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



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DATOS DEL TRABAJO PRÁCTICO** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | |  | |  | |  | |  | | |  | |  | | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| 2 | | | | | | | 2 | | 0 | | | 1 | | 6 | | | 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 | | | | | | | M | | e | | | r | | l | | | o | |  | | L | | e | | i | | v | | a | |  | | | N | | a | | | h | | u | | e | | l | |  | |  | |  | | 9 | | 2 | | 1 | | 1 | | 5 | |
| APELLIDO Y NOMBRE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | PADRÓN | | | | | | | | | |
| C | | o | | | z | | z | | | a | |  | | F | | a | | b | | r | | i | | z | | | i | | o | | |  | | L | | u | | i | | s | |  | |  | | 9 | | 7 | | 4 | | 0 | | 2 | |
| GRUPO | | | | | | | APELLIDO Y NOMBRE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | PADRÓN | | | | | | | | | |
| **DATOS DE LA ENTREGA** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | |  | |  | |  | |  | | |  | |  | | |  | |  | |  | |  | |  | |  | |  | | |  | |  | |  | | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| C | T | 1 | | 4 | | . | | t | | x | | | t | | | 4 | | 8 | | - | | 2 | | 0 | | 3 | | 9 | | 4 | | 1 | | | 7 | | 0 | 6 | | 2 | | 0 | | 1 | | 6 | | 1 | | 7 | | 0 | | 6 | | 2 | | 0 | | 1 | | 6 |
| 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 =

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 ()** | **Error del semieje mayor (∆a)** | **Semieje mayor (a)**  **(a =** ± **∆a)** |
| 1,00E+02 | 6.08083e+010 | 864522 |  |
| 1,00E+03 | 6,06744e+010 | 851339 |  |
| 1,00E+04 | 6.06648e+010 | 850238 |  |
| 1,00E+05 | 6.06639e+010 | 850130 |  |
| 1,00E+06 | 6.06638e+010 | 850119 |  |
| **1,00E+07** | **6.06638e+010** | **850118** |  |
| 1,00E+08 | 6.06638e+010 | 850118 |  |
| 1,00E+09 | 6.06638e+010 | 850118 |  |
| 1,00E+10 | 6.06637e+010 | 850118 |  |
| 1,00E+11 | 6.06638e+010 | 850118 |  |

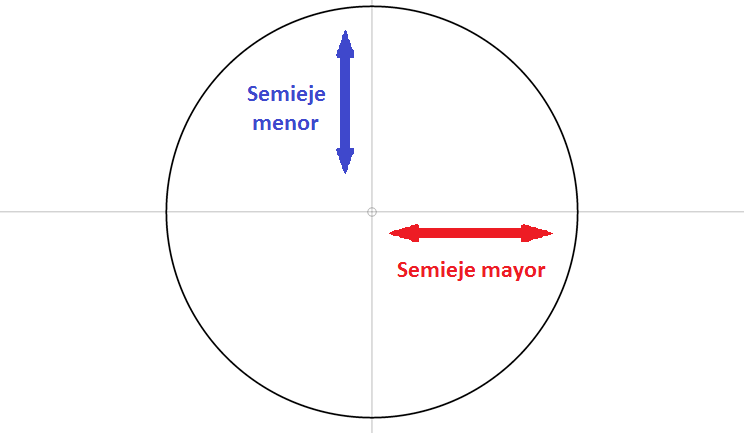
Cálculos del semieje menor de la órbita Mustafar a partir de los resultados del TP1:

|  |  |  |  |
| --- | --- | --- | --- |
| **PASOS (N)** | **Semieje menor aproximado ()** | **Error del semieje menor (∆b)** | **Semieje menor (b)**  **(b =** ± **∆b)** |
| 1,00E+02 | 5,95688e+010 | 846900 |  |
| 1,00E+03 | 5,94376e+010 | 833986 |  |
| 1,00E+04 | 5.94282e+010 | 832907 |  |
| 1,00E+05 | 5.94273e+010 | 832801 |  |
| 1,00E+06 | 5.94272e+010 | 832790 |  |
| **1,00E+07** | **5.94272e+010** | **832789** |  |
| 1,00E+08 | 5.94272e+010 | 832789 |  |
| 1,00E+09 | 5.94272e+010 | 832789 |  |
| 1,00E+10 | 5.94272e+010 | 832789 |  |
| 1,00E+11 | 5.94272e+010 | 832789 |  |

**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 recientemente 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)

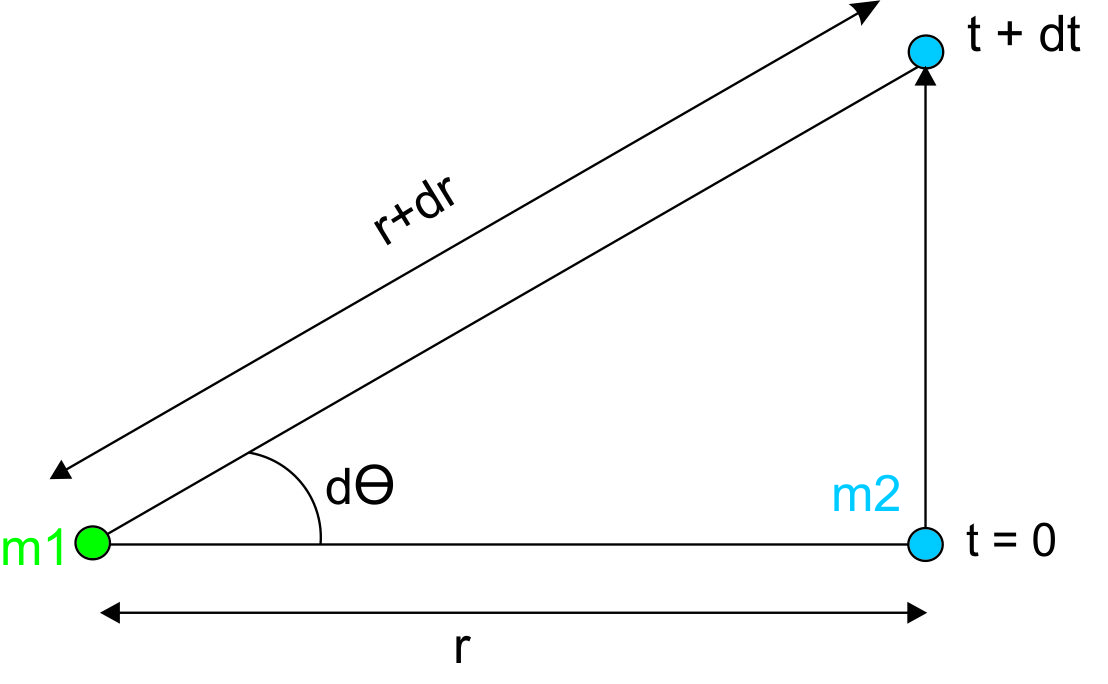
**Segunda ley** (1609): "*El radio vector que une un planeta y el Sol barre áreas iguales en tiempos iguales*".

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

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:

Siendo T el periodo y el área barrida de una órbita, que mediante el siguiente esquema:



Considerando dr chico y

Como el algoritmo uno programa el método de Euler explícito de (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:

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 ʓ un punto perteneciente al intervalo.

Utilizando este método se llega a los siguientes resultados:

|  |  |  |  |
| --- | --- | --- | --- |
| **PASOS (N)** | **Área aproximada ()** | **Error del área(∆)** | **Área ()**  **( =** ± **∆)** |
| 1,00E+02 | 1.13797e+022 | 3.23575e+017 |  |
| 1,00E+03 | 1.13297e+022 | 3.17939e+017 |  |
| 1,00E+04 | 1.13261e+022 | 3.17478e+017 |  |
| 1,00E+05 | 1.13258e+022 | 3.17432e+017 |  |
| 1,00E+06 | 1,13257E+27 | 3.17428e+017 |  |
| **1,00E+07** | **1.13257e+022** | **3.17427e+017** |  |
| 1,00E+08 | 1.13257e+022 | 3.17427e+017 |  |
| 1,00E+09 | 1.13257e+022 | 3.17427e+017 |  |
| 1,00E+10 | 1.13257e+022 | 3.17427e+017 |  |
| 1,00E+11 | 1.13257e+022 | 3.17427e+017 |  |
| **PASOS (N)** | **Período aproximada ()** | **Error del período (∆T)** | **Período (T)**  **(T =** ± **∆T)** |
| 1,00E+02 | 1.05982e+013 | 1.22033e+010 |  |
| 1,00E+03 | 1.05516e+013 | 1.19908e+010 |  |
| 1,00E+04 | 1.05482e+013 | 1.19734e+010 |  |
| 1,00E+05 | 1.05480e+013 | 1.19717e+010 |  |
| 1,00E+06 | 1.05479e+013 | 1.19715e+010 |  |
| **1,00E+07** | **1.05479e+013** | **1.19715e+010** |  |
| 1,00E+08 | 1.05479e+013 | 1.19715e+010 |  |
| 1,00E+09 | 1.05479e+013 | 1.19715e+010 |  |
| 1,00E+10 | 1.05479e+013 | 1.19715e+010 |  |
| 1,00E+11 | 1.05479e+013 | 1.19715e+010 |  |

**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 y menos visible, 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 lento, 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:

Donde 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 |  |
| 1,00E+03 | 4.98445e-007 | 1.15385e-009 |  |
| 1,00E+04 | 4.98366e-007 | 1.15235e-009 |  |
| 1,00E+05 | 4.98358e-007 | 1.15221e-009 |  |
| 1,00E+06 | 4.98358e-007 | 1.15219e-009 |  |
| **1,00E+07** | **4.98358e-007** | **1.15219e-009** |  |
| 1,00E+08 | 4.98357e-007 | 1.15219e-009 |  |
| 1,00E+09 | 4.98357e-007 | 1.15219e-009 |  |
| 1,00E+10 | 4.98357e-007 | 1.15219e-009 |  |
| 1,00E+11 | 4.98357e-007 | 1.15219e-009 |  |

**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 nota 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:

Donde , siendo la aproximación numérica de con y siendo

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)** |  |
| 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.

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 ()** | **Error del semieje mayor (∆a)** | **Semieje mayor (a)**  **(a =** ± **∆a)** |
| 1,00E+02 | 6.06638e+010 | 850118 |  |
| 1,00E+03 | 6.06638e+010 | 850118 |  |
| 1,00E+04 | 6.06638e+010 | 850118 |  |
| 1,00E+05 | 6.06638e+010 | 850118 |  |
| 1,00E+06 | 6.06638e+010 | 850118 |  |
| **1,00E+07** | **6.06638e+010** | 850118 |  |
| 1,00E+08 | 6.06638e+010 | 850118 |  |
| 1,00E+09 | 6.06638e+010 | 850118 |  |
| 1,00E+10 | 6.06638e+010 | 850118 |  |
| 1,00E+11 | 6.06638e+010 | 850118 |  |

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

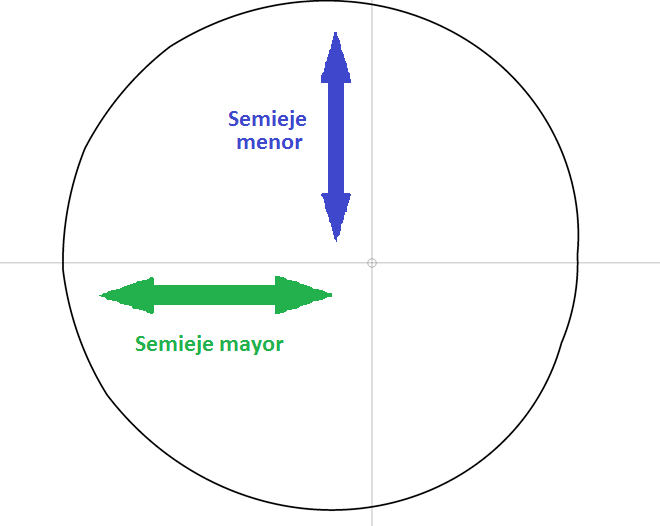
|  |  |  |  |
| --- | --- | --- | --- |
| **PASOS (N)** | **Semieje menor aproximado ()** | **Error del semieje menor (∆b)** | **Semieje menor (b)**  **(b =** ± **∆b)** |
| 1,00E+02 | 5.94272e+010 | 832789 |  |
| 1,00E+03 | 5.94272e+010 | 832789 |  |
| 1,00E+04 | 5.94272e+010 | 832789 |  |
| 1,00E+05 | 5.94272e+010 | 832789 |  |
| 1,00E+06 | 5.94272e+010 | 832789 |  |
| **1,00E+07** | **5.94272e+010** | **832789** |  |
| 1,00E+08 | 5.94272e+010 | 832789 |  |
| 1,00E+09 | 5.94272e+010 | 832789 |  |
| 1,00E+10 | 5.94272e+010 | 832789 |  |
| 1,00E+11 | 5.94272e+010 | 832789 |  |

**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 (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:

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 ()** | **Error del área(∆)** | **Área ()**  **( =** ± **∆)** |
| 1,00E+02 | 1.13257e+022 | 3.17427e+017 |  |
| 1,00E+03 | 1.13257e+022 | 3.17427e+017 |  |
| 1,00E+04 | 1.13257e+022 | 3.17427e+017 |  |
| 1,00E+05 | 1.13257e+022 | 3.17427e+017 |  |
| 1,00E+06 | 1.13257e+022 | 3.17427e+017 |  |
| **1,00E+07** | **1.13257e+022** | **3.17427e+017** |  |
| 1,00E+08 | 1.13257e+022 | 3.17427e+017 |  |
| 1,00E+09 | 1.13257e+022 | 3.17427e+017 |  |
| 1,00E+10 | 1.13257e+022 | 3.17427e+017 |  |
| 1,00E+11 | 1.13257e+022 | 3.17427e+017 |  |
| **PASOS (N)** | **Período aproximada ()** | **Error del período (∆T)** | **Período (T)**  **(T =** ± **∆T)** |
| 1,00E+02 | 1.05479e+013 | 1.19715e+010 |  |
| 1,00E+03 | 1.05479e+013 | 1.19715e+010 |  |
| 1,00E+04 | 1.05479e+013 | 1.19715e+010 |  |
| 1,00E+05 | 1.05479e+013 | 1.19715e+010 |  |
| 1,00E+06 | 1.05479e+013 | 1.19715e+010 |  |
| **1,00E+07** | **1.05479e+013** | **1.19715e+010** |  |
| 1,00E+08 | 1.05479e+013 | 1.19715e+010 |  |
| 1,00E+09 | 1.05479e+013 | 1.19715e+010 |  |
| 1,00E+10 | 1.05479e+013 | 1.19715e+010 |  |
| 1,00E+11 | 1.05479e+013 | 1.19715e+010 |  |

**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 |  |
| 1,00E+03 | 4.98357e-007 | 1.15219e-009 |  |
| 1,00E+04 | 4.98357e-007 | 1.15219e-009 |  |
| 1,00E+05 | 4.98357e-007 | 1.15219e-009 |  |
| 1,00E+06 | 4.98357e-007 | 1.15219e-009 |  |
| **1,00E+07** | **4.98357e-007** | **1.15219e-009** |  |
| 1,00E+08 | 4.98357e-007 | 1.15219e-009 |  |
| 1,00E+09 | 4.98357e-007 | 1.15219e-009 |  |
| 1,00E+10 | 4.98357e-007 | 1.15219e-009 |  |
| 1,00E+11 | 4.98357e-007 | 1.15219e-009 |  |

**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)** |  |
| 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). Como era el caso de manera similar con el periodo T, se tiene la misma energía total en órbitas elípticas para todos aquellos pertenecientes al eje mayor, independientemente de la excentricidad de la elipse.

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:

(1)

, (2)

, (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 = 3x108 m/s

Su desarrollo está dado por el siguiente pseudocódigo:

**Algoritmo 1-GR**

Desde a N - 1

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 (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 , que solo en este inciso será . También, por la misma razón del corrimiento se deberá realizar una interpolación debido a que por el análisis numérico la órbita posiblemente no vuelva al mismo lugar donde comenzó y se deberá interpolar entre los 3 puntos más próximos al inicio para deducir la precesión. Como este algoritmo no es muy preciso, es posible que la validación de la precesión se cumpla si o solo si el paso es muy pero muy grande.

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| **PASOS (N)** | **Precesión aproximada ()** | **Error de la precesión (∆)** | **Precesión ()**  **( =** ± **∆)** |
| 1,00E+02 |  |  |  |
| 1,00E+03 |  |  |  |
| 1,00E+04 |  |  |  |
| 1,00E+05 |  |  |  |
| 1,00E+06 |  |  |  |
| **1,00E+07** |  |  |  |
| 1,00E+08 |  |  |  |
| 1,00E+09 |  |  |  |
| 1,00E+10 |  |  |  |
| 1,00E+11 |  |  |  |

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)** |  |  |
| 1,00E+02 |  |  |
| 1,00E+03 |  |  |
| 1,00E+04 |  |  |
| 1,00E+05 |  |  |
| 1,00E+06 |  |  |
| **1,00E+07** |  |  |
| 1,00E+08 |  |  |
| 1,00E+09 |  |  |
| 1,00E+10 |  |  |
| 1,00E+11 |  |  |

Para los valores presentes en el sistema solar los resultados cuantitativos de la teoría einsteniana son numéricamente muy cercanos a la teoría newtoniana, se vería que el método convergería a un número y el problema aun tendería a ser conservativo, no sería correcto afirmarlo por el análisis no exacto a través del método numérico, en este caso aún menos por la aparición del termino relativista de Einstein.

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:

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

**Algoritmo 2-GR**

Desde a N - 1

Avanzar n

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 aproximada ()** | **Error de la precesión (∆)** | **Precesión ()**  **( =** ± **∆)** |
| 1,00E+02 |  |  |  |
| 1,00E+03 |  |  |  |
| 1,00E+04 |  |  |  |
| 1,00E+05 |  |  |  |
| 1,00E+06 |  |  |  |
| **1,00E+07** |  |  |  |
| 1,00E+08 |  |  |  |
| 1,00E+09 |  |  |  |
| 1,00E+10 |  |  |  |
| 1,00E+11 |  |  |  |

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)** |  |  |
| 1,00E+02 |  |  |
| 1,00E+03 |  |  |
| 1,00E+04 |  |  |
| 1,00E+05 |  |  |
| 1,00E+06 |  |  |
| **1,00E+07** |  |  |
| 1,00E+08 |  |  |
| 1,00E+09 |  |  |
| 1,00E+10 |  |  |
| 1,00E+11 |  |  |

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 aceptables para pasos no tan pequeños. 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. 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 por un lado un análisis más profundo del problema propuesto por Newton, y a través de los cálculos previos, verificar la convergencia de lo que fueron los métodos de Euler explícito y Runge-Kutta 4 (sin saber que eran estos en el TP1), teniendo siempre en cuenta el orden propuesto para la realización de la cuadratura y diferenciación numérica para ser consistente durante todo el proceso, como también ver de una manera más sencilla que el problema es conservativo a través del análisis de su energía. Como este es un PVI la idea de que el sistema converja habla de un problema bien condicionado, aunque sería conveniente verificar esto para ambos casos a través de perturbaciones de Von Neumann.

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 posiblemente para un problema tan complejo no sea lo mejor el uso de este algoritmo. Luego a través del método de RK4, tiene la ventaja de ser explicito por un lado pero la desventaja de tener una condición máxima de estabilidad, aunque si es recomendable para este caso por tener orden 4 para resolver este tipo de problema complejo.

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

* *Velocidad de cómputo*: la del algoritmo dos GR es 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.
* *Precesión*: el cálculo y grafico de la precesión debería ser menos brusco en el algoritmo 2GR que en el algoritmo 1GR debido a sus órdenes.
* Tanto para la primera como la segunda parte, el sistema poseería energía del tipo conservativa.
* El segundo algoritmo-GR está mejor condicionado que el primero, debido a que la solución cuando los pasos son pequeños se parece bastante a la solución con pasos grandes, 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.

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;

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();

}

void lagrange(double x, double X[], double y[], int Lit)

{

double r = 0, num = 1, den = 1;

for (int i = 0; i<Lit; i++){ //para el total de polinomios

for (int j = 0; j<Lit; j++){ //para cada polinomio

if (i != j){ num \*= (x - X[j]); den \*= (X[i] - X[j]); }

}

num \*= y[i];

printf("Interacion %d valor %lf\n", i, num / den);

\_getch();

r += num / den;

num = den = 1;

}

printf("\nEl resultado es: %lf", r);

}

void pointsInterpolator() //seria para ejecutar el main este proceso

{

int m;

double \*X, \*Y, x;

void clrscr();

printf("cuantas entradas tendra la tabla?\n\t\t");

scanf\_s("%d", &m);

X = (double\*)malloc(sizeof(double)\*m);

printf("Ingresa la tabla los valores de X:\n");

for (int i = 0; i<m; i++) scanf\_s("%lf", &X[i]);

printf("\nIngresa la tabla los valores de Y:\n");

Y = (double\*)malloc(sizeof(double)\*m);

for (int i = 0; i<m; i++) scanf\_s("%lf", &Y[i]);

printf("Escribe el valor X para el cual se encontrara el valor de Y\n");

scanf\_s("%lf", &x);

lagrange(x, X, Y, m);

\_getch();

}

double rectangularIntegral(double(\*f)(double x), double a, double b, int steps) { //function, I[a,b]

double step = (b - a) / steps; // width of each small rectangle

double area = 0.0; // signed area

for (int i = 0; i < steps; i++) {

area += f(a + (i + 0.5) \* step) \* step; // sum up each small rectangle

}

return area;

}

float f(float x) //function example

{

return(1 / (1 + pow(x, 2)));

}

void trapezoidalIntegral()

{

int i, n;

float x0, xn, h, y[20], so, se, ans, x[20];

printf("\n Enter values of x0,xn,h:\n");

scanf\_s("%f%f%f", &x0, &xn, &h);

n = (xn - x0) / h;

if (n % 2 == 1)

{

n = n + 1;

}

h = (xn - x0) / n;

printf("\nrefined value of n and h are:%d %f\n", n, h);

printf("\n Y values \n");

for (i = 0; i <= n; i++)

{

x[i] = x0 + i\*h;

y[i] = f(x[i]);

printf("\n%f\n", y[i]);

}

so = 0;

se = 0;

for (i = 1; i<n; i++)

{

if (i % 2 == 1)

{

so = so + y[i];

}

else

{

se = se + y[i];

}

}

ans = h / 3 \* (y[0] + y[n] + 4 \* so + 2 \* se);

printf("\nfinal integration is %f", ans);

\_getch();

}

float simpsonsOneIntegral(float(\*f)(float x), float a, float b, int n) {

float h = (b - a) / n;

float x;

float r;

char m = 0;

float s = 0.0;

for (x = a; x <= b; x += h) {

r = f(x);

if (x == a || x == b) {

s += r;

}

else {

m = !m;

s += r \* (m + 1) \* 2.0;

}

}

return s \* (h / 3.0);

}

float simpsonsTwoIntegral(float(\*f)(float x), float a, float b, int n)

{

float h = (b - a) / n;

float sum1 = f(a + h / 2);

float sum2 = 0;

for (int i = 1; i < n - 1; i++)

{

sum1 = sum1 + f(a + h \* i + h / 2);

sum2 = sum2 + f(a + h \* i);

}

float answer = (h / 6) \* (f(a) + f(b) + 4 \* sum1 + 2 \* sum2);

return answer;

}

void solve\_A\_1(alg\_params params, alg\_data data)

{

// 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;

// Guardar resultados

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();

}

//std::fstream statsOutput;

//statsOutput.open("mustafar\_stats.csv", std::ios\_base::app);

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);

//statsOutput << params.alg << ";";

//statsOutput << params.dp << ";";

//statsOutput << params.steps << ";";

//statsOutput << 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");

//REAL ddr = 0;

//if (dp)

//{

// ddr = computeDDR<double>(data.u\_n\_4\_4\_N, semiMinorAxis, epsilon);

//}

//else

//{

// ddr = computeDDR<float>(data.u\_n\_4\_4\_N, semiMinorAxis, epsilon);

//}

//printf("DDR=%e\n", ddr);

//printf("Saving stats...\n");

//statsOutput << ddr << ";";

//statsOutput << (double)tiempoDeCorrida << ";";

//statsOutput << 1.0 / data.u\_n\_min << ";";

//statsOutput << 1.0 / data.u\_n\_max << ";";

//statsOutput << "\n";

//statsOutput.close();

solve\_A\_1(params, data);

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

100;6.08083e+010;864523;5.95688e+010;846900;

1000;6.06744e+010;851339;5.94376e+010;833985;

10000;6.06641e+010;850217;5.94276e+010;832886;

100000;6.06691e+010;850300;5.94324e+010;832968;

1e+006;6.06275e+010;848947;5.93917e+010;831642;

1e+007;6.02572e+010;837038;5.90289e+010;819976;

1e+008;4.84777e+010;521823;4.74895e+010;511186;

1e+009;4.84777e+010;521823;4.74895e+010;511186;

1e+010;4.84777e+010;521823;4.74895e+010;511186;

1e+011;4.84777e+010;521823;4.74895e+010;511186;

100;6.06637e+010;850118;5.94272e+010;832789;

1000;6.06638e+010;850119;5.94272e+010;832790;

10000;6.06637e+010;850118;5.94272e+010;832789;

100000;6.06637e+010;850118;5.94272e+010;832789;

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

1e+007;6.06734e+010;850431;5.94367e+010;833096;

1e+008;4.84777e+010;521823;4.74895e+010;511186;

1e+009;4.84777e+010;521823;4.74895e+010;511186;

1e+010;4.84777e+010;521823;4.74895e+010;511186;

1e+011;4.84777e+010;521823;4.74895e+010;511186;