FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico N°2

Resolución de una EDO

Análisis Numérico I (75.12) – Curso 007

Fecha de entrega: 30/11/2011

Grupo N°: 004

Integrantes:

Rossi, Federico Martín - 92086

Montoya, Diego Ramiro - 91939

Bertucci, Juan Pablo - 92816

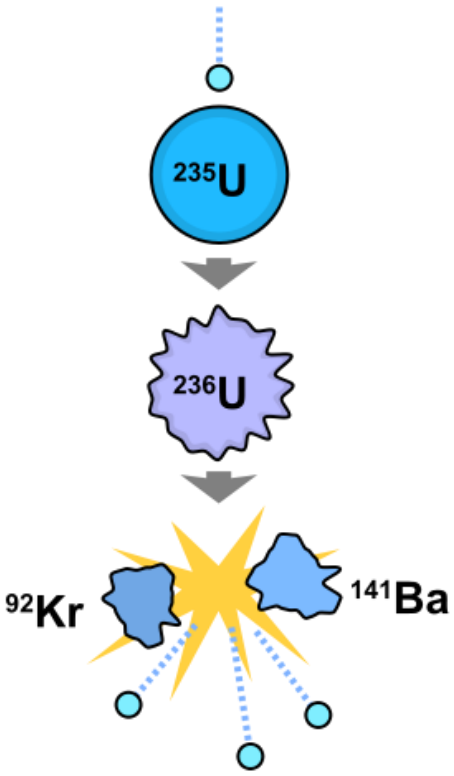
Di Nella, Camila - 91821

1. Introducción General

> "*El análisis numérico o cálculo numérico es la rama de las matemáticas que se encarga de diseñar algoritmos para, a través de números y reglas matemáticas simples, simular procesos matemáticos más complejos aplicados a procesos del mundo real*"[1]

El caso presentado en este trabajo es una clara exposición de lo enunciado anteriormente, en donde tenemos una situación inicial en la cual sólo conocemos en líneas generales el comportamiento de la fisión de los núcleos en los átomos del uranio 235 (U-235), un isótopo del uranio en el cual se induce la ruptura del núcleo y la consecuente radioactividad mediante el "bombardeo" de neutrones al mismo.

En la fisión nuclear, núcleos atómicos pesados se dividen para formar núcleos más pequeños. La fisión de núcleos pesados es un proceso exotérmico lo que supone que se liberan cantidades sustanciales de energía. Asimismo, encontramos otros productos tales como rayos alfa, beta y gamma. Por ultimo, luego de la fisión, se liberan también otros neutrones para producir lo que denominamos *reacción en cadena*, y así seguir fisionando todos los núcleos restantes.



[1] Nick Trefethen, The definition of numerical analysis, SIAM News; Noviembre 1992.

En las centrales nucleares, se trata de controlar estas reacciones en cadena. Tomando como *P* a la tasa de neutrones que se producen y *D* a la tasa de neutrones que se destruyen, obtenemos el cociente $K = P/D$. Si $k = 1$, la potencia de un reactor nuclear se mantiene constante; si $k > 1$ aumenta; si $k < 1$ disminuye.

Un reactor nuclear es básicamente un apilamiento de U-235, llamado “núcleo", en el que *D* se controla insertando y extrayendo barras de un material absorbente de neutrones, llamadas “barras de control".

Como cada reactor libera neutrones que “viven" un tiempo hasta que son absorbidos, constantemente hay una población de neutrones que depende de la tasa de fisiones y de la tasa de absorción. Además, como los neutrones fisionan átomos y cada fisión libera energía, la potencia del reactor está relacionada con la población de neutrones.

En el presente trabajo práctico se describe un proceso en el cual se quitan barras de control para aumentar la potencia del reactor. Sin embargo, la extracción termina siendo mayor que la prevista, y luego de 60 segundos la tasa de crecimiento es mayor que un umbral de seguridad. Habiéndose dado cuenta del imprevisto, el operador ordena la inmediata anulación del proceso, introduciendo una cantidad de barras de control tal que absorban todos los neutrones presentes, operación que transcurre en 12 segundos.

La ecuación que rige el comportamiento mencionado anteriormente es:

$$\frac{dN(t)}{dt} = (k(t) - 1).\frac{N(t)}{I} \qquad (1)$$

dónde:

- $N(t)$ es la cantidad de neutrones;
- $k(t)$ es la relación entre neutrones producidos y neutrones destruidos. Depende de t, porque la cantidad de neutrones absorbida se controla con las barras de control;
- $I = 0.3seg$ vida media de un neutrón en el núcleo.

1.1. Objetivos

Con el objetivo de obtener el número de neutrones producidos durante el infortunio, recurrimos a los métodos del Análisis Numérico ya que sería poco práctico obtener la solución analítica que gobierna el proceso. Sin embargo, a los fines prácticos, el análisis numérico nos ofrece una solución suficientemente cercana a la real. La cantidad de trabajo del ordenador y de las operaciones que sean necesarias efectuar dependerá de cuan precisa queramos la solución del problema; es decir, de cuan pequeño debamos hacer el error. De todas formas, somos conscientes de que las soluciones numéricas que hallemos no suplantan a las soluciones analíticas, sino que las complementan.

Para ello adoptamos distintos métodos numéricos, tales como el *Método de Euler*, el *Método Runge Kutta 4 (RK-4)* para la resolución de la ecuación diferencial (1) y el *Método de*

Romberg para la posterior integración numérica de los datos obtenidos por ambos métodos, de forma tal de poder realizar una comparación entre estos.

1.2. Comentarios

El desarrollo del trabajo práctico se dividirá en varios apartados de acuerdo a lo mencionado anteriormente, a fin de lograr un mayor entendimiento y organización de lo expuesto.

Todos los archivos y códigos fuente aquí mencionados, así como también el presente informe, pueden ser descargados de la sección *Downloads* del repositorio del grupo alojado en Google Code (*http://code.google.com/p/7512-analisis-numerico-grupo004/*).

2. Aplicación del *Método de Euler*

Como primer intento de dar solución al problema propuesto, utilizaremos el *Método de Euler* para ecuaciones diferenciales de primer orden. Esta sustituye la curva real que buscamos como respuesta de la ecuación diferencial por una serie de aproximaciones sucesivas, utilizando la pendiente obtenida en cada salto de paso h. En particular hallaremos la solución numérica de $N(t)$ con los pasos de tiempo h = 4 seg, 2 seg, 1 seg, 0.5 seg.

Para llevar a cabo este procedimiento se utilizó el software *Matlab*, el cual nos permite mediante sencillas implementaciones algorítmicas llegar a los resultados de forma más eficiente y rápida. Vale aclarar que se buscó un alto grado de modularización y de generalización de las funciones, a fin de que estas puedan ser lo más reutilizables posible.

2.1. Algoritmo e implementación del método

El algoritmo utilizado para el *Método de Euler* puede verse en el *Código 2.1*. Este algoritmo es una implementación general del método y es capaz de soportar cualquier tipo de ecuación que se presente. Esto se debe a que la función f que recibe como parámetro es lo que en *Matlab* se denomina *function handle*, lo que proporciona la posibilidad de llamar a una función externa de forma indirecta (sin ser conocida de antemano). Es así que la función f funciona como una caja negra, la cual recibe y devuelve valores. En la documentación se explica más detalladamente los requisitos que debe cumplir cada argumento.

Código 2.1 – metodoDeEuler.m

```
% Función que aplica el método de Euler para resolver una ecuación
% diferencial ordinaria a partir de un valor inicial dado.
%
%       metodoDeEuler(f,a,b,u0,h)
%
% PRE-CONDICIONES
% 'f': función que se desea procesar, la cual debe pasada como una function
% handler (ej: @miFuncion). Esta debe tener dos parametros: una variable
% independiente y una variable dependiente de la primera.
% 'a': extremo izquierdo del intervalo;
% 'b': extremo derecho del intervalo;
% 'u0': condición inicial de la función;
% 'h': tamaño del paso.
%
% POST-CONDICIONES
% Se devuelve una matriz de dos columnas por N filas, siendo
% N = (b-a)/h. La primer columna alberga los pasos en los que fue
% aplicado el método y la segunda columna almacena el resultado
% obtenido en cada paso.
function matrizDeResultados = metodoDeEuler(f,a,b,u0,h)

    % Calculamos el número de intervalos N.
    N = (b-a)/h;

    % Creamos el vector U que contiene los resultados de cada paso
    U = zeros(1, N+1);
    U(1) = u0;

    % Creamos el vector auxiliar P que contiene los pasos en donde se
    % aplicó el método.
    P = zeros(1, N+1);
    P = a:h:b;

    % Iteramos para aplicar el método.
    for j=1:N

        U(j+1) = U(j)+ (h * f(P(j), U(j)));

    end

    % Creamos una matriz que esta formada por los vectores P y U
    % traspuestos.
    matrizDeResultados = [P' U'];
```

En nuestro caso, la ecuación a procesar es la (1) ya antes expuesta. Por esta razón, se creó una función en *Matlab* que la representara, siendo su implementación la mostrada en el *Código 2.2*. En esta puede verse que se reciben dos parámetros, el primero es la variable independiente y el segundo es la variable dependiente de la primera.

Código 2.2 – tp2Ecuacion.m

```
% Función que representa la ecuación simplificada que rige el
% comportamiento temporal de un reactor nuclear.
function res = tp2Ecuacion(t, N)

    format longG

    % Vida media en segundos de un neutrón en el núcleo
    I = 0.3;

    valorK = obtenerValorK(t);

    res = (valorK(1,2) - 1) * N / I;%
```

Como puede notarse, en este último código se utiliza una función externa denominada *obtenerValorK(),* cuyo código fuente se muestra en el *Código 2.3*. Esta es la encargada de obtener los valores de *k(t)* de la tabla, la cual fue obtenida de forma empírica. Vale resaltar que esta tabla debe ser un archivo con formato CSV, por lo que más adelante se verá que el archivo *k.txt* provisto por la cátedra es convertido para cumplir con estas especificaciones. La razón por la cual se optó por esta medida fue la necesidad de lograr la mayor agilidad de procesamiento posible. La función de *Matlab* que carga los datos de un archivo con formato de texto plano no es lo suficientemente rápida para lograr una carga eficiente en cada consulta mientras que la función utilizada para cargar los archivos con formato CSV si lo es.

Cabe destacar (una importante aclaración con respecto a la utilización de los algoritmos aquí presentados para la resolución del problema propuesto) la utilización del comando *'format longG'* en la implementación de este último algoritmo, como así también en una gran parte de los restantes. *Matlab* utiliza por default 5 cifras significativas para la presentación de resultados por pantalla. En el caso de este Trabajo Práctico, consideramos necesario el aumento de este valor, por lo cual se decidió utilizar el formato *longG,* el cual presenta valores reales con la utilización de hasta 16 cifras significativas. De esta manera, los resultados obtenidos por los algoritmos numéricos, como son *Euler*, *Runge Kutta 4* y *Romberg,* son mucho más precisos, y el valor del error cometido se estrecha en gran medida.

Código 2.3 – obtenerValorK.m

```
% Función que busca el valor de 'k' correspondiente a un tiempo t dado por
% parametro.
%
%       obtenerValorK(t)
%
% PRE-CONDICIONES
% t: es el tiempo en segundos.
```

```
%
% POST-CONDICIONES
% Se devuelve una matriz de dimension 1x2, donde en la primer columna se
% encuentra el valor t al cual corresponde el k(t) buscado, y en la segunda
% columna se almacena este último.
function matrizResultado = obtenerValorK(t)

    % Configuración
    format longG;
    directorio = 'mediciones/';
    archivo = 'k.csv';

    % Abrimos el archivo y generamos una matriz con los datos extraidos
    datosDeK = dlmread(strcat(directorio, archivo),',');

    % Buscamos el valor de K para el tiempo t dado
    posT = datosDeK(:,1) == t;

    matrizResultado = [t datosDeK(posT,2)];

end
```

Por último, presentamos el código principal (*Código 2.4*) que invoca al método de Euler y le aplica la ecuación (1) para obtener los resultados para los distintos pasos *h* antes mencionados. Luego de obtener los valores se genera un archivo CSV por cada paso *h* procesado, el cual contiene su tabla correspondiente de valores. En el proceso de conversión de las distintas matrices resultantes del Método de Euler para los pasos indicados a un archivo de formato CSV, se utilizó el comando *'dlmwrite'* en el cual se seteó específicamente una precisión de 16 cifras significativas, para no generar errores de redondeo o truncamiento con los valores resultantes de dicho método. También se utilizó este valor de precisión para el traspaso de datos de los resultados obtenidos por el Método de Runge Kutta 4.

Código 2.4 – tp2MetodoDeEuler.m

```
% Función que utiliza el Método de Euler y lo aplica a la función propuesta
% en el TP2.
%
% POST-CONDICIONES
% Se generan cuatro archivos con formato CSV, almacenandose en cada uno los
% valores obtenidos al aplicar el método para distintos pasos h. Estos
% archivos se guardan en la carpeta 'CSV' que se debe encontrar en el mismo
% directorio raíz que este archivo.
function tp2MetodoDeEuler()

    format longG

    % Especificaciones
    funcion = @tp2Ecuacion;
    a = 0;
    b = 120;
    U0= 10^10;
```

```
    h = [4, 2, 1, 0.5];

    % Convertimos el archivo de datos empiricos k.txt en un archivo de formato
    % CSV
    convertirTXTaCSV();

    % Procesamos utilizando el método de Euler
    disp(' ');
    disp('Iniciando aplicación del Método de Euler.')

    disp(strcat('Procesando para el paso h=', num2str(h(1,1)), '...'));
    e1 = metodoDeEuler(funcion, a , b , U0 , h(1,1));

    disp(strcat('Procesando para el paso h=', num2str(h(1,2)), '...'));
    e2 = metodoDeEuler(funcion, a , b , U0 , h(1,2));

    disp(strcat('Procesando para el paso h=', num2str(h(1,3)), '...'));
    e3 = metodoDeEuler(funcion, a , b , U0 , h(1,3));

    disp(strcat('Procesando para el paso h=', num2str(h(1,4)), '...'));
    e4 = metodoDeEuler(funcion, a , b , U0 , h(1,4));

    % Generamos los archivos con formato CSV donde se albergan los
    % valores obtenidos para cada paso
    disp('Generando archivos CSV de datos...');
    dlmwrite('CSV/tp2MetodoDeEulerValoresPaso4.csv', e1, 'precision', 16);
    dlmwrite('CSV/tp2MetodoDeEulerValoresPaso2.csv', e2, 'precision', 16);
    dlmwrite('CSV/tp2MetodoDeEulerValoresPaso1.csv', e3, 'precision', 16);
    dlmwrite('CSV/tp2MetodoDeEulerValoresPaso05.csv', e4, 'precision', 16);
    disp('El proceso ha finalizado exitosamente.');
    disp(' ');

end
```

Tal como se destacó antes, el archivo *k.txt* debe ser convertido en un archivo con formato CSV y esto se logra invocando a la función auxiliar *convertirTXTaCSV(),* como se puede ver en el código. La implementación de esta función conversora se muestra en el *Apéndice A* de este informe.

Nota: *Para una mejor diferenciación entre archivos, se utilizó el prefijo 'tp2' en los nombres de archivos para indicar que estos son los que contienen las funciones centrales en lo que respecta al presente Trabajo Práctico, mientras que aquellos que obvian de él, son funciones auxiliares no sujetas a este trabajo en particular, sino que están pensadas para su futura reutilización en la resolución de otros problemas que requieran el uso de estos métodos numéricos.*

2.2. Resultados obtenidos

Pasaremos ahora a mostrar los resultados obtenidos con el *Método de Euler* para cada paso *h* aplicado.

2.2.1. Resultados para *h* = 4 seg

A continuación, en la *Tabla 2.1*, se muestra la tabla correspondiente a los valores de *N(t)* obtenidos al aplicar un paso de *h* = 4 seg. Además, en el *Gráfico 2.1*, se muestra el grafico correspondiente a dicha tabla.

Tiempo (t)	*N(t)*
0,000000000000	10000000000,000000000000
4,000000000000	10000000000,000000000000
8,000000000000	10017777773,333300000000
12,000000000000	10053396543,201900000000
16,000000000000	10107014658,099000000000
20,000000000000	10178886757,842400000000
24,000000000000	10269365755,769300000000
28,000000000000	10378905657,164200000000
32,000000000000	10508065367,396000000000
36,000000000000	10657513412,847100000000
40,000000000000	10828033627,452600000000
44,000000000000	11020531998,239300000000
48,000000000000	11236044628,880700000000
52,000000000000	11475746914,296800000000
56,000000000000	11740964171,215800000000
60,000000000000	12033183729,139800000000
64,000000000000	12354068628,583600000000
68,000000000000	12661547675,496800000000
72,000000000000	12954170105,036500000000
76,000000000000	13230525733,943900000000
80,000000000000	-4410175244,647990000000
84,000000000000	1470058414,882660000000
88,000000000000	-490019471,627554000000
92,000000000000	163339823,875851000000
96,000000000000	-54446607,958617000000
100,000000000000	18148869,319539000000
104,000000000000	-6049623,106513000000
108,000000000000	2016541,035504330000
112,000000000000	-672180,345168110000
116,000000000000	224060,115056036000
120,000000000000	-74686,705018678800

Tabla 2.1 – Valores para *h* = 4 seg.

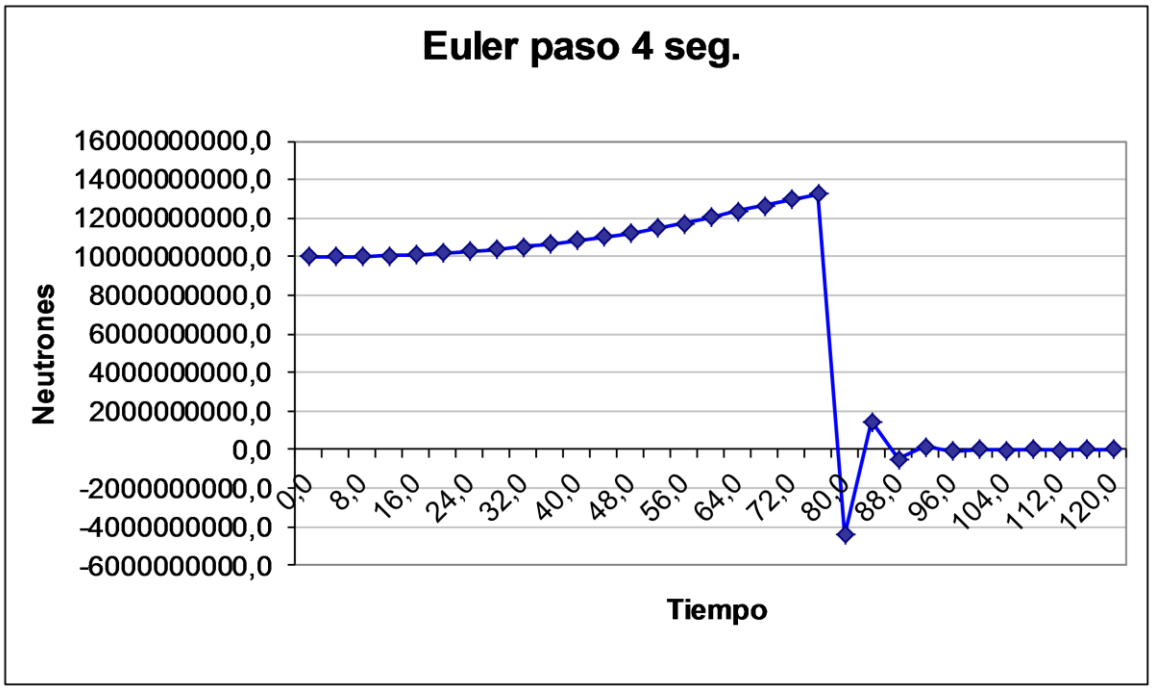


Gráfico 2.1 - Euler paso 4 segundos.

2.2.2. Resultados para *h* = 2 seg

A continuación, en la *Tabla 2.2*, se muestra la tabla correspondiente a los valores de *N(t)* obtenidos al aplicar un paso de *h* = 2 seg. Además, en el *Gráfico 2.2*, se muestra el grafico correspondiente a dicha tabla.

Tiempo (t)	*N(t)*
0,000000000000	10000000000,000000000000
2,000000000000	10000000000,000000000000
4,000000000000	10004444446,666600000000
6,000000000000	10013337283,951600000000
8,000000000000	10026688400,330200000000
10,000000000000	10044513626,381100000000
12,000000000000	10066834765,541000000000
14,000000000000	10093679658,249100000000
16,000000000000	10125082219,428900000000
18,000000000000	10161082509,514600000000
20,000000000000	10201726839,552700000000
22,000000000000	10247067849,995500000000
24,000000000000	10297164623,873900000000
26,000000000000	10352082835,201200000000

28,000000000000	10411894871,660600000000
30,000000000000	10476679992,992700000000
32,000000000000	10546524526,279400000000
34,000000000000	10621522036,365500000000
36,000000000000	10701773533,835500000000
38,000000000000	10787387722,106100000000
40,000000000000	10878481220,823400000000
42,000000000000	10975178829,257700000000
44,000000000000	11077613831,664100000000
46,000000000000	11185928280,479800000000
48,000000000000	11300273322,639000000000
50,000000000000	11420809571,413800000000
52,000000000000	11547707458,078600000000
54,000000000000	11681147630,583600000000
56,000000000000	11821321402,150600000000
58,000000000000	11968431182,226500000000
60,000000000000	12122690959,248900000000
62,000000000000	12284326838,705500000000
64,000000000000	12442658159,674600000000
66,000000000000	12597500130,648900000000
68,000000000000	12748670132,216700000000
70,000000000000	12895988095,355900000000
72,000000000000	13039276854,836800000000
74,000000000000	13178362474,621700000000
76,000000000000	4392787491,540580000000
78,000000000000	1464262497,180190000000
80,000000000000	488087499,060065000000
82,000000000000	162695833,020021000000
84,000000000000	54231944,340007200000
86,000000000000	18077314,780002400000
88,000000000000	6025771,593334140000
90,000000000000	2008590,531111380000
92,000000000000	669530,177037128000
94,000000000000	223176,725679042000
96,000000000000	74392,241893014200
98,000000000000	24797,413964338100
100,000000000000	8265,804654779370
102,000000000000	2755,268218259790
104,000000000000	918,422739419930
106,000000000000	306,140913139977
108,000000000000	102,046971046659

110,000000000000	34,015657015553
112,000000000000	11,338552338518
114,000000000000	3,779517446173
116,000000000000	1,259839148724
118,000000000000	0,419946382908
120,000000000000	0,139982127636

Tabla 2.2 – Valores para *h* = 2 seg.

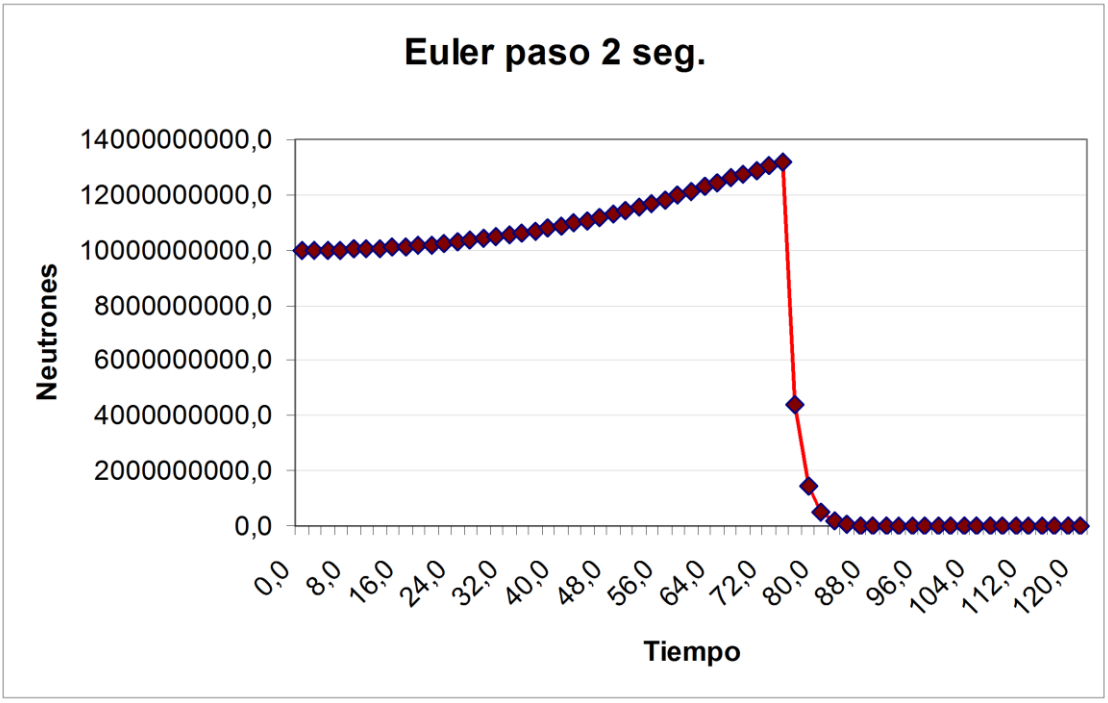


Gráfico 2.2 - Euler paso 2 segundos.

2.2.3. Resultados para *h* = 1 seg

A continuación, en la *Tabla 2.3*, se muestra la tabla correspondiente a los valores de *N(t)* obtenidos al aplicar un paso de *h* = 1 seg. Además, en el *Gráfico 2.3*, se muestra el grafico correspondiente a dicha tabla.

Tiempo (t)	*N(t)*
0,000000000000	10000000000,000000000000
1,000000000000	10000000000,000000000000
2,000000000000	10001111110,000000000000
3,000000000000	10003333580,246700000000
4,000000000000	10006668024,773500000000

5,000000000000	10011115431,672700000000
6,000000000000	10016677163,580400000000
7,000000000000	10023354948,356100000000
8,000000000000	10031150889,980000000000
9,000000000000	10040067469,663400000000
10,000000000000	10050107537,133100000000
11,000000000000	10061274322,168800000000
12,000000000000	10073571436,347100000000
13,000000000000	10087002864,928900000000
14,000000000000	10101572979,057500000000
15,000000000000	10117286538,147300000000
16,000000000000	10134148682,377600000000
17,000000000000	10152164945,575800000000
18,000000000000	10171341258,267600000000
19,000000000000	10191683940,784200000000
20,000000000000	10213199716,860100000000
21,000000000000	10235895717,365700000000
22,000000000000	10259779474,039500000000
23,000000000000	10284858933,836100000000
24,000000000000	10311142463,365300000000
25,000000000000	10338638843,267600000000
26,000000000000	10367357283,350200000000
27,000000000000	10397307427,765100000000
28,000000000000	10428499350,048400000000
29,000000000000	10460943569,089800000000
30,000000000000	10494651055,085900000000
31,000000000000	10529633225,269500000000
32,000000000000	10565901960,764400000000
33,000000000000	10603469613,354400000000
34,000000000000	10642349001,936700000000
35,000000000000	10682553430,317100000000
36,000000000000	10724096694,844200000000
37,000000000000	10766993081,623600000000
38,000000000000	10811257385,318400000000
39,000000000000	10856904917,702100000000
40,000000000000	10903951505,678800000000
41,000000000000	10952413511,159100000000
42,000000000000	11002307840,593600000000
43,000000000000	11053651943,849700000000
44,000000000000	11106463835,242100000000
45,000000000000	11160762104,115100000000

46,000000000000	11216565914,635700000000
47,000000000000	11273895028,064200000000
48,000000000000	11332769814,463500000000
49,000000000000	11393211253,473900000000
50,000000000000	11455240957,921400000000
51,000000000000	11518881186,738200000000
52,000000000000	11584154846,796400000000
53,000000000000	11651085517,957400000000
54,000000000000	11719697467,302200000000
55,000000000000	11790015652,106000000000
56,000000000000	11862065746,447700000000
57,000000000000	11935874156,854800000000
58,000000000000	12011468026,514800000000
59,000000000000	12088875263,573300000000
60,000000000000	12168124558,311100000000
61,000000000000	12249245388,699800000000
62,000000000000	12329545998,720100000000
63,000000000000	12409003071,564100000000
64,000000000000	12487593424,350700000000
65,000000000000	12565294007,045300000000
66,000000000000	12642081913,470000000000
67,000000000000	12717934404,950800000000
68,000000000000	12792828908,970800000000
69,000000000000	12866743030,134600000000
70,000000000000	12939654573,972000000000
71,000000000000	13011541545,265100000000
72,000000000000	13082382158,899200000000
73,000000000000	13152154863,746700000000
74,000000000000	8768103242,497800000000
75,000000000000	5845402161,665200000000
76,000000000000	3896934774,443470000000
77,000000000000	2597956516,295640000000
78,000000000000	1731971010,863760000000
79,000000000000	1154647340,575840000000
80,000000000000	769764893,717229000000
81,000000000000	513176595,811486000000
82,000000000000	342117730,540990000000
83,000000000000	228078487,027327000000
84,000000000000	152052324,684884000000
85,000000000000	101368216,456589000000
86,000000000000	67578810,971059900000

87,000000000000	45052540,647373200000
88,000000000000	30035027,098248800000
89,000000000000	20023351,398832500000
90,000000000000	13348900,932555000000
91,000000000000	8899267,288370030000
92,000000000000	5932844,858913350000
93,000000000000	3955229,905942230000
94,000000000000	2636819,937294820000
95,000000000000	1757879,958196550000
96,000000000000	1171919,972131030000
97,000000000000	781279,981420689000
98,000000000000	520853,320947126000
99,000000000000	347235,547298084000
100,000000000000	231490,364865389000
101,000000000000	154326,909910259000
102,000000000000	102884,606606839000
103,000000000000	68589,737737893200
104,000000000000	45726,491825262100
105,000000000000	30484,327883508100
106,000000000000	20322,885255672000
107,000000000000	13548,590170448000
108,000000000000	9032,393446965360
109,000000000000	6021,595631310240
110,000000000000	4014,397087540160
111,000000000000	2676,264725026770
112,000000000000	1784,176483351180
113,000000000000	1189,450988900780
114,000000000000	792,967325933859
115,000000000000	528,644883955906
116,000000000000	352,429922637270
117,000000000000	234,953281758180
118,000000000000	156,635521172120
119,000000000000	104,423680781413
120,000000000000	69,615787187609

Tabla 2.3 – Valores para h = 1 seg.

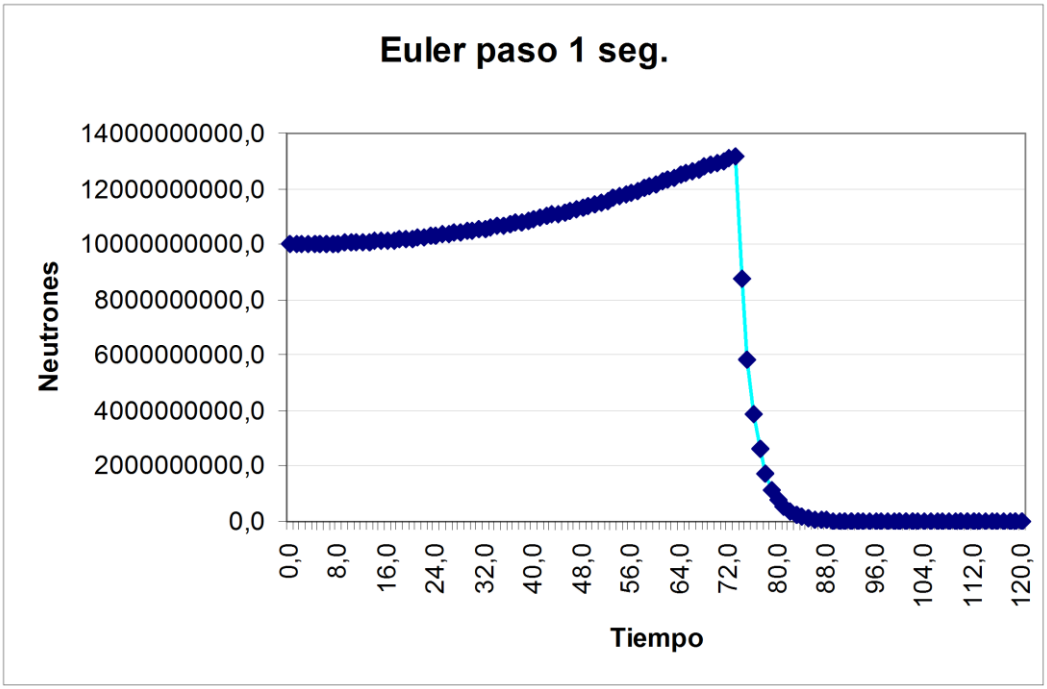


Gráfico 2.3 - Euler paso 1 segundo.

2.2.4. Resultados para *h* = 0,5 seg

A continuación, en la *Tabla 2.4*, se muestra la tabla correspondiente a los valores de *N(t)* obtenidos al aplicar un paso de *h* = 0,5 seg. Además, en el *Gráfico 2.4*, se muestra el grafico correspondiente a dicha tabla.

Tiempo (t)	*N(t)*
0,000000000000	10000000000,000000000000
0,500000000000	10000000000,000000000000
1,000000000000	10000277778,333300000000
1,500000000000	10000833348,765400000000
2,000000000000	10001666751,544500000000
2,500000000000	10002778048,405800000000
3,000000000000	10004167322,579100000000
3,500000000000	10005834683,799500000000
4,000000000000	10007780263,321700000000
4,500000000000	10010004213,935300000000
5,000000000000	10012506714,988800000000
5,500000000000	10015287967,410300000000
6,000000000000	10018348193,732900000000
6,500000000000	10021687643,130800000000

7,000000000000	10025306586,447600000000
7,500000000000	10029205316,229800000000
8,000000000000	10033384151,778200000000
8,500000000000	10037843434,180800000000
9,000000000000	10042583526,356000000000
9,500000000000	10047604818,119200000000
10,000000000000	10052907721,220300000000
10,500000000000	10058492669,395800000000
11,000000000000	10064360123,452900000000
11,500000000000	10070510566,309700000000
12,000000000000	10076944503,056500000000
12,500000000000	10083662466,058500000000
13,000000000000	10090665009,998000000000
13,500000000000	10097952711,944600000000
14,000000000000	10105526176,478500000000
14,500000000000	10113386030,732800000000
15,000000000000	10121532924,473500000000
15,500000000000	10129967535,243900000000
16,000000000000	10138690563,406400000000
16,500000000000	10147702732,232900000000
17,000000000000	10157004793,070700000000
17,500000000000	10166597520,384000000000
18,000000000000	10176481711,852900000000
18,500000000000	10186658193,564800000000
19,000000000000	10197127815,051900000000
19,500000000000	10207891449,401200000000
20,000000000000	10218949998,471400000000
20,500000000000	10230304387,926300000000
21,000000000000	10241955567,355300000000
21,500000000000	10253904515,517300000000
22,000000000000	10266152235,369300000000
22,500000000000	10278699754,197800000000
23,000000000000	10291548128,890500000000
23,500000000000	10304698440,960300000000
24,000000000000	10318151796,685800000000
24,500000000000	10331909332,414700000000
25,000000000000	10345972209,580000000000
25,500000000000	10360341614,851900000000
26,000000000000	10375018765,472900000000
26,500000000000	10390004904,266100000000
27,000000000000	10405301299,797900000000

27,500000000000	10420909251,747600000000
28,000000000000	10436830085,905600000000
28,500000000000	10453065154,348300000000
29,000000000000	10469615840,842700000000
29,500000000000	10486483555,834600000000
30,000000000000	10503669736,635200000000
30,500000000000	10521175852,862900000000
31,000000000000	10539003401,420300000000
31,500000000000	10557153906,692800000000
32,000000000000	10575628926,029500000000
32,500000000000	10594430044,707800000000
33,000000000000	10613558876,144400000000
33,500000000000	10633017067,417300000000
34,000000000000	10652806294,216800000000
34,500000000000	10672928261,069700000000
35,000000000000	10693384706,903400000000
35,500000000000	10714177399,983100000000
36,000000000000	10735308138,148900000000
36,500000000000	10756778754,425200000000
37,000000000000	10778591111,941500000000
37,500000000000	10800747104,183900000000
38,000000000000	10823248660,651000000000
38,500000000000	10846097741,758100000000
39,000000000000	10869296339,103200000000
39,500000000000	10892846481,171200000000
40,000000000000	10916750228,221200000000
40,500000000000	10941009672,566300000000
41,000000000000	10965626944,329600000000
41,500000000000	10990604206,312000000000
42,000000000000	11015943654,288200000000
42,500000000000	11041647522,814800000000
43,000000000000	11067718080,079400000000
43,500000000000	11094157628,211300000000
44,000000000000	11120968509,146200000000
44,500000000000	11148153099,453000000000
45,000000000000	11175713810,662900000000
45,500000000000	11203653095,189600000000
46,000000000000	11231973441,135900000000
46,500000000000	11260677372,639300000000
47,000000000000	11289767455,851900000000
47,500000000000	11319246293,725000000000

48,000000000000	11349116526,371200000000
48,500000000000	11379380837,108200000000
49,000000000000	11410041947,218200000000
49,500000000000	11441102616,329500000000
50,000000000000	11472565648,524400000000
50,500000000000	11504433887,074300000000
51,000000000000	11536710214,840600000000
51,500000000000	11569397560,449300000000
52,000000000000	11602498893,001100000000
52,500000000000	11636017222,491900000000
53,000000000000	11669955606,057500000000
53,500000000000	11704317142,657000000000
54,000000000000	11739104973,514100000000
54,500000000000	11774322288,434600000000
55,000000000000	11809972320,462100000000
55,500000000000	11846058346,340700000000
56,000000000000	11882583692,908600000000
56,500000000000	11919551731,724400000000
57,000000000000	11956965879,553500000000
57,500000000000	11994829604,838700000000
58,000000000000	12033146422,298400000000
58,500000000000	12071919893,435000000000
59,000000000000	12111153633,088700000000
59,500000000000	12150851304,003300000000
60,000000000000	12191016617,360900000000
60,500000000000	12231653339,418800000000
61,000000000000	12272085748,389000000000
61,500000000000	12312310919,023900000000
62,000000000000	12352325929,510700000000
62,500000000000	12392127867,930700000000
63,000000000000	12431713832,641700000000
63,500000000000	12471080926,445000000000
64,000000000000	12510226263,104600000000
64,500000000000	12549146967,729300000000
65,000000000000	12587840170,879800000000
65,500000000000	12626303015,147100000000
66,000000000000	12664532655,533300000000
66,500000000000	12702526253,499900000000
67,000000000000	12740280983,603200000000
67,500000000000	12777794033,873800000000
68,000000000000	12815062599,805900000000

68,500000000000	12852083891,049000000000
69,000000000000	12888855131,784600000000
69,500000000000	12925373554,658000000000
70,000000000000	12961636407,523800000000
70,500000000000	12997640953,820400000000
71,000000000000	13033384466,443400000000
71,500000000000	13068864234,544600000000
72,000000000000	13104077563,902600000000
72,500000000000	13139021770,739700000000
73,000000000000	10949184808,949700000000
73,500000000000	9124320674,124820000000
74,000000000000	7603600561,770680000000
74,500000000000	6336333801,475570000000
75,000000000000	5280278167,896310000000
75,500000000000	4400231806,580250000000
76,000000000000	3666859838,816880000000
76,500000000000	3055716532,347400000000
77,000000000000	2546430443,622830000000
77,500000000000	2122025369,685690000000
78,000000000000	1768354474,738080000000
78,500000000000	1473628728,948400000000
79,000000000000	1228023940,790330000000
79,500000000000	1023353283,991940000000
80,000000000000	852794403,326620000000
80,500000000000	710662002,772183000000
81,000000000000	592218335,643486000000
81,500000000000	493515279,702905000000
82,000000000000	411262733,085754000000
82,500000000000	342718944,238128000000
83,000000000000	285599120,198440000000
83,500000000000	237999266,832033000000
84,000000000000	198332722,360028000000
84,500000000000	165277268,633356000000
85,000000000000	137731057,194464000000
85,500000000000	114775880,995386000000
86,000000000000	95646567,496155500000
86,500000000000	79705472,913462900000
87,000000000000	66421227,427885800000
87,500000000000	55351022,856571500000
88,000000000000	46125852,380476200000
88,500000000000	38438210,317063500000

89,000000000000	32031841,930886200000
89,500000000000	26693201,609071900000
90,000000000000	22244334,674226500000
90,500000000000	18536945,561855500000
91,000000000000	15447454,634879500000
91,500000000000	12872878,862399600000
92,000000000000	10727399,051999700000
92,500000000000	8939499,209999750000
93,000000000000	7449582,674999800000
93,500000000000	6207985,562499830000
94,000000000000	5173321,302083190000
94,500000000000	4311101,085069320000
95,000000000000	3592584,237557770000
95,500000000000	2993820,197964810000
96,000000000000	2494850,164970670000
96,500000000000	2079041,804142230000
97,000000000000	1732534,836785190000
97,500000000000	1443779,030654320000
98,000000000000	1203149,192211930000
98,500000000000	1002624,326843280000
99,000000000000	835520,272369402000
99,500000000000	696266,893641168000
100,000000000000	580222,411367640000
100,500000000000	483518,676139700000
101,000000000000	402932,230116417000
101,500000000000	335776,858430347000
102,000000000000	279814,048691956000
102,500000000000	233178,373909963000
103,000000000000	194315,311591636000
103,500000000000	161929,426326363000
104,000000000000	134941,188605303000
104,500000000000	112450,990504419000
105,000000000000	93709,158753682600
105,500000000000	78090,965628068800
106,000000000000	65075,804690057300
106,500000000000	54229,837241714400
107,000000000000	45191,531034762000
107,500000000000	37659,609195635000
108,000000000000	31383,007663029200
108,500000000000	26152,506385857600
109,000000000000	21793,755321548000

109,500000000000	18161,462767956700
110,000000000000	15134,552306630600
110,500000000000	12612,126922192100
111,000000000000	10510,105768493400
111,500000000000	8758,421473744560
112,000000000000	7298,684561453800
112,500000000000	6082,237134544830
113,000000000000	5068,530945454030
113,500000000000	4223,775787878360
114,000000000000	3519,813156565300
114,500000000000	2933,177630471080
115,000000000000	2444,314692059230
115,500000000000	2036,928910049360
116,000000000000	1697,440758374470
116,500000000000	1414,533965312050
117,000000000000	1178,778304426710
117,500000000000	982,315253688929
118,000000000000	818,596044740774
118,500000000000	682,163370617312
119,000000000000	568,469475514426
119,500000000000	473,724562928689
120,000000000000	394,770469107240

Tabla 2.4 – Valores para *h* = 0,5 seg.

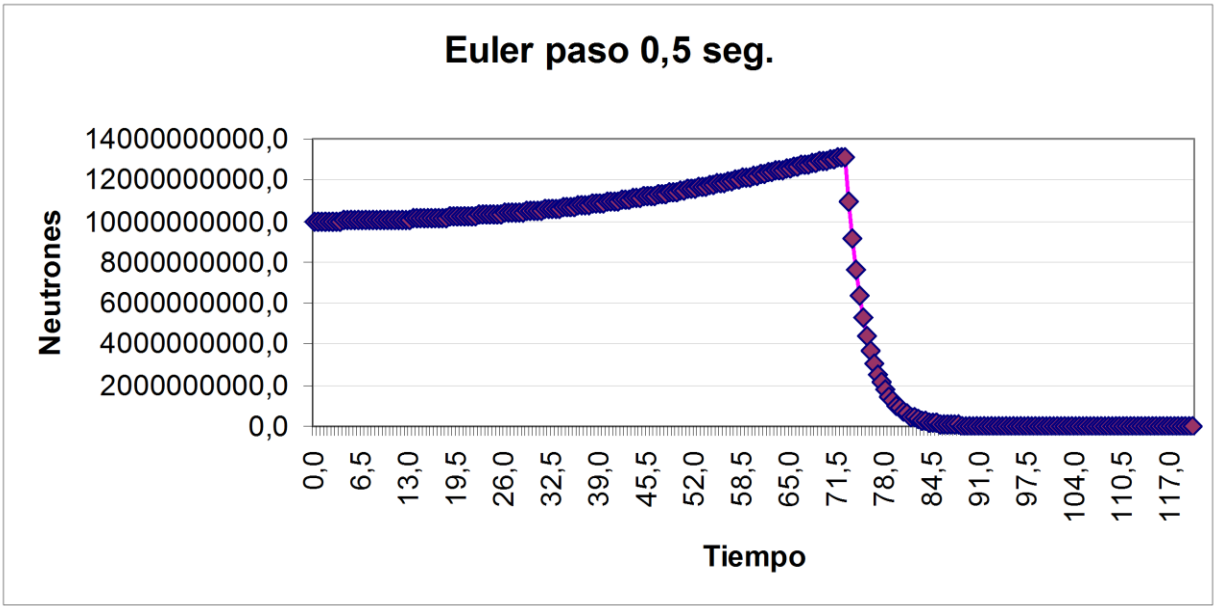


Gráfico 2.4 - Euler paso 0,5 segundos.

2.3. Análisis de resultados

En el *Gráfico2.5* podemos observar como a medida que afinamos el tiempo de paso, los resultados se vuelven cada vez más exactos. Para el tiempo de paso h=4, se puede ver como la función oscila en el intervalo 80-100, situación que ya para h=2 no ocurre. Esto es fundamental en cualquier tipo de resolución numérica, no sólo en la de Euler. Siempre que se disminuya el paso de la resolución, el conjunto de valores arrojados se vuelve cada vez más preciso. Esto puede llegar a parecer evidente, pues lo ideal sería obtener el valor que le corresponde al instante de tiempo infinitesimalmente contiguo al anterior. Como ya sabemos, esto en la práctica es imposible de obtener, por ello, se utilizan pasos medianamente razonables, si bien podemos ser exigentes a la hora de reducir el paso de la resolución hasta el valor que queramos, y así, llegar al resultado con la precisión deseada.

Veremos cómo sucede lo mismo para el método Runge Kutta 4, cuando, al disminuir el paso de la resolución, obtenemos curvas más definidas y precisas. Notaremos también como los valores arrojados son más precisos aún que los de Euler.

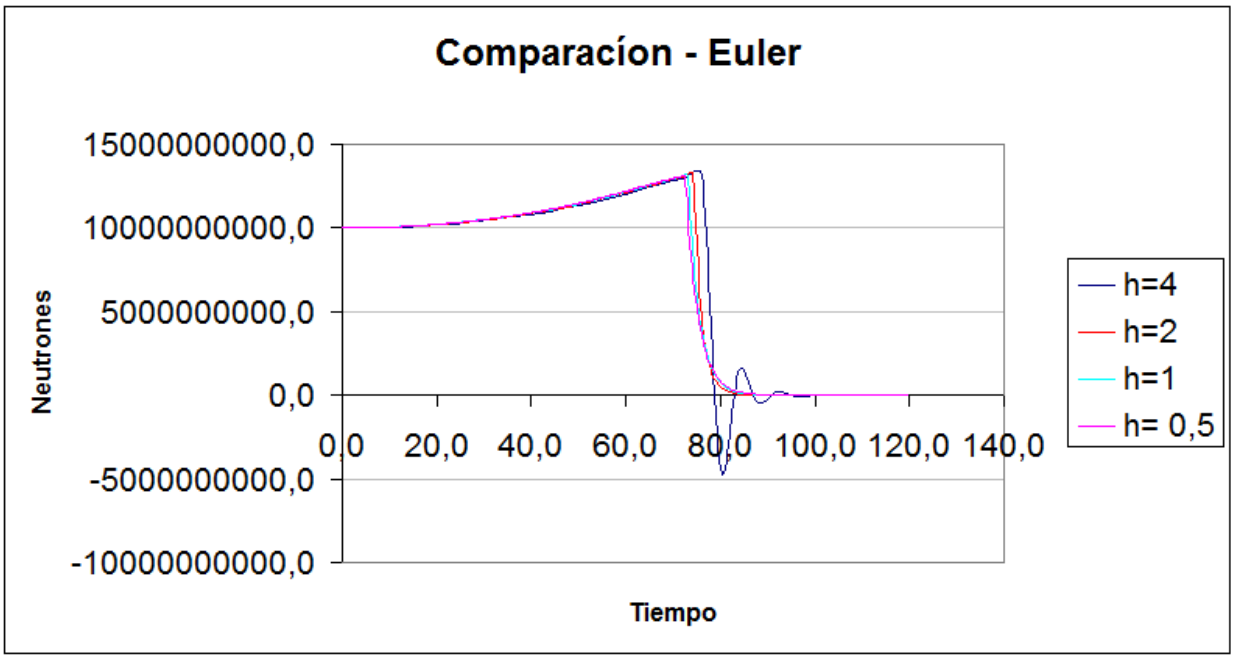


Gráfico 2.5 – Comparación del Método de Euler para los distintos pasos.

2.4. Análisis del error

Para el cálculo del error en los resultados obtenidos con el *Método de Euler*, en el instante de mayor cantidad de neutrones procedemos de la forma siguiente: si N_{h_1} responde a la curva de cantidad de neutrones para el paso h_1, y N_{h_2} a la curva obtenida con el paso h_2, y $h_2 < h_1$, efectuamos

$$\left|N_{h_1} - N_{h_2}\right| = \Delta N$$
$$\frac{\Delta N}{N_{h_2}} = \xi < 1\%$$

Con lo cual obtenemos el paso en el cual podemos garantizar un error menor al 1%.

En este caso, deberemos garantizar como máximo un error del 1% en el instante de mayor cantidad de neutrones, el cual se registra a los 72 segundos, ya que en ese punto debe comenzar a decrecer la cantidad de neutrones, producto del apagado de emergencia hecho por el operador.

Para el caso de la aplicación del Método de Euler y para un paso *h* = 0.5 seg, resulta:

$$\xi = \frac{|\ 13104077563{,}9026 - \ 13082382158{,}8992\ |}{13104077563{,}9026} = 0{,}001655622450$$

Es decir, obtenemos un error de aproximadamente 0.17%.

3. Aplicación del *Método de Runge Kutta 4*

El método de Runge Kutta 4 es un método multipaso e implícito, cuya idea inicial es similar a la de Euler pero aplica varias aproximaciones (4 en total) sobre las tangentes en distintos puntos dentro del paso aplicado. Este método tiene mayor aplicación en la práctica ya que obtiene mejores resultados en menos iteraciones, y demuestra ser mucho más estable que otros métodos.

3.1. Algoritmo e implementación del método

En la implementación del *Método de Runge Kutta 4* se ha mantenido la filosofía de lograr un código reutilizable y modularizado. Es por esto que se utilizará la misma función que representa la ecuación que se plantea en el enunciado del trabajo práctico (*Código 2.2*).

El algoritmo utilizado para el *Método de Runge Kutta 4* puede verse en el *Código 3.1*. Este, al igual que el del método de Euler, es una implementación general del método y es capaz de soportar cualquier tipo de ecuación que se presente. Esto se debe a que la función *f* que recibe como parámetro es nuevamente una *function handle*. En la documentación se explica más detalladamente los requisitos que debe cumplir cada argumento.

Código 3.1 – metodoRK4.m

```
% Función que utiliza el método de Runge-Kutta de orden 4 para la
% resolución de EDO's con un valor inicial.
%
%       metodoRK4(f,a,b,u0,h)
%
% PRECONDICIONES
% 'f': función que se desea procesar, la cual debe pasada como una function
% handler (ej: @miFuncion). Esta debe tener dos parametros: una variable
% independiente y una variable dependiente de la primera.
% 'a': Extremos izquierdo del intervalo;
% 'b': Extremos derecho del intervalo;
% 'u0': Condición inicial de la función;
% 'h': Tamaño de paso.
%
% POSTCONDICIONES
% Se devuelve una matriz de M filas y dos columnas.
% En la primera columna se guardan los instantes de tiempo en los cuales
% fue aplicado el método, en la segunda columna, los resultados obtenidos
% a cada instante
function matrizDeResultados = metodoRK4(f,a,b,u0,h)

    % Se calcula el número de intervalos 'M' dividiendo el largo total
    % del intervalo (a,b) por el tamaño del paso 'h'
    M = (b-a)/h;

    % Se inicializa un vector de tamaño M+1, el cual alberga los instantes
    % de tiempo donde se utilizó el método
    T = zeros(1,M+1);
    T = a:h:b;

    % Se crea el vector de ceros donde se guardarán los resultados obtenidos
    % en cada paso.
    U = zeros(1,M+1);
    % Guardamos la condición inicial en la primera posición
    U(1) = u0;

    % Iteramos para aplicar el método.
    for j=1:M

        % Calculamos los coeficientes necesarios para la utilización del
método
        q1 = h * f(T(j),U(j));
        q2 = h * f(T(j)+h/2,U(j)+q1/2);
        q3 = h * f(T(j)+h/2,U(j)+q2/2);
        q4 = h * f(T(j)+h,U(j)+q3);

        U(j+1) = U(j)+(q1+2*q2+2*q3+q4)/6;

    end

    % Creamos una matriz que esta formada por los vectores T y U
```

```
    % traspuestos.
    matrizDeResultados = [T' U'];
```

En el *Código 3.*2 se muestra el código principal que invoca al método de Runge Kutta 4 y le aplica la ecuación (1) para obtener los resultados para los distintos pasos *h* antes mencionados. Luego de obtener los valores se genera un archivo CSV por cada paso *h* procesado, el cual contiene su tabla correspondiente de valores.

Código 3.2 – tp2MetodoRK4.m

```
% Función que utiliza el Método Runge-Kutta 4 y lo aplica a la función
% propuesta en el TP2.
%
% POST-CONDICIONES
% Se generan cuatro archivos con formato CSV, almacenandose en cada uno los
% valores obtenidos al aplicar el método para distintos pasos h. Estos
% archivos se guardan en la carpeta 'CSV' que se debe encontrar en el mismo
% directorio raíz que este archivo.
function tp2MetodoRK4()

    format longG

    % Especificaciones
    funcion = @tp2Ecuacion;
    a = 0;
    b = 120;
    U0= 10^10;
    h = [4, 2, 1, 0.5];

    % Convertimos el archivo de datos empiricos k.txt en un archivo de formato
    % CSV
    convertirTXTaCSV();

    % Procesamos utilizando el método de RK4
    disp(' ');
    disp('Iniciando aplicación del Método de Runge-Kutta 4.')

    disp(strcat('Procesando para el paso h=', num2str(h(1,1)), '...'));
    rk1 = metodoRK4(funcion, a , b , U0 , h(1,1));

    disp(strcat('Procesando para el paso h=', num2str(h(1,2)), '...'));
    rk2 = metodoRK4(funcion, a , b , U0 , h(1,2));

    disp(strcat('Procesando para el paso h=', num2str(h(1,3)), '...'));
    rk3 = metodoRK4(funcion, a , b , U0 , h(1,3));

    disp(strcat('Procesando para el paso h=', num2str(h(1,4)), '...'));
    rk4 = metodoRK4(funcion, a , b , U0 , h(1,4));

    % Generamos los archivos con formato CSV donde se albergan los
    % valores obtenidos para cada paso
    disp('Generando archivos CSV de datos...');
    dlmwrite('CSV/tp2MetodoRK4ValoresPaso4.csv', rk1,'precision',16);
```

```
    dlmwrite('CSV/tp2MetodoRK4ValoresPaso2.csv', rk2,'precision',16);
    dlmwrite('CSV/tp2MetodoRK4ValoresPaso1.csv', rk3,'precision',16);
    dlmwrite('CSV/tp2MetodoRK4ValoresPaso05.csv', rk4,'precision',16);

    disp('El proceso ha finalizado exitosamente.');
    disp(' ');

end
```

Al igual que en la implementación del método de Euler, el archivo *k.txt* es convertido en un archivo con formato CSV invocando a la función auxiliar *convertirTXTaCSV()* como se puede ver en el código, obteniendo así un algoritmo más rápido y eficiente a la hora de utilizar dicho archivo para la resolución de la ecuación diferencial.

3.2. Resultados obtenidos

Pasaremos ahora a mostrar los resultados obtenidos con el *Método de Runge Kutta 4* para cada paso *h* aplicado.

3.2.1. Resultados para *h* = 4 seg

A continuación, en la *Tabla 3.1*, se muestra la tabla correspondiente a los valores de *N(t)* obtenidos al aplicar un paso de *h* = 4 seg. Además, en el *Gráfico 3.1*, se muestra el grafico correspondiente a dicha tabla.

Tiempo (t)	*N(t)*
0,000000000000	10000000000,000000000000
4,000000000000	10008892842,900900000000
8,000000000000	10035618842,641600000000
12,000000000000	10080320855,033400000000
16,000000000000	10143238394,160100000000
20,000000000000	10224709752,208100000000
24,000000000000	10325175052,963500000000
28,000000000000	10445180216,960500000000
32,000000000000	10585381887,168400000000
36,000000000000	10746553440,233300000000
40,000000000000	10929592098,465100000000
44,000000000000	11135527232,177000000000
48,000000000000	11365530025,484100000000

52,000000000000	11620924568,412000000000
56,000000000000	11903200519,367100000000
60,000000000000	12214027576,853700000000
64,000000000000	12532971229,495400000000
68,000000000000	12837401083,701200000000
72,000000000000	13125870004,622900000000
76,000000000000	3725370380,571330000000
80,000000000000	1088482703,788330000000
84,000000000000	318034041,024574000000
88,000000000000	92923526,389896100000
92,000000000000	27150495,364949000000
96,000000000000	7932860,785643550000
100,000000000000	2317831,752183920000
104,000000000000	677226,561337688000
108,000000000000	197872,781296197000
112,000000000000	57814,680954855900
116,000000000000	16892,355340719200
120,000000000000	4935,626457576400

Tabla 3.1 – Valores para *h* = 4 seg.

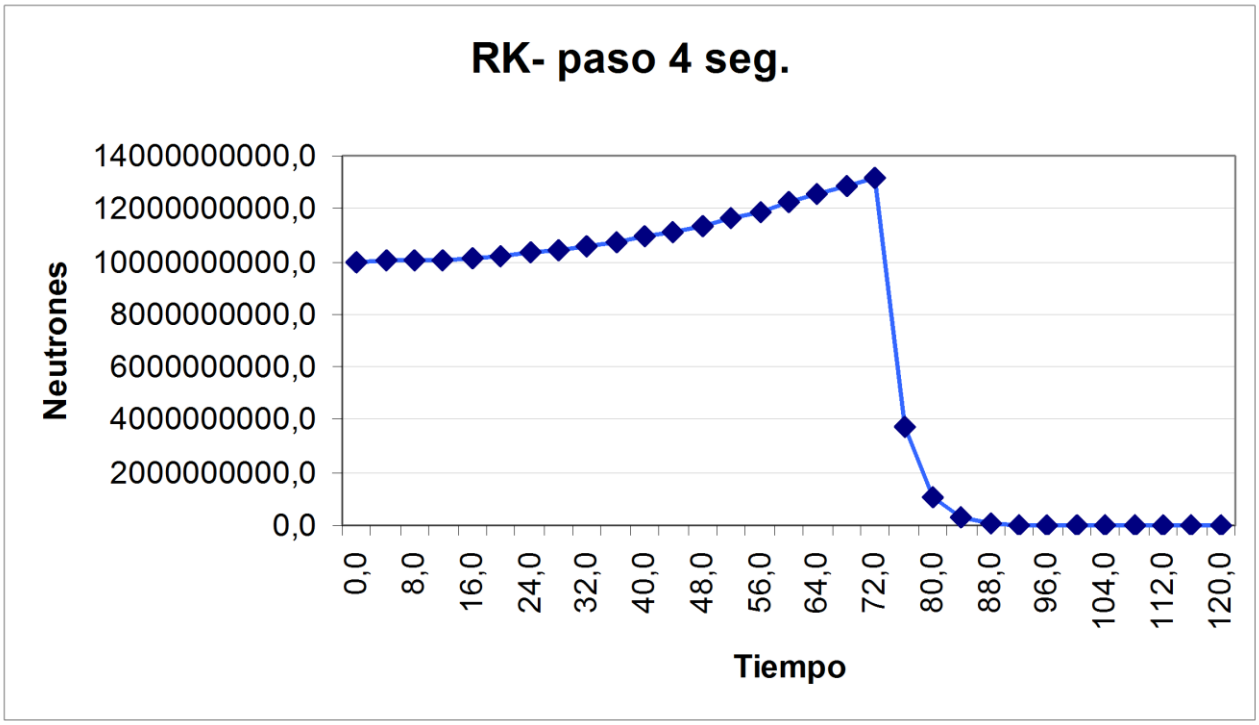


Gráfico 3.1 - RK-4 paso 4 segundos.

3.2.2. Resultados para *h* = 2 seg

A continuación, en la *Tabla 3.2*, se muestra la tabla correspondiente a los valores de *N(t)* obtenidos al aplicar un paso de *h* = 2 seg. Además, en el *Gráfico 3.2*, se muestra el grafico correspondiente a dicha tabla.

Tiempo (t)	*N(t)*
0,000000000000	10000000000,000000000000
2,000000000000	10002222468,042700000000
4,000000000000	10008892839,564800000000
6,000000000000	10020020013,339900000000
8,000000000000	10035618839,298900000000
10,000000000000	10055710161,602700000000
12,000000000000	10080320855,042400000000
14,000000000000	10109483884,929500000000
16,000000000000	10143238390,801300000000
18,000000000000	10181629763,896700000000
20,000000000000	10224709748,845600000000
22,000000000000	10272536570,125700000000
24,000000000000	10325175053,048000000000
26,000000000000	10382696770,125200000000
28,000000000000	10445180213,624800000000
30,000000000000	10512710963,752500000000
32,000000000000	10585381883,881600000000
34,000000000000	10663293343,888600000000
36,000000000000	10746553440,620900000000
38,000000000000	10835278247,616400000000
40,000000000000	10929592095,429400000000
42,000000000000	11029627851,048700000000
44,000000000000	11135527229,399500000000
46,000000000000	11247441138,613700000000
48,000000000000	11365530026,897600000000
50,000000000000	11489964265,043700000000
52,000000000000	11620924566,645200000000
54,000000000000	11758602413,070300000000
56,000000000000	11903200518,495900000000
58,000000000000	12054933337,495900000000
60,000000000000	12214027581,345300000000
62,000000000000	12375221461,726700000000
64,000000000000	12532971239,630500000000
66,000000000000	12687091927,867200000000

68,000000000000	12837401095,094700000000
70,000000000000	12983719210,661600000000
72,000000000000	13125870012,644300000000
74,000000000000	7465433097,561910000000
76,000000000000	3840243362,943370000000
78,000000000000	1975433828,674570000000
80,000000000000	1016169664,956050000000
82,000000000000	522721021,067930000000
84,000000000000	268889414,129594000000
86,000000000000	138317599,860902000000
88,000000000000	71151028,735032000000
90,000000000000	36600323,423370400000
92,000000000000	18827326,863873600000
94,000000000000	9684838,921745710000
96,000000000000	4981913,025589360000
98,000000000000	2562712,461722920000
100,000000000000	1318267,727223720000
102,000000000000	678121,258859941000
104,000000000000	348827,808055525000
106,000000000000	179438,172868068000
108,000000000000	92303,586866290400
110,000000000000	47481,268964141100
112,000000000000	24424,521072089000
114,000000000000	12564,054049428500
116,000000000000	6462,990766166940
118,000000000000	3324,583727452130
120,000000000000	1710,176814532990

Tabla 3.2 – Valores para h = 2 seg.

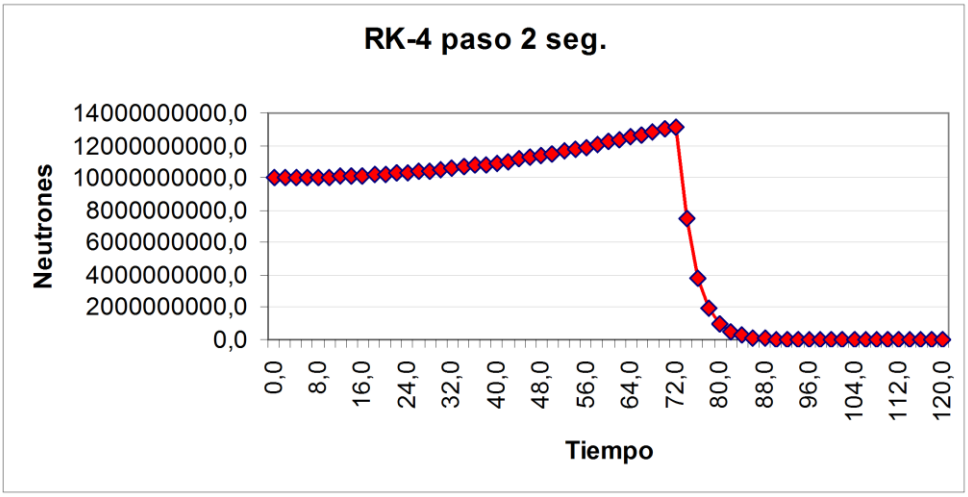


Gráfico 3.2 – RK-4 paso 2 segundos.

3.2.3. Resultados para *h* = 1 seg

A continuación, en la *Tabla 3.3*, se muestra la tabla correspondiente a los valores de *N(t)* obtenidos al aplicar un paso de *h* = 1 seg. Además, en el *Gráfico 3.3*, se muestra el grafico correspondiente a dicha tabla.

Tiempo (t)	*N(t)*
0,000000000000	10000000000,000000000000
1,000000000000	10000555571,543500000000
2,000000000000	10002222469,709700000000
3,000000000000	10005001250,208300000000
4,000000000000	10008892841,233000000000
5,000000000000	10013898538,973800000000
6,000000000000	10020020013,340000000000
7,000000000000	10027259308,893000000000
8,000000000000	10035618840,971600000000
9,000000000000	10045101402,046000000000
10,000000000000	10055710163,278800000000
11,000000000000	10067448671,267500000000
12,000000000000	10080320855,042700000000
13,000000000000	10094331028,269900000000
14,000000000000	10109483886,614900000000
15,000000000000	10125784515,406300000000
16,000000000000	10143238392,492600000000
17,000000000000	10161851386,243100000000
18,000000000000	10181629763,897900000000
19,000000000000	10202580195,100700000000
20,000000000000	10224709750,551300000000
21,000000000000	10248025911,071900000000
22,000000000000	10272536571,840000000000
23,000000000000	10298250041,714500000000
24,000000000000	10325175053,051000000000
25,000000000000	10353320766,665600000000
26,000000000000	10382696771,859800000000
27,000000000000	10413313097,024900000000
28,000000000000	10445180215,371200000000
29,000000000000	10478309045,683800000000
30,000000000000	10512710963,759900000000
31,000000000000	10548397808,944600000000
32,000000000000	10585381885,655500000000

33,000000000000	10623675975,702700000000
34,000000000000	10663293345,678500000000
35,000000000000	10704247749,293600000000
36,000000000000	10746553440,637300000000
37,000000000000	10790225182,474600000000
38,000000000000	10835278249,443400000000
39,000000000000	10881728442,318800000000
40,000000000000	10929592097,278100000000
41,000000000000	10978886090,015700000000
42,000000000000	11029627851,083200000000
43,000000000000	11081835376,189700000000
44,000000000000	11135527231,299200000000
45,000000000000	11190722569,125100000000
46,000000000000	11247441140,543400000000
47,000000000000	11305703300,743400000000
48,000000000000	11365530026,966700000000
49,000000000000	11426942931,116600000000
50,000000000000	11489964267,044800000000
51,000000000000	11554616949,631900000000
52,000000000000	11620924568,688800000000
53,000000000000	11688911397,469400000000
54,000000000000	11758602413,202200000000
55,000000000000	11830023312,388000000000
56,000000000000	11903200520,641900000000
57,000000000000	11978161214,794000000000
58,000000000000	12054933339,703300000000
59,000000000000	12133545619,539100000000
60,000000000000	12214027581,586900000000
61,000000000000	12295043381,510200000000
62,000000000000	12375221459,950200000000
63,000000000000	12454538479,538300000000
64,000000000000	12532971237,866800000000
65,000000000000	12610496684,951300000000
66,000000000000	12687091928,226900000000
67,000000000000	12762734243,675000000000
68,000000000000	12837401093,345100000000
69,000000000000	12911070130,154200000000
70,000000000000	12983719208,914100000000
71,000000000000	13055326403,838900000000
72,000000000000	13125870013,083900000000
73,000000000000	9933744851,383490000000

74,000000000000	7118161819,947120000000
75,000000000000	5100616983,120540000000
76,000000000000	3654917416,402730000000
77,000000000000	2618981461,444960000000
78,000000000000	1876667271,498370000000
79,000000000000	1344751805,142610000000
80,000000000000	963600444,734390000000
81,000000000000	690481182,878089000000
82,000000000000	494773810,570565000000
83,000000000000	354536994,920163000000
84,000000000000	254048371,359973000000
85,000000000000	182041862,810927000000
86,000000000000	130444606,427788000000
87,000000000000	93471881,046249900000
88,000000000000	66978564,967811800000
89,000000000000	47994414,094733400000
90,000000000000	34391059,070969000000
91,000000000000	24643387,492726200000
92,000000000000	17658559,041855800000
93,000000000000	12653483,922482000000
94,000000000000	9067028,345688020000
95,000000000000	6497104,159230150000
96,000000000000	4655589,554427770000
97,000000000000	3336026,877221130000
98,000000000000	2390476,049366790000
99,000000000000	1712928,568296260000
100,000000000000	1227422,580060030000
101,000000000000	879526,570999806000
102,000000000000	630236,889610457000
103,000000000000	451604,931701321000
104,000000000000	323603,739639887000
105,000000000000	231882,720842779000
106,000000000000	166158,760356991000
107,000000000000	119063,350399840000
108,000000000000	85316,485137333900
109,000000000000	61134,703598922900
110,000000000000	43806,914667335200
111,000000000000	31390,448627365200
112,000000000000	22493,258712921600
113,000000000000	16117,854622993700
114,000000000000	11549,470930982600

115,000000000000	8275,932616696950
116,000000000000	5930,233608569370
117,000000000000	4249,390646469710
118,000000000000	3044,959449862300
119,000000000000	2181,907671634870
120,000000000000	1563,476021907080

Tabla 3.3 – Valores para *h* = 1 seg.

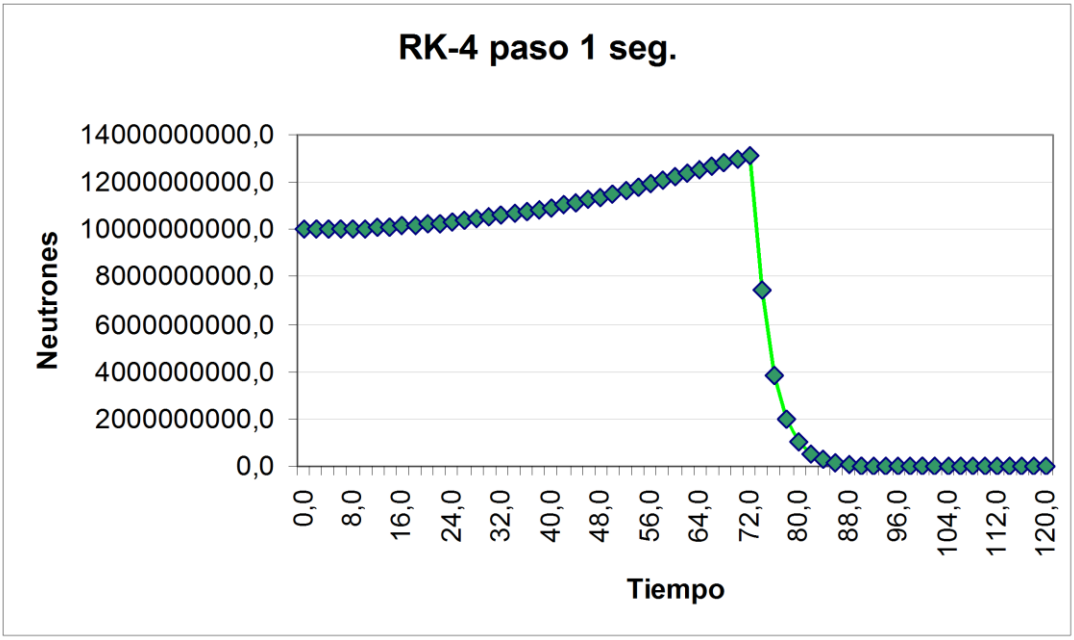


Gráfico 3.3 – RK-4 paso 1 segundo.

3.2.4. Resultados para *h* = 0,5 seg

A continuación, en la *Tabla 3.4*, se muestra la tabla correspondiente a los valores de *N(t)* obtenidos al aplicar un paso de *h* = 0,5 seg. Además, en el *Gráfico 3.4*, se muestra el grafico correspondiente a dicha tabla.

Tiempo (t)	*N(t)*
0,000000000000	10000000000,000000000000
0,500000000000	10000138889,575600000000
1,000000000000	10000555570,710100000000
1,500000000000	10001250078,128200000000
2,000000000000	10002222468,876200000000
2,500000000000	10003472824,830400000000

3,000000000000	10005001250,208300000000
3,500000000000	10006807871,582300000000
4,000000000000	10008892840,398900000000
4,500000000000	10011256330,498700000000
5,000000000000	10013898538,139300000000
5,500000000000	10016819684,526000000000
6,000000000000	10020020013,340000000000
6,500000000000	10023499790,770400000000
7,000000000000	10027259308,057400000000
7,500000000000	10031298879,027300000000
8,000000000000	10035618840,135300000000
8,500000000000	10040219553,019900000000
9,000000000000	10045101402,046000000000
9,500000000000	10050264794,356500000000
10,000000000000	10055710162,440800000000
10,500000000000	10061437961,684100000000
11,000000000000	10067448670,428500000000
11,500000000000	10073742792,555800000000
12,000000000000	10080320855,042700000000
12,500000000000	10087183408,031700000000
13,000000000000	10094331027,428700000000
13,500000000000	10101764312,463900000000
14,000000000000	10109483885,772500000000
14,500000000000	10117490396,007500000000
15,000000000000	10125784515,406300000000
15,500000000000	10134366939,880800000000
16,000000000000	10143238391,647300000000
16,500000000000	10152399616,796900000000
17,000000000000	10161851385,396300000000
17,500000000000	10171594494,134500000000
18,000000000000	10181629763,897900000000
18,500000000000	10191958039,880700000000
19,000000000000	10202580194,250500000000
19,500000000000	10213497123,726100000000
20,000000000000	10224709749,699300000000
20,500000000000	10236219020,918900000000
21,000000000000	10248025911,071900000000
21,500000000000	10260131418,916200000000
22,000000000000	10272536570,984100000000
22,500000000000	10285242419,166500000000
23,000000000000	10298250040,856400000000

23,500000000000	10311560541,673700000000
24,000000000000	10325175053,051100000000
24,500000000000	10339094732,389200000000
25,000000000000	10353320765,802900000000
25,500000000000	10367854365,709400000000
26,000000000000	10382696770,994700000000
26,500000000000	10397849249,782700000000
27,000000000000	10413313097,025100000000
27,500000000000	10429089634,679100000000
28,000000000000	10445180214,501000000000
28,500000000000	10461586215,636300000000
29,000000000000	10478309044,810800000000
29,500000000000	10495350139,147900000000
30,000000000000	10512710963,760200000000
30,500000000000	10530393011,952600000000
31,000000000000	10548397808,065900000000
31,500000000000	10566726905,068900000000
32,000000000000	10585381884,773800000000
32,500000000000	10604364360,707500000000
33,000000000000	10623675975,703200000000
33,500000000000	10643318402,129700000000
34,000000000000	10663293344,790400000000
34,500000000000	10683602538,515000000000
35,000000000000	10704247748,402200000000
35,500000000000	10725230772,748200000000
36,000000000000	10746553440,638000000000
36,500000000000	10768217612,202300000000
37,000000000000	10790225181,576300000000
37,500000000000	10812578074,490300000000
38,000000000000	10835278248,541400000000
38,500000000000	10858327696,184200000000
39,000000000000	10881728442,319900000000
39,500000000000	10905482544,583900000000
40,000000000000	10929592096,368600000000
40,500000000000	10954059224,412600000000
41,000000000000	10978886089,102300000000
41,500000000000	11004074887,531000000000
42,000000000000	11029627851,084900000000
42,500000000000	11055547245,762500000000
43,000000000000	11081835375,268200000000
43,500000000000	11108494578,597700000000

44,000000000000	11135527230,373400000000
44,500000000000	11162935743,976000000000
45,000000000000	11190722569,127600000000
45,500000000000	11218890192,244800000000
46,000000000000	11247441139,608900000000
46,500000000000	11276377974,947400000000
47,000000000000	11305703299,804500000000
47,500000000000	11335419756,752400000000
48,000000000000	11365530026,970300000000
48,500000000000	11396036830,633900000000
49,000000000000	11426942930,168500000000
49,500000000000	11458251127,825400000000
50,000000000000	11489964266,092000000000
50,500000000000	11522085230,987300000000
51,000000000000	11554616949,637200000000
51,500000000000	11587562390,703500000000
52,000000000000	11620924567,726300000000
52,500000000000	11654706536,695400000000
53,000000000000	11688911396,501900000000
53,500000000000	11723542292,326600000000
54,000000000000	11758602413,209500000000
54,500000000000	11794094992,523100000000
55,000000000000	11830023311,410400000000
55,500000000000	11866390696,351300000000
56,000000000000	11903200519,659100000000
56,500000000000	11940456202,969400000000
57,000000000000	11978161214,804200000000
57,500000000000	12016319071,092000000000
58,000000000000	12054933338,710000000000
58,500000000000	12094007633,045400000000
59,000000000000	12133545618,540500000000
59,500000000000	12173551012,290300000000
60,000000000000	12214027581,600800000000
60,500000000000	12254638733,882300000000
61,000000000000	12295043382,550000000000
61,500000000000	12335238598,368500000000
62,000000000000	12375221460,998000000000
62,500000000000	12414989056,257400000000
63,000000000000	12454538479,556000000000
63,500000000000	12493866836,260900000000
64,000000000000	12532971238,930100000000

64,500000000000	12571848810,774100000000
65,000000000000	12610496686,022100000000
65,500000000000	12648912007,125500000000
66,000000000000	12687091928,247700000000
66,500000000000	12725033615,629000000000
67,000000000000	12762734244,760300000000
67,500000000000	12800191003,900100000000
68,000000000000	12837401094,437500000000
68,500000000000	12874361728,036200000000
69,000000000000	12911070130,177600000000
69,500000000000	12947523540,521600000000
70,000000000000	12983719210,020300000000
70,500000000000	13019654404,486500000000
71,000000000000	13055326404,951700000000
71,500000000000	13090732504,749700000000
72,000000000000	13125870013,109400000000
72,500000000000	11424240056,919200000000
73,000000000000	9670422339,847800000000
73,500000000000	8185845865,028700000000
74,000000000000	6929177462,073710000000
74,500000000000	5865429314,523490000000
75,000000000000	4964984838,673130000000
75,500000000000	4202774106,784490000000
76,000000000000	3557575857,045040000000
76,500000000000	3011426657,026070000000
77,000000000000	2549120770,731710000000
77,500000000000	2157786804,674490000000
78,000000000000	1826529346,073640000000
78,500000000000	1546125615,765590000000
79,000000000000	1308768690,119990000000
79,500000000000	1107850142,816650000000
80,000000000000	937776054,855318000000
80,500000000000	793811270,199513000000
81,000000000000	671947560,863007000000
81,500000000000	568792030,927280000000
82,000000000000	481472652,465418000000
82,500000000000	407558303,329539000000
83,000000000000	344991080,515800000000
83,500000000000	292029004,594280000000
84,000000000000	247197519,996232000000
84,500000000000	209248440,843004000000

85,000000000000	177125199,297693000000
85,500000000000	149933428,893678000000
86,000000000000	126916063,829142000000
86,500000000000	107432260,948993000000
87,000000000000	90939557,565781100000
87,500000000000	76978768,362572300000
88,000000000000	65161200,881499700000
88,500000000000	55157833,655124900000
89,000000000000	46690155,681127300000
89,500000000000	39522412,195486200000
90,000000000000	33455040,853104300000
90,500000000000	28319115,567817100000
91,000000000000	23971643,318706800000
91,500000000000	20291582,977695200000
92,000000000000	17176475,315706500000
92,500000000000	14539590,360958000000
93,000000000000	12307512,686910400000
93,500000000000	10418097,400130700000
94,000000000000	8818739,919240020000
94,500000000000	7464911,372610290000
95,000000000000	6318918,837752580000
95,500000000000	5348855,905323680000
96,000000000000	4527714,349642080000
96,500000000000	3832632,173087910000
97,000000000000	3244257,088645560000
97,500000000000	2746207,718844810000
98,000000000000	2324617,509949360000
98,500000000000	1967748,663176980000
99,000000000000	1665665,334130230000
99,500000000000	1409957,001746230000
100,000000000000	1193504,304879650000
100,500000000000	1010280,827005410000
101,000000000000	855185,310385335000
101,500000000000	723899,628251526000
102,000000000000	612768,560707126000
102,500000000000	518698,027098055000
103,000000000000	439068,941469415000
103,500000000000	371664,292693809000
104,000000000000	314607,419056562000
104,500000000000	266309,758755794000
105,000000000000	225426,621601122000

105,500000000000	190819,750518774000
106,000000000000	161525,630510828000
106,500000000000	136728,662735326000
107,000000000000	115738,456827366000
107,500000000000	97970,609240217800
108,000000000000	82930,432442312700
108,500000000000	70199,181962887500
109,000000000000	59422,397823458900
109,500000000000	50300,035760476100
110,000000000000	42578,113475359300
110,500000000000	36041,639329113100
111,000000000000	30508,626604173700
111,500000000000	25825,026680211200
112,000000000000	21860,440054760800
112,500000000000	18504,485796096900
113,000000000000	15663,728347654100
113,500000000000	13259,076120929300
114,000000000000	11223,579449201000
114,500000000000	9500,566593300360
115,000000000000	8042,065902617190
115,500000000000	6807,470201582050
116,000000000000	5762,406215838920
116,500000000000	4877,777560983250
117,000000000000	4128,954649020310
117,500000000000	3495,088958142230
118,000000000000	2958,532573910970
118,500000000000	2504,346840872610
119,000000000000	2119,886444615970
119,500000000000	1794,447344402450
120,000000000000	1518,968754204350

Tabla 3.4 – Valores para *h* = 0,5 seg.

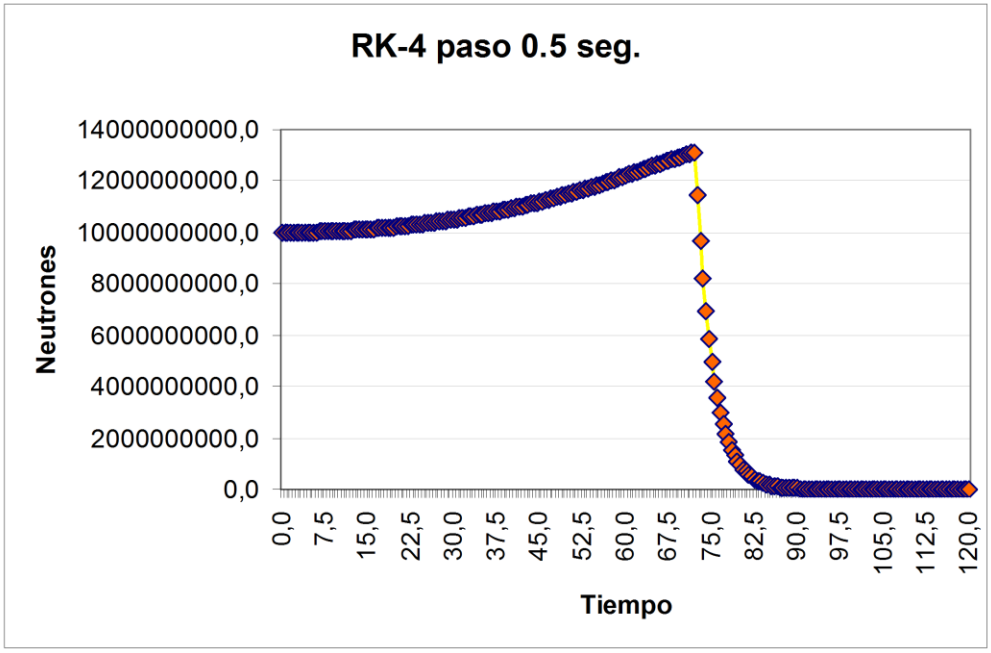


Gráfico 3.4 - RK-4 paso 0,5 segundos.

3.3. Análisis de resultados

En el *Gráfico 3.5* notamos un alto nivel de coincidencia entre todos los pasos, sobre todo entre los últimos. Podemos ver que Runge Kutta 4 es de por sí muy preciso desde el comienzo.

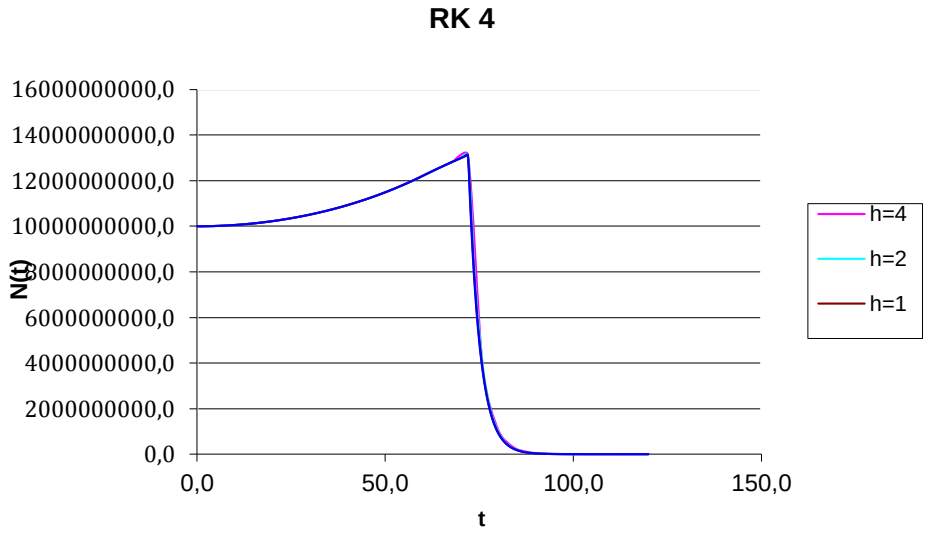


Grafico 3.5 - *RK – 4.*

Paso *h* = 4

Para comparar entre ambos métodos debemos hacer un análisis "*ceteris paribus*", es decir mantener todas las variables constantes excepto el método a evaluar.

En el caso de *h*=4 seg, Euler no logra converger a una única solución luego del salto en 72 segundos, sino que oscila entre valores positivos y negativos. Sin embargo para ese mismo paso el método Runge Kutta 4 ya se aproxima bastante bien a la solución buscada. Esto último se puede observar en el *Gráfico 3.6*.

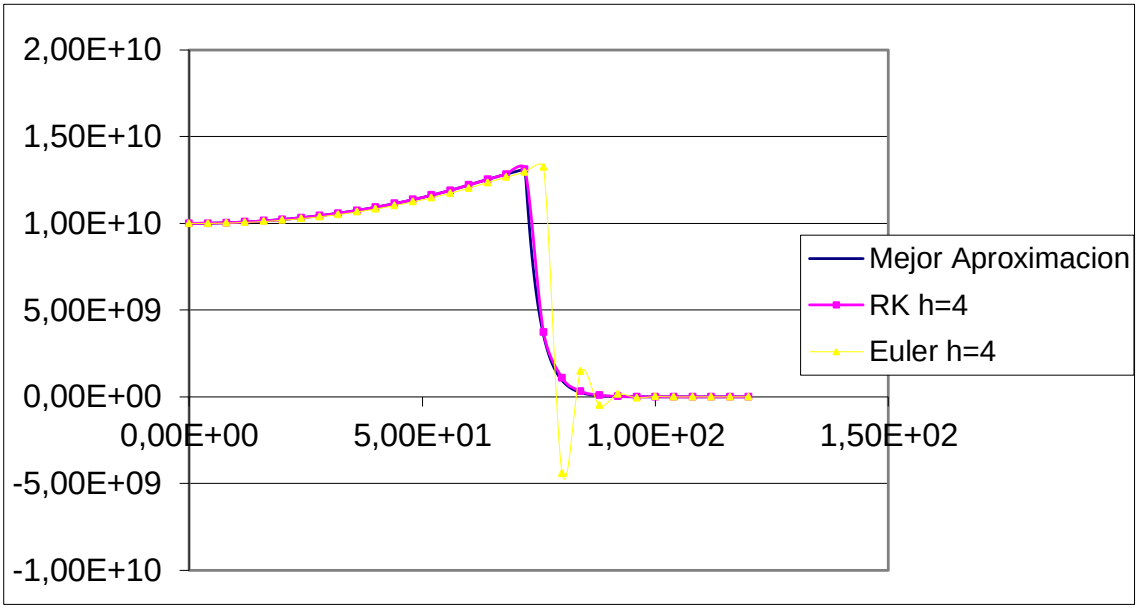


Grafico 3.6 - Comparación de Euler y RK-4 para el paso *h* = 4.

Paso *h* = 2

Este es un paso intermedio, en el cual Euler logra converger a una forma de las características de la solución esperada, pero aún conserva variaciones respecto a la mejor aproximación disponible. Esto se puede ver en el *Gráfico 3.7*.

Notamos que en los entornos con gran variación de pendiente Euler varia más notoriamente con respecto a las soluciones esperadas. Las soluciones por RK-4 se conservan muy similares a pesar del cambio de paso.

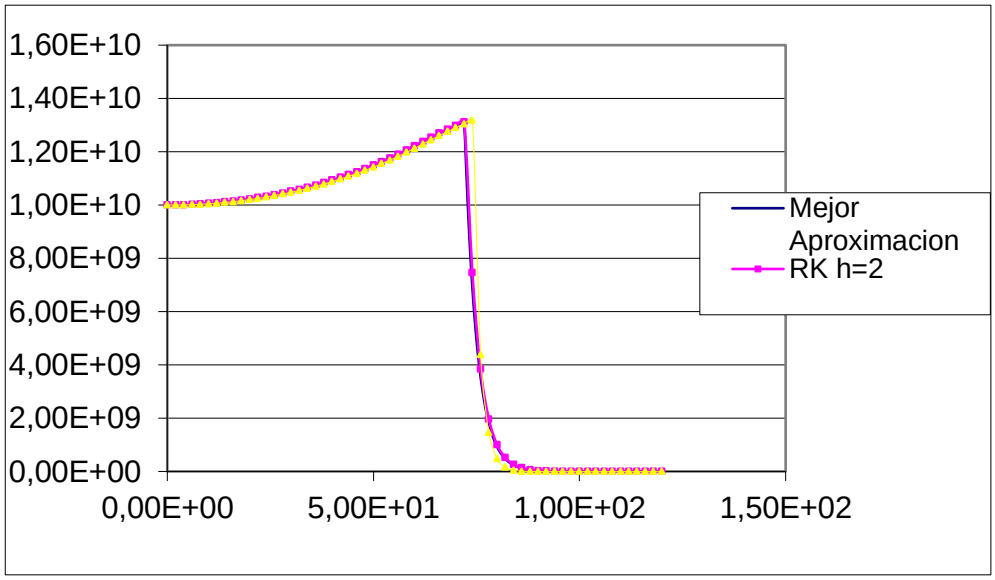


Grafico 3.7 - Comparación de Euler y RK-4 para el paso h = 2.

Paso *h* = 1

En este caso notamos como los dos métodos comienzan a semejarse (*véase* el *Gráfico 3.8*).

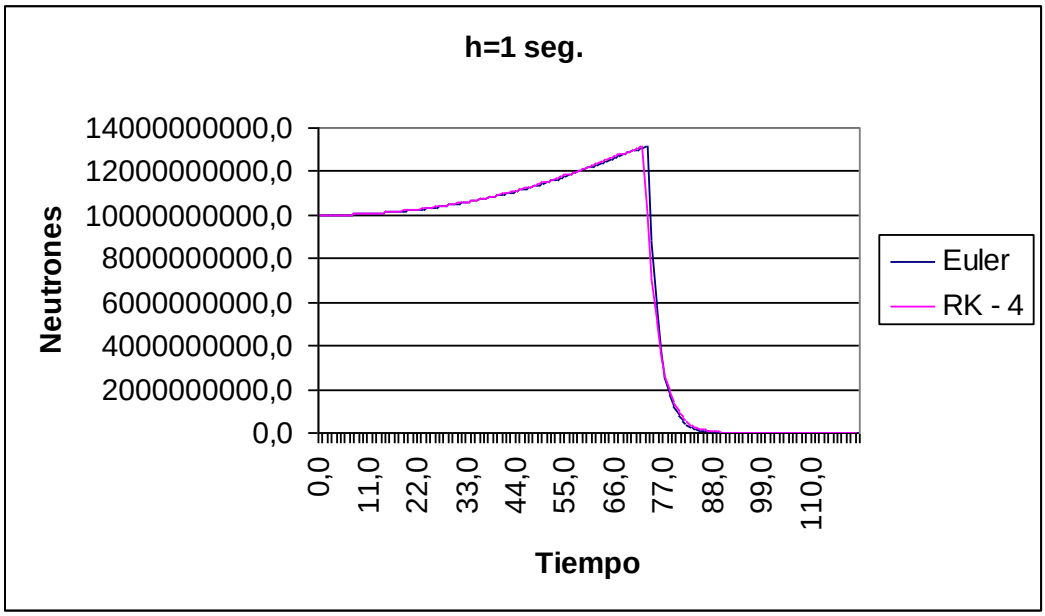


Grafico 3.8 - Comparación de Euler y RK-4 para el paso h = 1.

Paso h = 0.5

Notamos para este paso una gran similitud entre ambos métodos, como se puede apreciar en el *Gráfico 3.9*. En este observamos un alto nivel de coincidencia entre todos los pasos, sobre todo entre los últimos.

No disponemos de la función real, sin embrago podemos comparar nuestras aproximaciones entre sí. Sabiendo que el método converge a la función real en el infinito, comparar nuestras sucesivas aproximaciones nos da una cota del error que cometemos con nuestra aproximación.

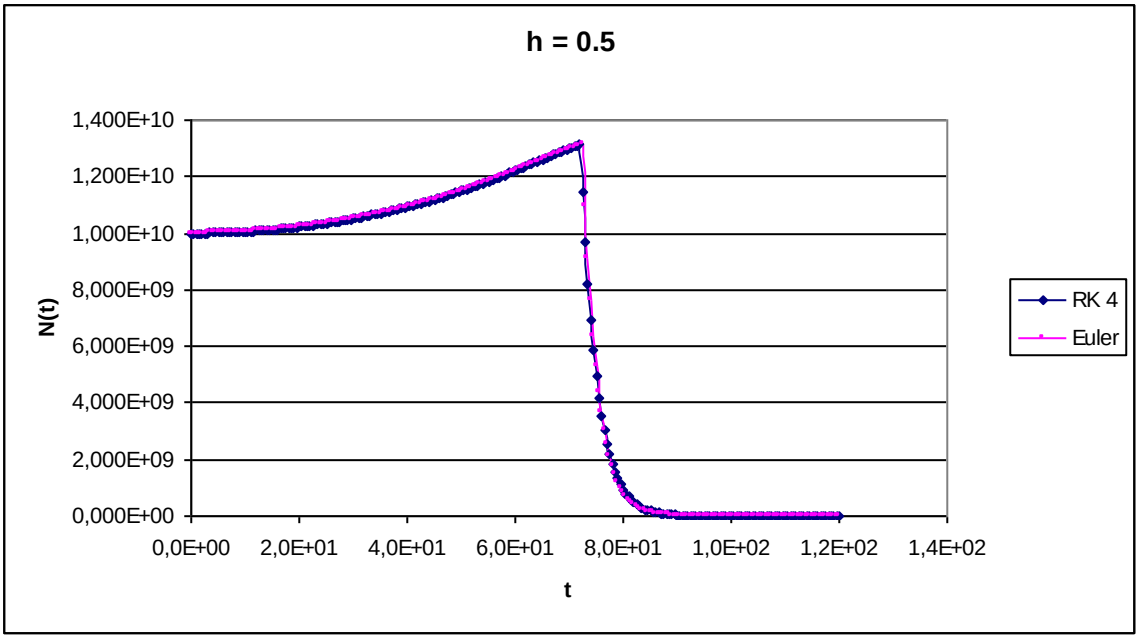


Grafico 3.9-- Comparación de Euler y RK-4 para el paso h = 0,5.

3.4. Análisis del error

No disponemos de la función real, sin embrago podemos comparar nuestras aproximaciones entre sí. Sabiendo que el método converge a la función real en el infinito, comparar nuestras sucesivas aproximaciones nos da una cota del error que cometemos con nuestra aproximación.

Si N_{h_1} responde a la curva de cantidad de neutrones para el paso h_1, y N_{h_2} a la curva obtenida con el paso h_2, y $h_2 < h_1$, entonces efectuamos,

$$\left|N_{h_1} - N_{h_2}\right| = \Delta N$$

$$\frac{\Delta N}{N_{h_2}} = \xi$$

Entonces podemos asegurar cuantos dígitos significativos tiene al menos nuestra solución más aproximada.[2] Comparando entre los pasos h = 1 y h = 0.5, en el momento de mayores neutrones, t = 72 seg:

$$\xi = \frac{|13125870013{,}1094 - 13125870013{,}0839|}{13125870013{,}1094} = 0{,}000000000002$$

Claramente el error tiende a 0%, lo cual evidencia el grado de igualdad que tienen ambos valores, ya que a pesar de ser para distintos valores de paso estos números de orden de magnitud 10 solo difieren en los decimales. Esto puede deberse a la cantidad de dígitos significativos utilizados en la aplicación del método, ya que durante todo el proceso fueron utilizadas 16 cifras significativas de precisión, tanto para el cálculo de los resultados, como para el pasaje de estos a archivos CSV.

Teniendo tal nivel de precisión, procedemos hacer la integración por *método de Romberg* (apartado 4) con esta última curva.

Como podemos observar en los gráficos previamente presentados, ambos métodos afinan su precisión a medida que reducimos el paso de la solución, si bien, para este caso, ya desde un comienzo Runge Kutta es preciso, a diferencia de Euler.

4. Aplicación del *Método de Romberg*

El *Método de Romberg* de integración consiste en una primera aproximación de la integral por trapecios, la cual se va mejorando con sucesivos pasos de la mitad de magnitud, y finalmente se le aplica extrapolación de Richardson a estos resultados para obtener una aproximación de mayor orden.

Para efectuar la integración pueden utilizarse directamente los puntos obtenidos, o interpolar la función y luego aplicar el método. En nuestro caso optamos por la primera opción utilizando los valores obtenidos para el paso h = 0.5 del método de Runge Kutta 4, ya

[2] Luego si aplicamos $\xi < 1\%$ obtenemos el paso en el cual podemos garantizar un error menor al 1%. Sobre esta curva trabajaremos para obtener el número total de neutrones durante toda la actividad del proceso.

que, además de asegurar la mejor precisión en los valores a causa del mínimo error que nos proporciona, con esta obtenemos la cantidad de puntos requerida para la integración.

La integración se realizará entre 0 y 72 segundos, que es lo que dura el período de estado crítico en donde aumenta la cantidad de neutrones en el núcleo del reactor.

4.1. Algoritmo e implementación del método.

Para realizar la implementación utilizaremos nuevamente el software *Matlab*. En el *Código 4.1* se muestra la implementación del algoritmo del Método de Romberg. Este, al igual que el del método de Euler y de Runge Kutta 4, es una implementación general del método y es capaz de soportar cualquier tipo de ecuación que se presente. Esto se debe a que la función *f* que recibe como parámetro es nuevamente una *function handle*. En la documentación se explica más detalladamente los requisitos que debe cumplir cada argumento.

Código 4.1 – metodoDeRomberg.m

```
% Función que utiliza el Método de Romberg para calcular la integral de
% una función entre un intervalo predeterminado.
%
%       metodoDeRomberg(f, a, b, K)
%
% PRE-CONDICIONES
% 'f': función que se desea procesar, la cual debe pasada como una function
% handler (ej: @miFuncion). Esta debe tener dos parametros: una variable
% independiente y una variable dependiente de la primera.
% 'a': extremo izquierdo del intervalo a integrar;
% 'b': extremo derecho del intervalo a integrar;
% 'K': número de paso máximo.
%
% POST-CONDICIONES
% Devuelve el resultado de la integración con una precisión de 15-16
% dígitos significativos.
function resultado = metodoDeRomberg(f, a, b, K)

    format longG;

    % Creamos la tabla en la cual se iran almacenando los valores
    % calculados en las iteraciones
    R = zeros(1, K);

    % Llenamos la primer columna de la tabla aplicando Romberg
    for k = 1:K

        hk = (b-a)/(2^(k-1));
        r = 0;
```

```
        for i = 1:((2^(k-1))-1)
           r = r + f(a+i*hk);
        end

        R(k,1) = (hk/2) * (f(a) + f(b) + 2*r);

    end

    % Calculamos los demas valores usando la extrapolación de Richardson
    for k = 2:K
        for j =2:k

           R(k,j) = R(k,j-1) + (R(k,j-1) - R(k-1,j-1)) / ((4^(j-1)) - 1);

        end
    end

    resultado = R(K,K);
```

A diferencia de los otros dos métodos, la función *f* que se pasará por parámetro a esta última será diferente, ya que en este caso los valores a utilizar se tomarán del archivo con formato CSV (*tp2MetodoRK4ValoresPaso05.csv*) creado al procesar el método de Runge Kutta 4 para el paso h = 0,5. Además, esta solo recibe por parámetro el tiempo en el que se desea saber la cantidad de neutrones en el núcleo. El código fuente de dicha función puede verse en el *Código 4.2*.

Código 4.2 – tp2EcuacionRomberg.m

```
% Función que modela la ecuación que rige la cantidad de neutrones
% en un reactor nuclear, a través de una tabla de valores en el tiempo
%
%            tp2EcuacionRomberg(t)
%
% PRE-CONDICIONES
% t: tiempo en el cual se desea saber la cantidad de neutrones.
%
% POST-CONDICIONES
% Devuelve el numero que representa la cantidad de neutrones en el tiempo t.
function res = tp2EcuacionRomberg(t)

    format longG

    directorio = 'CSV/';
    archivo = 'tp2MetodoRK4ValoresPaso05.csv';
```

```
    % Abrimos el archivo y generamos una matriz con los datos extraidos
    datosDeN = dlmread(strcat(directorio, archivo),',');

    posT = datosDeN(:,1)== t;

    res = datosDeN(posT,2);

end
```

En el *Código 4.*3 se muestra el código principal que invoca al método de Romberg y luego imprime por pantalla el resultado de la integración hecha.

Código 4.3 – tp2MetodoDeRomberg.m

```
% Función que utiliza el Método de Integración de Romberg y lo aplica a
%la función propuesta en el TP2.
%
% POST-CONDICIONES
% Se muestra por consola el resultado de la integración.
function tp2MetodoDeRomberg()

    format longG

    % Especificaciones
    funcion = @tp2EcuacionRomberg;
    a = 0;
    b = 72;
    K = 5;

    % Procesamos
    disp(' ');
    disp('Iniciando aplicación del Método de Romberg.')

    r = metodoDeRomberg(funcion, a, b, K);

    disp('El proceso ha finalizado exitosamente.');
    disp(' ');
    disp(['El resultado de la integración es ' num2str(r)]);

end
```

4.2. Resultados obtenidos

El resultado de la cantidad total de neutrones obtenido a través del *Matlab* es

$$N_{Total\ (Romberg)} = 7.94683864782 \times 10^{11}$$

Mientras que el resultado obtenido a partir de integrar la función de tendencia de los puntos generada por Microsoft Excel nos da el resultado:

$$N_{Total\ (Excel)} = 7.9305683 \times 10^{11}$$

Con lo cual podemos verificar los primeros ordenes de magnitud como seguros a partir del método de Romberg. Es aquí donde se ve la efectividad del método de Romberg, ya que a pesar de no tener una función definida hemos logrado, a partir de sólo tener conocimiento de puntos, una gran precisión en el resultado y una gran agilidad y eficiencia para su cálculo. Esto último será demostrado en el apartado siguiente al hacer un análisis del error.

4.3. Análisis del error

El resultado obtenido a través del *Matlab* contiene errores del tipo inherentes, así como de truncamiento, propios del método.

Como criterio para determinar el error de tipo inherente, buscamos el punto de máximo error del método; este crece a medida que avanzamos en el paso. Entonces podemos fijar como cota máxima del error la diferencia entre el último punto en RK-4 con paso 0.5 seg, con el RK-4 de paso 1 seg. Entonces tenemos,

$$\frac{\left|1518{,}96875420435 - 1563{,}47602190708\right|}{1518{,}96875420435} = 0{,}029300976455$$

Con lo cual podemos conservativamente tratar como un 3% de error inherente, como cota máxima.

El error de truncamiento podemos obtenerlo a través del método de las perturbaciones, sumándole una perturbación a los puntos que integraremos, y observando en que magnitud cambia el resultado final que devuelve el método de Romberg.

5. Integración entre 0 e ∞

Para realizar el cálculo de integración entre 0 e infinito de *N(t)*, nos encontramos con el problema principal de que, por métodos numéricos, no podemos realizar la integral hasta el infinito, ya que dicho límite no es alcanzable por una computadora.

Viendo los gráficos obtenidos que representan el valor de *N* en el tiempo, podemos observar que el mismo tiende a cero cuando el tiempo tiende a infinito, por lo cual, el resultado de la integral al aumentar el límite superior converge a un cierto valor.

Un método aceptable para resolver este problema, sería la comparación numérica entre los resultados de integrales que vayan de 0 a valores que consideremos "grandes". Pero, ¿Cómo saber cuándo un valor para el límite superior es lo suficientemente grande? La resolución propuesta es (suponiendo que tenemos una lista de valores lo suficientemente larga) tomar el resultado de la integral por Romberg de 0 a 120 segundos, y compararlo con el resultado de la misma integral, pero con límite superior en 240 segundos. Si la diferencia no satisface la precisión buscada, el límite superior puede duplicarse, llegando a 480 segundos, y compararlo con la integral anterior, es decir la que iba de 0 a 240.

Este paso puede repetirse indefinidas veces hasta alcanzar la precisión requerida.

El valor de precisión que se pide es discutible, teniendo en cuenta el orden de magnitud de resultados, que son superiores a 10^{11}, pero como podemos apreciar en los gráficos obtenidos, para tiempos cercanos a 120 segundos, los valores de *N(t)* son considerablemente más bajos.

Considerando las diferencias obtenidas con el método de Runge Kutta para los valores de *N* en 72 segundos, podemos aceptar una diferencia entre los resultados de integrales sucesivas del orden de 1, es decir, cuando la diferencia entre dos pasos consecutivos de cálculo de la integral sea igual o menor a 1, puede considerarse ese resultado como un representativo del valor de la integral de *N(t)* entre 0 e infinito.

6. Conclusión

Avocándonos a evaluar los resultados obtenidos, observamos que Euler tuvo un error del 0.16% mientras que RK-4 tuvo un error que tiende al 0%, atribuyendo semejante certeza a la cantidad de dígitos significativos utilizados (más precisamente, 15-16 dígitos significativos). Por esto último, diremos que Runge Kutta 4 es más exacto si bien la diferencia es mínima. Asimismo, observamos que cuanto más pequeño es el paso del tiempo, más exactos son los resultados, tanto para Euler como para Runge Kutta 4. Esto es de fundamental entendimiento para todo aquel que quiera aventurarse al uso de las resoluciones numéricas. *Al disminuir el paso de la solución, obtenemos mejores y más precisos resultados.*

El análisis numérico demuestra ser una herramienta excelente ante situaciones en las cuales no se conoce a priori la función exacta que domina el comportamiento de una

situación, tal como la fisión de los átomos de uranio demuestra ser. Al no poder obtenerse valores exactos, el análisis numérico nos provee la alternativa más lógica al problema: una estimación y su cota de error correspondiente.

En este trabajo práctico se evidencia de lleno esta situación, en la cual conocemos en líneas generales el proceso que domina la producción de neutrones a partir de la fisión de los U-235 y la absorción de los mismos a partir de las barras de control. El análisis numérico ofrece una alternativa discreta al problema: en lugar de resolver una elaborada ecuación diferencial la lleva a ser una ecuación en diferencias, regido su avance por el paso *h*. Algo análogo sucede en la integración numérica, la cual busca mediante el conocimiento de puntos pertenecientes a la curva, interpolarlos y obtener un valor cercano al real, sabiendo una cota máxima de su error en este proceso.

En particular, el análisis numérico demuestra ser la articulación entre el análisis matemático teórico y los resultados empíricos de la práctica, ya que parte de teoremas que garanticen la modelización congruente[3] de los procesos, y luego provee el andamiaje para evaluarlos.

Así es como podemos obtener estimaciones de cuan costosos serán nuestros errores, para este caso en particular, en una planta nuclear. Sin embargo, es esto una modelación. Ya que en la práctica real, la extracción de barras de control es sumamente controlada, luego, no podrá ella seguir sin advertencia del operador. Igualmente, si llegara a ocurrir el caso y necesitáramos obtener valores que cuantifiquen lo ocurrido en el proceso, los métodos numéricos son, sin duda, nuestra mejor opción.

Es esta aplicación de la matemática de inmensa utilidad en el campo científico e ingenieril, puesto que rara vez es necesario el cálculo de un valor perfecto teórico, sino una aproximación suficientemente buena. Estas cualidades y la forma de manipular los datos empíricos son provistas enteramente por el análisis numérico. Más aún, la posibilidad de poder defender la verdadera correspondencia de un resultado con la realidad, con un cierto nivel de confianza, es exclusivo alcance de la teoría numérica de errores; que es quizás lo mejor obtenible en un mundo empírico modulado por la incertidumbre.

[3] Esto se ve repetidas veces a través de la materia; ya sea en el teorema de punto fijo o en ecuaciones diferenciales que se exige que la función sea Lipchitziana en un entorno (lo cual significa cabalmente que la función no se comporte de forma muy extraña), la continuidad de la función en un entorno, o el teorema de Weierstrass en el caso de interpolación.

Apéndice A: Conversión de TXT a CSV

Para la conversión de un archivo de texto plano de extensión *.txt* a un archivo con formato *CSV* se utilizó la siguiente implementación realizada en *Matlab*, la cual se limita a funcionar con los nombres de archivos sujetos a este trabajo práctico. El código fuente puede verse a continuación:

Código A – convertirTXTaCSV.m

```
% Función que convierte el archivo k.txt de datos empiricos del TP2,
almacenado
% en el directorio 'mediciones', en un archivo con formato CSV.
function convertirTXTaCSV()

    directorio = 'mediciones/';
    archivoI = 'k.txt';
    archivoO = 'k.csv';
    datosDeK = load([directorio archivoI]);
    dlmwrite([directorio archivoO], datosDeK, 'precision', 16);

end
```

Bibliografía

- González, Hernán. *Análisis Numérico, Primer Curso*, Ed. Nueva Librería
- Richard L. Burden, J. Douglas Faires, *Analisis Numerico*, Ed. International Thomson
- *Matlab* Documentation (*http://www.mathworks.com/help/techdoc/)*