Reporte de evaluación 2

Carlos David Ureña Pérez

02 de Diciembre 2019

1 Problema 1

Se te proporciona el siguiente código que calcula el área de un triángulo de lados a,b,c, mediante una función externa Area(x,y,z):

```
PROGRAM Triangle
 IMPLICIT NONE
 REAL :: a, b, c, Area
 PRINT *, 'Welcome, please enter the&
          &lengths of the 3 sides.'
 READ *, a, b
 PRINT *, 'Triangle''s area:  '', Area(a,b,c)
END PROGRAM Triangle
FUNCTION Area(x,y,z)
 IMPLICIT NONE
 REAL :: Area             ! function type
 REAL, INTENT( IN ) :: x, y, z
 REAL :: theta, height
 theta = ACOS(((x**2+y**2-z**2)/(2.0*x*y))
 height = x*SIN(theta); Area = 0.5*y*height
END FUNCTION Area
```

Copia ese código a un archivo para compilarlo en Fortran. Es posible que intencionalmente se han eliminado o introducido modificaciones, las cuales se pide corregirlas par que compile tu programa.

Con el programa ya corregido y funcionando, utilizando la misma idea, añade una función adicional Volumen(a,b,c) para calcular el volumen de un Paralelepípedo dado por V=a*b*c

1. Resolución parte 1: Se encontraron muchos errores en el programa de ejemplo ya sea dentro del program como fuera de este. Por lo que se arregló de la siguiente manera:

```
        PROGRAM Triangle
     IMPLICIT NONE
     REAL :: a, b, c
!b= base
!height=altura

     REAL, EXTERNAL:: Area
     PRINT *, 'Welcome, please enter the&
              &lengths of the 3 sides.'
     READ *, a, b, c
     PRINT *, "Triangle''s area: ", Area(a,b,c)
    END PROGRAM Triangle
    FUNCTION Area(a,b,c)
     IMPLICIT NONE
     REAL :: Area            ! function type
     REAL, INTENT( IN ) :: a, b, c
     REAL :: theta, height
     theta = b*COS(((b**2+a**2-c**2)/(2.0*b*a)))
     height = b*SIN(theta); Area = 0.5*a*height
    END FUNCTION Area
```

Con este nuevo codigo el programa compila y da al usuario del programa un aproximado muy cercano de el area del triangulo que se da

2. Resolución parte 2(volumen): Se decidió seguir lo puesto por el profesor, se agregó una function con el nombre de volumen y se le preguntó al usuario si queria sacar el volumen o no.

```
        PROGRAM Triangle
     IMPLICIT NONE
     REAL :: a, b, c
     INTEGER:: s, n
!b= base
!height=altura

     REAL, EXTERNAL:: Area
     REAL, EXTERNAL:: Volumen
     PRINT *, 'Welcome, please enter the&
              &lengths of the 3 sides.'
     READ *, a, b, c
     PRINT *, "Triangle''s area: ", Area(a,b,c)
     PRINT*, "Si tu figura es un paralelepípedo, te gustaría saber su volumen? yes=1 no
     READ*, n
     IF(n==1) then
     PRINT*, Volumen(a,b,c)
```

```
        END IF
        IF(n==2) THEN
        END IF
      END PROGRAM Triangle
      FUNCTION Area(a,b,c)
       IMPLICIT NONE
       REAL :: Area            ! function type
       REAL, INTENT( IN ) :: a, b, c
       REAL :: theta, height
       theta = b*COS(((b**2+a**2-c**2)/(2.0*b*a)))
       height = b*SIN(theta); Area = 0.5*a*height
      END FUNCTION Area
  FUNCTION Volumen(a,b,c)
  REAL, INTENT(IN):: a,b,c
  REAL:: Volumen
  Volumen= a*b*c
  END FUNCTION Volumen
```

2 Problema 2

Se proporciona el siguiente programa que resuelve el movimiento de un objeto sujeto a un resorte, obedeciendo la ley de Hooke.

```
    PROGRAM ONE_D_MOTION
!
! Program for the motion of a particle subject to an external
! force f(x) = -x.   We have divided the total time 2*pi into
! 10000 intervals with an equal time step.   The position and
! velocity of the particle are written out at every 500 steps.
! Copyright (c) Tao Pang 1997.
!
  IMPLICIT NONE
  INTEGER, PARAMETER :: N=10001,IN=500
  INTEGER :: I
  REAL :: PI,DT
  REAL, DIMENSION (N):: T,V,X
!
! Assign constants, initial position, and initial velocity
!
  PI   = 4.0*ATAN(1.0)
  DT   = 2.0*PI/FLOAT(N-1)
  X(1) = 0.0
  T(1) = 0.0
  V(1) = 1.0
!
```

```
! Recursion for position and velocity at later time
!
  DO I = 1, N-1
    T(I+1) = DT*I
    X(I+1) = X(I)+V(I)*DT
    V(I+1) = V(I)-X(I)*DT
  END DO
!
! Write the position and velocity every 500 steps
!
  WRITE (6,"(3F16.8)") (T(I),X(I),V(I),I=1,N,IN)
END PROGRAM ONE_D_MOTION
```

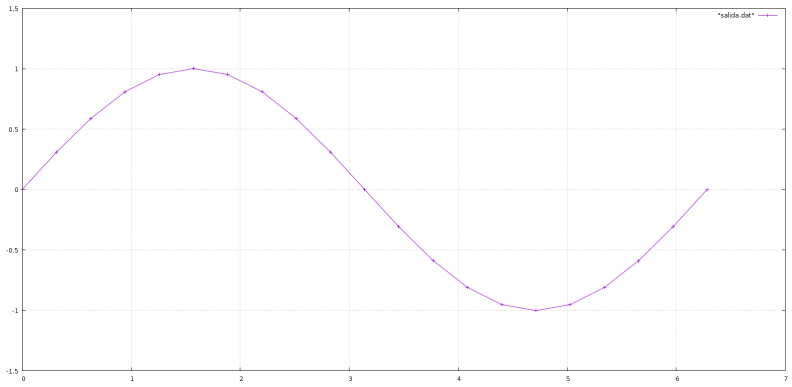
Copia el código a un archivo y compila. Se pide que modifique el código para que escriba la salida a un archivo salida.dat, para posteriormente graficarlo utilizando el programa Gnuplot.
También se pide que modifiques el código para que contemple resortes de constante k, y compares 3 casos (k=0.5, 1.0 y 2.0).

1. Resolución parte 1: El codigo con el open y el archivo de texto queda de la siguiente manera

```
    PROGRAM ONE_D_MOTION
!
! Program for the motion of a particle subject to an external
! force f(x) = -x.   We have divided the total time 2*pi into
! 10000 intervals with an equal time step.   The position and
! velocity of the particle are written out at every 500 steps.
! Copyright (c) Tao Pang 1997.
!
  IMPLICIT NONE
  INTEGER, PARAMETER :: N=10001,IN=500
  INTEGER :: i
  REAL :: pi,DT
  REAL, DIMENSION (N):: T,V,X
! Assign constants, initial position, and initial velocity
!
  pi   = 4.0*ATAN(1.0)
  DT   = 2.0*pi/FLOAT(N-1)
  X(1) = 0.0
  T(1) = 0.0
  V(1) = 1.0
!
! Recursion for position and velocity at later time
 OPEN(Unit=15, file="salida.dat")
  DO i = 1, N-1
    T(i+1) = DT*i
```

```
    X(i+1) = X(i)+V(i)*DT
    V(i+1) = V(i)-X(i)*DT
  END DO
!
! Write the position and velocity every 500 steps
!
  WRITE (15,"(3F16.8)") (T(i),X(i),V(i),i=1,N,IN)
close(15)
END PROGRAM ONE_D_MOTION
```

Y la grafica que arroja gnuplot es la siguiente:



Le modifiqué el nombre a algunas variables para poder entenderle de mejor manera sin confundirme.

2. Analisis del Problema: El análisis pertinente para poder realizar esta parte del programa fue peculiar y difícil de llegar a el, a pesar de que la solución fuera más fácil de lo esperado:
Para empezar tenemos que la ley de Hook establece que $F = -kx$ por lo que podemos decir que $ma = F = -kx$. Entonces volvemos al programa que nos brindó el profesor al incio el cual ya nos da ecuaciones para la posición y velocidad al inicio y al final. Nosotros nos enfocaremos en la de velocidad

```
    V(i+1) = V(i)-X(i)*DT
```

Esta es la ecuación. Con la expresión igualada $ma = F = -kx$ podemos despejar la aceleración y dejarla de la forma $a = F/m$.
Ahora sustituyendo podemos decir que $dv/dt = -kx/m$ o $\frac{dv}{dt} = \frac{-k}{m}x$
Expresamos la $\frac{dv}{dt}$ como $\frac{V_f - V_0}{dt}$ y despejamos V_f

$$V_f = V_0 - \frac{k}{m} * x * dt$$

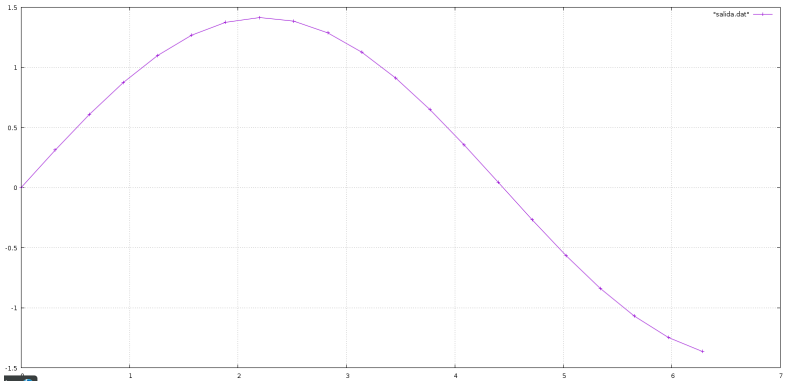
Esta expresión podemos compararla con la que nos brindó el profesor y podemos encontrar que son las mismas, solo que al principio del programa en un comentario se nos especifica que estamos tratando con una partícula por lo cual la masa la podemos expresar como uno y la constante del resorte también como uno.
El código del programa al final quedó de la siguiente manera:

```
    PROGRAM ONE_D_MOTION
!
! Program for the motion of a particle subject to an external
! force f(x) = -x.   We have divided the total time 2*pi into
! 10000 intervals with an equal time step.   The position and
! velocity of the particle are written out at every 500 steps.
! Copyright (c) Tao Pang 1997.
!
  IMPLICIT NONE
  INTEGER, PARAMETER :: N=10001,IN=500
  INTEGER :: i
  REAL :: pi,DT,k
  REAL, DIMENSION (N):: T,V,X
! Assign constants, initial position, and initial velocity
PRINT*,"Dame el valor de la k"
READ*, k
  pi   = 4.0*ATAN(1.0)
  DT   = 2.0*pi/FLOAT(N-1)
  X(1) = 0.0
  T(1) = 0.0
  V(1) = 1.0
!
! Recursion for position and velocity at later time
 OPEN(Unit=15, file="salida.dat")
  DO i = 1, N-1
    T(i+1) = DT*i
    X(i+1) = X(i)+V(i)*DT
    V(i+1) = V(i)-k*X(i)*DT
  END DO
!
! Write the position and velocity every 500 steps
!
  WRITE (15,"(3F16.8)") (T(i),X(i),V(i),i=1,N,IN)
close(15)
END PROGRAM ONE_D_MOTION
```

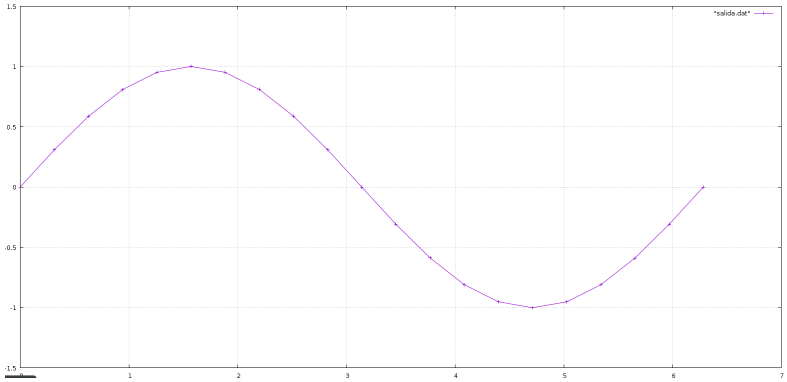
Las graficas con las diferentes k que el problema pide quedan de esta manera:

3. Resolución del problema

$k = 0.5$



$k = 1.0$



$k = 2.0$