Difusión y Permeabilidad del Hidrógeno en Coordenadas Cilíndricas

Juan David Parra

September 26, 2024

1 Introducción

Este documento describe las ecuaciones fundamentales para modelar tanto la difusión como la permeabilidad del hidrógeno en un material cilíndrico. Se presentan las ecuaciones de difusión en coordenadas cilíndricas, las condiciones de frontera, así como las ecuaciones que describen el comportamiento del flujo y la permeabilidad, basadas en la Ley de Fick.

2 Ecuación de difusión en coordenadas cilíndricas

La ecuación de difusión del hidrógeno en coordenadas cilíndricas en tres dimensiones es:

$$\frac{\partial C}{\partial t} = D\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial C}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 C}{\partial \theta^2} + \frac{\partial^2 C}{\partial z^2}\right)$$

Donde:

- $C(r, \theta, z, t)$ es la concentración de hidrógeno en el material,
- D es el coeficiente de difusión,
- r, θ, z son las coordenadas cilíndricas.

2.1 Simplificación radial

Para problemas con simetría axial, la ecuación se reduce a una dimensión en r:

$$\frac{\partial C}{\partial t} = D\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial C}{\partial r}\right)\right)$$

3 Condiciones de frontera

Las condiciones de frontera para resolver la ecuación de difusión en un cilindro son:

3.1 Condición de Dirichlet

Concentración fija de hidrógeno en la frontera del material:

$$C(r = R, t) = C_0$$

3.2 Condición de Neumann

Flujo de hidrógeno nulo en el centro del cilindro:

$$\frac{\partial C}{\partial r}(r = 0, t) = 0$$

4 Ecuaciones de permeabilidad del hidrógeno

La permeabilidad describe la capacidad de un material para permitir el paso de hidrógeno bajo un gradiente de concentración. El flujo de hidrógeno J a través de un material en coordenadas cilíndricas puede expresarse mediante la **Ley de Fick** como:

$$J = -P\frac{\partial C}{\partial r}$$

Donde:

- J es el flujo de hidrógeno en la dirección radial,
- P es el coeficiente de permeabilidad,
- $\frac{\partial C}{\partial r}$ es el gradiente de concentración de hidrógeno en la dirección radial.

El flujo de hidrógeno depende directamente de la permeabilidad del material y del gradiente de concentración. Para materiales donde el hidrógeno se difunde a través de barreras permeables, la ecuación general se ajusta a esta ley.

5 Relaciones entre difusión y permeabilidad

Es importante destacar la relación entre el coeficiente de difusión D, el coeficiente de permeabilidad P, y la solubilidad S. En muchos casos, la permeabilidad puede expresarse como:

$$P = D \cdot S$$

Donde:

- D es el coeficiente de difusión, que describe la rapidez con la que el hidrógeno se mueve a través del material,
- S es el coeficiente de solubilidad, que describe la cantidad de hidrógeno que puede ser absorbida por el material.

6 Deformaciones debido a la permeabilidad

El proceso de permeabilidad del hidrógeno puede generar deformaciones en el material debido a la absorción y difusión del gas. Las deformaciones radiales y tangenciales en coordenadas cilíndricas son:

6.1 Deformación radial

$$\varepsilon_r = \frac{\partial u_r}{\partial r}$$

Donde u_r es el desplazamiento radial.

6.2 Deformación tangencial

$$\varepsilon_\theta = \frac{u_r}{r}$$

7 Ecuación de equilibrio radial

Cuando el hidrógeno se difunde en el material, la ecuación de equilibrio radial asegura que los esfuerzos internos están balanceados. Esta ecuación es:

$$\frac{d\sigma_r}{dr} + \frac{\sigma_r - \sigma_\theta}{r} = 0$$

Donde:

- σ_r es el esfuerzo radial,
- σ_θ es el esfuerzo tangencial.

8 Relación esfuerzo-deformación con deformaciones térmicas

Cuando hay cambios de temperatura en el material debido a la difusión de hidrógeno, las ecuaciones esfuerzo-deformación incluyen el efecto de la expansión térmica. Las ecuaciones son:

8.1 Esfuerzo radial

$$\sigma_r = \frac{E}{(1+\nu)(1-2\nu)}\left((1-\nu)\varepsilon_r + \nu\varepsilon_\theta\right) - \alpha\Delta T$$

8.2 Esfuerzo tangencial

$$\sigma_\theta = \frac{E}{(1+\nu)(1-2\nu)}\left((1-\nu)\varepsilon_\theta + \nu\varepsilon_r\right) - \alpha\Delta T$$

Donde:

- E es el módulo de Young,
- ν es el coeficiente de Poisson,
- α es el coeficiente de expansión térmica,
- ΔT es el cambio de temperatura.

9 Método de Diferencias Finitas para EDOs

El método de diferencias finitas es una técnica numérica para resolver ecuaciones diferenciales ordinarias (EDOs) y parciales (EDPs) al aproximar las derivadas mediante diferencias finitas. Aquí se presenta la implementación en Python para una ecuación de difusión simplificada radialmente.

9.1 Ecuación de Difusión Radial

La ecuación de difusión en coordenadas cilíndricas, simplificada radialmente, es:

$$\frac{\partial C}{\partial t} = D\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial C}{\partial r}\right)\right) \tag{1}$$

9.2 Método de Diferencias Finitas

El método de diferencias finitas consiste en discretizar las derivadas de la ecuación en una malla. Para el caso de la ecuación de difusión simplificada radialmente:

$$\frac{\partial C}{\partial t} = D\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial C}{\partial r}\right)\right)$$

Podemos discretizar en tiempo usando un esquema explícito:

$$\frac{C_j^{n+1} - C_j^n}{\Delta t} = D\left(\frac{1}{r_j}\frac{C_{j+1}^n - 2C_j^n + C_{j-1}^n}{\Delta r^2} + \frac{C_{j+1}^n - C_{j-1}^n}{2r_j\Delta r}\right)$$

Donde:

- C_j^n es la concentración en el nodo j en el tiempo n,
- Δt es el paso de tiempo,
- Δr es el paso espacial.

El esquema explícito es sencillo de implementar, pero requiere que el paso de tiempo cumpla con la condición de estabilidad de Courant-Friedrichs-Lewy (CFL):

$$\Delta t < \frac{\Delta r^2}{2D}$$

Esta condición asegura que el método sea estable para la simulación.

9.3 Implementación en Python

Listing 1: Difusión Radial en Python

```
import numpy as np
import matplotlib.pyplot as plt

# Par metros
D = 1.0  # Coeficiente de difusi n
R = 1.0  # Radio m ximo
T = 1.0  # Tiempo m ximo
N = 100  # N mero de puntos espaciales
M = 1000  # N mero de puntos temporales
```

```
# Discretizaci n
dr = R / (N - 1)
dt = T / (M - 1)
r = np.linspace(0, R, N)
t = np.linspace(0, T, M)

# Condiciones iniciales y de frontera
C = np.zeros((M, N))
C[:, 0] = 1.0  # Condici n de Dirichlet en r = 0
C[:, -1] = 0.0  # Condici n de Dirichlet en r = R

# M todo de diferencias finitas
for n in range(0, M-1):
    for j in range(1, N-1):
        C[n+1, j] = C[n, j] + dt * D * (
            (C[n, j+1] - 2*C[n, j] + C[n, j-1]) / dr
               **2 +
            (C[n, j+1] - C[n, j-1]) / (2 * r[j] * dr
               )
        )

# Resultado final
plt.imshow(C, extent=[0, R, T, 0], aspect='auto')
plt.colorbar(label='Concentraci n')
plt.xlabel('Radio (r)')
plt.ylabel('Tiempo (t)')
plt.title('Difusi n de Hidr geno en Coordenadas
   Cil ndricas')
plt.show()
```

10 Elemjentos Finitos (FEM) en Nodos de Frontera Matricial Lagrangianos en Python

Los nodos de frontera matricial lagrangianos se utilizan en el método de elementos finitos (FEM) para resolver ecuaciones diferenciales parciales (EDPs). En este ejemplo, se usa FEniCSx para resolver un problema de difusión.

10.1 Método de Elementos Finitos (FEM)

En el método de elementos finitos, la ecuación de difusión se multiplica por una función de prueba v y se integra sobre el dominio. La forma débil de la ecuación de difusión radialmente simplificada es:

$$\int_\Omega v \frac{\partial C}{\partial t} \, d\Omega = -D \int_\Omega \nabla v \cdot \nabla C \, d\Omega$$

Donde Ω es el dominio cilíndrico. Utilizando el método de Galerkin, aproximamos C y v como combinaciones lineales de funciones de forma nodales:

$$C \approx \sum_i C_i \phi_i, \quad v \approx \sum_j v_j \phi_j$$

Sustituyendo en la forma débil, obtenemos un sistema de ecuaciones matriciales:

$$M \frac{\partial \mathbf{C}}{\partial t} = -DK\mathbf{C}$$

Donde:

- M es la matriz de masa, que proviene de la integración de las funciones de forma,
- K es la matriz de rigidez, que proviene de las derivadas de las funciones de forma.

Este sistema se resuelve en cada paso de tiempo usando un esquema temporal como el método de Euler hacia atrás o Crank-Nicolson para avanzar la solución.

10.2 Implementación en Python con FEniCSx

Listing 2: Nodos de Frontera Matricial Lagrangianos en Python con FEniCSx

```
import numpy as np
import matplotlib.pyplot as plt
from dolfinx import mesh, fem
from mpi4py import MPI
from ufl import TrialFunction, TestFunction, dx,
   inner, grad, div
```

```
# Par metros
L = 1.0  # Longitud del dominio
T = 0.1  # Tiempo m ximo
num_elements = 100  # N mero de elementos
   espaciales
num_steps = 100  # N mero de pasos de tiempo
dt = T / num_steps  # Paso de tiempo
D = 1.0  # Coeficiente de difusi n

# Malla
domain = mesh.create_interval(MPI.COMM_WORLD,
   num_elements, [0.0, L])

# Funci n de espacio
V = fem.FunctionSpace(domain, ("CG", 1))

# Funciones de prueba y ensayo
u = TrialFunction(V)
v = TestFunction(V)

# Condiciones iniciales
u_n = fem.Function(V)
u_n.interpolate(lambda x: np.sin(np.pi * x[0]))

# Definici n del problema variacional
a = u * v * dx + dt * D * inner(grad(u), grad(v)) *
   dx
L = u_n * v * dx

# Condiciones de frontera
boundary_facets = mesh.locate_entities_boundary(
   domain, dim=1, marker=lambda x: np.full(x.shape
   [1], True))
bc = fem.dirichletbc(fem.Constant(domain, 0.0), fem.
   locate_dofs_topological(V, 1, boundary_facets))

# Soluci n
u = fem.Function(V)
problem = fem.LinearProblem(a, L, bcs=[bc])

# Bucle de tiempo
```

```
for n in range(num_steps):
    u_n.x.array[:] = u.x.array[:]
    u = problem.solve()
    u_n.x.array[:] = u.x.array[:]

# Resultados
x = V.tabulate_dof_coordinates()
plt.plot(x[:, 0], u.x.array, label="t = {:.2f}".
   format(T))
plt.xlabel("x")
plt.ylabel("C")
plt.legend()
plt.title("Difusi n de Hidr geno")
plt.show()
```

11 Método de Monte Carlo Gran Canonical con Dinámica Molecular (MD)

El método de Monte Carlo Gran Canonical (GCMC) combinado con Dinámica Molecular (MD) permite simular sistemas en contacto con un reservorio de partículas y energía.

11.1 Método de Monte Carlo Gran Canonical (GCMC)

El método GCMC se utiliza para simular sistemas en equilibrio con un reservorio de partículas. En este método, el número de partículas puede variar, lo que lo hace ideal para estudiar la absorción de hidrógeno en materiales. El algoritmo se basa en la inserción y eliminación de partículas con una probabilidad que depende del potencial químico μ y la temperatura T del sistema.

La probabilidad de insertar una nueva partícula en una posición r es:

$$P_{\text{insertar}} = \min\left(1, \exp\left(\frac{\mu - \Delta E}{k_B T}\right)\right)$$

Donde ΔE es el cambio en energía debido a la inserción, k_B es la constante de Boltzmann, y T es la temperatura.

Por otro lado, la probabilidad de eliminar una partícula es:

$$P_{\text{eliminar}} = \min\left(1, \exp\left(\frac{\Delta E - \mu}{k_B T}\right)\right)$$

11.2 Dinámica Molecular (MD)

La dinámica molecular se utiliza para simular el movimiento de las partículas bajo la influencia de fuerzas calculadas a partir de interacciones interatómicas. La fuerza entre dos partículas i y j se puede calcular usando el potencial de Lennard-Jones:

$$V(r_{ij}) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]$$

Donde ϵ y σ son parámetros del potencial, y r_{ij} es la distancia entre las partículas i y j.

Las posiciones y velocidades de las partículas se actualizan usando el método de Verlet:

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{\mathbf{f}(t)}{2m}\Delta t^2$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \frac{\mathbf{f}(t) + \mathbf{f}(t + \Delta t)}{2m}\Delta t$$

Donde $\mathbf{r}(t)$ es la posición, $\mathbf{v}(t)$ es la velocidad, $\mathbf{f}(t)$ es la fuerza, m es la masa, y Δt es el paso de tiempo.

11.3 Implementación en Python

Listing 3: Monte Carlo Gran Canonical con Dinámica Molecular en Python

```
import numpy as np
import random

# Par metros
V = 10.0  # Volumen del sistema
kT = 1.0  # Energ a t rmica (k_B * T)
mu = 1.0  # Potencial qu mico
N_max = 100  # N mero m ximo de part culas
L = 10.0  # Longitud de la caja
dt = 0.01  # Paso de tiempo
steps = 1000  # N mero de pasos de tiempo
```

```
# Inicializaci n
N = 50  # N mero inicial de part culas
positions = np.random.rand(N, 3) * V**(1/3)
velocities = np.random.randn(N, 3)

# Funci n de energ a (ejemplo sencillo)
def energy(positions):
    return np.sum(positions**2)

# Potencial de Lennard-Jones
def lennard_jones(r):
    return 4 * ((1/r)**12 - (1/r)**6)

# Fuerzas
def compute_forces(positions):
    forces = np.zeros_like(positions)
    for i in range(N):
        for j in range(i+1, N):
            r_ij = positions[j] - positions[i]
            r = np.linalg.norm(r_ij)
            f = lennard_jones(r) * r_ij / r
            forces[i] += f
            forces[j] -= f
    return forces

# Funci n de inserci n
def insert_particle(positions, velocities, N):
    if N < N_max:
        new_position = np.random.rand(1, 3) * V
           **(1/3)
        new_velocity = np.random.randn(1, 3)
        dE = energy(new_position)  # Cambio en
           energ a (simplificado)
        acceptance_ratio = np.exp((mu - dE) / kT)
        if random.random() < acceptance_ratio:
            positions = np.vstack([positions,
               new_position])
            velocities = np.vstack([velocities,
               new_velocity])
            N += 1
    return positions, velocities, N
```

```
# Funci n de eliminaci n
def remove_particle(positions, velocities, N):
    if N > 0:
        index = random.randint(0, N-1)
        dE = -energy(positions[index])  # Cambio en
           energ a (simplificado)
        acceptance_ratio = np.exp((dE - mu) / kT)
        if random.random() < acceptance_ratio:
            positions = np.delete(positions, index,
               axis=0)
            velocities = np.delete(velocities, index
               , axis=0)
            N -= 1
    return positions, velocities, N

# Bucle de simulaci n
forces = compute_forces(positions)
for step in range(steps):
    if random.random() < 0.5:
        positions, velocities, N = insert_particle(
           positions, velocities, N)
    else:
        positions, velocities, N = remove_particle(
           positions, velocities, N)

    # Integraci n con Verlet
    positions += velocities * dt + 0.5 * forces * dt
       **2
    new_forces = compute_forces(positions)
    velocities += 0.5 * (forces + new_forces) * dt
    forces = new_forces

print("N mero final de part culas:", N)
print("Posiciones finales:", positions)
```

12 Conclusión

En este documento hemos presentado un conjunto de ecuaciones que describen la difusión y la permeabilidad del hidrógeno en un material cilíndrico, junto con tres métodos numéricos para resolver ecuaciones diferenciales y problemas relacionados. Estos métodos incluyen el método de diferencias finitas, los nodos de frontera matricial lagrangianos y el método de Monte Carlo Gran Canonical combinado con Dinámica Molecular. Estas técnicas se pueden implementar en Python y son útiles para simular sistemas físicos y químicos complejos.

Referencias

1. Barna, I. F., & Mátyás, L. (2023). Analytical Solution and Numerical Simulation of Heat Transfer in Cylindrical- and Spherical-Shaped Bodies. *Computation*, 11(7), 131. https://doi.org/10.3390/computation11070131.

2. Kim, K.-T., Chung, N.-K., Baek, U.-B., & Nahm, S.-H. (2022). Characterizing the Diffusion Property of Hydrogen Sorption and Desorption Processes in Several Spherical-Shaped Polymers. *Polymers*, 14(7), 1468. https://doi.org/10.3390/polym14071468.

3. Wang, H., Fu, W., Jiang, Z., Zhao, C., & Hua, Q. (2024). Mechanism for Adsorption, Dissociation, and Diffusion of Hydrogen in High-Entropy Alloy AlCrTiNiV: First-Principles Calculation. *Nanomaterials*, 14(17), 1391. https://doi.org/10.3390/nano14171391.

4. Barabás, J., Jovicic, V., & Delgado, A. (2022). Simulation of a Hydrogen-Air Diffusion Flame under Consideration of Component-Specific Diffusivities. *Applied Sciences*, 12(6), 3138. https://doi.org/10.3390/app12063138.

5. Numerical Modeling of Anisotropic Particle Diffusion through a Cylindrical Channel. MDPI. Retrieved from https://www.mdpi.com/2076-3417/10/12/4203.