Numerical solutions for hydrogen permeation in Python.

Arseniy A. Kuzmin

July 16, 2020

Abstract

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. Finite differnce methods

This section is a copy from GitHub/numerical-moc notebooks.

1.1. The Crank-Nicolson Method

The Crank-Nicolson method is a well-known finite difference method for the numerical integration of the heat equation and closely related partial differential equations. We often resort to a Crank-Nicolson (CN) scheme when we integrate numerically reaction-diffusion systems in one space dimension

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + f(u),$$
$$\left.\frac{\partial u}{\partial x}\right|_{x=0,L} = 0,$$

where u is our concentration variable, x is the space variable, D is the diffusion coefficient of u, f is the reaction term, and L is the length of our one-dimensional space domain.

Note that we use Neumann boundary conditions and specify that the solution u has zero space slope at the boundaries, effectively prohibiting entrance or exit of material at the boundaries (no-flux boundary conditions).

1.2. Finite difference methods

Many fantastic textbooks and tutorials have been written about finite difference methods, for instance a free textbook by Lloyd Trefethen. Here we describe a few basic aspects of finite difference methods. The above reaction-diffusion equation describes the time evolution of variable $u(x,t)$ in one space dimension (u is a line concentration). If we knew an analytic expression for $u(x,t)$ then we could plot u in a two-dimensional coordinate system with axes t and x.

To approximate $u(x,t)$ numerically we discretize this two-dimensional coordinate system resulting, in the simplest case, in a two-dimensional regular grid. This picture is employed commonly when constructing finite differences methods, see for instance Figure 3.2.1 of Trefethen.

Let us discretize both time and space as follows:

$$t_n = n\Delta t,\ n = 0,\ldots,N-1, \tag{1.1}$$

$$x_j = j\Delta x,\ j = 0,\ldots,J-1, \tag{1.2}$$

where N and J are the number of discrete time and space points in our grid respectively. Δt and Δx are the time step and space step respectively and defined as follows:

$$\Delta t = T/N, \tag{1.3}$$

$$\Delta x = L/J \tag{1.4}$$

where T is the point in time up to which we will integrate u numerically.

Our ultimate goal is to construct a numerical method that allows us to approximate the unknonwn analytic solution $u(x,t)$ reasonably well in these discrete grid points. That is we want construct a method that computes values $U(j\Delta x, n\Delta t)$ (note: capital U) so that

$$U(j\Delta x, n\Delta t) \approx u(j\Delta x, n\Delta t) \tag{1.5}$$

As a shorthand we will write $U_j^n = U(j\Delta x, n\Delta t)$ and (j,n) to refer to grid point $(j\Delta x, n\Delta t)$. Let us define $\sigma = \frac{D\Delta t}{2\Delta x^2}$ and reorder the above approximation of our reaction-diffusion equation:

$$\boxed{-\sigma U_{j-1}^{n+1} + (1+2\sigma)U_j^{n+1} - \sigma U_{j+1}^{n+1} = \sigma U_{j-1}^n + (1-2\sigma)U_j^n + \sigma U_{j+1}^n + \Delta t\, f(U_j^n).} \tag{1.6}$$

This equation makes sense for space indices $j = 1,\ldots,J-2$ but it does not make sense for indices $j = 0$ and $j = J-1$ (on the boundaries):

$$j = 0:\ -\sigma U_{-1}^{n+1} + (1+2\sigma)U_0^{n+1} - \sigma U_1^{n+1} = \sigma U_{-1}^n + (1-2\sigma)U_0^n + \sigma U_1^n + \Delta t\, f(U_0^n), \tag{1.7}$$

$$j = J-1:\ -\sigma U_{J-2}^{n+1} + (1+2\sigma)U_{J-1}^{n+1} - \sigma U_J^{n+1} = \sigma U_{J-2}^n + (1-2\sigma)U_{J-1}^n + \sigma U_J^n + \Delta t\, f(U_{J-1}^n). \tag{1.8}$$

The problem here is that the values U_{-1}^n and U_J^n lie outside our grid. However, we can work out what these values should equal by considering our Neumann boundary condition. Let us discretize our boundary condition at $j = 0$ with the backward difference and at $j = J-1$ with the forward difference:

$$\frac{U_1^n - U_0^n}{\Delta x} = 0, \tag{1.9}$$

$$\frac{U_J^n - U_{J-1}^n}{\Delta x} = 0. \tag{1.10}$$

These two equations make it clear that we need to amend our above numerical approximation for $j = 0$ with the identities $U_0^n = U_1^n$ and $U_0^{n+1} = U_1^{n+1}$, and for $j = J-1$ with the identities $U_{J-1}^n = U_J^n$ and $U_{J-1}^{n+1} = U_J^{n+1}$. Let us reinterpret our numerical approximation of the line concentration of u in a fixed point in time as a vector $\mathbf{U}^n$:

$$\mathbf{U}^n = \begin{bmatrix} U_0^n \\ \vdots \\ U_{J-1}^n \end{bmatrix}. \tag{1.11}$$

Using this notation we can now write our above approximation for a fixed point in time, $t = n\Delta t$, compactly as a linear system:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ -\sigma & 1+2\sigma & -\sigma & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & -\sigma & 1+2\sigma & -\sigma & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\sigma & 1+2\sigma & -\sigma \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_0^{n+1} \\ U_1^{n+1} \\ U_2^{n+1} \\ \vdots \\ U_{J-2}^{n+1} \\ U_{J-1}^{n+1} \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \sigma & 1-2\sigma & \sigma & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & \sigma & 1-2\sigma & \sigma & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma & 1-2\sigma & \sigma \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} U_0^n \\ U_1^n \\ U_2^n \\ \vdots \\ U_{J-2}^n \\ U_{J-1}^n \end{bmatrix} + \begin{bmatrix} f(U_1^n) \\ 0 \\ 0 \\ \vdots \\ 0 \\ g(U_{J-2}^n) \end{bmatrix}.$$

Note that since our numerical integration starts with a well-defined initial condition at $n = 0$, $\mathbf{U}^0$, the vector $\mathbf{U}^{n+1}$ on the left-hand side is the only unknown in this system of linear equations. Thus, to integrate numerically our reaction-diffusion system from time point n to $n+1$ we need to solve numerically for vector $\mathbf{U}^{n+1}$. Let us call the matrix on the left-hand side A, the one on the right-hand side B, and the vector on the right-hand side $\mathbf{f}^n$. Using this notation we can write the above system as

$$A\mathbf{U}^{n+1} = B\mathbf{U}^n + f^n. \tag{1.12}$$

In this linear equation, matrices A and B are defined by our problem: we need to specify these matrices once for our problem and incorporate our boundary conditions in them. Vector $\mathbf{f}^n$ is a function of $\mathbf{U}^n$ and so needs to be reevaluated in every time point n. We also need to carry out one matrix-vector multiplication every time point, $B\mathbf{U}^n$, and one vector-vector addition, $B\mathbf{U}^n + f^n$. The most expensive numerical operation is inversion of matrix A to solve for $\mathbf{U}^{n+1}$, however we may get away with doing this only once and store the inverse of A as A^{-1}:

$$\mathbf{U}^{n+1} = A^{-1}\left(B\mathbf{U}^n + f^n\right). \tag{1.13}$$

2. Permeation equation

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

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + f(u) \tag{2.1}$$

$$\Gamma_{in}(t) + D\frac{\partial u}{\partial x}\Big|_{x=0} - k_u u^2(0,t) = 0 \tag{2.2}$$

$$-D\frac{\partial u}{\partial x}\Big|_{x=L} - k_d u^2(L,t) = 0 \tag{2.3}$$

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. Now we can take the system from previous chapter. The matrix in the non-boundary points is same as in equation 1.6. But we have to reevaluate the boundary coefficients according to our boundary condition 2.2.

$$\Gamma^n_{inc} - k_u U_0^{n2} + D\frac{U_1^n - U_0^n}{\Delta x} = 0, \tag{2.4}$$

$$k_d U_{J-1}^{n2} + D\frac{U_{J-1}^n - U_{J-2}^n}{\Delta x} = 0 \tag{2.5}$$

By solving both quadratic equations for U_0^n and U_{J-1}^n, we can evaluate their values based on the calculated concentration from the previous time layer:

$$U_0^{n+1} = -\frac{D}{2k_u\Delta x} + \frac{1}{2}\sqrt{\Big(\frac{D}{k_u\Delta x}\Big)^2 + \frac{4DU_1^n}{k_u\Delta x} + \frac{4\Gamma^n_{inc}}{k_u}} \tag{2.6}$$

$$U_{J-1}^{n+1} = -\frac{D}{2k_d\Delta x} + \frac{1}{2}\sqrt{\Big(\frac{D}{k_d\Delta x}\Big)^2 + \frac{4DU_{J-2}^n}{k_d\Delta x}} \tag{2.7}$$

3. Stencil cheat sheet.

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

3.1. Different σ

The σ is defined in the finite difference approximation of the diffusion equation 2.1. For the Crank-Nicolson method, the second order derivative is approximated as half sum of both forward and backward Euler approximations.

$$\frac{u_x^{t+1} - u_x^t}{\Delta t} = \frac{D}{2\Delta x^2}\big((u_{x+1}^{t+1} - 2u_x^{t+1} + u_{x-1}^{t+1}) + (u_{x+1}^t - 2u_x^t + u_{x-1}^t)\big) \tag{3.1}$$

So the $\sigma = \frac{D\Delta t}{2\Delta x^2}$ is convenient.

For explicit stencils we have differnt finite difference equation:

$$\frac{u_x^{t+1} - u_x^t}{\Delta t} = \frac{D}{\Delta x^2}\big(u_{x+1}^t - 2u_x^t + u_{x-1}^t\big) \tag{3.2}$$

In this case $\sigma = \frac{D\Delta t}{\Delta x^2}$ is convenient.

3.2. Stensils

Here I use $\sigma = \frac{D\Delta t}{\Delta x^2}$, one for all.

Forward Euler method (explicit):

$$\boxed{U_x^{t+1} = \sigma U_{x-1}^t + (1-2\sigma)U_x^t + \sigma U_{x+1}^t} \tag{3.3}$$

Backward Euler method (implicit):

$$\boxed{-\sigma U_{x-1}^{t+1} + (1+2\sigma)U_x^{t+1} - \sigma U_{x+1}^{t+1} = U_x^t} \tag{3.4}$$

Crank-Nicolson method (implicit, second order in time), $\sigma = \frac{D\Delta t}{\Delta x^2}$:

$$\boxed{-\frac{\sigma}{2}U_{x-1}^{t+1} + (1+\sigma)U_x^{t+1} - \frac{\sigma}{2}U_{x+1}^{t+1} = \frac{\sigma}{2}U_{x-1}^t + (1-\sigma)U_x^t + \frac{\sigma}{2}U_{x+1}^t} \tag{3.5}$$

3.3. Backward Euler

$$-\sigma U_{x-1}^{t+1} + (1+2\sigma)U_x^{t+1} - \sigma U_{x+1}^{t+1} = U_x^t \tag{3.6}$$

$$U_0^{t+1} = -\frac{D}{2k_u\Delta x} + \frac{1}{2}\sqrt{\left(\frac{D}{k_u\Delta x}\right)^2 + 4\frac{D}{k_u\Delta x}U_1^{t+1} + \Gamma^{t+1}} \tag{3.7}$$

$$U_L^{t+1} = -\frac{D}{2k_d\Delta x} + \frac{1}{2}\sqrt{\left(\frac{D}{k_d\Delta x}\right)^2 + 4\frac{D}{k_d\Delta x}U_{L-1}^{t+1}} \tag{3.8}$$