UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

ESCUELA DE CIENCIAS FÍSICAS Y MATEMÁTICAS

Informe final de prácticas

Evaluación de distintos métodos numéricos para resolver ecuaciones particulares de la mecánica de fluidos

POR

Rodrigo Rafael Castillo Chong
201804566

ASESORADO POR

Enrique Pazos, Ph.D.

12 de marzo de 2023

Índice

1. Método de diferencias finitas

El **método de diferencias finitas** o **método DF** es un método numérico que sirve para resolver ecuaciones diferenciales ordinarias o parciales. El método consiste en discretizar el dominio de la ecuación en un conjunto finito de puntos llamado **grilla**; donde cada punto debe estar a la misma distancia de cada uno de sus vecinos. Posteriormente se deben aproximar las derivadas de la función con ecuaciones de diferencias utilizando series de potencias, para poder resolver la ecuación diferencial algebraica e iterativamente [1].

Un ejemplo de discretización, que se usa al aplicar el método DF en una ecuación diferencial parcial, es el siguiente: la segunda derivada parcial respecto a x de una función $u = u(x,t)$ valuada en (x,t) se puede aproximar expandiendo la función en dos series de Taylor centradas en dos diferentes valores sobre el eje x, que corresponden a los dos puntos vecinos a x, separándose de este punto por una distancia Δx; también llamada **tamaño de paso** en x.

Para obtener la aproximación de la segunda derivada de u respecto a x usamos las expansiones en series de Taylor:

$$u(x+\Delta x,t) \approx u(x,t) + \Delta x\frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2}\frac{\partial^2 u}{\partial x^2} \tag{1}$$

$$u(x-\Delta x,t) \approx u(x,t) - \Delta x\frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2}\frac{\partial^2 u}{\partial x^2} \tag{2}$$

Sumando ambas aproximaciones se obtiene:

$$(\Delta x)^2\frac{\partial^2 u}{\partial x^2} \approx u(x+\Delta x,t) + u(x-\Delta x,t) - 2u(x,t) \tag{3}$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u(x+\Delta x,t) + u(x-\Delta x,t) - 2u(x,t)}{(\Delta x)^2} \tag{4}$$

Por tanto, podemos aproximar una derivada de segundo orden en términos de valores conocidos, puesto que $u(x+\Delta x,t)$ y $u(x-\Delta x,t)$ corresponden a valores que toma la función en un dominio discretizado, en donde la distancia entre los puntos de la grilla es siempre Δx. Se puede escribir la función valuada en forma discreta:

$$u_i := u(x,t)$$

$$u_{i+1} := u(x+\Delta x,t)$$

$$u_{i-1} := u(x-\Delta x,t)$$

De tal forma que la aproximación de la segunda derivada parcial se puede representar de manera compacta

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} \tag{5}$$

En la figura 1 se observa la representación gráfica de esta discretización.

En general, el dominio temporal de la función también se discretiza, tomando intervalos de tiempo consecutivos separados por un intervalo de tiempo Δt o también llamado tamaño de paso en t.

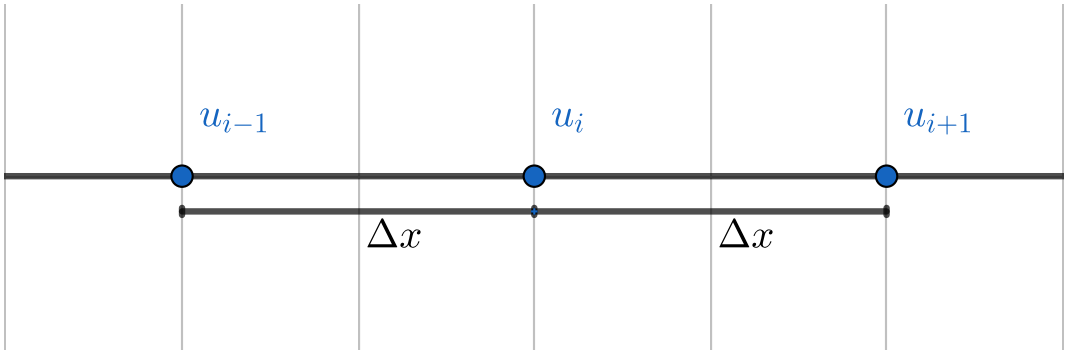


Figura 1: Discretización del dominio espacial

Para aproximar la primera derivada parcial en t de u se expande la función en una serie de Taylor centrada en Δt y el término donde la función está valuada en el instante temporal mayor se renombra como u_{nueva}.

$$u(x, t + \Delta t) \approx u(x, t) + \Delta t \frac{\partial u}{\partial t}$$

$$\frac{\partial u}{\partial t} \approx \frac{u(x, t + \Delta t) - u(x, t)}{\Delta t}$$

$$\frac{\partial u}{\partial t} \approx \frac{u_{\text{nueva},i} - u_i}{\Delta t}$$

Posteriormente se reemplazan las discretizaciones aproximadas en la ecuación diferencial a resolver y se itera sobre la relación de recurrencia encontrada para los valores actuales y futuros de la función en un punto dado.

En resumen, el método DF funciona aproximando y adaptando las derivadas de primer y segundo orden a ecuaciones de diferencias que dependen de los valores que la función devuelve cuando esta se valúa en los puntos de la grilla, o bien, en instantes discretos de tiempo.

A continuación se describen los problemas resueltos con el método de diferencias finitas y los resultados conseguidos con este.

1.1. Ecuación de Burgers no viscosa, en una dimensión

1.1.1. Descripción del problema

La ecuación de Burgers no viscosa en una dimensión espacial es una ecuación diferencial parcial de primer orden que expresa la evolución temporal de la cantidad $u = u(x, t)$, la cual se interpreta como la componente en x de la **velocidad** de un fluido o gas. La ecuación tiene la siguiente forma:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = 0 \tag{6}$$

Se resolvió esta ecuación en un dominio espacial I dado por $I = [0, \mathrm{L}]$, donde $\mathrm{L} = 100\mathrm{m}$, y sujeta a las siguientes condiciones iniciales:

$$u(0, 0) = 0 \tag{7}$$

$$u(\mathrm{L}, 0) = 0 \tag{8}$$

Estas condiciones pueden interpretarse como el modelado de un fluido sin presión ni viscosidad, o un gas, que se mueve en una dimensión cuyos extremos simulan un tope o frontera que impide que el fluido o gas en cuestión salga.

Como condición inicial se eligió una distribución gaussiana de velocidad. Esta se puede interpretar como un pulso centrado en el centro del dominio. Es común utilizar pulsos gaussianos como condiciones iniciales, para así visualizar su desplazamiento a lo largo de la simulación[1].

$$u(x,0) = A\exp\left(-b(x-\mu)^2\right) \tag{9}$$

Donde:

- $A = 3.5\text{m/s}$
- $b = 0.05$
- $\mu = L/2 = 50\text{m}$

1.1.2. Aplicación del método y código implementado

Se utilizó el lenguaje `C++` para resolver el problema numérico utilizando el método de diferencias finitas. El código completo está disponible en el siguiente enlace.

Para resolver numéricamente la ecuación 6 utilizando el método DF, primero se debe discretizar el dominio en el que esta se define. Para la simulación se definió un conjunto de `Nx` puntos sobre el eje x de tal manera que cada par de puntos vecinos estuvieran separados por una distancia `dx`, la cual equivale a Δx en la simbología algebraica de este documento.

Valores de los parámetros para discretización del dominio espacial

```
int Nx = 500; // Número de puntos en el eje x
double L = 100.0; // Largo del dominio en metros
double dx = L/(Nx-1); // Tamaño de paso en el eje x
```

Es destacable que el denominador de la fracción que define a `dx` es `Nx-1` dado que el intervalo contiene esta cantidad de veces la distancia `dx`.

Para almacenar los valores de la función u se utilizaron punteros a arreglos dinámicos de tipo `double`, al igual que para almacenar el conjunto de puntos que conforma el dominio discretizado en el eje x. Estos también fueron inicializados dependiendo de su definición; en el caso de u, se aplicó la condición inicial del problema y las condiciones de frontera.

Definición e inicialización de arreglos dinámicos

```
// Función de velocidad en el tiempo actual: u{x, t} = u_i
double *u = new double[Nx];
// Función de velocidad en el tiempo dt después u{x, t+dt} = u_i+1
double *u_nueva = new double[Nx];
// Puntos sobre el eje x
double *x = new double[Nx];
```

[1] En las ecuaciones se mantuvieron los nombres de las variables utilizadas en el código de la integración numérica de la ecuación, salvo por μ que se escribió como `mu`.

```
    // Inicialización de arreglos
    for (int i = 0; i < Nx; i++)
    {
        x[i] = i*dx;
    }
    // Aplicar condición inicial a u
    for (int i = 0; i < Nx; i++)
    {
        u[i] = f_cond_inicial(x[i]);
    }
    // Condiciones de frontera
    u[0] = 0.0;
    u[Nx-1] = 0.0;
```

Implementación de la ecuación 9

```
double f_cond_inicial(double x)
{
    double b = 0.05;
    double mu = 50;
    double A = 3.5;
    return A*exp(-b*pow(x - mu,2));
}
```

Para discretizar el dominio temporal se tomó `t_total` como la variable que almacena el tiempo total a simular y `dt` como el tamaño de paso temporal. Por tanto, el número de iteraciones necesarias para simular el tiempo completo se obtiene dividiendo `t_total` entre `dt`. Sin embargo, puede que el resultado de esta división no sea un número entero, por lo que se le aplica la función `floor()` para garantizarlo. La variable `num_outs` es un número entero que indica cuántos instantes temporales serán impresos en el archivo de datos; esta cantidad es importante ya que la velocidad de simulación depende de qué tantas veces se imprimen los datos. Las variables de tipo `const` pueden ser cambiadas a voluntad del usuario, siempre y cuando el cambio sea efectuado en la declaración de las mismas.

Valores de los parámetros para discretización del dominio temporal

```
    // Parámetros temporales
    const double t_total = 1.2; // Tiempo total en segundos
    const double dt = 0.000001; // Tamaño de paso temporal en segundos
    int Niter = floor(t_total/dt); // Número total de iteraciones
    const int num_outs = 48; // Número de gráficas de instantes temporales
    int out_cada = floor(Niter / num_outs); // Cada out_cada veces se
                                            // imprimen los valores

    double tiempo = 0.0; // Variable de tiempo en la simulación
```

El siguiente paso para implementar el método consistió en aproximar las derivadas utilizando las discretizaciones disponibles, esto es:

$$\frac{\partial u}{\partial x} \approx \frac{u(x+\Delta x,t)-u(x,t)}{\Delta x} \tag{10}$$

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1} - u_i}{\Delta x} \tag{11}$$

$$\frac{\partial u}{\partial t} \approx \frac{u_{\text{nueva},i} - u_i}{\Delta t} \tag{12}$$

Donde Δx y Δt son los tamaños de paso en la dimensión espacial y temporal respectivamente[2]. Sustituyendo las anteriores aproximaciones en 6, obtenemos

$$\frac{u_{\text{nueva},i} - u_i}{\Delta t} + u_i \left(\frac{u_{i+1} - u_i}{\Delta x} \right) = 0 \tag{13}$$

Luego se despeja $u_{\text{nueva},i}$

$$u_{\text{nueva},i} = u_i \left[1 + \frac{\Delta t}{\Delta x} (u_i - u_{i+1}) \right] \tag{14}$$

De esta forma, el valor de la función u en el i-ésimo elemento de la grilla, en un instante $t + \Delta t$, está dado por el miembro derecho de la ecuación 14, cuyos términos son valores de u en el mismo instante t. Esta es una relación de recurrencia sobre la cual se puede iterar para construir la solución general de la ecuación.

A continuación se muestra el ciclo principal de integración numérica de la ecuación de Burgers

Ciclo principal de integración

```
for (int j = 0; j < Niter; j++)
{
    for (int i = 1; i < Nx-1; i++)
    {
        u_nueva[i] = u[i]*(1 + (dt/dx)*(u[i]-u[i+1]));
    }

    // Condiciones de frontera
    u_nueva[0] = 0.0;
    u_nueva[Nx-1] = 0.0;

    // Actualizar u
    for (int i = 0; i < Nx; i++)
    {
        u[i] = u_nueva[i];
    }

    // Se imprime la solución de la iteración
    if (j % out_cada == 0)
        salida(outfile, u, x, tiempo, Nx);

    // Actualizamos el tiempo
    tiempo += dt;
}
```

[2]En el código, estas cantidades fueron nombradas como `dx` y `dt` respectivamente

Se puede notar que para realizar la integración numérica de la ecuación se necesitan anidar dos ciclos `for`, uno para iterar sobre el dominio temporal y otro para el dominio espacial. En la quinta línea se puede apreciar la implementación de la ecuación 14 en código y en las líneas consecuentes se observa la asignación de las condiciones de frontera.

Se utilizó una función especial llamada `salida` que toma como argumento una variable de tipo `ofstream` la cual sirve para enviar datos al archivo deseado.

Definición de función de salida de datos

```
void salida(ofstream &of, double *u, double *x, double t, int N)
{
    for (int i = 0; i < N; i++)
    {
        of << t << "\t" << x[i] << "\t" << u[i] << endl;
    }
    of << endl << endl;
}
```

1.1.3. Resultados

Se graficó una animación de la simulación numérica en **Gnuplot**, una herramienta de visualización de datos ampliamente utilizada en el ámbito científico. Se encontró que después de un tiempo de simulación de 1.2s, los resultados comenzaron a presentar un considerable error numérico y a perder significado físico. Sin embargo, se pudo explicar la razón de este fenómeno investigando sobre la ecuación de Burgers. En la sección 1.1.4 se discutirá con más detalle el tema del error numérico de la solución.

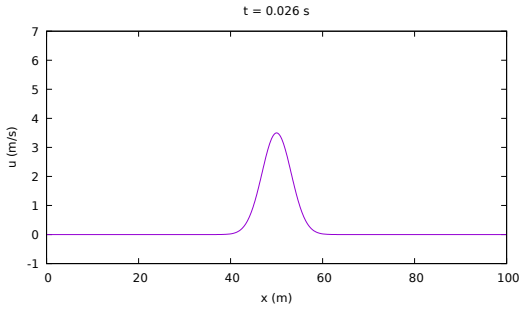
En la figura 2 se muestran seis instantes de la evolución temporal. La animación de la simulación completa se encuentra disponible en el siguiente enlace.

1.1.4. Discusión de resultados

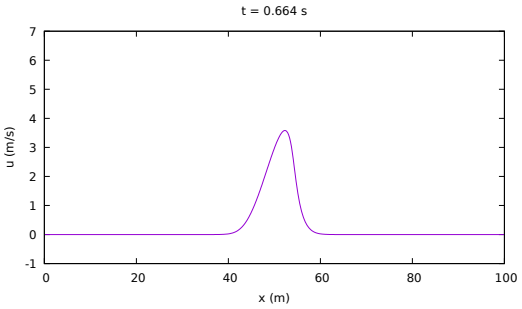
La ecuación de Burgers resuelta con el método de diferencias finitas mostró algunas irregularidades, principalmente la pérdida de continuidad en varias partes de la solución a partir del segundo 1.021 de la simulación. Este fenómeno es causado gracias al efecto de **onda de choque** (o **shockwave** en inglés) que la ecuación de Burgers presenta. Una onda de choque se define como una discontinuidad en la velocidad de propagación de una onda y dado que la ecuación de Burgers representa la velocidad del fluido modelado, la discontinuidad se manifiesta en la función solución de la ecuación.

Para explicar el fenómeno de onda de choque que se produce en la ec. de Burgers se puede utilizar el **método de características** para la resolución de la ecuación misma. El método de características consiste en transformar el dominio de la ecuación tal que, en éste, la ecuación diferencial parcial se pueda escribir como un sistema de ecuaciones diferenciales ordinarias, llamadas **ecuaciones características**. Las curvas del dominio en donde al valuar la ecuación esta se puede escribir como un sistema de ecuaciones diferenciales ordinarias se llaman **curvas características.** Para encontrar las curvas características de la ec. de Burgers se necesitan resolver las ecuaciones características:

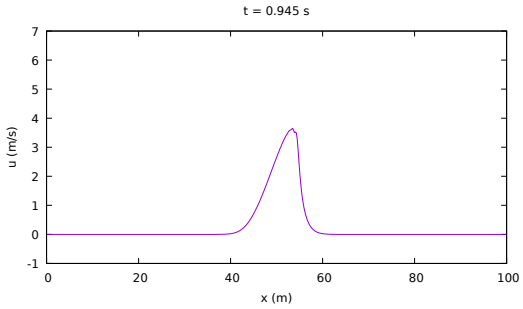
$$\frac{\mathrm{d}x}{\mathrm{d}t} = u(x,t) \tag{15}$$



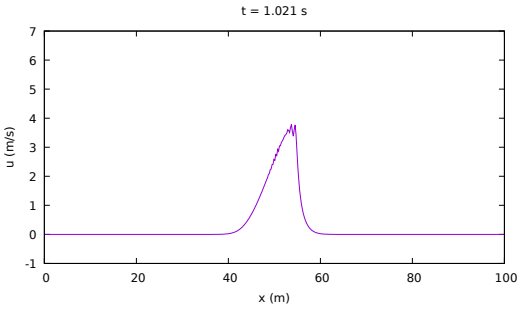
Gráfica de $u(x, t = 0.026\text{s})$



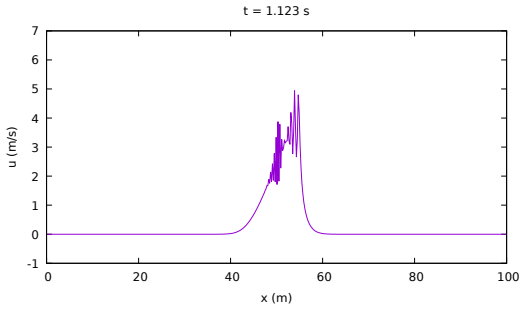
Gráfica de $u(x, t = 0.664\text{s})$



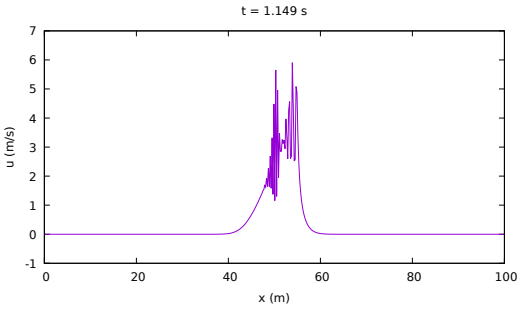
Gráfica de $u(x, t = 0.945\text{s})$



Gráfica de $u(x, t = 1.021\text{s})$



Gráfica de $u(x, t = 1.123\text{s})$



Gráfica de $u(x, t = 1.149\text{s})$

Figura 2: Seis instantes de tiempo de la evolución temporal de la ecuación de Burgers no viscosa con diferencias finitas.

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{\partial u}{\partial t} + \frac{\mathrm{d}x}{\mathrm{d}t}\frac{\partial u}{\partial x} = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \tag{16}$$

Notamos en la ecuación 16 que u es constante en el tiempo a lo largo de cualquier curva característica, por lo que resolviendo la ecuación 15 obtenemos:

$$x - x_0 = ut \tag{17}$$

$$x(t) = x_0 + u_0(x_0)t \tag{18}$$

Donde $u_0(x) = u(x, 0)$

Notamos que las curvas características son rectas en el plano $x - t$ que parten de un punto $(x_n, 0)$ y poseen pendiente $u_0(x_n)$, donde x_n es un punto que pertenece al dominio espacial. Consecuentemente se puede demostrar que estas curvas se intersectarán en un tiempo finito si $u_0'(x) < 0$ para algún x. Supongamos que las dos características $x(t) = x_1 + u_0(x_1)t$ y $x(t) = x_2 + u_0(x_2)t$

llegan a intersectarse en un tiempo t_{int}.

$$x_1 + u_0(x_1)t_{int} = x_2 + u_0(x_2)t_{int} \tag{19}$$

$$t_{int} = -\frac{x_2 - x_1}{u_0(x_2) - u_0(x_1)} \tag{20}$$

$$t_{int} = -\frac{1}{\frac{u_0(x_2) - u_0(x_1)}{x_2 - x_1}} \tag{21}$$

$$t_{int} = -\frac{1}{\frac{u_0(x_2) - u_0(x_1)}{x_2 - x_1}} \tag{22}$$

Por el teorema del valor medio, podemos asumir que para un x_0 tal que $x_1 < x_0 < x_2$ existe $u'(x_0) = \frac{u_0(x_2) - u_0(x_1)}{x_2 - x_1}$ y entonces, u' sí es negativa en algún punto, por lo tanto, el tiempo t_{int} es positivo y las características se intersectan.

Cuando las características se intersectan la onda viajera se rompe ya que según la ecuación diferencial que la rige esta debería seguir siendo constante en dos puntos distintos y esto produce la discontinuidad. El método numérico de diferencias finitas reacciona erróneamente ante la discontinuidad que se presenta en la solución, puesto que el término $u_i - u_{i+1}$ es una cantidad que crece cada vez más en cada iteración y el error numérico ya no se puede despreciar, por lo que la solución pierde el significado físico.

La segunda irregularidad presentada por los resultados tiene relación con la forma de la ecuación de Burgers, dado que al investigar sobre la resolución numérica se encontró que la ecuación 6 no es la adecuada para implementar en métodos numéricos, sino la forma conservativa de la misma:

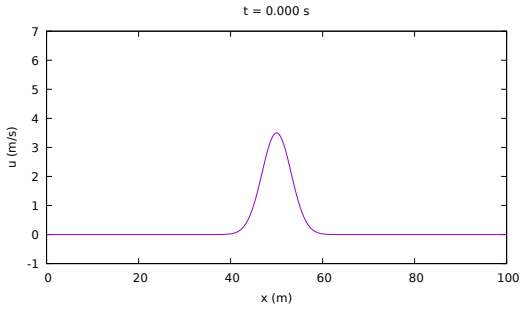
$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial (u^2)}{\partial x} = 0 \tag{23}$$

Con esta ecuación se puede obtener la ecuación 6 utilizando la regla de la cadena, al igual que calculando la derivada material de u; pero esta transformación es válida solamente si la ecuación es diferenciable en todo el dominio, lo cual no sucede en la ecuación de Burgers por el efecto de onda de choque previamente explicado. Por esta razón, es recomendable usar la versión conservativa (ecuación 23) para resolver la ecuación de Burgers.

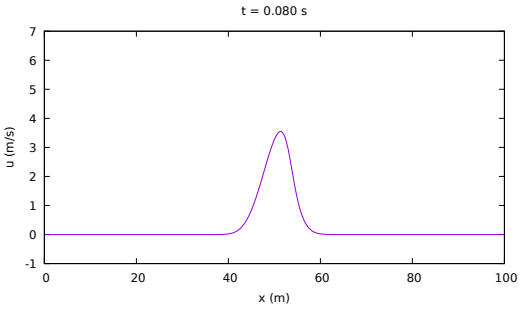
Se utilizó esta versión de la ecuación en una simulación usando la discretización siguiente:

$$u_{\text{nueva},i} = u_i + \frac{\Delta t}{2\Delta x}(u_i^2 - u_{i+1}^2) \tag{24}$$

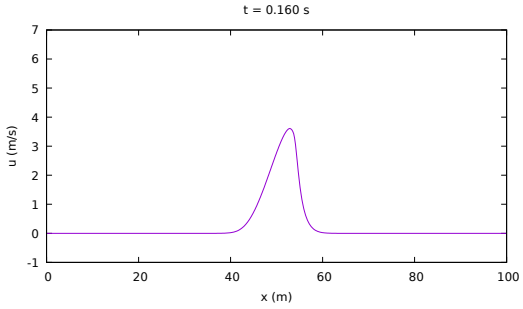
En la figura 3 se muestran cuatro instantes de la solución usando la versión conservativa de la ecuación de Burgers. Se puede notar que en esta solución la onda se propaga mucho más rápido que la onda de la solución de la ecuación de Burgers tradicional, pero también pierde significado físico a causa de la onda de choque. Por tanto el método de diferencias finitas es muy débil para simular la ecuación de Burgers no viscosa.



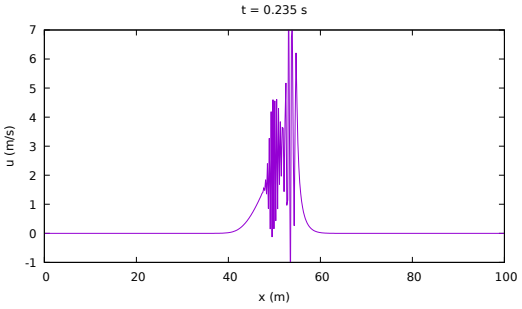
Gráfica de $u(x, t = 0.0\text{s})$



Gráfica de $u(x, t = 0.08\text{s})$



Gráfica de $u(x, t = 0.016\text{s})$



Gráfica de $u(x, t = 0.0235\text{s})$

Figura 3: Cuatro instantes de tiempo de la evolución temporal de la ecuación de Burgers en su versión conservativa, con diferencias finitas.

1.2. Ecuación de Burgers viscosa, en una dimensión

1.2.1. Descripción del problema

La ecuación de Burgers viscosa es una ecuación diferencial parcial no lineal de **segundo orden,** que se utiliza para modelar la velocidad de gases o fluidos con viscosidad considerable. La ecuación tiene la siguiente forma:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{25}$$

Donde ν es el coeficiente de viscosidad del medio modelado. Cabe destacar que ν es el inverso del **número de Reynolds** Re, una cantidad adimensional que sirve para medir la proporción entre fuerzas inerciales y viscosas.

En consecuencia de haber obtenido distintos resultados en la sección 1.1 al utilizar la versión conservativa (23) y no conservativa (6) de la ecuación de Burgers no viscosa, para este problema se optó por resolver la versión conservativa con el término de viscosidad incluido.

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial (u^2)}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{26}$$

La ecuación 26 se resolvió en el mismo dominio espacial I y bajo las mismas condiciones iniciales y de frontera que la versión no viscosa de la sección 1.1.

$$u(0, 0) = 0 \tag{27}$$

$$u(\text{L}, 0) = 0 \tag{28}$$

$$u(x,0) = A\exp\left(-b(x-\mu)^2\right) \tag{29}$$

Donde:

- $L = 100\text{m}$
- $A = 3.5\text{m/s}$
- $b = 0.05$
- $\mu = L/2 = 50\text{m}$

Se resolvieron tres ecuaciones con un coeficiente de viscosidad ν distinto para cada una: $1.60\text{m}^2/\text{s}$, $3.0\text{m}^2/\text{s}$ y $6.0\text{m}^2/\text{s}$

1.2.2. Aplicación del método y código implementado

Para este problema se utilizó como código base el programa de la solución no viscosa, haciendo algunos cambios; por ejemplo, el del tiempo de simulación total y la impresión de datos.

Variables generales de la simulación

```
// Tiempo total en segundos
const double t_total = 8;
// Tamaño de paso temporal
const double dt = 0.001;
// Número total de iteraciones
int Niter = floor(t_total/dt);
// Número de gráficas de instantes temporales
const int num_outs = 400;
// Cada out_cada veces se imprimen los valores
int out_cada = floor(Niter / num_outs);
// Variable de tiempo en la simulación
double tiempo = 0.0;

// Parámetros espaciales
int Nx = 500;        // Número de puntos en el eje x
double L = 100.0;    // Largo del dominio en metros
double dx = L/(Nx-1);    // Tamaño de paso en el eje x

// Arreglos y constantes
// Parámetro de viscosidad
const double nu = 0.3;
// Función de velocidad en el tiempo actual: u{x, t}
double *u = new double[Nx];
// Función de velocidad en el tiempo dt después u{x, t+dt}
double *u_nueva = new double[Nx];
// Función de distancia sobre el eje x
double *x = new double[Nx];
```

```
    // Variables y archivos de salida
    // Archivo donde se guarda la función solución u
    ofstream outfile;
    // Archivo de gnuplot para graficar la función u(x,t)
    ofstream gplotmain;
    // Nombres de los archivos de datos y de gráficas
    char name_datafile[31];
    char name_gplotmain[28];
    sprintf(name_datafile, "sol-burg-vis1DDF-nu-%.2f.dat", nu);
    sprintf(name_gplotmain, "burg-vis1DDF-nu-%.2f.gp", nu);
    // Se crean los archivos de datos y gráficas
    outfile.open(name_datafile, ios::out );
    gplotmain.open(name_gplotmain, ios::out);
```

Como puede notarse en el código presentado, los archivos `.dat` y `.gp` están nombrados de acuerdo al coeficiente de viscosidad utilizado en la solución que representan.

Se aplicó la discretización de la ecuación 5 para tratar la derivada de segundo orden de la ecuación de Burgers viscosa. Aplicando las discretizaciones en la ecuación 26 obtenemos:

$$\frac{u_{\text{nueva},i} - u_i}{\Delta t} + \frac{1}{2}\left(\frac{u_{i+1}^2 - u_i^2}{\Delta x}\right) - \nu\left(\frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}\right) = 0 \tag{30}$$

$$u_{\text{nueva},i} = u_i + \frac{\Delta t}{2\Delta x}\left(u_i^2 - u_{i+1}^2\right) + \nu\frac{\Delta t}{(\Delta x)^2}\left(u_{i+1} - 2u_i + u_{i-1}\right) \tag{31}$$

A continuación se muestra el ciclo principal de integración numérica de la ecuación de Burgers viscosa

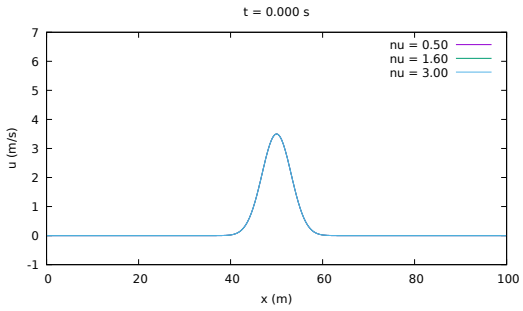
Ciclo principal de integración

```
    for (int j = 0; j < Niter; j++)
    {
        for (int i = 1; i < Nx-1; i++)
        {
            u_nueva[i] = u[i] + 0.5*(dt/dx)*(pow(u[i], 2)-pow(u[i+1], 2))
            + nu*(dt/dx)*(u[i+1]-2*u[i]+u[i-1])/dx;
        }

        // Condiciones de frontera
        u_nueva[0] = 0.0;
        u_nueva[Nx-1] = 0.0;

        // Actualizar u
        for (int i = 0; i < Nx; i++)
        {
            u[i] = u_nueva[i];
        }

        // Se imprime la solución de la iteración
        if (j % out_cada == 0)
```

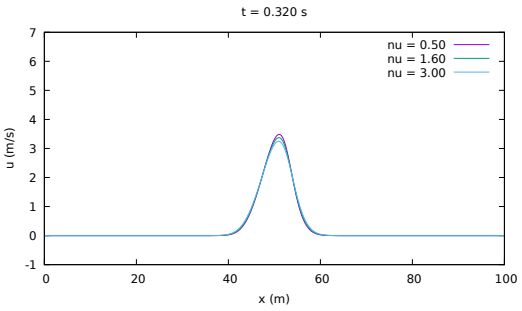
```
            salida(outfile, u, x, tiempo, Nx);
        // Actualizamos el tiempo
        tiempo += dt;
    }
```

1.2.3. Resultados

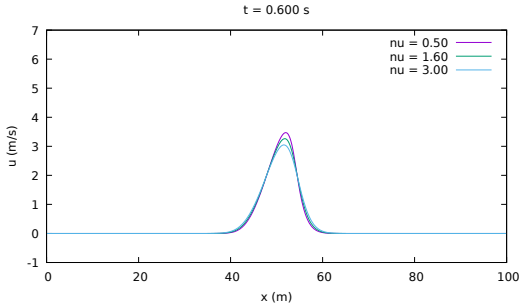
Se graficaron las tres soluciones obtenidas con los diferentes coeficientes de viscosidad considerados [3]. Se muestran en la figura 4 las gráficas para seis instantes de tiempo en orden ascendente. La animación completa de la simulación está disponible en el siguiente enlace.



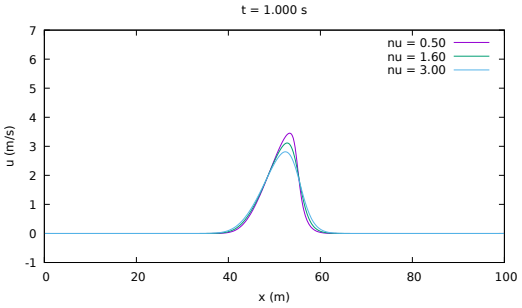
Gráficas para $t = 0.00$s



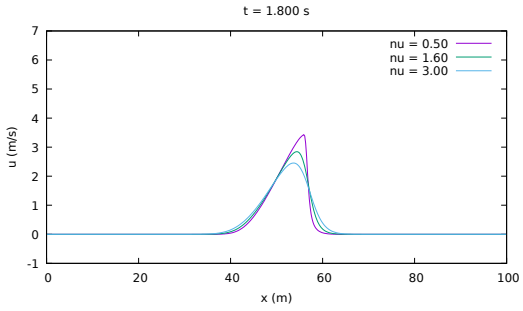
Gráficas para $t = 0.032$s



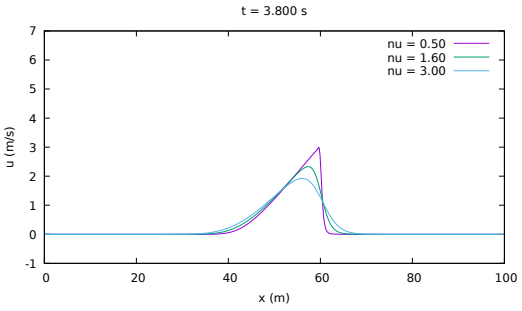
Gráfica de $u(x, t = 0.32$s$)$



Gráfica de $u(x, t = 1.0$s$)$



Gráfica de $u(x, t = 1.8$s$)$



Gráfica de $u(x, t = 3.8$s$)$

Figura 4: Seis instantes de tiempo de la evolución temporal de tres versiones de la ecuación de Burgers viscosa para diferentes coeficientes de viscosidad, con diferencias finitas.

[3] En la viñeta de cada gráfica, ν esta escrito como **nu**

2. Método de volúmenes finitos

3. Método de elementos finitos

4. Conclusiones

Referencias

[1] P. DeVries, P.L. DeVries y J. Hasbun. *A First Course in Computational Physics*. Jones & Bartlett Learning, 2011. ISBN: 9780763773144. URL: https://books.google.com.gt/books?id=X3FEPiebLH0C.