**UNIVERSIDAD DE BUENOS AIRES**
**FACULTAD DE INGENIERÍA**
**<75.12> ANÁLISIS NUMÉRICO**

## DATOS DEL TRABAJO PRÁCTICO

| | | |
|---|---|---|
| 2 | 2016 | Análisis profundizado de algoritmia |
| | AÑO | para la resolución de un mismo |
| | 1 | problema a partir de un nuevo modelo |
| TP NRO | CUAT | TEMA |

## INTEGRANTES DEL GRUPO

| | | |
|---|---|---|
| 14 | Merlo Leiva Nahuel | 92115 |
| | APELLIDO Y NOMBRE | PADRÓN |
| | Cozza Fabrizio Luis | 97402 |
| GRUPO | APELLIDO Y NOMBRE | PADRÓN |

## DATOS DE LA ENTREGA

| | | | |
|---|---|---|---|
| CT14.txt | 8-85395 | 29062016 | 29062016 |
| ARCHIVO | NRO CONTROL | FECHA VENC | FECHA ENTR |

## CORRECCIONES

| FECHA | NOTA | OBSERVACIONES |
|---|---|---|
| | | |
| | | |
| DOCENTE | FIRMA | |
| | | |
| | | |

## Introducción

En la actualidad, científicos e ingenieros trabajan en problemas cada vez más complejos. En general se requiere el uso de computadoras para estudiar modelos matemáticos a través del uso de la algoritmia.

Mediante esta metodología, una computadora es capaz de calcular aproximaciones a la solución de un problema determinado. Debido a que la aritmética con que opera tiene precisión finita, pueden introducirse errores de redondeo y truncamiento en las operaciones efectuadas.

Entonces, es inevitable pensar en la utilización de algoritmos simples para reducir el tiempo de cómputo, uso de memoria como también de errores.

## Objetivos

- A partir de los resultados del trabajo práctico número 1 (TP1), realizar un análisis más exhaustivo a partir de la interpretación del cálculo, desarrollo y resultado del problema presentado teniendo en cuenta en el análisis teórico sus errores, estabilidad, orden y convergencia de los algoritmos presentados.
- A partir del cambio del modelo del problema matemático respecto del TP1, implementar algoritmos propuestos que permitan calcular la resolución del problema del movimiento de un cuerpo celeste respecto de otro y su precesión, mediante los hallazgos de Newton y el término propuesto por Einstein.

## Resumen

En primer lugar, a partir de los resultados obtenidos en el TP1, se realizarán cálculos de los semiejes, periodo y un cociente determinado por las leyes de Kepler, para analizar el cumplimento de las mismas y la convergencia de los métodos utilizados, siendo el algoritmo uno el método de Euler explícito y el algoritmo dos el método de Runge-Kutta 4. También a partir de diferenciación numérica se analizara la conservación de la energía del sistema.

Luego, en una segunda parte se desarrollarán dos algoritmos modificados respecto de los propuestos en el TP1 para aplicar la resolución al problema propuesto por Einstein (cambio de modelo).

Durante el desarrollo se realizaran cálculos para la obtención termino llamado precesión, como también cálculos para la energía, aumentando cada vez el “paso”, siendo cada uno al menos un orden de magnitud superior al previo. Finalmente llegando a la obtención de conclusiones respecto a su precesión y orden a través del análisis teórico e interpretación de gráficos.

# A: Análisis clásico de orbitas – Leyes de Kepler

En esta sección se analizará, a partir de resultados del TP1, la cuadratura y la diferenciación numérica, la convergencia de los resultados a calcular, como también el cumplimiento de las leyes de Kepler y la conservación de la energía.

## A.1: Primera Ley de Kepler (Algoritmo 1)

**Primera ley** (1609): "*Todos los planetas se desplazan alrededor del Sol describiendo órbitas elípticas. El Sol se encuentra en uno de los focos de la elipse*".

Se propone el cálculo de los ejes a través de la siguiente manera:

*Cálculo del semieje mayor como*: a = (Radio perihelio + Radio afelio) / 2

*Cálculo del semieje menor como*: b = a $*$ $\sqrt{(1 - e^2)}$

Con sus respectivas propagaciones de errores.

Cálculos del semieje mayor de la órbita Mustafar a partir de los resultados del TP1:

| **PASOS (N)** | **Semieje mayor aproximado ($\tilde{a}$)** | **Error del semieje mayor ($\Delta$a)** | **Semieje mayor (a) (a = $\tilde{a} \pm \Delta$a)** |
|---|---|---|---|
| 1,00E+02 | 6.08083e+010 | 864522 | 6.080830000e+06 ± 1e+06 |
| 1,00E+03 | 6,06744e+010 | 851339 | 6.067440000e+06 ± 1e+06 |
| 1,00E+04 | 6.06648e+010 | 850238 | 6.066480000e+06 ± 1e+06 |
| 1,00E+05 | 6.06639e+010 | 850130 | 6.066390000e+06 ± 1e+06 |
| 1,00E+06 | 6.06638e+010 | 850119 | 6.066380000e+06 ± 1e+06 |
| **1,00E+07** | **6.06638e+010** | **850118** | **6.066380000e+06 ± 1e+06** |
| 1,00E+08 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |
| 1,00E+09 | 6.06638e+010 | 850118 | 6.06638e0000+06 ± 1e+06 |
| 1,00E+10 | 6.06637e+010 | 850118 | 6.066370000e+06 ± 1e+06 |
| 1,00E+11 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |

Cálculos del semieje menor de la órbita Mustafar a partir de los resultados del TP1:

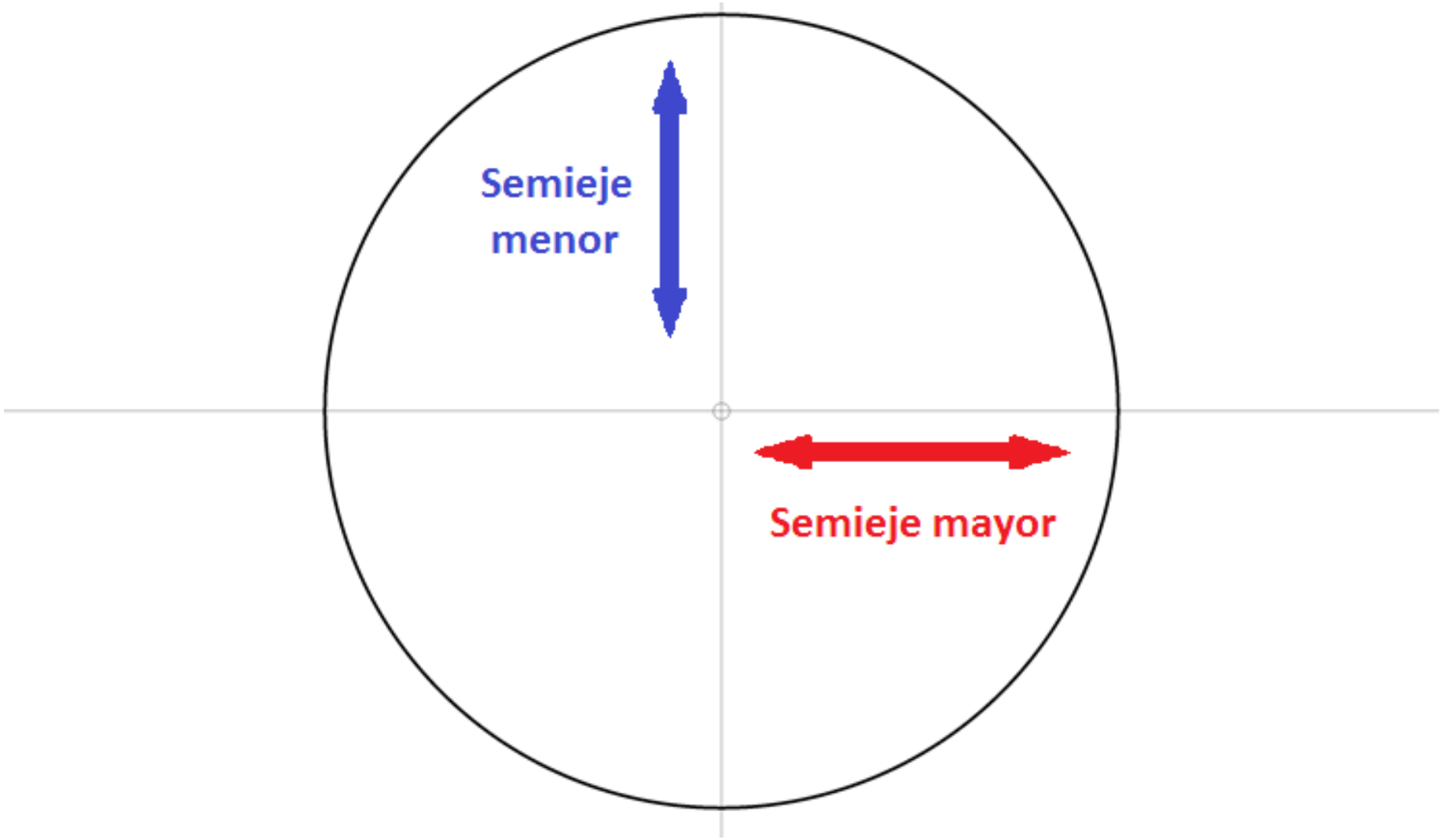
| **PASOS (N)** | **Semieje menor aproximado ($\widetilde{b}$)** | **Error del semieje menor ($\Delta$b)** | **Semieje menor (b) (b = $\widetilde{b} \pm \Delta$b)** |
|---|---|---|---|
| 1,00E+02 | 5.95688e+010 | 846900 | 5.956880000e+06 ± 1e+06 |
| 1,00E+03 | 5.94376e+010 | 833986 | 5.943760000e+06 ± 1e+06 |
| 1,00E+04 | 5.94282e+010 | 832907 | 5.942820000e+06 ± 1e+06 |
| 1,00E+05 | 5.94273e+010 | 832801 | 5.942730000e+06 ± 1e+06 |
| 1,00E+06 | 5.94272e+010 | 832790 | 5.942720000e+06 ± 1e+06 |
| **1,00E+07** | **5.94272e+010** | **832789** | **5.942720000e+06 ± 1e+06** |
| 1,00E+08 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |
| 1,00E+09 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |
| 1,00E+10 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |
| 1,00E+11 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |

**Análisis de la convergencia**

Se puede ver que el eje mayor converge a 6.06637e+010, mientras que el eje menor converge a 5.94272e+010, aunque es notable la diferencia entre los pasos pequeños con los pasos grandes, tal como se vio gráficamente en el TP1, hay una separación considerable en lo que refiere a semiejes entre el último paso (el que más se acerca a la solución real) con el primer paso mostrado, esto tiene que ver con el método utilizado (ahora se sabe que es un Euler explícito por las clases vistas) y su orden, por lo cual se destaca la variabilidad del error y en el caso del eje mayor se puede notar que no se llega a una solución concreta hasta un paso muy grande, es más, aun para un paso muy grande se ve una diferencia (mínima) entre el ultimo y anteúltimo paso, por lo cual no se podría confiar completamente en esta metodología de resolución.

Entonces, ¿Se cumple la primera ley de Kepler? La realidad es que como se mencionó, para pasos grandes la órbita si se podría parecer a la real de la forma elíptica y cerrada, incluso para pasos pequeños tiene una forma elíptica, pero también en este caso se ve un desfasaje notable en el lugar en el que la órbita finaliza, en relación al punto inicial y final el cual no debería ocurrir, como también la mencionada diferencia entre los semiejes, por lo cual un cálculo correcto para la primera ley requeriría de un cómputo sumamente grande si se quieren obtener soluciones confiables (como ya se mencionó se ve que no se llega a una solución concreta hasta un paso muy grande). En cuanto a que el Sol se encuentra en uno de los focos no interesa para este caso ya que un caso especial entre dos cuerpos, aquello se verá en la parte B.1 de este trabajo.

***Gráfico de la órbita y semiejes para el último paso:***



## A.2: Segunda Ley de Kepler (Algoritmo 1)

**Segunda ley** (1609): "*El radio vector que une un planeta y el Sol barre áreas iguales en tiempos iguales*".

Se propone la siguiente ecuación para el análisis del cumplimiento de la segunda ley en la órbita Mustafar:

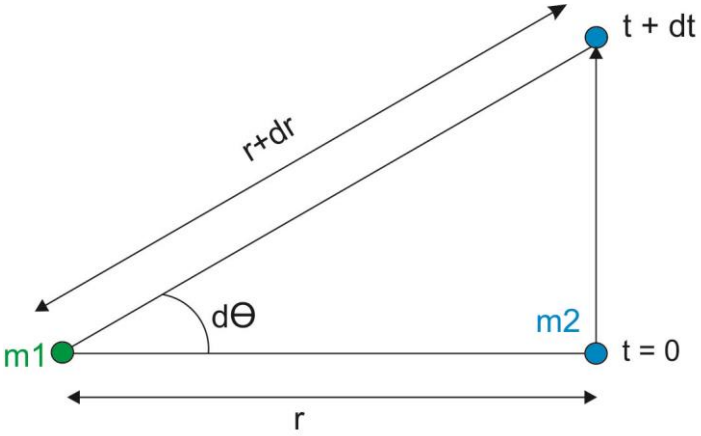
$$\frac{dA}{dt} = \frac{1}{2}h$$

Donde *dA* es el área barrida por el radio vector desde la estrella al planeta en el tiempo *dt* y h el momento angular especifico.

A partir de cuadratura numérica la idea es llegar al resultado del periodo tal que:

$$\int dA = \frac{h}{2}\int dt \Rightarrow T = \frac{2\Omega}{\mathrm{h}}$$

Siendo T el periodo y $\Omega = \int dA$ el área barrida de una órbita, que mediante el siguiente esquema:




$$\Rightarrow dA = \frac{1}{2}r^2\,d\theta$$

Considerando dr chico y $\sin\theta \cong \theta$

Como el algoritmo uno programa el método de Euler explícito de $\varphi(1)$ (orden uno), se podrán usar métodos de cuadratura numérica para la resolución respecto al orden correspondiente, por lo cual sería coincidente utilizar cuadratura numérica por medio de intervalos regulares estrechos a través del método de rectángulo.

A través de este método el error se calcula de la siguiente manera:

$$Err \cong \frac{(b-a)^3}{24\,N^2}|f''(\mathfrak{z})|$$

Siendo N la cantidad de pasos, b el extremo final del intervalo elegido y a el extremo inicial del intervalo elegido I[a,b] y $\mathfrak{z}$ un punto perteneciente al intervalo.

Utilizando este método se llega a los siguientes resultados:

| **PASOS (N)** | **Área aproximada ($\widetilde{\Omega}$)** | **Error del área($\Delta\Omega$)** | **Área ($\Omega$) ($\Omega = \widetilde{\Omega} \pm \Delta\Omega$)** |
|---|---|---|---|
| 1,00E+02 | 1.13797e+022 | 3.23575e+017 | 1.1380000e+020 ± 1.000e+020 |
| 1,00E+03 | 1.13297e+022 | 3.17939e+017 | 1.1329700e+020 ± 1.000e+020 |
| 1,00E+04 | 1.13261e+022 | 3.17478e+017 | 1.1326100e+020 ± 1.000e+020 |
| 1,00E+05 | 1.13258e+022 | 3.17432e+017 | 1.1325800e+020 ± 1.000e+020 |
| 1,00E+06 | 1.13257e+022 | 3.17428e+017 | 1.1325700e+020 ± 1.000e+020 |
| **1,00E+07** | **1.13257e+022** | **3.17427e+017** | **1.1325700e+020 ± 1.000e+020** |
| 1,00E+08 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| 1,00E+09 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| 1,00E+10 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| 1,00E+11 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| **PASOS (N)** | **Período aproximada ($\widetilde{T}$)** | **Error del período ($\Delta$T)** | **Período (T) (T = $\widetilde{T} \pm \Delta$T)** |
| 1,00E+02 | 1.05982e+013 | 1.22033e+010 | 1.06e+011 ± 1e+011 |
| 1,00E+03 | 1.05516e+013 | 1.19908e+010 | 1.06e+011 ± 1e+011 |
| 1,00E+04 | 1.05482e+013 | 1.19734e+010 | 1.06e+011 ± 1e+011 |
| 1,00E+05 | 1.05480e+013 | 1.19717e+010 | 1.05e+011 ± 1e+011 |
| 1,00E+06 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| **1,00E+07** | **1.05479e+013** | **1.19715e+010** | **1.05e+011 ± 1e+011** |
| 1,00E+08 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| 1,00E+09 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| 1,00E+10 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| 1,00E+11 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |

**Análisis de la convergencia**

Se puede ver que el área y periodo convergen a un resultado. Sin embargo, al igual que sucedió con el cálculo de los semiejes, se nota una diferencia entre las áreas calculadas y los periodos calculados respecto de pasos chicos y grandes, especialmente porque en los primeros pasos también hay un desvío producido por el análisis numérico en cuanto a donde finaliza la elipse (no es el mismo punto final que el inicial). La variabilidad del resultado no se refleja tanto en los cálculos del área ni del periodo (si bien se ve que para pasos pequeños no se llega a un resultado concreto) pero si en el error del problema debido a su orden.

Entonces, ¿Se cumple la segunda ley de Kepler? Como este ítem es más específico, en un comienzo se puede decir que no se cumple, al menos no exactamente lo esperado en esta segunda ley, principalmente por todo aquello mencionado en el TP1 acerca de los errores, en este ítem influye significativamente la propagación de errores, el método utilizado, tanto el de Euler como el de cuadratura utilizado obligado a tener un orden similar y lenta convergencia, entre otros factores como por ejemplo un modelo matemático incorrecto (ya que luego se descubrió lo propuesto por Einstein), como también las limitaciones computacionales.

## A.3: Tercera Ley de Kepler (Algoritmo 1)

**Tercera ley** (1618): "*Para cualquier planeta, el cuadrado de su período orbital es directamente proporcional al cubo de la longitud del semieje mayor de su órbita elíptica*".

Se analizará a través de la obtención de una constante por medio del cociente:

$$\frac{{T_N}^2}{\langle R_N \rangle^3} = C$$

Donde $\langle R_N \rangle = \frac{1}{2}\big((r(\theta = 0) + r(\theta = \pi)\big)$ es el semieje mayor calculado de la órbita para cada N.

Y su error se calculará como la propagación de errores del cociente.

Se llega a los siguientes resultados:

| PASOS (N) | Cociente aproximado | Error del cociente | Cociente |
|---|---|---|---|
| 1,00E+02 | 4.99545e-007 | 1.17171e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+03 | 4.98445e-007 | 1.15385e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+04 | 4.98366e-007 | 1.15235e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+05 | 4.98358e-007 | 1.15221e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+06 | 4.98358e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |

| | | | |
|---|---|---|---|
| **1,00E+07** | **4.98358e-007** | **1.15219e-009** | **5.00e-007 ± 1.00e-007** |
| 1,00E+08 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+09 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+10 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+11 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |

**Análisis de la convergencia**

Si bien se puede ver que el método converge a una constante, la tercera ley propone que siempre se obtenga el mismo número, por lo cual para este método no se cumple y se denota una convergencia lenta. Algo que ayudaría el cálculo en este caso es la presencia de estar elevadas las variables (aunque también es contraproducente en el cálculo del error).

## A.4: Integral de energía del sistema (Algoritmo 1)

Se analizará la conservación de la energía del sistema a través de su fórmula correspondiente, a partir de la diferenciación numérica:

$$E_n = \frac{1}{2} {v_n}^2 - \mu\, u_n$$

Donde ${v_n}^2 = h^2 \left[ {u_n}^2 + \left(\frac{du}{d\theta}\right)^2 \right]$, siendo $u_n$ la aproximación numérica de $u(\theta_n)$ con $\theta_n = nk$ y siendo $k = \frac{2\pi}{N}$

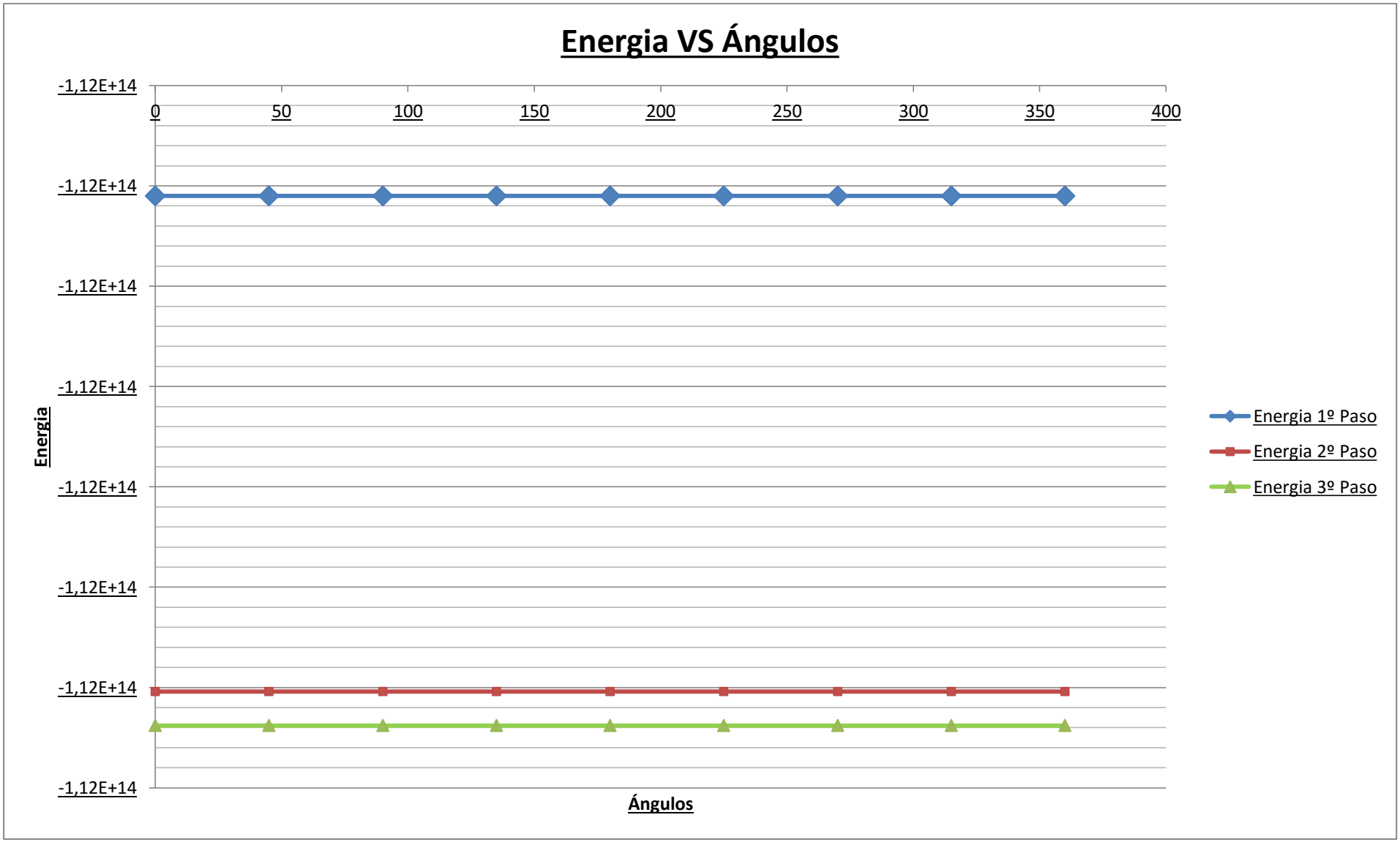
Como ya fue mencionado en la sección A.2, este algoritmo posee orden uno, por lo cual deberá realizarse una diferenciación numérica acorde al orden especificado, siendo la resolución una posible derivada en atraso o en adelanto, a partir de la conversión de la segunda derivada en una derivada primera, de otra manera se utilizaría un método directo para la segunda derivada.

Se llega a los siguientes resultados:

| **PASOS (N)** | $\boldsymbol{E_n}$ |
|---|---|
| 1,00E+02 | -1.11705e+009 |
| 1,00E+03 | -1.11952e+009 |
| 1,00E+04 | -1.11969e+009 |
| 1,00E+05 | -1.11971e+009 |

| | |
|---|---|
| 1,00E+06 | -1.11971e+009 |
| **1,00E+07** | **-1.11971e+009** |
| 1,00E+08 | -1.11971e+009 |
| 1,00E+09 | -1.11971e+009 |
| 1,00E+10 | -1.11971e+009 |
| 1,00E+11 | -1.11971e+009 |

Se ve que luego de algunos pasos el método converge a un resultado y se observa una diferencia entre los primeros 3 pasos en lo que respecta a la energía, pero por lo cual se podría decir que el problema, con este método, en un principio, tendería a ser conservativo, aunque no sería correcto afirmarlo. Esto sucede una vez más porque el método es de orden bajo y convergencia lenta. En el grafico se ve una diferencia notable sobre todo en el primer paso.


Energia VS Ángulos
-1,12E+14
-1,12E+14
-1,12E+14
-1,12E+14
-1,12E+14
-1,12E+14
-1,12E+14
Energia
0
50
100
150
200
250
300
350
400
Ángulos
Energia 1º Paso
Energia 2º Paso
Energia 3º Paso

## A.5: Primera Ley de Kepler (Algoritmo 2)

La resolución es análoga al ítem A.1.

Cálculos del semieje mayor de la órbita Mustafar a partir de los resultados del TP1:

| PASOS (N) | Semieje mayor aproximado ($\tilde{a}$) | Error del semieje mayor ($\Delta$a) | Semieje mayor (a) (a = $\tilde{a} \pm \Delta$a) |
|---|---|---|---|
| 1,00E+02 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |
| 1,00E+03 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |
| 1,00E+04 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |
| 1,00E+05 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |
| 1,00E+06 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |
| **1,00E+07** | **6.06638e+010** | **850118** | **6.066380000e+06 ± 1e+06** |
| 1,00E+08 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |
| 1,00E+09 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |
| 1,00E+10 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |
| 1,00E+11 | 6.06638e+010 | 850118 | 6.066380000e+06 ± 1e+06 |

Cálculos del semieje menor de la órbita Mustafar a partir de los resultados del TP1:

| PASOS (N) | Semieje menor aproximado ($\tilde{b}$) | Error del semieje menor ($\Delta$b) | Semieje menor (b) (b = $\tilde{b} \pm \Delta$b) |
|---|---|---|---|
| 1,00E+02 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |
| 1,00E+03 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |
| 1,00E+04 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |
| 1,00E+05 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |
| 1,00E+06 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |
| **1,00E+07** | **5.94272e+010** | **832789** | **5.942720000e+06 ± 1e+06** |
| 1,00E+08 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |
| 1,00E+09 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |
| 1,00E+10 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |

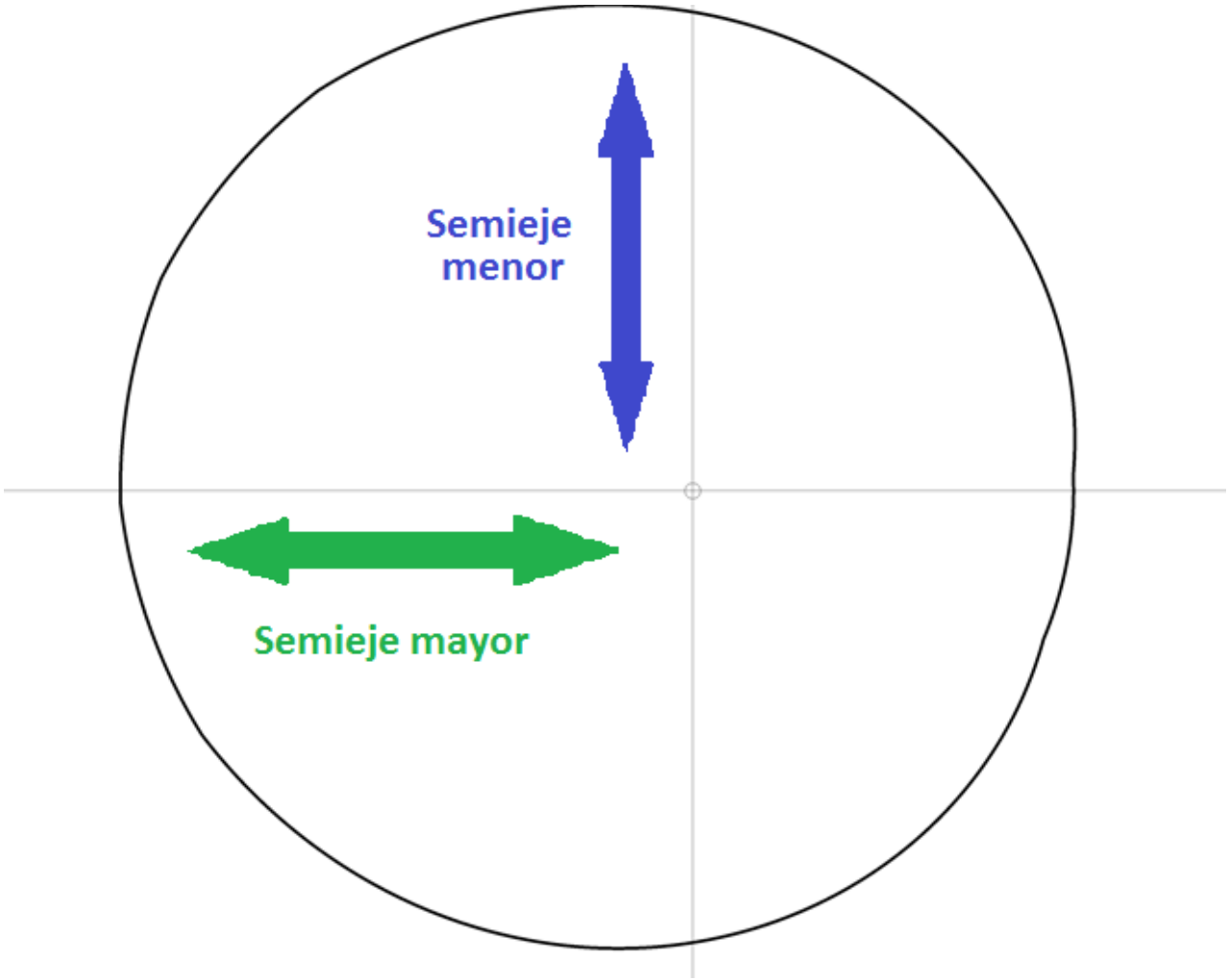
| | | | |
|---|---|---|---|
| 1,00E+11 | 5.94272e+010 | 832789 | 5.942720000e+06 ± 1e+06 |

**Análisis de la convergencia**

Se puede ver claramente que el eje mayor converge a 6.06638e+010, mientras que el eje menor converge a 5.94272e+010, pero a diferencia del primer algoritmo, la diferencia entre los pasos pequeños con los pasos grandes, tal como se vio gráficamente en el TP1, es prácticamente nula en lo que refiere a semiejes entre el último paso (el que más se acerca a la solución real) con el primer paso mostrado, esto tiene que ver con el método utilizado (ahora se sabe que es un RK4 por las clases vistas) y su orden, como también el cálculo (sencillo) propuesto para el cálculo de los semiejes. Es notable la consistencia de los cálculos a través de cualquier paso que se utilice, tanto para el resultado como para el error, y en relación a los pasos grandes del Euler, el error es igual.

Entonces, ¿Se cumple la primera ley de Kepler? Claramente se ve que el orden del método ayuda mucho a obtener hasta con un paso pequeño una solución efectiva buscada para cualquier paso utilizado, y el sistema cumple y realiza una órbita elíptica y cerrada. En cuanto la referencia al Sol, presenta el mismo análisis que el punto A.1.

***Gráfico de la órbita y semiejes para el último paso:***



## A.6: Segunda Ley de Kepler (Algoritmo 2)

La resolución es análoga al ítem A.2 pero como el algoritmo 2 programa el método de Runge-Kutta 4 de $\varphi(4)$ (orden cuatro), se podrán usar métodos de cuadratura numérica para la resolución respecto al orden correspondiente, por lo cual sería coincidente utilizar cuadratura numérica por método de Simpson.

A través de este método el error se calcula de la siguiente manera:

$$Err \cong \frac{h^5}{90} |f^{IV}(\xi)|$$

Siendo h = (b – a) / N, b, a y N aquellos mencionados en la sección A.2.

Utilizando este método se llega a los siguientes resultados:

| **PASOS (N)** | **Área aproximada ($\widetilde{\Omega}$)** | **Error del área($\Delta\Omega$)** | **Área ($\Omega$) ($\Omega = \widetilde{\Omega} \pm \Delta\Omega$)** |
|---|---|---|---|
| 1,00E+02 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| 1,00E+03 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| 1,00E+04 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| 1,00E+05 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| 1,00E+06 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| **1,00E+07** | **1.13257e+022** | **3.17427e+017** | **1.1325700e+020 ± 1.000e+020** |
| 1,00E+08 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| 1,00E+09 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| 1,00E+10 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| 1,00E+11 | 1.13257e+022 | 3.17427e+017 | 1.1325700e+020 ± 1.000e+020 |
| **PASOS (N)** | **Período aproximada ($\widetilde{T}$)** | **Error del período ($\Delta$T)** | **Período (T) (T = $\widetilde{T} \pm \Delta$T)** |
| 1,00E+02 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| 1,00E+03 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| 1,00E+04 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |

| | | | |
|---|---|---|---|
| 1,00E+05 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| 1,00E+06 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| **1,00E+07** | **1.05479e+013** | **1.19715e+010** | **1.05e+011 ± 1e+011** |
| 1,00E+08 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| 1,00E+09 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| 1,00E+10 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |
| 1,00E+11 | 1.05479e+013 | 1.19715e+010 | 1.05e+011 ± 1e+011 |

**Análisis y orden de la convergencia**

Se ve claramente que el área y periodo convergen a un resultado al igual que sucedió con los semiejes. Entonces, si bien nada es exacto, la consistencia del método en los resultados como también el error, se puede sugerir que este método, debido a su orden, propone una buena aproximación a la resolución del problema de la segunda ley de Kepler. Y entonces el movimiento en todas las órbitas elípticas con el eje mayor tiene el mismo periodo.

## A.7: Tercera Ley de Kepler (Algoritmo 2)

La resolución es análoga al ítem A.3.

Se llega a los siguientes resultados:

| PASOS (N) | Cociente aproximado | Error del cociente | Cociente |
|---|---|---|---|
| 1,00E+02 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+03 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+04 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+05 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+06 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |
| **1,00E+07** | **4.98357e-007** | **1.15219e-009** | **5.00e-007 ± 1.00e-007** |
| 1,00E+08 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+09 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |

| | | | |
|---|---|---|---|
| 1,00E+10 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |
| 1,00E+11 | 4.98357e-007 | 1.15219e-009 | 5.00e-007 ± 1.00e-007 |

**Análisis y orden de la convergencia**

El resultado converge indudablemente a una constante, por lo cual la tercera ley se cumple, la convergencia es más rápida por el método utilizado, como también más exacto.

## A.8: Integral de energía del sistema (Algoritmo 2)

La resolución es análoga al ítem A.4 pero como ya fue mencionado en la sección A.6, este algoritmo posee orden cuatro, por lo cual deberá realizarse una diferenciación numérica acorde al orden especificado, siendo la resolución una posible aproximación por polinomios de Taylor o por medio del método de Richardson.

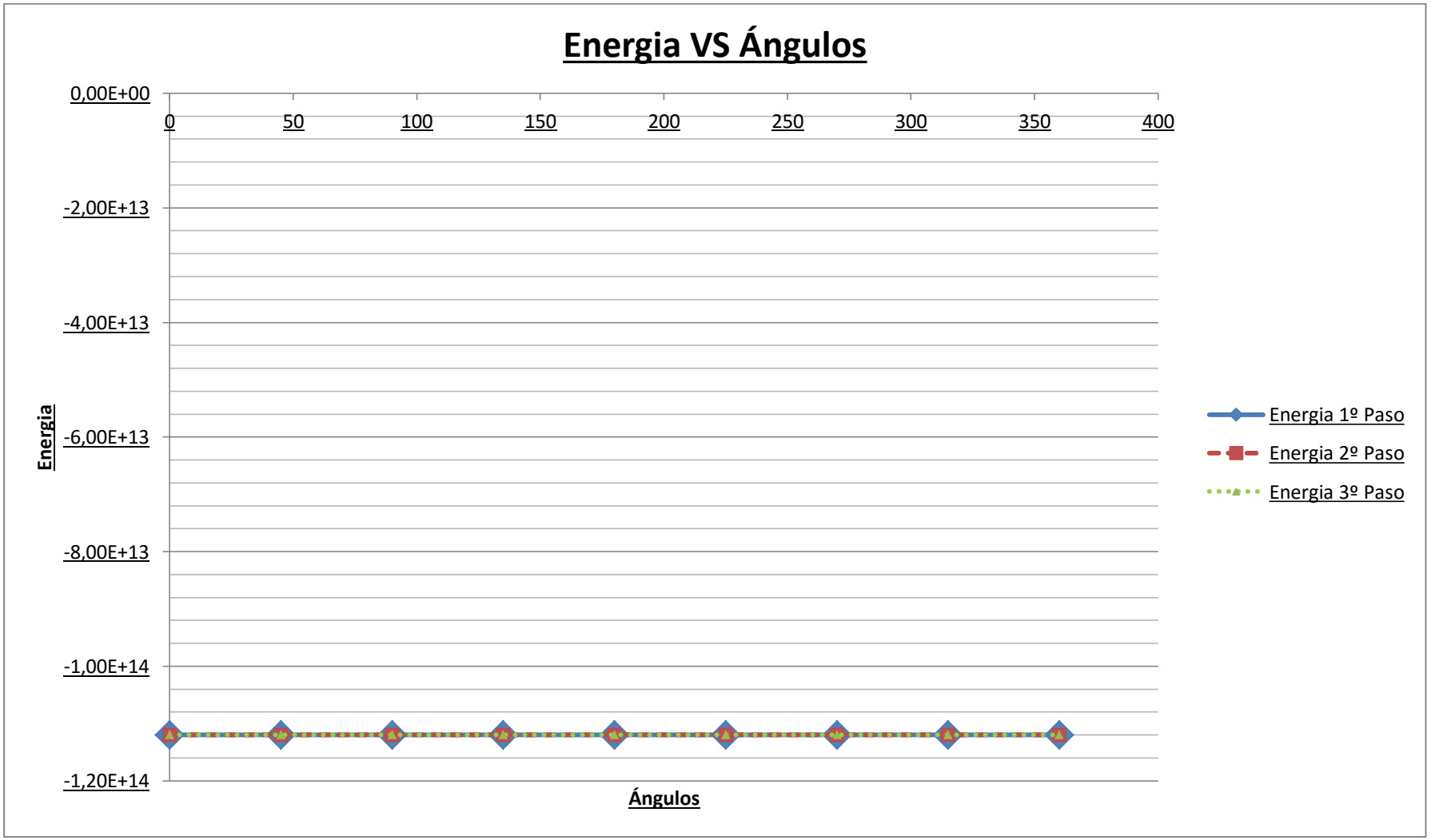
Se llega a los siguientes resultados:

| **PASOS (N)** | $E_n$ |
|---|---|
| 1,00E+02 | -1.11971e+009 |
| 1,00E+03 | -1.11971e+009 |
| 1,00E+04 | -1.11971e+009 |
| 1,00E+05 | -1.11971e+009 |
| 1,00E+06 | -1.11971e+009 |
| **1,00E+07** | **-1.11971e+009** |
| 1,00E+08 | -1.11971e+009 |
| 1,00E+09 | -1.11971e+009 |
| 1,00E+10 | -1.11971e+009 |
| 1,00E+11 | -1.11971e+009 |

**Análisis y convergencia**

Se ve que el método converge a un único resultado. El resultado es idéntico para las limitaciones computacionales, entonces se puede afirmar que el problema conservativo con el uso del RK4. . Esto sucede una vez más porque el método es de orden alto y convergencia rápida (aunque ocupa mayor tiempo de cómputo que Euler). Al igual como lo mencionado en

el caso del periodo, se tiene la misma energía total en órbitas elípticas para todos aquellos pertenecientes al eje mayor, independientemente del parámetro e de la elipse.


Energia VS Ángulos
0,00E+00
-2,00E+13
-4,00E+13
-6,00E+13
-8,00E+13
-1,00E+14
-1,20E+14
Energia
0
50
100
150
200
250
300
350
400
Ángulos
Energia 1º Paso
Energia 2º Paso
Energia 3º Paso

# B: Análisis relativista de orbitas – Leyes de Einstein

En esta sección se utilizará, a través de un cambio de modelo respecto del TP1, dos nuevos algoritmos generalizados con el objetivo, en un principio, de validar como también calcular un nuevo termino llamado precesión. Luego se averiguara la integral de energía del sistema para comprar con el resultado clásico y se analizará si los algoritmos nuevos son útiles para la resolución de este problema.

## B.1: Algoritmo 1-GR, precesión y su respectiva validación

La ley de gravitación de Newton propone una solución al problema del movimiento de cuerpos celestes, siendo este acotado a un plano.

Utilizando un sistema de coordenadas polares con origen en uno de los cuerpos y teniendo en cuenta esta vez las leyes de Einstein (hay un cambio de modelo matemático respecto del TP1), el movimiento del otro cuerpo puede ser representado por las siguientes ecuaciones:

$$\frac{d^2u}{d\theta} + u = \frac{\mu}{h^2} + 3\frac{GM}{c^2}\ u^2 \qquad (1)$$

$$u = \frac{1}{r}\ , \mu = G\ (m_1 + m_2) = GM \qquad (2)$$

$$\theta = 0, \frac{du}{d\theta} = 0 \qquad (3)$$

La idea es mostrar el modelo relativista planteado por Einstein para luego mostrar las soluciones por medio de algoritmia, independientemente de los parámetros especificados.

Los algoritmos a desarrollar se calcularan con simple precisión y utilizan las mismas variables del TP1 excepto por la aparición de la velocidad de la luz que está determinada por

c = $3x10^8$ m/s

Su desarrollo está dado por el siguiente pseudocódigo:

**Algoritmo 1-GR**

$u_0 = a^{-1}\ (1 - e)^{-1}$

$v_0 = 0$

Desde $n = 0$ a N - 1

$$u_{n+1} = u_n + kv_n$$

$$v_{n+1} = v_n - ku_n + k\mu h^{-2} + k\ 3\ GM\ c^{-2}\ {u_n}^2$$

Avanzar n

Debido a que la precesión es un corrimiento en el ángulo barrido por el planeta luego de 1 órbita completa, en este inciso se realizará una validación del valor teórico calculado de la precesión para el sistema Sol-Mercurio, siendo este valor $\Delta\vartheta = \frac{43seg\ de\ arco}{siglo}$ (precesión), por medio de la consideración de utilizar un N adecuado y lo suficientemente grande tal que se puedan obtener soluciones confiables teniendo en cuenta el análisis en el inciso B.8 del TP1, evitando la cancelación de términos dependiente del k obtenido según el paso, como también tener en cuenta la capacidad de cálculo del PC y la precisión. Se modificará únicamente para este caso el valor de la variable $\lambda$, que solo en este inciso será $\lambda = 1$.

| **PASOS (N)** | **Precesión ($\vartheta$)** |
|---|---|
| 1,00E+02 | 49.9853 |
| 1,00E+03 | 50.1129 |
| 1,00E+04 | 50.1224 |
| 1,00E+05 | 50.1233 |
| 1,00E+06 | 50.1234 |
| **1,00E+07** | **50.1234** |
| 1,00E+08 | 50.1234 |
| 1,00E+09 | 50.1234 |
| 1,00E+10 | 50.1234 |
| 1,00E+11 | 50.1234 |

Los datos mostrados se acercan al valor real y se mantienen en un valor luego de un paso relativamente grande. Dado que los valores encontrados no validan correctamente el resultado buscado, se debería realizar una interpolación (no se realizó) debido a la resolución por medio del análisis numérico realizado y el corrimiento del ángulo, de manera que de esta forma se llegue a el resultado esperado, pero es claro que este algoritmo da múltiples valores según el paso utilizado, y nunca se llegará al resultado real buscado, sino únicamente para un paso grande. Si se tuviera una CPU con mayor capacidad de cálculo y precisión quizás podrían continuar obteniéndose valores incluso luego de los pasos mostrados, para llegar al resultado real buscado aunque tal vez no valga la pena involucrar tanto tiempo y costo de cómputo para poder obtener un resultado más preciso si no es necesario.

## B.2: Cálculo de la precesión Estrella-Mustafar (Algoritmo 1-GR)

Aquí el valor de $\lambda$ vuelve a ser el establecido en el TP1. Y se calculó la precesión para este caso dando los siguientes resultados:

| PASOS (N) | Precesión ($\vartheta$) |
|---|---|
| 1,00E+02 | 48.7584 |
| 1,00E+03 | 48.8666 |
| 1,00E+04 | 48.8738 |
| 1,00E+05 | 48.8745 |
| 1,00E+06 | 48.8746 |
| **1,00E+07** | 48.8746 |
| 1,00E+08 | 48.8746 |
| 1,00E+09 | 48.8746 |
| 1,00E+10 | 48.8746 |
| 1,00E+11 | 48.8746 |

Como el valor de λ es más grande en este caso, es lógico obtener valores de precesión más pequeños a los calculados, pero al igual que a lo largo del TP, los valores de pasos pequeños son inconsistentes debido al orden y convergencia del método, pudiéndose esto mejor con el uso de interpolación y aumento de precisión del CPU, sin la garantía de que al aumentar el tiempo de computo como también de precisión aumente la calidad del resultado para este algoritmo.

## B.3: Integral de energía del sistema (Algoritmo 1-GR)

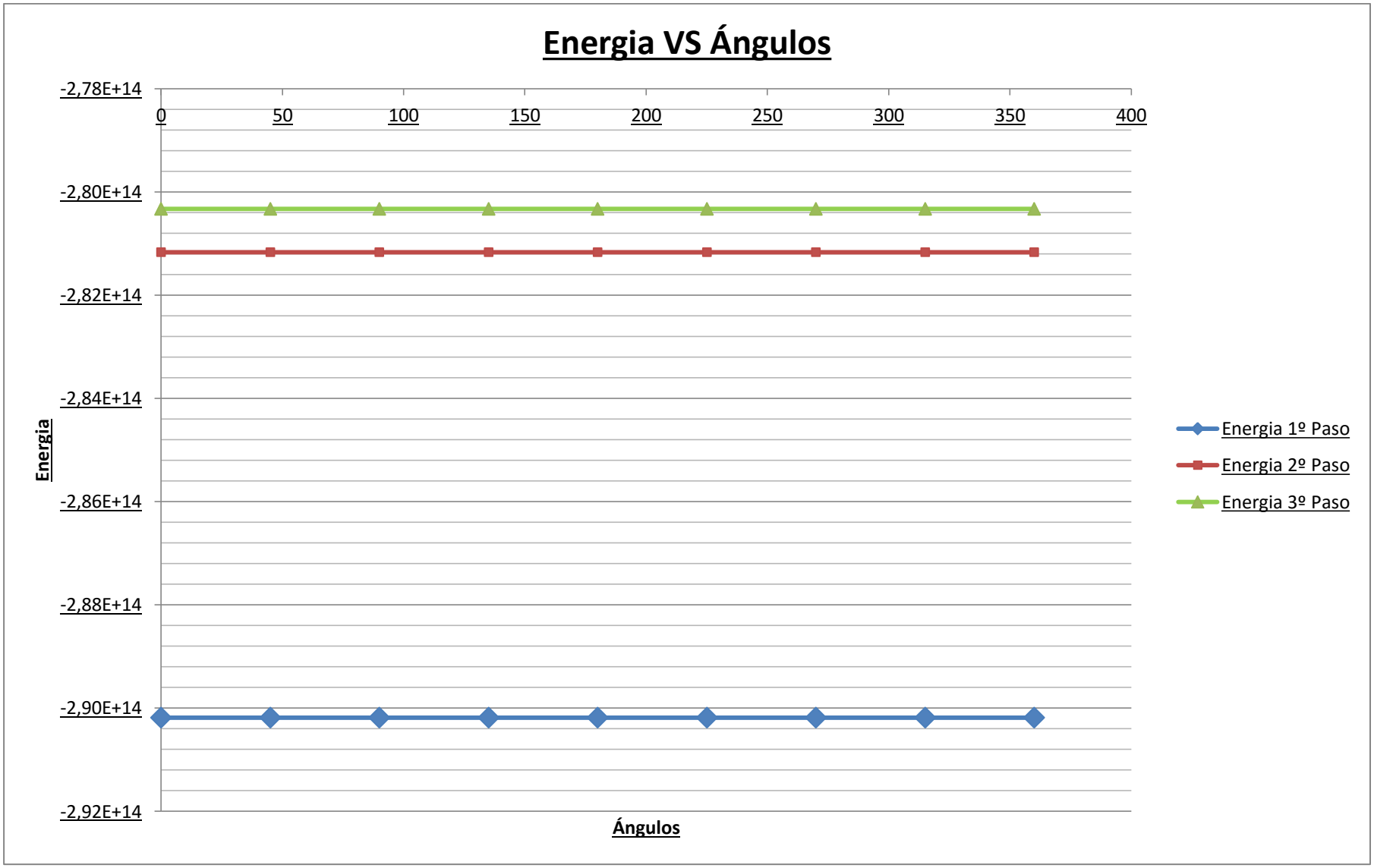
La resolución es similar al ítem A.4 para el cálculo de la energía pero como este algoritmo es otro, se tienen diferentes resultados, aunque igualmente se sigue poseyendo orden uno, por lo cual deberá realizarse una diferenciación numérica acorde al orden especificado, siendo la resolución una posible derivada en atraso o en adelanto, a partir de la conversión de la segunda derivada en una derivada primera, de otra manera se utilizaría un método directo para la segunda derivada.

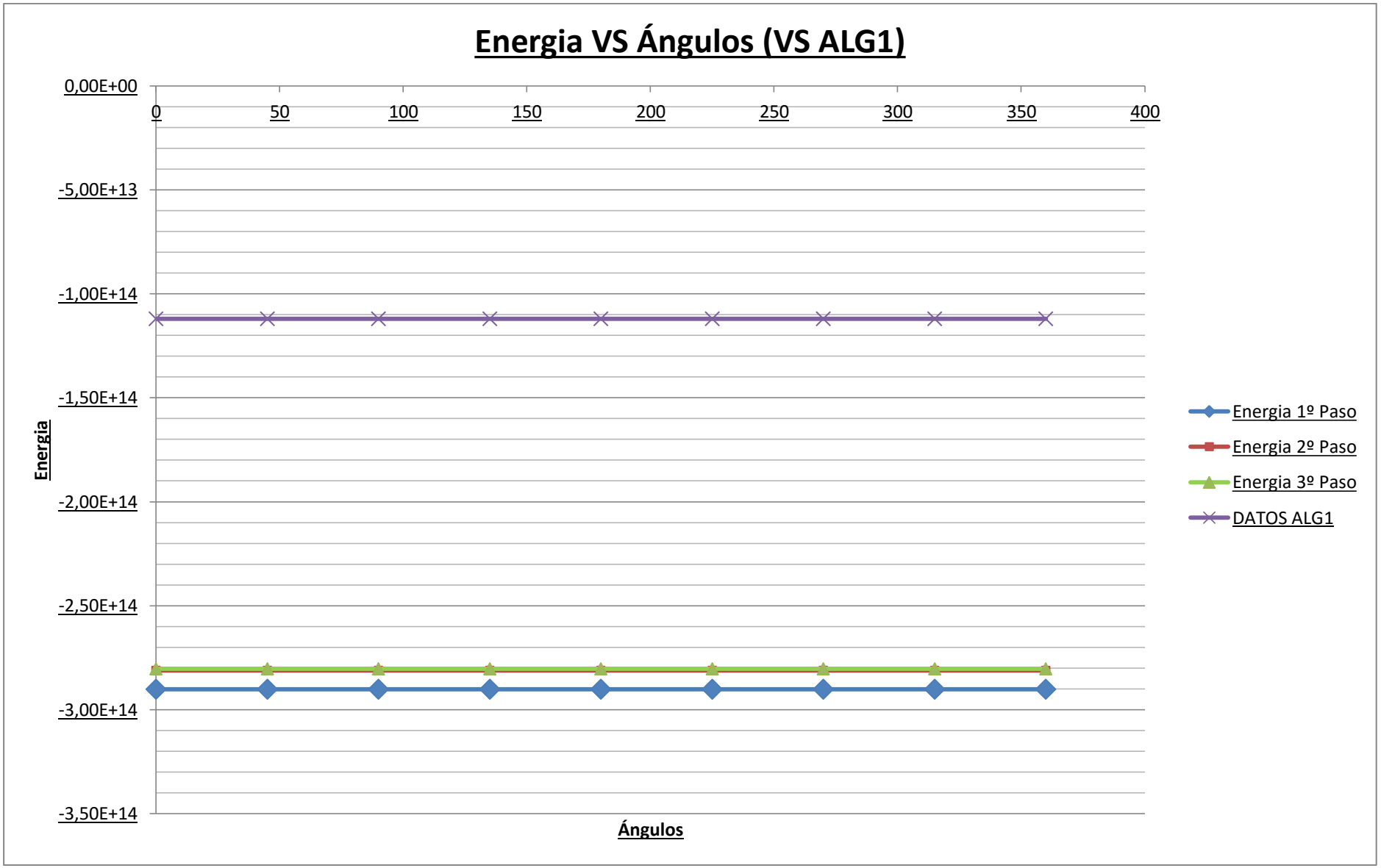
| PASOS (N) | $E_n$ |
|---|---|
| 1,00E+02 | -2.90188e+009 |
| 1,00E+03 | -2.81169e+009 |
| 1,00E+04 | -2.80329e+009 |
| 1,00E+05 | -2.80245e+009 |
| 1,00E+06 | -2.80237e+009 |

| **1,00E+07** | **-2.80236e+009** |
|---|---|
| 1,00E+08 | -2.80236e+009 |
| 1,00E+09 | -2.80236e+009 |
| 1,00E+10 | -2.80236e+009 |
| 1,00E+11 | -2.80236e+009 |

Se ve que luego de algunos pasos el método converge a un resultado y se observa una diferencia entre los primeros pasos en lo que respecta a la energía, pero por lo cual se podría decir que el problema, con este método, en un principio, tendería a ser conservativo, aunque no sería correcto afirmarlo. Para los valores presentes en el sistema solar los resultados cuantitativos de la teoría einsteniana son de alguna manera cercanos a la teoría newtoniana (e+009), pero diferentes en el sentido de su valor claramente por la aparición del nuevo termino de Einstein, que si bien, es un término despreciable para la teoría propuesta por Newton, denota su importancia por ejemplo a través de este cálculo en relación a la energía.

En los gráficos se puede ver la diferencia entre pequeños pasos, y la diferencia con el algoritmo uno sin la teoría relativista.


Energia VS Ángulos
-2,78E+14
-2,80E+14
-2,82E+14
-2,84E+14
-2,86E+14
-2,88E+14
-2,90E+14
-2,92E+14
Energia
0
50
100
150
200
250
300
350
400
Ángulos
Energia 1º Paso
Energia 2º Paso
Energia 3º Paso


Energia VS Ángulos (VS ALG1)
0,00E+00
-5,00E+13
-1,00E+14
-1,50E+14
-2,00E+14
-2,50E+14
-3,00E+14
-3,50E+14
Energia
0
50
100
150
200
250
300
350
400
Ángulos
Energia 1º Paso
Energia 2º Paso
Energia 3º Paso
DATOS ALG1

## B.4: Análisis de aptitud del algoritmo 1-GR

Si bien el nuevo termino de Einstein se utiliza para validar y verificar teoremas relativistas particulares, el pequeño termino que aparece influye en este algoritmo bruscamente por el orden que este posee, por lo cual se podría decir que este algoritmo será apto para una visión genérica del problema en cuestión, para pasos muy grandes, pasos que quizás deban ser calculados en lo posible (para verlo claramente) en una supercomputadora.

## B.5: Algoritmo 2-GR, precesión y su respectiva validación

Este inciso es análogo al B.1 pero con la generalización del algoritmo 2 del TP1.

Su desarrollo está dado por el siguiente pseudocódigo:

**Algoritmo 2-GR**

$u_0 = a^{-1}\,(1-e)^{-1}$

$v_0 = 0$

$\alpha = \mu h^{-2} + 3\,GM\,c^{-2}\,{u_n}^2$

Desde $n = 0$ a N - 1

$w_1 = u_n + k\,v_n\,/\,2$

$z_1 = v_n + k\,(\alpha - u_n)\,/\,2$

$w_2 = u_n + k\,z_1\,/\,2$

$z_2 = v_n + k\,(\alpha - w_1)\,/\,2$

$w_3 = u_n + k\,z_2$

$z_3 = v_n + k\,(\alpha - w_2)\,/\,2$

$u_{n+1} = u_n + k\,(v_n + 2\,z_1 + 2\,z_2 + z_3)\,/\,6$

$v_{n+1} = v_n + k\,(6\alpha - u_n - 2\,w_1 - 2\,w_2 - w_3)\,/\,6$

Avanzar n

| PASOS (N) | Precesión ($\vartheta$) |
|---|---|
| 1,00E+02 | 50.1234 |
| 1,00E+03 | 50.1234 |
| 1,00E+04 | 50.1234 |

| | |
|---|---|
| 1,00E+05 | 50.1234 |
| 1,00E+06 | 50.1234 |
| **1,00E+07** | **50.1234** |
| 1,00E+08 | 50.1234 |
| 1,00E+09 | 50.1234 |
| 1,00E+10 | 50.1234 |
| 1,00E+11 | 50.1234 |

El algoritmo es bastante consistente durante todos sus pasos por eso sería una mucha mejor aproximación a la validación del problema.

Dado que los valores encontrados no validan correctamente el resultado buscado, se debería realizar una interpolación (no se realizó) debido a la resolución por medio del análisis numérico realizado y el corrimiento del ángulo, de manera que de esta forma se llegue a el resultado esperado, pero es claro que este algoritmo valores consistentes para cualquier paso, y posiblemente si se utilizan la metodología mencionada se llegará al resultado real buscado.

## B.6: Cálculo de la precesión Estrella-Mustafar (Algoritmo 2-GR)

Inciso análogo al B.2. Se calculó la precesión para este caso dando los siguientes resultados:

| **PASOS (N)** | **Precesión ($\vartheta$)** |
|---|---|
| 1,00E+02 | 48.8746 |
| 1,00E+03 | 48.8746 |
| 1,00E+04 | 48.8746 |
| 1,00E+05 | 48.8746 |
| 1,00E+06 | 48.8746 |
| **1,00E+07** | **48.8746** |
| 1,00E+08 | 48.8746 |
| 1,00E+09 | 48.8746 |
| 1,00E+10 | 48.8746 |
| 1,00E+11 | 48.8746 |

Como el valor de $\lambda$ es más grande en este caso, es lógico obtener valores de precesión más pequeños a los calculados, pero al igual que a lo largo del TP, los valores de cualquier paso son

consistentes debido al orden y convergencia del método, pudiéndose esto mejorar con el uso de interpolación.

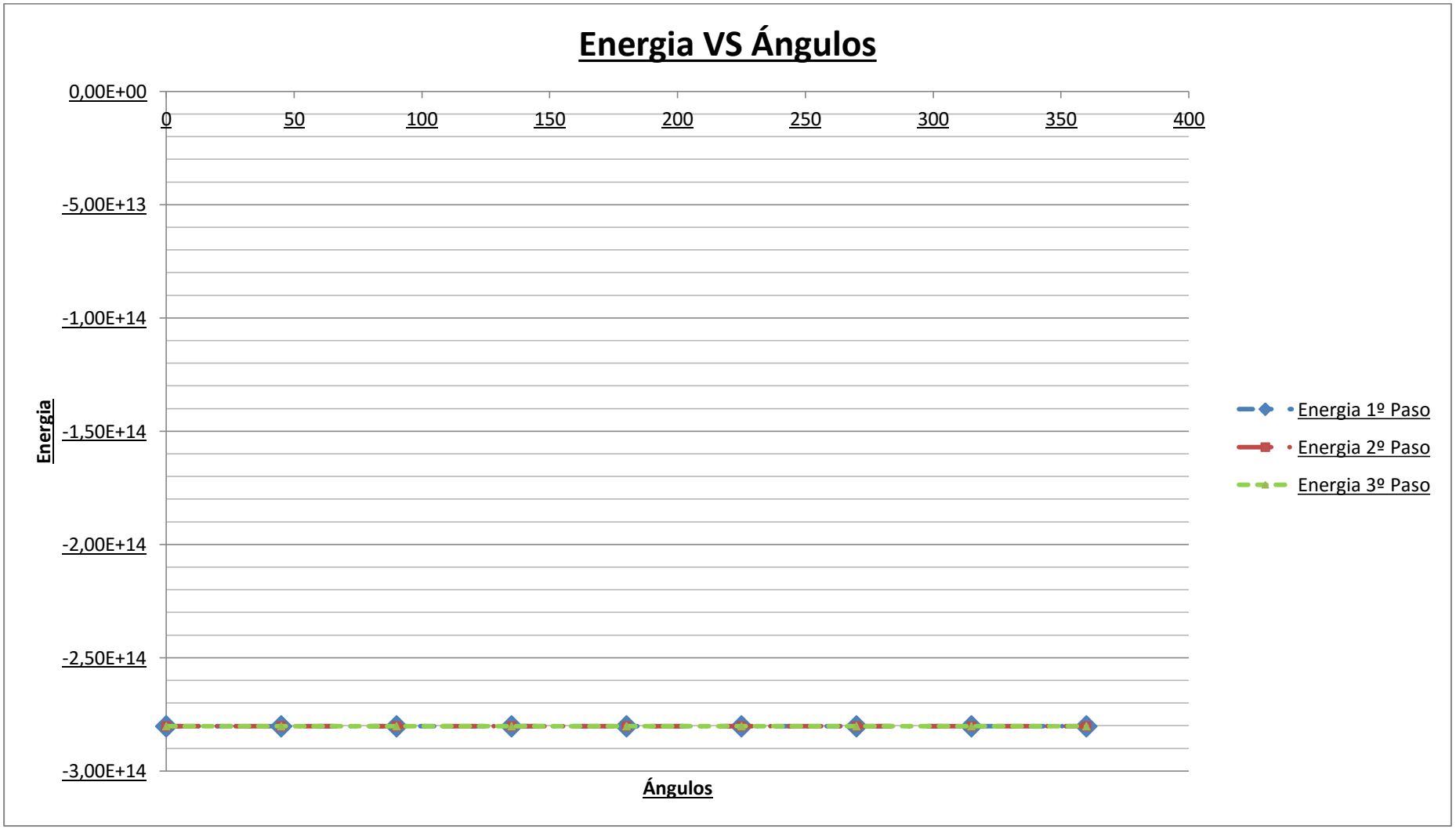
## B.7: Integral de energía del sistema (Algoritmo 2-GR)

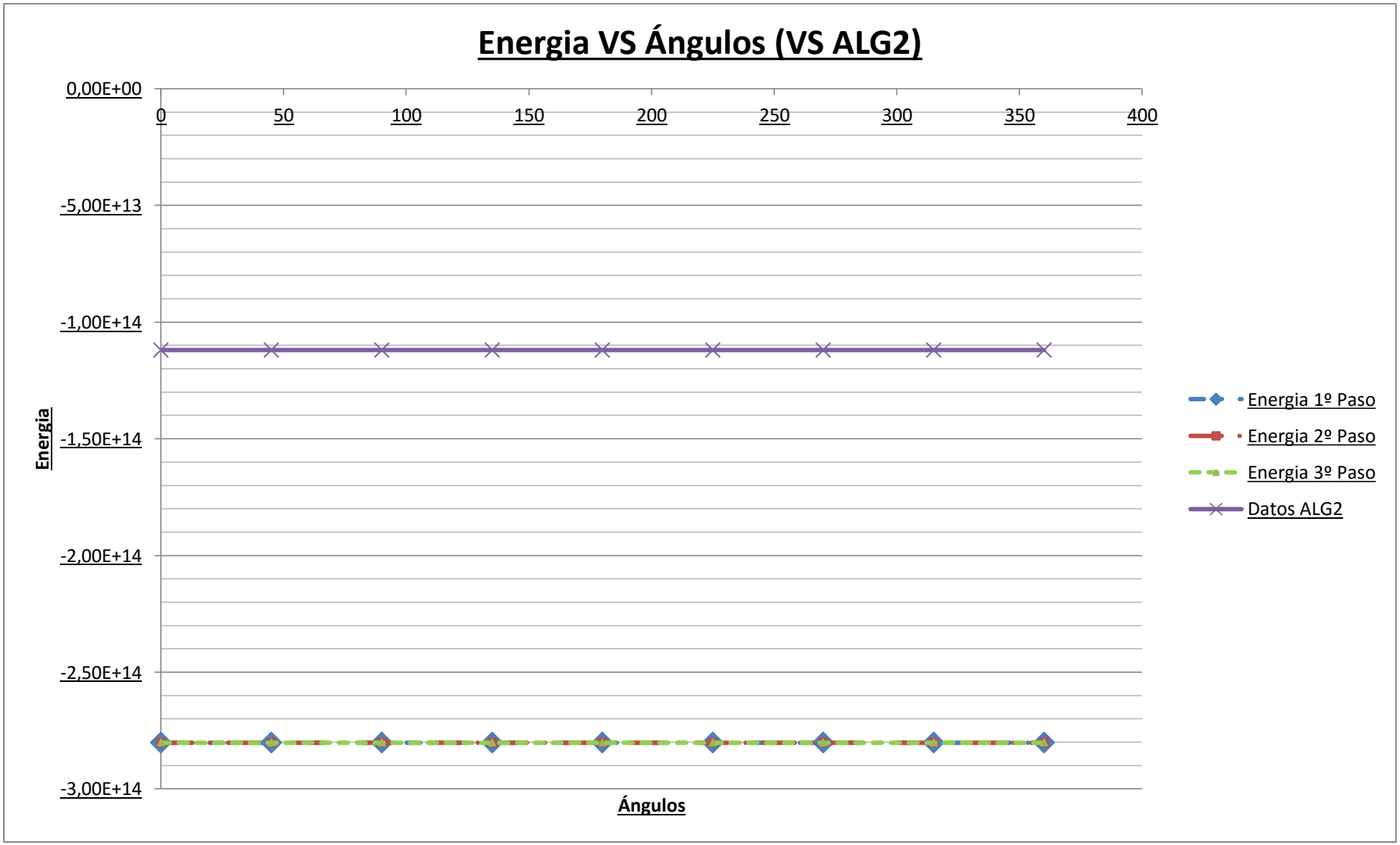
La resolución es similar al ítem B.3 para el cálculo de la energía pero como este algoritmo es otro, se tienen diferentes resultados, aunque igualmente se sigue poseyendo orden cuatro, por lo cual deberá realizarse una diferenciación numérica acorde al orden especificado, siendo la resolución una posible aproximación por polinomios de Taylor o por medio del método de Richardson.

| **PASOS (N)** | $E_n$ |
|---|---|
| 1,00E+02 | -2.80236e+009 |
| 1,00E+03 | -2.80236e+009 |
| 1,00E+04 | -2.80236e+009 |
| 1,00E+05 | -2.80236e+009 |
| 1,00E+06 | -2.80236e+009 |
| **1,00E+07** | **-2.80236e+009** |
| 1,00E+08 | -2.80236e+009 |
| 1,00E+09 | -2.80236e+009 |
| 1,00E+10 | -2.80236e+009 |
| 1,00E+11 | -2.80236e+009 |

Se ve que se converge a un resultado. Los valores presentes en el sistema solar los resultados cuantitativos de la teoría einsteniana son de alguna manera cercanos a la teoría newtoniana (e+009), pero diferentes en el sentido de su valor claramente por la aparición del nuevo término de Einstein, como se mencionó en la sección B.3. El problema es conservativo con el uso del ALG2-GR.

En los gráficos se puede ver la diferencia entre pequeños pasos, y la diferencia con el algoritmo uno sin la teoría relativista.


Energia VS Ángulos
0,00E+00
-5,00E+13
-1,00E+14
-1,50E+14
-2,00E+14
-2,50E+14
-3,00E+14
Energia
0
50
100
150
200
250
300
350
400
Ángulos
Energia 1º Paso
Energia 2º Paso
Energia 3º Paso


Energia VS Ángulos (VS ALG2)
0,00E+00
-5,00E+13
-1,00E+14
-1,50E+14
-2,00E+14
-2,50E+14
-3,00E+14
Energia
0
50
100
150
200
250
300
350
400
Ángulos
Energia 1º Paso
Energia 2º Paso
Energia 3º Paso
Datos ALG2

## B.8: Análisis de aptitud del algoritmo 2-GR

Si bien el nuevo término de Einstein se utiliza para validar y verificar teoremas relativistas particulares, el pequeño término que aparece influye en este algoritmo, pero por el orden que posee, el algoritmo es consistente y devuelve soluciones concretas para cualquier paso. Si bien no sea posiblemente suficiente para resolver el problema completamente, es definitivamente la mejor aproximación que se puede tener con los algoritmos dados y mientras mayor sea el paso, más precisa será la solución, aunque también posee una gran propagación de errores por la metodología compleja.

## Conclusión

El análisis numérico se ocupa de estudiar algoritmos para resolver problemas de la matemática continua. Dado que estos algoritmos son una aproximación al problema matemático, resulta evidente que los resultados obtenidos estarán afectados por algún tipo de error, siendo para este caso el problema matemático el planteado por Newton con la corrección relativista de Einstein, y el problema numérico el resuelto por los algoritmos propuestos. Aquí ya se puede apreciar el primer error cometido con diferencia del TP1, la idea de considerar el modelo propuesto por Newton el correcto, siendo el error no considerar en su modelo el término relativista adicional para la resolución de paradigmas especiales. Luego tanto como en el TP1 fue mencionado, existe la incidencia de los *errores de redondeo* como también de *truncamiento* y *entrada*. No vale la pena extenderse tanto en este tema ya que fue analizado en el TP1, aunque si vale la pena mencionar que estos algoritmos contienen tanto error de redondeo, debido a la precisión finita de la máquina, como error de truncamiento, debido a la naturaleza del problema matemático y errores de entrada por los datos de las variables propuestas.

Lo que se buscó ver en la sección A es un análisis más profundo del problema propuesto por Newton, teniendo siempre en cuenta el orden propuesto para la realización de la cuadratura y diferenciación numérica para ser consistente durante todo el proceso respecto al orden y convergencia de las metodologías de Euler y RK4. Como este es un PVI la idea de que el sistema converja habla de un problema bien condicionado para llegar a la conclusión de ver que el algoritmo de RK4 es una mucha mejor aproximación y un método más estable en cuanto al análisis de una órbita y su cumplimiento con las leyes de Kepler como también la conservación de energía producida por la relación entre la reducción del radio y el aumento de velocidad.

Luego se puede ver el análisis de estabilidad desde otro punto de vista en la sección B. Debido a la presencia del término nuevo llamado precesión, se buscó validar los resultados teóricos como también ver los resultados y analizar a partir de aquí si el problema y el algoritmo eran lo suficientemente útiles para poder resolverlos. Generalmente el método de Euler explícito suele ser incondicionalmente estable lo cual aquí se puede ver por los cálculos de la precesión, como también tiene la desventaja de ser explícito y de orden uno, por lo cual después de todo lo analizado, para un problema tan complejo probablemente no sea lo mejor el uso de este algoritmo. Luego a través del método de RK4, se obtiene una mucha mejor aproximación de

las soluciones con la el hecho de ser contraproducente el tener un tiempo de computo mucho mayor como también requerimientos del sistema para la realización de cálculos (si bien esto no está explicitado en números en este TP, en el anterior se vio claramente la diferencia y mientras se realizaban los nuevos procedimientos GR sucedió similarmente).

Las principales diferencias y similitudes entre los algoritmos uno y dos son:

- *Velocidad de cómputo*: la del algoritmo dos GR es mucho mayor que la del algoritmo uno GR
- *Orden*: el orden del algoritmo 2-GR es mayor que la del algoritmo 1-GR debido que los métodos utilizados son los mismos pero con la variación del término de Einstein, y los resultados reflejan la convergencia y resultados con una mejor o peor aproximación al problema.
- *Precesión*: con el algoritmo 2-GR se llega a una solución concreta, si bien no es la esperada, si se utilizaran herramientas para mejorar el cálculo (como interpolación no realizada) se llegaría a la solución buscada o una gran aproximación a ella, mientras que para el algoritmo 1-GR la solución será variable.
- Tanto para la primera como la segunda parte, el sistema poseería energía del tipo conservativa para el algoritmo 2 y 2-GR y no conservativa para el algoritmo 1 y 1-GR por su orden, convergencia y resultados, especialmente en pasos pequeños
- El segundo algoritmo-GR está mejor condicionado que el primero, debido a que la solución cuando los pasos son pequeños se parece o es idéntica a la solución con pasos grandes, si en cambio para el primero hay una cierta diferencia notable tanto durante la trayectoria como el punto en que finaliza. Como también posee un mayor orden y el método es explicito se puede imponer una condición de estabilidad para este caso, permitiendo una resolución correcta a partir de este análisis. Como tambien el segundo algoritmo posee, vale aclarar, un mayor requerimiento del CPU para el cálculo de los resultados.

## Anexo I: Código fuente

```
/*
**     Universidad de Buenos Aires
**     Facultad de Ingeniería
**     75.12 Análisis Numérico I
**     Trabajo Práctico I
**
**     Merlo Leiva Nahuel -   Fabrizio Luis Cozza
**     Padrón 92115       -   Padrón 97402
*/

#include "stdafx.h"
#include "D2D1Graph.h"

struct alg_data
{
	REAL u_n_min;
	REAL u_n_min_step;
	REAL u_n_max;
	REAL u_n_max_step;
	REAL u_n_4_4_N; // 2 PI
};

struct alg_params
{
	REAL np;
	REAL lambda;
	REAL steps;
	REAL alg;
	REAL dp;
	REAL print;
	REAL scale;
	REAL width;
	REAL height;
	REAL u_0;
	REAL v_0;
	REAL k;
	REAL alpha;
	REAL beta;
	REAL epsilon;
	CD2D1Graph* pGraph;
};

template<typename T>
T computeDDR(T u_n, T semiMinorAxis, T epsilon)
{
	T q = semiMinorAxis * (1 - epsilon);
	T DRR = (pow(u_n, -1) / q) - 1;
	return DRR;
}

template<typename T>
T disturbeSteps(T m1,T m2,T M, T epsilon,T semiMinorAxis) //pasar parametros
ByRef
{
	m1 = lambda * (1.9891e+30) + experimentalDisturbance; // kg
	m2 = lambda * (3.301e+23) + experimentalDisturbance; // kg
	M = m1 + m2; // kg
	semiMinorAxis = pow(lambda, 2) * (5.791e+10) + experimentalDisturbance; //
m
```

```
	epsilon = pow(lambda, -1) * 0.2056 + experimentalDisturbance; // -
	if (epsilon <= 0 || epsilon >= 1)
	{
		printf("Epsilon=%e out of range!\n", epsilon);
		return 0;
	}
}

template<typename T>
T computeCp(T disturbedResult)
{
	T originalResult = 10; //poner el resultado original aca
	T relativeError = (originalResult - disturbedResult) / originalResult;
	T C_p = relativeError * (1 / experimentalDisturbance); //numero de
condicion del problema
	return C_p;
}

template<typename T>
T computeStability(T resultInOnePrecision, T resultInAnotherPrecision)
{
	//utilizar la precision y los resultados de algun paso en dos precisiones
diferentes y calcular
	machineUnitErrorInAnotherPrecision = 8;
	machineUnitErrorInOnePrecision = 16;
	totalMachineError = machineUnitErrorInAnotherPrecision -
machineUnitErrorInOnePrecision;
	stability = (resultInOnePrecision - resultInAnotherPrecision) /
(resultInOnePrecision * totalMachineError);
	return stability;
}

template<typename T>
T computeCa(T stability, T C_p)
{
	Ca = stability / C_p;
}

#define INIT_DATA(data, u_n) \
data.u_n_min = u_0; \
data.u_n_max = u_0;  \

#define FILL_DATA_MIN_MAX(data, u_n, step) \
if (u_n < data.u_n_min) \
{ \
	data.u_n_min = u_n;  \
	data.u_n_min_step = step; \
} \
if (u_n > data.u_n_max) \
{ \
	data.u_n_max = u_n; \
	data.u_n_max_step = step; \
}

#define FILL_DATA_U_N_2_PI(data, u_n) \
data.u_n_4_4_N = u_n; \

#define PRINT(pGraph, n, k, u_n)   \
if (pGraph)    \
{ \
	T theta_n = n * k; \
	T radius_n = 1 / u_n; \
```

```
		pGraph->DrawPointPolar(radius_n, theta_n); \
}

template<typename T>
alg_data algorithm1(T u_0, T v_0, T k, T alpha, double steps, CD2D1Graph* pGraph
= nullptr)
{
		T u_n = u_0;
		T v_n = v_0;
		T k_alpha = k * alpha;

		double theta_n;

		alg_data data;
		INIT_DATA(data, u_n);

		for (double n = 0; n < steps; n++)
		{
			FILL_DATA_MIN_MAX(data, u_n, n);
			PRINT(pGraph, n, k, u_n);

			T u_n_1 = u_n + (k * v_n);
			T v_n_1 = v_n - (k * u_n) + k_alpha;

			u_n = u_n_1;
			v_n = v_n_1;
		}

		PRINT(pGraph, steps, k, u_n);
		FILL_DATA_U_N_2_PI(data, u_n);
		return data;
}

template<typename T>
alg_data algorithm2(T u_0, T v_0, T k, T alpha, double steps, CD2D1Graph* pGraph
= nullptr)
{
		T u_n = u_0;
		T v_n = v_0;

		alg_data data;
		INIT_DATA(data, u_n);

		for (double n = 0; n < steps; n++)
		{
			FILL_DATA_MIN_MAX(data, u_n, n);
			PRINT(pGraph, n, k, u_n);

			T w_1 = u_n + (k * v_n) / 2;
			T z_1 = v_n + (k * (alpha - u_n)) / 2;
			T w_2 = u_n + (k * z_1) / 2;
			T z_2 = v_n + (k * (alpha - w_1)) / 2;
			T w_3 = u_n + (k * z_2);
			T z_3 = v_n + (k * (alpha - w_2));

			T u_n_1 = u_n + (k * (v_n + ((2 * z_1) + (2 * z_2) + z_3))) / 6;
			T v_n_1 = v_n + (k * ((6 * alpha) - u_n - (2 * w_1) - (2 * w_2) -
w_3)) / 6;

			u_n = u_n_1;
			v_n = v_n_1;
		}
```

```
	PRINT(pGraph, steps, k, u_n);
	FILL_DATA_U_N_2_PI(data, u_n);
	return data;
}

template<typename T>
alg_data algorithm1GR(T u_0, T v_0, T k, T alpha, T beta, double steps,
CD2D1Graph* pGraph = nullptr)
{
	alg_data data;
	T u_n = u_0;
	T v_n = v_0;
	T k_alpha = k * alpha;
	T k_beta = k * beta;
	for (double n = 0; n < steps; n++)
	{
		if (pGraph)
		{
			T theta_n = n * k;
			T radius_n = 1 / u_n;
			pGraph->DrawPointPolar(radius_n, theta_n);
		}

		T u_n_1 = u_n + (k * v_n);
		T v_n_1 = v_n - (k * u_n) + k_alpha + k_beta * (pow(u_n, 2)); //
u_n^2

		u_n = u_n_1;
		v_n = v_n_1;
	}

	if (pGraph)
	{
		T theta_n = steps * k;
		T radius_n = 1 / u_n;
		pGraph->DrawPointPolar(radius_n, theta_n);
	}
	data.u_n_4_4_N = u_n;
	return data;
}

template<typename T>
alg_data algorithm2GR(T u_0, T v_0, T k, T alpha, T beta, double steps,
CD2D1Graph* pGraph = nullptr)
{
	alg_data data;
	T u_n = u_0;
	T v_n = v_0;
	for (double n = 0; n < steps; n++)
	{
		if (pGraph)
		{
			T theta_n = n * k;
			T radius_n = 1 / u_n;
			pGraph->DrawPointPolar(radius_n, theta_n);
		}

		T w_1 = u_n + (k * v_n) / 2;
		T z_1 = v_n + (k * (alpha + beta * pow(u_n, 2) - u_n)) / 2;
		T w_2 = u_n + (k * z_1) / 2;
		T z_2 = v_n + (k * (alpha + beta * pow(w_1, 2) - w_1)) / 2;
```

```
            T w_3 = u_n + (k * z_2);
            T z_3 = v_n + (k * (alpha + beta * pow(w_2, 2) - w_2));

            T u_n_1 = u_n + (k * (v_n + ((2 * z_1) + (2 * z_2) + z_3))) / 6;
            T v_n_1 = v_n + (k * (
                  (alpha + beta * pow(u_n, 2) - u_n) +
                  2 * (alpha + beta * pow(w_1, 2) - w_1) +
                  2 * (alpha + beta * pow(w_2, 2) - w_2) +
                  (alpha + beta * pow(w_3, 2) - w_3)
                  )) / 6;

            u_n = u_n_1;
            v_n = v_n_1;
      }

      if (pGraph)
      {
            T theta_n = steps * k;
            T radius_n = 1 / u_n;
            pGraph->DrawPointPolar(radius_n, theta_n);
      }
      data.u_n_4_4_N = u_n;
      return data;
}

template<typename T>
alg_data runAlgorithm(alg_params params)
{
      if (params.alg == 1)
      {
            return algorithm1<T>(params.u_0, params.v_0, params.k,
params.alpha, params.steps, params.pGraph);
      }
      else if (params.alg == 2)
      {
            return algorithm2<T>(params.u_0, params.v_0, params.k,
params.alpha, params.steps, params.pGraph);
      }
      else if (params.alg == 3)
      {
            return algorithm1GR<T>(params.u_0, params.v_0, params.k,
params.alpha, params.beta, params.steps, params.pGraph);
      }
      else if (params.alg == 4)
      {
            return algorithm2GR<T>(params.u_0, params.v_0, params.k,
params.alpha, params.beta, params.steps, params.pGraph);
      }
      return alg_data();
}

double secondDerivateValue(){
      return 0.5; //value input
}

double computeAreaError(double inicialIntervalPoint, double finalIntervalPoint,
double steps){
      double error, numerator, denominator;

      numerator = pow(finalIntervalPoint - inicialIntervalPoint, 3);
      denominator = 24 * (pow(steps, 2));
```

```
	error = (numerator / denominator) * secondDerivateValue();

	return error;
}

double computePeriodError(double areaError, double area, double
specificAngularMomentumSquared, double specificAngularMomentumSquaredError){
	double error, firstTerm, secondTerm;

	firstTerm = (2 / sqrt(specificAngularMomentumSquared)) * areaError;
	secondTerm = ((2 * area) * (1 / specificAngularMomentumSquared)) *
specificAngularMomentumSquaredError;

	error = firstTerm + secondTerm ;
	return error;
}

double periodCalculation(double area, int h){
	double period;
	period = ((area * 2) / h) * (-1);
	return period;
}

double rectangularIntegralGeneral(double inicialIntervalPoint, double
finalIntervalPoint, double specificAngularMomentumSquared, double semiMinorAxis,
double semiMayorAxis, int steps) {

	double area, x_i, h, areaError;
	double specificAngularMomentumSquaredError = 0.5e-16; //double
	h = (finalIntervalPoint - inicialIntervalPoint) / steps; // Computar ancho
del intervalo arriba
	area = 0.0;                          // Clear running area

	for (int i = 1; i <= steps - 1; i++)
	{
		x_i = inicialIntervalPoint + steps*h;
		area = area + (h * sqrt(((1 - (pow(x_i, 2) / pow(semiMayorAxis,
2))) * pow(semiMinorAxis, 2)))); //f(x) = (1 - (x^2 / a^2) * b^2)^1/2
		areaError = computeAreaError(inicialIntervalPoint,
finalIntervalPoint, steps);
	}

	// Guardar resultados
	std::fstream statsOutput;
	statsOutput.open("mustafar_solve_area_period.csv", std::ios_base::app);
	statsOutput << area << ";";
	statsOutput << areaError << ";";
	statsOutput << periodCalculation(area,
sqrt(specificAngularMomentumSquared)) << ";";
	statsOutput << computePeriodError(areaError, area,
specificAngularMomentumSquared, specificAngularMomentumSquaredError) << ";";
	statsOutput << "\n";
	statsOutput.close();

	return 0;
}

double computeKeplerThirdLawError(double period, double semiMajorAxis, double
periodError, double deltaSemiMajorAxis){
	double firstTerm, secondTerm, error;
```

```
	firstTerm = ((2 * period) / (pow(semiMajorAxis, 3))) * periodError; //
(2*T / b^3) * deltaT
	secondTerm = ((3 * (pow(period, 2))) / ((pow(semiMajorAxis, 4)))) *
deltaSemiMajorAxis; // (3*T^2/b^4) * deltaAxis

	error = firstTerm + secondTerm;

	return error;
}


double computeKeplerThirdLaw(double period, double semiMajorAxis){
	double quotient, numerator, denominator;

	numerator = pow(period, 2); //periodo^2
	denominator = pow(semiMajorAxis, 3);//b^3

	quotient = numerator / denominator; // cociente

	return quotient;
}

double fourthDerivateValue(){
	return 0.5; //value input
}

double computeSimpsonsRuleError(double h){
	double error, numerator, denominator;

	numerator = pow(h, 5);
	denominator = 90;

	error = (numerator / denominator) * fourthDerivateValue();
	return error;
}

double f(double x) //function f(x)
{
	return(x);
}

double simpsonsRule(double inicialIntervalPoint, double finalIntervalPoint, int
steps)
{
	double s1 = 0, s2 = 0; //Clear running area
	double area, h;
	h = (finalIntervalPoint - inicialIntervalPoint) / steps; // Computar ancho
del intervalo arriba
	if (steps % 2 == 0)
	{
		for (int i = 1; i <= steps - 1; i++)
		{
			if (i % 2 == 0)
			{
				s1 = s1 + f(inicialIntervalPoint + i*h);
			}
			else
			{
				s2 = s2 + f(inicialIntervalPoint + i*h);
			}
		}
```

```
		area = h / 3 * (f(inicialIntervalPoint) + f(finalIntervalPoint) + 4
* s2 + 2 * s1);
		printf("The value is = %f", area);
	}
	else
	{
		printf("The rule is not appliciable");
	}
	_getch();
	return area;
}

double lagrangeInterpolation(double x_1, double x_2, double x_3, double y_1,
double y_2, double y_3){
	{
		double x[100], y[100], quantityOfPoints, pointToInterpolate,
interpolatedPoint = 0, s = 1, t = 1;
		int i, j, cutCondition = 1;
		//Respective values of the variables x and y
		quantityOfPoints = 3;
		x[0] = x_1;
		x[1] = x_2;
		x[2] = x_3;
		y[0] = y_1;
		y[1] = y_2;
		y[2] = y_3;
		printf("\n\n Table entered is as follows :\n\n");
		for (i = 0; i<quantityOfPoints; i++)
		{
			printf("%0.3f\t%0.3f", x[i], y[i]);
			printf("\n");
		}

		while (cutCondition == 1)
		{
			//Value of the x to find the respective value of y
			pointToInterpolate = 3.5;
			for (i = 0; i<quantityOfPoints; i++)
			{
				s = 1;
				t = 1;
				for (j = 0; j<quantityOfPoints; j++)
				{
					if (j != i)
					{
						s = s*(pointToInterpolate - x[j]);
						t = t*(x[i] - x[j]);
					}
				}
				interpolatedPoint = interpolatedPoint + ((s /
t)*y[i]);
			}
			printf("\n\n The respective value of the variable y is: %f",
interpolatedPoint);
			printf("\n\n Do you want to continue?\n\n Press 1 to continue
and any other key to exit");
			scanf_s("%cutCondition", &cutCondition);
		}
		return interpolatedPoint;
	}
}
```

```
void solve_A_1(alg_params params, alg_data data, double specificMomentum)
{
	// Cálculo del semi eje mayor como: a =  (radio perihelio + radio afelio)
/ 2
	REAL semiMajorAxis = ((1 / data.u_n_min) + (1 / data.u_n_max)) / 2;

	// Propagación de error
	REAL deltaSemiMajorAxis = abs(1 / (2 * (pow(data.u_n_min, 2)))) *
REAL_EPSILON +
		abs(1 / (2 * (pow(data.u_n_max, 2)))) * REAL_EPSILON;

	// Cálculo del semi eje menor como: b = a * square_root(1 - e^2)
	REAL semiMinorAxis = semiMajorAxis * sqrt((1 - pow(params.epsilon, 2)));

	// Propagación de error
	REAL deltaSemiMinorAxis = abs(sqrt((1 - pow(params.epsilon, 2)))) *
deltaSemiMajorAxis +
		abs(semiMajorAxis * pow(2 * sqrt((1 - pow(params.epsilon, 2))), -1)
* 2 * params.epsilon) * REAL_EPSILON;

	std::fstream statsOutput;
	statsOutput.open("mustafar_solve_A_1.csv", std::ios_base::app);
	statsOutput << params.alg << ";";
	statsOutput << params.steps << ";";
	statsOutput << semiMajorAxis << ";";
	statsOutput << deltaSemiMajorAxis << ";";
	statsOutput << semiMinorAxis << ";";
	statsOutput << deltaSemiMinorAxis << ";";
	statsOutput << "\n";
	statsOutput.close();
	if (params.pGraph)
	{
	REAL min_theta_n = data.u_n_min_step * params.k;
	REAL min_radius_n = 1 / data.u_n_min;
	params.pGraph->DrawPointPolar(min_radius_n, min_theta_n, 3,
D2D1::ColorF(1, 0, 0, 1));

	REAL max_theta_n = data.u_n_max_step * params.k;
	REAL max_radius_n = 1 / data.u_n_max;
	params.pGraph->DrawPointPolar(max_radius_n, max_theta_n, 3,
D2D1::ColorF(1, 0, 0, 1));
	}

}

int main(int argc, char* argv[])
{
	alg_params params;
	params.np = 92115;
	params.lambda = params.np / LAMBDA_DEN;
	params.steps = 0;
	params.alg = 0;
	params.dp = 0;
	params.print = 0;
	params.scale = 0;
	params.width = 0;
	params.height = 0;
	params.pGraph = nullptr;
	for (int argIndex = 1; argIndex < argc; argIndex++)
	{
		if (!strcmp(argv[argIndex], "-alg") &&
			(argIndex + 1) < argc)
```

```
		{
			params.alg = atol(argv[argIndex + 1]);
			argIndex++;
		}
		else if (!strcmp(argv[argIndex], "-steps") &&
			(argIndex + 1) < argc)
		{
			params.steps = atof(argv[argIndex + 1]);
			argIndex++;
		}
		else if (!strcmp(argv[argIndex], "-dp") &&
			(argIndex + 1) < argc)
		{
			params.dp = atol(argv[argIndex + 1]);
			argIndex++;
		}
		else if (!strcmp(argv[argIndex], "-print") &&
			(argIndex + 1) < argc)
		{
			params.print = atol(argv[argIndex + 1]);
			argIndex++;
		}
		else if (!strcmp(argv[argIndex], "-width") &&
			(argIndex + 1) < argc)
		{
			params.width = atof(argv[argIndex + 1]);
			argIndex++;
		}
		else if (!strcmp(argv[argIndex], "-height") &&
			(argIndex + 1) < argc)
		{
			params.height = atof(argv[argIndex + 1]);
			argIndex++;
		}
		else if (!strcmp(argv[argIndex], "-scale") &&
			(argIndex + 1) < argc)
		{
			params.scale = atof(argv[argIndex + 1]);
			argIndex++;
		}
		else if (!strcmp(argv[argIndex], "-lambda") &&
			(argIndex + 1) < argc)
		{
			params.lambda = atof(argv[argIndex + 1]);
			argIndex++;
		}
	}

	if (params.steps == 0 ||
		params.alg < 1
		|| params.alg > 4)
	{
		printf("Wrong params!\n");
		printf("-alg \t Algorithm number {1: Euler, 2: RK4, 3: Euler GR, 4:
RK4 GR}.\n");
		printf("-steps \t Algorithm step count.\n");
		printf("-dp \t Use double precision. {0, 1}\n");
		printf("-print \t Print results. {0, 1}\n");
		printf("-scale \t Print scale.\n");
		printf("-width \t Print width.\n");
		printf("-height \t Print height.\n");
		printf("-lambda \t Override lambda.\n");
```

```
			return 0;
		}

		char runDatabaseFileName[255];
		memset(runDatabaseFileName, 0, sizeof(runDatabaseFileName) /
sizeof(runDatabaseFileName[0]));
		sprintf_s(runDatabaseFileName, "alg_%ld_steps_%.0f_dp_%ld_db.csv",
params.alg, params.steps, params.dp);

		printf("Creating output files...\n");

		if (params.print)
		{
			params.pGraph = new CD2D1Graph(params.width, params.height,
params.scale);
			params.pGraph->BeginDraw();
		}

		REAL G = 6.673e-11; // N m^2 / kg ^2
		REAL m1 = params.lambda * (1.9891e+30); // kg
		REAL m2 = params.lambda * (3.301e+23); // kg
		REAL M = m1 + m2; // kg
		params.epsilon = pow(params.lambda, -1) * 0.2056; // -
		if (params.epsilon <= 0 || params.epsilon >= 1)
		{
			printf("Epsilon=%e out of range!\n", params.epsilon);
			return 0;
		}
		REAL semiMinorAxis = pow(params.lambda, 2) * (5.791e+10); // m
		REAL mu = G * M; // m^3 / s^2
		REAL specificAngularMomentumSquared = semiMinorAxis * mu * (1 -
pow(params.epsilon, 2)); // m^4 / s^2
		params.u_0 = pow(semiMinorAxis * (1 - params.epsilon), -1);
		params.v_0 = 0;
		params.k = (2 * M_PI) / params.steps;
		REAL lightningSpeed = 3e+8;
		REAL lightningSpeedSquared = pow(lightningSpeed, 2); // c^2

		printf("ALGORITHM=%d\n", params.alg);
		printf("DOUBLE PRECISION=%d\n", params.dp);
		printf("NP=%e\n", params.np);
		printf("LAMBDA=%e\n", params.lambda);
		printf("M1=%e\n", m1);
		printf("M2=%e\n", m2);
		printf("MU=%e\n", mu);
		printf("EPSILON=%e\n", params.epsilon);
		printf("SEMI MINOR AXIS=%e\n", semiMinorAxis);
		printf("SPECIFIC ANGULAR MOMENTUM SQUARED=%e\n",
specificAngularMomentumSquared);
		printf("LIGHTNING SPEED SQUARED=%e\n", lightningSpeedSquared);
		printf("U_0=%e\n", params.u_0);
		printf("V_0=%e\n", params.v_0);
		printf("STEP COUNT=%e\n", params.steps);
		printf("RADIAN STEP=%e\n", params.k);

		params.alpha = mu * (pow(specificAngularMomentumSquared, -1));
		params.beta = 3 * mu * (pow(lightningSpeedSquared, -1));

		alg_data data;

		printf("Start!\n");
```

```
	auto startTimer = std::chrono::high_resolution_clock::now(); //inicio
calculo del tiempo en nanosegundos
	if (params.dp)
	{
		data = runAlgorithm<double>(params);
	}
	else
	{
		data = runAlgorithm<float>(params);
	}

	auto finishTimer = std::chrono::high_resolution_clock::now(); //inicio
calculo del tiempo en nanosegundos
	auto tiempoDeCorrida =
std::chrono::duration_cast<std::chrono::nanoseconds>(finishTimer -
startTimer).count();
	printf("End!\n");

	if (params.pGraph)
	{

		params.pGraph->EndDraw();
		params.pGraph->Present();

		std::wstring pngFileName = _bstr_t(runDatabaseFileName);
		pngFileName.append(L".png");
		params.pGraph->SavePNG(pngFileName.c_str());
	}

	printf("Done!\n");
	return 0;
}
```

# Anexo II: Salida

## *Salida A.1 – A.5*

100;6.08083e+010;864522;5.95688e+010;846900;

1000;6.06744e+010;851339;5.94376e+010;833986;

10000;6.06648e+010;850238;5.94282e+010;832907;

100000;6.06639e+010;850130;5.94273e+010;832801;

1e+006;6.06638e+010;850119;5.94272e+010;832790;

1e+007;6.06638e+010;850118;5.94272e+010;832789;

1e+008;6.06638e+010;850118;5.94272e+010;832789;

1e+009;6.06638e+010;850118;5.94272e+010;832789;

1e+010;6.06637e+010;850118;5.94272e+010;832789;

1e+011;6.06638e+010;850118;5.94272e+010;832789;

100;6.06638e+010;850118;5.94272e+010;832789;

1000;6.06638e+010;850118;5.94272e+010;832789;

10000;6.06638e+010;850118;5.94272e+010;832789;

100000;6.06638e+010;850118;5.94272e+010;832789;

1e+006;6.06638e+010;850118;5.94272e+010;832789;

1e+007;6.06638e+010;850118;5.94272e+010;832789;

1e+008;6.06638e+010;850118;5.94272e+010;832789;

1e+009;6.06638e+010;850118;5.94272e+010;832789;

1e+010;6.06638e+010;850118;5.94272e+010;832789;

1e+011;6.06638e+010;850118;5.94272e+010;832789;

### *Salida A.2 – A.6*

100;1.13797e+022;3.23575e+017;1.05982e+013;1.22033e+010;

1000;1.13297e+022;3.17939e+017;1.05516e+013;1.19908e+010;

10000;1.13261e+022;3.17478e+017;1.05482e+013;1.19734e+010;

100000;1.13258e+022;3.17432e+017;1.05480e+013;1.19717e+010;

1e+006;1.13257e+022;3.17428e+017;1.05479e+013;1.19715e+010;

1e+007;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1e+008;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1e+009;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1e+010;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1e+011;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

100;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1000;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

10000;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

100000;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1e+006;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1e+007;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1e+008;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1e+009;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1e+010;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

1e+011;1.13257e+022;3.17427e+017;1.05479e+013;1.19715e+010;

### *Salida A.3 – A.7*

100;4.99545e-007;1.17171e-009;

1000;4.98445e-007;1.15385e-009;

10000;4.98366e-007;1.15235e-009;

100000;4.98358e-007;1.15221e-009;

1e+006;4.98358e-007;1.15219e-009;

1e+007;4.98358e-007;1.15219e-009;

1e+008;4.98357e-007;1.15219e-009;

1e+009;4.98357e-007;1.15219e-009;

1e+010;4.98357e-007;1.15219e-009;

1e+011;4.98358e-007;1.15219e-009;

100;4.98357e-007;1.15219e-009;

1000;4.98357e-007;1.15219e-009;

10000;4.98357e-007;1.15219e-009;

100000;4.98357e-007;1.15219e-009;

1e+006;4.98357e-007;1.15219e-009;

1e+007;4.98357e-007;1.15219e-009;

1e+008;4.98357e-007;1.15219e-009;

1e+009;4.98357e-007;1.15219e-009;

1e+010;4.98357e-007;1.15219e-009;

1e+011;4.98357e-007;1.15219e-009;

### *Salida A.4 – A.8*

100;-1.11705e+009;

1000;-1.11952e+009;

10000;-1.11969e+009;

100000;-1.11971e+009;

1e+006;-1.11971e+009;

1e+007;-1.11971e+009;

1e+008;-1.11971e+009;

1e+009;-1.11971e+009;

1e+010;-1.11971e+009;

1e+011;-1.11971e+009;

100;-1.11971e+009;

1000;-1.11971e+009;

10000;-1.11971e+009;

100000;-1.11971e+009;

1e+006;-1.11971e+009;

1e+007;-1.11971e+009;

1e+008;-1.11971e+009;

1e+009;-1.11971e+009;

1e+010;-1.11971e+009;

1e+011;-1.11971e+009;

### *Salida B.1 – B.5*

100;49.9853;

1000;50.1129;

10000;50.1224;

100000;50.1233;

1e+006;50.1234;

1e+007;50.1234;

1e+008;50.1234;

1e+009;50.1234;

1e+010;50.1234;

1e+011;50.1234;

100;50.1234;

1000;50.1234;

10000;50.1234;

100000;50.1234;

1e+006;50.1234;

1e+007;50.1234;

1e+008;50.1234;

1e+009;50.1234;

1e+010;50.1234;

1e+011;50.1234;

### *Salida B.2 – B.6*

100;48.7584;

1000;48.866;

10000;48.8738;

100000;48.8745;

1e+006;48.8746;

1e+007;48.8746;

1e+008;48.8746;

1e+009;48.8746;

1e+010;48.8746;

1e+011;48.8746;

100;48.8746;

1000;48.8746;

10000;48.8746;

100000;48.8746;

1e+006;48.8746;

1e+007;48.8746;

1e+008;48.8746;

1e+009;48.8746;

1e+010;48.8746;

1e+011;48.8746;

### *Salida B.3 – B.7*

100;-2.90188e+009;

1000;-2.81169e+009;

10000;-2.80329e+009;

100000;-2.80245e+009;

1e+006;-2.80237e+009;

1e+007;-2.80236e+009;

1e+008;-2.80236e+009;

1e+009;-2.80236e+009;

1e+010;-2.80236e+009;

1e+011;-2.80236e+009;

100;-2.80236e+009;

1000;-2.80236e+009;

10000;-2.80236e+009;

100000;-2.80236e+009;

1e+006;-2.80236e+009;

1e+007;-2.80236e+009;

1e+008;-2.80236e+009;

1e+009;-2.80236e+009;

1e+010;-2.80236e+009;

1e+011;-2.80236e+009;