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.

25 de mayo de 2023

Índice

1. Introducción

En el estudio de la física, existen múltiples maneras de conocer más a profundidad las teorías que se estudian y también para comprobarlas. La física siempre ha dependido del área experimental, ya que las leyes y postulados que se definen en una teoría deben ser consistentes con cualquier experimento; de lo contrario, la teoría está destinada a fracasar.

Sin embargo, gracias al avance tecnológico exponencial que se ha presenciado en el último siglo, la física posee hoy en día una herramienta que le permitirá continuar evolucionando: **la simulación computacional.** Si bien el área experimental es la piedra angular de cualquier ciencia exacta como la física, la simulación computacional de un sistema físico puede resultar igual o aún más ilustrativa que un experimento del mismo sistema. Más importante, una simulación puede ser capaz de brindarle al investigador una idea más concreta y visual de fenómenos extremos (altas temperaturas, tamaños microscópicos, etc.) que están fuera del alcance de los experimentos.

2. 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 versión aproximada de la ecuación diferencial algebraica iterativamente [2].

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 se utilizan las expansiones en series de Taylor:

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

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

Sumando ambas aproximaciones se obtiene:

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

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

Por tanto, se puede 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.

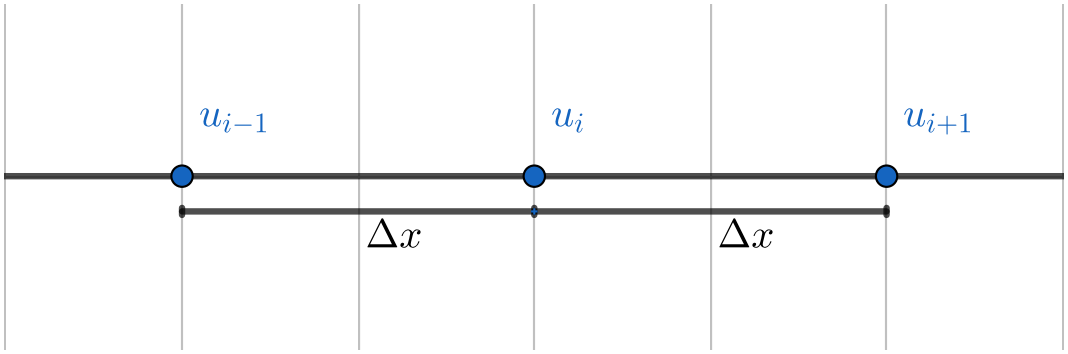


Figura 1: Discretización del dominio espacial

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}.

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

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.

2.1. Ecuación de Burgers no viscosa, en una dimensión

2.1.1. Descripción del problema

La ecuación de Burgers no viscosa en una dimensión espacial es una ecuación diferencial parcial no lineal 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 $I = [0, L]$, donde $L = 100\text{m}$, y sujeta a las siguientes condiciones de frontera:

$$u(0, t) = 0 \tag{7}$$
$$u(L, t) = 0 \tag{8}$$

La imposición de 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:

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

2.1.2. 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 los valores disponibles sobre la grilla, 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, se obtiene

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

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;
}
```

2.1.3. 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 2.1.4 se discutirá con más detalle el tema del error numérico de la solución.

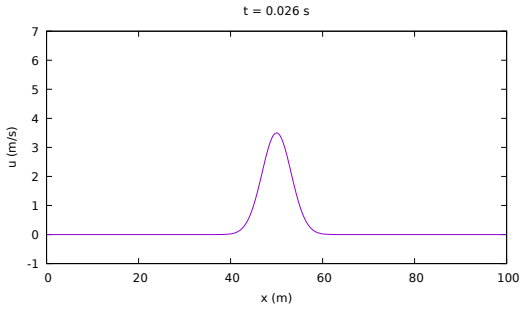
En la figura 2 se muestran seis instantes de la evolución temporal. La animación de la simulación completa se encuentra disponible en el siguiente enlace.

2.1.4. Discusión de resultados

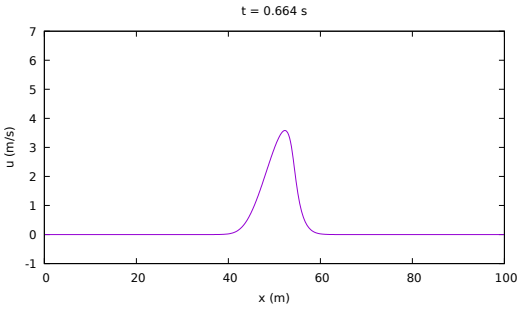
La ecuación de Burgers resuelta con el método de diferencias finitas mostró algunas irregularidades, principalmente la pérdida de continuidad en varias partes de la solución a partir del segundo 1.021 de la simulación. Este fenómeno es causado gracias al efecto de **onda de choque** (o **shockwave** 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:

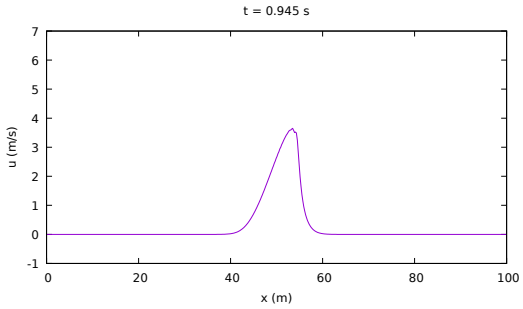
$$\frac{\mathrm{d}x}{\mathrm{d}t} = u(x,t) \tag{15}$$



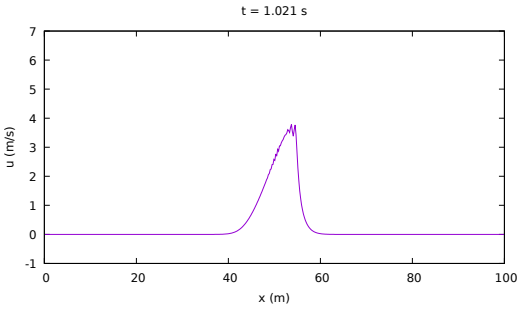
Gráfica de $u(x, t = 0.026\text{s})$



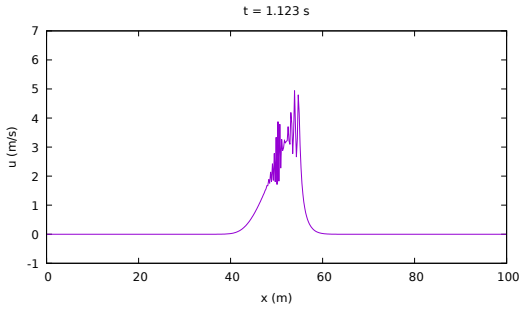
Gráfica de $u(x, t = 0.664\text{s})$



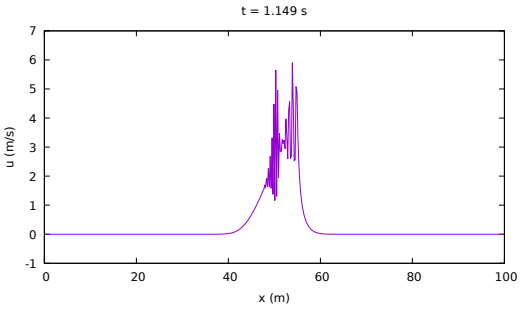
Gráfica de $u(x, t = 0.945\text{s})$



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.

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

Se puede observar 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 se obtiene:

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

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

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

Se puede notar 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. Si se supone 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, se puede asumir que para un x_0 tal que $x_1 < x_0 < x_2$ existe $u'(x_0) = \dfrac{u_0(x_2) - u_0(x_1)}{x_2 - x_1}$ 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:

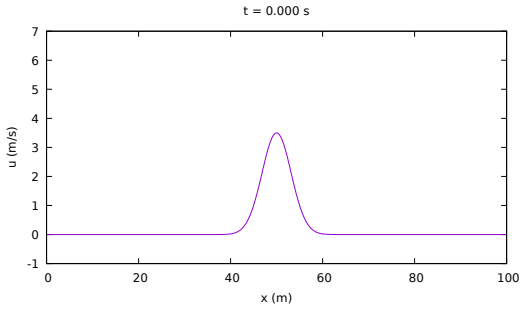
$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial (u^2)}{\partial x} = 0 \tag{23}$$

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.

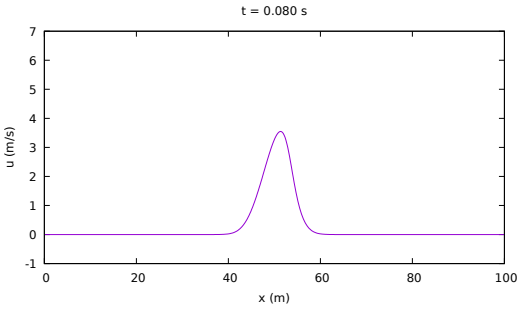
Se utilizó esta versión de la ecuación en una simulación usando la discretización siguiente:

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

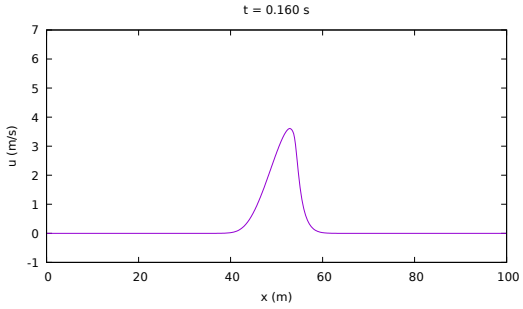
En la figura 3 se muestran cuatro instantes de la solución usando la versión conservativa de la ecuación de Burgers. Se puede notar que en esta solución la onda se propaga mucho más rápido que la onda de la solución de la ecuación de Burgers tradicional, pero también pierde significado físico a causa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



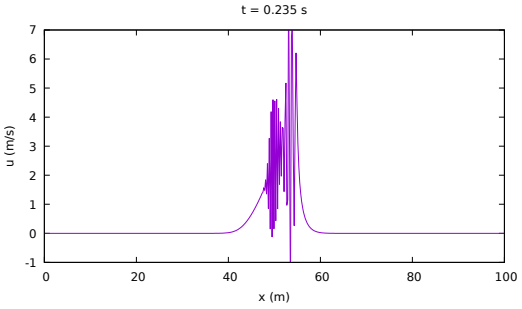
Gráfica de $u(x, t = 0.0\text{s})$



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.

2.2. Ecuación de Burgers viscosa, en una dimensión

2.2.1. 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 2.1 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 2.1.

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

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

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

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:$0.50\text{m}^2/\text{s}$, $1.60\text{m}^2/\text{s}$ y $3.0\text{m}^2/\text{s}$.

2.2.2. 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 = 0.5;
// 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 se obtiene:

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

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

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

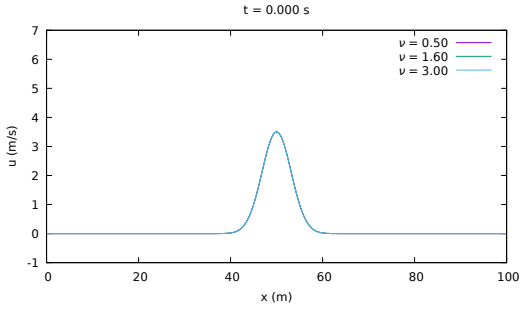
Ciclo principal de integración

```
    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)
```

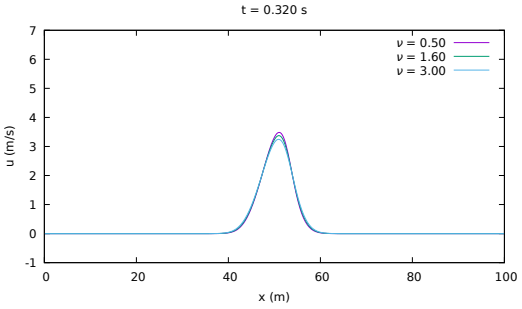
```
            salida(outfile, u, x, tiempo, Nx);
        // Actualizamos el tiempo
        tiempo += dt;
    }
```

2.2.3. Resultados

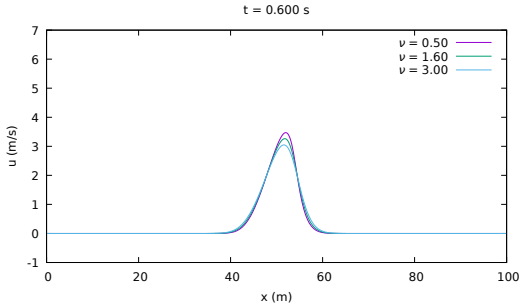
Se graficaron las tres soluciones obtenidas con los diferentes coeficientes de viscosidad considerados [3]. Se muestran en la figura 4 las gráficas para seis instantes de tiempo en orden ascendente. La animación completa de la simulación está disponible en el siguiente enlace.



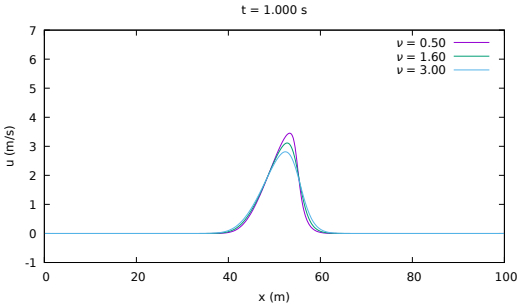
Gráficas para $t = 0.00$s



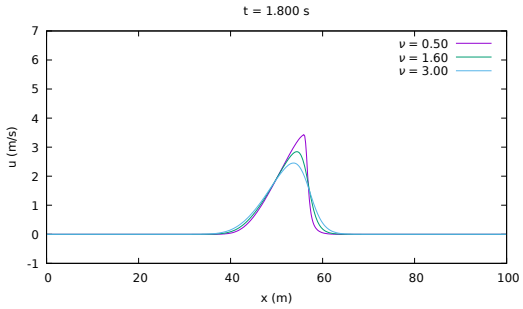
Gráficas para $t = 0.032$s



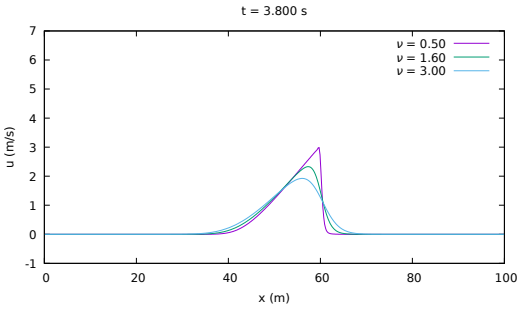
Gráfica de $u(x, t = 0.32\text{s})$



Gráfica de $u(x, t = 1.0\text{s})$



Gráfica de $u(x, t = 1.8\text{s})$



Gráfica de $u(x, t = 3.8\text{s})$

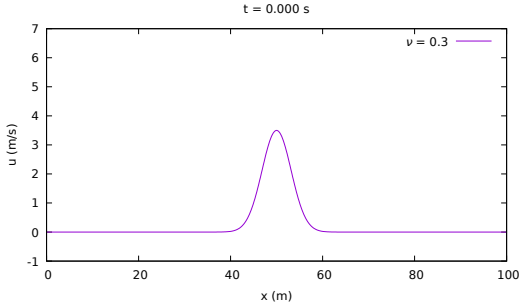
Figura 4: Seis instantes de tiempo de la evolución temporal de tres versiones de la ecuación de Burgers viscosa para diferentes coeficientes de viscosidad, con diferencias finitas.

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

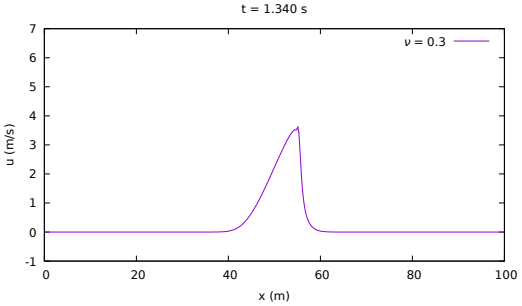
2.2.4. Discusión de resultados

La solución numérica de la ecuación de Burgers viscosa proporciona una comprensión más coherente de la evolución temporal de la velocidad de un fluido modelado en comparación con la ecuación de Burgers no viscosa. Esto se debe a que la solución numérica no presenta irregularidades considerables o efectos que hagan perder el significado físico del sistema. Según lo investigado, los métodos más frecuentemente utilizados para resolver la ecuación de Burgers suelen incluir un término viscoso para calibrar y afinar la solución que se desea analizar. Esto es lógico, ya que en la dinámica de un gas o fluido real siempre estará presente algún nivel de viscosidad.

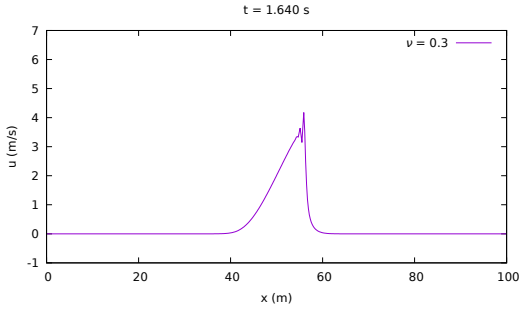
Al resolver este problema se consideraron varios valores para el coeficiente de viscosidad ν, como se puede ver en la figura 4. Cabe destacar que $\nu = 0.5$ fue el valor mínimo que este pudo tener sin que la solución perdiera significado físico. Por lo tanto, el método de diferencias finitas está limitado a resolver la ecuación viscosa de Burgers con $\nu > 0.5$; no obstante, el método es mucho más útil al resolver la versión viscosa que la versión no viscosa de la ecuación de Burgers, ya que hay soluciones que no presentan error numérico considerable. En la figura 5 se pueden observar cuatro instantes de tiempo de la solución de la ecuación viscosa con $\nu = 0.3$ utilizando diferencias finitas.



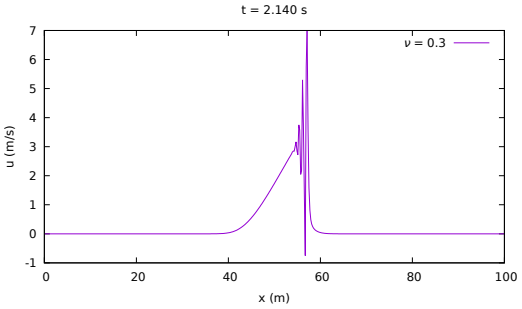
Gráfica de $u(x, t = 0.0\text{s})$



Gráfica de $u(x, t = 1.34\text{s})$



Gráfica de $u(x, t = 1.64\text{s})$



Gráfica de $u(x, t = 2.14\text{s})$

Figura 5: Cuatro instantes de tiempo de la evolución temporal de la ecuación de Burgers con viscosidad igual a 0.3, resuelta con diferencias finitas.

3. Método de volúmenes finitos

El siguiente método utilizado fue el **método de volúmenes finitos.** Este método se especializa en resolver ecuaciones diferenciales escritas en forma conservativa y es capaz de integrarlas aún cuando las soluciones poseen discontinuidades o en el contexto de las ecuaciones de la mecánica de fluidos, choques (*shocks* en inglés) [3]. El método de volúmenes finitos es aplicable utilizando métodos llamados *esquemas* o *marcos*, que sirven para calcular apropiadamente el **flujo** en las interfaces de las celdas de la grilla [3]. Se explicará la deducción del método de volúmenes finitos, los marcos utilizados y la teoría de su funcionamiento.

El principio básico del método de volúmenes finitos es la reinterpretación de los valores discretizados u_i^n de la función incógnita u; dado que en lugar de considerar a estos valores como la aproximación del valor exacto $u(x_i, t_n)$[4] se interpretan como el **valor promedio** de la función u en el intervalo $C_i = [x_i, x_{i+1}]$, también denominado **celda**. Entonces la discretización de la función consiste en dividir el intervalo $I = [0, L]$ en N intervalos C_i, donde $\Delta x = L/N = x_{i+1} - x_i$. De tal manera que, formalmente, el valor u_i^n se calcula con una integral en x sobre C_i:

$$u_i^n \approx \frac{1}{\Delta x} \int_{C_i} u(x, t_n)\, \mathrm{d}x \equiv \frac{1}{\Delta x} \int_{x_i}^{x_{i+1}} u(x, t_n)\, \mathrm{d}x \tag{32}$$

Ahora, el método toma ventaja de esta nueva interpretación de la discretización cuando se integra la versión conservativa de la ecuación diferencial a resolver, es decir, una ecuación de la forma:

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \tag{33}$$

donde $f(u)$ es la función del flujo de u; para la ecuación de Burgers $f(u) = \frac{u^2}{2}$. Entonces, integrando dicha ecuación en ambas variables:

$$\int_{C_i} \frac{\partial u}{\partial t}\, \mathrm{d}x + \int_{C_i} \frac{\partial f(u)}{\partial x}\, \mathrm{d}x = 0 \tag{34}$$

$$\int_{C_i} \frac{\partial u}{\partial t}\, \mathrm{d}x + f(u(x_{i+1}, t)) - f(u(x_i, t)) = 0 \tag{35}$$

$$\int_{t_n}^{t_{n+1}} \int_{C_i} \frac{\partial u}{\partial t}\, \mathrm{d}x\, \mathrm{d}t + \int_{t_n}^{t_{n+1}} [f(u(x_{i+1}, t)) - f(u(x_i, t))]\, \mathrm{d}t = 0 \tag{36}$$

$$\int_{C_i} [u(x, t_{n+1}) - u(x, t_n)]\, \mathrm{d}x + \int_{t_n}^{t_{n+1}} [f(u(x_{i+1}, t)) - f(u(x_i, t))]\, \mathrm{d}t = 0 \tag{37}$$

Esta última expresión es totalmente exacta y expresa cómo cambiará el valor promedio de u en cada celda, después de un instante Δt. Ahora, haciendo uso de la aproximación de la ecuación 32, se tiene:

$$\Delta x \left(u_i^{n+1} - u_i^n\right) + \int_{t_n}^{t_{n+1}} [f(u(x_{i+1}, t)) - f(u(x_i, t))]\, \mathrm{d}t = 0 \tag{38}$$

$$u_i^{n+1} - u_i^n + \frac{1}{\Delta x} \int_{t_n}^{t_{n+1}} [f(u(x_{i+1}, t)) - f(u(x_i, t))]\, \mathrm{d}t = 0 \tag{39}$$

[4] La notación introducida $u_i^n \approx u(x_i, t_n)$ es tal que, $t_n \equiv n\Delta t$ con $n \in \mathbb{N}$

Por otro lado, la siguiente integral se puede interpretar como el flujo promedio en el intervalo temporal $[t_n, t_{n+1}]$. Esto es:

$$F_i^n \approx \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} f(u(x_i, t))\, \mathrm{d}t \tag{40}$$

Sustituyendo en la integración de la ecuación, se obtiene:

$$u_i^{n+1} - u_i^n + \frac{\Delta t}{\Delta x} \left[F_{i+1}^n - F_i^n \right] = 0 \tag{41}$$

Entonces, la ecuación 41 es la forma de **diferencia de flujos** de la ecuación de conservación; y a partir de la misma se pueden desarrollar los métodos numéricos basados en el método de volúmenes finitos, puesto que la función F_i^n es una aproximación numérica. Para que el método funcione, la función F_i^n debe calcularse utilizando los valores de la grilla, u^n de tal manera que se obtenga un método numérico totalmente discretizado [3].

Se dice que la ecuación 41 es un método numérico **conservativo** ya que emula la idea de la solución exacta expuesta en el desarrollo del inicio de este capítulo (ecuación 34). Si se suman los valores que toma u, desde la celda L hasta la celda R, se obtiene:

$$\sum_{i=L}^{R} u_i^{n+1} - \sum_{i=L}^{R} u_i^n + \frac{\Delta t}{\Delta x} \left[\sum_{i=L}^{R} F_{i+1}^n - \sum_{i=L}^{R} F_i^n \right] = 0 \tag{42}$$

$$\Delta x \sum_{i=L}^{R} u_i^{n+1} - \Delta x \sum_{i=L}^{R} u_i^n + \Delta t \left[F_{R+1}^n - F_L^n \right] = 0 \tag{43}$$

Entonces, se concluye que el resultado del equivalente discreto de la integración de la función u varía en el tiempo solamente debido a los flujos en las celdas limitantes del dominio considerado. Esta es la propiedad conservativa de la ecuación de diferencia de flujos.

Volviendo a considerar el flujo F_i^n, se puede notar en la ecuación 40 que la expresión para el flujo promedio en una interfaz, F_i^n contiene una integral y generalmente, en consecuencia de que u depende del tiempo, esta integral no es trivial. Sin embargo, se sabe que la información de la onda se propaga a través del espacio con una velocidad finita, lo que implica que es posible obtener F_i^n a partir de los valores u_i y u_{i-1} [3]:

$$F_i^n = F(u_{i-1}^n, u_i^n) \tag{44}$$

De tal manera que, sustituyendo en la ecuación 41 se obtiene:

$$u_i^{n+1} - u_i^n + \frac{\Delta t}{\Delta x} \left[F(u_i^n, u_{i+1}^n) - F(u_{i-1}^n, u_i^n) \right] = 0 \tag{45}$$

A continuación se explican los marcos numéricos evaluados para calcular el flujo numérico F_i^n para resolver la ecuación de Burgers, donde $f(u) = \frac{u^2}{2}$.

3.1. Esquema de Lax-Friedrichs

La primera expresión que lógicamente se puede proponer para poder calcular $F(u_{i-1}^n, u_i^n)$ es promediar el valor del flujo $f(u)$ valuado en u_{i-1}^n y en u_i^n; esto es:

$$F(u_{i-1}^n, u_i^n) = \frac{1}{2} \left(f(u_{i-1}^n) + f(u_i^n) \right) \tag{46}$$

$$F(u_{i-1}^n, u_i^n) = \frac{1}{2}\left((u_{i-1}^n)^2 + (u_i^n)^2\right) \tag{47}$$

Sin embargo, esta elección para el flujo numérico no proporciona una integración estable para ningún valor de $\frac{\Delta t}{\Delta x}$ [3]. Por tanto, se le agrega un término al flujo que actúa como un *flujo difusivo* y compensa parcialmente los efectos de las discontinuidades en las soluciones. En el método de Roe se explicará la razón de la inclusión de este término.

$$\boxed{F(u_{i-1}^n, u_i^n) = \frac{1}{2}\left(\frac{(u_{i-1}^n)^2 + (u_i^n)^2}{2}\right) - \frac{\Delta t}{2\Delta x}(u_i^n - u_{i-1}^n)} \tag{48}$$

Este es el esquema de Lax-Friedrichs para el flujo numérico de la ecuación de Burgers.

3.2. Esquema de Godunov

El esquema de Godunov se justifica a través de conceptos avanzados de las ecuaciones diferenciales de conservación; pero la idea fundamental del método es simple. El esquema de Godunov consiste en resolver el **problema de Riemann** en cada una de las celdas de la solución, este problema se explicará con detalle después de considerar algunas propiedades de las ecuaciones diferenciales en forma conservativa. Primero, se debe comprender el concepto de **solución débil** de una ecuación diferencial.

3.2.1. Solución débil de una ecuación diferencial

Como fue visto en secciones anteriores, la ecuación de Burgers tiene soluciones discontinuas, lo cual parece contradecir la misma ecuación diferencial puesto que esta posee derivadas parciales en todas sus variables y una función discontinua no es derivable en todo su dominio. Con el objetivo de validar las funciones discontinuas como soluciones de las ecuaciones diferenciales el concepto de solución débil fue introducido a la teoría de ecuaciones diferenciales parciales [1].

Como motivación para hacer riguroso el concepto de la solución débil se define una función infinitamente suave $\phi(x,t)$ que posee un soporte compacto, es decir, que la función es distinta de cero solamente en un subconjunto compacto del espacio $(x,t) = \mathbb{R}\times[0,+\infty]$. Luego se define $u(x,t)$ tal que esta satisface la ecuación de conservación general $\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0$. Entonces integrando por partes se obtiene lo siguiente:

$$\int_0^\infty \int_{-\infty}^\infty \left(\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x}\right)\phi(x,t)\,\mathrm{d}x\,\mathrm{d}t = 0 \tag{49}$$

$$\left(\int_{-\infty}^\infty \phi u\,\mathrm{d}x\right)\Big|_0^\infty - \int_0^\infty \int_{-\infty}^\infty \left(u\frac{\partial \phi}{\partial t} + f(u)\frac{\partial \phi}{\partial x}\right)\mathrm{d}x\,\mathrm{d}t = 0 \tag{50}$$

A partir de este resultado, yace la definición de solución débil.

Definición 3.1. $u(x,t)$ es una solución débil de la ecuación diferencial $\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0$ si para cualquier función suave $\phi(x,t)$ con soporte compacto se satisface lo siguiente:

$$\int_0^\infty \int_{-\infty}^\infty \left(u\frac{\partial \phi}{\partial t} + f(u)\frac{\partial \phi}{\partial x}\right)\mathrm{d}x\,\mathrm{d}t = \left(\int_{-\infty}^\infty \phi u\,\mathrm{d}x\right)\Big|_0^\infty = -\int_{-\infty}^\infty \phi(x,0)u(x,0)\,\mathrm{d}x \tag{51}$$

Se puede notar que la definición de la solución débil fuerza que la función $u(x,t)$ satisfaga la forma integral de la ecuación conservativa (49), mediante la función ϕ denominada *función de prueba* [1].

3.2.2. Problema de Riemann

El problema de Riemann es un problema de valores iniciales para cualquier ecuación diferencial en forma conservativa (ecuación 33), que exige:

$$u(x,0) = \begin{cases} u_L & x < 0 \\ u_R & x > 0 \end{cases} \tag{52}$$

Se considerarán los casos no triviales para este problema, si $u_L > u_R$ o $u_L < u_R$.

3.2.3. Caso $u_L > u_R$ del problema de Riemann

Este caso (apreciable en la figura 6) de condición inicial tiene como solución general de una ecuación de conservación (ecuación 33):

$$u(x,t) = \begin{cases} u_L & x < st \\ u_R & x > st \end{cases} \qquad \text{donde,} \qquad s = \frac{f(u_L) - f(u_R)}{u_L - u_R} \tag{53}$$

Donde s es la *velocidad de choque.* La expresión para s también es denominada *condición Rankine-Hugoniot* [1]. A continuación se demostrará que esta solución es una solución débil de la ecuación (33).

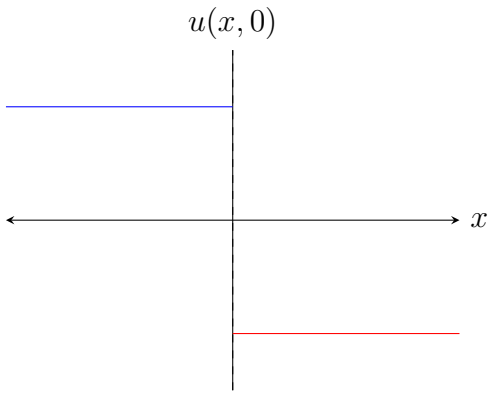


Figura 6: Gráfica de condición inicial del caso $u_L > u_R$

Prueba: Sea ϕ una función de prueba con soporte compacto U y que este último yace completamente en alguno de los dos conjuntos siguientes: $\{x < st\}$ o $\{x > st\}$. Dado que $u(x,t)$ es constante en ambos conjuntos mencionados, esta satisface la ecuación conservativa (49) y consecuentemente se satisface la condición 51[1].

Ahora se demostrará que $u(x,t)$ es la única solución débil de la ecuación 33. Suponiendo que el soporte U de ϕ está dividido por la recta $x = st$, de tal forma que el conjunto a la izquierda de la recta es U_L y a la derecha es U_R. Integrando, se obtiene

$$\int_0^\infty \int_{-\infty}^\infty \left(u\frac{\partial \phi}{\partial t} + f(u)\frac{\partial \phi}{\partial x} \right) \mathrm{d}x\,\mathrm{d}t = \int\int_{U_L} \left(u\frac{\partial \phi}{\partial t} + f(u)\frac{\partial \phi}{\partial x} \right) \mathrm{d}x\,\mathrm{d}t + \int\int_{U_R} \left(u\frac{\partial \phi}{\partial t} + f(u)\frac{\partial \phi}{\partial x} \right) \mathrm{d}x\,\mathrm{d}t \tag{54}$$

Usando el teorema de Green

$$\int\int_D \left(\frac{\partial P}{\partial x} - \frac{\partial Q}{\partial t}\right) \mathrm{d}x\,\mathrm{d}t = \int\int_{\partial D} P\,\mathrm{d}t + Q\,\mathrm{d}x \tag{55}$$

y dado que $u(x,t)$ es constante tanto en U_L como en U_R se tiene que:

$$\frac{\partial(u\phi)}{\partial t} = u\frac{\partial \phi}{\partial t} \tag{56}$$

$$\frac{\partial(f(u)\phi)}{\partial x} = f(u)\frac{\partial \phi}{\partial x} \tag{57}$$

Aplicando estas propiedades en la integración 54, se obtiene

$$\int_0^\infty \int_{-\infty}^\infty \left(u\frac{\partial \phi}{\partial t} + f(u)\frac{\partial \phi}{\partial x}\right) \mathrm{d}x\,\mathrm{d}t = \int\int_{U_L} \left(\frac{\partial(u\phi)}{\partial t} + \frac{\partial(f(u)\phi)}{\partial x}\right) \mathrm{d}x\,\mathrm{d}t + \int\int_{U_R} \left(\frac{\partial(u\phi)}{\partial t} + \frac{\partial(f(u)\phi)}{\partial x}\right) \mathrm{d}x\,\mathrm{d}t \tag{58}$$

$$\int_0^\infty \int_{-\infty}^\infty \left(u\frac{\partial \phi}{\partial t} + f(u)\frac{\partial \phi}{\partial x}\right) \mathrm{d}x\,\mathrm{d}t = \int_{\partial U_L} \phi\left(f(u_L)\,\mathrm{d}t - u_L\,\mathrm{d}x\right) + \int_{\partial U_R} \phi\left(f(u_R)\,\mathrm{d}t - u_R\,\mathrm{d}x\right) \tag{59}$$

$$\int_0^\infty \int_{-\infty}^\infty \left(u\frac{\partial \phi}{\partial t} + f(u)\frac{\partial \phi}{\partial x}\right) \mathrm{d}x\,\mathrm{d}t = \int_{x=st} \phi\left(\frac{f(u_L)}{s} - u_L - \frac{f(u_R)}{s} + u_R\right) \mathrm{d}x - \int_{-\infty}^0 \phi u_L\,\mathrm{d}x - \int_0^\infty \phi u_R\,\mathrm{d}x \tag{60}$$

$$\int_0^\infty \int_{-\infty}^\infty \left(u\frac{\partial \phi}{\partial t} + f(u)\frac{\partial \phi}{\partial x}\right) \mathrm{d}x\,\mathrm{d}t = \int_{x=st} \phi\left(\frac{f(u_L) - f(u_R)}{s} - (u_L - u_R)\right) \mathrm{d}x - \int_{-\infty}^\infty \phi(x,0)u(x,0)\,\mathrm{d}x \tag{61}$$

El integrando del primer término de la ecuación (61) se hace cero si y solo si $s = \frac{f(u_L)-f(u_R)}{u_L-u_R}$ y dado que esta integral debe ser cero por la condición 51, se demuestra que $u(x,t)$ definida en 53 es la única solución para el problema de Riemann en el caso $u_L > u_R$.

Si se sustituye el flujo de la ecuación de Burgers, se encuentra que la solución del problema de Riemann de esta ecuación, para este caso es

$$u(x,t) = \begin{cases} u_L & x < st \\ u_R & x > st \end{cases} \qquad \text{donde,} \qquad s = \frac{u_L + u_R}{2} \tag{62}$$

3.2.4. Caso $u_L < u_R$ del problema de Riemann

Hay múltiples soluciones débiles para este caso. Por ejemplo, se puede demostrar que la solución 62 satisface este caso también, tomando en cuenta que cambiará el signo de la velocidad de propagación s. Sin embargo esta solución no satisface la condición de entropía descrita por [1].

Definición 3.2. Una discontinuidad, propagándose en el espacio $x-t$, dada por la ecuación 53, satisface la condición de entropía si $f'(u_l) > s > f'(u_R)$.

Para la ecuación de Burgers, esta última condición equivale a $u_L > u_R$. Entonces la solución físicamente válida, o que satisface la condición de entropía, se encuentra resolviendo analíticamente el problema de Riemann para la ecuación de Burgers con viscosidad (ecuación 25) y luego en la

solución, hacer tender a cero el parámetro de viscosidad ν. Según [1], la solución débil que se obtiene es

$$u(x,t) = \begin{cases} u_L & x < u_L t \\ x/t & u_L t \leq x \leq u_R t \\ u_R & x > u_R t \end{cases}. \tag{63}$$

Esta solución se conoce como *rarefacción transónica*.

3.2.5. Cálculo del esquema de Godunov

El esquema de Godunov para encontrar el flujo entre celdas se consigue considerando las soluciones del problema de Riemann para la ecuación a resolver y evaluando que también se satisfagan las condiciones de entropía.

A partir de la ecuación conservativa discretizada en la forma de diferencia de flujos (ecuación 45) se puede notar que el flujo se define como

$$F_i^n = F(u_{i-1}^n, u_i^n) \tag{64}$$

entonces, Godunov incluye una función $u^*(u_L, u_R)$ que depende de las velocidades de las celdas vecinas a la interfaz en donde se quiere calcular el flujo, de tal manera que

$$F_i^n = f(u^*(u_{i-1}^n, u_i^n)) \tag{65}$$

La función $u^*(u_L, u_R)$ devuelve el valor correcto que tendrá la velocidad u en la interfaz, de acuerdo con la solución del problema de Riemann y la condición de entropía. Para calcular el valor de velocidad de u^* es preferible partir de los dos casos del problema de Riemann.

Recordando que para la ecuación de Burgers, $s = \frac{u_L + u_R}{2}$, se puede deducir u^*.

- Para el caso $u_L > u_R$ que produce **ondas de choque**, se tiene

$$u^* = \begin{cases} u_L & \frac{u_L+u_R}{2} > 0 \\ u_R & \frac{u_L+u_R}{2} < 0. \end{cases} \tag{66}$$

- Para el caso $u_L < u_R$ que produce **rarefacción transónica**, se tiene

$$u^* = \begin{cases} 0 & u_L < 0 < u_R \\ u_L & \frac{u_L+u_R}{2} > 0 \\ u_R & \frac{u_L+u_R}{2} < 0. \end{cases} \tag{67}$$

Se puede destacar que las opciones de u^* para el caso de ondas de choque y las últimas dos opciones del caso de rarefacción, se obtiene exactamente el mismo valor para u^*. La primera opción para el caso de rarefacción se deduce inmediatamente de la ecuación (63) [4].

Dado que el flujo F_i^n se define como una integral en el tiempo del flujo real $f(u)$ (ecuación 40), el cálculo de la velocidad que proporciona u^* debe considerar que al resolver el problema de Riemann la velocidad devuelta sea constante en un intervalo de tiempo Δt; esto da origen a la **condición de estabilidad**, también conocida como condición **CFL.** La condición de estabilidad para el marco de Godunov es la siguiente [4]

$$\left| u_{\max} \frac{\Delta t}{\Delta x} \right| \leq \frac{1}{2}. \tag{68}$$

Finalmente, a partir de la definición de u^*, se puede escribir el flujo numérico de la ecuación de Burgers bajo el esquema de Godunov:

$$\boxed{F(u_{i-1}^n, u_i^n) = \begin{cases} 0 & u_{i-1} < 0 < u_i \\ \frac{1}{2}u_{i-1}^2 & \frac{u_{i-1}+u_i}{2} > 0 \\ \frac{1}{2}u_i^2 & \frac{u_{i-1}+u_i}{2} < 0. \end{cases}} \tag{69}$$

3.3. Esquema de Roe

Al igual que el esquema de **Lax-Friedrichs**, el esquema de Roe es un **método aproximado** de la solución del problema de Riemann. Se diferencia principalmente del esquema de Godunov en que este último resuelve exactamente el problema de Riemann entre las celdas, mientras que Roe propone una aproximación de la solución basada en la linealización de la ecuación a resolver. Puede parecer poco motivador implementar el método de Roe para resolver la ecuación de Burgers, dado que ya se tiene el esquema de Godunov que aplica finamente la solución del problema de Riemann para obtener un método numérico. Sin embargo, resolver el problema de Riemann analíticamente es costoso, especialmente cuando las ecuaciones describen un modelo físico más realista y por ende, más complicado; es entonces en este tipo de situaciones en donde el método de Roe destaca más.

3.3.1. Linealización del problema

La ecuación de Burgers es no lineal, pero Roe propone resolver el problema de Riemann en las celdas linealizando la ecuación localmente. En otras palabras, Roe sugiere resolver la ecuación lineal

$$\frac{\partial u}{\partial t} + \bar{A}\frac{\partial u}{\partial x} = 0 \tag{70}$$

en dos celdas vecinas; donde $\bar{A}$ es una matriz **constante** que depende de los valores de u en cada celda vecina. En el caso de la ecuación de Burgers (o cualquier ecuación conservativa con una única función a resolver) $\bar{A}$ es solamente un escalar. Para construir $\bar{A}$ al aplicar el esquema de Roe para la ecuación de Burgers se necesitan satisfacer dos condiciones

1. Para cualquier u_i, u_{i+1} se tiene,

$$F_{i+1} - F_i = \bar{A} \cdot (u_{i+1} - u_i) \tag{71}$$

2. Cuando $u_j = u_{j+1} = u$ se tiene,

$$\bar{A}(u_j, u_{j+1}) = \bar{A}(u, u) = \frac{\partial F}{\partial u} = u. \tag{72}$$

La primera condición equivale a exigir que cuando se presenta una discontinuidad en el dominio, el flujo cambie de manera apropiada, dado que $\frac{\partial F}{\partial x} = \bar{A}\frac{\partial u}{\partial x}$. La segunda condición exige que en las

regiones continuas el valor de $\bar{A}$ devuelva la velocidad de onda; y para la ecuación de Burgers es u. Entonces se puede sustituir $\bar{A}$ por $\bar{u}$ en la ecuación 70 para obtener la ecuación de Burgers

$$\frac{\partial u}{\partial t} + \bar{u}\frac{\partial u}{\partial x} = 0, \tag{73}$$

así $\bar{u}$ se define tal que satisfaga los requerimientos anteriormente expuestos.

$$\bar{u}_{i+1} = \begin{cases} \frac{F_{i+1}-F_i}{u_{i+1}-u_i} & u_i \neq u_{i+1} \\ u_i & u_i = u_{i+1} \end{cases} \tag{74}$$

$$\bar{u}_{i+1} = \begin{cases} \frac{u_{i+1}+u_i}{2} & u_i \neq u_{i+1} \\ u_i & u_i = u_{i+1} \end{cases} \tag{75}$$

Roe consigue obtener una velocidad promedio utilizando la solución local lineal del problema de Riemann. De acuerdo a la condición Rankine-Hugoniot (53), se puede obtener la diferencia de flujos entre celdas [4]

$$F_{i+1} - F_i = \bar{u}_{i+1}\left(u_{i+1} - u_i\right) \tag{76}$$

Sin embargo, como se comentó anteriormente, la condición Rankine-Hugoniot solo devuelve la velocidad de propagación de una discontinuidad y por tanto, la anterior diferencia de flujos no es capas de diferenciar entre expansiones (caso $u_L > u_R$) y ondas de choque.

Para resolver este problema se debe considerar la **dirección** de la velocidad $\bar{u}$ de la onda, es decir, su signo. Si la velocidad es positiva, entonces la diferencia entre el flujo **en la celda** $i+1$, $f(u_{i+1})$ y el flujo en la interfaz a la derecha, F_{i+1} es

$$F_{i+1} - f(u_{i+1}) = \bar{u}_{i+1}(u_{i+1} - u_i). \tag{77}$$

Y si la velocidad $\bar{u}$ es negativa, se tiene que

$$f(u_i) - F_{i+1} = \bar{u}_{i+1}(u_{i+1} - u_i). \tag{78}$$

Ambas expresiones se pueden combinar, obteniendo

$$F_{i+1} = \frac{f(u_i) + f(u_{i+1})}{2} - \frac{1}{2}\left|\bar{u}_{i+1}\right|\left(u_{i+1} - u_i\right) \tag{79}$$

Junto a la definición de $\bar{u}_{i+1}$ en la ecuación 75, y recordando la notación $F_i = F(u^n_{i-1}, u^n_i)$ se obtiene el esquema de Roe

$$\boxed{F(u^n_{i-1}, u^n_i) = \frac{f(u^n_{i-1}) + f(u^n_i)}{2} - \frac{1}{2}\left|\bar{u}^n_i\right|\left(u^n_i - u^n_{i-1}\right)} \tag{80}$$

3.4. Ecuación de Burgers en una dimensión

3.4.1. Descripción del problema

Utilizando el método de volúmenes finitos y cada cada uno de los esquemas previamente explicados, se resolvió la ecuación de Burgers (6) en una dimensión, sujeta a los siguientes tipos de condiciones de frontera:

- **Fijas:** Los valores en las fronteras se mantienen constantes a lo largo de la evolución temporal

$$u(0,t) = u(0,0) \tag{81}$$

$$u(L,t) = u(L,0). \tag{82}$$

  Evidentemente, los valores de la frontera son los que toma la función de condición inicial $u(x,0)$.

- **Periódicas:** El dominio espacial es periódico. Esta condición se puede interpretar como el modelado de una tubería en forma de anillo en donde un fluido se propaga.

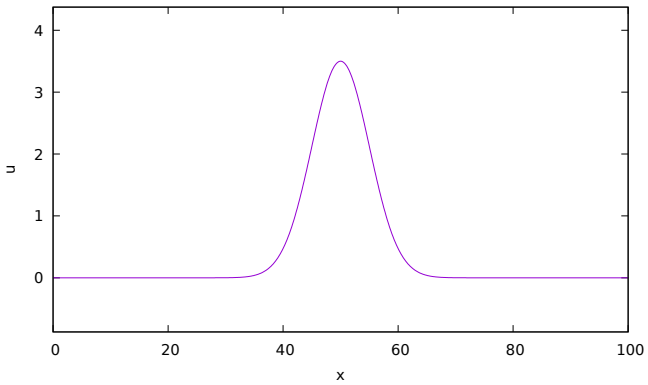
$$u(0,t) = u(L,t). \tag{83}$$

Como condiciones iniciales, se aplicaron varias funciones para examinar el funcionamiento de cada marco. A cada función se le asignó un nombre que fuera distintivo.

- `gauss`: Función gaussiana o curva de campana.

$$u(x,0) = A\exp\big(-b(x-\mu)^2\big). \tag{84}$$

  con A, b y μ constantes.

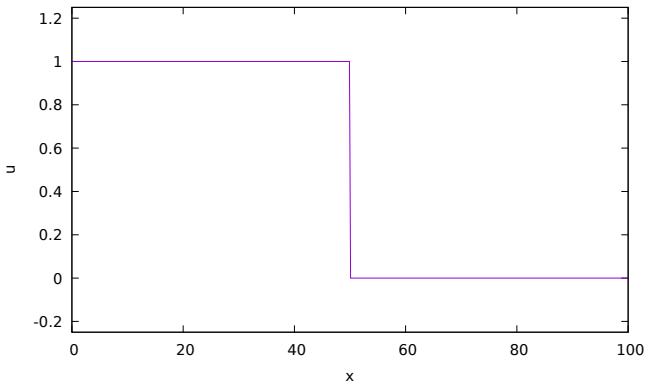


Gráfica de `gauss`.

- `step_neg`: Función Heaviside invertida respecto el eje horizontal

$$u(x,0) = \begin{cases} 1.0 & x < L/2 \\ 0.0 & x \geq L/2 \end{cases} \tag{85}$$

  donde L es el largo del dominio espacial.

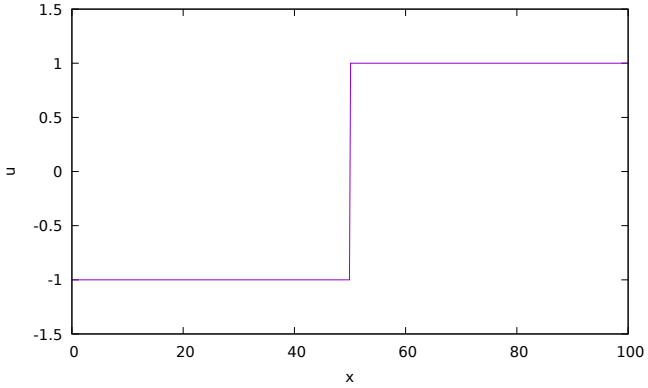


Gráfica de `step_neg`.

- `step_pos`: Similar a la función Heaviside, pero con -1.0 y 1.0 como sus valores constantes.

$$u(x,0) = \begin{cases} -1.0 & x < L/2 \\ 1.0 & x \geq L/2 \end{cases} \tag{86}$$

  donde L es el largo del dominio espacial.



Gráfica de `step_pos`.

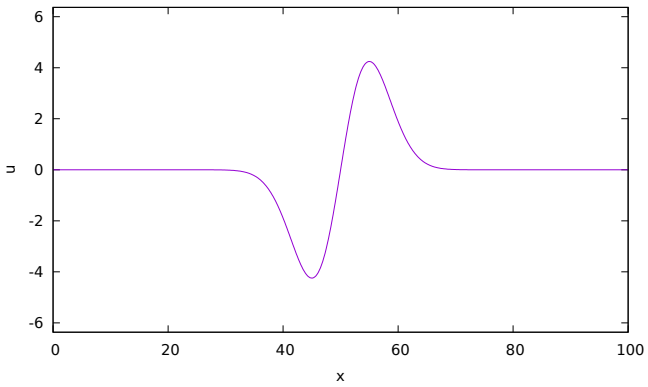
- `gauss_impar`: Multiplicación de una función lineal por una campana gaussiana, para obtener una función impar con dos pulsos.

$$u(x,0) = c\left(x - \frac{L}{2}\right) \cdot A\exp\left(-b(x-\mu)^2\right) \tag{87}$$

  con c constante y las mismas constantes definidas para `gauss`.

3.4.2. Implementación del método

La implementación de la solución numérica de la ecuación de Burgers con el método de volúmenes finitos tiene una estructura central muy similar a la implementación del método de diferencias finitas. Sin embargo, dado que el método de volúmenes finitos depende de la elección del esquema utilizado, el programa que se diseñó para ejecutar el método incluye la implementación de los



Gráfica de `gauss_impar`.

tres esquemas; esto para que se pudiera visualizar y comprobar cuál esquema era más preciso al producir la solución

Variables generales de la simulación

```
// Parámetros temporales
const double t_total = 30; // Tiempo total en segundos
const double dt = 0.001; // Tamaño de paso temporal en segundos
int Niter = floor(t_total/dt); // Número total de iteraciones
const int num_outs = 1000; // 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

// 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
```

Se definieron cadenas de caracteres (*strings*) que identificaran a cada uno de los esquemas (marcos) a utilizar. El string `roe-fix` será explicado en la discusión de resultados.

Arreglo de strings con nombres de los esquemas

```
string marcos[] = {"godunov", "roe", "LF", "roe-fix"};
int num_marcos = sizeof(marcos) / sizeof(marcos[0]);
```

Se utilizaron punteros a arreglos unidimensionales, así como en las anteriores simulaciones descritas,

Definición e inicialización de punteros a arreglos

```
// 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];
    // Variables de control
    double umax = 0;
    double umin = 0;

    // 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++)
    {
        if (funcion_inicial == "step_neg")
        {
            u[i] = step_neg(x[i]);
        }
        else if (funcion_inicial == "step_pos")
        {
            u[i] = step_pos(x[i]);
        }
        else if (funcion_inicial == "gauss")
        {
            u[i] = gauss(x[i]);
        }
        else if (funcion_inicial == "gauss_impar")
        {
            u[i] = gauss_impar(x[i]);
        }
        else if (funcion_inicial == "gauss_neg")
        {
            u[i] = -gauss(x[i]);
        }
```

Las funciones utilizadas al inicializar el arreglo u fueron definidas de la siguiente forma

Definición de funciones de condición inicial

```
#include <cmath>
double gauss(double x)
{
    double b = 0.02;
    double mu = 50;
    double A = 3.5;
    return A*exp(-b*pow(x - mu,2));
}

double step_neg(double x)
{
```

```
    double L = 100;
    double arg = x - L/2;
    if (arg < 0)
    {
        return 1.0;
    }
    else
    {
        return 0.0;
    }
}

double step_pos(double x)
{
    double L = 100;
    double arg = -(x - L/2);
    if (arg < 0)
    {
        return 1.0;
    }
    else
    {
        return -1.0;
    }
}

double gauss_impar(double x)
{
    double c = 0.4;
    double L = 100;
    return c*(x-L/2)*gauss(x);
```

Para aplicar todos los esquemas que fueron estudiados, se diseñó la función `Flujo` de tal manera que esta pudiese ser aplicada de acuerdo al marco a utilizar.

Definición de función que calcula el flujo entre celdas

```
/**
 * @brief Calcula el flujo por interfaz según el marco utilizado
 *
 * @param u Velocidad a la izquierda de la interfaz
 * @param v Velocidad a la derecha de la interfaz
 * @param marco Tipo de marco numérico usado: godunov, roe
 * o LF (Lax-Friedrichs)
 * @param dx Tamaño de paso en x
 * @param dt Tamaño de paso temporal
 * @return double: Flujo promedio en la i-ésima interfaz
 */
double Flujo(double u,
```

```
                double v,
                const string &marco,
                double dx,
                double dt)
{
    // Marco de Godunov
    if (marco == "godunov")
    {
        return FlujoBurgers(uPrime(u, v));
    }
    // Marco de Lax-Friedrichs
    else if (marco == "LF")
    {
        double FlujoPromedio = 0.5*(FlujoBurgers(u)+FlujoBurgers(v));
        return FlujoPromedio-0.5*dx/dt*(v-u);
    }
    // Marco de Roe
    else if (marco == "roe")
    {
        double FlujoPromedio = 0.5*(FlujoBurgers(u)+FlujoBurgers(v));
        return (FlujoPromedio-0.5*abs(uProm(u,v))*(v-u));
    }
```

La función `FlujoBurgers` calcula el flujo exacto $f(u) = \frac{1}{2}u^2$, ya que todos los esquemas necesitan esta función para calcular el flujo entre celdas. En el caso del esquema de Godunov, se define la función `uPrime` la cual es la implementación de la función definida en las ecuaciones 66 y 67.

Definición de función `uPrime`

```
/**
 * @brief Velocidad a usar para el flujo del marco de Godunov
 *
 * @param u Velocidad a la izquierda
 * @param v Velocidad a la derecha
 * @return double: velocidad elegida
 */
double uPrime(double u, double v)
{
    if (u > v)
    {
        if ((u+v)/2 > 0)
        {
            return u;
        }
        else return v;
    }
    else if (v > u )
    {
        if (u > 0)
```

```
        {
            return u;
        }
        else if (v < 0)
        {
            return v;
        }
        else if ((v > 0) && (u < 0))
        {
            return 0;
        }
        else return 0;
    }
    else return u;
}
```

En el caso del esquema de Roe, la función `uProm` calcula el promedio entre las velocidades de las celdas vecinas

Definición de función `uProm`

```
/**
 * @brief Velocidad promedio entre celdas para el flujo del
 * marco de Roe.
 *
 * @param u Velocidad a la izquierda. También denotada u_{i}
 * @param v Velocidad a la derecha, o u_{i+1}
 * @return double: Velocidad a usar
 */
double uProm(double u, double v)
{
    if (u != v)
    {
        return 0.5*(u + v);
    }
    else
        return u;
}
```

Para aplicar las condiciones de frontera se optó por definir una función de tipo `void` que modificara los arreglos de acuerdo al tipo de condición que se quisiera aplicar

Definición de función `condicion_frontera`

```
/**
 * @brief Aplica condiciones de frontera a la función u pasada como puntero
 *
 * @param u Puntero al arreglo con los valores que toma la función u
 * @param u_nueva Puntero al arreglo con los valores de la función u
 * en un instante dt después
 * @param N Tamaño del arreglo u
```

```
 * @param dt Tamaño de paso en t
 * @param dx Tamaño de paso en x
 * @param tipo Tipo de condición: periódica o fija
 * @param marco Marco numérico utilizado: roe, godunov o LF
 *
 */
void condicion_frontera(double *u, double *u_nueva, int N,
                        double dt, double dx, const string &tipo,
                        const string &marco)
{
    if (tipo == "fija")
    {
        u_nueva[0] = u[0];
        u_nueva[N-1] = u[N-1];
    }
    else
    {
        u_nueva[0] = u[0] -(dt/dx)*
            (Flujo(u[0], u[1], marco, dx, dt)-
             Flujo(u[N-1], u[0], marco, dx, dt));

        u_nueva[N-1] = u[N-1] -(dt/dx)*
            (Flujo(u[N-1], u[0], marco, dx, dt)-
             Flujo(u[N-2], u[N-1], marco, dx, dt));
    }
}
```

Por último, en el ciclo principal de integración se implementó la relación definida en la ecuación 41, junto con las funciones explicadas previamente

Ciclo principal de integración

```
 * en un instante dt después
 * @param N Tamaño del arreglo u
 * @param dt Tamaño de paso en t
 * @param dx Tamaño de paso en x
 * @param tipo Tipo de condición: periódica o fija
 * @param marco Marco numérico utilizado: roe, godunov o LF
 *
 */
void condicion_frontera(double *u, double *u_nueva, int N,
                        double dt, double dx, const string &tipo,
                        const string &marco)
{
    if (tipo == "fija")
    {
        u_nueva[0] = u[0];
        u_nueva[N-1] = u[N-1];
    }
```

```
    else
    {
        u_nueva[0] = u[0] -(dt/dx)*
            (Flujo(u[0], u[1], marco, dx, dt)-
             Flujo(u[N-1], u[0], marco, dx, dt));

        u_nueva[N-1] = u[N-1] -(dt/dx)*
            (Flujo(u[N-1], u[0], marco, dx, dt)-
             Flujo(u[N-2], u[N-1], marco, dx, dt));
    }
}
```

4. Método de elementos finitos

5. Conclusiones

Referencias

[1] M. Cameron. *Notes on Burger's Equation*. Universidad de Maryland. 2016.

[2] 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.

[3] Randall J. LeVeque. "Nonlinear Conservation Laws and Finite Volume Methods". En: ed. por O. Steiner y A. Gautschy. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, págs. 1-159. ISBN: 978-3-540-31632-9. DOI: 10.1007/3-540-31632-9_1. URL: https://doi.org/10.1007/3-540-31632-9_1.

[4] R.H. Pletcher, J.C. Tannehill y D. Anderson. *Computational Fluid Mechanics and Heat Transfer, Second Edition*. Series in Computational and Physical Processes in Mechanics and Thermal Sciences. Taylor & Francis, 1997. ISBN: 9781560320463. URL: https://books.google.com.gt/books?id=ZJPbtHeilCgC.