A domain decomposition strategy for a very high-order finite volumes scheme applied to cardiac electrophysiology

Yves Coudière, Rodolphe Turpault

A domain decomposition strategy for a very high-order finite volumes scheme applied to cardiac electrophysiology

Yves Coudière[a,b,d,e], Rodolphe Turpault[a,c]

[a] *Institut de Mathématiques de Bordeaux, F-33405 Talence, France*
[b] *Univ. Bordeaux, F-33400 Talence, France*
[c] *Bordeaux-INP, F-33400 Talence, France*
[d] *Carmen, Inria, F-33405 Talence, France*
[e] *IHU Liryc, F-33600 Pessac, France*

Abstract

In this paper, a domain decomposition technique for a very high-order finite volumes scheme is proposed. The objective is to obtain an efficient way to perform numerical simulations in cardiac electrophysiology. The aim is to extend a very high-order numerical scheme previously designed, where large stencils are used for polynomial reconstructions. Therefore, a particular attention has to be paid to maintain the scalability in parallel. Here, we propose to constrain the stencils inside the subdomains or their first layer of neighbors. The method is shown to remain accurate and to scale perfectly up to the level where there are not enough cells in the subdomains. Hence, these high-order schemes are proved to be efficient tools to perform realistic simulations in cardiac electrophysiology.

Keywords: Finite volumes, very high-order schemes, parallel computing, cardiac electrophysiology, MOOD
2000 MSC: 65M08, 65Y05, 35Q92, 92C99

1. Introduction

Numerical simulation plays an important role in the understanding and treatment of cardiac arrhythmia [1]. The efficiency of the cardiac pumping function is tightly related to the electrical synchronization of the cells. This phenomenon is due to the propagation of an electrical depolarization wave

that spreads in the heart. Indeed, numerous pathologies are associated to disorganizations of these waves. They may be described by a reaction-diffusion equation, namely the classical monodomain model:

$$\partial_t V + I_{ion}(V,w) = \operatorname{div}\big(D\nabla V\big),\ \mathbf{x}\in\Omega, \tag{1}$$

$$\partial_t w = G(V,w),\ \mathbf{x}\in\Omega, \tag{2}$$

where V is the transmembrane voltage in mV, $I_{ion}(V,w)$ is the normalized ionic current in $\mathrm{A\,F^{-1}}$, and $D=\sigma/A_mC_m$ is the normalized diffusion tensor in $\mathrm{cm^2\,ms^{-1}}$ (see [2]) . Equation (2) is the system of m nonlinear ordinary differential equations called the ionic model, and specified through the function $G(V,w)$, defined for $V\in\mathbb{R}$ and $w\in\mathcal{A}$, where $\mathcal{A}$ is a convex subset of $\mathbb{R}^m$. There exists a large diversity of ionic models, depending on the required level of complexity and cell type. In this paper, we will consider the simple phenomenological model of Aliev and Panfilov [3] and the more complex model of Ten Tusscher *et al* [4]. These two models are representative of the models used in biomedical simulations. In particular, the Ten Tusscher *et al* model is one of the few to describe human ventricular cells and its set of admissible states $\mathcal{A}$ is composed of positive ionic concentrations and gating variables that lie in $[0,1]$.

Since the phenomena under consideration involve complex dynamics with stiff gradients, very high-order numerical schemes capable of maintening the solution inside the admissible domain $\mathcal{A}$ are natural candidates. Moreover, long-time simulations have to be performed, hence parallel versions are required in many applications. In the field of cardiac electrophysiology, codes generally use at most schemes of order 2 and therefore unreasonably fine meshes have to be considered (or some physical parameters have to be tuned down). Furthermore, the reference codes (namely CARP[5], PROPAG [6] and CHASTE [7]) use parallel computing one way or another. The most comon strategy encountered in the domain is to rely on Petsc (see [6, 8, 9, 10]). Some codes use GPU through OpenCL or CUDA programming. This is the case of [11] where linear algebra involved in the implicit scheme is parallelized, and [12] where a parallel preconditionned multigrid conjugate gradient is developed. Finally, in [13] a Schwarz-based preconditionner is proposed.

In a previous article, an explicit very high order numerical method adapted to the monodomain model was described [2]. This numerical method

proved to be efficient and allows to perform simulations up to order 6 while retaining an admissible solution (*i.e.* positive ionic concentrations and gates in $[0, 1]$). The importance of using high-order schemes was underlined. Indeed, for a given level of accuracy, high-order methods require only reasonably refined meshes and hence save a lot of computational time.

The method proposed in [2] uses a cell-centered finite volume scheme based on a MOOD strategy (see [14] for the description of MOOD and [15, 16] for more related problems). High-order MOOD methods require large stencils for the sake of polynomial reconstructions. Hence, their effectiveness in parallel was questionable as opposed to some other techniques, *e.g.* Discontinuous Galerkin schemes for which there exists a natural way to do the parallel decomposition.

Indeed, for these MOOD schemes, a naive domain decomposition technique does scale very badly due to the large stencils.

The purpose of this paper is to develop an adapted domain decomposition technique in order to restore an efficient parallel scalability while maintaining the accuracy and properties of the sequencial method.

In the following section, the scheme proposed in [2] is briefly recalled, then an adapted domain decomposition technique is described. Several numerical tests are performed in order to show that the order and overall accuracy are maintained and an efficient scalability is obtained.

2. Preliminary description of the numerical method

We discretize equations (1) and (2) on a two-dimensional domain with the cell-centered finite volume scheme described in [2], which is a finite volumes scheme with a MOOD strategy. The domain is split into an unstructured mesh $\mathcal{M}$, and the unknonwns are functions which are piecewise constant on the cells K of the mesh, with values denoted by $V_K(t) \in \mathbb{R}$ and $w_K(t) \in \mathbb{R}^q$ for any $K \in \mathcal{M}$. They solve the semi-discrete finite volume scheme

$$\frac{d}{dt}V_K + I_K = \frac{1}{|K|}\sum_{e\in\mathcal{E}_K} |e| F_{K,e}, \tag{3}$$

$$\frac{d}{dt}w_K = G_K, \tag{4}$$

where I_K, G_K and $F_{K,e}$ are respectively approximations of the mean values of $I_{ion}(V, w)$ and $G(V, w)$ on the cell K and of the mean value of $D\nabla V \cdot n_{K,e}$ on the edge e; and where $\mathcal{E}_K$ is the set of the edges of K.

The computation of the terms I_K, G_K and $F_{K,e}$ relies on two series of weighted least squares polynomial approximations:

1. a polynomial approximation of degree m of the function V on each edge e of the mesh, that will be used to compute $F_{K,e}$;
2. and a polynomial approximation of degree p of the functions V and w on each cell K of the mesh, that will be used to compute I_K and G_K.

Approximation on each edge e. We define a stencil S_e, specifically a fixed collection of cells C that surround the edge e, and contains at least $(m+1)(m+2)/2$ cells, so as to have enough information to build a polynomial of degree m. Then, we construct the polynomial function $\widetilde{V}_e(\mathbf{x}) = \sum_{|\alpha|\leq m} \gamma_{\alpha,e}(\mathbf{x}-\mathbf{x}_e)^\alpha$ that best approximates the values of the unknown V_C on the cells $C \in S_e$ in a weighted least squares sense. Here, the point $\mathbf{x}_e$ is the midpoint of the edge e. The polynomial coefficients vector set $(\gamma_{\alpha,e})_{|\alpha|\leq m} := \Gamma_e \in \mathbb{R}^{(m+1)(m+2)/2}$ minimizes the function

$$J(\Gamma_e) = \frac{1}{2} \sum_{C \in S_e} \omega_{C,e} \left(\widetilde{V}_e(\mathbf{x}_C) - V_C \right)^2, \tag{5}$$

where $\mathbf{x}_C$ are the barycenters of the cells $C \in S_e$ and $\omega_{C,e} = \omega\left(|\mathbf{x}_C - \mathbf{x}_e|\right) \geq 0$ where ω is a decreasing weighting function, for instance $\omega(r) = r^{-s}$ for some $s > 0$. The approximate flux $F_{K,e}$ is finally defined by

$$F_{K,e} := \sum_{l=1}^{q_e} \xi_{l,e} \left(D(\mathbf{x}_{l,e}) \nabla \widetilde{V}_e \left(\mathbf{x}_{l,e} \right) \right) \cdot n_{K,e} \simeq \frac{1}{|e|} \int_e D(\mathbf{x}) \nabla \widetilde{V}_e(\mathbf{x}) \cdot n_{K,e}, \tag{6}$$

where $(\xi_{l,e})_{l=1\ldots q_e}$ and $(\mathbf{x}_{l,e})_{l=1\ldots q_e}$ are the weights and points of a quadrature formula of degree at least $m-1$ (see [2]). *The terms $F_{K,e}$ is ultimately a linear combination of the V_C for $C \in S_e$, and it depends on the choice of the stencils S_e.*

Approximation on each cell K. We similarly define a stencil S_K, a fixed collection of cells C that surround the cell K, and contains at least $(p+1)(p+2)/2$ cells, so as to have enough information to build a polynomial of degree p. Then we have to build a polynomial approximation for the variable $V \in \mathbb{R}$ and similarly for each of the variables in $w \in \mathbb{R}^q$. Here, we briefly recall the construction for V, which generalizes to w. It differs from the edge based reconstruction since it has to preserve the average value on the cell K, to garantee the order of accuracy of the method

(see [2]). Hence, we construct the polynomial function $\widetilde{V_K}(\mathbf{x}) = V_K + \sum_{1\leq|\alpha|\leq p} \lambda_{\alpha,K}\left[(\mathbf{x}-\mathbf{x}_K)^\alpha - \frac{1}{|K|}\int_K (\mathbf{x}-\mathbf{x}_K)^\alpha d\mathbf{x}\right]$ that best approximates the values of the unknown V_C on the cells $C \in S_K$ in a weighted least square sense. With the same notations as for the edges, the polynomial coefficients vector set $(\lambda_{\alpha,K})_{1\leq|\alpha|\leq p} := \Lambda_K \in \mathbb{R}^{(p+1)(p+2)/2-1}$ minimizes the function

$$J(\Gamma_K) = \frac{1}{2}\sum_{C\in S_K} \omega_{K,e}\left(\widetilde{V_K}(\mathbf{x}_C) - V_C\right)^2. \tag{7}$$

The function $\widetilde{w_K}(\mathbf{x})$ is reconstructed similarly, and finally, the approximate source terms I_K and G_K are defined by

$$I_K = \sum_{l=1}^{q_K} \xi_{l,K} I_{ion}\left(\widetilde{V_K}(\mathbf{x}_{l,K}), \widetilde{w_K}(\mathbf{x}_{l,K})\right), \tag{8}$$

$$G_K = \sum_{l=1}^{q_K} \xi_{l,K} G\left(\widetilde{V_K}(\mathbf{x}_{l,K}), \widetilde{w_K}(\mathbf{x}_{l,K})\right), \tag{9}$$

where $(\xi_{l,K})_{l=1\ldots q_e}$ and $(\mathbf{x}_{l,K})_{l=1\ldots q_e}$ are the weights and points of a quadrature formula of degree at least p. Again, *the terms I_K and G_K finally depend on the choice of the stencils S_K.*

Time-stepping method. The semi-discrete problem consists in solving the system of ODEs (3) and (4), with the flux and source terms computed from (6), (8) and (9). We discretize this system with standard strong stability preserving Runge Kutta (SSP RK) method from [17]. Specifically we will use SSP(k,k) methods, for $k = 1$ (forward Euler), $k = 2$ and $k = 3$, where the number of stages equals the order. Since, these are explicit methods where the CFL condition imposes $\Delta t = \mathcal{O}(\Delta x^2)$, we only need to have $2k$ at least equal to the order of discretization in space, hence we will finally use the forward Euler method up to order 2, the SSP(2,2) for the order 3 and 4, and SSP(3,3) for the order 5 and 6.

Degrees m and p, weights and quadrature rules. In practice, we look for polynomial reconstructions of the same degree $m = p := k$ on the edges and cells, so as to obtain a method of order of accuracy $k+1$. *We will investigate the properties of the methods of odrer $k+1 = 2, 4, 6$, built from polynomial of degreee $k = 1, 3, 5$.* To this aim, we also need quadrature rules that are exact

for polynomials of degree $k-1$ on the edges, and k on the cells. For all three schemes, we use the three-point Gauss formula on the edges (it is exact for polynomials of degree 5), and the 7 points formula of Dunavant [18] on the cells like proposed in our previous work [2] (it is also exact for polynomial of degree 5). We also fix the weighting function to $\omega(r) = r^{-1.5}$ on both the edges and the cells, since it is a good compromise as observed in [2].

Limitation strategy. For ionic models in our application (the functions F and G), there is a need to preserve values for part of the variables $w \in \mathbb{R}^q$ into a convex admissible set. Specifically, we want to preserve the gating variables in the interval $[0, 1]$, and positive ion concentrations. With the higher order methods, this will not be the case unless we limit the polynomial reconstruction when the admissible condition is violated when processing from time t^n to time t^{n+1}. To this aim, we adopt a MOOD strategy. Actually, assuming that all $w_C(t^n)$ are admissible, we replace the value $\widetilde{w_K}(\mathbf{x}_{l,K})$ reconstructed from the $w_C(t^n)$, whenever it is not admissible, by the average value $w_K(t^n)$. Since we use an SSP-RK method (and the admissible set is convex), the next value $w_K(t^{n+1})$ is admissible.

Stencils S_e and S_K. In order to build the stencil S_e of an edge e, we first consider the initial set S_e^0 of the two neighbors of e, then build iteratively the sets S_e^k by adding the neighbors of rank k of these two cells (two cells are neighbours iff they share a common node). We take $S_e = S_e^k$ the smallest such set that contains at least $(m+1)(m+2)/2$ cells. The stencil S_K of a cell K is built similarly, with $S_K^0 = \{K\}$ and S_K^k the neighbours of S_K^0 of ranks $\leq k$. The stencil S_K is the smallest S_K^k that contains at least $(p+1)(p+2)/2$ cells.

Implementation details. In practice, the minimization problems (5) and (7) amounts to solving local symetric and positive definite linear systems (which sizes are the number of elements in the stencils S_e or S_K), only depending on the geometry. They are solved in a preprocessing step and their inverses are stored. Hence each stage in the time-stepping methods requires one matrix vector products of size S_e for each edge e, and one of size S_K for each cell K of the mesh, quadrature formulas on the edges e, computation of the nonlinear terms F and G at the quadrature points, and application of the quadrature formulas in the cells K.

3. The domain decomposition technique

If the mesh is split so as to define geometrical subdomains of Ω (see example on figure 1), the computation of the flux (6),(8) and (9) near the boundary of each subdomain requires values of the unknowns in the neighbouring subdomains. More precisely, every unknown inside each of the stencils S_K and S_e has to be communicated. Hence, if nothing special is done, this results in heavy communications between threads which drastically slow down the computations.

In the following a strategy is proposed to avoid such an issue.

3.1. What happens near the boundary of the domain ?

Before going into the description of this strategy, let us investigate some features of the way reconstruction stencils are designed. Indeed, this is obviously a key point in terms of efficiency.

Let us first remark that even for the scheme on one domain with the stencils chosen as in [2] there is already a discrepancy in the treatment of these stencils between the boundaries and the interior of the domain. In order to avoid considering a large amount of ghost cells, only cells inside the domain are eligible for the stencil (plus the boundary condition). As a result, at the domain's boundaries, the stencils are not well balanced around the corresponding cells/interfaces. As order increases, cells located quite far from the boundary may be part of the stencils.

This does not change the order of the scheme since the degree of the polynomial reconstructions are unchanged. However, the use of distant information is expected to result in a reduction of the accuracy.

Hopefully, the least-squares procedures which determines the polynomial coefficients uses weights that decrease with the distance, so that remote cells only have a marginal influence. In practice, no significant reduction of the accuracy was observed at boundaries in numerical simulations [2].

3.2. The case of two subdomains

This observation suggests that it is possible to avoid deteriorating the order while imposing the stencils to lie inside a specific subdomain or the first layer of neighbors for the sake of consistency.

In order to validate this hint, a simple test-case is performed in a domain subdivised into two subdomains. As illustrated on figures 2 and 3, the reconstruction stencils are forced to belong to the related subdomain or the first layer of neighbors:

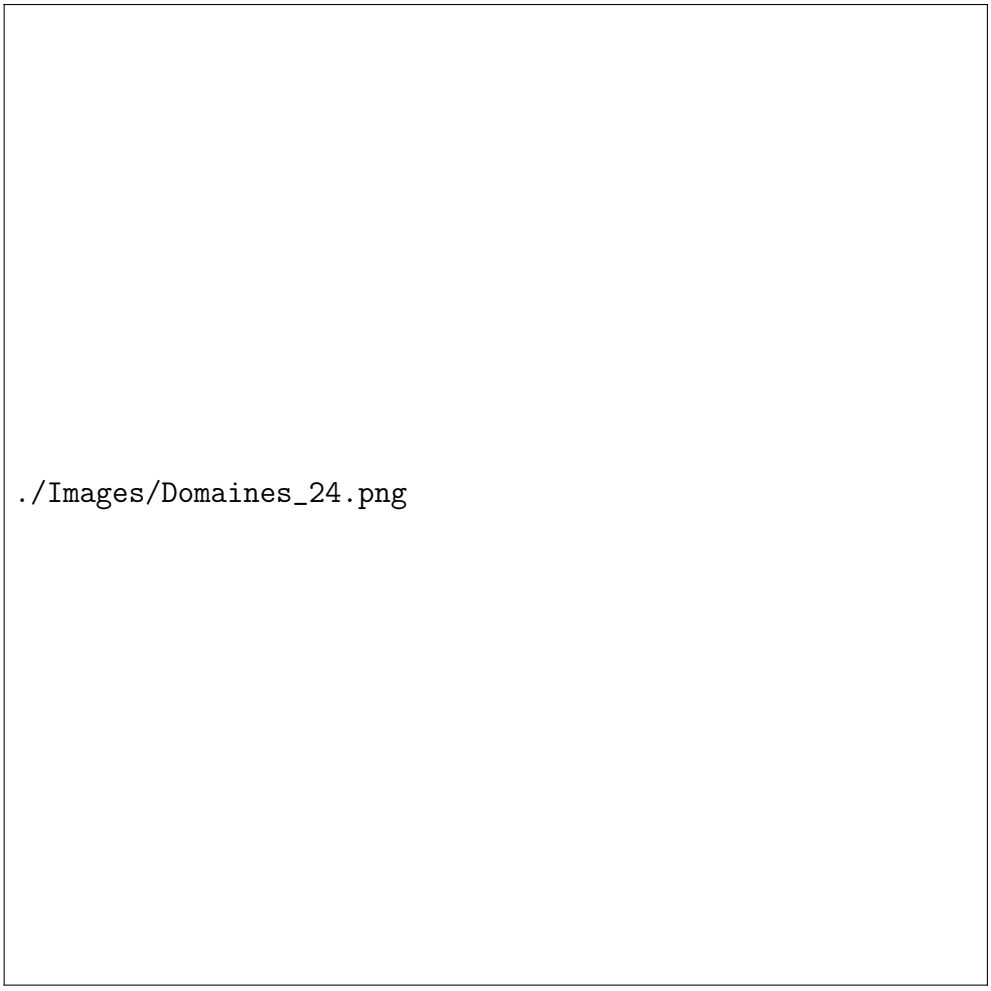


Figure 1: Example of mesh decomposition with 24 subdomains

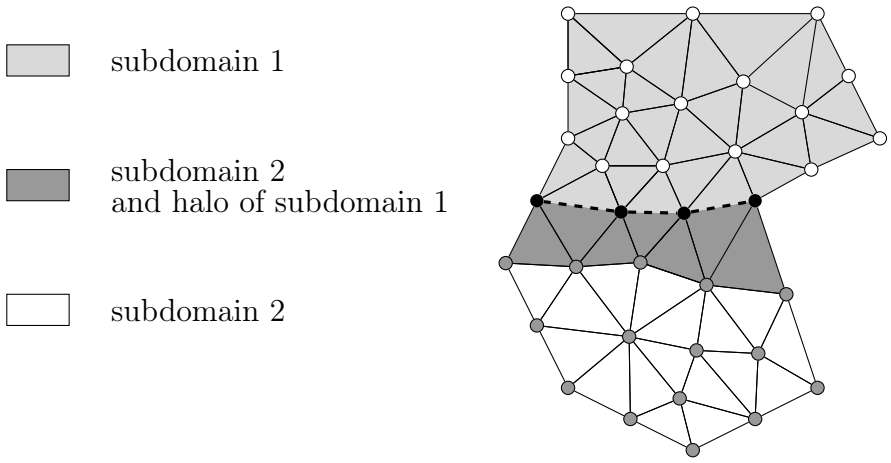


Figure 2: Subdomains and halo.

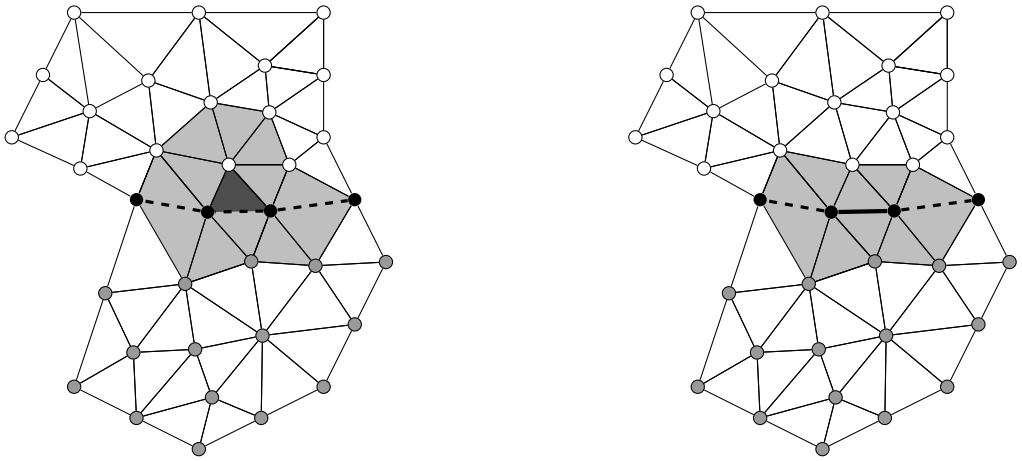
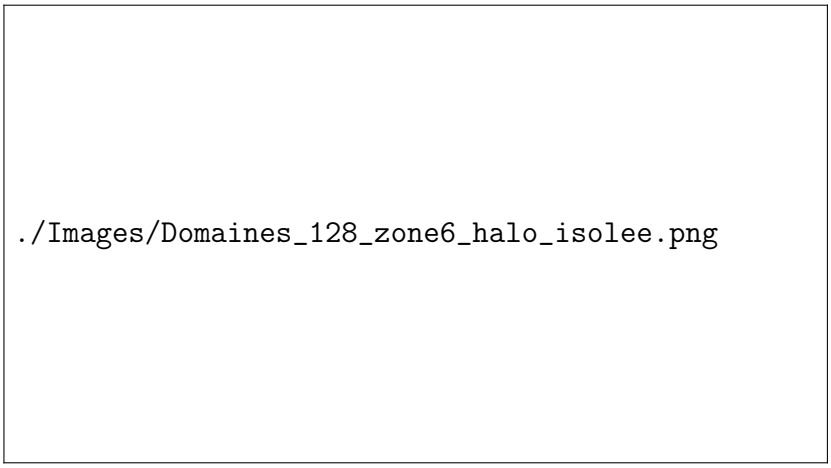
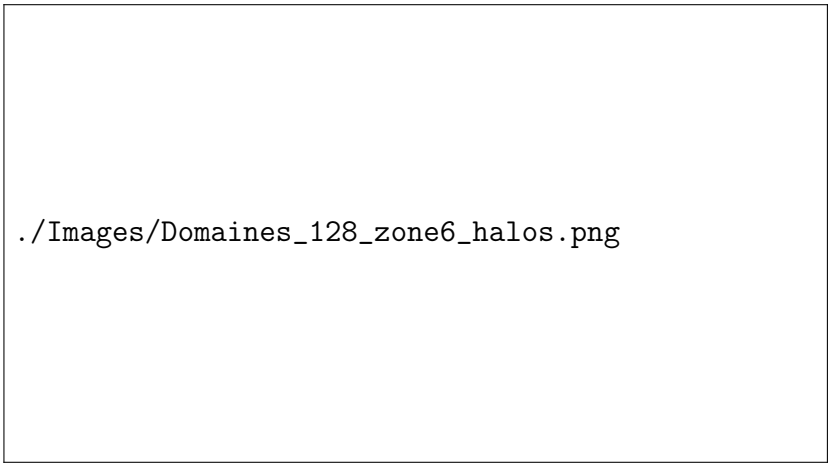


Figure 3: Stencils of a cell on the boundary of subdomain 1 (left) and stencils of an edge of the interface between the two subdomains (right).



(a) A subdomain in a decomposition of 128, and its halo.



(b) All subdomains around the previous one, and their halos.

Figure 4: Exemple of domains and halos

- the stencils associated to the cells are forced to belong to the cell's subdomain and if necessary to the first layer of neighbours (halo),
- the stencils associated to the edges are forced to belong the union of the two first layers of neighbours.

In order to evaluate the loss of accuracy of such a choice, an analytical test-case is performed. The ionic current is chosen such that:

$$\begin{aligned} V(t,x) &= e^{-\lambda t}\cos(8\pi x)\cos(8\pi y), \\ I_{ion}(V,w) &= \Delta V - \partial_t V. \end{aligned}$$

This analytical test-case was performed and detailed in [2] where the following error is computed:

$$e^2 := \sum_{K\in\mathcal{M}} |K||V_K - \sum_{l=1}^{q_K} \xi_{l,K}\widetilde{V_K}(\mathbf{x}_{l,K})|.$$

The results are in tables 1 and 2. The orders are conserved and, even if the accuracy is slightly decreased, the difference between the two simulations stays neglectable.

h	1	order	3	order	5	order
6.225E-003	5.920E-001	n/a	1.365E-001	n/a	4.562E-002	n/a
3.007E-003	1.058E-001	2.366	1.169E-002	3.378	1.053E-003	5.179
1.504E-003	1.859E-002	2.509	6.497E-004	4.170	2.359E-005	5.481

Table 1: L^2 errors for the analytical test case - one domain.

h	1	order	3	order	5	order
6.225E-003	5.922E-001	n/a	1.392E-001	n/a	4.771E-002	n/a
3.007E-003	1.058E-001	2.367	1.173E-002	3.400	1.102E-003	5.178
1.504E-003	1.859E-002	2.509	6.538E-004	4.167	2.447E-006	5.496

Table 2: L^2 errors for the analytical test case - two subdomains.

3.3. General case

In the general case, the domain Ω is partitionned into N subdomains $(\Omega_k)_{k=1...N}$.

A cell neighbors graph is defined as follows: two cells K and L are neighbors if they share a vertex. This relation is denoted by $K|L$.

Each closed subdomain Ω_k is then associated to an extended subdomain $\widehat{\Omega}_k$ consisting in all cells in Ω_k and their neighbors:

$$\widehat{\Omega}_k = \{L|K \ \forall K \in \Omega_k\}.$$

With these notations, the halo of Ω_k is simply $\widehat{\Omega}_k \backslash \Omega_k$. Now, the stencils are chosen such that:

- $\forall K \in \Omega_k, \ S_K \subset \widehat{\Omega}_k$,
- $\forall e \in \Omega_k, \ S_e \subset \widehat{\Omega}_k$.

Let us emphasize that if an edge e is located at the interface between subdomains Ω_k and Ω_l (*i.e.* $e \in \Omega_k \cap \Omega_l$), then S_e belongs to $\widehat{\Omega}_k \cap \widehat{\Omega}_l$.

As a consequence, we expect $\widehat{\Omega}_k \cap \widehat{\Omega}_l$ to contain at least $(m+1)(m+2)/2$ cells for a polynomial of degree m on the interfaces. Exceptionally, when there are too many subdomains, the intersection $\widehat{\Omega}_k \cap \widehat{\Omega}_l$ may contain too few cells to fulfill this condition. In this case, we use the second layer of neighbors.

For illustration, an example of subdomain and its halo is given on figure 4.

The results in tables 3 and 4 correspond to the analytical test-case described in the previous paragraph with 24 subdomains. Once again, the orders are conserved and, even if the accuracy is slightly decreased, the difference with the reference case (table 1) stays neglectable.

h	1	order	3	order	5	order
6.225E-003	5.927E-001	n/a	1.470E-001	n/a	5.946E-002	n/a
3.007E-003	1.058E-001	2.368	1.195E-002	3.450	1.415E-003	5.138
1.504E-003	1.859E-002	2.509	6.776E-004	4.141	3.084E-005	5.520

Table 3: L^2 errors for the analytical test case - 24 subdomains.

h	1	order	3	order	5	order
6.225E-003	7.609E-004	n/a	1.049E-002	n/a	1.384E-002	n/a
3.007E-003	8.267E-006	6.215	2.575E-004	5.095	3.612E-004	5.011
1.504E-003	6.337E-007	3.706	2.784E-005	3.210	7.248E-006	5.639

Table 4: L^2 errors for the analytical test case - difference between 1 and 24 subdomains.

3.4. Practical partitionning

As far as scalability is concerned, one of the difficulties in the scheme is the presence of two reconstructions. Ideally, when Ω is divided into N, one would want each subdomain to contain $\frac{1}{N}$ cells and edges. Unfortunately, this is not possible in most cases. For the sake of numerical experiments, the software Scotch was used [19] in order to define the subdomains. The default feature of Scotch was selected here, *i.e.* the partionning of a graph. In our case, the graph's vertices are the cells and the graph's edges are the interfaces between cells. This graph is different from the neighbouring graph defined in section 3.3.

In any case, we cannot have 100% scalable code since it would require a perfect balance of both cells and intefaces, though in practice we are not far from it. For instance, tables 6 and 7 show the minimum and maximum number of cells and edges in subdomains in the case of the two meshes that are used for numerical studies in this article (see table 5).

In these tables, an efficiency coefficient is computed as follows:

$$\text{eff}_{edge} = \frac{\#\text{edges in subdomain}}{\#\text{edges}} * 100 * \#\text{ threads}.$$

These numbers should ideally be 100 if cells and edges are evenly dispatched. Let us mention that edges which are located at the interface between two subdomains are arbitrarily affected in one subdomain or the other. These numbers are representative of the number of communications between two subdomains. Hence, it is possible to *a priori* have an idea of the deterioration of the parallel computing's efficiency.

Tables 6 and 7 show that one can expect each subdomain to contain at least $\frac{0.95}{N}$ edges and $\frac{0.99}{N}$ cells as long as each subdomain contains a reasonable number of cells.

From the implementation standpoint and in the sake of efficiency, let us mention that cells and edges are renumbered so that Ω_k contains consecutive

	# cells	# edges	h (cm)	dt (ms)
Mesh #1	37 928	57 132	2.051e-2	1.104e-2
Mesh #2	151 740	228 090	5.344e-3	2.856e-3

Table 5: The two meshes considered in the numerical studies below.

cell and edges numbers. This features favors memory locality, which is very important for general effectiveness, especially when OpenMP directives are used.

4. Numerical study

The goal of this section is to investigate:

- the accuracy of the method, even in parallel,
- its scalability,
- the best ratio accuracy / CPU time in parallel,

hence we will focus on these issues on a specific test-case.

Taking advantage of the experience on one single domain [2], we chose to perform the simulation of a spiral wave. To this aim, the standard Aliev-Panfilov model is modified in order to decrease the action potential duration ($k = 10$). The spiral wave is initiated by exciting a region which straddles the refractory and resting zones of a first planar wave.

As observed in [2], this test-case is a difficult one, very sensitive to the mesh and order. In particular, in the sequential case and on the considered meshes, a significant discrepancy is observed even between 5th and 6th order at time $t = 150$ ms.

We will consider the same meshes used in the previous article which features are summarized in table 5. Each mesh was partitionned in 1, 2, 4, 8, 12, 16, 24 for OpenMP simulations on CPU and in addition 32, 48, 64, 128, 256 subdomains for OpenMP simulations on KNL nodes. In practice, the CPU used were dual dodecacore Haswell Intel® Xeon® E5-2680 v3 @ 2.5 GHz and the KNL where Xeon Phi ™ 7230 @ 1.3 GHz.

In the sequencial simulations, huge discrepancies were visible on this particular mesh even between order 5 and 6 (the spirals obviously had a different number of revolutions). Here, as for the analytical test-case, we observe on

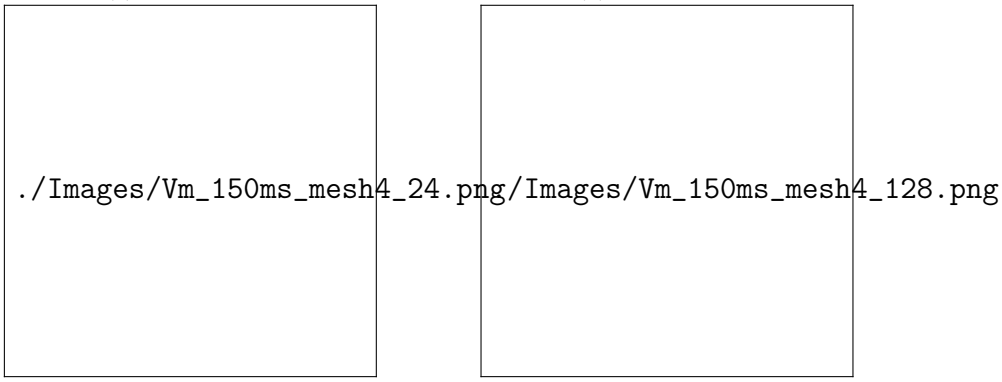
	# cells per subdomain		# boundary edges		# edges per subdomain	
# thrds	min	max	min	max	min	max
2	18 955	18 973	0	142	28 482	28 508
eff	99.95	100.05		0.50	99.71	99.80
4	9 460	9 503	0	142	14 175	14 249
eff	99.77	100.22		0.99	99.24	99.76
8	4 705	4 751	0	168	7 033	7 155
eff	99.24	100.21		2.35	98.48	100.19
12	3 137	3 183	0	152	4 625	4 753
eff	99.25	100.71		3.19	97.14	99.83
16	2 355	2 383	0	102	3 467	3 565
eff	99.35	100.53		2.86	97.09	99.84
24	1 565	1 595	0	99	2 293	2 370
eff	99.03	100.93		4.16	96.32	99.56
32	1 174	1 194	0	99	1 714	1 788
eff	99.05	100.74		5.55	96.00	100.15
48	783	797	0	87	1 134	1 176
eff	99.09	100.86		7.31	95.27	98.80
64	587	597	0	73	848	896
eff	99.05	100.74		8.18	94.99	100.37
128	294	298	0	54	416	443
eff	99.22	100.57		12.10	93.20	99.25
256	145	149	0	38	196	220
eff	97.87	100.57		17.03	87.82	98.58

Table 6: Minimum and maximun number of cells and edges in subdomains - Mesh 1. Boundary edges are edges located on the interfaces between two subdomains. These edges are arbitrarily affected inside one subdomain or the other.

	# cells per subdomain		# boundary edges		# edges per subdomain	
# thrds	min	max	min	max	min	max
2	75 830	75 910	0	294	113 837	113 959
eff	99.95	100.05		0.26	99.82	99.92
4	37 796	38 029		300	56 661	56 960
eff	99.63	100.25		0.53	99.37	99.89
8	18 873	19 075		368	28 282	28 510
eff	99.50	100.57		1.29	99.20	100.00
12	12 614	12 667		232	18 794	18 952
eff	99.75	100.17		1.22	98.88	99.71
16	9 414	9 533		214	14 025	14 236
eff	99.26	100.52		1.50	98.38	99.86
24	6 278	6 379		147	9 298	9 471
eff	99.30	100.89		1.55	97.84	99.66
32	4 711	4 784		217	6 964	7 139
eff	99.35	100.89		3.04	97.70	100.16
48	3 135	3 188		166	4 624	4 727
eff	99.17	100.85		3.49	97.31	99.48
64	2 347	2 393		139	3 461	3 575
eff	98.99	100.93		3.90	97.11	100.31
128	1 174	1 196		105	1 708	1 774
eff	99.03	100.89		5.89	95.85	99.55
256	587	600		71	842	882
eff	99.03	101.23		7.97	94.50	98.99

Table 7: Minimum and maximun number of cells and edges in subdomains - Mesh 2
Boundary edges are edges located on the interfaces between two subdomains. These edges are arbitrarily affected inside one subdomain or the other.

(a) 1 subdomain. (b) 4 subdomains.



(c) 24 subdomains. (d) 128 subdomains.

Figure 5: 6th order method: spiral wave (AP model) predicted at time $t = 150$ ms with different domain decompositions.

#threads	reaction	diffusion	total	scale factor	scalability
1	176.67	117.86	294.53	1.00	–
2	87.33	59.16	146.49	2.01	1.01
4	35.96	24.31	60.27	4.89	1.22
8	18.07	13.25	31.32	9.41	0.96
12	12.15	12.27	24.41	12.06	0.86
16	9.17	10.51	19.68	14.97	0.93
24	6.12	14.94	21.05	13.99	0.62

Table 8: Mesh 1, order 2

#threads	reaction	diffusion	total	scale factor	scalability
1	623.16	480.70	1103.86	1.00	–
2	327.98	260.08	588.06	1.88	0.94
4	169.02	125.43	294.45	3.75	1.00
8	86.03	67.82	153.86	7.17	0.96
12	65.72	59.41	125.13	8.82	0.82
16	46.64	43.83	90.47	12.20	1.04
24	34.50	45.45	79.95	13.81	0.75

Table 9: Mesh 1, order 4

figure 5 that the influence of domain decomposition stays neglectable, even on 128 subdomains. The result was far from being trivial given the stiffness of the case and the large stencils used for 6th order.

Now, scalability tests were performed in order to compare the schemes of order 2, 4 and 6 on both meshes with 1 to 24 subdomains. The CPU times and scalabilities of the methods are given on tables 8, 9 and 10 for mesh 1 and 11, 12 and 13 for mesh 2. The scale factors are also plotted on figures 6 and 7. They are defined as the ratio of the total time on 1 thread by the total time on N threads.

On the coarse mesh (Mesh 1), one can observe that the scalability is relatively good up to 16 threads and the suddently stalls. This is due to the fact that the subdomains become relatively small and there is not enough computations to be performed in each subdomain to keep the method effective. One also clearly sees that considering a refined mesh (Mesh 2) or more computations to do (order 6) helps restoring a good scalability. The graphs on figure 8 show the CPU time of computations. More precisely, the cost

#threads	reaction	diffusion	total	scale factor	scalability
1	3342.23	1938.90	5281.13	1.00	–
2	1699.76	1003.36	2703.11	1.95	0.98
4	877.15	546.53	1423.68	3.71	0.95
8	383.74	246.47	630.21	8.38	1.13
12	262.21	166.35	428.56	12.32	0.98
16	206.00	126.12	332.12	15.90	0.97
24	134.43	108.57	243.00	21.73	0.91

Table 10: Mesh 1, order 6

#threads	reaction	diffusion	total	scale factor	scalability
1	2872.00	2045.38	4917.38	1.00	–
2	1464.47	1077.84	2542.31	1.93	0.97
4	834.29	523.11	1357.40	3.62	0.94
8	400.06	273.49	673.55	7.30	1.01
12	273.72	189.79	463.51	10.61	0.97
16	202.51	151.56	354.07	13.89	0.98
24	138.00	139.52	277.52	17.72	0.85

Table 11: Mesh 2, order 2

#threads	reaction	diffusion	total	scale factor	scalability
1	11319.77	7758.15	19077.92	1.00	–
2	5800.49	4199.91	10000.40	1.91	0.95
4	2725.47	2381.66	5107.13	3.74	0.98
8	1414.60	1318.90	2733.50	6.98	0.93
12	928.06	811.24	1739.31	10.97	1.05
16	692.03	637.78	1329.81	14.35	0.98
24	478.69	502.71	980.40	19.44	0.90

Table 12: Mesh 2, order 4

#threads	reaction	diffusion	total	scale factor	scalability
1	48906.63	29253.54	78160.17	1.00	–
2	25643.07	16166.81	41809.88	1.87	0.93
4	13351.43	8678.10	22029.53	3.55	0.95
8	6577.43	4272.48	10849.91	7.20	1.02
12	4505.80	3102.36	7608.13	10.27	0.95
16	3065.58	2194.60	5260.18	14.86	1.08
24	2282.96	1615.26	3898.22	20.05	0.90

Table 13: Mesh 2, order 6

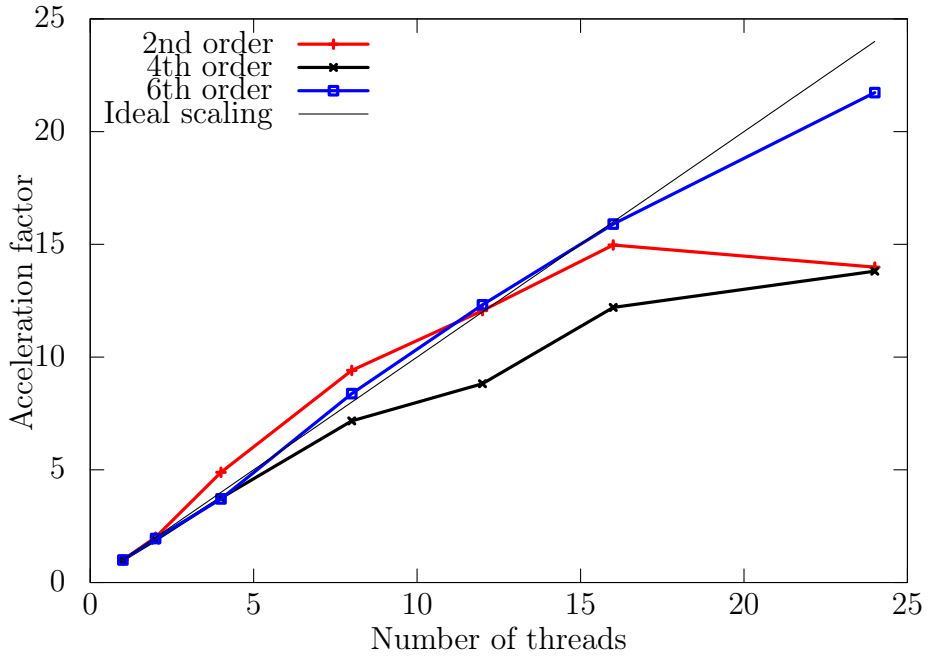


Figure 6: Scalability (AP model) on Mesh 1.

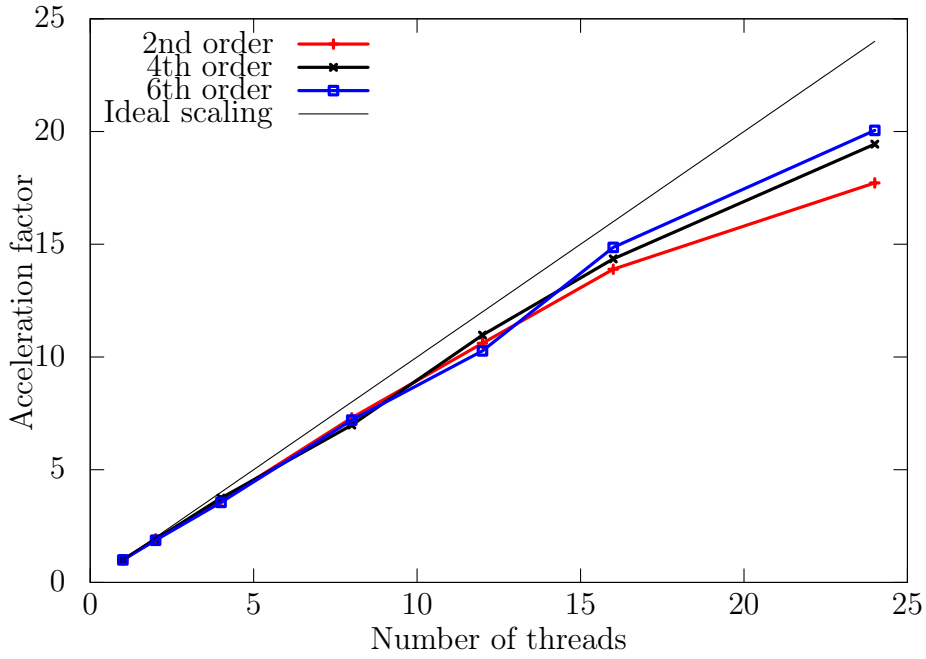


Figure 7: Scalability (AP model) on Mesh 2.

of order 2, 4 and 6 methods on Mesh 1 and order 2 method on mesh 2 are considered. This allows to see at a glance the relative cost of increasing the order of the method or refining the mesh.

On a given mesh, order 6 is roughly 10 times more expensive than order 2, but their CPU times improve at the same speed in parallel. One also sees that using order 6 on Mesh 1 has nearly the same cost as order 2 on Mesh 2. Since, the accuracy of order 6 on Mesh 1 is way better, this shows that using high-order is more efficient independently of the number of threads. In fact, the conclusions of [2] remain valid with our domain decomposition technique.

One can also see that the relative cost of the reaction is initially all the more important that the order is large. However, its scalability behaves way better than the diffusion due to the fact that only local computations are involved. Interestingly, the Aliev-Panfilov ionic model is indeed the worst case since only one simple ODE is involved.

To confirm this asumption, the same computations are done using the Ten Tussher *et al* ionic model which involves 16 ODEs, including stiff ones.

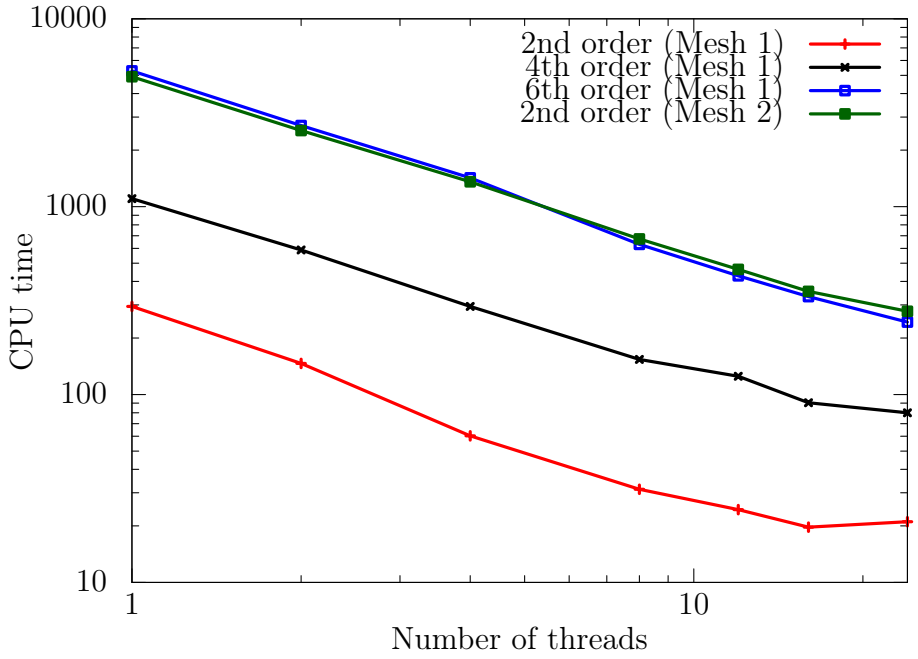


Figure 8: CPU times vs number of threads.

The results are displayed on figure 9. Here, even on the coarse mesh (Mesh 1), our scalability is excellent.

#threads	reaction	diffusion	total	scale factor	scalability
1	34111.27	2210.35	36321.62	1.00	–
2	16508.04	1143.83	17651.86	2.06	1.03
4	8379.74	601.04	8980.78	4.04	1.01
8	4175.42	341.47	4516.88	8.04	0.99
12	2790.62	232.51	3023.13	12.01	1.00
16	2109.46	183.39	2292.85	15.84	0.99
24	1439.09	133.79	1572.88	23.10	0.97

Table 14: TNNP model, Mesh 1, order 6

Finally, a set of computations is performed on a KNL node and the results are shown on table 15 and figure 10. Up to 24 threads, the scalability remains optimal. After that, the mesh is to coarse to observe a significant improvement. From 32 threads onward, there exist interfaces between sub-

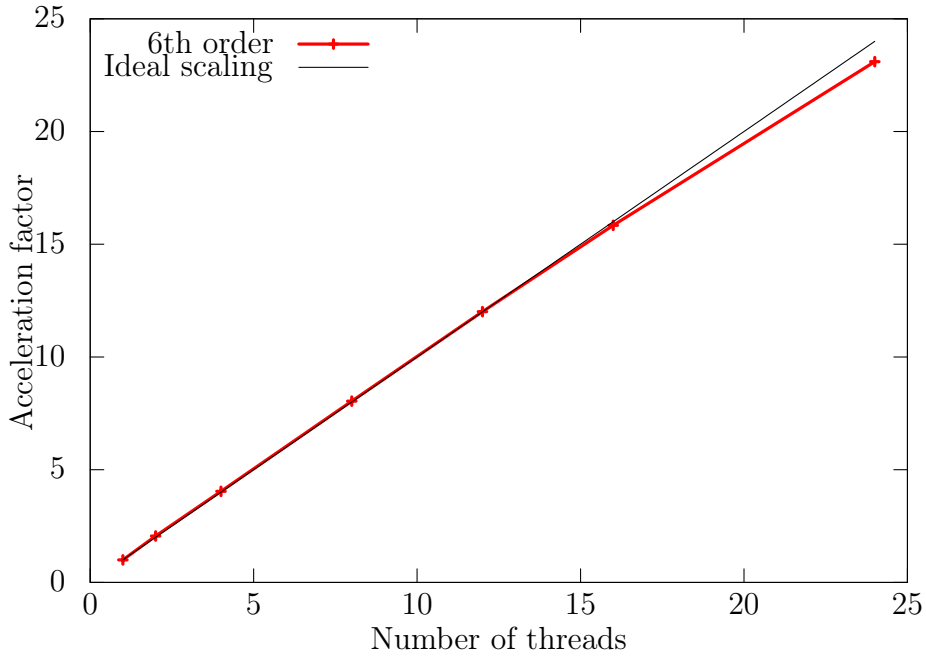


Figure 9: Scalability (TNNP model) on Mesh 1.

domains for which the second layer of neighbors has to be added to the halo. This spreads out the halo, which kills the scalability. Let us note nonetheless that the accuracy remains very good (see for instance the result with 128 subdomains on figure 5).

Only order 4 and Mesh 1 were used in this configuration since we were limited by a low system time-limit which disallow the computation in the case of one domain for either higher order or finer mesh.

5. Conclusion

In this paper, a technique to fit a very high-order MOOD method to parallel computing is proposed. Since MOOD methods rely on reconstruction stencils which may be large, their efficiency in parallel was questionable. Hopefully, numerical results show a very good scalability up to the point where the subdomains are small enough to have a number of cells comparable to their halo. When this happens, the scalablity obviously stalls very quickly. This is not a big issue however since the computational cost is already very low in such cases.

Moreover, the CPU time decreases at a speed which does not depend on the order of the method (as shown on figure 8). Since high-order methods were the most interesting in terms of accuracy vs CPU time ratio even in serial computing, they remain the best choice with our parallel extension.

Even though we only implemented our method with OpenMP routines, its extension to MPI is straightforward since communications are minimal (for instance, equivalent to a Lagrange P1 finite elements scheme). Obviously, the conclusions on the speed-up have to be confirmed.

Finally, let us empahsize that the method applies to any MOOD-type scheme and hence to other kind of equations.

Acknowledgements

This study received financial support from the French Government as part of the "Investments of the Future" program managed by the National Research Agency (ANR), Grant reference ANR-10-IAHU-04.

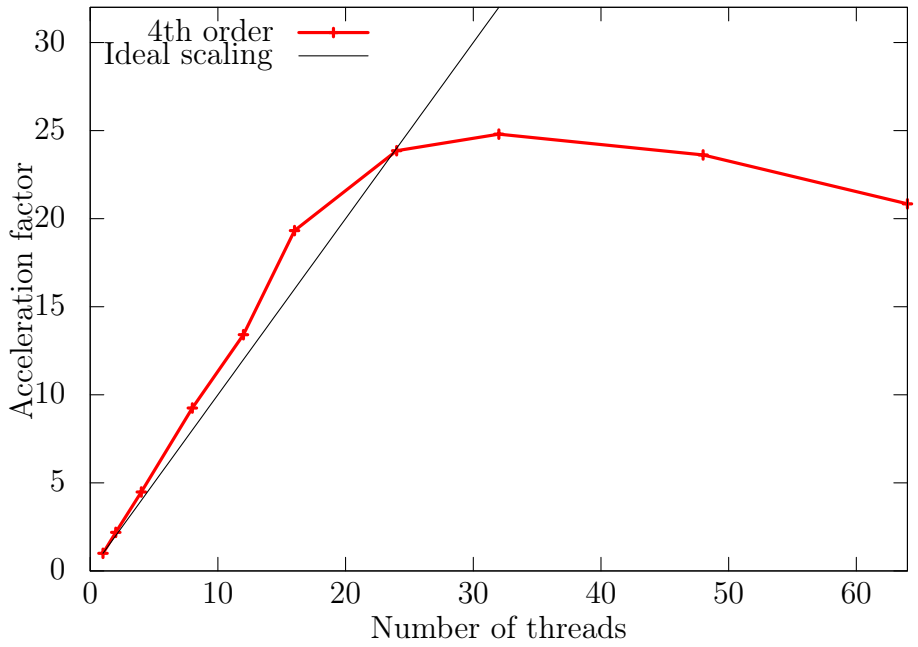


Figure 10: Scalability (AP model) on Mesh 1 - KNL node.

#threads	reaction	diffusion	total	scale factor	scalability
1	4524.86	2651.64	7176.51	1.00	–
2	1900.88	1371.80	3272.67	2.19	1.10
4	921.89	679.90	1601.79	4.48	1.02
8	445.99	329.60	775.59	9.25	1.03
12	306.13	229.05	535.18	13.41	0.97
16	193.86	177.62	371.47	19.32	1.08
24	128.59	172.13	300.73	23.86	0.82
32	100.31	189.09	289.40	24.80	0.78
48	103.35	200.43	303.78	23.62	0.64
64	56.02	288.34	344.36	20.84	0.66
128	45.39	624.82	670.21	10.71	0.26
256	37.85	1503.48	1541.32	4.66	0.22

Table 15: Mesh 1, order 4, KNL Node

References

[1] S. A. Niederer, J. Lumens, N. A. Trayanova, Computational models in cardiology, Nature Reviews Cardiology 16 (2019) 100 – 111.

[2] Y. Coudière, R. Turpault, Very high order finite volume methods for cardiac electrophysiology, Computers & Mathematics with Applications (2017) –.

[3] R. Aliev, A. Panfilov, A simple two-variable model of cardiac excitation, Chaos, Solitons and Fractals 3 (1996) 293–301.

[4] K. H. Ten Tusscher, D. Noble, P. J. Noble, A. V. Panfilov, A model for human ventricular tissue, Am. J. Physiol. Heart. Circ. Physiol. 286 (2004).

[5] E. J. Vigmond, M. Hughes, L. L. G. Plank, Computational tools for modeling electrical activity in cardiac tissue, J. Electrocardiol. 36 (2003) 69–74.

[6] D. Krause, M. Potse, T. Dickopf, R. Krause, A. Auricchio, F. Prinzen, Hybrid Parallelization of a Large-Scale Heart Model, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 120–132.

[7] G. R. Mirams, C. J. Arthurs, M. O. Bernabeu, R. Bordas, J. Cooper, A. Corrias, Y. Davit, S.-J. Dunn, A. G. Fletcher, D. G. Harvey, M. E. Marsh, J. M. Osborne, P. Pathmanathan, J. Pitt-Francis, J. Southern, N. Zemzemi, D. J. Gavaghan, Chaste: An open source c++ library for computational physiology and biology, PLOS Computational Biology 9 (2013) 1–8.

[8] P. Colli Franzone, L. Pavarino, A parallel solver for reaction–diffusion systems in computational electrocardiology, Mathematical Models and Methods in Applied Sciences 14 (2004) 883–911.

[9] R. W. dos Santos, G. Plank, S. Bauer, E. J. Vigmond, Parallel multigrid preconditioner for the cardiac bidomain model, IEEE Transactions on Biomedical Engineering 51 (2004) 1960–1968.

[10] G. Seemann, F. B. Sachse, M. Karl, D. L. Weiss, V. Heuveline, O. Dössel, Framework for Modular, Flexible and Efficient Solving the Cardiac Bidomain Equations Using PETSc, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 363–369.

[11] R. Sachetto Oliveira, B. M. Rocha, R. M. Amorim, F. O. Campos, W. Meira, E. M. Toledo, R. W. dos Santos, Comparing cuda, opencl and opengl implementations of the cardiac monodomain equations, in: R. Wyrzykowski, J. Dongarra, K. Karczewski, J. Waśniewski (Eds.), Parallel Processing and Applied Mathematics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 111–120.

[12] G. Haase, M. Liebmann, C. C. Douglas, G. Plank, A parallel algebraic multigrid solver on graphics processing units, in: W. Zhang, Z. Chen, C. C. Douglas, W. Tong (Eds.), High Performance Computing and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 38–47.

[13] M. Munteanu, L. Pavarino, S. Scacchi, A scalable newton–krylov–schwarz method for the bidomain reaction-diffusion system, SIAM Journal on Scientific Computing 31 (2009) 3861–3883.

[14] S. Clain, S. Diot, R. Loubère, A high-order finite volume method for systems of conservation laws—multi-dimensional optimal order detection (mood), Journal of Computational Physics 230 (2011) 4028 – 4050.

[15] S. Clain, G. Machado, J. Nóbrega, R. Pereira, A sixth-order finite volume method for multidomain convection-diffusion problem with discontinuous coefficients, Computer Methods in Applied Mechanics and Engineering 267 (2013) 43–64.

[16] S. Clain, G. Machado, A very high-order finite volume method for the time-dependent convection–diffusion problem with butcher tableau extension, Computers & Mathematics with Applications 68 (2014) 1292 – 1311.

[17] S. Gottlieb, C.-W. Shu, E. Tadmor, Strong stability preserving high-order time discretization methods, SIAM Rev. 43 (2001) 89–112.

[18] D. A. Dunavant, High degree efficient symmetrical gaussian quadrature rules for the triangle, International Journal for Numerical Methods in Engineering 21 (1985) 1129–1148.

[19] F. Pellegrini, Scotch and PT-Scotch Graph Partitioning Software: An Overview, in: O. S. Uwe Naumann (Ed.), Combinatorial Scientific Computing, Chapman and Hall/CRC, 2012, pp. 373–406. URL: https://hal.inria.fr/hal-00770422.