Física Computacional

Enrique Pazos

Viernes 5 de agosto de 2022

1. Método de Euler

Este método es uno de los más sencillos para integrar una ecuación diferencial ordinaria de primer orden. Necesitamos una ecuación de la forma

$$y'(t) = f(t, y(t)), \tag{1}$$

y una condición inicial que especifique un punto de la solución

$$y(t_0) = y_0. \tag{2}$$

El método de Euler es iterativo. La primera iteración toma la condición inicial como punto de partida $y_0(t_0)$ y va calculando los valores de y para instantes posteriores, es decir, $y_1(t_1)$, $y_2(t_2)$, $\cdots$, $y_n(t_n)$. En cada iteración el valor de la variable independiente t se incrementa en una cantidad h, que llamamos el *tamaño de paso*, de forma tal que $t_{i+1} = t_i + h$ o bien $t_n = t_0 + nh$.

La iteración está dada como

$$\begin{aligned} t_{i+1} &= t_i + h, \\ y_{i+1} &= y_i + hf(t_i, y_i). \end{aligned} \tag{3}$$

Por ejemplo, si queremos resolver la ecuación $y' = y/10$ sujeta la condición inicial $y(1.5) = 3.1$, con un valor de $h = 0.1$, podemos organizar la información como se muestra en la tabla 1. Ahora escribamos un programa que haga los cálculos por nosotros. Ver archivo `eurler.cpp`

i	t_i	y_i	$y_{i+1} = y_i + hf(t_i, y_i)$
0	1.5	3.1	3.131
1	1.6	3.131	3.16231
2	1.7	3.16231	3.19393
3	1.8	3.19393	3.22587
4	1.9	3.22587	3.25813
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Cuadro 1: Solución numérica de $y' = y/10$, sujeta a $y(1.5) = 3.1$, con $h = 0.1$.

Para compilar el programa escribimos en una terminal

```
g++ euler.cpp
```

si todo sale bien se genera un archivo ejecutable que por default se llama `a.out`. Para ejecutar el programa escribimos en la terminal

```
./a.out
```

Esto nos da dos columnas de datos con los valores de las variables t, y. Para redirigir lo que sale a la pantalla a un archivo basta con utilizar el símbolo $>$ de esta forma

```
./a.out > solucion
```

Ahora todo lo que salía en la pantalla está en el archivo llamado `solucion`.

Podemos comparar la solución numérica con la solución exacta de la ecuación. La solución exacta se puede encontrar por integración directa y es

$$y(t) = 2.66819e^{t/10}. \tag{4}$$

Utilizamos ahora el programa Gnuplot que nos permite hacer gráficas de forma sencilla. Escribimos `gnuplot` en la terminal y a continuación escribimos el comando que grafica tanto la solución numérica como la solución exacta

```
p 'solucion' pt 3, 2.66819*exp(x/10.0)
```

El resultado es una gráfica como la que se ve en la fig. 1. Podemos apreciar que al final del intervalo los asteriscos de la solución numérica se alejan un poco de la línea continua de la solución exacta. Esto se debe al error numérico inherente a la solución numérica.

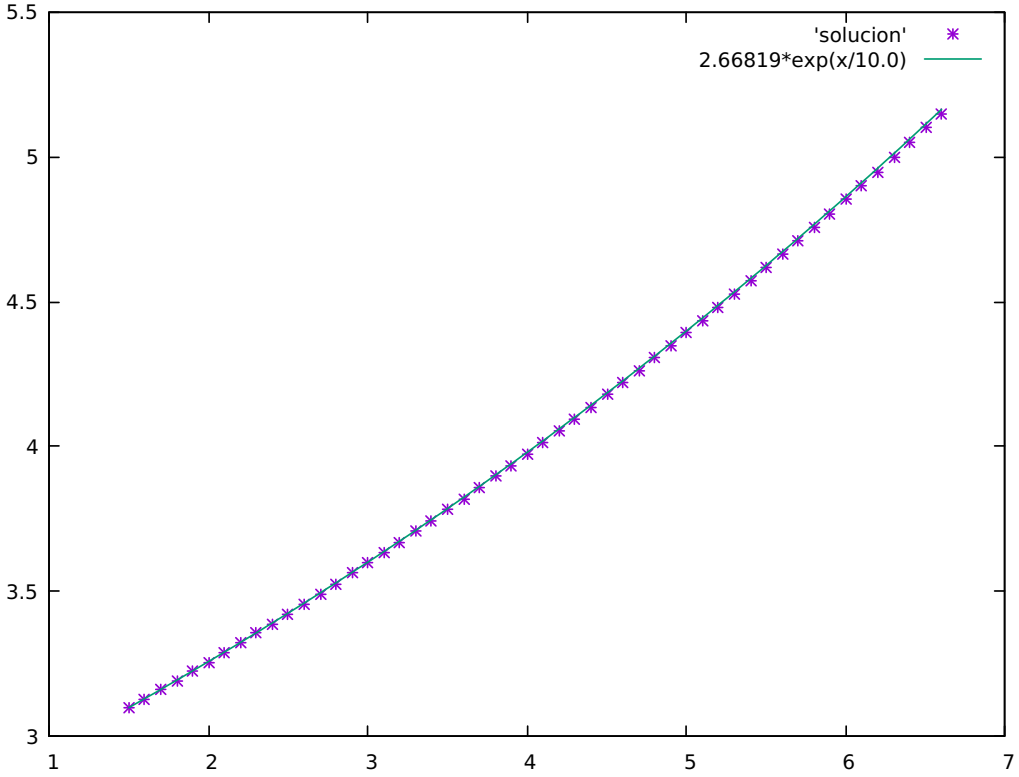

Figura 1: Solución numérica y solución exacta a la ecuación $y' = y/10$.

Viernes 12 de agosto de 2022

2. Método de Euler mejorado

También conocido como método de Heun. En este algoritmo hacemos un paso intermedio que mejora la aproximación de la solución. Al igual que en el método de Euler, necesitamos una ecuación diferencial ordinaria con su condición inicial. La iteración del método de Euler mejorado es

$$\begin{aligned} \tilde{y}_{i+1} &= y_i + hf(t_i, y_i), \\ y_{i+1} &= y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, \tilde{y}_{i+1})], \\ t_{i+1} &= t_i + h, \end{aligned} \tag{5}$$

donde la ecuación diferencial se escribe como

$$y'(t) = f(t, y(t)), \qquad y(t_0) = y_0. \tag{6}$$

Implementemos este algoritmo en un programa para resolver la siguiente ecuación

$$y' = \cos 2t, \qquad y(0) = 0. \tag{7}$$

Por integración directa, la solución es $y(t) = \frac{1}{2}\sin 2t$. Investiguemos ahora el error en la aproximación numérica.

2.1. Convergencia numérica

Llamemos $y_e(t) = \frac{1}{2}\sin 2t$ a la solución exacta. Llamemos también $y^{(1)}$ a la solución numérica que obtenemos cuando utilizamos un valor $h = h_1$ y de igual manera, llamemos $y^{(2)}$ a la solución numérica que obtenemos al utilizar un $h = h_2$. Decimos entonces, que si el método numérico tiene una exactitud de orden h^n, la diferencia entre la solución exacta y la solución numérica es

$$y_e - y^{(1)} = Ch_1^n + \cdots, \tag{8}$$

donde C es una constante y los puntos denotan términos de orden superior. Como h es pequeño, nos quedamos solamente con el primer término de la serie. De forma similar, podemos plantear la misma relación para solución $y^{(2)}$, de forma tal que tenemos

$$y_e - y^{(1)} = Ch_1^n, \tag{9}$$

$$y_e - y^{(2)} = Ch_2^n. \tag{10}$$

Dividiendo una ecuación entre la otra

$$\frac{y_e - y^{(1)}}{y_e - y^{(2)}} = \left(\frac{h_1}{h_2}\right)^n. \tag{11}$$

En el caso especial que $h_2 = \frac{1}{2}h_1$, la ecuación anterior nos dice que $h_1/h_2 = 2$. El valor de n está dado por el algoritmo que estamos utilizando. En el caso del método de Euler, la exactitud de la aproximación es de orden $n = 1$. Para el caso del método de Euler mejorado, este valor es $n = 2$. Por consiguiente, las soluciones numéricas del método de Euler deben cumplir con

$$\frac{y_e - y^{(1)}}{y_e - y^{(2)}} = \left(\frac{h_1}{h_2}\right)^2 = 2^2 = 4. \tag{12}$$

Esto lo podemos comprobar con el programa que hemos escrito. Por ejemplo, podemos utilizar una $h_1 = 0.1$ y una $h_2 = 0.05$. Si la razón de convergencia tiene un valor de 4, entonces podemos decir que la solución está convergiendo a la solución exacta.

2.2. Ejercicio

Encontrar la solución exacta y numérica de la ecuación

$$\frac{dy}{dt} = y \cos 2t, \qquad y(0) = 1. \tag{13}$$

Utilizar el método de Euler y el de Euler mejorado para verificar que la razón de convergencia es la apropiada.

Martes 16 de agosto de 2022

2.3. Caída libre con fricción del aire

Vamos a aplicar el método de Euler mejorado al caso de un objeto que cae bajo su propio peso y también experimenta una fuerza de fricción debida al aire, que es proporcional a la velocidad. La ecuación de movimiento es [ver ec. (2.66) de Symon, Mechanics]

$$F = -mg - bv, \tag{14}$$

donde m es la masa del objeto, g es la aceleración de la gravedad, v es la velocidad y b es una constante que depende de la forma y el tamaño del objeto. Expresando la aceleración como dv/dt, podemos escribir la ecuación de este forma

$$\begin{aligned} m\frac{dv}{dt} &= -mg - bv, \\ \frac{dv}{dt} &= -g - \frac{b}{m}v. \end{aligned} \tag{15}$$

Esta ecuación se puede resolver de forma exacta. Tomamos como condición inicial que $v(0) = 0$, es decir que el objeto parte del reposo. Después de integrar y aplicar la condición inicial tenemos

$$v(t) = -\frac{mg}{b}\left(1 - e^{-bt/m}\right). \tag{16}$$

Apliquemos el método de Euler mejorado para encontrar la solución de (15). Notamos que necesitamos darle un valor numérico a la constante b. Para esto vamos a experimentar con algunos valores hasta obtener una solución que parezca físicamente razonable.

Supongamos que los valores numéricos de las variables son $m = 0.5$ kg, $g = 9.8$ m/s^2, $b = 1$ kg/s. Encontraremos la solución para un tiempo total de 60 s escribiendo la salida del cálculo cada 1 s y el tamaño de paso será $h = 0.1$ s.

Este primer intento nos dice que es mejor escribir la solución a cada iteración y que es suficiente considerar los 10 primeros segundos.

2.4. Cambios en el programa

Vamos a modificar el programa del método de Euler mejorado para que las funciones no usen `return` sino que les enviemos la variable donde queremos el resultado.

Viernes 19 de agosto de 2022

3. Sistemas de ecuaciones diferenciales

Hasta el momento hemos aplicado métodos numéricos para resolver ecuaciones diferenciales ordinarias de primer orden. Sin embargo, ecuaciones como la segunda ley de Newton son de segundo orden. Para poder aplicar estos métodos numéricos a ecuaciones de orden superior debemos plantear las ecuaciones de una forma diferente.

Siempre es posible escribir una ecuación diferencial ordinaria de orden n como un sistema de n ecuaciones de primer orden. Para esto introducimos variables auxiliares de forma que solo aparezcan derivadas de primer orden. Por ejemplo, en el caso de la ec. (15), podemos escribir la ecuación original $F = -mg - bv$ como

$$m\frac{d^2y}{dt^2} = -mg - b\frac{dy}{dt}, \tag{17}$$

donde hemos utilizado el hecho que $v = dy/dt$ y $a = d^2y/dt^2$. Utilizando la notación de punto para las derivadas, nuestra ecuación queda

$$m\ddot{y} = -mg - b\dot{y}. \tag{18}$$

A fin de reescribir la segunda derivada, introducimos la velocidad v como variable auxiliar, de forma que

$$v = \dot{y}. \tag{19}$$

Con esta sustitución, la ec. (18) queda

$$m\dot{v} = -mg - bv. \tag{20}$$

Las ecs. (19) y (20) son el sistema de ecuaciones que buscamos. Para hacerlo evidente lo escribimos así

$$\begin{aligned} \dot{y} &= v, \\ \dot{v} &= -g - \frac{b}{m}v. \end{aligned} \tag{21}$$

Podemos definir un vector columna $Y = \begin{bmatrix} y \\ v \end{bmatrix}$, de forma que las ecuaciones se escriben de la siguiente forma

$$\frac{d}{dt}\begin{bmatrix} y \\ v \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -b/m \end{bmatrix}\begin{bmatrix} y \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ -g \end{bmatrix}, \tag{22}$$

es decir, que el sistema de ecuaciones de primer orden se puede plantear utilizando matrices. Es evidente que la forma del sistema es

$$\frac{dY}{dt} = AY + B, \tag{23}$$

lo cual representa la ecuación diferencial original de segundo orden. A este sistema de ecuaciones le podemos aplicar los métodos numéricos que ya conocemos, la solución será también la solución de la ecuación de orden superior que da origen al sistema de ecuaciones de primer orden.

3.1. Representación de las variables en el programa

En nuestro programa, a la función $y(t)$ la representamos con la variable independiente `t` y la variable dependiente `y`. En un sistema de ecuaciones tenemos varias variables dependientes. En el caso anterior, vemos en las ecs. (21) que las variables dependientes son y, v. En el programa, estas variables las representaremos como un arreglo (o vector) en donde y

$$y \to \texttt{y[0]}, \qquad v \to \texttt{y[1]}. \tag{24}$$

Es decir, que el "vector de estado" del sistema de ecuaciones que era $Y = \begin{bmatrix} y \\ v \end{bmatrix}$ se convierte en

$$Y = \begin{bmatrix} y[0] \\ y[1] \end{bmatrix}. \tag{25}$$

La forma de escribir la función `derivada` en el programa de C++ a partir del sistema (21) será así

```
dydt[0] = y[1];
dydt[1] = -g - b/m*y[1];
```

En donde `dydt[0], dydt[1]` son las componentes de otro arreglo que guarda el valor de la derivada de cada ecuación del sistema.

3.2. Variables, referencias y punteros en C++

Este pequeño programa ilustra las características de una variable, una referencia y un puntero.

```
int main()
{
  // variable
  double A = 3.1416;
  cout << A << " " << &A << endl;

  // copia de variables
  double B = A;
  cout << B << " " << &B << endl;

  // C es una referencia a A
  double &C = A;
  cout << C << " " << &C << endl;

  // D es un puntero hacia A
  double *D = &A;
  cout << D << " " << &D << " " << *D << endl;

  return 0;
}
```

3.3. Ejemplo: oscilador armónico amortiguado forzado

La ecuación de movimiento de una masa sujeta a un resorte con una fricción proporcional a la velocidad y que además está influenciada por una fuerza externa es la siguiente

$$m\ddot{x} + b\dot{x} + kx = F(t). \tag{26}$$

Resolvamos esta ecuación en forma numérica con el método de Euler mejorado. Dado que es una ecuación de segundo orden, tenemos que convertirla a dos ecuaciones de primer orden. Para ello definimos la variable auxiliar v de forma que

$$\dot{x} = v. \tag{27}$$

Esto implica que $\ddot{x} = \dot{v}$, por lo tanto la ec. (26) se puede reescribir así

$$\begin{aligned} m\dot{v} + bv + kx &= F(t), \\ \dot{v} &= \frac{F(t) - bv - kx}{m}. \end{aligned} \tag{28}$$

Las ecs. (27) y (28) forman nuestro sistema de ecuaciones. Para escribir este sistema en la computadora identificamos a x con la variable 0 y a v con la variable 1, es decir

$$x \to \texttt{y[0]}, \qquad v \to \texttt{y[1]}. \tag{29}$$

La función que calcula la derivada del sistema es

```
dydt[0] = y[1];
dydt[1] = (F(t) - b*y[1] - k*y[0])/m;
```

Para resolver la ecuación utilizamos como condición inicial $x(0) = 0$, $\dot{x}(0) = 0$. Asumimos que $F(t) = \sin(\pi t/5)$, $m = 0.1$ kg, $b = 0.001$ kg/s, $k = 0.05$ N/m, $h = 0.1$ para un tiempo total de 10 minutos. El resultado se muestra en la gráfica 2.

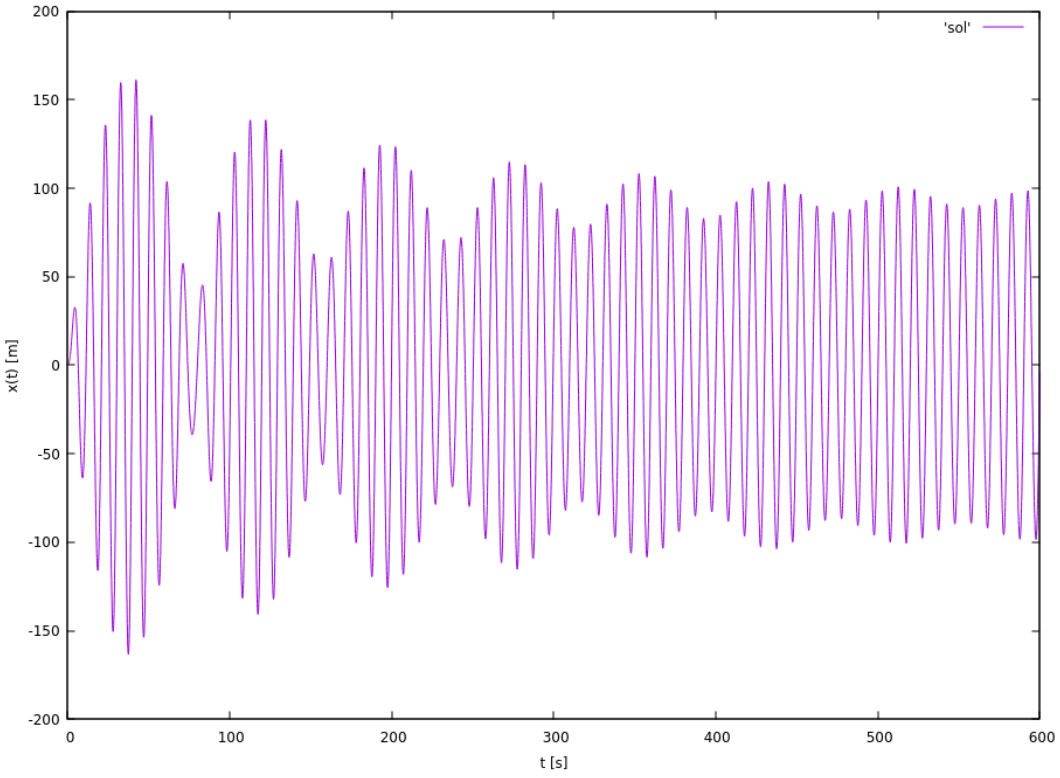

Figura 2: Oscilador armónico amortiguado forzado.

Martes 23 de agosto de 2022

4. Movimiento gravitacional en 1 dimensión

En este ejemplo ya no contaremos con la ventaja de tener una solución exacta al problema. Consideremos el movimiento vertical de un objeto que es lanzado sobre la superficie de la tierra a alturas arbitrariamente altas sobre la superficie de la Tierra. Ignorando la rotación del planeta y la fuerza de fricción del aire, la fuerza gravitacional sobre el objeto es

$$F = G\frac{Mm}{r^2}. \tag{30}$$

Tomaremos a M como la masa de la Tierra, r es la distancia entre el centro de la Tierra y el centro de masa del objeto, m es la masa del objeto y G es la constante de gravitación universal de Newton. La ecuación de movimiento es

$$\begin{aligned} m\ddot{r} &= -G\frac{Mm}{r^2}, \\ \ddot{r} &= -\frac{GM}{r^2}. \end{aligned} \tag{31}$$

Introducimos la variable auxiliar $v = \dot{r}$ que juega el papel de la velocidad vertical. Nuestro sistema de ecuaciones diferenciales de primer orden es

$$\begin{aligned} \dot{r} &= v, \\ \dot{v} &= -\frac{GM}{r^2}. \end{aligned} \tag{32}$$

La condiciones iniciales son tales que consideramos que el objeto se lanza hacia arriba con una cierta velocidad v_0 desde la superficie del planeta. Por lo tanto, las condiciones iniciales son

$$r(0) = r_0, \qquad v(0) = v_0, \tag{33}$$

donde $r_0 = 6.37 \times 10^6$ m y v_0 será un conjunto de valores: 10, 100, 1000, 10000 m/s. Para escribir la función `derivada` en el programa, hacemos la identificación de variables de forma que

$$r \to \texttt{y[0]}, \qquad v \to \texttt{y[1]}. \tag{34}$$

Las fórmulas en el programa quedarán así

```
dydt[0] = y[1];
dydt[1] = -G*M/(y[0]*y[0]);
```

4.1. Cambios en el programa

Para este problema seguiremos utilizando el método de Euler mejorado, haremos los siguientes cambios al programa con el que hemos venido trabajando.

1. Puntero a una función. Usaremos un puntero para la función que calcula la derivada del sistema de ecuaciones. De esta forma no tendremos que llamarle "derivada" a todas las ecuaciones que vayamos a resolver.

2. Crear una función que se encargue de la salida de la solución. Con esto haremos que las instrucciones de salida solo estén escritas una sola vez, en lugar de tenerlas en varias partes.

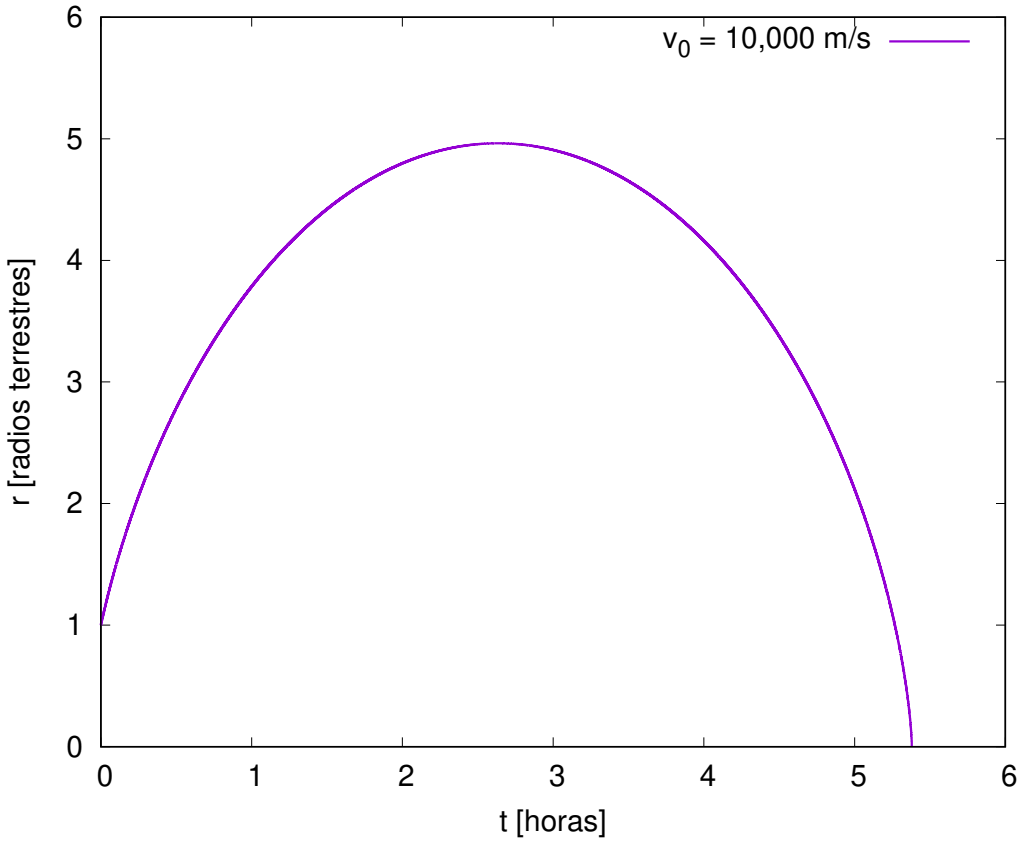

Figura 3: Movimiento gravitacional en una dimensión.

3. Cambiar el número de dígitos de la salida. Según el problema físico, puede que necesitemos más dígitos en la salida de la solución. Para esto utilizamos `fixed`, `scientific` y `setprecision`.

La solución numérica para el caso de $v_0 = 10,000$ m/s se muestra en la fig. 3

El script de Gnuplot para general la fig. 3 es el siguiente

```
set term postscript eps enhanced color size 5,4 font ',20'
set output 'movGrav1D.eps'

r0=6.37e6
set yrange [0:6]
set xrange [0:6]

set xlabel 't [horas]'
set ylabel 'r [radios terrestres]'

p 'sol' u ($1/3600.):($2/r0) w l lw 3 t 'v_0 = 10,000 m/s'
```

Para utilizarlo hay que copiar el texto anterior a un archivo de texto, por ejemplo `solucion.gp` y luego ejecutar en la terminal el comando `gnuplot solucion.gp`, esto creará el archivo `movGrav1D.eps`. El formato EPS es un tipo de gráfica vectorial que es independiente de la resolución. Muchas revistas científicas prefieren este formato.

Viernes 26 de agosto de 2022

5. Movimiento gravitacional en 2 dimensiones

Vamos a analizar el caso más general de un cuerpo de masa m que se mueve en un plano bajo la fuerza gravitacional de una masa M. Consideramos de $M \gg m$ de forma que M estará fija en el origen de coordenadas. La posición de m está dada por sus coordenadas cartesianas (x, y). Su distancia a la masa M es $r = \sqrt{x^2 + y^2}$. En estas circunstancias, escribimos la fuerza gravitacional como un vector

$$\mathbf{F} = -\frac{GMm}{r^2}\hat{\mathbf{r}}, \tag{35}$$

donde $\hat{\mathbf{r}}$ es el vector unitario

$$\hat{\mathbf{r}} = \frac{\mathbf{r}}{r} = \frac{x\hat{\mathbf{x}} + y\hat{\mathbf{y}}}{r} = \frac{x}{r}\hat{\mathbf{x}} + \frac{y}{r}\hat{\mathbf{y}}. \tag{36}$$

Sustituimos $\hat{\mathbf{r}}$ en (35) para expresar la fuerza en sus componentes cartesianas

$$\begin{aligned}\mathbf{F} &= -\frac{GMm}{r^2}\left[\frac{x}{r}\hat{\mathbf{x}} + \frac{y}{r}\hat{\mathbf{y}}\right] \\ &= -\frac{GMmx}{r^3}\hat{\mathbf{x}} - \frac{GMmy}{r^3}\hat{\mathbf{y}}.\end{aligned} \tag{37}$$

La segunda ley de Newton, $\mathbf{F} = m\ddot{\mathbf{r}}$, también se separa en componentes. Expresando la aceleración como $\ddot{\mathbf{r}} = \ddot{x}\hat{\mathbf{x}} + \ddot{y}\hat{\mathbf{y}}$, tenemos que

$$\begin{aligned} m\ddot{\mathbf{r}} &= \mathbf{F}, \\ m(\ddot{x}\hat{\mathbf{x}} + \ddot{y}\hat{\mathbf{y}}) &= -\frac{GMmx}{r^3}\hat{\mathbf{x}} - \frac{GMmy}{r^3}\hat{\mathbf{y}}, \\ \ddot{x}\hat{\mathbf{x}} + \ddot{y}\hat{\mathbf{y}} &= -\frac{GMx}{r^3}\hat{\mathbf{x}} - \frac{GMy}{r^3}\hat{\mathbf{y}}, \end{aligned}$$

igualando cada componente obtenemos

$$\ddot{x} = -\frac{GMx}{r^3}, \qquad \ddot{y} = -\frac{GMy}{r^3}, \tag{38}$$

estas son las ecuaciones de movimiento para las coordenadas x, y de nuestra partícula.

5.1. Ecuaciones de primer orden

Para poder aplicar el método de Euler mejorado debemos escribir las ecs. (38) como un sistema de ecuaciones de primer orden. Introducimos como variables auxiliares las velocidades $u = \dot{x}$ y $v = \dot{y}$, de tal forma que al tomar una derivada más tenemos que $\dot{u} = \ddot{x}$ y $\dot{v} = \ddot{y}$. Eliminando las segundas derivadas de (38) obtenemos lo siguiente

$$\begin{aligned} \dot{x} &= u, \\ \dot{u} &= -\frac{GMx}{r^3} \\ \dot{y} &= v, \\ \dot{v} &= -\frac{GMy}{r^3}. \end{aligned} \tag{39}$$

Como tenemos dos ecuaciones de segundo orden, nuestro sistema consiste en cuatro ecuaciones de primer orden. Este es el sistema de ecuaciones que vamos a alimentar en la computadora.

Lo que debemos escoger ahora es el orden de la ecuaciones, pues de eso depende qué variables representan `y[0], y[1], y[2], y[3]` en nuestro programa. Si queremos que `y[0]` sea x y que `y[1]` sea y, entonces reordenamos el sistema así

$$\begin{aligned}\dot{x} &= u,\\ \dot{y} &= v,\\ \dot{u} &= -\frac{GMx}{r^3}\\ \dot{v} &= -\frac{GMy}{r^3},\end{aligned} \tag{40}$$

de esta forma

$$x \to \texttt{y[0]}, \quad y \to \texttt{y[1]}, \quad u \to \texttt{y[2]}, \quad v \to \texttt{y[3]}. \tag{41}$$

La función que calcula las derivadas tendrá el siguiente texto

```
r3 = pow( y[0]*y[0] + y[1]*y[1], 1.5 );
dydt[0] = y[2];
dydt[1] = y[3];
dydt[2] = -G*M*y[0] / r3;
dydt[3] = -G*M*y[1] / r3;
```

Aquí hemos utilizado el hecho de que $r^3 = (x^2 + y^2)^{3/2}$.

5.2. Condiciones iniciales

Vamos a analizar el sistema Tierra-Luna. La Tierra estará en el origen de coordenadas y la Luna se moverá alrededor de la Tierra. Para lograrlo simular el movimiento debemos utilizar condiciones iniciales que reflejen la posición y velocidad de la Luna en un instante dado. Para posición inicial ubicaremos a la Luna sobre el eje x

$$x(0) = 3.844 \times 10^8 \text{ m}, \qquad y(0) = 0. \tag{42}$$

Dado que la órbita de la Luna es casi circular, necesitamos dirigir la velocidad de forma perpendicular al vector posición, es decir, la velocidad apunta en la dirección y

$$\dot{x}(0) = 0, \qquad \dot{y}(0) = 1.022 \times 10^3 \text{ m/s}. \tag{43}$$

En términos de las variables del programa las condiciones iniciales según (41) son

```
y[0] = 3.844e8;
y[1] = 0.0;
y[2] = 0.0;
y[3] = 1.022e3;
```

5.3. Cambios en el programa

Algunos cambios relevantes en el programa son

1. El número de ecuaciones ahora es 4.
2. Debemos enviar el número de ecuaciones a la función que saca la solución en pantalla.
3. Escoger un tamaño de paso adecuado al problema.
4. Crear una función que calcule la energía mecánica total y verificar si se conserva.
5. Hacer que la energía sea escrita a un archivo diferente.

Martes 30 de agosto de 2022

6. Energía del sistema

Cuando resolvemos ecuaciones que no tienen una solución exacta no es posible encontrar el error en la solución numérica de forma directa. Una manera de saber si la aproximación numérica por lo menos tiene significado físico consiste en calcular la energía total del sistema. Si la energía se conserva, esto implica que en todo momento la energía debe ser constante. Si esta condición se cumple en la solución de la ecuación diferencial que describe la dinámica del sistema, entonces podemos tener seguridad de que la solución obtenida no contiene suficiente error numérico para violar las leyes de conservación.

En el caso de la sección anterior, de una partícula moviéndose bajo una fuerza gravitacional, energía total es

$$E = \frac{1}{2}mv^2 - \frac{GMm}{r}, \tag{44}$$

donde v es la magnitud del vector velocidad. En el código podemos crear una función que calcule la energía total de la partícula. Las fórmulas quedarían así

```
  const double m = 7.34e22; //masa de la Luna
  const double M = 5.98e24; // masa de la Tierra
  const double G = 6.66e-11; // Constante de gravitacion universal

  Ecinetica  = 0.5*m*(y[2]*y[2] + y[3]*y[3]);
  Epotencial =-G*M*m/sqrt( y[0]*y[0] + y[1]*y[1] );

  Energia = Ecinetica + Epotencial;
```

7. Animaciones con Gnuplot

Gnuplot puede hacer animaciones escribiendo un script muy sencillo. La versión más simple de este script es la siguiente

```
set xrange [-1e8:4e8]
set yrange [-2e8:2e8]
set size ratio -1

do for [i=1:100] {
    plot 'solucion' every ::i::i u 2:3 w p pt 7 ps 2

    pause 0.5
}
```

El rango en los ejes lo escogimos para que se ajuste al sistema de una partícula que hemos trabajado hasta el momento. Estas instrucciones generan una animación que podemos ver en la ventana de Gnuplot.

7.1. GIF animado

Para hacer un GIF animado necesitamos generar todos los cuadros de la animación. Esto lo hacemos en Gnuplot con un script similar

```
set term pngcairo size 800,600 enhanced font "Verdana,12"
```

```
set xrange [-1e8:4e8]
set yrange [-2e8:2e8]
set size ratio -1

do for [i=1:100]{
    set output sprintf( "frames/frame-%04d.png",i)
    plot 'solucion' every ::i::i u 2:3 w p pt 7 ps 2

    print i
}
```

Este texto lo guardamos en un archivo, por ejemplo `movie.gp` y lo ejecutamos escribiendo `gnuplot movie.gp` en la terminal. La diferencia con el anterior es que aquí Gnuplot genera un archivo PNG por cada cuadro y los guarda en el directorio `frames` (hay que crearlo).

Para unir la secuencia de cuadros en un GIF utilizamos el siguiente comando en la terminal

```
convert -delay 10 -loop 0 frames/*png movie.gif
```

7.2. Animación en formato MP4

El formato GIF suele ser limitado. No solo no acepta sonido, sino que puede dar problemas si la animación contiene muchos cuadros. Por eso, en general es mejor el formato de película MP4. Para convertir una secuencia de imágenes en película utilizamos el siguiente comando

```
ffmpeg -i frames/frame-%04d.png -c:v libx264 -pix_fmt yuv420p -r 30 movie.mp4
```

Martes 6 de septiembre de 2022

8. Ejemplo: Problema de los tres cuerpos

8.1. Ecuaciones de movimiento

Vamos resolver numéricamente el problema de los tres cuerpos. Consideramos 3 cuerpos de masas m_1, m_2 y m_3. La fuerza $\mathbf{F}_i$ que experimenta la masa m_i ubicada en una posición $\mathbf{r}_i$, debido al resto de las masas, se expresa en forma general de la siguiente manera

$$\mathbf{F}_i = \sum_{j\neq i} -Gm_i m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}, \tag{45}$$

donde $|\mathbf{r}_i - \mathbf{r}_j| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. La segunda ley de Newton $\mathbf{F} = m\ddot{\mathbf{r}}$ se expresa como

$$\begin{aligned} m_i\ddot{\mathbf{r}}_i &= \sum_{j\neq i} -Gm_i m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}, \\ \ddot{x}_i\hat{\mathbf{x}} + \ddot{y}_i\hat{\mathbf{y}} &= \sum_{j\neq i} -\frac{Gm_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}\left[(x_i - x_j)\hat{\mathbf{x}} + (y_i - y_j)\hat{\mathbf{y}}\right]. \end{aligned}$$

Igualamos cada lado por componentes para obtener

$$\begin{aligned} \ddot{x}_i &= \sum_{j\neq i} -\frac{Gm_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}(x_i - x_j), \\ \ddot{y}_i &= \sum_{j\neq i} -\frac{Gm_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}(y_i - y_j). \end{aligned} \tag{46}$$

Para cada masa tenemos dos ecuaciones diferenciales de segundo orden. Siendo solo 3 masas, podemos escribir y expandir las sumatorias para cada una de ellas. Eso nos da las siguientes ecuaciones

$$\begin{aligned} \ddot{x}_1 &= -\frac{Gm_2}{|\mathbf{r}_1 - \mathbf{r}_2|^3}(x_1 - x_2) - \frac{Gm_3}{|\mathbf{r}_1 - \mathbf{r}_3|^3}(x_1 - x_3), \\ \ddot{y}_1 &= -\frac{Gm_2}{|\mathbf{r}_1 - \mathbf{r}_2|^3}(y_1 - y_2) - \frac{Gm_3}{|\mathbf{r}_1 - \mathbf{r}_3|^3}(y_1 - y_3), \\ \ddot{x}_2 &= -\frac{Gm_1}{|\mathbf{r}_2 - \mathbf{r}_1|^3}(x_2 - x_1) - \frac{Gm_3}{|\mathbf{r}_2 - \mathbf{r}_3|^3}(x_2 - x_3), \\ \ddot{y}_2 &= -\frac{Gm_1}{|\mathbf{r}_2 - \mathbf{r}_1|^3}(y_2 - y_1) - \frac{Gm_3}{|\mathbf{r}_2 - \mathbf{r}_3|^3}(y_2 - y_3), \\ \ddot{x}_3 &= -\frac{Gm_1}{|\mathbf{r}_3 - \mathbf{r}_1|^3}(x_3 - x_1) - \frac{Gm_2}{|\mathbf{r}_3 - \mathbf{r}_2|^3}(x_3 - x_2), \\ \ddot{y}_3 &= -\frac{Gm_1}{|\mathbf{r}_3 - \mathbf{r}_1|^3}(y_3 - y_1) - \frac{Gm_2}{|\mathbf{r}_3 - \mathbf{r}_2|^3}(y_3 - y_2). \end{aligned} \tag{47}$$

Este es un sistema de 6 ecuaciones diferenciales de segundo orden. Lo que corresponde a un sistema de 12 ecuaciones diferenciales de primer orden. Utilizamos las velocidades como variables auxiliares, de forma que u_i, v_i son las componentes en x, y para la i-ésima partícula,

respectivamente. El sistema de ecuaciones de primer orden es el siguiente

$$\begin{aligned}
\dot{x}_1 &= u_1,\\
\dot{y}_1 &= v_1,\\
\dot{x}_2 &= u_2,\\
\dot{y}_2 &= v_2,\\
\dot{x}_3 &= u_3,\\
\dot{y}_3 &= v_3,\\
\dot{u}_1 &= -\frac{Gm_2}{|\mathbf{r}_1-\mathbf{r}_2|^3}(x_1-x_2) - \frac{Gm_3}{|\mathbf{r}_1-\mathbf{r}_3|^3}(x_1-x_3),\\
\dot{v}_1 &= -\frac{Gm_2}{|\mathbf{r}_1-\mathbf{r}_2|^3}(y_1-y_2) - \frac{Gm_3}{|\mathbf{r}_1-\mathbf{r}_3|^3}(y_1-y_3),\\
\dot{u}_2 &= -\frac{Gm_1}{|\mathbf{r}_2-\mathbf{r}_1|^3}(x_2-x_1) - \frac{Gm_3}{|\mathbf{r}_2-\mathbf{r}_3|^3}(x_2-x_3),\\
\dot{v}_2 &= -\frac{Gm_1}{|\mathbf{r}_2-\mathbf{r}_1|^3}(y_2-y_1) - \frac{Gm_3}{|\mathbf{r}_2-\mathbf{r}_3|^3}(y_2-y_3),\\
\dot{u}_3 &= -\frac{Gm_1}{|\mathbf{r}_3-\mathbf{r}_1|^3}(x_3-x_1) - \frac{Gm_2}{|\mathbf{r}_3-\mathbf{r}_2|^3}(x_3-x_2),\\
\dot{v}_3 &= -\frac{Gm_1}{|\mathbf{r}_3-\mathbf{r}_1|^3}(y_3-y_1) - \frac{Gm_2}{|\mathbf{r}_3-\mathbf{r}_2|^3}(y_3-y_2).
\end{aligned} \tag{48}$$

La forma de enumerar las variables en el programa corresponde al orden en que hemos escrito las ecuaciones

$$\begin{aligned}
&x_1 \to \texttt{y[0]}, \quad y_1 \to \texttt{y[1]}, \quad x_2 \to \texttt{y[2]}, \quad y_2 \to \texttt{y[3]}, \quad x_3 \to \texttt{y[4]}, \quad y_3 \to \texttt{y[5]},\\
&u_1 \to \texttt{y[6]}, \quad v_1 \to \texttt{y[7]}, \quad u_2 \to \texttt{y[8]}, \quad v_2 \to \texttt{y[9]}, \quad u_3 \to \texttt{y[10]}, \quad v_3 \to \texttt{y[11]}.
\end{aligned}$$

8.2. Condiciones iniciales

8.2.1. Sistema Sol-Tierra-Luna

Verificamos que nuestro programa funcione bien con un sistema conocido. Tomamos las masas como

$$\begin{aligned}
m_1 &= 1.989 \times 10^{30}\ \text{kg} \qquad \text{(Sol)},\\
m_2 &= 5.972 \times 10^{24}\ \text{kg} \qquad \text{(Tierra)},\\
m_3 &= 7.348 \times 10^{22}\ \text{kg} \qquad \text{(Luna)}.
\end{aligned} \tag{49}$$

Las posiciones iniciales son

$$\begin{aligned}
&x_1(0) = 0, \qquad y_1(0) = 0, \qquad \text{(el Sol está en el origen)}\\
&x_2(0) = 1.5096 \times 10^{11}\ \text{m}, \qquad y_2(0) = 0,\\
&x_3(0) = 1.513444 \times 10^{11}\ \text{m}, \qquad y_2(0) = 0.
\end{aligned} \tag{50}$$

Notemos que la ubicación de la luna es con respecto del origen de coordenadas, por lo tanto su posición en el eje x es la suma de la distancia Sol-Tierra más la distancia Tierra-Luna.

Las velocidades iniciales son

$$\begin{aligned}
&u_1(0) = 0, \qquad v_1(0) = 0,\\
&u_2(0) = 0, \qquad v_2(0) = 2.978 \times 10^4\ \text{m/s},\\
&u_3(0) = 0, \qquad v_2(0) = 3.0802 \times 10^4\ \text{m/s}.
\end{aligned}$$

Utilizamos la velocidad de la luna respecto del Sol, por tal razón sumamos la velocidad de la Luna respecto a la Tierra más la velocidad de la Tierra.

8.3. Energía del sistema

La energía mecánica de los tres cuerpos es la suma de sus energías cinéticas más la energía potencial

$$E = \frac{1}{2}m_1V_1^2 + \frac{1}{2}m_2V_2^2 + \frac{1}{2}m_3V_3^2 - \frac{Gm_1m_2}{|\mathbf{r}_1 - \mathbf{r}_2|} - \frac{Gm_1m_3}{|\mathbf{r}_1 - \mathbf{r}_3|} - \frac{Gm_2m_3}{|\mathbf{r}_2 - \mathbf{r}_3|}, \tag{51}$$

donde V_i es la magnitud de la velocidad de la i-ésima partícula.

8.4. Tamaño de paso e iteraciones

Utilizamos $h = 3600$, es decir que el tamaño de paso es 1 hora. Vamos a cubrir un intervalo temporal de 1 año, por lo tanto el número de iteraciones $N = 24 \times 265$. La salida de la solución la haremos cada día, por lo tanto el parámetro `out_cada` será 24.

8.5. Gráficas de la solución

Con el siguiente script de Gnuplot podemos general la gráfica de las trayectorias que se muestra en la fig. 4. Debido a que la distancia Tierra-Luna es muy pequeña comparada con la distancia Sol-Tierra, hemos magnificado la primera para que sea visible en la gráfica.

```
set term postscript eps enhanced color size 5,4 font ',20'
set output 'orbitas_3cuerpos.eps'

set size ratio -1
set xlabel 'x (10^6 km)'
set ylabel 'y (10^6 km)'

# escala, para que la escala de los ejes sea en millones de kilometros
s = 1e9

# factor de magnificacion de la separacion Tierra-Luna
f = 15

plot 'solucion' u ($2/s):($3/s) w p pt 7 ps 3 lc rgbcolor '#ff9900' t 'Sol', \
     '' u ($4/s):($5/s) w l t 'Tierra', \
     '' u ((f*($6-$4)+$4)/s):((f*($7-$5)+$5)/s) w l lc 7 t 'Luna'
```

También podemos graficar la energía con el siguiente script. El resultado se muestra el la fig. 5. Podemos observar que la energía se mantiene constante, lo cual indica que nuestra solución no acumula una cantidad apreciable de error numérico.

```
set output 'energia_3cuerpos.eps'
set size noratio
set xlabel 'tiempo (meses)'
set ylabel 'Energia (J)'
set xrange [0:12.5]
set yrange [:0]
set grid

# para tener el tiempo en meses
```

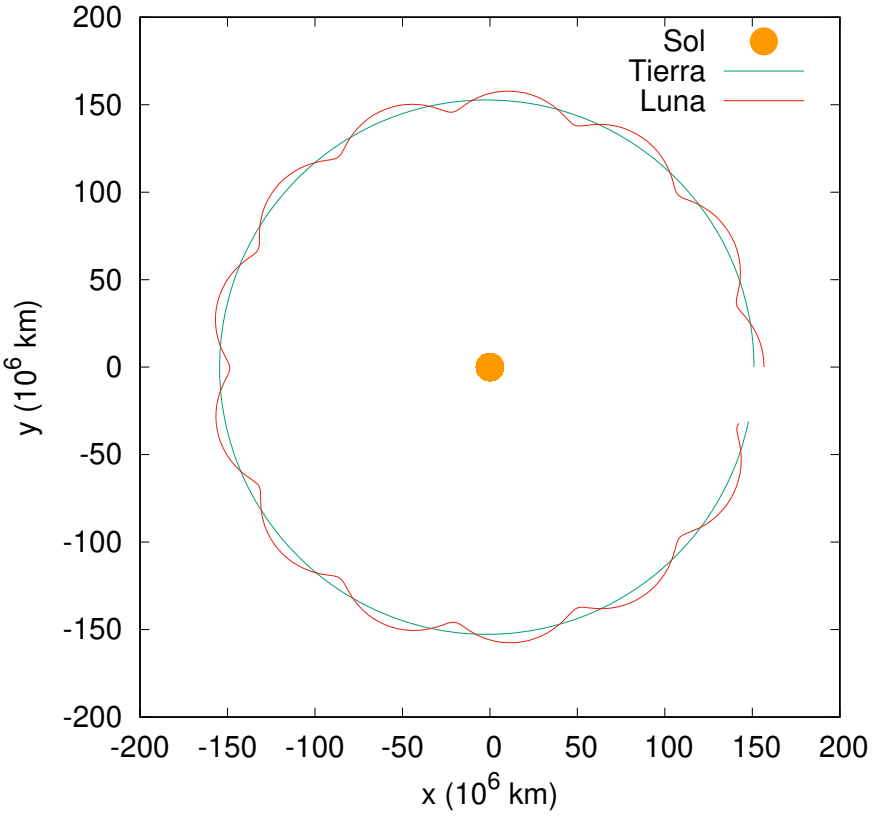

Figura 4: Órbitas de los tres cuerpos Sol, Tierra y Luna. La distancia Tierra-Luna está magnificada en un factor de 15.

```
s = 3600*24*30

plot 'energia.dat' u ($1/s):2 w l lw 6 t 'Energia'
```

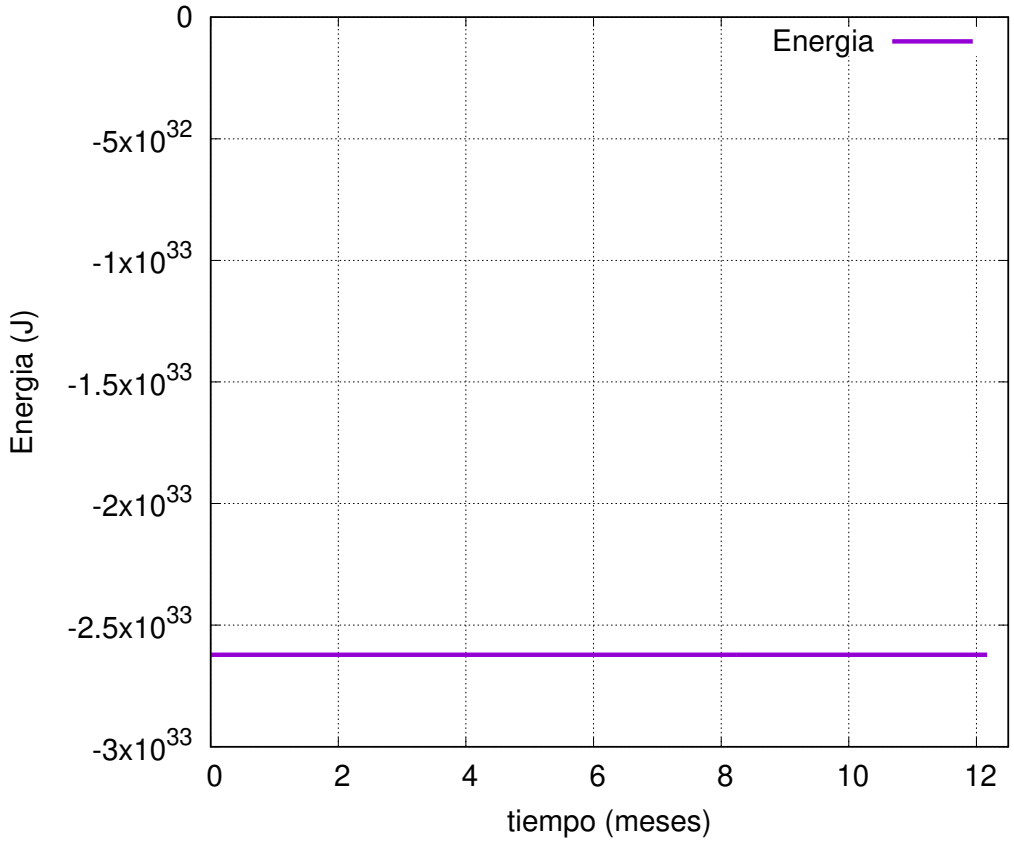

Figura 5: Energía mecánica para el sistema Sol, Tierra y Luna.

Viernes 9 de septiembre de 2022

9. Método de Runge-Kutta de 4to orden

Este algoritmo es más preciso que el método de Euler mejorado, el cual tiene un error de numérico de orden $O(h^2)$, mientras que este presenta un error numérico de orden $O(h^4)$. Dada una ecuación diferencial $y'(t) = f(t, y)$ y una condición inicial $y(t_0) = y_0$, el algoritmo es

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{6}(k_0 + 2k_1 + 2k_2 + k_3), \\ t_{i+1} &= t_i + h, \end{aligned} \tag{52}$$

donde las cantidades k son

$$\begin{aligned} k_0 &= f(t_i, y_i), \\ k_1 &= f\left(t_i + \frac{h}{2}, y_n + h\frac{k_0}{2}\right), \\ k_2 &= f\left(t_i + \frac{h}{2}, y_n + h\frac{k_1}{2}\right), \\ k_3 &= f(t_i + h, y_i + hk_2). \end{aligned} \tag{53}$$

Utilizaremos este método para resolver el sistema Sol-Tierra-Luna y comparar la solución con el método de Euler mejorado.

10. Péndulo doble compuesto

El péndulo doble compuesto consta de dos barras rígidas unidas en sus extremos, siendo libres de oscilar sin fricción bajo la influencia de la gravedad, ver Fig. 6.

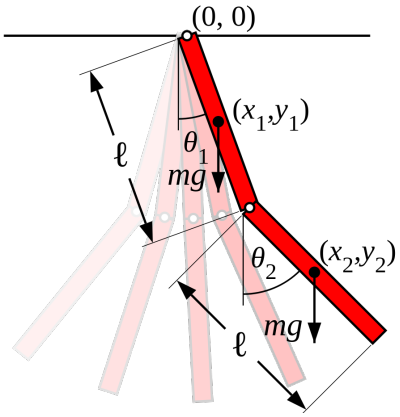

Figura 6: Péndulo doble compuesto. Fuente: wikipedia

Las ecuaciones de movimiento pueden obtenerse formulando el Lagrangiano del sistema. Aunque el péndulo doble no parezca un sistema muy complejo, las ecuaciones de movimiento no pueden resolverse en forma analítica, por lo que necesitamos recurrir a un método numérico para investigar su comportamiento.

Consideremos un péndulo doble compuesto en donde la longitud l y la masa m de cada una de las barras son iguales. El Lagrangiano del sistema está dado por

$$L = \frac{1}{6}ml^2\left[\dot{\theta}_2^2 + 4\dot{\theta}_1^2 + 3\dot{\theta}_1\dot{\theta}_2\cos(\theta_1-\theta_2)\right] + \frac{1}{2}mgl(3\cos\theta_1 + \cos\theta_2). \tag{54}$$

Las ecuaciones de movimiento son

$$\dot{\theta}_1 = \frac{6}{ml^2}\frac{2p_1 - 3p_2\cos(\theta_1-\theta_2)}{16 - 9\cos^2(\theta_1-\theta_2)}, \tag{55}$$

$$\dot{\theta}_2 = \frac{6}{ml^2}\frac{8p_2 - 3p_1\cos(\theta_1-\theta_2)}{16 - 9\cos^2(\theta_1-\theta_2)}, \tag{56}$$

$$\dot{p}_1 = -\frac{1}{2}ml^2\left[\dot{\theta}_1\dot{\theta}_2\sin(\theta_1-\theta_2) + 3\frac{g}{l}\sin\theta_1\right], \tag{57}$$

$$\dot{p}_2 = -\frac{1}{2}ml^2\left[-\dot{\theta}_1\dot{\theta}_2\sin(\theta_1-\theta_2) + \frac{g}{l}\sin\theta_2\right]. \tag{58}$$

Donde p_1 y p_2 es el momentum canónicamente conjugado con θ_1 y θ_2, respectivamente.

En síntesis, lo que tenemos es un sistema de cuatro ecuaciones diferenciales ordinarias, las cuales pueden ser resueltas numéricamente con un método numérico.

La forma de enumerar las variables en el programa corresponde al orden en que hemos escrito las ecuaciones

$$\theta_1 \to \texttt{y[0]}, \qquad \theta_2 \to \texttt{y[1]}, \qquad p_1 \to \texttt{y[2]}, \qquad p_2 \to \texttt{y[3]}. \tag{59}$$

10.1. Solución ejemplo

El problema consiste en resolver numéricamente el sistema de ecuaciones (55)-(58) con los métodos de Euler mejorado y Runge-Kutta de 4to orden, planteando condiciones iniciales razonables para posteriormente, visualizar la solución.

Utilizamos valores de $m = 0.5$ kg, $l = 0.3$ m y $g = 9.8$ m/s^2, imponiendo como condición inicial que las barras empiezan el movimiento desde una posición horizontal desde el reposo, es decir

$$\theta_1(0) = \pi/2, \tag{60}$$

$$\theta_2(0) = \pi/2, \tag{61}$$

$$p_1(0) = 0, \tag{62}$$

$$p_2(0) = 0. \tag{63}$$

Con esta configuración, la solución numérica para los ángulos θ_1 y θ_2 se muestran en la Fig. 7. Para este ejemplo utilizamos $h = 0.01$ s, con 12,000 iteraciones y salida cada 2 iteraciones.

Podemos verificar cuál de las dos soluciones es más precisa calculando la energía del sistema. Esta se muestra en la Fig. 8.

Estas gráficas están hechas con el siguiente script de Gnuplot

```
set term postscript eps enhanced color size 5,4 font ',20'

set output 'pendulo-doble-th1.eps'
set yrange [-2:2.2]
set xrange [0:10]
set key left
set xlabel 'tiempo (s)'
set ylabel '{/Symbol q}_1'
```

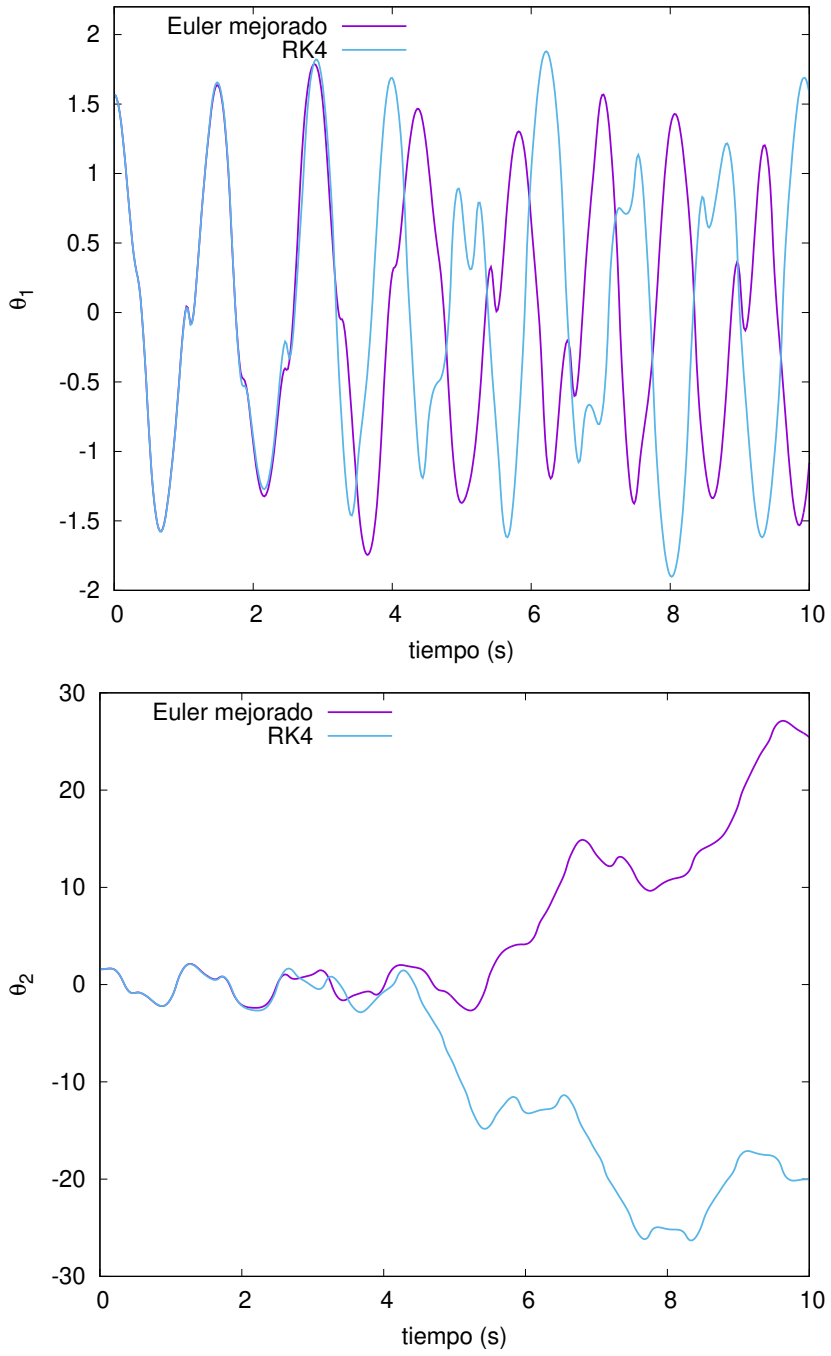

Figura 7: Soluciones para θ_1 y θ_2 en función del tiempo utilizando el método de Euler mejorado y Runge-Kutta de 4to orden.

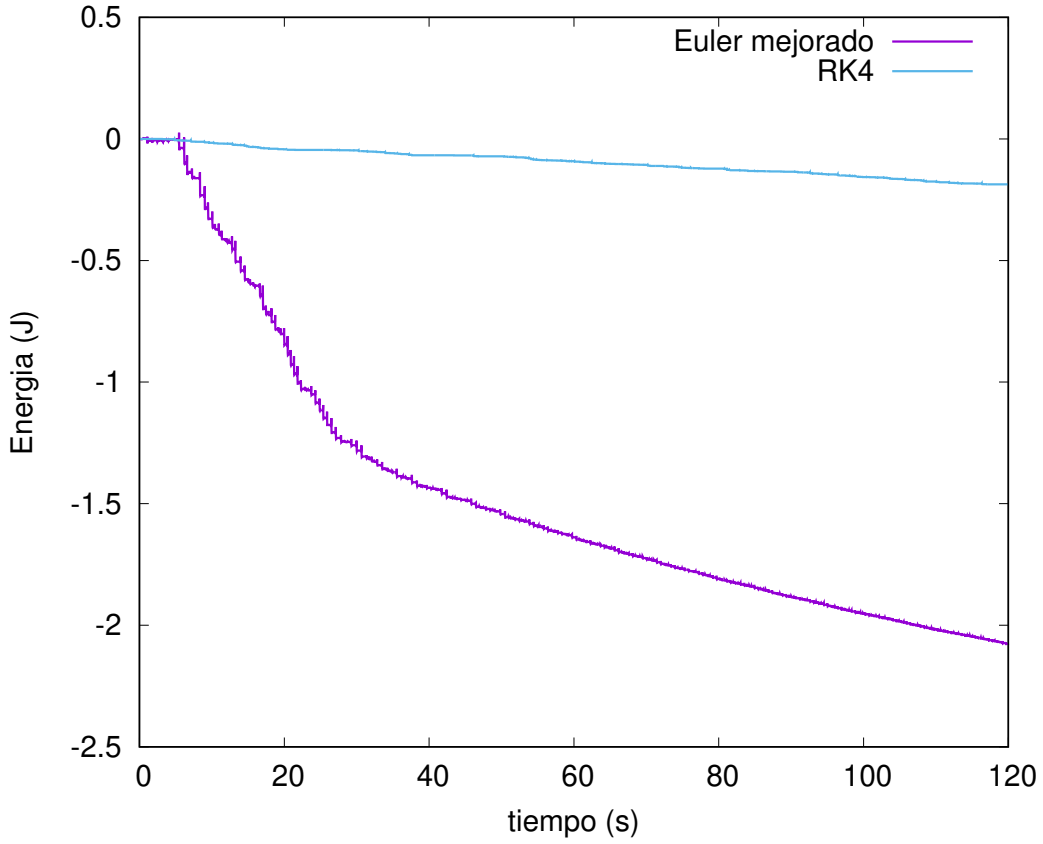

Figura 8: Energía para el péndulo doble. El método que mejor conserva la energía es RK4, sin embargo también se pierde un poco.

```
p 'solucion-em'  u 1:2 w l lw 3 lc 1 t 'Euler mejorado', \
  'solucion-rk4' u 1:2 w l lw 3 lc 3 t 'RK4'


set output 'pendulo-doble-th2.eps'
set yrange [-30:30]
set xlabel 'tiempo (s)'
set ylabel '{/Symbol q}_2'

p 'solucion-em'  u 1:3 w l lw 3 lc 1 t 'Euler mejorado', \
  'solucion-rk4' u 1:3 w l lw 3 lc 3 t 'RK4'



set output 'energia-pendulo-doble.eps'
set key right
set yrange [-2.5:0.5]
set xrange [0:120]
set xlabel 'tiempo (s)'
set ylabel 'Energia (J)'

p 'energia-em.dat'  u 1:2 w l lw 3 lc 1 t 'Euler mejorado', \
  'energia-rk4.dat' u 1:2 w l lw 3 lc 3 t 'RK4'
```

Martes 13 de septiembre de 2022

11. Convergencia de la solución numérica

En la Sec. 2.1 vimos cómo utilizar la solución exacta de una ecuación diferencial para verificar que la solución numérica se acerca cada vez más al valor exacto a medida que el tamaño de paso se hace más pequeño. En esta sección haremos el mismo análisis pero sin recurrir a la solución exacta, que en general es desconocida. Para ello suponemos que contamos con tres soluciones numéricas que han sido obtenidas con tres diferentes tamaños de paso. Llamaremos $y^{(1)}$, $y^{(2)}$ e $y^{(3)}$ a las soluciones numéricas, las cuales han sido obtenidas con los tamaños de paso h_1, h_2 y h_3, respectivamente. Para un método numérico de orden n, el error en la solución es de orden h^n. Decimos entonces que la diferencia entre la solución numérica y la solución exacta y_e es

$$y_e - y^{(1)} = Ch_1^n + \cdots , \tag{64}$$

esta es la misma expresión que vimos en la ec. (8). De la misma forma planteamos expresiones similares para las otras dos soluciones

$$y_e - y^{(2)} = Ch_2^n, \tag{65}$$

$$y_e - y^{(3)} = Ch_3^n. \tag{66}$$

En la mayoría de situaciones prácticas, la solución exacta y_e y la constante C son desconocidas, así que vamos a eliminarlas de las ecuaciones anteriores. Si restamos la ec. (66) de las ecs. (64) y (65) podemos eliminar y_e

$$y^{(3)} - y^{(1)} = C(h_1^n - h_3^n),$$
$$y^{(3)} - y^{(2)} = C(h_2^n - h_3^n).$$

Si tomamos el cociente de estas dos ecuaciones podemos eliminar C y nos queda que

$$\frac{y^{(3)} - y^{(1)}}{y^{(3)} - y^{(2)}} = \frac{h_1^n - h_3^n}{h_2^n - h_3^n}. \tag{67}$$

La expresión anterior nos dice que la razón de las diferencias entre las soluciones numéricas debe tener un valor constante que depende únicamente del los tamaños de paso. Si los tamaños de paso se escogen de forma que $h_2 = \frac{1}{2}h_1$ y $h_3 = \frac{1}{2}h_2 = \frac{1}{4}h_1$, la expresión anterior se reduce a

$$\begin{aligned}\frac{y^{(3)} - y^{(1)}}{y^{(3)} - y^{(2)}} &= \frac{h_1^n - \frac{h_1^n}{4^n}}{\frac{h_1^n}{2^n} - \frac{h_1^n}{4^n}}, \\ &= \frac{1 - \frac{1}{4^n}}{\frac{1}{2^n} - \frac{1}{4^n}}, \\ &= \frac{4^n - 1}{2^n - 1}.\end{aligned} \tag{68}$$

En este caso vemos que la razón entre las diferencias ni siquiera depende del tamaño de paso sino del orden del método numérico. Para el caso de Euler mejorado, $n = 2$, por lo tanto

$$\frac{y^{(3)} - y^{(1)}}{y^{(3)} - y^{(2)}} = 5. \tag{69}$$

Para el método de Runge-Kutta de 4to orden, $n = 4$, por lo tanto

$$\frac{y^{(3)} - y^{(1)}}{y^{(3)} - y^{(2)}} = 17. \tag{70}$$

También podemos verificar que para el método de Euler donde $n = 1$, la razón de convergencia sería 3.

Viernes 23 de septiembre de 2022

12. Ecuación de onda en 1D

La ecuación de onda en una dimensión es

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{v^2}\frac{\partial^2 u}{\partial t^2}, \tag{71}$$

donde v es la velocidad de la onda y $u = u(x,t)$ es cualquier magnitud física que presenta comportamiento ondulatorio. Puede ser el desplazamiento vertical de una cuerda tensada en sus extremos, un campo eléctrico, la presión de una onda sonora, etc.

Esta ecuación tiene una solución exacta y también se presta para soluciones numéricas. Vamos a utilizar el método de diferencias finitas para resolver esta ecuación. Lo primero que hacemos es utilizar una aproximación de la segunda derivada de una función

$$\frac{d^2}{dx^2}f(x) \approx \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}. \tag{72}$$

Para verificar esta expresión, podemos hacer una expansión en series de Taylor de la siguiente forma

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2!}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + \frac{1}{4!}h^4 f^{(4)}(x) + O(h^5), \\ f(x-h) &= f(x) - hf'(x) + \frac{1}{2!}h^2 f''(x) - \frac{1}{3!}h^3 f'''(x) + \frac{1}{4!}h^4 f^{(4)}(x) + O(h^5), \end{aligned}$$

Sumando las cantidades obtenemos

$$f(x-h) - 2f(x) + f(x+h) = h^2 f''(x) + \frac{2}{4!}h^4 f^{(4)}(x) + O(h^6), \tag{73}$$

despejando $f''(x)$ nos queda

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} - \frac{1}{12}h^2 f^{(4)}(x) + \cdots. \tag{74}$$

Esta expresión nos dice que la aproximación de la segunda derivada tiene un error de truncamiento del orden h^2.

Aplicamos esta fórmula a las derivadas respecto de x y t en la ecuación de onda y le llamamos δt y δx al tamaño de paso h respectivo

$$\begin{aligned} \frac{\partial^2 u(x,t)}{\partial x^2} &= \frac{u(x-\delta x,t) - 2u(x,t) + u(x+\delta x,t)}{\delta x^2}, \\ \frac{\partial^2 u(x,t)}{\partial t^2} &= \frac{u(x,t-\delta t) - 2u(x,t) + u(x,t+\delta t)}{\delta t^2}. \end{aligned}$$

En este punto conviene simplificar la notación. Dado que trabajaremos con una versión discretizada de la función $u(x,t)$, es decir, que los valores de u se conocen únicamente en los puntos $x_i = i\,\delta x$ y $t_j = j\,\delta t$ (siendo i y j enteros), llamaremos u_{ij} a la función $u(x,t)$ evaluada en los puntos (x_i, t_j), es decir

$$u(x_i, t_j) = u_{ij}. \tag{75}$$

En esta notación las derivadas quedan escritas como

$$\begin{aligned} \frac{\partial^2 u(x,t)}{\partial x^2} &= \frac{u_{i-1,j} - 2u_{ij} + u_{i+1,j}}{\delta x^2}, \\ \frac{\partial^2 u(x,t)}{\partial t^2} &= \frac{u_{i,j-1} - 2u_{ij} + u_{i,j+1}}{\delta t^2}. \end{aligned} \tag{76}$$

Ahora sustituimos las aproximaciones de las segundas derivadas (76) en la ecuación de onda (71)

$$\frac{u_{i-1,j} - 2u_{ij} + u_{i+1,j}}{\delta x^2} = \frac{1}{v^2}\frac{u_{i,j-1} - 2u_{ij} + u_{i,j+1}}{\delta t^2}.$$

Procedemos a despejar el término $u_{i,j+1}$

$$u_{i,j+1} = 2u_{ij} - u_{i,j-1} + \left(\frac{v\,\delta t}{\delta x}\right)^2 (u_{i-1,j} - 2u_{ij} + u_{i+1,j}),$$

$$u_{i,j+1} = 2(1-\alpha^2)u_{ij} + \alpha^2(u_{i-1,j} + u_{i+1,j}) - u_{i,j-1}, \tag{77}$$

donde $\alpha = v\,\delta t/\delta x$. La ec. (77) es esencialmente la fórmula que nos permite resolver la ecuación de onda junto con las condiciones iniciales y de frontera. Las condiciones iniciales están dadas como

$$u(x,0) = f(x), \qquad \frac{\partial u(x,0)}{\partial t} = g(x). \tag{78}$$

La primera condición implica que

$$u_{i0} = f(x_i), \tag{79}$$

es decir que tenemos todos los valores de u para $t = 0$ o bien, $j = 0$. La segunda condición la podemos escribir como

$$\left.\frac{u(x,t+\delta t) - u(x,t)}{\delta t}\right|_{t=0} = \left.\frac{u_{i,j+1} - u_{ij}}{\delta t}\right|_{j=0} = \frac{u_{i1} - u_{i0}}{\delta t} = g(x_i), \tag{80}$$

lo cual implica que

$$u_{i1} = u_{i0} + g(x_i)\delta t. \tag{81}$$

Las condiciones de frontera están dadas como

$$u(0,t) = w(t), \qquad u(L,t) = z(t). \tag{82}$$

La evaluación de estas expresiones es directa, la podemos escribir como

$$u_{0j} = w(t_j), \qquad u_{Nj} = z(t_j), \tag{83}$$

donde N es el último punto en la dirección x del dominio considerado.

La fig. 9 ilustra el dominio de la solución en la ecuación de onda unidimensional. Los puntos negros son aquellos en donde se calcula en valor de u utilizando la ec. (77). Esta fórmula se puede utilizar ya que los valores de u en el lado derecho de dicha ecuación son conocidos. Los valores en los puntos verdes están dados por las condiciones iniciales, mientras que los valores en los puntos rojos se conocen gracias a las condiciones de frontera. Los puntos con un círculo alrededor muestran el *stencil* del la ec. (77), es decir, aquellos puntos que están relacionados mediante dicha fórmula en una iteración.

12.1. Estabilidad

Este método para resolver la ecuación de onda tiene una condición de estabilidad que es

$$\alpha < 1 \qquad \Longrightarrow \qquad \frac{v\,\delta t}{\delta x} < 1. \tag{84}$$

Esto establece una relación entre la discretización en espacio y tiempo. Sabiendo la velocidad de la onda v y δx, el valor de δt esta dado como

$$\delta t < \frac{\delta x}{v}. \tag{85}$$

Normalmente se toma un valor tal que $\delta t = \frac{1}{2}\delta x/v$, es decir, $\alpha = \frac{1}{2}$.

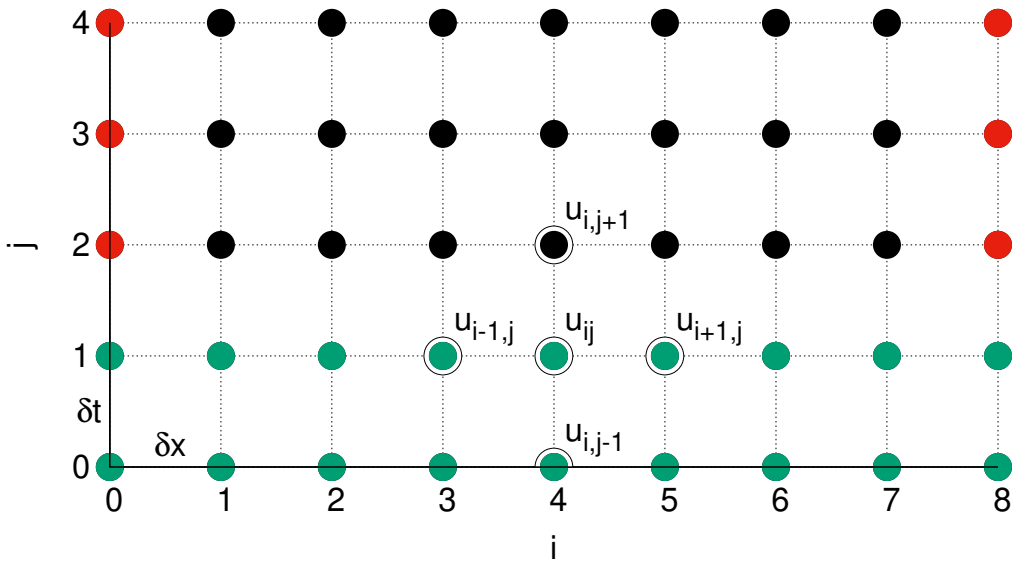

Figura 9: Esquema de puntos en el dominio de la solución $u(x_i, t_j)$ para la ecuación de onda en 1 dimensión.

Viernes 30 de septiembre de 2022

12.2. Ejemplos

Analicemos algunos casos de soluciones con la ecuación de onda en una dimensión.

1. Condiciones iniciales, con los extremos fijos

$$f(x) = e^{-\sigma(x-L/2)^2}, \qquad g(x) = 0, \tag{86}$$

$$f(x) = 0, \qquad g(x) = e^{-\sigma(x-L/2)^2} \tag{87}$$

2. Condiciones iniciales nulas, $f(x) = 0$, $g(x) = 0$ con condiciones de frontera

$$u(0,t) = w(t) = e^{\sigma(t-t_0)^2}, \qquad u(L,t) = 0 \tag{88}$$

$$u(0,t) = w(t) = 0, \qquad u(L,t) = z(t) = e^{\sigma(t-t_0)^2} \tag{89}$$

3. Interferencia constructiva

$$u(0,t) = w(t) = e^{\sigma(t-t_0)^2}, \qquad u(L,t) = z(t) = e^{\sigma(t-t_0)^2} \tag{90}$$

4. Interferencia destructiva

$$u(0,t) = w(t) = e^{\sigma(t-t_0)^2}, \qquad u(L,t) = z(t) = -e^{\sigma(t-t_0)^2} \tag{91}$$

12.3. Profiling

En inglés *profiling* se refiere a cuantificar qué tanto tiempo tarda cada parte del programa en ejecutarse. Esto es muy útil para optimizar la velocidad y detectar cuellos de botella en el programa. Para obtener esta información hacemos lo siguiente:

1. Compilamos con `g++ -pg programa.cpp`
2. Ejecutamos el programa con `./a.out` (o el nombre que se le asigne)
3. Al terminar habrá un archivo llamado `gmon.out` con los resultados en formato binario
4. Para generar la tabla de tiempo escribimos `gprof a.out > tiempos`
5. Por último abrimos el archivo `tiempos` en un editor de texto o con el comando `less`.

Martes 4 de octubre de 2022

13. Ecuación de onda en 2 dimensiones

La ecuación de onda en dos dimensiones es útil en el análisis de propagación de ondas en una superficie plana o en un volumen con simetría a lo largo de una dimensión. En coordenadas cartesianas, la ecuación de onda bidimensional se escribe así

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{1}{v^2}\frac{\partial^2 u}{\partial t^2}, \tag{92}$$

donde $u = u(x, y, t)$ es una función de las coordenadas x, y y del tiempo t.

La aplicación del método de diferencias finitas a esta ecuación es bastante directa. Solamente debemos sustituir la $\partial^2 u/\partial y^2$ por la aproximación de diferencias finitas. Es decir

$$\frac{\partial^2 u}{\partial y^2} \to \frac{u(x, y - \delta y, t) - 2u(x, y, t) + u(x, y + \delta y, t)}{\delta y^2}. \tag{93}$$

Para simplificar la notación ahora utilizaremos

$$u(x_i, y_j, t_k) = u_{ijk}. \tag{94}$$

Con esta notación, las derivadas las representamos de la siguiente manera

$$\begin{aligned}
\frac{\partial^2 u(x, y, t)}{\partial x^2} &= \frac{u_{i-1,j,k} - 2u_{ijk} + u_{i+1,jk}}{\delta x^2}, \\
\frac{\partial^2 u(x, y, t)}{\partial y^2} &= \frac{u_{i,j-1,k} - 2u_{ijk} + u_{i,j+1,k}}{\delta y^2}, \\
\frac{\partial^2 u(x, y, t)}{\partial t^2} &= \frac{u_{ij,k-1} - 2u_{ijk} + u_{ij,k+1}}{\delta t^2}.
\end{aligned} \tag{95}$$

Sustituimos las expresiones (95) en la ec. (92) para obtener

$$\frac{u_{i-1,j,k} - 2u_{ijk} + u_{i+1,jk}}{\delta x^2} + \frac{u_{i,j-1,k} - 2u_{ijk} + u_{i,j+1,k}}{\delta y^2} = \frac{1}{v^2}\frac{u_{ij,k-1} - 2u_{ijk} + u_{ij,k+1}}{\delta t^2}. \tag{96}$$

Ahora despejamos el término $u_{ij,k+1}$

$$u_{ij,k+1} = 2u_{ijk} - u_{ik,k-1} + v^2\,\delta t^2 \left(\frac{u_{i-1,j,k} - 2u_{ijk} + u_{i+1,jk}}{\delta x^2} + \frac{u_{i,j-1,k} - 2u_{ijk} + u_{i,j+1,k}}{\delta y^2} \right).$$

Para simplificar las cosas en esta ocasión tomaremos $\delta x = \delta y$. Con lo que la expresión anterior se reduce a

$$u_{ij,k+1} = 2u_{ijk} - u_{ik,k-1} + \alpha^2 \left(u_{i-1,j,k} + u_{i+1,jk} + u_{i,j-1,k} + u_{i,j+1,k} - 4u_{ijk} \right), \tag{97}$$

donde $\alpha = v\,\delta t/\delta x$. Esta ecuación nos da la fórmula iterativa para calcular la solución numérica.

13.1. Condiciones iniciales

Las condiciones iniciales son

$$u(x, y, 0) = f(x, y), \qquad \frac{\partial u(x, y, 0)}{\partial t} = g(x, y). \tag{98}$$

De forma similar a la ecuación de onda en una dimensión, la primera condición implica que

$$u_{ij0} = f(x_i, y_i). \tag{99}$$

La segunda condición nos dice que

$$u_{ij1} = u_{ij0} + g(x_i, y_i)\,\delta t. \tag{100}$$

13.2. Condiciones de frontera

Para un plano en un sistema de coordenadas cartesiano, las fronteras son las cuatro rectas que limitan dicho plano. Si consideramos que las dimensiones del plano son L_x, L_y y alineamos una esquina con el origen de coordenadas, las condiciones de frontera son

$$\begin{aligned} u(x,0,t) &= p_1(x,t), \\ u(x,L_y,t) &= p_2(x,t), \\ u(0,y,t) &= q_1(y,t), \\ u(L_x,y,t) &= q_2(y,t). \end{aligned} \tag{101}$$

Lo anterior nos provee de los valores de u en los bordes del dominio

$$\begin{aligned} u_{i0k} &= p_1(x_i,t_k), \\ u_{i,N_y,k} &= p_2(x_i,t_k), \\ u_{0jk} &= q_1(y_j,t_k), \\ u_{N_x,0k} &= q_2(y_j,t_k), \end{aligned} \tag{102}$$

donde N_x y N_y es el número de puntos que utilizamos en cada dimensión.

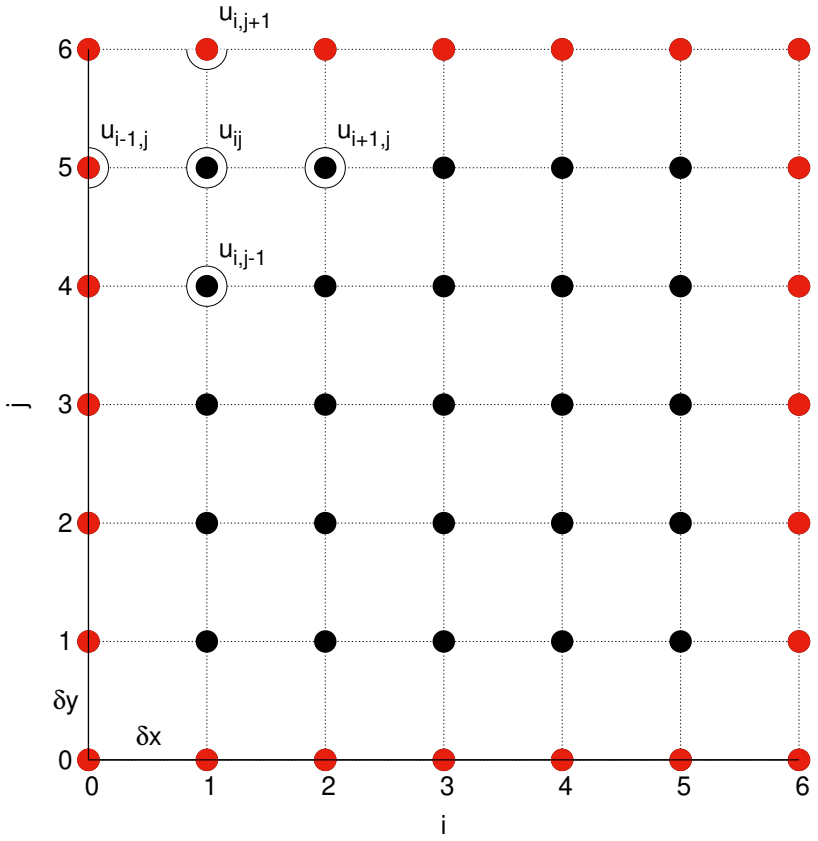

Figura 10: Esquema de puntos en el dominio de la solución $u(x_i, y_j, t_k)$ para la ecuación de onda en 2 dimensiones. Los puntos rojos marcan la frontera del dominio.

Viernes 14 de octubre de 2022

14. Paralelizando ciclos con OpenMP

OpenMP provee una manera muy simple de utilizar varios núcleos del procesador de la computadora. Después de bajar la librería OpenMP, para utilizarlo necesitamos tomar hacer lo siguiente.

1. Colocar `#include <omp.h>` al inicio del programa.
2. Colocar `#pragma omp parallel for` al inicio del ciclo `for` que se quiere ejecutar en paralelo
3. Compilar con la opción `-fopenmp`
4. El número de núcleos a utilizar se puede especificar con la variable de ambiente `OMP_NUM_THREADS`. Para ello basta con escribir en la terminal, por ejemplo
   `$> OMP_NUM_THREADS=2`
   eso hará que el programa utilice 2 núcleos al correr en paralelo.

Para tener más información sobre OpenMP pueden consultar la información breve que se encuentra en el artículo de wikipedia `https://en.wikipedia.org/wiki/OpenMP`. Un tutorial bastante conciso que se puede consultar es "Introduction to the OpenMP with C++ and some integrals approximation"
`https://medium.com/swlh/introduction-to-the-openmp-with-c-and-some-integrals-approximation-a7f03`

Martes 19 de octubre de 2022

15. Ondas en medios diferentes

Utilizando nuestro programa para resolver la ecuación de onda en 2D, vamos a simular la propagación de ondas en dos medios diferentes. La forma de caracterizar los medios es mediante la velocidad de propagación de la onda. La forma más sencilla de implementar esta propiedad es definiendo un valor para la velocidad en cada punto del dominio, es decir

$$v \to v_{ij}. \tag{103}$$

Supongamos que dividimos el domino en dos partes, de tal forma que

$$v_{ij} = \begin{cases} v_1, & x < L_x/2 \\ v_2, & x \geq L_x/2 \end{cases} \tag{104}$$

Para implementar esto en nuestro programa en lugar de utilizar la velocidad como una constante, declaramos un arreglo de dos dimensiones que contenga los valores de la velocidad tal como la deseamos.

15.1. Condición de frontera

Podemos utilizar las condiciones de frontera para simular que perturbamos ligeramente el medio con una onda entrante. Una condición de frontera que representa esta situación la podemos escribir como

$$u(0, y, t) = Ae^{-w(y-L_y/2)^2} \sin \omega t. \tag{105}$$

Aquí, A controla la amplitud, w fija el ancho del pulso y ω se encarga de la velocidad de oscilación.

16. Comandos y funciones útiles

Casi todos los comandos de la terminal de linux tienen ayuda o documentación que podemos revisar utilizando el comando `man`, por ejemplo `man paste` despliega la ayuda o documentación del comando `paste`.

`comando > salida`

Redirige el standard output producido por `comando` al archivo llamado `salida`. El signo `>` siempre sobrescribe lo que haya en `salida`, si el archivo no existe entonces lo crea. Si lo que uno quiere es continuar agregando la salida de `comando` a un archivo ya existente se puede usar `comando >> salida`, esto agrega el standard output al final de lo que haya en el archivo `salida`.

`paste archivo1 archivo2`

Toma dos archivos de texto con datos en columnas y coloca todas las columnas de forma contigua.

`cut -f 1,2 archivo`

Selecciona las columnas 1 y 2 de `archivo`

`diff archivo1 archivo2`

Toma dos archivos de texto y los compara línea por línea. Es una manera fácil de verificar sidos archivos son iguales.

`less archivo`

Es una forma muy rápida de ver lo que hay en `archivo`.

`ln`

Crea un "link" hacia un archivo o carpeta que actúa como si el archivo o carpeta estuviera copiado en el lugar donde el link es creado.

`time`

Mide el tiempo que toma la ejecución de un comando en la terminal.