2017 29th International Symposium on Computer Architecture and High Performance Computing Workshops

Tuning up TVD HOPMOC method on Intel MIC Xeon Phi Architectures with Intel Parallel Studio Tools

Frederico L. Cabral , Carla Osthoff, Gabriel P. Costa
Laboratório Nacional de Computação Científica (LNCC)
Petropólis, RJ, Brasil
Email: {fcabral,osthoff,gcosta}@lncc.br

Diego Brandão
Centro Federal de Educação Tecnológica (CEFET/RJ)
Rio de Janeiro, RJ, Brazil
Email: diego.brandao@eic.cefet-rj.br

Mauricio Kischinhevsky
Instituto de Computação (IC-UFF),
Universidade Federal Fluminense,
Niterói, RJ, Brazil
Email: kisch@ic.uff.br

Sanderson L. Gonzaga de Oliveira
Departamento de Ciência da Computação,
Universidade Federal de Lavras, Lavras, MG, Brazil
Email: sanderson@dcc.ufla.br

***Abstract*—This paper focuses on the parallelization of TVD Method scheme for numerical time integration of evolutionary differential equations. The Hopmoc method for numerical integration of differential equations was developed aiming at benefiting from both the concept of integration along characteristic lines as well as from the spatially decomposed Hopscotch method. The set of grid points is initially decomposed into two subsets during the implementation of the integration step. Then, two updates are performed, one explicit and one implicit, on each variable in the course of the iterative process. Each update requires an integration semi step. This is carried out along characteristic lines in a Semi-Lagrangian scheme based on the Modified Method of Characteristics. This work analises two strategies to implement the parallel version of TVD Hopmoc based on the analysis performed by Intel Tools such as Parallel and Threading Advisor. A naive solution is substituted by a chunk loop strategy in order to avoid fine-grain tasks inside main loops.**

***Keywords*-Total Variation Diminishing; Hopmoc method; Intel Tools; Xeon Phi**

I. Introduction

The study of transport phenomena is essential in several problems in science and engineering. More specifically, the numerical solution of advection-diffusion transport arises from various important applications in engineering, physics, and chemistry. Significant examples of its use are found in geophysical flows, such as meteorology and oceanography, as well as in transport of contaminants in air, ground water, rivers, and lagoons, oil reservoir flow, aerodynamics, astrophysics, biomedical applications, in the modeling of semiconductors, and so forth. In environment or reactive fluid flow problems, contaminant or chemical species are principally transported by the fluid in which it is dissolved. Thus, the subject of study where the modeling of transport processes is ample. Therefore, it is a significant topic in numerical mathematics [12].

The Hopmoc method was proposed as a method to solve parabolic problems with convective dominance in parallel machines [14], [15]. This method addresses the issue of parallelization from the beginning of the process, by devising a spatial decoupling that allows message-passing minimization. The Hopmoc method is based on the Hopscotch method [7], [8], [9], [6], [10]. The Hopscotch method is a general-purpose scheme for the solution of second-order parabolic and elliptic partial differential equations. The Hopmoc method follows Hopscotch concepts in the sense that the set of unknowns is decoupled into two subsets. These subsets are calculated alternately by explicit and implicit approaches. The semi-steps are evaluated along characteristic lines by a Semi-Lagrangian scheme considering concepts of the Modified Method of Characteristics [13]. In this method, the time derivative and the advection term are combined as a direction derivative. To provide more specific detail, time steps are considered in the flow direction along characteristics of the velocity field of the fluid. In this approach, large time steps are accepted without inserting serious time truncation errors since the time-stepping direction is tracking along characteristic lines where the unknowns are modified gradually [2].

The Hopmoc method is a direct method in the sense that the cost per time step is known *a priori*. In addition, it employs a strategy based on tracking values along characteristic lines during time stepping. Moreover, the Hopmoc method is a type of Eulerian-Lagrangian localized adjoint method (ELLAM [3]) since it completely discretizes the domain along characteristic lines. More specifically, an ELLAM-like method provides a methodology that preserves the accuracy and efficiency of an Eulerian-Lagrangian approach, whereas also maintaining mass quantity and systematically handling any sort of boundary condition [18].

Discretization of the advective term in transport equations

978-1-5386-4819-3/17 $31.00 © 2017 IEEE
DOI 10.1109/SBAC-PADW.2017.12

is often plagued with serious difficulties. To avoid spurious numerical oscillations, Harten [11] introduced the concepts of Total Variation Diminishing (TVD) techniques and flux limiter, which provide monotonicity-preserving properties of stable higher-order accurate solutions of convection-diffusion problems. TVD techniques have been successfully applied in conjunction with numerical methods; recent examples are the publications of Fernandes et al. [5] and Silveira and Barros [19].

This present paper is a result of our work in implementing TVD techniques in conjunction with the Hopmoc method. Specifically, this work shows a total variation diminishing (TVD) scheme for numerical time integration of hyperbolic equations that does not involve solving linear systems and it's parallel implementation.

The remainder of this paper is divided as follows. Section II presents the Hopmoc method in details, including a figure that illustrates the main idea. Specifically, this section provides a brief background on the traditional Hopmoc scheme which we modify and improve in this manuscript. Section III describes two parallel strategies for the TVD Hopmoc method, assisted by Intel Tools analysis and respective results. Section IV discusses some related work. Finally, V discusses conclusion and future work.

II. The Hopmoc method

Consider the one-dimensional advection-diffusion equation in the form

$$u_t + vu_x = du_{xx}, \tag{1}$$

with adequate initial and boundary conditions, where v represents a constant positive velocity, d is a positive constant of diffusivity, and $0 \leq x \leq 1$. In equation (1), u_t refers to the time derivative and not u evaluated at the discrete time step t. Nevertheless, we abuse the notation and now use t to denote a discrete time step so that $0 \leq t \leq T$, for T time steps.

Consider a conventional finite difference discretization for this problem, with $\Delta t = u^{t+1} - u^t$ and $\delta t = \frac{\Delta t}{2} = u^{t+\frac{1}{2}} - u^t$ is a time semi-step of the method. The same characteristic line allows to obtain $\overline{u}\left(\overline{x}_i^{t+\frac{1}{2}}\right)$ and $\overline{\overline{u}}\left(\overline{\overline{x}}_i^{t}\right)$ in the previous two time semi-steps, for $\overline{x}_i^{t+\frac{1}{2}} = x_i - v \cdot \delta t$ and $\overline{\overline{x}}_i^{t} = x_i - 2v \cdot \delta t$, respectively, and the variable value $\overline{\overline{u}}\left(\overline{\overline{x}}_i^{t}\right)$ is obtained by interpolation [16], as described below. For clarity, a variable in a previous time semi-step and in a previous time step are written as $\overline{u}_i^{t+\frac{1}{2}} = u\left(\overline{x}_i^{t+\frac{1}{2}}\right)$ and $\overline{\overline{u}}_i^{t} = u\left(\overline{\overline{x}}_i^{t}\right)$, respectively. In addition, $\overline{u}_i^{t+\frac{1}{2}}$ is the variable in the previous time semi-step in the foot of the characteristic line originated at x_i^{t+1}. Additionally, we use a uniform spatial discretization so that $\Delta x = x_{i+1} - x_i$ [16].

In this work, discretization of diffusive terms is a three-point finite difference scheme. When t is even, $\overline{\overline{u}}_i^{t+1}$ is a numerical approximation of u in (x_i, u^{t+1}). Using the finite-difference operator

$$L_h(u_i^t) = d\frac{u_{i-1}^t - 2u_i^t + u_{i+1}^t}{\Delta x^2}, \tag{2}$$

both consecutive time semi-steps of the Hopmoc method can be written as $\overline{u}_i^{t+\frac{1}{2}} = \overline{\overline{u}}_i^t + \delta t\left(\theta_i^t L_h \overline{\overline{u}}_i^t + \theta_i^{t+1} L_h \overline{u}_i^{t+\frac{1}{2}}\right)$ or $u_i^{t+1} = \overline{u}_i^{t+\frac{1}{2}} + \delta t\left(\theta_i^t L_h \overline{u}_i^{t+\frac{1}{2}} + \theta_i^{t+1} L_h u_i^{t+1}\right)$, for $\theta_i^t = 1\ (=0)$ if $t+i$ is even (odd).

Figure 1 represents the Hopmoc method for a one-dimensional problem. The discretization of the advective term requires to calculate the values of the concentration at midpoints of the sides of each grid interval, which are obtained when employing a flux limiter.

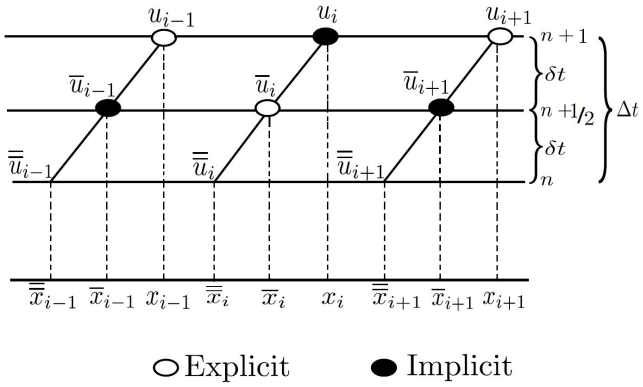

Figure 1. Variable values $\overline{\overline{u}}_i^t$ are used to calculate $\overline{u}_i^{t+\frac{1}{2}}$ in a first time semi-step and subsequently the variable values $\overline{u}_i^{t+\frac{1}{2}}$ are used to calculate u_i^t in the second time semi-step in the Hopmoc method.

Oliveira et al. [16] presented the convergence analysis of the Hopmoc method for an advection-diffusion equation.

HOPMOC method accuracy can be improved by a strategy called Total Variation Diminishing (TVD) leading to a TVD HOPMOC Method that does not present any difference from the traditional HOPMOC's parallelization. So, the strategy presented in this paper is focused on the TVD HOPMOC Method.

III. Parallel TVD Hopmoc

Since the Hopmoc and TVD Hopmoc methods does not solve any linear system [14], [15], [1], its parallelization is easily employed because it allows division of the unknowns into two disjoint subsets. Another advantage of the Hopmoc method is that its computational cost is linear in the number of unknowns per time step [1]. Each first step of the Hopmoc method employs a linear interpolation to obtain the initial estimative of the function value in the foot of the characteristic line.

Despite the fact that the method is easily parallelized its always a challenge to achieve this goal in modern machines when one considers the parallel efficiency and scalability.

To accomplish this on MIC Xeon Phi co-processors, one should use Intel Tools such as, Vtune, Thread Advisor and Parallel Advisor. These tools provide information about microarchitecture level that can assist programmers to find the best way to achieve required performance.

A. *Intel Tools*

This section presents the analysis performed by Intel Parallel Studuio 2017 version for Haswell/Broadwell architectures and identifies the parts inside the code that can permit optimization via vectorization and parallelization. To accomplish that, Intel tools such as Intel VTune Amplifier, reports generated by compiler, and Intel Advisor are used. Such tools perform some kinds of analysis inside the source and binary (executable) code and give hints on how to optimize the code. This task is mostly applied to performance bottlenecks analysis inside the CPU pipeline. Modern CPUs implement a pipeline together with other techniques like hardware threads, out-of-order execution, and instruction-level parallelism to use hardware resources as efficiently as possible. The main challenge for an application programmer is to take full advantage of these resources, because they rely on the microarchitecture level, which is far distant from the programming level available in modern and widely used programming languages such as C, C++, and Fortran*. Intel tools can help by analyzing compiler/linker and execution times to identify how those resources are being used. First, the compiler is able to inform a report where one can discover what was automatically done and what was not, for optimization goals. Sometimes a loop will be vectorized and other times it will not, and in both cases the compiler tells that and it's useful to the programmer to know whether an intervention is needed or not. Second, Intel VTune Amplifier shows the application behavior in the pipeline and on the memory usage. Its also possible to know how much time is spent in each module and whether the execution is efficient or not. Intel VTune Amplifier creates high-level metrics to analyze those details, such as cycles-per-instruction, which is a measure of how fast instructions are being executed. Finally, Intel Advisor can speculate and tell how much performance can be improved with thread parallelism. By combining all these tips and results, the programmer is able to make changes in the source code as well as in compiler options to achieve the optimization goal.

B. *Naive Attempt vs. Task Chunk versions of Paralleization*

The first and simplest approach to parallelize Hopmoc for Xeon Phi is to insert OpenMP pragmas in each loop that solves: (1) linear interpolation; (2) explicit operators; (3) implicit operators. The problem regarding this strategy is that each of these calculations are too fine grained to take advantage of parallelism.

Analysis performed by Intel Thread Advisor revealed that even with most part of the code vectorized, the estimated gain with OpenMP is strongly limited by this fine grain content of each loop. The Advisor itself suggested a strategy called task chunk to solve this issue. Chunking means that the parallel framework will merge several tasks into a single task, with little or no overhead between them. For instance, if tasks are loop iterations, chunking would mean that several iterations are executed together (as a chunk) before heavyweight task control is performed.

This naive implementation in Xeon Phi KNC with the offload code shown below confirms this prediction from Intel Advisor. The strategy adopted regards on offloading the main loop (time loop) into the co-processor and speedup was limited to about 30 threads not more than that.

```
1  #pragma simd
2  #pragma omp for
3  for (int i = head+1 ; i <= N-2 ; i+=2) {
4    ANNOTATE_ITERATION_TASK(loop_HOP_EXP_1);
5    U_old[i] = alfa*(U_new[i-1] + U_new[i+1]) +
         (1 - 2*alfa)*U_new[i];
6  }
7
8  #pragma omp single
9    head = (head+1)%2;
10
11 #pragma simd
12
13 #pragma omp for
14 for (int i = head+1; i <= N-2 ; i+=2) {
15   ANNOTATE_ITERATION_TASK(loop_HOP_IMP_1);
16   U_old[i] = (U_new[i] + alfa*U_old[i-1] + alfa
         *U_old[i+1])/(1+2*alfa);
17 }
18
19 #pragma omp single
20   head = (head+1)%2;
21
22 #pragma simd
23 #pragma omp for
24 for (int i = head+1 ; i <= N-2 ; i+=2) {
25   ANNOTATE_ITERATION_TASK(loop_HOP_EXP_2);
26   U_new[i] = alfa*(U_old[i-1] + U_old[i+1]) +
         (1 - 2*alfa)*U_old[i];
27 }
28
29 #pragma omp single
30   head = (head+1)%2;
31
32 #pragma simd
33 #pragma omp for
34 for (int i = head+1; i <= N-2 ; i+=2) {
35   ANNOTATE_ITERATION_TASK(loop_HOP_IMP_2);
36   U_new[i] = (U_old[i] + alfa*U_new[i-1] + alfa
         *U_new[i+1])/(1+2*alfa);
37 }
```

Suitability Analysis performed by Intel Threading Advisor, indicates that this Naive version presents 74.7% of load imbalance and 100% of runtime overhead, specifically thread schedule, due to small tasks inside each loop and so, parallelism would not be as efficient as naturally expected.

The same Advisor tool suggests a strategy called Task Chunk which means that all loops inside the analyzed site wold be theoretically merged into one singe loop. According to Advisor, if this is done the load imbalance is reduced to 3.7% with no runtime overhead at all.

Unfortunately, because data dependency its impossible to merge all loops inside the parallel site into a single one, and a middle ground strategy was adopted, which its being called herein as Chunk version.

```
1  t_ini = omp_get_wtime();
2  #pragma omp parallel
3  {
4          ANNOTATE_SITE_BEGIN(time_loop);
5          while (tempo < tempoFinal){
6          [...]
7          }
8          ANNOTATE_SITE_END();
9  }
10 t_fim = omp_get_wtime();
```

The annotations ANNOTATE SITE BEGIN ... ANNOTATE SITE END and ANNOTATE ITERATION TASK are inserted in order to permit Intel Parallel Advisor run the suitability analysis described in previous paragraph.

This analysis leaded to a version in which all Hopscoth operators are merged into two separated loops, instead of the traditional four loops. To accomplish this, a dependency data was analyzed in order to guarantee that the loops are suitable to parallelization, which means that the task inside the loop may be computed concurrently. Each of these two loops execute over a cluster of four point. The first one, called E-cluster, is composed by the following points given a i-th iteration (figure 2): for the $t + 1/2$ time step, explicit $i - 1$, explicit $i + 1$, implicit i and finally explicit i for the $t+1$ time step. The second one, called I-cluster, is composed by (3): implicit i for $t+1/2$ time step and explicit i, implicit $i - 1$ and implicit $i + 1$, for $t + 1$ time step.

```
1  #pragma omp for
2  for (int i = 4 ; i <= N-3 ; i+=4) {
3          U_til[i-1] = alfa*(U_til_til[i-2] +
               U_til_til[i]) + (1 - 2*alfa)*
               U_til_til[i-1];
4          U_til[i+1] = alfa*(U_til_til[i] +
               U_til_til[i+2]) + (1 - 2*alfa)*
               U_til_til[i+1];
5          U_til[i] = (U_til_til[i] + alfa*U_til[i
               -1] + alfa*U_til[i+1])/(1+2*alfa);
6          U_new[i] = alfa*(U_til[i-1] + U_til[i
               +1]) + (1 - 2*alfa)*U_til[i];
7  }
8
9  #pragma omp for
10 for (int i = 2 ; i <= N-5 ; i+=4) {
11         U_til[i] = (U_til_til[i] + alfa*U_new[i
               -1] + alfa*U_new[i+1])/(1+2*alfa);
12         U_new[i] = alfa*(U_til[i-1] + U_til[i
               +1]) + (1 - 2*alfa)*U_til[i];
13         U_new[i-1] = (U_til[i-1] + alfa*U_new[i
               -2] + alfa*U_new[i])/(1+2*alfa);
14         U_new[i+1] = (U_til[i+1] + alfa*U_new[i
               ] + alfa*U_new[i+2])/(1+2*alfa);
15 }
```

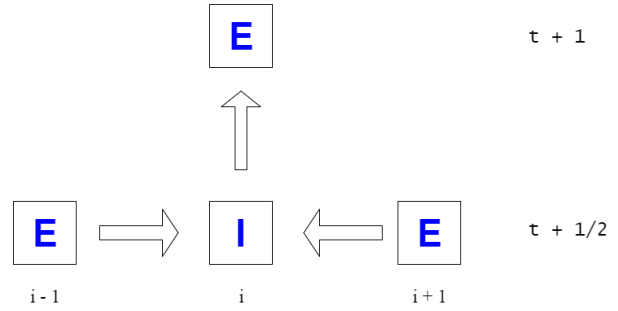

Figure 2. E-cluster points

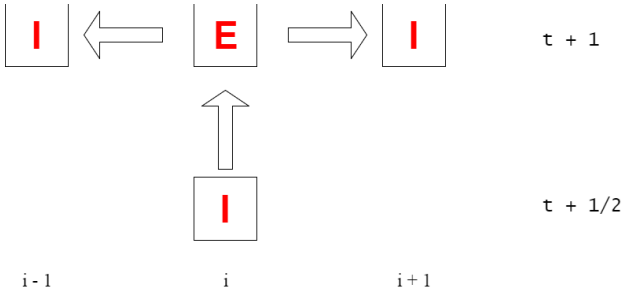

Figure 3. I-cluster points

Figure 4 exhibits the finite difference mesh composed by the two cluster points. The first loop in the chunk version is executed in parallel for all E-cluster points and then the second loop executes, also in parallel, all the I-cluster points. This cluster split strategy avoids data dependency permitting safe parallelism.

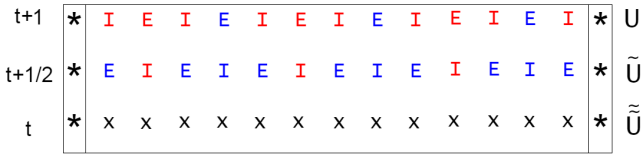

Figure 4. Mesh composed by two cluster points

Both ”Naive” and ”Chunk” versions presents 71% of vectorization efficiency, according to Intel Vector Advisor, but due to augmented granularity of tasks the chunk version is 50% faster than the naive version. This result is still far from the gain estimated by Intel Parallel Advisor but indicates that chunk task strategy is promising, specially for two and three spatial dimensions because the number of mathematical operations are naturally bigger than in one dimension case.

Table I shows execution times (in seconds) in a Xeon Phi KNC by two versions. Time was measured between only the offload parallel region.

Figure 5 shows the speedup in function of the number of threads for Naive version and figure 6 shows the same result

Simulation Results		
Threads	**Naive Version Time**	**Chunk Version Time**
1	769.17	784.94
5	163.38	163.89
10	88.74	87.20
20	46.24	42.21
30	39.05	32.57
40	34.90	26.42
50	33.70	24.94
60	31.93	22.98
120	29.52	19.08
180	28.51	18.84
240	29.60	18.79

Table I
COMPARISON BETWEEN NAIVE AND CHUNK VERSIONS

for the Chunk version. Both graphics exhibit results taken from table I. Maximum speedup for Naive version is up to 26 with 120 threads and for Chunk version is up to 42 also with 120 threads, which corresponds to a 50% increase in speedup.

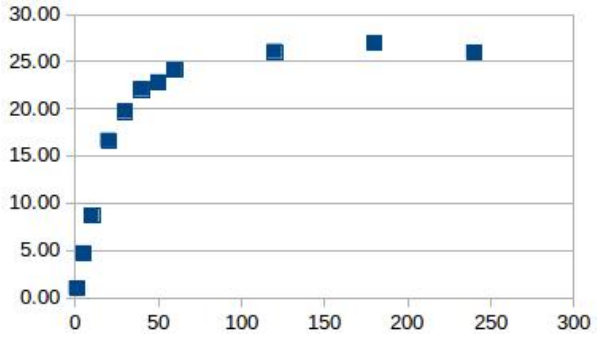

Figure 5. Naive version: Speedup x number of threads

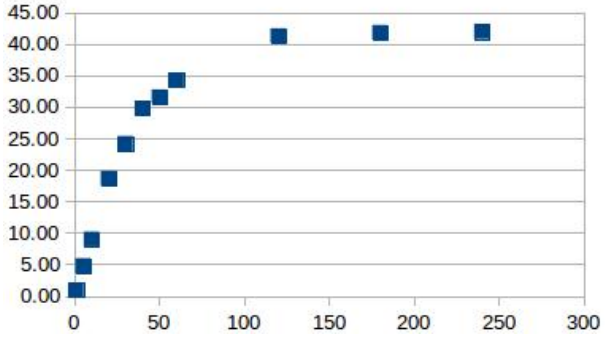

Figure 6. Chunk version: Speedup x number of threads

IV. RELATED WORK

Load imbalance due to fine-grained task and data issues have been investigated recently, specially for Xeon PHI coprocessors. In [24] authors discusse TILE Loop strategy in order to guarantee data locality for a matrix transposition code. This approach addresses a nested loop structure creating some kind of chunk task as well, leading to a slightly performance improvement in CPU but 30 % in coprocessor. This result seems to be similar to strategy described herein as augmented granularity was able to decrease load imbalance and therefore increased speedup.

V. CONCLUSION AND FUTURE WORK

Methods for Partial Differential Equations can be tricky as the tasks become too small in one-dimensional cases. To overcome deficiencies about obtaining speedup one can use Intel Tools such VTune, Parallel and Threading Advisor as those tools provide hints on how programmers can change original code in order to obtain desired speedup. In the case explored herein a merge of all individual small tasks proved to be a possible solution to obtain a task large enough to become speedup possible, without preventing vectorization.

Other approaches using positive schemes and universal limiters will be evaluated in future studies. In addition, we intend to evaluate those schemes when applying them to 2-D and 3-D parallel cases so that the scalability of this approach will be evaluated in future investigations. Additionally, we plan to compare how well the proposed new method performs compared with other approaches, especially one based on an approach of solving systems of linear algebraic equations and analyse all those possibilities with profiler tools in order to implement them in Intel Xeon Phi KNC and KNL architectures.

ACKNOWLEDGMENT

This work was developed with the support of CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico (National Council for Scientific and Technological Development, in Brazil), CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Coordination for Enhancement of Higher Education Personnel, in Brazil), FAPERJ - Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (Rio de Janeiro Research Support Foundation, in Brazil), and FAPEMIG - Fundação de Amparo à Pesquisa do Estado de Minas Gerais (Minas Gerais Research Support Foundation, in Brazil).

REFERENCES

[1] F. L. Cabral, C. Osthoff, M. Kischinhevsky, and D. Brandão. Hybrid MPI/OpenMP/OpenACC implementations for the solution of convection diffusion equations with Hopmoc method. In B. Apduhan, A. M.Rocha, S. Misra, D. Taniar, O. Gervasi, and B. Murgante, editors, *14th International Conference on Computational Science and Its Applications (ICCSA)*, CPS, pages 196–199. IEEE, July 2014.

[2] M. A. Celia, C. A. Brebbia L. A. Ferrand, W. G. Gray, and G. F. Pinder, editors. *Developments in Water Science, 35, Computational Methods in Water Resources, Modeling Surface and Sub-Surface Flows, Proceedings of the VII International Conference, MIT*, volume 1. Elsevier - Computational Mechanics Publications, Amsterdam, June 1988.

[3] M. A. Celia, T. F. Russel, I. Herrera, and R. E. Ewing. An Eulerian-Lagrangian localized adjoint method for the advection-diffusion equation. *Advances in Water Resources*, 13:187–206, 1990.

[4] R. Courant, E. Isaacson, and M. Rees. On the solution of nonlinear hyperbolic differential equations by finite differences. *Communications on Pure and Applied Mathematics*, 5(3):243–255, August 1952.

[5] B. Ramon B. Fernandes, A. D. R. Gonçalves, E. P. D. Filho, I. C. M. Lima, F. Marcondes, and K. Sepehrnoori. A 3d total variation diminishing scheme for compositional reservoir simulation using the element-based finite-volume method. *Numerical Heat Transfer, Part A*, 67(8):839–856, 2015.

[6] C. R. Gane and A. R. Gourlay. Block Hopscotch procedures for second order parabolic differential equations. *Journal of the Institute of Mathematics and its Applications*, 19:205–216, 1977.

[7] P. Gordon. Nonsymmetric difference equations. *SIAM Journal on Applied Mathematics*, 13:667–673, 1965.

[8] A. R. Gourlay. Hopscotch: a fast second order partial differential equation solver. *IMA Journal of Applied Mathematics*, 6:375–390, 1970.

[9] A. R. Gourlay and G. R. McGuire. General Hopscotch algorithm for the numerical solution of partial differential equations. *Journal of the Institute of Mathematics and its Applications*, 7:216–227, 1971.

[10] A. R. Gourlay and S. McKee. The construction of Hopscotch methods for parabolic and elliptic equations in two space dimensions with mixed derivative. *Journal of Computational and Applied Mathematics*, 3:201–206, 1977.

[11] A. Harten. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49:357–393, 1983.

[12] A. Holstad. The Koren upwind scheme for variable gridsize. *Applied Numerical Mathematics*, 37:459–487, 2001.

[13] J. Douglas Jr. and T. F. Russel. Numerical methods for convection-dominated diffusion problems based on combining the method of characteristics with finite element or finite difference procedures. *SIAM Journal on Numerical Analysis*, 19:871–885, 1982.

[14] M. Kischinhevsky. An operator splitting for optimal message-passing computation of parabolic equation with hyperbolic dominance. In *SIAM Annual Meeting*, Kansas City, MO, 1996.

[15] M. Kischinhevsky. A spatially decoupled alternating direction procedure for convection-diffusion equations. In *Proceedings of the XXth CILAMCE Iberian Latin American Congress on Numerical Methods in Engineering*, 1999.

[16] S. R. F. Oliveira, S. L. Gonzaga de Oliveira, and M. Kischinhevsky. Convergence analysis of the Hopmoc method. *International Journal of Computer Mathematics*, 86:1375–1393, 2009.

[17] P. L. Roe and M. J. Baines. Algorithms for advection and shock problems. In H. Viviand, editor, *Proceedings of the Fourth GAMM Conference on Numerical Methods in Fluid Mechanics*, volume 5 of *Notes on Numerical Fluid Mechanics*, pages 281–290, Paris, France, 1982. Vieweg, Vieweg.

[18] T. F. Russel and M. A. Celia. An overview of research on Eulerian–Lagrangian localized adjoint methods (ELLAM). *Advances in Water Resources*, 25:1215–1231, 2002.

[19] G. P. Silveira and L. C. de Barros. Analysis of the dengue risk by means of a Takagi-Sugeno-style model. Fuzzy Sets and Systems. Available online: http://www.sciencedirect.com/science/article/pii/S016501141500127X, March 2015.

[20] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 21(5):995–1011, 1984.

[21] J. Thuburn. TVD schemes, positive schemes and universal limiter. *Monthly Weather Review*, 125:1990–1993, December 1997.

[22] B. van Leer. Towards the ultimate conservative difference schemes. *Journal of Computational Physics*, 14:361–370, 1974.

[23] N. P. Waterson and H. Deconinck. Design principles for bounded higher-order convection schemes - a unified approach. *Journal of Computational Physics*, 224:182–207, 2007.

[24] A.Vladimirov, A.Ryo, V.Karpusenko. Parallel Programming and Optimization with Intel Xeon Phi Coprocessors. *Colfax International*, March 2015.