Numerical solutions for hydrogen permeation in Python.

Arseniy A. Kuzmin

July 14, 2020

I will describe several methods of solving the diffusion problem for the hydrogne permeation through a metal membrane, starting with the simplest cases. One dimensional membrane, no traps. There are explicit and implicit methods. We need to select the onse which are faster and more precise. The numerical solution is implemented in Python programming language. Calculations in pure Python are slow, but thera several methods to make them fast. One - to use specialized packages such as numpy and scipy. They have built in methods for operations on matrices and several solvers for systems of linear equations. Another option is to use numba and jit, which compile the python code into some machine code, which improves the performance of python loops dramatically.

But first things first, we will start the explanation with some explicit stencils, then explore some implicit stencils and hopefully in the future will address more sofisticated methods with variable space steps. I combined explanations from several resources, one of most helpful was a jupyter notebook course for simulation from 2014, their code is on GitHub/numerical-moc. I also used works by A. A. Pisarev and E. D. Marenkov from MEPhI, as well as papers of S. K. Sharma-san.

1. Permeation equation.

The general equation for hydrogen permeation trough a membrane, one-dimensional case.

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

$$\Gamma_{in}(t) + D\frac{\partial u}{\partial x}\Big|_{x=0} - k_u u^2(0,t) = 0 \tag{2}$$

$$-D\frac{\partial u}{\partial x}\Big|_{x=L} - k_d u^2(L,t) = 0 \tag{3}$$

Here u is the concentration, t - time, x - coordinate, D - diffusion coefficient, k_u and k_d - are hydrogen recombination coefficients on the upstream and downstream sides, respectively. The function $f(u)$ may express the contribution of hydrogen traps. The $\Gamma_{in}(t)$ is the incident atomic hydrogen flux.

There are several ways to approximate the actual derivatives with finite differences. The explicit-implicit approximation of Crank-Nicolson:

$$\frac{\partial^2 u}{\partial x^2}\Big|_{x=j\Delta x, t=n\Delta t} \approx \frac{1}{2\Delta x^2}\left(U_{j+1}^n - 2U_j^n + U_{j-1}^n + U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}\right). \tag{4}$$

Here Δx and Δt are the coordinate and time steps, respectively. When they are decided, we can write down the vectors for descrete coordinate and time:

$$t_n = n\Delta t, n = 0, \ldots, N-1 \tag{5}$$

$$x_j = j\Delta t, j = 0, \ldots, J-1 \tag{6}$$

where N and J - are number of points for time and space coorinates, respectively.

Now we have the finite difference approximation, we can rewrite euqtion 1 as follows:

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = \frac{D}{2\Delta x^2}\left(U_{j+1}^n - 2U_j^n + U_{j-1}^n + U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}\right) + f(U_j^n) \tag{7}$$

By introducing $\sigma = \frac{\Delta t D}{2\Delta x^2}$ we can simplify the above equation:

$$\boxed{-\sigma U_{j-1}^{n+1} + (1+2\sigma)U_j^{n+1} - \sigma U_{j+1}^{n+1} = \sigma U_{j-1}^n + (1-2\sigma)U_j^n + \sigma U_{j+1}^n + \Delta t f(U_j^n)} \tag{8}$$

This gives us a system of linear equation for unknown vector $\vec{U}^{n+1}$. Two boundary conditions are outside our stencil, and we have to approach them separately, depending on the condition. In our case, we have squeared concentration in our boundary condition, which makes this approach slightly more complex. After the whole system is rewriten using the finite differneces, one can solve that system. One of the basic algrorythms for doing this is to use the tridiagonal matrix algrorythm.

2. Stencil cheat sheet.

Here I changed the space and time coordinate indices from j and n to more straight forward x and t.

2.1. Stensils

Forward Euler method:

$$\boxed{U_x^{t+1} = \sigma U_{x-1}^t + (1-2\sigma)U_x^t + \sigma U_{x+1}^t} \tag{9}$$

Backward Euler method:

$$\boxed{-\sigma U_{x-1}^{t+1} + (1+2\sigma)U_x^{t+1} - \sigma U_{x+1}^{t+1} = U_x^t} \tag{10}$$

Crank-Nicolson method:

$$\boxed{-\frac{\sigma}{2}U_{x-1}^{t+1} + (1+\sigma)U_x^{t+1} - \frac{\sigma}{2}U_{x+1}^{t+1} = \frac{\sigma}{2}U_{x-1}^t + (1-\sigma)U_x^t + \frac{\sigma}{2}U_{x+1}^t} \tag{11}$$

2.2. Backward Euler

$$-\sigma U_{x-1}^{t+1} + (1+2\sigma)U_x^{t+1} - \sigma U_{x+1}^{t+1} = U_x^t \tag{12}$$

$$U_0^{t+1} = -\frac{D}{2k_u\Delta x} + \frac{1}{2}\sqrt{\left(\frac{D}{k_u\Delta x}\right)^2 + 4\frac{D}{k_u\Delta x}U_1^{t+1} + \Gamma^{t+1}} \tag{13}$$

$$U_L^{t+1} = -\frac{D}{2k_d\Delta x} + \frac{1}{2}\sqrt{\left(\frac{D}{k_d\Delta x}\right)^2 + 4\frac{D}{k_d\Delta x}U_{L-1}^{t+1}} \tag{14}$$