Ejercicio 1: Vamos a implementar el método multi-paso explícito (orden 2) de la familia Adam-Bashford, cuya fórmula es:

$$y_{k+1} = y_k + \frac{h}{2}(3f_k - f_{k-1})$$

Cread una función met_AB.m con el siguiente template:

```
function [T S]=met_AB(fun, Tspan, y0, h)
...
return
```

Como luego lo vamos a aplicar a una ecuación diferencial 1D, no hace falta considerar una implementación vectorial: asumiremos que la condición inicial y0 es un número y que la solución final S es un vector 1 x N con una sola fila.

La inicialización es muy similar a un método de un paso:

1) Creamos el vector de tiempos T con un salto h desde t inicial a t final (dados en el intervalo Tspan) y determinamos su longitud N.
2) Reservamos vector F (1 x N) para guardar las sucesivas evaluaciones de fun(t,y).
3) Reservamos un vector S de tamaño 1 x N para guardar la solución calculada y asignamos la condición inicial al primer punto de la solución S(1)=y0.

Como tenemos un método de 2 pasos necesitamos conocer también S(2) antes de poder aplicar la fórmula del método. Para "arrancar" el método usaremos un RK2 (método de Heun, también de orden 2) para calcular el siguiente valor de la solución, S(2):

```
K1=fun(T(1),S(1)); K2=fun(T(2),S(1)+h*K1); K=(K1+K2)/2; S(2)=S(1)+K*h;
```

Como en un método multipaso usamos las evaluaciones anteriores de la función, evaluad también fun(T(k),S(k)) para k=1,2 y guardar los valores obtenidos en F(1) y F(2).
Con eso ya tenemos todo listo. Solo falta montar un bucle (desde k = 2 hasta N-1) y aplicar la fórmula del método, que con en notación es:

$$S_{k+1} = S_k + \frac{h}{2}(3 \cdot F_k - F_{k-1})$$

donde S(k) es la solución en el punto anterior y F(k), F(k-1) los valores de la función fun en los 2 puntos anteriores. Una vez obtenido S(k+1) evaluar fun() en T(k+1), S(k+1) y guardar el resultado en F(k+1), estando listos para dar el siguiente paso del bucle.
Adjuntar código de vuestra función met_AB.m una vez completada.

Vamos a aplicar este método a la ecuación diferencial $\begin{cases} y'(t) = \cos(t) \cdot \sin(\exp(y)) \\ y(0) = 2 \end{cases}$ que tenéis ya codificada en la función suministrada ed.m.

Usando met_AB(), resolver esta ED en el intervalo [0,8] con un paso h=0.2 y guardar la solución en S1. Repetir con un paso h=0.02, guardando la nueva solución en S2.

Hacer una gráfica con las dos soluciones usando `plot(T1,S1,T2,S2)`.Tras hacer el plot, usad `legend({'h=0.2','h=0.02'});` para identificar ambos casos. Adjuntar gráfica.
¿Parece fiable la solución con h=0.2?

Ejercicio 2: En este ejercicio implementaremos un método predictor-corrector para ver si el método implícito del corrector puede reducir los problemas observados al resolver el problema anterior con un paso relativamente grande (h=0.2).

Para ahorrarnos trabajo el predictor (explícito) será el mismo método de Adam_Bashford (orden 2) que se implementó en el ejercicio anterior. De esta forma, gran parte del código será muy similar. Copiaremos el código anterior met_AB.m en otra función met_PC.m y la modificaremos para añadirle la parte del corrector. Como método corrector (implicito) usaremos Adam-Moulton de orden 2 (método del trapecio):

$$y_{k+1} = y_k + \frac{h}{2}\left(f_{k+1} + f_k\right)$$

El proceso inicial de reservar los vectores (T, S, F) y usar Heun para inicializar el 2º punto de la solución es igual que antes y no hay que tocar nada en el código. Las diferencias están dentro del bucle donde vamos calculando las soluciones. En cada paso del bucle:

1) Calcular el valor dado por el método predictor anterior. Esta parte es idéntica al código que ya tenéis. La diferencia es que el resultado no es la solución definitiva, por lo que en vez de guardarlo en S(k+1) le llamaremos P (solución del Predictor).

2) A continuación montamos un bucle con la iteración del corrector, aplicando de forma reiterada la fórmula iterativa del método implícito al valor de P:

$$P_{new} = y_k + \frac{h}{2}\left(f(t_{k+1}, P) + f(t_k, y_k)\right)$$

Notad que en la fórmula anterior el valor de $f(t_k,y_k)$ lo tenemos guardado en F(k), así que no hay que volver a evaluarlo. Con nuestra notación la fórmula quedaría:

$$P_{new} = S(k) + \frac{h}{2}\left(fun(T(k+1), P) + F(k)\right)$$

   a) En cada paso del bucle, tras aplicar esta fórmula calcular la diferencia entre P y Pnew, delta =abs(Pnew-P) y actualizar P con el valor recién calculado Pnew.

   b) Repetir el bucle mientras que la diferencia entre pasos sucesivos del corrector (delta) sea mayor que 10^-5.

3) Una vez que salimos del bucle el último resultado P se acepta como la solución final del método y se guarda en S(k+1). Igual que en el código del ejercicio anterior, una vez que tenemos S(k+1) se evalúa la función fun() en T(k+1), S(k+1) y se guarda el resultado en F(k+1).

Al terminar tendremos en el vector S la solución obtenida del método predictor-corrector. Adjuntar vuestro código de met_PC.

Vamos a comparar los resultados de los dos métodos (met_AB y met_PC) para resolver la ecuación diferencial del ejercicio anterior en el mismo intervalo [0,8] con el paso h=0.2 que antes nos daba problemas. Guardar las soluciones en S1 e S2.

Añadiremos a la comparación los resultados de la función ode23 de MATLAB:

```
[T3 S3]= ode23(@ed,[0 8],y0);
```

Siguiendo el ejemplo anterior haced una gráfica superponiendo las tres soluciones. Añadid una leyenda etiquetando los tres casos como 'AB' (método Adam-Bashford 2), 'PC' (Predictor-Corrector) y 'ode23' (función de MATLAB). Adjuntar código y gráfica resultante. ¿Cuántos puntos ha usado el método adaptativo (ode23) para recorrer el intervalo dado?

Veréis que no hay acuerdo entre ninguno de los tres métodos que nos indique cuál sería la verdadera solución.

El método de MATLAB ode23 es adaptativo y se puede modificar la cota de error deseada cambiando sus opciones. Volver a correr ode23 con una especificación más estrictas (10^-6) para el máximo error relativo deseado:

- Usar opt=odeset() con la propiedad `'RelTol'` para asignar la tolerancia requerida de 10^-6. Añadid también `'Stats','on'` para que al finalizar MATLAB os de algunos datos sobre el número de pasos necesarios, evaluaciones de fun(), etc.

- Llamar a ode23 pasándole las nuevas opciones: `[T3 S3]= ode23(@ed,T,y0,opt);`

¿Cuántos puntos necesita ahora el método adaptativo ode23 para la nueva precisión? ¿Cuántas evaluaciones de la función ed(t,y) ha necesitado?

Repetir el gráfico anterior con las tres soluciones y adjuntadlo.
¿Cuál de las 3 soluciones era la mejor en la comparación inicial?

Opcional (dejad para el final): Añadir código en vuestra función met_pc para que en cada paso se guarde el número de iteraciones hechas por el corrector. Guardar los resultados en un vector n_iter y devolverlo como 3er argumento de salida. El vector n_iter se inicializará previamente con ceros con el mismo tamaño 1xN que la solución. Volver a resolver (con el mismo paso h=0.2) y haced una gráfica del número de iteraciones que necesita el corrector en función del tiempo T. Adjuntar la figura resultante.

¿Cuál es el máximo número de iteraciones que tiene que hacer el corrector en un paso?

En el método Predictor-Corrector ¿cuántas evaluaciones de ed(t,y) se hacen?
¿Cuántas de ellas corresponden a la parte del predictor? ¿Y a la parte del corrector?
Comparar con el numero de evaluaciones de la función usadas por el método ode23().

Ejercicio 3: (Este ejercicio corresponde al primer apartado de la práctica final)

La fuerza gravitatoria ejercida por un cuerpo masivo (Tierra o Luna) sobre una nave viene dada por:

$$\bar{F} = -\frac{GMm}{\|r\|^3}\bar{r} \quad , \quad \text{donde}$$

- $\bar{r}$ es el vector desde el centro del planeta a la nave: si $\bar{x}$ es la posición de la nave y $\bar{X}$ la del planeta $\bar{r} = \bar{x} - \bar{X}$.
- $\|r\|^3$ es la norma de dicho vector al cubo. En MATLAB tenéis la función norm().
- G es la constante gravitación universal y M la masa del planeta (Tierra/Luna).
- m es la masa de la nave.

Una nave en su viaje entre la Tierra y la Luna (despreciando la influencia del Sol y otros efectos menores) sentirá tanto la fuerza de la Tierra como la de la Luna:

$$\bar{F} = -\frac{GM_1 m}{r_1^3}\bar{r}_1 - \frac{GM_2 m}{r_2^3}\bar{r}_2$$

siendo M1, M2 las masas de Tierra y Luna y r1 y r2 los vectores desde la Tierra y la Luna a la nave Apolo. Si la posición de la nave es r en unos ciertos ejes y las posiciones de la Tierra y la Luna son rT y rL respectivamente, los vectores r1 y r2 se calculan como:

$$\bar{r}_1 = \bar{r} - \bar{r}_T \qquad \bar{r}_2 = \bar{r} - \bar{r}_L$$

La aceleración sobre la nave vendrá dada por: $\bar{a} = \frac{\bar{F}}{m} = -\frac{GM_1}{r_1^3}\bar{r}_1 - \frac{GM_2}{r_2^3}\bar{r}_2$

Durante toda la práctica trabajaremos en **unidades de km para las distancias y segundos para el tiempo**, con las velocidades en km/s. En estas unidades, las constantes GM1, GM2 son `GM1=3.99e+05` km^3/s^2 y `GM2=4.90e+03` km^3/s^2 (estas constantes están ya definidas en el esqueleto del programa suministrado).

El problema se complica porque también la Luna está girando en su órbita alrededor de la Tierra. La aceleración sobre la Luna, debida a la Tierra, viene dada por:

$$\bar{a}_L = -\frac{GM_1}{r_{TL}^3}\bar{r}_{TL}$$

donde M1 es la masa de la Tierra y $\bar{r}_{TL}$ el vector Tierra-Luna $\bar{r}_{TL} = (\bar{r}_L - \bar{r}_T)$.

Finalmente, aunque mucho menos, la Tierra también se mueve por la influencia de la Luna. La aceleración sobre la Tierra, debida a la Luna es:

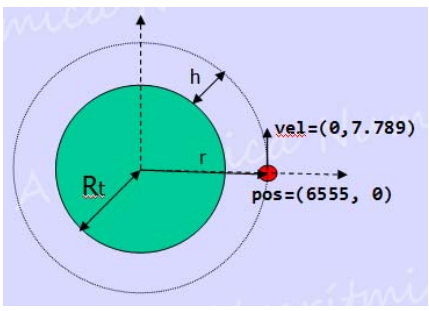
$$\bar{a}_T = -\frac{GM_2}{r_{LT}^3}\bar{r}_{LT}$$

con M2 la masa de la Luna y $\bar{r}_{LT}$ el vector Luna-Tierra $\bar{r}_{LT} = (\bar{r}_T - \bar{r}_L) = -\bar{r}_{TL}$.

Simplificaciones: Como tenemos que seguir la pista a la nave, la Luna y la Tierra, el vector de estado tendrá 3 posiciones (r_n, r_L, r_T) y 3 velocidades (V_n, V_L, V_t). Para que no sea demasiado complicado nos limitarnos al caso 2D, suponiendo que Tierra, Luna y nave están en el mismo plano XY y por lo tanto bastan dos coordenadas para describir tanto sus posiciones (x,y) como sus velocidades (vx,vy). **Por lo tanto el vector de estado tendrá en principio 12 componentes**.

Como siempre, además de las ecuaciones diferenciales se deben especificar las condiciones iniciales. Para la nave su posición y velocidad iniciales (ejes X/Y) serán:

Posición nave (6555 , 0) km
Velocidad nave (0, 7.789) km/s



Esto es, empezamos (ver gráfica) sobre el eje X y girando en el sentido contrario a las agujas del reloj (vy positiva).

Como el radio de la Tierra son 6370 km, las condiciones anteriores corresponden a una órbita casi circular de unos 185 km (6555-6370) de altura (~100 millas náuticas). La nave ha llegado hasta dicha órbita impulsada por las tres etapas de un cohete Saturno.

Respecto a la Luna, su posición inicial está sobre el eje X a 384000 km (x=384000, y=0). Su velocidad inicial será de vx=0.0, vy=1.0 (en km/s):

Posición Luna (384000 , 0) km
Velocidad Luna (0.0, 1.0) km/s

Finalmente, la posición inicial de la Tierra está en el origen (x=0,y=0) y su velocidad (muy pequeña) será de vx=0, vy=-0.0123:

Posición Tierra (0 , 0) km
Velocidad Tierra (0.0, -0.0123) km/s

En esta práctica usaremos la función ode45 de MATLAB para resolver las ecuaciones diferenciales. El programa suministrado x_apolo.m comprende un script principal donde se especifican con la función odeset de MATLAB las opciones que usaremos en ode45:

```
opt=odeset('RelTol',1e-8,'OutputFcn',@graf,'Refine',8);
```

Con estas opciones se especifica un error relativo de 1e-8 y también se asigna la función graf() a la propiedad `'OutputFcn'`. La asignación de `'OutputFcn'` hace que el ode45, tras cada paso de integración llame a la función graf(), que se ocupa de actualizar los gráficos con las posiciones de la nave, Luna y Tierra. La función graf() ya está escrita e incluida en x_apolo, por lo que no hay que hacer nada con ella.

El fichero x_apolo incluye también la función apolo (ahora incompleta) que será donde se codifiquen las ecuaciones diferenciales del problema. Como siempre, la función recibe un tiempo t y un vector de estado s, y debe devolver su vector de derivadas en sp:

```
function sp = apolo(t,s)
% s = vector estado con posición s(1:2) y velocidad s(7:8) de la nave
%       (12x1)         posición s(3:4) y velocidad s(9:10) de la LUNA
%                      posición s(5:6) y velocidad s(11:12) de la TIERRA
GM1 = 3.99e+005; GM2 = 4.90e+003;
sp=NaN*s;

% Extraer del vector de entrada s posiciones/velocidades de Nave/Luna/Tierra
rn=s(1:2);  rL=s(3:4);  rT=s(5:6);
vn=s(7:8);  vL=s(9:10); vT=s(11:12);

sp(1:6)=s(7:12);  % Las velocidades son como siempre la 2ª parte del vector s de entrada

% Calculo de aceleraciones Nave/Luna/Tierra
...
end
```

Vuestra primera tarea será rellenar la función apolo(), para que codifique las ecuaciones diferenciales del problema gravitatorio. Basta codificar las expresiones presentadas antes para las aceleraciones de la nave, Luna y Tierra y guardarlas en las respectivas posiciones de sp que faltan por rellenar.

Como en problemas anteriores es mejor trabajar con vectores, de forma que obtengáis las aceleraciones como vectores 2x1 en vez de intentar hallar sus componentes x e y por separado.

Una vez completada la función apolo(t,s), en el script principal haced la llamada a ode45 para resolver nuestra ecuación diferencial:

- Definir el intervalo de tiempo en el que estamos interesados. En un principio resolved para un intervalo de 8 horas (t0 = 0, tf = 8 x 3600 segundos).

- Crear un **vector columna** S0 (12x1) con las 12 condiciones iniciales del problema (posiciones nave/Luna/Tierra + velocidades nave/Luna/Tierra).

- Usar ode45 para resolver la ED: `[T S]=ode45(@apolo,[t0 tf],S0,opt);`

Si todo va bien veréis el gráfico actualizarse (debido a la función graf). La nave debe verse dando vueltas en la órbita de aparcamiento y la Luna girando en torno a la Tierra (el movimiento de la Tierra es inapreciable en esas 8 horas). Adjuntar código de apolo().

De la solución S extraer las coordenadas (x,y) con la trayectoria de la nave y representar su órbita haciendo `plot(x,y,'r)`. Superponer sobre ella un círculo de radio Re que represente la Tierra haciendo:

```
th=(0:0.01:2*pi);plot(Re*cos(th),Re*sin(th),'b');
```

Adjuntad el gráfico. ¿Observáis algún problema?

Lo que sucede es que lo importante son las coordenadas de la nave RESPECTO a la Tierra. Para obtenerlas extraer también las coordenadas (xT,yT) de la Tierra de la solución S y hacer plot(x-XT,y-yT). Adjuntad nueva gráfica. ¿Es ahora una órbita estable?

Finalmente, con la información anterior (x-xT, y-yT), cread un gráfico de la ALTURA de la nave sobre la superficie, usando el radio terrestre RT = 6370km definido en el programa.

Adjuntad gráfica/código usado. ¿Es una órbita circular perfecta? ¿Entre qué alturas oscila?

Determinar sobre la gráfica cuánto tiempo tarda en dar la nave una vuelta a la Tierra.