TALLER 1 - CFD

Johan Mendez[a]

[a]Universidad Nacional de Colombia , Bogotá, Departamento de Física

1. EXACTITUD DE ESQUEMAS DE DIFERENCIAS FINITAS

Se busca hallar la primera y segunda derivada de la función (1), usando esquemas de diferencias finitas uniformes de 2°, 4° y 6° orden de exactitud

$$f(x) = e^{-k_o x^2}(1.1\cos(k_1 x) - 1.3\sin(k_2 x)) \tag{1}$$

donde $xE[-\pi, \pi]$, $k_0 = 1.125$, $k_1 = \frac{\pi}{8 * \triangle x_{max}}$ y $k_2 = \frac{3.5k_1}{4}$. Y como objetivo principal se quiere comparar entre si los esquemas propuestos, teniendo en cuenta, principalmente, el error medido con norma L_2 en función del tamaño de espaciado de la malla $\triangle x$.

Formulación de esquemas de diferencias finitas

Primera derivada ($\frac{\partial f}{\partial x}$)

1. **Esquema 2° orden de exactitud.**
   Se usa un esquema centrado de 3 nodos, con $\triangle x$ el espacio entre nodos, así que se tienen en cuenta las posiciones antes y después del punto donde se quiere evaluar la derivada de la función. Luego, se toma las series de Taylor para:

$$f_{i+1} = f_i + \triangle x f_i' + \frac{(\triangle x)^2}{2!} f_i'' + ... + \frac{(\triangle x)^n}{n!} f_i^{(n)} \tag{2}$$

$$f_{i-1} = f_i - \triangle x f_i' + \frac{(\triangle x)^2}{2!} f_i'' + ... + \frac{(-\triangle x)^n}{n!} f_i^{(n)} \tag{3}$$

   Restando (2) y (3)

$$\begin{aligned} f_{i+1} - f_{i-1} &= f_i + \triangle x f_i' + \frac{(\triangle x)^2}{2!} f_i'' + ... - f_i + \triangle x f_i' - \frac{(\triangle x)^2}{2!} f_i'' + ... \\ &= 2\triangle x f_i' + 2\frac{(\triangle x)^3}{3!} f_i''' + ... \end{aligned}$$

   Luego,

$$f_i' = \frac{(f_{i+1} - f_{i-1})}{2\triangle x} + \frac{(\triangle x)^2}{3!} f_i''' + ... \tag{4}$$

$$= \frac{(f_{i+1} - f_{i-1})}{2\triangle x} + O((\triangle x)^2) \tag{5}$$

2. **Esquema 4° orden de exactitud.**
   Se usa un esquema centrado de 5 nodos, con $\triangle x$ el espacio entre nodos, así que se tienen en cuenta las posiciones antes y después del punto donde se quiere evaluar la derivada de la función. Luego, se toma las series de Taylor para:

$$f_{i+1} = f_i + \triangle x f_i' + \frac{(\triangle x)^2}{2!} f_i'' + \frac{(\triangle x)^3}{3!} f_i''' + ... \tag{6}$$

$$f_{i-1} = f_i - \triangle x f_i' + \frac{(\triangle x)^2}{2!} f_i'' - frac(\triangle x)^3 3! f_i''' + ... \tag{7}$$

E-mail: jsmendezh@unal.edu.co

$$f_{i+2} = f_i + 2\triangle x f_i' + \frac{(2\triangle x)^2}{2!} f_i'' + \frac{(2\triangle x)^3}{3!} f_i''' + ... \tag{8}$$

$$f_{i-2} = f_i - 2\triangle x f_i' + \frac{(2\triangle x)^2}{2!} f_i'' - \frac{(2\triangle x)^3}{3!} f_i''' + ... \tag{9}$$

Restando (6) y (7), (8) y (9)

$$f_{i+1} - f_{i-1} = 2\triangle x f_i' + \frac{(\triangle x)^3}{3!} f_i''' + ... \tag{10}$$

$$f_{i+2} - f_{i-2} = 4\triangle x f_i' + 2\frac{(2\triangle x)^3}{3!} f_i''' + ... \tag{11}$$

Luego, se multiplica (10) por 8 y se resta con (11)

$$\begin{aligned} 8f_{i+1} - 8f_{i-1} - f_{i+2} + f_{i-2} &= 16\triangle x f_i' + 16\frac{(\triangle x)^3}{3!} f_i''' + ... - 4\triangle x f_i' - 16\frac{(\triangle x)^3}{3!} f_i''' - ... \\ &= 12\triangle x f_i' + 16\frac{(\triangle x)^5}{!} f_i^{(5)} - 64\frac{(\triangle x)^5}{5!} f_i^{(5)} + ... \\ &= 12\triangle x f_i' - 48\frac{(\triangle x)^5}{!} f_i^{(5)} + ... \end{aligned}$$

Luego,

$$f_i' = \frac{(-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2})}{12\triangle x} - \frac{48}{12}\frac{(\triangle x)^4}{5!} f_i^{(5)} + ...$$

$$f_i' = \frac{(-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2})}{12\triangle x} + O((\triangle x)^4) \tag{12}$$

3. **Esquema** 6° **orden de exactitud.**
Se usa un esquema compacto, sacado de (referencia), la forma general de este esquema es:

$$\beta f_{i-2}' + \alpha f_{i-1}' + f_i' + \alpha f_{i+1}' + \beta f_{i+2}' = c\frac{f_{i+3} - f_{i-3}}{6h} + b\frac{f_{i+2} - f_{i-2}}{4h} + a\frac{f_{i+1} - f_{i-1}}{2h}$$

Ahora, como se asume que la función es periódica, el sistema se puede resolver como un sistema lineal de ecuaciones para los valores de derivada desconocidas. Este sistema lineal es triagonal cuando $\beta = 0$, y para tener un esquema de 6° orden, los coeficiente deben cumplir la relación

$$a + 2^4 b + 3^4 c = 2\frac{5!}{4!}(\alpha + 2^4\beta)$$

Si se elige $\alpha = \frac{1}{3}$ se tiene que $a = \frac{14}{9}$, $b = \frac{1}{9}$ y $c = 0$. de acuerdo a

$$a = \frac{1}{6}(\alpha + 9), b = \frac{1}{15}(32\alpha - 9), c = \frac{1}{10}(-3\alpha + 1)$$

Entonces, la expansión de la derivada es de la forma

$$\frac{1}{3} f_{i-1}' + f_i' + \frac{1}{3} f_{i+1}' = \frac{1}{9}\left(\frac{f_{i+2} - f_{i-2}}{4\triangle x}\right) + \frac{14}{9}\left(\frac{f_{i+1} - f_{i-1}}{2\triangle x}\right) \tag{13}$$

Segunda derivada $\left(\frac{\partial^2 f}{\partial x^2}\right)$

A partir de la primera derivada es posible hallar la segunda, teniendo el concepto de matriz de diferenciación: Conjunto de coeficientes ordenados de forma triangular o pentagonal que permite evaluar la derivada en todos los puntos del dominio.

$$\begin{bmatrix} f_1' \\ f_2' \\ . \\ . \\ . \\ f_k' \end{bmatrix} = \begin{bmatrix} b & c & 0 & 0 & ... & a \\ a & b & c & 0 & ... & 0 \\ 0 & a & b & c & ... & 0 \\ : & 0 & a & b & c & : \\ : & : & : & : & : & : \\ c & & ... & ... & a & b \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ . \\ f_N \end{bmatrix}$$

Así que se puede expresar como $F' = \mathbf{A}F$.

Recordando que $\frac{\partial^2 f}{\partial^2} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right)$. Así se tendrá que $F'' = \mathbf{A}^2 F$

1. **Esquema** 2° **orden de exactitud.**
   De acuerdo con lo anterior, se genera la matriz de diferenciación, reemplaza la ecuación (5) para diferentes valores

$$f_1' = \frac{f_2 - f_0}{2}$$
$$f_2' = \frac{f_3 - f_1}{2}$$
$$f_3' = \frac{f_4 - f_2}{2}$$
$$f_4' = \frac{f_5 - f_3}{2}$$
$$f_n' = \frac{f_{n+1} - f_{n-1}}{2}$$

   Poniendo este sistema de ecuaciones de forma matricial, se tiene

$$\begin{bmatrix} f_1' \\ f_2' \\ . \\ . \\ . \\ f_k' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & ... & -1 \\ -1 & 0 & 1 & 0 & ... & 0 \\ 0 & -1 & 0 & 1 & ... & 0 \\ : & 0 & -1 & 0 & 1 & : \\ : & : & : & : & : & : \\ 1 & & ... & ... & -1 & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ . \\ f_N \end{bmatrix}$$

   Después haciendo $\mathbf{A}^2$, se tendrá que

$$\begin{bmatrix} f_1'' \\ f_2'' \\ . \\ . \\ . \\ f_k'' \end{bmatrix} = \begin{bmatrix} -2 & 1 & 0 & 0 & ... & 1 \\ 1 & -2 & 1 & 0 & ... & 0 \\ 0 & 1 & -2 & 1 & ... & 0 \\ : & 0 & 1 & -2 & 1 & : \\ : & : & : & : & : & : \\ 1 & & ... & ... & 1 & -2 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ . \\ f_N \end{bmatrix}$$

   Así que la discretización para la segunda derivada es

$$f_i'' = \frac{f_{i-1} - 2f_i + f_{i+1}}{(\triangle x)^2} + O((\triangle x)^2) \tag{14}$$

2. **Esquema** 4° **orden de exactitud.**
Siguiendo el mismo procedimiento del esquema de orden 2 se tiene que la matriz de diferenciación para este caso es:

$$\begin{bmatrix} f_1' \\ f_2' \\ . \\ . \\ . \\ f_k' \end{bmatrix} = \frac{1}{12} \begin{bmatrix} 0 & 8 & -1 & 0 & ... & 1 & -8 \\ -8 & 0 & 8 & -1 & 0 & ... & 1 \\ 1 & -8 & 0 & 8 & -1 & 0 & 0 \\ 0 & 1 & -8 & 0 & 8 & 1 & 0 \\ : & : & : & : & : & : & : \\ 8 & -1 & ... & ... & 1 & -8 & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ . \\ f_N \end{bmatrix}$$

Después haciendo $\mathbf{A}^2$ y factorizando, se tendrá que

$$\begin{bmatrix} f_1'' \\ f_2'' \\ . \\ . \\ . \\ f_k'' \end{bmatrix} = \frac{1}{12} \begin{bmatrix} -30 & 16 & -1 & 0 & ... & -1 & 16 \\ 16 & -30 & 16 & -1 & 0 & ... & -1 \\ -1 & 16 & -30 & 16 & -1 & 0 & 0 \\ 0 & -1 & 16 & -30 & 16 & 1 & 0 \\ : & : & : & : & : & : & : \\ 16 & -1 & ... & ... & -1 & 16 & -30 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ . \\ f_N \end{bmatrix}$$

Así que la discretización para la segunda derivada es

$$f_i'' = \frac{-f_{i+2} + 16f_{i+1} - 30f_i + 16f_{i-1} - f_{i-2}}{12(\triangle x)^2} + O((\triangle x)^4) \tag{15}$$

3. **Esquema** 6° **orden de exactitud.**

Como para este orden se esta usando un esquema compacto, es mejor y más certero hallar la forma de la segunda derivada análogo a la primera derivada. Partiendo de una expresión general[1]

$$\beta f_{i-2}'' + \alpha f_{i-1}'' + f_i'' + \alpha f_{i+1}'' + \beta f_{i+2}'' = c\frac{f_{i+3} - 2f_i + f_{i-3}}{9h^2} + b\frac{f_{i+2} - 2f_i + f_{i-2}}{4h^2} + a\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}$$

Para funciones que son periódicas en x, el sistema tridiagonal o pentadiagonal definido por la expresión anterior, cada nodo puede ser resuelto para obtener las segundas derivadas. Para obtenr un esquema tridiagonal se define $\beta = 0$ y $\alpha = 2/11$ se calculan los otros parámetros con

$$a = \frac{6 - 9\alpha - 12\beta}{4}, b = \frac{-3 + 24\alpha - 6\beta}{5}, c = \frac{2 - 11\alpha + 124\beta}{20}$$

Se tiene $a = 12/11$, $b = 3/11$ y $c = 0$.
Finalmente, la expansión de la derivada es de la forma

$$\frac{2}{11}f_{i-1}'' + f_i' + \frac{2}{11}f_{i+1}'' = \frac{3}{11}\frac{f_{i+2} - 2f_i + f_{i-2}}{4h^2} + \frac{12}{11}\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} \tag{16}$$

Solución numérica

Se presentan los códigos (Apéndice) con los que se hicieron cada derivadas en diferentes esquemas, definiendo primero la función y sus características

Se usaron dos códigos diferentes el primero para las derivadas con CDS (segundo y cuarto orden de exactitud) y otro a parte para la derivación con un esquema compacto.

Y para todos los sistemas se calculo el error absoluto que esta dado por la función:

$$L_2 = \frac{1}{N}\sqrt{\sum (f_i' - f'_{exacta})^2}$$

Así que para ver la exactitud que tienen los diferentes esquemas se compara de acuerdo a sus errores y tamaño de malla elegido $\triangle x$. Para la primera derivada se tiene

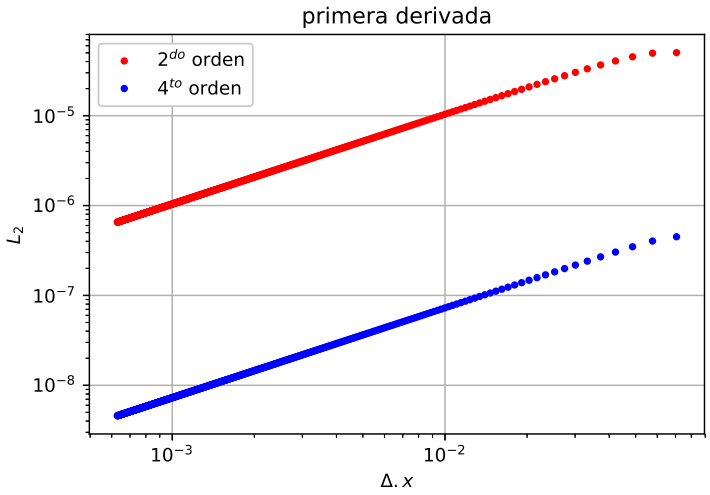

Figura 1. Error promedio en la primera derivada, comparando el segundo y cuarto orden de exactitud

La figura (1) presenta los resultados para los esquemas de segundo y cuarto orden de diferencias finitas, se encuentra que el cuarto orden presenta menor error promedio en el estudio de los modelos desarrollados, esto evidencia que es más preciso considerar más nodos en las soluciones de las ecuaciones, sin embargo es más costoso computacionalmente ser un poco más precisos. A continuación se presenta el mismo análisis para la segunda derivada

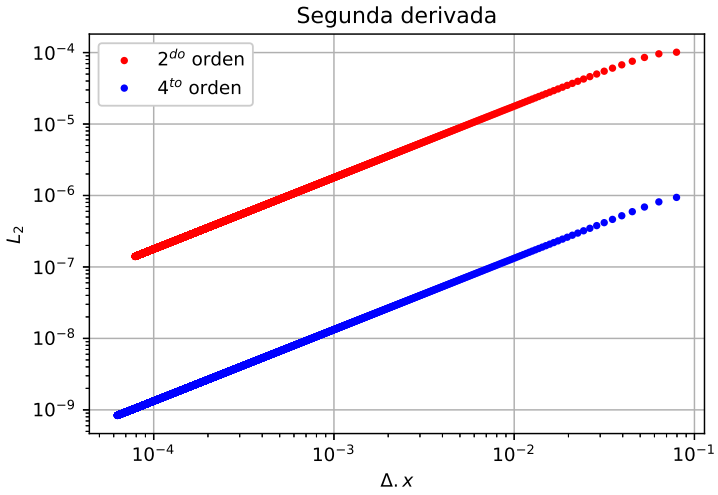

Figura 2. Error promedio en la primera derivada, comparando el segundo y cuarto orden de exactitud

Efectivamente el esquemas que más error presenta de acuerdo al tamaño de malla es el de diferencias finitas de 2° orden. Por otro lado, hablando específicamente de Δx se encuentra una relación lineal con el error promedio de los datos, estas gráficas pueden dar una idea clara para cuando en futuras ocasiones se quiera tener menos o más resolución espacial en la solución de las ecuaciones diferenciales, esto porque si se quiere obtener un resultado más preciso, se hace necesario estar dispuesto a utilizar más nodos en la malla computacional y esto implica tiempo de computo o si por el contrario no se requiere una precisión tan grande se puede estudiar el sistema con baja resolución aceptando que las soluciones distan un poco más de las reales.

Para el estudio de los esquemas compactos se utilizó un algoritmo de Tomas pasa invertir la matriz y poder encontrar las derivadas , sin embargo son contrario a los esperados, dado que el esquema compacto que se estudio de manera teórica debería ser más preciso , a continuación se presenta el resultado obtenido

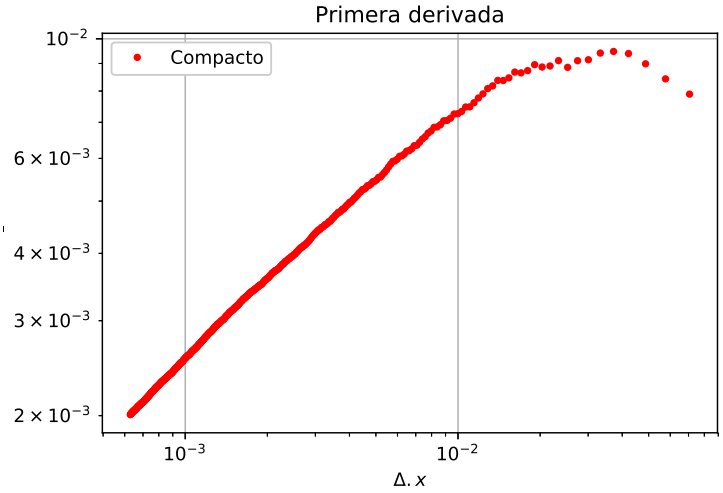

Figura 3. Error promedio en la primera derivada utilizando esquemas compactos

Se atribuye este resultado incorrecto a la implementación computacional, y esta es una observación, dado que la implementación de esquemas compactos es un poco más dispendiosa que la que se utiliza en diferencias finitas. Se debe revisar y plantear de nuevo los códigos con los cuales se estudio los modelos compactos.

2. ECUACIÓN DE DIFUSIÓN CONVECCIÓN 1D

La ecuacion de difusi';on convección unidimensional está dada por la siguiente ecuación

$$\frac{\partial(\rho u \phi)}{\partial x} = \frac{\partial}{\partial x}\left(\Gamma \frac{\partial \phi}{\partial x}\right) \tag{17}$$

Se considera para esta ecuación condiciones de frontera tipo Dirichlet: $\phi(x = 0) = 0.0$, $\phi(x = L) = 1.0$, Además se considera un campo de velocidades dado por $u = 1.0$ y una densidad constante $\rho = 1.0$, con $\Gamma = 0.01667$. La solución analítica está dada por

$$\phi = \phi_o + \frac{x^{Pe/L} - 1}{e^{Pe} - 1}(\phi_L - \phi o), \qquad Pe = \frac{\rho u L}{\Gamma} \tag{18}$$

Este problema puede ser utilizado como un test para corroborar la validez de los métodos numéricos implementados. Físicamente representa cuando la convección hace un balance con la difusión en las direcciones de las líneas de corriente

2.0.1 Esquema de diferencia central

A continuación se presenta de la ecuación discreta considerando diferencias finitas centradas, en un enmallado no uniforme , por tanto

$$
\begin{aligned}
Pe\frac{\partial \phi}{\partial x} &= \frac{\partial^2 \phi}{\partial x^2} \\
Pe\left(\frac{\phi_{i+1}-\phi_{i-i}}{x_{i+1}-x_i}\right) &= \frac{\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}-\frac{\phi_i-\phi_{i-1}}{x_i-x_{i-1}}}{1/2(x_{i+1}-x_{i-1})} \\
Pe\left(\frac{\phi_{i+1}-\phi_{i-i}}{x_{i+1}-x_i}\right) &= \frac{(x_i-x_{i-1})\phi_{i+1}-(x_{i+1}-x_{i-1})\phi_i+(x_{i+1}-x_i)\phi_{i-1}}{(x_{i+1}-x_i)(x_i-x_{i-1})(x_{i+1}-x_{i-1})} \\
\frac{(x_{i+1}-x_{i-1})}{(x_{i+1}-x_i)(x_i-x_{i-1})}\phi_i &= \left[\frac{1}{x_{i+1}-x_i}-\frac{Pe}{2}\right]\phi_{i+1}+\left[\frac{1}{x_i-x_{i-1}}+\frac{Pe}{2}\right]\phi_{i-1} \\
\phi_i &= \frac{(x_{i+1}-x_{i-1})}{(x_{i+1}-x_i)(x_i-x_{i-1})}\left(\left[\frac{1}{x_{i+1}-x_i}-\frac{Pe}{2}\right]\phi_{i+1}+\left[\frac{1}{x_i-x_{i-1}}+\frac{Pe}{2}\right]\phi_{i-1}\right)
\end{aligned}
$$

Definamos $r1 = x_{i+1} - x_i$, $r2 = x_i - x_{i-1}$, obteniendo una forma más compacta

$$
\boxed{\phi_i = \frac{r_1 r_2}{r_1+r_2}\left(\left[\frac{1}{r_1}-\frac{Pe}{2}\right]\phi_{i+1}+\left[\frac{1}{r_2}+\frac{Pe}{2}\right]\phi_{i-1}\right)} \tag{19}
$$

Para resolver espacialmente la variable ϕ_i se utilza un enmallado dado por el siguiente coeficiente de expansión

$$
\Delta x_{i+1} = r_e \Delta x_i, \quad r_e = 0.7, \qquad \Delta x_{i+1} = x_{i+1} - x_i, \quad \Delta x_i = x_i - x_{i-1}
$$

En donde se utiliza

$$
\boxed{x_{i+1} = (1+r_e)x_i - x_{i-1}} \tag{20}
$$

Cuando se considera un enmallado uniforme se impone la condición $r_1 = r_2 = \Delta x$, obteniendose

$$
\boxed{\phi_i = \frac{\Delta x}{2}\left(\left[\frac{1}{\Delta x}-\frac{Pe}{2}\right]\phi_{i+1}+\left[\frac{1}{\Delta x}+\frac{Pe}{2}\right]\phi_{i-1}\right)} \tag{21}
$$

Este esquema considera la dirección del campo de velocidades sobre el cual evoluciona ϕ para considerar la mejor forma discreta de la primera derivada, es decir

$$
\left[u\frac{\partial \phi}{\partial x}\right]_i \approx \begin{cases} u(\phi_i - \phi_{i-1})/(x_i - x_{i-1}) & u > 0 \\ u(\phi_{i+1} - \phi_i)/(x_{i+1} - x_i) & u < 0 \end{cases} \tag{22}
$$

Se considera entonces para la parte convectiva la formulación upwind en un enmallado uniforme , dada la condición inicial del campo de velocidaddes u = 1.0, entonces la ecuación resulta de la siguiente manera

$$
\begin{aligned}
Pe\frac{\partial \phi}{\partial x} &= \frac{\partial^2 \phi}{\partial x^2} \\
Pe\left(\frac{\phi_i - \phi_{i-1}}{\Delta x}\right) &= \frac{2}{\Delta x^2}\left(\phi_{i-1} - 2\phi_i + \phi_{i+1}\right) \\
\left(Pe + \frac{4}{\Delta x}\right)\phi_i &= \left(Pe + \frac{2}{\Delta x}\right)\phi_{i-1} + \frac{2}{\Delta x}\phi_{i+1}
\end{aligned}
\tag{23}
$$

Entonces la ecuación que queremos calcular es la siguiente

$$
\boxed{\phi_i = \frac{\Delta x Pe + 2}{\Delta x Pe + 4}\phi_{i-1} + \frac{2}{\Delta x + 4}\phi_{i-1}, \qquad Pe = \frac{\rho u L}{\Gamma}}
\tag{24}
$$

A continuación se presentan los códigos correspondientes a cada esquema utilizado las siguientes siglas (CDS) esquema centrado uniforme, (UDS) esquema upwind uniforme, (CDS NU) esquema centrado no uniforme , las variables definidas y condiciones iniciales son las siguientes

```
integer, parameter :: dp = selected_real_kind(15)
integer :: i,n,j,k,e,nodos
real(dp), parameter :: l=1.0_dp,rho=1.0_dp,u=1.0_dp,gamma=0.01667_dp
real(dp) :: dx,beta,b,Pe,r1,r2
real(dp), allocatable :: phi(:),x(:),phi1(:)

dx = 1.0/41.0
n = l/dx
nodos = 11.0
allocate(phi(n),phi1(nodos),x(nodos))
phi(1) = 0.0_dp
phi1(1) = 0.0_dp
x(1) = 0.0_dp
x(2) = 0.31_dp
Pe = rho*u*l/gamma
b = 2*gamma/(rho*u*l)
beta =b/dx
```

En primer lugar se considera el bucle principal de cada uno de los siguientes esquemas, para poder observar el modo en que se resuelve cada ecuación , se empieza con el CDS NU, el cual utiliza la ecuacion (19), entonces

```
do i = 2,nodos-1
 x(i+1) = 0.7_dp*(x(i)-x(i-1)) + x(i)
 r1 = x(i+1)-x(i) ; r2 = x(i)-x(i-1)
 phi1(i) = (r1*r2/(r1+r2))*((1.0_dp/r1 - 0.5_dp*Pe)*phi1(i+1) + (1.0_dp/r2 + 0.5_dp*Pe)*phi1(i-1))
end do
phi1(nodos) = 1.0_dp
```

Luego se presenta el proceso iterativo para las diferencias centradas uniformes (CDS), utilizando la ecuacion (21). Para este código se define una variable auxiliar $\beta = 1/(\Delta x Pe)$ esto con el fin de hacer más fácil la sintaxis

```
do i = 2,n-1
 phi(i) = ( (beta+1.0_dp)*phi(i-1)+(beta-1.0_dp)*phi(i+1) )/(2.0_dp*beta)
end do
phi(n) = 1.0_dp
```

Finalmente el esquema upwind (UDS) el cual se relaciona con al ecuacion (24)

```
do i = 2,n-1
 phi(i) = (2.0_dp/(dx*Pe+4.0_dp))*phi(i+1) + ((dx*Pe+2.0_dp)/(dx*Pe+4.0_dp))*phi(i-1)
end do
phi(n) = 1.0_dp
```

Sin embargo el código completo se anexa al final del documento. Las soluciones de los moedelos se comparan con el módelo teórico y se muestran en el siguiente grafico

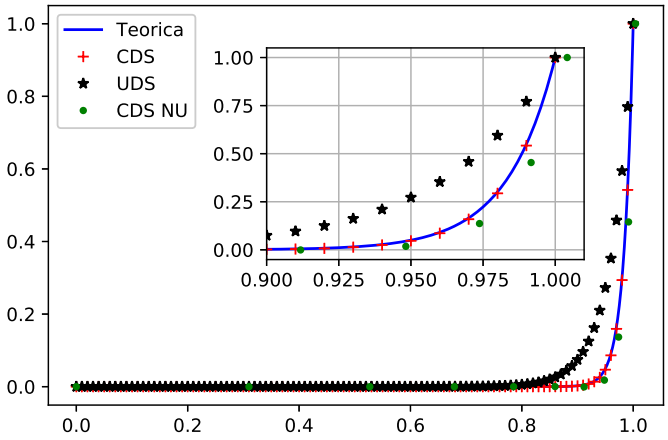

Figura 4. Error promedio en la solución unidimensional de la ecuación de convección difusión para un numero de Peclet $Pe \approx 60$ como función del espacio del enmallado

Dado que en este caso se conoce la solución de (17) podemos estimar el error fácilmente utilizando (18) utilizando la siguiente expresión[2]

$$\epsilon = \frac{1}{N} \sum_i |\phi_i^{eq} - \phi_i| \tag{25}$$

Ahora se presenta el error en función del tamaño del enmallado

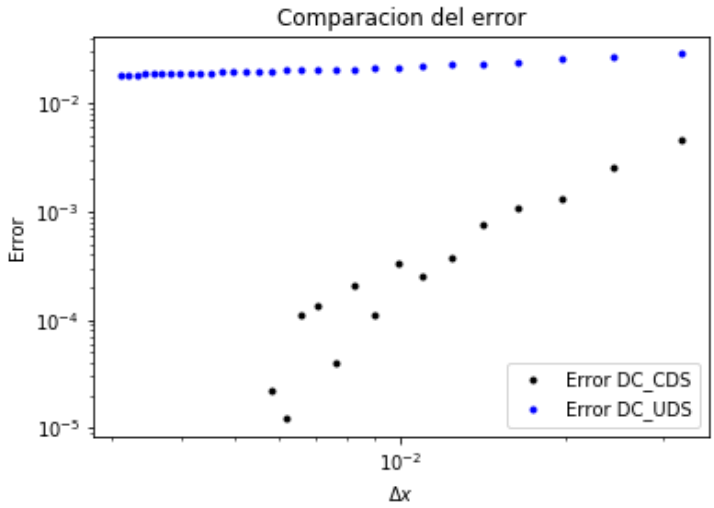

Figura 5. Solución de la ecuación convección difusión cono un Número de Peclet ($Pe \approx 60$) usando tres esquemas. Diferencias centradas uniformes, no uniformes y un esquema upwind

La anterior relación muestra que el esquema de diferencias finitas centrado presenta un error promedio más bajo que el esquema de diferencias finitas upwind, además de esto en cuanto el enmallado es más fino cae mucho más rápido el error para CDS mientras que para UDS queda acotado alrededor de 10^2. Estos cálculos se hicieron para enmallados desde 30 nodos hasta 320 nodos.

Las ventajas de estos métodos estudiados radica principalmente en la facilidad de implementación computacional, sin embargo se debe tener en cuenta que esto depende exclusivamente de cada ecuación diferencial a resolver, que en este caso particular es la ecuación de advección difusión una simplificación de las ecuaciones de Navier Stokes un "toy model", en donde se considera una aproximación lineal con una velocidad advectiva constante $u(\geq 0)$ y difusividad $\Gamma(\geq 0)$ entonces la ecuación (17) sirve como test para diferentes esquemas de discretización.

La eficiencia de estos métodos es bastante buena, dado que siempre se pudo considerar esquemas explícitos a la hora de encontrar la ecuación de evolución del sistema para cada uno de los modelos discretos, ahora, la diferencia entre UDS y CDS radica principalmente en la precisión dado que en la parte convectiva si tiene en cuenta la dirección del campo de velocidades sobre el cual evoluciona ϕ y se emplean primeras derivadas. En cambio para CDS tenemos tenemos segundo orden de precisión y no importa la dirección en la cual el campo de velocidades esté dispuesto, si se considera el espaciamiento no uniforme se tiene la ventaja de dar resolución al sistema en lugares cuando el error es mayor. Una desventaja de estos sistemas se puede evidenciar en el hecho de modelar geometrías más complicadas , hablando de simulaciones más robustas, dado que por ahora solamente tenemos sistemas cartesianos a la hora de construir el enmallado sobre el cual se solucionan las ecuaciones diferenciales. Como el objetivo de las diferencias finitas es convertir las derivadas continuas en combinaciones lineales de las funciones evaluadas en los nodos , cuando resolvemos el sistema estamos calculando ecuaciones en proporción a los nodos considerados esto en simulaciones más grandes puede llevar un costo computacional importante

3. ECUACIÓN DE DIFUSIÓN-CONVECCIÓN TRANSITORIA 1D

La ecuación de difusión convección transitoria unidimensional está dada por la siguiente ecuación

$$\frac{\partial \rho\phi}{\partial t} + \frac{\partial(\rho u\phi)}{\partial x} = \frac{\partial}{\partial x}\left(\Gamma \frac{\partial \phi}{\partial x}\right) \tag{26}$$

Se considera para esta ecuación condiciones de frontera tipo Dirichlet: $\phi(x=0)=0.0$, $\phi(x=L)=1.0$, además se tiene la condición inicial: $\phi(x,0)=\sin(x\pi/2)$. Se considera un campo de velocidades dado por $u=1.0$ y una densidad constante $\rho=1.0$, con $\Gamma=0.025$. Como condición inicial del caso transitorio

$$\phi(x,0)=\phi_o\cos\left(x\frac{\pi}{2}\right)+\phi_L\sin\left(x\frac{\pi}{2}\right) \tag{27}$$

El problema transitorio evoluciona en el tiempo desde la condición inicial $t_1\approx 0$ hasta cuando se tiene una condición estacionaria en el tiempo $t_6\approx 1$, esto se puede comparando este último tiempo con el problema estudiado anteriormente en el caso estacionario, veáse la figura (4)

Se realizó un estudio numérico del perfil de ϕ a lo largo de x en función del tiempo, usando esquemas de diferencias finitas centradas uniformes y centradas no-uniformes para la parte espacial. El marco teórico de este tema se desarrollo en el anterior ejercicio sección 2.0.1.

Por otro lado, para el caso transitorio se debe ver la discretización temporal que se aplica a problemas definidos por ecuaciones diferenciales ordinarias de primer orden en el tiempo junto con las condiciones iniciales correspondientes. A este tipo de problemas se les denomina problemas de Cauchy. Para este problema se usaran dos esquemas de avance temporal: Euler hacia adelante y Runge-Kutta-2.

3.1 Esquemas de avance temporal

EL modelo implementado en particular está dado por

$$\frac{\partial\phi}{\partial t}=-u\frac{\partial\phi}{\partial x}+\frac{\Gamma}{\rho}\frac{\partial^2\phi}{\partial x^2}=f(t,\phi) \tag{28}$$

Se utiliza entonces diferencias finitas centradas para la forma discreta de las derivadas espaciales , en este caso se utiliza el esquema de enmallado no uniforme, de tal manera que se describen completamente el sistema no uniforme y uniforme, entonces se tiene

$$\frac{\partial\phi}{\partial t} = -u\left(\frac{\phi^n_{i+1}-\phi^n_{i-1}}{x_{i+1}-x_{i-1}}\right)+\frac{2\Gamma}{\rho}\left(\frac{(x_i-x_{i-1})\phi^n_{i+1}-(x_{i+1}-x_{i-1})\phi^n_i+(x_{i+1}-x_i)\phi^n_{i-1}}{\frac{1}{2}(x_{i+1}-x_{i-1})(x_{i+1}-x_i)(x_i-x_{i-1})}\right)$$

Este es el punto de partida para calcular la evolución temporal.

3.1.1 Euler hacia adelante

Este es el método más elemental en el cual el flujo y la fuente son evaluadas en definidos t^n.

$$\frac{\partial\phi}{\partial t}=\frac{\phi^{n+1}_i-\phi^n_i}{\triangle t}$$

Primero, se utiliza la forma discreta de CDS con el avance temporal dado por la anterior formula de avance temporal de Euler, entonces

$$\frac{\phi^{n+1}_i-\phi^n_i}{\triangle t}=-u\left(\frac{\phi^n_{i+1}-\phi^n_{i-1}}{x_{i+1}-x_{i-1}}\right)+\frac{2\Gamma}{\rho}\left(\frac{(x_i-x_{i-1})\phi^n_{i+1}-(x_{i+1}-x_{i-1})\phi^n_i+(x_{i+1}-x_i)\phi^n_{i-1}}{\frac{1}{2}(x_{i+1}-x_{i-1})(x_{i+1}-x_i)(x_i-x_{i-1})}\right) \tag{29}$$

Ahora se escribe la ecuación de manera conveniente

$$\phi_i^{n+1} = \left[1 - \frac{2\Gamma}{\rho}\frac{\Delta t}{(x_{i+1}-x_i)(x_i-x_{i-1})}\right]\phi_i^n + \left[\frac{2\Gamma}{\rho}\frac{\Delta t}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)} - \frac{u\Delta t}{x_{i+1}-x_{i-1}}\right]\phi_{i+1}^n + \left[\frac{2\Gamma}{\rho}\frac{\Delta t}{(x_{i+1}-x_{i-1})(x_i-x_{i-1})} + \frac{u\Delta t}{x_{i+1}-x_i}\right]\phi_{i-1}^n \tag{30}$$

Definiendo

$$a_1 = \frac{2\Gamma}{\rho}\frac{\Delta t}{(x_{i+1}-x_i)(x_i-x_{i-1})} \qquad a_2 = \frac{2\Gamma}{\rho}\frac{\Delta t}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}$$
$$a_3 = \frac{u\Delta t}{x_{i+1}-x_{i-1}} \qquad a_4 = \frac{2\Gamma}{\rho}\frac{\Delta t}{(x_{i+1}-x_{i-1})(x_i-x_{i-1})}$$

Entonces la ecuación a resolver es

$$\boxed{\phi_i^{n+1} = [a_4 + a_3]\phi_{i-1}^n + [1 - a_1]\phi_i^n + [a_2 - a_3]\phi_{i+1}^n} \tag{31}$$

Para esto se muestra el código del ciclo principal.

```
do j = 1, tf
 do i = 2,n-1
  a1 = (2.0*gamma*dt)/( rho*(x(i+1)-x(i))*(x(i)-x(i-1))  )
  a2 = (2.0*gamma*dt)/( rho*(x(i+1)-x(i-1))*(x(i+1)-x(i)) )
  a3 = u*dt/(x(i+1)-x(i-1))
  a4 = (2.0*gamma*dt)/(rho*(x(i+1)-x(i-1))*(x(i)-x(i-1)) )
  phi2(i) = (1-a1)*phi(i) + (a2-a3)*phi(i+1) + (a4+a3)*phi(i-1)

  phi(i)= (1.0-2.0*d)*phi(i)+(d-0.5*c)*phi(i+1)+(d+0.5*c)*phi(i-1)
 end do
```

Ahora se muestra la implementación del método descrito anteriormente , para esto se comparar los perfiles numéricos para mallas uniformes y no uniformes

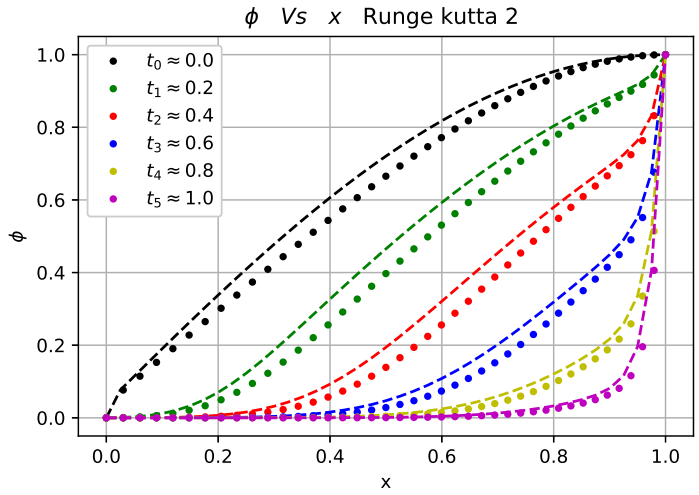

Figura 6. Perfil numérico obtenido mediante Euler hacia adelante considerando algunos tiempos en el esquema de evolución temporal , además vemos que se identifica la malla no uniforme con el símbolo +

3.1.2 Runge-Kutta-2

Este es un esquema numérico multietapa , es decir, halla la solución iterativamente usando 2 etapas. Utiliza no sólo los instantes t^n y t^{n+1} sino también otros intermedios. La primera etapa es en $t = t^n$ donde

$$f^n = k_1 = f(t^n, y^n)$$

la segunda etapa es en $t = t^n + \triangle t^n$, y se tiene

$$f^{n+1} = k_2 = f\left(t^n + \triangle t^n, y^n + \frac{\triangle t^n}{2} k_1\right)$$

Así ϕ es de la forma

$$\phi^{n+1} = \phi^n + \triangle t(k_1 + k_2)$$

Para este caso se definen k_1 y k_2 utilizando (29)

$$\begin{aligned}\frac{\phi_i^{n+1} - \phi_i^n}{\triangle t} &= -u\left(\frac{\phi_{i+1}^n - \phi_{i-1}^n}{x_{i+1} - x_{i-1}}\right) + \frac{2\Gamma}{\rho}\left(\frac{(x_i - x_{i-1})\phi_{i+1}^n - (x_{i+1} - x_{i-1})\phi_i^n + (x_{i+1} - x_i)\phi_{i-1}^n}{\frac{1}{2}(x_{i+1} - x_{i-1})(x_{i+1} - x_i)(x_i - x_{i-1})}\right) \\ &= \left[\frac{u}{x_{i+1} - x_{i-1}} + \frac{2\Gamma}{\rho}\frac{1}{(x_{i+1} - x_{i-1})(x_i - x_{i-1})}\right]\phi_{i-1}^n - \frac{2\Gamma}{\rho}\frac{1}{(x_{i+1} - x_i)(x_i - x_{i-1})}\phi_i^n + \\ &\quad \left[-\frac{u}{x_{i+1} - x_{i-1}} + \frac{2\Gamma}{\rho}\frac{1}{(x_{i+1} - x_{i-1})(x_i - x_{i-1})}\right]\phi_{i+1}^n \end{aligned} \tag{32}$$

Definiendo

$$
\begin{aligned}
a_1 &= \frac{2\Gamma}{\rho}\frac{1}{(x_{i+1}-x_i)(x_i-x_{i-1})} \\
a_2 &= \frac{2\Gamma}{\rho}\frac{1}{(x_{i+1}-x_{i-1})(x_i-x_{i-1})} \\
a_3 &= \frac{u}{x_{i+1}-x_{i-1}} \\
a_4 &= \frac{2\Gamma}{\rho}\frac{1}{(x_{i+1}-x_{i-1})(x_i-x_{i-1})}
\end{aligned}
$$

Se llega a

$$k_1 = [a_3+a_4]\phi^n_{i-1} - a_1\phi^n_i + [a_2-a_3]\phi^n_{i+1} \tag{33}$$

$$k_1 = [a_3+a_4]\phi^n_{i-1} - a_1\phi^n_i + [a_2-a_3]\phi^n_{i+1} + \frac{\Delta t}{2}k_1 \tag{34}$$

$$\phi^{n+1} = \phi^n + \triangle t(k_1+k_2) \tag{35}$$

Se presenta entonces el bucle principal del desarrollo uniforme y no uniforme para el método anterior

```
do j = 1, tf
  do i = 2,n-1


   k1 = (0.5*(-phi(i+1)+phi(i-1))/dx) + (gamma/rho)*( (phi(i+1)-2.0*phi(i)+phi(i-1) )/(dx*dx) )
   k2 = (0.5*(-phi(i+1)+phi(i-1))/dx) + (gamma/rho)*( (phi(i+1)-2.0*phi(i)+phi(i-1) )/(dx*dx) ) + &
   0.5*dt*k1

   phi(i) = phi(i) + dt*(k2+k1)

   a1 = (2.0*gamma)/( rho*(x(i+1)-x(i))*(x(i)-x(i-1))  )
   a2 = (2.0*gamma)/( rho*(x(i+1)-x(i-1))*(x(i+1)-x(i)) )
   a3 = u/(x(i+1)-x(i-1))
   a4 = (2.0*gamma)/(rho*(x(i+1)-x(i-1))*(x(i)-x(i-1)) )

   l1 = (a3+a4)*phi1(i-1) -a1*phi1(i) + (a2-a3)*phi1(i+1)
   l2 = (a3+a4)*phi1(i-1) -a1*phi1(i) + (a2-a3)*phi1(i+1) + 0.5*dt*k1
   phi1(i) = phi1(i) + dt*(l1+l2)

  end do
```

Se muestra el perfil numérico obtenido de $\phi = \phi(x,t)$

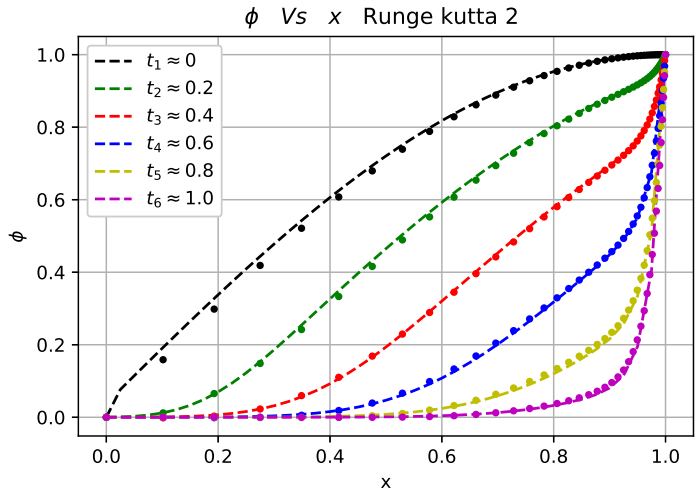

Figura 7. Perfil numérico obtenido mediante Runge-kutta 2 considerando algunos tiempos en el esquema de evolución temporal , además vemos que se identifica la malla no uniforme con la línea punteada y con los puntos el esquema no uniforme

3.2 Análisis de resultados

Para analizar la influencia que tiene el espacio entre dos nodos en el resultado de perfil numérico se hace necesario ver la generación de la malla. Se considera una contracción del espacio entre nodos en cuanto más se avance en el dominio, definimos esta tasa como r, entonces podemos ver que considerando un vector x con n nodos

$$\begin{aligned} x(1) &= 0 \\ x(2) &= h \\ x(3) &= h + rh \\ x(4) &= h + rh + r^2 h \\ x(5) &= h + rh + r^2 h + r^3 h \\ &\vdots \\ x(n) &= h + rh + r^2 h + \cdots + r^{n-2} h = L \\ L &= h \sum_{k=0}^{n-2} \end{aligned} \tag{36}$$

Ahora como se considera una malla que se comprime s tiene que $0 < r < 1$, por tanto una serie geométrica será de gran utilidad , entonces se tiene

$$L = h\frac{1-r^{n-1}}{1-r} \longrightarrow \boxed{h = L\frac{1-r}{1-r^{n-1}}} \tag{37}$$

Entonces basta con definir el tamaño del dominio el número de nodos y la tasa de compresión para poder obtener el dominio de integración discreto, a continuación el código de la implementación

```
h = (l*(1-r))/(1-r**(n-1))
x(1) = 0.0
x(2) = h

do i=2, n-1
 x(i+1) = h + r*x(i)
end do
```

Entonces bajo varios experimentos realizados con los programas construidos en fortran, se evidencia que para el caso de Euler hacia adelante se evidencian dos situaciones particulares. Cuando estudiamos el sistema uniforme se encuentra que se comporta muy bien en la elección siempre y cuando respete la condición de estabilidad dada por

$$\Delta t < \frac{\rho(\delta x)^2}{2\Gamma} \qquad \Delta x < \frac{2\Gamma}{\rho u} \tag{38}$$

La primera condición considera que todos los coeficientes de la ecuación (31) deben ser positivos, la segunda condición considera una relación entre los términos convectivos y difusivos. Entonces tenemos un esquema condicionalmente estable.

El problema radica en la variación del Δx en el esquema no uniforme, es muy sensible cuando se cambia la cantidad de nodos, recordemos que bajo la elección de generación de malla el sistema provee el punto inicial h, la tasa de compresión r, los nodos n y la longitud del dominio L para poder obtener tal vector, sin embargo, cuando se varía la cantidad de nodos es muy fácil que la simulación diverga, por otra parte considerando diferentes tasas de compresión ocurre lo mismo , entonces se puede ver que es una debilidad de este sistema cuando se tienen mallas no uniforme y quizás se deba a la condición inicial del problema, dado que cuando se llega al límite derecho las distancias entre nodos serán bastante pequeñas y las cantidades $x_{i+1} - x_i$ serán muy pequeñas las cuales frecuentemente se encuentran en el denominador de las expresiones, por lo tanto la estabilidad se calcula en función de tales distancias y tendremos que modificar nuestros criterios de convergencia en tales puntos, por tanto se planta la dificultad a la hora de mantener la estabilidad en tales esquemas.

Para el Runge kutta 2 se hizo el mismo estudio encontrando que es un poco flexible en la elección del Δx sin embargo cuando se quiere hacer refinamiento de malla es muy restringida la resolución que queremos encontrar, la ventaja de este método radica en que es más precisa la solución entre el enmallado uniforme y no uniforme , sin embargo el consumo de memoria es más grande que en el caso anterior, entonces tienes ventajas y desventajas a la hora de resolver problemas. En definitiva la elección de la evolución temporal depende de la cantidad que se quiera estudiar y en que condiciones esté de acuerdo a esto se plantea el tiempo de computo que estemos dispuestos a invertir o por el contrario el error que estemos dispuesto a tolerar.

4. USO DE ESQUEMAS DE AVANCE TEMPORAL

Para estudiar los esquemas de avance temporal se considera la simulación se una esfera en dos dimensiones inmersa en una corriente de fluido sujeta a varias consideraciones , la primera de ellas es la acción de una fuerza viscosa de arrastre en la dirección $\hat{x}$ causada por la velocidad relativa entre la partícula y el fluido , además de esto se tiene en cuenta la acción de fuerzas verticales de flotación y gravitacional , asociadas con el volumen y la densidad sw ka esfera, entonces el modelo que estamos estudiando se modela a través de las siguientes ecuaciones

$$\frac{d^2x}{dt^2} = \frac{\rho C_d \pi R^2}{2M}\left(V_x - \frac{dx}{dt}\right)^2 \qquad \frac{d^2y}{dt^2} = \frac{\rho C_d \pi R^2}{2M}\left(V_y - \frac{dy}{dt}\right)^2 + \frac{\rho_{fluido}}{\rho_{esfera}}g - g \tag{39}$$

$$\vec{V} = \frac{A_o}{k_k}\sin(\omega t)\sin(k_x x)\sin(k_y y)\hat{x} + \frac{A_o}{k_k}\sin(\omega t)\sin(k_x x)\sin(k_y y)\hat{y} \tag{40}$$

Ahora se busca dejar sin dimensiones las ecuaciones para poder escribir los códigos de manera correcta, por tanto definimos las siguientes cantidades auxiliares

$$x = Rx, \quad y = Ry, \quad t = \tau t \tag{41}$$

Al considerar tales definiciones las ecuaciones de nuestro modelo resultan escritas de la siguiente manera

$$\frac{d^2x}{dt^2} = \alpha\left(\beta V_x - \frac{dx}{dt}\right)^2 \qquad \frac{d^2y}{dt^2} = \alpha\left(\beta V_y - \frac{dy}{dt}\right)^2 + fe \tag{42}$$

donde

$$\alpha = \frac{\rho C_d \pi R^3}{2M} \qquad \beta = \frac{\tau}{R} \qquad fe = \tau\beta g\left(\frac{\rho_f}{\rho_{es}} - 1\right) \tag{43}$$

Entonces podemos ver el sistema como cuatro ecuaciones diferenciales de primer orden , las cuales utilizaremos para la solución del problema

$$\frac{dx}{dt} = u \qquad \frac{du}{dt} = \alpha\left(\beta V_x - u\right)^2 \tag{44}$$

$$\frac{dy}{dt} = v \qquad \frac{dv}{dt} = \alpha\left(\beta V_y - v\right)^2 + fe \tag{45}$$

A continuación se presentan diferentes esquemas de evolución temporal para resolver el sistema presentando , las condiciones consideradas en este caso fue el agua como fluido y una esfera de hierro de 10 cm de radio.

4.1 Forward - Euler

Consideramos la evolución temporal de Euler hacia adelante en la velocidad

$$u_i^{n+1} = u_i^n + \left(\frac{du}{dt}\right)_i^n \Delta t \tag{46}$$

$$x_i^{n+1} = x_i^n + u_i^n \Delta t \tag{47}$$

Entonces se implementa este método considerando las ecuaciones (44) y (45), obteniéndose el siguiente perfil de trayectoria de la esfera

4.2 Cranck - Nicholson

El método se reduce principalmente en escribir las ecuaciones (44) y (45) de la siguiente manera para que tenga evolución temporal

$$u_i^{n+1} - u_i^n = \frac{\Delta t}{2}(F_i^{n+1})_x + \frac{\Delta t}{2}(F_i^n)_x \qquad v_i^{n+1} - v_i^n = \frac{\Delta t}{2}(F_i^{n+1})_y + \frac{\Delta t}{2}(F_i^n)_y \tag{48}$$

En donde se considera las funciones F_i como

$$(F_i^n)_x = \alpha\left((\beta^2 V_x^2)_i^n - 2(\beta V_x)_i^n u_i^n + (u^2)_i^n\right), \qquad (F_i^n)_y = \alpha\left((\beta^2 V_x^2)_i^{n+1} - 2(\beta V_x)_i^{n+1} u_i^{n+1} + (u^2)_i^{n+1}\right) + fe \quad (49)$$

Por tanto se hace para cada componente , pero se presenta el desarrollo para la componente y , dado que el tratamiento para x es análogo

$$\begin{aligned}
v_i^{n+1} - \frac{\Delta t}{2}(F_i^{n+1})_y &= v_i^n + \frac{\Delta t}{2}(F_i^n)_y \\
v_i^{n+1} - \frac{\Delta t}{2}\left[\alpha\left((\beta^2 V_x^2)_i^{n+1} - 2(\beta V_x)_i^{n+1} u_i^{n+1} + (u^2)_i^{n+1}\right) + fe\right] &= v_i^n + \frac{\Delta t}{2}(F_i^n)_y \\
\left(1 + \alpha\Delta t(\beta V_y)_i^{n+1} - \frac{\alpha\Delta t}{2} v_i^{n+1}\right) v_i^{n+1} - \frac{\Delta t}{2} fe - \frac{\alpha\Delta t}{2}\left(\beta^2 V_y^2\right)_i^{n+1} &= v_i^n + \frac{\Delta t}{2}(F_i^n)_y \\
\underbrace{\left(1 + \alpha\Delta t(\beta V_y)_i^{n+1} - \frac{\alpha\Delta t}{2} v_i^{n+1}\right)}_{\text{Linealización} \quad v_i^{n+1} = v_i^n} v_i^{n+1} &= v_i^n + \frac{\Delta t}{2}(F_i^n)_y + \Delta t fe + \frac{\alpha\Delta t}{2}\left(\beta^2 V_y^2\right)_i^{n+1} \\
v_i^{n+1} = \frac{v_i^n + \frac{\Delta t}{2}(F_i^n)_y + \Delta t fe + \frac{\alpha\Delta t}{2}\left(\beta^2 V_y^2\right)_i^{n+1}}{1 + \alpha\Delta t(\beta V_y)_i^{n+1} - \frac{\alpha\Delta t}{2} v_i^n} &
\end{aligned} \quad (50)$$

Definamos

$$\alpha_1 = 1 + \alpha\Delta t(\beta V_y)_i^{n+1} - \frac{\alpha\Delta t}{2} v_i^n \qquad \gamma_1 = 1 + \alpha\Delta t(\beta V_x)_i^{n+1} - \frac{\alpha\Delta t}{2} u_i^n \quad (51)$$

$$\alpha_2 = \frac{\alpha\Delta t}{2}\left(\beta^2 V_y^2\right)_i^{n+1} \qquad \gamma_2 = \frac{\alpha\Delta t}{2}\left(\beta^2 V_x^2\right)_i^{n+1} \quad (52)$$

Entonces se obtiene la ecuación de evolución del sistema dada por

$$\boxed{v_i^{n+1} = \frac{v_i^n + \frac{\Delta t}{2}(F_i^n)_y + \Delta t fe + \alpha_2}{\alpha_1} \qquad u_i^{n+1} = \frac{u_i^n + \frac{\Delta t}{2}(F_i^n)_x + \gamma_2}{\gamma_1}} \quad (53)$$

Finalmente para encontrar las componentes de las coordenadas se se hace

$$x_i^{n+1} = x_i^n + \Delta t(u_i^{n+1}) \quad (54)$$

$$y_i^{n+1} = y_i^n + \Delta t(v_i^{n+1}) \quad (55)$$

Se muestran las trayectorias para unas condiciones iguales a las consideradas en el método cranck-Nicholson

4.3 Runge-Kutta 4

Al igual que los anteriores métodos consideramos las ecuaciones (44) y (45) entonces se desarrolla la componente y , sabiendo que para la otra componente es un procedimiento análogo. En primer lugar consideramos la derivada de la posición de la siguiente manera

$$\frac{dy}{dt} = v \qquad \frac{dv}{dt} = \Delta t f(y,v,t)$$

$$\begin{aligned} k1 &= \Delta t(v) & l_1 &= \Delta t f(y,v,t) \\ k2 &= \Delta t\left(v + \frac{l_1}{2}\right) & l_2 &= \Delta t f\left(y + \frac{k_1}{2}, v + \frac{l_1}{2}, t + \frac{\Delta t}{2}\right) \\ k3 &= \Delta t\left(v + \frac{l_1}{2}\right) & l_3 &= \Delta t f\left(y + \frac{k_2}{2}, v + \frac{l_2}{2}, t + \frac{\Delta t}{2}\right) \\ k4 &= \Delta t\left(v + l_3\right) & l_4 &= \Delta t f(y + k_3, v + l_3, t + \Delta t) \\ x_i^{n+1} &= x_i^n + \frac{1}{6}(k1 + 2k_2 + 2k_3 + k_4) & v_i^{n+1} &= v_i^n + \frac{1}{6}(l_1 + 2l_2 + 2l_3 + l_4) \end{aligned} \tag{56}$$

Teniendo que

$$f(x,u,t) = \alpha\left(\beta V_x - u\right)^2 \qquad f(y,v,t) = \alpha\left(\beta V_y - v\right)^2 + fe \tag{57}$$

Entonces para cada componente se tiene que realizar este análisis, dado que las ecuaciones diferenciales están acopladas. Ahora se presenta una caso particular en el cual se hace la comparación por los tres métodos de avance temporal y eligiendo cuatro tiempos característicos

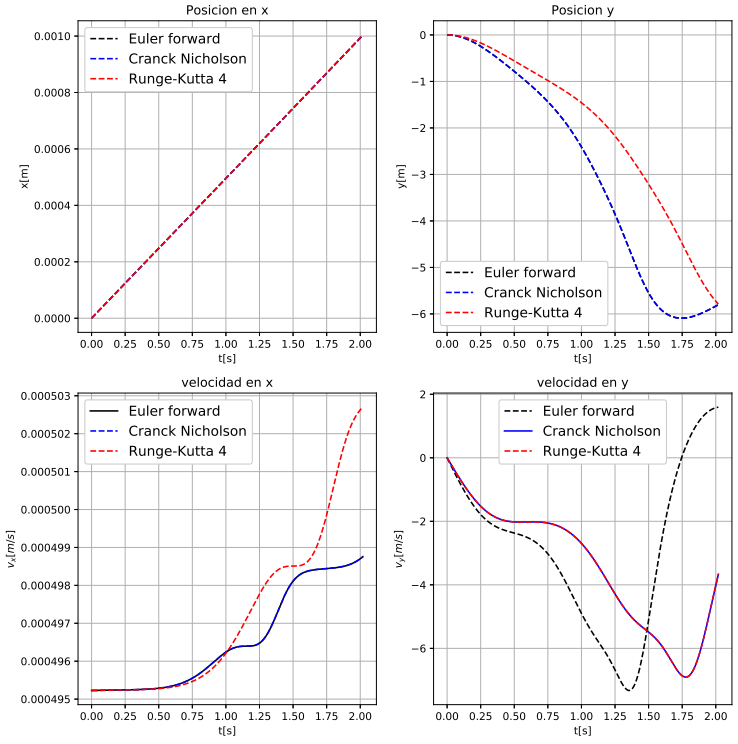

Figura 8. calculo de posiciones y velocidades en el primer tiempo característico elegido τ_1

En este test se quiere simular una esfera de metal con densidad $\rho_{es} \approx 7874 kg/m^3$ que esta totalmente sumergida dentro agua $\rho_f = 997 kg/m^3$ con un coeficiente de arrastre $C_d = 0.47$ que es el correspondiente al de una esfera

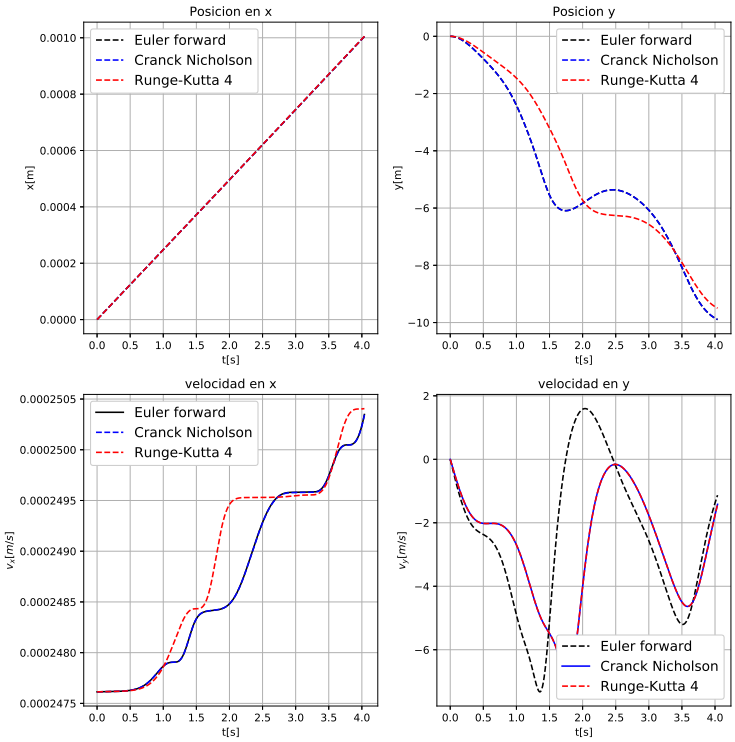

Figura 9. calculo de posiciones y velocidades en el segundo tiempo característico elegido τ_2

Dada la elección de tales parámetros, el valor sin dimensión que caracteriza el fluido, está dado por el siguiente valor. utilizando la definición (43) se tiene que

$$\alpha = 2.23 \times 10^{-2} \tag{58}$$

La elección del tiempo característico se hizo de la siguiente manera, se tomo como longitud de referencia el radio de la esfera, dado que es una cantidad representativa del sistema a modelar, luego de esto se tuvo en cuenta que al fluido en la componente vertical actúan fuerzas externas y estas están almacenadas dentro de la variable fe , dada por

$$fe = \frac{\tau^2 g}{R}\left(\frac{\rho_f}{\rho_{es}} - 1\right) \tag{59}$$

La elección del tiempo característico se puede hacer de varias maneras, en esta ocasión se elige como tiempo característico una relación entre la gravedad y el radio de la esfera, teniendo en cuenta la anterior relación

$$\tau = \sqrt{\frac{R}{g}} \tag{60}$$

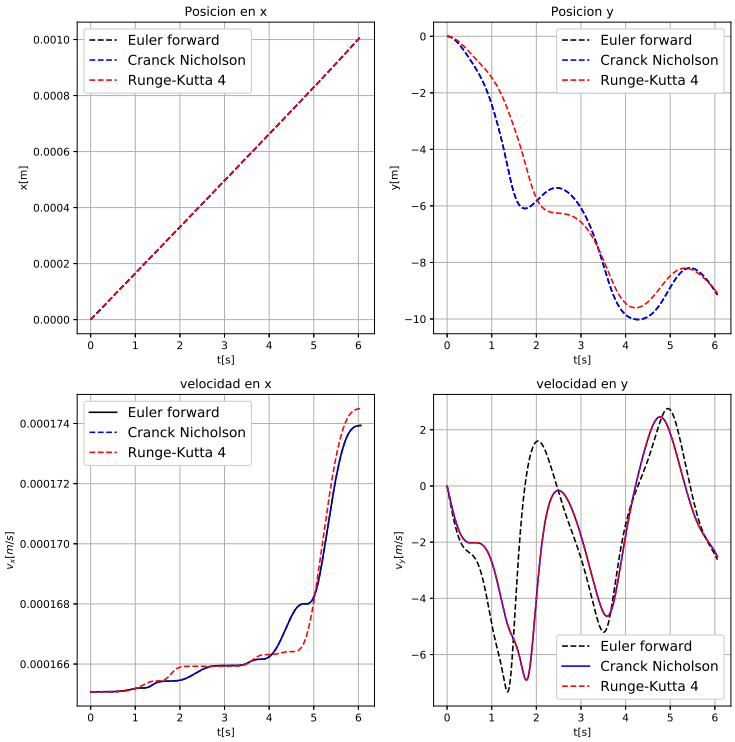

Figura 10. calculo de posiciones y velocidades en el tercer tiempo característico elegido τ_3

Al momento de realizar las simulaciones y hacer varios experimentos se encontró que los tiempos característicos que dejan ver de la evolución de mejor manera para el sistema son los correspondientes a $\tau_1 = 20\tau$, $\tau_1 = 40\tau$, $\tau_1 = 60\tau$, $\tau_1 = 80\tau$, los cuales se muestran en las figuras 8,9,10 y 11.

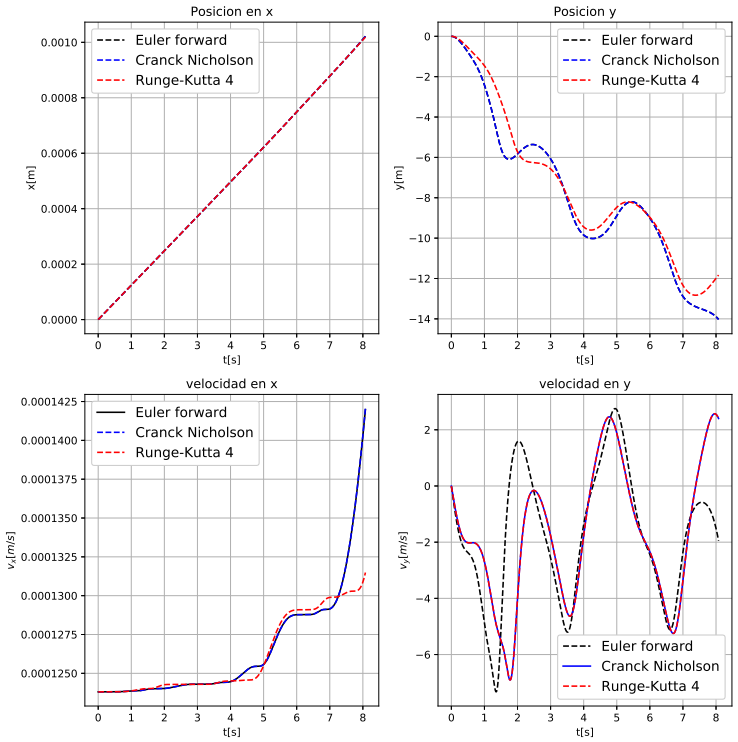

Figura 11. calculo de posiciones y velocidades en el cuarto tiempo característico elegido τ_4

Ahora se presenta para el tiempo característico de τ_4 la trayectoria que sigue la esfera de hierro con cada uno de los métodos considerados al principio de está sección. Se muestra que el Euler hacia adelante coincide casi perfecto con el Cranck-Nicholson para esta elección de parámetros , esto se puede asegurar por que se realizaron varias pruebas con diferentes valores y se tenían diferencias importantes entre los tres métodos, al parecer el programa es susceptible de las condiciones iniciales adecuadas para que no existan diferencias grandes entre los modelos considerados.

Otra cosa importante para mencionar es la consideración si el fluido es o no incompresible. Podemos decir que el fluido es incompresible, dado que la densidad es constante a través de toda la malla de simulación, esto quiere decir que se respeta la condición de $\nabla \cdot \vec{v} = 0$, en ningún momento se tuvo que actualizar en los nodos la densidad para resolver el sistema de ecuaciones planteado en la forma discreta de las ecuaciones.

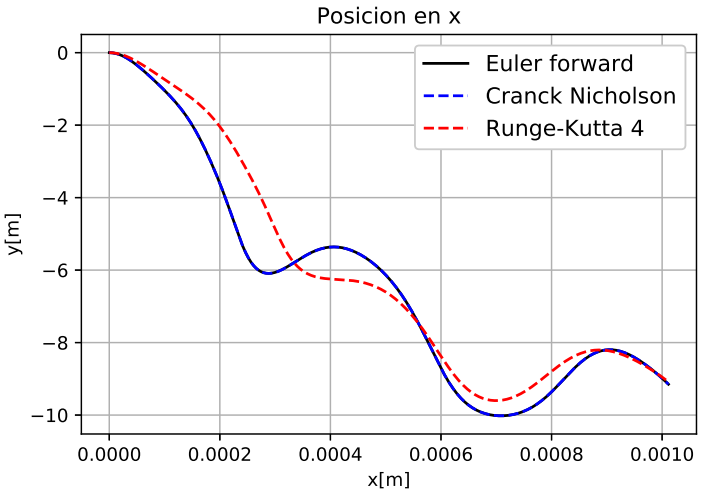

Figura 12. Comparación de las trayectorias utilizando tres métodos distintos de simulación

APPENDIX A. APÉNDICE

A.1 Códigos para el estudio de exatitud en las derivadas de una fución

A.1.1 Primera derivada diferencias finitas

```
program compacto
implicit none

integer, parameter :: dp=selected_real_kind(15, 307)

integer :: n,i
real(kind=dp),allocatable:: y(:), dydx(:), dy2dx(:)
real(kind=dp) :: x, h, error
real(kind=dp), parameter :: pi=4.0D0*atan(1.0D0)
integer, parameter :: xmin=90.0d0
double precision :: k0,k1,k2, x1

open(1, file = "c1.dat")

 k0=1.1250d0
 k1=4.D0*DATAN(1.D0)/(8.0*xmin)
 k2=(3.5d0*k1)/4.0

do n=xmin, 10000, 20

allocate (y(n), dydx(n))

h= (2.0d0*pi)/(n-1.0d0)

do i=1,n
   x= -pi + h*(DBLE(i)- 1.0d0)
   y(i)=exp(-k0*(x**2))*(1.1d0*COS(k1*x)-1.3d0*SIN(k2*x))
end do

call dpadex (y, n, h, dydx)


do i=2, n
if (i>20 .AND. i <(n-20)) then
x= -pi + h*(DBLE(i)- 1.0d0)
error= error + (((dydx(i) - exactder(x))**2)*h)
end if

end do
write(1, *) h, SQRT(error)/n

deallocate(y, dydx)

end do

close(1)
contains
```

```
subroutine dpadex(f, nx, h, s)
real(kind=dp), intent(in):: f(nx), h
real(kind=dp),intent(out) :: s(nx)
real(kind=dp) :: hx(nx)
real(kind=dp) :: b0, c0, d(nx)
integer :: i, nx

b0= 14.0/9.0
c0= 1.0/9.0

do i=1,nx
   hx(i)= pi
end do

d(1)= 0.0
d(2)= 0.5* b0*(f(3)-f(1))/h + 0.25*c0*(f(4)-f(2))/h

do i=3, nx-2
d(i)= 0.5* b0*(f(i+1)-f(i-1))/h + 0.25*c0*(f(i+2)-f(i-2))/h
end do
d(nx)=d(1)
d(nx-1)=d(2)

call thomas(nx,d,s)

do i=1, nx
s(i)= s(i)/hx(i)
end do
return
end

double precision function exactder(xp)
  double precision, INTENT(IN) :: xp
  double precision, parameter :: pi=4.D0*DATAN(1.D0)
  double precision, parameter :: xmin=90.0d0
  double precision :: k0,k1,k2
    k0=1.1250d0
    k1=pi/(8.0*xmin)
    k2=(3.5d0*k1)/4.0

    exactder=exp(-k0*(xp**2))*(-2.2d0*k0*xp*COS(k1*xp)-1.1d0*k1*SIN(k1*xp)+2.6d0*k0*xp*SIN(k2*xp)-1.3d0*k
end function exactder


subroutine thomas(n,d,s)

real(kind=dp), INTENT(IN):: d(n)
real(kind=dp) :: s(n)
real(kind=dp) :: alpha(n), beta(n), y(n), a(n), b(n), c(n)
integer :: i, n

do i=1, n
```

```
b(i)= 1.0
a(i)= 1.0/3.0
c(i)= 1.0/3.0
end do

do i=1, n
if (n .EQ. 1) then
beta(1)= c(1)/b(1)
y(1)= 0.0
return
end if
alpha(i) = b(i) - a(i) * beta(i-1)
beta(i) = c(i)/ alpha(i)
y(i) = (d(i) - a(i)*y(i-1))/alpha(i)
end do
do i=1, n-1
s(i) = y(i) - beta(i)*s(i+1)
end do
s(n)= y(n)
return
end


end program compacto
```

A.1.2 Segunda derivada diferencias finitas

```
program compacto2
implicit none

integer, parameter :: dp=selected_real_kind(15, 307)

integer :: n,i
real(kind=dp),allocatable:: y(:), dy2dx(:)
real(kind=dp) :: x, h, error
real(kind=dp), parameter :: pi=4.0D0*atan(1.0D0)
integer, parameter :: xmin=80.0d0
double precision :: k0,k1,k2

open(1,file = "c2.dat")

 k0=1.1250d0
 k1=4.D0*DATAN(1.D0)/(8.0*xmin)
 k2=(3.5d0*k1)/4.0

do n=xmin, 100000, 20

allocate (y(n), dy2dx(n))

h= (2.0d0*pi)/(n-1.0d0)

do i=1,n
   x= -pi + h*(DBLE(i)- 1.0d0)
```

```
   y(i)=exp(-k0*(x**2))*(1.1d0*COS(k1*x)-1.3d0*SIN(k2*x))
end do

call d2padx (y, n, h, dy2dx)

do i=1, n
if (i>20 .AND. i <(n-20)) then
x= -pi + h*(DBLE(i)- 1.0d0)
error= error + (((dy2dx(i) - exact2der(x))**2)*h)
end if

end do
write(1,*) h, SQRT(error)/n

deallocate(y, dy2dx)

end do
close(1)
contains

subroutine d2padx(f, nx, h, s)
real(kind=dp), intent(in):: f(nx), h
real(kind=dp),intent(out) :: s(nx)
!real(kind=dp) :: x(nx), hx(nx), h2x(nx)
real(kind=dp) :: b0, c0, d(nx), m
integer :: i, nx

b0= 12.0/11.0
c0= 3.0/11.0
m= h**2

d(1)= b0*(2*f(2)-2*f(1))/m + 0.25*c0*(2*f(3)-2*f(1))/m
d(2)= b0*(f(3)-2*f(2)+f(1))/m + 0.25*c0*(f(4)-2*f(2)+f(2))/m

do i=3, nx-2
d(i)= b0*(f(i+1)-2*f(i)+f(i-1))/m + 0.25*c0*(f(i+2)-2*f(i)+f(i-2))/m
end do
d(nx)=d(1)
d(nx-1)=d(2)

call thomas(nx,d,s)

do i=1, nx
s(i)= s(i)/(2*pi)
end do
return
end

real(kind=dp) function exact2der(xp)
  double precision, INTENT(IN) :: xp
  double precision, parameter :: pi=4.D0*DATAN(1.D0)
  double precision, parameter :: xmin=80.0d0
```

```
  double precision :: k0,k1,k2
    k0=1.1250d0
    k1=pi/(8.0*xmin)
    k2=(3.5d0*k1)/4.0

    exact2der= 4.4*exp(-k0*(xp**2))*(((k0**2)*(xp**2)*COS(k1*xp)) -1.18182*(k0**2)*(xp**2)*SIN(k2*xp) &
+(k0*k1*xp)*SIN(k1*xp)-0.5*k0*COS(k1*xp)+0.590909*k0*SIN(k2*xp)+1.18182*k0*k2*xp*COS(k2*xp)-0.25*(k1**2)*
+0.295455*(k2**2)*SIN(k2*xp))
end function exact2der

subroutine thomas(n,d,s)

real(kind=dp), INTENT(IN):: d(n)
real(kind=dp) :: s(n)
real(kind=dp) :: alpha(n), beta(n), y(n), a(n), b(n), c(n)
integer :: i, n

do i=1, n
b(i)= 1.0
a(i)= 2.0/11.0
c(i)= 2.0/11.0
end do

do i=1, n
if (n .EQ. 1) then
beta(1)= c(1)/b(1)
y(1)= d(1)/b(1)
return
end if
alpha(i) = b(i) - a(i) * beta(i-1)
beta(i) = c(i)/ alpha(i)
y(i) = (d(i) - a(i)*y(i-1))/alpha(i)
end do
do i=1, n-1
s(i) = y(i) - beta(i)*s(i+1)
end do
s(n)= y(n)

return
end


end program compacto2
```

A.2 Código en fortran para el Desarrollo de la ecuación de difusión convección 1D estacionaria

```
program main
 implicit none

 integer, parameter :: dp = selected_real_kind(15)
 integer :: i,n,j,k,e,nodos
 real(dp), parameter :: l=1.0,rho=1.0,u=1.0,gamma=0.01667
```

```
real(dp) :: dx,beta,b,Pe,r1,r2
real(dp), allocatable :: phi(:),x(:),phi1(:)

dx = 1.0/41.0
n = l/dx
nodos = 11.0
allocate(phi(n),phi1(nodos),x(nodos))
phi(1) = 0.0_dp
phi1(1) = 0.0_dp
x(1) = 0.0_dp
x(2) = 0.31_dp
Pe = rho*u*l/gamma
b = 2*gamma/(rho*u*l)
beta =b/dx

print*, "Upwind ----------------------------------->1"
print*, "Diferencias centradas no uniformes -------->2"
print*, "Diferencias centradas uniformes ----------->3"
read*, e

select case(e)

case(1)

 open(1, file  = "DC_UDS.dat")
 do j = 1,1000000
  do i = 2,n-1
   phi(i) = (2.0_dp/(dx*Pe+4.0_dp))*phi(i+1) + ((dx*Pe+2.0_dp)/(dx*Pe+4.0_dp))*phi(i-1)
   end do
  phi(n) = 1.0_dp
 end do
 write(*,*) phi
 write(1,*) phi

case(2)
 open(2, file = "CDS_NU.dat")
 do j = 1,6
  do i = 2,nodos-1
   x(i+1) = 0.7_dp*(x(i)-x(i-1)) + x(i)
   r1 = x(i+1)-x(i) ; r2 = x(i)-x(i-1)
   phi1(i) = (r1*r2/(r1+r2))*((1.0_dp/r1 - 0.5_dp*Pe)*phi1(i+1) + (1.0_dp/r2 + 0.5_dp*Pe)*phi1(i-1))
  end do
 phi1(1) = 0.0_dp
 phi1(nodos) = 1.0_dp
 end do
 write(*,*) phi1!x(1)
 write(2,*) x
 write(2,*) phi1

case(3)
 open(1, file  = "DC_CDS.dat")
 do j = 1,1000000
```

```
   do i = 2,n-1
    phi(i) = ( (beta+1.0_dp)*phi(i-1)+(beta-1.0_dp)*phi(i+1) )/(2.0_dp*beta)
    end do
   phi(n) = 1.0_dp
  end do
  write(*,*) phi
  write(1,*) phi

 end select
end program main
```

A.3 Código en fortran para el Desarrollo de la ecuación de difusión convección 1D transitoria

En este apéndice se presetan dos códigos , en primer lugar uno correspondiente al Euler hacia adelante en donde se desarrollan dos esquemas, uno uniforme y uno no uniforme. En segundo lugar se muestra pero para un Runge Kutta 2 .

A.3.1 Euler hacia adelante

```
program main
 implicit none

 integer, parameter :: dp = selected_real_kind(15)
 integer :: i,n,j,k,tf
 real(dp), parameter :: l=1.0,rho=1.0,u=1.0,gamma=0.025
 real(dp), parameter :: pi = 4.0_dp*atan(1.0_dp)
 real(dp) :: dx,dt,d,c,t,r,h
 real(dp) :: a1,a2,a3,a4
 real(dp), allocatable :: phi(:),x(:),phi2(:)

 t = 0.0;tf = 1000000
 dx = 1/41.0!0.001
 n = int(l/dx); r = 0.99
 allocate(phi(n),phi1(n),x(n),phi2(n))

 h = (l*(1-r))/(1-r**(n-1))
 x(1) = 0.0
 x(2) = h

 do i=2, n-1
  x(i+1) = h + r*x(i)
 enddo

 do k = 1,n
  phi(k) = 1.0*sin(k*0.5*pi/n)
  phi2(k) = 1.0*sin(x(k)*0.5*pi)
 end do
 phi(1) = 0.0
 phi1(1) = 0.0

 dt = 0.000001
 d = gamma*dt/(rho*dx*dx)
 c = u*dt/dx
```

```
open(1,file = "DCT_CDS.dat")
open(2,file = "DCT_CDS_t.dat")
open(4,file = "DCT_CDS_NU.dat")
open(5,file = "malla_CDS.dat")

if (d < 0.5 .and. c < 2*d) then
do j = 1, tf
 do i = 2,n-1
  a1 = (2.0*gamma*dt)/( rho*(x(i+1)-x(i))*(x(i)-x(i-1))  )
  a2 = (2.0*gamma*dt)/( rho*(x(i+1)-x(i-1))*(x(i+1)-x(i)) )
  a3 = u*dt/(x(i+1)-x(i-1))
  a4 = (2.0*gamma*dt)/(rho*(x(i+1)-x(i-1))*(x(i)-x(i-1)) )
  phi2(i) = (1-a1)*phi(i) + (a2-a3)*phi(i+1) + (a4+a3)*phi(i-1)

  phi(i)= (1.0-2.0*d)*phi(i)+(d-0.5*c)*phi(i+1)+(d+0.5*c)*phi(i-1)
 end do

 phi(n)= 1.0
 phi2(n) = 1.0


 if (j == 1) then
  write(1,*) phi
  write(2,*) t
  write(4,*) phi2
 endif
 if (j == 1*tf/5) then
  write(1,*) phi
  write(2,*) t
  write(4,*) phi2
 endif
 if (j == 2*tf/5 ) then
  write(1,*) phi
  write(2,*) t
  write(4,*) phi2
 endif
 if (j == 3*tf/5 ) then
  write(1,*) phi
  write(2,*) t
  write(4,*) phi2
 endif
 if (j == 4*tf/5) then
  write(1,*) phi
  write(2,*) t
  write(4,*) phi2
 endif
 if (j == tf ) then
  write(1,*) phi
  write(2,*) t
  write(4,*) phi2
 endif
```

```
 end do

 write(*,*) "Done Euler!"
 else
  print*, "Para que sea estable el programa se debe asegurar"
  print*, "valores adecuados para dt y dx "
  print*, "d = ",d,"c = ", c
 endif

 write(5,*) x
 close(1)
 close(2)
 close(4)
 close(5)
 deallocate(phi,phi1,phi2)

end program main
```

A.3.2 Runge Kutta 2

```
program main
 implicit none
 integer, parameter :: dp = selected_real_kind(15)

 integer :: i,n,j,k,tf
 real(dp), parameter :: l=1.0,rho=1.0,u=1.0,gamma=0.025
 real(dp), parameter :: pi = 4.0_dp*atan(1.0_dp)
 real(dp) :: dx,dt,t,k1,k2,a1,a2,a3,a4,l1,l2,r,Pe,h
 real(dp), allocatable :: phi(:),x(:),phi1(:)
 character(6) :: name

 !read*, n,name #
 !Nombra archivos para cuando se utilzia varios nodos
 t = 0.0;tf = 50000
 dx = 1/41.0!0.001
 n = int(l/dx)
 Pe = rho*u/gamma
 r=0.9
 allocate(phi(n),x(n),phi1(n))


 h = (l*(1-r))/(1-r**(n-1))
 x(1) = 0.0
 x(2) = h
 do i=2, n-1
  x(i+1) = h + r*x(i)
 enddo


 do k = 1,n
  phi(k) = 1.0*sin(k*0.5*pi/n)
  phi1(k) = 1.0*sin(x(k)*0.5*pi)
 end do
```

```
phi(1) =0.0
phi1(1) = 0.0
dt = 0.00001

open(2,file = "DCT_RK2_t_.dat")
open(1,file = "DCT_RK2_.dat")
open(4,file = "DCT_RK2_NU_.dat")
open(3,file = "malla_RK2_.dat")
!open(2,file = "DCT_RK2_t_"//name//".dat")
!open(1,file = "DCT_RK2_"//name//".dat")
!open(4,file = "DCT_RK2_NU_"//name//".dat")
!open(3,file = "malla_RK2_"//name//".dat")

do j = 1, tf
 do i = 2,n-1


  k1 = (0.5*(-phi(i+1)+phi(i-1))/dx) + (1.0/Pe)*( (phi(i+1)-2.0*phi(i)+phi(i-1) )/(dx*dx) )
  k2 = (0.5*(-phi(i+1)+phi(i-1))/dx) + (1.0/Pe)*( (phi(i+1)-2.0*phi(i)+phi(i-1) )/(dx*dx) ) + &
  0.5*dt*k1
  phi(i) = phi(i) + dt*(k2+k1)

  a1 = (2.0*gamma)/( rho*(x(i+1)-x(i))*(x(i)-x(i-1))  )
  a2 = (2.0*gamma)/( rho*(x(i+1)-x(i-1))*(x(i+1)-x(i)) )
  a3 = u/(x(i+1)-x(i-1))
  a4 = (2.0*gamma)/(rho*(x(i+1)-x(i-1))*(x(i)-x(i-1)) )

  l1 = (a3+a4)*phi1(i-1) -a1*phi1(i) + (a2-a3)*phi1(i+1)
  l2 = (a3+a4)*phi1(i-1) -a1*phi1(i) + (a2-a3)*phi1(i+1) + 0.5*dt*l1
  phi1(i) = phi1(i) + dt*(l1+l2)

 end do

 phi(n) = 1.0
 phi1(n) = 1.0

 if (j == 1) then
  write(1,*) phi
  write(2,*) t
  write(4,*) phi1
 endif
 if (j == 1*tf/5) then
  write(1,*) phi
  write(2,*) t
  write(4,*) phi1
 endif
 if (j == 2*tf/5 ) then
  write(1,*) phi
  write(2,*) t
  write(4,*) phi1
 endif
 if (j == 3*tf/5 ) then
```

```
   write(1,*) phi
   write(2,*) t
   write(4,*) phi1
  endif
  if (j == 4*tf/5) then
   write(1,*) phi
   write(2,*) t
   write(4,*) phi1
  endif
  if (j == tf ) then
   write(1,*) phi
   write(2,*) t
   write(4,*) phi1
  endif
 end do
 write(3,*) x
 write(*,*) "Done RK2!"
 close(1)
 close(2)
 close(3)
 close(4)
 deallocate(phi)

end program main
```

A.4 Código en fortran para los esquemas de avance temporal

```
program FEesfera
 implicit none

 integer, parameter :: dp = selected_real_kind(15)
 real(dp), parameter :: pi = 4.0_dp*atan(1.0_dp)
 integer :: i, iteraciones
 real(dp) :: Ao,kx,ky,w,alpha,dt,g,beta,R,tau,M,C,rho,rhoe,fe
 real(dp) :: t,dxdt,dudt,dydt,dvdt
 real(dp) :: xFE,yFE,xCN,yCN,xRK,yRK,uFE, vFE,uCN,vCN,uRK,vRK
 real(dp) :: unew,vnew,xnew,ynew
 real(dp) :: alphau1,alphau2,alphav1,alphav2
 real(dp) :: kx1,kx2,kx3,kx4,lx1,lx2,lx3,lx4
 real(dp) :: ky1,ky2,ky3,ky4,ly1,ly2,ly3,ly4

 iteraciones = 10000

 uFE = 0.01; vFE = 0.01
 xFE = 0.0; yFE = 0.0

 uCN = 0.01; vCN = 0.01
 xCN = 0.0; yCN = 0.0

 uRK = 0.01; vRK = 0.01
 xRK = 0.0; yRK = 0.0

 dt = 0.0001; t = 0.0
```

```
Ao = 10.0; w = 1.0
kx = 1.0; ky = 1.0

rho = 997.0 !kg/m3
rhoe = 7874.0 !kg/m3
C = 0.47
R = 0.1     !m
M = 4*pi*R**3*rhoe/3.0 ! kg
g = 9.81

tau = 80*sqrt(R/g)!s
alpha = (rho*C*pi*R**3)/(2*M)
beta = tau/R !s/m
fe = tau*beta*g*((rho/rhoe)-1)

print*, alpha

open(1,file = "FEesfera.dat")
open(2,file = "posiciones")
open(3,file = "velocidades")
open(4,file = "Reporte general")



do i = 1,iteraciones


 !-----------------forward euler ---------------------------------
 dxdt = uFE
 dydt = vFE
 dudt = fx(xFE,yFE,uFE,vFE,t)
 dvdt = fy(xFE,yFE,uFE,vFE,t)
 uFE = uFE + dt*dudt
 vFE = vFE + dt*dvdt
 xFE = xFE + uFE*dt
 yFE = yFE + vFE*dt
 !----------------- termina forward euler ---------------------



 !-----------------Cranck Nicholson ---------------------------------
 alphau1 = 1.0 + alpha*dt*beta*vx(xCN,yCN,t+dt)-alpha*dt*0.5*uCN
 alphau2 = alpha*dt*0.5*beta*beta*vx(xCN,yCN,t+dt)*vx(xCN,yCN,t+dt)
 alphav1 = 1.0 + alpha*dt*beta*vy(xCN,yCN,t+dt)-alpha*dt*0.5*vCN
 alphav2 = alpha*dt*0.5*beta*beta*vy(xCN,yCN,t+dt)*vy(xCN,yCN,t+dt)
 unew = (uCN + 0.5*dt*alpha*(beta*beta*vx(xCN,yCN,t)*vx(xCN,yCN,t)-2.0*beta*vx(xCN,yCN,t)*uCN + uCN*uCN)
 vnew = (vCN + 0.5*dt*alpha*(beta*beta*vy(xCN,yCN,t)*vy(xCN,yCN,t)-2.0*beta*vy(xCN,yCN,t)*vCN + vCN*vCN)

 xnew = xCN + dt*(unew)
 ynew = yCN + dt*(vnew)
```

```
uCN = unew
vCN = vnew
xCN = xnew
yCN = ynew

!-----------------termina Cranck Nicholson --------------------------------



!-----------------Runge kutta 4 --------------------------------
kx1 = dt*uRK
lx1 = dt*fx(xRK,yRK,uRK,vRK,t)

ky1 = dt*vRK
ly1 = dt*fy(xRK,yRK,uRK,vRK,t)

kx2 = dt*(uRK + 0.5*lx1)
lx2 = dt*fx(xRK+0.5*kx1,yRK+0.5*ky1,uRK+0.5*lx1,vRK+0.5*ly1,t+0.5*dt)

ky2 = dt*(vRK + 0.5*ly1)
ly2 = dt*fy(xRK+0.5*kx1,yRK+0.5*ky1,uRK+0.5*lx1,vRK+0.5*ly1,t+0.5*dt)

kx3 = dt*(uRK + 0.5*lx2)
lx3 = dt*fx(xRK+0.5*kx2,yRK+0.5*ky2,uRK+0.5*lx2,vRK+0.5*ly2,t+0.5*dt)

ky3 = dt*(vRK + 0.5*ly2)
ly3 = dt*fy(xRK+0.5*kx2,yRK+0.5*ky2,uRK+0.5*lx2,vRK+0.5*ly2,t+0.5*dt)

kx4 = dt*(uRK + lx3)
lx4 = dt*fx(xRK+kx3,yRK+ky3,uRK+lx3,vRK+ly3,t+dt)

xRK = xRK+ (1.0/6.0)*(kx1 + 2.0*kx2 + 2.0*kx3 + kx4)
yRK = yRK + (1.0/6.0)*(ky1 + 2.0*ky2 + 2.0*ky3 + ky4)
uRK = uRK + (1.0/6.0)*(lx1 + 2.0*lx2 + 2.0*lx3 + lx4)
vRK = vRK + (1.0/6.0)*(ly1 + 2.0*ly2 + 2.0*ly3 + ly4)

t = t + dt

write(*,*) error(R*xRK,R*xFE),error(R*xRK,R*xCN)

write(1,*) t*tau," ",R*xFE," ",R*yFE," ",R*xCN," ",R*yCN," ",R*xRK," ", &
           R*yRK," ",uFE/beta," ",vFE/beta," ",uCN/beta," ",vRK/beta," ", &
           uRK/beta," ",vRK/beta, error(xRK,xFE),error(xRK,xCN),&
           error(yRK,yFE),error(yRK,yCN),error(uRK,uFE),error(uRK,uCN),&
           error(vRK,vFE),error(vRK,vCN)
write(2,*) R*xFE," ",R*yFE,R*xCN," ",R*yCN," ",R*xRK," ",R*yRK
write(3,*) uFE/beta," ",vFE/beta," ",uCN/beta," ",vRK/beta,uRK/beta," ",vRK/beta
write(4,*) tau," ", A," ",w," ", kx," ",ky

end do

close(1)
```

```
 close(2)
 close(3)
 close(4)

contains
real function fx(x,y,u,v,t)
 implicit none
 real(dp) , intent(in) :: x,y,u,v,t
 fx = alpha*(beta*vx(x,y,t)-u)**2
end function fx

real function fy(x,y,u,v,t)
 implicit none
 real(dp) , intent(in) :: x,y,u,v,t
 fy = alpha*(beta*vy(x,y,t)-v)**2 + fe
end function fy

real function gx(x,y,u,v,t)
 implicit none
 real(dp) , intent(in) :: x,y,u,v,t
 gx = u
end function gx

real function gy(x,y,u,v,t)
 implicit none
 real(dp) , intent(in) :: x,y,v,u,t
 gy = v
end function gy

real function vx(x,y,t)
 implicit none
 real(dp) , intent(in) :: x,y,t
 vx = (Ao/kx)*sin(w*t*tau)*sin(kx*x*R)*sin(ky*y*R)
end function vx

real function vy(x,y,t)
 implicit none
 real(dp) , intent(in) :: x,y,t
 vy = (Ao/kx)*sin(w*t*tau)*cos(kx*x*R)*cos(ky*y*R)
end function vy

real function error(xref,x)
 implicit none
 real(dp) , intent(in) :: xref,x
 error = abs((xref-x)/xref)
end function

end program FEesfera
```

REFERENCES

[1] s. Lele, "Compact finite difference schemes with spectral-like resolution," *J.of Computational physics, 103(1)* **16-42.**

[2] Ferziger, J. H. and Peric, M., [*Computational Methods for fluid Dynamics*], Springer Berlin Heildelberg, Berlin (2001).