División de Ciencias Exactas y Naturales
Departamento de Física
Lic. en Física

Programación y Lenguaje Fortran
Reporte – Evaluación 2

Prof. Carlos Lizárraga Celaya

Camargo Loaiza Julio Andrés

Hermosillo, Sonora Diciembre 2019

EVALUACIÓN 2 - REPORTE

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 para que compile tu programa.

```
PROGRAM Triangle
     IMPLICIT NONE
     REAL :: a, b, c, Area
     PRINT *, 'Welcome, please enter the&
              &lengths of the 3 sides.'
     READ *, a, b !le hace falta leer el lado c
     PRINT *, 'Triangle''s area:  '', Area(a,b,c)' !Sigue abierta la
comilla para el Area
    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)) !Tiene un paréntesis de más
     height = x*SIN(theta); Area = 0.5*y*height
    END FUNCTION Area
```

```
      PROGRAM Triangle
           IMPLICIT NONE
           REAL :: a, b, c, 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(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
```

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

```
PROGRAM Triangle
     IMPLICIT NONE
     REAL :: a, b, c, Area, Vol
     PRINT *, 'Welcome, please enter the&
              & lengths of the 3 sides.'
     READ *, a, b, c
     PRINT *, 'Triangle''s area:  ', Area(a,b,c)
     PRINT *, 'Parallelepiped''s volume:  ', Vol(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

    FUNCTION Vol(x,y,z)
    IMPLICIT NONE
    REAL :: Vol
    REAL, INTENT (IN) :: x,y,z
    Vol = x*y*z
    END FUNCTION Vol
```

Calcula el volumen de un paralelepípedo de lados a=1, b=2, c=3.

```
julioand96@atena:~/Documentos/ProgFortran/Evaluación2$ gfortran Problema1.f90 -o
 Prob1
julioand96@atena:~/Documentos/ProgFortran/Evaluación2$ ./Prob1
 Welcome, please enter the lengths of the 3 sides.
1, 2, 3
 Triangle's area:    -8.74227766E-08
 Parallelepiped's volume:     6.00000000
julioand96@atena:~/Documentos/ProgFortran/Evaluación2$ gfortran Problema1.f90 -o
 Prob1
julioand96@atena:~/Documentos/ProgFortran/Evaluación2$ ./Prob1
 Welcome, please enter the lengths of the 3 sides.
1, 2, 3
 Triangle's area:    -8.74227766E-08
 Parallelepiped's volume:     6.00000000
```

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.

  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.

```
julioand96@atena:~/Documentos/ProgFortran/Evaluación2$ gfortran Hooke.f90 -o Hooke
julioand96@atena:~/Documentos/ProgFortran/Evaluación2$ ./Hooke
      0.00000000      0.00000000      1.00000000
      0.31415927      0.30904728      0.95115054
      0.62831855      0.58790123      0.80917645
      0.94247776      0.80925632      0.58795923
      1.25663710      0.95143223      0.30913907
      1.57079625      1.00049424      0.00000002
      1.88495553      0.95162076     -0.30920008
      2.19911480      0.80957693     -0.58819181
      2.51327419      0.58825105     -0.80965704
      2.82743335      0.30929297     -0.95190275
      3.14159250      0.00000124     -1.00098920
      3.45575190     -0.30935171     -0.95209181
      3.76991105     -0.58848166     -0.80997849
      4.08407021     -0.81005651     -0.58854175
      4.39822960     -0.95237219     -0.30944610
      4.71238899     -1.00148284     -0.00000151
      5.02654839     -0.95256162      0.30950430
      5.34070730     -0.81037760      0.58877224
      5.65486670     -0.58883196      0.81045544
      5.96902609     -0.30959910      0.95284235
      6.28318501     -0.00000198      1.00197709
```

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).

```
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.

  IMPLICIT NONE
  INTEGER, PARAMETER :: N=10001,IN=500
  INTEGER :: I
  REAL :: PI,DT, k, m
  REAL, DIMENSION (N):: T,V,X

PRINT *, 'Enter the k value'
     READ *, k
! Assign constants, initial position, and initial velocity
  m = 1
  PI   = 4.0*ATAN(1.0)
  DT   = 2.0*PI*(sqrt(m/k))/FLOAT(N-1) !k afecta el periodo
  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*k/m    !Constante k
  END DO

! Write the position and velocity every 500 steps
open (6, file = 'salida.dat', status = 'unknown')
  WRITE (6,"(3F16.8)") (T(I),X(I),V(I),I=1,N,IN)
close (6)
END PROGRAM ONE D MOTION
```

Se añade la constante *k* funcionando como la constante elástica del resorte, para no afectar otro parámetro, se añade un valor de masa *m=1*.

Compilación correcta del programa. El programa pide que se especifique el valor de *k*.

```
julioand96@atena:~/Documentos/ProgFortran/Evaluación2$ gfortran Problema2.f90 -o
 Prob2
julioand96@atena:~/Documentos/ProgFortran/Evaluación2$ ./Prob2
 Enter the k value
0.5
```

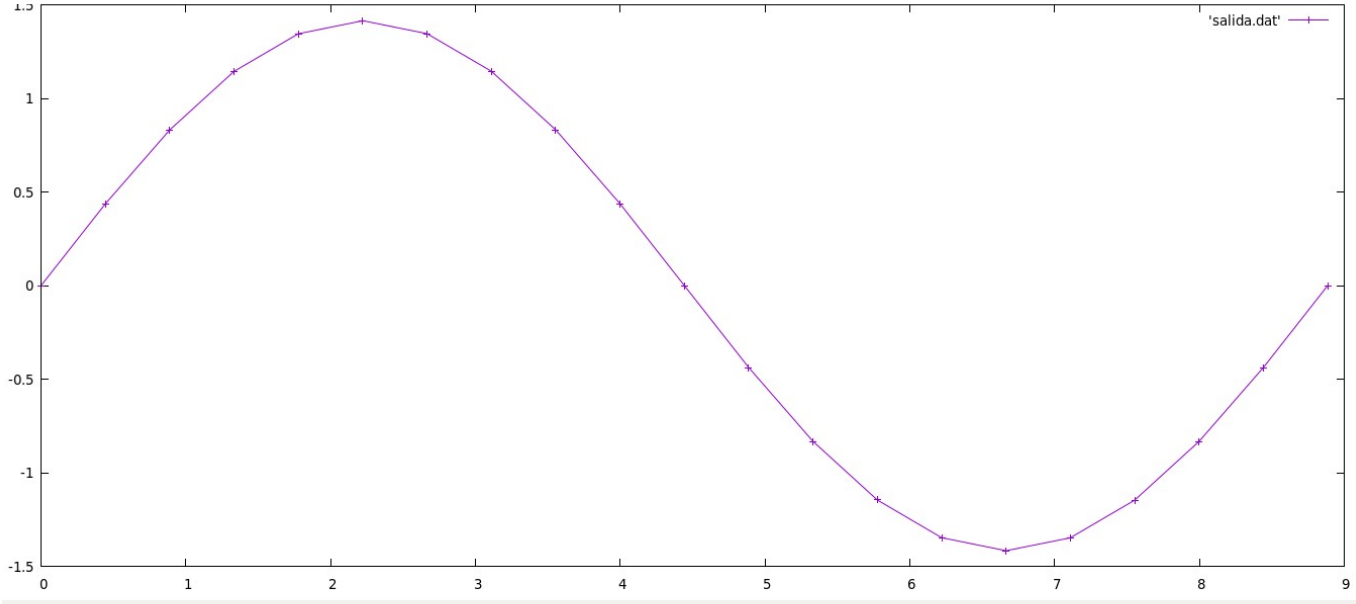
A continuación, se muestran los resultados numérica y gráficamente para los tres casos de valor de *k*.

***K* = 0.5**

- Datos:

0.00000000	0.00000000	1.00000000
0.44428831	0.43705919	0.95115054
0.88857663	0.83141756	0.80917645
1.33286488	1.14446080	0.58795935
1.77715325	1.34552824	0.30913919
2.22144151	1.41491055	0.00000032
2.66572976	1.34579289	-0.30919945
3.11001825	1.14491284	-0.58819079
3.55430651	0.83191049	-0.80965561
3.99859476	0.43740517	-0.95190108
4.44288301	0.00000069	-1.00098741
4.88717127	-0.43749017	-0.95209002
5.33145952	-0.83223909	-0.80997664
5.77574778	-1.14559221	-0.58853996
6.22003651	-1.34685814	-0.30944419
6.66432476	-1.41630900	0.00000018
7.10861301	-1.34712267	0.30950540
7.55290127	-1.14604390	0.58877259
7.99718952	-0.83273184	0.81045526
8.44147778	-0.43783575	0.95284122
8.88576603	0.00000009	1.00197530

- Gráfica:

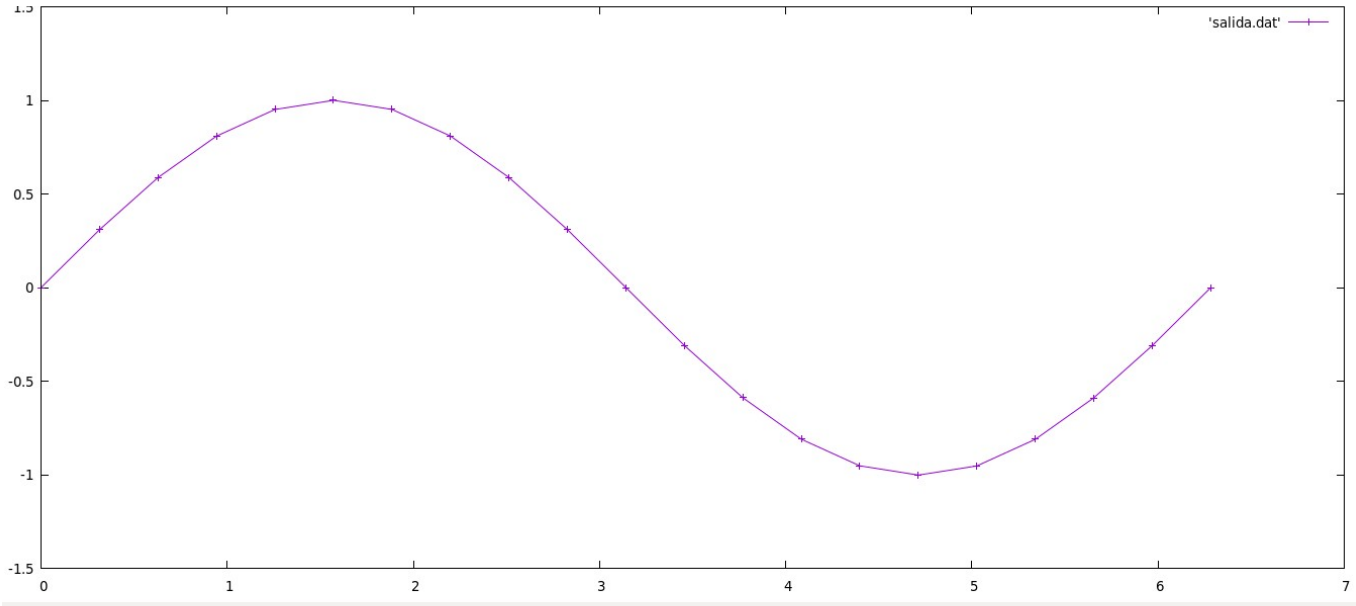


K = 1

- Datos:

0.00000000	0.00000000	1.00000000
0.31415927	0.30904728	0.95115054
0.62831855	0.58790123	0.80917645
0.94247776	0.80925632	0.58795923
1.25663710	0.95143223	0.30913907
1.57079625	1.00049424	0.00000002
1.88495553	0.95162076	-0.30920008
2.19911480	0.80957693	-0.58819181
2.51327419	0.58825105	-0.80965704
2.82743335	0.30929297	-0.95190275
3.14159250	0.00000124	-1.00098920
3.45575190	-0.30935171	-0.95209181
3.76991105	-0.58848166	-0.80997849
4.08407021	-0.81005651	-0.58854175
4.39822960	-0.95237219	-0.30944610
4.71238899	-1.00148284	-0.00000151
5.02654839	-0.95256162	0.30950430
5.34070730	-0.81037760	0.58877224
5.65486670	-0.58883196	0.81045544
5.96902609	-0.30959910	0.95284235
6.28318501	-0.00000198	1.00197709

- Gráfica:

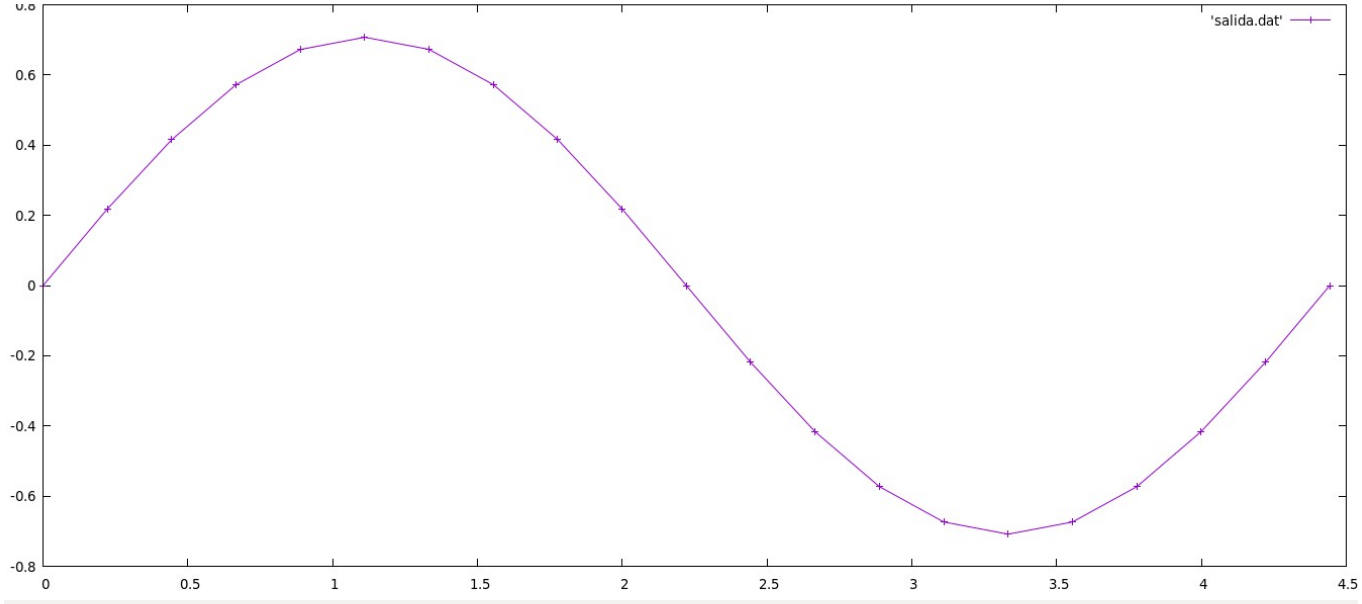


K = 2

- Datos:

0.00000000	0.00000000	1.00000000
0.22214416	0.21852960	0.95115054
0.44428831	0.41570878	0.80917645
0.66643244	0.57223040	0.58795935
0.88857663	0.67276412	0.30913919
1.11072075	0.70745528	0.00000032
1.33286488	0.67289644	-0.30919945
1.55500913	0.57245642	-0.58819079
1.77715325	0.41595525	-0.80965561
1.99929738	0.21870258	-0.95190108
2.22144151	0.00000035	-1.00098741
2.44358563	-0.21874508	-0.95209002
2.66572976	-0.41611955	-0.80997664
2.88787389	-0.57279611	-0.58853996
3.11001825	-0.67342907	-0.30944419
3.33216238	-0.70815450	0.00000018
3.55430651	-0.67356133	0.30950540
3.77645063	-0.57302195	0.58877259
3.99859476	-0.41636592	0.81045526
4.22073889	-0.21891788	0.95284122
4.44288301	0.00000004	1.00197530

- Gráfica:



Discusión

Problema 1

En el primer problema, se hace uso de la ley de cosenos para resolver cualquier caso de triángulos, el caso mostrado solo hacía falta de leer el tercer lado del triángulo, además de algunos errores de sintaxis. Para añadir la función de obtener el volumen dado por un paralelepípedo de lados a, b, c, se añadió una función siguiendo el mismo patrón que en el ejemplo mostrado, pero obteniendo el resultado al multiplicar los tres lados mencionados.

Problema 2

En el segundo problema, se tiene un programa que compila sin problemas, al cual se le deben hacer algunas pequeñas modificaciones para que sea considerada la constante de elasticidad *k*. Las únicas modificaciones que hacen falta es la de incluir esta constante en el periodo que está representado como:

DT = 2.0*PI/FLOAT(N-1), el cual pasa a convertirse en: DT = 2.0*PI*(sqrt(m/k))/FLOAT(N-1)

Además, se debe considerar ahora en las ecuaciones diferenciales de primer orden, el cual pasa de ser:

V(I+1) = V(I)-X(I)*DT , para ser modificado a: V(I+1) = V(I)-X(I)*DT*k/m

Además del cambio realizado en el cálculo de la posición y velocidad, también se añadió la apertura de un archivo a donde se mandan los resultados para poder ser graficados después en gnuplot. Este cambio es sencillo.
Se observa que en todos los casos de *k* diferentes, el movimiento se limita a un periodo. La diferencia radica en que a mayor valor de *k*, el periodo disminuye su valor, es decir, tarda menos tiempo en volver a su posición inicial.

Conclusiones

Los cambios realizados a los ejemplos proporcionados son mínimos y fáciles de realizar si se tienen los conocimientos básicos necesarios para resolver el problema, los cuales se adquirieron en el transcurso del curso. Además de ello, se hizo uso de conocimientos básicos de Ley de Hooke y Ley de Cosenos para resolver el cálculo pedido.
El primer ejemplo se trata de detectar los errores de sintaxis y resolverlos, además de añadir una función básica donde sólo se calcula un producto de variables. Mientras en el segundo ejemplo, es necesario tener conocimientos básicos de la Ley de Hooke para reconocer dónde deben ser introducidos los cambios pedidos, además de conocer la instrucción “open” y el uso de gnuplot, conocimiento que se obtuvo en la actividad 3 del curso.