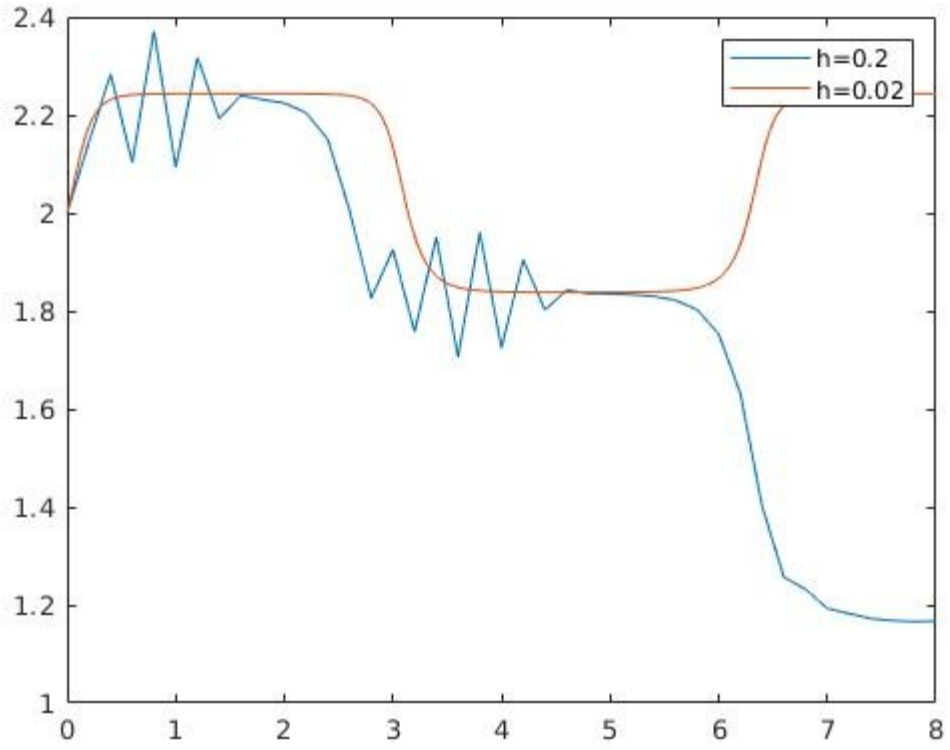
García Martínez, Máximo

Ejercicio 1: Multipaso-paso explícito (orden 2) de la familia Adam-Bashford

Adjuntar código de vuestra función met_AB.m una vez completada.

```
function [T S] = met_AB(fun, Tspan, y0, h)
    t0 = Tspan(1);
    tfin = Tspan(2);

    T = t0:h:tfin;
    N = length(T);


    % Como es de orden 2, necesitamos al menos tener dos elementos en S y F
    S = NaN * zeros(1, N);
    S(1) = y0;
    K1=fun(T(1),S(1)); K2=fun(T(2),S(1)+h*K1); K=(K1+K2)/2; S(2)=S(1)+K*h;


    % Vector con el historico de las evaluaciones de F
    F = NaN * zeros(1, N);
    F(1) = K1;
    F(2) = fun(T(2), S(2));


    for k=2:N-1
        S(k + 1) = S(k) + (h/2)*(3*F(k) - F(k - 1));
        F(k + 1) = fun(T(k+1), S(k+1));
    end
return
```

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?



Como se puede observar para la función que usa h=0.2, tiene cierto retraso respecto a la h=0.02. Esto es debido a qué la primera h hace solo una evalución y la otra función hace 10 evaluaciones en el mismo espacio. Por eso, h=0.02 es más precisa.

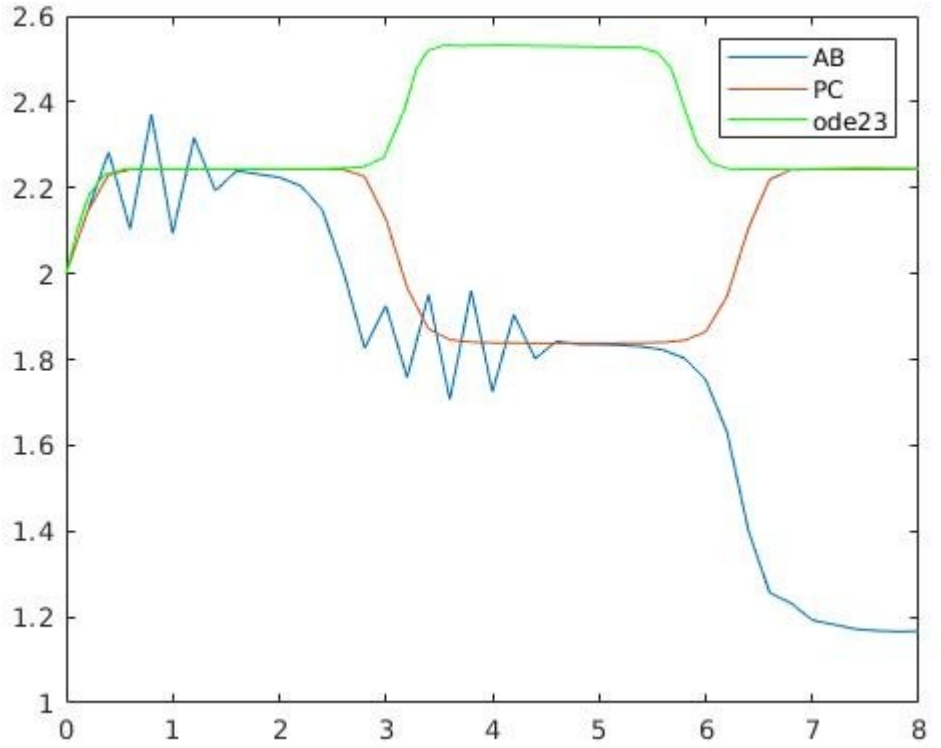
Ejercicio 2: Predictor-Corrector

Adjuntar vuestro código de met_PC.

```
function [T S] = met_PC(fun, Tspan, y0, h)
   t0 = Tspan(1);
   tfin = Tspan(2);

   T = t0:h:tfin;
   N = length(T);


   % Como es de orden 2, necesitamos al menos tener dos elementos en S y F
   S = NaN * zeros(1, N);
   S(1) = y0;
   K1=fun(T(1),S(1)); K2=fun(T(2),S(1)+h*K1); K=(K1+K2)/2; S(2)=S(1)+K*h;


   % Vector con el historico de las evaluaciones de F
   F = NaN * zeros(1, N);
   F(1) = K1;
   F(2) = fun(T(2), S(2));


   for k=2:N-1
      % solucion del predictor
      P = S(k) + (h/2)*(3*F(k) - F(k - 1));
      delta = 99;
      while delta>10^-5
         Pnew = S(k) + (h/2)*(fun(T(k+1), P) + F(k));
         F(k + 1) = fun(T(k+1), S(k+1));
         delta = abs(Pnew - P);
         P = Pnew;
      end
      S(k+1) = P;
      F(k+1) = fun(T(k+1), S(k+1));
   end
return
```

Adjuntar código y gráfica resultante.

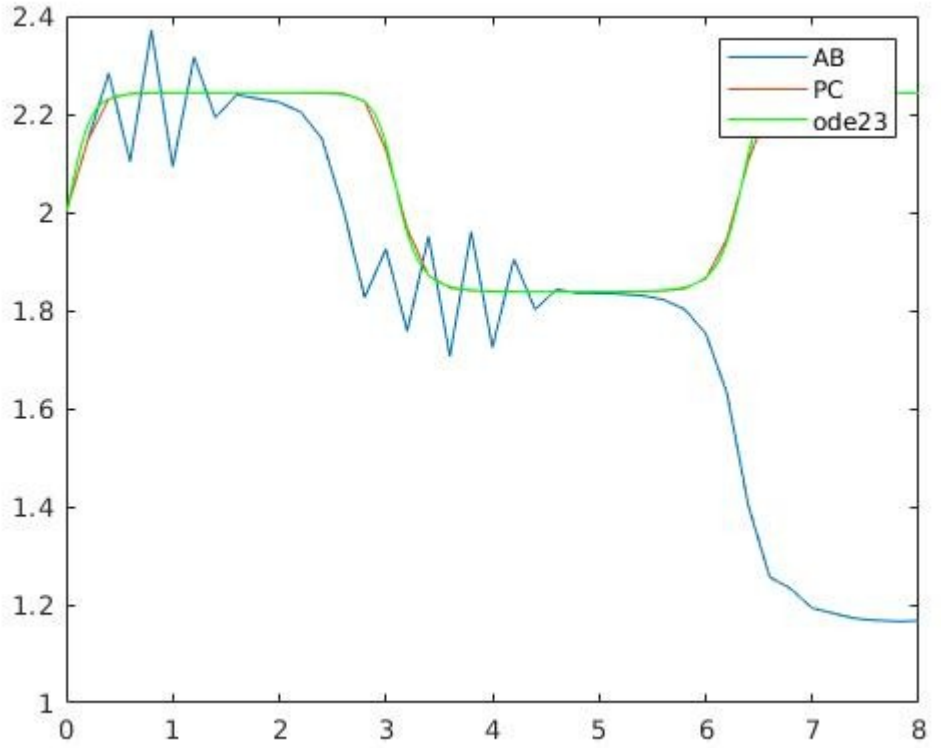
```
y0 = 2;
intervalo = [0 8];
h= 0.2;
[T1 S1] = met_AB(@fun, intervalo, y0, h);
[T2 S2] = met_PC(@fun, intervalo, y0, h);
[T3 S3]= ode23(@fun,intervalo,y0);

plot(T1,S1,T2,S2,T3,S3,'g');
legend({'AB','PC','ode23'});
```



¿Cuántos puntos ha usado el método adaptativo (ode23) para recorrer el intervalo dado?
Ode23 usa 28 puntos en el eje X

¿Cuántos puntos necesita ahora el método adaptativo ode23 para la nueva precisión? ¿Cuántas evaluaciones de la función ed(t,y) ha necesitado?
Ode23 usa 231 puntos en el eje X y 733 evaluaciones en total

Repetir el gráfico anterior con las tres soluciones y adjuntadlo.



¿Cuál de las 3 soluciones era la mejor en la comparación inicial?
La mejor solución porque es la que más evaluaciones y puntos en el eje horizontal tiene.

Opcional (dejad para el final):

Adjuntar la figura resultante. ¿Cuál es el máximo número de iteraciones que tiene que hacer el corrector en un paso?
El máximo número de iterraciones que tiene que hacer el corrector en un paso es de 61

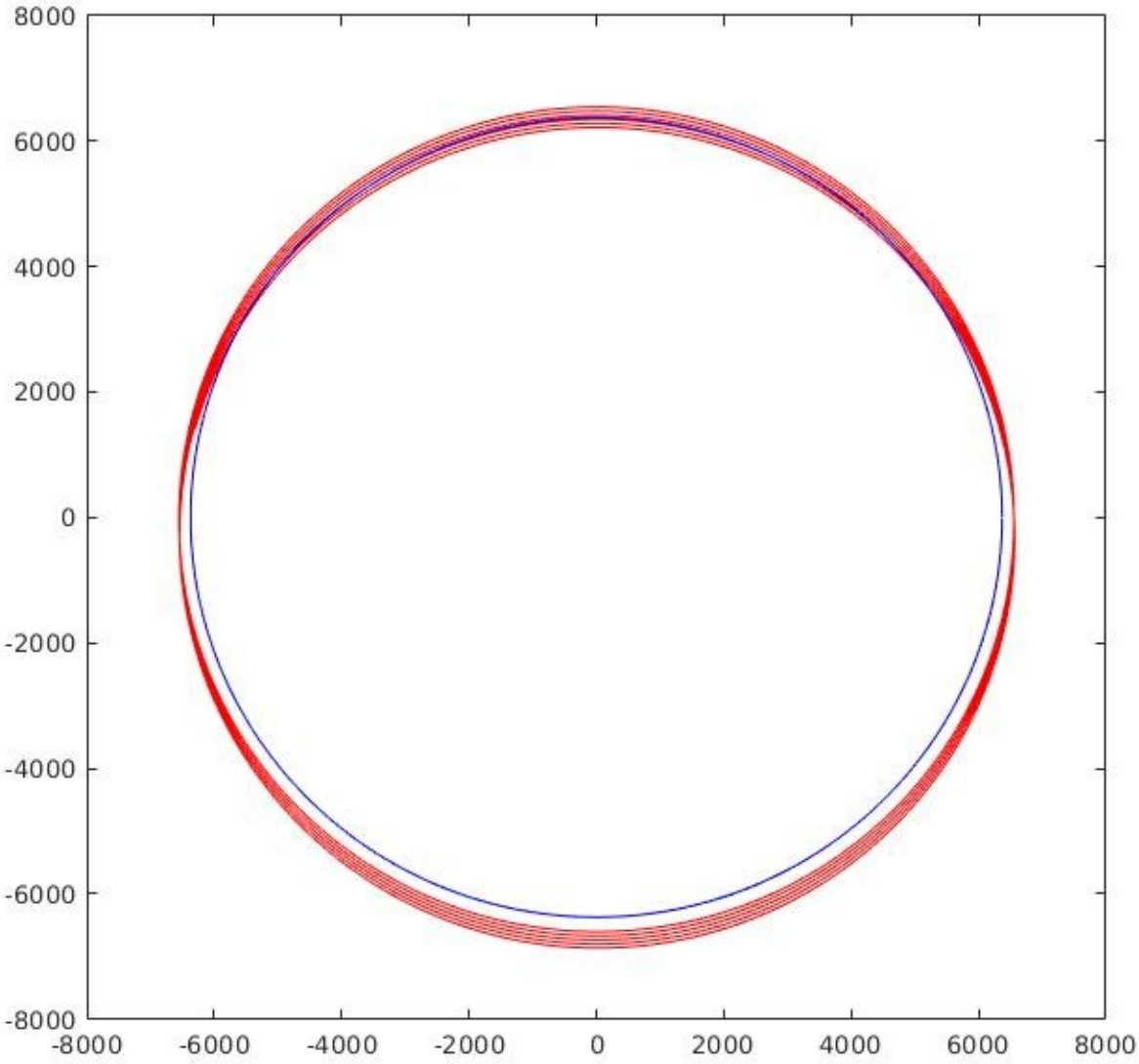
En el método Predictor-Corrector ¿cuántas evaluaciones de ed(t,y) se hacen?
¿Cuántas de ellas corresponden a la parte del predictor? ¿Y a la parte del corrector?
Sumando todos los números de iteraciones,
Comparar con el numero de evaluaciones de la función usadas por el método ode23().
En el método Predictor-Corrector se realizan 488 evaluaciones de las cuales 41 pertenecen a evaluaciones y 447 se usan para corregir.
En el método ode23() se realizan 733 evaluaciones.

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

Adjuntar código de apolo().

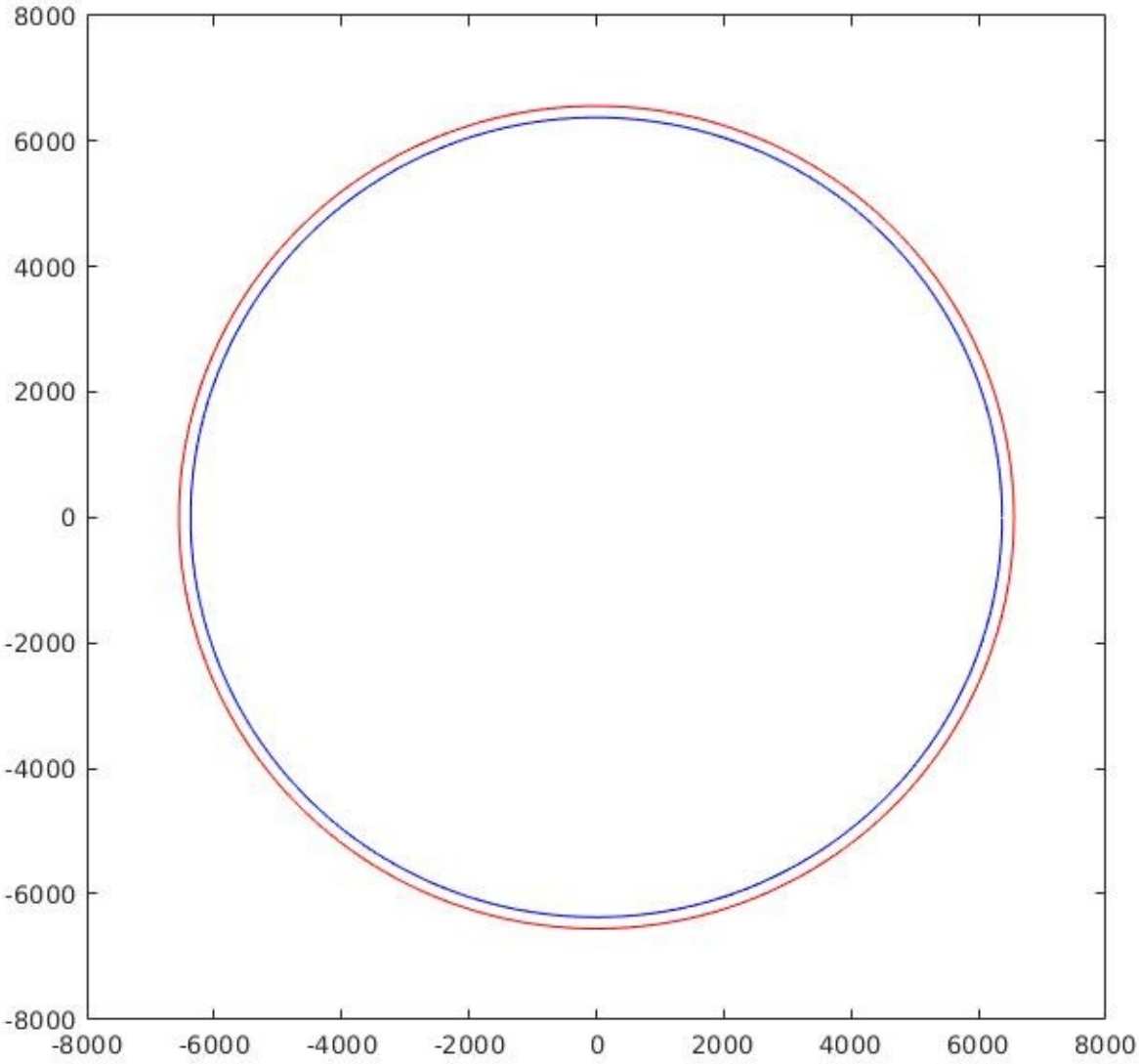
```
function sp = apolo(t,s)
   % s = vector estado con posiciï¿½n s(1:2) y velocidad s(7:8) de la nave
   %       (12x1)        posiciï¿½n s(3:4) y velocidad s(9:10) de la LUNA
   %                   posiciï¿½n s(5:6) y velocidad s(11:12) de la TIERRA

   global motor

   GM1 = 3.99e+005; GM2 = 4.90e+003;

   % Extraer del vector de entrada s la posiciï¿½n y velocidad de la nave
   rn=s(1:2); rL=s(3:4); rT=s(5:6);
   vn=s(7:8); vL=s(9:10); vT=s(11:12);

   % Vector posicion tierra y luna respecto a nave.
   r1 = rn - rT; r2 = rn- rL;
   % distancia entre luna y tierra
   rTL = rL - rT;

   % Establecemos velocidades
   sp(1:6) = s(7:12);

   % Calculo aceleraciones nave
   sp(7:8) = (-(GM1 * r1)/norm(r1)^3) - (GM2 * r2 / norm(r2)^3);

   % Calculo aceleracion Luna
   sp(9:10) = (-(GM1 * rTL)/norm(rTL)^3);

   % Calculo aceleracion Tierra
   sp(11:12) = (-(GM2 * -rLT)/norm(-rLT)^3);

   % Devuelve en sp las velocidades y aceleraciones de las variables de estado
end
```

Adjuntad el gráfico. ¿Observáis algún problema?



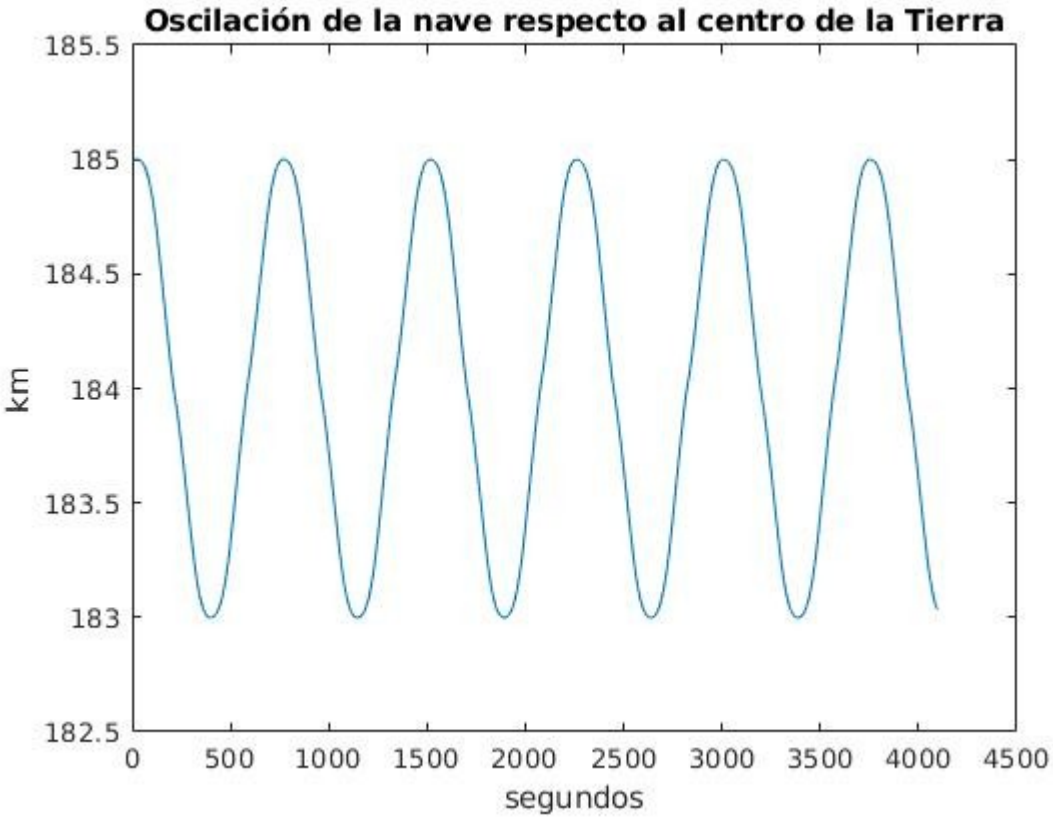
No es una órbita estable. Esto es debido a qué la Tierra está quieta cuando en realidad no es así, se debería estar moviendo. La nave se mueve alrededor de la Tierra.

Adjuntad nueva gráfica. ¿Es ahora una órbita estable?



Ahora es una órbita estable.

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



La nave oscila entre 183km y 185km sobre el nivel del mar.

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

Un período de la gráfica anterior es igual a una vuelta a la Tierra, es decir, la nave tarda unos 880 segundos (14,67 minutos) en dar una vuelta.