Unidad 1

Problemas de valor inicial para ecuaciones diferenciales ordinarias (b)

El péndulo plano. El movimiento de un péndulo plano ideal se puede describir por la ecuación diferencial de segundo orden

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + \frac{g}{L}\operatorname{sen}\theta = 0$$

para $t > 0$, donde L es la longitud del péndulo, g es la aceleración de la gravedad terrestre y θ es el ángulo que hace con la vertical, siendo $-\pi < \theta \leq \pi$ (ver fig. 1). Si además especificamos la posición y velocidad del péndulo cuando comienza el movimiento, digamos:

$$\theta(0) = \theta_0, \quad \theta'(0) = \theta'_0$$

tenemos un *problema de valor inicial* para el movimiento del péndulo. Para poder aplicar un método de integración numérica debemos primero reescribir esta ecuación como un *sistema* de ecuaciones diferenciales de *primer orden*. Para ello definimos la función $w(t) = \theta'(t)$ (cuyo significado físico en nuestro problema es el de velocidad angular), obteniendo así el sistema de dos ecuaciones diferenciales de primer orden:

$$\begin{aligned} \theta' &= w, \\ w' &= -k\operatorname{sen}\theta, \end{aligned}$$

siendo $k = g/L$ una constante positiva para un péndulo de longitud dada. El sistema se completa con las condiciones iniciales:

$$\theta(0) = \theta_0, \quad w(0) = \theta'_0.$$

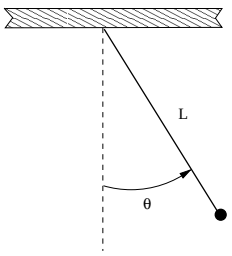

Figura 1. Geometría del problema del péndulo.

Este sistema puede ser escrito en forma vectorial definiendo los vectores bidimensionales:

$$\mathbf{y}(t) = \begin{pmatrix} \theta(t) \\ w(t) \end{pmatrix}, \quad \mathbf{f}(t, \mathbf{y}) = \begin{pmatrix} w \\ -k\operatorname{sen}\theta \end{pmatrix},$$

y

$$\mathbf{y}_0 = \begin{pmatrix} \theta_0 \\ \theta'_0 \end{pmatrix},$$

con lo cual,

$$\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0.$$

Nótese que en nuestro caso la función $\mathbf{f}$ no depende de t, ésto es, la ecuación diferencial es *autónoma*. Pero además, este problema es *conservativo*, es decir, existe una función $E(\theta, w)$ que permanece constante a lo largo de la solución $\theta(t)$, $w(t)$ para cada elección de los valores iniciales (físicamente tal función se identifica con la energía total del sistema por unidad de masa m y el cuadrado de la longitud L del péndulo). Analíticamente ésto resulta de multiplicar ambos miembros de la segunda ecuación del sistema de ecuaciones diferenciales por la primera, obteniendo

$$w'\, w = -k(\operatorname{sen}\theta)\, \theta',$$

de donde

$$\mathrm{d}\left(\frac{w^2}{2}\right) = -k(\operatorname{sen}\theta)\, \mathrm{d}\theta,$$

o sea, integrando,

$$E = \frac{1}{2}w^2 - k\, \cos\theta = \frac{1}{2}w_0^2 - k\, \cos\theta_0.$$

(Físicamente, el primer término de E es la energía cinética y el segundo la energía potencial, ambas medidas por unidad de masa y longitud, midiendo la energía potencial respecto del punto de suspensión $\theta = \pi/2$). Podemos determinar E evaluándola en el instante inicial. Considerando, en particular, la condición inicial en que el péndulo comienza su movimiento desde el reposo, esto es, $w(0) = 0$, tenemos que $E = -k\cos\theta_0$, con lo cual la ecuación anterior permite escribir:

$$w^2 = 2k[\cos\theta - \cos\theta_0]$$

Esta ecuación define la trayectoria seguida por el péndulo en el *plano de las fases* $\theta - w$ para distintos valores de la amplitud inicial, como se muestra en la fig. 2. Claramente para cualquier $0 < \theta_0 < \pi$ el movimiento queda restringido a una oscilación entre $-\theta_0$ y θ_0.

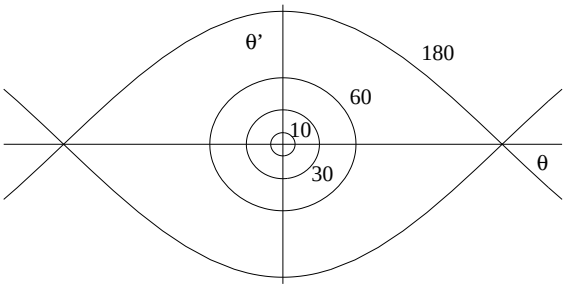

Figura 2. Trayectorias del péndulo en el plano de las fases.

Ejercicio 1. Mostrar que la solución del sistema de ecuaciones diferenciales del péndulo admite infinitos puntos de equilibrio de la forma $\mathbf{y} = (n\pi, 0)^t$, para $n \in \mathbb{Z}$, correspondientes a las situaciones donde el péndulo está en posición vertical con velocidad cero.

Ejercicio 2. Mostrar que para n par la posición de equilibrio es estable, mientras que, para n impar, es inestable. *Ayuda:* linealice el sistema en un entorno de las posiciones de equilibrio.

Ejercicio 3. Determine las soluciones correspondientes a los sistemas linealizados del punto anterior para las condiciones iniciales $\mathbf{y}_0 = (\theta_0, 0)^t$ y $\mathbf{y}_0 = (\pi+\theta_0, 0)^t$, respectivamente, siendo $|\theta| \gg 0$. Analice las correspondientes trayectorias en el plano de las fases.

Consideremos ahora la situación más general en que el péndulo parte con una velocidad angular no nula $w(0) = \theta_0'$. La ecuación de la trayectoria en el plano de las fases es ahora

$$w^2 = 2(E + k\cos\theta).$$

Si $E < k$, entonces el movimiento del sistema sólo podrá tener lugar para θ inferior a una cota $\theta_{\text{máx}}$, definida por la ecuación $\cos\theta_{\text{máx}} = -E/k$. Bajo esta condición, el péndulo oscila entre $-\theta_{\text{máx}}$ y $\theta_{\text{máx}}$, describiendo en el plano de las fases una curva cerrada. Si, en cambio, $E > k$, todos los valores de θ corresponden a un movimiento físico y θ puede crecer sin límite sobre una curva abierta del plano de las fases con infinitos pasajes períodicos a través de los puntos de equilibrio $\theta = 0$ y $\theta = \pi$, pero con velocidad no nula. Lo que sucede físicamente en este caso es que el péndulo tiene tanta energía que puede sobrepasar la posición vertical $\theta = \pi$ y, en consecuencia, sigue girando. En el caso límite en que $E = k$ el péndulo llega a la posición vertical $\theta = 0$ con energía cinética nula, es decir, $w = 0$, aunque este punto de equilibrio inestable es alcanzado sólo asintóticamente para $t \to \infty$.

Ejercicio 4. Mostrar que para $\theta(0) = 0$ el valor límite de la velocidad angular que separa el régimen de oscilación del rotacional está dado por $w_L = 2\sqrt{k}$.

Consideremos ahora la solución *numérica* del sistema que define la ecuación del péndulo. Dicha solución es construida determinando para ciertos valores discretos t_i de la variable independiente t, valores aproximados $\boldsymbol{\eta}_i$ de la solución exacta $\mathbf{y}(t_i)$. Tomaremos aquí los valores t_i *equidistantes*, esto es

$$t_i = i\,\Delta t,$$

donde Δt es el *paso de integración*. Si t_f denota el último valor discreto de t, podemos asegurar la uniformidad del espaciamiento imponiendo que el cociente $t_f/\Delta t$ sea igual a un entero N. De esta forma la solución numérica es obtenida en $(N + 1)$ puntos igualmente espaciados. En particular, el *método de Euler* determina la solución numérica por las relaciones

$$\begin{aligned} \boldsymbol{\eta}_0 &= \mathbf{y}_0 \\ \boldsymbol{\eta}_{i+1} &= \boldsymbol{\eta}_i + \Delta t\,\mathbf{f}(t_i, \boldsymbol{\eta}_i), \end{aligned}$$

para $i = 0, 1, \ldots, N-1$. Usando la expresión de $\mathbf{f}$ para nuestro problema en particular, tenemos explícitamente que el método de Euler procede como

$$\begin{pmatrix} \theta_{i+1} \\ w_{i+1} \end{pmatrix} = \begin{pmatrix} \theta_i \\ w_i \end{pmatrix} + \Delta t \begin{pmatrix} w_i \\ -k\,\text{sen}\,\theta_i \end{pmatrix}.$$

Al final de estas notas se lista un programa que implementa la resolución numérica del problema del péndulo con el método de Euler. Dicho programa consta de tres bloques: el programa principal que contiene las características del problema planteado, una subrutina de propósito general que implementa el método en cuestión y una subrutina que permite evaluar $\mathbf{f}$ para nuestro problema. La entrada y salida de datos se realiza por archivos.

Ejercicio 5. Determine la solución numérica, con el método de Euler, del problema del péndulo para $k = 1$ con la condición inicial $\theta_0 = \pi/10$, $w_0 = 0$, en el intervalo temporal $0 \le t \le 30$. Grafice θ, w y E en función de t, así como también la trayectoria en el plano de fases. Utilice

los pasos de integración $\Delta t = 0.1$ y $\Delta t = 0.01$. Comente sus resultados.

Ejercicio 6. Determine las soluciones numéricas para el mismo problema del ejercicio anterior pero considerando ahora las condiciones iniciales $(\pi/10, 1)$, $(\pi/10, 2)$ y $(\pi/10, -2)$. Comente sus resultados.

Ejercicio 7. Modifique el programa para implementar el *método de Heun*, el cual procede según el esquema:

$$\begin{aligned} \boldsymbol{\eta}_0 &= \mathbf{y}_0 \\ \mathbf{k}_1 &= \Delta t\, \mathbf{f}(t_i, \boldsymbol{\eta}_i), \\ \mathbf{k}_2 &= \Delta t\, \mathbf{f}(t_i + \Delta t, \boldsymbol{\eta}_i + \mathbf{k}_1), \\ \boldsymbol{\eta}_{i+1} &= \boldsymbol{\eta}_i + \frac{1}{2}(\mathbf{k}_1 + \mathbf{k}_2), \end{aligned}$$

para $i = 0, \ldots, N-1$.

Ejercicio 8. Determinar y analizar la solución numérica del problema del pendulo con el método de Heun para las mismas condiciones iniciales que en los ejercicios que implementan el método de Euler. Comente sus resultados.

Métodos numéricos de integración de mayor orden y que permiten una estimación eficiente del error y, en consecuencia, adaptan el tamaño del paso según sea conveniente, son los denominados *métodos de Runge-Kutta-Fehlberg*. En vez de realizar nuestra propia implementación, utilizarmos las subrutina `dderkf` de la biblioteca de rutinas SLATEC[1]. Esta biblioteca de rutinas, o *libreria* en la jerga, proporciona un conjunto de más de 1400 rutinas matemáticas y estadísticas escritas en Fortran 77. En particular, la subrutina `dderkf` implementa en doble precisión el método de Runge-Kutta-Fehlberg de orden 4-5. La mejor documentación sobre su uso está contenida al inicio del propio código, el cual puede verse *on-line* en `http://www.netlib.org/slatec/src/dderkf.f`. Si la libreria SLATEC está disponible en su sistema, entonces la compilación de un programa que use dicha librería procede como sigue:

```
$ gfortran -Wall -o exe codigo.f -lslatec
```

Si la libreria no se encuentra disponible en su sistema es posible descargar el código de `dderkf` y sus dependencias desde el sitio Web de SLATEC y crear una libreria propia. En este caso es de utilidad el `Makefile` generico presentado al final de estas notas. Con el código y sus dependencias alojados en un directorio simplemente copie dicho `Makefile` en el mismo directorio y ejecute

```
$ make
```

El resultado final es la libreria `libderkf.a`. Copiando esta libreria en el mismo lugar que su código, puede compilar el mismo como sigue:

```
$ gfortran -Wall -o exe codigo.f -L. -lderkf
```

Ejercicio 9. Implemente la resolución numérica del problema del péndulo con la subrutina `dderkf`.

Ejercicio 10. Determine la solución numérica con el código anterior del problema del péndulo para las condiciones iniciales planteadas en ejercicios anteriores. Comente sus resultados.

Ejercicio 11. Investigue otras implementaciones de métodos numéricos de integración de ecuaciones diferenciales ordinarias proporcionados por SLATEC, tales como las rutinas `ddeabm`, `ddebdf`, `ddriv1`, etc.

[1] Sitio Web: `http://www.netlib.org/slatec/`.

Programa de integración de las ecuaciones de Euler por el método de Euler.

```
*       ================================================================
        program pendulo
*       ----------------------------------------------------------------
*       Bloque de identificación del programa
*       ----------------------------------------------------------------
*       Resolución numérica de la ecuación de movimiento
*       del péndulo plano por el método de Euler.
*       ----------------------------------------------------------------
*       Formato del archivo de entrada (ej.):
*       -------
*       Archivo de salida
*       pendulo.sal
*       Condicion inicial (ángulos en radianes)
*       10.0 0.0
*       Constante del pendulo
*       1.0
*       Tiempos: inicial, final, paso de integracion
*       0.0 800.0 0.1
*       -------
*       Archivo de salida: especificado en el archivo de entrada
*       Formato:
*       -------
*       t, theta, theta', energia
*       ----------------------------------------------------------------
*       Bloque de declaración de variables
*       ----------------------------------------------------------------
        implicit none
        integer n
        parameter (n=2) ! Grado de la ecuación diferencial, dimensión
                        ! del vector solución
*       ----------------------------------------------------------------
        external euler,         ! Subrutina de integración
     &           derivs_pendulo ! Subrutina de las derivadas
*       ----------------------------------------------------------------
        character(80)infodato,     ! Nombre de descripción
     &               nombreentrada,! Archivo de datos de entrada
     &               nombresalida  ! Archivo de datos de salida
        character(*) fmt           ! Formato de impresion
        parameter (fmt = '(4(1pe13.6,1x))')
*       ----------------------------------------------------------------
        double precision
     &         tetaini,   ! Valor inicial del ángulo
     &         omegaini,  ! Valor inicial de la velocidad angular
     &         k,         ! Constante del péndulo
     &         deltat,    ! Paso de integración en el tiempo
     &         tinicial,  ! Tiempo inicial
     &         tfinal     ! Tiempo final
*       ----------------------------------------------------------------
        common/datos/k    ! Datos compartidos con subrutinas
*       ----------------------------------------------------------------
        double precision
     &         t,         ! Variable de independiente, el tiempo
     &         ener,      ! Energía del sistema
     &         y(n)       ! Vector solución:
                          ! y(1) = ángulo, y(2) = velocidad angular
        integer i
        integer ok
```

```
*       ----------------------------------------------------------------
*       Bloque de lectura de datos
*       ----------------------------------------------------------------
        if (command_argument_count().eq.0) then
           write(0,*) "Error: Indicar el archivo de datos"
           stop
        endif
        call get_command_argument(1,nombreentrada)
*       ----------------------------------------------------------------
        open(8,file=nombreentrada,status='old',iostat=ok)
        if (ok.ne.0) then
           write(0,*) 'El archivo de datos no puede ser leído'
           stop
        endif
*       ----------------------------------------------------------------
        read(8,'(A)')  infodato
        write(6,'(A)') infodato
        read(8,'(A)')  nombresalida
        write(6,'(A)') nombresalida
*       ----------------------------------------------------------------
        read(8,'(A)')  infodato
        write(6,'(A)') infodato
        read(8,*)  tetaini, omegaini
        write(6,*) tetaini, omegaini
*       ----------------------------------------------------------------
        read(8,'(A)')  infodato
        write(6,'(A)') infodato
        read(8,*)  k
        write(6,*) k
*       ----------------------------------------------------------------
        read(8,'(A)')  infodato
        write(6,'(A)') infodato
        read(8,*)  tinicial, tfinal, deltat
        write(6,*) tinicial, tfinal, deltat
*       ----------------------------------------------------------------
        close(8)
*       ----------------------------------------------------------------
*       Bloque de procesamiento
*       ----------------------------------------------------------------
        open(8,file=nombresalida,status='unknown',iostat=ok)
        if (ok.ne.0) then
           write(0,*) 'El archivo de salida no puede ser escrito'
           stop
        endif
*       ----------------------------------------------------------------
        t=tinicial    ! Inicialización de la variable independiente
        y(1)=tetaini  ! Asignación de las condiciones iniciales
        y(2)=omegaini ! Idem
*       ----------------------------------------------------------------
*       Considerar el caso particular de los valores iniciales
        ener=0.5d0*y(2)**2-k*cos(y(1))
        write(8,fmt) t,y(1),y(2),ener
*       ----------------------------------------------------------------
*       Integrar mientras el tiempo sea menor que el final
        i=0
        do while (t.lt.tfinal)
*         Llamar al método de integración
          t=tinicial+dble(i)*deltat
```

```
          call euler(t,deltat,y,n,derivs_pendulo)
*         Incrementar el tiempo
          i=i+1
          t=tinicial+dble(i)*deltat
*         Calcular la energía
          ener=0.5d0*y(2)**2-k*cos(y(1))
*         Imprimir los resultados
          write(8,fmt) t,y(1),y(2),ener
        end do
*       ---------------------------------------------------------------
        close(8)
        end
*       ===============================================================

*       ===============================================================
        subroutine derivs_pendulo(t,y,yp,n)
*       ---------------------------------------------------------------
*       Subrutina para la evaluación de las derivadas del sistema de
*       ecuaciones diferenciales ordinarias de primer orden del pendulo.
*       ---------------------------------------------------------------
*       Bloque de declaración de variables
*       ---------------------------------------------------------------
        implicit none
        integer n       ! Dimensión del sistema de ecuaciones
                        ! diferenciales ordinarias de primer orden
        double precision
     &          t,      ! Variable independiente
     &          y(n),   ! Vector solución
     &          yp(n)   ! Vector de derivadas
*       ---------------------------------------------------------------
        double precision k ! Constante del péndulo
*       ---------------------------------------------------------------
*       Asignación de variables compartidas con el programa principal
        common/datos/k
*       ---------------------------------------------------------------
*       Bloque de procesamiento
*       ---------------------------------------------------------------
        yp(1)=y(2)
        yp(2)=-k*sin(y(1))
        return
*       ---------------------------------------------------------------
        end
*       ===============================================================

*       ===============================================================
        subroutine euler(t,deltat,y,n,derivs)
*       ---------------------------------------------------------------
*       Subrutina de propósito general para la solución numérica del
*       problema de valor inicial de un sistema de ecuaciones
*       diferenciales ordinarias de primer orden por el método de
*       de un paso de Euler.
*       Se utiliza una subrutina pasada como argumento (derivs)
*       para el cálculo de las derivadas del problema particular.
*       ---------------------------------------------------------------
*       Bloque de declaración de variables
*       ---------------------------------------------------------------
        implicit none
        integer n       ! Dimensión del sistema de ecuaciones dif.
```

```
      double precision
     &          t,      ! Variable independiente
     &          deltat, ! Paso de integración
     &          y(n)    ! Vector solución
      external derivs
*     ----------------------------------------------------------------
      double precision yp(n) ! Arreglo automático para las
                             ! derivadas (válido en F90)
      integer i
*     ----------------------------------------------------------------
*     Bloque de procesamiento
*     ----------------------------------------------------------------
      call derivs(t,y,yp,n)   ! Cálculo de las derivadas
      do i=1,n
         y(i)=y(i)+deltat*yp(i) ! Solución numérica
      end do
      return
*     ----------------------------------------------------------------
      end
*     ================================================================
```

Makefile general para crear una libreria estática

```
# Lines starting with the pound sign are comments
#
# Especify the name of the library

MYLIB    = libderkf.a

# You can modify the below as well, but probably
# won't need to.

FORTRAN   = gfortran
OPTS      = -Wall -O2
ARCH      = ar
ARCHFLAGS = crv
RANLIB    = ranlib
SRCS     := $(wildcard *.f)
LIBOBJS  := $(patsubst %.f,%.o,$(SRCS))

# "all" is the default target. Simply make it point to the library.

all: $(MYLIB)

# Define the components of the library, and how to link and archive
# them together.
# These components are defined as dependencies; that is, they must
# be made up-to-date before the library is linked.

$(MYLIB): $(LIBOBJS)
        $(ARCH) $(ARCHFLAGS) $@ $(LIBOBJS)
        $(RANLIB) $@

# Specify that all .o files depend on .f files, and indicate how
# the .f files are compiled to the .o files.

.f.o:
        $(FORTRAN) $(OPTS) -c -o $@ $<

clean:
        -rm -f $(LIBOBJS) $(MYLIB) *~

explain:
        @echo "The following information represents your library:"
        @echo "Library name : $(MYLIB)"
        @echo "Source files : $(SRCS)"
        @echo "Objects files: $(LIBOBJS)"
```