Proyecto programado I-2022

Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática

CI0202 - Principios de Informática, Grupo 02 y 13

Prof. Jose Pablo Ramírez Méndez

Objetivo: Implementar de manera conjunta, los tópicos vistos hasta este momento en el curso para resolver problemas de mayor complejidad.
Fecha de entrega: Domingo 3 de julio, 11:59 p.m
Formato: Individual
Entregar: Código fuente (.py, .ipynb), 1 archivo .py o .ipynb

- No olvide incluir su **nombre completo y carnet** en su entrega, ya sea como comentarios de códigos o bien, como celdas de textos si decide utilizar cuadernos de Jupyter.
- El proyecto es de elaboración y entrega individual. No se aceptarán entregas después de la fecha y hora límite. Proyectos elaborados en grupos serán invalidadas; y serán considerados como **plagio** aquellas que sean idénticas o casi idénticas.
- No olvide aplicar cada una de las etapas de resolución de problemas vista en clase. Se recomienda que en la fase de Diseño e implementación, escriba su algoritmo en psudocódigo como comentarios de línea en su código.
- Se calificarán buenas prácticas de programación. Estas incluyen, pero no se limitan a: nombres significativos para variables, separación del programa de manera modular en subrutinas, manejo apropiado de excepciones, legibilidad del código, entre otros.
- Podrá usar la biblioteca `numpy` **SOLAMENTE** en las secciones señaladas en el enuncidado. Si utiliza otras bibliotecas o `numpy` en partes del proyectos no indicadas, se evaluará dicha parte con 0 puntos.

Resumen

Implemente un programa que sea capaz de resolver un sistema de ecuaciones de una cantidad arbitraria de variables $(v_1, v_2, ..., v_n)$ utilizando el método de eliminación Gauss-Jordan. Para esto, se tomará una matriz de tamaño $n \times m$ que represente este sistema como entrada, tal que $n + 1 = m$ (o que tenga una columna más que filas). Esta matriz deberá ser cargada de un archivo en formato *comma-separated values* (csv). Luego de aplicar el método de reducción Gauss-Jordan, el programa deberá mostrar, para cada variable del sistema de ecuaciones $v_1, v_2, ..., v_n$, el valor que satisfaga todas las ecuaciones del sistema. Es posible que el sistema no tenga solución, y debería mostrarse un mensaje en este caso. Igualmente, si el archivo de la matriz no se encuentra o no está bien constituido, debería mostrarse un mensaje que refleje esto y no realizar el proceso.

Enunciado

Sistemas de Ecuaciones

Un problema estudiado en matemática (álgebra, particularlmente) son los sistemas de ecuaciones lineales. Estos denotan una serie de n ecuaciones lineales de n variables: $v_1, v_2, ..., v_n$. Un problema de esta naturaleza trata de encontrar los valores de estas variables tal que satisfagan todas las ecuaciones. Un ejemplo de este tipo de sistema es el siguiente:

$$\begin{alignedat}{2} 2x + y - z &= 8 \qquad & (L_1) \\ -3x - y + 2z &= -11 \qquad & (L_2) \\ -2x + y + 2z &= -3 \qquad & (L_3) \end{alignedat}$$

En general, estos sistemas son de la forma:

$$\begin{matrix} a_{11}x_1 & +a_{12}x_2 & +\ldots & +a_{1n}x_n & = b_1 \\ a_{21}x_1 & +a_{22}x_2 & +\ldots & +a_{2n}x_n & = b_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m1}x_1 & +a_{m2}x_2 & +\ldots & +a_{mn}x_n & = b_m \end{matrix}$$

Este tipo de sistemas se pueden representar utilizando una representación matricial. Esto permite separar los coeficientes de las incógnitas. Esto es como sigue:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Sea A la matriz de coeficientes de tamaño $n \times m$, x el vector de incógnitas de tamaño n, y b el vector de términos independientes de tamaño m, esta representación se puede resumir de la forma $A \cdot x = b$.

La ganancia de realizar esta representación es poder utilizar álgebra matricial.En particular, el algoritmo de eliminación Gauss-Jordan permite resolver un sistema de este tipo: dejando cada ecuación en términos de solo una variable. Despejando de esta manera, se obtienen los valores de cada incógnita que satisfacen el sistema original.

Eliminación Gauss-Jordan

El método Gauss-Jordan permite resolver sistemas de ecuaciones lineales representados como matriz. Para esto, primero se debe pasar la matriz a su forma aumentada, que es "uniendo" horizontalmente la matriz A con el vector b, que lo llamaremos la matriz M. Para el ejemplo visto anteriormente, esta sería su forma aumentada:

$$\left[\begin{array}{rrr|r} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]$$

El algoritmo Gauss-Jordan trata de convertir esta matriz en su forma escalonada reducida. Dicho de otra manera, la matriz A se convierte en la matriz identidad, y el vector b contiene los valores de cada constante que satisfacen el sistema. Para nuestra ecuación anterior, su forma escalonada reducida es la siguiente:

$$\left[\begin{array}{rrr|r} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$$

Y esta es la que finalmente nos permite obtener los valores que satisfacen la ecuación original: $x = 2, y = 3, z = -1$.

Vale la pena mencionar que este algoritmo solo funciona si la matriz A es cuadrada: de tamaño $n \times n$. Esto tiene sentido, ya que se requieren n piezas de información para despejar n variables. De lo contrario, el sistema no tiene una única solución, si no que una familia de soluciones. Para efectos de este proyecto, vamos a considerar que no tienen solución. Entonces, la matriz aumentada debería ser de tamaño $n \times (n + 1)$.

Ahora, ¿cómo es este proceso de eliminación? Este se basa en 3 operaciones elementales de álgebra matricial:

- Intercambiar 2 filas
- Multiplicar una fila por un número (excepto 0)
- Añadir a una fila el múltiplo de otra

Estas operaciones pueden cambiar la forma de la matriz sin cambiar el espacio de solución. Eliminación Gauss-Jordan se puede ver como un «juego» de aplicar estas operaciones para obtener su forma escalonada reducida.

El algoritmo de Gauss-Jordan trata de «eliminar» una columna a la vez, de tal manera que todas las filas, excepto una, tengan 0 en esta columna. Esto tiene como resultado que solo una ecuación queda en términos de la primer variable. Si se repite para el resto, la ecuación 1 va a quedar solo en términos de la variable 1, la ecuación 2 solo en términos de la 2, y así sucesivamente.

La forma de eliminar la i-ésima columna es un proceso de 2 pasos:

1. Dividir la fila i entre el valor de la posición $M_{i,i}$
2. Para todas las demás $n - 1$ filas, realizar la operación $M_j = M_j - M_{j,i} * M_i$

El primer paso consiste en convertir la posición de la matriz $M_{i,i}$ en 1. Para esto, simplemente se divide toda la fila i entre este número. En términos de nuestras 3 operaciones elementales, esto se logra multiplicando la fila i por $1/M_{i,i}$. Note que esto deja el valor de la columna i en esta fila como 1.

El segundo paso es en el que todas las demás filas adquieren valor 0 en la columna i. Para esto, se debe restar hasta que sea 0. Sabemos que la fila i tiene un 1. Entonces lo que hacemos es simplemente es hacer la operación $M_j - M_{j,i} * M_i$. Como el número en la posición $M_{i,i}$ es 1, lo que estamos haciendo en el fondo es $M_{j,i} - M_{j,i}$, que resulta en 0.

Realizando este proceso, dejamos la columna i como si fuera la columna i de la matriz identidad. Si realizamos esto para todas las filas, queda una matriz de la forma escalonada reducida.

Hay un detalle que no ha sido cubierto. ¿Qué pasa si el número en la posición $M_{i,i}$ es 0 antes de dividir? En este caso, no podemos aplicar la operación. La solución es sencilla: busquemos otra columna que sí contenga un número distinto a 0. Si ocurre esto, hay que buscar entre las filas $i + 1, i + 2, \ldots, n$ por alguna que no tenga 0 en la columna i. Una vez que la encontramos, la intercambiamos con la fila i y procedemos de forma normal. Si no existe tal columna, el sistema de ecuaciones no tiene solución.

El siguiente algoritmo en pseudocódigo resume el método de reducción Gauss-Jordan:

Note que la notación Mi, i o Mj, i corresponde a $M_{i,i}$ y $M_{j,i}$ que son posiciones de la matriz M

```
Gauss Jordan(M):
    para i = 0, 1, 2,...,n:
        # Reducir una fila
        # Primero, revisar si el número es 0
        si Mi,i = 0:
            # Buscar un sustituto
            para j = i+1, i+2, ..., n:
                si Mj,i ≠ 0:
                    intercambiar fila Mi con fila Mj
                    romper ciclo
        # Revisar si sigue siendo 0
        # Si es así, el sistema no tiene solución
        si Mi,i = 0:
            retornar que el sistema no tiene solución

        # Ahora, se divide la fila por el número
        multiplicar fila Mi por el número 1/Mi,i

        # Luego, para el resto de filas, se resta
        # del tal forma que la columna i se convierta en 0
        para j = 0, 1, ..., n:
            si j ≠ i:
                sumar a la fila Mj la fila Mi multiplicada por -Mj,i
    retornar última columna como resultado al sistema de ecuaciones
```

Archivos

Como la entrada de datos mediante input es molesta, especialmente para matrices; vamos a hacer que nuestro programa permita leer arcivos de matriz. Estos deberían venir en formato Comma-Separated Values, o csv. Además, debería revisar que el archivo sea consistente y válido. O sea:

1. Cada fila esté separada con enter (\n), cada elemento de la fila separado mediante coma (,), y el separador decimal es punto (.)
2. Cada elemento de la matriz sea un número real
3. Cada fila tenga igual número de columnas (la matriz sea cuadrada)
4. La matriz sea de tamaño $n \times (n + 1)$

En caso de no cumplir con estas propiedades, su programa simplemente deberá mostrar un error y no realizar la eliminación.

Una vez que se carga la matriz a memoria, su programa deberá aplicar Gauss-Jordan, y mostrar los valores de las variables $v_1, v_2, ..., v_n$ que satisfacen el sistema. En el caso de no existir, simplemente muestre que el sistema no tiene solución.

El siguiente es un ejemplo del contenido de este tipo de archivo, correspondiendo a la matriz de ejmplo que se utilizó en los puntos anteriores:

```
2,1,-1,8
-3,-1,2,-11
-2,1,2,-3
```

Este ejemplo, junto con otros, serán adjuntados en con este enunciado.

Fases

Se recomienda realizar un proceso de diseño y desarrollo iterativo para este problema, puesto que es un ejercicio complejo. Para esto, se recomienda seguir la siguiente estructura de fases. Cada fase debería ser analizada, diseñada, implementada y probada de manera independiente. Siempre considere que el producto de una fase es la base de la siguiente.

Fase 1: 3 operaciones

Puntaje: 30 puntos, 10 cada función

La primer fase recomendada es realizar las 3 operaciones básicas de matrices. Estas son las siguientes:

- `intercambiar_filas(M, i, j)` que retorna la matriz resultado de intercambiar las filas i y j de la matriz M. (Tiene derecho a usar `numpy` para esta función si fuera es necesario)
- `multiplicar_fila(M, i, k)` que retorna la matriz resultado de multiplicar cada uno de los elementos de la fila M_i por el número k. (Tiene derecho a usar `numpy` para esta función si fuera es necesario)
- `suma_fila(M, j, i, k)` que retorna la matriz resultado de sumar, a cada elemento de la fila j de la matriz, su valor actual más el resultado de multiplicar el número correspondiente de la fila i por el número k. Tal que: $M_j = M_j + kM_i$. (Tiene derecho a usar `numpy` para esta función si fuera es necesario)

Fase 2: Gauss-Jordan

Puntaje: 40 puntos

Implemente el algoritmo Gauss-Jordan, utilizando sus 3 funciones de operaciones básicas. Esta debería ser una función `gauss_jordan(M)` que retorne el vector de soluciones. Caso que la matriz M no sea de tamaño $n \times (n + 1)$, o no tenga solución, retorne `None`, o algún otro valor que usted considere significativo para denotar esto.

Fase 3: Archivos

Puntaje: 30 puntos

Implemente la función `cargar_matriz(nombre_archivo)` que a partir del nombre de un archivo, lo abra, y cargue la matriz correspondiente, verificando de paso que sea válida. En el caso de que no sea así, o que el archivo no se pueda abrir, puede retornar `None` o algún otro valor que considere significativo para denotar esto.

Cree un programa principal que lea de input el nombre de un archivo, lo abra con su función de `cargar_matriz`, y si la matriz es válida, que la resuelva utilizando `gauss_jordan` y que finalmente muestre el vector resultado.

Ejemplos

A continuación se detallan varios ejemplos de cómo debería funcionar nuestro programa. Cada ejemplo corresponde a un archivo de ejemplo distinto, todos adjuntados con el enunciado e igualmente desplegados aquí

Ejemplo 1: m1.csv

Este ejemplo es el caso esperado, se introduce una matriz de tamaño 3×4.

Archivo:

```
In [1]: print(open("m1.csv", "r").read())
```

```
2,1,-1,8
-3,-1,2,-11
-2,1,2,-3
```

Ejecución:

```
Ingrese el nombre de un archivo de matriz en formato csv: m1.csv
v1 = 2.00000
v2 = 3.00000
v3 = -1.00000
```

Ejemplo 2: m2.csv

Este es un ejemplo de un sistema de ecuaciones que no tiene soluciones puntuales, si no una familia de soluciones. Note que la fila 1 y la 3 de la matriz son iguales. Por esto es que ocurre la indefinición.

Archivo:

```
In [37]: print(open("m2.csv", "r").read())
```

```
2,1,-1,8
-3,-1,2,-11
2,1,-1,8
```

Ejecución:

```
Ingrese el nombre de un archivo de matriz en formato csv: m2.csv
No existe solución para el sistema
```

Ejemplo 3: m3.csv

Este es un ejemplo de un sistema con una solución válida.

Archivo:

```
In [38]: print(open("m3.csv", "r").read())
```

```
3,2,1,1
2,2,4,-2
-1,0.5,-1,0
```

Ejecución:

```
Ingrese el nombre de un archivo de matriz en formato csv: m3.csv
v1 = 0.69231
v2 = -0.15385
v3 = -0.76923
```

Ejemplo 4: m4.csv

Este es otro ejemplo de un sistema con una solución válida.

Archivo:

```
In [39]: print(open("m4.csv", "r").read())
```

```
1,1,1,30
1,3,-2,20
1,1,-2,0
```

Ejecución:

```
Ingrese el nombre de un archivo de matriz en formato csv: m4.csv
v1 = 10.00000
v2 = 10.00000
v3 = 10.00000
```

Ejemplo 5: me1.csv

Ejemplo de un archivo incorrecto. En este caso, el archivo no corresponde con una matriz. La última fila le falta un elemento.

Archivo:

```
In [40]: print(open("me1.csv", "r").read())
```

Ejecución:

```
Ingrese el nombre de un archivo de matriz en formato csv: me1.csv
Archivo de matriz no aceptado
```

Ejemplo 6: me2.csv

Ejemplo de un archivo incorrecto. En este caso, el archivo es una matriz, pero de tamaño 2×4, por o que su sistema correspondiente no tiene una única solución.

Archivo:

```
In [10]: print( "Abrir archivo" )
         print(open("me2.csv", "r").read())
```

```
Abrir archivo
2,1,-1,8
-3,-1,2,-11
```

Ejecución:

```
Ingrese el nombre de un archivo de matriz en formato csv: me2.csv
No existe solución para el sistema
```

Ejemplo 7: me3.csv

Archivo inexistente. Nuestro programa debería detectar si el archivo que le dimos no existe.

Archivo: (Nuevamente, no existe, por lo que cargarlo sin más debería dar error)

```
In [11]: print(open("me3.csv", "r").read())
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/var/folders/gl/4p2lmsl14hg98byp467mlt2w0000gn/T/ipykernel_5849/2755237916.py in <module>
----> 1 print(open("me3.csv", "r").read())

FileNotFoundError: [Errno 2] No such file or directory: 'me3.csv'
```

Ejecución:

```
Ingrese el nombre de un archivo de matriz en formato csv: me3.csv
Archivo de matriz no aceptado
```

Repaso de álgebra lineal

Les recomiendo corroborar sus resultados con este sitio web que resuelve sistemas de ecuaciones lineales con el método de Gauss-Jordan:

https://matrix.reshish.com/es/gauss-jordanElimination.php

Tomemos como ejemplo la siguiente matriz y resolvammos paso a paso el sistema lineal de ecuaciones:

$$\begin{aligned} 2x + y - z &= 8 && (L_1) \\ -3x - y + 2z &= -11 && (L_2) \\ -2x + y + 2z &= -3 && (L_3) \end{aligned}$$

Como vimos, al pasarlo a su forma de matriz aumentada obtenemos la siguiente matriz:

$$\begin{pmatrix} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{pmatrix}$$

Paso 1: Dividir fila1 entre 2

$$\begin{pmatrix} 1 & 1/2 & -1/2 & 4 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{pmatrix}$$

Paso 2: Hacer fila2 = fila2 + 3 * fila1

$$\begin{pmatrix} 1 & 1/2 & -1/2 & 4 \\ 0 & 1/2 & 1/2 & 1 \\ -2 & 1 & 2 & -3 \end{pmatrix}$$

Paso 3: Hacer fila3 = fila3 + 2 * fila1

$$\begin{pmatrix} 1 & 1/2 & -1/2 & 4 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 2 & 1 & 5 \end{pmatrix}$$

Paso 4: Dividir fila2 entre 1/2

$$\begin{pmatrix} 1 & 1/2 & -1/2 & 4 \\ 0 & 1 & 1 & 2 \\ 0 & 2 & 1 & 5 \end{pmatrix}$$

Paso 5: Hacer fila1 = fila1 + -1/2 * fila2

$$\begin{pmatrix} 1 & 0 & -1 & 3 \\ 0 & 1 & 1 & 2 \\ 0 & 2 & 1 & 5 \end{pmatrix}$$

Paso 6: Hacer fila3 = fila3 + - 2 * fila2

$$\begin{pmatrix} 1 & 0 & -1 & 3 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

Paso 7: Dividir fila3 entre -1

$$\begin{pmatrix} 1 & 0 & -1 & 3 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

Paso 8: Hacer fila2 = fila2 + - 1 * fila3

$$\begin{pmatrix} 1 & 0 & -1 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

Paso 9: Hacer fila1 = fila1 + 1 * fila3

$$\begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

Finalmente el resultado del sistema de ecuaciones está en la última columna:

$$\begin{pmatrix} 2 \\ 3 \\ -1 \end{pmatrix}$$

$x = 2, y = 3, z = -1$