Métodos Numéricos
1er Cuatrimestre 2024
Trabajo Práctico 1

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Eliminación Gaussiana, matrices tridiagonales y difusión

Introducción

El objetivo de este trabajo es la implementación y experimentación con algoritmos de eliminación gaussiana y el estudio en particular del caso de matrices tridiagonales y su aplicación en el modelado de un problema de difusión.

A continuación describimos ciertos conceptos que se van a utilizar para realizar las consignas.

Solución de sistemas de ecuaciones

Dado un conjunto de n ecuaciones con n incógnitas podemos escribir la forma matricial del sistema de la siguiente forma:

$$Ax = b$$

donde $A \in \mathbb{R}^{n \times n}$ es una matriz cuadrada, $x \in \mathbb{R}^{n \times 1}$ un vector columna con las incógnitas y $b \in \mathbb{R}^{n \times 1}$ un vector columna con los términos independientes.

El algortimo de Eliminación Gaussiana (EG) [1] permite resolver el sistema de ecuaciones en los casos que exista una única solución. También permite identificar si la solución no puede ser encontrada ya sea porque el sistema no tiene soluciones o tiene infinitas.

Para comenzar es útil armar una matriz aumentada $A' \in \mathbb{R}^{n \times n+1}$ en la cual se concatena el vector columna b a la derecha de A. Luego, el algoritmo se puede dividir en dos etapas. En una primera, se realiza la eliminación hacia adelante. Se busca obtener una matriz escalonada a partir de la matriz A'. Para esto se realizan las operaciones necesarias desde la primera fila hasta la última para generar valores nulos en la parte inferior izquierda de la matriz. Las operaciones pueden ser: Multiplicar una fila por un escalar no nulo, Intercambiar de posición dos filas, Sumar a una fila un múltiplo de otra. Se llama pivot al elemento de la matriz que se toma para realizar las operaciones y generar valores nulos debajo de él. En la segunda parte, con la matriz escalonada, se puede despejar una de las incógnitas, y luego sustituir hacia atrás e ir despejando progresivamente las otras incógnitas. La cantidad de operaciones aritméticas es de orden $O(n^3)$.

Consideramos una versión del algoritmo sin pivoteo, es decir sin permutaciones de filas. Existen casos donde el pivot por defecto vale cero y el algoritmo no termina y no entrega una solución cuando en realidad sí existe. Si incluimos pivoteo parcial, podemos realizar permutación de filas para poder tomar un pivot distinto de cero y avanzar con la primera etapa. En este caso se suele aprovechar que si el valor que se toma como pivot es el más grande dentro de las opciones, se terminan relizando operaciones numéricas con menor error. También existe una versión de pivoteo, pivoteo total, donde se realizan permutaciones de columnas para encontrar el mejor pivot.

Sistema tridiagonal

Un sistema de ecuaciones tridiagonal tiene la pinta:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i,$$

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}. \qquad (1)$$

El sistema está definido por a, b, c y d, 4 vectores de largo n (los elementos a_1 y c_n se toman como cero). Al verlo de forma matricial se puede apreciar que la mayoría de los elementos de la matriz tienen un valor nulo, y puede considerarse como un caso particular de una matriz rala. A partir de esta observación podemos pensar que si uno aplica el algoritmo de eliminación gaussiana de forma naive va a realizar muchas operaciones redundantes sobre valores nulos.

Si se realiza una versión cuidadosa del algoritmo de eliminación gaussiana utilizando solo los vectores que definen el problema, se puede lograr un algoritmo con menor uso de memoria y con una cantidad de operaciones aritméticas de orden $O(n)$.

La idea es similar a las dos etapas del algoritmo de eliminación gaussiana: una etapa de eliminación hacia adelante donde se actualizan los valores de la matriz (definida a partir de los 4 vectores) y luego una substitución hacia atrás.
En la práctica muchas veces ocurre que problemas del tipo $Ax = d$ se realizan de forma sucesiva para distintos casos de d manteniendo A. Por lo tanto, podría resultar ventajoso precomputar las actualizaciones de la primera etapa por única vez y luego para cada nuevo término independiente d realizar las operaciones que faltan junto con la substitución hacia atrás.

A continuación un caso de aplicación de una matriz tridiagonal basada en un operador diferencial.

Laplaciano Discreto

El operador de Laplace o Laplaciano (∇^2) es un operador diferencial dado por la divergencia del gradiente de una función escalar. En el caso unidimensional es equivalente a la derivada segunda $(\frac{d^2}{dx^2})$. La versión discreta del operador es ampliamente utilizada en procesamiento de imágenes y grafos. Para el caso unidimensional la operación discreta de $\frac{d^2}{dx^2}u = d$ está dada por:

$$u_{i-1} - 2u_i + u_{i+1} = d_i \tag{2}$$

con $u_{-1} = 0$, $u_n = 0$ (para probar esto, buscar la definición del concepto de diferencias finitas). Notar como la derivada de un valor constante es 0, los coeficientes $(1, -2, 1)$ suman 0. En forma matricial obtenemos:

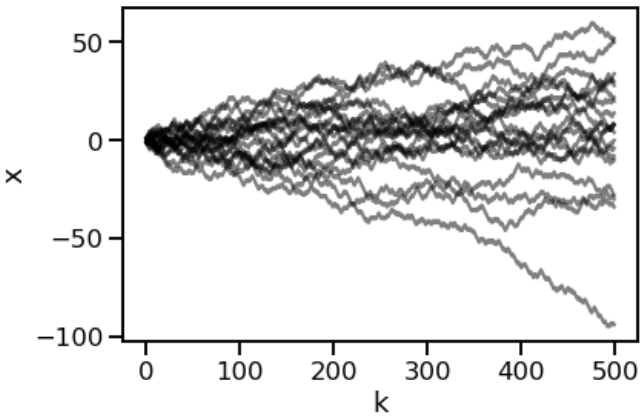
$$\begin{bmatrix} -2 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & 1 & -2 & \ddots & \\ & & \ddots & \ddots & 1 \\ 0 & & & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}.$$

Se puede utilizar el algoritmo de eliminación gaussiana para sistemas tridiagonales para encontrar el vector u para cada vector d. Cuando $n \to \infty$ la solución del sistema de ecuaciones es equivalente a encontrar una función u cuya segunda derivada es la función d (para funciones continuas).

Difusión

La difusión es un proceso estocástico donde una entidad se difunde típicamente desde un lugar de mayor concentración hacia uno de menos. Estos procesos pueden utilizarse para modelar muchos escenarios estocásticos de la naturaleza y se aplican en física, estadística, probabilidad, redes neuronales, finanzas, etc.

Como ejemplo sencillo, en este gráfico se ven varias simulaciones de la evolución temporal de una variable $x^{(k)}$ que sigue la fórmula: $x^{(k)} = x^{(k-1)} + \xi$, donde ξ es una variable aleatoria que toma valores -1 o 1.



De forma global se puede pensar como que al inicio tenemos una cantidad de partículas en la posición $u^{(0)} = 0$ y durante la evolución estas se difunden de forma aleatoria movíendose hacia sus regiones aledañas. Una pregunta que se desprende de ver estas trayectorias individuales es ¿cómo será la evolución promedio de una densidad de partículas inicial?

Podemos resolver esta pregunta si encontramos una función densidad que varíe en el tiempo. Para esto resolveremos la ecuación de difusión de forma discreta utilizando eliminación gaussiana de un sistema tridiagonal [2]. Tendremos un vector inicial $u^{(0)}$ con valores positivos indicando una magnitud que podrá difundirse hacia los costados, reduciendo su valor localmente. Es decir, para cada punto discreto i, u_i se atenuará y se dispersará hacia u_{i-1} y u_{i+1}. Si queremos que la magnitud se conserve, es decir la suma total se mantenga constante, debemos usar una operación donde el ritmo de atenuación y de dispersión sean iguales. En el caso del operador laplaciano la suma de sus coeficienes es $1 - 2 + 1 = 0$ con lo cual se garantiza la conservación. Para expresar la ecuación de difusión consideremos el incremento para el paso k, $u^{(k)} - u^{(k-1)}$ como una fracción (α) del operador laplaciano aplicado a $u^{(k-1)}$:

$$u_i^{(k)} - u_i^{(k-1)} = \alpha(u_{i-1}^{(k-1)} - 2u_i^{(k-1)} + u_{i+1}^{(k-1)}) \tag{3}$$

Si se acomodan los términos, esta formulación permite despejar de forma explícita el estado en el paso k en función del estado $k-1$. Si se repite este cómputo de forma iterada es posible obtener la evolución del vector u para $k = \{1, 2, 3, \cdots, m\}$.

Otra formulación posible es la que considera que el incremento depende del laplaciano aplicado en la magnitud en el paso k, $u^{(k)}$, en vez de $k-1$:

$$u_i^{(k)} - u_i^{(k-1)} = \alpha(u_{i-1}^{(k)} - 2u_i^{(k)} + u_{i+1}^{(k)}) \tag{4}$$

En este caso, reordenando los términos se obtiene una solución de forma implícita, ya que se obtiene el sistema de ecuaciones de forma matricial $Au^{(k)} = u^{(k-1)}$.
Ahora, para obtener la evolución, hay que resolver el sistema de ecuaciones para cada paso k.

Ambas formulaciones permiten generar la evolución del sistema, y el resultado que se obtiene es equivalente al de simular infinitas trayectorias individuales y analizar como se distribuyeron en promedio. No obstante, la formulación explícita (ec. 3) no genera soluciones estables para todos los valores de α mientras que la formulación implícita si (ec. 4) (para una demostración ver [2]).

Experimentos relacionados.
Máquina de galton: https://youtu.be/8AD7b7_HNak (https://youtu.be/8AD7b7_HNak).
Difusión de tinturas: https://youtu.be/eebPrMpKG7c (https://youtu.be/eebPrMpKG7c).

Consignas

1. **Eliminación Gaussiana sin pivoteo**
   a) Implementar una función que acepta la matriz A y el término independiente b y devuelve la respuesta x. Debe devolver una excepción si no puede encontrar la solución.
   b) Reportar un ejemplo donde el algoritmo no se puede aplicar pero si hubiese pivoteo sí.

2. **Eliminación Gaussiana con pivoteo**
   a) Implementar EG con pivoteo buscando el pivot parcial por filas con valor más grande. Incluir una advertencia en la implementación, que imprima un mensaje cuando haya operaciones que puedan incurrir en error numérico al dividir por valores cercanos a cero. Utilizar una tolerancia como argumento para determinar cuando un valor es cercano a cero.
   b) Reportar un ejemplo donde el algoritmo no se puede aplicar por que el sistema no tiene soluciones o tiene infinitas.
   c) Exploración de resultados con error numérico: Buscaremos medir el error (ϵ) entre las soluciones esperadas x y las entregadas por la implementación $\hat{x}$, para tipo flotante de 32 bits y de 64 bits. Para esto definimos el problema $Ax = b$ con A, x y b conocidos. Explorar un rango de valores de ϵ espaciado logarítmicamente entre 10^{-6} y 10^{0}. Computar el valor máximo de las diferencias absolutas, (norma infinito $||\hat{x} - x||_\infty$) y graficarlo vs ϵ.

$$A = \begin{bmatrix} 1 & 2+\epsilon & 3-\epsilon \\ 1-\epsilon & 2 & 3+\epsilon \\ 1+\epsilon & 2-\epsilon & 3 \end{bmatrix}, x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, b = \begin{bmatrix} 6 \\ 6 \\ 6 \end{bmatrix}$$

3. **Eliminación Gaussiana para un sistema tridiagonal**

   a) Derivar la formulación de EG para el caso tridiagonal.

   b) Implementar una función que tome los vectores a, b, c, d y resuelva el sistema.

   c) Implementar la variante del algortimo que a partir de precomputar las actualizaciones necesarias sobre los vectores a, b y c, resuelve el sistema para un d. *Sugerencia*: Definir un método auxiliar que se encargue únicamente del precomputo.

4. **Verificación de la implementación**

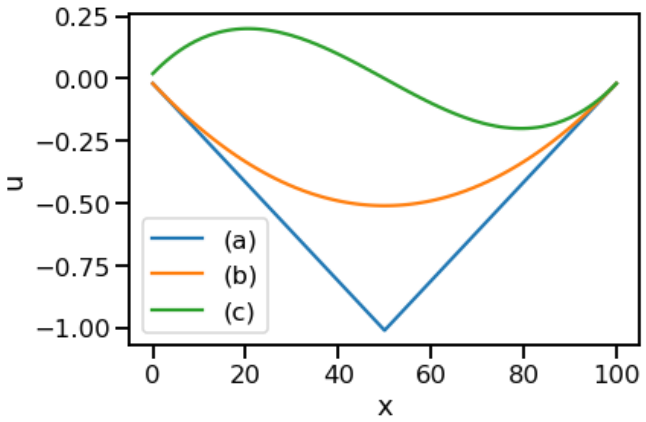
   Para probar nuestra implementación de sistema tridiagonal resolveremos una simple ecuación diferencial. Encontrar u para el problema $\frac{d^2}{dx^2}u = d$ utilizando la matriz tridiagonal del operador laplaciano, para los siguientes d:

   a) $d_i = \begin{cases} 0 \\ 4/n \quad i = \lfloor n/2 \rfloor + 1 \end{cases}$

   b) $d_i = 4/n^2$

   c) $d_i = (-1 + 2i/(n-1))12/n^2$

   Usar $n = 101$. Graficar la función u para los tres items y obtener la siguiente figura. Interpretar la formas de las curvas y la soluciones de la ecuación diferencial.

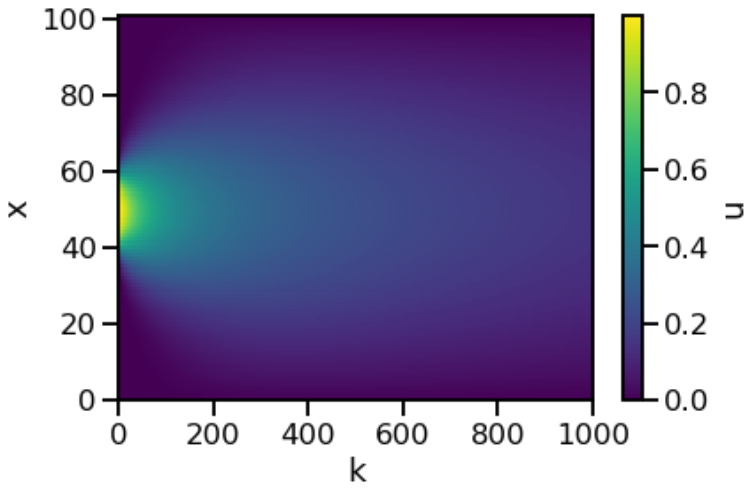


5. **Tiempos de cómputo**
   Utilizando un sistema de ecuaciones en base a la matriz laplaciana y el valor mínimo de tiempo obtenido de varias repeticiones
   a) Reportar los tiempos de cómputo, usando distintos tamaños de matrices, para los siguientes casos: EG con pivoteo y EG en un sistema tridiagonal. Graficar tiempos vs tamaño con ambos ejes en escala logarítmica. Interpretar los resultados en términos de la complejidad algorítmica de las implementaciones.
   b) Comparar los tiempos entre la implementación de EG para un sistema tridiagonal y su variante con precómputo. Para esto evaluar el sistema n veces de forma sucesiva y graficar el tiempo versus n con ambos ejes en escala logarítmica.

6. **Simulación de difusión**
   Para simular la difusión se requiere encontrar u^k en función de u^{k-1} e iterar el proceso una cantidad de pasos dada a partir de una condición inicial u^0. Esto se puede hacer con el método explícito o el implícito.
   a) Para el caso explícito, expresar la ecuación (3) de forma matricial, como la operación producto matriz-vector $u^{(k)} = Au^{(k-1)}$.
   b) Computar la difusión para m pasos utilizando la condición incial dada. Graficar la evolución de u de forma similar a la figura. Encontrar empíricamente el rango de α donde los resultados se vuelven inestables (la magnitud crece sin límites).
   c) Para el caso implícito, expresar la ecuación (4) de forma matricial como un sistema tridiagonal $Au^{(k)} = u^{(k-1)}$
   d) Computar la difusión para m pasos utilizando la condición inicial dada. Graficar la evolución de u y generar la figura para $\alpha = 1$ y comparar con otros resultados similares utilizando otros valores de α.

   Parámetros: $n = 101$, $r = 10$, $m = 1000$.
   Condición inicial:
   $$u_i^{(0)} = \begin{cases} 0 & \\ 1 & \lfloor n/2 \rfloor - r < i < \lfloor n/2 \rfloor + r \end{cases}$$

Mapa de calor de la evolución del vector u para el caso implícito con $\alpha = 1$.



IMPORTANTE

Las consignas están pensadas para resolverse utilizando python. Para las consignas de implementación de algoritmos se pide que las operaciones de matrices se realicen con python puro, es decir no utilizar numpy ya que se quiere evaluar como se planteó el algoritmo. Solamente se puede utilizar numpy para crear vectores o arreglos. Para el análisis, gráficos y otras cosas se puede usar numpy y otras herramientas.

Forma y Fecha de entrega

1. Se debe entregar un informe hecho en Latex. Este debe incluir una introducción al problema, el desarrollo y la solución a las consignas mostrando pseudocódigos con una explicación de su funcionamiento, figuras o tablas para visualizar resultados junto con sus interpretaciones (referenciadas en el texto, con descripción y debidamente rotuladas) y una sección de conclusiones con un resumen de lo mostrado y los principales resultados. Para más detalles ver el archivo pautas (https://campus.exactas.uba.ar/mod/resource/view.php?id=354262) de escritura de informes en el campus. **Intentar que el informe no supere las 14 carillas**, para esto intentar condensar resultados en pocas figuras.

2. Se debe entregar los códigos que generan los resultados, figuras, etc. Si se requieren procedimientos especiales para ejecutar el código deben ser descriptos.

3. La entrega es el **Domingo 21 de Abril 23:58h**. Para la entrega se facilitará un formulario online donde deberan adjuntar el informe y los archivos (incluir en el nombre de los archivos el nombre del grupo).

4. Recuperatorio: **Domingo 19 de Mayo hasta las 23.58hs**, enviando el trabajo corregido.

Cronograma sugerido

- Semana 25/03 (no hay labo por feriado):
    - Lectura del enunciado
    - Inicio de informe: Motivación
    - Familiarización con Python, Numpy, Matplotlib
- Semana 01/04:
    - Experimentación con medición de tiempos de procesos
    - Realizar item (1) del TP: Eliminación Gausiana sin pivoteo
    - Pensar item (6a y 6c) y pasar el desarrollo al informe
    - Pensar ítem (3a) y pasar el desarrollo al informe. Comenzar implementación (3b).
- Semana 08/04:
    - Realizar item (2) del TP: Eliminación Gausiana con pivoteo
    - Completar item (3) del TP: Variante de EG para sistema tridiagonal
    - Avanzar en el informe: Desarrollo
- Semana 15/04:
    - Realizar ítem (4) del TP: Verificación de Implementación
    - Realizar item (5) del TP: Tiempo de cómputo
    - Completar item (6) del TP: Simulación de difusión
    - Terminar informe: Resultados y Conclusiones

Referencias

[1] Higham, N. J. (2011). Gaussian elimination. Wiley Interdisciplinary Reviews: Computational Statistics, 3(3), 230-238.. pdf (https://wires.onlinelibrary.wiley.com/doi/pdfdirect/10.1002/wics.164?casa_token=cHtuxpW0tGYAAAAA:_st9iQ-vtw2-t8h5e_44oPPukDY1juQMaRR0il5LWcpRkSSFCCMkHg-xJCYg8AVl4J0bGp3hao0TffDy)

[2] Langtangen, H. P. (2013). Finite difference methods for diffusion processes. University of Oslo. pdf (https://hplgit.github.io/fdm-book/doc/pub/diffu/pdf/diffu-4print.pdf)