TRABAJO PRÁCTICO N°6
Problema de Valor Inicial

Esta clase de problemas corresponden al dominio de la solución de ecuaciones diferenciales. Como recordarán del curso específico de resolución de ecuaciones diferenciales, una ecuación diferencial es aquella que involucra una función y sus derivadas:

$$F(x, y, y', .., y^{(n-1)}) = y^{(n)}$$

Muchas ecuaciones diferenciales tienen una solución trivial. En el caso

$$y' = f(x)$$

la solución puede hallarse por integración (a veces, no en forma analítica), aunque faltará algo: la integral resuelve salvo por una constante (la constante de integración). Esa constante se fija con la condición inicial, que debe ser dato del problema. En esta parte de la materia nos concentraremos en ecuaciones del tipo

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

A la ecuación (al problema) le vamos a pedir tres condiciones:

1. Existencia de una solución
2. Unicidad de la solución
3. Problema bien planteado

Las dos primeras condiciones son triviales de entender, no así de demostrar. Una demostración de existencia y unicidad de las soluciones de las ecuaciones diferenciales escapa al alcance de esta materia (de hecho, es tema de cursos previos). La tercera condición se relaciona con cuánto varía la solución al cambiar las condiciones iniciales, y es especialmente importante con los métodos numéricos. Un problema cuya solución varía mucho para pequeños cambios de las condiciones iniciales se denomina *mal planteado.*

Aquellos que hayan leído el nombre de este curso, habrán notado que se trata de métodos numéricos. Esto puede indicar que trataremos de resolver estas ecuaciones diferenciales por métodos numéricos, a los que les pediremos tres condiciones:

1. Consistencia
2. Convergencia
3. Estabilidad

Recordemos que los métodos numéricos suelen involucrar la discretización del dominio; es decir, pasar de un conjunto continuo a un conjunto discreto, finito, que pueda "contarse". Cuando se pasa del continuo al discreto, los operadores diferenciales se transforman en *operadores en diferencias*. Se dice que un método es *consistente* cuando el operador diferencial se recupera si la diferencia entre los puntos (discretos) del dominio de cálculo tienden a cero.

La condición de convergencia implica que, al reducir la distancia entre puntos del dominio, se obtienen soluciones cada vez más cercanas entre sí.

Por último, que un método numérico sea *estable* significa que el error no debe aumentar en forma descontrolada por errores en pasos anteriores.

En esta práctica, usaremos los métodos de Euler simple, Euler mejorado y Runge-Kutta.

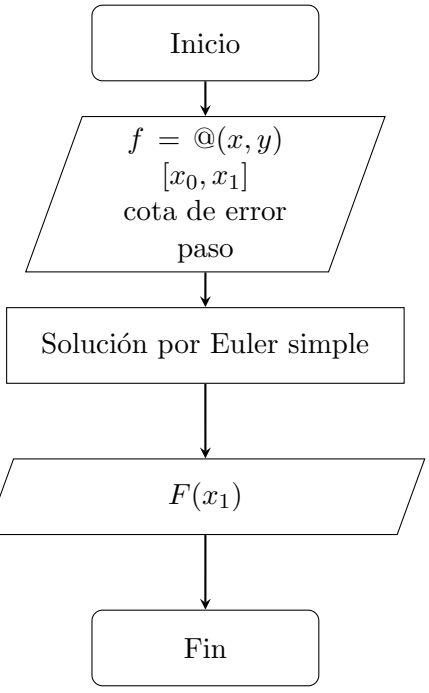
1. (Esta es ~~pregunta de examen~~)¿Cuál es la ~~pronunciación~~ correcta de "Euler"?

1. **Métodos de Euler**

   a) a) Euler hizo una "hipótesis arriesgada" al desarrollar el método de "Euler Simple". ¿Cuál es esa hipótesis?

   b) ¿Cuál es la consecuencia fundamental de la hipótesis de Euler?

   c) Realizar un diagrama de flujo del método de Euler simple. El diagrama de flujo tiene que considerar la *convergencia* de la solución dentro de un error dado, y tiene que representar el funcionamiento de una *función* (ffunction). Se considera que la función descripta por el diagrama de fluojo toma un handle, un intervalo (dominio), un paso, un valor inicial para la solución y una cota de error, y devuelve la solución en el extremo del dominio:



   El programa también debe informar en pantalla la cantidad de ciclos usados para alcanzar el error deseado.

   d) Escribir un programa que se corresponda con el diagrama de flujo del punto anterior.

   e) Modificar el código anterio para que use la mejor información disponible en un punto para hacer el cálculo de la solución en el punto siguiente (Euler mejorado).

f) Aplicar ambos métodos para resolver la ecuación diferencial

$$y' = 0,5y$$

con un error menor al 0,001 %, y comparar la cantidad de ciclos usados por uno y otro método.

g) ¿Cuál es el "orden" de cada método?

h) ¿Cómo se reduce el error obtenido con estos métodos? Respuesta: achicando el paso. ¿Siempre se reduce el error achicando el paso? Respuesta: NO. ¿Por qué? Mostrarlo con un ejemplo.

2. Métodos de Runge-Kutta

a) Conceptualmente, ¿En qué se diferencian los métodos de Runge-Kutta del método de Euler simple?

b) En general, el método de Runge-Kutta más usado es el de 4to orden. ¿Cuál es el orden del error de este método?

c) El bloque de código siguiente implementa Runge-Kutta de 4to orden en Python. O no. Analizar el código, traducirlo a Octave/Matlab y verificar si funciona y corregirlo si no lo hace.

```python
# Python program to implement Runge Kutta method
# A sample differential equation "dy / dx = (x - y)/2"
def dydx(x, y):
    return ((x - y)/2)

# Finds value of y for a given x using step size h
# and initial value y0 at x0.
def rungeKutta(x0, y0, x, h):
    n = (int)((x - x0)/h)
    for i in range(1, n + 1):
        k1 = h * dydx(x0, y)
        k2 = h * dydx(x0 + 0.5 * h, y + 0.5 * k1)
        k3 = h * dydx(x0 + 0.5 * h, y + 0.5 * k2)

        y = y + (1.0 / 6.0)*(k1 + 2 * k2 + 2 * k3)
        x0 = x0 + h
    return y

x0 = 0
y = 1
x = 2
h = 0.2
print 'The value of y at x is:', rungeKutta(x0, y, x, h)

# This code is contributed by Prateek Bhindwar
```

Nota: las estructuras de control (funciones, ciclos *for*, *while*, instrucciones de decisión *if-then-else*) en Python se "definen" dentro de su nivel de indentación, es decir, por la cantidad de "espacios" antes de cada línea de código. Aumentar o disminuir la cantidad de espacios implica entrar o salir de una dada estructura de control, por lo que no se usan comandos tipo *end* en Python.

d) Una vez resuelto el punto anterior (spoiler: el código no funciona; no porque su autor original lo haya programado mal, sino porque "alguien" lo alteró voluntariamente), tendrías que contar con un programa que resuelve ecuaciones diferenciales por el método de Runge-Kutta de 4to orden. Sin embargo, no es un "solver" general. Para hacerlo más general, habría que modificarlo para que tome como argumento una función. También hay dos maneras de suministrar la información necesaria al solver para obener el resultado: x_0, y_0, x_1, h y x_0, y_0, x_1, n (h es el tamaño del paso, y n es el número de intervalos); ambas deberían estar implementadas. Y además, el solver tendría que recibir un argumento que permita decidir si se usa h o n. Hacer lo necesario con el código para que cumpla con estos requisitos.

e) El código, hasta ahora, toma una función (ecuación diferencial) y una condición inicial y devuelve una "condición final". Hacer algo con el código anterior para que permita "graficar la solución con un paso p o un número de intervalos m", y graficar la solución. Esto requerirá almacenar los valores de las variables independiente y dependiente (x e y) en todos los puntos para los que se calcule la solución. Hacer esto de tres formas distintas:

   i) Dos vectores, uno para x y otro para y.
   ii) Una matriz de dos columnas y tantas filas como número de puntos.
   iii) Un vector de 2 × numero_de_puntos (hay que generar el vector primero, y "llenarlo" en el orden correcto).

f) Utilizar el código anterior en una función que permita calcular una solución convergida con un error inferior a un valor dado, variando el paso o el número de intervalos (el paso o número de intervalos inicial tiene que ser dato de entrada de la función).

g) Estimar el número de operaciones necesario (y el espacio en memoria necesario) para:

   i) Calcular un paso de los tres métodos vistos.
   ii) Calcular n pasos de los tres métodos vistos.
   iii) Calcular una solución con un error aproximado de $1x10^{-5}$ con los tres métodos (recordar el orden del error en cada método).

h) Utilizar los tres métodos para resolver la ecuación diferencial

$$\frac{dy}{dt} = -y + t + 1$$

con condición inicial $y(0) = 1$, para t en el intervalo $[0, 5]$ con un paso inicial de 0,1 y un error máximo de 10^{-4}.