Ejercicio 1: Vamos implementar el método RK4 clásico, Partir del código euler.m (en su versión vectorial) usado en la última clase y guardarlo como rk4.m.

Dentro del código sólo hay que modificar el bucle donde se van calculando las soluciones. Basaros en el código dado en las transparencias para el método de Heun (RK2):

```
for k=1:N-1,  % Iteracion para calcular y(k+1)
  tt=T(k); yy=S(:,k);  % Punto de partida (tk,yk)

                       K1 = f(tt,yy);             % 1ª eval de f(t,y) al inicio
  m = K1; delta=h/2;   K2 = f(tt+dt,yy+delta*m);  % 2ª evaluación en h/2

  K=(K1+K2)/2;         % Pendiente media a usar = promedio de K1, K2, ...
  S(:,k+1) = yy + h * K;  % Siguiente valor y(k+1)
end
```

y modificarlo para implementar RK4 clásico. Sólo hay que añadir el cálculo de K3 y K4 y recalcular la pendiente final K como:

$$K = \frac{1}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right)$$

Adjuntar vuestra función rk4.m

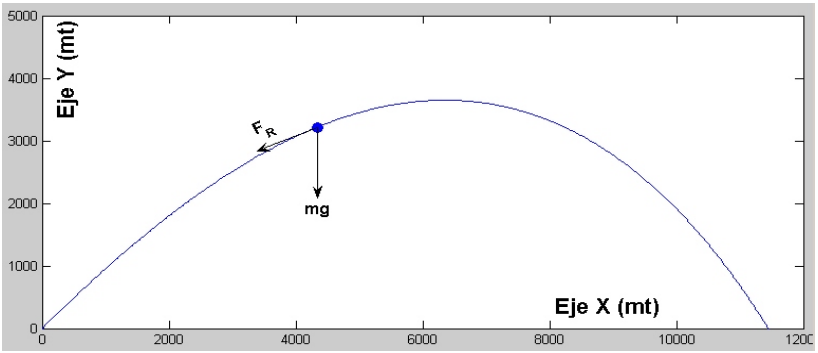
Vamos a verificar que el método rk4 es efectivamente del orden esperado. Para ello lo usaremos para resolver una ecuación sencilla cuya solución sea conocida. Por ejemplo, resolveremos y=y' (con y0=1) en el intervalo [0,5]. La solución exacta es y(t)=exp(t).

Usar un paso h=0.1. Calcular media del error absoluto entre la solución real y la solución de rk4 (en los puntos T donde el método os ha dado solución). Adjuntad error medio.

Según la teoría, ¿cuánto se debería reducir dicho error si usamos un paso h=0.01?

Repetir el cálculo del error medio para h=0.01. Adjuntad el resultados.
¿Se verifican vuestras previsiones? De no ser así vuestro método está mal implementado.

Ejercicio 2: Una de las primeras aplicaciones en el siglo XIX de los métodos numéricos para ED's fue estudiar trayectorias balísticas con objeto de elaborar tablas de artillería.



La física del movimiento de un obús es relativamente sencilla. El proyectil abandona la boca del cañón con una velocidad (muzzle velocity) inicial v0 (m/s) que viene especificada para el proyectil y pieza usada. Una vez que el proyectil sale del cañón las principales

fuerzas que actúan sobre él (ver figura adjunta) son la gravedad (magnitud mg y dirigida hacia abajo en el eje Y) y la resistencia del aire Fr que se opone al movimiento.

a) En un primer caso ignoraremos la resistencia del aire. La única fuerza es el peso que actúa hacia abajo en el eje Y (-mg), y las ecuaciones diferenciales quedan:

$$\begin{aligned} m\cdot x'' &= 0 \\ m\cdot y'' &= -mg \end{aligned}$$ o escrito vectorialmente $\bar{r}''(t)=\begin{pmatrix} x'' \\ y'' \end{pmatrix}=\begin{pmatrix} 0 \\ -g \end{pmatrix}$ siendo g=9.8m/s2.

Para resolver numéricamente el problema lo convertimos en un sistema de 4 ecuaciones diferenciales de 1er orden. El vector de estado s del sistema tendrá como componentes la posición r = (x,y) seguida de la velocidad r'=(x', y'):

vector de estado $\bar{s}=\begin{pmatrix} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \\ \begin{pmatrix} x'(t) \\ y'(t) \end{pmatrix} \end{pmatrix}=\begin{pmatrix} \bar{r}(t) \\ \bar{r}'(t) \end{pmatrix}$ y su derivada será $\bar{s}'=\begin{pmatrix} \begin{pmatrix} x'(t) \\ y'(t) \end{pmatrix} \\ \begin{pmatrix} x''(t) \\ y''(t) \end{pmatrix} \end{pmatrix}=\begin{pmatrix} \bar{r}'(t) \\ \bar{r}''(t) \end{pmatrix}$.

Se ve que las dos primeras componentes de la derivada (r') se sacan directamente de la entrada (3ª y 4ª componentes) y las dos últimas (vector r'') vienen especificadas por las ecuaciones diferenciales del problema (en este caso, r''=[0 , -g]').

Sólo tenéis que escribir una función sp=obus(t,s) que reciba como parámetros el tiempo t y el vector de estado s = (x, y, x', y') = (r, r') de 4 componentes y devuelva otro vector con sus derivadas sp = (x',y', x'',y'')=(r', r''). Trabajad directamente con r, r', r'' y no con las componentes (x, y, x', y', ...) individuales.
Adjuntad vuestra función (obus.m) codificando las ecuaciones diferenciales anteriores.

Las condiciones iniciales serán la posición inicial del cañón, que podemos suponer en el origen, luego r(0)=(0,0), y la velocidad inicial de la bala. La velocidad inicial V0 en la boca del cañón se reparte entre las componentes Vx y Vy en función del ángulo de tiro α (la elevación del cañón sobre la horizontal). Por lo tanto:

$$\bar{r}(0)=\begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \bar{r}'(0)=\begin{pmatrix} V_0\cos(\alpha) \\ V_0\sin(\alpha) \end{pmatrix}$$

Juntando r y r' en un vector (columna) 4x1 obtenemos el vector de condiciones iniciales.

Usaremos algunos datos de un ejemplo real. La pieza de artillería más común del ejército americano en la 2ª guerra mundial fue el obús M2A1 de 105 mm. Según Wikipedia este arma disparaba proyectiles de 15 kg con velocidad inicial V0 = 470 m/s. Resolver numéricamente usando RK4 entre t=0 y t=70 segundos, usando h=0.1 segundo. Usad un ángulo de tiro de 45º (el óptimo para un máximo alcance). Si usáis grados acordaros de usar cosd() y sind() en MATLAB para calcular los senos y cosenos.

Hacer la gráfica de la trayectoria de la bala (plot de coordenada Y frente a coordenada X). Podéis añadir una línea que marque el suelo haciendo `plot([0 S(1,end)],[0 0],'r:')`.
El proyectil debe describir una parábola volviendo a tierra poco antes de los 70 segundos. Si no es así tal vez habéis codificado de forma incorrecta el método RK4 o las ecuaciones diferenciales. Para diferenciar entre ambos casos correr el mismo programa usando euler

en vez de rk4. Si ahora sale una parábola el error está en rk4. Si tampoco funciona el error va a estar en la función obus.m donde habéis codificado las ecuaciones diferenciales.

Adjuntar gráfica y determinar el alcance del arma. ¿Altura máxima del proyectil?

Para este caso las ecuaciones diferenciales tiene una solución exacta:

$$x(t) = V0 \cdot \cos(\alpha) \cdot t \qquad y(t) = V0 \cdot \sin(\alpha) \cdot t - g\frac{t^2}{2}$$

Con las fórmulas anteriores, calcular la solución exacta en los tiempos T donde rk4 ha calculado la solución y haced un gráfico de la **diferencia** entre la solución exacta y la numérica) para la x(t) y la y(t). Adjuntad gráfica. ¿De qué orden son las diferencias?

b) Este primer resultado es bastante malo porque el modelo ignora la resistencia del aire, que es muy importante a las velocidades de un obús. En este apartado consideraremos la fuerza de rozamiento con el aire. Esta fuerza, Fr, es de dirección opuesta a la velocidad, y su magnitud es proporcional a la densidad del aire ρ y a la velocidad v del proyectil al cuadrado. También depende de una serie de factores del proyectil (sección, coeficiente aerodinámico, masa) que se agrupan en el llamado coeficiente balístico K. La ecuación diferencial es:

$$m\bar{r}'' = m\bar{g} + \bar{F}_R \Rightarrow \bar{r}'' = \bar{g} + \frac{\bar{F}_R}{m} = \bar{g} - \frac{\rho}{K}\|\bar{v}\| \cdot \bar{v}$$

Como el vector gravedad es $\bar{g} = \begin{pmatrix} 0 \\ -g \end{pmatrix}$ la aceleración queda: $\bar{r}'' = \begin{pmatrix} 0 \\ -g \end{pmatrix} - \frac{\rho}{K} \cdot \|\bar{r}'\| \cdot \bar{r}'$

donde g es la gravedad (9.8 m/s2) y ρ la densidad aire (1.2 kg/m3). En cuanto a K, los proyectiles lanzados por el obús M2A1 (de nuevo según Wikipedia), tenían un coeficiente balístico K~14000 (kg/m2), siendo el alcance máximo del arma unos 11400 mt.

Modificar la función obus.m para que codifique las nuevas ecuaciones diferenciales usando K=14000 para el coeficiente balístico y ρ=1.2 kg/m3 para la densidad del aire.

Al igual que antes, trabajar vectorialmente, extrayendo r y r' del vector de estado s y calcular directamente r'' aplicando la ecuación anterior usando vectores. Luego usar r' y r'' para montar el vector columna sp (4x1) que devuelve la función. Recordad que en MATLAB, ||r|| = norm(r). Adjuntad la función obus.m modificada.

Resolver con las mismas condiciones iniciales del apartado anterior (usando rk4 con el mismo intervalo [0,70] y paso h=0.1). Adjuntar la nueva gráfica y determinar de nuevo el alcance con el cursor de datos. ¿Obtenemos ahora el valor (11400 mt) que nos han dado en las especificaciones del arma?

c) El (principal) fallo del modelo anterior es que la densidad del aire no es constante (ya que depende de la altura) y puede cambiar significativamente para las alturas que alcanza el proyectil. Resolver una tercera vez (RK4, mismo paso h=0.1, etc.) pero ahora teniendo en cuenta que la densidad del aire disminuye con la altura. En vez de usar ρ=1.2 constante, usar la fórmula:

$$\rho = 1.2 \cdot e^{-\frac{altura}{7000}}$$

Adjuntar código de la función obus.m con esta última modificación. Fijaros que la altura corresponde a la coordenada Y (2ª componente del vector de estado s).
Resolver y adjuntad la gráfica. ¿Alcance máximo? ¿Es un resultado más ajustado al real?

La simulación realizada corresponde a usar el arma a nivel del mar, ya que la altura (coordenada Y) inicial es 0. Si disparásemos el arma en un terreno a una altitud de 1000m, ¿su alcance sería mayor o menor? ¿Por qué? (contestar sin volver a resolver).

Ejercicio 3: En este ejercicio escribiremos un método adaptativo del tipo descrito en la clase anterior. Se basará en un método de orden 1 (Euler) usando Heun (RK2 orden 2) como control. En la "jerga" de los métodos adaptativos a esto se le llama un método 1+2. La idea es que en cada paso, partiendo del punto más reciente (t_k, y_k) evaluaremos:

$$K_1 = f(t_k, y_k)$$
$$K_2 = f(t_k + h, y_k + hK_1)$$

, y a partir de ellas euler=s1= $y_k + hK_1$ y heun=s2= $y_k + h\frac{(K_1 + K_2)}{2}$.

usando Heun como el resultado "correcto" para estimar el error cometido.

Nuestra función tendrá la siguiente definición:

```
function [T S]=adapta12(fun,Tspan,y0,tol)
```

- **fun:** puntero a la función yp=f(t,y) con la ecuación diferencial a resolver.
- **Tspan:** intervalo a resolver desde t0=Tspan(1) hasta t_fin=Tspan(2)
- **y0:** vector de condiciones iniciales de la solución en t0 de dimensión m x 1.
- **tol:** es un nuevo parámetro (en vez del paso h) que indica la cota de error que el usuario desea alcanzar. Ahora el paso h ya no es constante, sino que el algoritmo lo va ajustando tratando que el error de la solución no exceda este valor tol.

Los argumentos de salida son los mismos que antes:

- T es el conjunto de instantes de tiempo donde estimamos la solución.
- S es la solución, organizada como una matriz m x N. Contiene la estimación de las m componentes de la solución en los N instantes de tiempo.

Partir del template adapta12.m suministrado y completarlo con los siguientes pasos de un algoritmo adaptativo:

1. Extraer el punto inicial (T0) y final (Tfin) del intervalo y usarlos para fijar el h del primer paso que demos. Usad como paso inicial el ancho del intervalo/20.

2. Inicializar T al tiempo inicial T0 y la solución S al vector de condiciones iniciales y0.

3. while (T(end)<Tfin) (mientras no se ha alcance el final del intervalo), repetir:

   a) Definir el punto de partida de este paso: t = T(end) y = S(:,end)

   b) Calcular K1 y K2 a partir de (t,y). Usando K1 y K2 calcular los resultados s1 y s2 que nos dan las fórmulas de Euler y Heun respectivamente.

c) Estimar el error absoluto cometido en el paso si suponemos que el resultado de Heun (s2) es el "correcto" frente a Euler (s1). Por simplicidad usaremos como estimación del error:

$$E = \|s_1 - s_2\|$$

donde ||·|| es la norma de un vector y se calcula como norm(·) en Matlab.

d) Calcular el ratio r entre el error E estimado y la tolerancia especificada por el usuario: r=E/tol

e) Si r<=1 es que Erel<=tol y aceptamos la iteración, por lo que añadimos los resultados recién calculados a la solución:

T = [T t+h]; S = [S s2];

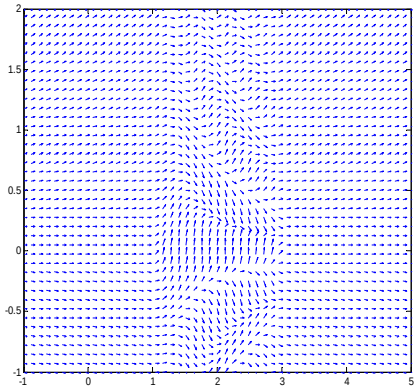
f) Independientemente de si el paso se ha aceptado o no, modificamos el paso h según la fórmula vista en clase para ajustarnos a la tolerancia pedida. Para estar más seguros de que cumplimos las especificaciones multiplicar el nuevo paso h obtenido por un factor adicional de 0.8.

g) Finalmente, antes de terminar el bucle, comprobad si (T(end)+h) es mayor que Tfin, lo que haría que en el siguiente paso nos saliéramos del intervalo. En ese caso haced h=(Tfin-T(end)) asegurando así que el último paso nos lleva directamente a Tfin.

Adjuntar vuestro código de la función adapta12 una vez completada

---

Aplicación del método: Consideremos la ecuación diferencial codificada en la función fun_rara.m suministrada. La figura adjunta muestra el campo de pendientes de dicha ecuación diferencial. Hay una zona (entre t=1 y t=3) donde las pendientes son muy "cambiantes", pero antes y después las cosas están más "tranquilas".



Usar el método adapta12 para resolver esta ecuación diferencial en el intervalo [0, 4] con la condición inicial y(0)=0.84. Especificad una tolerancia TOL=5·10^-4. ¿En cuántos puntos calcula la solución el método adaptativo?

Hacer un plot(T,S) para mostrar la solución obtenida. Repetir el plot con la opción 'bo:' que además de la curva muestra los puntos donde se ha calculado la solución. Adjuntar ambas gráficas. ¿Cuál de ellas ilustra mejor el funcionamiento de un método adaptativo?

Ahora el parámetro T (tiempos) de salida sí que nos aporta información adicional, al no ser solo una colección de puntos equiespaciados (como cuando h era constante). Haced un plot(S) en vez de un plot(T,S). Adjuntad la figura resultante. ¿Por qué los resultados parecen diferentes en una y otra gráfica?

Usando los datos del parámetro T de salida (los tiempos donde se ha evaluado la solución) hacer una gráfica del tamaño de paso h que usa el programa adaptativo en función del tiempo. Podéis usar la función diff() de MATLAB que resta los valores consecutivos de un vector. Adjuntar la gráfica. ¿Cuál es el paso máximo y mínimo usado durante el proceso adaptativo? ¿En qué zonas se dan ambos casos?