MÉTODOS NÚMERICOS

PROYECTO FINAL

OPCION 3

Integrantes

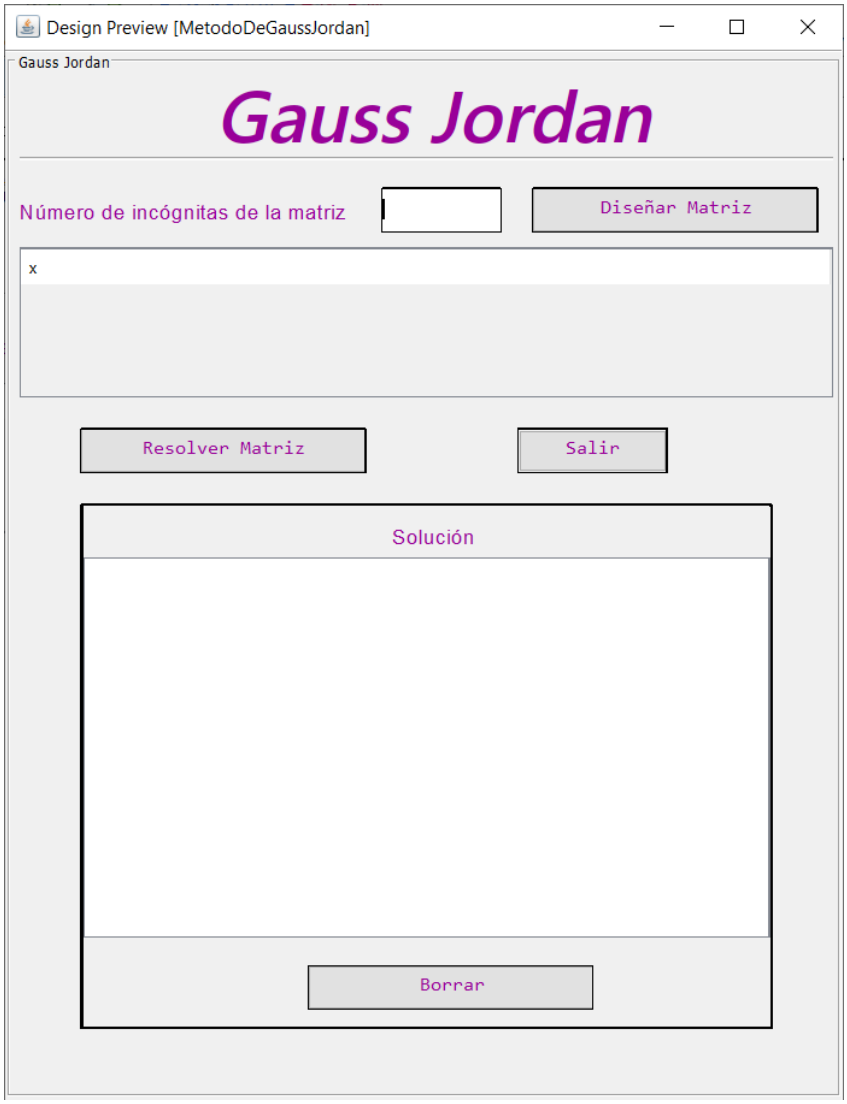
- Merino Luna Gabriela Guadalupe
- Arango Aquino Edgar Eduardo
- González Luna Alberto Carlos
- Mendoza Martínez Ángel Fernando

Se entregaran los siguientes códigos programados en un lenguaje de alto nivel, además deberá documentar como utilizar dichas funciones.
mostrando al menos un ejemplo de aplicación

1. Sistemas de ecuaciones lineales.

a) Método de Gauss.

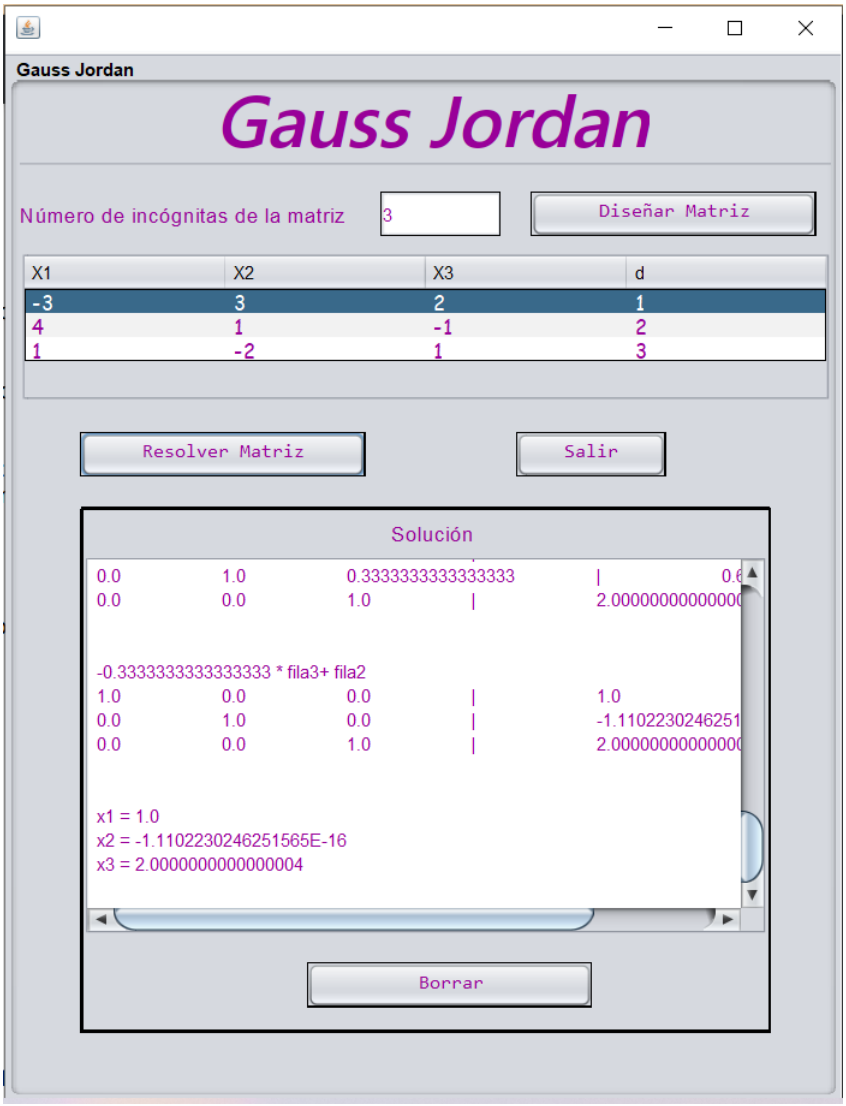
Consiste en la aplicación sucesiva del **método de reducción**, utilizando los **criterios de equivalencia** de sistemas, para transformar la **matriz ampliada con los términos independientes** (**A***) en una **matriz triangular**, de modo que cada fila (ecuación) tenga una incógnita menos que la inmediatamente anterior. Se obtiene así un sistema, que llamaremos **escalonado**, tal que la última ecuación tiene una única incógnita, la penúltima dos incógnitas, la antepenúltima tres incógnitas, ..., y la primera todas las incógnitas.



PASOS PARA EJECUTAR EL PROGRAMA:

1. Para obtener resultados correctos en esta pequeña interfaz es necesario tener en cuenta el tamaño de la matriz a ingresar y que esta debe ser cuadrada, es decir, 2x2, 3x3, 4x4, etc. ese dato se debe introducir en el numero de incógnitas el cual no va a permitir crear una tabla en la cual deberemos introducir nuestra matriz.
2. Después de introducir los datos de nuestra matriz para generar los resultados se debe presionar el botón **"Resolver matriz"**.
3. Para borrar los datos de la matriz e introducir una nueva debemos darle clic al botón **"borrar"** y luego introducir el tamaño de la matriz (repetir los pasos desde el 1).

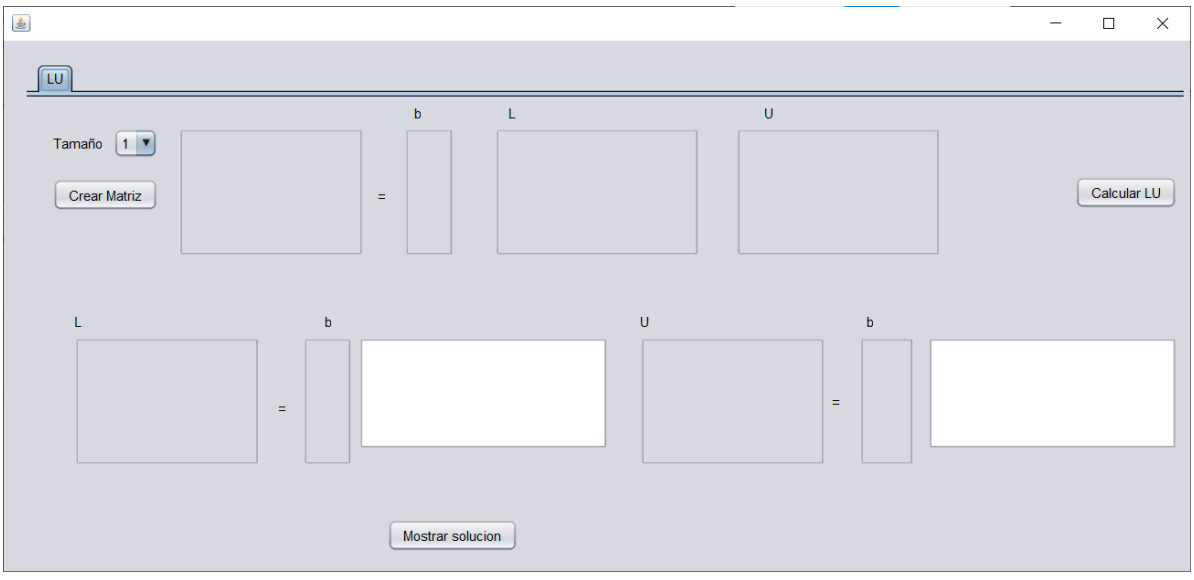
PRUEBA



Ingresamos una matriz y obtenemos los procedimientos y los valores de **x** en algunos casos son valores aproximados.

b) Factorización LU y PLU.

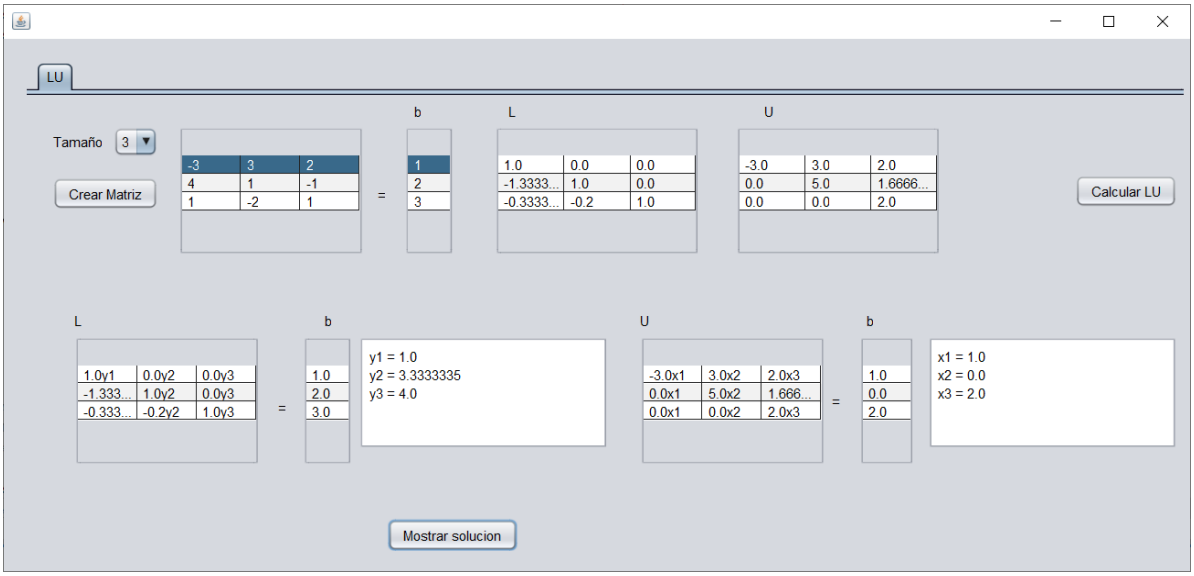
Es una forma de factorización de una matriz como el producto de una matriz triangular inferior y una superior. Debido a la inestabilidad de este método, deben tenerse en cuenta algunos casos especiales, por ejemplo, si uno o varios elementos de la diagonal principal de la matriz a factorizar es cero, es necesario premultiplicar la matriz por una o varias matrices elementales de permutación.



PASOS PARA EJECUTAR EL PROGRAMA:

1. Como en el anterior programa, debemos introducir el tamaño que tendrá nuestra matriz y con ayuda del botón **"Crear Matriz"** esta nos va a permitir ingresar nuestros datos mediante un jTable.
2. Cuando los datos ingresados sean correctos continuamos dando clic al botón **"Calcular LU"**
3. Finalmente con el botón mostrar solución nos dará nuestros datos, los cuales se mostraran en la parte de abajo, así como los valores de **x** y **y**.

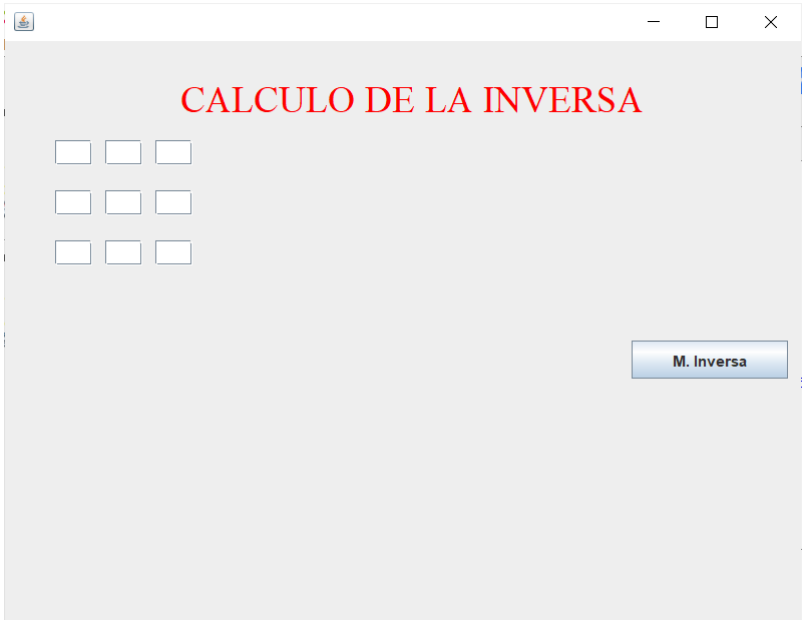
PRUEBA



Usando la matriz de la prueba anterior, obtenemos los valores de **X** y **Y**, con valores mas exactos.

c) Inversa de una matriz.

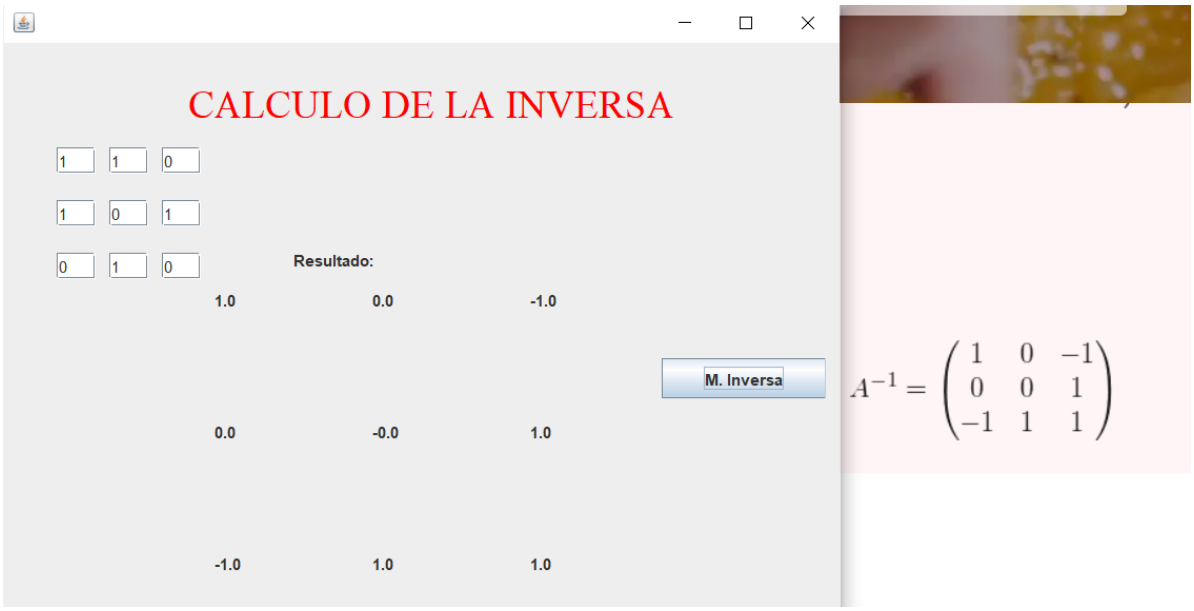
Una matriz inversa es la transformación lineal de una matriz mediante la multiplicación del inverso del determinante de la matriz por la matriz adjunta traspuesta.



PASOS PARA EJECUTAR EL PROGRAMA:

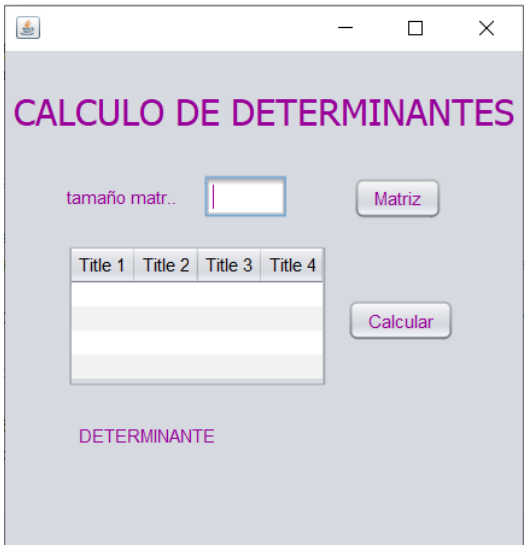
1. Por defecto se crea una matriz de 3x3
2. Debemos introducir nuestros datos en los espacios correspondientes y con el botón se calcula la inversa de la matriz

PRUEBA



d) Determinantes.

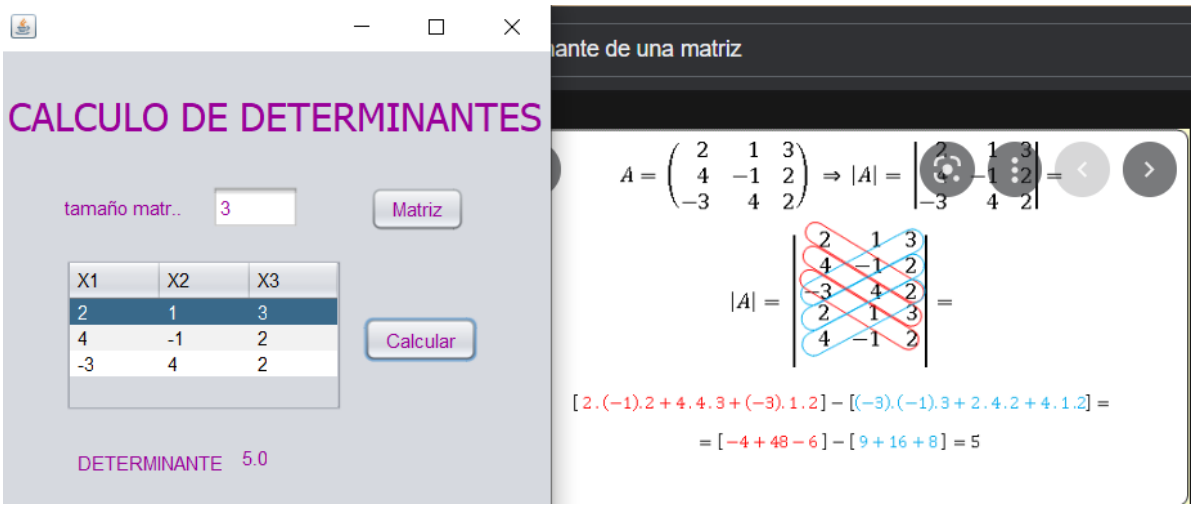
El **determinante de una matriz** siempre es igual al de su **matriz** traspuesta. El **determinante de una matriz** será siempre cero (nulo) si la **matriz** contiene dos filas o columnas iguales, si los elementos de una fila o columna son todo ceros o si los elementos de una fila o columna son una combinación lineal de las demás.



PASOS PARA EJECUTAR EL PROGRAMA:

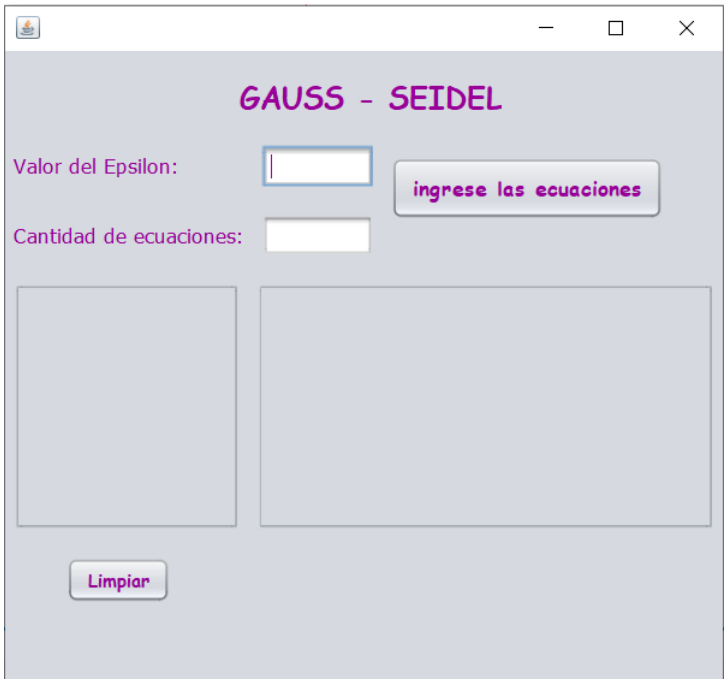
1. Tenemos que tener en cuenta el tamaño de la matriz a ingresar y generala con el botón **"Matriz"**, esto nos va a crear un jTable para poder introducir los datos de la matriz.
2. Con el botón **"Calcular"** se realizan los cálculos para realizar los cálculos para obtener el determinante y los muestra los resultado en una etiqueta.
3. Para poder ingresar nuevos datos solo se debe ingresar nuevamente el tamaño de la matriz (repetir desde el paso 1).

PRUEBA



e) Gauss Seidel.

El Método de Gauss-Seidel consiste en hacer iteraciones, a partir de un vector inicial, para encontrar los valores de las incógnitas hasta llegar a una tolerancia deseada, la diferencia radica en que cada vez que se desee encontrar un nuevo valor de una xi, además de usar los valores anteriores de las x, también utiliza valores actuales de las x encontradas antes (desde x0 hasta xi-1).



PASOS PARA EJECUTAR EL PROGRAMA:

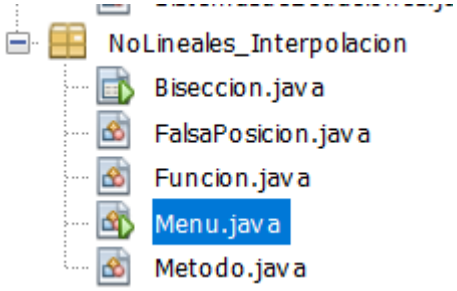
PRUEBA

#####

2. Ecuaciones no lineales.

Método de bisección

El método de bisección es un algoritmo de búsqueda de raíces que trabaja dividiendo el intervalo a la mitad y seleccionando el subintervalo que tiene la raíz. Esto se logra llevar a cabo a través de varias interacciones que son aplicadas en un intervalo para por medio de ello encontrar la raíz de la función. Para ejecutar este programa debemos abrir el paquete llamado **NoLineales_Interpolacion,** posteriormente la clase **Menu.java** y daremos clic derecho sobre el para darle clic en la opción **Run file**



Al ejecutarse nos desplegará un menú con las opciones de los métodos numéricos de ecuaciones no lineales y de interpolación, elegimos la opción 1.

```
                                METODOS NUMERICOS

ECUACIONES NO LINEALES:
        1.-Biseccion
        2.-Falsa Posición
        3.-Newton-Raphson

INTERPOLACIÓN:
        4.-Interpolacion Newton
        5.-Interpolacion Lagrange
```

Nos mostrará una interfaz gráfica la cuál nos pide que escribamos la función, los límites inferior y superior y la tolerancia, y al darle clic en el botón **Calcular** nos arroja el resultado en la parte donde dice **Raíz.**

PRUEBA

Método de falsa posición

El método de la falsa posición pretende conjugar la seguridad del método de la bisección con la rapidez del método de la secante. Este método, como en el método de la bisección, parte de dos puntos que rodean a la raíz f(x) = 0, es decir, dos puntos x0 y x1tales que f(x0)f(x1) < 0.
Para el Método de Falsa posición presionamos la opción 2 del menú anteriormente visto y damos enter, nos pedirá los coeficientes de la ecuación y el error.

PRUEBA

```
                    "METODO DE FALSA POSICIÓN"
Coeficiente 1:
-0.5
Exponente:
2
Coeficiente 2:
2.5
Coeficiente 3:
4.4
Error:
0.0004
Ejemplo a realizar:
        ---    -0.5X^2.0 + 2.5X + 4.4    ---
        ---   error < 4.0E-4     ---
```

Posteriormente nos pedirá que escribamos el valor de a y b, y al darle enter nos desplegará las iteraciones que realizó para llegar al resultado.

```
Iteracin : 6
a = 6.357545153150706
b = 12.0
f(a) = 0.08467269570174807
f(b) = -37.6
xi = 6.370223035909992
f(xi) = 0.03568682615582297
Exi = 0.0019901787877469866

Iteracin : 7
a = 6.370223035909992
b = 12.0
f(a) = 0.03568682615582297
f(b) = -37.6
xi = 6.375561290337008
f(xi) = 0.015012342420673619
Exi = 8.372995229623512E-4

Iteracin : 8
a = 6.375561290337008
b = 12.0
f(a) = 0.015012342420673619
f(b) = -37.6
xi = 6.377806032384808
f(xi) = 0.006310187599995132
Exi = 3.519614796063137E-4
```

```
ingrese el valor de a:
5
ingrese el valor de b:
12

Iteracion : 1

a = 5.0
b = 12.0
f(a) = 4.4
f(b) = -37.6
xi = 5.733333333333333
f(xi) = 2.297777777777778
Exi = 0.0

Iteracin : 2
a = 5.733333333333333
b = 12.0
f(a) = 2.297777777777778
f(b) = -37.6
xi = 6.094240837696336
f(xi) = 1.0657164003179709
Exi = 0.059221076746850135
```

Método de Newton/Raphson (pendiente)

El método de Newton-Raphson para funciones vectoriales sigue el mismo esquema que para funciones de variable real; se trata de ir generando aproximaciones. El esquema iterativo de Newton puede derivarse del desarrollo de Taylor de la función alrededor de la estimación inicial. Para que el método de Newton-Raphson converja deben cumplirse ciertas condiciones de convergencia. Existencia de la Raíz. Dado un cierto intervalo de trabajo [a,b], dentro del mismo debe cumplirse que f(a)*f(b)<0.Unicidad de la Raíz. Dentro del intervalo de trabajo [a,b], la derivada de f(x) debe ser diferente de cero. Concavidad. La gráfica de la función f(x) dentro del intervalo de trabajo [a,b], debe ser cóncava, hacia arriba o hacia abajo.

3. Interpolación.

Método de Interpolación de Lagrange.

Es una forma de presentar el polinomio que interpola un conjunto de puntos dado. La resolución de un problema de interpolación lleva a un problema de álgebra lineal en el cual se debe resolver un sistema de ecuaciones. Usando una base monómica estándar para nuestro polinomio interpolador, llegamos a la matriz de Vandermonde. Eligiendo una base distinta, la base de Lagrange, llegamos a la forma más simple de matriz identidad = δi,j, que puede resolverse inmediatamente.
En el menú anteriormente visto, el método de interpolación de Lagrange es la opción 5, entonces digitalizamos el numero 5, y damos enter. Después nos pedirá el numero de puntos que vamos a ocupar, los digitamos y al final nos pide el valor a interpolar y al darle enter nos arroja el resultado

Ejemplo

```
                "METODO DE INTERPOLACION LAGRANGE  2-4 PTOS."
Dame el numero de puntos
3
Dame los  pares de puntos
Dame x 0
0
Dame f(x) 0
7
Dame x 1
3
Dame f(x) 1
7
Dame x 2
6
Dame f(x) 2
6
Dame el valor a interpolar
5
f(x) en ese punto es: 6.444444444444445
```

Método de Interpolación de Newton.

Es un método de interpolación polinómica. Aunque sólo existe un único polinomio que interpola una serie de puntos, existen diferentes formas de calcularlo. Este método es útil para situaciones que requieran un número bajo de puntos para interpolar, ya que a medida que crece el número de puntos, también lo hace el grado del polinomio. Existen ciertas ventajas en el uso de este polinomio respecto al polinomio interpolador de Lagrange. Por ejemplo, si fuese necesario añadir algún nuevo punto o nodo a la función, tan solo habría que calcular este último punto, dada la relación de recurrencia existente y demostrada anteriormente.
En el menú de los métodos de ecuaciones no lineales e interpolación podemos observar que la opción para la interpolación de Newton es la 4, así que digitamos el numero 4 y damos enter. Este método nos pide el valor a interpolar y 5 puntos, los digitamos y al darle enter nos arroja el valor de f(x) en ese punto.

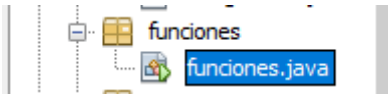
Ejemplo

```
                    "INTERPOLACION DE DIFERENCIAS DE NEWTON
Valor a interpolar:
0.35
Dame los 5 pares de puntos
Dame x 0
0
Dame f(x) 0
0.3
Dame x 1
0.1
Dame f(x) 1
0.31
Dame x 2
0.2
Dame f(x) 2
0.32
Dame x 3
0.3
Dame f(x) 3
0.33
Dame x 4
0.4
Dame f(x) 4
0.34
f(x) en ese punto es: 0.335
```

4. Cálculo numérico

Derivación e integración de funciones

La **integración**, proceso inverso de la **derivación**, se basa **en** la idea de sumar todas las partes constituyentes de un todo. La **derivación** numérica evalúa la derivada de una **función en** un punto a partir de valores numéricos de dicha **función**, sin necesidad por tanto de conocer la expresión analítica de dicha derivada.Para ejecutar este programa debemos de abrir el paquete **calculo_numérico**, seguido de la clase **Derivación_e_integración_de_funciones,** daremos click derecho y run file para su ejecución.



La regla de Simpson de 1/3 o simplemente regla de Simpson consiste en aproximar la curva con polinomios de grado 2, es decir, con parábolas. Omitiendo la deducción se tiene que el resultado de la integral es:

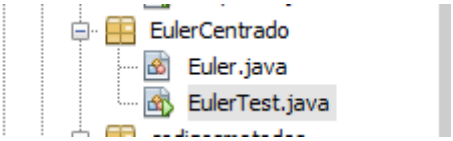
$$S_N = \frac{H}{3}\left(y_0 + 4\sum_{i=1,3,5,\ldots}^{N-1} y_i + 2\sum_{i=2,4,6,\ldots}^{N-2} y_i + y_N\right)$$

5. Ecuaciones diferenciales

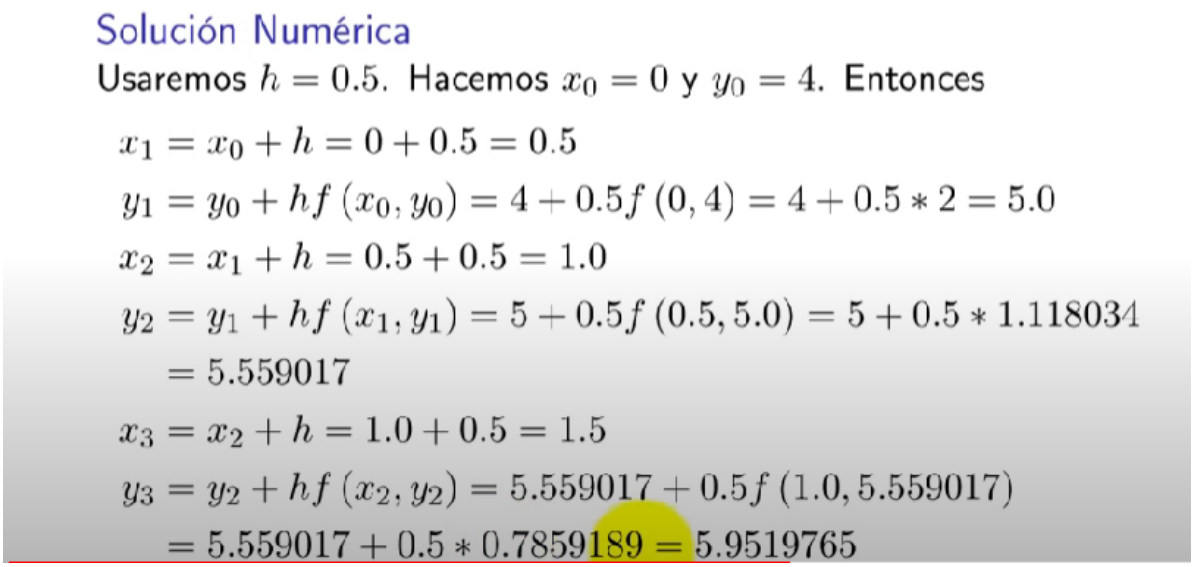
a) Métodos para resolver una ecuación diferencial, problema de condiciones iniciales

1. Euler centrado.

El método de Euler es un **método de primer orden**, lo que significa que el error local es proporcional al cuadrado del tamaño del paso, y el error global es proporcional al tamaño del paso. El método de Euler regularmente sirve como base para construir métodos más complejos.Para ejecutar este programa debemos de abrir el paquete **EulerCentrado**, seguido de la clase **EulerTest,** daremos click derecho y run file para su ejecución.



Realizaremos el siguiente ejemplo



Solución Numérica

Usaremos $h = 0.5$. Hacemos $x_0 = 0$ y $y_0 = 4$. Entonces

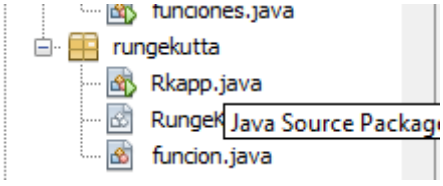
$$x_1 = x_0 + h = 0 + 0.5 = 0.5$$
$$y_1 = y_0 + hf(x_0, y_0) = 4 + 0.5f(0, 4) = 4 + 0.5 * 2 = 5.0$$
$$x_2 = x_1 + h = 0.5 + 0.5 = 1.0$$
$$y_2 = y_1 + hf(x_1, y_1) = 5 + 0.5f(0.5, 5.0) = 5 + 0.5 * 1.118034$$
$$= 5.559017$$
$$x_3 = x_2 + h = 1.0 + 0.5 = 1.5$$
$$y_3 = y_2 + hf(x_2, y_2) = 5.559017 + 0.5f(1.0, 5.559017)$$
$$= 5.559017 + 0.5 * 0.7859189 = 5.9519765$$

Y realizamos la inserción de nuestros datos en la consola de nuestro código, que seria h=0.5, x0=0, y y0= 4, y como podemos observar puede llegar con un código para resolver algunas ecuaciones diferenciales ordinarias particulares que se escribieron como funciones en el programa.

```
x           explicit    actual    error
0.000000        4.000000        0.904800        4.000000
valor aproximado es is 4.000000

ingrese otro valor inicial de x:
0
ingrese otro valor inicial de y:
4
ingrese otro valor inicial de h:
0.5
ingrese otro valor inicial de x:
0
x           explicit    actual    error
0.000000        4.000000        0.904800        4.000000
valor aproximado es 4.000000
```

Y como prueba final nuestro programa entrega una aproximación de la ecuación diferencial.**5. Ecuaciones diferenciales.****Métodos de Runge/Kutta 3o orden. Los métodos de Runge-Kutta son una serie de métodos numéricos para resolver ecuaciones diferenciales (o bien sistemas de ecuaciones diferenciales. Para ejecutar este programa debemos de abrir el paquete **runge_kutta,** seguido de la clase **Rkapp**, daremos click derecho y run file para su ejecución.



Utilizaremos el siguiente ejemploSea una ecuación diferencial de primer orden, con la condición inicial $x(t_0) = x_0$Se elige una anchura de paso *h*, y se calculan cuatro números *k**1**, k**2**, k**3**, k***4* de acuerdo con el procedimiento esquematizado en la tabla adjunta. Según el procedimiento ordinario de Runge-Kutta, a partir del valor de *x* en el instante *t* se determina el valor de *x* en el instante *t+h* mediante la fórmula que figura en la última fila de dicha tabla.

$\frac{dx}{dt} = f(x,t)$
$k_1 = hf(x,\ t)$ $k_2 = hf\left(x + \frac{k_1}{2},\ t + \frac{h}{2}\right)$ $k_3 = hf\left(x + \frac{k_2}{2},\ t + \frac{h}{2}\right)$ $k_4 = hf\left(x + k_3,\ t + h\right)$
$x(t+h) = x(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

Definimos una clase base abstracta denomina *RungeKutta*, cuya función miembro *resolver* nos calcule la solución aproximada de la ecuación diferencial *x* en el instante final *tf*, cuando le pasamos el estado inicial, es decir, el valor de *x0* en el instante inicial *t0*. La función devuelve el estado del sistema que puede ser la posición del móvil, la intensidad de la corriente eléctrica en un circuito, etc., es decir, el valor de *x*, en el instante *t*.En la clase *Funcion* derivada de *RungeKutta* describimos el modelo físico particular, redefiniendo la función *f(x, t)*. Consideremos la ecuación diferencial que describe la desintegración de una sustancia radioactiva en otra estable.

$\frac{dx}{dt} = -ax$ $x = x_0 e^{-at}$ Donde *a* es la constante de desintegración radioactiva.

A la izquierda tenemos la ecuación diferencial y a la drecha su solución analítica.La definición de la clase *Funcion* define la función *f(x)*, tomando el valor de la constante de desintegración *a* igual a 0.1La llamada a la función *resolver* se efectuará desde un objeto de la clase derivada *Funcion*. Se nos pedirá los siguientes datos: el estado inicial, es decir, el número de núcleos *x0* en el instante inicial *t0*, el instante final *t* en el que queremos calcular el nuevo estado del sistema *x*, y el paso *h* para resolver numéricamente la ecuación diferencial. La función miembro *resolver* devuelve el número de núcleos quedan sin desintegrar en dicho instante *t*.

```
    public static void main(String[] args) {
        double h=0.5;        //paso

        double x0=5000;      //número inicial   en el instante t=0
        double t=20.0;       //resolver la e. d. hasta este instante

        double x=new funcion().resolver(t, 0, x0, h);
        System.out.println("valor aproximado de x: "+(int)x);// valor exacto
        x=(int)(x0*Math.exp(-0.1*t));
        System.out.println("valor exacto de x: "+(int)x);
```

Comparamos el valor exacto y el valor aproximado, tomando como paso *h* el valor 0.5, el número inicial de núcleos, 5000, y el instante *t*, 20. Se obtiene para el resultado aproximado 676 y para el resultado exacto 676, lo que nos confirma la exactitud del procedimiento de Runge-Kutta.

```
run:
valor aproximado de x: 676
valor exacto de x: 676
```

####

2. MÉTODOS DE RUNGE/KUTTA 4O ORDEN (ECUACIÓN DIFERENCIAL)

Para una ecuación diferencial de primer orden con una condición de inicio, la fórmula de Runge-Kutta de 4to orden se obtiene de la expresión con cinco términos:

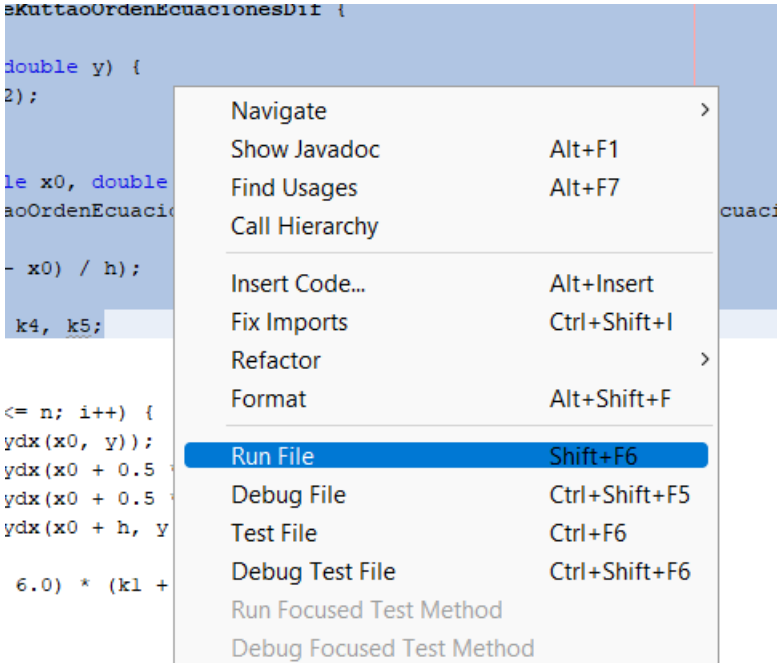
$$y_{i+1} = y_i + aK_1 + bK_2 + cK_3 + dK_4$$

siendo:

$$y'(x) = f(x_i, y_i)$$

$$y(x_0) = y_0$$

Para ejecutar el problema necesitamos un main, y luego le damos Shift + F6 o:



Nos muestra en el apartado de **Output**.

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Metodos ---

El valor de y en x es: 1.1036393232374955
------------------------------------------------------------------------
BUILD SUCCESS
```

PROBLEMA

El método de Runge-Kutta encuentra el valor aproximado de y para una x dada. Solo las ecuaciones diferenciales ordinarias de primer orden se pueden resolver utilizando el método de cuarto orden de Runge Kutta.A continuación, se muestra la fórmula utilizada para calcular el siguiente valor y n+1 a partir del valor anterior y n. Los valores de n son 0, 1, 2, 3, ... (x – x0) /h. Aquí **h es la altura del escalón** y xn+1 = x0 +h. Un tamaño de paso más bajo significa más precisión.

$$K_1 = hf(x_n, y_n)$$
$$K_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$K_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$K_4 = hf(x_n + h, y_n + k_3)$$
$$y_{n+1} = y_n + k_1/6 + k_2/3 + k_3/3 + k_4/6 + O(h^5)$$

La fórmula básicamente calcula el siguiente valor y n+1 usando el y n actual más el promedio ponderado de cuatro incrementos. k1 es el incremento basado en la pendiente al comienzo del intervalo, usando yk2 es el incremento basado en la pendiente en el punto medio del intervalo, usando y + hk1/2.k3 es nuevamente el incremento basado en la pendiente en el punto medio, usando y + hk2/2.k4 es el incremento basado en la pendiente al final del intervalo, usando y + hk3.El método es un método de cuarto orden, lo que significa que el error de truncamiento local es del orden de O (h 5), mientras que el error total acumulado es del orden de O (h 4).

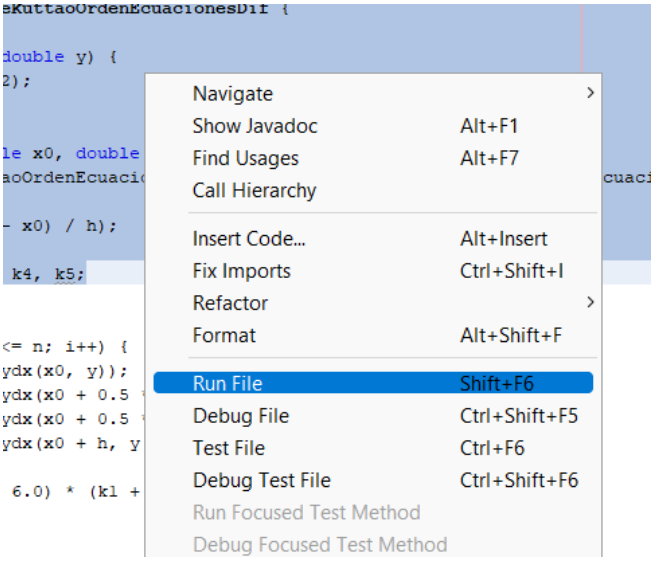
EJERCICIO

En una ecuación diferencial ordinaria se define el valor de dy/dx en la forma x e y, el cual en el código a y le damos un valor inicial de 0, es decir y(0). Al empezar el código importamos un java.io.Realizamos una variable de tipo doble la cual será dydx, en donde asigne los valores de x e y, para retornar la función de x - y entre de 2.Realizamos una función la cual encontrará el valor de y para un x dado usando el tamaño de paso h y el valor inicial y0 en x0. Tendremos que encontrar n, para eso hacemos una operación para que cuente el número de iteraciones utilizando el tamaño de paso o la altura de paso h, a lo que agregamos variables de tipo doble como: k1, k2, k3, k4, k5.Realizamos una serie de interacciones por números para aplicar las fórmulas de Runge Kutta para encontrar el siguiente valor de y. Ya de ultimo solamente actualizamos y e x, para en la siguiente función realizar la impresión de los valores x e y.

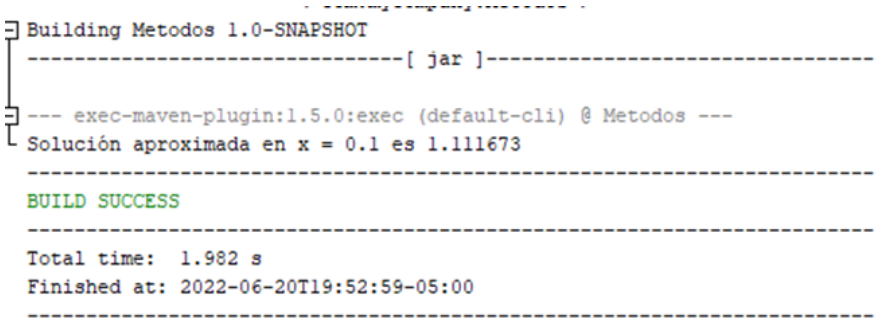
B) MÉTODOS PARA RESOLVER UN SISTEMA DE ECUACIONES, PROBLEMA DE CONDICIONES INICIALES.

1. EULER IZQUIERDO (SISTEMA DE ECUACIONES)

En matemáticas y ciencias computacionales, el método de Euler (también llamado método de Euler directo) es un procedimiento numérico de primer orden para resolver ecuaciones diferenciales ordinarias (EDO) con un valor inicial dado. Considere una ecuación diferencial dy/dx = f(x, y) con la condición inicial y(x0) =y0, entonces la aproximación sucesiva de esta ecuación puede estar dada por: y(n+1) = y(n) + h * f(x(n), y(n))donde h = (x(n) – x (0)) / n*h indica el tamaño del paso. Elegir valores más pequeños de h conduce a resultados más precisos y más tiempo de cálculo. *Para ejecutar el problema necesitamos un main, y luego le damos Shift + F6 o:



Nos muestra en el apartado de **Output**.



PROBLEMA

Considere la siguiente ecuación diferencial dy/dx = (x + y + xy)con condición inicial y (0) = 1 y tamaño de paso h = 0,025. Encuentre y (0.1).

SOLUCIÓN:

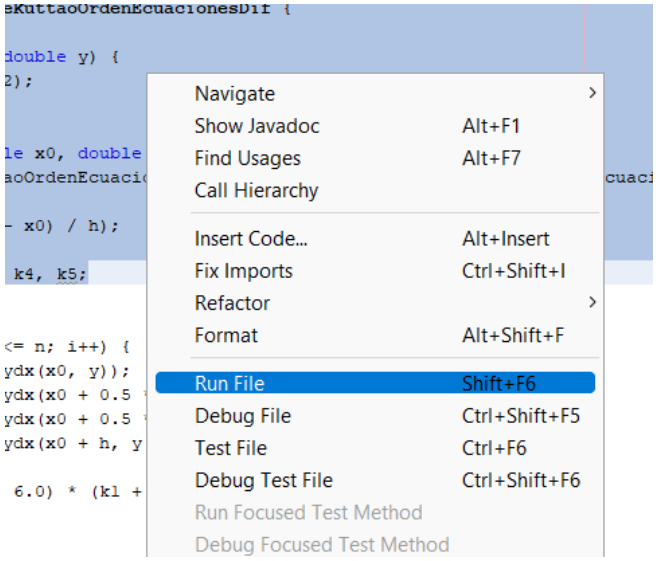
Realizaremos un programa de Euler de manera hacia atrás, una solución aproximada al programa. Lo primero que haremos es considerar una función diferencial dy/dx=(x + y + xy). Para luego crear una función en la cual vendrá la fórmula de Euler para iterar hasta el punto en el que necesitemos la aproximación del valor, realizamos una impresión de líneas para la aproximación. Ya al final hacemos una función para el programa del conductor, le damos valores iniciales para que el valor de x en el que necesitamos aproximar.
f (x, y) = (x + y + xy)x0 = 0, y0 = 1, h = 0,025Ahora podemos calcular y1 usando la fórmula de Eulery1 = y0 + h * f (x0, y0)y1 = 1 + 0.025 *(0 + 1 + 0 * 1)y1 = 1.025y (0,025) = 1,025. Similarmente podemos calcular y (0.050), y (0.075), ...y (0.1).y (0.1) = 1.11167

2. EULER CENTRADO (SISTEMA DE ECUACIONES)

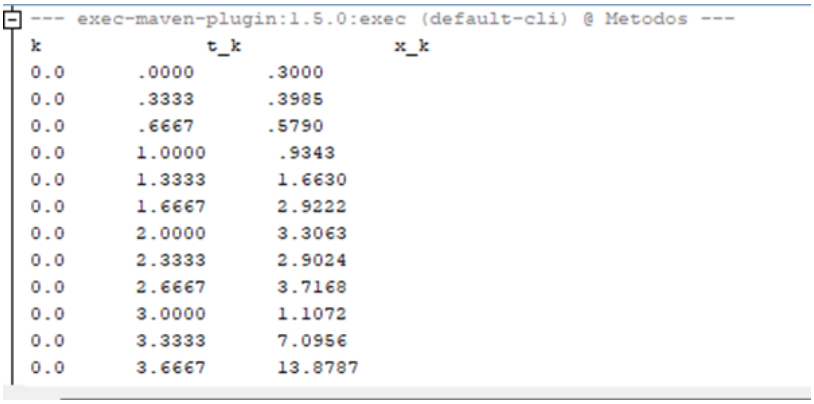
En dinámica de fluidos, las ecuaciones de Euler son las que describen el movimiento de un fluido compresible no viscoso. Su expresión corresponde a las ecuaciones de Navier-Stokes cuando las componentes disipativas son despreciables frente a las convectivas, esto nos lleva a las siguientes condiciones que se pueden deducir a través del análisis de magnitudes de las Navier-Stokes:

$$Re = \frac{\rho_0 U_0 L_0}{\mu} \gg 1$$

Para ejecutar el problema necesitamos un main, y luego le damos Shift + F6 o:



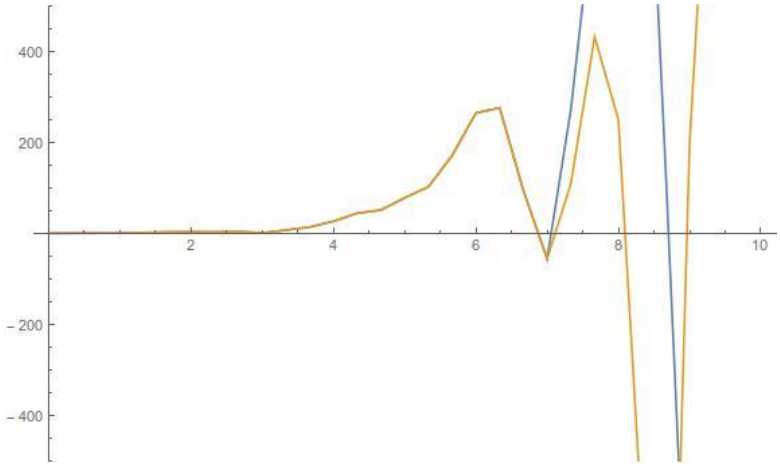
Nos muestra en el apartado de **Output**.



```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Metodos ---
k              t_k              x_k
0.0       .0000       .3000
0.0       .3333       .3985
0.0       .6667       .5790
0.0       1.0000       .9343
0.0       1.3333       1.6630
0.0       1.6667       2.9222
0.0       2.0000       3.3063
0.0       2.3333       2.9024
0.0       2.6667       3.7168
0.0       3.0000       1.1072
0.0       3.3333       7.0956
0.0       3.6667       13.8787
```

PROBLEMA

Método de Euler para encontrar un valor aproximado para x (10) y compararlo con el valor de x (10) dado por la solución exacta dada en ODE separable. Sin embargo, mi código muestra un número caótico para x (10).Ejecuté su código y lo comparé con otra implementación del mismo algoritmo. Los resultados se superponen en el régimen en el que la aproximación está funcionando, y bastante tiempo después. Solo difieren una vez que el método se descompone fuertemente:



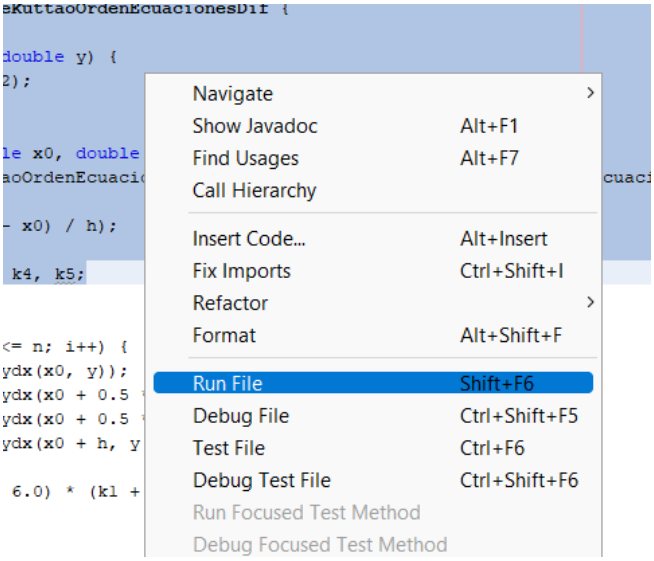
La solución real se vuelve plana, acercándose, por lo que debe ir a cero, haciendo que la derivada sea cero también. Sin embargo, explota, por lo que la derivada es numéricamente inestable 1/31/10000t=10exp(t)sin(x)pisin(x)exp(t).

SOLUCIÓN:

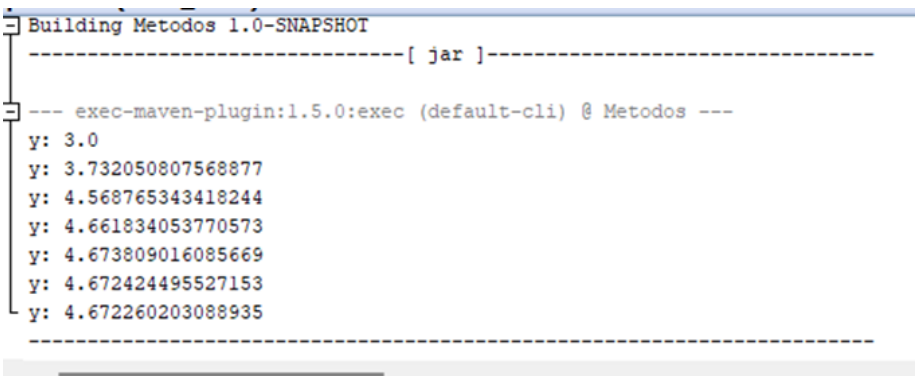
Lo primero que hacemos es importar un java de formato decimal para que nos de valores de tipo decimar y no entero. Realizamos una función en la cual guardaremos las variables para guardar los datos, todas las variables las hacemos de tipo doble. Por ejemplo, la variable h es el tamaño-paso, t_0 para condición inicial al igual que x_0 y x_f para encontrar x(10) usando este método y compararlo con un valor exacto de x(10).Crearemos dos matrices las cuales contienen los valores de x_k y t_k, para luego hacer un contador de tipo entero la cual llamaremos i. Hacemos una impresión para darle el encabezado de la tabla. La función que tenemos como if realiza las condiciones iniciales para ya al último realizamos las operaciones que son para hacer esto de tipo Euler, al final solamente lo imprimimos.

3. EULER DERECHO (SISTEMA DE ECUACIONES)

Una ecuación de Euler es una ecuación diferencial o de diferencia que es una condición Inter temporal de primer orden para un problema de elección dinámico. Describe la evolución de las variables económicas a lo largo de una trayectoria óptima. Es una condición necesaria pero no suficiente para una trayectoria óptima candidata, por lo que es útil para caracterizar parcialmente las implicaciones teóricas de una serie de modelos de comportamiento dinámico. En los modelos con incertidumbre, las ecuaciones de Euler de expectativa son condiciones sobre los momentos y, por lo tanto, proporcionan directamente una base para probar los modelos y estimar los parámetros del modelo utilizando el comportamiento dinámico observado. Para ejecutar el problema necesitamos un main, y luego le damos Shift + F6 o:



Nos muestra en el apartado de **Output**.

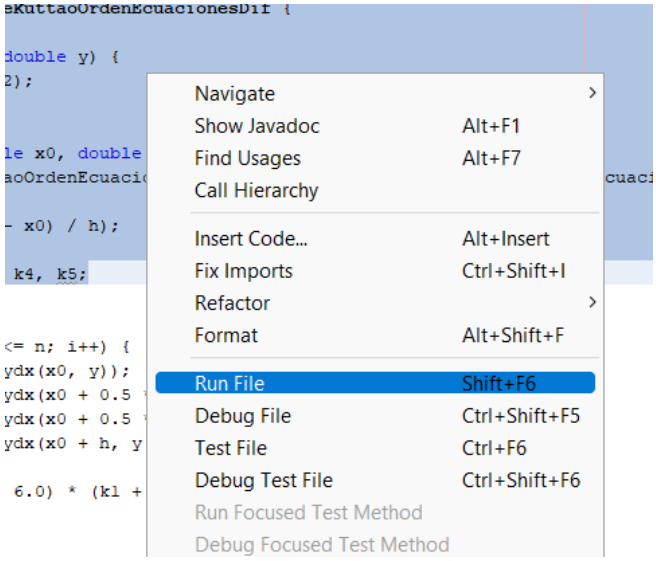


SOLUCIÓN:

Este código es un poco rápido y que creamos una función la cual asignará el orden al formato el cual imprimirá. Después de eso asignamos el valor de y de forma double en donde crearemos una función en la cual imprima otro formato agregando el valor de y. Luego hacer la función del método de Euler lo realizamos en código y lo aplicamos a este método en el cual declaramos nuevas variables todas de manera double como son; y, t y h. Para al final aplicarlo a una nueva función la cual la llamaremos Y este nos dará el resultado final.

4. MÉTODOS DE RUNGE/KUTTA 3O ORDEN (SISTEMA DE ECUACIONES)

La convergencia lenta del método de Euler y lo restringido de su región de estabilidad absoluta nos lleva a considerar métodos de orden de convergencia mayor. El método de Euler se mueve a lo largo de la tangente de una cierta curva que esta "cerca" a la curva desconocida o buscada. Los métodos Runge-Kutta extienden esta idea geométrica al utilizar varias derivadas o tangentes intermedias, en lugar de solo una, para aproximar la función desconocida. Estos métodos están basados en evaluaciones de f (x, y) en puntos intermedios del intervalo [xn, xn+1] de modo que resulten métodos equivalentes a un método de Taylor de cierto orden. Así, de forma más precisa, un método tipo Runge-Kutta se basa en m-evaluaciones en puntos intermedios del intervalo. Dada una partición del intervalo [a, b] donde existe solución del P.V.I. en N−sub intervalos, se define la solución numérica, {yn} n=0, ...N como combinación lineal de esos valores. Para ejecutar el problema necesitamos un main, y luego le damos Shift + F6 o:



Nos muestra en el apartado de **Output**.

```
                Runge - Kutta de 3er Orden

Calcula y1 & y2 con los siguientes datos
       y'y² - 2yt - 1 = 0      y0 = 1.9141     h = 2.9569      t0 = 0.0

k1 = 0.80706291
k2 = 4.32298499
k3 = 1.82402139
y1 = 5.23460404
--------------------
k1 = 3.44847302
```

PROBLEMA

Se escoge aleatoriamente la función a resolver e imprime el problema propuesto

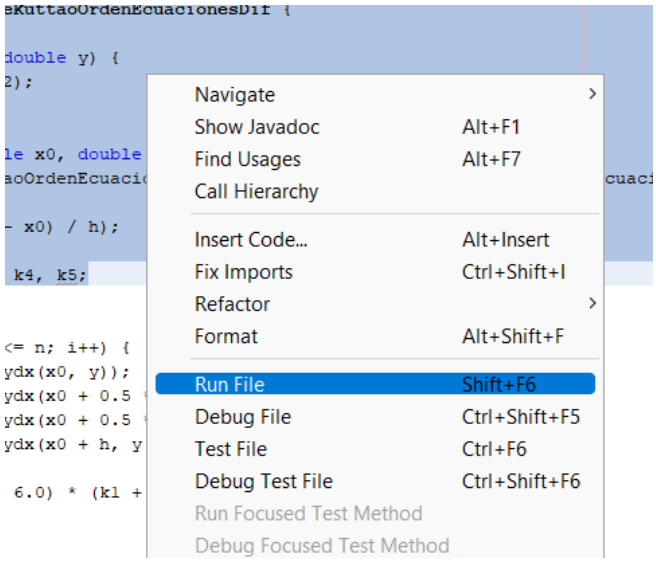
SOLUCIÓN:

A lo que sabemos de los códigos anterior, en este código necesitamos importar el java el cual hace que genere números random, a lo que investigue como tal un ejecutador de números random no genera 100% un número random. Luego de eso generamos una función la cual hará un redondeo de números para que los valores no se impriman demasiados números. Lugo de eso calcularemos el valor de Y con variables dobles de K la cual yn se sumará a (1.0/6.0) * (k1+ (4*k2) +k3).Para escoger aleatoriamente la función a resolver e imprimir el problema que se proponga. Se ordena y obteniendo nuevas variables, obteniendo máximos y mínimos, para después calcular y1 & y2 realizándolo con la formula de Runge 3er orden. Redondeando los valores como comentábamos anteriormente para ubicar el segundo caso si no se llegara realizar el primer caso, en ambos de usan las fórmulas de Runge.

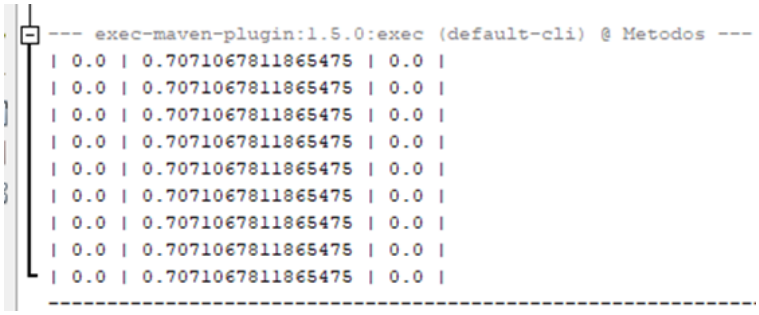
5. MÉTODOS DE RUNGE/KUTTA 4O ORDEN (SISTEMA DE ECUACIONES)

Al método de Runge-Kutta de cuarto orden se le conoce como el método clásico de Runge-Kutta. También, algunos autores lo denominan como “método RK4”.Existen métodos de Runge-Kutta de distintos órdenes, el orden del método está sujeto a la cantidad de veces que es necesaria la evaluación de la ecuación diferencial. Todos los métodos de Runge-Kutta son generalizaciones de la forma básica de Euler. Las principales características de los métodos de Runge-Kutta son:

1. Necesitan únicamente la información del punto anterior para calcular el próximo.
2. El orden del método depende de la cantidad de veces que se evalúa la función. Por lo tanto, entre mayor orden el método, más cantidad de cálculos y menor velocidad.
3. No poder estimar el error cometido sin la utilización conjunta con otro método de distinto orden.
4. Entre mayor el orden del método, se tiene mayor exactitud.
5. Para ejecutar el problema necesitamos un main, y luego le damos Shift + F6 o:



Nos muestra en el apartado de **Output**.



SOLUCIÓN:

Este Método es el más popular de los métodos RK es el de 4to orden. Como sabemos hay un numero infinito de versiones. Lo primero que debemos hacer es importar los *javas*, luego continuamos con asignar x como estático de valor doble y terminamos dando valores a (a y b). Al igual que declaramos Y1 e Y2 los cuales los volvemos de tipo doble para que retornen un valor, al momento de declararlos les hacemos unas operaciones por el valor antes asignado **x**. Realizamos unas funciones las cuales harán el papel de derivar Y1 e Y2, al momento de darles valores como 1,2,10 y 20 para las matrices, al multiplicarlo por x ala vez haciendo una división entre las 2 variables. Luego de obtener Y1 e Y2 comenzamos a hacer lo que nos retornara el código, lo que se va a mostrar al usuario. Empezamos con más matrices, hacemos más variables las cuales guardaran los cálculos que se realizaran para actualizar x. Después de todo lo anterior ahora si empezamos a realizar el cálculo por Runge-Kutta de cuarto orden para que se termine actualizando el valor de x.

[]: <>