# Primer proyecto de desarrollo grupal

Segundo cuatrimestre 2021

Cada grupo deberá desarrollar un programa que resuelva uno de los problemas alternativos que se enuncian, siguiendo una metodología de desarrollo de prototipación evolutiva o incremental en las etapas que se indican.

Para cada etapa, el prototipo del programa debe respetar asimismo la metodología que determina la plantilla de desarrollo de programas con funciones, con todas las secciones de documentación completas y un informe independiente que reporte, igual que para el TP1, cada encuentro del grupo con detalle de fecha, duración, objetivo y resultados, incluyendo la descripción de trabas o inconvenientes, si los tuvieran.

## RESOLUCIÓN DE UN SISTEMA DE N ECUACIONES LINEALES CON N INCÓGNITAS

Un ejemplo de un sistema lineal de ecuaciones sería:

$$\begin{cases} 2x & +y & -z & = & 8 \\ -3x & -y & +2z & = & -11 \\ -2x & +y & +2z & = & -3 \end{cases}$$

El problema es encontrar los valores de las variables $x_1$, $x_2$ y $x_3$ que satisfagan las tres ecuaciones.

Los sistemas lineales en los que los coeficientes de las ecuaciones son números reales pueden ser:

- Compatibles, si tienen solución, y en este caso pueden distinguirse entre
  - compatibles determinados, cuando tienen una solución única
  - compatibles indeterminados, cuando tienen infinitas soluciones
- Incompatibles, si no tienen solución

### Método de Gauss

El método de eliminación de Gauss consiste en convertir un sistema lineal de n ecuaciones con n incógnitas en uno escalonado, en el que la primera ecuación tiene n incógnitas y cada una de las siguientes tiene una menos, de modo que la última ecuación queda con una única incógnita. De esta forma, es fácil partir de la última ecuación e ir subiendo para despejar todas las incógnitas.

Para resolver un sistema con este método, el sistema se puede representar como una matriz ampliada de n filas y n+1 columnas que incluya a los términos independientes en la última columna, y realizar operaciones entre filas de manera de obtener el sistema escalonado. Para el ejemplo anterior, el sistema quedaría

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

Si en el proceso queda en 0 un elemento de la diagonal principal, se debe intercambiar la fila con una posterior; si queda una fila totalmente anulada, el sistema es compatible indeterminado, y si queda con los primeros n elementos nulos y el término independiente distinto de 0, el sistema es incompatible.

Una variante de este método es el llamado de **Gauss-Jordan**, que consiste realizar operaciones entre filas de la matriz ampliada hasta obtener ecuaciones de una sola incógnita, cuyo valor es el que queda en la última columna (en las primeras n columnas queda la matriz identidad).

### Métodos de inversión de la matriz de coeficientes

Otra forma de representar estos sistemas lineales es usando letras para la matriz de coeficientes del sistema, el vector de incógnitas y el vector de términos independientes

$$Ax=b$$

donde A es una matriz de n×n con los coeficientes del sistema, x es un vector columna de longitud n para las incógnitas, y b es otro vector columna de la misma longitud para los términos independientes.

Si se premultiplica cada miembro de la igualdad con la inversa de la matriz de coeficientes

$$A^{-1}Ax=A^{-1}b$$

se obtiene

$$Ix=A^{-1}b \equiv \boxed{x=A^{-1}b}$$

Para obtener la inversa de la matriz de coeficientes, que sólo existe si el sistema es compatible determinado, se puede optar por uno de los siguientes métodos.

### Eliminación de Gauss-Jordan

Comenzando con la matriz de coeficientes del sistema y la matriz identidad, se les realiza a ambas las mismas operaciones en sus filas de manera que la matriz original de coeficientes se convierta en la matriz identidad, y la que originalmente era la identidad resulta la inversa de la de coeficientes.

### Método analítico

$$\mathbf{A}^{-1} = \frac{1}{\det \mathbf{A}} \operatorname{adj}(\mathbf{A})$$

donde adj(A) es la matriz de cofactores traspuesta, o sea,

$$\operatorname{adj}(\mathbf{A}) = \operatorname{cof}(\mathbf{A})^T = \mathbf{C}^T$$

Por ejemplo, para

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}$$

su matriz de cofactores es

$$\operatorname{cof}(\mathbf{A}) = \begin{pmatrix} +\begin{vmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{vmatrix} & -\begin{vmatrix} A_{21} & A_{23} \\ A_{31} & A_{33} \end{vmatrix} & +\begin{vmatrix} A_{21} & A_{22} \\ A_{31} & A_{32} \end{vmatrix} \\ \\ -\begin{vmatrix} A_{12} & A_{13} \\ A_{32} & A_{33} \end{vmatrix} & +\begin{vmatrix} A_{11} & A_{13} \\ A_{31} & A_{33} \end{vmatrix} & -\begin{vmatrix} A_{11} & A_{12} \\ A_{31} & A_{32} \end{vmatrix} \\ \\ +\begin{vmatrix} A_{12} & A_{13} \\ A_{22} & A_{23} \end{vmatrix} & -\begin{vmatrix} A_{11} & A_{13} \\ A_{21} & A_{23} \end{vmatrix} & +\begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} \end{pmatrix} = \begin{pmatrix} A_{22}A_{33} - A_{23}A_{32} & A_{23}A_{31} - A_{21}A_{33} & A_{21}A_{32} - A_{22}A_{31} \\ A_{32}A_{13} - A_{33}A_{12} & A_{33}A_{11} - A_{31}A_{13} & A_{31}A_{12} - A_{32}A_{11} \\ A_{12}A_{23} - A_{13}A_{22} & A_{13}A_{21} - A_{11}A_{23} & A_{11}A_{22} - A_{12}A_{21} \end{pmatrix}$$

## Etapas para resolver un sistema de ecuaciones lineales

El programa para resolver este problema deberá

- solicitar al usuario el nombre de un archivo de texto con los coeficientes y el término independiente de cada ecuación en una línea
- dependiendo del método que se escoja para resolverlo, cargar los números de cada línea en una matriz ampliada de n filas y n+1 columnas en la que la última columna es la de términos independientes, o en una matriz de coeficientes A y un vector de términos independientes b
- transformar la matriz ampliada de manera que quede escalonada o hallar la inversa de la matriz de coeficientes A
- calcular los valores de las incógnitas en un vector x y almacenar el resultado en un archivo con el nombre que indicó el usuario y posfijo "_r" (p.e. si el usuario ingresa se1.txt, el resultado debe guardarse en un archivo

se1_r.txt), representando el sistema original en una convención amigable para el usuario (notación de ecuaciones con incógnitas x1, x2, …) e informando qué tipo de sistema es al final, y en caso de que sea compatible determinado, los valores de cada incógnita

- evaluar b-Ax e informar el resultado en el mismo archivo como la precisión del resultado

El problema debe descomponerse en funciones para

- cargar la matriz ampliada o la matriz A y el vector b, según el método de resolución, tomando como fuente un archivo de texto
- transformar la matriz ampliada u obtener la inversa de la matriz de coeficientes
- buscar la solución del sistema
- almacenar el resultado en un archivo de texto
- en caso de que el sistema sea compatible determinado obtener la precisión del resultado (funciones de resta y de multiplicación de matrices)
- si corresponde, almacenar la precisión del resultado en un archivo de texto

Las entregas que cumplir se dividirán en las siguientes tres etapas, que deben comprender

1. prototipo del programa para cargar la matriz ampliada del sistema o la matriz de coeficientes A y el vector de términos independientes b y que los muestre en pantalla, más otro prototipo incremental que transforme la matriz ampliada o bien halle la inversa de A o concluya que es singular y lo reporte en pantalla
2. prototipo incremental que busque la solución del sistema e informe el resultado en pantalla, más prototipo incremental que almacene el resultado en un archivo de texto
3. prototipo incremental que para sistemas compatibles determinados evalúe b-Ax y lo informe en pantalla, más el programa definitivo que almacene en el archivo de resultados la precisión del resultado

## ANÁLISIS DE POLINOMIOS DE UNA VARIABLE

El problema que se debe resolver es, dados los coeficientes de un polinomio de una variable de grado mayor a 2, hallar las raíces reales, máximos y mínimos locales, y puntos de inflexión, en un intervalo real determinado con centro en el origen de las abscisas del sistema de coordenadas cartesianas.

La solución del problema implica recorrer el intervalo determinado a saltos de distancia fija buscando cambios de signo del polinomio (acotación de raíces reales). Para esto se requiere una función para evaluar el polinomio dados sus coeficientes y el valor de la variable, y otra función para devolver la lista de subintervalos en los que se detecta cambios de signos del polinomio dados sus coeficientes y algún dato sobre el intervalo de búsqueda (p.e., su longitud o su extremo positivo).

Luego de obtenida una lista de subintervalos en los que el polinomio cambia de signo, en una segunda etapa se encuentran aproximaciones de las raíces mediante un proceso iterativo de reducción de los subintervalos a la mitad de su tamaño, de manera de conservar signos opuestos del polinomio en los extremos, hasta que en un punto medio de un intervalo reducido a la mitad el valor del polinomio se aproxime lo suficiente a 0.0 (p.e., que el valor absoluto del valor del polinomio sea menor que $10^{-10}$ -aproximación de raíces reales).

Para la búsqueda de máximos y mínimos locales, se deben buscar las raíces de la derivada primera del polinomio, y para los puntos de inflexión, las raíces de la derivada segunda. Por esto, se debe pensar en una función que, dada una lista de coeficientes de un polinomio, devuelva la lista de coeficientes de su derivada primera, que se debe aplicar también es esta última lista para obtener su derivada segunda.

Para optimizar la acotación de raíces, se puede ingeniar una estrategia que comience evaluando en zigzag desde el origen del sistema de coordenadas (0, 1, -1, 2, -2, 3, -3…), tanto al polinomio como a sus derivadas, y en caso de que no se hayan encontrado tantas raíces reales como el grado del polinomio en un intervalo lo suficientemente amplio, se puede informar al usuario cuántas raíces reales se acotaron en ese intervalo y preguntar si desea que se amplíe la búsqueda.

También se pide mostrar una imagen del polinomio y de los puntos encontrados usando funciones de matplotlib.pyplot.

En caso de que se escoja resolver este problema, las entregas que cumplir se dividirán en las siguientes tres etapas, que deben comprender

1. prototipo del programa para cargar los coeficientes del un polinomio de un archivo de texto cuyo nombre se solicita al usuario, donde se encuentren dispuestos en una única línea y ordenados por el grado del término que representan, y que devuelva una lista de pares de tuplas $[[(x_0, p(x_0)), (x_1, p(x_1))], [(x_2, p(x_2)), (x_3, p(x_3))], \ldots [(x_{2n}, p(x_{2n})), (x_{2n+1}, p(x_{2n+1}))]]$ que representen los subintervalos de acotación tales que $p(x_{2i})*p(x_{2i+1})<0.0$ y la muestre en pantalla
2. prototipo incremental que agregue a la versión previa las listas de acotación de raíces de p' y p'' y luego la lista de raíces aproximadas de p, p'y p'' y muestre las raíces reales, máximos y mínimos locales, y puntos de inflexión
3. prototipo incremental que agregue al archivo de donde se tomaron los coeficientes del polinomio la información encontrada en un formato amigable para el usuario, y que muestre en pantalla la curva del polinomio en el intervalo de búsqueda y los puntos encontrados

# CRONOGRAMA DE ENTREGAS

| Etapa | Fecha y hora límite |
|---|---|
| **1** | 15/11, 23:59 |
| **2** | 22/11, 23:59 |
| **3** | 29/11, 23:59 |

Cada entrega debe comprender el o los prototipos de programas solicitados, el informe de trabajo de grupo de la etapa, y archivos de texto con datos de prueba que representen distintos escenarios (para sistemas de ecuaciones, sistemas de cantidades distintas de ecuaciones y uno compatible determinado, otro compatible indeterminado y otro incompatible; y para análisis de polinomios, polinomios de distintos grados, uno con raíces múltiples indetectables y otro con raíces complejas).