Álgebra Lineal Computacional

2do cuatrimestre de 2023

Trabajo Práctico N° 1.

En este trabajo práctico experimentaremos la resolución de sistemas lineales con diferentes estrategias que incluyen la utilización de la descomposición LU y la inversa de una matriz, trabajando en diferentes precisiones.

Descomposición LU

Tanto para resolver el sistemas como para calcular la inversa vamos a usar la descomposición LU.
Sea $A \in \mathbb{R}^{n \times n}$ a la cual le queremos calcular la descomposición LU. Para ello, lo calcularemos en bloques en vez del método tradicional de triangulación.
Supongamos que A tiene descomposición LU; es decir podemos descomponerla en $L, U \in \mathbb{R}^{n \times n}$ escribiendo la descomposición en bloques de la forma:

$$A = LU$$

$$\begin{pmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} u_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}$$

donde a_{11}, u_{11} son escalares, $A_{12} \in \mathbb{R}^{1 \times (n-1)}$ (vector fila), $L_{21} \in \mathbb{R}^{(n-1) \times 1}$ (vector columna), $U_{12} \in \mathbb{R}^{1 \times (n-1)}$ (vector fila), y $L_{22}, U_{22} \in \mathbb{R}^{(n-1) \times (n-1)}$. Realizando la multiplicación por bloques obtenemos la siguiente igualdad:

$$\begin{pmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} u_{11} & U_{12} \\ L_{21}u_{11} & L_{21}U_{12} + L_{22}U_{22} \end{pmatrix}$$

De aquí obtenemos las siguientes ecuaciones:

$$\begin{aligned} u_{11} &= a_{11} \\ U_{12} &= A_{12} \\ L_{21} &= \frac{A_{21}}{u_{11}} \end{aligned}$$

Estas 3 ecuaciones nos permitirán despejar y obtener el contenido de estos 3 bloques. Ahora, nos queda la última ecuación:

$$A_{22} = L_{21}U_{12} + L_{22}U_{22}$$

Despejando, nos queda:

$$A_{22} - L_{21}U_{12} = L_{22}U_{22}$$

con $L_{21}U_{12}$ un producto de vector columna por vector fila (que resulta en una matriz). Aquí llegamos a una ecuación donde nuestras incógnitas son las dos matrices L_{22} (triangular inferior con unos en la diagonal) y U_{22} (triangular superior), es decir, una descomposción LU. Si resolvemos (recursivamente) la descomposición LU de $A_{22} - L_{21}U_{12}$, podemos obtener los dos bloques que nos faltan.

Ejercicio 1. Implementar una función `descompLU` que dada una matriz $A \in \mathbb{R}^{n \times n}$ retorne las dos matrices L y U de la descomposición $A = LU$ siguiendo el esquema antes descripto. La descomposición hallada es sin pivoteo. En caso de que aparezca un 0 en la diagonal en algún paso del algoritmo, imprimir un mensaje de error y devolver las matrices I_n y A.

Ejercicio 2. (Opcional) Implementar una función `descompPLU` que dada una matriz $A \in \mathbb{R}^{n \times n}$ retorne las matrices P, L y U de la descomposición $PA = LU$. La descomposición hallada debe utilizar pivoteo parcial. Se puede utilizar cualquier algoritmo para hallar la descomposición. Si se utilizara el esquema antes descripto, ¿cómo debería modificarlo para incorporar las permutaciones de filas?

Ejercicio 3. Implementar un programa `resolverLU` que reciba una matriz A cuadrada de tamaño $n \times n$ y un vector b de tamaño n y devuelva la solución x del sistema $Ax = b$ calculando primero la descomposición LU y resolviendo luego los sistemas $Ly = b$ y $Ux = y$. Estos sistemas pueden resolverse mediante el comando `scipy.linalg.solve_triangular` de la biblioteca `scipy.linalg`.

Ejercicio 4. Probar la función `resolverLU` implementada tomando una matriz $A \in \mathbb{R}^{10 \times 10}$ de números aleatorios en $[-1, 1)$ y un vector $b \in \mathbb{R}^{10}$ de números aleatorios en $[-1, 1)$. Calcular el error relativo en norma-2: $e = \frac{\|A\tilde{x} - b\|_2}{\|b\|_2}$ para $\tilde{x}$ la solución calculada.
Sugerencia: utilizar el comando `numpy.random.rand` para generar matrices y vectores aleatorios.

Cálculo de la inversa

Ejercicio 5. Implementar la función `inversa` que reciba una matriz A de tamaño $n \times n$ inversible y calcule la inversa de A realizando los siguientes pasos:

1. Calcular la descomposición LU de A utilizando las rutinas anteriores.
2. Para cada vector e_i, $1 \leq i \leq n$, de la base canónica de $\mathbb{R}^n$, resolver los sistemas $Ly = e_i$ y $Ux = y$ mediante el comando `scipy.linalg.solve_triangular`.
3. Guardar las soluciones como columnas de la matriz inversa.

Ejercicio 6. Probar la función `inversa` implementada tomando una matriz $A \in \mathbb{R}^{10 \times 10}$ de números aleatorios en $[-1, 1)$. Calcular $A^{-1}A$ (siendo A^{-1} la matriz retornada por la función `inversa`) y el error en norma Frobenius: $\|A^{-1}A - I\|_F$.

Experimentos

Ejercicio 7. Para n desde 10 hasta 200, realizar el siguiente experimento.

1. Repetir 10 veces:
   (a) Tomar una matriz $A \in \mathbb{R}^{n \times n}$ de números aleatorios en $[-1, 1)$ y un vector $x \in \mathbb{R}^n$ de números aleatorios en $[-1, 1)$. Definir $b = Ax$.
   (b) Calcular la solución x_1 del sistema $Ax = b$ mediante la función `resolverLU`.
   (c) Calcular el logaritmo natural del error relativo $\frac{\|Ax_1 - b\|_2}{\|b\|_2}$.
   (d) Calcular la solución x_2 del sistema $Ax = b$ mediante la fórmula $x_2 = A^{-1}b$, utilizando la función `inversa` implementada anteriormente que calcula A^{-1}.
   (e) Calcular el logaritmo natural del error relativo $\frac{\|Ax_2 - b\|_2}{\|b\|_2}$.
2. Grabar en un vector v_1 los errores cometidos por el primer método (b)-(c), asignando en la coordenada i del vector la suma de los logaritmos de los errores obtenidos para matrices de $i \times i$.
3. Grabar en un vector v_2 los errores cometidos por el segundo método (d)-(e), asignando en la coordenada i del vector la suma de los logaritmos de los errores obtenidos para matrices de $i \times i$.

Importante: en cada una de las 10 repeticiones tomar una nueva matriz aleatoria, y utilizar la misma matriz para comparar los dos métodos.
Graficar los errores cometidos por ambos métodos en un solo gráfico. Puede utilizar el siguiente código:

```
import matplotlib.pyplot as plt #biblioteca para graficar
n = np.arange(10,len(v1))
plt.plot(n,v1[10:], label='x = Inversa(A)*b')
plt.plot(n,v2[10:], label='Resolver (LU)x=b')
plt.title('Errores relativos ||Ax-b||/||b|| ')
plt.legend()
plt.show()
```

¿Qué conclusión pueden obtener del gráfico?

Ejercicio 8. Repetir todo el experimento del Ejercicio 7 midiendo los errores relativos entre las soluciones halladas y la solución original $\frac{\|x_i - x\|_2}{\|x\|_2}$. ¿Observan el mismo comportamiento?

Ejercicio 9. Repetir todo el experimento anterior del Ejercicio 7 y 8 utilizando los comandos `np.linalg.solve` y `np.linalg.inv` de `numpy` en lugar de las funciones `resolverLU` e `inversa`. ¿Observan el mismo comportamiento?

Ejercicio 10. Repetir el experimento del Ejercicio 7 para $n = 2, \ldots, 20$ utilizando únicamente una matriz $A \in \mathbb{R}^{n \times n}$ definida como $a_{ij} = \frac{1}{i+j-1}$. Calcular el número de condición de A y sacar conclusiones.

Ejercicio 11. (Opcional) Modificar todas las implementaciones de los ejercicios anteriores para que soporten operaciones en floats de precisión simple (32 bits) y doble (64 bits). Las constantes pueden definirse en 32 bits utilizando el comando `np.float32()`, y, por ejemplo, la creación de vectores de

ceros con n elementos puede realizarse con el comando `np.zeros(n, dtype='float32')`. Asegurarse que las operaciones son realizadas en 32 bits verificando que las variables tienen ese formato, por ejemplo, `assert L.dtype == 'float32'` siendo `L` una matriz de Numpy.
Comparar los resultados del Ejercicio 7 cuando se utiliza precisión simple y sacar conclusiones.

Entrega y lineamientos

La entrega se realizará a través del campus virtual de la materia con las siguientes fechas y formato:

- Fecha de entrega: hasta el domingo **1ro de octubre** a las 23:59 hs.
- Fecha de re-entrega: hasta el **25 de octubre** a las 23:59 hs.
- Formato: Jupyter Notebook (no se acepta entrega de un archivo .py).

Prestar especial atención a las siguientes indicaciones:

- El TP1 se realizará en grupos de dos personas. Deberán inscribir su grupo en el foro 'Foro de Grupos de TP' destinado para tal fin, dentro de la sección Laboratorio/TP1 del campus de la materia.
- Leer el enunciado completo antes de comenzar a generar código y sacarse todas las dudas de cada ítem antes de implementar. Para obtener un código más legible y organizado, pensar de antemano qué funciones deberán implementarse y cuáles podrían reutilizarse.
- El código debe estar correctamente comentado. Cada función definida debe contener un encabezado donde se explique los parámetros que recibe y qué se espera que retorne. Además las secciones de código dentro de la función deben estar debidamente comentados. Los nombre de las variables deben ser explicativos.
- Las conclusiones y razonamientos que respondan los ejercicios, o cualquier experimentación agregada, debe estar debidamente explicada en bloques de texto de las notebooks (markdown cells), separado de los bloques de código. Aprovechen a utilizar código LaTeX si necesitan incluir fórmulas.

  Importante: Se deberá agregar un bloque de texto final en la notebook entregada. Este bloque de texto se titulará CONCLUSIONES FINALES y allí deberán escribir sus conclusiones generales que englobe a todos los ejercicios. Una entrega sin conclusiones generales se considerará incompleta.
- Gráficos: deben contener título, etiquetas en cada eje y leyendas indicando qué es lo que se muestra.
- En caso de no haberse entregado de forma completa los ejercicios 1, 3, 4, 5 y 6 en la primera instancia, los ejercicios opcionales deberán entregarse **obligatoriamente** para la reentrega del TP 1.