# Introducción a la solución numérica de ODE's

Guía computacional 1 - Mecánica Clásica 2016 - Clase G. Mindlin

Ignacio Poggi - L.U: 567/07 - ignaciop.3@gmail.com

12 de abril de 2016

## 1. Enunciado

En la sección de materiales adicionales de la cátedra se encuentra un programa principal y un integrador Runge-Kutta de orden 4 (rk4) en lenguaje C. En el programa principal se encuentra escrita una ecuación diferencial a integrar, los parámetros y las condiciones iniciales. Sobre este código van a trabajar en las siguientes actividades realizando las modificaciones pertinentes para su problema en particular.

En ubuntu es posible compilar y ejecutar el código directamente desde una terminal abierta en una carpeta que contenga tanto el programa principal como el integrador rk4:

```
gcc ODE_ejX.c -o ode_ejX -lm rk4.c
```

```
./ode_ejX
```

Se obtendrá como salida un archivo llamado $ejX.dat$ , donde $X$ es el número del ejercicio, que contiene el resultado de la integración. Los resultados pueden ser analizados gráficamente mediante un graficador, en nuestro caso utilizaremos **gnuplot** que se controla mediante comandos en terminal.

## Actividad 1

Editar el código de ODE.c para analizar los siguientes puntos:

- Cómo varía el resultado según el paso de integración. Programe una integración con el método de Euler y compare.
- Analizar cómo evoluciona el sistema dadas distintas condiciones iniciales.

Qué tipo de conclusiones puede obtener a partir de los análisis anteriormente realizados.

# Actividad 2 - Oscilador armónico amortiguado

El oscilador armónico amotiguado es un problema del cual se conoce la solución analítica cuya ecuación diferencial que rige el movimiento es:

$$\frac{d^2x}{dt^2} + 2\gamma\frac{dx}{dt} + \omega_0^2 x = 0 \tag{1}$$

Estudie numéricamente las soluciones del sistema según la relación de los parámetros, para ello: escriba la ecuación de segundo orden como dos ecuaciones de primer orden, varíe $\gamma$ y $\omega$ e integre. También analice distintas condiciones iniciales. Compare con lo conocido de la solución analítica, para ello grafique como evoluciona la posición en el tiempo, la velocidad y cuál es la trayectoria en el espacio de fases $x\dot{x}$.

# Actividad 3 - Oscilador de Van der Pol

Es un tipo de oscilador con un amortiguamiento no lineal descripto a principio de siglo por Van der Pol quien estudió circuitos eléctricos con componentes no lineales obteniendo la ecuación:

$$\frac{d^2x}{dt^2} + \mu(x^2 - 1)\frac{dx}{dt} + x = 0 \tag{2}$$

Este sistema presenta soluciones oscilatorias para ciertos valores del parámetro $\mu$ que son conocidas como oscilaciones de relajación. Esta ecuación tiene una importancia en la ciencia ya que fue usada en distintos campos para describir por ejemplo, el comportamiento de una falla tectónica o el potencial de acción de una neurona. Esto se debe a que el sistema según los valores de $x$ presenta un amortiguamiento positivo (como el de la actividad 2 donde el sistema pierde energía), y para otros presenta un amortiguamiento "negativo"donde el sistema gana energía. Esto produce que eventualmente la energía perdida en un ciclo sea igual a la ganada generando oscilaciones autosostenidas. Este sistema se verá con más detalle avanzado el curso, en esta práctica se propone realizar un acercamiento de forma numérica para tener cierta comprensión de cómo se comporta el mismo.

- Escriba el sistema como dos ecuaciones de primer orden.
- Inspeccione numéricamente las soluciones posibles del sistema, estudie como varían según la variación del parámetro $\mu$. Para ello grafique la trayectoria $x$ en función del tiempo, la velocidad $\dot{x}$ en función del tiempo y también el espacio de fases $x\dot{x}$.
- Modifique también las condiciones iniciales y estudie numéricamente las respuestas del sistema. Para ello grafique la trayectoria $x$ en función del tiempo, la velocidad $\dot{x}$ en función del tiempo y también el espacio de fases $x\dot{x}$.

# A entregar

Se deberá entregar un trabajo de la actividad 2 y 3, con los códigos, los gráficos obtenidos para las integraciones numéricas propuestas y el correspondiente análisis para cada caso.

# 2. Análisis de datos y conclusiones

Las unidades utilizadas a lo largo de este trabajo son las siguientes:

- Tiempo $[t] = s$
- Posición $[x] = cm$
- Velocidad $[\dot{x}] = \frac{cm}{s}$
- Constante de frecuencia natural $[\omega_0] = Hz$
- Constante de amortiguamiento $[\gamma] = \frac{Ns}{cm}$

## 2.1. Oscilador armónico amortiguado

El oscilador armónico amotiguado es el caso generalizado de los osciladores armónicos libres ya que su comportamiento contempla la disipación de energía pero no la presencia de fuerzas externas. Su comportamiento está dado por (1). Para obtener las soluciones de dicha ecuación, se propone una con la forma $x(t) = e^{zt}$, con $z \in \mathbb{C}$ (luego $\dot{x}(t) = ze^{zt}$ y $\ddot{x}(t) = z^2e^{zt}$) **[I]**.

Reemplazando $x(t)$ y sus derivadas en (1), se obtiene la siguiente ecuación cuadrática para $z$:

$$(z^2 + 2\gamma z + \omega_o^2)e^{zt} = 0$$

Esta ecuación es igual a 0 si y solo si $(z^2 + 2\gamma z + \omega_o^2) = 0$. Las raíces de este polinomio son:

$$z_{1,2} = \frac{-2\gamma \pm \sqrt{4\gamma^2 - 4\omega_0^2}}{2} = -\gamma \pm \sqrt{\gamma^2 - \omega_0^2}$$

Se pueden distinguir tres casos que, a continuación, se analizan por separado.

- Si $\omega_0^2 > \gamma^2$, se obtienen dos raices complejas

$$z_1 = -\gamma + i\sqrt{\omega_0^2 - \gamma^2}$$

$$z_2 = -\gamma - i\sqrt{\omega_0^2 - \gamma^2}$$

Sea $\omega_1 = \sqrt{\omega_0^2 - \gamma^2}$. Reemplazando esta nueva frecuencia en la solución propuesta y tomando su parte real:

$$x(t) = e^{-\gamma t}(acos(\omega_1 t) + bsin(\omega_1 t)) \tag{3}$$

$$\dot{x}(t) = e^{-\gamma t}((\omega_1 b - \gamma a)acos(\omega_1 t) + (\omega_1 a - \gamma b)bsin(\omega_1 t))$$

donde $a$ y $b$ se determinan con las condiciones iniciales:

$$x(0) = a, \dot{x}(0) = \omega_1 b - \gamma a$$

Luego, la ecuación (3) queda de la siguiente manera [**II**]:

$$x(t) = e^{-\gamma t}(x(0)cos(\omega_1 t) + \frac{\dot{x}(0) + \gamma x(0)}{\omega_1} sin(\omega_1 t)) \tag{4}$$

Este movimiento corresponde a una oscilación armónica de frecuencia $\omega_1$, diferente de la frecuencia natural $\omega_0$; y se denomina *movimiento oscilatorio subamortiguado.*

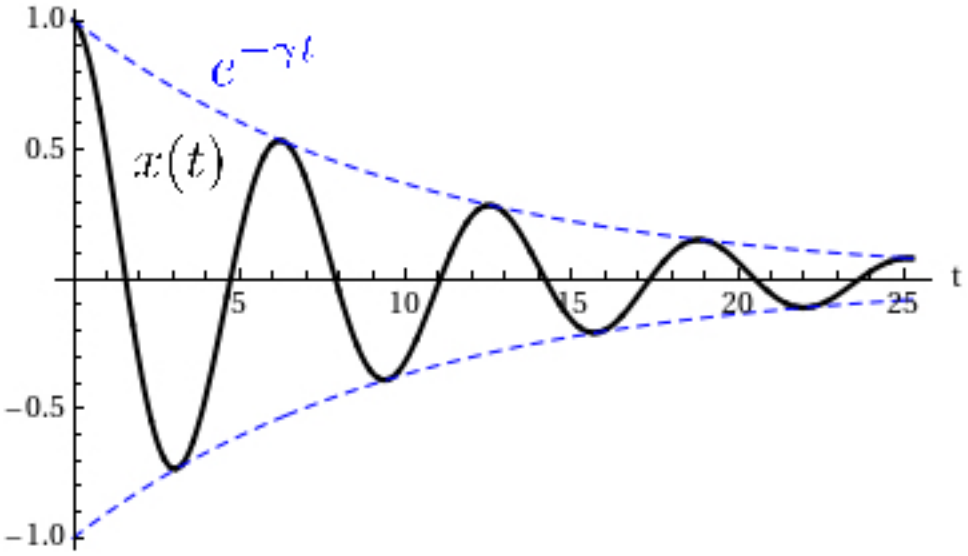


Figura 1: Movimiento oscilatorio subamortiguado.
Se observa como la curva esta modulada por un termino exponencial relacionado con la constante de amortiguamiento $\gamma$.

- Si $\omega_0^2 < \gamma^2$, se obtienen dos raices reales

$$z_1 = -\gamma + \sqrt{\gamma^2 - \omega_0^2}$$

$$z_2 = -\gamma - \sqrt{\gamma^2 - \omega_0^2}$$

En este caso, la ecuación $x(t)$ quedará expresada en términos de cosenos y senos hiperbólicos. Para las condiciones iniciales se procede como en el caso del oscilador subamortiguado, por lo tanto la ecuación de movimiento queda [**III**]:

$$x(t) = x(0)e^{-\gamma t}cosh(\sqrt{\gamma^2 - \omega_0^2}t) + \frac{\dot{x}(0) + \gamma x(0)}{\sqrt{\gamma^2 - \omega_0^2}} e^{-\gamma t} sinh(\sqrt{\gamma^2 - \omega_0^2}t) \tag{5}$$

Este movimiento se denomina *oscilatorio sobreamortiguado.*

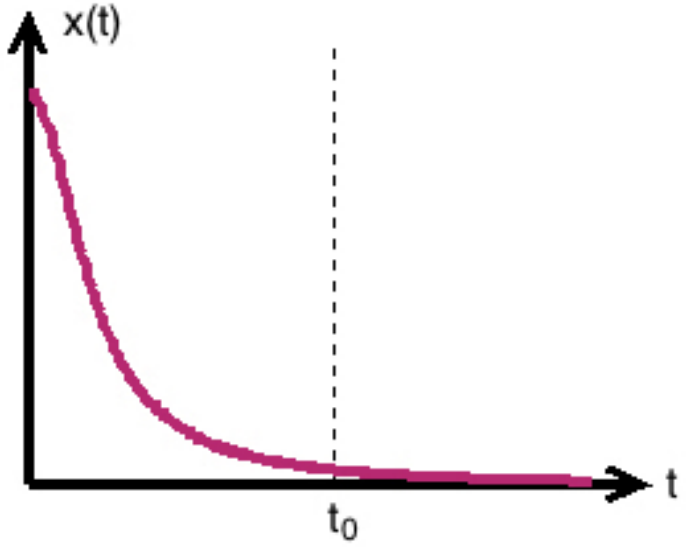


Figura 2: Esquema del movimiento oscilatorio sobreamortiguado. En $t_0$ el sistema decae a 0 sin llegar a completar un periodo de oscilación.

- Si $\omega_0^2 = \gamma^2$, se obtiene una raíz real doble:

$$z_{1,2} = -\gamma$$

Al tener una única raíz doble, considero soluciones del siguiente tipo:

$$x(t) = (a + bt)e^{-\gamma t}$$

Al imponer las condiciones iniciales sobre $x(t)$, la solución queda [**IV**]:

$$x(t) = [x(0) + (\dot{x}(0) + \gamma x(0))t]e^{-\gamma t} \tag{6}$$

Este movimiento se denomina *oscilatorio con amortiguamiento crítico.*

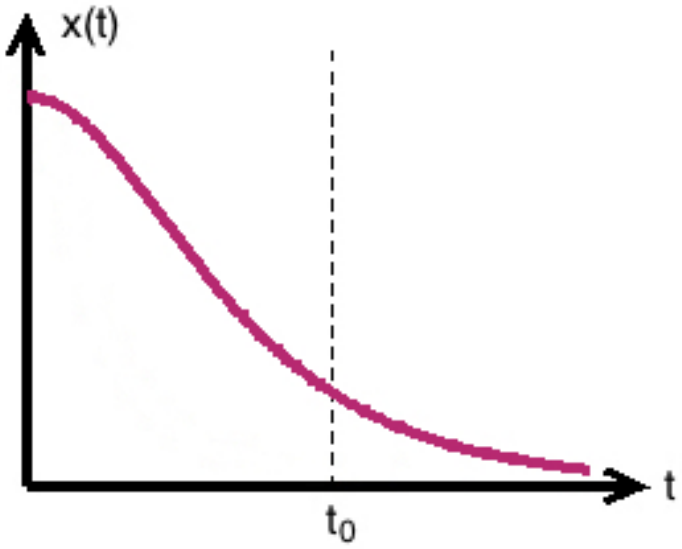


Figura 3: Esquema del movimiento oscilatorio con amortiguamiento crítico.

Para el análisis numérico, se reescribió la ecuación de segundo orden (1) como dos ecuaciones de primer orden de la siguiente manera [**V**]:

$$\begin{cases} \dot{x} = y \\ \dot{y} = -2\gamma y - \omega_0^2 x \end{cases}$$

Este sistema se implementó en el código fuente *ODE_ej2.c* (ver sección Apéndice). Se utilizó una variable $j$ dentro del bucle *for* para ir iterando los distintos valores de $\gamma$ y $\omega_0$, así como también las condiciones iniciales $x(0)$ y $\dot{x}(0)$; desde $j = 1$ hasta $j = 10$. El tiempo máximo de muestreo fue de 2 segundos.

Para destacar los casos sobre, sub y amortiguado crítico, es necesario aclarar que se utilizaron incrementos de dicha variable con un valor igual a 4, por lo tanto $j$ solo toma 3 valores durante la ejecución del programa: $j = 1$, $j = 5$ y $j = 9$; obteniendo una representación numérica y gráfica de los 3 casos mencionados anteriormente.

En el siguiente cuadro se pueden ver los valores numéricos para los parámetros y condiciones iniciales correspondientes a cada iteración:

| $j$ | $\gamma$ | $\omega_0$ | $x(0)$ | $\dot{x}(0)$ |
|---|---|---|---|---|
| 1 | 9 | 1 | 0.5 | 4.5 |
| 5 | 5 | 5 | 2.5 | 2.5 |
| 9 | 1 | 9 | 4.5 | 0.5 |

Cuadro 1: Valores de los parámetros $\gamma$, $\omega_0$ y las condiciones iniciales en cada iteración. La primera fila corresponde al caso sobreamortiguado ($\gamma^2 > \omega_0^2$), la segunda al caso de amortiguamiento crítico ($\gamma^2 = \omega_0^2$) y la tercera al caso subamortiguado ($\gamma^2 < \omega_0^2$).

Los gráficos correspondientes a estos resultados son los siguientes:

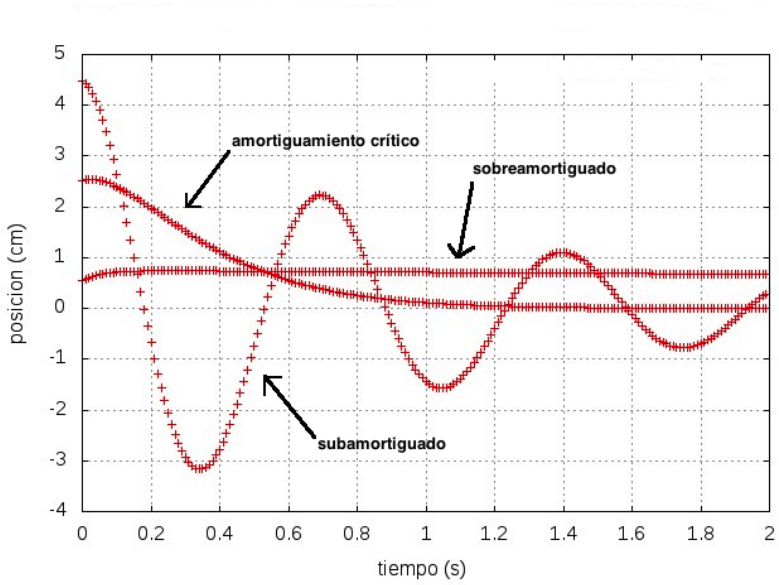


Figura 4: Posición en función del tiempo para los 3 casos estudiados del oscilador armónico amortiguado.

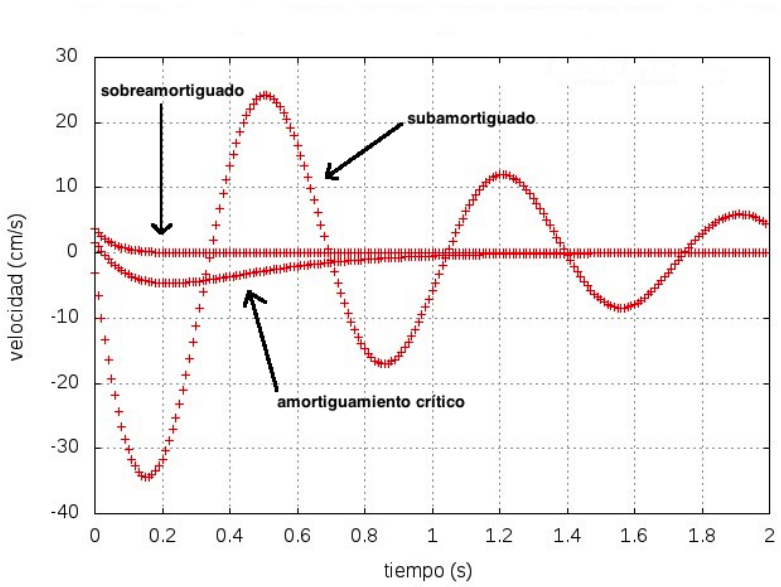


Figura 5: Velocidad en función del tiempo para los 3 casos estudiados del oscilador armónico amortiguado.

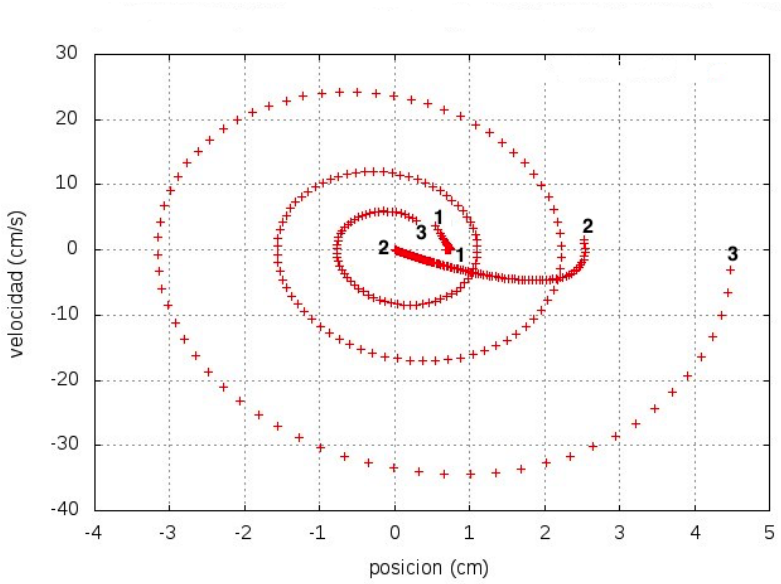


Figura 6: Espacio de fases para los 3 casos estudiados del oscilador armónico amortiguado.

Se observa en los gráficos que el integrador de Runge-Kutta de orden 4 provisto por la cátedra, y los valores escogidos para destacar los 3 casos, aproximan satisfactoriamente a las soluciones analíticas del oscilador armónico amortiguado detalladas en la introducción teórica del mismo.

También podemos ver que, en el caso sobreamortiguado, como $\gamma^2 = 81$ y $\omega_0^2 = 1$, el amortiguamiento es excesivo frente a la oscilación natural del sistema, por lo tanto la posición y velocidad decaen muy rapidamente, sin llegar a 0. Esta situación en el espacio de fases esta representada por una linea muy corta, señalada al principio y al final con el numero 2.

Para el caso de amortiguamiento crítico, $\gamma^2 = \omega_0^2 = 25$, la función $x(t)$ presenta un decaimiento exponencial, similar al caso sobreamortiguado. En un oscilador de estas características, el retorno a la posición de equilibrio se da rápidamente. En el espacio de fases se puede ver como la línea gruesa marcada con el numero 1 al principio y al final de la misma.

En el caso subamortiguado, $\gamma^2 = 1$ y $\omega_0^2 = 81$. Esto se ve reflejado en una gráfica cosenoidal con una envolvente exponencial que decae mas lentamente que en los otros dos casos, y se hace 0 en un tiempo mas largo que el tiempo de muestreo de 2 segundos.
Se observa además la gran amplitud y frecuencia inicial de esta función con respecto a los casos sobreamortiguado y amortiguado crítico, debido a los valores de $x(0)$ y $\omega_0^2$ asignados a este caso. En el espacio de fases se puede ver como la curva marcada al principio y al final con el numero 3.

## 2.2. Oscilador de Van der Pol

El oscilador de Van der Pol tiene un solo punto de equilibrio, que es el punto P(0,0), el cual siempre resulta ser inestable, pero todas las trayectorias a partir de él tienden en espiral a un ciclo límite y a éste también tienden a decaer aquellas trayectorias cuyas amplitudes son más grandes que la amplitud del ciclo límite [**VI**].

Para realizar el análisis numérico de este oscilador no-lineal, se reescribió la ecuación de segundo orden (2) como dos ecuaciones de primer orden utilizando las transformaciones de Liénard [**VII**]:

$$\begin{cases} \dot{x} = \mu(y - (x^3 - x)) \\ \dot{y} = -\frac{1}{\mu}x \end{cases}$$

Podemos destacar dos regímenes interesantes para este oscilador, dependientes del parámetro $\mu$:

- Si $\mu = 0$, no hay amortiguamiento y la ecuación (2) queda:

$$\frac{d^2x}{dt^2} + x = 0$$

  Es la fórmula del oscilador armónico simple sin pérdida de energía.

- Si $\mu > 0$, el sistema alcanzará un ciclo límite, en el que se conservará la energía. Cerca del origen $x = \frac{dx}{dt} = 0$, el sistema es inestable; y lejos del origen hay amortiguamiento.

Este último caso es el más interesante y para el cual se analizaron numéricamente dos posibilidades: amortiguamiento muy pequeño ($\mu << 1$) y amortiguamiento muy grande ($\mu >> 1$).

Este sistema se implemento en el código fuente *ODE_ej3.c* (ver sección Apéndice). Se aumento el tiempo de muestreo a 20 segundos con respecto al ejercicio anterior, para poder ver detalladamente el espacio de fases de este oscilador. Se decidió graficar con líneas continuas para facilitar la visualización de la posición, velocidad y espacio de fases de este oscilador.

- $\mu << 1$:

  En primera instancia, se dejaron las condiciones iniciales fijas ($x(0)$ = -0.01 y $\dot{x}(0) = 0.05$). El parámetro $\mu$ toma los valores $\mu_1 = 0.05$ y $\mu_2 = 0.005$ en cada ejecución. Los gráficos correspondientes a estos resultados son los siguientes:

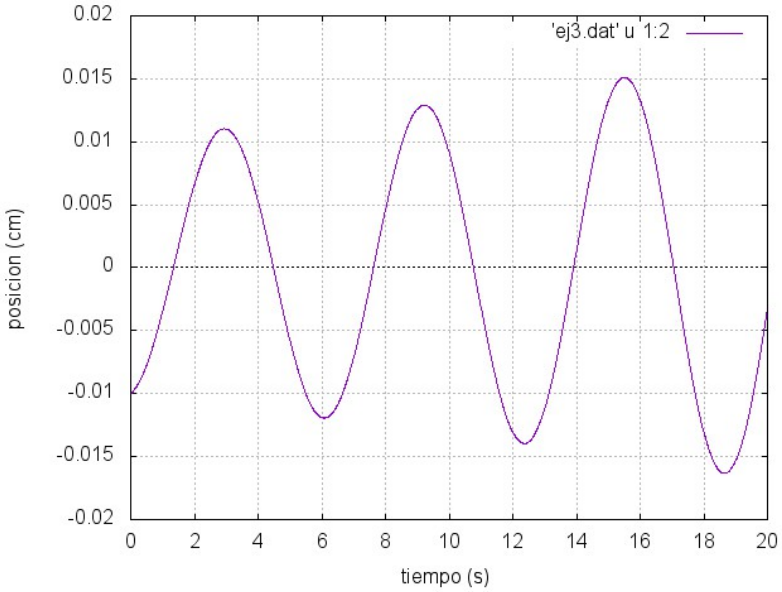


Figura 7: Posición en función del tiempo para $\mu_1 = 0.05$ y condiciones iniciales fijas.

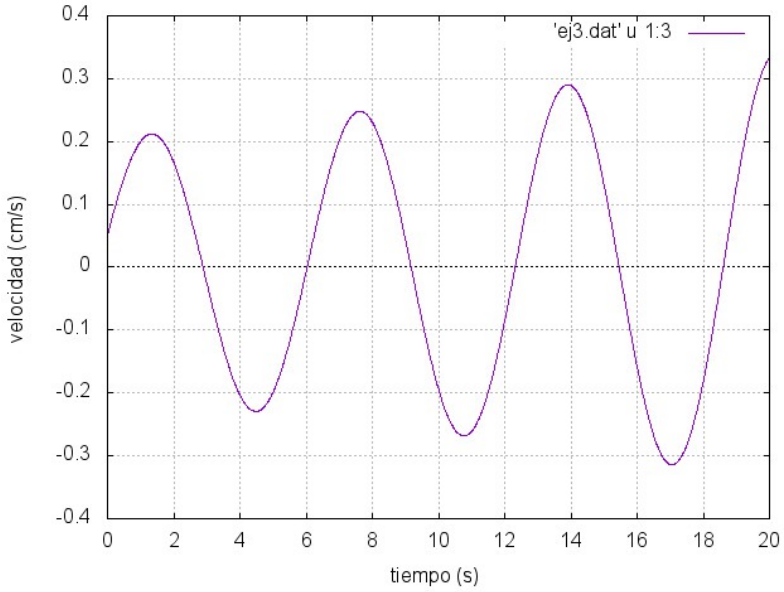


Figura 8: Velocidad en función del tiempo para $\mu_1 = 0.05$ y condiciones iniciales fijas.

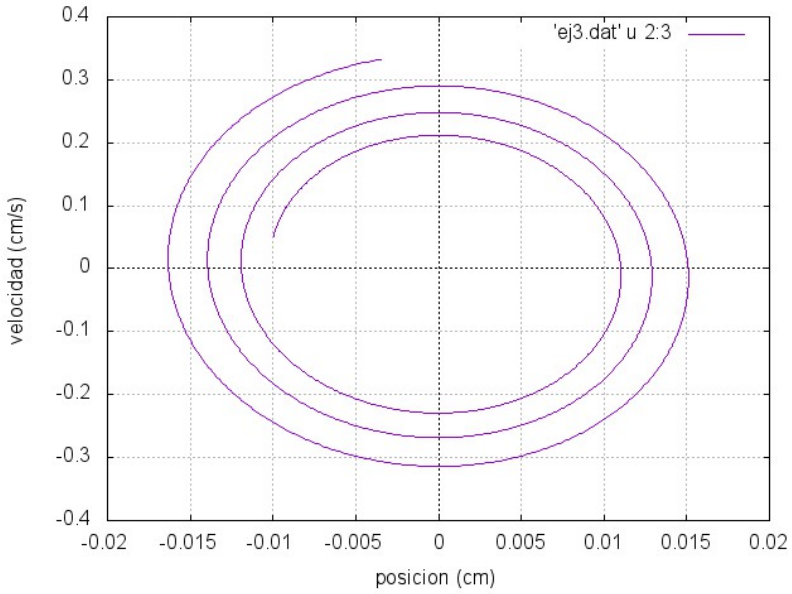


Figura 9: Espacio de fases para $\mu_1 = 0.05$ y condiciones iniciales fijas.

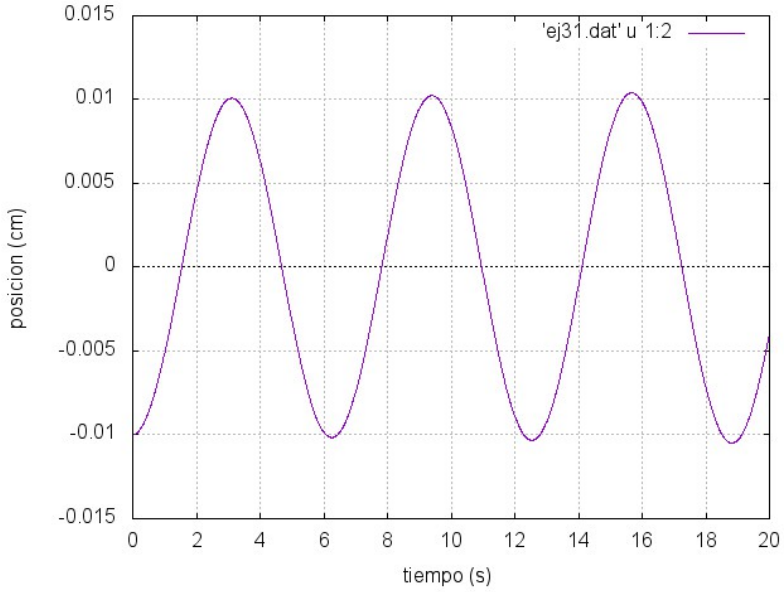


Figura 10: Posición en función del tiempo para $\mu_2 = 0.005$ y condiciones iniciales fijas.

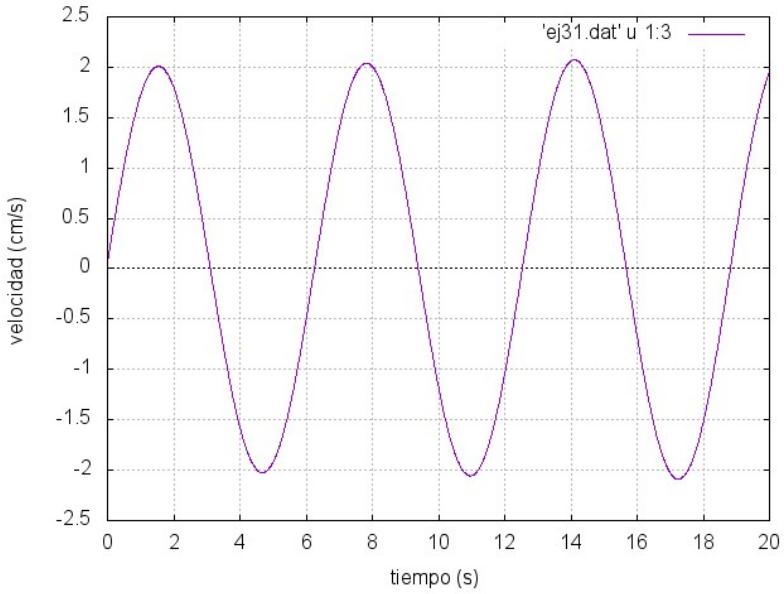


Figura 11: Velocidad en función del tiempo para $\mu_2 = 0.005$ y condiciones iniciales fijas.

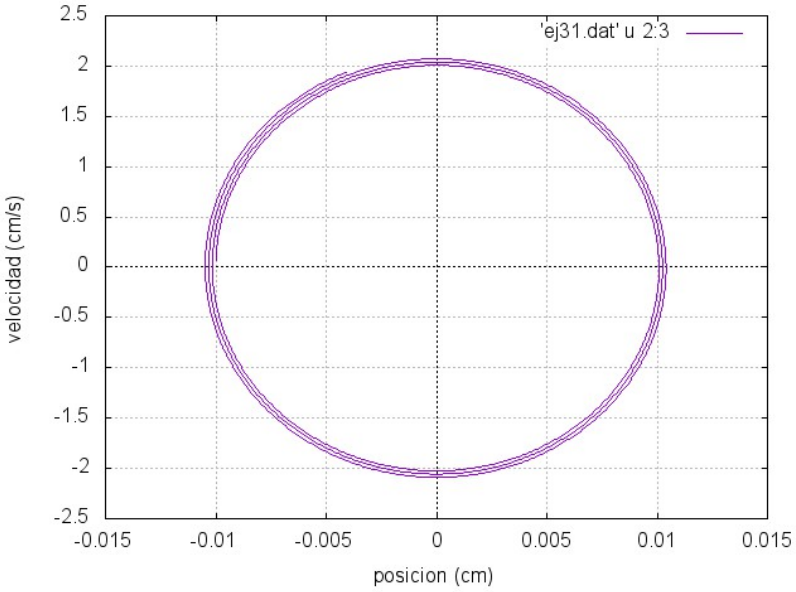


Figura 12: Espacio de fases para $\mu_2 = 0.005$ y condiciones iniciales fijas.

En una segunda instancia, las condiciones iniciales se variaron dentro de un bucle *for*, esta vez con un iterador $j$ incrementado de a 1, con los mismos valores de $\mu_1$ y $\mu_2$. La siguiente tabla muestra los valores correspondientes a este caso:

| $j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $x(0)$ | 0.2 | 0.4 | 0.6 | 0.8 | 1 | 1.2 | 1.4 | 1.6 | 1.8 | 2 |
| $\dot{x}(0)$ | -0.2 | -0.4 | -0.6 | -0.8 | -1 | -1.2 | -1.4 | -1.6 | -1.8 | -2 |

Cuadro 2: Condiciones iniciales en cada iteración para valores de $\mu_1 = 0.05$ y $\mu_2 = 0.005$.

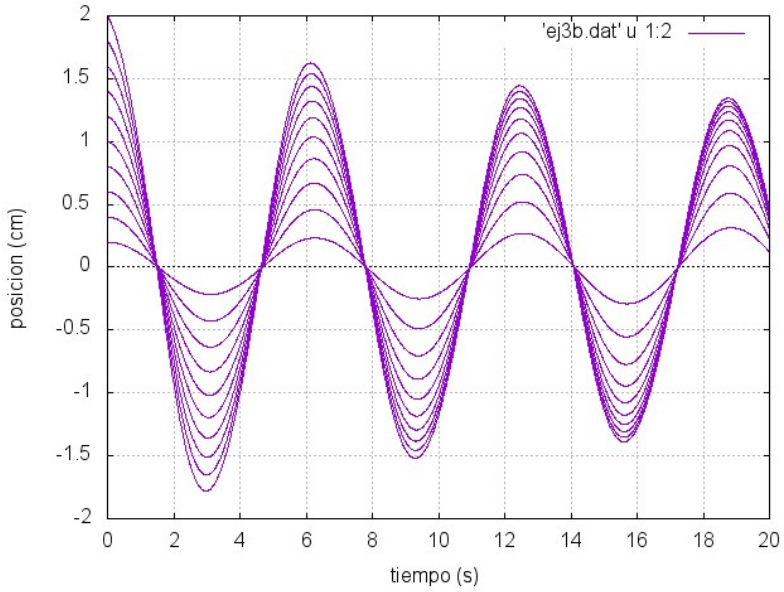


Figura 13: Posición en función del tiempo para $\mu_1 = 0.05$ y condiciones iniciales variables.

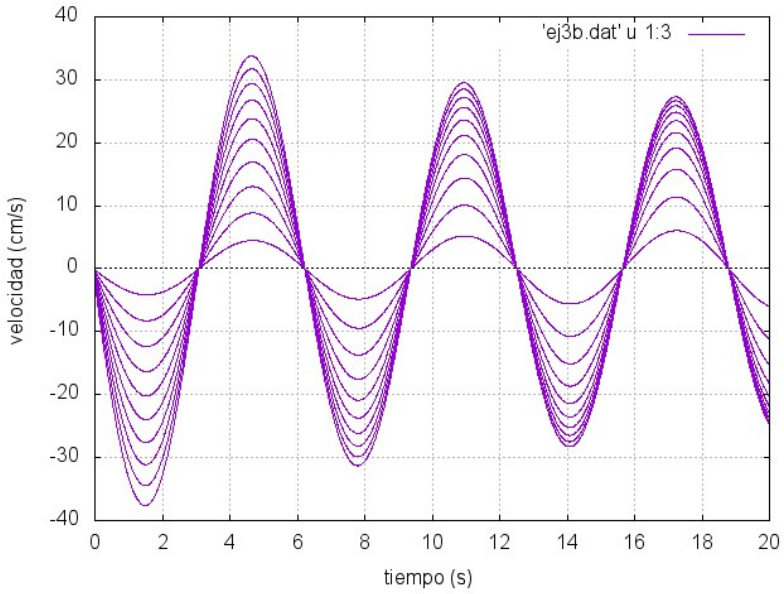


Figura 14: Velocidad en función del tiempo para $\mu_1 = 0.05$ y condiciones iniciales variables.

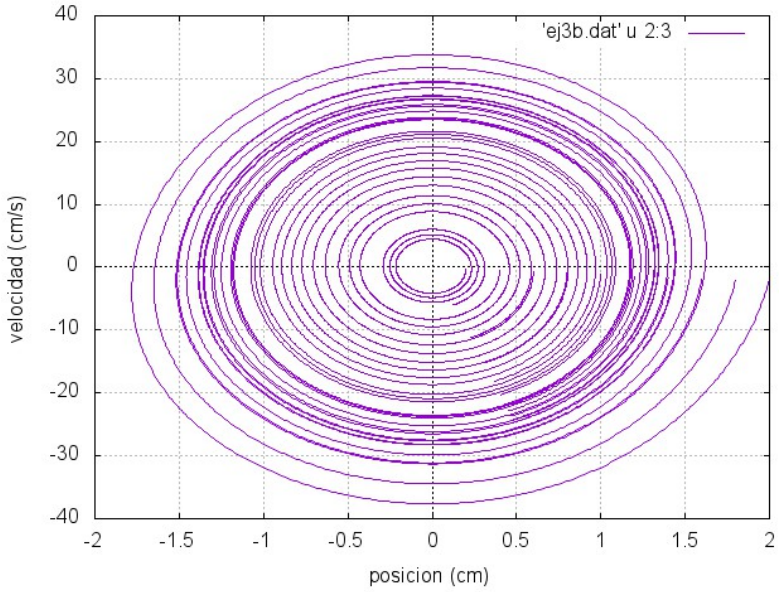


Figura 15: Espacio de fases para $\mu_1 = 0.05$ y condiciones iniciales variables.

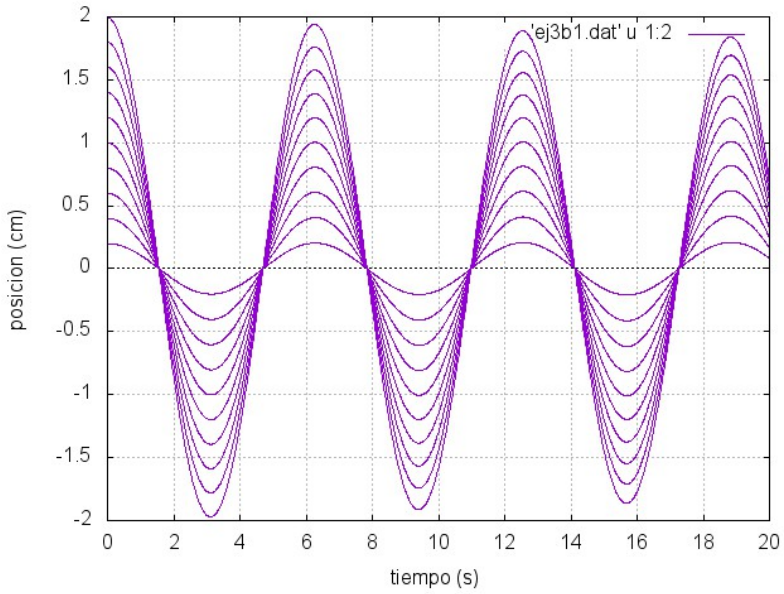


Figura 16: Posición en función del tiempo para $\mu_2 = 0.005$ y condiciones iniciales variables.

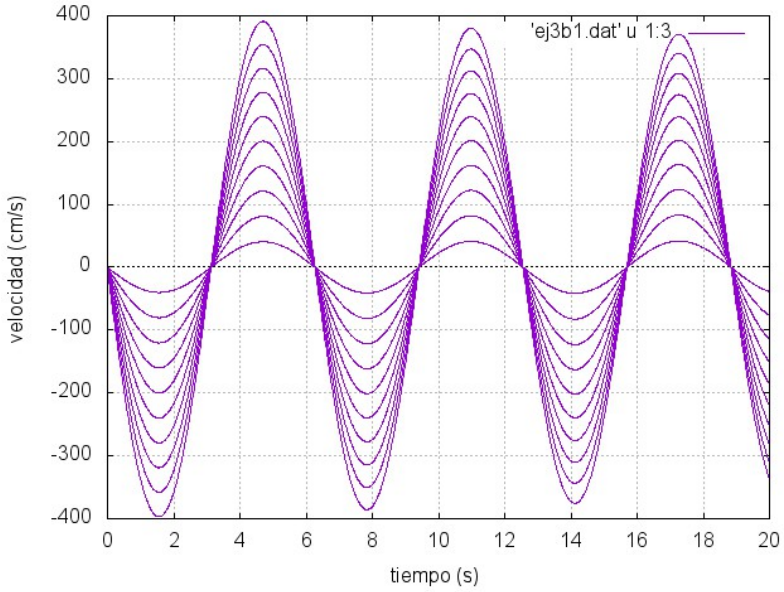


Figura 17: Velocidad en función del tiempo para $\mu_2 = 0.005$ y condiciones iniciales variables.

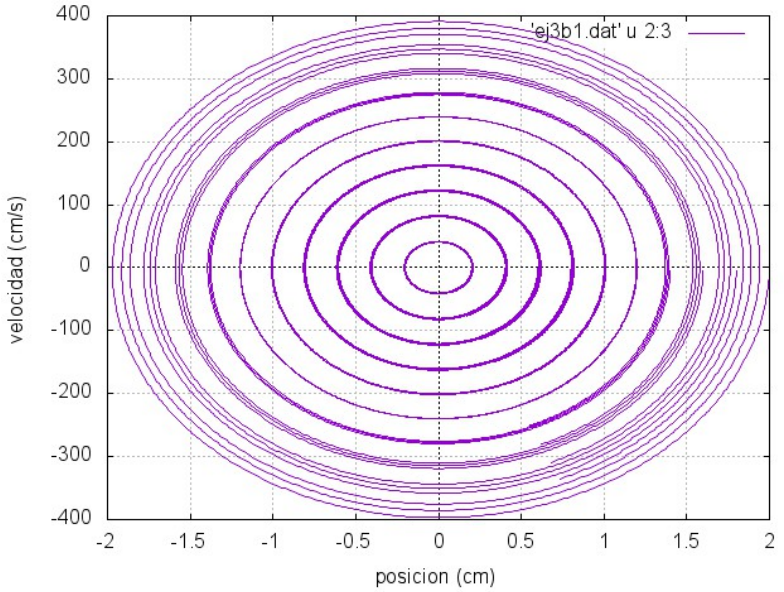


Figura 18: Espacio de fases para $\mu_2 = 0.005$ y condiciones iniciales variables.

Se puede observar que, para $\mu << 1$, el termino cuadrático de la ecuación (2) es muy pequeño y el comportamiento es similar al de un oscilador armónico amortiguado. Esto se puede ver claramente en los espacios de fases en los casos analizados; sobre todo cuando $\mu = \mu_2 = 0.005$ y las condiciones son variables, podemos distinguir varios ciclos límite representados por círculos marcados con líneas más gruesas alrededor del punto de equilibrio (0,0), a los

cuales convergen las distintas trayectorias impuestas por las condiciones iniciales variables (Figura 18). Estos resultados son consistentes con los obtenidos para las condiciones iniciales fijas (Figura 12), repitiéndose el mismo diagrama de fases con amplitudes más grandes, según el valor de la variable $j$ detallada en el Cuadro 2.

- $\mu >> 1$:

  Como en el caso anterior, primero se dejaron las condiciones iniciales fijas ($x(0)$ = -0.01 y $\dot{x}(0) = 0.05$), variando el parámetro $\mu$ en cada ejecución por separado ($\mu_1 = 5$ y $\mu_2 = 10$). Los gráficos de posición, velocidad y espacio de fases para dichos $\mu$ son los siguientes:

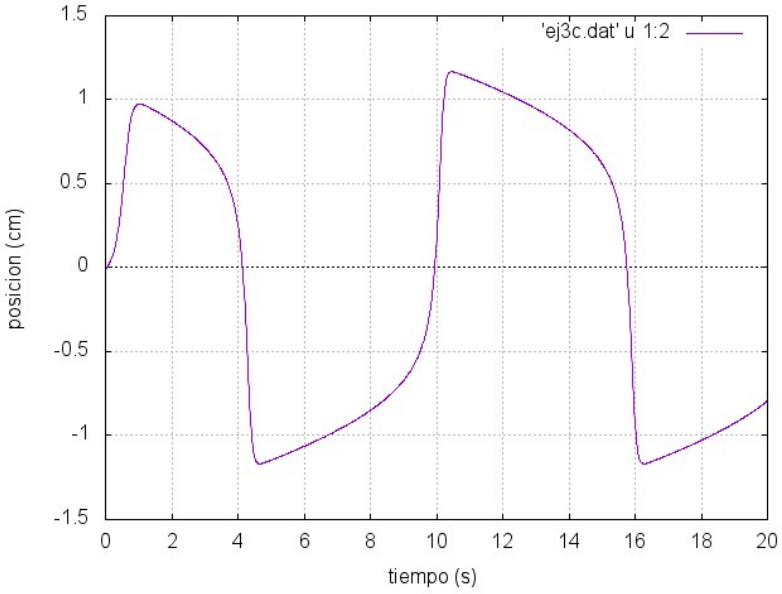


Figura 19: Posición en función del tiempo para $\mu_1 = 5$ y condiciones iniciales fijas.

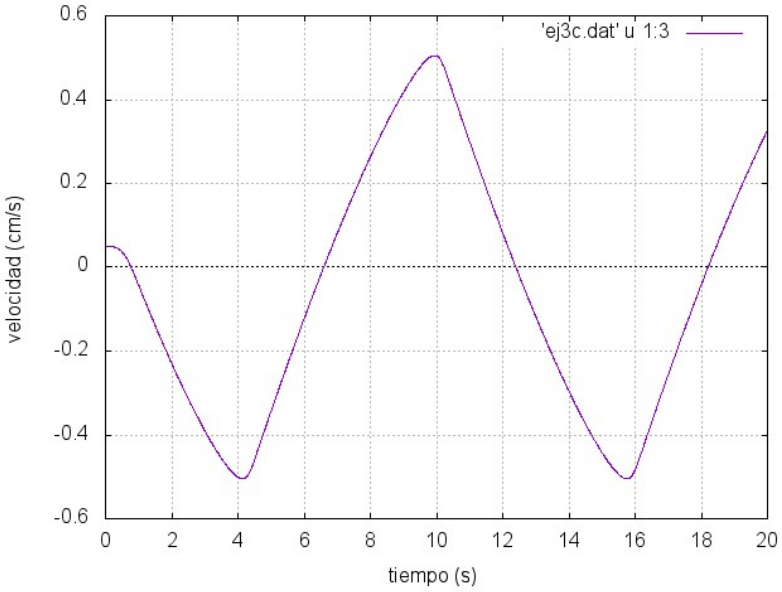


Figura 20: Velocidad en función del tiempo para $\mu_1 = 5$ y condiciones iniciales fijas.

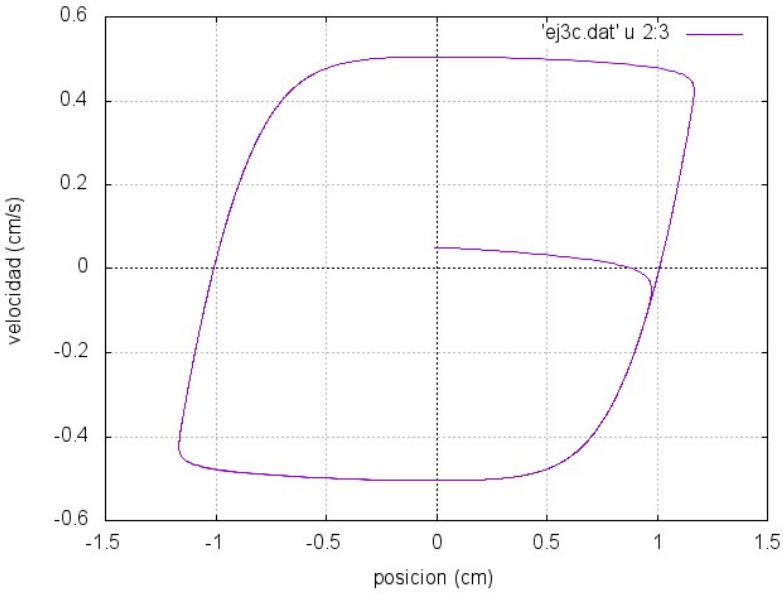


Figura 21: Espacio de fases para $\mu_1 = 5$ y condiciones iniciales fijas.

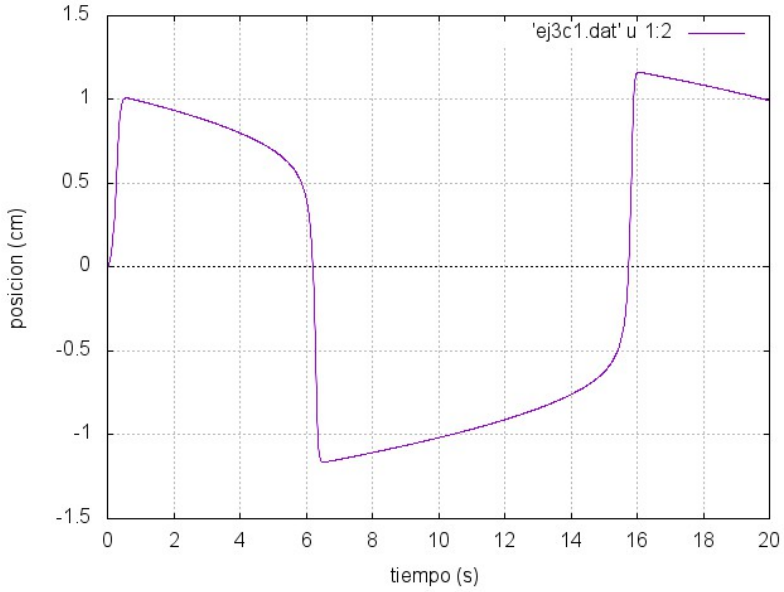


Figura 22: Posición en función del tiempo para $\mu_2 = 10$ y condiciones iniciales fijas.

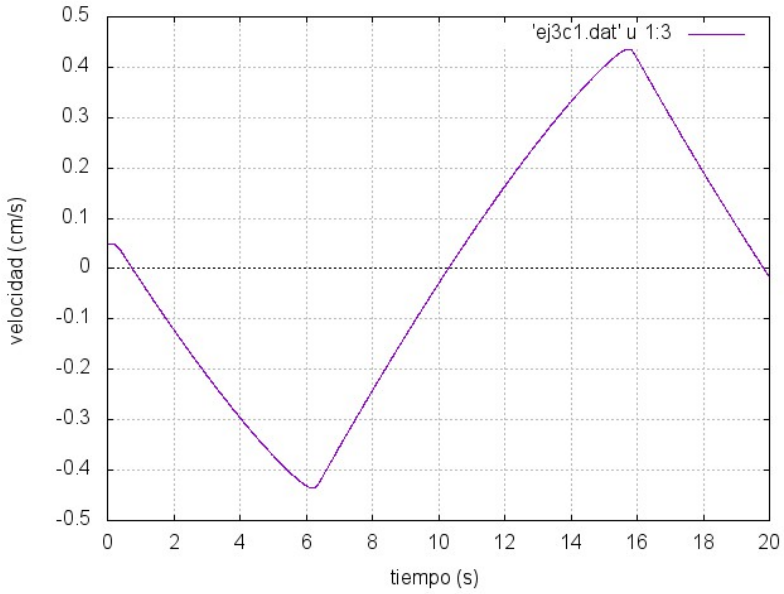


Figura 23: Velocidad en función del tiempo para $\mu_2 = 10$ y condiciones iniciales fijas.

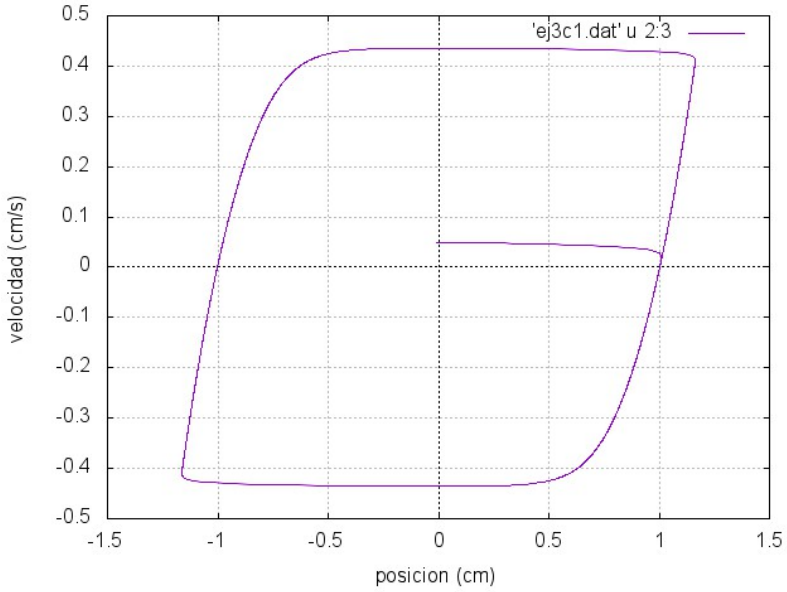


Figura 24: Espacio de fases para $\mu_2 = 10$ y condiciones iniciales fijas.

Finalmente, las condiciones iniciales también se variaron dentro de un bucle *for*, con los mismos valores que muestra el Cuadro 2. Los gráficos para este caso son los siguientes:

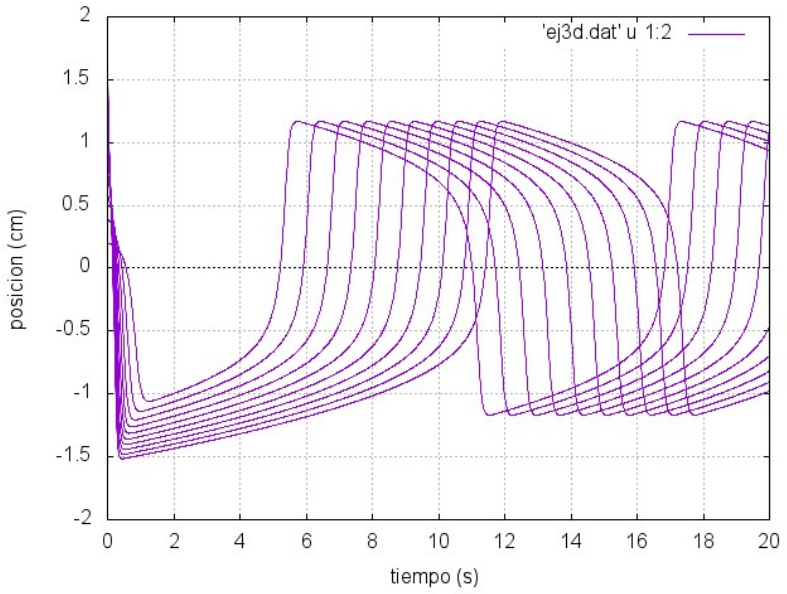


Figura 25: Posición en función del tiempo para $\mu_1 = 5$ y condiciones iniciales variables.

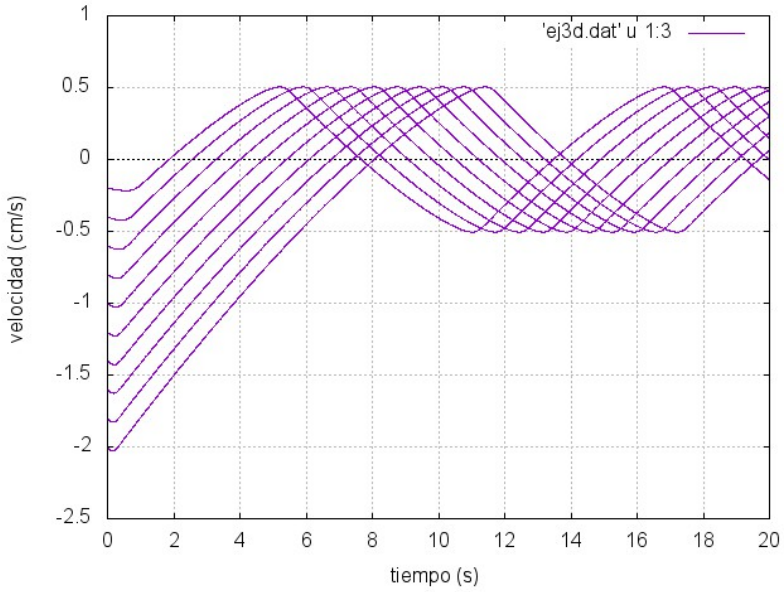


Figura 26: Velocidad en función del tiempo para $\mu_1 = 5$ y condiciones iniciales variables.

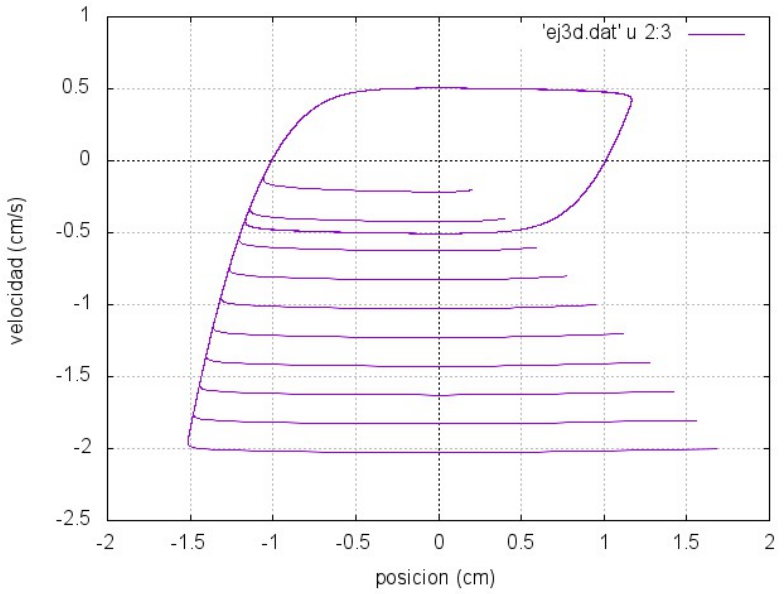


Figura 27: Espacio de fases para $\mu_1 = 5$ y condiciones iniciales variables.

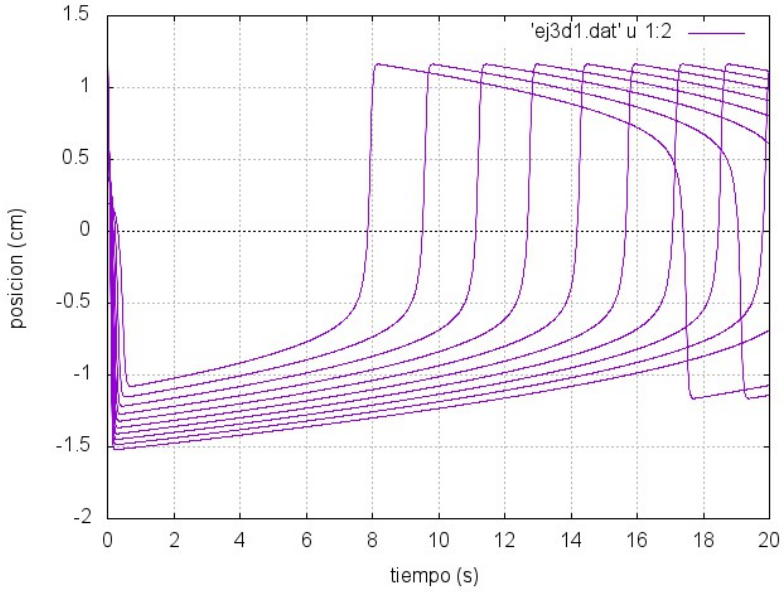


Figura 28: Posición en función del tiempo para $\mu_2 = 10$ y condiciones iniciales variables.

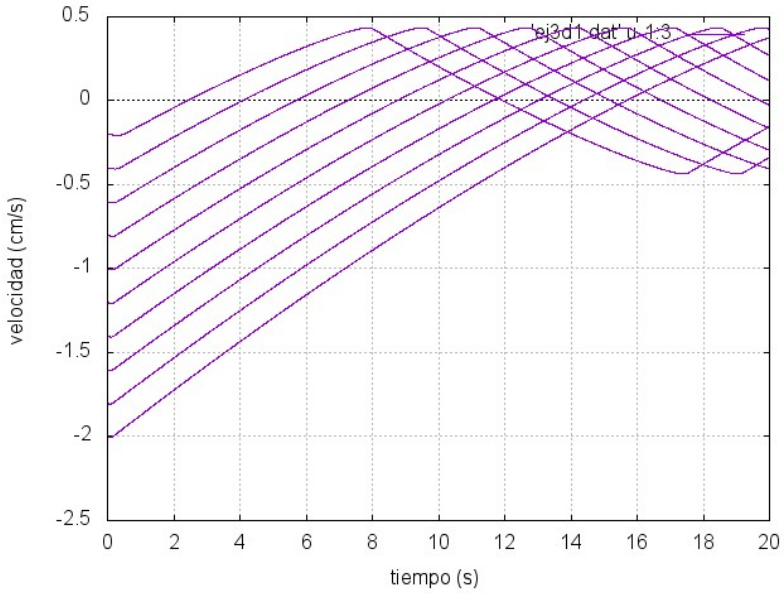


Figura 29: Velocidad en función del tiempo para $\mu_2 = 10$ y condiciones iniciales variables.

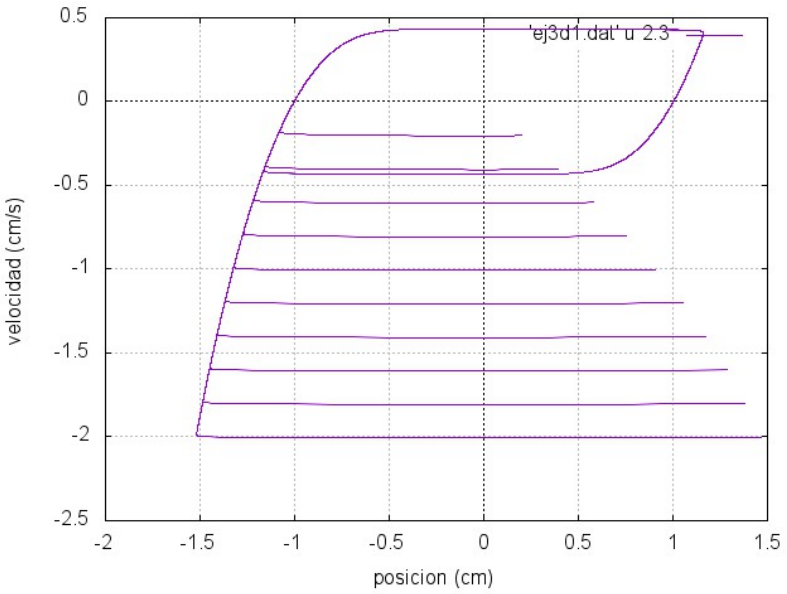


Figura 30: Espacio de fases para $\mu_2 = 10$ y condiciones iniciales variables.

Para $\mu >> 1$, y de acuerdo a los gráficos correspondientes, es más claro ver qué pasa cuando las condiciones iniciales están fijas.

También aquí se presenta el punto de equilibrio inestable (0,0), por ejemplo en el caso $\mu = \mu_1 = 5$ (Figura 21); en el cual en un instante inicial, el sistema se desplaza rápidamente desde este punto hacia $x \approx 1$; para luego realizar un desplazamiento mucho más lento en el sentido vertical desde $(x,y) \approx (1,0)$ hasta $(x,y) \approx (1,-0.5)$.

Llegado este punto, se repite nuevamente el movimiento rápido en sentido horizontal hasta $x \approx$ -1, y luego el lento hacia arriba, desde $(x,y) \approx$ (-1, -0.5) hasta $(x,y) \approx$ (-1, 0.5); repitiéndose el ciclo pero ahora desde el punto $(x,y) \approx$ (-1, 0.5), y no desde el (0,0).

Para el caso $\mu = \mu_2 = 10$, el movimiento es idéntico al detallado arriba, salvo los puntos $(x,y)$ que cambian debido al parámetro $\mu$.

Para finalizar, el siguiente gráfico da una idea de la rapidez, dirección y sentido de los movimientos descriptos:

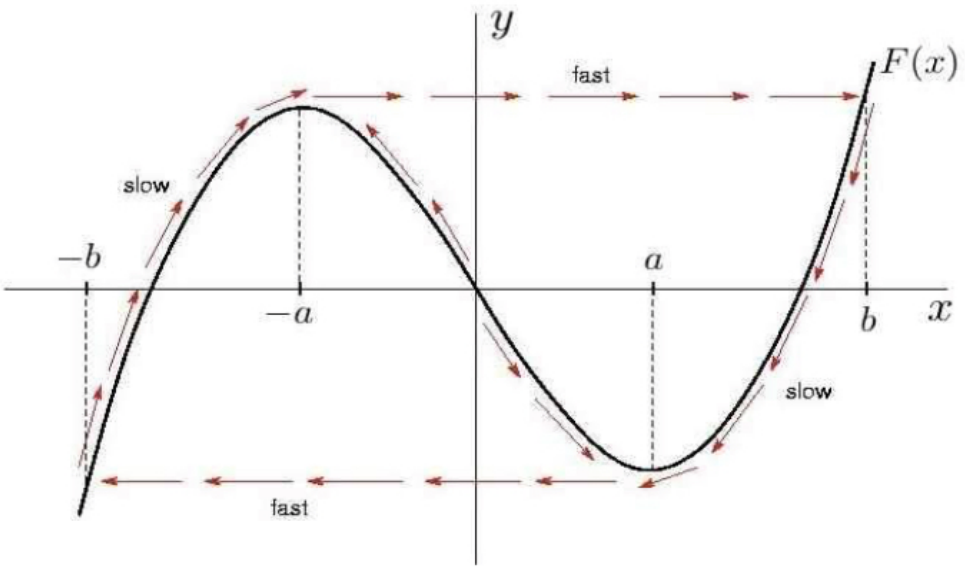


Figura 31: Movimientos rápidos y lentos del oscilador de Van der Pol. Para los casos estudiados en este trabajo, $F(x) = x^3 - x$.

# 3. Apéndice

Descargar el archivo *guia1_poggi.zip* y descomprimirlo en una carpeta a elección del usuario, abrir una terminal dentro de la carpeta $\sim \backslash mecanica - clasica \backslash practica1$; y seguir las instrucciones provistas en la sección Enunciado.

Los archivos que contienen los datos numéricos (*ej2.dat* y *ej3X.dat*) no fueron transcriptos en este informe dada la extensión de los mismos.

## 3.1. Código fuente en C del oscilador armónico amortiguado (ODE_ej2.c)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define a -1
5
6  // Define parametros para usar en el sistema en todo el codigo
7  struct Par{
8      double gamma, omega;
9  } aa;
10
11
12 // Ecuaciones del sistema
13 void ecuaciones(int n, double v[], double dv[], double t){
14     double x,y;
```

```
15      x=v[0];
16      y=v[1];
17
18      // En este caso son 2 ecuaciones acopladas
19      dv[0]= y;
20      dv[1]= -2*aa.gamma*y-aa.omega*aa.omega*x;
21
22    return;
23  }
24
25
26  // Programa principal
27  int main(){
28
29      int i,j;
30      FILE *ptr;
31      double v[2],t,dt,t_pre,t_max;
32
33      // Archivo de salida
34      ptr=fopen("ej2.dat","w");
35
36      dt=0.01;
37      t_max=2;
38
39      // Condiciones iniciales. Se analizaron 3 casos: sub, sobre y amortiguamiento
            critico variando gamma y omega con valores opuestos en el rango de 1 a 10,
            con pasos de 5
40      for (j=1;j<=10;j=j+4) {
41          v[0] = 0.5*j;
42          v[1] = 0.5*(10-j);
43          aa.gamma=(10-j);
44          aa.omega=j;
45
46          t=0.;
47
48          while(t<t_max) {
49              // Integra las ecuaciones utilizando el metodo de Runge Kutta
50              rk4(ecuaciones,v,2,t,dt);
51
52              // Imprime la integracion
53              fprintf(ptr,"%lg\t%lg\t%lg\n",t,v[0],v[1]);
54
55              t+=dt;
56          }
57
58          fprintf(ptr,"\n");
59      }
60
61      fclose(ptr);
62      return(0);
63  }
```

## 3.2. Código fuente en C del oscilador de Van der Pol con $\mu = 0.05$ y condiciones iniciales fijas (ODE_ej3.c)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define a -1
5
6  // Define parametros para usar en el sistema en todo el codigo
7  struct Par{
8      double mu;
9  } aa;
10
11
12 // Ecuaciones del sistema
13 void ecuaciones(int n, double v[], double dv[], double t){
14      double x,y;
15      x=v[0];
16      y=v[1];
17
18      // En este caso son 2 ecuaciones acopladas
19      dv[0]= aa.mu*(y-(x*x*x-x));
20      dv[1]= (-1/aa.mu)*x;
21
22    return;
23 }
24
25
26 // Programa principal
27 int main(){
28
29      int i,j;
30      FILE *ptr;
31      double v[2],t,dt,t_pre,t_max;
32
33      // Archivo de salida
34      ptr=fopen("ej3.dat","w");
35
36      dt=0.01;
37      t_max=20;
38
39      // Condiciones iniciales fijas y mu=0.05
40      v[0] = -0.01;
41      v[1] = 0.05;
42      aa.mu= 0.05;
43
44      t=0.;
45
46      while(t<t_max) {
47          // Integra las ecuaciones utilizando el metodo de Runge Kutta
48          rk4(ecuaciones,v,2,t,dt);
49
50          // Imprime la integracion
```

```
51          fprintf(ptr,"%lg\t%lg\t%lg\n",t,v[0],v[1]);
52
53          t+=dt;
54      }
55
56      fprintf(ptr,"\n");
57
58      fclose(ptr);
59      return(0);
60  }
```

### 3.3. Código fuente en C del oscilador de Van der Pol con $\mu = 0.005$ y condiciones iniciales fijas (ODE_ej31.c)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define a -1
5
6  // Define parametros para usar en el sistema en todo el codigo
7  struct Par{
8      double  mu;
9  } aa;
10
11
12 // Ecuaciones del sistema
13 void ecuaciones(int n, double v[], double dv[], double t){
14      double x,y;
15      x=v[0];
16      y=v[1];
17
18      // En este caso son 2 ecuaciones acopladas
19      dv[0]= aa.mu*(y-(x*x*x-x));
20      dv[1]= (-1/aa.mu)*x;
21
22    return;
23 }
24
25
26 // Programa principal
27 int main(){
28
29      int i,j;
30      FILE *ptr;
31      double v[2],t,dt,t_pre,t_max;
32
33      // Archivo de salida
34      ptr=fopen("ej31.dat","w");
35
36      dt=0.01;
37      t_max=20;
38
39      // Condiciones iniciales fijas y mu=0.005
```

```
40      v[0] = -0.01;
41      v[1] = 0.05;
42      aa.mu= 0.005;
43
44      t=0.;
45
46      while(t<t_max) {
47          // Integra las ecuaciones utilizando el metodo de Runge Kutta
48          rk4(ecuaciones,v,2,t,dt);
49
50          // Imprime la integracion
51          fprintf(ptr,"%lg\t%lg\t%lg\n",t,v[0],v[1]);
52
53          t+=dt;
54      }
55
56      fprintf(ptr,"\n");
57
58      fclose(ptr);
59      return(0);
60  }
```

## 3.4. Código fuente en C del oscilador de Van der Pol con $\mu = 0.05$ y condiciones iniciales variables (ODE_ej3b.c)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define a -1
5
6  // Define parametros para usar en el sistema en todo el codigo
7  struct Par{
8      double mu;
9  } aa;
10
11
12 // Ecuaciones del sistema
13 void ecuaciones(int n, double v[], double dv[], double t){
14     double x,y;
15     x=v[0];
16     y=v[1];
17
18     // En este caso son 2 ecuaciones acopladas
19     dv[0]= aa.mu*(y-(x*x*x-x));
20     dv[1]= (-1/aa.mu)*x;
21
22   return;
23 }
24
25
26 // Programa principal
27 int main(){
28
```

```
29      int i,j;
30      FILE *ptr;
31      double v[2],t,dt,t_pre,t_max;
32
33      // Archivo de salida
34      ptr=fopen("ej3b.dat","w");
35
36      dt=0.01;
37      t_max=20;
38
39      aa.mu= 0.05;
40
41      // Condiciones iniciales variables.
42
43      for (j=1;j<=10;j=j+1) {
44
45
46          v[0] = 0.2*j;
47          v[1] = -0.2*j;
48
49          t=0.;
50
51          while(t<t_max) {
52              // Integra las ecuaciones utilizando el metodo de Runge Kutta
53              rk4(ecuaciones,v,2,t,dt);
54
55              // Imprime la integracion
56              fprintf(ptr,"%lg\t%lg\t%lg\n",t,v[0],v[1]);
57
58              t+=dt;
59          }
60
61          fprintf(ptr,"\n");
62      }
63
64      fclose(ptr);
65      return(0);
66  }
```

## 3.5. Código fuente en C del oscilador de Van der Pol con $\mu = 0.005$ y condiciones iniciales variables (ODE_ej3b1.c)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define a -1
5
6  // Define parametros para usar en el sistema en todo el codigo
7  struct Par{
8      double  mu;
9  } aa;
10
11
```

```
12 // Ecuaciones del sistema
13 void ecuaciones(int n, double v[], double dv[], double t){
14     double x,y;
15     x=v[0];
16     y=v[1];
17
18     // En este caso son 2 ecuaciones acopladas
19     dv[0]= aa.mu*(y-(x*x*x-x));
20     dv[1]= (-1/aa.mu)*x;
21
22   return;
23 }
24
25
26 // Programa principal
27 int main(){
28
29     int i,j;
30     FILE *ptr;
31     double v[2],t,dt,t_pre,t_max;
32
33     // Archivo de salida
34     ptr=fopen("ej3b1.dat","w");
35
36     dt=0.01;
37     t_max=20;
38
39     aa.mu= 0.005;
40
41     // Condiciones iniciales variables.
42
43     for (j=1;j<=10;j=j+1) {
44
45
46         v[0] = 0.2*j;
47         v[1] = -0.2*j;
48
49         t=0.;
50
51         while(t<t_max) {
52             // Integra las ecuaciones utilizando el metodo de Runge Kutta
53             rk4(ecuaciones,v,2,t,dt);
54
55             // Imprime la integracion
56             fprintf(ptr,"%lg\t%lg\t%lg\n",t,v[0],v[1]);
57
58             t+=dt;
59         }
60
61         fprintf(ptr,"\n");
62     }
63
64     fclose(ptr);
65     return(0);
```

```
66 }
```

## 3.6. Código fuente en C del oscilador de Van der Pol con $\mu = 5$ y condiciones iniciales fijas (ODE_ej3c.c)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define a -1
5
6  // Define parametros para usar en el sistema en todo el codigo
7  struct Par{
8      double  mu;
9  } aa;
10
11
12 // Ecuaciones del sistema
13 void ecuaciones(int n, double v[], double dv[], double t){
14      double x,y;
15      x=v[0];
16      y=v[1];
17
18      // En este caso son 2 ecuaciones acopladas
19      dv[0]= aa.mu*(y-(x*x*x-x));
20      dv[1]= (-1/aa.mu)*x;
21
22    return;
23 }
24
25
26 // Programa principal
27 int main(){
28
29      int i,j;
30      FILE *ptr;
31      double v[2],t,dt,t_pre,t_max;
32
33      // Archivo de salida
34      ptr=fopen("ej3c.dat","w");
35
36      dt=0.01;
37      t_max=20;
38
39      // Condiciones iniciales fijas
40      v[0] = -0.01;
41      v[1] = 0.05;
42
43      aa.mu = 5;
44
45
46      t=0.;
47
48      while(t<t_max) {
```

```
49          // Integra las ecuaciones utilizando el metodo de Runge Kutta
50          rk4(ecuaciones,v,2,t,dt);
51
52          // Imprime la integracion
53          fprintf(ptr,"%lg\t%lg\t%lg\n",t,v[0],v[1]);
54
55          t+=dt;
56      }
57
58      fprintf(ptr,"\n");
59
60      fclose(ptr);
61      return(0);
62  }
```

## 3.7. Código fuente en C del oscilador de Van der Pol con $\mu = 10$ y condiciones iniciales fijas (ODE_ej3c1.c)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define a -1
5
6  // Define parametros para usar en el sistema en todo el codigo
7  struct Par{
8      double mu;
9  } aa;
10
11
12 // Ecuaciones del sistema
13 void ecuaciones(int n, double v[], double dv[], double t){
14      double x,y;
15      x=v[0];
16      y=v[1];
17
18      // En este caso son 2 ecuaciones acopladas
19      dv[0]= aa.mu*(y-(x*x*x-x));
20      dv[1]= (-1/aa.mu)*x;
21
22    return;
23 }
24
25
26 // Programa principal
27 int main(){
28
29      int i,j;
30      FILE *ptr;
31      double v[2],t,dt,t_pre,t_max;
32
33      // Archivo de salida
34      ptr=fopen("ej3c1.dat","w");
35
```

```
36      dt=0.01;
37      t_max=20;
38
39      // Condiciones iniciales fijas
40      v[0] = -0.01;
41      v[1] = 0.05;
42
43      aa.mu = 10;
44
45
46      t=0.;
47
48      while(t<t_max) {
49          // Integra las ecuaciones utilizando el metodo de Runge Kutta
50          rk4(ecuaciones,v,2,t,dt);
51
52          // Imprime la integracion
53          fprintf(ptr,"%lg\t%lg\t%lg\n",t,v[0],v[1]);
54
55          t+=dt;
56      }
57
58      fprintf(ptr,"\n");
59
60      fclose(ptr);
61      return(0);
62  }
```

## 3.8. Código fuente en C del oscilador de Van der Pol con $\mu = 5$ y condiciones iniciales variables (ODE_ej3d.c)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define a -1
5
6  // Define parametros para usar en el sistema en todo el codigo
7  struct Par{
8      double mu;
9  } aa;
10
11
12 // Ecuaciones del sistema
13 void ecuaciones(int n, double v[], double dv[], double t){
14      double x,y;
15      x=v[0];
16      y=v[1];
17
18      // En este caso son 2 ecuaciones acopladas
19      dv[0]= aa.mu*(y-(x*x*x-x));
20      dv[1]= (-1/aa.mu)*x;
21
22    return;
```

```
23 }
24
25
26 // Programa principal
27 int main(){
28
29     int i,j;
30     FILE *ptr;
31     double v[2],t,dt,t_pre,t_max;
32
33     // Archivo de salida
34     ptr=fopen("ej3d.dat","w");
35
36     dt=0.01;
37     t_max=20;
38
39     aa.mu= 5;
40     // Condiciones iniciales variables.
41
42     for (j=1;j<=10;j=j+1) {
43
44
45         v[0] = 0.2*j;
46         v[1] = -0.2*j;
47
48         t=0.;
49
50         while(t<t_max) {
51             // Integra las ecuaciones utilizando el metodo de Runge Kutta
52             rk4(ecuaciones,v,2,t,dt);
53
54             // Imprime la integracion
55             fprintf(ptr,"%lg\t%lg\t%lg\n",t,v[0],v[1]);
56
57             t+=dt;
58         }
59
60         fprintf(ptr,"\n");
61     }
62
63     fclose(ptr);
64     return(0);
65 }
```

## 3.9. Código fuente en C del oscilador de Van der Pol con $\mu = 10$ y condiciones iniciales variables (ODE_ej3d1.c)

```
1 #include <stdio.h>
2 #include <math.h>
3
4 #define a -1
5
6 // Define parametros para usar en el sistema en todo el codigo
```

```
7  struct Par{
8      double  mu;
9  } aa;
10
11
12 // Ecuaciones del sistema
13 void ecuaciones(int n, double v[], double dv[], double t){
14     double x,y;
15     x=v[0];
16     y=v[1];
17
18     // En este caso son 2 ecuaciones acopladas
19     dv[0]= aa.mu*(y-(x*x*x-x));
20     dv[1]= (-1/aa.mu)*x;
21
22   return;
23 }
24
25
26 // Programa principal
27 int main(){
28
29     int i,j;
30     FILE *ptr;
31     double v[2],t,dt,t_pre,t_max;
32
33     // Archivo de salida
34     ptr=fopen("ej3d1.dat","w");
35
36     dt=0.01;
37     t_max=20;
38
39     aa.mu= 10;
40     // Condiciones iniciales variables.
41
42     for (j=1;j<=10;j=j+1) {
43
44
45         v[0] = 0.2*j;
46         v[1] = -0.2*j;
47
48         t=0.;
49
50         while(t<t_max) {
51             // Integra las ecuaciones utilizando el metodo de Runge Kutta
52             rk4(ecuaciones,v,2,t,dt);
53
54             // Imprime la integracion
55             fprintf(ptr,"%lg\t%lg\t%lg\n",t,v[0],v[1]);
56
57             t+=dt;
58         }
59
60         fprintf(ptr,"\n");
```

```
61      }
62
63      fclose(ptr);
64      return(0);
65  }
```

## 3.10. Código fuente en C del método de Runge-Kutta de orden 4 provisto por la cátedra (rk4.c)

```
1  /* Runge Kutta integrator from numerical recipies plus improvements */
2  /* void *deri(int n,double h[],double D[],double t);  */
3  /* function argument not tested yet */
4
5  void rk4(void deri(int , double [], double [], double ), \
6  double h[], int n, double t, double dt)
7  {
8  #define naux 26
9
10 int i;
11 double k1[naux],k2[naux],k3[naux],k4[naux],h0[naux];
12 double dt2, dt6;
13
14 dt2=dt/2.;
15 dt6=dt/6.;
16
17 for (i = 0 ; i<n; i++)
18      h0[i] = h[i];
19
20 deri(n,h0,k1,t);
21 for (i =0 ; i<n ; i++)
22      h0[i]=h[i]+dt2*k1[i];
23
24 deri(n,h0,k2,t+dt2);
25 for (i =0 ; i<n ; i++)
26      h0[i]=h[i]+dt2*k2[i];
27
28 deri(n,h0,k3,t+dt2);
29 for (i =0 ; i<n ; i++)
30      h0[i]=h[i]+dt*k3[i];
31
32 deri(n,h0,k4,t+dt);
33 for (i = 0 ; i<n ; i++)
34      {h0[i]=h[i]+dt*k4[i];};
35
36 for (i =0; i<n ; i++)
37      h[i]=h[i]+dt6*(2.*(k2[i]+k3[i])+k1[i]+k4[i]);
38
39 return;
40 }
```

# 4. Bibliografía

**[I], [II], [III], [IV]** F. S. Crawford. *Ondas – Berkeley Physics Course Vol. 3.* Editorial Reverté S.A., 2da edición, Barcelona (1994). Págs. 72-80

**[V]** J. D. Hoffman. *Numerical Methods for Engineers and Scientists.* Editorial Marcel Dekker Inc., Second Edition, (2001). Pág. 398

**[VI]** S. H. Strogatz. *Nonlinear Dynamics and Chaos with Applications to Physics, Biology, Chemistry and Engineering.* Westview Press, Second Edition, (2015). Pág. 198

**[VII]** S. H. Strogatz. *Nonlinear Dynamics and Chaos with Applications to Physics, Biology, Chemistry and Engineering.* Westview Press, Second Edition, (2015). Pág. 212