A Numerical Simulation of the Smoluchowski Coagulation Equation

Anh-Tuan Nguyen*, †

Abstract

In this report, a simulation on a conservation-law numerical scheme of the Smoluchowski coagulation equation is presented.

Key words: Finite volume method (FVM).

1 Initialization

1.1 Parameter initialization

In order to verify the correctness of the results, main input values are forced to be declared exactly as given by Filbet[1]. More precisely, they are listed in the following table:

Parameter	Concept	Value
R	Truncation size	50
I^h	Mesh size	≥ 125
T	Finish time	2
N	Number of time nodes	21
a(x,x')	Coagulation coefficient	a(x,x') $\equiv 1$
$f_0(x)$	Initial value of $f(t,x)$	$f_0(x) = M_0 e^{-M_0 x}, M_0 \in \mathbb{R}_+^*$

Table 1: Parameter initialization

It should be noted in this small project that the chosen mesh is uniform. Hence, a quantity Δx is defined to put the calculation more simply:

$$\Delta x := \Delta x_i \text{ for all } i = 0, 1, ..., I^h - 1. \tag{1}$$

Additionally, we would also like to mention that:

$$\Delta t := \frac{T}{N-1}, \tag{2}$$

*Applied Mathematics Research Division, Viettel Group, 1 Tran Huu Duc Str, Nam Tu Liem Dist, Hanoi 100000, Vietnam.

†Corresponding Email: anhtuan299@gmail.com.

where N is the number of nodes in the time domain.

1.2 Variable declaration

For any indices $i \in \{0, 1, ..., I^h\}$ and $n \in \{0, 1, ..., N-1\}$, the equality between the equations (12) and (13) both in [1] derives that a matrix $\mathbf{J} \in \mathbb{R}^{(Ih+1)\times N}$ could be declared to save the value of the approximate fluxes J so that:

$$\mathbf{J}_{i,n} := J^{h,n}_{i-1/2}. \tag{3}$$

As a result, for any value n we have:

$$\mathbf{J}_{0,n} := J^{h,n}_{-1/2} \approx J^R_{nc}(f)(x_{-1/2}) = \int_0^{x_{-1/2}} \{\text{function}\} = \int_0^0 \{\text{function}\} = 0. \tag{4}$$

Similarly, for any indices $i \in \{0, ..., I^h - 1\}$[1] and $n \in \{0, 1, ..., N\}$, the equations (11) and (13) both in [1] derive that the matrix $\mathbf{G} \in \mathbb{R}^{Ih\times(N+1)}$ should be used to save the approximation of $g(t, x)$ so that:

$$\mathbf{G}_{i,n} = g^n_i. \tag{5}$$

As a consequence, the values $\mathbf{G}_{i,0}$ could be computed from the equation (10) in [1] as followed:

$$\mathbf{G}_{i,0} = g^{0,h}_i = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} g_0(x)dx. \tag{6}$$

Finally, from the equation (11) in [1] and the definition of matrices $\mathbf{J}$ and $\mathbf{G}$:

$$\mathbf{G}_{i,n+1} = \mathbf{G}_{i,n} - \frac{\Delta t}{\Delta x} \left(\mathbf{J}_{i+1,n} - \mathbf{J}_{i,n}\right). \tag{7}$$

2 Implementation

2.1 Pseudo-codes

2.1.1 J-G Update

According to the equation (12) in [1], $J^{h,n}_{i+1/2}$ might be seen as a function J of the variables $\{g^n_i : i = 0, ..., I^h - 1\}$. Consequently, from the equations (11) and (12) both in [1], the alternating updates of $\mathbf{J}_{i,n}$ and $\mathbf{G}_{i,n}$ could be described as the Algorithm 1 given below.
The most difficult part of this implementation is definitely to compute $\mathbf{J}_{i,n}$ using the equation (12) in [1]. This could be done by a deep-nested loop which will be shown at the end of this section.

[1] The last index $i = I^h$ is kindly skipped here, which will be explained afterward.

Algorithm 1: $\mathbf{J}-\mathbf{G}$ Update

Input: I^h, Δt, Δx, $\left\{\mathbf{G}_{i,0} : i = 0, ..., I^h - 1\right\}$, $\{\mathbf{J}_{0,n} : n = 0, ..., N-1\}$.
Output: Matrices $\mathbf{G}$ and $\mathbf{J}$.

```
for n = 0 : N − 1
    for i = 0 : I^h − 1
        Update J_{i+1,n} ← J({G_{i,0} : i = 0, ..., I^h − 1})
        Update G_{i,n+1} ← G_{i,n} − (Δt/Δx)(J_{i+1,n} − J_{i,n})
```

For any fixed value i, the finite sequences $\{F1_k\}_{k=0,1,...,i}$ and $\{F2_k\}_{k=0,1,...,i}$ are declared to save the sum-of-integral and the integral in the deepest inner loops. In detail, from the equation (12) in [1] we define:

$$F1_k := \sum_{j=\alpha_{i,k}}^{I^h} \int_{\Lambda_j^h} \frac{a(x', x_k)}{x'} dx' g_j^n, \tag{8}$$

$$F2_k := \int_{x_{i+1/2}-x_k}^{x_{\alpha_{i,k}-1/2}} \frac{a(x', x_k)}{x'} dx' g_{\alpha_{i,k}-1}^n. \tag{9}$$

There is a confusion here in Filbet's convention: For the value $j = I^h$, the interval Λ_j^h will be $\Lambda_{I^h}^h = \left[x_{I^h-1/2}, x_{I^h+1/2}\right)$, which is invalid since $x_{I^h-1/2} = R$ is the last x-node in the mesh $(0, R)$. This happened at all the mathematical equations in [1] having the term "$\Lambda_{I^h}^h$". Therefore, in the above the author of this report removed all invalid terms whose index $i = I^h$. Fortunately, the implementation (will be given at the end of the report) seems to return the correct simulation as though skipping this term.

The sequence $\{F1_k\}_{k=0,1,...,i}$ is now re-defined as followed:

$$F1_k := \sum_{j=\alpha_{i,k}}^{I^h-1} \int_{\Lambda_j^h} \frac{a(x', x_k)}{x'} dx' g_j^n. \tag{10}$$

Since the mesh is chosen uniformly, it's easy to derive from the dependence $x_{i+1/2} - x_k \in \Lambda_{\alpha_{i,k}}^h$ that:

$$\alpha_{i,k} = i - k + 1. \tag{11}$$

2.1.2 Filbet's numerical scheme

Consequently, an 4-nested loop is in needed to calculate $\mathbf{J}_{i,n}$, which is expressed in the Algorithm 2 below.
It should be noted that the bolded $\mathbf{0}$ stands for a vector containing zeros only.

Algorithm 2: Filbet's numerical scheme

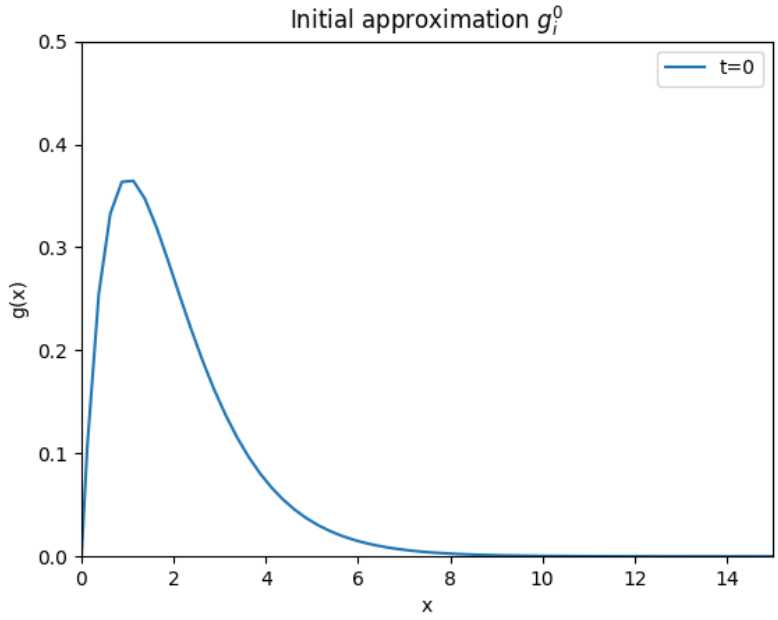
Input: R, I^h, T, N, $a(x,x')$, $g_0(x) = xf_0(x)$, Δt, Δx, $\left\{\mathbf{G}_{i,0} : i = 0, ..., I^h - 1\right\}$, $\{\mathbf{J}_{0,n} : n = 0, ..., N-1\}$.

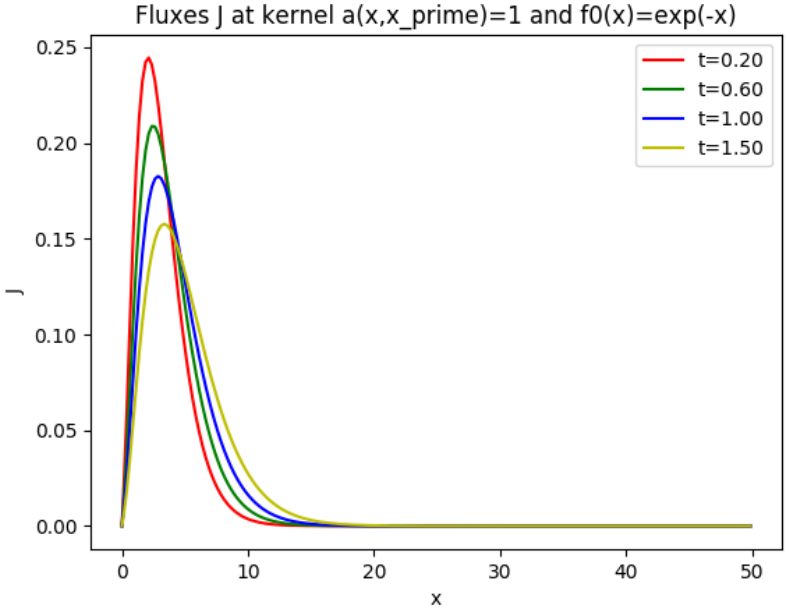
Output: Matrices $\mathbf{G}$ and $\mathbf{J}$.

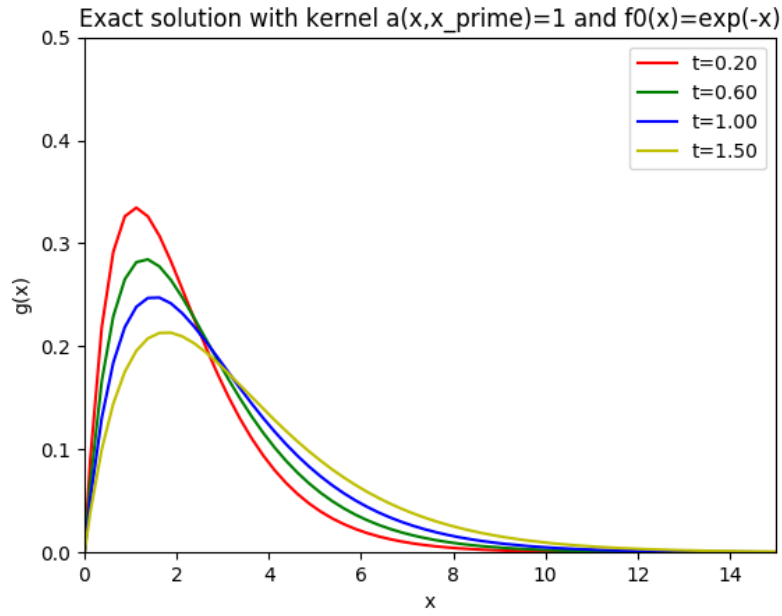
for $n = \overline{0 : N-1}$
  for $i = \overline{0 : I^h - 1}$
    $\mathbf{J}_{i+1,n} \leftarrow 0$
    $F2 \leftarrow \mathbf{0}$
    $F1 \leftarrow \mathbf{0}$
    for $k = \overline{0 : i}$
      $F2_k \leftarrow \int_{x_{i+1/2}-x_k}^{x_{\alpha_{i,k}-1/2}} \frac{a\left(x', x_k\right)}{x'} dx' \mathbf{G}_{\alpha_{i,k}-1,n}$
      for $j = \overline{\alpha_{i,k} : I^h - 1}$
        $F1_k \leftarrow F1_k + \int_{\Lambda_j^h} \frac{a\left(x', x_k\right)}{x'} dx' \mathbf{G}_{j,n}$
      $\mathbf{J}_{i+1,n} \leftarrow \mathbf{J}_{i,n} + \Delta x \cdot \mathbf{G}_{k,n} \left(F1_k + F2_k\right)$
    $\mathbf{G}_{i,n+1} \leftarrow \mathbf{G}_{i,n} - \frac{\Delta t}{\Delta x} \left(\mathbf{J}_{i+1,n} - \mathbf{J}_{i,n}\right)$

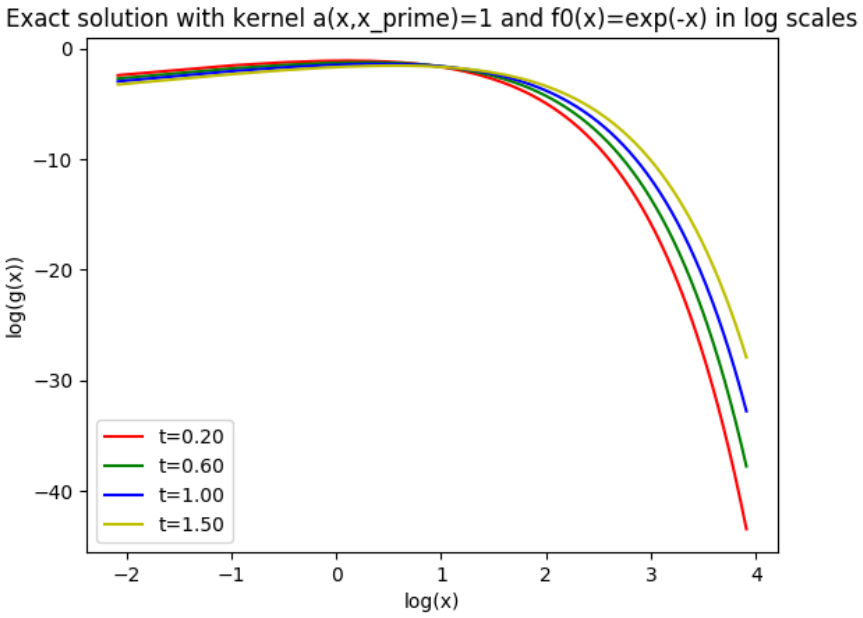
2.2 Results

The values of t are chosen as closed as Filbet's [1]. The implementation is performed on a computer with Intel® Core i7-8700 CPU@3.20GHz x 12, RAM 31,3 GB working in Linux operating system. The running time is around 5 minutes in the case that $I^h = 200$ and $N = 21$. The first, second and third pictures are corresponding to the equations (10), (12) and (11)+(13) in [1], respectively.


Initial approximation g_i^0
t=0
g(x)
x
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
10
12
14


Fluxes J at kernel a(x,x_prime)=1 and f0(x)=exp(-x)
t=0.20
t=0.60
t=1.00
t=1.50
J
x
0.00
0.05
0.10
0.15
0.20
0.25
0
10
20
30
40
50


Exact solution with kernel a(x,x_prime)=1 and f0(x)=exp(-x)
t=0.20
t=0.60
t=1.00
t=1.50
g(x)
x
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
10
12
14


Exact solution with kernel a(x,x_prime)=1 and f0(x)=exp(-x) in log scales
t=0.20
t=0.60
t=1.00
t=1.50
log(g(x))
log(x)
0
−10
−20
−30
−40
−2
−1
0
1
2
3
4

3 Conclusion

Comparing to the results by Filbet, the simulation in this report is exactly the same.

Acknowledgement

The author would like to thank Dr. Enrique Zuazua and Dr. Minh-Binh Tran for proposing this work.

References

[1] Filbet, F., Laurencot, P.. Numerical simulation of the Smoluchowski coagulation equation. *SIAM J. Sci. Comput.*, **25**, (6), pp. 2004-2028, (2004).