



# Component-based implicit parallelism model for multi-stencil numerical simulations

Hélène Coullon, Christian Perez

**RESEARCH  
REPORT**

**N° 7003**

April 2015

Project-Teams Avalon





## Component-based implicit parallelism model for multi-stencil numerical simulations

Hélène Coullon<sup>\*†</sup>, Christian Perez<sup>‡</sup>

Project-Teams Avalon

Research Report n° 7003 — April 2015 — 5 pages

**Abstract:** abstract

**Key-words:** No keywords

---

<sup>\*</sup> Footnote for first author

<sup>†</sup> Shared foot note

<sup>‡</sup> Footnote for second author

**RESEARCH CENTRE  
GRENOBLE – RHÔNE-ALPES**

Inovallée  
655 avenue de l'Europe Montbonnot  
38334 Saint Ismier Cedex

**Exemple de document  
utilisant le style  
rapport de recherche  
Inria**

**Résumé :** Resume en francais

**Mots-clés :** mots-cles

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Multi-stencil programs</b>	<b>3</b>
2.1	Definitions . . . . .	3
2.2	Parallelization . . . . .	5
<b>3</b>	<b>Component Stencil Model</b>	<b>5</b>
3.1	A primitive component model . . . . .	5
3.2	Control components . . . . .	5
3.3	Transformation to a parallel assembly . . . . .	5
<b>4</b>	<b>Component Stencil Language</b>	<b>5</b>
4.1	CSL language and example . . . . .	5
4.2	CCSL compiler and example . . . . .	5
4.3	Evaluation . . . . .	5
<b>5</b>	<b>Related work</b>	<b>5</b>
<b>6</b>	<b>Conclusion</b>	<b>5</b>

## 1 Introduction

## 2 Multi-stencil programs

To numerically solve a set of PDEs, iterative methods are frequently used to approximate the solution by a step by step phenomena. Thus, the continuous time and space domains are discretized so that a set of numerical computations are iteratively (time discretization) applied on a mesh (space discretization). In other words, the PDEs are transformed to a set of numerical computations applied at each time step on all elements of the discretized space domain. Among the numerical computations is found a set of numerical schemes, also called *stencil computations*. A formal definition of a *stencil program*, and the parallelization of such program, are presented in this section.

### 2.1 Definitions

A mesh  $\mathcal{M}$  defines the discretization of the continuous space domain  $\Omega$  of a set of PDEs.

**Definition 1** *A mesh is a connected undirected graph without bridges.*

$E_i$  denotes a set of elements constructed by a function  $elem_i$  which defines a precise association of nodes and edges of a mesh. If we denote *nodes* the function to get nodes of a mesh and *edges* the same for edges, the function  $elem_i$  is defined as

$$elem_i : nodes(\mathcal{M}) \times edges(\mathcal{M}) \rightarrow E_i$$

For example, the set of cells  $E_0$  in a Cartesian 2D mesh could be defined by exactly four nodes and four edges connected as a cycle. But we could also define another set of elements  $E_1$  as the simple set of nodes etc.

A mesh can be structured (as Cartesian or curvilinear meshes), unstructured, regular or irregular (without the same topology for each element) and hybrid if mixing those.

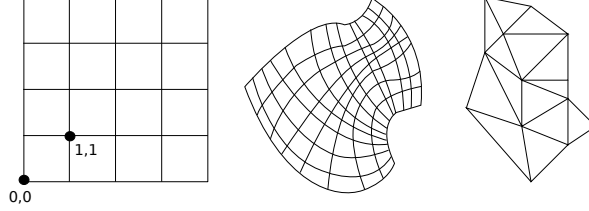


Figure 1: From left to right, Cartesian, curvilinear and unstructured meshes.

The discretization of the continuous time domain  $\mathcal{T}$  is denoted  $T$ .  $T$  is responsible for the iteration time steps of the numerical simulation.

In a numerical simulation a set of data, or quantities, are applied onto the mesh and represent the set of values to compute, or to use, for computation. We denote the set of data applied on the mesh by  $\Delta$ , such that  $\delta \in \Delta$  is a function which associates each element  $el \in E_i$  to a value  $v \in V$ ,

$$\delta : E_i \rightarrow V$$

One can notice that in applied mathematics, the signature of  $\delta$  would be  $\delta : E_i \times T \rightarrow V$ , however when programming a numerical simulation it is not wise to store all values of each time iteration.

Finally, in a numerical simulation are performed a set of ordered numerical computations denoted  $\Gamma$ . A numerical computation in  $\Gamma$  is denoted

$$c(R, w, D, e).$$

In this definition  $R \subset \Delta, w \in \Delta$  are respectively the data read and written to compute the numerical expression  $e$  (a single data is written in a single computation).  $D$  is one of the subsets  $E_i \subset \mathcal{M}$ . It has to be noticed that at each time iteration, all the elements of a mesh are computed. However, it happens that the computation of the mesh elements is splitted in different computations (for example the computation of the physical border). In this case additional  $E_i$  can be specified for the mesh  $\mathcal{M}$ . The numerical expression  $e$  computes the data  $w$  from a set of data  $R$ .

$$e : R \rightarrow w$$

A *multi-stencil program* is defined by the quadruplet

$$\mathcal{MSP}(T, \mathcal{M}, \Delta, \Gamma), \tag{1}$$

If the number of computations in  $\Gamma$  is  $\text{card}(\Gamma) = m$ , such that  $\bigcup_{i=0}^{m-1} c_i = \Gamma$ , then  $\bigcup_{i=0}^{m-1} R_i \cup w_i \subseteq \Delta$ .

A computation  $c \in \Gamma$  can be of two different types. The first type is called a *stencil computation* and involves an additional information denoted  $\mathcal{N}$  which represents a neighborhood function around an element

$$\mathcal{N} : E_i \rightarrow E_k \times E_k \times \dots \tag{2}$$

, where  $E_k$  is a set of elements of  $\mathcal{M}$  which could be equal to  $E_i$  or not. In other words, from an element  $el \in E_i$  the function  $\mathcal{N}$  returns a set of elements of  $E_k$  which represents the

neighborhood of  $el$ . The function  $\mathcal{N}$  is also sometimes called the *stencil shape*, or the *stencil* in applied mathematics.

A *stencil computation* is a quintuplet

$$s(R, w, D, e, \mathcal{N}), \quad (3)$$

where  $R \subset \Delta, w \in \Delta$  are read and written in the numerical expression  $e$ , and  $D$  is one the subsets  $E_i$ . In a stencil computation  $s$ ,  $\forall d \in D$ , the stencil numerical expression  $e$  is applied such that  $w(d) = e(R(d), R(\mathcal{N}(d)))$ . In this work, a stencil computation  $s(R, w, D, e, \mathcal{N})$  always verifies  $R \cap w = \emptyset$ , otherwise an implicit numerical scheme has to be solve which is over the scope of this paper.

Figure ?? gives an example of a stencil computation  $s(R, w, D, e, \mathcal{N})$ , on a two dimensional Cartesian mesh, where  $R = \{A, B\}$ ,  $w = C$ , and where for  $(x, y) \in D$

$$e(R(x, y), R(\mathcal{N}(x, y))) = A(x, y) + B(x, y + 1) + B(x, y - 1) + B(x + 1, y) + B(x - 1, y).$$

Finally, the second type of numerical computation is a local computation where  $e$  does not involve a neighborhood function  $\mathcal{N}$

$$l(R, w, D, e) \quad (4)$$

## 2.2 Parallelization

# 3 Component Stencil Model

## 3.1 A primitive component model

Definitions of a primitive component (ports, interfaces), a component instantiation, an assembly + simple semantics on the behavior of a primitive component. Cite L2C as an implementation of such a model + introduction of notations (use,provide,use-multiple,sync).

## 3.2 Control components

Definition of three control components, and of the kernel component. Explain the behavior.

## 3.3 Transformation to a parallel assembly

Transformation from a parallel algorithm using the directives: PARALLEL, PARALLEL SECTIONS, SECTION and FORALL, to a component assembly

# 4 Component Stencil Language

## 4.1 CSL language and example

## 4.2 CCSL compiler and example

## 4.3 Evaluation

# 5 Related work

# 6 Conclusion



**RESEARCH CENTRE  
GRENOBLE – RHÔNE-ALPES**

Inovallée  
655 avenue de l'Europe Montbonnot  
38334 Saint Ismier Cedex

Publisher  
Inria  
Domaine de Voluceau - Rocquencourt  
BP 105 - 78153 Le Chesnay Cedex  
[inria.fr](http://inria.fr)

ISSN 0249-6399