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.

28 de febrero 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.

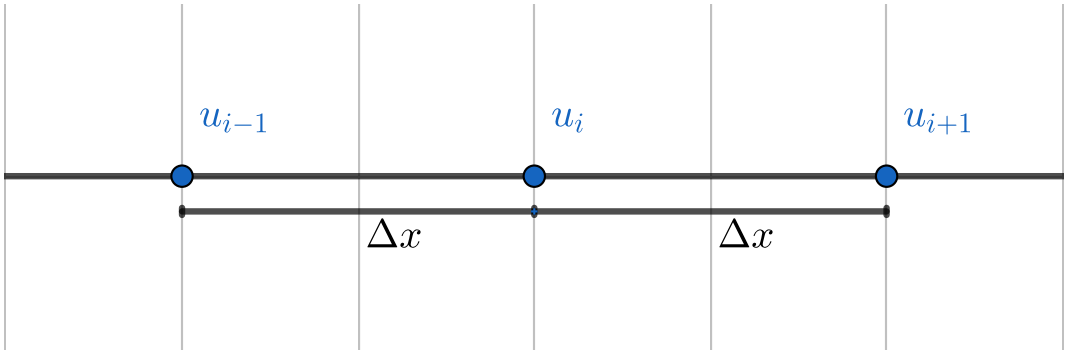

Figura 1: Discretización del dominio espacial

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 más futuro 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 presentes 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

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 puede ser interpretada 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:

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

Aplicación del método y código implementado [2]

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

```
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];

// Inicialización de arreglos
for (int i = 0; i < Nx; i++)
```

[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`.

[2] Código completo disponible en el siguiente enlace

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

void salida(ofstream &of, double *u, double *x, double t, int N)
```

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 aplicó 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 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[3]. 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;
}
```

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.

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

Se utilizó una función especial llamada `salida` que toma como argumento una variable de tipo `ofstream` que 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. 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.