Modeling Impurity Migration in Multilayer Systems Using Parallelization

Călin Ioan Hojbotă, Valer Toșa

Departament of Molecular and Biomolecular Physics
National Institute for Research and Development of Isotopic and Molecular Technologies
Cluj-Napoca, Romania

Table of Contents

Introduction

- In science, diffusion appears in many systems (electrons, molecules, photons, etc.)

Introduction

- In science, diffusion appears in many systems (electrons, molecules, photons, etc.)
- A chemical agent can migrate from the packaging foil into the product which it wraps

Introduction

- In science, diffusion appears in many systems (electrons, molecules, photons, etc.)
- A chemical agent can migrate from the packaging foil into the product which it wraps
- In EU and USA, models help decision-making regarding safe packaging

Introduction

- In science, diffusion appears in many systems (electrons, molecules, photons, etc.)
- A chemical agent can migrate from the packaging foil into the product which it wraps
- In EU and USA, models help decision-making regarding safe packaging
- We present migration of chemical impurities in a multilayer system

Introduction

- In science, diffusion appears in many systems (electrons, molecules, photons, etc.)
- A chemical agent can migrate from the packaging foil into the product which it wraps
- In EU and USA, models help decision-making regarding safe packaging
- We present migration of chemical impurities in a multilayer system
- We try to improve execution times when the simulation is repeated a large number of times

Mathematical model

- Fick's Law of Diffusion:

$$\frac{\partial c}{\partial t} = D\nabla \cdot (\nabla c)$$

Mathematical model

- Fick's Law of Diffusion:

$$\frac{\partial c}{\partial t} = D\nabla \cdot (\nabla c)$$

- Initial concentration :

$$c(x,0) = c_0(x)$$

Boundary conditions:

$$\left.\frac{\partial c}{\partial x}\right|_{x=0} = 0$$

Interlayer condition (partition coefficient):

$$\frac{c_A(x = L_A)}{c_B(x = L_A)} = K_{AB}$$

Mathematical model

- Finite Differences equation (Crank-Nicolson):

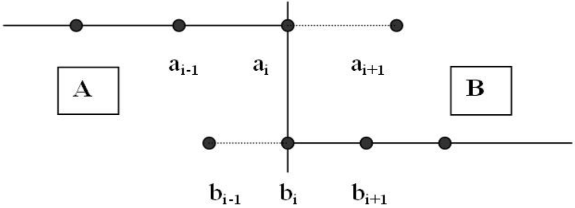
$$\frac{c_i^{n+1} - c_i^n}{\delta t} = D \left[\frac{c_{i-1}^{n+1} - 2c_i^{n+1} + c_{i+1}^{n+1}}{2\delta x^2} + \frac{c_{i-1}^{n} - 2c_i^{n} + c_i^{n}}{2\delta x^2} \right]$$

Mathematical model

- Finite Differences equation (Crank-Nicolson):

$$\frac{c_i^{n+1} - c_i^n}{\delta t} = D\left[\frac{c_{i-1}^{n+1} - 2c_i^{n+1} + c_{i+1}^{n+1}}{2\delta x^2} + \frac{c_{i-1}^n - 2c_i^n + c_i^n}{2\delta x^2}\right]$$

- Fictitious point method

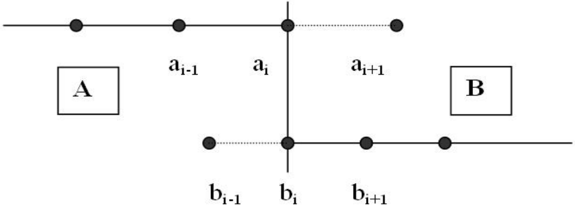


Mathematical model

- Finite Differences equation (Crank-Nicolson):

$$\frac{c_i^{n+1} - c_i^n}{\delta t} = D\left[\frac{c_{i-1}^{n+1} - 2c_i^{n+1} + c_{i+1}^{n+1}}{2\delta x^2} + \frac{c_{i-1}^n - 2c_i^n + c_i^n}{2\delta x^2}\right]$$

- Fictitious point method

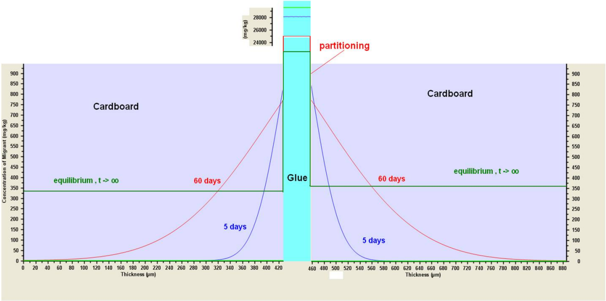


- We end up solving a system of equations

$$\mathbf{Ac^{n+1}} = \mathbf{Bc^n}$$

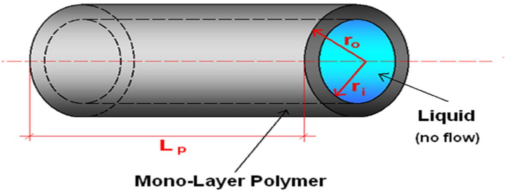
Mathematical model

Figure 1: Planar configuration : cardboard-glue-cardboard.



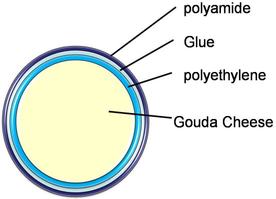
Mathematical model

Figure 2: Cylindrical configuration.



Mathematical model

Figure 3: Spherical configuration.



Implementation

- The program will be part of a large Monte Carlo simulation

Implementation

- The program will be part of a large Monte Carlo simulation
- The numerical routine was implemented in FORTRAN

Implementation

- The program will be part of a large Monte Carlo simulation
- The numerical routine was implemented in FORTRAN
- It will be called from the main application as a library file (*.dll)

Implementation

- The program will be part of a large Monte Carlo simulation
- The numerical routine was implemented in FORTRAN
- It will be called from the main application as a library file (*.dll)
- Main program written in C/C++

Implementation

- We tried to improve execution times using OpenMP

Implementation

- We tried to improve execution times using OpenMP
- Parallelization was implemented in the main application

Implementation

- We tried to improve execution times using OpenMP
- Parallelization was implemented in the main application
- Data is passed and read using arrays

Implementation

- We tried to improve execution times using OpenMP
- Parallelization was implemented in the main application
- Data is passed and read using arrays
- We must have a way to store large amounts of data taking into account the number of threads

Implementation

- We tried to improve execution times using OpenMP
- Parallelization was implemented in the main application
- Data is passed and read using arrays
- We must have a way to store large amounts of data taking into account the number of threads
- Race conditions must be avoided

Implementation

- 1. check the maximum number of threads

Implementation

- 1. check the maximum number of threads
- 2. dynamically create an array of file pointers

Implementation

- 1. check the maximum number of threads
- 2. dynamically create an array of file pointers
- 3. on each thread, write the results to its corresponding file

Implementation

- 1. check the maximum number of threads
- 2. dynamically create an array of file pointers
- 3. on each thread, write the results to its corresponding file
- After the computation is performed, assemble them into a single data file

Results

- Intel i3 CPU, (M 350, 2.27 GHz) - 2 physical cores

Results

- Intel i3 CPU, (M 350, 2.27 GHz) - 2 physical cores
- Simulated for 1000 runs

Results

- Intel i3 CPU, (M 350, 2.27 GHz) - 2 physical cores
- Simulated for 1000 runs
- $S_p = \frac{T_s}{T_p} = 2.4$

	With OpenMP(T_p)	**Without OpenMP(T_s)**
Exec. time(s)	5.2	12.5

Discussion

- We presented a model suitable for modeling migration
- The model was applied for a planar configuration can be (and was) extended for other geometries (cylindrical, spherical)
- Parallelization improved the execution times in shared memory systems
- Can be further extended to distributed memory systems (MPI)
- The work was performed within the EU FP7 Project FACET (Flavorings Additives and food Contact materials Exposure Task)

Thank You !