# Prueba de Evaluación Continua 1

Adolfo Hilario Garcia

30 de octubre de 2020

*El firmante de este trabajo reconoce que todo él es original, de su única autoía, escritura y redacción, y que allí donde han sido empleadas ideas o datos de otros autores, su trabajo ha sido reconocido y ubicado, con suficiente detalle, como para que el lector pueda consultar lo afirmado sobre él.*

## 1 Introducción y enunciado

Una partícula de masa $m$ se mueve bajo la acción de una fuerza central de la forma $\vec{F}(\vec{r}) = F(r)\vec{u}_r$ siendo $\vec{r}$ el vector posición respecto al centro y $\vec{u}_r$ el vector unitario en la dirección radial.

El hecho de que la partícula se encuentre sometida a este tipo de fuerza le confiere dos características fundamentales: $(i)$ la energía se conserva, esto es, la fuerza es conservativa y $(ii)$ el torque respecto al centro es nulo, por lo que el momento angular $L$ es constante. Esta segunda propiedad hace que el movimiento de la partícula quede confinado a un plano perpendicular al momento angular.

Usando coordenadas polares y reordenando en términos del módulo del momento angular $L = mr^2\theta$, tenemos que

$$E = \frac{1}{2}m\dot{r}^2 + V_{ef}(r)\,, \qquad V_{ef}(r) = \frac{L^2}{2mr^2} + V(r) \tag{1}$$

que son las componentes cinética y potencial del movimiento y donde $V(r) = -\alpha/r$. Además, fijaremos los valores constantes $m = 1$, $\alpha = 5$, $L = 1$ en unidades del SI. Esta memoria consistirá en

(a) calcular el valor de $r$ que hace que el potencial efectivo sea mínimo ($r_{min}$), mediante el método de Newton partiendo de varias semillas. Además, se analizará el criterio de convergencia de dicho método.

(b) analizar el movimiento de la partícula en el caso en que ésta tiene una energía $E = V_{ef}(r_{min})$.

(c) utilizar el método númerico de bisección para encontrar la solución a la ecuación $E = V_{ef}(r)$ para $E = -1$ con un error absoluto menor a $10^{-5}$ y discutir los resultados comparándolos con el apartado $(a)$.

(d) utilizar el método de punto fijo y la aceleración de Aitken para calcular la solución de $E = V_{ef}(r)$ para $E = +1$.

(e) discutir el significado físico de las soluciones del apartado $(d)$.

(f) estimar el orden de convergencia de cada método numérico implementado y compararlos entre sí.

# 2 Metodología

Para realizar esta PEC se han escrito programas que agilicen la tarea a la hora de realizar cálculos y obtener resultados. Se ha utilizado para escribirlos el lenguaje de programación C en el editor de texto Geany, que fue introducido en Física Computacional I. También han sido utilizados programas con funciones de cálculo simbólico como Maple o Matlab para comprobar las soluciones obtenidas en cada ejercicio.

En el Subsección 6.2 se encuentran listados de código que ilustran la metodología de trabajo con la que será enfocada tanto la PEC como la asignatura en general. La metodología sigue las pautas generales que se enlistan a continuación:

- Los algoritmos que efectuan los métodos numéricos están basados en el pseudocódigo de los algoritmos de Burden y Faires (2011). Éstos serán llamados desde `main` con las condiciones iniciales requeridas por las funciones. También habrá funciones (REF) que simplemente consistan en retornarse a ellas mismas (veáse el listado 2). Este último caso es especialmente útil para definir aplicaciones matemáticas como punteros que se pueden utilizar en otras funciones más sofisticadas.
- Las funciones que realizan métodos iterativos crean una matriz en que se escriben los resultados obtenidos para cada iteración. Esta matriz es luego utilizada para, a partir de los datos que contiene, escribir un archivo de LaTeX y un archivo de texto plano en .CSV. Las gráficas de la Subsección 6.1 han sido creadas en Microsoft Excel a partir de un archivo de datos .csv generado también desde C. Las figuras 1, 2 y 3 han sido creadas desde C utilizando GNUplot.
- A lo largo del código aparecen comentarios e indicaciones que facilitan la lectura. No obstante, en el Subsección 6.2 se han eliminado la mayoría de ellos por cuestiones de espacio. Éstos serán sustituidos por comentarios fuera del listado acerca del programa en cuestión.

# 3 Resultados

A continuación se exponen los resultados junto con los razonamientos e interpretaciones pertinentes para cada apartado.

## 3.1 Apartados (a) y (b)

El valor de $r$ que hace que el potencial efectivo sea mínimo debe cumplir que $dV_{ef}/dr = 0$ en $r_{min}$. Se trata esencialmente de un problema de optimización. Es por eso que el método de

Newton se implementará sobre la derivada de la expresión para el potencial mostrada en la Sección 1. Al sustituir los valores fijados por el enunciado, tenemos que

$$f(r) = \frac{dV_{ef}}{dr}(r) = -\frac{1}{r^3} + \frac{5}{r^2} = 0 \tag{2}$$

Además, para el algoritmo método de Newton debemos obtener la derivada de esta expresión, esto es:

$$f'(r) = \frac{d^2V_{ef}}{dr^2}(r) = \frac{3}{r^4} - \frac{10}{r^3} \quad \Longrightarrow \quad r_{n+1} = r_n - \frac{f(r_n)}{f'(r_n)}$$

donde $r_n$ empezará siendo una semilla $r_0$ que elegiremos y que tiene que cumplir el teorema 2.6 de la página 70 de Burden y Faires (2011). Utilizaremos cuatro semillas, dos que estén por encima de la raíz a encontrar y dos que estén por debajo: $r_{01} = 0.05$, $r_{02} = 0.1$, $r_{03} = 0.22$, $r_{04} = 0.26$. A continuación se expone una tabla con los resultados para $r_{01} = 0.05$ y para $r_{03} = 0.22$. El resto de tablas se pueden encontrar en la Subsección 6.1. La primera columna nos indica el paso de iteración en que nos encontramos, la segunda, el valor de $r_{min}$ calculado en cada $i$-iteración, y la tercera columna muestra el error relativo de cada cálculo.

***Tabla 1:*** *Semilla* $r_{01} = 0.05$

| $n$ | $r_{min-n}$ | $\varepsilon_r$ |
|---|---|---|
| 1 | 0.065000 | 1.000000 |
| 2 | 0.083670 | 0.223140 |
| 3 | 0.106167 | 0.211898 |
| 4 | 0.131864 | 0.194877 |
| 5 | 0.158583 | 0.168483 |
| 6 | 0.181805 | 0.127732 |
| 7 | 0.195799 | 0.071470 |
| 8 | 0.199746 | 0.019762 |
| 9 | 0.199999 | 0.001266 |
| 10 | 0.200000 | 0.000005 |
| 11 | 0.200000 | 0.000000 |

***Tabla 2:*** *Semilla* $r_{03} = 0.22$

| $n$ | $r_{min-n}$ | $\varepsilon_r$ |
|---|---|---|
| 1 | 0.192500 | 1.000000 |
| 2 | 0.199215 | 0.033708 |
| 3 | 0.199991 | 0.003879 |
| 4 | 0.200000 | 0.000046 |
| 5 | 0.200000 | 0.000000 |

Con esto, podemos concluir que la solución es $r_{min} = 0.2$ unidades de longitud. Nótese cómo $|0{,}2 - 0{,}05| > |0{,}2 - 0{,}22|$, lo que provoca una convergencia más rápida hacia la solución. Cabe esperar el mismo efecto si tomamos $r_{02}$ y $r_{04}$.

Analicemos ahora el teorema de convergencia, Teorema 2.6., Burden y Faires (2011, pág. 70). El método de Newton genera una sucesión $\{r_n\}_{n=1}^{\infty}$ que converge a $r$ para cualquier aproximación inicial $r_0 \,\epsilon\, [r-\delta, r+\delta]$, $\delta > 0$ si $(i)$ $f(r) = 0$ y $(ii)$ $f'(r) \neq 0$ para $r \,\epsilon\, [a, b]$ y para $f \,\epsilon\, \mathcal{C}^2[a, b]$. La primera de las condiciones se cumple en cualquier subintervalo de $(0, \infty)$ que contenga a $r_{min}$. La segunda de las condiciones también se cumple pues $f'(r_{min}) = -1.0625$. En consecuencia, $\exists \delta > 0$ tal que la sucesión de raíces converge dada una semilla $r_0$ que se encuentre en el intervalo $[r - \delta, r + \delta]$. En nuestro caso, para valores mayores que 0.26 la sucesión diverge a infinito, lo cual nos da un rango seguro de semillas a aplicar: $\delta = 0{,}06 \rightarrow r_0 \,\epsilon\, [r - 0{,}06, r + 0{,}06] = [0{,}14, 0{,}26]$, aunque también podemos salirnos de ese rango de seguridad y obtener una raíz, como es el caso de $r_{01}$.

En el caso de $E = V_{ef}(r_{min}) = -2.375$ unidades de energía, el potencial efectivo mínimo se

corresponde con la energía mínima. En este caso, la trayectoria de la partícula vendrá descrita por una órbita circular de radio $r_{min}$. A esta distancia del centro, la fuerza centrífuga producida por el momento angular se cancela con la fuerza atractiva producida por el potencial $V(r)$. La Figura 1 muestra la evolución del potencial efectivo respecto a la distancia $r$, con su correspondiente mínimo en $r = 0.2$. A la derecha, la Figura 2 muestra el potencial sin su componente centrífuga (sin la componenente dependiente de $L$) y su evolución en función de $r$.

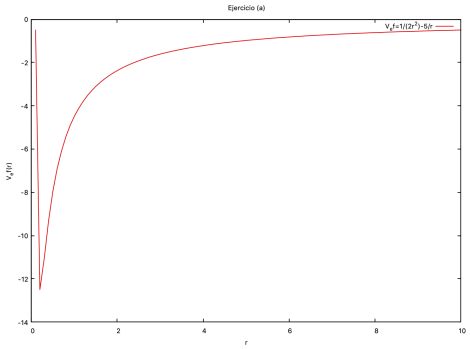


***Figura 1:*** *Representación gráfica del potencial efectivo en función de la distancia al centro r*

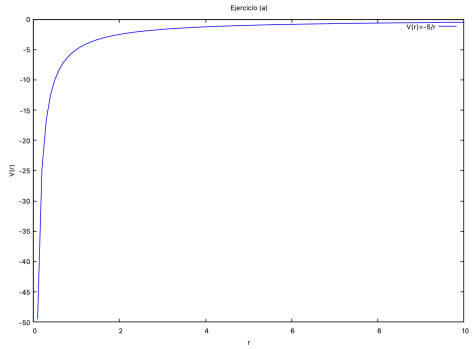


***Figura 2:*** *Representación del potencial atractivo $V(r)$ en función de la distancia al centro r*

## 3.2 Apartado (c)

Fijando el valor $E = -1$, podemos calcular los valores de $r$ para los que la partícula tiene una velocidad nula, llamados puntos de retroceso, resolviendo la ecuación

$$E = \frac{L^2}{2mr^2} - \frac{\alpha}{r} \quad \Longrightarrow \quad g(r) = \frac{1}{2r^2} - \frac{5}{r} + 1 = 0 \tag{3}$$

Como se puede apreciar en la Figura 3, la (3) tiene dos ceros, uno en el intervalo $(0, 1]$, y otro en el intervalo $[4, 5]$, donde además se puede aplicar el método de bisección pues las funciones son contínuas y tienen distinto signo en los extremos de los respectivos intervalos.

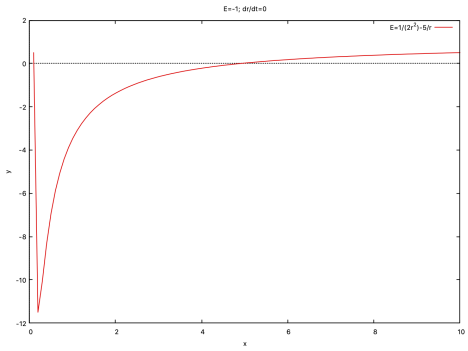


***Figura 3:*** *Representación gráfica de la ecuación (3)*

El número de iteraciones a realizar viene dado por una expresión fácil de aislar para obtener la información que necesitamos:

$$|r_n - r_{n-1}| \leq \frac{b-a}{2^n} \quad \Longrightarrow \quad n \geq \log_2 10^5 \cong 16{,}61 \tag{4}$$

donde hemos utilizado la aproximación de que que $b-a \approx 1$ en ambos casos, y que el enunciado pide un error absoluto menor que $10^{-5}$. El resultado anterior nos dice que deberemos realizar almenos 17 iteraciones en nuestro programa para asegurar que las raíces se han obtenido de acuerdo con la tolerancia exigida por el enunciado.

A continuación la Tabla 3 y la Tabla 4 muestran el proceso seguido mediante el método de bisección para obtener las raíces deseadas en el intervalo [0,05, 1] y [4, 5], en las cuales se muestra los sucesivos recortes en el intervalo de evaluación de la función, los sucesivos valores $r_{01}$ y $r_{45}$[1]. calculados en cada iteración del algoritmo, el valor de la función $g(r)$ en ese punto (que tiende a cero) y el error relativo en cada $n$-iteración.

El hecho de que la partícula tenga dos puntos de retroceso significa que al solucionar la ecuación

$$\dot{r} = \sqrt{\frac{2}{m}(E - V_{ef}(r))} \tag{5}$$

para $\dot{r} = 0$ y $E = -1$ se obtienen dos soluciones. Esta expresión no es más que una reformulación de la ecuación (1). Ahora bien, ¿qué significado físico tienen dos soluciones?. En este caso, la partícula posee un punto de mínima distancia con el centro y un punto de máximo alejamiento, que corresponden a $r_{01}$ y $r_{45}$, respectivamente. La partícula puede o no describir una trayectoria cerrada, pero una vez se encuentre bajo la influencia de la fuerza $F$, no se alejará del centro más que $r_{45}$ unidades de longitud y no se acercará más que $r_{01}$.

***Tabla 3:*** *Intervalo* [0,05, 1]

| $n$ | $a$ | $b$ | $r_{01}$ | $g(r_{01})$ | $\varepsilon_r$ |
|---|---|---|---|---|---|
| 1 | 0.050000 | 0.525000 | 0.525000 | -6.709751 | 1.000000 |
| 2 | 0.050000 | 0.287500 | 0.287500 | -10.342155 | 0.826087 |
| 3 | 0.050000 | 0.168750 | 0.168750 | -11.071331 | 0.703704 |
| 4 | 0.050000 | 0.109375 | 0.109375 | -2.918368 | 0.542857 |
| 5 | 0.079687 | 0.109375 | 0.079687 | 16.993851 | 0.372549 |
| 6 | 0.094531 | 0.109375 | 0.094531 | 4.059900 | 0.157025 |
| 7 | 0.101953 | 0.109375 | 0.101953 | 0.060495 | 0.072797 |
| 8 | 0.101953 | 0.105664 | 0.105664 | -1.536548 | 0.035120 |
| 9 | 0.101953 | 0.103809 | 0.103809 | -0.767131 | 0.017874 |
| 10 | 0.101953 | 0.102881 | 0.102881 | -0.360893 | 0.009018 |
| 11 | 0.101953 | 0.102417 | 0.102417 | -0.152128 | 0.004529 |
| 12 | 0.101953 | 0.102185 | 0.102185 | -0.046305 | 0.002270 |
| 13 | 0.102069 | 0.102185 | 0.102069 | 0.006972 | 0.001136 |
| 14 | 0.102069 | 0.102127 | 0.102127 | -0.019698 | 0.000568 |
| 15 | 0.102069 | 0.102098 | 0.102098 | -0.006371 | 0.000284 |
| 16 | 0.102084 | 0.102098 | 0.102084 | 0.000298 | 0.000142 |
| 17 | 0.102084 | 0.102091 | 0.102091 | -0.003037 | 0.000071 |

[1]Los subíndices corresponden a los intervalos de solución, si bien $r_{01}$ será evaluado en $(0{,}05, 1)$ para no dividir por cero

***Tabla 4:*** *Intervalo* $[4, 5]$

| $n$ | $a$ | $b$ | $r_{45}$ | $g(r_{45})$ | $\varepsilon_r$ |
|---|---|---|---|---|---|
| 1 | 4.500000 | 5.000000 | 4.500000 | -0.086420 | 1.000000 |
| 2 | 4.750000 | 5.000000 | 4.750000 | -0.030471 | 0.052632 |
| 3 | 4.875000 | 5.000000 | 4.875000 | -0.004602 | 0.025641 |
| 4 | 4.875000 | 4.937500 | 4.937500 | 0.007851 | 0.012658 |
| 5 | 4.875000 | 4.906250 | 4.906250 | 0.001663 | 0.006369 |
| 6 | 4.890625 | 4.906250 | 4.890625 | -0.001460 | 0.003195 |
| 7 | 4.890625 | 4.898438 | 4.898438 | 0.000104 | 0.001595 |
| 8 | 4.894531 | 4.898438 | 4.894531 | -0.000677 | 0.000798 |
| 9 | 4.896484 | 4.898438 | 4.896484 | -0.000286 | 0.000399 |
| 10 | 4.897461 | 4.898438 | 4.897461 | -0.000091 | 0.000199 |
| 11 | 4.897461 | 4.897949 | 4.897949 | 0.000007 | 0.000100 |
| 12 | 4.897705 | 4.897949 | 4.897705 | -0.000042 | 0.000050 |
| 13 | 4.897827 | 4.897949 | 4.897827 | -0.000018 | 0.000025 |
| 14 | 4.897888 | 4.897949 | 4.897888 | -0.000006 | 0.000012 |
| 15 | 4.897888 | 4.897919 | 4.897919 | 0.000001 | 0.000006 |
| 16 | 4.897903 | 4.897919 | 4.897903 | -0.000002 | 0.000003 |
| 17 | 4.897911 | 4.897919 | 4.897911 | -0.000001 | 0.000002 |

## 3.3 Apartados (d) y (e)

Finalmente, fijaremos el valor de de la energía total a $E = +1$ unidades de energía. Calcularemos los valores de $r$ que son solución de la ecuación

$$\frac{1}{2r^2} - \frac{5}{r} - 1 = 0 \tag{6}$$

mediante el método de punto fijo, y más tarde aplicando la aceleración de Aitken (método de Steffensen). Antes de poner a funcionar el algoritmo, debemos convertir (5) a la forma $r = h(r)$. Los despejes obtenidos son[2]:

$$r = \left(\frac{r}{10} + \frac{1}{2}\right)^{1/2} \tag{7}$$

$$r = \frac{1}{2r} - 5 \tag{8}$$

$$r = \left(\frac{1}{2} - 5r\right)^{1/2} \tag{9}$$

La primera de las ecuaciones es una raíz cuadrada estrictamente positiva dados los elementos que tiene en su interior y sabiendo que $r > 0$. No es el caso de la tercera ecuación, que para $r > \frac{1}{10}$ la raíz será negativa y el programa nos devolverá `nan` ("not a number"). Por tanto, no utilizaremos la tercera ecuación para hallar la solución. La segunda expresión no nos traerá problemas con los números reales, pero sí a la hora de encontrar una raíz mayor que cero.

El método del punto fijo consiste en obtener, a partir de una determinada semilla, la solución a una ecuación haciendo $r = h(r_0)$ y renovando $r_0 = r$. El método de Steffensen incorpora la aceleración de Aitken, que no es mas que implementar en el programa la ecuación:

[2] Las raíces cuadradas siempre son positivas pues no tiene sentido físico obtener $r < 0$

$$r = r_0 - \frac{(r_1 - r_0)^2}{p_2 - 2p_1 + p_0} \qquad r_1 = h(r_0) \quad r_2 = h(r_1) \tag{10}$$

e ir renovando $r$ hasta realizar las iteraciones necesarias o rebajar el error absoluto deseado.

Las tablas 5, 6,7 y 8 presentan los resultados obtenidos mediante ambos métodos utilizando la semilla $r_0 = 0{,}05$.

***Tabla 5:*** *Método del punto fijo para la primera ecuación*

| $n$ | $r$ | $\varepsilon_r$ |
|---|---|---|
| 1 | 0.070711 | 1.000000 |
| 2 | 0.084090 | 0.159104 |
| 3 | 0.091700 | 0.082996 |
| 4 | 0.095760 | 0.042397 |
| 5 | 0.097857 | 0.021428 |
| 6 | 0.098923 | 0.010772 |
| 7 | 0.099460 | 0.005401 |
| 8 | 0.099730 | 0.002704 |
| 9 | 0.099865 | 0.001353 |
| 10 | 0.099932 | 0.000677 |
| 11 | 0.099966 | 0.000338 |
| 12 | 0.099983 | 0.000169 |
| 13 | 0.099992 | 0.000085 |

***Tabla 6:*** *Método de Steffensen para la primera ecuación*

| $n$ | $r_1$ | $r_2$ | $r$ | $\varepsilon_r$ |
|---|---|---|---|---|
| 1 | 0.070711 | 0.084090 | 0.108504 | 1.000000 |
| 2 | 0.104165 | 0.102061 | 0.100081 | 0.084161 |
| 3 | 0.100040 | 0.100020 | 0.100000 | 0.000808 |
| 4 | 0.100000 | 0.100000 | 0.100000 | 0.000000 |

***Tabla 7:*** *Método del punto fijo para la segunda ecuación*

| $n$ | $r$ | $\varepsilon_r$ |
|---|---|---|
| 1 | 5.000000 | 1.000000 |
| 2 | -4.900000 | 2.020408 |
| 3 | -5.102041 | 0.039600 |
| 4 | -5.098000 | 0.000793 |
| 5 | -5.098078 | 0.000015 |
| 6 | -5.098076 | 0.000000 |

***Tabla 8:*** *Método de Steffensen para la segunda ecuación*

| $n$ | $r_1$ | $r_2$ | $r$ | $\varepsilon_r$ |
|---|---|---|---|---|
| 1 | 5.000000 | -4.900000 | 1.700000 | 1.000000 |
| 2 | -4.705883 | -5.106250 | -5.132941 | 1.331194 |
| 3 | -5.097410 | -5.098089 | -5.098076 | 0.006839 |
| 4 | -5.098076 | -5.098076 | -5.098076 | 0.000000 |

Como se puede apreciar, la incorporación de la aceleración de Aitken acelera la convergencia de forma significativa, especialmente para la primera expresión. Con esto, las soluciones obtenidas son $r_1 = 0{,}100$ unidades de longitud y $r_2 = -5{,}098$ unidades de longitud. Como hemos mencionado anteriormente, si bien la computadora no nos da problemas a la hora de calcular $r_2$, sabemos que $r > 0$, por lo que $r_2$ no tiene ningun significado físico al tratarse de una distancia negativa.

Por otro lado, $r_1$ es la solución a la ecuación (6), es decir, de la ecuación $E = V_{ef}(r) = +1$. En este caso, una partícula que viene del infinito se acercará no más cerca que el radio mínimo $r_1$, en el cual se cumple que $E = V_{ef}(r)$. En este punto la partícula nota el efecto de la fuerza centrífuga provocada por el momento angular y es despedida de nuevo hacia el infinito.

En definitiva, hemos podido comprobar que si la energía total es igual al potencial mínimo ésta también se minimiza, y la partícula describe una órbita circular alrededor del centro. Por otro lado, si la energía es menor que cero, existe un punto de máximo acercamiento de la partícula al centro y otro de máximo alejamiento dónde $\dot{r} = 0$. Por último, sabemos que

si la energía total es mayor que cero una partícula que viene desde lejos pasa por un radio mínimo que es solución de la ecuación y es expulsada de nuevo hacia el infinito a causa de la fuerza centrífuga que causa el momento angular.

## 3.4 Apartado (f)

En esta sección analizaremos el error relativo en los distintos métodos iterativos empleados a lo largo de la memoria con el objetivo de estimar el orden de convergencia de cada uno de ellos. Para ello se ha realizado gráficas desde Microsoft Excel a partir de los datos generados por cada programa. Las gráficas 4 y 5 muestran el valor absoluto del logaritmo en base 10 del error relativo a lo largo de las $n$-iteraciones para el método de Newton para las dos semillas presentadas.

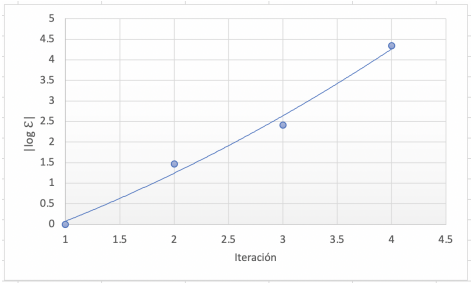


***Figura 4:*** *Representación logarítmica del error en cada iteración para el método de Newton*

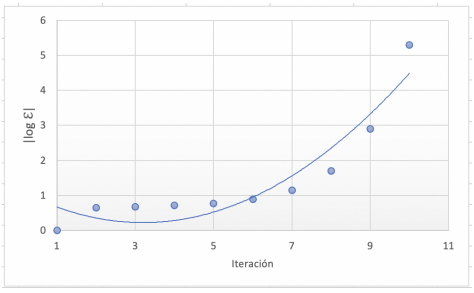


***Figura 5:*** *Representación logarítmica del error en cada iteración para el método de Newton*

Las gráficas 6 y 7 muestran el valor absoluto del logaritmo en base 10 del error relativo a lo largo de las $n$-iteraciones para el método de bisección en cada intervalo solución.

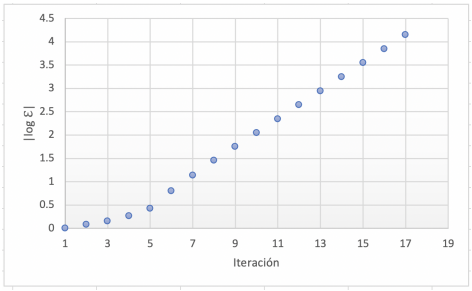


***Figura 6:*** *Representación logarítmica del error en cada iteración para el método de bisección*

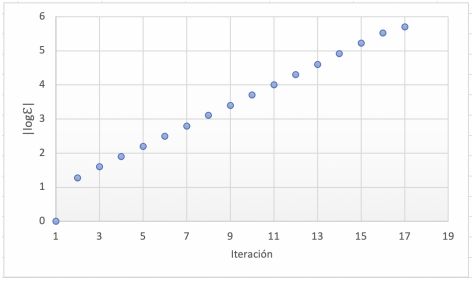


***Figura 7:*** *Representación logarítmica del error en cada iteración para el método de bisección*

Las gráficas 8 y 9 muestran el valor absoluto del logaritmo en base 10 del error relativo a lo largo de las $n$-iteraciones para el método de punto fijo y de Steffensen, respectivamente, para la solución físicamente válida.

Las representaciones gráficas expuestas nos dan información valiosa acerca del orden de convergencia. Podemos afirmar que:

1. El método de Newton converge cuadráticamente en los casos que se presentan en esta PEC, esto es, tiene orden de convergencia 2. Además, $\varepsilon_{n+1} = x_{n+1} - r$.

2. El método de bisección converge linealmente, esto es, tiene orden de convergencia 1. De hecho $|\varepsilon_{n+1}| = 0{,}5|\varepsilon_n|$.

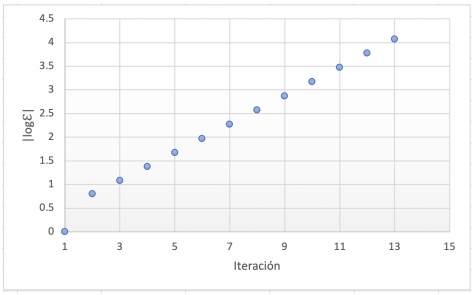


***Figura 8:*** *Representación logarítmica del error en cada iteración para el método de punto fijo*

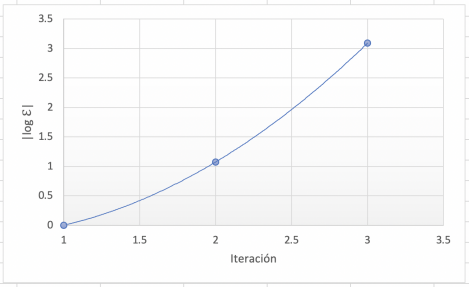


***Figura 9:*** *Representación logarítmica del error en cada iteración para el método de Steffensen.*

3. La iteración de punto fijo converge linealmente a la solución, por lo tanto el orden de convergencia es 1.
4. La incorporación de la aceleraciópn de Aitken al método de punto fijo provoca una convergencia cuadrática hacia la solución, esto es, pasa a orden de convergencia 2.

# 4 Ampliación: Campo gravitatorio

Una analogía válida para este caso teórico de mecánica podría ser considerar que la fuerza central $\vec{F}(\vec{r})$ está producida por un campo vectorial conservativo como, por ejemplo, el gravitatorio. Éste cumple todos los requisitos, pues en él se conserva la energía y el momento angular, y la fuerza depende de la distancia del centro a la que se encuentre una determinada partícula.

Consideremos la ecuación (1) aplicada en el sistema solar, suponiendo que tenemos al Sol como centro que genera un campo gravitatorio como el descrito en el párrafo anterior, y con una energía potencial atractiva del tipo $V(r) = -\alpha/r$. En este caso, las distintas situaciones discutidas a lo largo de la memoria para los potenciales mínimos y los puntos de retroceso se pueden aplicar a cuerpos celestes que actúan como partículas. Por ejemplo:

- Una partícula con $E = V_{ef}(r_{min})$ puede ser comparada con el planeta Venus, pues su trayectoria alrededor del Sol, si bien es elíptica, tiene tan baja excentricidad que parece casi circular.
- Una partícula con $E < 0$ tiene dos puntos donde $\dot{r} = 0$. Éste fenomeno se da, por ejemplo, cuando un cometa queda atrapado en un campo gravitatorio.
- Un cometa que venga desde el espacio exterior al sistema solar con energía $E > 0$ describirá una trayectoria parabólica con vértice en un punto de máximo acercamiento antes de volver a salir disparado hacia el espacio interestelar.

Tanto para la interpretación de las soluciones como para las analogías y ejemplos se ha consultado *Potenciales centrales* (2020).

# 5 Conclusiones

En esta PEC 1 se ha estudiado el movimiento de una partícula sometida a una fuerza conservativa central para diferentes valores de su energía total, utilizando diferentes métodos

iterativos, discutiendo su eficiencia y razonando sobre la validez matemática y física de los resultados obtenidos. Además se ha conseguido comparar ésta situación matemática o ïdealçon un fenómeno presente en el universo, como es el movimiento de cuerpos celestes bajo el campo gravitatorio creado por el Sol en nuestro sistema solar.

En definitiva, conocer en profundidad las propiedades del movimiento de partículas sometidas a una fuerza central es vital para entender como actúa la gravedad sobre los distintos planetas del sistema solar, además de sobre los distintos cuerpos celestes que pueden o no llegar de otros sistemas estelares, como es el caso de cometas o asteroides.

# 6 Anexo

## 6.1 Tablas y gráficos

En esta sección se presentan los datos obtenidos para $r_{min}$ mediante el método de Newton para las semillas $r_{02}$ (Tabla 9) y $r_{04}$ (Tabla 10).

***Tabla 9:*** *Semilla* $r_{02} = 0.1$

| $n$ | $r_{min-n}$ | $\varepsilon_r$ |
|---|---|---|
| 1 | 0.125000 | 1.000000 |
| 2 | 0.151786 | 0.176471 |
| 3 | 0.176474 | 0.139896 |
| 4 | 0.193279 | 0.086948 |
| 5 | 0.199365 | 0.030528 |
| 6 | 0.199994 | 0.003145 |
| 7 | 0.200000 | 0.000030 |
| 8 | 0.200000 | 0.000000 |

***Tabla 10:*** *Semilla* $r_{04} = 0.26$

| $n$ | $r_{min-n}$ | $\varepsilon_r$ |
|---|---|---|
| 1 | 0.065000 | 1.000000 |
| 2 | 0.083670 | 0.223140 |
| 3 | 0.106167 | 0.211898 |
| 4 | 0.131864 | 0.194877 |
| 5 | 0.158583 | 0.168483 |
| 6 | 0.181805 | 0.127732 |
| 7 | 0.195799 | 0.071469 |
| 8 | 0.199746 | 0.019762 |
| 9 | 0.199999 | 0.001266 |
| 10 | 0.200000 | 0.000005 |
| 11 | 0.200000 | 0.000000 |

## 6.2 Listados de código

Los algoritmos implementados en el código de los distintos programas están basados en los algoritmos de Burden y Faires (2011). En las sigue se presentan los códigos utilizados.

### Método de Newton

Como se puede ver en el listado 1, la entrada de la función `newton` consta de 8 elementos, el puntero a la función matemática `f` y a su derivada `fprima` (veáse el listado 2), la semilla `p_0`, además de la tolerancia `TOL` y el número de iteraciones `N_0`. Por último, creamos un puntero `*M` que creará un array de `nparam` parámetros. El puntero `*ir` se guardará el número de iteraciones realizadas, para utilizarlo como cota superior a la hora de escribir los archivos en `main`, en caso de que `ir` $<$ `N_0`.

El cuerpo del programa consiste en un bucle `while` en que, en cada iteración, el programa

1. define la raíz `p` mediante la fórmula de Newton, y el error apsoluto como $|p - p_0|$.
2. para el programa si el error absoluto es menor que la tolerancia exigida (aquí es importante `*ir`).
3. define el error relativo como $|p - p_0|/|p|$ en términos del puntero/array `*M`.
4. escribe la iteración en que nos encontramos, el valor de la raíz calculada y el error relativo de ésta en `*M`.
5. renueva la semilla con el valor obtenido en la iteración en cuestión.

***Listado 1:*** *Método de Newton*

```
float newton(float (*f)(float), float (*fprima)(float), float p_0,
                float TOL, int N_0, float *M, int nparam, int *ir){
    int i=0;
    float p;
    float e_abs, e_rel;
    int cond_salida=0;

    while (i<N_0){
        p = p_0 - (*f)(p_0)/(*fprima)(p_0);

        e_abs = fabs(p-p_0); //definimos el error absoluto

        if (e_abs<TOL){
            cond_salida = 1;
        }

        e_rel = fabs(p-*((M+(i-1)*nparam) + 1))/fabs(p);

        *((M+i*nparam) + 0) = i+1;  //escribiendo la matriz
        *((M+i*nparam) + 1) = p;
        *((M+i*nparam) + 2) = e_rel;

        i++;
        p_0 = p; //renovamos p al final de cada iteracion

        if(cond_salida==1){break;}
    }

    *ir = i;
    return p;
}
```

***Listado 2:*** *Funciones matemáticas empleadas en el método de Newton*

```
float funcion(float x)      //función sin derivar
{
    return -1/pow(x, 3) + 5/pow(x,2);
}

float derivada(float x)     //derivada de la funcion
{
    return 3/pow(x, 4) - 10/pow(x,3);
}
```

A continuación se muestra como, desde `main` se llama a la función algoritmo y, además, se escriben los archivos de datos .tex y .csv. Dado que el proceso es ánalogo para los demás

métodos, se mostrará el listado 3 como referencia general de como están estructurada `main` en todos los programas de la PEC.

### Método de Bisección

Como se puede ver en el listado 4, la entrada de la función `biseccion` consta de 8 elementos, el puntero a la función matemática `f`, el intervalo de evaluación `[a,b]`, además de la tolerancia `TOL` y el número de iteraciones `N_0`. Por último, creamos un puntero `*M` que creará un array de `nparam` parámetros. El puntero `*ir` funciona de forma análoga a en `newton`.

El cuerpo del programa consiste en un bucle `while` en que, en cada iteración, el programa

1. define la raíz `p` mediante el teorema del Valor Intermedio, y calcula el valor de la función en ese punto.
2. para el programa si el error absoluto es menor que la tolerancia exigida (aquí es importante `*ir`).
3. renueva el intervalo de evaluación de la función mediante un `if-else` que asegura que $a$ y $b$ tengan signo opuesto. Renueva entonces el valor de la raíz como extremo superior o inferior del intervalo.
4. escribe la iteración en que nos encontramos, los extremos del intervalo de la iteración en cuestión, el valor de la raíz calculada y el error relativo de ésta en `*M`.

### Iteración de punto fijo

El algoritmo que se muestra en el listado 5 es el utilizado para realizar la iteración de punto fijo a una función $p = g(p)$. Cómo se puede apreciar, es relativamente sencillo, pues se trata únicamente de evaluar la función en la semilla proporcionada inicialmente e ir renovando la raíz hasta rebajar la tolerancia requerida o completar el número de iteraciones.

### Método de Steffensen

Éste método es una extensión del método de punto fijo, en que se incorpora la llamada aceleración de Aitken, cuya fórmula aparece en la línea 17 del listado 6. La clave de este algoritmo es asegurarnos de que pare el bucle en caso de que $p_2 = p_1 = p$ pues el denominador se anularía, lo que causa problemas en la respuesta recibida. Para evitar esto, introducimos un `if-else` que controla el polinomio conflictivo antes de convertirlo en denominador. En caso de que éste sea cero, el programa ha llegado a la solución, pues significa que el error relativo es cero.

### Gráficas con GNUplot

El listado 7 muestra un ejemplo de la metodología seguida para generar una gráfica desde C mediante GNUplot. Primero, se define una lista de comandos para ejecutar y configurar la visualización que tendrán los puntos en la gráfica con GNUplot: datos a utilizar, estilo, color leyenda, etc,. A continuación se crea una archivo con la función `popen` que utilizamos para crear un canal que ejecute los comandos de gnuplot y los represente.

# 7 Bibliografía

Burden, Richard L. y J. Douglas Faires (2011). *Análisis numérico*. 9.ª ed. (vid. págs. 2, 3, 10).

*Potenciales centrales* (2020). URL: http://www.ugr.es/~pittau/MECANICA/t2.pdf (vid. pág. 9).

***Listado 3:*** *Ejemplo general de la función* `main`

```
int main()
{
    int N_0 = 100;
    int ncol = 3;
    float p, p_0, TOL = pow(10,-9);
    float M[N_0][ncol];
    int i,j;
    int ir;

    char nombrearchivo_datos[100] = "datos_newton.csv";
    char nombrearchivo_tex[100] = "tabla_newton.tex";
    FILE *archivodatos; //para guardar los datos en un archivo

    char cabeceras[100] = "$i$ & $p_n$ & $\\varepsilon_r$";
    char columnesTabular[100] = "*3{r}";

    // -------------------------------------------------
    // -------------------------------------------------

    //p_0 = 0.26;
    p_0 = 0.05;
    //p_0 = 0.22;
    //p_0 = 0.1;    //distintas semillas

    p = newton(&funcion, &derivada, p_0, TOL, N_0, (double *)M, ncol, &ir);

    // -------------------------------------------------
    // -------------------------------------------------
    // Escribimos el archivo de datos -> Microsoft Excel

    archivodatos = fopen(nombrearchivo_datos, "w");
    for (i=0;i<ir;i++){

        fprintf(archivodatos, "%d; ", (int)M[i][0]);
            for (j=1;j<ncol;j++){
                fprintf(archivodatos,"%f; ",M[i][j]); //&??
            }
        fprintf(archivodatos,"\n");
    }
    fclose(archivodatos);

    // -------------------------------------------------
    // Escribimos la tabla para LaTeX
    archivodatos = fopen(nombrearchivo_tex, "w");

    fprintf(archivodatos,"\\begin{tabular}{%s}\n\t\\toprule\n\t
    %s\\\\\n\t\\midrule\n\t",columnesTabular,cabeceras);

    for (i=0;i<ir;i++)
    {
        fprintf(archivodatos, "%d & ", (int)M[i][0]);
        for (j=1;j<ncol;j++){
            fprintf(archivodatos,"%f",M[i][j]);
            if (j<ncol-1) {fprintf(archivodatos," & ");}
            else {fprintf(archivodatos,"\\\\\n\t");}
        }
    }
        fprintf(archivodatos,"\\bottomrule\n\\end{tabular}");
        fclose(archivodatos);

    // -------------------------------------------------
    // -------------------------------------------------
    // -------------------------------------------------
    // -------------------------------------------------

    return 0;
}
```

***Listado 4:*** *Algoritmo para ejecutar el método de bisección*

```
float biseccion(float (*f)(float), float a, float b, float TOL,
                float N_0, float *M, int nparam, int *ir){
    int i=0;
    float FA=(*f)(a);
    float FP, p;
    float e;
    int cond_salida = 0;

    while (i<N_0){

        p=a+(b-a)/2;
        FP=(*f)(p);

        if (FP==0 || ((b-a)/2)<TOL){cond_salida = 1;}

        if (FA*FP>0){
            a=p;
            FA=FP;
        }else{b=p;}

        e = fabs(p-*((M+(i-1)*nparam) + 3))/fabs(p);

        *((M+i*nparam) + 0) = i+1;
        *((M+i*nparam) + 1) = a;
        *((M+i*nparam) + 2) = b;
        *((M+i*nparam) + 3) = p;
        *((M+i*nparam) + 4) = FP;
        *((M+i*nparam) + 5) = e;

        i++;

        if(cond_salida==1){break;}
    }

    *ir = i;
    return p;
}
```

***Listado 5:*** *Función que realiza la iteración de punto fijo*

```
float puntofijo(float (*f)(float), float p_0, float TOL,
                  float N_0, float *M, int nparam, int *ir)
{
    int i=0;
    float p;
    float e_rel, e_abs;
    int cond_salida = 0;

    while (i<N_0){

        p = (*f)(p_0);
        e_abs = fabs(p-p_0);

        if (e_abs<TOL)
        {
            cond_salida = 1;
        }

        e_rel = fabs(p-*((M+(i-1)*nparam) + 1))/fabs(p);

        //escribiendo la matriz

        *((M+i*nparam) + 0) = i+1;  //iteraciones
        *((M+i*nparam) + 1) = p;    //p_i
        *((M+i*nparam) + 2) = e_rel;    //e_i

        i++;
        p_0=p;

        if(cond_salida==1){break;}
    }

    *ir = i;
    return p;
}
```

***Listado 6:*** *Función que realiza la iteración de punto fijo con la aceleración de Aitken*

```
float puntofijo_aitken(float (*f)(float), float p_0, float TOL,
                float N_0, float *M, int nparam, int *ir)
{
    int i = 0;
    float p, p_1, p_2;
    float e_rel, e_abs;
    int cond_salida = 0;

    while (i<N_0){

        p_1 = (*f)(p_0);
        p_2 = (*f)(p_1);

        //nos aseguramos de que no dividimos por cero

        if((p_2-2*p_1+p_0)!=0){
            p = p_0 - pow(p_1-p_0, 2)/(p_2-2*p_1+p_0);
            e_abs = fabs(p-p_0);
        }else{cond_salida = 1; e_abs = 0;}

        if (e_abs<TOL)
        {
            cond_salida = 1;
        }

        e_rel = fabs(p-*((M+(i-1)*nparam) + 3))/fabs(p);

        //escribiendo la matriz

        *((M+i*nparam) + 0) = i+1;
        *((M+i*nparam) + 1) = p_1;
        *((M+i*nparam) + 2) = p_2;
        *((M+i*nparam) + 3) = p;
        *((M+i*nparam) + 4) = e_rel;

        i++;
        p_0=p;

        if(cond_salida==1){break;}
    }

    *ir = i;
    return p;
}
```

***Listado 7:*** *Función que genera gráficas con GNUplot*

```
int graficaplot()
{
    #define NUM_COMANDOS 6
    int i;

    char * comandosGNUplot[] =
    {
        "set title 'E=+1; dr/dt=0'",
        "set xzeroaxis",
        "set yzeroaxis",
        "set ylabel 'y'",
        "set xlabel 'x'",
        "plot [0:10] 1/(2*x**2)-5/x-1 with lines
         lc rgb 'red' title 'E=1/(2r^2)-5/r'"
    };

    FILE * p = popen ("gnuplot -persist", "w");

    //ejecutamos los comandos en un bucle for

    for (i=0;i<NUM_COMANDOS;i++)
    {
        fprintf(p, "%s\n", comandosGNUplot[i]);
    }

    return 0;
}
```