
Instituto Tecnológico de Costa Rica

Métodos numéricos

Profesor: Alfredo Rodríguez

Examen 2

Estudiantes

Dávila Rodríguez Stephanny 2019033581

Rodríguez Morales Marco 2019163031

Rodríguez Rivas Daniel 2019039694

Problema 1)

1-A [5 puntos]
En la figura 1 se presenta el grafico discretizado del espacio de trabajo que se describe en el problema 1. Para el eje de distancia (x), se usaron 21 puntos igualmente espaciados, por lo que se uso un $dx = 0.5\ cm$. Para el eje de tiempo se utilizó un $dt = 5\ s$, y solo se presentaron los 10 puntos iniciales, tal y como decia la instruccion. Para las condiciones de borde se considera que las dos placas se encuentran inmoviles, se tiene un tiempo inicial de $t = 0\ s$, y una velocidad inicial para la placa superior de $v = 9\ cm/s$.

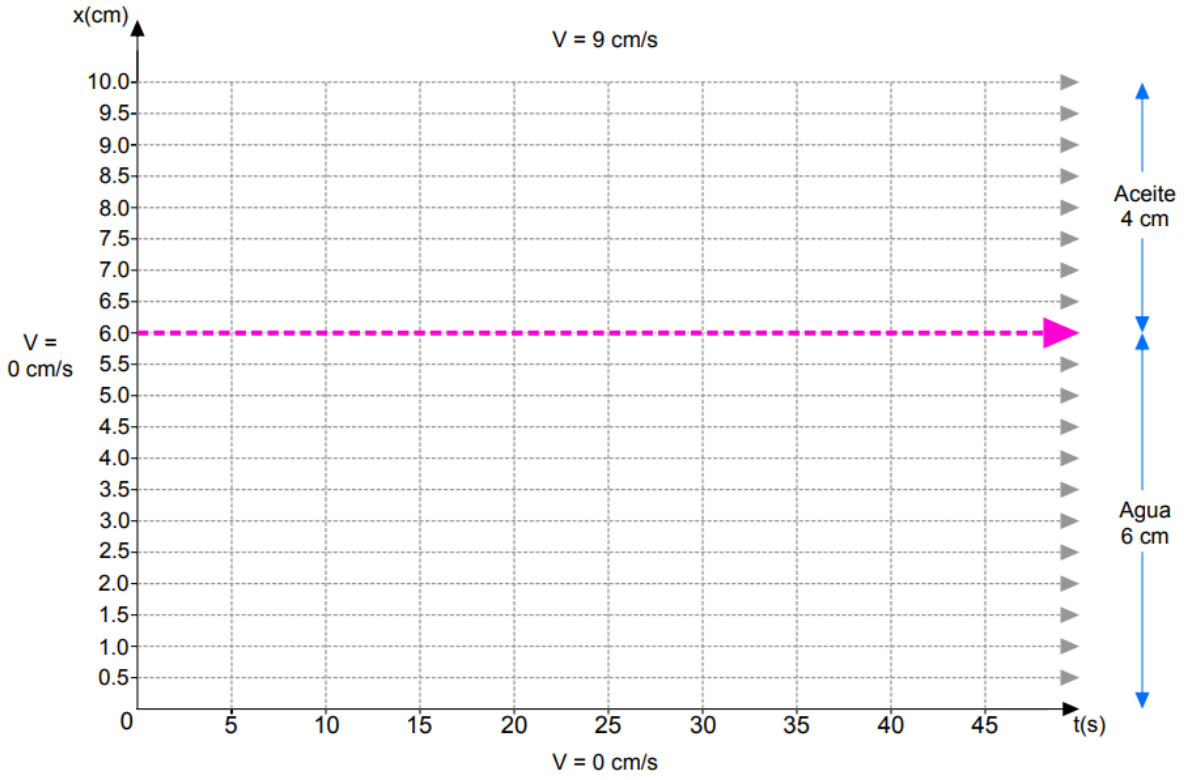


Figura 1. Gráfico discretrizado del espacio de trabajo.

1-B [5 puntos]

Las ecuaciones que representan la velocidad de los fluidos son ecuaciones diferenciales de tipo parabólico, cuya forma se puede identificar en la ecuación 1.

$$\frac{\partial v}{\partial t} = k\frac{\partial^2 v}{\partial x^2} \qquad (1)$$

Resolver ecuaciones parabólicas implica reemplazar las derivadas parciales con diferencias finitas, teniendo en cuenta variaciones tanto en el tiempo como en el espacio. En este contexto, se aproximó la ecuación utilizando el método explícito. Para ello, se comenzó representando la segunda derivada en el espacio mediante una diferencia finita centrada. El resultado es la ecuación 2, donde "$i$" denota el dato de distancia y "$l$" el dato de tiempo:

$$\frac{\partial^2 v}{\partial x^2} = \frac{v_{i+1}^{l} - 2 * v_i^{l} + v_{i-1}^{l}}{\Delta x^2} \quad (2)$$

Después, se estima la derivada con respecto al tiempo mediante una diferencia finita hacia adelante, lo que conduce a la ecuación 3.

$$\frac{\partial v}{\partial t} = \frac{v_i^{l+1} - v_i^{l}}{\Delta t} \quad (3)$$

En última instancia, se insertan las ecuaciones 2 y 3 en la ecuación 1, logrando así la aproximación definitiva para la ecuación de velocidad de los fluidos, que se representa en la ecuación 4, para el metodo explicito.

$$v_i^{l+1} = v_i^{l} + \lambda * \left(v_{i+1}^{l} - 2 * v_i^{l} + v_{i-1}^{l}\right); \lambda = \frac{\Delta t}{(\Delta x)^2} \quad (4)$$

Para el metodo implicito, la velocidad en la frontera de los fluidos será:

$$-\lambda * v_{i-1}^{l+1} + (1 + 2 * \lambda) * v_i^{l+1} = v_i^{l} + \lambda * f_{i+1} * \left(t^{l+1}\right); \lambda = \frac{k*\Delta t}{(\Delta x)^2} \quad (5)$$

Para los puntos intermedios usando el metodo implicito se usa:

$$-\lambda * v_{i-1}^{l+1} + (1 + 2 * \lambda) * v_i^{l+1} - \lambda * v_{i+1}^{l+1} = v_i^{l}; \lambda = \frac{k*\Delta t}{(\Delta x)^2} \quad (6)$$

Al emplear la ecuación 4 y sustituir los datos proporcionados en el problema, se derivan las aproximaciones para agua y oil, representadas en las ecuaciones 7 y 8, respectivamente.

$$v_{(H2O)_i}^{l+1} = v_i^{l} + 0.20 * \left(v_{i+1}^{l} - 2 * v_i^{l} + v_{i-1}^{l}\right) \quad (7)$$

$$v_{(Oil)_i}^{l+1} = v_i^{l} + 0.67 * \left(v_{i+1}^{l} - 2 * v_i^{l} + v_{i-1}^{l}\right) \quad (8)$$

En el caso de las ecuaciones que representan el comportamiento de la velocidad de los fluidos en la interfaz, no es posible aproximarse utilizando la ecuación 4. Esto se debe a que, aunque dicha ecuación utiliza la diferencia finita centrada, que es más precisa, emplea los puntos $i + 1$ e $i - 1$, cuando lo que se busca es utilizar únicamente el punto $i$ para determinar la velocidad justo en la interfaz. Para la aproximación en este caso, se utiliza una diferencia finita, ya sea hacia adelante o hacia atrás, y el resultado se presenta en la ecuación 9:

$$\mu_{H20} * \frac{v_{(H2O)_i}^{l} - v_{(H2O)_{i-1}}^{l}}{\partial x} = \mu_{Oil} * \frac{v_{(Oil)_{i+1}}^{l} - v_{(Oil)_i}^{l}}{\partial x} \quad (9)$$

Dado que es necesario conocer la velocidad de ambos fluidos en el punto de la interfaz, podemos considerar $v_{(H2O)_i}^{l}$ y $v_{(Oil)_i}^{l}$ como una única variable denominada $v_B$. Teniendo en cuenta esto y sabiendo que $\partial x = 1$, se obtiene la ecuación 10.

$$\mu_{H20} * \left(v_B - v_{H2O}\right) = \mu_{Oil} * \left(v_{Oil} - v_B\right) \quad (10)$$

Al resolver la ecuación 10 para despejar vB, se obtiene la aproximación, para la velocidad del oil y el agua en la interfaz, como se presenta en la ecuación 12.

$$v_B = \frac{\mu_{H20} * v_{(H2O)_{i-1}} + \mu_{Oil} * v_{(Oil)_{i+1}}}{\mu_{H20} + \mu_{Oil}} \quad (11)$$

$$\frac{\mu_{H20}}{\left(\mu_{H20} + \mu_{Oil}\right)} * v_{H2O} - v_B - \frac{\mu_{Oil}}{\left(\mu_{H20} + \mu_{Oil}\right)} * v_{Oil} = 0 \quad (12)$$

1-C [45]

Se realiza un código que implementa la resolución del problema planteado en la primera pregunta del desarrollo.

Este programa primero pide los siguientes parámetros de entrada:

k1: el valor de la viscosidad dinámica de la primer sustancia

k2 valor de la viscosidad dinámica de la segunda sustancia

x: valor de la distancia entre láminas

n: número de intervalos deseados

t: tiempo total que se desea calcular

dt: tamaño del intervalo de tiempo

v0: velocidad inicial de la placa superior

i: distancia a la cuál se encuentra la interfaz entre sustancias

Luego se definió la precisión y las iteraciones máximas a las que debe de llegar el método Gauss-Seidel para detenerse, por enunciado estas deben de ser 100 iteraciones y una precisión de 0,05.

Luego, se verificó que las condiciones de tiempo y de distancia que se ingresaron cumplieran las condiciones necesarias para poder ser utilizadas.

En las siguiente imágenes se puede apreciar la forma en la que se programó este segmento:

```
function examen2()
    % Se piden los valores de entrada
    k1 = input('Ingrese el valor de la viscosidad dinámica de la primer sustancia (k1): ');
    k2 = input('Ingrese el valor de la viscosidad dinámica de la segunda sustancia (k2): ');
    x = input('Ingrese el valor de la distancia entre láminas (x): ');
    n = input('Ingrese el número de intervalos deseados (n): ');
    t = input('Ingrese el tiempo total que se desea calcular (t): ');
    dt = input('Ingrese el tamaño del intervalo de tiempo (dt): ');
    v0 = input('Ingrese la velocidad inicial de la placa superior: ');
    i = input('Ingrese la distancia a la cuál se encuentra la interfaz entre sustancias (i): ');

    precision = 0.05;
    itermax = 100;

    % Se verifica que las condiciones necesarias de tiempo se cumplan:
    dx = x / n;
    dt_min_1 = 0.5 * dx / k1;
    dt_min_2 = 0.5 * dx / k2;

    if dt_min_1 < dt || dt_min_2 < dt
        disp('El dt seleccionado no es válido.');
        return;
    end
```

Figura 2. Ingreso de datos por parte del usuarios y verificación de las condiciones de tiempo.

```
% Se verifica que las condiciones necesarias de distancia se cumplan:
if rem(n, x) ~= 0 || rem(i, dx) ~= 0
    disp('El n seleccionado no es válido para la distancia x seleccionada.');
    return;
end
```

Figura 3. Verificación de las condiciones de distancia.

Luego de esto, se definen las constantes que hacen falta, las cuales son T (el vector de tiempos) y X (el vector de posiciones).
Seguidamente, se inició el cálculo de lambda1 y lambda2 para evitar recalcularlas en funciones sucesivas.

```
% Se calculan las constantes faltantes:
X = 0:dx:x; % Vector de posiciones

% Se crea el vector de tiempos
T = 0:dt:t;

lambda1 = k1 * dt / dx;
lambda2 = k2 * dt / dx;
```

Figura 4. Verificación de las condiciones de distancia.

Posteriormente se inició la resolución del problema utilizando el método Gauss-Seidel, para esto primero se llama a la función “tic” para tomar el tiempo de ejecución de la función, luego se llama a la función “calcularVelocidadesGaussSeidel”, la cual contiene todos los pasos necesarios para el cálculo mediante este método.
Cuando se termina de ejecutar, se tiene que la variable “tiempo” guarda el tiempo de ejecución de la función, este tiempo se imprime en la terminal. Este mismo esquema de programación se repite para llamar la función que soluciona el problema mediante “calcularVelocidadesLU”.

Por último se grafican los resultados obtenidos de las funciones anteriores.
Todo lo mencionado anteriormente se puede observar en la siguiente imagen.

```
% Se realiza el cálculo de velocidades utilizando Gauss-Seidel
tic;
Mr_GS = calcularVelocidadesGaussSeidel(X, T, v0, i, lambda1, lambda2, precision, itermax);
tiempo = toc; % Detener temporizador y almacenar tiempo de ejecución
disp('Tiempo de ejecución Gauss-Seidel:');
disp(tiempo);

vb = 0;

% Se realiza el cálculo de velocidades utilizando Factorización LU
tic;
Mr_LU = calcularVelocidadesLU(X, T, v0, i, lambda1, lambda2);
tiempo = toc; % Detener temporizador y almacenar tiempo de ejecución
disp('Tiempo de ejecución LU:');
disp(tiempo);

% Se grafican los resultados
graficarResultados(X, Mr_GS, T, 'Gauss-Seidel');
graficarResultados(X, Mr_LU, T, 'Factorización LU');
```

Figura 5. Llamado de las funciones para resolver el problema mediante Gauss-Seidel y LU y gráfico de la respuesta.

Ahora bien, la función “calcularVelocidadesGaussSeidel” implementa dentro de ella un ciclo “for” que permite llegar al límite de 100 iteraciones, al mismo tiempo se tiene un condicional que sale de este ciclo si se llega a la precisión deseada antes de terminar las iteraciones. Al mismo tiempo, dentro de esta función se llama a una nueva función que calcula la nueva iteración. Esta nueva iteración se calcula en función de la posición y toma en cuenta el límite entre ambos fluídos.
En las siguientes figuras se puede apreciar el código implementado.

```
function [Mr] = calcularVelocidadesGaussSeidel(X, T, v0, i, lambda1, lambda2, precision, itermax)
    % Inicialización de la matriz de resultados
    Mr = zeros(length(T), length(X));
    Mr(1, end) = v0; % Condición inicial

    %tiempo = zeros(1, length(T)); % Vector para almacenar el tiempo de ejecución

    % Cálculo de velocidades utilizando Gauss-Seidel con parámetros adicionales
    for i_tiempo = 2:length(T)
        for iter = 1:itermax
            Mr(i_tiempo, :) = calcularNuevaIteracionGaussSeidel(Mr(i_tiempo - 1, :), X, i, lambda1, lambda2);
            % Verificar convergencia
            if norm(Mr(i_tiempo, :) - Mr(i_tiempo - 1, :)) < precision
                break;
            end
        end

    end
end
```

Figura 6. Implementación de la función “calcularVelocidadesGaussSeidel”

```
function nuevo = calcularNuevaIteracionGaussSeidel(anterior, X, i, lambda1, lambda2)
    nuevo = anterior;
    for i_longitud = 2:length(X)-1
        if i_longitud < i / X(2) + 1
            derecha = nuevo(i_longitud + 1);
            izquierda = nuevo(i_longitud - 1);
            nuevo(i_longitud) = nuevo(i_longitud) + lambda1 * (izquierda - 2 * nuevo(i_longitud) + derecha);
        elseif i_longitud == i / X(2) + 1
            derecha = nuevo(i_longitud + 1);
            izquierda = nuevo(i_longitud - 1);
            nuevo(i_longitud) = (derecha + (lambda1 / lambda2) * izquierda) / (1 + lambda1 / lambda2);
        elseif i_longitud > i / X(2) + 1
            derecha = nuevo(i_longitud + 1);
            izquierda = nuevo(i_longitud - 1);
            nuevo(i_longitud) = nuevo(i_longitud) + lambda2 * (izquierda - 2 * nuevo(i_longitud) + derecha);
        end
    end
end
```

Figura 7. Implementación de la función “calcularNuevaIteracionGaussSeidel”

La función “calcularVelocidadesLU” resuelve el sistema de ecuaciones. Esta función toma en cuenta los valores de borde, la posición en la que está (según la posición decide si realizar los cálculos con lambda1 o lambda2, además de utilizar vb = 9 o vb = 0).
Esta misma función llama a las subfunciones matriz() y vector() para crear la matriz y el vector que se utilizará cuando se aplique esta función.
Se puede observar en las siguientes imágenes su aplicación:

```
function Mr = calcularVelocidadesLU(X, T, v0, i, lambda1, lambda2)
    % Inicialización de la matriz de resultados
    Mr = zeros(length(T), length(X));
    Mr(1, end) = v0; % Condición inicial

    for i_tiempo = 2:length(T)
        for i_longitud = 1:length(X)
            if i <= i_longitud
                lambda = lambda2;
                vb = 9;
            else
                lambda = lambda1;
                vb = 0;

            end
            mat1 = matriz(lambda1, lambda2, length(X), i);
            [L, U] = factLU(mat1);

            b_anterior = vector(lambda, Mr(i_tiempo - 1, :), matriz(lambda1, lambda2, length(X), i), vb);

            y = sust_adelante(L, b_anterior');
            x = sust_atras(U, y);
            Mr(i_tiempo, :) = x';
            a = Mr;
        end
    end
```

Figura 8. Función “calcularVelocidadesLU”

```
function M = matriz(lambda1, lambda2, nx, i)
    % Función para generar la matriz
    M = zeros(nx, nx);

    for j = 1:nx
        if j < i
            M(j, j) = 1 + 2 * lambda1;
        elseif j == i
            M(j, j) = 1 + lambda1 / lambda2;
        else
            M(j, j) = 1 + 2 * lambda2;
        end

        if j > 1
            M(j, j - 1) = -lambda1 * (j < i) - lambda2 * (j >= i);
        end

        if j < nx
            M(j, j + 1) = -lambda1 * (j < i) - lambda2 * (j >= i);
        end
    end
end
```

Figura 9. Función “matriz”

```
function b = vector(lambda, prev, vb)
    % Función para generar el vector de constantes

    n = length(prev);
    b = zeros(size(prev));

    for p = 1:n
        if p == 1
            b(p) = 0;
        elseif p==2
            b(p) = lambda * vb + prev(p);
        elseif p == n-1
            b(p) = lambda * vb + prev(p);
        elseif p == n
            b(p) = 9;
        else
            b(p) = prev(p);
        end
    end
end
```

Figura 10. Función “vector”

A continuación se presentan las imágenes generadas para tiempos de 300, 1000 y 2000 segundos.

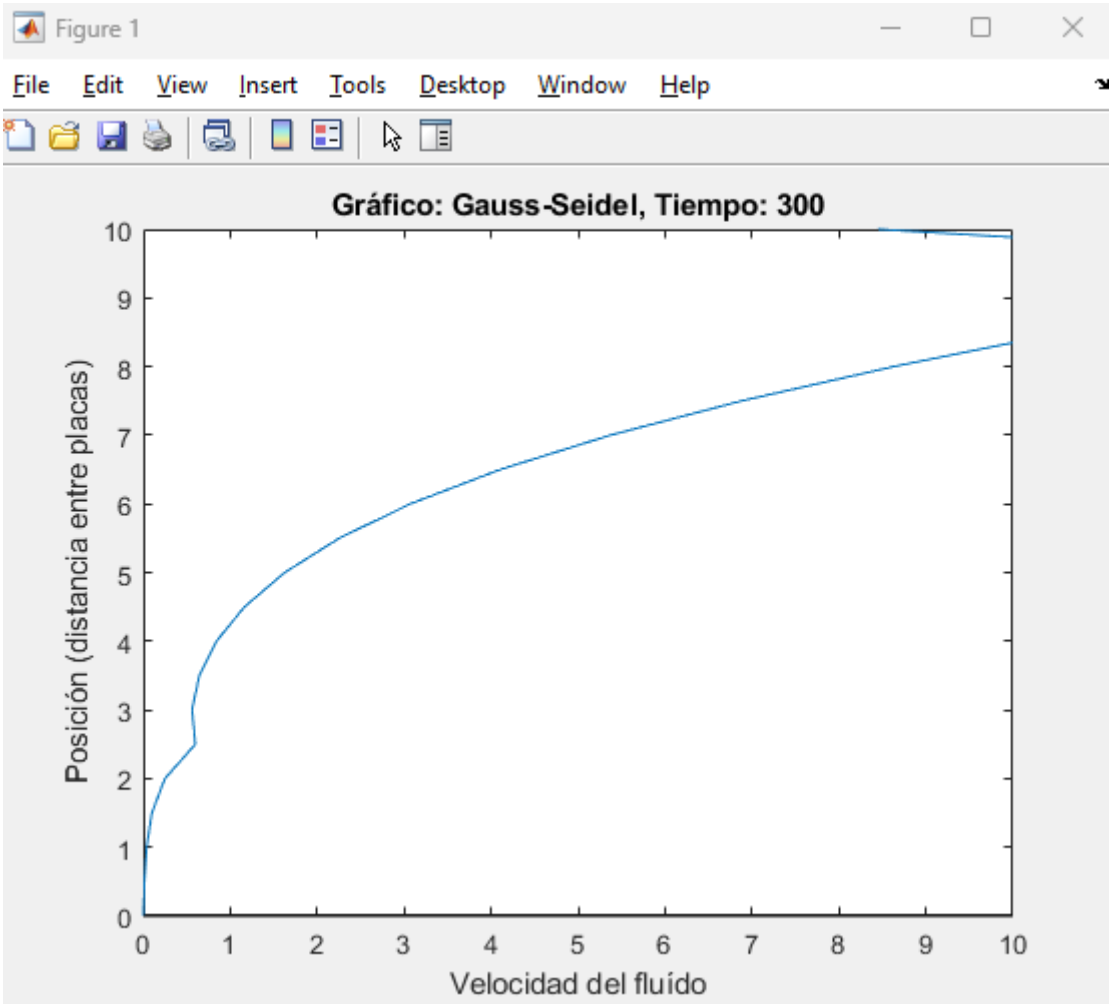


Figura 11. Factorización Gauss-Seidel para T = 300s

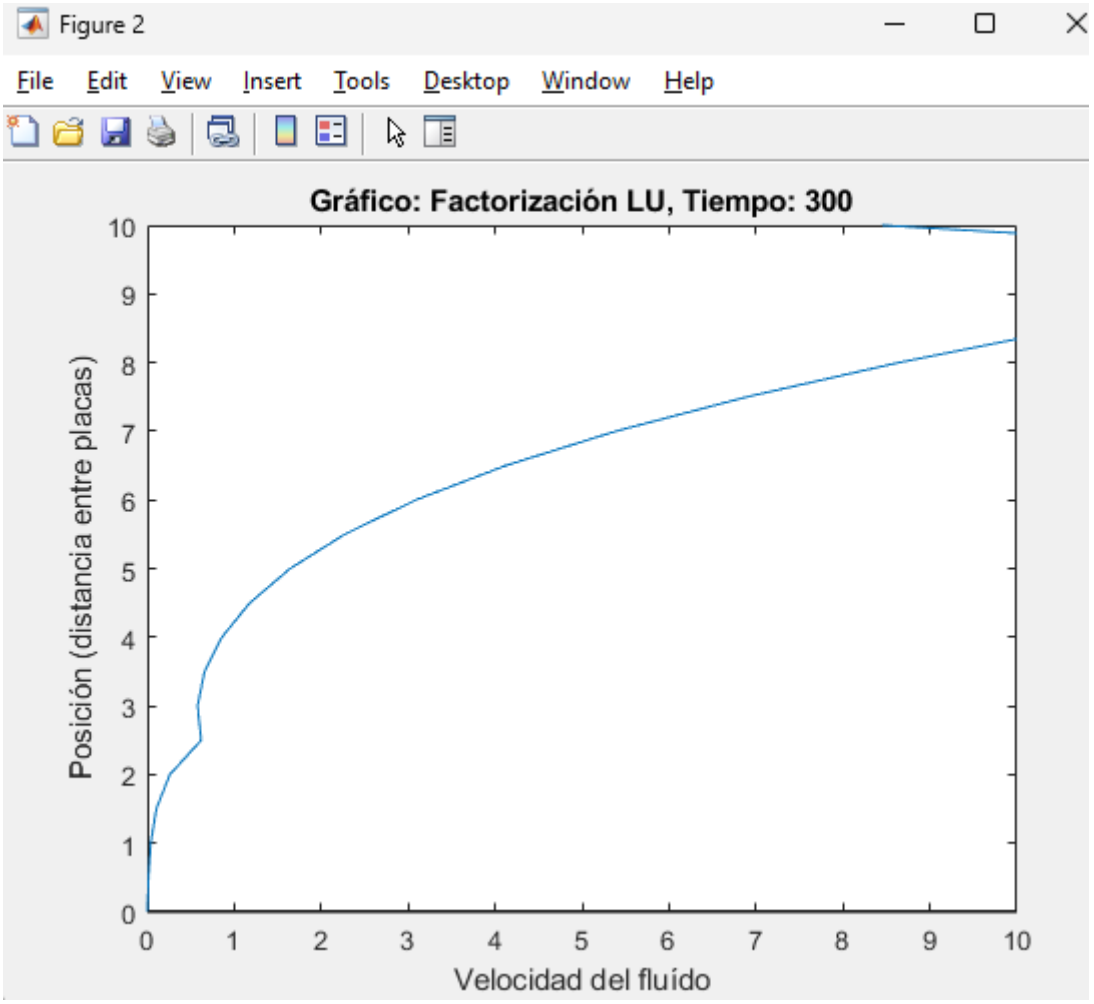


Figura 12. Factorización LU para T = 300s

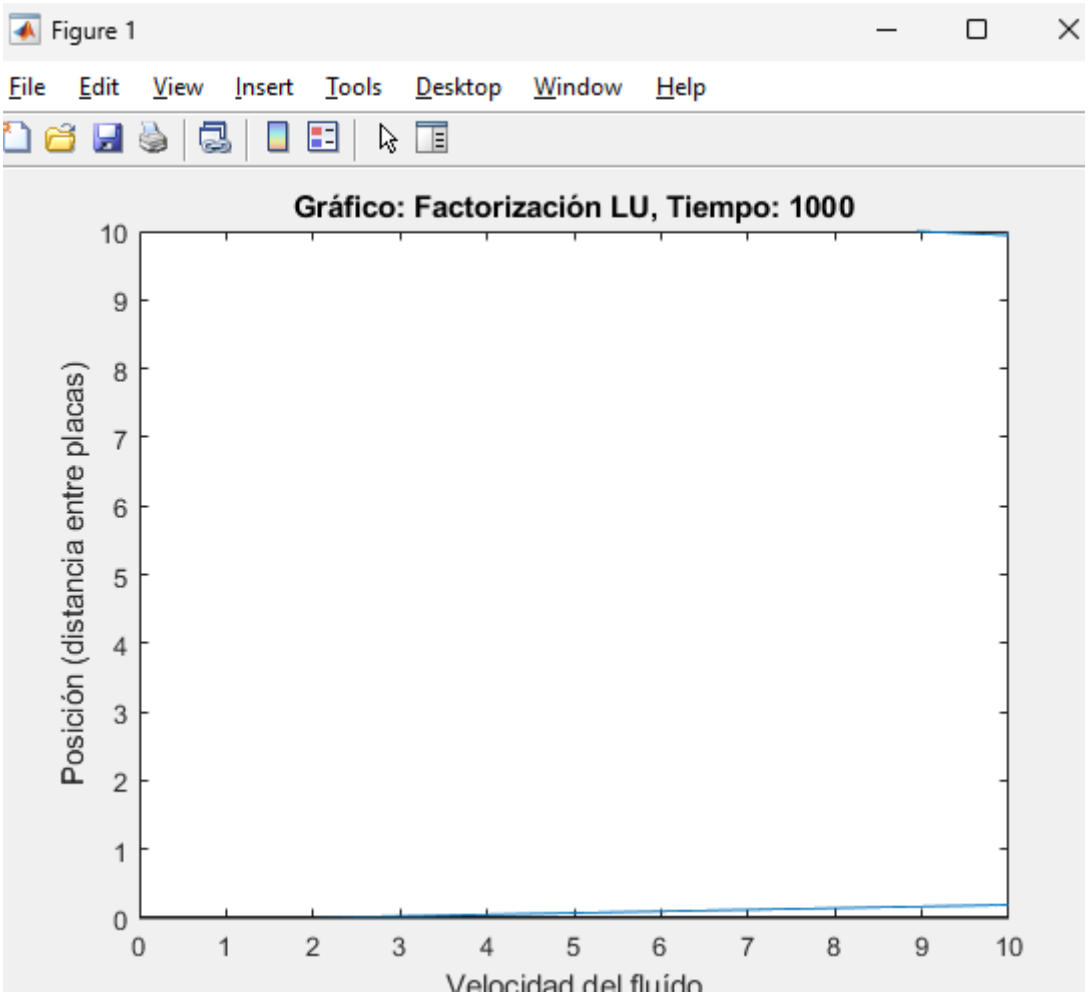


Figura 13. Factorización Gauss-Seidel para T = 1000s

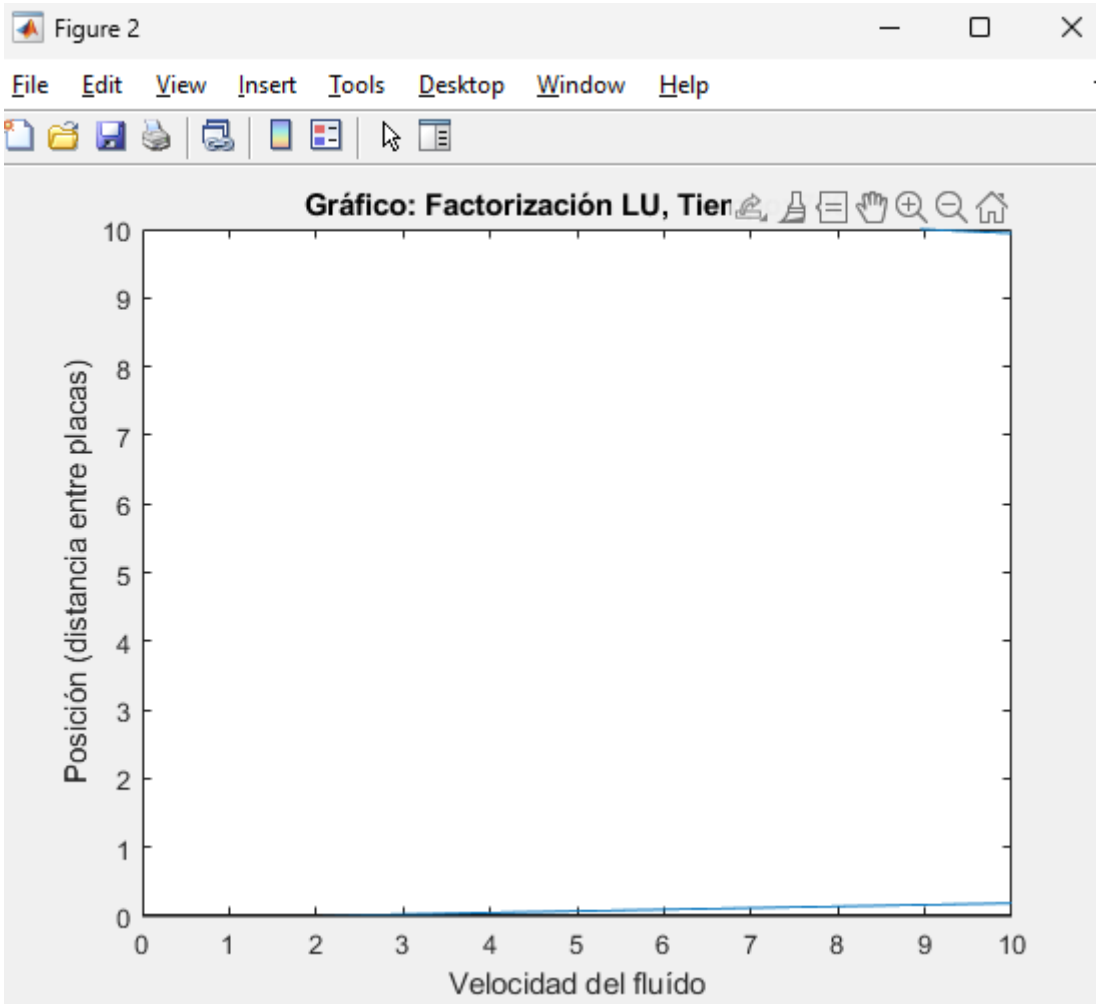


Figura 14. Factorización LU para T = 1000s

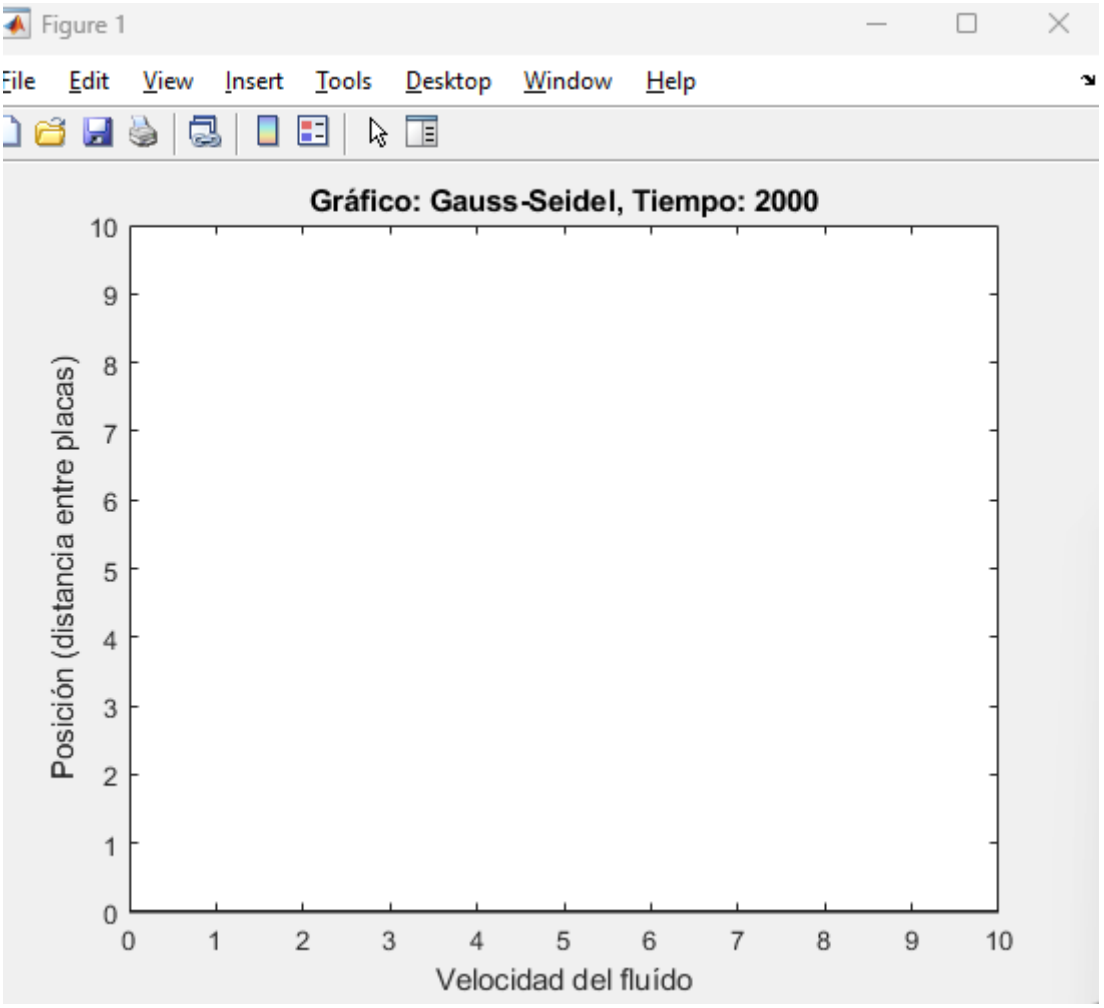


Figura 15. Factorización Gauss-Seidel para T = 2000s

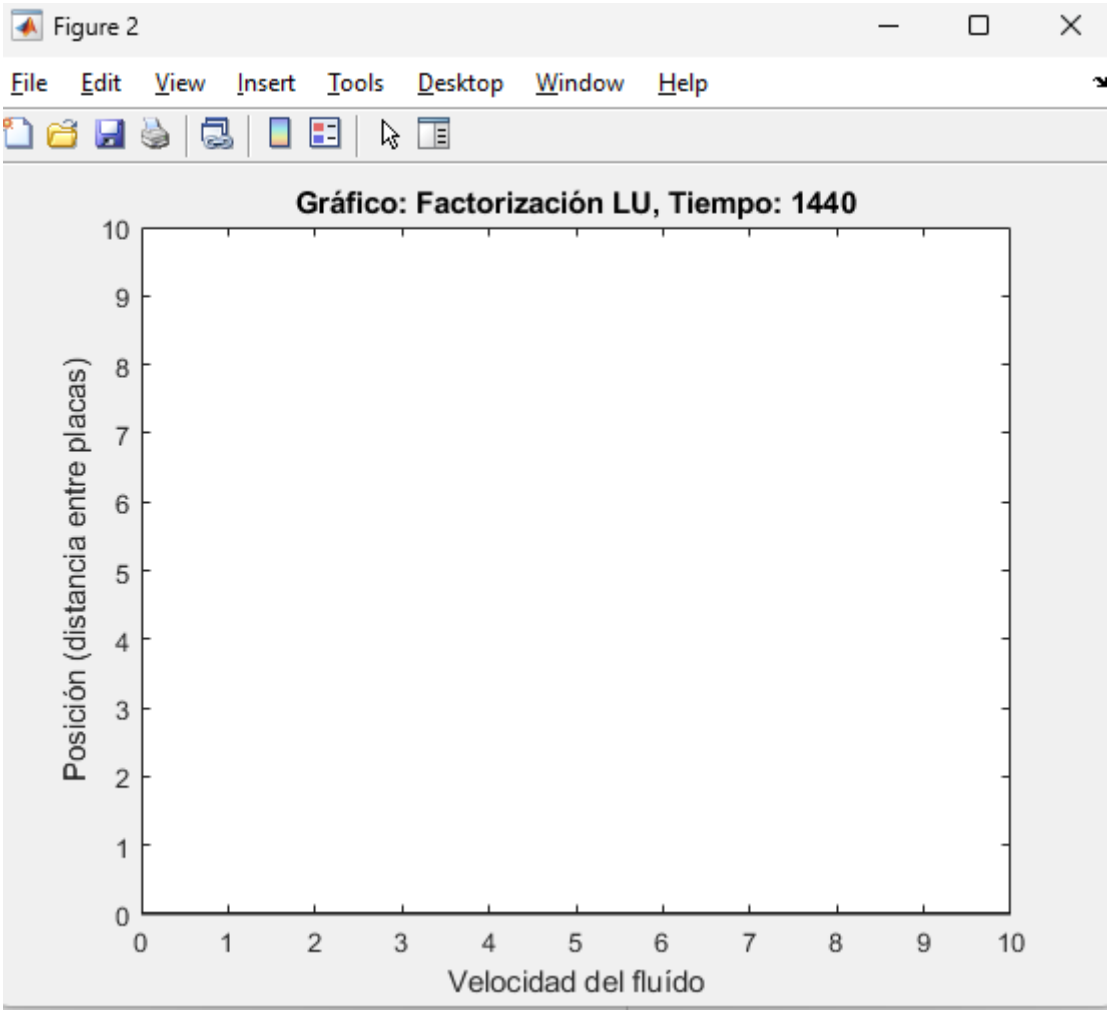


Figura 16. Factorización LU para T = 2000s

1-D [15 puntos]

En las figuras 15 y 16 se puede ver una ejecución del programa para 2000 s y dt = 5. No consideramos que estos resultados sean representativos de la realidad, debido a que no se toman en cuenta variables importantes que se estudian en la dinámica de fluidos. Por ejemplo, no se toma en cuenta el material del que está hecha la tubería.

Es importante tomar en cuenta el material por temas de rugosidad ya que esto afecta las velocidades.

Por último, dados los resultados anteriores se puede concluir que ninguno de los métodos logra converger a un resultado realista, sin embargo, el método gauss seidel se reitera hasta llegar a un precisión deseada, esto genera resultados más precisos, por lo que, aunque este método sea más lento de acuerdo a la Figura 17, se puede tomar como el más acertado debido a lo anterior.

```
Tiempo de ejecución Gauss-Seidel:
    0.5134

Tiempo de ejecución LU:
    0.0209
```

Figura 17. tiempos de ejecución para T = 2000s

Problema 2) [10 puntos]

Se definen varios parámetros que se van a utilizar en las ecuaciones y en la resolución numérica.

En ellos tenemos la viscosidad del oil y del agua (mu_oil y mu_H2O), la densidad del oil (rho_oil), la distancia entre las placas (x), el tiempo total de simulación (t), la altura de las capas de agua y oil (h_water y h_oil), y la velocidad de la placa superior (v).

También se definen los parámetros para la resolución numérica de las ecuaciones diferenciales parciales. Para esto se ocupa el tamaño del paso espacial (dx), el tamaño del paso temporal (dt), el número de puntos espaciales (nx) y el número de puntos temporales (nt).

El tamaño del paso espacial dx y el tamaño del paso temporal dt son los incrementos que se utilizan en la discretización del espacio y el tiempo, respectivamente. La discretización es un proceso en el que se divide el dominio del problema (en este caso, el espacio entre las placas y el tiempo de simulación) en una serie de pequeños incrementos o pasos. Esto permite aproximar las derivadas en las EDP utilizando diferencias finitas.

El número de puntos espaciales nx y el número de puntos temporales nt son el número total de puntos en los que se calculará la solución en el espacio y el tiempo, respectivamente. Se calculan dividiendo la longitud total del dominio (la distancia entre las placas para nx y el tiempo total de simulación para nt) por el tamaño del paso correspondiente y sumando 1.

Después de esto se inicializan las matrices de v_oil y v_H2O para almacenar las velocidades del agua y el oil en cada punto espacial y temporal. Las velocidades de inicializan a v, la velocidad de la placa superior.

En la siguiente figura se observa la programación descrita en Matlab:

```
mu_oil = 3; % cP
mu_water = 1; % cP
rho_oil = 0.9; % g/cm^3
x = 10; % cm
t = 200; % segundos
h_water = 6; % cm
h_oil = 4; % cm
v = 9; % cm/s

% Parámetros para la resolución numérica
dx = 0.1; % tamaño del paso espacial
dt = 1; % tamaño del paso temporal
nx = round(x / dx) + 1; % número de puntos espaciales
nt = round(t / dt) + 1; % número de puntos temporales

% Inicialización de las velocidades
v_oil = zeros(nx, nt);
v_oil(:, 1) = v;
v_water = zeros(nx, nt);
v_water(:, 1) = v;
```

Figura 18. Declaración de constantes para la resolución del ejercicio 2

Seguidamente se crean las matrices A_oil y A_water, para así aplicar el metodo implicito. Estas respresentan el sistemas de ecuaciones lineales que se resuelve en cada paso temporal para obtener las velocidades del aceite y el agua.

Se aplican las condiciones de contorno para el agua y el aceite, estas aseguran que la solucion sea realista.

Despues se resuelven numericamente las ecuaciones parciales diferenciales del agua y el aceite usando nuestro metodo implicito. Se destaca que este metodo es implicito porque wl valor de la velocidad en un punto y un tiempo dado se va acalcular resolviendo un sistema de ecuaciones lineales que incluye valores de la velocidad en los puntos y tiempos actuales y anteriores.

Lo descrito se puede observar en la siguiente figura:

```
21
22    % Matrices para el método implícito
23 -  A_oil = diag(ones(nx, 1)) - (dt * mu_oil / rho_oil) * diag(ones(nx-1, 1), -1) + (2 * dt * mu_oil / rho_oil) * diag(ones(nx, 1)) - (dt * mu_oil / rho_oil) * diag(ones(nx-1, 1), 1);
24 -  A_water = diag(ones(nx, 1)) - (dt * mu_water / rho_oil) * diag(ones(nx-1, 1), -1) + (2 * dt * mu_water / rho_oil) * diag(ones(nx, 1)) - (dt * mu_water / rho_oil) * diag(ones(nx-1,
25
26    % Condiciones de contorno
27 -  A_oil(1, 1) = 1;
28 -  A_oil(1, 2) = 0;
29 -  A_oil(nx, nx-1) = 0;
30 -  A_oil(nx, nx) = 1;
31
32 -  A_water(1, 1) = 1;
33 -  A_water(1, 2) = 0;
34 -  A_water(nx, nx-1) = 0;
35 -  A_water(nx, nx) = 1;
36
37    % Resolución numérica de las ecuaciones diferenciales parciales
38 -  for j = 2:nt
39        % Ecuaciones de movimiento para el oil (método implícito)
40 -      v_oil(:, j) = A_oil \ v_oil(:, j-1);
41
42        % Ecuaciones de movimiento para el water (método implícito)
43 -      v_water(:, j) = A_water \ v_water(:, j-1);
44 -  end
```

Figura 19. Resolución de las ecuaciones diferenciales

Es importante saber que el mejor método para solucionar el problema es el de Simpson. El método del trapecio compuesto busca aproximar la curva que se desea integrar utilizando rectas espaciadas a cierta distancia, sin embargo, el método de Simpson utiliza curvas de grado mayor a 1, esto permite aproximar las funciones de una manera más exacta, generando un error más bajo.

```
% Cálculo de velocidades promedio para el oil utilizando la regla de Simpson
integral_oil_simpson = SimpsonComp(@(h) v_oil(:, end), h_water, h_water + h_oil, nx);

% Cálculo de velocidades promedio para el water utilizando la regla de Simpson
integral_water_simpson = SimpsonComp(@(h) v_water(:, end), 0, h_water, nx);

% Velocidad promedio
v_oil_avg_simpson = integral_oil_simpson / h_oil;
v_water_avg_simpson = integral_water_simpson / h_water;
```

Figura 20. Cálculos utilizando Simpson

Finalmente, se imprimen las velocidades promedio del oil y del agua. Además de los valores de la integral de cada uno.

```
56      % Imprimir resultados
57 -    fprintf('Velocidad promedio de oil (Simpson) a %d segundos: %.2f cm/s\n', t, v_oil_avg_simpson);
58 -    fprintf('Velocidad promedio de water (Simpson) a %d segundos: %.2f cm/s\n', t, v_water_avg_simpson);
59
60      % Imprimir resultados de las integrales
61 -    fprintf('Integral de oil (Simpson) a %d segundos: %.2f\n', t, integral_oil_simpson);
62 -    fprintf('Integral de water (Simpson) a %d segundos: %.2f\n', t, integral_water_simpson);
```

Figura 21. Impresión de las integrales y velocidades promedios utilizando Simpson

Obteniendo así:

```
>> Ejercicio
Velocidad promedio de oil (Simpson) a 200 segundos: 8.91 cm/s
Velocidad promedio de water (Simpson) a 200 segundos: 8.91 cm/s
Integral de oil (Simpson) a 200 segundos: 35.64
Integral de water (Simpson) a 200 segundos: 53.47
>>
```

Figura 22. Cálculo de las integrales y velocidades promedios utilizando Simpson

Problema 3) [10 puntos]
La resolución implica abordar un problema diferencial de valor inicial, donde las condiciones iniciales, el criterio y el rango se describen en las ecuaciones 11 y 12, respectivamente. Para llevar a cabo la solución, es necesario trabajar con 16 puntos equidistantes, por lo que n=15.

$$y(1) = 2 \tag{11}$$

$$y' = t - y\,;\ t \in [1, 6] \tag{12}$$

Con este propósito, se empleará el método de Euler para calcular las condiciones iniciales. Este método se basa en la idea de que, dado que $dy/dt$ representa la pendiente de la función en un punto, podemos utilizarlo para calcular el siguiente valor de la función a partir de un valor inicial. Esto se expresa en la ecuación 13.

$$y_{(i+1)} = y_i + h\frac{dy}{dt} \tag{13}$$

A continuación, se aplica el método multipaso de corrección Adams-Bashforth 2. Este método se fundamenta en la premisa de que, una vez iniciado el cálculo con métodos como Euler, los puntos calculados previamente pueden emplearse para perfeccionar las aproximaciones. La formulación de este método se detalla en la ecuación 14, donde los coeficientes β0 y β1 son 1.5 y -0.5, respectivamente.

$$y_{(i+1)} = y_i + h\left(\frac{3}{2}f_i - \frac{1}{2}f_{i-1}\right) \tag{14}$$

Para lograr una precisión adicional, se llevan a cabo tres correcciones adicionales sobre las primeras aproximaciones calculadas mediante el método Adams-Bashforth 2. En este caso, el método multipaso que se empleará es el Adams-Moulton de orden 4, cuya formulación y parámetros β0, β1, β2 y β3 se presentan en la ecuación 15.

$$y_{(i+1)} = y_i + h\left(\frac{9}{24}f_{i+1} + \frac{19}{24}f_i + \frac{5}{24}f_{i-1} + \frac{1}{24}f_{i-2}\right) \tag{15}$$

Los resultados de la primera aproximación se presentan en la Tabla I, mientras que las aproximaciones y los errores obtenidos mediante el método Adams-Moulton de orden 4 (AM4) se detallan en la Tabla II.

Tabla 1. Valores obtenidos mediante Euler y AB2

| AB2 y Euler | | | |
|---|---|---|---|
| x | yi | yi-1 | y |
| 1 | | | 2 |
| 1.33333333 | | | 1.66666667 |
| 1.66666667 | -0.33333333 | -1 | 1.66666667 |
| 2 | 0 | -0.33333333 | 1.72222222 |
| 2.33333333 | 0.27777778 | 0 | 1.86111111 |
| 2.66666667 | 0.47222222 | 0.27777778 | 2.05092593 |
| 3 | 0.61574074 | 0.47222222 | 2.28009259 |
| 3.33333333 | 0.71990741 | 0.61574074 | 2.53742284 |
| 3.66666667 | 0.79591049 | 0.71990741 | 2.81539352 |
| 4 | 0.85127315 | 0.79591049 | 3.10837834 |
| 4.33333333 | 0.89162166 | 0.85127315 | 3.41231031 |
| 4.66666667 | 0.92102302 | 0.89162166 | 3.72421821 |
| 5 | 0.94244845 | 0.92102302 | 4.0419386 |
| 5.33333333 | 0.9580614 | 0.94244845 | 4.36389456 |
| 5.66666667 | 0.96943877 | 0.9580614 | 4.68893705 |
| 6 | 0.97772962 | 0.96943877 | 5.01622873 |

Tabla 2. Valores obtenidos mediante la tercera corrección AM4

| AM4 - Correcion 3 | | | | | | |
|---|---|---|---|---|---|---|
| x | fi+1 | fi | fi-1 | fi-2 | yi mejorado | Error |
| 1 | | | | | 2 | 0 |
| 1.33333333 | | | | | 1.66666667 | 0 |
| 1.66666667 | | | | | 1.66666667 | 0 |
| 2 | 0.28790509 | 0 | -0.33333333 | -1 | 1.71191406 | 0.00018084 |
| 2.33333333 | 0.48840181 | 0.28808594 | 0 | -0.33333333 | 1.84435734 | 0.00057418 |
| 2.66666667 | 0.63302287 | 0.488976 | 0.28808594 | 0 | 2.03251456 | 0.00112923 |
| 3 | 0.73628285 | 0.63415211 | 0.488976 | 0.28808594 | 2.26194014 | 0.00177701 |
| 3.33333333 | 0.81015924 | 0.73805986 | 0.63415211 | 0.488976 | 2.52072883 | 0.00244526 |
| 3.66666667 | 0.86299687 | 0.8126045 | 0.73805986 | 0.63415211 | 2.80059425 | 0.00307554 |
| 4 | 0.90081156 | 0.86607241 | 0.8126045 | 0.73805986 | 3.09556255 | 0.00362589 |
| 4.33333333 | 0.92789924 | 0.90443745 | 0.86607241 | 0.8126045 | 3.4013632 | 0.00407089 |
| 4.66666667 | 0.94733078 | 0.93197013 | 0.90443745 | 0.86607241 | 3.71493674 | 0.00439915 |
| 5 | 0.96129843 | 0.95172993 | 0.93197013 | 0.90443745 | 4.03409148 | 0.00461009 |
| 5.33333333 | 0.97136605 | 0.96590852 | 0.95172993 | 0.93197013 | 4.35725644 | 0.00471085 |
| 5.66666667 | 0.97864841 | 0.9760769 | 0.96590852 | 0.95172993 | 4.68330483 | 0.00471343 |
| 6 | 0.98393969 | 0.98336184 | 0.9760769 | 0.96590852 | 5.01142783 | 0.00463249 |