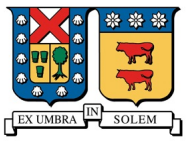

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Tarea N°3: Método de RK4 Modelación de la Amplitud de Grandes Burbujas de Gas Ascendiendo por un Líquido

Computación Científica

IMP-458 - 2022

Departamento Ing. Mecánica UTFSM

Martín Achondo Mercado

Rol: 201860005-9

Profesor: Franco Perazzo M.

21 de Julio de 2022

Resumen

En este trabajo se intentó reproducir los resultados obtenidos por G. K. Bachelor [1] respecto a la estabilidad de grandes burbujas de gas al ascender por un líquido. Para esto, se resolvió la ecuación diferencial adimensionalizada que gobierna este problema por un código de elaboración propia con el método de Runge Kutta de orden 4. Respecto al método, se utilizó un paso de 0.01 y de esta manera, se pudo reproducir los resultados en la publicación de G. K. Bachelor. Para la modelación se variaron 2 parámetros adimensionales del problema (α y β), los cuales tienen relación con el tamaño de la burbuja y con la razón entre la longitud de onda de la perturbación inicial con la crítica respectivamente. De esta manera, se obtuvo que la amplitud es máxima cuando la longitud de la perturbación alcanza casi 4 veces la longitud crítica, $\lambda \approx 3.7\lambda_c$ para los distintos tamaños de burbujas y cuando la longitud de onda de la perturbación inicial es cercana a la crítica, $\lambda_0 = 0.91\lambda_c$. Además, mientras más grande el tamaño de la burbuja, se alcanza una mayor amplitud de la perturbación en el punto detallado anteriormente.

Índice

1. Introducción

En este trabajo se programará el método de Runge Kutta de orden 4 para resolver un sistema de ecuaciones diferenciales ordinarias de primer orden. El propósito es resolver una ecuación diferencial de segundo orden que aparece al estudiar el comportamiento de la amplitud de las fluctuaciones en la superficie de una burbuja al ascender a través de un líquido.

Para validar los resultados, se comparará la solución con la publicación de G. K. Batchelor [1], creando gráficos para la amplitud de la burbuja adimensionalizada para distintos valores de los parámetros característicos del problema.

1.1. Presentación del Problema

En la actualidad se han estudiado distintos tipos de burbujas [1], de donde se ha encontrado que burbujas "grandes" al ascender por flotación en un líquido toman la forma de un segmento esférico, mientras que su superficie inferior queda relativamente plana. Estas burbujas son llamadas Spherical-Cap Bubbles, SCB. La superficie superior de la SCB está sujeta a una inestabilidad de Rayleigh-Taylor, y experimenta fluctuaciones (que pueden llegar a destruir la burbuja). Se puede establecer que la amplitud A de las fluctuaciones de la superficie de la burbuja (respecto a su forma esférica) está determinada por la ecuación:

$$\frac{\mathrm{d}^2 A}{\mathrm{d}t^2} + 3k\frac{\mathrm{d}A}{\mathrm{d}t} + \left(2k^2 - gn\left(1 - \frac{n^2}{n_c^2}\right)\right) A = 0 \tag{1}$$

En donde $n_c = 2\pi/\lambda_c = (\rho g/\sigma)^{1/2}$ es el número de onda crítico en la interfaz de ambos fluidos y λ_c la longitud de onda crítica, $n = 2\pi/\lambda$ el número de onda de la perturbación con longitud de onda λ, $k = (g/R)^{1/2}$ donde R es el radio de curvatura de la burbuja, ρ la densidad del líquido, σ la tensión superficial en la interfaz y g la aceleración de gravedad.

Para la resolución de esta ecuación, se introducirá la siguiente adimensionalización, con λ_0 la longitud de onda de la perturbación inicial con número de onda n_0:

$$\mathcal{A} = \frac{A}{L} \ ; \ \tau = kt \ ; \ \alpha = \frac{gn_c}{k^2} = n_c R \ ; \ \beta = \frac{n_c}{n_0} = \frac{\lambda_0}{\lambda_c} \tag{2}$$

De esta manera, la ecuación 1 queda como:

$$\frac{\mathrm{d}^2 \mathcal{A}}{\mathrm{d}\tau^2} + 3\frac{\mathrm{d}\mathcal{A}}{\mathrm{d}\tau} + \left(2 - \frac{\alpha}{\beta \mathrm{e}^{\tau}}\left(1 - \frac{1}{\beta^2 \mathrm{e}^{2\tau}}\right)\right) \mathcal{A} = 0 \tag{3}$$

Se nota que la evolución de la perturbación en la superficie está determinada por los parámetros α y β. Notar que para un sistema gas-líquido, α es proporcional al tamaño de la burbuja y β es proporcional a la longitud de onda de la perturbación inicial. A esta ecuación se le añadirán las siguientes condiciones iniciales:

$$\mathcal{A}(\tau = 0) = 0 \ ; \ \left.\frac{\mathrm{d}\mathcal{A}}{\mathrm{d}\tau}\right|_{\tau=0} = 1 \tag{4}$$

2. Metodología

Para resolver la ecuación 3 se programará en Fortran el método de Runge Kutta de orden 4 para resolver sistemas de ecuaciones diferenciales de primer orden. Dado esto, la ecuación 3 se escribirá como un sistema de ecuaciones de primer orden bajo el siguiente cambio de variables:

$$\phi(\tau) = \mathcal{A}(t) \ ; \ \psi(\tau) = \frac{\mathrm{d}\mathcal{A}}{\mathrm{d}\tau} \tag{5}$$

De esta manera, la ecuación 3 se transforma en el sistema:

$$\begin{cases} \dfrac{\mathrm{d}\phi}{\mathrm{d}\tau} = \psi(t) \\ \dfrac{\mathrm{d}\psi}{\mathrm{d}\tau} = -3\psi(t) - \left(2 - \dfrac{\alpha}{\beta \mathrm{e}^{\tau}}\left(1 - \dfrac{1}{\beta^2 \mathrm{e}^{2\tau}}\right)\right)\phi(t) \end{cases} \tag{6}$$

Con las condiciones iniciales:

$$\phi(\tau = 0) = 0 \ ; \ \psi(\tau = 0) = 1 \tag{7}$$

2.1. Método de Runge Kutta de orden 4

El método de RK-4 [2] sirve para resolver ecuaciones diferenciales de primer orden. La idea se basa en aproximar la solución usando evaluaciones en la ecuación diferencial para así aproximar con orden 4. Lo dicho entonces lleva a resolver la siguiente ecuación:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y) \tag{8}$$

Para la cual entonces se genera la siguiente iteración, con h el paso en la variable t:

$$y(t + h) = y(t) + \int_t^{t+h} f(s, y)\,\mathrm{d}s \tag{9}$$

Los métodos de Runge Kutta aproximan la integral planteada como:

$$\int_t^{t+h} f(s,y)\,\mathrm{d}s \approx h\sum_{\ell=1}^{m}\gamma_\ell k_\ell \tag{10}$$

En donde γ corresponden a los pesos y k a las evaluaciones convenientes de f.

El método de RK-4 es la aplicación a 4 etapas, dadas por:

$$\begin{aligned} k_1 &= f(t_j, y^j) \\ k_2 &= f(t_j + h/2, y^j + k_1 h/2) \\ k_3 &= f(t_j + h/2, y^j + k_2 h/2) \\ k_4 &= f(t_j + h, y^j + k_3 h) \end{aligned} \tag{11}$$

Así, la iteración en el paso j queda como:

$$y^{j+1} = y^j + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{12}$$

Notar que el método tiene una forma explícita. Además, este método se generaliza fácilmente a sistemas de ecuaciones, siguiendo la misma iteración con la variable y y la función f como vectores. El algoritmo a programar es:

Algoritmo 1 Código a programar para RK-4 para ecuación 8

Require: t_0, t_f, h, y_0, f
  $t \leftarrow t_0$
  $y \leftarrow y_0$
  while $t < t_f$ **do**
    $k_1 \leftarrow f(t, y)$
    $k_2 \leftarrow f(t + h/2, y + k_1 h/2)$
    $k_3 \leftarrow f(t + h/2, y + k_2 h/2)$
    $k_4 \leftarrow f(t + h, y + k_3 h)$
    $y \leftarrow y + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
    $t \leftarrow t + h$
  end while
  return t, y

2.2. Modelación y Presentación de los Resultados

Por lo dicho anteriormente, la ecuación 3 se resolverá utilizando el método de RK-4 aplicado al sistema en la ecuación 6. Para esto, se utilizará un paso temporal $\Delta\tau = h = 0.01$ y se guardarán los resultados cada 3 pasos y se imprimirán en un archivo de salida. Todo el código será programado en Fortran y con variables de doble precisión.

Para la modelación, se presentarán gráficos para la amplitud adimensionalizada $\mathcal{A}$ en función de $\tau' = \tau + \ln(\beta)$ para diferentes valores de estos parámetros. En general, se modelará para $\beta =$ 0.33, 0.5, 0.67, 0.91, 1.0, 1.33 y 2.0 con $\alpha = 50$, 65 y 80. El intervalo a graficar irá entre $\tau' = 0$ hasta $\tau' = 8$

3. Resultados

En esta sección se presentan los resultados para los distintos valores de α y β.

Se muestra la distribución de la amplitud adimensionalizada $\mathcal{A}$ para $\alpha = 50$

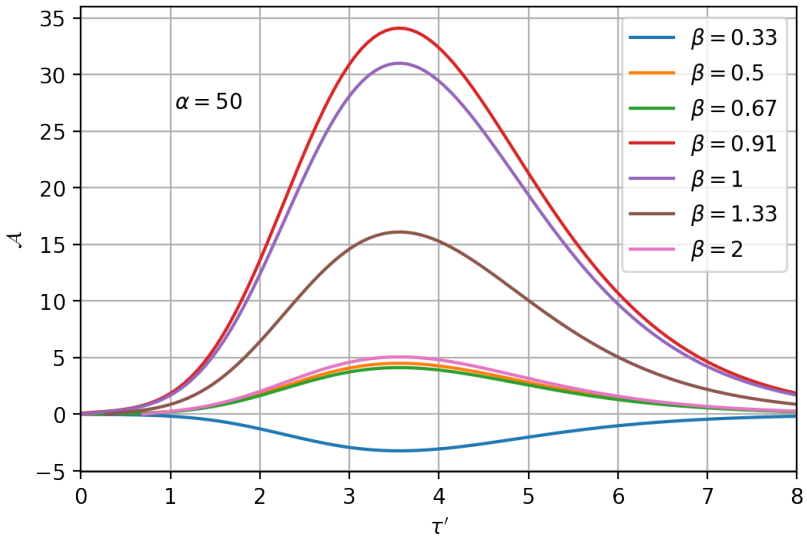

Figura 1: Amplitud adimensionalizada para $\alpha = 50$ y distintos valores de β

Se muestra la distribución de la amplitud adimensionalizada $\mathcal{A}$ para $\alpha = 65$

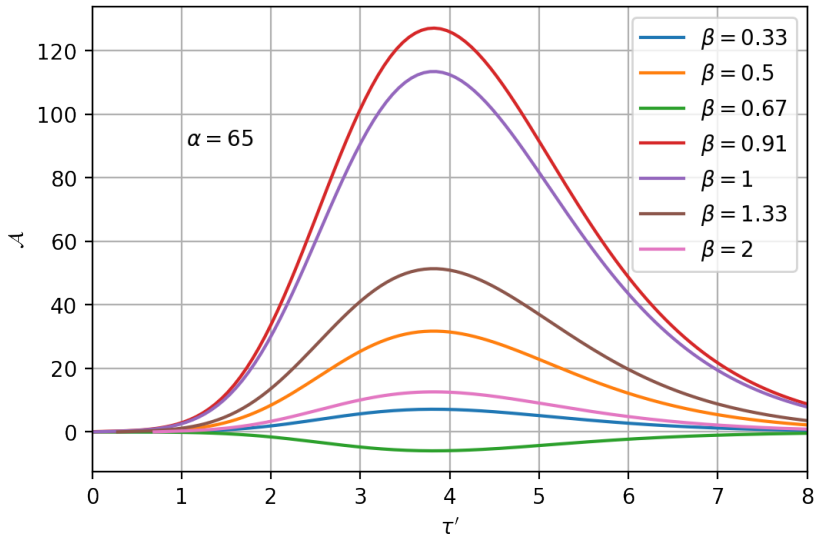

Figura 2: Amplitud adimensionalizada para $\alpha = 65$ y distintos valores de β

Se muestra la distribución de la amplitud adimensionalizada $\mathcal{A}$ para $\alpha = 80$

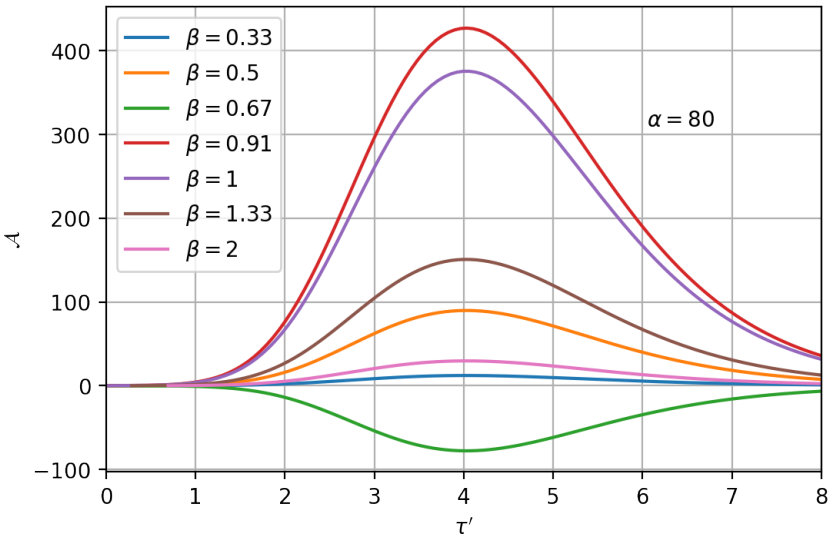

Figura 3: Amplitud adimensionalizada para $\alpha = 80$ y distintos valores de β

4. Análisis de Resultados

Con los resultados presentados es valido destacar que el método de Runge Kutta de orden 4 logró obtener una solución precisa de la ecuación diferencial 3, mediante la resolución del sistema 6. De esta forma, se pudo variar los parámetros de esta ecuación para estimar el comportamiento de las fluctuaciones en la amplitud dada una perturbación en la burbuja.

Para los resultados obtenidos, se nota que se eligió graficar la amplitud adimensionalizada respecto a $\tau' = \ln(\lambda/\lambda_c)$ al igual que [1] para mostrar la relación del crecimiento de la amplitud de la perturbación con el valor instantáneo de λ/λ_c. De las 3 figuras presentadas se nota que cuando $\lambda > \lambda_c$, la amplitud crece exponencialmente hasta un máximo aproximado en $\tau' = 3.5$ para $\alpha = 50$ y $\tau' = 4$ para $\alpha = 80$.

Además, de las mismas figuras para distintos α desde $\tau' = 0$ hasta $\tau' = 8$, se nota que a medida que aumenta el α, la amplitud alcanza un máximo mayor a los otros casos. Esto quiere decir que para los casos en que la burbuja es más grande (mayor radio de curvatura), la amplitud de las perturbaciones es mayor. Para $\alpha = 50$ se alcanza una amplitud máxima $\mathcal{A} \approx 34$, mientras que para $\alpha = 65$ se alcanza una amplitud máxima $\mathcal{A} \approx 130$ y para $\alpha = 80$ se alcanza una amplitud máxima $\mathcal{A} \approx 420$. En todos estos casos se llegó que el máximo ocurre cuando $\beta = 0.91$

De lo anterior, dado que $\beta = \lambda_0/\lambda_c$, se nota que para cuando la longitud de onda de la perturbación inicial es similar a la crítica, la amplitud crece y llega al máximo. Además, para los casos cuando $\lambda_0 < \lambda_c$, la amplitud tiende a oscilar y luego crece hasta su máximo, como se ve en la siguiente figura:

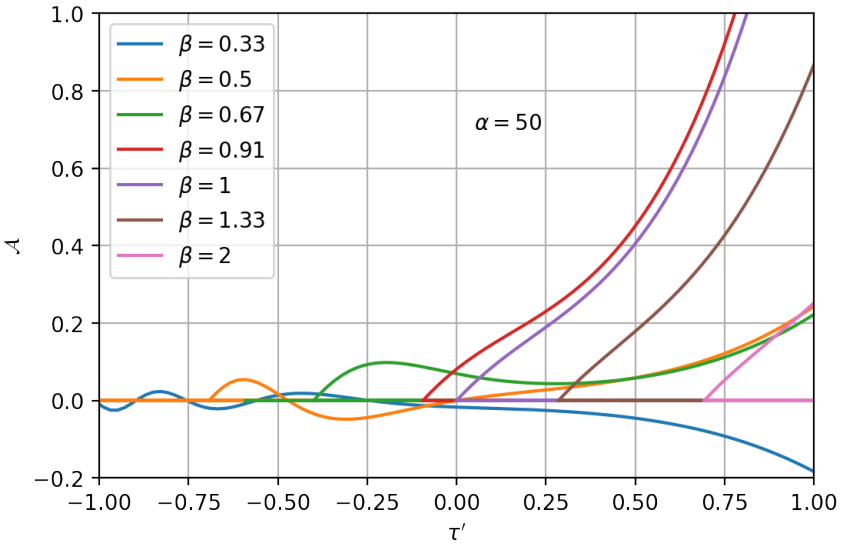

Figura 4: Oscilaciones en la amplitud adimensionalizada para $\alpha = 50$

Con lo dicho anteriormente, si una burbuja asciende por un líquido y experimenta una perturbación infinitesimal, la amplitud crece de la manera descrita anteriormente. Lo importante es que este crecimiento depende de la longitud de la perturbación inicial, obteniendo máximos cuando esta es similar a la crítica y además, el máximo es mayor si la burbuja es más grande.

Respecto al método se nota que el método de RK-4 programado resuelve con muy buena precisión la ecuación diferencial planteada. De esta forma se pudo obtener las curvas presentadas y obtener conclusiones respecto a la amplitud de las perturbaciones en la burbuja. El hecho de utilizar variables de doble precisión contribuye a un menor error en los cálculos.

5. Conclusión

Para finalizar, se nota que mediante el método programado de Runge Kutta 4 se pudo obtener el comportamiento de la amplitud de la perturbación en la superficie de una burbuja ascendiendo por un líquido. La ecuación que gobierna este fenómeno es una ecuación diferencial lineal de segundo orden, la cual mediante un cambio de variables se pudo llevar a un sistema de ecuaciones lineales de primer orden. El propósito fue utilizar este sistema de ecuaciones para resolverlo mediante el método de Runge Kutta.

Es importante señalar que se eligió un paso bastante pequeño $h \approx 0.01$ para obtener una solución con un gran orden de convergencia, dado que RK-4 es de orden 4. Esto se ve reflejado en las gráficas presentadas, siendo bastante similar a las presentadas en la publicación de G. K. Bachelor [1]. Asimismo, se logró con una gran precisión modelar las oscilaciones que ocurren en la amplitud para $\tau' < 0$. Es importante destacar que la ecuación diferencial que gobierna este fenómeno pudo haber sido resuelta con otros métodos, como Euler explícito, Euler implícito, método del trapecio, entre otros. Lo único que variaría sería el orden de convergencia, el cual en estos casos es menor. Dado esto, se necesitaría un paso temporal menor y por ende, mayor costo computacional.

Respecto a lo modelado, se obtuvo que la amplitud es máxima cuando la longitud de la perturbación alcanza casi 4 veces la crítica, $\lambda \approx 3.7\lambda_c$ para los distintos tamaños de burbujas y cuando la longitud de onda de la perturbación inicial es cercana a la crítica, $\lambda_0 = 0.91\lambda_c$. Además, mientras más grande el tamaño de la burbuja, se alcanza una mayor amplitud de la perturbación en el punto detallado anteriormente.

6. Referencias

[1] G. K. Batchelor, *The stability of a large gas bubble rising through liquid*, Journal of Fluid Mechanics, Vol. 184, pp. 399-422 (1987)

[2] Steven C. Chapra Raymond P. Canale. (2007). *Métodos numéricos para ingenieros*. The McGraw-Hill.

[3] A. Quarteroni, R. Sacco, F. Saleri (2000). *Numerical Mathematics*. Springer

7. Anexos

Se presenta el código programado para la resolución:

Código 1: Programa para RK-4 aplicado al problema.

```
1
2  !Programa que resuleve sistema de edo con Runge Kutta 4
3  program main
4      implicit none
5      integer,parameter :: n=2
6      integer :: i,mm
7      real(8) :: tf,tout,ti,dt,ai(n),alpha,beta
8      real(8), allocatable, dimension(:) :: tp,tp2
9      real(8), allocatable, dimension(:,:) :: ap
10
11     ti = 0
12     ai(1) = 0
13     ai(2) = 1
14
15     !Se lee de archivo tf, alpha y beta
16     call ReadFile(tf,alpha,beta)
17
18     dt = 0.01
19     tout = 0.03
20     mm = floor(tf/tout) + 2
21
22     allocate (tp(mm))
23     allocate (ap(n,mm))
24     allocate (tp2(mm))
25
26     !Se corre RK4
27     call Main_RK4(n,ti,tf,tout,dt,ai,ap,tp,mm,alpha,beta)
28
29     !write(*,*) tp
30     write(*,*)
31     !write(*,*) ap(1,:)
32
33     do i=1,mm
34         tp2(i) = tp(i) + log(beta)
35     end do
36
37     !Se escribe en archivo de salida
38     call WriteFile(mm,tp,ap(1,:),tp2)
39
40
41 end program main
42
43
```

```
44 ! Edos a resolver
45 subroutine Derivs(x,y,dy,n,a,b)
46
47     implicit none
48     integer, intent(in) :: n
49     real(8), intent(in) :: x,y(n),a,b
50     real(8), intent(out) :: dy(n)
51     real(8) :: k
52
53     k = 1/(b*exp(x))
54
55     dy(1) = y(2)
56     dy(2) = -3*y(2) - (2-a*k*(1-k**2))*y(1)
57
58 end subroutine
59
60
61 !Main de RK4
62 subroutine Main_RK4(n,xi,xf,xout,dx,yi,yp,xp,mm,a,b)
63
64     implicit none
65     integer, intent(in) :: n,mm
66     real(8), intent(in) :: xi,xf,dx,yi(n),xout,a,b
67     real(8), intent(out) :: yp(n,mm),xp(mm)
68     integer :: m,i
69     real(8) :: x,xend,h,y(n)
70
71     x = xi
72     m = 1
73     xp(m) = x
74
75     do i=1,n
76         yp(i,m) = y(i)
77         y(i) = yi(i)
78     end do
79
80     do while(.true.)
81         xend = x + xout
82         if(xend.gt.xf) then
83             xend = xf
84         end if
85         h = dx
86         call Integrator(x,y,n,h,xend,a,b)
87         m = m + 1
88         xp(m) = x
89         do i=1,n
90             yp(i,m) = y(i)
91         end do
```

```
92          if(x.ge.xf) then
93              exit
94          end if
95      end do
96
97  end subroutine
98
99
100 subroutine Integrator(x,y,n,h,xend,a,b)
101
102     implicit none
103     integer, intent(in) :: n
104     real(8), intent(in) :: a,b
105     real(8), intent(inout) :: x,y(n),xend,h
106
107     do while(.true.)
108
109         if((xend-x).lt.h) then
110             h = xend - x
111         end if
112         call RK4(x,y,n,h,a,b)
113         if(x.ge.xend) then
114             exit
115         end if
116
117     end do
118
119 end subroutine
120
121
122 subroutine RK4(x,y,n,h,a,b)
123
124     implicit none
125     integer, intent(in) :: n
126     real(8), intent(in) :: h,a,b
127     real(8), intent(inout) :: x,y(n)
128     real(8) :: ye(n),ym(n),sl(n),k1(n),k2(n),k3(n),k4(n)
129
130     call Derivs(x,y,k1,n,a,b)
131     ym(:) = y(:) + k1(:)*h/2
132
133     call Derivs(x+h/2,ym,k2,n,a,b)
134     ym(:) = y(:) + k2(:)*h/2
135
136     call Derivs(x+h/2,ym,k3,n,a,b)
137     ye(:) = y(:) + k3(:)*h
138
139     call Derivs(x+h,ye,k4,n,a,b)
```

```
140     sl(:) = (k1(:) + 2*(k2(:)+k3(:)) + k4(:))/6
141     y(:) = y(:) + sl(:)*h
142 
143     x = x + h
144 
145 end subroutine
146 
147 
148 ! Para escribir archivo salida
149 subroutine WriteFile(n,x,y,z)
150     implicit none
151     integer, intent(in) :: n
152     integer :: i,stat
153     real(kind=8), intent(in) :: x(n),y(n),z(n)
154 
155     open(unit=20,file='Results_data.txt',iostat=stat,action='write')
156 
157         if(stat/=0) then
158             write(*,*)'Error al abrir el archivo con iostat',stat
159         end  if
160 
161         do i=1,n
162             write(20,'(3F12.6)')x(i),y(i),z(i)
163         end do
164     close(unit=20,iostat=stat)
165 
166     if(stat/=0) then
167         write(*,*)'Error al cerrar el archivo con iostat',stat
168     end if
169 
170     write(*,*)
171     write(*,*) "Archivo Guardado"
172     write(*,*)
173 
174 end subroutine WriteFile
175 
176 
177 ! Para leer archivo de entrada
178 subroutine ReadFile(tf,a,b)
179     implicit none
180     integer :: stat
181     real(8), intent(inout) :: tf,a,b
182 
183     open(unit=10,file='input_data.txt',iostat=stat,action='read')
184 
185         if(stat/=0) then
186             write(*,*)'Error al abrir el archivo con iostat',stat
187         end  if
```

```
188
189         read(10,*)tf
190         read(10,*)a
191         read(10,*)b
192
193     close(unit=10,iostat=stat)
194
195     if(stat/=0) then
196         write(*,*)'Error al cerrar el archivo con iostat',stat
197     end if
198
199     write(*,*)
200     write(*,*) "Archivo Leido:"
201     write(*,*)
202     write(*,*) 'Tiempo de Integracion', tf
203     write(*,*) 'Alpha', a
204     write(*,*) 'Beta', b
205     write(*,*)
206
207 end subroutine ReadFile
```

Código 2: Programa para creación de gráficos utilizando el main.exe creado

```
1
2  # Programa para graficar valores iterando el archivo exe creado por
       fortran
3
4  import numpy as np
5  import matplotlib.pyplot as plt
6  plt.rcParams['figure.dpi'] = 200
7  plt.rcParams['savefig.dpi'] = 200
8  #plt.rcParams["figure.figsize"] = (10,6)
9
10 import pandas as pd
11 import os
12 import time
13
14 tf = 13
15 alpha = 50
16 betas = [0.33, 0.5, 0.67, 0.91, 1, 1.33, 2]
17
18 file = 'Results_data.txt'
19 data = pd.DataFrame()
20
21 #Loop que corre el main.exe creado en fortran con los valores de beta
22 for beta in betas:
23     with open('input_data.txt','w') as f:
24         f.write(str(tf)+"\n")
25         f.write(str(alpha)+"\n")
```

```
26          f.write(str(beta))
27
28      time.sleep(0.5)
29
30      os.startfile('main.exe')
31      time.sleep(0.5)
32
33      datax = pd.read_csv("Results_Data.txt", header=None,
            delim_whitespace=True)
34
35      data['x_beta'+str(beta)] = datax[2]
36      data['y_beta'+str(beta)] = datax[1]
37
38  #Valores guardados en dataframe son graficados
39  for beta in betas:
40      x = 'x_beta'+str(beta)
41      y = 'y_beta'+str(beta)
42      plt.plot(data[x],data[y], ls='-',linewidth=1.5, label=r'$\beta=$'+
            str(beta))
43
44  plt.grid()
45  plt.xlabel(r"$\tau'$")
46  plt.ylabel(r'$\mathcal{A}$')
47  plt.xlim([0,8])
48  #plt.text(6.05,310,r'$\alpha=$'+str(alpha))
49  plt.legend()
50  plt.show();
```