El objetivo de esta práctica será estudiar la trayectoria de las naves del programa Apolo en su viaje de la Tierra a la Luna. En esta práctica usaremos las funciones que tiene MATLAB para la resolución de nuestras ecuaciones diferenciales. En particular usaremos ode45 (un método adaptativo basado en un RK de 4º orden junto con un RK de 5º orden para monitorizar el error).

**Para saber más:**

- "A Man on the Moon: The voyages of the Apollo astronauts" ,Andrew Chaikin
- "Returning from space (reentry)" (recurso web).
- "How Apollo flew to the Moon", David Woods
- Artículos de Wikipedia para los datos de los diferentes sistemas de propulsión.

A continuación tenéis un resumen de algunos de los datos/constantes que usaremos. Las unidades que usaremos serán: kg para la masa, km para las distancias y segundos para el tiempo. Por lo tanto las velocidades estarán en km/s.

**Tamaños, distancias y masas Tierra/Luna:**

$R_T$ = Radio Tierra = 6370 km
$R_L$ = Radio Luna = 1735 km
Distancia media Tierra-Luna: 384000 km
$GM_T$ = 3.99e+005;
$GM_L$ = 4.90e+003;

**Parámetros de la nave en el momento de la TLI (Trans Lunar Insertion):**

Masa de la nave en el momento de la TLI: 140 Ton = 140000 kg.

Duración de la ignición TLI: unos 5 minutos, en el sentido del movimiento (deltaV positiva ganando velocidad)

Gasto de combustible del motor (J2 engine): 240 kg/s
Fuerza de empuje (thrust) del cohete: 1000 kg·km/s2

**Parámetros del módulo lunar (LM) en su descenso:**

Masa del LM: 14500 kg, de los cuales 8200 kg eran de combustible para el descenso.

Empuje del motor de descenso (DPS): máximo de 45 kg·km/s2
Gasto de combustible del motor DPS: máximo 15 kg/s

El motor DPS era regulable, pudiendo bajar su empuje (y su gasto de combustible).

## 1.Planteamiento y órbita inicial:

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 cuerpo atractor 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 usaremos 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:

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

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: $\bar{a} = \frac{\bar{F}}{m} = -\frac{GM_1}{\|\bar{r}_1\|^3}\bar{r}_1 - \frac{GM_2}{\|\bar{r}_2\|^3}\bar{r}_2$

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` $\mathrm{km}^3/\mathrm{s}^2$ y `GM2=4.90e+03` $\mathrm{km}^3/\mathrm{s}^2$ (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:

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

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 poco, la Tierra también se mueve por la influencia de la Luna. La aceleración sobre la Tierra, debida a la Luna es:

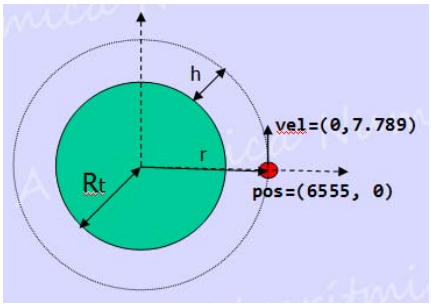
$$\bar{a}_T = -\frac{GM_2}{\|\bar{r}_{LT}\|^3}\bar{r}_{LT}$$

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 hay 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/seg**

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**

El programa suministrado x_apolo.m comprende un script principal que usará la función ode45 para la resolución de las ecuaciones diferenciales. En este script se especifican las opciones del "solver" ode45 usando la función odeset de MATLAB.

```
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. ¿Cuál parece ser el 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.

## 2. Fase de TLI (Trans Lunar Injection)

La órbita de aparcamiento se usaba para que los astronautas y el Centro de Control verificasen que todos los sistemas funcionaban correctamente antes de proseguir el viaje a la Luna. En caso afirmativo se autorizaba a la tripulación a iniciar la siguiente etapa del viaje, la llamada fase **TLI (Trans Lunar Injection).**

La TLI consistía en encender la última etapa del cohete Saturno que aún estaba acoplada a la nave, dotada de un propulsor J2 con una fuerza de empuje (thrust) de 1000 kg·km/s2. Como la masa de la nave era de unos 140000 kg, el propulsor proporcionaba una aceleración:

$$a = F/m = 1000/140000 = 1/140 \text{ km/s2} = 0.00742 \text{ km/s2} = 7.42 \text{ m/s2}$$

ligeramente inferior a la de la gravedad (g=9.8 m/s2). Sin embargo, debido al gasto de combustible (240 kg por segundo) la masa de la nave disminuía rápidamente durante el proceso, mientras que el empuje del motor era constante. Al final de la ignición la nave

había perdido la mitad de su masa (en combustible gastado), por lo que la aceleración se duplicaba, siendo al final de unos 15 m/s2 (~1.5 veces la gravedad g).

Tras la ignición TLI la última etapa del Saturno se separaba y el módulo de control y servicios (CSM) con una masa total de 30 toneladas) estaba en rumbo a la Luna. Para el resto del viaje, el motor con el que se contaba era el llamado SPS (Service Propulsión System), mucho más pequeño que el propulsor Saturno inicial. Para evitar la pérdida de la nave (y su tripulación) si el motor SPS fallaba, la mayoría de las naves Apolo que han ido a la Luna (sobre todo en los primeros viajes) seguían lo que se llama una trayectoria de regreso gratuita. Se trataba de planificar la TLI para obtener una trayectoria de regreso a la Tierra incluso cuando nos fuera imposible usar ningún tipo de propulsión a partir de la TLI inicial.


El emblema oficial de la tripulación del Apolo 8 (la primera nave tripulada en orbitar la Luna) hacía un juego de palabras gráfico entre el número de la misión y la representación de la trayectoria en forma de 8 alrededor de la Luna. En esta primera misión tener asegurada la vuelta era especialmente importante pues sería la primera vez que se usase el propulsor SPS en órbita lunar.

La idea es usar la fuerza gravitatoria de la Luna como una especie de montaña rusa para girar y retomar una trayectoria de vuelta a la Tierra. Diseñando cuidadosamente la TLI (momento del inicio y duración de la ignición) es posible lanzar la nave hacia la Luna en una trayectoria que le lleve de vuelta a la Tierra sin necesidad de hacer nada más. Gracias a esta trayectoria "de seguridad" el Apolo 13 pudo volver a la Tierra a pesar de sufrir graves daños al inicio de su viaje que impidieron el uso del propulsor SPS.

El efecto de una ignición lo incorporaremos en nuestro programa a través de la variable global motor, que modificaremos en el programa principal y que (al ser global) es visible dentro de nuestra función apolo(). En la función apolo() comprobaremos su valor y si motor=1, a la aceleración de la nave (además de la gravedad siempre presente) se le añadirá la aceleración producida por el motor. Obviamente las aceleraciones de Luna y Tierra no se ven afectadas por el motor.

Como gran parte de la masa de la nave es combustible, al encender el motor la masa de la nave deja de ser constante y por lo tanto se debe añadir a nuestro vector de estado que pasa así a tener 13 componentes. El valor inicial de esta componente será la masa inicial de la nave en el inicio de la TLI (140000 kg). En cuanto a su derivada, si el motor está parado (motor=0) es nula (la masa de la nave no cambia) y con el motor encendido (motor=1) la derivada coincide con el gasto de combustible (-240 kg/s). El signo es negativo porque la masa se reduce (derivada negativa).

La ignición de la TLI es lo que se llama una ignición hacia delante, es decir, la aceleración se aplica en la dirección de la velocidad de la nave respecto de la Tierra, de forma que aumentamos su velocidad con respecto a la Tierra para conseguir abandonar la órbita terrestre. La fórmula a aplicar sería:

$$\bar{a}_{motor} = \left(\frac{F}{m}\right) \cdot \frac{(\bar{v}_N - \bar{v}_T)}{\|\bar{v}_N - \bar{v}_T\|}$$ en nuestras unidades de $\frac{\text{km}}{\text{s}^2}$

con $v_N$ la velocidad de la nave y $v_T$ la de la Tierra. F es la fuerza del propulsor (F=1000 kg·km/s2) y m la masa actual de la nave (componente 13ª del nuevo vector de estado).

Esta aceleración debe sumarse (cuando el motor esté encendido, motor=1) a la ya calculada por los efectos de la gravedad (siempre presente). Adjuntad vuestro código de la función apolo() añadiendo el empuje del motor en función del valor de motor y con la nueva componente (masa de la nave) en el vector de estado.

En el programa principal definiremos el momento inicial T0 de la ignición y su duración dT0 (ambos en segundos) y haremos 3 llamadas a ode45:

1. En la primera usamos el intervalo [0 T0] con el motor apagado (motor=0). Las 13 condiciones iniciales son las posiciones/velocidades/masa especificadas al inicio. Durante esta fase nos mantenemos en la órbita inicial.

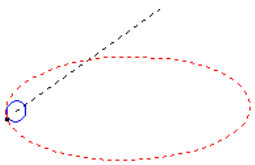
2. En la segunda (intervalo [T0 T0+dT0]) encenderemos el motor (motor=1). Como condición inicial usaremos el estado final del paso 1. Sólo durante ese tiempo dT0 (~ 5 minutos) estaba encendido el propulsor durante el viaje a al Luna.

3. Partiendo del estado final del paso 2, volvemos a resolver la ecuación gravitatoria (motor=0) desde T0+dT0 hasta el final. Para el tiempo final usaremos ahora **8 días** (tf=8*24*3600 segundos) que es algo más de lo que solía durar el viaje ida/vuelta a la Luna.

Recordad que la función ode45 de MATLAB devuelve la solución como una **matriz N x 13**. Transponerla (S=S') y usar su última columna S(:,end) como punto inicial de la siguiente fase del viaje. Código de las 3 llamadas a ode45 en el programa principal.

Una vez modificada apolo() y añadidas las llamadas anteriores a ode45 en el programa podemos experimentar para determinar los valores de T0 y dT0 necesarios para conseguir una buena trayectoria en 8 pasando por detrás de la Luna y que nos traiga de vuelta.

El tiempo T0 marca la dirección de la trayectoria, ya que la nave saldrá en la dirección de la velocidad que tuviera en ese momento. La duración del encendido (dT0) determina lo lejos que llegue la nave. Se trata de dar valores a T0 y dT0 en el script principal, ejecutarlo, y observar los resultados. Si probáis con T0=2500 y dT0=300, obtendréis una figura similar a la adjunta. Se ve que la trayectoria se queda corta y además, apunta hacia donde estaba la Luna inicialmente (pero cuando llega ya no está).

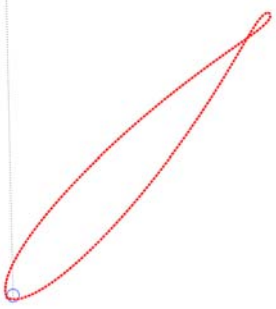


Para ser capaces de obtener tanteando un resultado decente hemos de atacar el problema de una forma más o menos metódica. Un posible proceso es el siguiente:

1. Partir de los datos anteriores y aumentar dT0 (a saltos de 5 segundos) hasta ver el primer dT0 (múltiplo de 5) para el que la trayectoria de la nave sobrepasa la órbita lunar (aunque no acierte con la posición actual de la Luna). Mantener este valor de dT0 durante el resto de las pruebas.

2. Con dT0 fijado, aumentad T0 (en múltiplos de 50 o 100) para apuntar mejor a la Luna, hasta que veáis que la nave cruza la trayectoria lunar justo "por delante" de la Luna, describiendo un bucle y volviendo más o menos en dirección a la Tierra.

3. Afinad T0 (a saltos de 10 o de 5) hasta ver una trayectoria simétrica en forma de 8 que vuelva a la Tierra con una curva similar a la de ida (ver figura adjunta).



**Ángulo de reentrada:** Si vuestra trayectoria vuelve a la Tierra veréis que la atraviesa sin pararse. Para evitar esto he escrito en x_apolo la función vuelta(), para indicarle a ode45 que detenga el proceso si la nave llega a los 120 km de altura (altura de la reentrada):

```
function [val,term,dir] = vuelta(t,s)
global Re
% val = valor que se anula si se cumple la condicion deseada (altura llega a 120 km)
rn=s(1:2); rT=s(5:6); h = norm(rn-rT)-Re; val=h-120;
term =  1;   % Detiene la iteración si se cumple condición
dir  = -1;   % Activar si cruce por cero es en sentido negativo (bajando)
end
```

La función de eventos recibe el tiempo t y vector de estado s y devuelve tres argumentos:

1) El parámetro val es el valor que define el evento al anularse. En este caso es que la altura sobre la superficie terrestre llegue a los 120 km.
2) El parámetro term lo fijaremos a 1 (detener el proceso si se alcanza la condición).
3) Finalmente dir indica si el evento sucede al alcanzar el valor dado subiendo (1) o bajando (-1). En este caso definimos el evento en caso de descenso (dir=-1).

Para que funcione, asociar la función vuelta (@vuelta) a las opciones de ode45 a través de la propiedad `'Events'` de odeset(). Buscar ahora una trayectoria que vuelva a la Tierra de forma que la función de eventos detenga el proceso. Valores de T0 y dT0. Adjuntar la gráfica de trayectoria. ¿Duración del viaje (en segundos y en días/horas/minutos)?

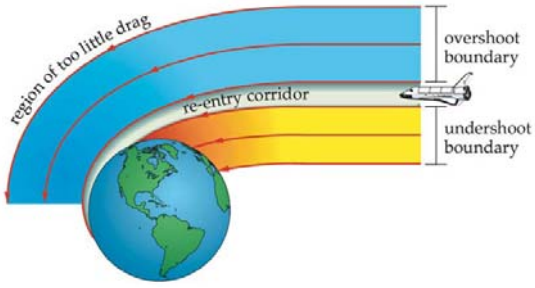
De la última solución de S podéis velocidad de la nave con respecto a la Tierra en el momento de la reentrada. Recordad que las rutinas de MATLAB devuelven sus resultados traspuestos respecto a los nuestros (el vector de estado en el momento de la reentrada no será la última columna de S, sino la última fila). ¿Velocidad de la nave con respecto a la Tierra en el momento de la reentrada (en km/s y en km/hora)?

En la reentrada es fundamental conocer el ángulo entre la dirección de la nave (dada por el vector velocidad) y el plano horizontal en el momento de la reentrada. Para una posición (x,y) el vector que define la horizontal (tangente) en dicho punto es $\underline{t}$=(-y,x). Dada la posición (x,y) y velocidad (vx,vy) de reentrada el ángulo de reentrada es:

$$\cos(\varphi) = \frac{\bar{t}\cdot\bar{v}}{\|\bar{t}\|\cdot\|\bar{v}\|} = \frac{\left(x\cdot v_y - y\cdot v_x\right)}{\|\bar{t}\|\cdot\|\bar{v}\|}$$

Tanto la posición como la velocidad en la fórmula anterior son **con respecto a la Tierra**, por lo que tendréis que tener en cuenta la posición/velocidad de la Tierra.

Una entrada demasiado "perpendicular" (un ángulo demasiado grande, en la zona amarilla) implica una frenada demasiado brusca en la atmósfera y la pérdida de la nave por las excesivas fuerzas y/o calentamiento. Si la entrada es demasiado tangente (ángulo bajo, en la zona azul) la nave no frena lo suficiente y en el peor de los casos podría llegar a "rebotar" en la atmósfera superior, perdiéndose de nuevo en el espacio.




Para las naves Apolo volviendo de la Luna, el ángulo de reentrada ideal era ~ 6.5º y no podía apartarse más allá de 1 grado en un sentido u otro (5.5º / 7.5º).

Completad la función phi(s) en x_apolo para que reciba el vector de estado s y calcule la posición/velocidad de la nave con respecto a la Tierra, devolviendo el ángulo de reentrada en grados (usad la función acosd(x) para obtener el ángulo en grados).
Adjuntar vuestro código de la función phi() y valor del ángulo para la trayectoria anterior.

Volver a jugar con el momento inicial T0 de la ignición para intentar volver con un ángulo de reentrada que permita que vuestra tripulación sobreviva (idealmente entre 6º y 7º). Es posible que tengáis que usar un valor fraccionario para T0. Es mejor no tocar dT0, que es mucho más sensible y difícil de ajustar con este enfoque de prueba y error.

Mejores valores encontrados para T0 y dT0. Ángulo de reentrada obtenido.
Gráfica de la trayectoria obtenida para vuestra mejor combinación de T0 y dT.

**Adjuntad fichero x_apolo.m junto con el fichero de respuestas en un .rar o similar.**
**El fichero x_apolo debe incluir todas las funciones pedidas (apolo, phi, ...), de forma que sea autosuficiente para ejecutarse.**

**---------- Hasta aquí 45% de la nota ----------**

## 3. Optimización para conseguir objetivos de la misión (25%)

Aunque nos ha servido para hallar una trayectoria aproximada, está claro que el enfoque anterior de prueba y error no es muy profesional. Hemos conseguido ajustar el ángulo de reentrada óptimo tanteando con los parámetros T0 y dT0, pero el ángulo de reentrada no es la única especificación a cumplir por la trayectoria. Por ejemplo, para colocarse en una órbita lunar adecuada la nave tenía que pasar muy cerca de la Luna, a tan solo unos 110 km de altura. En caso contrario la órbita lunar sería demasiado alta, y el módulo lunar no tendría suficiente combustible para descender.

Extraer de la última solución las coordenadas de la nave y la Luna y restarlas entre sí para obtener las coordenadas (x,y) de la nave **respecto** a la Luna. Calculad la distancia de la nave al centro de la Luna como r=sqrt(x.^2+y.^2). Usando la función min() calcular el valor mínimo del vector r y con el radio lunar Rm definido en el programa calculad la distancia mínima de la trayectoria a la superficie lunar. ¿Distancia mínima de vuestra trayectoria?

En este apartado automatizaremos el proceso de determinar T0 y dT0 planteándolo como un problema de optimización. Para ello, definiremos una función de coste que queremos que sea lo más baja posible y usaremos un algoritmo de optimización para intentar hallar su mínimo.

```
function coste=opt_trayectoria(X)
global motor
T0=X(1); dT=X(2);
opt=odeset('RelTol',1e-6,'events',@vuelta,'Refine',8);  % SIN GRAFICOS
% Partir de las condiciones iniciales S0,
% Resolver las tres etapas [0,T0] + [T0,T0+dT], [T0+dT, 8 dias]
% A partir de la trayectoria de la última etapa evaluar función de coste
end
```

La función de coste recibe un argumento de entrada X con los dos parámetros de la ignición (T0=X(1), dT0=X(2)) y devuelve un valor que se hace más pequeño cuanto más nos acerquemos a nuestros objetivos ideales (p.e que el ángulo de reentrada ~ 6.5º).

Se trata de completar la función opt_trayectoria dentro del script x_apolo. Tras resolver las ecuaciones diferenciales (como se hizo en el 1er apartado) partiendo del estado inicial S0 la función calculará varios parámetros a partir de la solución de la última etapa (entre T0+dT y 8 días) en función de los objetivos deseados para la misión. El valor devuelto en coste debe reflejar lo cerca que estamos de cumplir los objetivos.

### a) Distancia mínima a Tierra en el regreso y ángulo de reentrada.

Obviamente nuestro primer objetivo es que los astronautas regresen. Para ello vamos a calcular el momento del regreso en el que la trayectoria pasa lo más cerca de los 120 km de altura que definen el punto de reentrada. Lo primero es extraer las coordenadas (x,y) de la nave y de la Tierra durante la última etapa de la trayectoria y calcular las alturas de la nave sobre la superficie de la Tierra, obteniendo un vector h de alturas de la nave.

A partir de h buscaremos su valor mínimo (min_h), con la condición adicional de que h debe estar disminuyendo en ese punto para que el valor mínimo sea válido (evitando así escoger un valor del viaje de ida, cuando nos estábamos alejando de la Tierra).

Inicializar al valor de la altura del último punto de la trayectoria idx=N; min_h=h(idx), que probablemente será el valor más cercano. Luego haced un bucle desde k=2:N y en cada paso, comprobad si se verifican simultáneamente las condiciones h(k)<h(k-1) (indicando que nos estamos acercando a la Tierra) y (h(k)<min_h). Si se cumplen AMBAS condiciones actualizar el valor de min_h y guardar el índice en el que hemos encontrado dicho mínimo en idx. En MATLAB el operador lógico AND es &&.

Al terminar tendréis en min_h la altura mínima durante el viaje de regreso y en idx el índice de tiempos en el que ocurre. Pasar el vector de estado correspondiente a ese punto a la función phi() para obtener el ángulo de reentrada. Luego evaluamos la función de coste como:

```
coste = (min_h-120)^2 + (ang-6.5)^2
```

penalizando cualquier alejamiento (en uno u otro sentido) de las condiciones deseadas (altura 120 km, ángulo 6.5º).

El problema con esta función de coste es que podemos terminar con una trayectoria que nos lleve de regreso a 120 km de altura con un ángulo de 6.5º, pero que pase muy lejos de la Luna. Debemos añadir a coste una penalización adicional teniendo en cuenta nuestro objetivo de pasar muy cerca (~ 110 km) de la Luna.

### b) Determinación de la distancia mínima a la Luna durante trayectoria.

Hallar la mínima altura de la trayectoria respecto a la superficie lunar como hicimos antes. A partir de ella, añadir un nuevo término de error a la función coste calculada antes:

```
coste = coste + (min_h_lunar-110)^2
```

Al final de la función de coste poned un fprintf() para que vuelque los valores de min_h, ang y min_h_lunar junto con el valor de la función de coste.

Probad la función de coste con valores de ignición T0=3240 y dT=310 (partiendo de las condiciones iniciales S0). Debéis obtener una distancia mínima a la Tierra de 11697 km, un ángulo de 0.1º y una distancia mínima a la Luna de 1710.7 km. Con esos valores el valor de la función de coste sería de 136588561.3 ~ 1.3659e8

Volver a llamar a la función opt_reentrada() con los siguientes valores T0=3245, dT0=310 para la ignición. Volcar los nuevos resultados obtenidos.

Adjuntad código de la función opt_trayectoria.

Una vez completada la función de coste podemos usar una función de optimización de MATLAB (fminsearch) para hallar los valores de [T0,dT] que minimicen dicha función (idealmente haciendo cero los 3 errores y consiguiendo así cumplir con los tres objetivos). Usarla es muy sencillo:

```
opciones_opt = optimset('MaxFunEvals',1000,'MaxIter',1000);
X = fminsearch(@opt_reentrada,X0,opciones_opt);
```

- X0=[T0 dT] es una hipótesis inicial de los datos de ignición. Podéis usar algunas de vuestras pruebas iniciales que volvían más o menos a la Tierra.

- X es el resultado del algoritmo de optimización, un vector con los valores [T0, dT0] que mejor consiguen minimizar el valor de coste (idealmente reducirlo a 0).

Dependiendo de vuestros valores iniciales pudiera ser que el proceso de minimización no converja. Si es así, repetir el proceso cambiando ligeramente la hipótesis inicial. Con los datos anteriores (T0=3240, dT=310) a mi me converge en unas 150 iteraciones.

Indicad vuestros datos de partida (T0,dT) y los parámetros T0 y dT obtenidos tras el proceso de optimización (con 4 decimales). ¿Cuántas iteraciones necesitáis?

¿A qué distancia mínima de la Luna pasa vuestra trayectoria? ¿En qué instante de tiempo se alcanza dicha distancia?

Si todo había ido bien, una vez que la nave alcanzaba el punto más cercano a la Luna, encendía el propulsor SPS para frenar y quedar atrapada en una órbita lunar. Esta era la llamada fase LOI (Lunar Orbit Insertion), que no modelaremos en esta práctica.

Por seguridad este proceso se llevaba a cabo en dos etapas. Una primera frenada (LOI-1) les colocaba en una órbita elíptica de unos 320 x 110 km. Posteriormente una segunda frenada (LOI-2) más corta les dejaba en una órbita cuasi-circular de 110 km.

Desde esta órbita circular se iniciaba la fase final del descenso lunar. Esta es la fase que vamos a simular en el siguiente apartado. **El siguiente apartado es independiente de los anteriores. Aunque no hayáis implementado la parte de optimización podéis continuar haciendo la práctica.**

## 4. Descenso Lunar (30%):

Al estar muy cerca de la Luna en este apartado ignoraremos la influencia de la Tierra y consideraremos la Luna fija y en el centro de coordenadas. **El vector de estado tendrá ahora solo 5 componentes: la posición r (x,y) y velocidad r' (vx,vy) de la nave + su masa.** Como la Luna está fija (velocidad 0) en el origen (0,0) la posición y velocidad de la nave coinciden con su posición y velocidad con respecto a la Luna.

El módulo lunar (LM) se separaba del módulo de control y servicios (CSM) para inicial el descenso lunar. El LM estaba dotado de 2 motores independientes, para el descenso y ascenso. Al volver, el motor de descenso y la estructura de aterrizaje se quedaban en la superficie. En este apartado vamos a simular el descenso a la superficie, y solo nos preocuparemos de los datos del motor de descenso o **DPS** (Descent Propulsion System) con **empuje máximo de 45 kg·km/s2** y un **consumo de combustible de 15 kg/s**.

La principal característica de este motor es que podía ser regulado (los otros motores funcionaban únicamente en modo on/off dando siempre el 100% de su potencia). Por el contrario, el DPS tenía un modo 100% (con un empuje = 45) y un modo regulable, en el que su empuje variaba entre un 10% y un 60% del empuje máximo. Esta potencia variable era fundamental para conseguir controlar el descenso y llegar a la superficie con la mínima velocidad. En esta práctica ignoraremos estas limitaciones y consideraremos que este empuje regulable podía variar de forma continua entre 0 y 100%. Si se usaba un empuje reducido, el consumo de combustible era también proporcionalmente menor. Esto es, si p (entre 0 y 1) representa el % de impulso aplicado, el empuje E y gasto G del motor serían:

$$E = (45 \cdot p) \text{ kg·km/s2}$$
$$G = -(15 \cdot p) \text{ kg/s}$$

El descenso consistía de 4 fases diferenciadas de las cuales, en esta práctica, vamos a simular las tres primeras:

1. Descenso a una órbita más baja: un encendido del DPS de 30 segundos (al 25% de potencia) en dirección opuesta a su velocidad hacía caer al LM a una órbita más baja. Esta fase duraba hasta alcanzar la altura de 50000 pies (unos 15 km).

2. A partir de los 15km se iniciaba la fase de "descenso propulsado". Se encendía el motor al 100% pero no exactamente en la dirección opuesta a la velocidad de la nave, sino un poco (8º o 10º) desviada hacia "arriba", para ir reduciendo velocidad vertical. Esta fase terminaba al alcanzar los 7500 pies de altura (unos 2.25 km).

3. Al llegar a los 7500 pies se iniciaba la fase de "descenso controlado". A esta altura ya se disponía de datos fiables de radar para conocer la altura de la nave. Con estos datos de altura junto con la velocidad de descenso de la nave el sistema de control regulaba de forma variable el empuje del motor. El objetivo ideal era llegar a la superficie con velocidad nula, tanto en el sentido horizontal como en la vertical.

4. Como en el alunizaje siempre podía haber imprevistos, a 500 pies (150 mt) de altura se pasaba a un control semi-manual, que permitía al piloto alterar el empuje dado por el sistema de control, con objeto de elegir el punto más adecuado para aterrizar. Al llegar a esta fase era necesario que la velocidad de la nave fuese ya muy pequeña (del orden de 1-2 m/s o unos 5-10 km/h). Además debemos llegar a esta fase con una reserva de combustible que le dure al piloto dos o tres minutos para que pueda buscar un punto de aterrizaje adecuado.

El programa adjunto x_descenso.m (similar a x_apolo.m) es un template en el que iremos aplicando las distintas etapas del descenso. Los datos iniciales de la nave (órbita circular de unos 110 km de altura) ya están cargados en el vector de estado inicial:

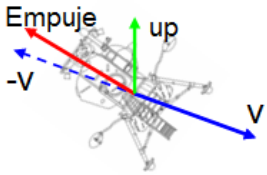
Radio Lunar: 1735 km. Posición LM: X=RL+15; Y=0; (en km)
Velocidad LM: Vx=1.63; Vy=0.00 (km/s)
Masa del LM: 14500 kg (de los cuales 8200 son de combustible para el descenso).

Dentro de x_descenso tenéis la función sp=descenso(t,s) (sin completar) que será la que codifique las ecuaciones diferenciales del proceso (similar a apolo dentro de x_apolo).

Tras separarse del CSM y antes de encender sus motores, el LM se mantenía en órbita sometido únicamente a la gravedad lunar. Completar la función descenso() para que refleje la fuerza de la gravedad lunar sobre la nave y propagar las condición iniciales durante 3 horas usando dichas ecuaciones. Al terminar hacer una gráfica de la ALTURA de la nave respecto a la superficie lunar durante ese tiempo. Adjuntad el gráfico ¿Entre qué alturas oscila? ¿Cuánto tarda la nave en completar una órbita alrededor de la Luna?

Durante esas 3 horas el módulo lunar verificaba sus sistemas y si estaban correctos se iniciaba la fase de descenso. Se trata de ir completando la función descenso() para simular el comportamiento del motor en las diferentes etapas. Como antes, usaremos una variable global motor para indicar en qué modo está funcionando el motor:

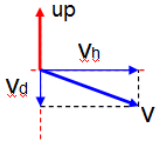
- **motor=0**) Caso anterior. Motor parado, solamente actúan fuerzas gravitatorias.
- **motor=1**) motor encendido al 25% (empuje=11.25 kg·km/s2) en sentido contrario a la velocidad. Se usa al principio de la primera etapa para iniciar el descenso.
- **motor=2)** Usado en la 2ª etapa, desde los 15 km a los 2.25 km de altura. El motor se enciende al 100% (empuje=45 kg·km/s2), pero con una dirección de aplicación que es más o menos opuesta a la velocidad como antes, pero añadiendo una cierta componente hacia arriba, para ir reduciendo su velocidad de bajada.




Para conseguir esto, además del vector unitario **vn,** contrario al vector velocidad **v,** se construye un vector unitario hacia arriba **up**, a partir de la posición r: **up**=r/norm(r). A partir de **vn** y **up** construimos el vector **d** = **vn** + 0.14·**up**. Este vector es el usado como dirección de aplicación del empuje de la nave y corresponde a apartarse unos 8º hacia arriba (**up**) del vector antivelocidad (**vn**). La aceleración del motor sería:

$$\bar{a}_{motor} = \left(\frac{45}{m}\right) \cdot \frac{\bar{d}}{\|\bar{d}\|}$$

- **motor=3)** Usado en la 3ª etapa. Se determina el vector "abajo" **down=-up** y se usa para hallar la velocidad de descenso vd, proyectando (con un producto escalar) el vector velocidad **v** sobre la dirección "abajo". Luego se calcula la altura h de la nave sobre la superficie. Conocidas la velocidad vd de descenso y la altura h, la aceleración que es necesario aplicar al LM para llegar a la superficie con velocidad nula es:




$$a_{motor} = \left( GM_2 \frac{\bar{r}}{\|\bar{r}\|^3} - 2 \cdot \left(\frac{v_d}{h}\right) \cdot \bar{v} \right)$$

El primer término de esta fórmula anterior cancela la gravedad lunar y el segundo se opone a la velocidad actual **v** de tal forma que (idealmente) dicha velocidad se anule al alcanzar la superficie.

La idea es usar el "throttle" del motor DPS para aplicar un empuje que cause justo la aceleración anterior. Para comunicar la aceleración $a_{motor}$ a la nave (de masa m) el motor debe ejercer un fuerza de:

$$E_{motor} = m \cdot \|a_{motor}\|$$

Este sería el empuje necesario. Si lo dividimos por el empuje máximo $E_{max}=45$ obtenemos el "throttle" p = ($E_{motor}$/45) a usar por el motor. El porcentaje p, aplicado al gasto máximo de 15kg/s, nos dará también el consumo de combustible.

El problema es que si p>1, el sistema de control nos pide una aceleración mayor de la que puede darnos el motor. En ese caso, lo único que puede hacer el motor es dar la máxima aceleración posible (que será inferior a la deseada por un factor p)

a_motor = a_motor/p

En este caso, al estar el motor funcionando a máxima potencia el gasto de combustible será de 15 kg/s.

Adjuntad el código de descenso()con las diferentes opciones del motor. Tras completarlo podremos programar las sucesivas etapas del descenso. De cara a depurar vuestro programa, en cuanto tengáis programada la opción de motor=1, probad a implementar la primera fase del descenso. Luego, tras programar la opción motor=2, implementad la 2ª fase y así sucesivamente. De esta forma será más fácil detectar errores y podéis tener un crédito parcial en esta parte aunque no completéis todas las fases del descenso.

**Para la primera etapa usad como tiempo y condición inicial el tiempo y estado final tras las 3h en órbita. Para las siguientes etapas usad el tiempo y vector de estado con el que terminái s cada fase para arrancar la siguiente.**

Fijaros que en las 3 etapas lo que gobierna el paso a la siguiente fase no es que pase un cierto tiempo, sino que se alcance la siguiente condición de alturas. Por ello al llamar a ode45 podéis usar un intervalo de tiempo exagerado (por ejemplo 3 horas) a partir del tiempo inicial de cada fase, porque quién va a cortar realmente el proceso es la función de eventos evento_altura().

Completar la función de eventos evento_altura() para que el proceso se detenga a la altura indicada por la variable global h_stop. De esta forma podemos usar la misma función de eventos para detener el proceso a otras alturas sin más que modificar la variable h_stop en el programa principal. Adjuntad la función una vez completada.

**Etapa 1)** Tras las 3h en órbita se enciende el motor al 25% en dirección opuesta a la velocidad (motor=1) durante 30 segundos, lo que provoca que la nave empiece a perder altura. Transcurridos esos 30 segundos se deja caer la nave (solo con la gravedad, motor=0) y se detiene el proceso (usando función de eventos) al llegar a 15 km de altura.

Tiempo (segundos) en el que la nave alcanza los 15 km de altura. Duración de esta fase.
Hacer una gráfica de la altura de la trayectoria durante esta fase. Adjuntadla.
Módulo de la velocidad en km/s y km/h al llegar a ese punto.

**Etapa 2)** De nuevo, usando como tiempo inicial y vector inicial el último tiempo y vector de estado del apartado anterior propagar la solución en el modo motor=2. Cambiar h_stop para que el proceso se detenga al alcanzar los 2.25 km (7500 pies) de altura.

Tiempo (segundos) al final de esta fase (2.25 km de altura). Duración de la fase.
Gráfica de la altura de la trayectoria en esta fase.
Velocidad horizontal y vertical en km/h al llegar a ese punto.
% de combustible restante en ese momento.

**Etapa 3)** Partiendo del punto anterior propagar la solución usando el modo motor=3. Cambiar h_stop para que el proceso se detenga a 150 mt (500 pies) de altura.

Tiempo (segundos) al final de esta fase. Duración de esta fase.
Gráfica de la altura de la trayectoria durante esta fase.
Velocidad horizontal y vertical en km/h al final de esta fase.
Porcentaje de combustible restante en ese momento (de los 8200 kg iniciales).

Si todo ha ido bien, al terminar la 3ª fase debéis tener una velocidad muy pequeña, por debajo de 10 km/hora, y os debería quedar del orden de un 10% de combustible.

Combustible restante en kg al final de la fase 3 (150 mt).

Durante esta 3ª fase, de media, ¿cuál ha sido el % de empuje usado por el motor?

Sabiendo que en esos instantes finales el motor puede "sostener" la nave usando un 25% de su empuje total, ¿de cuántos minutos dispone el piloto para buscar un buen sitio para aterrizar?

Tras ejecutar cada etapa, guardar los tiempos y alturas de cada una de ellas y al terminar superponer en una sola gráfica la altura de la nave en función del tiempo de descenso durante las 3 fases del vuelo. Pintad las tres fase con colores distintos (negro 'k', azul 'b' y rojo 'r'). Adjuntad la gráfica resultante.

**Además del fichero doc con las respuestas adjuntad (en un .rar o similar) con la entrega vuestros programas x_apolo.m y x_dencenso.m.**

**Ambos programas deben ser autosuficientes (deben contener todas las funciones auxiliares con las ecuaciones diferenciales, funciones de eventos, funciones auxiliares, etc.). Para comprobar que son autosuficientes comprobad que se ejecutan correctamente después de hacer un clear, desde un directorio sin ninguna otra función.**