces, luego de obtener todos los valores y vectores propios, podemos hacer el siguiente análisis. Sea λ uno de los valores propios encontrados y $\mathbf{x} = [\mathbf{u}, \mathbf{v}]$ su vector propio asociado, entonces se obtiene la siguiente relación,

$$\begin{aligned} B \mathbf{x} &= \lambda \mathbf{x} \\ \begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} &= \lambda \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \\ \begin{bmatrix} A^* \mathbf{v} \\ A \mathbf{u} \end{bmatrix} &= \lambda \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \end{aligned}$$

³La idea en este caso es evitar la computación explícita de $A^* A$.

Del cual se obtienen las siguientes ecuaciones,

$$A^* \mathbf{v} = \lambda \mathbf{u}, \quad (\text{A.2})$$

y,

$$A \mathbf{u} = \lambda \mathbf{v} \quad (\text{A.3})$$

Al multiplicar por la izquierda la ecuación A.3 por A^* se obtiene,

$$A^* A \mathbf{u} = \lambda A^* \mathbf{v}$$

Ahora, utilizando la ecuación (A.2):

$$A^* A \mathbf{u} = \lambda^2 \mathbf{u},$$

donde \mathbf{u} es efectivamente un vector propio de $A^* A$, con su correspondiente valor propio λ^2 . Esto nos demuestra que podemos obtener los valores y vectores propios de $A^* A$ sin nunca calcular $A^* A$ explícitamente! Solo debemos obtener los valores y vectores propios de B . Luego de haber obtenido V y Σ , podemos recuperar \hat{U} de la misma forma que se obtuvo anteriormente.

A.4. SVD Reducida y SVD Completa

Hasta este momento hemos trabajado implícitamente considerando que la matriz A asociada a la transformación lineal respectiva es cuadrada, sin embargo este no es realmente un requerimiento explícito. Más aún, en el apéndice A.3 se consideró que la matriz $A \in \mathbb{R}^{m \times n}$. Entonces, la importancia recae para los casos donde la matriz A no es cuadrada, y más particularmente cuando tiene más filas que columnas. En el caso de más columnas que filas el análisis es similar, pero por simplicidad acá consideraremos principalmente que $m > n$, es decir el caso en que A tiene más filas que columnas. En estos casos es cuando aparece la diferencia entre la SVD reducida y la SVD completa⁴.

Antes de explicar la diferencia entre la SVD reducida y la SVD completa es bueno presentar un ejemplo numérico nuevamente. En la Figura A.4 se observa lo que ocurre al hacer una transformación lineal en el caso de que la matriz tenga 3 filas y 2 columnas, en particular la matriz es la siguiente,

$$A_3 = \begin{bmatrix} 6 & 1 \\ 4 & 4 \\ 8 & 4 \end{bmatrix}.$$

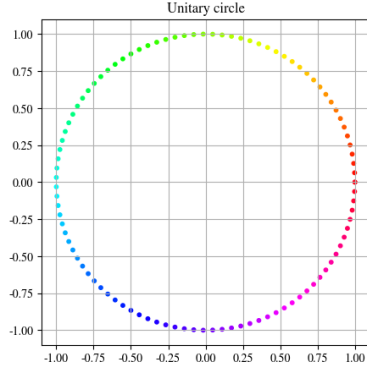
En la ecuación (A.1) se presentó la formulación matricial de la SVD, la cual era la SVD reducida. También podemos expresar esta factorización por medio de la relación entre los vectores singulares izquierdos, \mathbf{u}_j , y vectores singulares derechos, \mathbf{v}_j , de la siguiente forma⁵,

$$A \mathbf{v}_j = \sigma_j \mathbf{u}_j, \quad 1 \leq j \leq n. \quad (\text{A.4})$$

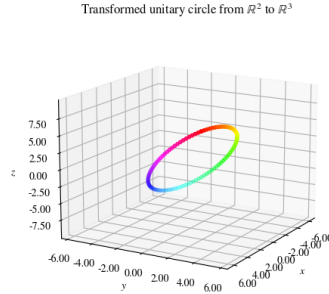
En particular, podemos reconstruir la relación matricialmente desde las expresiones anteriores de la siguiente for-

⁴En general uno trabaja con la SVD reducida para reducir los requerimientos de memoria asociados, así que tenga mucho cuidado cuando quiera la SVD y asegúrese que sea la SVD reducida!

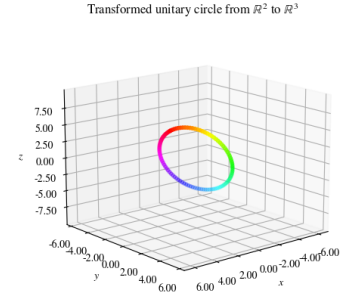
⁵Notar que no utilizaremos A_3 en este análisis porque esa matriz corresponde al ejemplo particular presentado en la Figura A.4, sin embargo este análisis es para matrices en $\mathbb{R}^{m \times n}$.



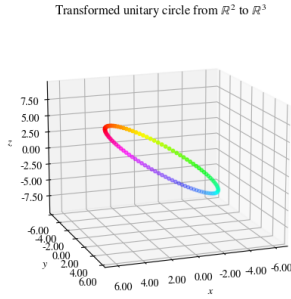
(a) Vectores en el círculo unitario: $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$.



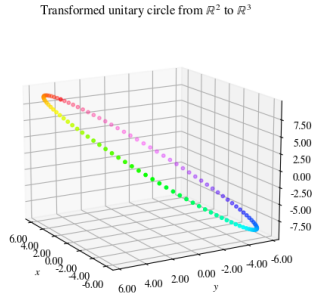
(b) Vectores del círculo unitario transformados por la matriz A_3 . Se utilizan los parámetros de visualización elevación= 15 y ángulo azimut= 30.



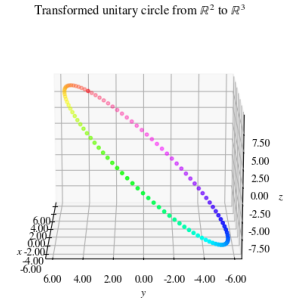
(c) Vectores del círculo unitario transformados por la matriz A_3 . Se utilizan los parámetros de visualización elevación= 15 y ángulo azimut= 50.



(d) Vectores del círculo unitario transformados por la matriz A_3 . Se utilizan los parámetros de visualización elevación= 15 y ángulo azimut= 70.



(e) Vectores del círculo unitario transformados por la matriz A_3 . Se utilizan los parámetros de visualización elevación= 15 y ángulo azimut= 150.



(f) Vectores del círculo unitario transformados por la matriz A_3 . Se utilizan los parámetros de visualización elevación= 15 y ángulo azimut= 150.

Figura A.4: Este ejemplo muestra que le ocurre a los vectores del círculo unitario al ser transformados por la matriz

$A_3 = \begin{bmatrix} 6 & 1 \\ 4 & 4 \\ 8 & 4 \end{bmatrix}$. La transformación lineal se expresa de la siguiente forma: $A_3 \mathbf{x} = \begin{bmatrix} 6 & 1 \\ 4 & 4 \\ 8 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$. Al igual que en las

figuras anteriores, los colores son utilizados para denotar que ocurre con cada vector unitario al ser transformado por la matriz A_3 . En este caso la transformación lineal va de \mathbb{R}^2 a \mathbb{R}^3 . Notar que si bien el espacio de llegada es \mathbb{R}^3 , aún se forma una elipse y no un elipsoide debido a que el espacio de salida es \mathbb{R}^2 .

ma, es decir, columna a columna,

$$\begin{aligned} \begin{bmatrix} A\mathbf{v}_1 & A\mathbf{v}_2 & \dots & A\mathbf{v}_n \end{bmatrix} &= \begin{bmatrix} \sigma_1 \mathbf{u}_1 & \sigma_2 \mathbf{u}_2 & \dots & \sigma_n \mathbf{u}_n \end{bmatrix} \\ \begin{bmatrix} A \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}}_{V \in \mathbb{R}^{n \times n}} &= \underbrace{\begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_n \end{bmatrix}}_{\hat{U} \in \mathbb{R}^{m \times n}} \underbrace{\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix}}_{\hat{\Sigma} \in \mathbb{R}^{n \times n}} \\ AV &= \hat{U}\hat{\Sigma}. \end{aligned} \tag{A.5}$$

donde $\hat{\Sigma}$ es una matriz diagonal de dimensión $n \times n$ con elementos positivos (asumiendo que A es full rank), \hat{U} es una matriz de dimensión $m \times n$, con columnas ortonormales, V es una matriz de dimensión $n \times n$ con columnas ortonormales y V es unitaria, por lo tanto $V^{-1} = V^*$, se obtiene la SVD reducida,

$$A = \hat{U} \hat{\Sigma} V^*. \quad (\text{A.6})$$

Esta relación es en general la relación que uno construye para distintas aplicaciones, sin embargo no cumple con lo mencionado al principio de este anexo. En particular se mencionó que U y V debían ser matrices unitarias, en este caso solo V sería unitaria. Notar que U no es unitaria porque no es una matriz cuadrada, es solo una matriz rectangular de dimensión $m \times n$. Entonces surge la pregunta, ¿cómo se obtiene U a partir de \hat{U} ?

Para responder la pregunta solo debemos notar que \hat{U} tiene efectivamente n vectores ortonormales en \mathbb{R}^m , por lo cual solo necesitamos obtener $m - n$ vectores adicionales ortonormales a los vectores columnas almacenados en \hat{U} y también ortonormales entre sí. Es decir, podemos construir U de la siguiente forma,

$$\begin{aligned} U &= [\hat{U}, \quad \hat{U}^\perp] \\ &= [\hat{U}, \quad \mathbf{u}_{n+1}, \dots, \mathbf{u}_m], \end{aligned}$$

donde $\hat{U}^\perp = [\mathbf{u}_{n+1}, \dots, \mathbf{u}_m]$ y $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = 0$ para $i \neq j$, y 1 en todo otro caso. Debemos notar que si reemplazamos \hat{U} por U en la ecuación (A.6) vamos a tener una incompatibilidad al querer hacer el producto entre U y $\hat{\Sigma}$. ¿Qué podemos hacer acá? Es importante destacar que no solo por compatibilidad debemos modificar $\hat{\Sigma}$, sino que también debemos asegurar que la matriz que se obtiene al hacer el producto sigue siendo A , entonces consideremos que extendemos la matriz $\hat{\Sigma}$ con la matriz X de la siguiente forma,

$$\begin{aligned} A &= \hat{U} \hat{\Sigma} V^* \\ &= [\hat{U}, \quad \hat{U}^\perp] \begin{bmatrix} \hat{\Sigma} \\ X \end{bmatrix} V^* \\ &= [\hat{U} \hat{\Sigma}, \quad \hat{U}^\perp X] V^* \\ &= \underbrace{\hat{U} \hat{\Sigma} V^*}_A + \hat{U}^\perp X V^* \\ &= A + \hat{U}^\perp X V^*. \end{aligned}$$

Por lo tanto podemos concluir que X debe ser la matriz nula, es decir, una matriz de puros 0 de dimensión $(m - n) \times n$, entonces,

$$\begin{aligned} A &= [\hat{U}, \quad \hat{U}^\perp] \begin{bmatrix} \hat{\Sigma} \\ \underline{0} \end{bmatrix} V^* \\ &= U \Sigma V^*. \end{aligned} \quad (\text{A.7})$$

Entonces hemos podido construir la factorización SVD reducida y SVD completa haciendo notar que ambas generan la misma matriz A , sin embargo la SVD completa sí genera 2 matrices unitarias y una matriz diagonal pero no cuadrada.

A.4.1. Descomposición de Valores Propios

En el apéndice A.3 se mencionó la descomposición de valores propios y su relación para la computación de la SVD. En esta sección presentaremos brevemente la similitud entre la SVD y la descomposición de valores propios. La descomposición de valores propios tiene la siguiente forma matricial,

$$A W = W \Lambda. \quad (\text{A.8})$$

Similar a la SVD, también podemos escribirla componente a componente de la siguiente forma,

$$\begin{bmatrix} A \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_n \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}.$$

Si hacemos una comparación columna a columna obtenemos,

$$A \mathbf{w}_i = \lambda_i \mathbf{w}_i.$$

Notar que los vectores propios \mathbf{w}_i no son necesariamente ortonormales, por lo que obtenemos la siguiente relación si W es no singular,

$$A = W \Lambda W^{-1}. \quad (\text{A.9})$$

Generalmente, ambas descomposiciones son distintas, aunque podría darse el caso de que sean iguales⁶. Otro aspecto a considerar es que toda matriz tiene una descomposición en valores singulares, pero no todas las matrices tienen una descomposición en valores propios.

A.5. Algunas propiedades importantes de la SVD

Consideremos $A \in \mathbb{R}^{m \times n}$, $p = \min\{m, n\}$, y $r \leq p$ denota el número de valores singulares no nulos.

Thm 21. *El rango de A es r , i.e. la cantidad de valores singulares no nulos.*

Demostración. Como U y V^* son matrices unitarias, $\text{Range}\{A\} = \text{Range}\{\Sigma\}$, donde la última corresponde al número de valores diagonales distintos de cero. \square

Thm 22. $\text{Range}\{A\} = \text{span}\langle \mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r \rangle$ y $\text{Null}\{A\} = \text{span}\langle \mathbf{v}_{r+1}, \dots, \mathbf{v}_n \rangle$.

Demostración. Ya que r es el número de valores singulares no nulos. En general, se cumple lo siguiente para la matriz A :

$$\begin{array}{c} \overbrace{\begin{bmatrix} | & | & | & | & | & | \\ \mathbf{u}_1 & \dots & \mathbf{u}_r & \mathbf{u}_{r+1} & \dots & \mathbf{u}_m \\ | & | & | & | & | & | \end{bmatrix}}^{\text{col}\{A\}} \quad \overbrace{\begin{bmatrix} | & | & | & | & | & | \\ \mathbf{u}_{r+1} & \dots & \mathbf{u}_m & \mathbf{u}_{r+1} & \dots & \mathbf{u}_m \\ | & | & | & | & | & | \end{bmatrix}}^{\text{null}\{A^*\}} \end{array} \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \begin{array}{c} \left[\begin{array}{c} \mathbf{v}_1^* \\ \vdots \\ \mathbf{v}_r^* \\ \mathbf{v}_{r+1}^* \\ \vdots \\ \mathbf{v}_n^* \end{array} \right] \left. \begin{array}{l} \text{row}\{A\} \\ \text{null}\{A\} \end{array} \right\} \end{array}$$

\square

Thm 23. $\|A\|_2 = \sigma_1$ y $\|A\|_F = \sqrt{\sigma_1^2 + \dots + \sigma_r^2}$, donde $_F$ denota la norma de Frobenious.

⁶Por ejemplo si A es simétrica y definida positiva.

Demostración. La norma 2 es invariante frente a la multiplicación de matrices unitarias, por lo tanto:

$$\|A\|_2 = \|U\Sigma V^*\|_2 = \|\Sigma\|_2 = \max_{1 \leq i \leq m} |\Sigma_{i,i}| = \sigma_1$$

$$\|A\|_F = (\text{traza}(A^* A))^{1/2} = \|\Sigma\|_F = \sqrt{\sigma_1^2 + \dots + \sigma_r^2}$$

□

Thm 24. Los valores singulares no nulos de A , son la raíz cuadrada de los valores propios no nulos de $A^* A$ o $A A^*$ (Estas matrices tienen los mismos valores propios no nulos).

Demostración. $A^* A$ es una matriz cuadrada, simétrica y semidefinida positiva que acepta una diagonalización del tipo $W \Lambda W^{-1}$, con Λ la matriz diagonal de los valores propios y W la matriz de vectores propios. Además:

$$A^* A = W \Lambda W^{-1} = (U\Sigma V^*)^* (U\Sigma V^*) = V\Sigma^* U^* U\Sigma V^* = V\Sigma^* \Sigma V^*$$

Se cumple que:

$$\begin{aligned} \Lambda &= \Sigma \Sigma^* \\ \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} &= \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \\ \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} &= \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_n^2 \end{bmatrix}, \end{aligned}$$

donde $\lambda_i = \sigma_i^2$. Este procedimiento es análogo con la matriz $A A^*$.

□

Thm 25. A es la suma de r matrices de rango 1

$$A = \sum_{k=1}^r \sigma_k \mathbf{u}_k \mathbf{v}_k^*,$$

donde r es el rango de A y \mathbf{u}_i y \mathbf{v}_i son las i -ésimas columnas de las matrices U y V , respectivamente.

Demostración.

$$\begin{aligned}
 A &= U \Sigma V^* \\
 &= U \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} V^* \\
 &= U \left(\begin{bmatrix} \sigma_1 & & \\ & & \\ & & \end{bmatrix} + \begin{bmatrix} & \sigma_2 & \\ & & \\ & & \end{bmatrix} + \dots + \begin{bmatrix} & & \sigma_r \end{bmatrix} \right) V^* \\
 &= \sigma_1 \mathbf{u}_1 \mathbf{v}_1^* + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^* + \dots + \sigma_r \mathbf{u}_r \mathbf{v}_r^*
 \end{aligned}$$

□

A.6. Ejemplo numérico de la computación de la SVD

Considerar la siguiente matriz,

$$A = \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

La matriz A no tiene grandes dimensiones y está bien condicionada. Por lo tanto se seguirá el primer método explicado para obtener los valores singulares. En primer lugar, se calcula el producto $A^* A$,

$$\begin{aligned}
 A^* A &= \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}.
 \end{aligned}$$

A continuación, se determinan los valores propios de la matriz $A^* A$, resultando $\lambda_1 = \sigma_1^2 = 4$ y $\lambda_2 = \sigma_2^2 = 2$. Por lo tanto, la matriz $\hat{\Sigma}$ está dada por,

$$\begin{aligned}
 \hat{\Sigma} &= \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \\
 &= \begin{bmatrix} 2 & 0 \\ 0 & \sqrt{2} \end{bmatrix}
 \end{aligned}$$

Luego se calculan los vectores propios de \mathbf{v}_1 y \mathbf{v}_2 , es decir, los vectores propios asociados a λ_1 y λ_2 ,

$$\begin{aligned}
 \tilde{\mathbf{v}}_1 &= \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \\
 \tilde{\mathbf{v}}_2 &= \begin{bmatrix} 1 \\ -1 \end{bmatrix}.
 \end{aligned}$$

Sin embargo, debido a que la matriz V presenta columnas ortonormales, cada vector propio debe ser normalizado tal que tenga norma 2 igual a 1.

$$\mathbf{v}_1 = \frac{\tilde{\mathbf{v}}_1}{\|\tilde{\mathbf{v}}_1\|_2} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{2}{\sqrt{2}} \\ \frac{\sqrt{2}}{2} \end{bmatrix}$$

$$\mathbf{v}_2 = \frac{\tilde{\mathbf{v}}_2}{\|\tilde{\mathbf{v}}_2\|_2} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{2}{-\sqrt{2}} \\ \frac{\sqrt{2}}{2} \end{bmatrix}$$

Por lo tanto, cada vector normalizado corresponde a una columna para la matriz V ,

$$V = [\mathbf{v}_1, \mathbf{v}_2] = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{2}{\sqrt{2}} & \frac{2}{-\sqrt{2}} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}.$$

Ya que se tienen las matrices $\hat{\Sigma}$ y V , se obtiene la matriz \hat{U} realizando el producto $AV\hat{\Sigma}^{-1} = \hat{U}$,

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{2}{\sqrt{2}} & \frac{-\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix} = \hat{U}$$

$$= \begin{bmatrix} 0 & 1 \\ \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & 0 \end{bmatrix}.$$

Lo cual nos entrega la *descomposición en valores singulares reducida* de la matriz A ,

$$A = \hat{U} \hat{\Sigma} V^*$$

$$= \begin{bmatrix} 0 & 1 \\ \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{2}{\sqrt{2}} & \frac{-\sqrt{2}}{2} \end{bmatrix}.$$

Para obtener la *descomposición en valores singulares completa* de A , se debe agregar un vector columna ortonormal a los vectores columnas en la matriz \hat{U} y un correspondiente vector fila nulo en la matriz $\hat{\Sigma}$, entonces,

$$A = U \Sigma V^*$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ \frac{\sqrt{2}}{2} & 0 & \frac{-\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{2}{\sqrt{2}} & \frac{-\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

Apéndice B

PCA: Análisis de Componentes Principales

El análisis de componentes principales o *Principal Component Analysis* (PCA del inglés) es una técnica utilizada para la reducción de dimensionalidad de conjuntos de datos. El objetivo es encontrar una base conveniente para expresar el conjunto de datos manteniendo sus características o incluso reproduciendo exactamente los mismos datos.

B.1. Introducción

El análisis de componentes principales corresponde a una manipulación algebraica de los datos, en particular se puede entender como la re-escritura de los datos pero tomando ventaja de una posible dependencia lineal existente. Para ejemplificar este análisis, considere los datos presentados en la Figura B.1. Estos datos corresponden a puntos en \mathbb{R}^3 sin embargo generan una superficie bidimensional, en particular se puede entender que son puntos sobre un plano. Este es el punto principal de PCA, es decir, uno puede tener datos de alta dimensión, pero puede representarlos algebraicamente con un espacio de dimensión inferior. En este ejemplo uno podría re-escribir *cada punto de la data*¹ como la combinación lineal de 2 vectores en \mathbb{R}^3 . Esto significa que uno no necesita realmente 3 vectores (dado que están en \mathbb{R}^3) para representar los datos.

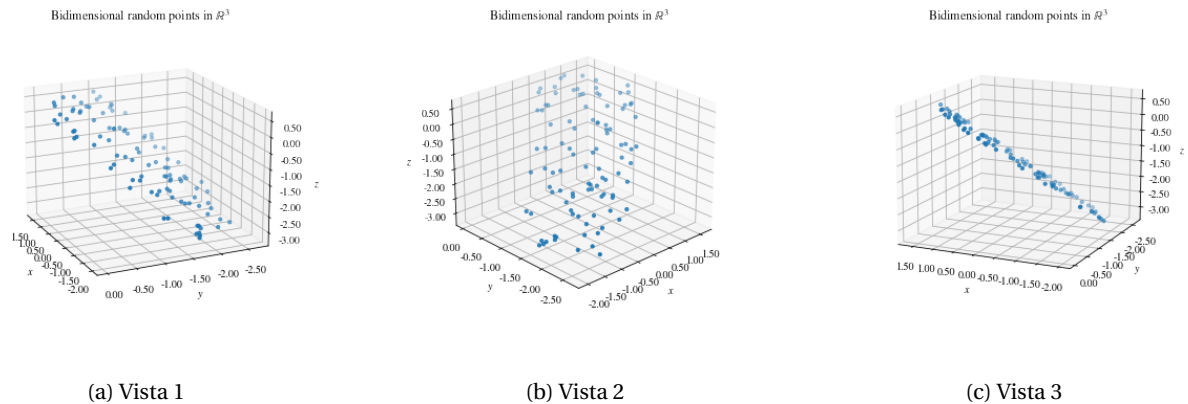


Figura B.1: Datos en \mathbb{R}^3 pero que son bidimensionales.

¹Notar que acá se destaca que nos estamos refiriendo a cada punto de la data, no a cualquier punto.

En la Figura B.2 se muestran los puntos en \mathbb{R}^2 utilizados para generar los datos en \mathbb{R}^3 . Los puntos en $\mathbf{x} \in \mathbb{R}^3$ se generaron de la siguiente forma $\mathbf{x}_i = a_i \mathbf{q}_1 + b_i \mathbf{q}_2$, donde $\mathbf{q}_1 \in \mathbb{R}^3$, $\mathbf{q}_2 \in \mathbb{R}^3$, y son ortonormales entre si.

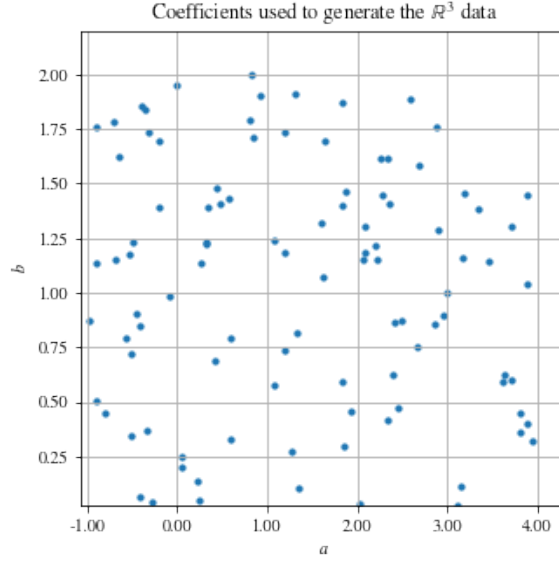


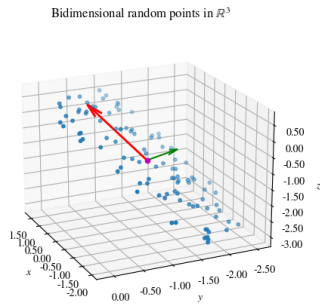
Figura B.2: Data 2D usada para generar los puntos en \mathbb{R}^3 , donde cada punto en $\mathbf{x} \in \mathbb{R}^3$ se generó de la siguiente forma $\mathbf{x}_i = a_i \mathbf{q}_1 + b_i \mathbf{q}_2$.

Ahora, si consideramos que los puntos en \mathbb{R}^3 los organizamos de la siguiente forma:

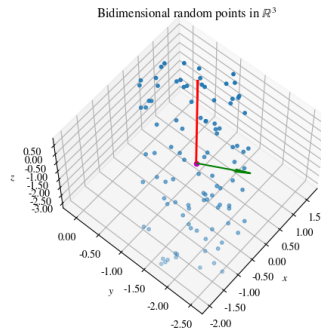
$$X = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix}.$$

Entonces la pregunta que nos podríamos hacer es la siguiente: ¿Cómo podemos determinar si podemos representar los datos almacenados en la matriz X de una forma más compacta? o, equivalentemente, ¿Cómo podemos reducir la dimensión del conjunto de datos almacenados en X ? La respuesta es PCA!² Pero, ¿Qué hace PCA? Para responder esta pregunta se presenta la Figura B.3. En esta figura se observan la base obtenida por PCA en rojo, verde y negro, además se incluye en magenta la media de los datos. El objetivo de incluir la media de los datos es porque ese es específicamente el punto donde se centra el sistema de coordenadas generados por PCA. En las siguientes secciones se formalizarán algebraicamente los conceptos presentados y la relación de PCA con la SVD.

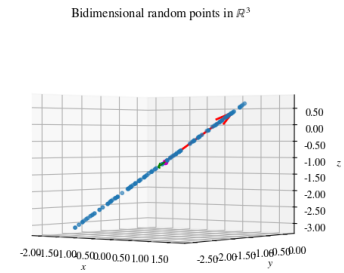
²Pero no es la única! Sin embargo, es importante entender PCA profundamente antes de analizar otros algoritmos y/o variantes.



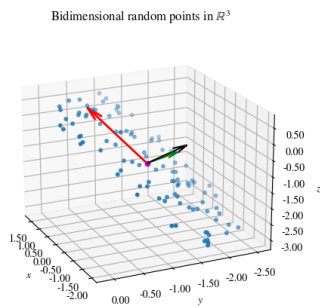
(a) Vista 1 mostrando 2 componentes principales



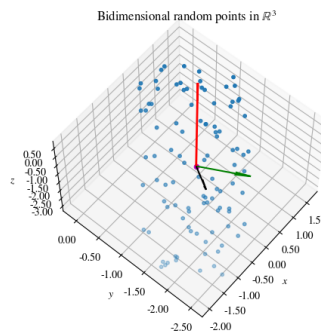
(b) Vista 2 mostrando 2 componentes principales



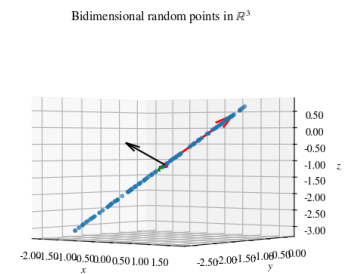
(c) Vista 3 mostrando 2 componentes principales



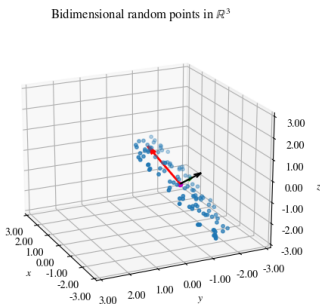
(d) Vista 1 mostrando toda la base



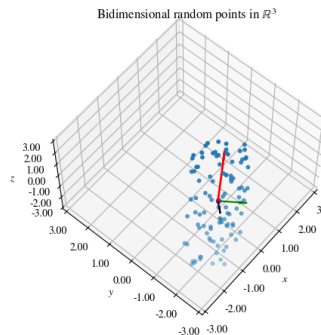
(e) Vista 2 mostrando toda la base



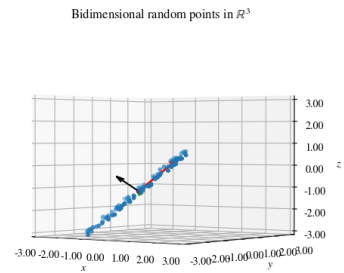
(f) Vista 3 mostrando toda la base



(g) Vista 1 mostrando toda la base con ejes en el mismo rango de valores



(h) Vista 2 mostrando toda la base con ejes en el mismo rango de valores



(i) Vista 3 mostrando toda la base con ejes en el mismo rango de valores

Figura B.3: Componentes principales obtenidas por PCA. En rojo se presenta la componente más significativa y en verde la segunda más significativa. La tercera componente se incluye en color negro, en este caso se observa que no hay variación en esa componente. En color magenta se incluye la media de los datos. Notar que los vectores pueden parecer no ortonormales por artefactos de visualización, por ejemplo en la segunda fila de gráficos, pero sí lo son. La tercera fila se agregó para minimizar este efecto.

B.2. Algunos conceptos preliminares

¡MUY IMPORTANTE!

En esta sección analizaremos algebraicamente el caso unidimensional y en la siguiente pasaremos de inmediato al caso multidimensional, por lo cual es importante entender la diferencia en este momento. Por ejemplo considere que tenemos un conjunto de la siguiente forma,

$$X = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix}.$$

El cual también podría ser escrito de la siguiente forma,

$$X = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix},$$

donde $\mathbf{x}_i = [x_i, y_i, z_i]^T$ ^a. Entonces aquí aparece una posible dualidad de interpretación que hay que tener cuidado. Por ejemplo uno puede definir como $\mathbf{x} = [x_1, x_2, \dots, x_n]$, $\mathbf{y} = [y_1, y_2, \dots, y_n]$, y $\mathbf{z} = [z_1, z_2, \dots, z_n]$, entonces ¿Cómo se puede interpretar \mathbf{x}_i ? La respuesta es simple^b, hay que tener cuidado en las definiciones que se incluyen justo antes de su uso. Sería equivalente a re-definir una variable en Python. De todos modos se explicitará en cada caso la definición que se usará respectivamente, pero es importante que usted esté atent@ de todos modos!

^aNotar que usamos el operador transpuesta porque estamos considerando que los vectores son vectores columna.

^bEs decir la respuesta es **depende**!

Considere que tenemos un conjunto de datos o mediciones $\{x_1, x_2, x_3, \dots, x_n\}$. Entonces, podemos definir su media \bar{x} ³ de la siguiente forma,

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i. \quad (\text{B.1})$$

Luego de obtener la media \bar{x} , podemos obtener la varianza muestral de la siguiente forma,

$$\text{var}([x_1, x_2, \dots, x_n]) = S^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2,$$

donde se utiliza el factor $\frac{1}{n-1}$ en lugar de $\frac{1}{n}$ para eliminar el sesgo. Al considerar un segundo conjunto de n datos y_i , se define la covarianza, que mide el grado de relación lineal entre dos variables de la siguiente forma,

$$\text{cov}([x_1, x_2, \dots, x_n], [y_1, y_2, \dots, y_n]) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \quad (\text{B.2})$$

Al convertir los datos x_i e y_i en vectores de n componentes, i.e. $\mathbf{x} = [x_1, x_2, \dots, x_n]$ y $\mathbf{y} = [y_1, y_2, \dots, y_n]$, se puede

³También se acostumbra a utilizar μ para la media, sin embargo en esta sección, por simplicidad, utilizaremos \bar{x} porque de otra forma tendríamos que agregar un sub-índice para diferenciar entre \bar{x} y \bar{y} . En la siguiente sección sí se utilizará μ pero de forma vectorial, i.e. $\boldsymbol{\mu}$, porque trabajaremos con n variables.

definir la covarianza (B.2) realizando un producto interno conveniente,

$$\text{cov}(\mathbf{x}, \mathbf{y}) = \frac{1}{n-1} \langle \mathbf{x} - \bar{\mathbf{x}}, \mathbf{y} - \bar{\mathbf{y}} \rangle \quad (\text{B.3})$$

Notar que en este caso la operación $\mathbf{x} - \bar{\mathbf{x}}$ corresponde a hacer la resta *element-wise* entre cada elemento x_i de \mathbf{x} con \bar{x} . En caso de que se obtenga la covarianza de un conjunto de datos consigo mismo se recupera la varianza muestral del conjunto original, es decir se obtiene la siguiente identidad: $\text{cov}(\mathbf{x}, \mathbf{x}) = \text{var}(\mathbf{x})$.

B.3. PCA en 5 pasos, paso 1: centrar los datos

Finalmente llegamos a la sección donde construiremos las *componentes principales*! Para la realización de esta tarea considere que tenemos m puntos en \mathbb{R}^n , es decir, $\mathbf{x}_i \in \mathbb{R}^n$ para $i \in \{1, 2, 3, \dots, m\}$. Los cuales almacenaremos en la matriz de datos X de la siguiente manera,

$$X = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_{m,1} & x_{m,2} & x_{m,3} & \dots & x_{m,n} \end{bmatrix}.$$

Notar que en este formato cada columna representa alguna característica propia de los datos. Por ejemplo si consideramos que cada dato corresponde a las características de un objeto geométrico, una característica podría ser su área superficial y otra su volumen, y así sucesivamente.

El siguiente paso necesario antes de obtener las componentes principales es *centrar* los datos. Esto significa obtener la media de cada columna de la matriz X y luego restarla de cada columna. Entonces, podemos obtener la media de cada columna simplemente sumando los elementos de cada columna y luego dividiendo por la cantidad de datos, en este caso la cantidad de datos es m , entonces obtenemos la media de la j -ésima columna de X , de la misma forma que se obtuvo antes en la ecuación (B.1),

$$\mu_j = \frac{1}{m} \sum_{k=1}^m x_{k,j}, \quad j \in \{1, 2, 3, \dots, n\},$$

y ahora podemos almacenar las medias en el vector de medias $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_n]^T$. Ahora ya que conocemos el vector de medias, podemos centrar los datos de la siguiente forma,

$$Z = X - \mathbf{1} \boldsymbol{\mu}^T,$$

donde $\mathbf{1}$ es un vector de “1”s de m componentes. El resultado de hacer el producto entre el vector $\mathbf{1}$ y $\boldsymbol{\mu}^T$ es una

matriz repitiendo el vector $\boldsymbol{\mu}^T$ m -veces, es decir,

$$\begin{aligned} Z &= \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_{m,1} & x_{m,2} & x_{m,3} & \dots & x_{m,n} \end{bmatrix} - \begin{bmatrix} \mu_1 & \mu_2 & \mu_3 & \dots & \mu_n \\ \mu_1 & \mu_2 & \mu_3 & \dots & \mu_n \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \mu_1 & \mu_2 & \mu_3 & \dots & \mu_n \end{bmatrix} \\ &= \begin{bmatrix} x_{1,1} - \mu_1 & x_{1,2} - \mu_2 & x_{1,3} - \mu_3 & \dots & x_{1,n} - \mu_n \\ x_{2,1} - \mu_1 & x_{2,2} - \mu_2 & x_{2,3} - \mu_3 & \dots & x_{2,n} - \mu_n \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_{m,1} - \mu_1 & x_{m,2} - \mu_2 & x_{m,3} - \mu_3 & \dots & x_{m,n} - \mu_n \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_m^T \end{bmatrix} \end{aligned}$$

Con el desarrollo anterior hemos finalizado el proceso de centrar los datos⁴.

B.4. PCA en 5 pasos, paso 2: entender qué significa un cambio de base

Antes de construir las *famosas* componentes principales, necesitamos entender qué significa hacer un cambio base cuando la **nueva** base es ortonormal, es decir, los vectores basales tiene norma-2 igual a 1 y son ortogonales entre sí.

La primera base que se estudia que cumple con lo solicitado es la base canónica. Por ejemplo, en \mathbb{R}^2 la base canónica consiste en los vectores $\mathbf{e}_1 = [1, 0]^T$ y $\mathbf{e}_2 = [0, 1]^T$. En el caso de \mathbb{R}^n los vectores canónicos se denotan de la misma forma, es decir, \mathbf{e}_i corresponde al vector con un 1 en la i -ésima componente y 0 en todas las otras. Para la base canónica es directo verificar que los vectores son ortonormales. Por completitud se presenta a continuación el desarrollo respectivo,

$$\begin{aligned} \langle \mathbf{e}_i, \mathbf{e}_j \rangle &= \sum_{k=1}^n e_{i,k} e_{j,k} = \underbrace{e_{i,i}}_1 \underbrace{e_{j,i}}_0 + \underbrace{e_{i,j}}_0 \underbrace{e_{j,j}}_1 = 0, \quad \forall i \neq j, \\ \|\mathbf{e}_i\|_2 &= \sqrt{\sum_{k=1}^n e_{i,k}^2} = \sqrt{e_{i,i}^2} = 1. \end{aligned}$$

Entonces, cada dato \mathbf{z}_i puede ser expresado como una combinación lineal de la base canónica,

$$\mathbf{z}_i = \sum_{k=1}^n c_{i,k} \mathbf{e}_k,$$

donde $c_{i,k}$ corresponde al coeficiente que debemos escalar \mathbf{e}_k , para que la combinación lineal efectivamente represente \mathbf{z}_i exactamente. Ahora, la pregunta que surge es ¿Cómo se obtiene $c_{i,k}$? Aquí es donde utilizamos lo que

⁴Un paso adicional que podría realizarse, pero que no lo realizaremos ahora, es dividir cada columna de la matriz Z por la desviación estándar de cada variable, es decir, cada elemento quedaría de la siguiente forma $c_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j}$, donde $\sigma_j = \sqrt{\text{var}([x_{1,j}, x_{2,j}, \dots, x_{m,j}])}$. Se sugiere explorar que es lo que ocurre si se hace este cambio de variables luego de entender PCA! En este caso se requiere almacenar, además de $\boldsymbol{\mu}$, el vector de varianzas o de desviaciones estándar, digamos $\boldsymbol{\sigma}$, para hacer la reconstrucción posterior de la data.

conocemos de los vectores basales ortonormales \mathbf{e}_i , es decir, que son ortonormales! Por ejemplo, si calculamos el producto interno con respecto a \mathbf{e}_j se obtiene lo siguiente,

$$\begin{aligned}\langle \mathbf{e}_j, \mathbf{z}_i \rangle &= \left\langle \mathbf{e}_j, \sum_{k=1}^n c_{i,k} \mathbf{e}_k \right\rangle \\ &= \sum_{k=1}^n c_{i,k} \langle \mathbf{e}_j, \mathbf{e}_k \rangle \\ &= c_{i,j} \underbrace{\langle \mathbf{e}_j, \mathbf{e}_j \rangle}_1 + \sum_{\substack{k=1 \\ k \neq j}}^n c_{i,k} \underbrace{\langle \mathbf{e}_j, \mathbf{e}_k \rangle}_0 \\ &= c_{i,j}.\end{aligned}$$

Lo que nos permite expresar el cambio de base de la siguiente forma,

$$\begin{aligned}\mathbf{z}_i &= \sum_{k=1}^n c_{i,k} \mathbf{e}_k \\ &= \sum_{k=1}^n \langle \mathbf{e}_k, \mathbf{z}_i \rangle \mathbf{e}_k.\end{aligned}$$

Ahora, si hacemos el mismo cambio de base a la matriz Z obtenemos la siguiente expresión,

$$Z^T = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \dots \quad \mathbf{e}_n] \underbrace{\begin{bmatrix} c_{1,1} & c_{2,1} & c_{3,1} & \dots & c_{m,1} \\ c_{1,2} & c_{2,2} & c_{3,2} & \dots & c_{m,2} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ c_{1,n} & c_{2,n} & c_{3,n} & \dots & c_{m,n} \end{bmatrix}}_{C^T},$$

donde la matriz $[\mathbf{e}_1 \quad \mathbf{e}_2 \quad \dots \quad \mathbf{e}_n]$ es la matriz identidad de $n \times n$ y la matriz C contiene los coeficientes transformados. Al aplicar la transpuesta a la ecuación anterior obtenemos su versión simplificada,

$$\begin{aligned}Z &= C \underbrace{[\mathbf{e}_1 \quad \mathbf{e}_2 \quad \dots \quad \mathbf{e}_n]^T}_{I_n} \\ &= C I_n,\end{aligned}$$

donde I_n es la matriz identidad de dimensión $n \times n$. En este caso, como usamos los vectores canónicos obtuvimos la matriz identidad, y no se logra cambio algunos entre los coeficientes en Z y C , por lo tanto son iguales. Lo valioso es el análisis realizado. Lo interesante surge cuando cambiamos los vectores basales \mathbf{e}_i por otro conjunto de vectores ortonormales, digamos \mathbf{v}_i . En este caso el análisis anterior sigue siendo válido y podemos ir directamente a las expresiones finales. Por lo tanto obtenemos,

$$\begin{aligned}\mathbf{z}_i &= \sum_{k=1}^n y_{i,k} \mathbf{v}_k \\ &= \sum_{k=1}^n \langle \mathbf{v}_k, \mathbf{z}_i \rangle \mathbf{v}_k,\end{aligned}$$

donde $y_{i,k}$ son los coeficientes en la nueva base, y de forma matricial,

$$\begin{aligned}Z &= Y \underbrace{[\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_n]^T}_{V^T} \\ &= Y V^T.\end{aligned}\tag{B.4}$$

En esta sección hemos logrado escribir matricialmente un cambio de base y describir explícitamente los coeficientes considerando que la nueva base sea ortonormal. Notar que el cambio sigue siendo válido si la base es *full-rank* pero no ortonormal, lo que se pierde en ese caso es la descripción explícita de los coeficientes y se traduce en resolver m sistemas de ecuaciones lineales!

B.5. PCA en 5 pasos, paso 3: varianza y covarianza

Antes de analizar la varianza y covarianza, introduciremos una conveniente notación para evitar confundir sobre qué obtendremos las varianzas y covarianzas. Recordando la definición de Z , podemos definir los vectores $\hat{\mathbf{z}}_j \in \mathbb{R}^m$ de la siguiente forma,

$$\begin{aligned} Z &= \begin{bmatrix} x_{1,1} - \mu_1 & x_{1,2} - \mu_2 & x_{1,3} - \mu_3 & \dots & x_{1,n} - \mu_n \\ x_{2,1} - \mu_1 & x_{2,2} - \mu_2 & x_{2,3} - \mu_3 & \dots & x_{2,n} - \mu_n \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_{m,1} - \mu_1 & x_{m,2} - \mu_2 & x_{m,3} - \mu_3 & \dots & x_{m,n} - \mu_n \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_m^T \end{bmatrix} \\ &= [\hat{\mathbf{z}}_1 \quad \hat{\mathbf{z}}_2 \quad \hat{\mathbf{z}}_3 \quad \dots \quad \hat{\mathbf{z}}_n]. \end{aligned}$$

En este caso podemos notar que los vectores \mathbf{z}_i^T representan la i -ésima fila de la matriz Z y los vectores $\hat{\mathbf{z}}_j$ representan la j -ésima columna de la misma matriz. Es importante que quede clara esta diferencia. Ahora, habiendo introducido los vectores $\hat{\mathbf{z}}_j$ podemos re-escribir las ecuaciones (B.2) y (B.2) sobre nuestros conjuntos de datos almacenados en los vectores $\hat{\mathbf{z}}_j$.

$$\begin{aligned} \text{var}(\hat{\mathbf{z}}_j) &= \frac{1}{m-1} \sum_{k=1}^m \hat{z}_{j,k}^2 \\ &= \frac{1}{m-1} \|\hat{\mathbf{z}}_j\|_2^2. \\ \text{cov}(\hat{\mathbf{z}}_i, \hat{\mathbf{z}}_j) &= \frac{1}{m-1} \langle \hat{\mathbf{z}}_i, \hat{\mathbf{z}}_j \rangle. \end{aligned}$$

Notar que en este caso no es necesario restar las medias, dado que por construcción sabemos que son 0. Sí es importante destacar que en este caso la varianza es solamente la norma-2 del vector $\hat{\mathbf{z}}_j$ y la covarianza es el producto interno⁵ entre $\hat{\mathbf{z}}_i$ y $\hat{\mathbf{z}}_j$ escalado por $\frac{1}{m-1}$. Es interesante recordar que podemos escribir el producto interno, por ende la covarianza, en función de la norma-2 de los vectores involucrados y el coseno del ángulo entre ellos en \mathbb{R}^m de la siguiente forma,

$$\begin{aligned} \text{cov}(\hat{\mathbf{z}}_i, \hat{\mathbf{z}}_j) &= \frac{1}{m-1} \langle \hat{\mathbf{z}}_i, \hat{\mathbf{z}}_j \rangle \\ &= \frac{1}{m-1} \|\hat{\mathbf{z}}_i\|_2 \|\hat{\mathbf{z}}_j\|_2 \cos(\theta_{i,j}). \end{aligned}$$

Esta representación nos permite entender mejor el rol de la covarianza, que mide la dependencia lineal entre variables. Si la covarianza fuera 0, entonces sabemos que los vectores $\hat{\mathbf{z}}_i$ y $\hat{\mathbf{z}}_j$ son ortogonales. Por el contrario, el máximo (o mínimo) de la covarianza se obtiene cuando $\theta_{i,j} = \pm \frac{\pi}{2}$. Notar que la covarianza puede ser positiva o negativa.

⁵También conocido en el barrio como el producto punto!

El siguiente paso natural es obtener las covarianzas entre todos los vectores $\hat{\mathbf{z}}_i$, lo cual puede escribirse de la siguiente manera,

$$\frac{1}{m-1} \begin{bmatrix} \hat{\mathbf{z}}_1^T \\ \hat{\mathbf{z}}_2^T \\ \hat{\mathbf{z}}_3^T \\ \vdots \\ \hat{\mathbf{z}}_n^T \end{bmatrix} \begin{bmatrix} \hat{\mathbf{z}}_1 & \hat{\mathbf{z}}_2 & \hat{\mathbf{z}}_3 & \dots & \hat{\mathbf{z}}_n \end{bmatrix} = \frac{1}{m-1} \begin{bmatrix} \hat{\mathbf{z}}_1^T \hat{\mathbf{z}}_1 & \hat{\mathbf{z}}_1^T \hat{\mathbf{z}}_2 & \hat{\mathbf{z}}_1^T \hat{\mathbf{z}}_3 & \dots & \hat{\mathbf{z}}_1^T \hat{\mathbf{z}}_n \\ \hat{\mathbf{z}}_2^T \hat{\mathbf{z}}_1 & \hat{\mathbf{z}}_2^T \hat{\mathbf{z}}_2 & \hat{\mathbf{z}}_2^T \hat{\mathbf{z}}_3 & \dots & \hat{\mathbf{z}}_2^T \hat{\mathbf{z}}_n \\ \hat{\mathbf{z}}_3^T \hat{\mathbf{z}}_1 & \hat{\mathbf{z}}_3^T \hat{\mathbf{z}}_2 & \hat{\mathbf{z}}_3^T \hat{\mathbf{z}}_3 & \dots & \hat{\mathbf{z}}_3^T \hat{\mathbf{z}}_n \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \hat{\mathbf{z}}_n^T \hat{\mathbf{z}}_1 & \hat{\mathbf{z}}_n^T \hat{\mathbf{z}}_2 & \hat{\mathbf{z}}_n^T \hat{\mathbf{z}}_3 & \dots & \hat{\mathbf{z}}_n^T \hat{\mathbf{z}}_n \end{bmatrix},$$

donde $\hat{\mathbf{z}}_i^T \hat{\mathbf{z}}_j = \langle \hat{\mathbf{z}}_i, \hat{\mathbf{z}}_j \rangle$, es decir, hacer el producto entre un vector columna transpuesto y otro vector columna es equivalente a hacer el producto punto. Por lo tanto lo que se obtiene son las covarianzas de todos con todos⁶, lo interesante ahora es que la notación se puede simplificar significativamente si escribimos la operación anterior en función de la matriz Z , es decir,

$$\frac{1}{m-1} \begin{bmatrix} \hat{\mathbf{z}}_1^T \\ \hat{\mathbf{z}}_2^T \\ \hat{\mathbf{z}}_3^T \\ \vdots \\ \hat{\mathbf{z}}_n^T \end{bmatrix} \begin{bmatrix} \hat{\mathbf{z}}_1 & \hat{\mathbf{z}}_2 & \hat{\mathbf{z}}_3 & \dots & \hat{\mathbf{z}}_n \end{bmatrix} = \frac{1}{m-1} Z^T Z.$$

Este resultado es muy interesante. Por ejemplo podemos listar las siguientes propiedades para la matriz $\frac{1}{m-1} Z^T Z$,

- Es de dimensión $n \times n$.
- Es simétrica.
- Es, por lo menos, semi-definida positiva. Esto significa que los valores propios son reales y mayores e iguales a 0.
- En algunos casos incluso es positiva definida, es decir, los valores propios son reales y mayores que 0.
- Los términos de la diagonal corresponden a las varianzas de los vectores de datos $\tilde{\mathbf{z}}_i$.
- Los términos fuera de la diagonal corresponden a las covarianzas respectivas.

¡MUY IMPORTANTE!

Una consideración importante corresponde al caso en que existiera dependencia lineal entre un par de vectores, por ejemplo $\tilde{\mathbf{z}}_i$ y $\tilde{\mathbf{z}}_j$. Esto puede detectarse cuando el $\cos(\theta_{i,j}) = 1$, lo cual se obtiene de la siguiente forma,

$$\cos(\theta_{i,j}) = \frac{\langle \hat{\mathbf{z}}_i, \hat{\mathbf{z}}_j \rangle}{\|\hat{\mathbf{z}}_i\|_2 \|\hat{\mathbf{z}}_j\|_2}.$$

Si efectivamente se determina que $\cos(\theta_{i,j}) = 1$, entonces una alternativa sería eliminar uno de los 2 vectores del conjunto de datos porque no están entregando información *nueva*, solo repiten la información. Se sugiere revisar de todos modos esto para todos los pares de vectores involucrados, es decir, solo se debe analizar la parte triangular superior o triangular inferior de $Z^T Z$, sin considerar la diagonal. ¿Qué significa que en la diagonal se obtenga algún valor nulo? ¿Qué se debería hacer si se detecta que hay un coeficiente nulo en la diagonal?

⁶Esto es muy similar a un *n-body-problem*, es decir, la interacción de todos con todos!

En resumen, la matriz de varianzas y covarianzas se obtiene de la siguiente forma,

$$M = \frac{1}{m-1} Z^T Z. \quad (\text{B.5})$$

B.6. PCA en 5 pasos, paso 4: construyendo las componentes principales

Finalmente llegamos a la sección donde construiremos las *componentes principales*!

- Primero, debemos recordar la ecuación (B.4), donde re-escribimos el conjunto de datos bajo una nueva base, aún desconocida pero distinta a la original. El cambio corresponde a $Z = Y V^T$.
- Segundo, en la sección anterior determinamos que podemos escribir la matriz de varianzas y covarianzas de la siguiente forma: $M = \frac{1}{m-1} Z^T Z$.

Estos 2 puntos mencionados son la clave para la construcción de las componentes principales. Por ejemplo si reemplazamos $Z = Y V^T$ en la definición de la matriz de varianzas y covarianzas obtenemos,

$$\begin{aligned} M &= \frac{1}{m-1} Z^T Z \\ &= \frac{1}{m-1} (Y V^T)^T (Y V^T) \\ &= \frac{1}{m-1} V Y^T Y V^T. \end{aligned}$$

En el desarrollo anterior hemos logrado conectar la matriz de varianzas y covarianzas M con la nueva base V y los coeficientes respectivos Y . Este aún no nos entrega el algoritmo para determinar la nueva base, el paso final lo presentamos a continuación. Lo único que debemos recordar es que estamos considerando construir una nueva base ortonormal, lo que implica que $V^{-1} = V^T$. Entonces al multiplicar por la izquierda por V^T y por la derecha por V obtenemos,

$$\begin{aligned} M &= \frac{1}{m-1} V Y^T Y V^T, \\ V^T M V &= \frac{1}{m-1} V^T V Y^T Y V^T V \\ \underbrace{V^T M V}_{M_V} &= \frac{1}{m-1} Y^T Y \\ M_V &= \frac{1}{m-1} Y^T Y. \end{aligned}$$

En este caso hemos obtenido una expresión explícita para la matriz de varianzas y covarianzas, i.e. M_V , de los vectores de datos transformados almacenados en Y . Por lo cual surge la pregunta, ¿Cuál será la mejor estructura de la matriz de varianzas y covarianzas que uno puede obtener? La respuesta es simple, una matriz diagonal! Entonces ahora sí tenemos implícitamente un algoritmo para obtener V , y se basa en el siguiente hecho,

$$\begin{aligned} M_V &= V^T M V \\ \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n) &= V^T M V. \end{aligned}$$

Lo cual nos indica que debemos **diagonalizar** M , o también lo podemos re-escribir de la siguiente forma,

$$M = V \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n) V^T. \quad (\text{B.6})$$

Ahora surge la pregunta, ¿Qué es lo que realmente nos indica la ecuación (B.6)? Lo que nos indica la ecuación (B.6) es que debemos obtener la descomposición de valores propios de M , donde los coeficientes de la matriz diagonal, λ_i , son los valores propios y V contiene los vectores propios de M .

Entonces las componentes principales corresponden a los vectores propios de la matriz M , donde la más significativa corresponde a la asociado al valor propio de mayor, y así sucesivamente. Recuerde que en este caso los valores propios son reales y mayores o iguales a 0, por lo tanto sí podemos ordenarlos de mayor a menor.

B.6.1. La conexión secreta de PCA con la SVD

En el apéndice A se presentó la descomposición de valores singulares, o simplemente SVD. Ahora, se presenta como se relaciona con PCA. Lo primero que debemos hacer es recordar la ecuación (B.5), es decir,

$$M = \frac{1}{m-1} Z^T Z.$$

Aquí tenemos 2 alternativas:

- (I) Obtener la SVD de la matriz M .
- (II) Obtener la SVD de la matriz Z .

Lo interesante es que ambas alternativas nos entregan lo que queremos, es decir la matriz V . Por lo que es importante analizar cómo ocurre eso en cada caso.

En el primer caso, i.e. (I), si obtenemos la SVD de M debemos recordar que M es simétrica⁷ y semi-definida positiva (o incluso definida positiva), por lo que la SVD obtenida será al mismo tiempo la SVD completa y reducida, es decir, $M = U \Sigma_M V^*$ ⁸. Pero al ser simétrica y con coeficientes reales, tenemos $U = V$ y el operador $*$ se convierte en la transpuesta, por lo tanto recuperamos $M = V \Sigma_M V^T$. Lo que es exactamente lo mismo que la descomposición de valores propios de M , por lo tanto esta alternativa no nos entrega nada nuevo respecto a lo que ya teníamos.

En el segundo caso, i.e. (II), el resultado es distinto, ya que podemos obtener la SVD completa y reducida de Z . Notar que utilizaremos el operador transpuesta T en vez de la transpuesta conjugada $*$ porque los datos son números reales. Entonces,

$$\begin{aligned} Z &= U \Sigma V^T \\ &= [\hat{U}, \hat{U}^\perp] \begin{bmatrix} \hat{\Sigma} \\ 0 \end{bmatrix} V^T \\ &= \hat{U} \hat{\Sigma} V^T. \end{aligned}$$

Es muy importante indicar en este punto que si bien matemáticamente la SVD completa y reducida generan *exactamente*⁹ Z , hay una diferencia computacional **muy** importante. La diferencia recae en los requerimiento de memoria. Por ejemplo considere que $m \gg n$, los requerimiento de memoria son los siguientes,

- SVD completa: $U \Sigma V^T$,
 - $U \in \mathbb{R}^{m \times m}$.
 - $\Sigma \in \mathbb{R}^{m \times n}$, pero solo se necesitan almacenar los n coeficientes de la diagonal, ya que el resto son 0.
 - $V \in \mathbb{R}^{n \times n}$.
 - Cantidad de elementos a almacenar: $m^2 + n + n^2$.

⁷Lo que implica que la matriz sea cuadrada también.

⁸Notar que hemos introducido la notación de Σ_M para destacar que la matriz diagonal Σ_M almacena los valores singulares de M .

⁹Recuerde que lo "exacto" es considerando aritmética de punto flotante!

- SVD reducida: $\hat{U} \hat{\Sigma} V^T$,
 - $\hat{U} \in \mathbb{R}^{m \times n}$.
 - $\hat{\Sigma} \in \mathbb{R}^{n \times n}$.
 - $V \in \mathbb{R}^{n \times n}$.
 - Cantidad de elementos a almacenar: $m n + n + n^2$.

Por lo tanto la diferencia es muy significativa si $m \gg n$. Este análisis nos indica que en este caso es computacionalmente conveniente trabajar con la SVD reducida!¹⁰

Entonces, si obtenemos la SVD reducida de $Z = \hat{U} \hat{\Sigma} V^T$, podemos reemplazarla en la definición de M , entonces,

$$\begin{aligned}
 M &= \frac{1}{m-1} Z^T Z \\
 &= \frac{1}{m-1} (\hat{U} \hat{\Sigma} V^T)^T (\hat{U} \hat{\Sigma} V^T) \\
 &= \frac{1}{m-1} V \hat{\Sigma} \underbrace{\hat{U}^T \hat{U}}_{I_n} \hat{\Sigma} V^T \\
 &= \frac{1}{m-1} V \hat{\Sigma}^2 V^T \\
 &= \frac{1}{m-1} V \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2) V^T \\
 &= V \text{diag}\left(\frac{\sigma_1^2}{m-1}, \frac{\sigma_2^2}{m-1}, \dots, \frac{\sigma_n^2}{m-1}\right) V^T.
 \end{aligned}$$

Por lo tanto, V efectivamente diagonaliza M . Más aún, sabemos que los valores propios λ_i de M son efectivamente $\frac{\sigma_i^2}{m-1}$ ¹¹. También es importante destacar que al trabajar con la SVD reducida de Z no tenemos nunca que calcular $Z^T Z$ ¹².

En resumen, hemos presentado la conexión secreta y *conveniente* entre PCA y la SVD!

B.7. PCA en 5 pasos, paso 5: escribiendo los datos originales con la nueva base

Este último paso conecta todas las componentes presentados anteriormente, por completitud se presenta un resumen:

- Paso 1, centrar los datos: $\mathbf{x}_i = \mathbf{z}_i + \boldsymbol{\mu}$, para $i \in \{1, 2, 3, \dots, m\}$, donde se obtiene la media vectorial $\boldsymbol{\mu}$ y se empieza a trabajar con los datos centrados \mathbf{z}_i . En este paso también se presentó la versión matricial: $Z = X - \mathbf{1} \boldsymbol{\mu}^T$.
- Paso 2, entender qué significa un cambio de base: $\mathbf{z}_i = \sum_{k=1}^n y_{i,k} \mathbf{v}_k$ y la versión matricial para todo el conjunto de datos $Z = Y V^T$. En este caso se analiza que es lo que significa re-escribir el dato \mathbf{z}_i en una nueva base \mathbf{v}_k , en particular se considera que \mathbf{v}_k sea una base ortonormal, por lo que se obtiene que $y_{i,k} = \langle \mathbf{v}_k, \mathbf{z}_i \rangle$.

¹⁰Por lo tanto cuando usted quiera obtener la SVD de una matriz debe asegurarse de cual SVD construirá, por ejemplo en algunas librerías numéricas la SVD que se entrega por defecto es la SVD completa, por lo que debe indicarle a la librería que usted quiere la SVD reducida. Si no lo hace, probablemente saturará su memoria RAM rápidamente!

¹¹Dependiendo de la referencia que se use el coeficiente $\frac{1}{m-1}$ pudiera ser o no incluido. Notar sin embargo que V no depende de m , así que el impacto de incluirlo o omitirlo afecta la magnitud de los valores propios o valores singulares, respectivamente.

¹²Se sugiere recordar el análisis presentado en la apéndice A.3.

- Paso 3, varianza y covarianza: $M = \frac{1}{m-1} Z^T Z$. En este caso se obtiene las varianzas de cada columna de Z ¹³ y las covarianzas entre los distintos conjuntos de datos.
- Paso 4, construyendo las componentes principales: $M = V \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n) V^T = V \text{diag}\left(\frac{\sigma_1^2}{m-1}, \frac{\sigma_2^2}{m-1}, \dots, \frac{\sigma_n^2}{m-1}\right) V^T$. En este caso se presentan 2 alternativas para obtener la nueva base, i.e. la matriz V . La primera basada en la descomposición de valores propios de M y la segunda basada en la SVD reducida de Z .

Entonces, con lo 4 pasos previos, podemos construir explícitamente la relación entre la data original y su nueva estructura basada en las componentes principales. El formato para un vector de datos corresponde a,

$$\begin{aligned} \mathbf{x}_i &= \boldsymbol{\mu} + \mathbf{z}_i \\ &= \boldsymbol{\mu} + \sum_{k=1}^n y_{i,k} \mathbf{v}_k \\ &= \boldsymbol{\mu} + \sum_{k=1}^n \langle \mathbf{v}_k, \mathbf{z}_i \rangle \mathbf{v}_k, \end{aligned}$$

y la versión matricial,

$$\begin{aligned} X &= Z + \mathbf{1} \boldsymbol{\mu}^T \\ &= Y V^T + \mathbf{1} \boldsymbol{\mu}^T \\ &= (Z V) V^T + \mathbf{1} \boldsymbol{\mu}^T. \end{aligned}$$

Por lo tanto hemos finalizado la descripción de los datos utilizando las componentes principales.

B.8. ¿Por qué queremos re-escribir la matriz de datos utilizando las componentes principales?

En las secciones anteriores hemos estudiado profundamente la componentes principales: lo que son y como se obtienen. Sin embargo hasta el momento no hay ventaja aparente de porque hacer todo es esfuerzo sin ver su ventaja. La ventaja del uso de las componentes principales, o incluso cualquier otro cambio de base, es que podemos *aproximar* los datos originales sin incurrir en un error *significativo*¹⁴. Esta aproximación se logra truncando las expresiones anteriores, por ejemplo en el caso de la versión vectorial se obtiene,

$$\begin{aligned} \mathbf{x}_i &= \boldsymbol{\mu} + \mathbf{z}_i \\ &= \boldsymbol{\mu} + \sum_{k=1}^n y_{i,k} \mathbf{v}_k \\ &= \underbrace{\boldsymbol{\mu} + \sum_{k=1}^l y_{i,k} \mathbf{v}_k}_{\text{aproximación con } l \text{ términos}} + \underbrace{\sum_{k=l+1}^n y_{i,k} \mathbf{v}_k}_{\text{error}}, \end{aligned}$$

donde podemos definir los siguientes términos,

- $\tilde{\mathbf{z}}_i = \sum_{k=1}^l y_{i,k} \mathbf{v}_k$: aproximación de \mathbf{z}_i .

¹³Aquí es importante recordar que se analizan el conjunto de las distintas componentes de cada vector de datos.

¹⁴Este dependerá de cada aplicación en particular

- $\tilde{\mathbf{y}}_i = [y_{i,1}, y_{i,2}, \dots, y_{i,l}]$: primeros l componentes de $\mathbf{y}_i = [y_{i,1}, y_{i,2}, \dots, y_{i,n}]$.
- $\tilde{\mathbf{x}}_i = \boldsymbol{\mu} + \tilde{\mathbf{z}}_i$: aproximación de \mathbf{x}_i con l términos.

Y su versión matricial,

$$\begin{aligned}
 X &= Y V^T + \mathbf{1} \boldsymbol{\mu}^T \\
 &= [\tilde{Y} \quad \check{Y}] \begin{bmatrix} \tilde{V}^T \\ \check{V}^T \end{bmatrix} + \mathbf{1} \boldsymbol{\mu}^T \\
 &= \underbrace{\tilde{Y} \tilde{V}^T + \mathbf{1} \boldsymbol{\mu}^T}_{\substack{\text{aproximación con} \\ l \text{ términos}}} + \underbrace{\check{Y} \check{V}^T}_{\text{error}}
 \end{aligned}$$

donde $\tilde{V} = [\mathbf{v}_1 \quad \dots \quad \mathbf{v}_l]$ y $\check{V} = [\mathbf{v}_{l+1} \quad \dots \quad \mathbf{v}_n]$. Esta aproximación tiene 2 puntos interesantes,

- El error está determinado de forma explícita, es decir, $\check{Y} \check{V}^T$. Se puede notar que al aumentar la cantidad de términos en la aproximación el error disminuye, y en el caso límite, i.e. $l = n$, la aproximación representa exactamente los datos.
- El éxito de la aproximación es cuando podemos determinar que con un coeficiente l mucho menos que n el error es bajo. Esto debe verificarse problema a problema, sin embargo se pueden destacar 2 cosas: (i) se necesita mucho menos memoria para almacenar la aproximación respecto a los datos originales, (ii) al trabajar con su versión proyectada, es decir los coeficientes almacenados en \tilde{Y} y no con X directamente, la computación se puede reducir significativamente porque estaríamos trabajando con datos en \mathbb{R}^l y no en \mathbb{R}^n , donde $n \gg l$.

Respecto al segundo punto podemos analizar explícitamente los requerimientos de memoria involucrados,

- X requiere almacenar $m n$ coeficientes.
- La aproximación con l componentes principales requiere almacenar los siguientes coeficientes:
 - $m l$ para \tilde{Y} .
 - $n l$ para \check{V} .
 - n para $\boldsymbol{\mu}$.
 - En total: $l(m + n) + n$.

Por lo tanto almacenar la aproximación es conveniente siempre y cuando $l < \left\lfloor \frac{n(m-1)}{m+n} \right\rfloor$.

B.9. EXTRA: Compresión de imágenes con la SVD

En esta sección se presentará una muy breve explicación de como utilizar la SVD para comprimir imágenes¹⁵. Por simplicidad trabajaremos con una imagen en escala de grises, sin embargo la misma idea aplica para imágenes RGB, donde uno debería aplicar la compresión a cada capa de forma independiente.

Llamemos G a una imagen en escala de grises de dimensión $m \times n$, donde $G_{i,j}$ corresponde al píxel (i, j) y lo que se almacena en el $G_{i,j}$ es la intensidad respectiva. Entonces, dado que G es una matriz, podemos obtener la SVD reducida,

$$G = \hat{U} \hat{\Sigma} V^T,$$

¹⁵En realidad pueden haber muchas formas de como hacerlo, incluso algunas basadas en PCA.

o, según el teorema 25, equivalentemente,

$$G = \sum_{k=1}^r \sigma_k \mathbf{u}_k \mathbf{v}_k^T,$$

donde r corresponde al rango de G . Siguiendo el mismo argumento utilizado en PCA, uno puede truncar la sumatoria, es decir,

$$\begin{aligned} G &= \underbrace{\sum_{k=1}^l \sigma_k \mathbf{u}_k \mathbf{v}_k^T}_{\tilde{G}, \text{aproximación con } l \text{ términos}} + \underbrace{\sum_{k=l+1}^r \sigma_k \mathbf{u}_k \mathbf{v}_k^T}_{\check{G}, \text{error}} \\ &= \tilde{G} + \check{G}. \end{aligned}$$

Aquí podemos indicar que la imagen resultante luego de haberla aproximado con l términos es \tilde{G} y \check{G} corresponde al error de la aproximación. Lo interesante de esta aproximación es que podemos elegir l en función de como decaen los valores singulares σ_k , i.e. si decaen muy rápido, solo con unos pocos términos la aproximación puede ser bastante buena!

Note que dado que G tiene $m \times n$ coeficientes, \tilde{G} también tendrá $m \times n$ coeficientes. Entonces, ¿Por qué sería interesante construir \tilde{G} si necesitamos, en principio, almacenar la misma cantidad de coeficientes que para almacenar G ? Más aún, si almacenamos \tilde{G} de forma explícita, estaríamos *empeorando* la imagen original porque tendría una diferencia \check{G} con respecto a la imagen original. El punto clave para lograr efectivamente una compresión, i.e. almacenar menos coeficientes, es almacenar $(\sigma_k, \mathbf{u}_k, \mathbf{v}_k)$ para $k \in \{1, 2, 3, \dots, l\}$, y no el producto. Esto significa que cada vez que se necesite acceder a la imagen comprimida se debe hacer el producto. Nuevamente, si se logra una buena compresión, i.e. l pequeño, entonces el producto no sería un costo tan significativo.

A modo resumen, podemos comparar la cantidad de coeficientes que se requieren almacenar en cada caso,

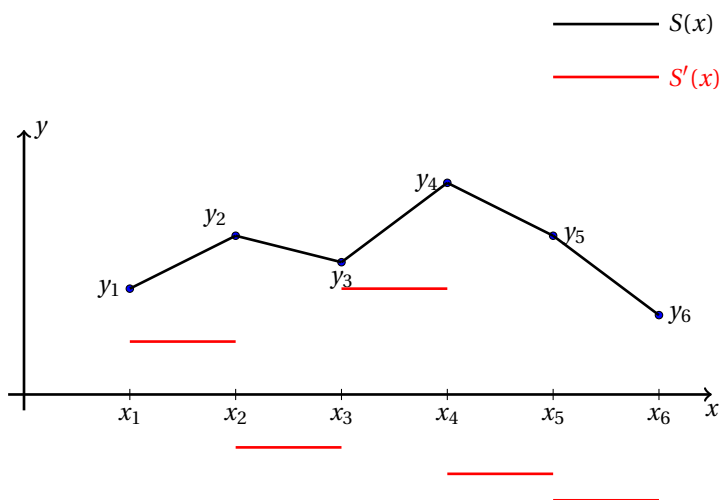
- Cantidad de coeficientes de la imagen original: $m n$.
- Cantidad de coeficientes de la imagen comprimida:
 - Se necesitan almacenar l coeficientes σ_k : l .
 - Se necesitan almacenar l vectores \mathbf{u}_k : $l m$.
 - Se necesitan almacenar l vectores \mathbf{v}_k : $l n$.
 - Total: $l(m + n + 1)$.

Entonces, si $l < \left\lfloor \frac{m n}{m + n + 1} \right\rfloor$ se obtiene una compresión efectiva!

Apéndice C

Splines Cúbicas

Splines es un enfoque alternativo a la interpolación de datos por medio de polinomios, la idea es definir una función por intervalos. El ejemplo más simple de Spline es la Spline lineal, aquí se conectan los puntos a través de rectas. Es decir para n puntos: $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ con $x_1 < x_2 < \dots < x_n$ la spline lineal se compone de $n - 1$ segmentos. Veamos un ejemplo para $n = 6$:



Algebraicamente:

$$S_1(x) = y_1 + b_1(x - x_1), \quad x \in [x_1, x_2]$$

$$S_2(x) = y_2 + b_2(x - x_2), \quad x \in [x_2, x_3]$$

$$S_3(x) = y_3 + b_3(x - x_3), \quad x \in [x_3, x_4]$$

$$S_4(x) = y_4 + b_4(x - x_4), \quad x \in [x_4, x_5]$$

$$S_5(x) = y_5 + b_5(x - x_5), \quad x \in [x_5, x_6]$$

¿Cómo encontramos los b_i 's?

$$\begin{aligned}
 S_1(x_2) &= S_2(x_2) \\
 y_1 + b_1(x_2 - x_1) &= y_2 \\
 b_1 \cdot (x_2 - x_1) &= y_2 - y_1
 \end{aligned}$$

Repitiendo el mismo análisis para x_3 , x_4 , x_5 y x_6 y re-escribiendo las ecuaciones como un sistema de ecuaciones lineales, obtenemos:

$$\begin{bmatrix}
 (x_2 - x_1) & 0 & 0 & 0 & 0 \\
 0 & (x_3 - x_2) & 0 & 0 & 0 \\
 0 & 0 & (x_4 - x_3) & 0 & 0 \\
 0 & 0 & 0 & (x_5 - x_4) & 0 \\
 0 & 0 & 0 & 0 & (x_6 - x_5)
 \end{bmatrix}
 \begin{bmatrix}
 b_1 \\
 b_2 \\
 b_3 \\
 b_4 \\
 b_5
 \end{bmatrix}
 =
 \begin{bmatrix}
 y_2 - y_1 \\
 y_3 - y_2 \\
 y_4 - y_3 \\
 y_5 - y_4 \\
 y_6 - y_5
 \end{bmatrix}$$

Resolviendo el sistema de ecuaciones lineales obtenemos:

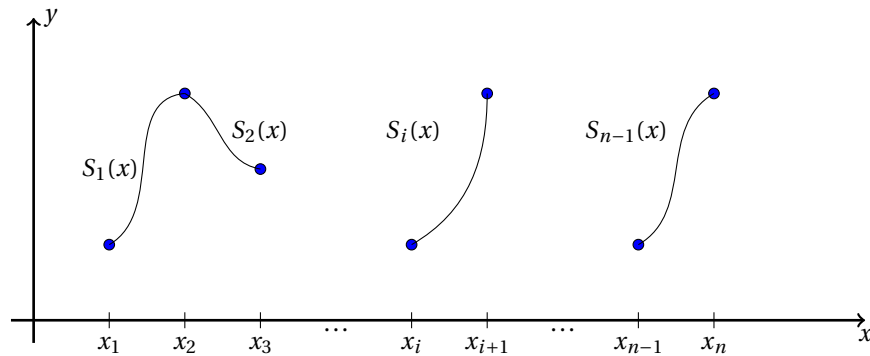
$$b_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad \forall i \in \{1, 2, 3, 4, 5\}$$

C.1. Propiedades de Spline Cúbica

Una Spline cúbica con n puntos $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ donde los x_i son distintos y en orden creciente, se define de la siguiente forma:

$$S(x) = \begin{cases} S_1(x), & x \in [x_1, x_2] \\ S_2(x), & x \in]x_2, x_3] \\ \vdots \\ S_{n-1}(x), & x \in]x_{n-1}, x_n] \end{cases}$$

donde $S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$, para $x \in [x_i, x_{i+1}]$. Y gráficamente



Además se requiere que se cumplan las siguientes propiedades para ser considerada una spline cúbica.

C.1.1. Propiedad 1 (Continuidad)

$$\begin{aligned}
 S_i(x_i) &= y_i \\
 S_i(x_{i+1}) &= y_{i+1}
 \end{aligned}
 \quad \text{para } i = 1, 2, \dots, n-1$$

C.1.2. Propiedad 2 (Diferenciabilidad)

$$S'_{i-1}(x_i) = S'_i(x_i) \text{ para } i = 2, \dots, n-1$$

C.1.3. Propiedad 3 (Continuidad en la segunda derivada)

$$S''_{i-1}(x_i) = S''_i(x_i) \text{ para } i = 2, \dots, n-1$$

C.2. Interpretación de las 3 propiedades

- La propiedad 1 garantiza que la spline interpole los puntos de datos.
- La 2da propiedad obliga a las pendientes de las partes adyacentes de la spline a ser iguales.
- La 3ra propiedad hace lo mismo pero con la segunda derivada.

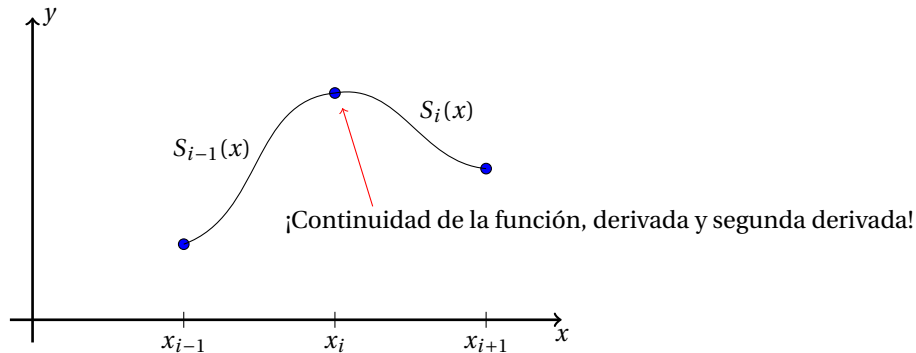
C.3. Cantidad de ecuaciones a satisfacer

La construcción de un Spline a partir de un conjunto de puntos significa que debemos encontrar los coeficientes: b_i, c_i, d_i , que hacen que se cumplan las 3 propiedades.

¿Cuántas ecuaciones se deben satisfacer?

La propiedad 1 entrega $(n-1)$ ecuaciones independientes, por otra parte las propiedades 2 y 3 nos dan $(n-2)$ cada una. En total tenemos $n-1+2(n-2) = 3n-5$ ecuaciones.

Pero tenemos 3 coeficientes por cada ecuación, es decir: $3(n-1) = 3n-3$. Finalmente la determinación de los coeficientes consiste en resolver un sistema de $3n-5$ ecuaciones lineales con $3n-3$ incógnitas. Por lo tanto aún necesitamos dos ecuaciones más para la unicidad de la solución.



Resumen:

- Cantidad de variables: $3n - 3$
- Cantidad de ecuaciones: $3n - 5$
- ¿Es cuadrada la matriz?¹
- ¿Qué falta o qué sobra?²
- ¿Qué opciones tenemos?³

Las spline también tiene otras condiciones, dependiendo de estas la spline adquiere otro nombre, veremos las más usadas.

C.4. Tipos de condiciones de borde para splines cúbicas

C.4.1. Spline Natural

Se llama **spline natural** a la spline cúbica que cumple las condiciones de borde:

$$S_1''(x_1) = S_{n-1}''(x_n) = 0$$

C.4.2. Spline con curvatura ajustada

Ahora en vez de definir las condiciones de borde en 0, podemos elegir sus valores arbitrariamente, es decir establecer las curvaturas deseadas en los extremos, es decir:

$$\begin{aligned} S_1''(x_1) &= k_1 \\ S_{n-1}''(x_n) &= k_2 \end{aligned}$$

A estas se les llama **spline con curvatura ajustada**.

C.4.2.1. Spline cúbica fijada (Clamped cubic spline)

De manera similar, definimos los valores a las pendientes en el borde de la siguiente forma:

$$\begin{aligned} S_1'(x_1) &= k_1 \\ S_{n-1}'(x_n) &= k_2 \end{aligned}$$

C.4.3. Spline Terminada parabólicamente

Si obligamos a las primera parte de la spline y a la última a ser un polinomio de grado como máximo 2, al hacer $d_1 = d_n = 0$, de esta forma requerimos que $c_1 = c_2$ y $c_{n-1} = c_n$.

¹No

²Faltan dos ecuaciones

³¡Aún falta definir las condiciones de borde!

C.4.4. Spline cúbica sin nodo (Not-a-knot cubic spline)

Como S_1 y S_2 son polinomios de grado 3 o menor, se requiere que sus terceras derivadas concuerden en x_2 , como ya concuerda en su primera y segunda derivada. Para obtener la otra condición hacemos lo mismo para S_{n-1} y S_{n-2} , es decir debe cumplir

$$\begin{aligned} S_1'''(x_2) &= S_2'''(x_2) \\ S_{n-2}'''(x_{n-1}) &= S_{n-1}'''(x_{n-1}) \end{aligned}$$

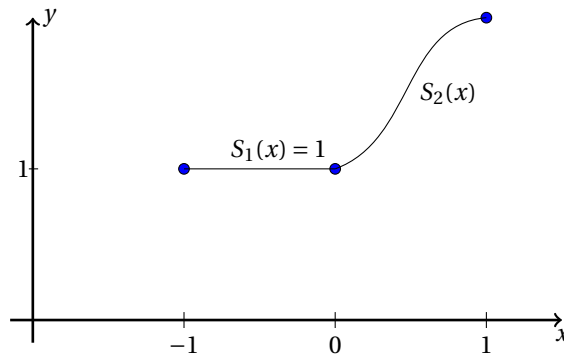
C.5. Unicidad de la Spline Cúbica

Thm 26. Asuma $n \geq 2$. Entonces, para el conjunto de datos $(x_1, y_1), \dots, (x_n, y_n)$ y para cualquiera de las condiciones de borde (C.4.1, C.4.2, C.4.2.1), existe una única spline cúbica que satisface las condiciones de borde y se ajusta a los puntos dados.

Lo mismo es válido para C.4.3 con $n \geq 3$ y para C.4.4 con $n \geq 4$.

C.5.1. Ejemplo

Asuma que la parte izquierda de una spline cúbica natural es $S_1(x) = 1$ en el intervalo $[-1, 0]$. Encuentre 3 diferentes posibilidades para la definición de $S_2(x)$ en $[0, 1]$.



$$S_2(x) = 1 + b(x-0) + c(x-0)^2 + d(x-0)^3$$

$$S_1(0) = S_2(0) \quad \checkmark \checkmark$$

$$S_1'(0) = 0, S_2'(x) = b + 2cx + 3dx^2$$

$$S_2'(0) = b$$

$$\therefore b = 0$$

$$S_1''(0) = 0, S_2'(x) = 2c + 6dx$$

$$S_2''(0) = 2c$$

$$\therefore c = 0$$

$$\Rightarrow S_2(x) = 1 + dx^3$$

- Sub-sección C.4.1 $\Rightarrow S_2''(1) = 6d = 0 \Rightarrow d = 0 \Rightarrow S_2(x) = 1$
- Sub-sección C.4.2 $\Rightarrow S_2''(1) = 6d = K_2 \Rightarrow d = K_2/6 \Rightarrow S_2(x) = 1 + x^3 \cdot K_2/6$
- Sub-sección C.4.2.1 $\Rightarrow S'(1) = 3d \cdot 1^2 = P_2 \Rightarrow d = P_2/3 \Rightarrow S_2(x) = 1 + x^3 \cdot P_2/3$
- Sub-sección C.4.3 $\Rightarrow d = 0 \Rightarrow S_2(x) = 1$
- Sub-sección C.4.4 $\Rightarrow S_1'''(0) = 0, S_2'''(0) = 6d \Rightarrow d = 0 \Rightarrow S_2(x) = 1$

C.6. Ejercicios Propuestos

1. (0,0), (1,1), (2,4). Determine las ecuaciones para construir la spline cúbica considerando todas las posibles condiciones de borde.
2. Encuentre "c" para las siguientes splines cúbicas.

a)

$$S(x) = \begin{cases} 4 - \frac{11}{4}x + \frac{3}{4}x^3, & x \in [0, 1] \\ 2 - \frac{1}{2}(x-1) + c(x-1)^2 - \frac{3}{4}(x-1)^3, & x \in [1, 2] \end{cases}$$

b)

$$S(x) = \begin{cases} 3 - 9x + 4x^2, & x \in [0, 1] \\ -2 - (x-1) + c(x-1)^2, & x \in [1, 2] \end{cases}$$

3. Decida si las siguientes funciones son splines cúbicas.

a)

$$S(x) = \begin{cases} x^3 + x - 1, & x \in [0, 1] \\ -(x-1)^3 + 3(x-1)^2 + 3(x-1) + 1, & x \in [1, 2] \end{cases}$$

b)

$$S(x) = \begin{cases} 2x^3 + x^2 + 4x + 5, & x \in [0, 1] \\ (x-1)^3 + 7(x-1)^2 + 12(x-1) + 12, & x \in [1, 2] \end{cases}$$

Apéndice D

Algoritmos para matrices simétricas y definidas positivas

En este anexo se presentan 3 algoritmos especializados para resolver sistemas de ecuaciones lineales cuando se conoce que la matriz asociada al sistema es simétrica y definida positiva. Es importante destacar también que si bien se presenta el algoritmo del Gradiente Descendente como un algoritmo para resolver sistemas de ecuaciones lineales también se utiliza bastante en problemas de optimización, en particular para *entrenar* redes neuronales o en cualquier problema donde se requiere ajuste de parámetros continuos.

D.1. La Factorización de Cholesky

Importante: Este algoritmo sólo funciona para matrices definidas positivas y simétricas.

Def 20 (Matriz definida positiva). *La matriz A de $n \times n$ es definida positiva si $\mathbf{x}^* A \mathbf{x} > 0$ para todo vector no nulo \mathbf{x} .*

Recordar: A es simétrica si $A = A^T$

D.1.1. Submatriz

Una submatriz principal de A es una submatriz cuadrada de A donde los elementos de su diagonal son también elementos de la diagonal de A .

D.1.2. Propiedades de una matriz definida positiva y simétrica

1. Si una matriz A de $n \times n$ es simétrica, entonces A es definida positiva si y sólo si todos sus valores propios son positivos.
2. Si una matriz A de $n \times n$ es simétrica y definida positiva, y X es una matriz de $n \times m$ “full rank” con $n \geq m$, entonces $X^T A X$ es simétrica y definida positiva.
3. Cualquier submatriz principal de una matriz A simétrica y definida positiva es también simétrica y definida positiva.

D.1.2.1. Caso base

Para matriz de 2×2 .

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \det(A) = \underline{\hspace{2cm}} > \underline{\hspace{2cm}}$$

Ahora queremos encontrar la siguiente factorización:

$$A = R^T R = \begin{bmatrix} \diagup & & 0 \\ & \diagup & \\ & & \diagup \end{bmatrix} \begin{bmatrix} \diagup & & \\ & \diagup & \\ & & \diagup \end{bmatrix}$$

donde R es una matriz triangular superior... ¿Cómo se obtiene?

D.1.3. Descomposición de matrices simétricas y definida positiva

Thm 27. Si una matriz A de $n \times n$ es simétrica y definida positiva, se puede descomponer en $A = R^T R$.

D.1.3.1. Caso Base

Caso base: 2×2

$$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} r_{11} & 0 \\ r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} r_{11} & r_{21} \\ 0 & r_{22} \end{bmatrix}$$

Multiplicando $R^T \cdot R$:

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} r_{11}^2 & r_{11} r_{21} \\ r_{11} r_{21} & r_{21}^2 + r_{22}^2 \end{bmatrix}$$

Comparando término a término:

$$a = r_{11}^2 \implies r_{11} = \pm\sqrt{a} = \sqrt{a}$$

$$b = r_{11} r_{21} \implies r_{21} = \frac{b}{\sqrt{a}}$$

$$c = r_{21}^2 + r_{22}^2 \implies c = \frac{b^2}{a} + r_{22}^2 \implies r_{22} = \frac{\pm\sqrt{ac - b^2}}{\sqrt{a}}$$

$$r_{22} = \frac{\sqrt{ac - b^2}}{\sqrt{a}} \text{ (Por convención se elige la raíz positiva)}$$

¿Es $a \cdot c - b^2 > 0$?

Sí/No ¿Por qué?

$$\Rightarrow \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} \sqrt{a} & 0 \\ \frac{b}{\sqrt{a}} & \sqrt{\frac{ac-b^2}{a}} \end{bmatrix} \begin{bmatrix} \sqrt{a} & \frac{b}{\sqrt{a}} \\ 0 & \sqrt{\frac{ac-b^2}{a}} \end{bmatrix} \\ = R^T R$$

¿Cómo se utiliza para resolver un sistema de ecuaciones lineales?

$$\begin{array}{rcl} A & \mathbf{x} & = \mathbf{b} \quad ("A" \text{ es simétrica y positiva definida}) \\ \downarrow & & \\ R^T R & \mathbf{x} & = \mathbf{b} \end{array}$$

$$\left. \begin{array}{l} R^T \mathbf{c} = \mathbf{b}, \text{ Forward substitution} \\ R \mathbf{x} = \mathbf{c}, \text{ Backward substitution} \end{array} \right\} \text{ Similar a } PA = LU \text{ o } A = LU.$$

D.1.3.2. Caso General

En general: $A = R^T R$

$$A = \left[\begin{array}{c|c} a & \mathbf{b}^T \\ \hline \mathbf{b} & C \end{array} \right] = \left[\begin{array}{c|c|c|c} R_{11} & 0 & \cdots & 0 \\ R_{21} & R_{22} & & \\ \vdots & & \ddots & \\ R_{n1} & & & R_{nn} \end{array} \right] \left[\begin{array}{c|c|c|c} R_{11} & R_{21} & \cdots & R_{n1} \\ \hline 0 & & & \\ \vdots & & \ddots & \\ 0 & & & R_{nn} \end{array} \right]$$

$$\Rightarrow \left[\begin{array}{c} a \\ \mathbf{b} \end{array} \right] = R_{11} \left[\begin{array}{c} R_{11} \\ R_{21} \\ \vdots \\ R_{n1} \end{array} \right] \Rightarrow \begin{array}{l} R_{11} = \sqrt{a} \\ R_{2:n,1} = \frac{\mathbf{b}}{\sqrt{a}} \end{array}$$

$$\Rightarrow A = \left[\begin{array}{c|c} a & \mathbf{b}^T \\ \hline \mathbf{b} & C \end{array} \right] = \left[\begin{array}{c|c} \sqrt{a} & \mathbf{0} \\ \hline \frac{\mathbf{b}}{\sqrt{a}} & R_1^T \end{array} \right] \cdot \left[\begin{array}{c|c} \sqrt{a} & \mathbf{b}^T/\sqrt{a} \\ \hline \mathbf{0} & R_1 \end{array} \right] \\ = \left[\begin{array}{c|c} a & \mathbf{b}^T \\ \hline \mathbf{b} & \frac{\mathbf{b}}{\sqrt{a}} \cdot \frac{\mathbf{b}^T}{\sqrt{a}} + R_1^T R_1 \end{array} \right]$$

$$\therefore C = \frac{\mathbf{b} \cdot \mathbf{b}^T}{a} + R_1^T R_1$$

¿Cómo encontramos $R_1^T R_1$? (¡Tenemos que hacer Cholesky otra vez!)

$\Rightarrow C - \frac{1}{a} \mathbf{b} \cdot \mathbf{b}^T = R_1^T R_1$, Pero para una matriz más pequeña. Hasta que ... ¡lleguemos a una matriz de 2×2 !

Recuerde: $A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} \sqrt{a} & 0 \\ \frac{b}{\sqrt{a}} & \sqrt{c - \frac{b^2}{a}} \end{bmatrix} \begin{bmatrix} \sqrt{a} & \frac{b}{\sqrt{a}} \\ 0 & \sqrt{c - \frac{b^2}{a}} \end{bmatrix}$

Ejemplo: Resolver $\begin{bmatrix} 4 & 0 & -2 \\ 0 & 1 & 1 \\ -2 & 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 0 \end{bmatrix}$

$$\begin{array}{ccc} \begin{bmatrix} 4 & 0 & -2 \\ 0 & 1 & 1 \\ -2 & 1 & 3 \end{bmatrix} & \rightarrow & \begin{bmatrix} 4 & 0 & -1 \\ & R_1 & \\ 0 & & R_0 \end{bmatrix} \end{array} \quad \Rightarrow C = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}, \frac{\mathbf{b}}{\sqrt{a}} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$\frac{\mathbf{b} \cdot \mathbf{b}^T}{a} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Rightarrow R_1^T R_1 = C - \frac{1}{a} \mathbf{b} \cdot \mathbf{b}^T = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\begin{array}{ccc} \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} & c=2 & \\ \begin{matrix} A_1 & R_1^T & R_1 \end{matrix} & c - \frac{\mathbf{b} \cdot \mathbf{b}^T}{a} = 2 - \frac{1 \cdot 1}{1} = 1 & \end{array}$$

$$\begin{bmatrix} 4 & 0 & -2 \\ 0 & 1 & 1 \\ -2 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Forward Substitution}$$

$$\Rightarrow \begin{array}{ccc} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 0 \end{bmatrix} & \Rightarrow & \begin{array}{l} c_1 = 2 \\ c_2 = 2 \\ -1 \cdot 2 + 1 \cdot 2 + c_3 = 0 \\ c_3 = 0 \end{array} \\ R^T & \mathbf{c} = \mathbf{b} & \end{array}$$

$$R \cdot \mathbf{x} = \mathbf{c}$$

Backward Substitution

$$\begin{bmatrix} 2 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} \quad \Rightarrow \quad \begin{array}{l} \downarrow \\ x_3 = 0 \\ x_2 = 2 \\ x_1 = 1 \end{array}$$

Recuerde:

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

D.2. Método del Gradiente Descendente

La idea principal del gradiente descendente como un problema de optimización, es encontrar el mínimo de una función cuadrática convexa al moverse en la dirección de máximo decrecimiento, es decir es un proceso iterativo. El gradiente ∇f es la dirección de máximo crecimiento, por lo tanto la dirección de máximo decrecimiento es $-\nabla f$. Después de localizar el mínimo a lo largo de esta dirección de decrecimiento se debe repetir el proceso a partir de este punto y hacer una nueva minimización unidimensional en la nueva dirección.

```

1  for i in range(n) :
2       $\mathbf{v} = \nabla f$ 
3      Minimizar  $f(\mathbf{x}_i - s\mathbf{v})$  para escalar  $s = s^*$ 
4       $\mathbf{x}_{i+1} = \mathbf{x}_i - s^* \mathbf{v}$ 

```

D.2.1. Minimizar una función cuadrática convexa

$$\Phi(\mathbf{y}) = \frac{1}{2} \mathbf{y}^T A \mathbf{y} - \mathbf{y}^T \cdot \mathbf{b}$$

$$\Rightarrow \nabla \Phi(\mathbf{y}) = \frac{1}{2} (\underbrace{A^T + A}) \mathbf{y} - \mathbf{b} = 0$$

$$\begin{array}{l} A \text{ es simétrica} \\ \text{y positiva definida} \end{array} \Rightarrow \underbrace{A \mathbf{y} - \mathbf{b} = 0}_{A \mathbf{y} = \mathbf{b}}$$

Recuerde:

$$\text{Jacobi} \Rightarrow \mathbf{x}_{k+1} = \mathbf{x}_k + D^{-1} \cdot \mathbf{r}_k,$$

$$\text{G-S} \Rightarrow \mathbf{x}_{k+1} = \mathbf{x}_k + (L + D)^{-1} \cdot \mathbf{r}_k,$$

$$\text{SOR}(\omega) \Rightarrow \mathbf{x}_{k+1} = \mathbf{x}_k + (L + \frac{D}{\omega})^{-1} \cdot \mathbf{r}_k,$$

$$\Rightarrow \mathbf{x}_{k+1} = \mathbf{x}_k + \text{"Vector"}$$

$$= \mathbf{x}_k - \alpha_k \cdot \mathbf{d}_k$$

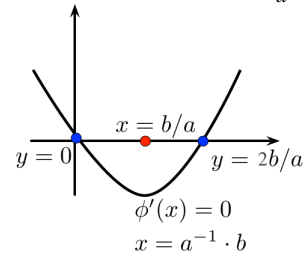
$$\mathbf{r}_k = \mathbf{b} - A \mathbf{x}_k$$

1D-sketch

$$\phi(x) = \frac{1}{2} a x^2 - x b$$

$$\phi'(y) = a y - b$$

$$\phi'(x) = 0 = a x - b \Rightarrow x = \frac{b}{a}$$



Primera opción : Definir \mathbf{d}_k en la dirección de máximo decrecimiento.

$$\Rightarrow \nabla \Phi(\mathbf{x}_k) = A \mathbf{x}_k - \mathbf{b} = -\mathbf{r}_k$$

$$\Rightarrow \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{r}_k$$

D.2.2. ¿Qué es y cómo se obtiene α_k ?

Nota: Por simplicidad se utilizará α en vez de α_k en el siguiente desarrollo.

$$\Phi(\mathbf{x}_{k+1}) = \frac{1}{2} (\mathbf{x}_k + \alpha \mathbf{r}_k)^T A (\mathbf{x}_k + \alpha \mathbf{r}_k) - (\mathbf{x}_k + \alpha \mathbf{r}_k)^T \mathbf{b}$$

donde $\Phi(\mathbf{x}_{k+1})$ es una función de α cuando \mathbf{x}_k y \mathbf{r}_k son conocidos, $\Rightarrow f(\alpha) = \Phi(\mathbf{x}_k + \alpha \cdot \mathbf{r}_k)$.

D.2.2.1. ¿Cómo se obtiene el “óptimo” de “ $f(\alpha)$ ”?

$$f'(\alpha) = 0$$

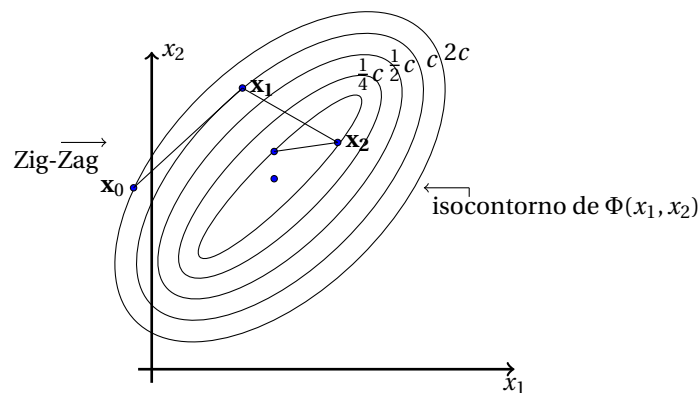
$$\Rightarrow \alpha = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_k^T A \mathbf{r}_k} \quad (\text{D.1})$$

D.2.2.2. Algoritmo

```

1   $\mathbf{x}_0 = \text{"dato"}$ 
2  for  $k$  in  $\text{range}(n)$  :
3       $\mathbf{r}_k = b - A \mathbf{x}_k$ 
4       $\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_k^T A \mathbf{r}_k}$ 
5       $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{r}_k$ 

```

D.2.2.3. Gráficamente

$$\Phi(x_1, x_2) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} x_1 & x_2 \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

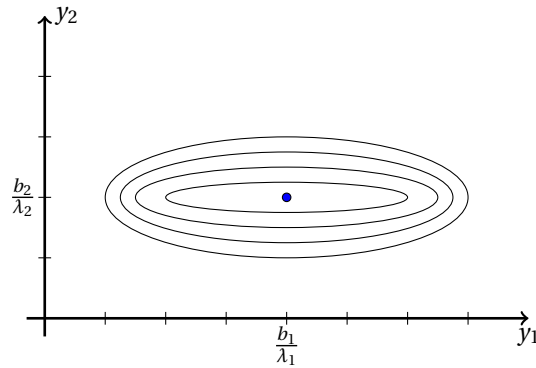
D.2.3. Convergencia en matrices simétricas y definida positiva

Sea A una matriz simétrica y positiva definida, entonces el método del gradiente descendente converge a la solución de $A\mathbf{x} = \mathbf{b}$, para cualquier \mathbf{x}_0 y \mathbf{b} , y

$$\|\mathbf{e}^{k+1}\|_A \leq \frac{\kappa_2(A) - 1}{\kappa_2(A) + 1} \|\mathbf{e}^k\|_A \quad \left| \quad \kappa(A) = \|A\| \|A^{-1}\| \right.$$

donde $\|\cdot\|_A$ es A -norma, i.e. $\|\mathbf{x}\|_A = \sqrt{\mathbf{x}^T A \mathbf{x}}$ y $\mathbf{e}^k = \mathbf{x}_k - \mathbf{x}$
Ejemplo:

$$\begin{aligned}
A &= \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \lambda_i > 0, 0 < \lambda_2 \leq \lambda_1, \mathbf{b} = \langle b_1, b_2 \rangle^T \\
\Rightarrow \Phi(\mathbf{y}) &= \frac{1}{2} \langle y_1, y_2 \rangle \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \langle y_1, y_2 \rangle \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \\
&= \frac{1}{2} (\lambda_1 y_1^2 + \lambda_2 y_2^2) - b_1 y_1 - b_2 y_2 = C \\
\Rightarrow &\left(\frac{y_1 - b_1/\lambda_1}{\sqrt{2/\lambda_1}} \right)^2 + \left(\frac{y_2 - b_2/\lambda_2}{\sqrt{2/\lambda_2}} \right)^2 = C + \frac{b_1^2}{2\lambda_1} + \frac{b_2^2}{2\lambda_2} \\
&\quad \uparrow \\
&\text{"ellipse"}
\end{aligned}$$



D.3. Método del Gradiente Conjugado

Recuerde del capítulo [D.2](#) que construimos la solución del sistema de ecuaciones lineales iterativamente de la siguiente forma:

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 + \alpha_0 \mathbf{r}_0 \\
\mathbf{x}_2 &= \mathbf{x}_1 + \alpha_1 \mathbf{r}^1 \\
&\vdots \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k \mathbf{r}_k
\end{aligned}$$

$$\begin{aligned} \Rightarrow \mathbf{x}_2 &= \mathbf{x}_0 + \alpha_0 \mathbf{r}_0 + \alpha_1 \mathbf{r}_1 \\ \mathbf{x}_2 &= \mathbf{x}_0 + \underbrace{\begin{bmatrix} \mathbf{r}_0 & | & \mathbf{r}_1 \end{bmatrix}}_{\text{Base}} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \\ &\vdots \\ \mathbf{x}_{k+1} &= \mathbf{x}_0 + \underbrace{\begin{bmatrix} \mathbf{r}_0 & | & \dots & | & \mathbf{r}_k \end{bmatrix}} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_k \end{bmatrix} \end{aligned}$$

“k” puede ser mayor que “n”,
i.e. la dimensión del espacio

¿Cuántos vectores “linealmente independientes” necesitamos para construir una base de \mathbb{R}^n ?

¿Necesitamos más de “n” iteraciones entonces?

¿Qué tal si entonces elegimos la dirección convenientemente? i.e. si usamos \mathbf{d}_k en vez de \mathbf{r}_k .

D.3.1. Derivación del Método del Gradiente Conjugado

El método del Gradiente Conjugado (CG) se utiliza para resolver sistemas de ecuaciones lineales, es decir: $A\mathbf{x} = \mathbf{b}$, donde la matriz $A \in \mathbb{R}^{n \times n}$ es simétrica ($A = A^T$) y positiva definida ($\mathbf{x}^T A \mathbf{x} > 0$ para $\mathbf{x} \neq \mathbf{0}$). En este caso se busca construir la solución \mathbf{x} como una combinación lineal de vectores en \mathbb{R}^n linealmente independiente, es decir:

$$\mathbf{x} = \mathbf{x}_0 + \sum_{i=0}^{n-1} \alpha_i \mathbf{d}_i. \quad (\text{D.2})$$

Donde definiremos la siguiente notación:

$$\mathbf{x} = \underbrace{\mathbf{x}_0 + \alpha_0 \mathbf{d}_0}_{\mathbf{x}_1} + \alpha_1 \mathbf{d}_1 + \alpha_2 \mathbf{d}_2 + \cdots + \alpha_{n-1} \mathbf{d}_{n-1},$$

$$\underbrace{\hspace{1.5cm}}_{\mathbf{x}_2}$$

$$\underbrace{\hspace{2.5cm}}_{\mathbf{x}_3}$$

$$\underbrace{\hspace{4.5cm}}_{\mathbf{x}_n}$$

donde obtenemos,

$$\begin{aligned}\mathbf{x}_1 &= \mathbf{x}_0 + \alpha_0 \mathbf{d}_0 \\ \mathbf{x}_2 &= \mathbf{x}_1 + \alpha_1 \mathbf{d}_1 \\ &\vdots \\ \mathbf{x}_n &= \mathbf{x}_{n-1} + \alpha_{n-1} \mathbf{d}_{n-1},\end{aligned}$$

las cuales se pueden expresar de la siguiente forma:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k, \quad k \in \{0, 1, \dots, n-1\}. \quad (\text{D.3})$$

Multiplicando (D.3) por $-A$ por la izquierda,

$$-A\mathbf{x}_{k+1} = -A\mathbf{x}_k - \alpha_k A\mathbf{d}_k$$

y sumando \mathbf{b} en ambos lados obtenemos,

$$\begin{aligned}\underbrace{\mathbf{b} - A\mathbf{x}_{k+1}}_{\mathbf{r}_{k+1}} &= \underbrace{\mathbf{b} - A\mathbf{x}_k}_{\mathbf{r}_k} - \alpha_k A\mathbf{d}_k \\ \mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_k A\mathbf{d}_k,\end{aligned} \quad (\text{D.4})$$

por lo tanto hemos encontrado una relación entre los vectores residuales en función de las direcciones \mathbf{d}_k . Sin embargo, aún nos falta por definir como encontraremos los α_k y \mathbf{d}_k que se necesitan en las ecuaciones (D.3) y (D.4).

Una primera alternativa a elegir α_k es similar a la utilizada en la ecuación (D.1), pero teniendo la salvedad que ahora la dirección de búsqueda es \mathbf{d}_k y no \mathbf{r}_k como se utilizó en el caso del Gradiente Descendente. En este caso utilizaremos otro camino, el cual consiste en reemplazar (D.2) en $A\mathbf{x} = \mathbf{b}$ obteniendo,

$$\begin{aligned}A \left(\mathbf{x}_0 + \sum_{i=0}^{n-1} \alpha_i \mathbf{d}_i \right) &= \mathbf{b} \\ A\mathbf{x}_0 + \sum_{i=0}^{n-1} \alpha_i A\mathbf{d}_i &= \mathbf{b}.\end{aligned}$$

Re-escribiendo obtenemos,

$$\alpha_0 A\mathbf{d}_0 + \alpha_1 A\mathbf{d}_1 + \cdots + \alpha_{n-1} A\mathbf{d}_{n-1} = \mathbf{b} - A\mathbf{x}_0 = \mathbf{r}_0. \quad (\text{D.5})$$

Ahora es conveniente incluir la definición de A -ortogonalidad:

Def 21. *A -ortogonalidad:* Sea $\mathbf{u} \in \mathbb{R}^n$ no nulo, $\mathbf{v} \in \mathbb{R}^n$ no nulo y, $A \in \mathbb{R}^{n \times n}$ simétrica y positiva definida, entonces decimos que \mathbf{u} y \mathbf{v} son A -ortogonales (o conjugados) si $\langle \mathbf{u}, \mathbf{v} \rangle_A = 0$, donde $\langle \mathbf{u}, \mathbf{v} \rangle_A = \mathbf{u}^T A \mathbf{v}$.

Recuerde que la noción tradicional de ortogonalidad implica $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T \mathbf{v} = \mathbf{u}^T I \mathbf{v} = 0$, i.e. donde la matriz utilizada es la identidad $I \in \mathbb{R}^{n \times n}$.

Multiplicando la ecuación (D.5) por \mathbf{d}_0^T por la izquierda obtenemos,

$$\alpha_0 \mathbf{d}_0^T A \mathbf{d}_0 + \alpha_1 \mathbf{d}_0^T A \mathbf{d}_1 + \cdots + \alpha_{n-1} \mathbf{d}_0^T A \mathbf{d}_{n-1} = \mathbf{d}_0^T \mathbf{r}_0.$$

Entonces si consideramos ahora la definición 21 de A -ortogonalidad para el conjunto de vectores $\{\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{n-1}\}$ obtenemos,

$$\begin{aligned} \alpha_0 \mathbf{d}_0^T A \mathbf{d}_0 + \alpha_1 \underbrace{\mathbf{d}_0^T A \mathbf{d}_1}_0 + \cdots + \alpha_{n-1} \underbrace{\mathbf{d}_0^T A \mathbf{d}_{n-1}}_0 &= \mathbf{d}_0^T \mathbf{r}_0, \\ \alpha_0 \mathbf{d}_0^T A \mathbf{d}_0 &= \mathbf{d}_0^T \mathbf{r}_0, \end{aligned}$$

por lo cual obtenemos $\alpha_0 = \frac{\mathbf{d}_0^T \mathbf{r}_0}{\mathbf{d}_0^T A \mathbf{d}_0}$, lo que es equivalente al valor obtenido por (D.1) considerando la dirección \mathbf{d}_0 en este caso. Antes de pasar al caso general, debemos hacer la siguiente simplificación. Considerando que uno ya obtiene el valor de α_0 y conoce el vector \mathbf{d}_0 uno podría mover al lado derecho de la ecuación lo conocido, es decir obtiene lo siguiente,

$$\alpha_1 A \mathbf{d}_1 + \cdots + \alpha_{n-1} A \mathbf{d}_{n-1} = \underbrace{\mathbf{r}_0 - \alpha_0 A \mathbf{d}_0}_{\mathbf{r}_1}.$$

es decir, va obteniendo vector residual de la siguiente iteración en el lado derecho de la ecuación, ver ecuación (D.4). En este caso, utilizando la A -ortogonalidad, obtendríamos $\alpha_1 = \frac{\mathbf{d}_1^T \mathbf{r}_1}{\mathbf{d}_1^T A \mathbf{d}_1}$, entonces en el caso general se obtiene,

$$\alpha_k = \frac{\mathbf{d}_k^T \mathbf{r}_k}{\mathbf{d}_k^T A \mathbf{d}_k}. \quad (\text{D.6})$$

Nótese que también se podría usar $\alpha_k = \frac{\mathbf{d}_k^T \mathbf{r}_0}{\mathbf{d}_k^T A \mathbf{d}_k}$, sin embargo por ahora se prefiere utilizar (D.6) dado que necesitaremos \mathbf{r}_k posteriormente. ¿Hay alguna diferencia Matemática en usar α_k como $\frac{\mathbf{d}_k^T \mathbf{r}_0}{\mathbf{d}_k^T A \mathbf{d}_k}$ o $\frac{\mathbf{d}_k^T \mathbf{r}_k}{\mathbf{d}_k^T A \mathbf{d}_k}$? ¿Hay alguna diferencia Computacional¹ en usar α_k como $\frac{\mathbf{d}_k^T \mathbf{r}_0}{\mathbf{d}_k^T A \mathbf{d}_k}$ o $\frac{\mathbf{d}_k^T \mathbf{r}_k}{\mathbf{d}_k^T A \mathbf{d}_k}$?

Hasta este punto solo hemos obtenido α_k , sin embargo también necesitamos definir como obtendremos \mathbf{d}_k , es decir, los vectores linealmente independientes $\{\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{n-1}\}$ que además deben ser A -ortogonales!

Para obtener los vectores \mathbf{d}_k se propone la siguiente ecuación:

$$\mathbf{d}_{k+1} = \mathbf{r}_{k+1} - \beta_k \mathbf{d}_k, \quad (\text{D.7})$$

donde el coeficiente β_k se obtiene nuevamente utilizando la A -ortogonalidad, es decir multiplicando la ecuación (D.7) por $\mathbf{d}_k^T A$ por la izquierda,

$$\underbrace{\mathbf{d}_k^T A \mathbf{d}_{k+1}}_0 = \mathbf{d}_k^T A \mathbf{r}_{k+1} - \beta_k \mathbf{d}_k^T A \mathbf{d}_k,$$

por lo tanto,

$$\beta_k = \frac{\mathbf{d}_k^T A \mathbf{r}_{k+1}}{\mathbf{d}_k^T A \mathbf{d}_k}. \quad (\text{D.8})$$

Por lo cual ahora podemos construir nuestra primera versión del algoritmo,

¹¡Le sugiero implementarlo y ver qué ocurre! En realidad ya está implementado en los jupyter notebooks del curso, solo debe modificarlo un poco.

```

1  $\mathbf{x}_0 = \text{"dato"}$ 
2  $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ 
3  $\mathbf{d}_0 = \mathbf{r}_0$ 
4 for  $k$  in  $\text{range}(0, n)$  :
5      $\alpha_k = \mathbf{d}_k^T \mathbf{r}_k / \mathbf{d}_k^T A \mathbf{d}_k \longrightarrow A\mathbf{y} = \text{afun}(\mathbf{y})$ 
6      $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$ 
7      $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A \mathbf{d}_k = \mathbf{b} - A\mathbf{x}_{k+1}$ 
8      $\beta_k = \mathbf{d}_k^T A \mathbf{r}_{k+1} / \mathbf{d}_k^T A \mathbf{d}_k$ 
9      $\mathbf{d}_{k+1} = \mathbf{r}_{k+1} - \beta_k \mathbf{d}_k$ 

```

Notar algo muy importante que hasta ahora no se había definido, esto es, ¿como se define o inicializa \mathbf{d}_0 ? Answer: Look back at the algorithm!.

D.3.2. Análisis de la A-ortogonalidad

Para que todo el desarrollo de la sección D.3.1 se sustente, necesitamos demostrar que:

$$\mathbf{d}_j^T A \mathbf{d}_{k+1} = 0, \quad \text{para } j \in \{0, 1, \dots, k\}, \text{ y } k \in \{0, 1, \dots, n-2\}, \quad (\text{D.9})$$

donde $\mathbf{d}_{k+1} = \mathbf{r}_{k+1} - \beta_k \mathbf{d}_k$, ver ecuación (D.7), y $\beta_k = \frac{\mathbf{d}_k^T A \mathbf{r}_{k+1}}{\mathbf{d}_k^T A \mathbf{d}_k}$, ver ecuación (D.8).

Para demostrar (D.9) procederemos por inducción en k . El caso base es para $k = 0$, es decir necesitamos demostrar que $\mathbf{d}_0^T A \mathbf{d}_1 = 0$. Multiplicando por la izquierda la ecuación (D.7) por $\mathbf{d}_0^T A$ y considerando $k = 0$ obtenemos,

$$\mathbf{d}_0^T A \mathbf{d}_1 = \mathbf{d}_0^T A \mathbf{r}_1 - \beta_0 \mathbf{d}_0^T A \mathbf{d}_0,$$

reemplazando ahora β_0 en la ecuación anterior. Es decir, utilizar la definición de β_k en ecuación (D.8) con $k = 0$, i.e. $\beta_0 = \frac{\mathbf{d}_0^T A \mathbf{r}_1}{\mathbf{d}_0^T A \mathbf{d}_0}$ obtenemos,

$$\begin{aligned} \mathbf{d}_0^T A \mathbf{d}_1 &= \mathbf{d}_0^T A \mathbf{r}_1 - \underbrace{\beta_0}_{\frac{\mathbf{d}_0^T A \mathbf{r}_1}{\mathbf{d}_0^T A \mathbf{d}_0}} \mathbf{d}_0^T A \mathbf{d}_0 \\ \mathbf{d}_0^T A \mathbf{d}_1 &= \mathbf{d}_0^T A \mathbf{r}_1 - \frac{\mathbf{d}_0^T A \mathbf{r}_1}{\mathbf{d}_0^T A \mathbf{d}_0} \mathbf{d}_0^T A \mathbf{d}_0 \\ \mathbf{d}_0^T A \mathbf{d}_1 &= \mathbf{d}_0^T A \mathbf{r}_1 - \mathbf{d}_0^T A \mathbf{r}_1 \\ \mathbf{d}_0^T A \mathbf{d}_1 &= 0. \end{aligned}$$

Por lo tanto se cumple el caso base $\mathbf{d}_0^T A \mathbf{d}_1 = 0$.

Ahora asumamos que el conjunto de vectores $\{\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{k-1}\}$ son A-ortogonales. Considere ahora la ecuación (D.4) donde la multiplicamos por la izquierda por \mathbf{d}_j^T ,

$$\mathbf{d}_j^T \mathbf{r}_k = \mathbf{d}_j^T \mathbf{r}_{k-1} - \alpha_{k-1} \mathbf{d}_j^T A \mathbf{d}_{k-1}, \quad (\text{D.10})$$

donde tenemos 2 casos: (i) $j = k-1$ y (ii) $0 \leq j < k-1$. Para el primer caso, (i), obtenemos,

$$\begin{aligned} \mathbf{d}_{k-1}^T \mathbf{r}_k &= \mathbf{d}_{k-1}^T \mathbf{r}_{k-1} - \alpha_{k-1} \mathbf{d}_{k-1}^T A \mathbf{d}_{k-1} \\ &= \mathbf{d}_{k-1}^T \mathbf{r}_{k-1} - \frac{\mathbf{d}_{k-1}^T \mathbf{r}_{k-1}}{\mathbf{d}_{k-1}^T A \mathbf{d}_{k-1}} \mathbf{d}_{k-1}^T A \mathbf{d}_{k-1} \\ &= \mathbf{d}_{k-1}^T \mathbf{r}_{k-1} - \mathbf{d}_{k-1}^T \mathbf{r}_{k-1} \\ &= 0. \end{aligned}$$

Para el segundo caso, (ii), obtenemos,

$$\begin{aligned}\mathbf{d}_j^T \mathbf{r}_k &= \mathbf{d}_j^T \mathbf{r}_{k-1} - \alpha_{k-1} \underbrace{\mathbf{d}_j^T A \mathbf{d}_{k-1}}_{0 \text{ por } A\text{-ortogonalidad}} \\ &= \mathbf{d}_j^T \mathbf{r}_{k-1},\end{aligned}$$

reemplazando por la definición de \mathbf{r}_{k-1} obtenemos,

$$\begin{aligned}\mathbf{d}_j^T \mathbf{r}_k &= \mathbf{d}_j^T \mathbf{r}_{k-1} \\ &= \mathbf{d}_j^T (\mathbf{r}_{k-2} - \alpha_{k-2} A \mathbf{d}_{k-2}) \\ &= \mathbf{d}_j^T \mathbf{r}_{k-2} - \alpha_{k-2} \underbrace{\mathbf{d}_j^T A \mathbf{d}_{k-2}}_0 \\ &= \mathbf{d}_j^T \mathbf{r}_{k-2},\end{aligned}$$

siguiendo el mismo desarrollo obtenemos la misma recurrencia hasta que lleguemos al $j+1$ -ésimo residuo,

$$\begin{aligned}\mathbf{d}_j^T \mathbf{r}_k &= \mathbf{d}_j^T \mathbf{r}_{k-1} \\ &= \mathbf{d}_j^T \mathbf{r}_{k-2} \\ &\vdots \\ &= \mathbf{d}_j^T \mathbf{r}_{j+1} \\ &= \mathbf{d}_j^T (\mathbf{r}_j - \alpha_j A \mathbf{d}_j),\end{aligned}$$

al reemplazar por la definición de α_j obtenemos,

$$\begin{aligned}\mathbf{d}_j^T \mathbf{r}_k &= \mathbf{d}_j^T (\mathbf{r}_j - \alpha_j A \mathbf{d}_j) \\ &= \mathbf{d}_j^T \mathbf{r}_j - \alpha_j \mathbf{d}_j^T A \mathbf{d}_j \\ &= \mathbf{d}_j^T \mathbf{r}_j - \frac{\mathbf{d}_j^T \mathbf{r}_j}{\mathbf{d}_j^T A \mathbf{d}_j} \mathbf{d}_j^T A \mathbf{d}_j \\ &= \mathbf{d}_j^T \mathbf{r}_j - \mathbf{d}_j^T \mathbf{r}_j \\ &= 0.\end{aligned}$$

Por lo tanto,

$$\mathbf{d}_j^T \mathbf{r}_k = 0, \quad j \in \{0, 1, \dots, k-1\}. \quad (\text{D.11})$$

Note que la ecuación (D.11) es válida para \mathbf{r}_k , ahora consideremos \mathbf{r}_{k+1} , i.e. obtengamos el producto interno entre \mathbf{d}_j y \mathbf{r}_{k+1} ,

$$\mathbf{d}_j^T \mathbf{r}_{k+1} = \underbrace{\mathbf{d}_j^T \mathbf{r}_k}_{0 \text{ por (D.11)}} - \alpha_k \underbrace{\mathbf{d}_j^T A \mathbf{d}_k}_0, \quad \text{para } j \in \{0, 1, \dots, k-1\} \quad (\text{D.12})$$

Por lo tanto podemos concluir que el conjunto $\{\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{k-1}\}$ es ortogonal a \mathbf{r}_{k+1} .

Otra relación interesante es entre los vectores residuales. Considere el producto interno entre \mathbf{r}_k y j -ésimo vector \mathbf{d}_j , ver ecuación (D.7),

$$\begin{aligned}\underbrace{\mathbf{r}_k^T \mathbf{d}_j}_{0 \text{ por (D.11)}} &= \mathbf{r}_k^T \mathbf{r}_j - \beta_{j-1} \underbrace{\mathbf{r}_k^T \mathbf{d}_{j-1}}_{0 \text{ por (D.11)}} \\ &= \mathbf{r}_k^T \mathbf{r}_j.\end{aligned} \quad (\text{D.13})$$

Por lo tanto $\mathbf{r}_k^T \mathbf{r}_j = 0$ para $j \in \{0, 1, \dots, k-1\}$, i.e. los vectores residuales son ortogonales.

Nos falta el último paso, determinar si efectivamente \mathbf{d}_{k+1} es A -ortogonal a \mathbf{d}_j para $j \in \{0, 1, \dots, k\}$. Multipliquemos entonces \mathbf{d}_{k+1} (ver ecuación (D.7)) por la izquierda por $\mathbf{d}_j^T A$, lo que nos da:

$$\mathbf{d}_j^T A \mathbf{d}_{k+1} = \mathbf{d}_j^T A \mathbf{r}_{k+1} - \beta_k \mathbf{d}_j^T A \mathbf{d}_k.$$

Nuevamente tenemos 2 casos: (i) $j = k$ y (ii) $j < k$. Para el primer caso, (i), obtenemos:

$$\begin{aligned} \mathbf{d}_k^T A \mathbf{d}_{k+1} &= \mathbf{d}_k^T A \mathbf{r}_{k+1} - \beta_k \mathbf{d}_k^T A \mathbf{d}_k \\ &= \mathbf{d}_k^T A \mathbf{r}_{k+1} - \frac{\mathbf{d}_k^T A \mathbf{r}_{k+1}}{\mathbf{d}_k^T A \mathbf{d}_k} \mathbf{d}_k^T A \mathbf{d}_k \\ &= \mathbf{d}_k^T A \mathbf{r}_{k+1} - \mathbf{d}_k^T A \mathbf{r}_{k+1} \\ &= 0. \end{aligned}$$

Ahora, para el segundo caso, (ii), obtenemos,

$$\begin{aligned} \mathbf{d}_j^T A \mathbf{d}_{k+1} &= \mathbf{d}_j^T A \mathbf{r}_{k+1} - \beta_k \underbrace{\mathbf{d}_j^T A \mathbf{d}_k}_{=0} \\ &= \mathbf{d}_j^T A \mathbf{r}_{k+1}. \end{aligned} \tag{D.14}$$

Donde notamos que $\mathbf{d}_j^T A = (A \mathbf{d}_j)^T$, y considerando la ecuación (D.4) para el caso del $j+1$ -ésimo vector residual,

$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j A \mathbf{d}_j,$$

podemos despejar $A \mathbf{d}_j$ en función de los vectores residuales \mathbf{r}_j y \mathbf{r}_{j+1} con $\alpha_j \neq 0$ ²,

$$A \mathbf{d}_j = \frac{\mathbf{r}_j}{\alpha_j} - \frac{\mathbf{r}_{j+1}}{\alpha_j}. \tag{D.15}$$

Reemplazando la ecuación (D.15) en la ecuación (D.14) obtenemos,

$$\mathbf{d}_j^T A \mathbf{d}_{k+1} = \mathbf{d}_j^T A \mathbf{r}_{k+1} \tag{D.16}$$

$$= (A \mathbf{d}_j)^T \mathbf{r}_{k+1} \tag{D.17}$$

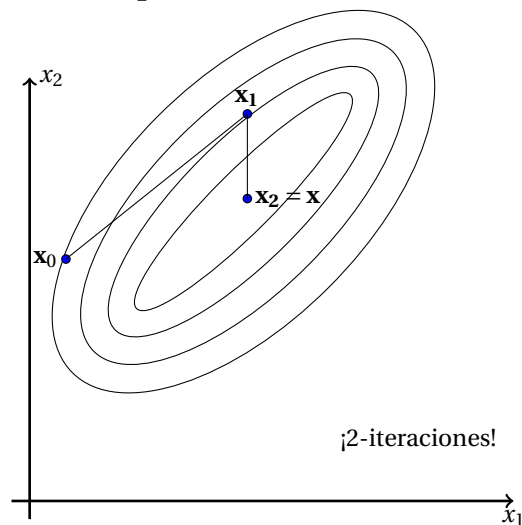
$$= \left(\frac{\mathbf{r}_j}{\alpha_j} - \frac{\mathbf{r}_{j+1}}{\alpha_j} \right)^T \mathbf{r}_{k+1} \tag{D.18}$$

$$= \frac{\mathbf{r}_j^T \mathbf{r}_{k+1}}{\alpha_j} - \frac{\mathbf{r}_{j+1}^T \mathbf{r}_{k+1}}{\alpha_j}, \tag{D.19}$$

por la ecuación (D.13) sabemos que los vectores residuales son ortogonales, por lo tanto $\mathbf{d}_j^T A \mathbf{d}_{k+1} = 0$ para $j \in \{0, 1, \dots, k\}$, y esto completa la demostración.

²¿Qué pasa en el caso de $\alpha_j = 0$?

D.3.3. Gradiente Conjugado - Interpretación Gráfica



D.3.4. Convergencia del Algoritmo del Gradiente Conjugado para una matriz simétrica y definida positiva

Thm 28. Sea A una matriz simétrica y positiva definida. El método del gradiente conjugado para resolver $Ax = b$ converge a lo más en “ n ” pasos usando aritmética exacta.

D.3.5. Gradiente Conjugado - Versión 2

¿Podría usted demostrar que la siguiente implementación es equivalente a la anterior? Warning: Watch out for the sign used for β_k .

```

1  $\mathbf{x}_0 = \text{"dato"}$ 
2  $\mathbf{d}_0 = \mathbf{r}_0 = \mathbf{b} - A \cdot \mathbf{x}_0$ 
3 for  $k$  in  $\text{range}(n)$  :
4     if ( $\|\mathbf{r}_k\| == 0$ ) :
5         break
6      $\alpha_k = \frac{\mathbf{r}_k^T \cdot \mathbf{r}_k}{\mathbf{d}_k^T A \mathbf{d}_k}$ 
7      $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$ 
8      $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A \mathbf{d}_k$ 
9      $\beta_k = \frac{\mathbf{r}_{k+1}^T \cdot \mathbf{r}_{k+1}}{\mathbf{r}_k^T \cdot \mathbf{r}_k}$ 
10     $\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k$ 

```

- ¿Cómo se puede mejorar esta propuesta?
- ¿Qué convendría hacer al momento de la implementación?

D.3.5.1. Ejemplo

p.129, 14.(a)

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$$

$$\mathbf{r}_0 = \mathbf{b} - A \mathbf{x}_0 = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$$

$$\mathbf{d}_0 = \mathbf{r}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\alpha_0 = \mathbf{d}_0^T \cdot \mathbf{r}_0 / \mathbf{d}_0^T A \mathbf{d}_0$$

$$= \frac{\begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}}{\begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}}$$

$$= \frac{\boxed{1}}{\begin{bmatrix} 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} -1 \\ 2 \end{bmatrix}}_{A \mathbf{d}_0}} \rightarrow \mathbf{r}_0^T \mathbf{r}_0$$

$$\boxed{\alpha_0 = \frac{1}{2}}$$

$$\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0 \mathbf{d}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 1/2 \end{bmatrix}$$

$$\mathbf{r}_1 = \mathbf{r}_0 - \alpha_0 A \mathbf{d}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

$$\mathbf{r}_1 = \begin{bmatrix} 1/2 \\ 0 \end{bmatrix}$$

$$\beta_0 = \frac{\mathbf{r}_1^T \mathbf{r}_1}{\mathbf{r}_0^T \mathbf{r}_0} = \frac{\begin{bmatrix} 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/2 \\ 0 \end{bmatrix}}{1}$$

$$\boxed{\beta_0 = 1/4}$$

$$\mathbf{d}_1 = \mathbf{r}_1 + \beta_0 \mathbf{d}_0$$

$$= \begin{bmatrix} 1/2 \\ 0 \end{bmatrix} + \frac{1}{4} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/4 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}}_A \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_x = \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_b$$

 $k = 1$

$$\alpha_1 = \frac{\mathbf{r}_1^T \mathbf{r}_1}{\mathbf{d}_1^T A \mathbf{d}_1} = \frac{1/4}{\begin{bmatrix} 1/2 & 1/4 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 1/2 \\ 1/4 \end{bmatrix}}_{\begin{bmatrix} 1/4 \\ 0 \end{bmatrix}}}$$

$$\boxed{\alpha_1 = \frac{1/4}{1/8} = 2}$$

$$\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{d}_1 = \begin{bmatrix} 0 \\ 1/2 \end{bmatrix} + 2 \begin{bmatrix} 1/2 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{r}_2 = \mathbf{r}_1 - \alpha_1 A \mathbf{d}_1$$

$$= \begin{bmatrix} 1/2 \\ 0 \end{bmatrix} - 2 \begin{bmatrix} 1/4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

↑
; residual nulo !

$$\beta_1 = 0$$

$$\mathbf{d}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0 \begin{bmatrix} 1/2 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\therefore \mathbf{x} = \mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \& \mathbf{r}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

¡CG terminó en sólo 2 iteraciones!

Apéndice E

Computación de Pesos y Nodos de la Cuadratura Gaussiana

Este anexo ha sido preparado por los profesores Ariel Sanhueza y Pablo Ibarra en Abril de 2019. El anexo ha sido actualizado en Junio de 2022 por el Prof. Claudio Torres manteniendo el espíritu de la formulación inicial.

E.1. Definiciones y teoremas bases

Para mayor detalle en cuanto a las definiciones y teoremas que serán expuestos en esta sección, por favor revisar [2, 5, 3]. Una *cuadratura* tiene la estructura:

$$I(f) = \int_a^b f(x) dx \approx Q(f) = \int_a^b \sum_{j=1}^n f(x_j) L_j(x) dx = \sum_{j=1}^n w_j f(x_j), \quad (\text{E.1})$$

con

$$w_j = \int_a^b L_j(x) dx.$$

En la notación anterior el funcional $I(f)$ corresponde al operador de integración en el intervalo $[a, b]$ y el funcionar $Q(f)$ corresponde a la cuadratura numérica $\sum_{j=1}^n w_j f(x_j)$, en ambos casos se están aplicando a la función $f(x)$.

Thm 29. Sea $\{p_k(x)\}_{k=0}^n$ un conjunto de polinomios ortogonales en $[a, b]$ con respecto al producto interno:

$$\langle f, g \rangle = \int_a^b f(x) g(x) dx.$$

Sean $\{x_k\}_{k=1}^n$ las n raíces de $p_n(x)$. Entonces la cuadratura (E.1) es exacta para polinomios de grado $2n - 1$ o menor.

Notar que el teorema anterior es una variante del teorema 20, la demostración se puede obtener en [3], capítulo 5, teorema 5.9.

E.2. Nodos de la Cuadratura Gaussiana

Consideremos la recurrencia obtenida al aplicar Lanczos a la base polinómica mónica, la cual es:

$$x p_n(x) = \beta_{n-1} p_{n-1}(x) + \alpha_n p_n(x) + \beta_n p_{n+1}(x). \quad (\text{E.2})$$

Una posible explicación de esta recurrencia es considerar que uno está construyendo una secuencia de polinomios ortogonales, y para encontrar el polinomio $p_{n+1}(x)$ uno multiplica por x el polinomio anterior y luego le resta las proyecciones sobre $p_{n-1}(x)$ y $p_n(x)$. Este tipo de procedimiento lo vimos anteriormente en la factorización QR con el algoritmo de Gram-Schmidt.

Utilizando la recurrencia (E.2), comenzando por $n = 0$ y tomando $p_{-1}(x) = 0$, tenemos:

$$\begin{aligned} x p_0(x) &= \alpha_0 p_0(x) + \beta_0 p_1(x), \\ x p_1(x) &= \beta_0 p_0(x) + \alpha_1 p_1(x) + \beta_1 p_2(x), \\ x p_2(x) &= \beta_1 p_1(x) + \alpha_2 p_2(x) + \beta_2 p_3(x), \\ &\vdots \\ x p_{n-1}(x) &= \beta_{n-2} p_{n-2}(x) + \alpha_{n-1} p_{n-1}(x) + \beta_{n-1} p_n(x). \end{aligned}$$

Si los polinomios $p_n(x)$ son ortogonales, entonces tenemos que

$$\alpha_n = \frac{\langle p_n, x p_n \rangle}{\langle p_n, p_n \rangle}. \quad (\text{E.3})$$

$$\beta_{n-1} = \frac{\langle p_n, x p_{n-1} \rangle}{\langle p_{n-1}, p_{n-1} \rangle}, \quad (\text{E.4})$$

Definiendo:

$$\mathbf{p}(x) = \begin{pmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_{n-1}(x) \end{pmatrix}, T_n = \begin{pmatrix} \alpha_0 & \beta_0 & 0 & 0 & \cdots & 0 \\ \beta_0 & \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_1 & \alpha_2 & \beta_2 & \cdots & 0 \\ 0 & 0 & \beta_2 & \alpha_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \alpha_{n-1} \end{pmatrix},$$

entonces tenemos que

$$x \mathbf{p}(x) = T_n \mathbf{p}(x) + \beta_{n-1} p_n(x) \mathbf{e}_{n-1}. \quad (\text{E.5})$$

Notar que $p_n(x)$ es un polinomio de grado n , por lo que tiene n raíces. Sea x_j la j -ésima raíz de $p_n(x)$, vemos que:

$$x_j \mathbf{p}(x_j) = T_n \mathbf{p}(x_j) + \beta_{n-1} p_n(x_j) \mathbf{e}_n = T_n \mathbf{p}(x_j),$$

dado que $p_n(x_j) = 0$. Así, los valores propios del sistema anterior son los nodos de la cuadratura Gaussiana. Esto se obtiene de la escritura de la ecuación anterior de la siguiente forma,

$$T_n \mathbf{p}(x_j) = x_j \mathbf{p}(x_j), \quad (\text{E.6})$$

donde claramente notamos que x_j es un valor propio con vector propio $\mathbf{p}(x_j)$ de T_n .

E.3. Pesos de la Cuadratura Gaussiana

Para encontrar los pesos w_j , debemos considerar que:

1. Cuadratura Gaussiana es exacta para $p_0(x), \dots, p_{n-1}(x)$, pues son polinomios de grado menor a $2n - 1$.
2. $\mathbf{p}(x_j) = (p_0(x_j), p_1(x_j), \dots, p_{n-1}(x_j))^T$ es el vector propio.
3. $p_0(x)$ es un polinomio constante.
4. Los polinomios $\{p_k\}$ son ortonormales.

5. Los polinomios unitarios se puede definir con una versión alternativa de la formula de Rodrigues¹ de la siguiente forma:

$$p_n(x) = \sqrt{\frac{2n+1}{2}} \frac{1}{2^n n!} \frac{d^n}{dx^n} [(x^2 - 1)^n]. \quad (\text{E.7})$$

Notar que el único cambio es que se agregó el coeficiente $\sqrt{\frac{2n+1}{2}}$ para asegurar que sean unitarios.

6. Por simplicidad trabajaremos en el intervalo $[-1, 1]$.

Así, considerando el intervalo $[-1, 1]$, tenemos que:

$$\langle p_i, p_k \rangle = \int_{-1}^1 p_i(x) p_k(x) dx = \delta_{ik},$$

donde δ_{ik} indica la función Kronecker delta. Como $\deg(p_i(x) \cdot p_k(x)) \leq 2n - 1$, para $i, k \in \{0, \dots, n-1\}$, entonces por el Teorema 29 la cuadratura es exacta para el producto:

$$\delta_{ik} = \langle p_i, p_k \rangle = \int_{-1}^1 p_i(x) p_k(x) dx = \sum_{j=0}^n p_i(x_j) p_k(x_j) w_j.$$

Sea $W = \text{diag}(w_0, w_1, \dots, w_{n-1})$ y $\mathbf{p}_k = (p_k(x_0), p_k(x_1), \dots, p_k(x_{n-1}))^T$, entonces:

$$\delta_{ik} = \langle \mathbf{p}_i, W \mathbf{p}_k \rangle.$$

Ahora, escribiendo de forma matricial todas las combinaciones de productos obtenemos efectivamente la matriz identidad I , en particular obtenemos,

$$\begin{aligned} I &= \begin{bmatrix} \langle \mathbf{p}_0, W \mathbf{p}_0 \rangle & \langle \mathbf{p}_0, W \mathbf{p}_1 \rangle & \dots & \langle \mathbf{p}_0, W \mathbf{p}_{n-1} \rangle \\ \langle \mathbf{p}_1, W \mathbf{p}_0 \rangle & \langle \mathbf{p}_1, W \mathbf{p}_1 \rangle & \dots & \langle \mathbf{p}_1, W \mathbf{p}_{n-1} \rangle \\ \vdots & \vdots & \dots & \vdots \\ \langle \mathbf{p}_{n-1}, W \mathbf{p}_0 \rangle & \langle \mathbf{p}_{n-1}, W \mathbf{p}_1 \rangle & \dots & \langle \mathbf{p}_{n-1}, W \mathbf{p}_{n-1} \rangle \end{bmatrix}, \\ &= Q^T [W \mathbf{p}_0 | W \mathbf{p}_1 | \dots | W \mathbf{p}_{n-1}], \\ &= Q^T W Q. \end{aligned}$$

con $Q = [\mathbf{p}_0 | \mathbf{p}_1 | \dots | \mathbf{p}_{n-1}]$. Así:

$$I = Q^T W Q \iff W = (Q^T)^{-1} Q^{-1} = (Q Q^T)^{-1},$$

entonces,

$$W^{-1} = Q Q^T$$

$$\text{diag}\left(\frac{1}{w_0}, \frac{1}{w_1}, \dots, \frac{1}{w_{n-1}}\right) = [\mathbf{p}_0 \quad \mathbf{p}_1 \quad \dots \quad \mathbf{p}_{n-1}] \begin{bmatrix} \mathbf{p}_0^T \\ \mathbf{p}_1^T \\ \vdots \\ \mathbf{p}_{n-1}^T \end{bmatrix}.$$

Ahora, notando que el resultado del producto entre $Q Q^T$ es una matriz diagonal, necesitamos obtener la k -ésima fila de Q o, equivalentemente, la k -ésima columna de Q^T , que contienen efectivamente los mismos términos! Por lo tanto, recordando la definición anterior: $\mathbf{p}(x_k) = (p_0(x_k), p_1(x_k), \dots, p_{n-1}(x_k))$. Entonces, podemos concluir que,

$$\frac{1}{w_k} = \langle \mathbf{p}(x_k), \mathbf{p}(x_k) \rangle = \|\mathbf{p}(x_k)\|_2^2 \quad (\text{E.8})$$

¹https://en.wikipedia.org/wiki/Rodrigues%27_formula

Por otro lado, sea \mathbf{v}_k un vector propio de T_n , entonces \mathbf{v}_k puede escribirse como un escalamiento de $\mathbf{p}(x_k)$, es decir $c\mathbf{p}(x_k)$. Sin embargo conocemos de ecuación (E.7) que,

$$p_0(x) = \frac{1}{\sqrt{2}}$$

Por lo que podemos obtener c de la expresión anterior, es decir tenemos la siguiente ecuación,

$$(\mathbf{v}_k)_1 = c p_0(x_k) = c \frac{1}{\sqrt{2}}.$$

Por lo tanto $c = \sqrt{2}(\mathbf{v}_k)_1$. Entonces tenemos que:

$$\mathbf{p}(x_k) = \frac{1}{c} \mathbf{v}_k = \frac{1}{\sqrt{2}(\mathbf{v}_k)_1} \mathbf{v}_k,$$

Por lo que podemos explícitamente obtener la norma al cuadrado de $\mathbf{p}(x_k)$, es decir,

$$\|\mathbf{p}(x_k)\|_2^2 = \frac{1}{2(\mathbf{v}_k)_1^2} \|\mathbf{v}_k\|_2^2.$$

Finalmente, considerando la ecuación (E.8) llegamos al resultado final para obtener los pesos de la cuadratura, Por lo que el peso es:

$$w_k = \frac{1}{\|\mathbf{p}(x_k)\|_2^2} = 2 \frac{(\mathbf{v}_k)_1^2}{\|\mathbf{v}_k\|_2^2}.$$

Por completitud repetimos un extracto del algoritmo 14 para obtener los nodos y pesos de la cuadratura Gaussiana, ver algoritmo 19. Notar que en la implementación, en la línea 9 no se divide por la norma del vector propio porque en este caso la librería utilizada entrega los vectores propios de forma unitaria, es decir, tienen norma 1, ver [numpy.linalg.eigh](#).

```

1 def gaussian_nodes_and_weights(m):
2     if m==1:
3         return np.array([1]), np.array([2])
4     # Why an eigenvalue problems is related to the roots of the
      Legendre polynomial mentioned before? (This is a tricky
      question!)
5     beta = .5 / np.sqrt( 1. - (2.*np.arange(1.,m))**(-2) )
6     T = np.diag(beta,1) + np.diag(beta,-1)
7     D, V = np.linalg.eigh(T)
8     x = D
9     w = 2 * V[0, :]**2
10    return x, w

```

Algoritmo 19: Nodos y pesos de la Cuadratura Gaussiana

Antes de finalizar, usted puede notar que en el algoritmo 19 aparece el coeficiente $\alpha_n = 0$, ver ecuación (E.6), en particular en la definición de T_n . También podemos notar que los coeficientes β_{n-1} tienen definido un valor explícito en función de m . Esto se debe a que en este caso particular uno puede obtener una expresión explícita de ecuación (E.4) y de ecuación (E.3). Antes de obtener los coeficientes de forma explícita, es necesario recordar que

si consideramos los polinomios de Legendre unitarios, es decir ecuación (E.7), los coeficientes se simplifican de la siguiente forma²:

$$\begin{aligned}\alpha_n &= \langle p_n, x p_n \rangle, \\ \beta_{n-1} &= \langle p_n, x p_{n-1} \rangle.\end{aligned}$$

El coeficiente α_n se puede expresar de la siguiente forma,

$$\begin{aligned}\alpha_n &= \langle p_n, x p_n \rangle \\ &= \int_{-1}^1 x p_n^2(x) dx \\ &= 0.\end{aligned}$$

Esto se debe a que la función $x p_n^2(x)$ es impar³. Ahora, para el segundo término se obtiene lo siguiente,

$$\begin{aligned}\beta_{n-1} &= \langle p_n, x p_{n-1} \rangle \\ &= \int_{-1}^1 x p_n(x) p_{n-1}(x) dx.\end{aligned}$$

Utilizando la formula recursiva de Bonnet⁴, la cual es la siguiente,

$$(n+1) P_{n+1}(x) = (2n+1) x P_n(x) - n P_{n-1}(x), \quad (\text{E.9})$$

donde $P_n(x)$ corresponde a los polinomios no unitarios y satisfacen la relación $p_n(x) = \sqrt{\frac{2n+1}{2}} P_n(x)$. Despejando $x P_n(x)$ en la expresión anterior tenemos,

$$x P_n(x) = \frac{n+1}{2n+1} P_{n+1}(x) + \frac{n}{2n+1} P_{n-1}(x).$$

Ahora, reemplazando en la computación de β_{n-1} obtenemos,

$$\begin{aligned}\beta_{n-1} &= \int_{-1}^1 x p_n(x) p_{n-1}(x) dx \\ &= \sqrt{\frac{2n+1}{2}} \sqrt{\frac{2n-1}{2}} \int_{-1}^1 x P_n(x) P_{n-1}(x) dx \\ &= \sqrt{\frac{2n+1}{2}} \sqrt{\frac{2n-1}{2}} \int_{-1}^1 \left(\frac{n+1}{2n+1} P_{n+1}(x) + \frac{n}{2n+1} P_{n-1}(x) \right) P_{n-1}(x) dx \\ &= \sqrt{\frac{2n+1}{2}} \sqrt{\frac{2n-1}{2}} \int_{-1}^1 \left(\frac{n+1}{2n+1} P_{n+1}(x) \right) P_{n-1}(x) dx + \sqrt{\frac{2n+1}{2}} \sqrt{\frac{2n-1}{2}} \int_{-1}^1 \left(\frac{n}{2n+1} P_{n-1}(x) \right) P_{n-1}(x) dx \\ &= \sqrt{\frac{2n+1}{2}} \sqrt{\frac{2n-1}{2}} \frac{(n+1)}{2n+1} \underbrace{\int_{-1}^1 P_{n+1}(x) P_{n-1}(x) dx}_0 + \sqrt{\frac{2n+1}{2}} \sqrt{\frac{2n-1}{2}} \frac{n}{2n+1} \int_{-1}^1 P_{n-1}(x) P_{n-1}(x) dx \\ &= \sqrt{\frac{2n+1}{2}} \sqrt{\frac{2n-1}{2}} \frac{n}{2n+1} \underbrace{\int_{-1}^1 P_{n-1}^2(x) dx}_{\frac{2}{2n-1}} \\ &= \sqrt{\frac{2n+1}{2}} \sqrt{\frac{2n-1}{2}} \frac{n}{2n+1} \frac{2}{2n-1},\end{aligned}$$

²Al considerar que son unitarios, se sabe que $\langle p_i, p_i \rangle = \|p_i\|^2 = 1$ para $i \in \{0, 1, \dots\}$

³También es posible obtener este resultado utilizando la formula recursiva de Bonnet que se presenta en la ecuación (E.9).

⁴https://en.wikipedia.org/wiki/Legendre_polynomials#Recurrence_relations

simplificando,

$$\beta_{n-1} = \frac{1}{2\sqrt{1 - \frac{1}{(2n)^2}}}.$$

Lo cual es exactamente lo implementado en el algoritmo 19! Se sugiere ver las siguientes referencias para más detalles en el cálculo anterior [4, 9].

Bibliografía

- [1] S. 1.9.3, “Newton-Krylov,” https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.newton_krylov.html, accessed: 2022-11-21. 166, 219
- [2] M. Cameron, “Gaussian quadrature,” <http://www2.math.umd.edu/~mariakc/teaching/gaussian.pdf> or <https://www.cs.umd.edu/~onwunta/Gaussian.pdf>, accessed: 2022-06-22, second link. 217, 268
- [3] A. Gil, J. Segura, and N. M. Temme, Numerical Methods for Special Functions. Philadelphia: Society for Industrial and Applied Mathematics, 2007. 178, 182, 268
- [4] G. H. Golub and J. H. Welsch, “Calculation of Gauss quadrature rules,” Mathematics of Computation, vol. 23, no. 106, pp. 221–230, 1969. [Online]. Available: <https://www.ams.org/mcom/1969-23-106/S0025-5718-69-99647-1/> 273
- [5] A. Iserles, “Lecture notes, numerical analysis,” <http://www.damtp.cam.ac.uk/user/na/PartIB/Lect03.pdf>, accessed: 2022-06-22. 268
- [6] D. Knoll and D. Keyes, “Jacobian-free Newton–Krylov methods: a survey of approaches and applications,” Journal of Computational Physics, vol. 193, no. 2, pp. 357–397, Jan. 2004. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0021999103004340> 166
- [7] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed. Society for Industrial and Applied Mathematics, 2003. [Online]. Available: <https://epubs.siam.org/doi/abs/10.1137/1.9780898718003> 8
- [8] T. Sauer, Numerical Analysis, 3rd ed. Hoboken, NJ: Pearson, 2018. 8
- [9] L. N. Trefethen and D. Bau, Numerical Linear Algebra. Philadelphia: Society for Industrial and Applied Mathematics, 1997. 8, 273