Analysis and optimization of the diffusive representation method for distributed control applications with distributed computers

Huu Quan Do

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UFC UNIVERSITÉ DE FRANCHE-COMTÉ

Analyse et optimisation de la méthode de représentation diffusive en vue d'applications au contrôle distribué avec calculateurs distribués

HUU QUAN DO

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UFC UNIVERSITÉ DE FRANCHE-COMTÉ

N° 2016001

THÈSE présentée par

HUU QUAN DO

pour obtenir le

Grade de Docteur de
l'Université de Franche-Comté

Spécialité : **Informatique**

Analyse et optimisation de la méthode de représentation diffusive en vue d'applications au contrôle distribué avec calculateurs distribués

Unité de Recherche :
FEMTO-ST, Département Informatique de Belfort, Université de Franche-Comté

Soutenue publiquement le 16 February 2016 devant le Jury composé de :

FRÉDÉRIC MAGOULÈS	Rapporteur	Professeur de Ecole Centrale Paris
PIERRE SPITERI	Rapporteur	Professeur Emérite des Universités, Université de Toulouse
RAPHAËL COUTURIER	Directeur de Thèse	Professeur des Universités, Université de Franche-Comté
MICHEL LENCZNER	Co-directeur de Thèse	Professeur des Universités, Université de Technologie Belfort-Montbéliard

REMERCIEMENTS

I would like to express the deepest appreciation to my advisors, Professor Raphaël COUTURIER and Professor Michel LENCZNER, for their support, guidance, and supervision. Their knowledge, dedication, enthusiasm, patience, encouragement, and personality have made my graduate experience at University of Franche-Comté rewarding and indeed unforgettable. . I would like to acknowledge the financial support by Labex ACTION [1]. Thanks to University of Franche-Comté for giving me opportunity to work on my dissertation.

I would like to thank my colleagues for providing me a lot of helps in my years at Institute of FEMTO-ST, Informatic department at Belfort, and Time-frequency department at Besançon. I would like to extend my special thanks to my colleagues : Ali Kadhum IDREES, Jean-Claude CHARR, Jean-Francois COUCHOT, Ahmed FANFAKH, Yousra Ahmed FADIL, Stephane DOMAS, Arnaud GIERSCH, Christophe GUYEUX, David LAIYMANI, Gilles PERROT for their support. I would like to thank also Duy Duc NGUYEN, Thi Trang NGUYEN, for their helpful discussions with mathematical problems, simulation and control issues that I encountered in my research work. I would also like to express my thanks to Dr. Lilia Ziane KHODJA, PostDoc at LT AS-A&M, Liege, Belgium. I would like to express my thanks and my best wishes to Ingrid COUTURIER for all the received assistance during my study.

Last but the least, I want to give my deepest grateful to my parents for supporting and encouraging me to get through all these years, and for their endless love. Thanks to my brothers and my sisters for supporting and being proud of me. I would also like to thank my wife PhD. Vy Thuy Lynh HOANG, and her family for their supporting me to finish my dissertation over last two years. I am grateful to my wife for sharing her time with me in preparing and writing this dissertation. Again, I am thankful to everyone who have contributed to this dissertation.

1. Contract ANR-11-LABX-0001-01, www.labexaction.fr.

Labex Action
Integrated smart systems

SOMMAIRE

INTRODUCTION

MOTIVATION OF THE DISSERTATION

In this dissertation, we are interested in control problems, in view of distributed control on distributed computing architectures. Namely, we consider a class of systems governed by linear partial differential equations in which the observations and controls are distributed in large numbers. Achieving global control of such systems remains a challenging task.

The applications mainly concern matrices of micro-systems and smart distributed systems. In such applications, the realization of optimal controllers seems to be impossible if the calculation is performed on centralized architectures computers. This is due to the limitations firstly in transmitting information and secondly computing power. To deal with this difficulty, we consider the use of semi-decentralized architectures, that is, elementary matrices interconnected only between neighbouring computers as illustrated in Fig. 1.

The design of semi-decentralized controllers (that is, controllers are suitable for implantation on semi-decentralized architectures) has attracted the attention of many researchers. This interest is then reinforced since the technological progress in the field of matrices MEMS (i.e., MicroElectroMechanical Systems). Indeed, the manufacture of large arrays of micro-mechanical devices with distributed sensors and actuators has become both feasible and promising from an economic point of view, as for example, the micro-cantilever arrays and micro-mirrors.

The problem encountered is that the optimal control operators are by nature very non-local and therefore not suitable for implantation in semi-decentralized architectures. For

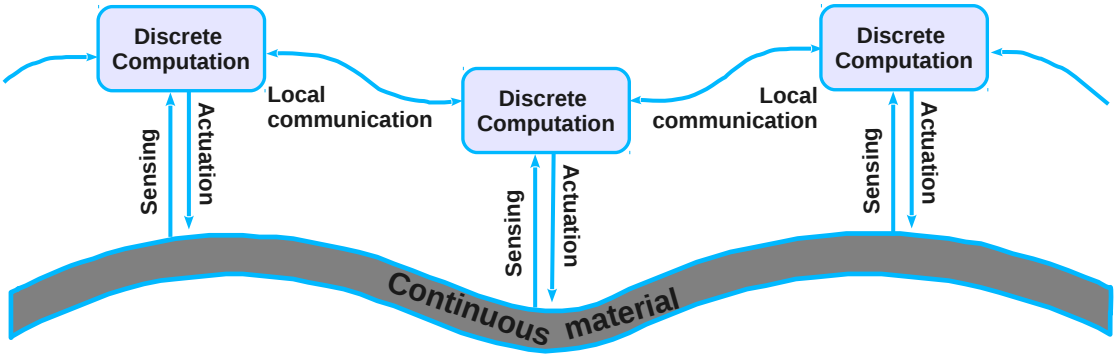


FIGURE 1 – Continuous system properties can be sensed, processed in a computing element, and actuated. Sensors, actuators, and computing elements are at discrete locations and can communicate locally.

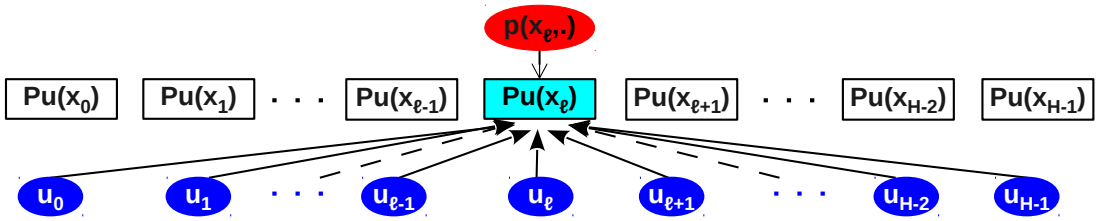


FIGURE 2 – An illustration of a computation in distributed architecture with the global inputs requirement.

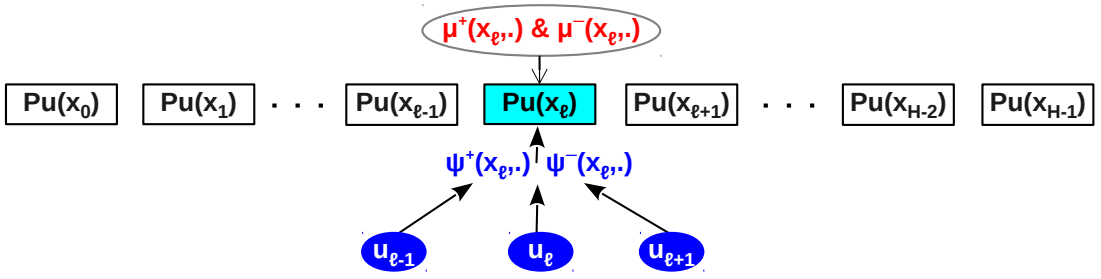


FIGURE 3 – An illustration of a computation in distributed architecture with the local inputs requirement.

example, we consider a kernel operator P in the form

$$Pu(x) = \int_0^1 p(x,y)u(y)dy, \tag{1}$$

where p is the kernel of P and u is an input function. In view of distributed control, the computing unit receives discrete values of the input u from sensors, and Pu is the control function whose values are used to send to actuators. In view of distributed architecture, this integral can be approximated by a quadrature rule

$$Pu(x_\ell) \approx h \sum_{k=0}^{H-1} p(x_\ell, y_k)u(y_k), \ell = 0, 1, ..., H-1. \tag{2}$$

To compute the $Pu(x_\ell)$, global inputs are required. Any node involved in the computation needs all inputs $u_k = u(y_k)$ as illustrated in Fig. 2. Our method provides an approximation such that the same computation only requires local inputs. In practice, our computation only needs the information from few neighbours as illustrated in Fig. 3.

The work presented in this thesis focuses on that method in the context of approximation of linear operators, solutions of linear partial differential operatorial equations. Such approximations allow us to implement real time applications on semi-decentralized network architectures. Our method is based on the diffusive representation [26]. We propose two approaches to approximate those equations in one-dimensional domains. We emphasize that our goal is not to solve control problems. We aim at building a basic tool that will help researchers to make control problems with distributed architectures. Namely, we continue the development of the diffusive realization (DR) theory [15], [36], then we study its implementation in distributed computing environment. For the latter, we

study some parallel topologies and study the implementation of the DR method on these topologies.

One of the main recognized advantages of DR of a linear operator is its low computational cost, see for example the papers of G. Montseny et al. [2], [3], [13], [17], [25], and by D. Matignon et al. [9], [10], [22] for representations of various pseudo-differential operators, and for their approximation. C. Lubich et al. [6], [21] and López-Fernández et al. [18], [19], [20], [29], apply a similar idea to convolution operators and develop optimized numerical methods. They have reported detailed performance analysis of their methods in comparison with a direct approach via quadratures.

MAIN CONTRIBUTIONS OF THIS DISSERTATION

This thesis includes three main contributions. In **Chapter 1**, we complete a theoretical framework of diffusive realizations for state-realizations of some linear operators developed in Yakoubi [36] and Lenczner et al. [15]. The realization of a linear operator $u \to z = Pu$, by the DR method, was addressed to causal operators when the kernel p of P is explicitly known, and analytic in its second variable, see Montseny [26]. The case where P is the solution of an operator equation, so p is neither explicitly given nor analytic, was reported in Yakoubi et al. [37]. Their method was announced in Lenczner and Montseny [26] and fully developed in Lenczner et al. [16]. The numerical method was implemented for an operator P solution of Lyapunov equation issued from optimal control of the heat equation. A slightly simplified version reads as

$$-\frac{d^2}{dx^2}Pu - P\frac{d^2}{dx^2}u = Qu \text{ for all } u \in H_0^1(0,1), \tag{3}$$

Q being a given operator. An overview of the theory and of the numerical results was presented in the PhD thesis of Yakoubi [36]. We notice that their results were suffering from a poor accuracy. Precisely, the DR theory requires an analytic extension of the kernel p which was obtained after approximation by Legendre polynomials and extension out of the interval $\widehat{\omega} = (-1,1)$. This leads to non-uniformly bounded extensions with respect to the number of polynomials which causes high numerical errors. In this thesis, we introduce an additional change of variables to the kernel p and its extension so that the latter is defined in $\widehat{\omega}$ which eliminates an important source of error.

In **Chapter 2**, we implement our method in view of parallel computation, which is a good choice for real-time, embedded, massive, and low-cost computation. Communication speed is highly dependent on the network topology [24] that affects the computational properties of the system. Moreover, a main part of our algorithm can be reformulated into a prefix sum which is very suitable for parallel computation [14]. A study of our algorithm in a parallel implementation is thus presented for three parallel topologies introduced in [4], [8], [27], [28] : line topology, hypercube topology and binary topology. In order to implement the DR method close to the system to control, we use microcontrolers whose advantages are small, cheap and which have real-time computing capacity [12], [23], [30]. In this implementation, we estimate the number of operations and transmissions, namely, the number of additions and multiplications and the number of real values sent during communication process. Three topologies with the DR and the line topology for the direct

method have been considered. Based on that, we also estimate their computation time and make comparison of the computation time. These results show us the effeciency of our method. Therefore, we can choose a suitable network for designing realistic applications which reach high-effect economic.

As in [35], but with a significantly different theoretical approach, in **Chapter 3** we also provide, for the first time, an error estimate of the DR and use it in a contour optimization method. In this related field of numerical inversion of the Laplace transform of a function, Stenger [31], [32], [33] provided a discretization error based on the trapezoidal formula. His important results was applied in many works. Above all, Weideman and Trefethen [35] provided an error estimate and its application to contour optimization for computation cost reduction. It is based on the balancing of discretization and truncation errors in which they assumes that the function decays rapidly. López-Fernández et al. [20] showed that a faster decay of the function provides a better error estimate. Optimal parameters for three classes of contours (parabolas, hyperbolas, and cotangent contours) in the framework of numerical inversion of the Laplace transform were studied by various authors. In this thesis, we only consider hyperbolic contours as the best ones. The method has been extensively tested and significant results are reported.

DISSERTATION OUTLINE

This dissertation is organized as follows. In Chapter 1, we introduce the DR theory. We also formulate the extension of the kernel. Then we build the approximation of the diffusive realization. We illustrate our method on a Lyapunov equation arising from the optimal control theory of the heat equation. Chapter 2 implements this theory on different parallel computer topologies. Results are presented for three parallel topologies : line topology, hypercube topology, binary topology. We also compare the computation times between our algorithm with these topologies and a direct spectral method with a line topology. With a significantly different theoretical approach, we provide an error estimate of the DR and use it in a contour optimization method in Chapter 3. There are three sources of errors in the algorithm. They come from the discretization of the contour integral, the piecewise constant interpolation of the input u and the application of the Galerkin method in the kernel computation. The latter is not taken into account in this dissertation. Moreover, the method has been extensively tested and some significant results are reported. We draw our conclusion in Chapter 4 with some remarks on future research work.

1

DIFFUSIVE REALIZATION : THEORY AND NUMERICAL APPROXIMATION

This chapter is devoted to the diffusive realization theory introduced in Subsection 1.1.1, to the formulation of the extension of the kernel in Subsection 1.1.2, and to the approximation of the diffusive realization in Subsection 1.1.3. The diffusive realization of an operator P is based on a decomposition into causal and anti-causal parts. Sufficient conditions of existence of the diffusive realization on the contour $\theta^{\pm}$ are stated in Theorem 1. A hyperbolic contour lying the singularities in a sectorial region around the negative real axis is used. We also use a change of variables to transform the kernel. Then, its transformation is approximated based on a Legendre basis. The inversion of the approximation extension is now an approximation extension of the kernel. Finally, the approximations of the diffusive symbol, of history functions and the computation algorithm for the diffusive realization are derived. A complete example illustrates this theory in Section 1.2.

1.1/ DIFFUSIVE REALIZATION OF INTEGRAL OPERATORS

1.1.1/ DIFFUSIVE REALIZATION THEORY

We consider operators P bounded in $L^2(\omega)$ formulated in the general integral form

$$Pu(x) = \int_{\omega} p(x,y)u(y)\,dy,$$

where $\omega = (0,1)$, and where the regularity of the kernel $p(x,y)$ will be specified later.
An operator P is said to be causal (respectively anti-causal) if $p(x,y) = 0$ for $y \geq x$ (respectively for $y \leq x$). Diffusive realizations of P are based on its decomposition into causal and anti-causal parts : $P = P^+ + P^-$ where $P^+u(x) = \int_0^x p(x,y)u(y)\,dy$ and $P^-u(x) = \int_x^1 p(x,y)u(y)\,dy$. Throughout this work, we shall use the superscripts $+$ or $-$ to refer to causal or anti-causal operators, and the convention $\mp = -(\pm)$. The so-called impulse response $\widetilde{p}$ is defined by $\widetilde{p}(x,y) = p(x,x-y)$. We introduce the modified impulse responses $\widetilde{p}^{\pm}(x,y) = \widetilde{p}(x,\pm y)$. The variables x and y are treated on an unequal footing, assuming that the causal (resp. anti-causal) impulse response is analytic with respect to y, with analytic extension to $\mathbb{R}_y^{+*}$ (resp. $\mathbb{R}_y^{-*}$) locally integrable, and that for each y, the function $x \longmapsto p(x,y)$ belongs to $L^2(\omega)$.
For given $a^{\pm} \in \mathbb{R}$, we consider $\xi \longmapsto \theta^{\pm}(\xi)$ two complex Lipschitz functions from $\mathbb{R}$ to

$[a^{\pm}, +\infty) + i\mathbb{R} \subset \mathbb{C}$ such that $|\theta^{\pm\prime}| \geq b > 0$ almost everywhere which define simple arcs closed at infinity. Moreover we assume that they are included in some sector $k + e^{i[-\alpha,+\alpha]}\mathbb{R}^+$ with $0 \leq \alpha < \frac{\pi}{2}$.
From now on, we use the convenient notation, $\langle \mu, \psi \rangle := \int_{\mathbb{R}} \mu(\xi)\psi(\xi)d\xi$ and we remark that in the case where μ would not be a locally integrable function, a more general duality product, to be specified in each concrete case, would be involved in place of the integral. Consider also the so-called $\theta^{\pm}-$representations of u, denoted by $\psi^{\pm}(u)$ and defined as the unique solutions of the following direct and backward Cauchy problems (parameterized by $\xi \in \mathbb{R}$), of diffusive type thanks to the sector condition on $\theta^{\pm}$:

$$\begin{aligned} &\partial_x \psi^+(x,\xi) = -\theta^+(\xi)\psi^+(x,\xi) + u(x), \forall x \in (0,1), \psi^+(0,\xi) = 0 \\ &\text{and } \partial_x \psi^-(x,\xi) = \theta^-(\xi)\psi^-(x,\xi) + u(x), \forall x \in (0,1), \psi^-(1,\xi) = 0. \end{aligned} \tag{1.1}$$

We shall say that a causal operator P^+ (resp. anti-causal operator P^-) admits a θ^+-diffusive realization (resp. θ^- -diffusive realization) if there exists a so-called diffusive symbol $\mu^+(x,\xi)$ (resp. $\mu^-(x,\xi)$) so that $P^+u(x) = \langle \mu^+, \psi^+(u) \rangle$ (resp. $P^-u(x) = \langle \mu^-, \psi^-(u) \rangle$). Similarly, we say that an operator P admits a $\theta^{\pm}$-diffusive realization if both its causal and anti-causal parts P^+ and P^- admit a diffusive realization associated respectively to θ^+ and θ^-.
Let us state some sufficient conditions for the existence of the so-called canonical diffusive realization of an operator P for general paths $\theta^{\pm}$. They pertain to the Laplace transforms with respect to y of the causal and anti-causal parts of the impulse response : $\mathcal{P}^+(x,y) := \mathcal{L}_y(\widetilde{p}(x,y))$ and $\mathcal{P}^-(x,y) := \mathcal{L}_y(\widetilde{p}(x,-y))$. Therefore, the function $y \mapsto \widetilde{p}^{\pm}(x,y)$ should be extended to $\mathbb{R}^+$. Their extensions are considered in the next section.

Theorem 1 : Adapted from [16]

For a given path θ^+ (resp. θ^-), a causal (resp. anti-causal) operator P^+ (resp. P^-) admits a diffusive symbol if the two following conditions are fulfilled :
(i) the Laplace symbol $\lambda \mapsto \mathcal{P}^+(x,\lambda)($ resp. $\lambda \mapsto \mathcal{P}^-(x,\lambda))$ is holomorphic in a domain D^+ (resp. D^-) that contains the closed set located at right of the arc $-\theta^+$ (resp. $-\theta^-$) ;
(ii) $\mathcal{P}^{\pm}(x,\lambda)$ vanish when $|\lambda| \to \infty$ uniformly with respect to $\arg \lambda$.
Then the so-called canonical $\theta^{\pm}$-symbols are given by

$$\mu^+(x,\xi) = -\frac{\theta^{+\prime}(\xi)}{2i\pi}\mathcal{P}^+(x,-\theta^+(\xi)) \text{ and } \mu^-(x,\xi) = \frac{\theta^{-\prime}(\xi)}{2i\pi}\mathcal{P}^-(x,-\theta^-(\xi)) \tag{1.2}$$

and have the same regularity as $\theta^{\pm}$.

In this thesis, we only use the hyperbolic contours $-\theta^{\pm}$ proposed by Weideman and Trefethen in [35], in the context of inverse Laplace transform computation, namely

$$-\theta^{\pm}(\xi) = \theta_h^{\pm}(1 + \sin(i\xi - \alpha^{\pm})) \text{ for } \xi \in \mathbb{R}, \tag{1.3}$$

where $\theta_h^{\pm} > 0$ regulates the width of the contours and $\alpha^{\pm}$ is the hyperbola's asymptotic angle. This contour allows for singularities with an unbounded imaginary part, provided that they lay in a sectorial region around the negative real axis ; see Figure 1.1.

1.1.2/ FORMULATION OF THE EXTENSION

As mentioned before, to define the Laplace transform the function $y \mapsto p(x, x \mp y)$ should be extended to $\mathbb{R}^+$. In other words, $y \mapsto p(x,y)$ has an analytic extension to $(-\infty, 1)$ in the

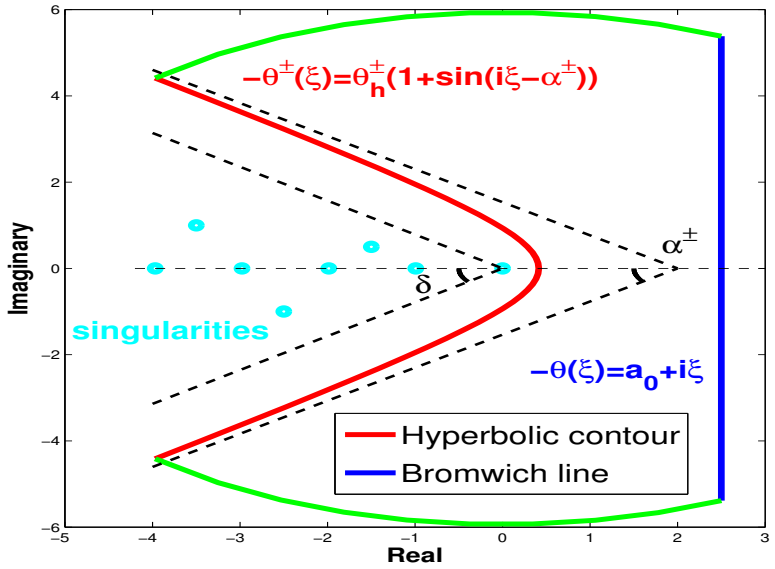


FIGURE 1.1 – The red color refers to a hyperbolic contour $-\theta^{\pm}$ for $\theta_h^{\pm} = 2, \delta = \frac{\pi}{4}, \alpha^{\pm} = \frac{\pi}{6}$. The blue color refers to the Bromwich line. The singularities (cyan color) lay in a sectorial region Σ_δ defined by $\{z \in \mathbb{C} : |arg(z)| \leq \delta\}$. The green color refers to two parts of a circle $C(O, R)$, with $R \to +\infty$, which connect two vertices of the contour and the Bromwich line.

causal case and to $(0, +\infty)$ in the anti-causal case. Indeed, the function $y \mapsto p(x, x \mp y)$ defined over $\mathbb{R}^+$ implies that $-\infty < x - y < x$ for the causal part and $x < x + y < +\infty$ for the anti-causal part. Since $x \in (0, 1)$, $-\infty < x - y < 1$ for the causal part and $0 < x + y < +\infty$ for the anti-causal part.
The aforesaid change of variables is defined in the following subsection.

1.1.2.1/ CHANGE OF VARIABLES

Since an analytic extension cannot be built numerically, we build an extension based on a change of variable mapping Ω to $\widehat{\Omega}^{*\pm}$ and an approximation by Legendre polynomials, where $\Omega = (0, 1)^2$, $\widehat{\Omega}^{*\pm} = (-1, 1) \times \widehat{\omega}^{*\pm}$ and $\widehat{\omega}^{*\pm} \subset (-1, 1)$. Precisely, let us define the change of variable $T^{\pm} : (x, y) \mapsto (s, z)$ to transform the domain Ω of $p(x, y)$ into a subset $\widehat{\Omega}^{*\pm}$. The method amounts to building approximations $z \mapsto \widehat{p}^{N\pm}(s, z)$ of the mapping $z \mapsto \widehat{p}^{\pm}(s, z) = p \circ (T^{\pm})^{-1}(s, z)$ in an N dimensional Legendre basis in z and the basis of modified Legendre polynomials in s. The calculations have been carried out in the set $\Omega^{*\pm}$. Then the approximation of $\widehat{p}^{\pm}$ is naturally extended to the entire set $(-1, 1)^2$. We observe that $T^{\pm}$ transforms $\Omega_\infty^{\pm}$ into $(-1, 1)^2$ with $\Omega_\infty^{+} = (0, 1) \times (-\infty, 1)$ and $\Omega_\infty^{-} = (0, 1) \times (0, +\infty)$, thus the inverse mapping of the extension approximation of $\widehat{p}^{\pm}$ is an extension approximation of p over $\Omega_\infty^{\pm}$.

Remark 1 :

We have used a spectral method to discretize both $s-$ and $z-$directions. In the $z-$direction we actually need to use global basis functions so that they can be analytically extended. On the contrary, there is no particular restriction regarding approximations in the $s-$direction. For instance a local basis as a finite element basis might be used.

The domain ω is extended by its left- and right-extensions on $\omega_\infty^+ = (-\infty, 1)$ and $\omega_\infty^- = (0, +\infty)$. The kernel p is defined on Ω extended by its lower- and upper-extensions $\Omega_\infty^\pm = \omega \times \omega_\infty^\pm$. The mapping $x \mapsto T_x^\pm(x) = 2x - 1$ changes ω into its transformed set $\widehat{\omega} = (-1, 1)$. The mapping $y \mapsto T_y^+(y) = 2e^{\sigma(y-1)} - 1$ changes ω_∞^+ into $\widehat{\omega}$ and changes ω into $\widehat{\omega}^{*+} = (-z^*, 1)$ with $z^* = 1 - 2e^{-\sigma}, \sigma > 0$. Moreover, the mapping $y \mapsto T_y^-(y) = -2e^{-\sigma y} + 1$ changes ω_∞^- into $\widehat{\omega}$ and changes ω into $\widehat{\omega}^{*-} = (-1, z^*)$. We define $\widehat{\Omega} = \widehat{\omega}^2$. Therefore, the change of variables $(s, z) = T^\pm(x, y)$ maps the unbounded sets $\Omega_\infty^\pm$ into the reference set $\widehat{\Omega}$ and the kernel domain Ω into $\widehat{\Omega}^{*\pm} \subset \widehat{\Omega}$,

$$T^\pm(x, y) = (T_x^\pm(x), T_y^\pm(y)). \tag{1.4}$$

Figure 1.2 shows the domain Ω_∞^+ and its image through T^+. Similarly, Figure 1.3 shows the domain Ω_∞^- and its image through T^-. In particular,

$$\begin{gathered} T_x^\pm(0) = -1,\ T_x^\pm(1) = 1, \\ T_y^+(-\infty) = -1,\ T_y^+(0) = -z^*,\ T_y^+(1) = 1, \\ T_y^-(-1) = -1,\ T_y^-(1) = z^*,\ T_y^-(\infty) = 1. \end{gathered}$$

The inverse of $T^\pm$ is

$$(T^\pm)^{-1}(s, z) = \left(\frac{s+1}{2}, (T_y^\pm)^{-1}(z) \right),$$

with

$$(T_y^+)^{-1}(z) = 1 + \frac{1}{\sigma} \log(\frac{z+1}{2}) \text{and } (T_y^-)^{-1}(z) = -\frac{1}{\sigma} \log(\frac{1-z}{2}).$$

So

$$D(T^\pm)^{-1} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{\chi^\pm(z)} \end{pmatrix} \text{ and } \det(D(T^\pm)^{-1}) = \frac{1}{2\chi^\pm(z)},$$

with $\chi^\pm(z) = \sigma(1 \pm z)$. Moreover, since

$$DT^\pm = \begin{pmatrix} \partial_x T_x^\pm & \partial_y T_x^\pm \\ \partial_x T_y^\pm & \partial_y T_y^\pm \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & \partial_y T_y^\pm \end{pmatrix}, \tag{1.5}$$

with

$$\partial_y T_y^+(y) = 2\sigma e^{\sigma(y-1)} \text{ and } \partial_y T_y^+(y) = 2\sigma e^{-\sigma y},$$

so

$$DT^\pm \circ (T^\pm)^{-1}(s, z) = \begin{pmatrix} 2 & 0 \\ 0 & \sigma(1 \pm z) \end{pmatrix}.$$

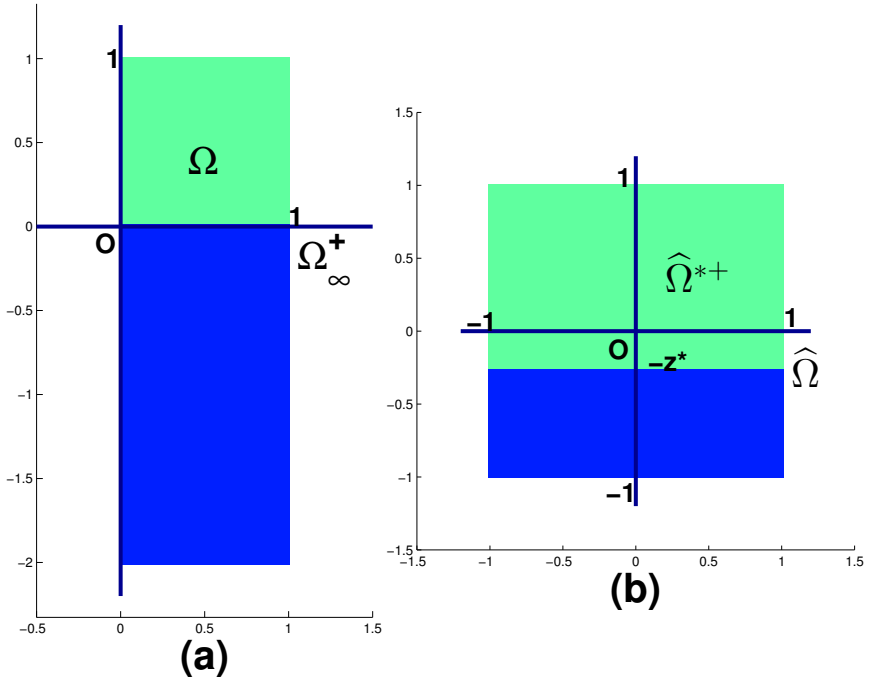


FIGURE 1.2 – (a) The domain Ω_∞^+ and (b) its image $\widehat{\Omega} = T^+(\Omega_\infty^+)$. The cyan color refers to the original domains, and the blue color refers to the extended parts.

1.1.2.2/ APPROXIMATION OF THE KERNEL AND ITS EXTENSION

The extension approximation of the kernel p states as

$$p^{N\pm}(x, x \mp y) = \sum_{k=2}^{N_1} \sum_{\ell=0}^{N_2} p_{k\ell}^{\pm} \eta_k(2x-1) L_\ell(\rho^\pm(x) e^{-\sigma y} + \tau^\pm). \tag{1.6}$$

where $\rho^+(x) = 2e^{\sigma(x-1)}$, $\rho^-(x) = -2e^{-\sigma x}$ and $\tau^\pm = \mp 1$. Moreover, the coefficients $p_{k\ell}^\pm$ are the coefficients of the spectral decomposition of p based on the Legendre basis $L_\ell(z)$ and a modal basis $\eta_k(s)$.

First of all we discuss an approximation of the kernel with respect to y and its extension. Since the transformed kernel $z \mapsto \widehat{p}^\pm(x,z) = p(x, (T_y^\pm)^{-1}(z))$ is defined over $\widehat{\omega}^{*\pm} \subset (-1,1)$, it can be decomposed with a spectral method. Using the Legendre basis $L_\ell(z), \ell = 0, ..., N$, made of the first $N+1$ Legendre polynomials detailed in Section 2.3 of [5] as a spectral basis, its decomposition is of the form

$$\widehat{p}^\pm(x,z) = \sum_{\ell=0}^{\infty} \widehat{p}_\ell^\pm(x) L_\ell(z) \text{ for } z \in \widehat{\omega}^{*\pm}, \tag{1.7}$$

where $\widehat{p}_\ell^\pm$ are the coefficients of the projection of $\widehat{p}^\pm(x,z)$ in $L^2(\widehat{\omega}^{*\pm})$, i.e., they are solution of

$$\sum_{\ell=0}^{+\infty} a_{k\ell} \widehat{p}_\ell^\pm(x) = b_k, \text{ for all } k \in \mathbb{N}. \tag{1.8}$$

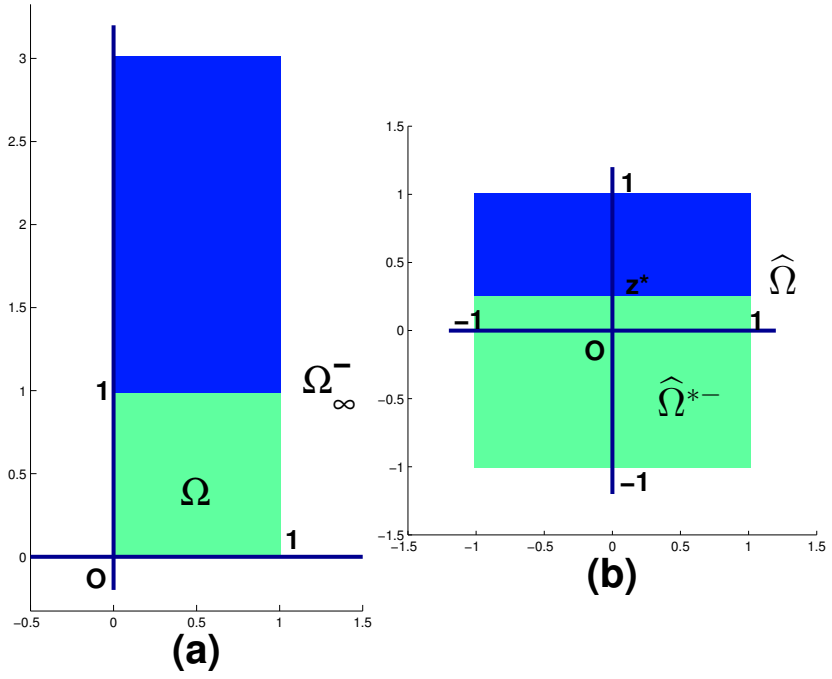


FIGURE 1.3 – (a) The domain Ω_∞^- and (b) its image $\widehat{\Omega} = T^-(\Omega_\infty^-)$. The cyan color refers to the original domains, and the blue color refers to the extended parts.

with $a_{k\ell} = \int_{\widehat{\omega}^{*\pm}} L_k(z)L_\ell(z)dz$ and $b_k = \int_{\widehat{\omega}^{*\pm}} \widehat{p}^{\pm}(x,z)L_k(z)dz$.

We can compute the exact value of a_{kl} and the approximation value of b_k by using a change of variable to change the integration domain into $(-1,1)$ and applying Lobatto quadrature (see Legendre Gauss-Lobato nodes as (2.3.12) in [5]). Equation (1.8) reads

$$A\mathbf{p} = b,$$

where **p** designates an $(N+1)$-column vector $(\widehat{p}_0^{\pm}(x), ..., \widehat{p}_N^{\pm}(x))$, and A is the $(N+1)\times(N+1)$ matrix

$$A = \begin{pmatrix} a_{00} & a_{01} & \dots & a_{0N} \\ a_{10} & a_{11} & \dots & a_{1N} \\ \dots & \dots & \dots & \dots \\ a_{N0} & a_{N1} & \dots & a_{NN} \end{pmatrix}$$

and the $(N+1)\times 1$-column vector b is $(b_0, ..., b_N)^T$.

Now the truncation of the sum yields the approximation

$$\widehat{p}^{N\pm}(x,z) = \sum_{\ell=0}^{N} \widehat{p}_\ell^{\pm}(x)L_\ell(z),$$

which is analytic in $\widehat{\omega}$, so we can naturally extend it to $\widehat{\omega}$. Its extension is still denoted by $\widehat{p}^{N\pm}$. It is bounded over $\widehat{\omega}$ since $L_\ell(z)$ is bounded by 1. Such bounded extensions allow good approximation accuracy.

We determine two approximated extensions of p for the causal and the anti-causal parts,

$$\begin{aligned} p^{N\pm}(x,y) &= \widehat{p}^{N\pm}(x, T_y^{\pm}(y)) \\ &= \sum_{\ell=0}^{N} \widehat{p}_\ell^{\pm}(x) L_\ell(T_y^{\pm}(y)) \text{ for } y \in \omega_\infty^{\pm}, \end{aligned}$$

giving $T_y^{\pm}(x \mp y) = \rho^{\pm}(x)e^{-\sigma y} + \tau^{\pm}$: the extension of $y \mapsto p^{N\pm}(x, x \mp y)$ over $y \in (0, +\infty)$ states as

$$p^{N\pm}(x, x \mp y) = \sum_{\ell=0}^{N} \widehat{p}_\ell^{\pm}(x) L_\ell(\rho^{\pm}(x)e^{-\sigma y} + \tau^{\pm}),$$

for any $x \in \omega$.

For the sake of the uniformity of the method, the change of variable $s = 2x - 1$ is used to map ω into $\widehat{\omega}$, so the discretization with respect to x is done with a basis of modified Legendre polynomials as the formula (2.3.30) in [5], vanishing at the boundaries,

$$\eta_k(s) = (L_{k-2}(s) - L_k(s))/\sqrt{2(2k-1)}, \tag{1.9}$$

for $k \geq 2$ with $\eta_0(s) = \frac{1-s}{2}$ and $\eta_1(s) = \frac{1+s}{2}$. The extension is thus of the form

$$\widehat{p}^{N\pm}(s, T_y^{\pm}(y)) = \sum_{k=2}^{N_1} \sum_{\ell=0}^{N_2} p_{k\ell}^{\pm} \eta_k(s) L_\ell(T_y^{\pm}(y)),$$

N_1, N_2 being the numbers of base functions of a spectral method. The two first terms with respect to $k = 0$, $k = 1$ are ignored since the kernel satisfies the Dirichlet boundary conditions. The expression (1.6) is obtained by using $p^{N\pm}(x, y) = \widehat{p}^{N\pm}(2x - 1, T_y^{\pm}(y))$.

1.1.3/ DIFFUSIVE REALIZATION APPROXIMATION

The proposed approximation includes three steps : kernel approximation, discretization of the history functions with respect to x, and finally evaluation of the integrals thanks to the trapezoidal rule. It is stated under the form of algorithms in Subsection 1.1.3.1. Next subsections detail all steps of derivation. In Subsection 1.1.3.2, we present an approximation of the diffusive symbols $\mu^{\pm}$ based on the approximation extension of the kernel p. Then, computational algorithms for history functions $\psi^{\pm}$ and for $P^{\pm}u$ are derived in Subsection 1.1.3.3 and 1.1.3.4.

1.1.3.1/ STATEMENT OF THE APPROXIMATION

We recall the diffusive realization of an operator $Pu(x) = P^{+}u(x) + P^{-}u(x)$ where

$$P^{\pm}u(x) = \int_{\mathbb{R}} \mu^{\pm}(x,\xi)\psi^{\pm}(x,\xi)d\xi. \tag{1.10}$$

The approximations $\mu^{N\pm}$ of the symbols $\mu^{\pm}$ are based on the approximated extension of the kernel p, so they are the linear combinations,

$$\mu^{N\pm}(x,\xi) = \mp \frac{\theta^{\pm\prime}(\xi)}{2i\pi} \sum_{k=2}^{N_1} \sum_{\ell=0}^{N_2} p_{k\ell}^{\pm} \nu_k(x) \zeta_\ell^{\pm}(x, -\theta^{\pm}(\xi)), \tag{1.11}$$

of basis functions $\nu_k(x)$ and $\zeta_\ell^\pm(x, -\theta^\pm(\xi))$. Here $\nu_k(x) = \eta_k(2x-1)$ defined in (1.9) are modified Legendre polynomials in x, satisfying the Dirichlet condition. The rational fractions $\zeta_\ell^\pm$ in ξ are determined by the recurrence equation

$$\begin{aligned} \zeta_0^\pm(x,\lambda) = \frac{1}{\lambda},\ \zeta_1^\pm(x,\lambda) = \frac{\beta^\pm(x)}{\lambda+1} + \frac{\gamma^\pm}{\lambda}, \\ \zeta_{k+1}^\pm(x,\lambda) = \frac{2k+1}{k+1}\left(\beta^\pm(x)\zeta_k^\pm(x,\lambda+1) + \gamma^\pm\zeta_k^\pm(x,\lambda)\right) - \frac{k}{k+1}\zeta_{k-1}^\pm(x,\lambda). \end{aligned} \tag{1.12}$$

Indeed, the Legendre polynomials satisfy the recursion relation (see the expression (2.3.3) in [5])

$$L_{k+1}(z) = \frac{2k+1}{k+1} z L_k(z) - \frac{k}{k+1} L_{k-1}(z), \tag{1.13}$$

where $L_0(z) = 1$ and $L_1(z) = z$. Substituting z by $\beta^\pm(x)e^{-y} + \gamma^\pm$ and applying the Laplace transform to this expression, we get (1.12).

To discretize $\psi^\pm$ with respect to x, two x-discretizations are considered. They are based on two different interpolations of discrete inputs $(u_n)_{n+\frac{1}{2}}$ or $(u_n)_n$ located at regularly spaced nodes $(x_{n+\frac{1}{2}})_n$ or $(x_n)_n$ separated by a distance h, where $x_n = nh$ and $x_{n+\frac{1}{2}} = (n+\frac{1}{2})h$. In the interval $[x_n, x_{n+1})$, the first one is piecewise constant $u(x) = u_{n+\frac{1}{2}}$, and the second one is piecewise linear and continuous, $u(x) = u_n + \frac{u_{n+1}-u_n}{h}(x - x_n)$. We define $H = \frac{1}{h}$ the number of discretization nodes in x. With the piecewise constant interpolation of u, the recurrence relations of the approximations $\psi^{h\pm}$ of $\psi^\pm$ yield

$$\begin{aligned} \psi^{h+}(x_{n+1},\xi) &= \alpha^+(\xi)\psi^{h+}(x_n,\xi) + \beta^+(\xi)u_{n+\frac{1}{2}},\ \text{with } \psi^{h+}(0,\xi) = 0, \\ \psi^{h-}(x_n,\xi) &= \alpha^-(\xi)\psi^{h-}(x_{n+1},\xi) - \beta^-(\xi)u_{n+\frac{1}{2}},\ \text{with } \psi^{h-}(1,\xi) = 0, \end{aligned} \tag{1.14}$$

where $\alpha^\pm(\xi) = e^{-\theta^\pm(\xi)h}$, and $\beta^\pm(\xi) = \frac{\alpha^\pm(\xi)-1}{-\theta^\pm(\xi)}$. Moreover, with the piecewise linear interpolation of u, the recurrence relations of the approximations $\psi^{h\pm}$ of $\psi^\pm$ yield

$$\begin{aligned} \psi^{h+}(x_{n+1},\xi) &= \alpha^+(\xi)\psi^{h+}(x_n,\xi) + \gamma^+(\xi)u_n + \delta^+(\xi)u_{n+1},\ \text{with } \psi^{h+}(0,\xi) = 0, \\ \psi^{h-}(x_n,\xi) &= \alpha^-(\xi)\psi^{h-}(x_{n+1},\xi) + \delta^-(\xi)u_n + \gamma^-(\xi)u_{n+1},\ \text{with } \psi^{h-}(1,\xi) = 0, \end{aligned} \tag{1.15}$$

where $\delta^\pm(\xi) = \mp\frac{1}{-\theta^\pm(\xi)} + \frac{\beta^\pm(\xi)}{-\theta^\pm(\xi)h}$, and $\gamma^\pm(\xi) = \pm\frac{\alpha^\pm(\xi)}{-\theta^\pm(\xi)} - \frac{\beta^\pm(\xi)}{-\theta^\pm(\xi)h}$.

We replace $\mu^\pm$ and $\psi^\pm$ by their approximations $\mu^{N\pm}$ and $\psi^{h\pm}$. So the diffusive realization is approximated by

$$P^\pm u(x_n) \simeq \int_{\mathbb{R}} \mu^{N\pm}(x_n,\xi)\psi^{h\pm}(x_n,\xi)d\xi,\ \text{for all } n. \tag{1.16}$$

Evaluating the integrals thanks to the trapezoidal rule with $2M+1$ quadrature nodes regularly spaced at a distance $\hbar$ yields the final approximations, for the piecewise constant interpolation of u

$$\begin{aligned} P^+u_{n+1} &= \hbar\sum_{k=-M}^{M} \mu_{n+1,k}^{N+}\left(\alpha_k^+\psi_{n,k}^{h+} + \beta_k^+u_{n+\frac{1}{2}}\right), \\ P^-u_n &= \hbar\sum_{k=-M}^{M} \mu_{n,k}^{N-}\left(\alpha_k^-\psi_{n+1,k}^{h-} - \beta_k^-u_{n+\frac{1}{2}}\right), \end{aligned} \tag{1.17}$$

and for the piecewise linear interpolation of u

$$\begin{aligned} Pu_{n+1}^{+} &= \hbar \sum_{k=-M}^{M} \mu_{n+1,k}^{N+}(\alpha_k^{+}\psi_{n,k}^{h+} + \gamma_k^{+}u_n + \delta_k^{+}u_{n+1}), \\ Pu_{n}^{-} &= \hbar \sum_{k=-M}^{M} \mu_{n,k}^{N-}(\alpha_k^{-}\psi_{n+1,k}^{h-} + \delta_k^{-}u_n + \gamma_k^{-}u_{n+1}), \end{aligned} \tag{1.18}$$

where $P^{\pm}u_n = P^{\pm}u(x_n)$, $\mu_{n,k}^{N\pm} = \mu^{N\pm}(x_n,\xi_k)$, $\psi_{n,k}^{h\pm} = \psi^{h\pm}(x_n,\xi_k)$, $\alpha_k^{\pm} = \alpha^{\pm}(\xi_k)$, $\beta_k^{\pm} = \beta^{\pm}(\xi_k)$, $\gamma_k^{\pm} = \gamma^{\pm}(\xi_k)$, $\delta_k^{\pm} = \delta^{\pm}(\xi_k)$.

Moreover, we notice that a direct application of the residue theorem yields to eliminate the terms without exponential,

$$\int_{\mathbb{R}} \frac{\mu^{N\pm}(x,\xi)}{\theta^{\pm}(\xi)}d\xi = \int_{\mathbb{R}} \frac{\mu^{N\pm}(x,\xi)}{\theta^{\pm^2}(\xi)}d\xi = 0. \tag{1.19}$$

These results will be proved in Lemma 1. So for the piecewise constant interpolation of u, from (1.19) the diffusive realization is approximated by

$$\begin{aligned} P^{\pm}u(x_n) &\simeq \int_{\mathbb{R}} \mu^{N\pm}(x_n,\xi)\psi^{h\pm}(x_n,\xi)d\xi, \text{ for all } n, \\ &= \int_{\mathbb{R}} \mu^{N\pm}(x_n,\xi)\left(\psi^{h\pm}(x_n,\xi) \pm \frac{u_{n\mp\frac{1}{2}}}{-\theta^{\pm}(\xi)}\right)d\xi. \end{aligned} \tag{1.20}$$

Therefore, the final reformulated approximations,

$$\begin{aligned} P^{+}u_{n+1} &= \hbar \sum_{k=-M}^{M} \mu_{n+1,k}^{N+}\left(\alpha_k^{+}\psi_{n,k}^{h+} + \widetilde{\beta}_k^{+}u_{n+\frac{1}{2}}\right), \\ P^{-}u_{n} &= \hbar \sum_{k=-M}^{M} \mu_{n,k}^{N-}\left(\alpha_k^{-}\psi_{n+1,k}^{h-} - \widetilde{\beta}_k^{-}u_{n+\frac{1}{2}}\right), \end{aligned} \tag{1.21}$$

where $\widetilde{\beta}_k^{\pm} = \widetilde{\beta}^{\pm}(\xi_k) = \beta^{\pm}(\xi_k) + \frac{1}{-\theta^{\pm}(\xi_k)}$.

For the piecewise linear interpolation of u, we follow the same route to find

$$\begin{aligned} Pu_{n+1}^{+} &= \hbar \sum_{k=-M}^{M} \mu_{n+1,k}^{N+}(\alpha_k^{+}\psi_{n,k}^{h+} + \overline{\gamma}_k^{+}u_n + \overline{\delta}_k^{+}u_{n+1}), \\ Pu_{n}^{-} &= \hbar \sum_{k=-M}^{M} \mu_{n,k}^{N-}(\alpha_k^{-}\psi_{n+1,k}^{h-} + \overline{\delta}_k^{-}u_n + \overline{\gamma}_k^{-}u_{n+1}), \end{aligned} \tag{1.22}$$

with

$$\overline{\gamma}_k^{\pm} = \overline{\gamma}^{\pm}(\xi_k) = \frac{\alpha^{\pm}(\xi_k)}{-\theta^{\pm}(\xi_k)} - \frac{\gamma^{\pm}(\xi_k)}{-\theta^{\pm}(\xi_k)h},\ \overline{\delta}_k^{\pm} = \overline{\delta}^{\pm}(\xi_k) = \frac{\gamma^{\pm}(\xi_k)}{-\theta^{\pm}(\xi_k)h}.$$

For the sake of clarity, our method is synthesized in Algorithms 1 - 4. In the next chapter, these algorithms will be rewritten under a form of a prefix sum which is very suitable for parallel computation.

Offline computation of diffusive symbol $\mu^{N+}(x,\xi)$
Online computation
for $\ell = 0, ..., H$ **do**
  for k=1,...,M **do**
    $\psi^{h+}_{\ell+1,k} = \alpha^+_k \psi^{h+}_{\ell,k} + \beta^+_k u_{\ell+\frac{1}{2}}$, $\psi^{h+}_{0,k} = 0$
  end
  $P^+ u_{\ell+1} \simeq 2\hbar\Re\left(\sum_{k=1}^M \mu^{N+}_{\ell+1,k}(\alpha^+_k \psi^{h+}_{\ell,k} + \widetilde{\beta}^+_k u_{\ell+\frac{1}{2}})\right)$
end

Algorithme 1: Diffusive realization of $P^+u(x)$ for the piecewise constant interpolation.

Offline computation of diffusive symbol $\mu^{N-}(x,\xi)$
Online computation
for $\ell = 0, ..., H$ **do**
  for k=1,...,M **do**
    $\psi^{h-}_{\ell,k} = \alpha^-_k \psi^{h-}_{\ell+1,k} - \beta^-_k u_{\ell+\frac{1}{2}}$, $\psi^{h-}_{H,k} = 0$
  end
  $P^- u_\ell \simeq 2\hbar\Re\left(\sum_{k=1}^M \mu^{N-}_{\ell,k}(\alpha^-_k \psi^{h-}_{\ell+1,k} - \widetilde{\beta}^-_k u_{\ell+\frac{1}{2}})\right)$
end

Algorithme 2: Diffusive realization of $P^-u(x)$ for the piecewise constant interpolation.

1.1.3.2/ APPROXIMATION OF DIFFUSIVE SYMBOLS

As mentioned before, the approximations of the diffusive symbols are based on the approximated extension of the kernel p, and its final expression is stated in (1.11). In Chapter 3, an explicit approximation form of $\mu^{N\pm}$ is used for contour optimization which avoids the recurrence scheme. For convenience, we mention it here, namely,

$$\begin{aligned}
\mu^{N+}(x,\xi) &= -\frac{\theta^{+\prime}(\xi)}{2i\pi}\sum_{\ell=0}^{N_2}\widehat{p^+_\ell}(x)\sum_{s=0}^{\ell}\frac{e^{s(x-1)}d^+(\ell,s)}{-\theta^+(\xi)+s},\\
\mu^{N-}(x,\xi) &= \frac{\theta^{-\prime}(\xi)}{2i\pi}\sum_{\ell=0}^{N_2}\widehat{p^-_\ell}(x)\sum_{s=0}^{\ell}\frac{e^{-sx}d^-(\ell,s)}{-\theta^-(\xi)+s},
\end{aligned} \tag{1.23}$$

where $\widehat{p^\pm_\ell}$ are defined by

$$\widehat{p^\pm_\ell}(x) = \sum_{k=2}^{N_1} p^\pm_{k\ell}\nu_k(x), \text{ with } p^\pm_{k\ell} \text{ being already defined in (1.11)} \tag{1.24}$$

and

$$d^+(\ell,s) = (-1)^{\ell+s}C^s_\ell C^s_{\ell+s},\ d^-(\ell,s) = (-1)^s C^s_\ell C^s_{\ell+s}. \tag{1.25}$$

Indeed, to derive these expression, we use explicit expressions of the shifted Legendre polynomials $L^+_\ell(x) = L_\ell(2x-1)$ and $L^-_\ell(x) = L_\ell(-2x+1)$. From the formula (6) of Bhrawy et al. [1],

$$L^+_\ell(x) = \sum_{s=0}^{\ell} d^+(\ell,s)x^s \text{ and } L^-_\ell(x) = \sum_{s=0}^{\ell} d^-(\ell,s)x^s,$$

Offline computation of diffusive symbol $\mu^{N+}(x,\xi)$
Online computation
for $\ell = 0, ..., H$ **do**
  for k=1,...,M **do**
    $\psi^{h+}_{\ell+1,k} = \alpha^{+}_{k}\psi^{h+}_{\ell,k} + \gamma^{+}_{k}u_{\ell} + \delta^{+}_{k}u_{\ell+1}$, $\psi^{h+}_{0,k} = 0$
  end
  $P^{+}u_{\ell+1} = 2\hbar\Re\left(\sum_{k=1}^{M}\mu^{N+}_{\ell+1,k}(\alpha^{+}_{k}\psi^{h+}_{\ell,k} + \overline{\gamma}^{+}_{k}u_{\ell} + \overline{\delta}^{+}_{k}u_{\ell+1})\right)$
end

Algorithme 3: Diffusive realization of $P^{+}u(x)$ for the piecewise linear interpolation.

Offline computation of diffusive symbol $\mu^{N-}(x,\xi)$
Online computation
for $\ell = 0, ..., H$ **do**
  for k=1,...,M **do**
    $\psi^{h-}_{\ell,k} = \alpha^{-}_{k}\psi^{h-}_{\ell+1,k} + \delta^{-}_{k}u_{\ell} + \gamma^{-}_{k}u_{\ell+1}$, $\psi^{h-}_{H,k} = 0$
  end
  $P^{-}u_{\ell} = 2\hbar\Re\left(\sum_{k=1}^{M}\mu^{N-}_{\ell,k}(\alpha^{-}_{k}\psi^{h-}_{\ell+1,k} + \overline{\delta}^{-}_{k}u_{\ell} + \overline{\gamma}^{-}_{k}u_{\ell+1})\right)$
end

Algorithme 4: Diffusive realization of $P^{-}u(x)$ for the piecewise linear interpolation.

and since

$$L_{\ell}(x) = L^{+}_{\ell}\left(\frac{1+x}{2}\right) = L^{-}_{\ell}\left(\frac{1-x}{2}\right),$$

thus

$$L_{\ell}(\rho^{+}(x)e^{-y} + \tau^{+}) = L^{+}_{\ell}\left(\frac{1+\rho^{+}(x)e^{-y}+\tau^{+}}{2}\right) = L^{+}_{\ell}\left(e^{x-1}e^{-y}\right),$$
$$L_{\ell}(\rho^{-}(x)e^{-y} + \tau^{-}) = L^{-}_{\ell}\left(\frac{1-\rho^{-}(x)e^{-y}-\tau^{-}}{2}\right) = L^{-}_{\ell}\left(e^{-x}e^{-y}\right),$$

and

$$\zeta^{+}_{\ell}(x,-\theta^{+}(\xi)) = \mathcal{L}_{y}\left(L^{+}_{\ell}\left(e^{x-1}e^{-y}\right)\right)(-\theta^{+}(\xi)) = \sum_{s=0}^{\ell}\frac{e^{s(x-1)}d^{+}(\ell,s)}{-\theta^{+}(\xi)+s},$$
$$\zeta^{-}_{\ell}(x,-\theta^{-}(\xi)) = \mathcal{L}_{y}\left(L^{-}_{\ell}(e^{-x}e^{-y})\right)(-\theta^{-}(\xi)) = \sum_{s=0}^{\ell}\frac{e^{-sx}d^{-}(\ell,s)}{-\theta^{+}(\xi)+s}.$$

1.1.3.3/ DISCRETIZATION OF $\psi^{\pm}$ WITH RESPECT TO x

We detail the calculation for both causal and anti-causal parts used in Subsection 1.1.3.1, since it is not so trivial to deduce from each other. To proceed, we firstly consider the integral forms of (1.1), i.e.,

$$\psi^{+}(x,\xi) = \int_{0}^{x} e^{-\theta^{+}(\xi)(x-y)}u(y)\,dy, \tag{1.26}$$
$$\psi^{-}(x,\xi) = -\int_{x}^{1} e^{\theta^{-}(\xi)(x-y)}u(y)\,dy.$$

In particular, at a point $x = x_{n+1}$, we have

$$\begin{aligned}\psi^+(x_{n+1},\xi) &= \int_0^{x_{n+1}} e^{-\theta^+(\xi)(x_{n+1}-y)}u(y)\,dy \\ &= \int_0^{x_n} e^{-\theta^+(\xi)(x_{n+1}-y)}u(y)\,dy + \int_{x_n}^{x_{n+1}} e^{-\theta^+(\xi)(x_{n+1}-y)}u(y)\,dy,\end{aligned}$$

and at a point $x = x_n$,

$$\begin{aligned}\psi^-(x_n,\xi) &= -\int_{x_n}^{1} e^{\theta^-(\xi)(x_n-y)}u(y)\,dy \\ &= -\int_{x_{n+1}}^{1} e^{\theta^-(\xi)(x_n-y)}u(y)\,dy - \int_{x_n}^{x_{n+1}} e^{\theta^-(\xi)(x_n-y)}u(y)\,dy.\end{aligned}$$

Remarking that $x_{n+1} - x_n = h$, for all $n \in \{0, 1, ..., H-1\}$, we deduce the recurrence relations

$$\begin{aligned}\psi^+(x_{n+1},\xi) &= e^{-\theta^+(\xi)h}\psi^+(x_n,\xi) + \int_{x_n}^{x_{n+1}} e^{-\theta^+(\xi)(x_{n+1}-y)}u(y)\,dy,\ \psi^+(0,\xi) = 0, \\ \psi^-(x_n,\xi) &= e^{-\theta^-(\xi)h}\psi^-(x_{n+1},\xi) - \int_{x_n}^{x_{n+1}} e^{\theta^-(\xi)(x_n-y)}u(y)\,dy,\ \psi^-(1,\xi) = 0.\end{aligned}$$

Piecewise constant interpolation of u : The integrals turns to be equal to

$$\int_{x_n}^{x_{n+1}} e^{-\theta^+(\xi)(x_{n+1}-y)}u(y)\,dy \simeq u_{n+\frac{1}{2}}\Big[\frac{e^{-\theta^+(\xi)(x_{n+1}-y)}}{\theta^+(\xi)}\Big]_{y=x_n}^{y=x_{n+1}} = \beta^+(\xi)u_{n+\frac{1}{2}},$$

$$\text{and } \int_{x_n}^{x_{n+1}} e^{\theta^-(\xi)(x_n-y)}u(y)\,dy \simeq u_{n+\frac{1}{2}}\Big[\frac{e^{\theta^-(\xi)(x_n-y)}}{-\theta^-(\xi)}\Big]_{y=x_n}^{y=x_{n+1}} = \beta^-(\xi)u_{n+\frac{1}{2}}.$$

So the recurrence relations yield (1.14). We denote by $\psi^{h\pm}$ the approximations of $\psi^\pm$ which are solution to the exact recurrence relations,

$$\psi^{h+}(x_{n+1},\xi) = \alpha^+(\xi)\psi^{h+}(x_n,\xi) + \beta^+(\xi)u_{n+\frac{1}{2}} \text{ with } \psi^{h+}(0,\xi) = 0,$$
$$\text{and } \psi^{h-}(x_n,\xi) = \alpha^-(\xi)\psi^{h-}(x_{n+1},\xi) - \beta^-(\xi)u_{n+\frac{1}{2}} \text{ with } \psi^{h-}(1,\xi) = 0.$$

In Chapter 2, we require the solution of the above recurrence relation,

$$\begin{aligned}\psi^{h+}(x_{n+1},\xi) &= \beta^+(\xi)\sum_{j=0}^{n}(\alpha^+(\xi))^{n-j}u_{j+\frac{1}{2}} \\ \text{and } \psi^{h-}(x_n,\xi) &= -\beta^-(\xi)\sum_{j=n}^{H-1}(\alpha^-(\xi))^{j-n}u_{j+\frac{1}{2}}.\end{aligned} \tag{1.27}$$

Piecewise linear interpolation of u : The integrals are

$$\begin{aligned}&\int_{x_n}^{x_{n+1}} e^{-\theta^+(\xi)(x_{n+1}-y)}u(y)\,dy \simeq \int_{x_n}^{x_{n+1}} e^{-\theta^+(\xi)(x_{n+1}-y)}\Big(u_n + \frac{u_{n+1}-u_n}{h}(y-x_n)\Big)dy \\ =\ &\left[\frac{e^{-\theta^+(\xi)(x_{n+1}-y)}}{\theta^+(\xi)}\Big(u_n + \frac{u_{n+1}-u_n}{h}(y-x_n)\Big) - \Big(\frac{u_{n+1}-u_n}{h}\Big)\frac{e^{-\theta^+(\xi)(x_{n+1}-y)}}{\theta^+(\xi)^2}\right]_{y=x_n}^{y=x_{n+1}} \\ =\ &\frac{1}{\theta^+(\xi)}u_{n+1} - \frac{e^{-\theta^+(\xi)h}}{\theta^+(\xi)}u_n - \Big(\frac{u_{n+1}-u_n}{h}\Big)\frac{1-e^{-\theta^+(\xi)h}}{\theta^+(\xi)^2} \\ =\ &\gamma^+(\xi)u_n + \delta^+(\xi)u_{n+1},\end{aligned}$$

and

$$\begin{aligned}
&\int_{x_n}^{x_{n+1}} e^{\theta^-(\xi)(x_n-y)}u(y)\,dy \simeq \int_{x_n}^{x_{n+1}} e^{\theta^-(\xi)(x_n-y)}\left(u_n+\frac{u_{n+1}-u_n}{h}(y-x_n)\right)dy\\
=\;& \left[\frac{e^{\theta^-(\xi)(x_n-y)}}{-\theta^-(\xi)}\left(u_n+\frac{u_{n+1}-u_n}{h}(y-x_n)\right)-\left(\frac{u_{n+1}-u_n}{h}\right)\frac{e^{\theta^-(\xi)(x_n-y)}}{\theta^-(\xi)^2}\right]_{y=x_n}^{y=x_{n+1}}\\
=\;& \frac{e^{-\theta^-(\xi)h}}{-\theta^-(\xi)}u_{n+1}-\frac{1}{-\theta^-(\xi)}u_n-\left(\frac{u_{n+1}-u_n}{h}\right)\frac{e^{-\theta^-(\xi)h}-1}{\theta^-(\xi)^2}\\
=\;& \delta^-(\xi)u_n+\gamma^-(\xi)u_{n+1}.
\end{aligned}$$

So the recurrence relations are rewritten in (1.15). The approximations $\psi^{h\pm}$ of $\psi^\pm$ are solution to

$$\psi^{h+}(x_{n+1},\xi)=\alpha^+(\xi)\psi^{h+}(x_n,\xi)+\gamma^+(\xi)u_n+\delta^+(\xi)u_{n+1} \text{ with } \psi^{h+}(0,\xi)=0,$$
$$\text{and } \psi^{h-}(x_n,\xi)=\alpha^-(\xi)\psi^{h-}(x_{n+1},\xi)-\delta^-(\xi)u_n-\gamma^-(\xi)u_{n+1} \text{ with } \psi^{h-}(1,\xi)=0.$$

1.1.3.4/ APPROXIMATION OF DIFFUSIVE REALIZATIONS OF $P^\pm u$

Using the above recurrence relations, we establish the final approximations Pu_{n+1}^+ and Pu_n^- of P^+u and P^-u at the input nodes. Before describing them, we give an elementary lemma.

Lemma 1 :

For $\mu^{N\pm}$ defined as in (1.11) and $\theta^\pm$ defined as in (1.3), we have

$$\int_{\mathbb{R}}\frac{\mu^{N\pm}(x,\xi)}{-\theta^\pm(\xi)}d\xi=\int_{\mathbb{R}}\frac{\mu^{N\pm}(x,\xi)}{\theta^{\pm 2}(\xi)}d\xi=0.$$

Proof of Lemma 1

A standard proof of this lemma can be implemented by using a direct application of the residue theorem. However, we here give a more intuitive way. Namely, from (1.23), the diffusive symbol can be rewritten as

$$\mu^{N\pm}(x,\xi)=\frac{\theta^{\pm\prime}(\xi)}{2i\pi}\sum_{s=0}^{N}\frac{a_s^\pm(x)}{-\theta^\pm(\xi)+s},$$

where

$$a_s^+(x)=-\sum_{\ell=s}^{N}\widehat{p_\ell^+}(x)e^{s(x-1)}d^+(\ell,s) \text{ and } a_s^-(x)=\sum_{\ell=s}^{N}\widehat{p_\ell^-}(x)e^{-sx}d^-(\ell,s).$$

So we seek the value of

$$I^\pm(x,t)=\int_{\mathbb{R}}e^{-\theta^\pm(\xi)t}\frac{\mu^{N\pm}(x,\xi)}{-\theta^\pm(\xi)}d\xi \text{ at } t=0.$$

We operate a change of the variable by posing $\lambda=-\theta^\pm(\xi)$, so $d\lambda=-\theta^{\pm\prime}(\xi)d\xi$, then

$$I^\pm(x,t) \;=\; -\frac{1}{2\pi i}\int_{-\theta^\pm(\mathbb{R})}e^{\lambda t}\frac{1}{\lambda}\sum_{s=0}^{N}\frac{a_s^\pm(x)}{\lambda+s}d\lambda.$$

As in Talbot [34] and Weideman et al.[35], the contour $-\theta^{\pm}(\mathbb{R})$ can be replaced by an equivalent Bromwich contour $a+i\mathbb{R}$ where $a > a_0$ and a_0 is the convergence abscissa, i.e.,

$$\begin{aligned} I^{\pm}(x,t) &= -\frac{1}{2\pi i}\int_{a+i\mathbb{R}} e^{\lambda t}\frac{1}{\lambda}\sum_{s=0}^{N}\frac{a_s^{\pm}(x)}{\lambda+s}d\lambda \\ &= -a_0^{\pm}(x)tH(t) - \sum_{s=1}^{N}\frac{1}{s}(1-e^{-st})a_s^{\pm}(x)H(t), \end{aligned}$$

where $H(t)$ is the Heaviside step function. So $I^{\pm}(x,0) = 0$.

Similarly, we also have

$$\begin{aligned} J^{\pm}(x,t) &:= \int_{\mathbb{R}} e^{-\theta^{\pm}(\xi)t}\frac{\mu^{N\pm}(x,\xi)}{\theta^{\pm 2}(\xi)}d\xi \\ &= -\frac{1}{2\pi i}\int_{a+i\mathbb{R}} e^{\lambda t}\frac{1}{\lambda^2}\sum_{s=0}^{N}\frac{a_s^{\pm}(x)}{\lambda+s}d\lambda \\ &= -\frac{1}{2\pi i}\int_{a+i\mathbb{R}} e^{\lambda t}\left(\frac{a_0^{\pm}(x)}{\lambda^3} + \sum_{s=1}^{N}\left(\frac{1}{\lambda^2} - \frac{1}{\lambda(\lambda+s)}\right)\frac{a_s^{\pm}(x)}{s}\right)d\lambda \\ &= -a_0^{\pm}(x)\frac{t^2H(t)}{2} - \sum_{s=1}^{N}\left(tH(t) - \frac{1}{s}(1-e^{-st})H(t)\right)\frac{a_s^{\pm}(x)}{s}. \end{aligned}$$

Therefore, $J^{\pm}(x,0) = 0$. ■

Piecewise constant interpolation of u : We recall the formula of diffusive realizations $P^{\pm}u = \langle \mu^{\pm}, \psi^{\pm}\rangle$. We replace $\mu^{\pm}$ and $\psi^{\pm}$ by their approximations $\mu^{N\pm}$ and $\psi^{h\pm}$. So, from Lemma (1) the diffusive realization is approximated by

$$\begin{aligned} P^{\pm}u(x_n) &\simeq \int_{\mathbb{R}} \mu^{N\pm}(x_n,\xi)\psi^{h\pm}(x_n,\xi)d\xi, \text{ for all } n, \\ &= \int_{\mathbb{R}} \mu^{N\pm}(x_n,\xi)\overline{\psi}^{h\pm}(x_n,\xi)d\xi, \end{aligned} \tag{1.28}$$

where

$$\begin{aligned} \overline{\psi}^{h+}(x_n,\xi) &= \psi^{h+}(x_n,\xi) + \frac{u_{n-\frac{1}{2}}}{-\theta^{+}(\xi)}, \\ \overline{\psi}^{h-}(x_n,\xi) &= \psi^{h-}(x_n,\xi) - \frac{u_{n+\frac{1}{2}}}{-\theta^{-}(\xi)}. \end{aligned}$$

These changes of $\psi^{h\pm}$ are introduced to improve the numerical simulation. Indeed, when the input function u is constant in a neighbourhood of x, then $\psi^{\pm}(x,\xi)$ contain the terms $\frac{\mp u_{n\mp\frac{1}{2}}}{-\theta^{\pm}(\xi)}$. The diffusive realization $z^{N\pm}$ thus consist of the terms without exponential $\mp u_{n\mp\frac{1}{2}}\int_{\mathbb{R}}\frac{\mu^{N\pm}(x,\xi)}{-\theta^{\pm}(\xi)}d\xi$ and so decaying less fast than other parts. Moreover, we notice that an applying of the formula (1.19) allows to eliminate these terms. The approximation of the integrals $\int_{\mathbb{R}}$ using the trapezoidal rule thus avoids errors caused by these terms. Equivalently,

$$\begin{aligned} \overline{\psi}^{h+}(x_{n+1},\xi) &= \alpha^{+}(\xi)\psi^{h+}(x_n,\xi) + \widetilde{\beta}^{+}(\xi)u_{n+\frac{1}{2}}, \\ \overline{\psi}^{h-}(x_n,\xi) &= \alpha^{-}(\xi)\psi^{h-}(x_{n+1},\xi) - \widetilde{\beta}^{-}(\xi)u_{n+\frac{1}{2}}. \end{aligned} \tag{1.29}$$

Then we consider the discretization of the parameter $\xi \in \mathbb{R}$ of the contours $-\theta^{\pm}$ by $2M+1$ points ξ_k with $k = -M, -M+1, ..., M-1, M$. From this discretization, we can deduce the approximation of diffusive realizations of $P^{\pm}u(x_n)$ by the trapezoidal rule

$$\left(P^{N,h,M\pm}u\right)(x_n) = \hbar \sum_{k=-M}^{M} \mu^{N\pm}(x_n, \xi_k)\overline{\psi}^{h\pm}(x_n, \xi_k), \text{ for all } n,$$

with the step size $\hbar > 0$. This constitutes the approximation of diffusive realization

$$Pu_n \simeq \left(P^{N,h,M}u\right)(x_n) = \hbar \sum_{k=-M}^{M} \mu_{n,k}^{N+}\overline{\psi}_{n,k}^{h+} + \mu_{n,k}^{N-}\overline{\psi}_{n,k}^{h-}, \tag{1.30}$$

where $\overline{\psi}_{n,k}^{h\pm} = \overline{\psi}^{h\pm}(x_n, \xi_k)$.

Piecewise linear interpolation of u : We follow the same route to find

$$P^{\pm}u_n = \hbar \sum_{k=-M}^{M} \mu_{n,k}^{N\pm}\overline{\psi}_{n,k}^{h\pm}, \tag{1.31}$$

where

$$\begin{aligned}
\overline{\psi}_{n+1,k}^{h+} &= \alpha_k^{+}\psi^{h+}(x_n, \xi_k) + \overline{\gamma}_k^{+}u_n + \overline{\delta}_k^{+}u_{n+1}, \\
\overline{\psi}_{n,k}^{h-} &= \alpha_k^{-}\psi^{h-}(x_{n+1}, \xi_k) + \overline{\delta}_k^{-}u_n + \overline{\gamma}_k^{-}u_{n+1}.
\end{aligned}$$

1.1.3.5/ Relationship between the approximation of DR and the inversion of Laplace transforms

We establish that the approximation of the realization $P^{\pm}u$ can be written like a linear combination of inverse Laplace transforms $\mathcal{L}^{-1}$. We recall that

$$P^{\pm}u(x_n) \simeq \int_{\mathbb{R}} \mu^{N\pm}(x_n, \xi)\psi^{h\pm}(x_n, \xi)d\xi, \text{ for all } n. \tag{1.32}$$

Piecewise constant interpolation of u **:** Replacing $\psi^{h\pm}(x_n, \xi)$ by the formula (2.1), we get

$$\begin{aligned}
P^{\pm}u(x_n) \simeq \pm \sum_{j \in J_n^{\pm}} u_{j+\frac{1}{2}} \times \\
\left(\int_{\mathbb{R}} \frac{e^{-\theta^{\pm}(\xi)(\pm t_{n\pm1,j})}}{-\theta^{\pm}(\xi)}\mu^{N\pm}(x_n, \xi)d\xi - \int_{\mathbb{R}} \frac{e^{-\theta^{\pm}(\xi)(\pm t_{n,j})}}{-\theta^{\pm}(\xi)}\mu^{N\pm}(x_n, \xi)d\xi\right)
\end{aligned} \tag{1.33}$$

for all $n \in \{0, 1, ..., H-1\}$, with $J_n^{+} = \{0, 1, ..., n-1\}$, $J_n^{-} = \{n, n+1, ..., H-1\}$, $t_{n,j} = (n-j)h$. We see that

$$\begin{aligned}
\int_{\mathbb{R}} \frac{e^{-\theta^{\pm}(\xi)t}}{-\theta^{\pm}(\xi)}\mu^{N\pm}(x_n, \xi)d\xi &= \frac{1}{2\pi i}\int_{\mathbb{R}} \frac{e^{-\theta^{\pm}(\xi)t}}{-\theta^{\pm}(\xi)}\mathcal{P}(x_n, -\theta^{\pm}(\xi))(-\theta^{\pm\prime}(\xi))d\xi \\
&= \mathcal{L}^{-1}\left(\frac{-\theta^{\pm\prime}(\xi)\mathcal{P}(x_n, -\theta^{\pm}(\xi))}{-\theta^{\pm}(\xi)}\right)(t).
\end{aligned}$$

So if we pose

$$F_n^{\pm}(-\theta^{\pm}) = \frac{-\theta^{\pm\prime}(\xi)\mathcal{P}(x_n, -\theta^{\pm}(\xi))}{-\theta^{\pm}(\xi)},$$

then

$$P^{\pm}u(x_n) \simeq \pm \sum_{j \in J_n^{\pm}} u_{j+\frac{1}{2}} \times$$
$$\Big(\mathcal{L}^{-1}(F_n^{\pm}(-\theta^{\pm}))(\pm t_{n\pm1,j}) - \mathcal{L}^{-1}(F_n^{\pm}(-\theta^{\pm}))(\pm t_{n,j})\Big). \tag{1.34}$$

Weideman and Trefethen [35] have developed a contour optimization based on a balance between the truncation error and the discretization error for the numerical integration of the Laplace inversion at points $t_{\ell,j} \in I = \{h, 2h, ..., 1\}$ excluding the point 0. This approach is not effective in the present case since the ratio between the upper and lower bounds of the set I, i.e. $H = \frac{1}{h}$, is very large for a fine mesh and the numerical inversion of Laplace transform is relatively expensive. To circumvent this problem, we apply the results regarding the error estimate from Theorem 4.1 of Stenger [32]. They are applied for the evaluation of the discretization error of each pair $\mathcal{L}^{-1}[F^{\pm}(-\theta^{\pm})](\pm t_{\ell,j}) - [F^{\pm}(-\theta^{\pm})](\pm t_{\ell,j+1})$ of Laplace inverses. Moreover, the combination of these evaluations and the corresponding truncation errors gives us a global error estimate. Contour optimization based on such error estimates is reported in Chapter 3.

1.2/ AN EXAMPLE OF LYAPUNOV EQUATION AND THE NUMERICAL APPROXIMATIONS OF ITS SOLUTION

We consider a non-negative constant c and a positive, self-adjoint operator $Q \in \mathcal{L}(L^2(\omega))$ with kernel q and admitting a diffusive representation with symbols $\nu^{\pm}$. The Lyapunov equation states : find $P \in \mathcal{L}(H_0^1(\omega))$ such that

$$\int_\omega \left(\frac{du}{dx}\frac{d(Pv)}{dx} + \frac{d(P^*u)}{dx}\frac{dv}{dx} \right) dx = \int_\omega (Quv + c\, uv) dx \text{ for all } u, v \in H_0^1(\omega), \tag{1.35}$$

We divide Ω into two open sets Ω^+ and Ω^- corresponding to the causal ($y < x$) and anti-causal ($y > x$) parts of Ω. The boundary of Ω^+ is split into $\Gamma_y^+ = \{1\} \times \omega$, $\Gamma_x^+ = \omega \times \{0\}$ and $\Gamma_0 = \{(x, y) \in \Omega \text{ s.t. } x = y\}$, when the boundary of Ω^- is split into $\Gamma_y^- = \{0\} \times \omega$, $\Gamma_x^- = \omega \times \{1\}$ and Γ_0. We define $\widehat{\Gamma}_x^{\pm} = T^{\pm}(\Gamma_x^{\pm})$, $\widehat{\Gamma}_y^{\pm} = T^{\pm}(\Gamma_y^{\pm})$ and $\widehat{\Gamma}_0^{\pm} = T^{\pm}(\Gamma_0)$.

We will seek the solution P as a kernel operator with kernel $p(x, y)$, the kernel of its adjoint P^* being $p^*(x, y) = p(y, x)$. We can readily prove that p is symmetric, i.e., that $p^* = p$, by interchanging u with v, and then x with y. Now, we derive the equations satisfied by the kernel p in Ω^+ and in Ω^-. For brevity, we use the notations p^+ and p^- for $p_{|\Omega^+}$ and $p_{|\Omega^-}$ the kernel of the causal and anti-causal parts.

Proposition 1 : Adapted from Proposition 19 in [16]

The causal part p^+ and the anti-causal part p^- of the kernel p are the unique solutions to the two uncoupled boundary value problems

$$\begin{aligned} -\Delta p^{\pm} &= q^{\pm} \text{ in } \Omega^{\pm}, \\ (-\partial_x + \partial_y) p^{\pm} &= \pm \frac{c}{2} \text{ on } \Gamma_0, \; p^{\pm} = 0 \text{ on } \Gamma_x^{\pm} \cup \Gamma_y^{\pm}. \end{aligned} \tag{1.36}$$

In the particular case $c = 0$, p is the unique solution to

$$-\Delta p = q \text{ in } \Omega, \text{ and } p = 0 \text{ on } \partial\Omega. \tag{1.37}$$

1.2.1/ CHANGE OF VARIABLES

The construction of the approximation of $p^{\pm}$ is built by using the mapping $T^{\pm}$. The weak formulation corresponding to (1.36) is : $p^{\pm} \in H^1_{\Gamma_x^{\pm} \cup \Gamma_y^{\pm}}(\Omega)$ satisfies

$$\int_{\Omega^{\pm}} \nabla p^{\pm} \nabla w^{\pm} dy dx = \frac{1}{\sqrt{2}} \int_{\Gamma_0} c w^{\pm} ds + \int_{\Omega^{\pm}} q^{\pm} w^{\pm}\, dy dx \text{ for all } w^{\pm} \in H^1_{\Gamma_x^{\pm} \cup \Gamma_y^{\pm}}(\Omega^{\pm}).$$

We remark that with parameterized curves

$$\begin{aligned} \Gamma_0 &= \{(x(t), y(t)) = (t, t) : t \in (0, 1)\} \\ \text{and } \widehat{\Gamma}_0^{\pm} &= \left\{ (\widehat{x}(t), \widehat{y}(t)) = \left(t, T_y^{\pm}(\frac{t+1}{2}) \right) : t \in (-1, 1) \right\}, \end{aligned}$$

$$
\begin{aligned}
I &:= \frac{1}{\sqrt{2}}\int_{\Gamma_0} c\, w^{\pm}(x,y)ds = \int_0^1 c\, w^{\pm}(t,t)dt \\
&= \int_0^1 c\, \left(w^{\pm}\circ(T^{\pm})^{-1}\right)\circ(T^{\pm})(t,t)dt = \int_0^1 c\, \widehat{w^{\pm}}\left(2t-1, T_y^{\pm}(t)\right)dt \\
&= \frac{1}{2}\int_{-1}^1 c\, \widehat{w^{\pm}}\left(t, T_y^{\pm}(\frac{t+1}{2})\right)\left(\sqrt{1+\left(\partial_t T_y^{\pm}(\frac{t+1}{2})\right)^2}\right) r^{\pm}\left(T_y^{\pm}(\frac{t+1}{2})\right)dt,
\end{aligned}
$$

where

$$
\begin{aligned}
r^{\pm}(\widehat{y}) &= \left(\sqrt{1+(\partial_t\widehat{y}(t))^2}\right)^{-1} \\
&= \left(\sqrt{1+\frac{\sigma^2}{4}(1\pm\widehat{y}(t))^2}\right)^{-1} \\
&= \left(\sqrt{1+\frac{\chi^{2\pm}(\widehat{y})}{4}}\right)^{-1} \text{ (note : } \chi \text{ was defined in (1.5)).}
\end{aligned}
$$

So

$$
I = \frac{1}{2}\int_{\widehat{\Gamma}_0^{\pm}} c\, \widehat{w^{\pm}}(\widehat{x},\widehat{y}) r^{\pm}(\widehat{y})ds.
$$

Thus the weak formulation satisfied by $\widehat{p^{\pm}} = p^{\pm}\circ(T^{\pm})^{-1}$ is : $\widehat{p^{\pm}} \in H^1_{\widehat{\Gamma}_x^{\pm}\cup\widehat{\Gamma}_y^{\pm}}(\widehat{\Omega}^{*\pm})$

$$
\int_{\widehat{\Omega}^{*\pm}} \widehat{a^{\pm}}\nabla\widehat{p^{\pm}}.\nabla\widehat{w^{\pm}}\, dsdz = \frac{1}{2}\int_{\widehat{\Gamma}_0^{\pm}} c\widehat{w^{\pm}} r^{\pm} ds + \int_{\widehat{\Omega}^{*\pm}} \widehat{q^{\pm}}\widehat{w^{\pm}}\frac{dsdz}{2\chi^{\pm}}, \tag{1.38}
$$

$$
\text{for all } \widehat{w^{\pm}} \in H^1_{\widehat{\Gamma}_x^{\pm}\cup\widehat{\Gamma}_y^{\pm}}(\widehat{\Omega}^{*\pm}),
$$

where

$$
\widehat{a}^{\pm}_{jk} = \left[\frac{\partial T_j^{\pm}}{\partial x}\circ(T^{\pm})^{-1}\frac{\partial T_k^{\pm}}{\partial x}\circ(T^{\pm})^{-1} + \frac{\partial T_j^{\pm}}{\partial y}\circ(T^{\pm})^{-1}\frac{\partial T_k^{\pm}}{\partial y}\circ(T^{\pm})^{-1}\right]|\det(D(T^{\pm})^{-1})|
$$

$$
\text{and } \widehat{q^{\pm}} = q\circ(T^{\pm})^{-1}.
$$

If we detail the expressions

$$
\widehat{a}^{\pm}_{11} = \frac{2}{\chi^{\pm}},\ \widehat{a}^{\pm}_{22} = \frac{\chi^{\pm}}{2},\ \widehat{a}^{\pm}_{12} = \widehat{a}^{\pm}_{21} = 0,
$$

then the associated boundary value problem is

$$
\begin{aligned}
-4\frac{\partial^2\widehat{p^{\pm}}}{\partial s^2} - \chi^{\pm}\frac{\partial}{\partial z}(\chi^{\pm}\frac{\partial\widehat{p^{\pm}}}{\partial z}) &= \widehat{q^{\pm}} \text{ in } \widehat{\Omega}^{*\pm}, \\
\text{and } (-2\partial_s + \chi^{\pm}\partial_z)\widehat{p^{\pm}} &= \pm\frac{c}{2} \text{ on } \widehat{\Gamma}_0^{\pm},\ \widehat{p^{\pm}} = 0 \text{ on } \widehat{\Gamma}_x^{\pm}\cup\widehat{\Gamma}_y^{\pm}.
\end{aligned} \tag{1.39}
$$

We restart from

$$
-\frac{4}{\chi^{\pm}}\frac{\partial^2\widehat{p^{\pm}}}{\partial s^2} - \frac{\partial}{\partial z}(\chi^{\pm}\frac{\partial\widehat{p^{\pm}}}{\partial z}) = \frac{\widehat{q^{\pm}}}{\chi^{\pm}} \text{ in } \widehat{\Omega}^{*\pm},
$$

and multiply by any $v \in H^1_{\widehat{\Gamma}^{\pm}_x \cup \widehat{\Gamma}^{\pm}_y}(\widehat{\Omega}^{*\pm})$ so that to get

$$\int_{\widehat{\Omega}^{*\pm}} \frac{4}{\chi^{\pm}} \frac{\partial \widehat{p^{\pm}}}{\partial s} \frac{\partial \widehat{v^{\pm}}}{\partial s} + \chi^{\pm} \frac{\partial \widehat{p^{\pm}}}{\partial z} \frac{\partial \widehat{v^{\pm}}}{\partial z}\, dsdz = \frac{1}{2} \int_{\widehat{\Gamma}^{\pm}_0} c\widehat{v^{\pm}} r^{\pm} dS + \int_{\widehat{\Omega}^{*\pm}} \frac{\widehat{q^{\pm}}}{\chi^{\pm}} \widehat{v^{\pm}}\, dsdz. \tag{1.40}$$

Finally, Dirichlet condition is penalized,

$$\begin{aligned} &\int_{\widehat{\Omega}^{*\pm}} \frac{4}{\chi^{\pm}} \frac{\partial \widehat{p^{\pm}}}{\partial s} \frac{\partial \widehat{v^{\pm}}}{\partial s} + \chi^{\pm} \frac{\partial \widehat{p^{\pm}}}{\partial z} \frac{\partial \widehat{v^{\pm}}}{\partial z}\, dsdz + \frac{1}{\varepsilon} \int_{\widehat{\Gamma}^{\pm}_x \cup \widehat{\Gamma}^{\pm}_y} \widehat{p^{\pm}} \widehat{v^{\pm}} ds \\ &= \frac{1}{2} \int_{\widehat{\Gamma}^{\pm}_0} c\widehat{v^{\pm}} r^{\pm} ds + \int_{\widehat{\Omega}^{*\pm}} \frac{\widehat{q^{\pm}}}{\chi^{\pm}} \widehat{v^{\pm}}\, dsdz \text{ for all } \widehat{v^{\pm}} \in H^1(\widehat{\Omega}^{*\pm}). \end{aligned} \tag{1.41}$$

1.2.2/ Approximation of $\widehat{p^{\pm}}$ and its extension

The solution $\widehat{p^{\pm}}$ and the test function $\widehat{v^{\pm}}$ are decomposed on the same modal basis

$$\Phi_{k\ell}(s,z) = \eta_k(s) L_\ell(z)$$

defined in $\widehat{\Omega}^{*\pm}$ with polynomial degrees N_1 and N_2.

The Galerkin method states as

$$\begin{aligned} &\sum_{k'=2}^{N_1} \sum_{\ell'=0}^{N_2} \left[\int_{\widehat{\omega}} \int_{\widehat{\omega}^{*\pm}} (\nabla \Phi_{k'\ell'})^T B^{\pm} (\nabla \Phi_{k\ell}) B^{\pm} \frac{dsdz}{\chi^{\pm}} + \frac{1}{\varepsilon} \int_{\widehat{\Gamma}^{\pm}_x \cup \widehat{\Gamma}^{\pm}_y} \Phi_{k'\ell'} \Phi_{k\ell} ds \right] \widehat{p^{\pm}_{k'\ell'}} \\ = \;& \frac{1}{2} \int_{\widehat{\Gamma}^{\pm}_0} c\, \Phi_{k\ell} r^{\pm} ds + \int_{\widehat{\omega}} \int_{\widehat{\omega}^{*\pm}} \widehat{q^{\pm}} \Phi_{k\ell} \frac{dsdz}{\chi^{\pm}}, \end{aligned}$$

with

$$B^{\pm} = \begin{pmatrix} 2 & 0 \\ 0 & \chi^{\pm}(z) \end{pmatrix}.$$

We pose

$$\begin{aligned} f_{k\ell k'\ell'} &= \frac{1}{\chi^{\pm}} [(\nabla \Phi_{k'\ell'})^T B^{\pm}].[(\nabla \Phi_{k\ell})^T B^{\pm}], \\ h_{k\ell k'\ell'} &= \frac{1}{\varepsilon} \Phi_{k'\ell'} \Phi_{k\ell},\ g_{k\ell} = \frac{\widehat{q^{\pm}} \Phi_{k\ell}}{\chi^{\pm}}, \\ d_{k\ell}(s) &= \frac{1}{2} c\, \Phi_{k\ell} \left(s, T^{\pm}_y(\frac{s+1}{2}) \right). \end{aligned}$$

The G-NI method states as (based on the Legendre Gauss-Lobato quadrature rule)

$$\begin{aligned} &\sum_{k'=2}^{N_1} \sum_{\ell'=0}^{N_2} \sum_{i=0}^{N_3} \left[\sum_{j=0}^{N_4} f_{klk'\ell'}(x_i, x'_j) w'_j + h_{k\ell k'\ell'}(x_i, \pm 1) \right] w_i \widehat{p^{\pm}_{k'\ell'}} \\ &= \sum_{i=0}^{N_3} d_{k\ell}(x_i) w_i + \sum_{i=0}^{N_3} \sum_{j=0}^{N_4} g_{k\ell}(x_i, x'_j) w_i w'_j. \end{aligned}$$

This yields the linear system

$$\sum_{k'=2}^{N_1} \sum_{\ell'=0}^{N_2} A^{\pm}_{k\ell k'\ell'} \widehat{p^{\pm}_{k'\ell'}} = F^{\pm}_{k\ell},$$

where

$$A^{\pm}_{k\ell k'\ell'} = \sum_{i=0}^{N_3}\left[\sum_{j=0}^{N_4} f_{k\ell k'\ell'}(x_i,x'_j)w'_j + h_{k\ell k'\ell'}(x_i,\pm 1)\right]w_i,$$
$$\text{and } F^{\pm}_{k\ell} = \sum_{i=0}^{N_3} d_{k\ell}(x_i)w_i + \sum_{i,j=0}^{N_3,N_4} g_{k\ell}(x_i,x'_j)w_i w'_j.$$

The transformed kernel $\widehat{p}^{\pm}$ of $p^{\pm}$ defined on $\widehat{\Omega}^{*\pm}$ has the form,

$$\widehat{p}^{N\pm}(s,z) = \sum_{k=2}^{N_1}\sum_{\ell=0}^{N_2} p^{\pm}_{k\ell}\eta_k(s)L_\ell(z).$$

Then this approximation will be naturally extended to $\widehat{\Omega}$. We still keep denoting by $\widehat{p}^{N\pm}$ these two extensions

$$\widehat{p}^{N\pm}(s,T^{\pm}_y(y)) = \sum_{k=2}^{N_1}\sum_{\ell=0}^{N_2} p^{\pm}_{k\ell}\eta_k(s)L_\ell(T^{\pm}_y(y)), \text{ for } z\in\widehat{\Omega}^{\pm}.$$

1.2.3/ DIFFUSIVE REALIZATION APPROXIMATION

- The approximations $\mu^{N\pm}$ of the diffusive symbols $\mu^{\pm}$ based on the above approximation extension of the kernel p read as (1.11).
- The discretizations of $\psi^{\pm}$ with respect to x based on the piecewise constant interpolation of u state as (1.14) and the discretizations of $\psi^{\pm}$ based on the piecewise linear interpolation of u state as (1.15).
- The approximations of diffusive realizations of $P^{\pm}u$ based on some changes of $\psi^{\pm}$ to improve the numerical simulation state as (1.17) for the piecewise constant interpolation of u, and state as (1.31) for the piecewise linear interpolation of u.

1.2.4/ NUMERICAL TEST

Referring to the example (1.37) introduced above, wherein the kernel

$$q(x,y) = -1(1-3z)(1-y)y^2 - 2(1-x)x^2(1-3y). \tag{1.42}$$

The analytic solution of this system is the kernel p

$$p(x,y) = (1-x)x^2(1-y)y^2, \text{ for all } (x,y)\in\Omega. \tag{1.43}$$

Offline computation : In Figure 1.4, we present the relative errors in $L^2(\Omega)$ norm

$$e^{\pm} = \frac{\|p^{\pm} - p^{N\pm}\|_{L^2(\Omega)}}{\|p^{\pm}\|_{L^2(\Omega)}} \tag{1.44}$$

between $p^{\pm}$ and their approximation $p^{N\pm}$ as a function of $N = N_1 = N_2$, with $N = 1, 2, ..., 20$. In the following, the polynomial degrees of $p^{N\pm}$ are fixed at sufficiently large values, $N_1 = N_2 = 10$, so that the relative error $e^{\pm}$ is negligible, namely in the order of magnitude of e^{-10} as shown in Figure 1.5 representing p and its approximations.

1.3/ CONCLUSION

In this chapter, we have presented both the general theory of the DR and its approximation. The approximation of diffusive symbols is based on the dedicated spectral approximation of the kernel. The spectral method combines the basis of Legendre polynomials in both x and y with an extension in y. Two x-discretizations of the history function are based on two different interpolations of discrete inputs located at regularly spaced nodes $(x_n)_n$ separated by a distance h. In the interval $[x_n, x_{n+1})$, the first one is a piecewise constant interpolation $u(x) = u_{n+\frac{1}{2}}$, and the second one is a piecewise linear and continuous interpolation, $u(x) = u_n + \frac{u_{n+1}-u_n}{h}(x - x_n)$, where $u_n = u(nh), u_{n+\frac{1}{2}} = u(nh + \frac{1}{2}h)$. Moreover, our general approach is presented through the example of a Lyapunov equation arising in optimal control theory of the one-dimensional heat equation.

In the next chapter, we evaluate the performance of the method for three parallel topologies : line topology, hypercube topology, binary topology. We also compare the computation times between our algorithm with these topologies and a direct spectral method with a line topology. All these results are carried out on the same operator P as in [16].

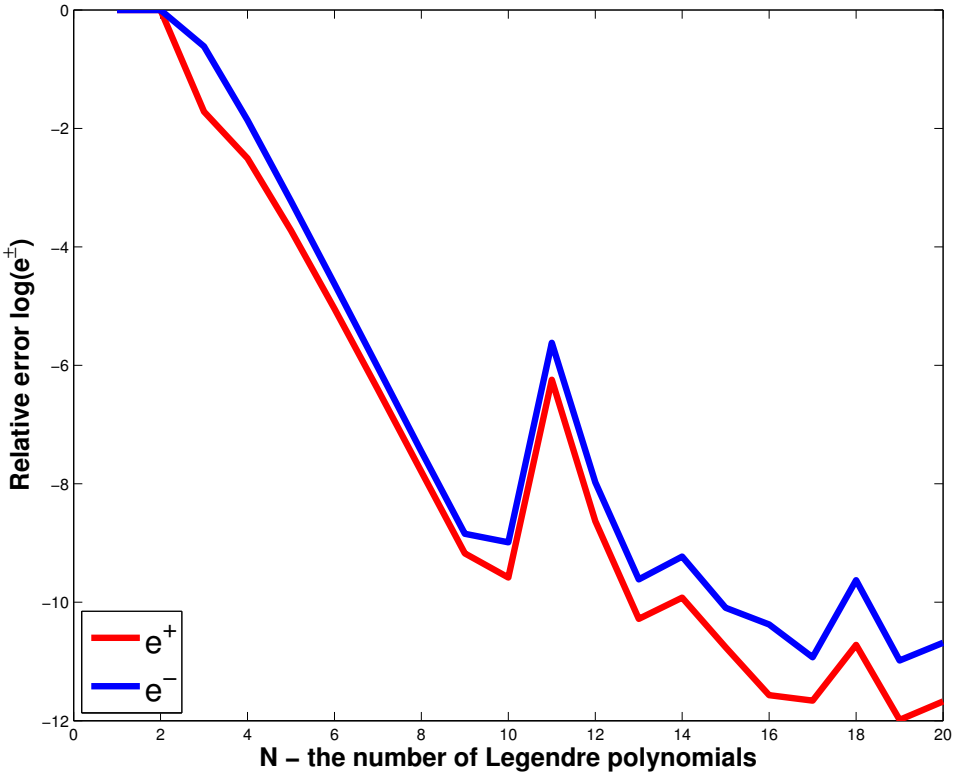


FIGURE 1.4 – Relative errors e^+ (red color) and e^- (blue color) in the norm L^2, defined by (1.44) in the function of $N = (N_1, N_2)$ in a logarithmic scale.

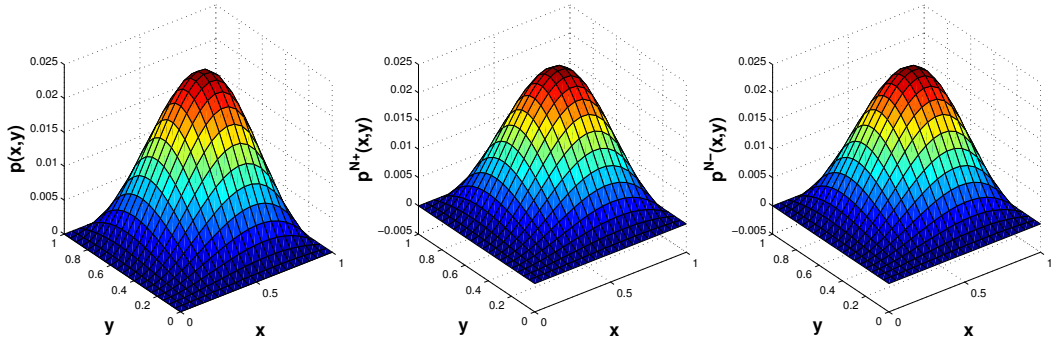


FIGURE 1.5 – Profile $x-y$ of the kernels (a) $p(x,y)$, (b) $p^{N+}(x,y)$, (c) $p^{N-}(x,y)$ for $N = (10, 10)$.

2

PARALLEL COMPUTATION

As we have already mentioned in the previous chapter, one of main advantages of the diffusive realization (DR) is its very low computational cost. We also observed that the diffusive realization decomposes an operator as a linear combination of local operators namely of differential operators. Such decomposition seems to be well suited for implementation on distributed computing architectures. Combining the two features opens a new direction to develop embedded real-time computation for distributed system on distributed architectures with parallel implementation. Hence, it yields embedded, massive, scalable and low-cost computation.

In order to implement the DR method close to the system to control, it seems interesting to use microcontrollers. In fact, they have the advantages to be small, to be cheap and they have real-time computing capacity. In order to control a system which is distributed a network of microcontroller can be used. In fact, field-programmable gate array (FPGA) based solutions can also be viewed as microcontrollers because they can be cheap and they are real-time. In this chapter, our effort are focused on the algorithms and not on a real implementation.

In order to propose a scalable solution, an architecture with communications between direct neighbours is preferred. Considering only the case where the control is performed in one dimension (1D), there are different possible topologies to make the microcontrolers communicate. In this chapter, three network topologies are considered : a line, a binary tree and a hypercube. The line topology is the simplest one. The binary tree and the hypercube allow faster diffusion of the information in the network.

The computation of $\psi^{\pm}$ - a main part of our algorithm synthesized in Algorithm 5 - can be reformulated into a prefix sum which is very suitable for parallel computation (see also Leighton [14], Chapter 1, Section 1.2.2). We will describe and present results for three parallel architectures well suited for prefix computation. We also compute and compare the computation time between our algorithm with topologies and a direct method with a line topology.

The parallel computation of a prefix sum with a line topology is impossible with one sensor per processor or microcontroller. Hence, we consider that our network is composed of k processors and that a processor controls $m \geq 2$ sensors (i.e., inputs). So the input u is discretized by $k \times m$ points.

The outline of this chapter is as follows. In Section 2.1 we estimate not only the number of operations but also the execution time of three topologies of the DR method. The similar estimation for direct method is considered in Section 2.2. Some comparisons of number

of operations and the computation time between two methods are presented in Section 2.3.

2.1/ PARALLEL ALGORITHM FOR THE DR METHOD

In this section we describe the implementations of our algorithm on different parallel topologies. We recall that both ψ^+ and ψ^- store a part of the *history* of the input data u. They are respectively solution to the forward and backward ordinary differential equation in x,

$$\begin{aligned} \partial_x\psi^+(x,\xi) + \theta^+(\xi)\psi^+(x,\xi) &= u(x), \text{ with } \psi^+(0,\xi) = 0, \\ \partial_x\psi^-(x,\xi) - \theta^-(\xi)\psi^-(x,\xi) &= u(x), \text{ with } \psi^-(1,\xi) = 0, \end{aligned}$$

parametetrized by $\xi \in \mathbb{R}$.

The algorithm is based on the *history function* ψ^+ for the diffusive realizations of the causal part z^+. Applying the elementary lemma 2, the equation

$$\psi^+(x_{\ell+1},\xi_k) \simeq \alpha^+(\xi_k)\psi^+(x_\ell,\xi_k) + \beta^+(\xi_k)u_\ell,\ \psi^+(0,\xi_k) = 0, \tag{2.1}$$

can be rewritten under a general form.

Lemma 2 :

The general form of the sequence of numbers $y_{\ell+1} = \alpha y_\ell + b_\ell$, with $y_0 = 0, \ell = 0, 1, ...$ is

$$y_{\ell+1} = \sum_{i=0}^{\ell} \alpha^{\ell-i} b_i.$$

Proof A standard proof of this lemma can be implemented by using the inductive reasoning. However, we here give a more intuitive way. We have

$$\begin{aligned} y_{\ell+1} &= \alpha y_\ell + b_\ell \\ \alpha y_\ell &= \alpha^2 y_{\ell-1} + \alpha b_{\ell-1} \\ \alpha^2 y_{\ell-1} &= \alpha^3 y_{\ell-2} + \alpha^2 b_{\ell-2} \\ \cdots \quad &\cdots \quad \cdots \\ \alpha^\ell y_1 &= \alpha^{\ell+1} y_0 + \alpha^\ell b_0. \end{aligned}$$

So the summation of all left sides is equal to the summation of all right sides. It means

$$y_{\ell+1} = \alpha^{\ell+1} y_0 + \sum_{i=0}^{\ell} \alpha^{\ell-i} b_i.$$

The term $\alpha^{\ell+1} y_0$ is suppressed since $y_0 = 0$, so

$$y_{\ell+1} = \sum_{i=0}^{\ell} \alpha^{\ell-i} b_i. \blacksquare$$

Namely, the equation (2.1) is rewritten as

$$\psi^+_{\ell+1,k} = \sum_{i=0}^{\ell} (\alpha^+_k)^{\ell-i} \beta^+_k u_i,$$

with $\psi^+_{\ell,k} = \psi^+(x_\ell, \xi_k), \alpha^+_k = \alpha^+(\xi_k), \beta^+_k = \beta^+(\xi_k)$. So if we define $\varphi^+_{\ell,k}$ as

$$\varphi^+_{\ell,k} = \sum_{i=0}^{\ell} (\alpha^+_k)^{-i} \beta^+_k u_i,$$

then $\psi^+_{\ell+1,k} = \varphi^+_{\ell,k}(\alpha^+_k)^\ell$. The variable φ^+ is used to make a prefix sum.

Definition The prefix sum (scan, or cumulative sum) of a sequence of numbers $x_0, x_1, x_2, ...$ is a second sequence of numbers $y_0, y_1, y_2, ...$, the sums of prefixes (running totals) of the input sequence :

$$\begin{aligned} y_0 &= x_0 \\ y_1 &= x_0 + x_1 \\ y_2 &= x_0 + x_1 + x_2 \\ &... \end{aligned}$$

It should be noticed that prefix sums have been well studied in parallel algorithms.

The resulting algorithm is summarized in Algorithm 5. Note that the implementation of the

Offline computation of diffusive symbol
$\mu^{N+}(x,\xi)$, $(\alpha^+)^{-\ell}\beta^+$ and $(\alpha^+)^\ell$, $\ell \in \{0, ..., H-1\}$.
Online computation
for $\ell = 0, ..., H-2$ **do**
  for k=1,...,M **do**
    $\varphi^+_{\ell+1,k} = \varphi^+_{\ell,k} + (\alpha^+_k)^{-\ell-1}\beta^+_k u_{\ell+1}$, $\varphi^+_{0,k} = 0$;
  end
end
for $\ell = 0, ..., H-2$ **do**
  for k=1,...,M **do**
    $\psi^+_{\ell+1,k} = \varphi^+_{\ell,k}(\alpha^+_k)^\ell$;
  end
  $z^+_{\ell+1} = 2h_\xi \Re\left(\sum_{k=1}^n \mu^{N+}_{\ell+1,k}(\alpha^+_k \psi^+_{\ell,k} + \gamma^+_k u_\ell)\right)$;
end

Algorithme 5: Reformulated algorithm for the diffusive realization of $z^+(x)$.

anti-causal part is done in a similar way, and will not be described in detail. Consequently, we will drop all upper indices "+" without any risk of confusion.

To be able to implement this algorithm in parallel, the first loop, which intends to compute a prefix sum, needs to be executed in parallel. Then the estimate of the contour is a computation where no communication between the nodes is required. The computation time is optimized by adapting the topologies in order to implement efficiently a prefix sum. In Subsection 2.1.1-2.1.3, we derive three parallel topologies, well suited for the diffusive realization. Namely, they are a line topology, a hypercube topology and a binary topology. Each of them is considered by designing its corresponding network, estimating the number of operations and transmissions, then inferring its execution time. A line topology for a direct spectral method is also described in Section 2.2.

2.1.1/ DR ALGORITHM WITH A LINE TOPOLOGY

2.1.1.1/ DESCRIPTION OF A LINE NETWORK

We consider a line network. In the general case, a processor in this network is connected with bidirectional links to its *left neighbour* and *right neighbour*. The outermost processors may have just one connection each. We consider that each processor has a local program which can read a few inputs. The complexity of the local program and the size of the local memory may vary. We assume that the local program is simple (i.e., it consists of a few operations). At each step, each processor :

1. receives input from its neighbours,
2. reads a few inputs,
3. performs a computation,
4. generates output for its neighbours.

If each processor of the line network only controls one sensor (i.e., one input), the parallel computation of a prefix sum on this line is impossible. In this case, the H^{th} prefix sum cannot be computed until the $(H-1)^{st}$ prefix sum has been computed, which in turn cannot be computed until the $(H-2)^{nd}$ prefix sum has been computed, and so on. Therefore, with this line network we need at least H steps to add H inputs in the H^{th} processor (where the H^{th} prefix sum is stored). In other word, a prefix sum on a line can not be computed in parallel.

In order to maximize parallelism we consider that our network is composed of k processors and that a processor controls $m \geq 2$ sensors (i.e., inputs). So the input u is discretized by $k \times m$ points. In this case there are $(k-1)$ links. We assume that there are two lines in our network, one for the causal part and another one for the anti-causal part. The communication of the causal part starts from the first processor to the last one, while that of the anti-causal starts from the last one to end on the first one. In fact, in order to implement the computation of the causal part, a processor should receive data from its *left neighbour*. In order to implement the computation of the anti-causal part it should receive data from its *right neighbour*. The detail of communications is shown in Figure 2.1.

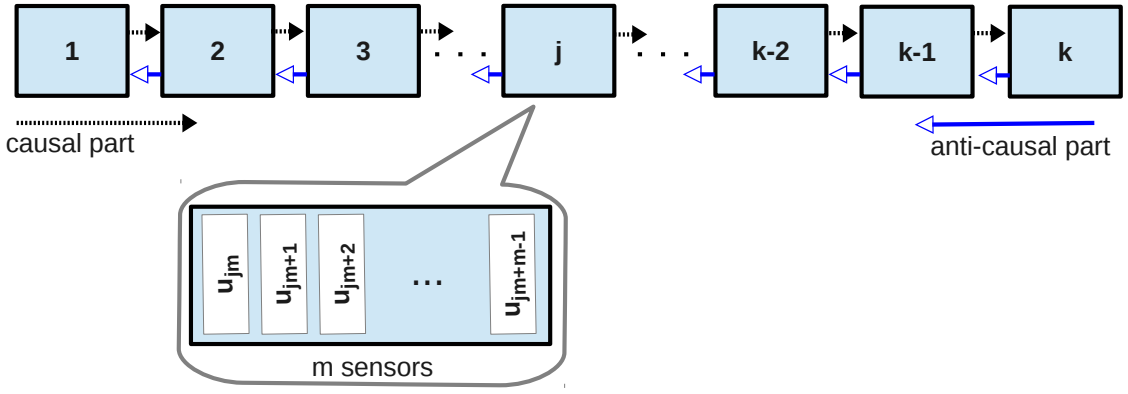


FIGURE 2.1 – A line network with k processors.

2.1.1.2/ ESTIMATION OF THE NUMBER OF OPERATIONS AND TRANSMISSIONS FOR THE DR METHOD WITH A LINE

We emphasize that we just consider the computation of the causal part if there are no other mentions. As mentioned the first loop of Algorithm 5 needs to be executed in parallel. This loop is rewritten as the following prefix sum :

$$\varphi_{\ell+1} = \varphi_\ell + B_{\ell+1}U_{\ell+1},$$

where the output vector is defined by $\varphi_{\ell+1} = (\varphi^+_{\ell+1,1}, \varphi^+_{\ell+1,2}, \ldots, \varphi^+_{\ell+1,M})$, the vector $B_{\ell+1} = (\alpha_1^{-\ell-1}\beta_1, \alpha_2^{-\ell-1}\beta_2, \ldots, \alpha_M^{-\ell-1}\beta_M)$ is computed offline and where $U_{\ell+1} = u(x_{\ell+1})$ is the input value.

We remark that in our computation the value of φ_ℓ and B_ℓ are complex-valued vectors. Thus the additions and the multiplications here are complex operations. To be convenient we count the number of operations and transmissions with real numbers using a below remark :

Remark 2 :

Assume that $z_1 = a + bi, z_2 = c + di \in \mathbb{C}$, we have

$$z_1 + z_2 = (a \oplus c) + (b \oplus d)i,$$
$$z_1 \times z_2 = (a \otimes c \oplus (-b) \otimes d) + (b \otimes c \oplus a \otimes d)i.$$

So a complex addition can be carried out using only two real additions, whereas a complex multiplication can be carried out using four real multiplications and two real additions.

To be efficient, we suggest proceed by using three steps : Initial cumulation, Communication and Ending cumulation.

The first step is to compute the local prefix sum on all processors. Each processor receives m inputs from its m sensors, then m local prefix sums are computed. Namely, at $(j+1)^{st}$ processor, m local prefix sums are computed by

$$\textbf{(local prefix)}_{(mj+p)} = \sum_{i=0}^{p-1} B_{mj+i} \times U_{mj+i}, \text{ with } p = 1, ..., m.$$

The second step is to communicate. Any processor except the first one receives the message from its *left neighbour*. In order to compute this step quickly, the processor adds this value to its last prefix sum. Its last prefix sum is now updated. This updated result is saved in its message to send to its *right neighbour*. The last local prefix sum of the first processor is considered as its last updated prefix sum. This process continues until the last processor receives and adds the message of its *left neighbour*. The computation of this step can be summarized by

$$\textbf{(last updated prefix)}_1 = \textbf{(last local prefix)}_1,$$
$$\textbf{(last updated prefix)}_j = \textbf{(last updated prefix)}_{j-1} + \textbf{(last local prefix)}_j,$$
$$\text{with } j = 2, 3, ..., k,$$

where **(last local prefix)**$_j$ = **(local prefix)**$_{mj}$. Then the third step is to add previous updated prefix sum to all local elements. Now m final prefix sums at $(j+1)^{st}$ processor are computed by

$$\varphi_{mj+p} = \textbf{(last updated prefix)}_j + \textbf{(local prefix)}_{(mj+p)}, \text{ with } p = 1, ..., m-1,$$

We note that φ_{mj+m} is **(last updated prefix)**$_{j+1}$ and it is computed in previous step. To complete our online computation part, we add more a new step called Estimation of the contour which performs the computation of z_ℓ^+.

1. Step 1 - Initial cumulation :

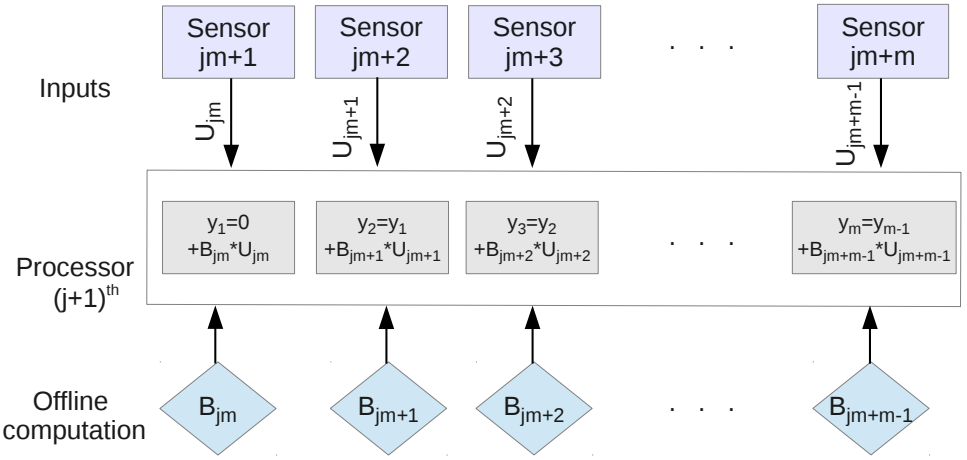


FIGURE 2.2 – Computation of step 1 at the $(j+1)^{st}$ processor.

Local prefix sums are computed on all processors. As mentioned previously, each processor needs to compute its local prefix sum with m inputs. This step is illustrated in Figure 2.2. Each computation requires a complex multiplication since we multiply the input U_{jm+i} by the vector B_{jm+i} (which contains M elements), with $i = 0, 1, ..., m-1$. It also requires a complex addition since we add results of the previous element to it. Therefore, this step involves $(m \times M)$ complex multiplications and $(m \times M)$ complex additions at each processor. Thanks to Remark 2, we have the total number of real operations for the causal part of this step

$$\mathcal{O}_{init\_cumul} = (4k \times m \times M) \otimes + (4k \times m \times M) \oplus, \tag{2.2}$$

where the symbols $\otimes, \oplus$ and $\textcircled{S}$ refer to a real multiplication, a real addition and a real-valued transmission, respectively.

2. Step 2 - Communication : The goal of this step is to communicate the local sum with the next processor if the processor is not the first one. It consists of receiving the last updated prefix sum of the previous processor and adding to its last local prefix sum. We note that this last updated prefix sum has M complex numbers. It requires M complex additions. If the processor is not the last one, the result of the last updated prefix sum is sent to the next processor. So a vector with M complex numbers is sent. We note that this step is executed by all the processors except

the first one, so there are $(k-1)$ communications. So $(k-1)\times M$ complex additions are used and $(k-1)\times M$ complex numbers are transmitted. Figure 2.3 illustrates this step. The $(j+1)^{st}$ processor receives the value **(last updated prefix)**$_j$ from its *left neighbour* and adds to its last local prefix sum. The result of this addition called **(last updated prefix)**$_{j+1}$ is used to send to its *right neighbour*. We note that the communication C_j is only executed if **(last updated prefix)**$_j$ is already computed. Thus the total number of operations and transmissions for the causal part of this step is given by

$$O_{comm} = 2(k-1)\times M \ \oplus + 2(k-1)\times M \ \text{Ⓢ}. \tag{2.3}$$

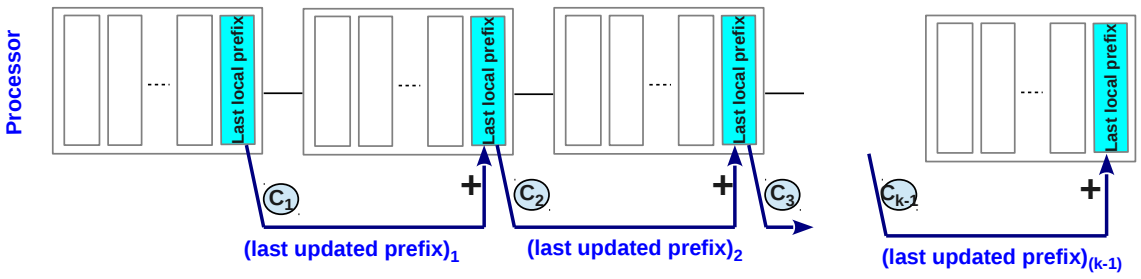


FIGURE 2.3 – Communication of the causal part between k processors.

3. Step 3 - Ending cumulation : The goal of this step is to complete all prefix sums in all processors. This step is executed by all the processors except the first one. It includes adding the last updated prefix sum of the previous processor to all local elements. Figure 2.4 illustrates this computation.

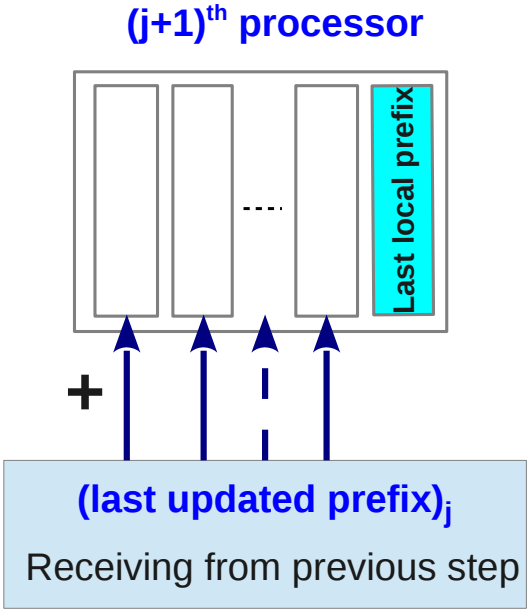


FIGURE 2.4 – Prefix sums completed on all elements.

We also note that this last updated prefix sum is received in the previous step. We only need to add it to $(m-1)$ local elements since the last local element was added in the previous step. So this step requires $(m-1)\times M$ complex additions at each processor except the first one. Thus the total number of operations for the causal part of this step is given by

$$O_{end\_cumul} = 2(k-1)\times(m-1)\times M \ \oplus . \tag{2.4}$$

Therefore the total number of operations and transmissions to compute prefix sums in parallel for the causal part is

$$\begin{aligned} O_{Cumul} &= O_{init\_cumul} + O_{comm} + O_{end\_cumul} \\ &= (4k\times m\times M)\ \otimes + (2(3k-1)\times m\times M)\ \oplus + (2(k-1)\times M)\ \text{Ⓢ}. \end{aligned} \tag{2.5}$$

4. Step 4 - Estimation of the contour : After the computation of prefix sums, the estimation of the contour which corresponds to the second loop of Algorithm 5 is executed. We note that no communication between the nodes is required in this step. For each processor, the computation of ψ^+ involves $(2m\times M)$ complex multiplications and $(m\times M)$ complex additions. Then the computation of z^+ involves a summation of real part of multiplications, so it involves $(2m\times M)$ real additions and $(2m\times M)$ real multiplications. Thus the number of operations for the causal part of this step is given by

$$O_{Estim\_contour} = (10k\times m\times M)\ \otimes + (8k\times m\times M)\ \oplus . \tag{2.6}$$

Therefore the total number of operations and transmissions for the causal part is

$$\begin{aligned} O_{DR\_causal} &= O_{Cumul} + O_{Estim\_contour} \\ &= (14k\times m\times M)\ \otimes + (2(7k-1)\times m\times M)\ \oplus + (2(k-1)\times M)\ \text{Ⓢ}. \end{aligned} \tag{2.7}$$

Basic step	Notation	Basic step	Notation
Step **1** of **c**ausal part	Step 1c	Step **1** of **a**nti-causal part	Step 1a
Step **2** of **c**ausal part	Step 2c	Step **2** of **a**nti-causal part	Step 2a
Step **3** of **c**ausal part	Step 3c	Step **3** of **a**nti-causal part	Step 3a
Step **4** of **c**ausal part	Step 4c	Step **4** of **a**nti-causal part	Step 4a

TABLE 2.1 – Some notations of some basic steps of causal and anti-causal part.

We remark that the anti-causal part is symmetric and that its communication starts from the last processor to end on the first. So a processor should receive the message from its *right neighbour*. The computation of both parts is similar. Thus here is the total number of operations and transmissions for the DR method

$$\begin{aligned} O_{DR} &= 2O_{DR\_causal} \\ &= (28k \times m \times M) \ \otimes + (4(7k-1) \times m \times M) \ \oplus + (4(k-1) \times M) \ \text{Ⓢ}. \end{aligned} \tag{2.8}$$

2.1.1.3/ ESTIMATION OF THE EXECUTION TIME OF THE DR METHOD WITH A LINE

To clarify the explanation, we give some notations as Table 2.1. We need some assumptions as follows :

- Each step of both parts is executed for all processors.
- The step $2c$ only starts on the second processor when its step $1c$ finishes, whereas the step $2a$ starts on the processor $(k-1)^{st}$ when its step $1a$ finishes.
- At each processor, the step jc (resp. ja) only starts after step $(j-1)c$ (resp. $(j-1)a$) finishes, with $j = 2, 3, 4$.

The computation of both causal and anti-causal parts uses the same network, so we should find the best effective way to arrange the order of steps of both parts such that the computation time is the least. In order to do that we choose a strategy to overlap some steps of both parts, namely

- The step $1c$ is executed at all processors of the first half of the network while the step $1a$ is executed at all processors of its second half ahead.
- Then the step $2c$ starts on the second processor, and the step $2a$ starts on the $(k-1)^{st}$ processor. We note that on each processor there is an idle time between above steps during which some other operations can be executed. Figure 2.5 illustrates our strategy, for example, the idle time between steps $1c$ and $2c$ or between $2c$ and $2a$ in the first half network can be seen. We have three idle times between these steps. We denote these idle times by I_1, I_2 and I_3 respectively as in Table 2.2.
- The step $1a$ of each processor in the first half of the network should be executed in the idle time I_1 or the idle time I_2. The step $3c$ and $4c$ of each processor in the first half of the network can be executed in the idle time I_2 or the idle time I_3 while the step $3a$ and $4a$ of these processors should be executed in the idle time I_3. It is similar for the steps of processors in the second half of the network.

Therefore, it is possible, in some cases, to overlap some steps of both parts. Before describing these different cases, we introduce some notations given in Table 2.3 and estimate the computation time of each step.

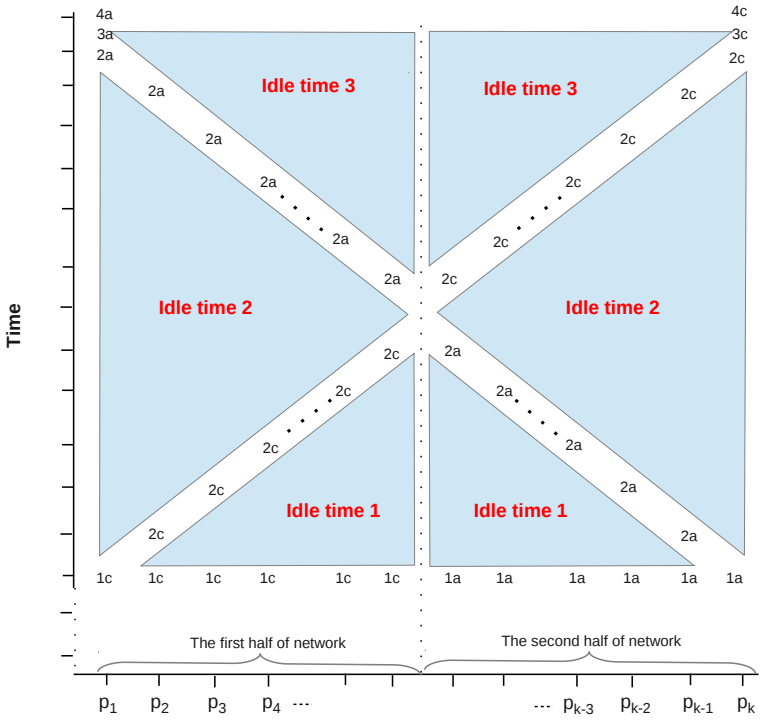


FIGURE 2.5 – The computation time scheme consisting of some basis steps and idle times.

Order	Basis step	Note
1	Initial cumulation (causal part)	$1c$
2	First idle time	I_1
3	Communication (causal part)	$2c$
4	Second idle time	I_2
5	Communication (anti-causal part)	$2a$
6	Third idle time	I_3

TABLE 2.2 – The order of some basic steps at each processor in the first half of the network.

	Note
The computation time of the j^{th} step, $j = 1, 2, 3, 4$	T_j
The computation time of the step jc (resp. ja), $j = 1, 2, 3, 4$	$T_j(c)$ (resp. $T_j(a)$)
The first idle time of the j^{th} processor	I_{1j}
The second idle time of the j^{th} processor	I_{2j}
The third idle time of the j^{th} processor	I_{3j}
The time taken a real multiplication	t_m
The time taken a real addition	t_a
The time taken a real-valued transmission (sent)	t_s

TABLE 2.3 – The notations concerning the computation time.

We have the following remark :

Remark 3 :

The computation time of each step is

$$\begin{aligned} T_1 &= 4m \times M \times t_m + 4m \times M \times t_a, \\ T_2 &= 2M \times t_a + 2M \times t_s, \\ T_3 &= 2(m-1) \times M \times t_a, \\ T_4 &= 10m \times M \times t_m + 8m \times M \times t_a. \end{aligned} \tag{2.9}$$

The overlapping now depends on the computation time of each step and the number of nodes in the network. Precisely, it depends on the ratio of the time taken by a multiplication t_m, an addition t_a and a transmission t_s. It also depends on the number of processors in network. We consider all possible cases where other steps can be added into each idle time :

Case 1 : If the communication time to cross a half of the network is not greater than the computation time T_1 of the step 1, i.e.,

$$(k/2-1) \times T_2 \leq T_1,$$

then there is no more overlap. In this case, data are awaited in the first idle time I_1 and the step $1a$ of the first half (resp. the step $1c$ of the second half) of the network must be executed in the second idle time I_2. Step 3 and 4 for both parts are executed in the third idle time I_3.
We note that the first processor and the last one receive data (in the step 2) finally, so the computation time at these ones is greater than others as the illustration in Figure 2.6. However, the computation time should be an upper bound of the time of all processors. It is easy to compute this estimation by counting at the first processor. Namely,

$$\begin{aligned} T_{no\_overlap} &= T_1(c) + T_1(a) + k/2 \times T_2(c) + k/2 \times T_2(a) \\ &+ T_3(c) + T_4(c) + T_3(a) + T_4(a). \end{aligned}$$

In other words, the necessary time of the DR method in this case is

$$\begin{aligned} T_{no\_overlap} &= 2T_1 + (k-1) \times T_2 + 2T_3 + 2T_4 = 28m \times M \times t_m \\ &+ 2(14m + k - 3) \times M \times t_a + 2(k-1) \times M \times t_s. \end{aligned} \tag{2.10}$$

The estimation time and the order of steps in this case are shown in Figure 2.6.

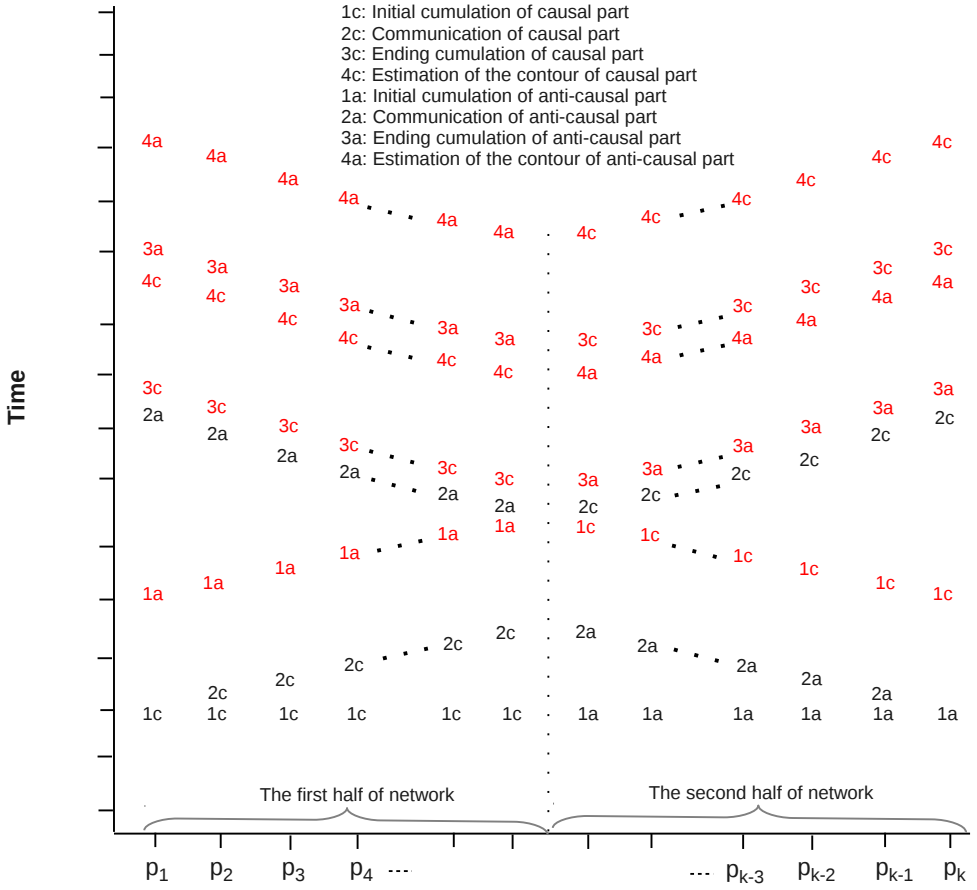


FIGURE 2.6 – The estimation time and the order of steps in Case 1 : no overlap. The red ones represent the new steps added into idle times.

We note that in this case there is no overlap, so we can execute both step $1a$ and $1c$ before starting the communication step 2.

Case 2 : Otherwise, if the communication time to cross the half of network is greater than the implementation time of the step 1, i.e.,

$$T_1 < k/2 \times T_2, \tag{2.11}$$

then the overlapping situation is possible, namely, the step $1a$ (resp. the step $1c$) of some processors in the first half (resp. the second half) of the network can be executed during the idle time I_1. We just consider the first half of the network sufficiently since another half is symmetric. The step $1a$ is able to execute in the first idle time I_1 from some processors of the first half of the network. So the first idle time and the second idle time at the j^{th} processor are given by

$$\begin{aligned} I_{1j} &= (j-2) \times T_2(c) = (j-2) \times T_2 \\ &\quad \text{if } 2 \le j \le k/2, \text{ and } I_{11} = 0, \end{aligned} \tag{2.12}$$

and

$$\begin{aligned} I_{2j} &= (k/2-j) \times T_2(c) + (k/2-j) \times T_2(a) \\ &= (k-2j) \times T_2, \ j \le k/2. \end{aligned} \tag{2.13}$$

We only firstly consider the case (2.12) and (2.13). The condition to overlap can be rewritten by

$$T_1 < I_{1j} \quad \text{or} \quad T_1 < I_{2j}, \text{ for all } j = 1, 2, ..., k/2.$$

It is equivalent to

$$\begin{aligned} & T_1 < \max\{I_{1j}, I_{2j}\}, \text{ for all } j = 1, 2, ..., k/2 \\ \Longleftrightarrow\ & T_1 < \min_{j\in\{1,2,...,k/2\}} \Big(\max\{I_{1j}, I_{2j}\}\Big). \end{aligned} \tag{2.14}$$

Since

$$\begin{aligned} \max\{I_{1j}, I_{2j}\} &= \max\{(j-2)\times T_2, (k-2j)\times T_2\} = \max\{j-2, k-2j\}\times T_2 \\ &= \begin{cases} (j-2)\times T_2 & \text{if} \quad j \geq \frac{k+2}{3} \\ (k-2j)\times T_2 & \text{if} \quad j \leq \frac{k+2}{3} \end{cases}, \end{aligned} \tag{2.15}$$

the inequality (2.14) is equivalent to

$$T_1 < \frac{k-4}{3} \times T_2. \tag{2.16}$$

In this case, the step $1a$ can be put in the first idle time I_1 from the $(\frac{k+2}{3})^{th}$ processor to the $\frac{k}{2}^{th}$ processor and in the second idle time I_2 from the first processor to the $(\frac{k+2}{3}-1)^{st}$ processor. This scheme is illustrated in Figure 2.7.

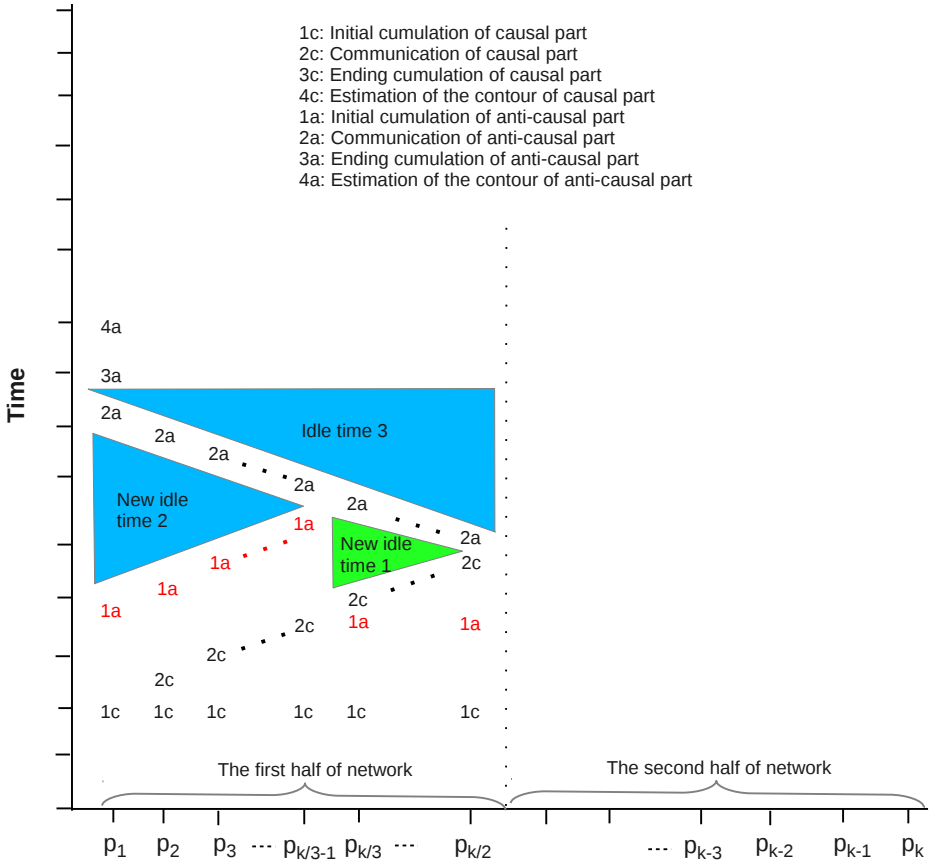


FIGURE 2.7 – The estimation time in Case 2 : overlap step 1. The red ones represent step $1a$ added into idle times.

With the condition (2.16), the step $1a$ of some processors can be executed during idle times. Now we have two new idle times as shown in Figure 2.7 in which the step $3c$, and $4c$ can be executed for some conditions considered later. If there is only the overlap of the step $1a$ then the total time to complete computations in this case is

$$\begin{aligned} T_{overlap\_step1} &= T_1 + (k-1) \times T_2 + 2 \times T_3 + 2 \times T_4 \\ &= 24m \times M \times t_m + 2(12m + k - 3) \times M \times t_a \\ &\quad +2(k-1) \times M \times t_s, \end{aligned} \tag{2.17}$$

since it is easy to compute the time by counting at the first processor, i.e.

$$\begin{aligned} T_{overlap\_step1} = T_1(c) + (k/2-1) \times T_2(c) + k/2 \times T_2(a) \\ +T_3(c) + T_4(c) + T_3(a) + T_4(a). \end{aligned}$$

An illustration of this case is shown in Figure 2.8.

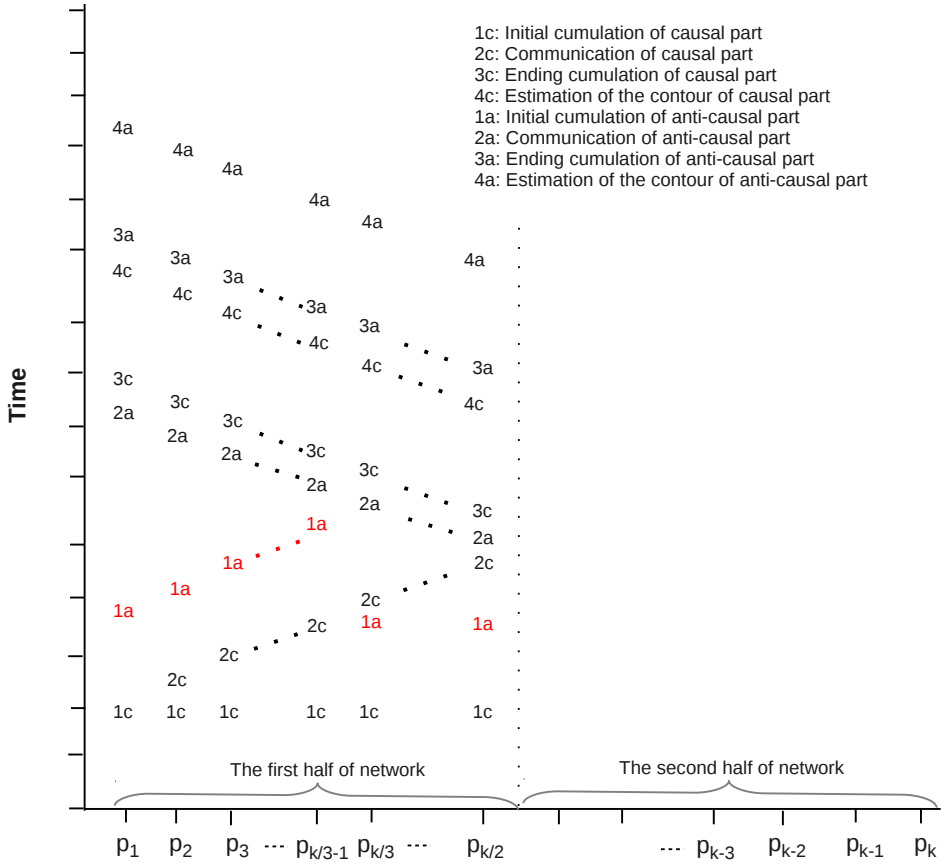


FIGURE 2.8 – The overlap only occurs for step $1a$ of the first half of network.

Case 3 : Now we consider conditions in order to overlap the step $3c$ of the first half of the network. From Figure 2.7 we can see that the step $3c$ of the first $(k/3-1)$ processors can be executed in the second new idle time or the third idle time I_3 while the step $3c$ of the last $k/6$ processors can be executed in the first new idle time or the third idle time I_3. We denote by I'_{2j} the second new idle time, and we denote by I_{3j} the third idle time of the j^{th} processor , $j \leq k/2$. We also give their formula

$$\begin{aligned} I'_{2j} = I_{2j} - T_1(a) = (k-2j) \times T_2 - T_1 \quad &\text{if} \quad j < k/3, \\ I'_{2j} = 0 \quad &\text{if} \quad k/3 \leq j \leq k/2, \\ I_{3j} = (j-1) \times T_2 \quad &\text{if} \quad j \leq k/2. \end{aligned} \tag{2.18}$$

In fact the idle time I_{3j} can be extended infinitely. Theoretically, I_{3j} can be infinity if other steps haven't started yet, but in order to overlap the step $3c$, the formula (2.18) should be satisfied. The condition in order to overlap the step $3c$ is for all processors if the time of the step 3 is less than the second new idle time I'_2 or the third idle time I_3 at each processor. This condition can be rewritten by

$$T_3 < I'_{2j} \quad \text{or} \quad T_3 < I_{3j}, \text{ for all } j = 1, 2, ..., k/2.$$

It is equivalent to

$$\begin{aligned} T_3 &< \max\{I'_{2j}, I_{3j}\}, \text{ for all } j = 1, 2, ..., k/2 \\ \Longleftrightarrow T_3 &< \min_{j\in\{1,2,...,k/2\}} \left(\max\{I'_{2j}, I_{3j}\}\right). \end{aligned} \tag{2.19}$$

Since

$$\begin{aligned} \max\{I'_{2j}, I_{3j}\} &= \max\{(k-2j)\times T_2 - T_1, (j-1)\times T_2\} \\ &= \begin{cases} (j-1)\times T_2 & \text{if} \quad j \geq \frac{k+1-T_1/T_2}{3} := k_0 \\ (k-2j)\times T_2 - T_1 & \text{if} \quad j \leq k_0 \end{cases}, \end{aligned}$$

the inequality (2.19) is equivalent to

$$T_3 < \frac{k-2}{3} \times T_2 - \frac{T_1}{3}. \tag{2.20}$$

In this case, the step $3c$ can be executed in the third idle time I_3 from the processor k_0 to $(k/2)$ and in the second new idle time I'_2 from the first processor to $(k_0 - 1)$. This case is illustrated in Figure 2.9.

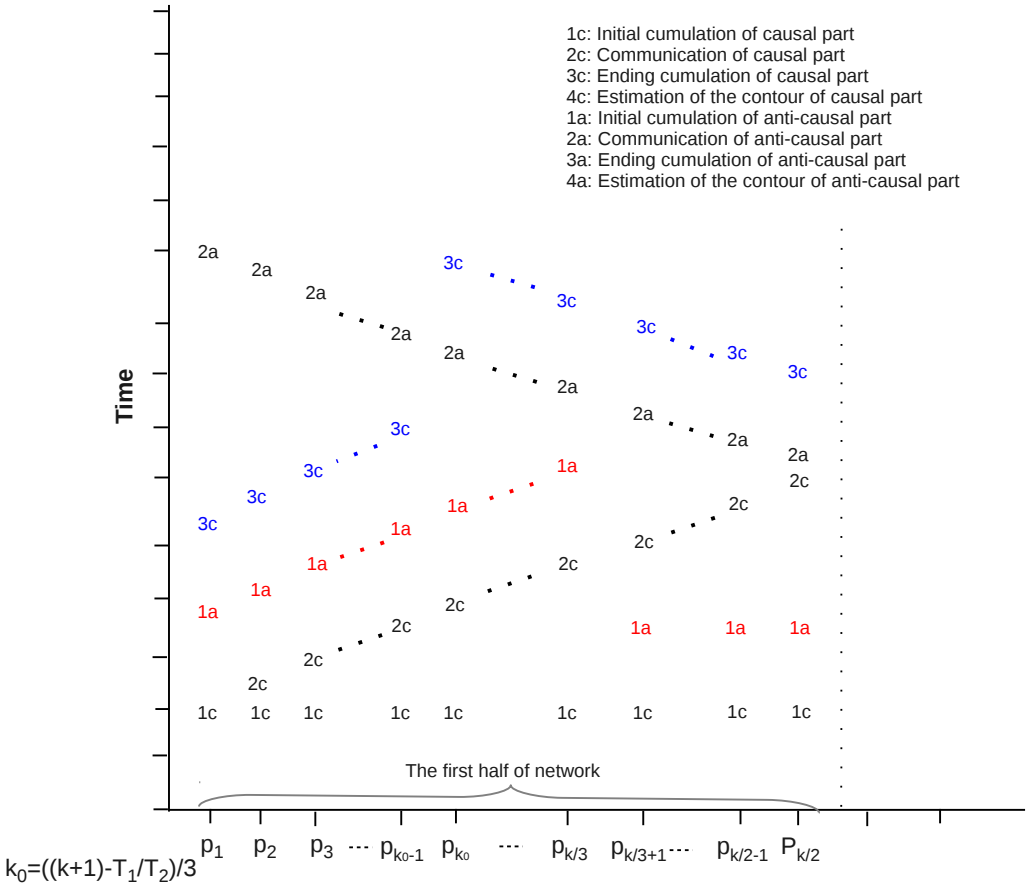


FIGURE 2.9 – The overlap of both step $1a$ and step $3c$.

So the total time in this case is

$$\begin{aligned} T_{overlap\_step3} &= T_1 + (k-1) \times T_2 + T_3 + 2 \times T_4 \\ &= 24m \times M \times t_m + 2(11m + k - 2) \times M \times t_a \\ &\quad +2(k-1) \times M \times t_s, \end{aligned} \tag{2.21}$$

since it is easy to see by counting at the first processor, i.e.,

$$\begin{aligned} T_{overlap\_step3} &= T_1(c) + (k/2 - 1) \times T_2(c) + k/2 \times T_2(a) \\ &\quad +T_3(a) + T_4(c) + T_4(a). \end{aligned}$$

We remark that the overlap of the step $4c$ seems to be impossible since its computation time gets a large proposion. Therefore, we ignore this case. Moreover, the steps $3a$ and $4a$ should be executed after the step $2a$, so there are no overlap of these steps.

Finally, our parallel computation is summarized as follows

1. Case 1 : No overlap if the condition holds

$$T_1 \geq \frac{k-4}{3} \times T_2,$$

then the total computation time is

$$\begin{aligned} T_{no\_overlap} = 28m \times M \times t_m + 2(14m + k - 3) \times M \times t_a \\ +2(k-1) \times M \times t_s. \end{aligned} \tag{2.22}$$

2. Case 2 : Only overlap of the step 1 if the conditons hold

$$T_1 < \frac{k-4}{3} \times T_2, \text{ and } T_3 \geq \frac{k-2}{3} \times T_2 - \frac{T_1}{3},$$

then the total computation time is

$$\begin{aligned} T_{overlap\_step1} &= T_1 + (k-1) \times T_2 + 2 \times T_3 + 2 \times T_4 \\ &= 24m \times M \times t_m + 2(12m + k - 3) \times M \times t_a \\ &\quad +2(k-1) \times M \times t_s. \end{aligned} \tag{2.23}$$

3. Case 3 : Overlap of both step 1 and step 3 if the conditons hold

$$T_1 < \frac{k-4}{3} \times T_2, \text{ and } T_3 < \frac{k-2}{3} \times T_2 - \frac{T_1}{3},$$

then the total computation time is

$$\begin{aligned} T_{overlap\_step3} &= T_1 + (k-1) \times T_2 + T_3 + 2T_4 \\ &= 24m \times M \times t_m + 2(11m + k - 2) \times M \times t_a \\ &\quad +2(k-1) \times M \times t_s. \end{aligned} \tag{2.24}$$

2.1.2/ DR ALGORITHM WITH A GENERAL HYPERCUBE TOPOLOGY

A general hypercube topology is also used to implement our algorithm in parallel. This topology executes a prefix sum algorithm in the first loop of Algorithm 5.

2.1.2.1/ DESCRIPTION OF A HYPERCUBE NETWORK

We consider that *the $d-$dimensional hypercube network* has $k = 2^d$ processors and $d \times 2^{d-1}$ links. Each processor corresponds to a $d-$bit binary string, and two processors are connected with a link if and only if their binary strings differ in precisely one bit. As a consequence, each processor is incident to $d = \log_2 k$ other processors, one for each bit position. For example, we have drawn the hypercube networks with 2, 4, 8 and 16 processors in Figure 2.10.

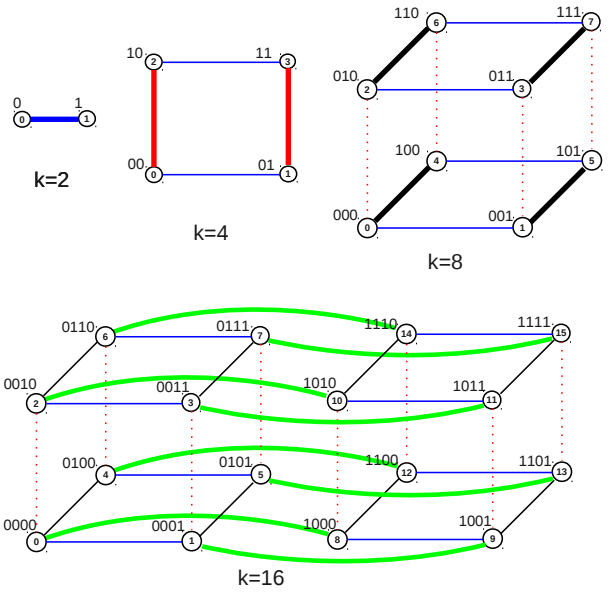


FIGURE 2.10 – The $k-$processor hypercube for $k = 2, 4, 8$ and 16. Two processors are connected with a link if and only if their strings differ in precisely one bit position. *Dimension 1 links* are shown in boldface.

The links of a hypercube network can be naturally partitioned according to the dimensions that they traverse. In particular, a link is called a *dimension j link* if it connects two processors that differ in the j^{th} bit position. We will call the neighbour of a processor across dimension j link to be its $(d - j + 1)^{st}$ neighbour. We also consider that each processor in our network controls $m \geq 1$ sensors.

Our network is designed for both causal and anti-causal parts. In order to use as less links as possible, only one link is used to connect each pair of processors. It means we only use $d \times 2^{d-1}$ links as a classical hypercube. Each processor maintains a *causal result buffer* and an *anti-causal result buffer*. It also requires a *causal message* and an *anti-causal message*.

2.1.2.2/ PREFIX SUM ALGORITHM FOR A HYPERCUBE

A prefix sum algorithm for a hypercube can be implemented very similarly to the all-to-all broadcast on a hypercube. It is outlined in Algorithm 6 (referenced from the page 168 in

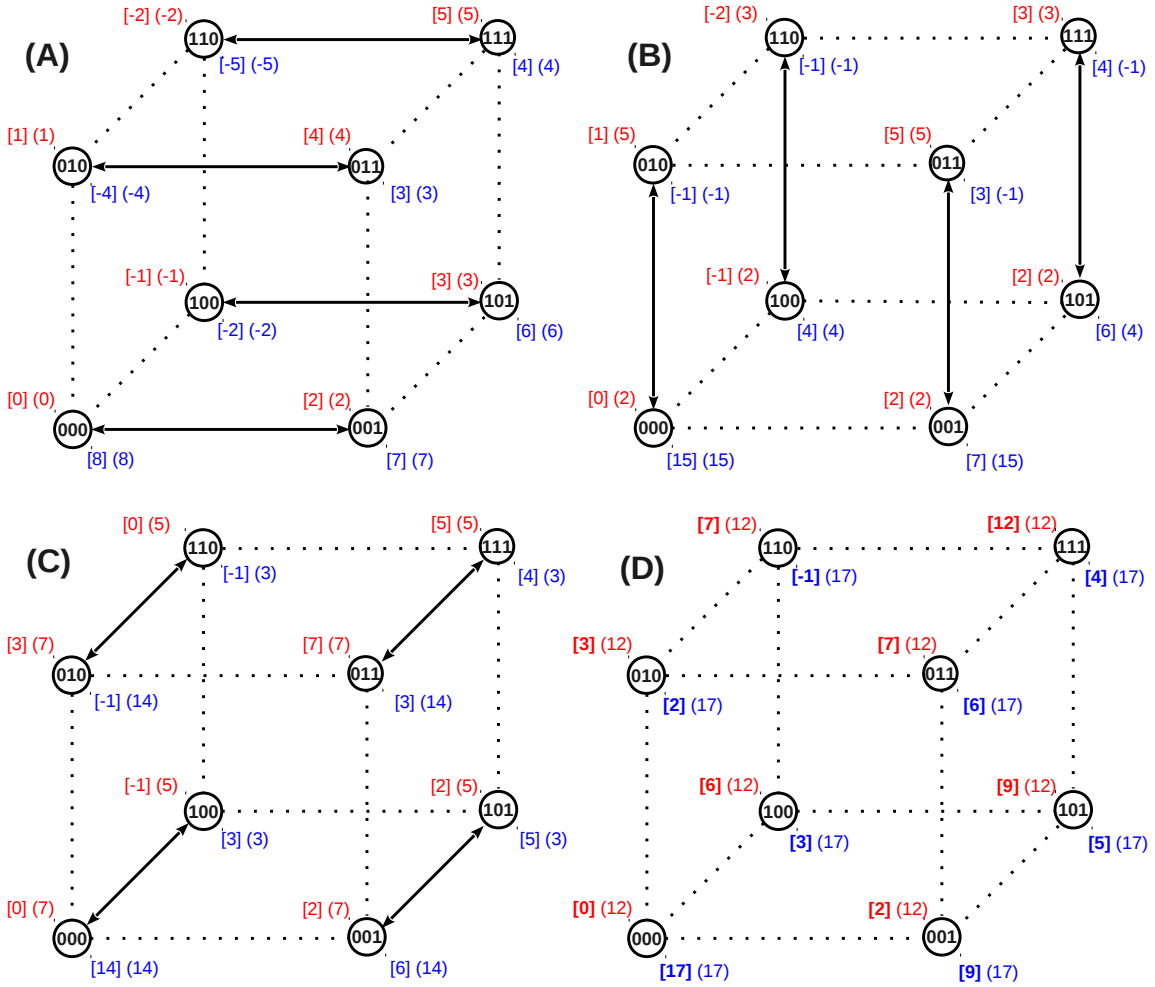
Ananth Grama [8]).

Require : An associative binary operator $\oplus$ and $k = 2^d$ processes.
Ensure : Computes the partial reductions $data_o \oplus ... \oplus data_h$ for $h = 0, 1, ..., k-1$, where k is the number of processors and $data_h$ denotes the data initially stored on $(h+1)^{th}$ processor, and stores the h^{th} partial on $(h+1)^{th}$ processor. The $(h+1)^{th}$ processor has the rank h.
Let me denote the rank (between 0 and $k-1$) of the calling processor.
$msg \leftarrow data_{me}$
$res \leftarrow data_{me}$
for $h = 0, 1, ...d-1$ **do**
  partner $\leftarrow me$ XOR 2^h
  Send msg to $partner$ and at the same time receive $data$ from $partner$
  $msg \leftarrow msg \oplus data$
  if $partner < me$ **then**
    $res \leftarrow res \oplus data$
  end
end
Return res.

Algorithme 6: Prefix sum on d-dimensional hypercube for the causal part.

It should be noticed that a hypercube prefix sum algorithm of the anti-causal part is performed by replacing $partner < me$ in the algorithm of the causal part into $partner > me$.

See Figure 2.11 for an illustration of the hypercube prefix sum algorithm for both causal and anti-causal parts on an eight-node hypercube (it is also a hypercube of dimension 3). The contents of the `msg` buffers are shown in $(.)$ and the contents of the `res` buffers are shown in $[.]$. The contents of causal (resp. anti-causal) result buffers are shown in upper-left (resp. lower-right) buffers. At the end of a communication step, the content of a causal (resp. anti-causal) incoming message is added to the causal (resp. anti-causal) result buffer only if the message comes from a processor with a smaller (resp. larger) rank than that of the recipient processor. The contents of the causal (resp. anti-causal) outgoing message (denoted by parentheses in the figure) are updated with every causal (resp. anti-causal) incoming message. For instance, after the first communication step, processors $0, 2$, and 4 do not add the data of the causal part received from processors $1, 3$, and 5 to their causal result buffers. However, the contents of the causal outgoing messages for the next step are updated.



Processor	1	2	3	4	5	6	7	8
Input of causal	0	2	1	4	-1	3	-2	5
Output of causal	**0**	**2**	**3**	**7**	**6**	**9**	**7**	**12**

Processor	1	2	3	4	5	6	7	8
Input of anti-causal	8	7	-4	3	-2	6	-5	4
Output of anti-causal	***17***	***9***	***2***	***6***	***3***	***5***	***-1***	***4***

FIGURE 2.11 – Computing prefix sums on an eight-node hypercube for both parts. At each node, square brackets show the local prefix sum accumulated in the result buffer and parentheses enclose the contents of the outgoing message buffer for the next step. Two small tables illustrate the input-output for both parts in detail.

2.1.2.3/ ESTIMATION OF THE NUMBER OF OPERATIONS AND TRANSMISSIONS

For simplicity we only consider the causal case. The anti-causal case is treated in subsequent remarks. We suggest to proceed in four steps to implement the algorithm for the causal part. Four similar steps for the anti-causal part are presented in subsequent remarks.

1. Step 1 - Initial cumulation : This step is similar as Step 1 of the line topology. It

means the local prefix sums are computed on all processors. So the number of operations of this step for the causal part is given by

$$O_{init\_comp\_causal} = (4m \times M \times 2^d) \otimes +(4m \times M \times 2^d) \ \oplus . \tag{2.25}$$

The causal outgoing message of each processor saves its *last local prefix sum*. For example, if the inputs for the prefix sum computation of a processor are $1, 2, -2, 3, 4$ then the local prefix sums of the causal part are $1, 3, 1, 4, 8$ respectively. The causal outgoing message thus saves 8 since it is the last local prefix sum.

Remark 4 :

The computation of the anti-causal part is similar as the causal part. Since we use the same links for both parts, this computation can be executed right after the computation of the causal part. However, the local prefix sums of the anti-causal part are the results of an inverse progress of the causal part. So the anti-causal outgoing message of each processor saves its *first local prefix sum*. For example, if the inputs for the prefix sum computation of a processor are $1, 2, -2, 3, 4$ then the local prefix sums of the anti-causal part are $8, 7, 5, 7, 4$ respectively. The anti-causal outgoing message saves 8 since it is the first local prefix sum.
Therefore, the number of operations of this step for the anti-causal part is given by

$$O_{init\_comp\_anti} = O_{init\_comp\_causal}.$$

2. Step 2 - Communication : The goal of this step is to communicate and it is divided by d small steps named the sub-step $2.j$ (with $j = 1, ..., d$). At each sub-step $2.j$, all pairs of processors connected by a *dimension* $(d - j + 1)$ *link* communicate together. Each processor receives M complex-numbers, namely the contents of the causal incoming message, from its j^{th} neighbour. In order to compute this step quickly, at the end of this communication step, the contents of the causal outgoing message are updated with the causal incoming message. At the same time, the content of this causal incoming message is added to the causal result buffer only if the rank of the processor is larger than that of its neighbour. So there is always a half of processors executing additions for their causal result messages. The number of operations and transmissions for these processors is thus M complex-numbers received and $2M$ complex-additions (add to its causal outgoing message and its result buffer), whereas for other processors there is only M complex-numbers received and M complex-additions.

   So the number of operations and transmissions for the causal part at each sub-step is

$$\begin{aligned} O_{comm\_subs\_causal} &= (4M \times 2^{d-1}) \ \oplus +(2M \times 2^{d-1}) \ \text{Ⓢ} + (2M \times 2^{d-1}) \ \oplus \\ &+ (2M \times 2^{d-1}) \ \text{Ⓢ} = 3M \times 2^d \ \oplus +2M \times 2^d \ \text{Ⓢ}. \end{aligned} \tag{2.26}$$

   There are d sub-steps as above, so the number of operations and transmissions of this step for the causal part is given by

$$O_{comm\_causal} = (3M \times d \times 2^d) \ \oplus +(2M \times d \times 2^d) \ \text{Ⓢ}. \tag{2.27}$$

Remark 5 :

To save the total computation time, communications of the anti-causal part are executed right after that of the causal part finishes. Its computation is executed similarly as the computation of the causal part. The only different point is that the content of the anti-causal incoming message is added to the anti-causal result buffer only if the rank of the processor is smaller than that of its neighbour. So the number of operations and transmissions of this step for the anti-causal part is given by

$$O_{comm\_anti} = O_{comm\_causal}.$$

We consider an example as in Figure 2.11. Let us consider the communication of the causal part between the 3^{rd}-processor with the initial value $\texttt{res}_3^c = 1, \texttt{mes}_3^c = 1$ and the 4^{th}-processor with the initial value $\texttt{res}_4^c = 4, \texttt{mes}_4^c = 4$, see Figure 2.11(A). Recalling that in this case the rank of the 3^{rd}-processor is 2 which corresponds to $3-$bit binary string 010 and the rank of the 4^{th}-processor is 3 which corresponds to $3-$bit binary string 011. So the 3^{rd}-processor is the first neighbour of the 4^{th}-processor, and thus this communication is implemented in the sub-step 2.1.

The 3^{rd}-processor receives M complex-numbers (in this example, $M = 1$, the imaginary part is equal to 0) as the content of its causal incoming message from the 4^{th}-processor and add this value into only its message since its rank (equal to 2) is less than that of its neighbour (equal to 3). Its causal outgoing message is now updated, namely $\texttt{mes}_3^c = 5$.

The 4^{th}-processor receives the value 1, the old content of the causal outgoing message of the 3^{rd}-processor, as the content of its causal incoming message and add this value into both its causal outgoing message and its result buffer since its rank is larger. Now its causal outgoing message is updated, namely $\texttt{mes}_4^c = 5$. The causal result buffer is also updated, namely $\texttt{res}_4^c = 5$. Other processors and other sub-steps communicate and compute similarly. The results of the sub-step 2.2, 2.3 and 2.4 are shown in Figure 2.11(B), 2.11(C) and 2.11(D), respectively.

Remark 6 :

With the same example for the anti-causal part, considering the same communication, the 3^{rd}-processor with the initial value $\texttt{res}_3^a = -4, \texttt{mes}_3^a = -4$ receives the value 3 from the 4^{th}-processor with the initial value $\texttt{res}_4^a = 3, \texttt{mes}_4^a = 3$. This value is added to both its anti-causal outgoing message and its anti-causal result buffer. Its anti-causal outgoing message and its anti-causal result buffer are now updated, namely $\texttt{mes}_3^a = -1$, and $\texttt{res}_3^a = -1$.

The 4^{th}-processor receives the value -4, the old content of the anti-causal outgoing message of the 3^{rd}-processor, as the content of its anti-causal incoming message. This value is then added to its anti-causal outgoing message. This message is thus updated, namely $\texttt{mes}_4^a = -1$.

3. Step 3 - Ending cumulation : This step is executed by all processors except the first processor since the first processor already executed its computation in Step 1. It consists of adding the *previous prefix sum* to all the local elements.

   It means the j^{th} processor requires the last prefix sum of the $(j-1)^{st}$ processor. However, this previous prefix sum is not available in the j^{th} processor. It should be sent from the $(j-1)^{st}$ processor or it should be computed at the j^{th} processor. We

note that this prefix sum is the result of a subtraction between the context of the causal result buffer as a minuend and the last initial input as a subtrahend.
For example, seeing Figure 2.11(D), the processor 111 requires the last prefix sum of the processor 110, i.e., the value 7. This value is the result of subtracting 5 (the initial input of the processor 111, see Figure 2.11(A)) from 12 (the context of the causal result buffer of the processor 111).
We avoid adding more than one communication step by computing this previous prefix sum at the j^{th} processor. Namely, we use an addition storage to save its last initial input which will be used as above subtrahend. This storage is updated by substituting its old value by the result of subtracting "its old value" from "the context of the causal result buffer of the j^{th} processor". This result is used as the *previous prefix sum* in this step. In this case, we should add k subtractions (i.e., additions) in the number of operations of this step.
It thus requires $[(m-1)\times M+1]$ complex-additions at each processor except the first processor. So the number of operations of this step for the causal part is given by

$$O_{end\_cumul\_causal} = 2(m-1)\times M\times(2^d-1)+2(2^d-1)\ \oplus. \tag{2.28}$$

So the total number of operations and transmissions to implement the prefix sums for the causal part is

$$\begin{aligned} O_{cumul\_causal} &= O_{init\_comp\_causal}+O_{comm\_causal}+O_{end\_cumul\_causal} \\ &= (4m\times M\times 2^d)\ \otimes + M\times(6m\times 2^d+3d\times 2^d-2^{d+1}-2m+2) \\ &\quad +2(2^d-1)\ \oplus +(2M\times d\times 2^d)\ \text{Ⓢ}. \end{aligned} \tag{2.29}$$

Remark 7 :

Similarly to the anti-causal part, this step is executed by all processors except the last processor. But it adds the *last prefix sum* to all local elements. By similar deductions, the number of operations of this step for the anti-causal part is

$$O_{end\_cumul\_anti} = O_{end\_cumul\_causal},$$

and the total number of operations and transmissions to implement the prefix sums for the anti-causal part

$$O_{cumul\_anti} = O_{cumul\_causal}.$$

4. Step 4 - Estimation of the contour : This step is similar as the step 4 of the line topology. So the number of operations and transmissions for the causal part is

$$O_{estim\_contour\_causal} = (10m\times M\times 2^d)\ \otimes +(8m\times M\times 2^d)\ \oplus. \tag{2.30}$$

Remark 8 :

This step of the anti-causal part is also computed similarly to the one of the causal part. Therefore, the number of operations and transmissions for the anti-causal part is

$$O_{estim\_contour\_anti} = O_{estim\_contour\_causal}.$$

We see that the number of operations and transmissions of all steps of the anti-causal part is equal to that of the causal part. Since the total operations and transmissions for the causal part are

$$\begin{aligned} O_{DR\_causal} &= O_{cumul\_causal} + O_{estim\_contour\_causal} \\ &= 14m \times M \times 2^d \otimes + M \times (14m \times 2^d + 3d \times 2^d \\ &\quad -2^{d+1} - 2m + 2) + 2(2^d - 1) \ \oplus + (2M \times d \times 2^d) \ \text{Ⓢ}, \end{aligned} \tag{2.31}$$

and the total number of operations and transmissions for both parts is

$$\begin{aligned} O_{DR} &= 2 \times O_{DR\_causal} = 28m \times M \times 2^d \otimes \\ &\quad + 2M \times (14m \times 2^d + 3d \times 2^d - 2^{d+1} - 2m + 2) + 2(2^d - 1) \ \oplus \\ &\quad + (4M \times d \times 2^d) \ \text{Ⓢ}. \end{aligned} \tag{2.32}$$

2.1.2.4/ ESTIMATION OF THE EXECUTION TIME OF THE DR METHOD WITH A HYPERCUBE

We only consider the causal case. The anti-causal case is similar. In Step 1, there are 2^d processors to compute parallelly. Thus the computation time of this step is

$$T_{init\_comp\_causal} = 4m \times M \times t_m + 4m \times M \times t_a. \tag{2.33}$$

At every sub-step $2.j$, a half of processors executes $2M$ complex-additions (M complex additions for the message, M complex-additions for the result buffer) and another half of processors only executes M complex-additions for the message. However, we have to take the upper bound of computation time. So

$$T_{small\_step\_causal} = 4M \times t_2 + 2M \times t_s. \tag{2.34}$$

In Step 3, all processors expect the first processor compute parallelly, so

$$T_{end\_cumul\_causal} = 2(m-1) \times M \times t_a + 2t_a. \tag{2.35}$$

To implement the estimation of the contour, all the processors should implement the computation parallelly. So total time of this computation in parallel is

$$T_{estim\_contour\_causal} = 10m \times M \times t_m + 8m \times M \times t_a. \tag{2.36}$$

We use the same network for both parts, so the computation time of DR is twice as much as the computation time of the causal part, namely,

$$\begin{aligned} T &= 2 \times T_{init\_comp\_causal} + 2d \times T_{small\_step\_causal} + 2 \times T_{end\_cumul\_causal} \\ &\quad + 2 \times T_{estim\_contour\_causal} = 28m \times M \times t_m \\ &\quad + 2M \times (14m + 4d - 2) \times t_a + 2t_a + 4M \times d \times t_s. \end{aligned} \tag{2.37}$$

2.1.3/ DR ALGORITHM WITH A BINARY TREE TOPOLOGY

2.1.3.1/ DESCRIPTION OF A BINARY TREE TOPOLOGY

We consider a binary tree network which contains $k = 2^d$ processors and each of them is able to control $m \geq 1$ sensors in general case. The $(2^i \times j + 2^{i-1})^{th}$ processor communicates

with i other processors, $i \leq d-1, j \leq 2^{d-i}-1$. The k^{th} processor communicates with d other processors. Therefore, there are $k-1$ links in this network. See Figure 2.12 for an illustration of a binary tree network with $k = 2^4$ processors.

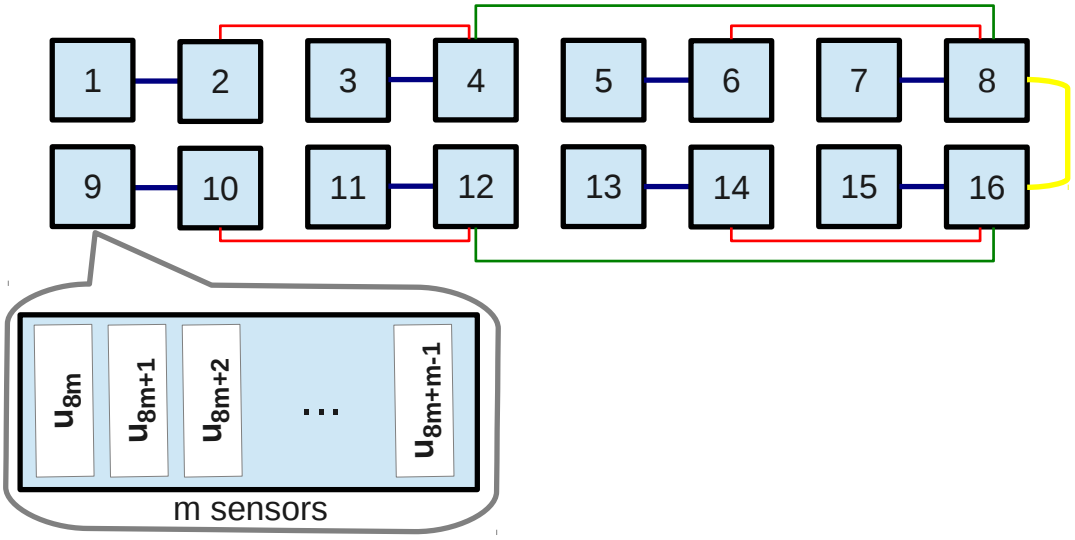


FIGURE 2.12 – A binary tree network which is suitable for the causal part with 16 processors.

This network is suitable to compute prefix sums for the causal part. For the anti-causal part, we have two choices. The first choice is to use another network which is symmetric with the network of the causal part. Figure 2.13 illustrates for this choice. In this case,

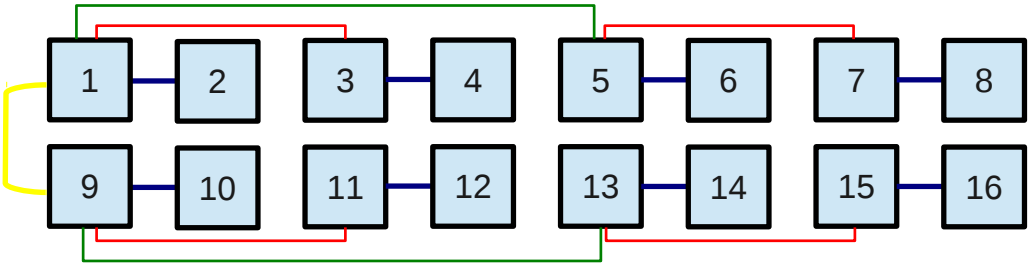


FIGURE 2.13 – A binary tree network which is suitable for the anti-causal part with 16 processors.

the parallel algorithm of the anti-causal part can be implemented similarly to the one of the causal part. Therefore, a common network for both parts is a combination of both networks as in Figure 2.12 and Figure 2.13.
The second choice is to use the same network as in the causal part. In this case, we should build a new algorithm for the anti-causal part.

The advantages of the first choice are a similar algorithm used for both parts and its capacity for saving computation time. Namely, both parts are implemented simultaneously. However, its biggest drawback is to use more $(k-1)$ links. This implies that its cost in-

creases. To avoid this drawback, we use the second choice. This is more realistic in application. In Subsection 2.1.3.2, we describe algorithms for both parts with the second choice. However, to have a comprehensive view, we also describe the first choice through subsequent remarks.

2.1.3.2/ ESTIMATION OF THE NUMBER OF OPERATIONS AND TRANSMISSIONS

For the causal part

] We suggest to proceed in 3 steps to implement the prefix sum and one step to estimate the contour. In order to understand easily, we give an example throughout a series of figures from Figure 2.12-A to Figure 2.12-L. This example is considered with 16 processors, and each of them controls three sensors by using three result buffers. Its initial data for the prefix sum computation are illustrated in Figure 2.12-A.

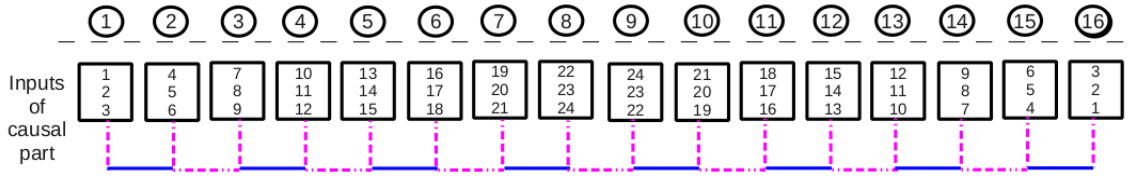


Figure 2.12-A : Input data illustrated for the causal part.

1. Step 1 - Initial cumulation : The local prefix sums are computed on all processors. This step is illustrated in Figure 2.12-B. This step involves $m \times M$ complex multiplications and $m \times M$ complex additions at each processor. So the number of operations of this step is given by

$$O_{init\_comp\_causal} = (4m \times M \times 2^d) \otimes +(4m \times M \times 2^d) \oplus . \tag{2.38}$$

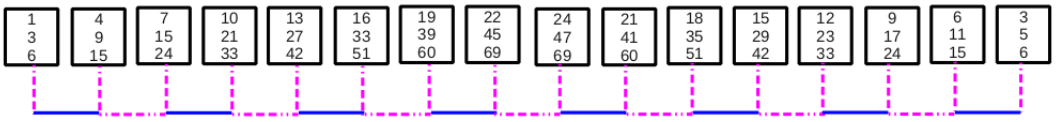


Figure 2.12-B : The result of step 1 illustrated for the causal part.

2. Step 2 - Communication : The goal of this step is to communicate. This step is divided into two steps since it requires both up-sweep and down-sweep to complete the prefix sum as in the first loop of Algorithm 5. We denote by Step $2a$ Up-sweep computation and denote by Step $2b$ Down-sweep computation.

   2a. **Step 2a - Up-sweep :** In this step all leaves send their results to their parents. It is divided into d sub-steps. They are illustrated from Figure 2.12-C to Figure 2.12-F.

   At sub-step 2a-i^{th} $(1 \leq i \leq d)$, the $(2^i j)^{th}$ processor receives and adds the M complex values from the $(2^{i-1}(2j-1))^{th}$ processor $(1 \leq j \leq 2^{d-i})$ and other processors have no computation. So at this each sub-step, there are $M \times 2^{d-i}$ complex numbers sent and $M \times 2^{d-i}$ complex numbers added. The number of operations and transmission of this each sub-step is summarized in Table 2.4.

Sub-step	Ⓢ (real number)	⊕ (real number)
2a-1	$2M \times 2^{d-1}$	$2M \times 2^{d-1}$
2a-2	$2M \times 2^{d-2}$	$2M \times 2^{d-2}$
...	...	...
2a-i	$2M \times 2^{d-i}$	$2M \times 2^{d-i}$
...	...	...
2a-d	$2M \times 2^{0}$	$2M \times 2^{0}$
Total	$2M \times (2^{d}-1)$	$2M \times (2^{d}-1)$

TABLE 2.4 – The number of operations and transmission of each sub-step.

Thus the number of operations and transmissions of this step is given by

$$O_{comm\_up\_causal} = 2M \times (2^d - 1)\ \text{Ⓢ} + 2M \times (2^d - 1) \oplus . \qquad (2.39)$$
$$(\text{ since } 2^d - 1 = 2^{d-1} + 2^{d-2} + ... + 2^0)$$

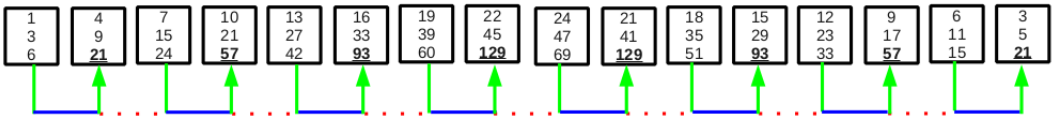


Figure 2.12-C : The result of step 2a-1 illustrated for the causal part.

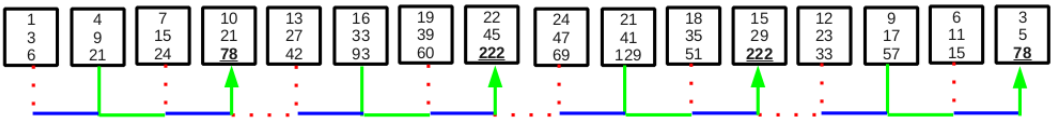


Figure 2.12-D : The result of step 2a-2 illustrated for the causal part.

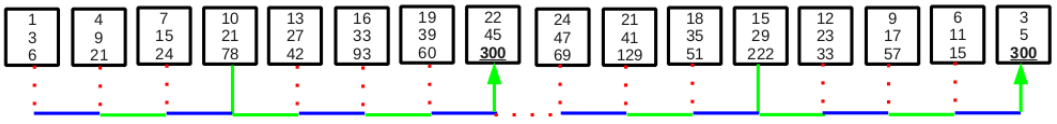


Figure 2.12-E : The result of step 2a-3 illustrated for the causal part.

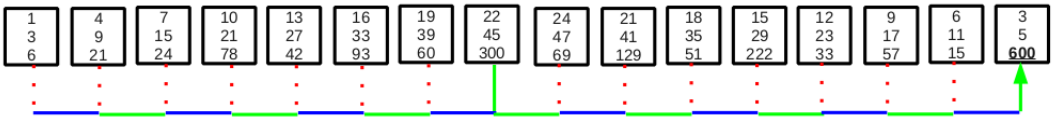


Figure 2.12-F : The result of step 2a-4 illustrated for the causal part.

2b. **Step 2b - Down-sweep** : This step should be executed since the following remark.

Remark 9 :

The second loop in Algorithm 5 shows that the computation of the history function ψ^+ at the $(\ell+1)^{st}$ element requires the prefix sum φ^+ of the previous element. Therefore, this prefix sum should be sent quickly or be implemented at the precessor which contains this element.

In a binary tree network, this step (Down-sweep) fulfills Remark 9, i.e., the ℓ^{th} prefix sum is implemented right on the processor containing the $(\ell+1)^{st}$ element. Before beginning this step, we replace the last prefix sum of last processor by 0. It only takes a replacement, so we don't count it. The result of this replacement is illustrated in Figure 2.12-G.

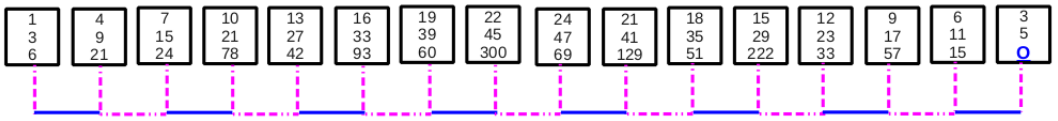


Figure 2.12-G : The replacement of the last prefix sum of the last processor.

This step is also divided by d sub-steps. They are also illustrated from Figure 2.12-H to Figure 2.12-K.

At the sub-step 2b-i^{th} $(1 \leq i \leq d)$, the $(2^{d-i+1}j)^{th}$ processor receives and adds the M complex numbers from the $(2^{d-i}(2j-1))^{th}$ processor $(1 \leq j \leq 2^{i-1})$, while the $(2^{d-i}(2j-1))^{th}$ processor receives the M complex values from the $(2^{d-i+1}j)^{th}$ processor and replaces its old values. Other processors have no computation. We count a replacement as an addition. So at this each sub-step, there are $2M\times 2^{i-1}$ complex numbers sent and $2M\times 2^{i-1}$ complex numbers added. Therefore the number of operations and transmissions of this step is given by

$$O_{comm\_down\_causal} = 4M \times (2^d - 1)\,\text{Ⓢ} + 4M \times (2^d - 1)\,\oplus\,. \qquad (2.40)$$
$$(\text{ since } 2^d - 1 = 2^0 + 2^1 + ... + 2^{d-1})$$

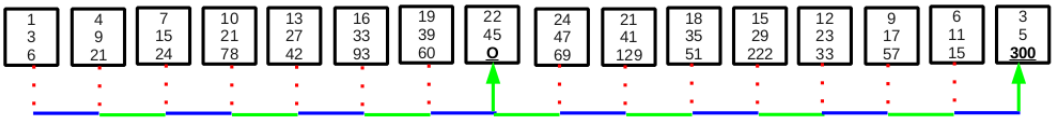


Figure 2.12-H : The result of Step 2b-1 illustrated for the causal part.

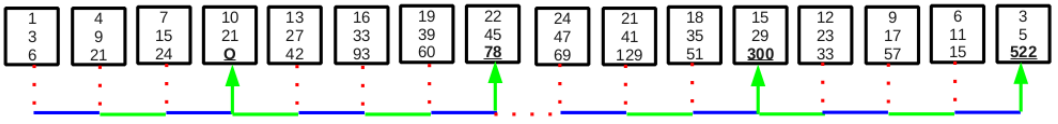


Figure 2.12-I : The result of Step 2b-2 illustrated for the causal part.

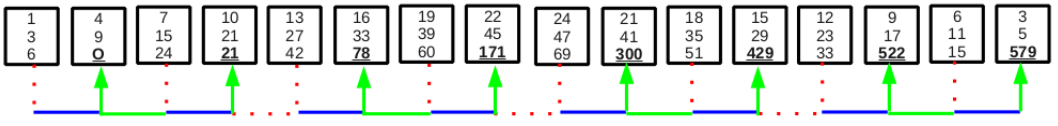


Figure 2.12-J : The result of Step 2b-3 illustrated for the causal part.

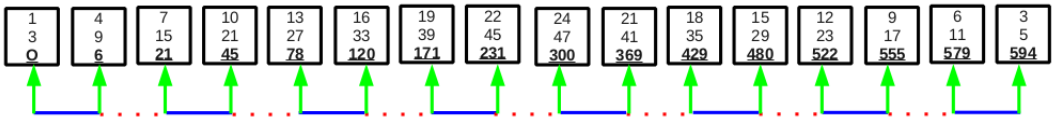


Figure 2.12-K : The result of Step 2b-4 illustrated for the causal part.

3. Step 3 - Ending cumulation : This last step is executed by all processors. It consists of adding the last prefix sum to all the local elements, then putting it in first location. Here an opration which puts the last prefix sum in the first location is considered as an addition. So the number of operations of this step is given by

$$O_{end\_cumul\_causal} = 2m \times M \times 2^d \ \oplus . \tag{2.41}$$

Figure 2.12-L : The result of Step 4 illustrated for the causal part.

The results of this step of the example are illustrated in Figure 2.12-L.

For the anti-causal part, it is symmetric and starts for the last processor to end on the first.

So the total number of operations and transmissions to compute the prefix sums in parallel for the causal part is

$$\begin{aligned} O_{cumul\_causal} &= O_{init\_comp\_causal} + O_{comm\_up\_causal} \\ &\quad + O_{comm\_down\_causal} + O_{end\_cumul\_causal} \\ &= (4m \times M \times 2^d) \ \otimes + \left(6m \times M \times 2^d + 6M \times 2^d - 6M\right) \ \oplus \\ &\quad + (6M \times (2^d - 1)) \ \text{Ⓢ}. \end{aligned} \tag{2.42}$$

4. Step 4 - Estimation of the contour : After the computation of the prefix sums, the estimation of the contour can be executed. It corresponds to the second line of the second loop in Algorithm 5. At each processor, the computation of ψ^+ involves $(2m \times M)$ complex multiplications and $(m \times M)$ complex additions. Then the computation of z^+ involves a summation of real part multiplications, so it is real part of $(m \times M)$ complex additions and complex multiplications. So the number of operations of this step is given by

$$O_{estim\_contour\_causal} = (10m \times M \times 2^d) \ \otimes + (8m \times M \times 2^d) \ \oplus . \tag{2.43}$$

Therefore, the total number of operations and transmissions for the causal part is

$$\begin{aligned} O_{BT\_causal} &= O_{cumul\_causal} + O_{estim\_contour\_causal} \\ &= (14m \times M \times 2^d) \ \otimes + \left(14m \times M \times 2^d + 6M \times 2^d - 6M\right) \ \oplus \\ &\quad + (6M \times (2^d - 1)) \ \text{Ⓢ}. \end{aligned} \tag{2.44}$$

Remark 10 :

With the first choice, the causal part is implemented as above. The anti-causal part is symmetric, so all computation of the anti-causal part is similar as that of the causal part, but it is executed from the last processor to the end at the first processor. Therefore the total number of operations and transmissions for the anti-causal part is

$$\begin{aligned} O_{BT\_anti} &= (14m \times M \times 2^d) \ \otimes + \left(14m \times M \times 2^d + 6M \times 2^d - 6M\right) \ \oplus \\ &\quad + (6M \times (2^d - 1)) \ \text{Ⓢ}. \end{aligned}$$

For the anti-causal part

We also proceed in three steps to implement the prefix sum and one step to estimate the contour. Some steps of this algorithm are illustrated throughout a series of figures from Figure 2.14-A to Figure 2.14-L with the initial inputs for the prefix sum computation as in Figure 2.14.

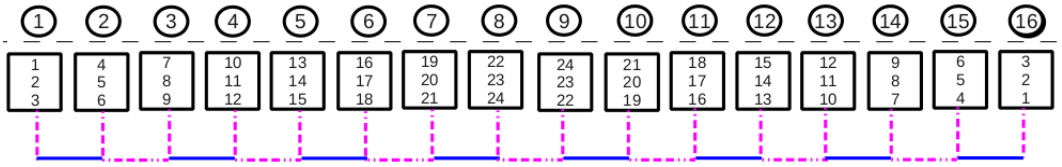


FIGURE 2.14 – Input data illustrated for anti-causal part

1. Step 1-Initial cumulation : This step is similar as in causal part. It means the local prefix sums are computed. However, the local prefix sum starts from the last element to the first element. The content of the anti-causal outgoing message is the first local prefix sum. This step is illustrated in Figure 2.14-A. The number of operations is given by

$$O_{init\_comp\_anti} = (4m \times M \times 2^d) \otimes + (4m \times M \times 2^d) \oplus . \tag{2.45}$$

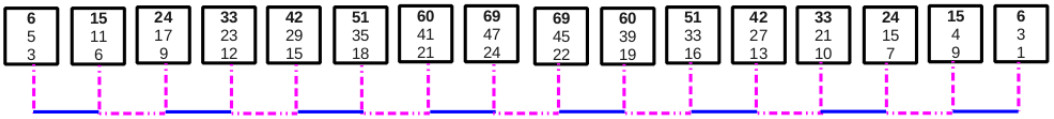


Figure 2.14-A : The result of Step 1 illustrated for the anti-causal part.

2. Step 2-Communication : This step is divided into two steps since it requires both up-sweep and down-sweep to complete the prefix sum.

   2a. **Step 2a - Up-sweep** : In this step all leaves send their results to their parents. It is divided into d sub-steps. At sub-step $2a - i^{th} (1 \leq i \leq d)$, the $(2^i j)^{th}$ processor receives and adds the M complex values from the $(2^{i-1}(2j-1))^{th}$ processor $(1 \leq j \leq 2^{d-i})$. At this step, a difference point of anti-causal part here is that the $(2^{i-1}(2j-1))^{th}$ processor also receives M complex values from the $(2^i j)^{th}$ processor and replaces its local prefix sum by these M complex values. Other processors have no computation. So at this each sub-step, there are $2M \times 2^{d-i}$ complex numbers sent and $2M \times 2^{d-i}$ complex numbers added. These sub-steps are illustrated from Figure 2.14-B to Figure 2.14-E.

   We consider a replacement is an addition. Thus the number of operations and transmissions of this step for the anti-causal part is given by

$$O_{comm\_up\_anti} = 4M \times (2^d - 1) \text{Ⓢ} + 4M \times (2^d - 1) \oplus . \tag{2.46}$$

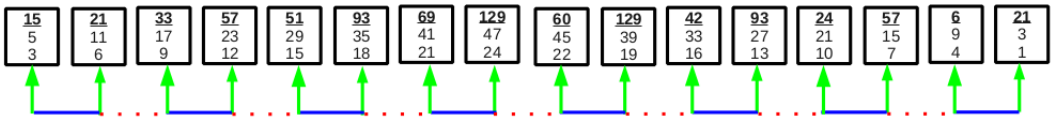


Figure 2.14-B : The result of step 2a-1 illustrated for the anti-causal part.

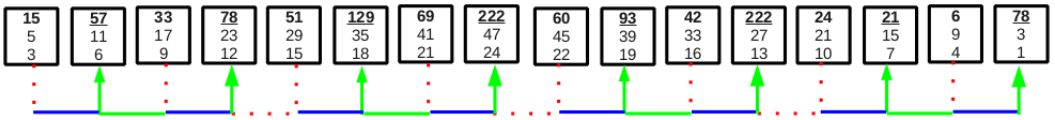


Figure 2.14-C : The result of step 2a-2 illustrated for the anti-causal part.

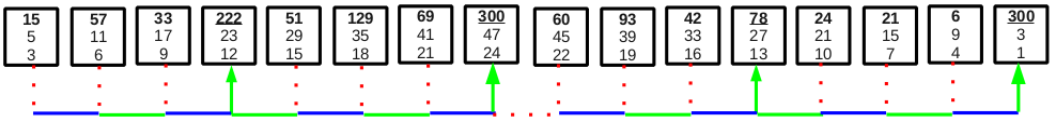


Figure 2.14-D : The result of step 2a-3 illustrated for the anti-causal part.

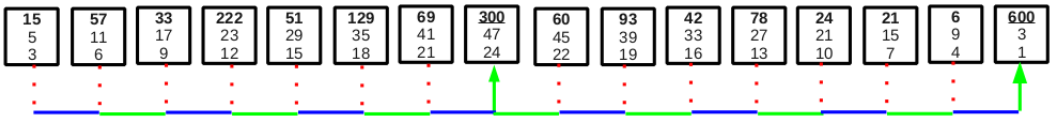


Figure 2.14-E : The result of step 2a-4 illustrated for the anti-causal part.

2b. **Step 2b - Down-sweep** : Before beginning this step, we replace the last prefix sum of last processor by 0. It only takes a replacement, so we don't count it. It is illustrated in Figure 2.14-F.

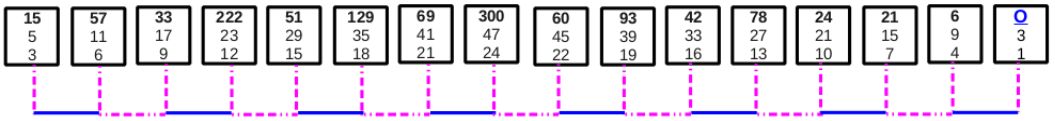


Figure 2.14-F : The replacement of the first prefix sum of last processor.

This step is also divided by d sub-steps. At the sub-step 2b-i^{th} $(1 \leq i \leq d)$, the $(2^{d-i}(2j-1))^{th}$ processor receives and adds the M complex values from the $(2^{d-i+1}j)^{th}$ processor. Other processors have no computation. We count a replacement as an addition. So at this each sub-step, there are $M \times 2^{i-1}$ complex numbers sent and $M \times 2^{i-1}$ complex numbers added. These sub-steps are illustrated from Figure 2.14-H to Figure 2.14-K. Therefore, the number of operations and transmissions of this step for the anti-causal part is given by

$$O_{comm\_down\_anti} = 2M \times (2^d - 1)\ \text{Ⓢ} + 2M \times (2^d - 1) \oplus . \tag{2.47}$$

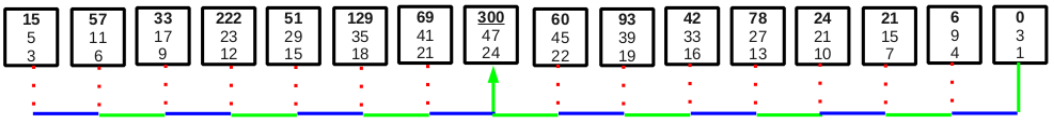


Figure 2.14-H : The result of Sub-step 2b-1 illustrated for the anti-causal part.

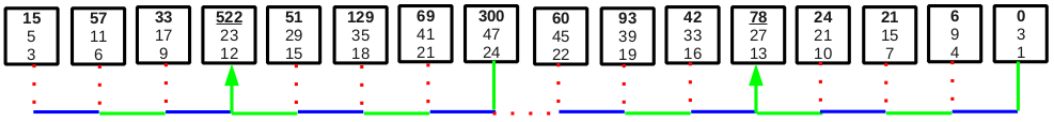


Figure 2.14-I : The result of Sub-step 2b-2 illustrated for the anti-causal part.

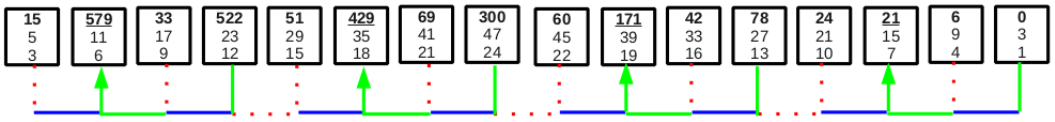


Figure 2.14-J : The result of Sub-step 2b-3 illustrated for the anti-causal part.

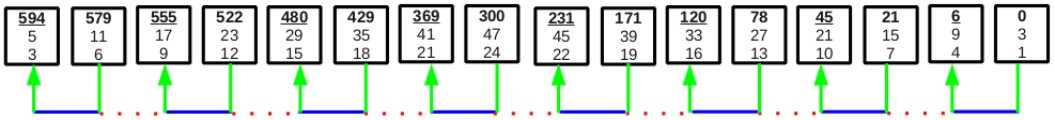


Figure 2.14-K : The result of Sub-step 2b-4 illustrated for the anti-causal part.

3. Step 3 - Ending cumulation : This last step is executed by all processors. It consists in adding the first prefix sum to all the local elements, then putting it in the last location. Here an operation which puts the first prefix sum in the last location is considered as an addition. The results of this step of the example are illustrated in Figure 2.14-L.
   So the number of operations of this step for the anti-causal part is given by

$$O_{end\_cumul\_anti} = 2m \times M \times 2^d \ \oplus . \tag{2.48}$$

Figure 2.14-L : The result of Step 3 illustrated for the anti-causal part.

   So the total number of operations and transmissions to compute prefix sums in parallel for the anti-causal part is

$$\begin{aligned} O_{cumul\_anti} &= O_{init\_comp\_anti} + O_{comm\_up\_anti} \\ &\quad + O_{comm\_down\_anti} + O_{end\_cumul\_anti} \\ &= (4m \times M \times 2^d) \ \otimes + \left(6m \times M \times 2^d + 6M \times 2^d - 6M\right) \ \oplus \\ &\quad + (6M \times (2^d - 1)) \ \text{Ⓢ}. \end{aligned} \tag{2.49}$$

4. Step 4 - Estimation of the contour : It is as similar as for the causal part. It means the number of operations of this step is given by

$$O_{estim\_contour\_anti} = (10m \times M \times 2^d) \ \otimes + (8m \times M \times 2^d) \ \oplus . \tag{2.50}$$

   Therefore, the total number of operations and transmissions for the anti-causal part is

$$\begin{aligned} O_{BT\_anti} &= O_{cumul\_anti} + O_{estim\_contour\_anti} \\ &= (14m \times M \times 2^d) \ \otimes + \left(14m \times M \times 2^d + 6M \times 2^d - 6M\right) \ \oplus \\ &\quad + (6M \times (2^d - 1)) \ \text{Ⓢ}. \end{aligned} \tag{2.51}$$

   Finally, the total number of operations and transmissions for both parts is given

$$\begin{aligned} O_{BT} &= O_{BT\_causal} + O_{BT\_anti} \\ &= (28m \times M \times 2^d) \ \otimes + \left(28m \times M \times 2^d + 12M \times 2^d - 12M\right) \ \oplus \\ &\quad + (12M \times (2^d - 1)) \ \text{Ⓢ}. \end{aligned} \tag{2.52}$$

Remark 11 :

In the first choice, the communication of both parts can be implemented independently. Therefore, the computation time in this case is

$$\begin{aligned} T_{Bt\_2} &= T_{BT\_causal} \\ &= 14m \times M \times t_m + 2M \times (7m + 2d) \times t_a + 6M \times d \times t_s. \end{aligned} \tag{2.53}$$

2.1.3.3/ ESTIMATION OF THE EXECUTION TIME OF THE DR METHOD WITH A BINARY TREE

We consider the second choice as above. At the first step, all processors are computing in parallel, so

$$T_{init\_comp} = 4m \times M \times t_m + 4m \times M \times t_a. \tag{2.54}$$

At each sub-step 2a-j ($j = 1, ..., 4$), some processors receive and add M complex numbers at the same time. Other processors have no computation but we take the upper bound time. So the computation time of this step is given by

$$T_{comm\_up} = 2M \times d \times t_a + 2M \times d \times t_s. \tag{2.55}$$

At each sub-step $2b - j$, some processors send M complex numbers, then they receive and add M complex numbers from the processor which they sent. There are only 2^j executing this task. We remark that these processors are separated into each pair and there are two communicated directions (forward and backward) between that each pair. The addition is overlapped, but the communication is not overlapped if we use only one link between each pair of these processors. So the computation time of this step is given by

$$T_{comm\_down} = 2M \times d \times t_a + 4M \times d \times t_s. \tag{2.56}$$

At the third step, all processors execute their computation in parallel.

So the computation time of this step is the computation time at each processor

$$T_{end\_cumul} = 2m \times M \times t_a. \tag{2.57}$$

For the estimation of the contour, its computation time is

$$T_{Estim\_contour} = 10m \times M \times t_m + 8m \times M \times t_a. \tag{2.58}$$

For the anti-causal part, it is symmetric. So

$$\begin{aligned} T_{Bt1} &= 2 \times T_{init\_comp} + 2 \times T_{comm\_up} + 2 \times T_{comm\_down} \\ &\quad +2 \times T_{end\_cumul} + 2 \times T_{Estim\_contour} \\ &= 28m \times M \times t_m + 4M \times (7m + 2d) \times t_a + 12M \times d \times t_s. \end{aligned} \tag{2.59}$$

2.2/ PARALLEL ALGORITHM FOR THE DIRECT METHOD WITH A LINE TOPOLOGY

We recall the formula of the direct method

$$z(x_\ell) \approx h_x \sum_{j=0}^{H-1} p(x_\ell, y_j) u(y_j), \text{ for all } \ell. \tag{2.60}$$

In fact, this computation is represented as a product of a $H \times H$ real-valued matrix P multiplied by a $H \times 1$ real vector U with

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1H} \\ P_{21} & P_{22} & \dots & P_{2H} \\ \vdots & \vdots & \ddots & \vdots \\ P_{H1} & P_{H2} & \dots & P_{HH} \end{bmatrix}, \; U = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_H \end{bmatrix}$$

where $P_{\ell+1,j+1} = p(x_\ell, y_j)$ and $U_{j+1} = u(x_j)$, $\ell, j = 0, 1, ..., H-1$. This method is summarized in Algorithm 7. A similar line network as in DR method is used to implement this method in parallel. An example for parallel computation of the direct method is shown in Figure 2.15.

Offline computation of the kernel p : $P = (P_{\ell,j}) = p(x_\ell, y_j)$.
Online computation
for $\ell = 0, ..., H-1$ **do**
  $z_\ell = 0$;
  for $j = 0, ..., H-1$ **do**
    $z_\ell = z_\ell + P_{\ell,j} u_j$;
  end
end

Algorithme 7: The algorithm of the direct method.

2.2.1/ ESTIMATION OF THE NUMBER OF OPERATIONS AND TRANSMISSIONS

We suggest to proceed in k steps. The first step is to compute the local sums. The remaining steps are to communicate in order to exchange the inputs U_j. In these communication steps, there are two classes of processors. Each processor in the first class receives only one input from either its right neighbour or its left neighbour.

They are illustrated in Figure 2.16. The parallel computation of this method is described as follows :

1. Step 1 - Initial summation : The local sums are computed on all processors. Namely, at the j^{th} processor, $j = 1, ..., k$, there are m local sums

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(j-1)+1}^{mj} P_{\ell,s} U_s, \text{ for all } \ell = m(j-1)+1, ..., mj.$$

   We assume that the initial results $\texttt{res}_\ell$ are assigned by 0. This step involves m^2 multiplications and m^2 additions at each processor. So the total number of operations of this step is given

$$O_{step\_1} = m^2 \times k \otimes + m^2 \times k \oplus .$$

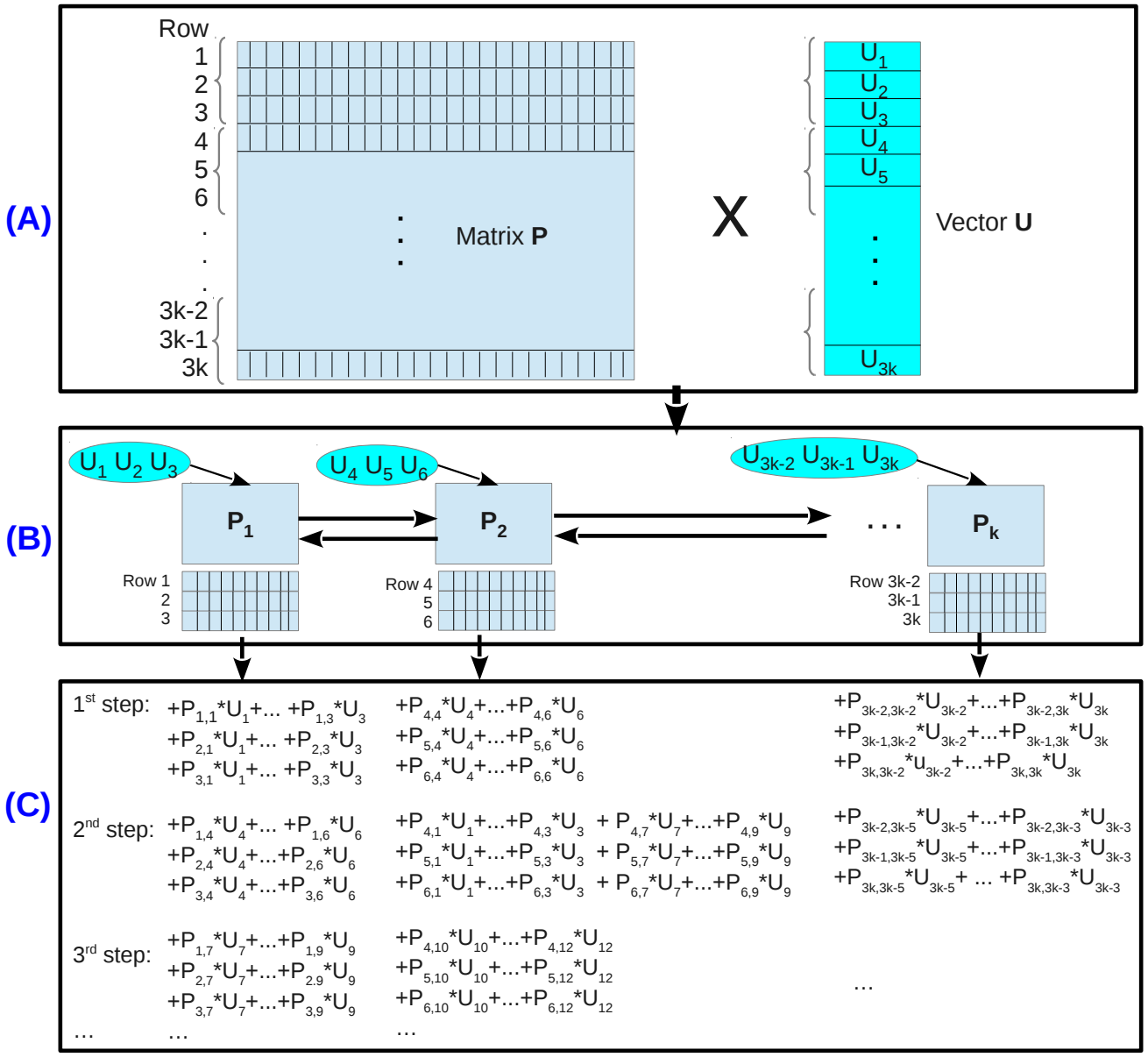


FIGURE 2.15 – An illustration for parallel computation of a matrix-vector multiplication on a line network with k processors. **(A)** Preparation step : the partitioning of the matrix and the vector. **(B)** The distribution of the matrix, in this example each processor controls 3 sensors (i.e., inputs), namely, the j^{th} processor stores three rows $3j-2, 3j-1, 3j$ of the matrix as its initial data. **(C)** Main computation steps of parallel algorithm.

2. Step 2 - The first communication : The first processor and the last processor belong to the first class. Each of them only receives m real values (i.e., inputs) from its neighbour and implements the computations as follows, at the first processor

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m+1}^{2m} P_{\ell,s} U_s \text{ for all } \ell = 1, ..., m,$$

FIGURE 2.16 – A k-steps process of the direct method on a line network with k processors. The $V_j, j = 1, ..., k$ represents the inputs $U_{(m-1)j+1}, U_{(m-1)j+2}, ..., U_{mj}$ which are exchanged throughout communication steps. The yellow lines represent the connection links without communication, the red arrows represent communications along one direction, and the blue arrows represent communications along two directions.

and at the last processor

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(k-2)+1}^{m(k-1)} P_{\ell,s} U_s \text{ for all } \ell = m(k-1)+1, ..., mk).$$

Thus each of these two processors requires

$$O_{class\_1} = m\,\text{Ⓢ} + m^2 \otimes + m^2 \oplus .$$

All processors expect the first processor and the last processor belong to the second class. Each of them receives $2m$ real values from its two neighbours and implement the computations. Precisely, at the j^{th} processor, $j = 2, ..., k-1$, there are m double

local sums

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(j-2)+1}^{m(j-1)} P_{\ell,s}U_s + \sum_{s=mj+1}^{m(j+1)} P_{\ell,s}U_s \text{ for all } \ell = m(j-1)+1, ..., mj.$$

So each processor in the second class requires

$$O_{class\_2} = 2m\ \textcircled{S} + 2m^2 \otimes + 2m^2 \oplus .$$

It is clear that there are two processors in the first class and $(k-2)$ processors in the second class. Therefore, the total number of operations and transmissions of this step is given by

$$\begin{aligned} O_{step\_2} &= 2 \times O_{class\_1} + (k-2) \times O_{class\_2} \\ &= 2m^2 \times (k-1) \otimes + 2m^2 \times (k-1) \oplus + 2m \times (k-1)\ \textcircled{S}. \end{aligned} \tag{2.61}$$

3. Step 3 - The second communication :
The two first processors and the two last processor belong to the first class. So each of them only receives m real values from neither its right neighbour nor its left neighbour. Namely, two first processors receive data from their right neighbours, and two last processor receive data from their left neighbours. Then they implement the computations, at the two first processors (with $j = 1$ and $j = 2$)

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(j+1)+1}^{m(j+2)} P_{\ell,s}U_s \text{ for all } \ell = m(j-1)+1, ..., mj,$$

and at the two last processors (with $j = k-1$ and $j = k$)

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(j-3)+1}^{m(j-2)} P_{\ell,s}U_s \text{ for all } \ell = m(j-1)+1, ..., mj.$$

These processors therefore require $O_{class\_1} = m\ \textcircled{S} + m^2 \otimes + m^2 \oplus$.

The $(k-4)$ remaining processors belong to the second class. Each of them receives $2m$ real values from its two neighbours and implement the computations. Precisely, at the j^{th} processor, $j = 3, ..., k-2$, there are m double local sums

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(j-3)+1}^{m(j-2)} P_{\ell,s}U_s + \sum_{s=m(j+1)+1}^{m(j+2)} P_{\ell,s}U_s \text{ for all } \ell = m(j-1)+1, ..., mj.$$

Each of these $(k-4)$ processors requires $O_{class\_2} = 2m\ \textcircled{S} + 2m^2 \otimes + 2m^2 \oplus$.

So the total number of operations and transmissions of this step is given by

$$\begin{aligned} O_{step\_3} &= 4 \times O_{class\_1} + (k-4) \times O_{class\_2} \\ &= 2m^2 \times (k-2) \otimes + 2m^2 \times (k-2) \oplus + 2m \times (k-2)\ \textcircled{S}. \end{aligned}$$

In generally, if we assume that k is even then

k/2-r. Step $(k/2-r)^{th}$ - The $(k/2-r-1)^{th}$ communication, with $r = 0, ..., k/2-2$:
The $(k/2-r-1)$ first processors and the $(k/2-r-1)$ last processors belong to the first class. Each of the $(k/2-r-1)$ first processors receives m real values from its *right neighbour* and executes (with $j = 1, ..., k/2-r-1$)

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(j+k/2-r-2)+1}^{m(j+k/2-r-1)} P_{\ell,s}U_s \text{ for all } \ell = m(j-1)+1, ..., mj,$$

Moreover, each of the $(k/2-r-1)$ last processors receives m real values from its *left neighbour* and executes (with $j = k/2+r+2, ..., k$)

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(j-k/2+r)+1}^{m(j-k/2+r+1)} P_{\ell,s}U_s \text{ for all } \ell = m(j-1)+1, ..., mj.$$

Each of these $(k-2r-2)$ processors requires $O_{class\_1} = m\,\textcircled{S} + m^2 \otimes + m^2 \oplus$.
The remaining processors belong to the second class. Each of them receives $2m$ real values from its two neighbours and implements the computations. Namely, at the j^{th} processor, $j = k/2-r, ..., k/2+r+1$, there are m double local sums

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(j-k/2+r)+1}^{m(j-k/2+r+1)} P_{\ell,s}U_s + \sum_{s=m(j+k/2-r-2)+1}^{m(j+k/2-r-1)} P_{\ell,s}U_s$$

for all $\ell = m(j-1)+1, ..., mj$. Each of these $(2r+2)$ processors requires $O_{class\_2} = 2m\,\textcircled{S} + 2m^2 \otimes + 2m^2 \oplus$. So the total number of operations and transmissions of this step is given by

$$\begin{aligned} O_{step\_(k/2-r)} &= (k-2r-2) \times O_{class\_1} + (2r+2) \times O_{class\_2} \\ &= 2m^2(k/2+r+1) \otimes + 2m^2(k/2+r+1) \oplus \\ &\quad + 2(k/2+r+1) \times m\,\textcircled{S}. \end{aligned}$$

Moreover, the next steps are generalized as follows :

k/2+r. Step $(k/2+r)^{th}$ - The $(k/2+r-1)^{th}$ communication, $r = 1, ..., k/2$:
The $(k/2-r+1)$ first processors and the $(k/2-r+1)$ last processors belong to the first class. Each of the $(k/2-r+1)$ first processors receives m real values from its *right neighbour* and implements the computation (with $j = 1, ..., k/2-r+1$)

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(j+k/2+r-2)+1}^{m(j+k/2+r-1)} P_{\ell,s}U_s \text{ for all } \ell = m(j-1)+1, ..., mj,$$

Moreover, each of the $(k/2-r+1)$ last processors receives m real values from its *left neighbour* and executes (with $j = k/2+r, ..., k$)

$$\texttt{res}_\ell = \texttt{res}_\ell + \sum_{s=m(j-k/2-r)+1}^{m(j-k/2-r+1)} P_{\ell,s}U_s \text{ for all } \ell = m(j-1)+1, ..., mj.$$

Each of these $(k-2r+2)$ processors requires $O_{class\_1} = m\,\textcircled{S} + m^2 \otimes + m^2 \oplus$.
The $(2r-2)$ remaining processors have no computation since they have finished their

	The first class	The second class	Time ($^*t_s$)
Step 1	0	0	0
Step 2	2	k-2	2m
Step 3	4	k-4	2m
...	...	...	...
Step k/2-1	k-4	4	2m
Step k/2	k-2	2	2m
Step k/2+1	k	0	m
Step k/2+2	k-2	0	m
...	...	...	...
Step k-1	4	0	m
Step k	2	0	m
SUM	$2\times(2+4+\ldots$ $+(k-2))+k=\frac{k^2}{2}$	$2+4+\ldots+(k-2)$ $=\frac{k}{2}\left(\frac{k}{2}-1\right)$	$2m\times(k/2-1)$ $+m\times k/2=m\left(\frac{3k}{2}-2\right)$

TABLE 2.5 – The number of processor of each class and the computation time at each step.

computations from previous steps. So the total number of operations and transmissions is given by

$$\begin{aligned} O_{step\_(k/2+r)} &= (k-2r+2)\times O_{class\_1} \\ &= 2m^2\times(k/2-r+1)\otimes+2m^2\times(k/2-r+1)\oplus \\ &\quad +2m\times(k/2-r+1)\,\text{Ⓢ}. \end{aligned}$$

The number of processors of each class at each step is summarized by the second and the third column of Table 2.5. From this table, we can compute the total number of operations and transmissions of this method as follows

$$\begin{aligned} O_{direct} &= O_{step\_1}+...+O_{step\_k} \\ &= [2\times(2+4+\ldots+(k-2))+k]\times O_{class\_1} \\ &\quad +[2+4+\ldots+(k-2)]\times O_{class\_2} \\ &= m^2\times k^2\otimes+m^2\times k^2\oplus+m\times k\times(k-1)\,\text{Ⓢ}. \end{aligned} \tag{2.62}$$

2.2.2/ ESTIMATION OF THE EXECUTION TIME OF THE DIRECT METHOD WITH A LINE

To estimate the entire computation time, we evaluate the computation time at each step. The communication time of each step is summarized in the last column of Table 2.5.

1. Step 1 : Since k processors are computed in parallel, the computation time is

$$T_{step\_1}=m^2\times t_m+m^2\times t_a. \tag{2.63}$$

2. Step 2 : There are two processors which only receive m real values from either only from its right neighbour or from its left neighbour and implements their computations. Other ones receive $2m$ real values and execute their computations. Their

computations are also executed in parallel. So we have to take the upper bound of the computation time, namely, it takes

$$T_{step\_2} = 2m^2 \times t_m + 2m^2 \times t_a + 2m \times t_s. \tag{2.64}$$

Generally,

k/2-j. Step $(k/2-j)^{th}$ with $j = 0, 1, ..., k/2-2$: There are $2(k/2-j-1)$ processors which only receive m real values from its first or second neighbour and implement the computations. There are $2j+2$ processors which receive $2m$ real values from its two neighbours and implement the computations. So we should use the upper bound of the computation time, namely, it takes

$$T_{step\_(k/2-j)} = 2m^2 \times t_m + 2m^2 \times t_a + 2m \times t_s. \tag{2.65}$$

Moreover,

k/2+j. Step $(k/2+j)^{th}$ with $j = 1, ..., k/2$: There are $(2j-2)$ processors which execute the computation from the last steps. The remaining $2(k/2-j+1)$ processors only receive m real values from their right or left neighbour and implement the computations. So we have to use the upper bound of the computation time, namely, it takes

$$T_{step\_(k/2+j)} = m^2 \times t_m + m^2 \times t_a + m \times t_s. \tag{2.66}$$

Therefore, the total computation time of the direct method is

$$\begin{aligned} T_{direct} &= T_{step\_1} + ... + T_{step\_k} \\ &= m^2 \times (1.5k-1) \times t_m + m^2 \times (1.5k-1) \times t_a \\ &\quad + m \times (1.5k-2) \times t_s. \end{aligned} \tag{2.67}$$

2.3/ COMPARISON THE NUMBER OF OPERATIONS AND THE COMPUTATION TIME

In this section, the computation time is considered as a function $T(H)$ of the number H of discretization points. This function also depends on some parameters such as the number M of discretization nodes of the contour, the time t_m taken a multiplication, the time t_a taken an addition and the time t_s taken a transmission of a type processor used. We report some simulations based on choosing different parameters for this function.

We also consider the number $O(H)$ of operations and transmissions as a function of H. We can summarize the functions $O(H)$ and $T(H)$ of the DR with three topologies and the direct method with a line topology in Table 2.6.

We note that the discretization points (or inputs) are distributed in all processors through sensors. It is assumed that each processor controls m sensors, so we have a new summary as in Table 2.7 if we fix the number m of sensors. Therefore, the number k of processors equals $\frac{H}{m}$. In this case, the functions $O(H)$ and $T(H)$ depend on the variable H and the parameter m.

	H	$O(H)$	$T(H)$
DR with a line & no overlap	$k \times m$	$28k \times m \times M \otimes$ $+4m \times M \times (7k-1) \oplus$ $+4M \times (k-1)$ Ⓢ	$28m \times M \times t_m$ $+2M \times (14m + k - 3) \times t_a$ $+2M \times (k-1) \times t_s$
DR with line and overlap Step 1			$24m \times M \times t_m$ $+2M \times (12m + k - 3) \times t_a$ $+2M \times (k-1) \times t_s$
DR with a line and overlap Step 1 & 3			$24m \times M \times t_m$ $+2M \times (11m + k - 2) \times t_a$ $+2M \times (k-1) \times t_s$
DR with a hypercube	$2^d \times m$	$28m \times M \times 2^d \otimes$ $+2M \times (14m \times 2^d + 3d \times 2^d$ $-2^{d+1} - 2m + 2) + 2(2^d - 1) \oplus$ $+4M \times d \times 2^d$ Ⓢ	$28m \times M \times t_m$ $+4M \times (7m + 2d - 1) \times t_a$ $+2t_a + 4M \times d \times t_s$
DR with a binary tree	$2^d \times m$	$28m \times M \times 2^d \otimes$ $+2M \times (14m \times 2^d + 6 \times 2^d - 6) \oplus$ $+12M \times (2^d - 1)$ Ⓢ	$28m \times M \times t_m$ $+4M \times (7m + 2d) \times t_a$ $+12M \times d \times t_s$
Direct method with a line	$k \times m$	$m^2 \times k^2 \otimes$ $+m^2 \times k^2 \oplus$ $+m \times k \times (k-1)$ Ⓢ	$m^2 \times (1.5k - 1) \times t_m$ $+m^2 \times (1.5k - 1) \times t_a$ $+m \times (1.5k - 2) \times t_s$

TABLE 2.6 – The number $O(H)$ of operations and transmissions, and the computation time $T(H)$ considered as the functions of the number H of discretization points.

We recall that the time taken by an operation or a transmission of computation devices, namely processors, depends on the type of processors used and the type of connection links between processors. In the following, we report simulation results showing the computation time for three sets of speeds. In order to have realistic values, we took the values of a ADUC841 microcontroller. It takes 9 cycles at 20Mhz to execute an operation, so it is $9/(20 \times 10^6)$ second. It is about 0.45 micro-second. For the communication, we can consider a speed of 4 *mbits/s*. If a float is coded with 18 bits then it means $18/(4000000)$. It is about 4.5 micro-second. In order to see the influence of the communication, three configurations are considered :

1. $t_m = t_a = 0.45\ \mu s$, $t_s = 0.45\ \mu s$,
2. $t_m = t_a = 0.45\ \mu s$, $t_s = 4.5\ \mu s$,
3. $t_m = t_a = 0.45\ \mu s$, $t_s = 45\ \mu s$.

We use three different numbers of nodes $m = 1$, $m = 16$ or $m = 64$. The number of quadrature points along the contour is fixed at $M = 20$. In the case $m = 1$, the result of the DR with a line is not plotted to show the computation time. These simulations are shown in a series of the figures in this section.

Obviously, the communication time has a significant effect on the performance of all the parallel implementations and the number of sensors per processor is also very important. The line topology is especially influenced by the number of sensors. This topology is very efficient if the number of sensors per processor is large such in Figures 2.23, 2.24, 2.25. Moreover, the binary tree is less efficient. This topology is more efficient in the case the ratio between the time taken an operator and the time taken a transmission

	k	$O(H)$	$T(H)$
DR with a line & no overlap	$\frac{H}{m}$	$28H \times M \otimes$ $+4M \times (7H - m) \oplus$ $+4M \times (\frac{H}{m} - 1)$ Ⓢ	$28m \times M \times t_m$ $+2M \times (14m + \frac{H}{m} - 3) \times t_a$ $+2M \times (\frac{H}{m} - 1) \times t_s$
DR with a line & overlap Step 1			$24m \times M \times t_m$ $+2M \times (12m + \frac{H}{m} - 3) \times t_a$ $+2M \times (\frac{H}{m} - 1) \times t_s$
DR with a line and overlap Step 1 & 3			$24m \times M \times t_m$ $+2M \times (11m + \frac{H}{m} - 2) \times t_a$ $+2M \times (\frac{H}{m} - 1) \times t_s$
DR with a hypercube	$\frac{H}{m}$	$28H \times M \otimes$ $+2M \times (14H + 3\log_2(\frac{H}{m}) \times \frac{H}{m}$ $-2\frac{H}{m} - 2m + 2) + 2(\frac{H}{m} - 1) \oplus$ $+4M \times \log_2(\frac{H}{m}) \times \frac{H}{m}$ Ⓢ	$28m \times M \times t_m$ $+4M \times (7m + 2\log_2(\frac{H}{m}) - 1) \times t_a$ $+2t_a + 4M \times \log_2(\frac{H}{m}) \times t_s$
DR with a binary tree	$\frac{H}{m}$	$28H \times M \otimes$ $+2M \times (14H + 6\frac{H}{m} - 6) \oplus$ $+12M \times (\frac{H}{m} - 1)$ Ⓢ	$28m \times M \times t_m$ $+4M \times (7m + 2\log_2(\frac{H}{m})) \times t_a$ $+12M \times \log_2(\frac{H}{m}) \times t_s$
Direct method with a line	$\frac{H}{m}$	$H^2 \otimes$ $+H^2 \oplus$ $+H \times (\frac{H}{m} - 1)$ Ⓢ	$m \times (1.5H - m) \times t_m$ $+m \times (1.5H - m) \times t_a$ $+(1.5H - 2m) \times t_s$

TABLE 2.7 – The functions $O(H)$ and $T(H)$ of the number H of discretization points with the number k sensors per processor fixed.

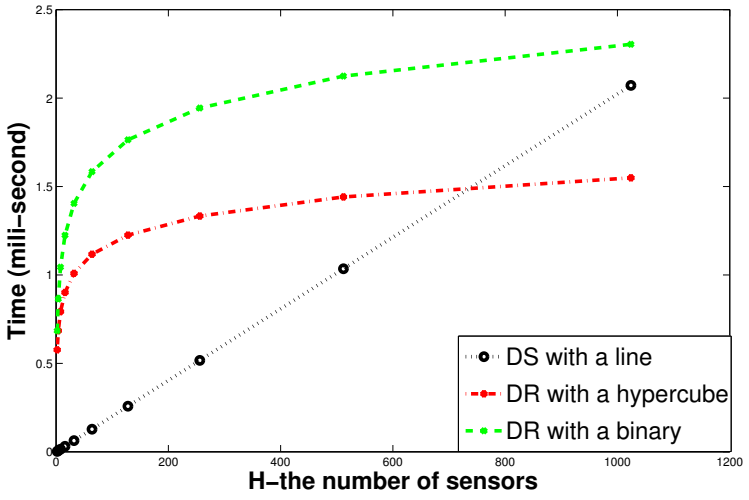


FIGURE 2.17 – Computation time in seconds with $M = 20, m = 1, t_s = 0.45\ \mu s$.

is not so big such as in Figures 2.20, 2.23. As mentioned in the previous part, the only solution to improve it, is to use more links so that some communications can be performed in parallel. In all these simulations, the hypercube technology is, without surprise, the most efficient. This topology has been used in many algorithms since it offers a good compromise between the number of communication links used and the efficiency. The next step will be to implement a real distributed control algorithm.

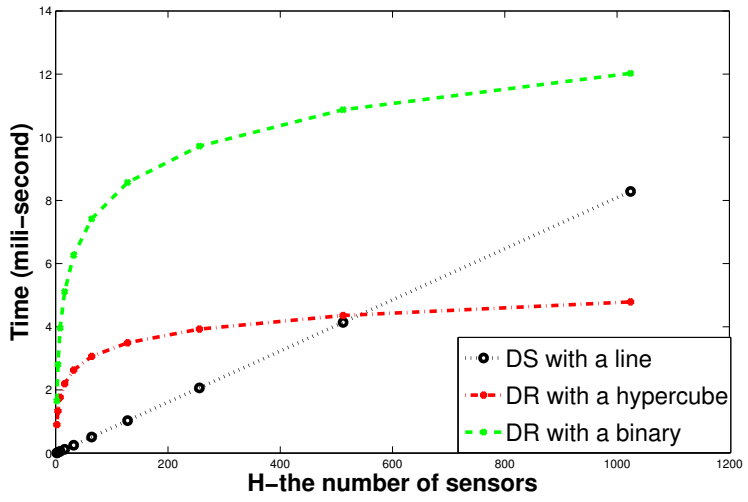


FIGURE 2.18 – Computation time in seconds with $M = 20, m = 1, t_s = 4.5\ \mu s$.

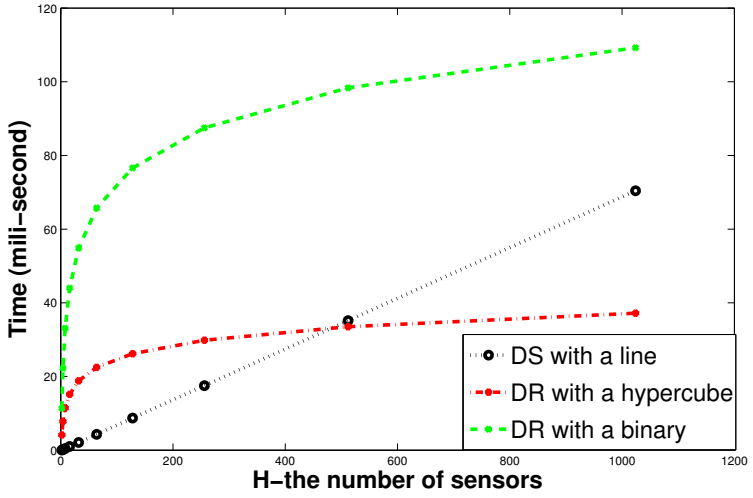


FIGURE 2.19 – Computation time in seconds with $M = 20, m = 1, t_s = 45\ \mu s$.

2.4/ CONCLUSION

This chapter described three network topologies : a line, a binary and a hypercube. The number of operations and transmissions for DR method are evaluated for all three topologies. A similar evaluation is executed for the direct method with a line topologies. Moreover, we also computed and compared the computation time between our algorithm with these topologies and the direct method with a line topology. Each node of these networks only requires a few operations and each local program can read a few inputs. Therefore, microcontrolers are well suitable for implementation of the DR method. They are small, cheap, and they have real-time computing capacity, thus they are the best choice for a control system.

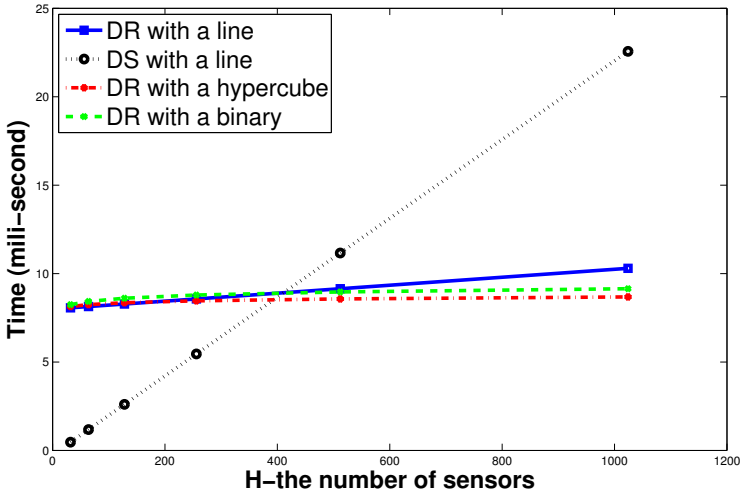


FIGURE 2.20 – Computation time in seconds with $M = 20, m = 16, t_s = 0.45\ \mu s$.

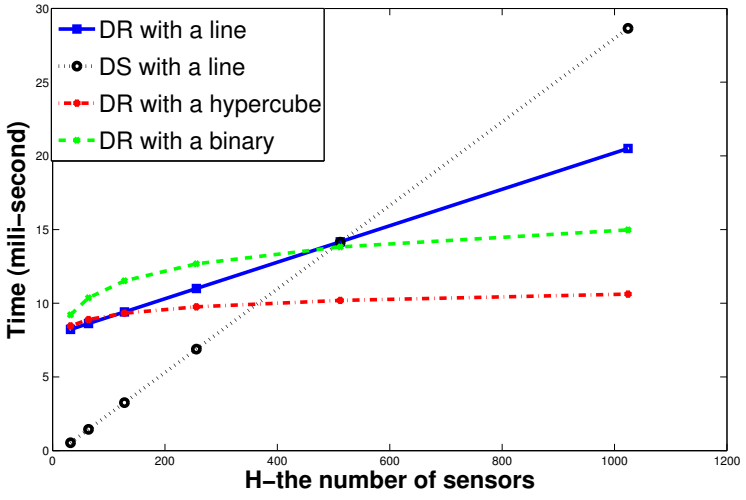


FIGURE 2.21 – Computation time in seconds with $M = 20, m = 16, t_s = 4.5\ \mu s$.

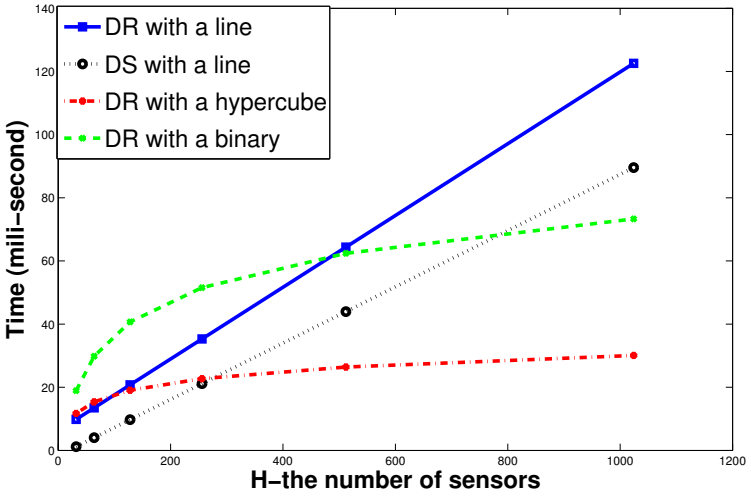


FIGURE 2.22 – Computation time in seconds with $M = 20, m = 16, t_s = 45\ \mu s$.

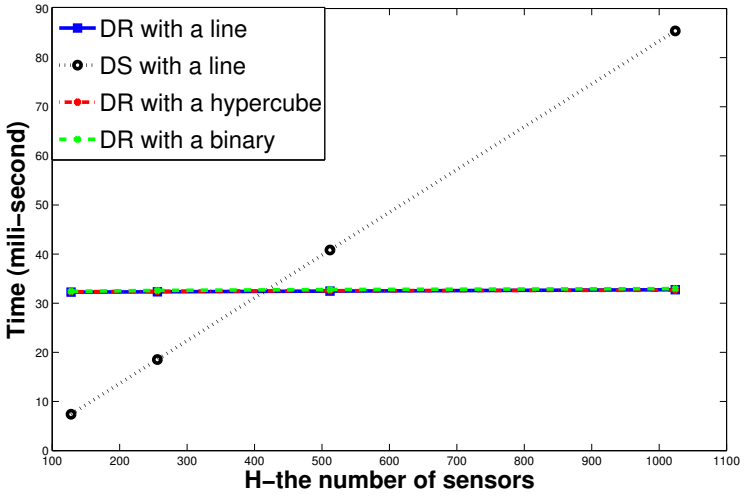


FIGURE 2.23 – Computation time in seconds with $M = 20, m = 64, t_s = 0.45\ \mu s$.

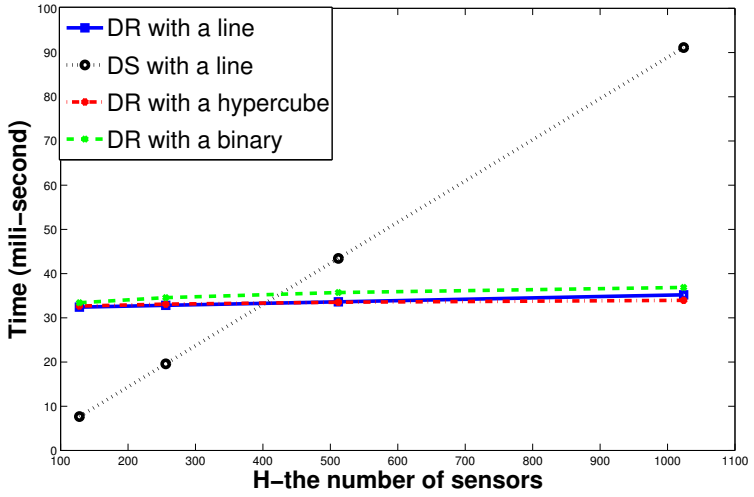


FIGURE 2.24 – Computation time in seconds with $M = 20, m = 64, t_s = 4.5\ \mu s$.

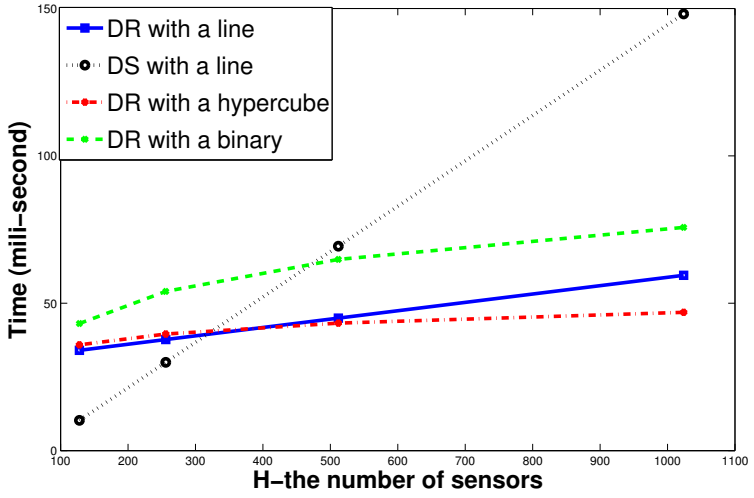


FIGURE 2.25 – Computation time in seconds with $M = 20, m = 64, t_s = 45\ \mu s$.

3

ERROR ESTIMATES

In Chapter 1, we recall the DR theory of integral operators, then a numerical method is detailed. The numerical method consists in the truncated trapezoidal rule for the inverse Laplace transform computation, the spectral Galerkin method for the kernel approximation, and the piecewise constant interpolation or the piecewise linear interpolation for discretization of inputs u. In Chapter 2, estimates of numbers of operations and computation times on parallel architectures are discussed in the framework of parallel implementation. In this chapter, we establish the error estimates and its application to contour optimization.

As already mentioned, one of the main advantages of this method is its very low computation cost. However, in the framework of parallel implementation, this algorithm is more effective if the number M of quadrature nodes along the contours is very small such as $M = 20$. To achieve this, our main objective in this work is to derive estimates for the parameters of the contour that will ensure, for given M, a small error.

Applying an idea similar to one of Weideman and Trefethen [35] in the framework of the numerical inversion of the Laplace transform, we find optimal parameters that define an hyperbolic contour based on the evaluation of an error estimate in our method. There are three sources of errors in the algorithm. The first one is the quadrature error of the contour integral, which is well controlled. The second one is due to the piecewise constant interpolation of the input u by $\widetilde{u}$ as mentioned in Chapter 1. The third one comes from the approximation of the kernel by the Galerkin method. We study two approaches in evaluation of error estimates : one in view of discretization of input u executed before discretization of the contour ; the other is the opposite case, i.e., in view of discretization of the contour executed firstly as in [19]. To be convenient, we call them to be the first way of error evaluation and the second way of error evaluation, respectively.

The methods for inverting the Laplace transform that Weideman and Trefethen [35] considered are all based on numerical integration of the Bromwich complex contour integral

$$f(x) = \frac{1}{2\pi i} \int_{\mathbb{R}} e^{\theta(\xi)x} F(\theta(\xi))\theta'(\xi) d\xi.$$

In the diffusive realization, we consider the numerical integration

$$z^{\pm}(x) = \int_{\mathbb{R}} \psi^{\pm}(x,\xi)\mu^{\pm}(x,\xi) d\xi,$$

where $\psi^{\pm}$ are the integral forms of (1.1) as in (1.26) and $\mu^{\pm}$ are the diffusive symbols defined in (1.2).

Evidently, not only the form of $\psi^{\pm}(x,\xi)$ is different from the corresponding form $e^{\theta(\xi)x}$ but also the Laplace transform $\mathcal{P}^{\pm}(x,-\theta^{\pm}(\xi))$ is more precise than $F(\theta(\xi))$. Therefore, the Weideman and Trefethen's results [35] cannot be directly applied. However, if $\psi^{\pm}(x,\xi)$ is discretized with respect to x by approximating the input u by a piecewise constant function then the numerical integration becomes as in (1.34)

$$z^{\pm}(x_n) \simeq \pm \sum_{j\in J_n^{\pm}} u_{j+\frac{1}{2}}\Big(\mathcal{L}^{-1}(F_n^{\pm}(-\theta^{\pm}))(\pm t_{n\pm1,j}) - \mathcal{L}^{-1}(F_n^{\pm}(-\theta^{\pm}))(\pm t_{n,j})\Big).$$

This is a linear combination of inverse Laplace transforms at the times $t_{n,j}$. Following Weideman and Trefethen, the numerical integration formula along the contour can be used to approximate each of these inverse Laplace transforms. The optimization of the contours is found on a balance between the truncation error estimate and the quadrature error estimate for the Laplace inversion at points $t_{n,j} \in I = \{h, 2h, ..., H.h = 1\}$ excluding the point 0. The ratio $H = \frac{1}{h}$ between the upper and lower bounds of the set I is very large for a fine mesh and the numerical inversion of Laplace transform is relatively expensive. To avoid this problem, our strategy is to use a Stenger's result to evaluate each of these Laplace inversions, then the error estimate is a sum of these sub-error evaluations.

Remark 12 :

Another strategy would be to use a contour per Laplace inversion or use Remark 3.2 in Helie [10], at least each contour used for a given time interval, that is, long time spans should be split into adjacent intervals of equal diameter in order to maintain the desired convergence rate :

$$[h,1] = [h,\Lambda h] \cup [\Lambda h, \Lambda^2 h] \cup ... \cup [\Lambda^n h, 1].$$

This is not so costly as the parameters are optimized once for all for a fixed time interval diameter Λ. However, this may not be the ideal solution for real-time computation. The idea using many contours is not thus considered.

In the next section, we present analytical preliminaries in the framework of numerical inversion of the Laplace transform which will be applied in our error estimates. We provide then proof for the convenience of the reader. The error estimate problem and the results are stated in Section 3.2. The proofs of theorems and lemmas are detailed in Section 3.3. Numerical simulations are presented in Section 3.4. Finally, other approaches are discussed in Section 3.5.

3.1/ ERROR ESTIMATE OF INTEGRALS OVER $\mathbb{R}$ APPROXIMATED BY A QUADRATURE FORMULA

Let $f : \mathbb{R} \to \mathbb{R}$ be an integrable mapping. We recall an accurate estimate for the quadrature error

$$E_M(f,\hbar) = \int_{\mathbb{R}} f(\xi)d\xi - \hbar \sum_{k=-M}^{M} f(\xi_k), \text{ with } \xi_k = k\hbar, \hbar > 0, M \geq 1, \tag{3.1}$$

of the truncated trapezoidal rule, in the spirit of Stenger [32], [33], and adapted from López [20].

For $d = (d_1, d_2)$ with $d_1 > 0$ and $d_2 > 0$, let $\mathcal{B}_d$ denote the family of all functions that are analytic in the horizontal strip

$$D_d = \{z \in \mathbb{C} : -d_1 \leq \Im z \leq d_2\}, \tag{3.2}$$

such that

$$\int_{-d_1}^{d_2} |f(\xi + iy)|dy \to 0 \text{ as } \xi \to \pm\infty, \tag{3.3}$$

and

$$\mathcal{N}(f, D_d) := \int_{\mathbb{R}} |f(\xi - id_1)|d\xi + \int_{\mathbb{R}} |f(\xi + id_2)|d\xi < \infty. \tag{3.4}$$

We assume that f admits a holomorphic extension $f : \mathbb{C} \setminus \Sigma_\delta \to \mathbb{R}$ to the complement of some acute sector

$$\Sigma_\delta = \{z \in \mathbb{C} : |\arg(-z)| \leq \delta\},\ 0 < \delta < \frac{\pi}{2}. \tag{3.5}$$

Given δ, let us select $\alpha > 0, d_1 > 0$ and $d_2 > 0$ such that

$$d_1 < \alpha < \frac{\pi}{2} - d_2 \text{ and } \alpha + d_2 + \delta < \frac{\pi}{2}. \tag{3.6}$$

The estimate is stated as follows.

Theorem 2 :

Assume that $f \in \mathcal{B}_d$, for some $d = (d_1, d_2)$ with $d_1, d_2 > 0$ and $|f(\xi)| \leq \dfrac{C}{|\xi|^\alpha}$ with some constant $C > 0$ and $\alpha > 1$. Suppose further that for some $\mathcal{N}^+ > 0$ and $\mathcal{N}^- > 0$ the function f satisfies

$$\int_{\mathbb{R}} |f(\xi + ir)|d\xi \leq \mathcal{N}^+, \quad \int_{\mathbb{R}} |f(\xi - is)|d\xi \leq \mathcal{N}^-, \tag{3.7}$$

for all $0 < r < d_2$ and $0 < s < d_1$. Then

$$\Big| \int_{\mathbb{R}} f(\xi)d\xi - \hbar \sum_{k\in\mathbb{Z}} f(\xi_k) \Big| \leq E^+ + E^-, \tag{3.8}$$

where

$$E^+ = \frac{\mathcal{N}^+}{e^{2\pi d_2/\hbar} - 1},\ E^- = \frac{\mathcal{N}^-}{e^{2\pi d_1/\hbar} - 1}.$$

Proof

The first step is to establish a Poisson summation formula. We define a $\hbar-$periodic analytic function

$$T_\hbar(z) = \hbar \sum_{j\in\mathbb{Z}} f(j\hbar + z). \tag{3.9}$$

The condition $|f(\xi)| \leq \dfrac{C}{|\xi|^{\alpha}}$ ensures that this sum converges uniformly for $0 \leq z \leq \hbar$. Hence, we define the Fourier coefficients of $T_\hbar$

$$t_k^\hbar = \frac{1}{\hbar}\int_0^\hbar T_\hbar(z)e^{-i2k\pi\frac{z}{\hbar}}dz. \tag{3.10}$$

so that we have

$$T_\hbar(z) = \sum_{k\in\mathbb{Z}} t_k^\hbar e^{i2k\pi\frac{z}{\hbar}}. \tag{3.11}$$

Substituting equation (3.9) into (3.10), and inverting the summation and the integration yields

$$t_k^\hbar = \sum_{j\in\mathbb{Z}}\int_0^\hbar f(j\hbar + z)e^{-i2k\pi\frac{z}{\hbar}}dz.$$

Performing the change of variable $\xi = j\hbar + z$, we obtain

$$t_k^\hbar = \sum_{j\in\mathbb{Z}}\int_{j\hbar}^{(j+1)\hbar} f(\xi)e^{-i2k\pi\frac{\xi}{\hbar}}e^{i2k\pi j}d\xi = \int_{\mathbb{R}} f(\xi)e^{-i2k\pi\frac{\xi}{\hbar}}d\xi,$$

since $e^{i2k\pi j} = 1, \forall k, j$. We introduce the Fourier Transform of the function f

$$\widehat{f}(u) = \int_{\mathbb{R}} f(t)e^{-iut}dt,$$

thus

$$t_k^\hbar = \widehat{f}\left(\frac{2k\pi}{\hbar}\right),$$

wherefrom we infer

$$T_\hbar(z) = \sum_{k\in\mathbb{Z}}\widehat{f}\left(\frac{2k\pi}{\hbar}\right)e^{i2k\pi\frac{z}{\hbar}}. \tag{3.12}$$

We remark that $\displaystyle\int_{\mathbb{R}} f(\xi)d\xi = \widehat{f}(0)$, so

$$T_\hbar(z) - \int_{\mathbb{R}} f(\xi)d\xi = \sum_{k\neq 0}\widehat{f}\left(\frac{2k\pi}{\hbar}\right)e^{i2k\pi\frac{z}{\hbar}}. \tag{3.13}$$

Note that the above identity is valid for all real z thanks to the analytic condition in the strip D_d and according to Theorem 10.6e in Henrici [11].
The next step uses the holomorphic extension to infer the estimate from the above Poisson summation formula (3.12). We use the holomorphic extension to obtain (since $f(t)$ is analytic on the strip $0 < r < d_2$, $\int_{\mathbb{R}} f(t)e^{-iut}dt = \int_{\mathbb{R}+ir} f(t)e^{-iut}dt$)

$$\widehat{f}(u) = e^{ur}\int_{\mathbb{R}} f(\xi + ir)e^{-iu\xi}d\xi,$$

Thanks to the condition (3.7), we obtain

$$\left|\widehat{f}\left(\frac{2k\pi}{\hbar}\right)\right| \leq e^{\frac{2k\pi}{\hbar}r}\int_{\mathbb{R}}\left|f(\xi+ir)\right|e^{-i\frac{2k\pi}{\hbar}\xi}|d\xi \leq e^{\frac{2k\pi}{\hbar}r}\mathcal{N}^{+}, \forall r\in(0,d_2).$$

In the same manner, using the holomorphic extension in the remaining part of the strip yields

$$\left|\widehat{f}\left(\frac{2k\pi}{\hbar}\right)\right| \leq e^{-\frac{2k\pi}{\hbar}s}\mathcal{N}^{-}, \forall s\in(0,d_1).$$

Now, the formula (3.13) can be written as follows :

$$T_\hbar(z)-\int_{\mathbb{R}}f(\xi)d\xi=\sum_{k\in\mathbb{N}^*}\widehat{f}\left(-\frac{2k\pi}{\hbar}\right)e^{-i2k\pi\frac{z}{\hbar}}+\widehat{f}\left(\frac{2k\pi}{\hbar}\right)e^{i2k\pi\frac{z}{\hbar}}.$$

So

$$\begin{aligned}\left|T_\hbar(z)-\int_{\mathbb{R}}f(\xi)d\xi\right| &\leq \sum_{k\in\mathbb{N}^*}\left|\widehat{f}\left(-\frac{2k\pi}{\hbar}\right)\right|\times\left|e^{-i2k\pi\frac{z}{\hbar}}\right|+\left|\widehat{f}\left(\frac{2k\pi}{\hbar}\right)\right|\times\left|e^{i2k\pi\frac{z}{\hbar}}\right|\\ &\leq \sum_{k\in\mathbb{N}^*}\left|\widehat{f}\left(-\frac{2k\pi}{\hbar}\right)\right|+\left|\widehat{f}\left(\frac{2k\pi}{\hbar}\right)\right|.\end{aligned}$$

Therefore, using the two estimates obtained above at the limit points $r=d_2$ and $s=d_1$, we get

$$\left|T_\hbar(z)-\int_{\mathbb{R}}f(\xi)d\xi\right|\leq\sum_{k\in\mathbb{N}^*}\mathcal{N}^{+}e^{-\frac{2k\pi}{\hbar}d_2}+\mathcal{N}^{-}e^{-\frac{2k\pi}{\hbar}d_1}.$$

Computing the sums in the above expression establishes the estimate (by using the sum of a geometric progression)

$$\left|T_\hbar(z)-\int_{\mathbb{R}}f(\xi)d\xi\right|\leq\frac{\mathcal{N}^{+}}{e^{\frac{2\pi}{\hbar}d_2}-1}+\frac{\mathcal{N}^{-}}{e^{\frac{2\pi}{\hbar}d_1}-1}, \forall z\in[0,\hbar].$$

We have the proof by choosing $z=0$. ■
Furthermore, we have

$$\begin{aligned}|E_M(f,\hbar)| &= \Big|\int_{\mathbb{R}}f(\xi)d\xi-\hbar\sum_{k=-M}^{M}f(\xi_k)\Big|\\ &\leq \Big|\int_{\mathbb{R}}f(\xi)d\xi-\hbar\sum_{k=-\infty}^{+\infty}f(\xi_k)\Big|+\Big|\hbar\sum_{|k|=M+1}^{+\infty}f(\xi_k)\Big|.\end{aligned}$$

The truncation error can be evaluated as follows

$$\Big|\hbar\sum_{|k|=M+1}^{+\infty}f(\xi_k)\Big| \leq \hbar\sum_{|k|=M+1}^{+\infty}|f(\xi_k)|. \tag{3.14}$$

Thus the global quadrature error estimate are

$$|E_M(f,\hbar)| \leq \frac{\mathcal{N}^{+}}{e^{\frac{2\pi}{\hbar}d_2}-1}+\frac{\mathcal{N}^{-}}{e^{\frac{2\pi}{\hbar}d_1}-1}+\hbar\sum_{|k|=M+1}^{+\infty}|f(\xi_k)|.$$

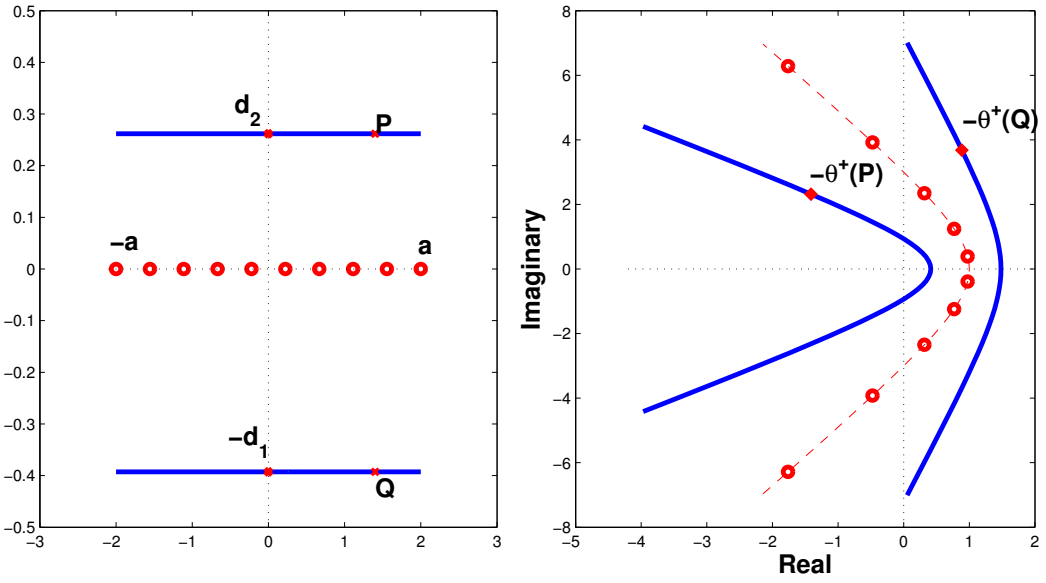


FIGURE 3.1 – D_d and $-\theta^+(D_d)$ for $\theta_h^+ = 2, \delta = \frac{\pi}{4}, \alpha = \frac{\pi}{6}$ and $d_1 = d_2 = \frac{\pi}{8}$.

As mentioned in Chapter 1, we only use the hyperbolic contours

$$-\theta^{\pm}(\xi) = \theta_h^{\pm}(1 + \sin(i\xi - \alpha^{\pm})),\ \xi \in \mathbb{R}, \theta_h^{\pm} > 0. \tag{3.15}$$

This contour allows for singularities with an unbounded imaginary part, provided they lay in a sectorial region around the negative real axis. It is clear that $-\theta^{\pm}(\xi)$ transforms the strip D_d into a region laying outside the sector Σ_δ (see Figure 3.1).

Theorem 2 is applied to evaluate each of the Laplace inversions which has the form as

$$f(x) = \frac{1}{2\pi i}\int_{\mathbb{R}} e^{\theta(\xi)x} F(\theta(\xi))\theta'(\xi)d\xi. \tag{3.16}$$

Therefore, we should find the conditions on the parameter d in Theorem 2 and the conditions on the parameters of the contours. We start from

$$\theta^{\pm}(w) = \theta_h^{\pm}(1 + \sin(iw - \alpha^{\pm})), \tag{3.17}$$

which is identical to (3.15) when w is real. The image of the horizontal line $w = \xi + id_2$ is

$$\theta^{\pm}(w) = \theta_h^{\pm}(1 - \sin(\alpha^{\pm} + d_2)\cosh\xi) + i\theta_h^{\pm}\cos(\alpha^{\pm} + d_2)\sinh\xi,$$

which can be expressed as the hyperbola

$$\left(\frac{\theta_h^{\pm} - \xi}{\sin(\alpha^{\pm} + d_2)}\right)^2 - \left(\frac{y}{\cos(\alpha^{\pm} + d_2)}\right)^2 = \theta_h^{\pm 2},\ \theta^{\pm} = \xi + iy. \tag{3.18}$$

When d_2 increases from 0, the hyperbola closes and degenerates into the negative real axis when d_2 reaches the value $\frac{\pi}{2} - \alpha^{\pm}$. Therefore, the condition $d_2 < \frac{\pi}{2} - \alpha^{\pm}$ guarantee f to admit a holomorphic extension in the strip $0 < y < d_2$.

We examine the lower half of the w-plane by considering $-d_1 < 0$. When $-d_1$ decreases from 0 the hyperbola widens until it reaches the limiting value $-d_1 = -\alpha^{\pm}$, at which point the image in the z-plane becomes a vertical line. For Theorem 2 to remain valid, one cannot decrease $-d_1$ beyond this point, since the factor $e^{\theta^{\pm}x}$ then grows unboundedly. Therefore we take $-d_1 = -\alpha^{\pm}$, or $d_1 = \alpha^{\pm}$. Thus the condition of $d = (d_1, d_2)$ is

$$d_1 < \alpha^{\pm}, \text{ and } d_2 < \frac{\pi}{2} - \alpha^{\pm}.$$

It means that the condition of $\alpha^{\pm}$ is

$$d_1 < \alpha^{\pm} < \frac{\pi}{2} - d_2. \tag{3.19}$$

3.2/ STATEMENT OF THE PROBLEM AND RESULTS

We consider the kernel operator

$$Pu(x) = \int_0^1 p(x,y)u(y)dy,$$

and its unique decomposition $Pu = z^+ + z^-$ into causal and anti-causal parts

$$z^+(x) = \int_0^x p(x,y)u(y)dy \text{ and } z^-(x) = \int_x^1 p(x,y)u(y)dy. \tag{3.20}$$

To achieve the final approximation (1.17), a process can be considered as follows. Firstly, the realizations $z^{\pm}$ are approximated with respect to x as

$$z^{h+}(x) = \int_0^x p(x,y)\widetilde{u}(y)dy \text{ and } z^{h-}(x) = \int_x^1 p(x,y)\widetilde{u}(y)dy, \tag{3.21}$$

where $\widetilde{u}$ is a piecewise constant approximation of u equal to $u(x_\ell)$ on each interval $[x_\ell, x_{\ell+1}), x_\ell = \ell h$. The realizations of $z^{h\pm}$ are formulated, thanks to the diffusive representation, in the form

$$z^{h\pm}(x) = \int_{\mathbb{R}} \mu^{\pm}(x,\xi)\psi^{h\pm}(x,\xi)d\xi, \tag{3.22}$$

where $\psi^{h\pm}$ are defined by

$$\begin{aligned} &\partial_x\psi^{h+}(x,\xi) = -\theta^+(\xi)\psi^{h+}(x,\xi) + \widetilde{u}(x), \forall x \in (0,1), \psi^{h+}(0,\xi) = 0 \\ &\text{and } \partial_x\psi^{h-}(x,\xi) = \theta^-(\xi)\psi^{h-}(x,\xi) + \widetilde{u}(x), \forall x \in (0,1), \psi^{h-}(1,\xi) = 0. \end{aligned} \tag{3.23}$$

Secondly, the diffusive symbols are approximated by $\mu^{N\pm}$ defined as in (1.11) which is based on a dedicated spectral approximation of the kernel p. The diffusive realizations are approximated by

$$z^{N,h\pm}(x) = \int_{\mathbb{R}} \mu^{N\pm}(x,\xi)\psi^{h\pm}(x,\xi)d\xi. \tag{3.24}$$

Finally, the approximations of $z^{\pm}$ are given by (1.17), namely

$$z_\hbar^{N,h\pm}(x) = \hbar \sum_{k=-M}^{M} \mu^{N\pm}(x,\xi_k)\psi^{h\pm}(x,\xi_k), \tag{3.25}$$

where the integrals $\int_{\mathbb{R}}$ are evaluated thanks to the trapezoidal rule with $2M+1$ quadrature nodes regularly spaced at the distance $\hbar$.
Our problem is to evaluate the global error estimate in L^j-norm, with $j = 1, 2$,

$$\|e\|_{L^j(0,1)} := \left(\int_0^1 e(x)^j dx \right)^{\frac{1}{j}}, \text{ with } j = 1, 2, \tag{3.26}$$

where $e(x) = |z(x) - z_\hbar^{N,h}(x)|$. This error is bounded by the sum $e^h(x) + e^{N,h}(x) + e_\hbar^{N,h}(x)$ of the error $e^h(x) = |z(x) - z^h(x)|$ of quadrature in x, the error $e^{N,h}(x) = |z^h(x) - z^{N,h}(x)|$ of the approximation of the symbols $\mu^\pm$ based on approximating the kernel by a spectral method and the error $e_\hbar^{N,h}(x) = |z^{N,h}(x) - z_\hbar^{N,h}(x)|$ of quadrature in ξ. Here $z = z^+ + z^-$, $z^h = z^{h+} + z^{h-}$, $z^{N,h} = z^{N,h+} + z^{N,h-}$ and $z_\hbar^{N,h} = z_\hbar^{N,h+} + z_\hbar^{N,h-}$. Thus the L^j-norm of the global error, with $j = 1, 2$, is

$$\|e\|_{L^j(0,1)} \leq \|e^h\|_{L^j(0,1)} + \|e^{N,h}\|_{L^j(0,1)} + \|e_\hbar^{N,h}\|_{L^j(0,1)}.$$

The error of quadrature in x states as follows.

Theorem 3 :

For a piecewise constant approximation of u,

i) if $u \in C^1(0,1)$ and $p \in L^1(\Omega)$, then

$$\|e^h\|_{L^1(0,1)} \leq h \times \max_{y\in[0,1]} |u'| \times \|p\|_{L^1(\Omega)},$$

ii) if $u \in H^1(0,1)$ and $p \in L^2(\Omega)$, then

$$\|e^h\|_{L^2(0,1)} \leq \frac{\sqrt{2}}{2} h \times \|u'\|_{L^2(0,1)} \times \|p\|_{L^2(\Omega)}.$$

The error estimate $e^{N,h}$ from approximating the kernel by a spectral method is assumed to be small enough. Then we ignore it in our estimate. Theorem 4 below provides the error estimate of quadrature in ξ. Before stating it, we give some definitions. We define

$$g^{N,h\pm}(x,\xi,t) = e^{-\theta^\pm(\xi)|x-t|} \frac{(e^{-\theta^\pm(\xi)h} - 1)}{h} \frac{\mu^{N\pm}(x,\xi)}{-\theta^\pm(\xi)}, \tag{3.27}$$

$$g_0^{N,h\pm}(x,\xi,t) = e^{\theta_h^\pm|x-t|}(1 + L(\sin|x-t|)) \frac{(e^{-\theta^\pm(\xi)h} - 1)}{h} \frac{\mu^{N\pm}(x,\xi)}{-\theta^\pm(\xi)}, \tag{3.28}$$

where the function L is defined by

$$L(x) = 1 + |\ln(1 - e^{-x})|, x > 0. \tag{3.29}$$

Notice that L is decreasing, $L(x) \to 1$ as $x \to +\infty$ and $L(x) \sim |\ln x|$ as $x \to 0^+$. To be convenient, we give an elementary lemma provided in Lopez [20]

Lemma 3 :

There hold :

$$(1) \qquad \int_0^{+\infty} e^{-\gamma \cosh x} dx \leq L(\gamma),\ \gamma > 0.$$

$$(2) \qquad \int_r^{+\infty} e^{-\gamma \cosh x} dx \leq (1 + L(\gamma)) e^{-\gamma \cosh r},\ r > 0, \gamma > 0.$$

We also define the functions

$$f^{\pm}(x,t) \quad = \quad \sum_{\ell=1}^{2} \int_{\mathbb{R}} |g^{N,h\pm}(x, \xi + i(-1)^{\ell} d_{\ell}, t)| d\xi + \max_{\xi \geq M\hbar} |g_0^{N,h\pm}(x,\xi,t)|.$$

Remark 13 :

(i) We note that $f^{\pm}$ are bounded for all $x \neq t$ since $g^{N,h\pm}$ decrease exponentially.

(ii) Since the kernel p is approximated by Legendre polynomials which are analytic, the diffusive symbols $\mu^{N\pm}$ are analytic. Therefore, we can apply Theorem 2 for our analytic function on the strip D_d. Namely,

$$\begin{aligned} e^{N,h\pm}(x,t) \quad &:= \quad \Big| \int_{\mathbb{R}} g^{N,h\pm}(x,\xi,t) d\xi - \hbar \sum_{k=-M}^{M} g^{N,h\pm}(x,\xi_k,t) \Big| \qquad (3.30) \\ &\leq \quad \sum_{\ell=1}^{2} \frac{\int_{\mathbb{R}} |g^{N,h\pm}(x, \xi + i(-1)^{\ell} d_{\ell}, t)| d\xi}{e^{2\pi d_{\ell}/\hbar} - 1} + \hbar \sum_{|k|=M+1}^{+\infty} |g^{N,h\pm}(x,\xi_k,t)| \\ &\leq \quad K_{h,\alpha}(\hbar) \times f^{\pm}(x,t), \end{aligned}$$

where $K_{h,\alpha}(\hbar)$ is defined by

$$K_{h,\alpha}(\hbar) := \max\Big\{ \frac{1}{e^{2\pi d_1/\hbar} - 1}, \frac{1}{e^{2\pi d_2/\hbar} - 1}, e^{-h \sin\alpha \cosh(M\hbar)} \Big\}. \qquad (3.31)$$

As in Weideman and Trefethen [35], we state a sufficient condition for such estimates to exist.

Assumption 1 :

For each x, t, h,

(i) the functions $w \mapsto g^{N,h\pm}(x, \xi + iw, t)$ are analytic in the strip $-d_1 < w < d_2$ for some $d_1 > 0$ and $d_2 > 0$,

(ii) $g^{N,h\pm}(x, \xi + iw, t)$ tend to 0 uniformly with respect to ξ as $|\xi + iw| \to +\infty$ in that strip.

For simplicity's sake, we only use the same contour $\theta^+ = \theta^- = \theta$, and we define

$$F_\hbar(a,\theta_h,\alpha) = h\sum_{n=1}^{H}\int_{x_n}^{x_{n+1}}\left(\sum_{j\in J_n^+} f^+(x,x_{j+1}) + \sum_{j\in J_n^-} f^-(x,x_j)\right)dx. \tag{3.32}$$

and

$$G_\hbar^2(a,\theta_h,\alpha) = h^2\sum_{n=1}^{H}\int_{x_n}^{x_{n+1}}\left(\sum_{j\in J_n^+} f^+(x,x_{j+1})^2 + \sum_{j\in J_n^-} f^-(x,x_j)^2\right)dx.$$

where $J_n^\pm$ are defined in Section 1.1.3.5 of Chapter 1.

Theorem 4 :

Assume that Assumption 1 is satisfied, and

i) if $u \in C^0(0,1)$ then

$$\|e_\hbar^{N,h}\|_{L^1(0,1)} \le \max_{y\in[0,1]}|u(y)| \times F_\hbar(a,\theta_h,\alpha) \times K_{h,\alpha}(\hbar).$$

ii) if $u \in L^2(0,1)$ then

$$\|e_\hbar^{N,h}\|_{L^2(0,1)} \le \max_{y\in[0,1]}|u(y)|^2 \times G_\hbar(a,\theta_h,\alpha) \times K_{h,\alpha}(\hbar).$$

where $F_\hbar, G_\hbar$ are bounded functions defined above and $K_{h,\alpha}$ is defined in (3.31).

3.3/ PROOFS

3.3.1/ PROOF OF THEOREM 3

Before proving it, we give an elementary lemma.

Lemma 4 :

Assume that $\widetilde{u}$ is a piecewise constant approximation of the input function u,

i) if we assume that $u \in C^1(0,1)$ then

$$|u(y) - \widetilde{u}(y)| \le h \times \max_{y\in[0,1]}|u'(y)|,$$

ii) if we assume that $u \in H^1(0,1)$ then

$$\|u - \widetilde{u}\|_{L^2(0,1)} \le \frac{\sqrt{2}}{2} h \times \|u'\|_{L^2(0,1)}.$$

Proof of Lemma 4 :

i) Using the mean value theorem, $\forall y \in (x_i, x_{i+1})$:

$$\begin{aligned} |u(y) - \widetilde{u}(y)| &= |u(y) - u(x_i)| = |(y-x_i)u'(c)|,\ c \in (x_i, y) \\ &\le h \times \sup_{c\in(x_i,x_{i+1})}|u'(c)| \le h \times \max_{y\in[0,1]}|u'(y)|. \end{aligned}$$

ii) Since $u \in H^1(0,1),\ \forall y \in (x_i, x_{i+1})$,

$$\begin{aligned} |u(y) - u(x_i)|^2 &= \Big| \int_{x_i}^{y} u'(t)dt \Big|^2 \leq \int_{x_i}^{y} 1dt \int_{x_i}^{y} |u'(t)|^2 dt \\ &\leq |y - x_i| \int_{x_i}^{x_{i+1}} |u'(t)|^2 dt. \end{aligned}$$

Moreover,

$$\begin{aligned} \|u - \widetilde{u}\|^2_{L^2(0,1)} &:= \int_0^1 |u(y) - \widetilde{u}(y)|^2 dy = \sum_{i=0}^{H-1} \int_{x_i}^{x_{i+1}} |u(y) - u(x_i)|^2 dy \\ &\leq \sum_{i=0}^{H-1} \int_{x_i}^{x_{i+1}} |y - x_i| dy \int_{x_i}^{x_{i+1}} |u'(t)|^2 dt = \frac{h^2}{2} \sum_{i=0}^{H-1} \int_{x_i}^{x_{i+1}} |u'(t)|^2 dt \\ &\leq \frac{h^2}{2} \|u'\|^2_{L^2(0,1)}. \end{aligned}$$

Therefore,

$$\|u - \widetilde{u}\|_{L^2(0,1)} \leq \frac{\sqrt{2}}{2} h \times \|u'\|_{L^2(0,1)}. \blacksquare$$

Returning to the proof of Theorem 3, setting

$$\|e^h\|_{L^1(0,1)} = \int_0^1 |z^+(x) + z^-(x) - z^{h+}(x) - z^{h-}(x)| dx.$$

Using the formulas (3.20) and (3.21),

$$\begin{aligned} \|e^h\|_{L^1(0,1)} &= \int_0^1 \Big| \int_0^1 p(x,y)(u(y) - \widetilde{u}(y)) dy \Big| dx \\ &\leq \int_0^1 \int_0^1 |p(x,y)| \times |u(y) - \widetilde{u}(y)| dy dx. \end{aligned}$$

By using *i)* of Lemma 4, it leads to

$$\|e^h\|_{L^1(0,1)} \leq h \times \max_{y \in [0,1]} |u'(y)| \int_0^1 \int_0^1 |p(x,y)| dy dx.$$

Similarly,

$$\begin{aligned} \|e^h\|^2_{L^2(0,1)} &= \int_0^1 |z^+(x) + z^-(x) - z^{h+}(x) - z^{h-}(x)|^2 dx. \\ &= \int_0^1 \Big| \int_0^1 p(x,y)(u(y) - \widetilde{u}(y)) dy \Big|^2 dx \\ &\leq \int_0^1 \left(\int_0^1 |p(x,y)|^2 dy \int_0^1 |u(y) - \widetilde{u}(y)|^2 dy \right) dx. \end{aligned}$$

By using *ii)* of Lemma 4, it leads to

$$\|e^h\|^2_{L^2(0,1)} \leq \frac{h^2}{2} \|u'\|^2_{L^2(0,1)} \int_0^1 \int_0^1 |p(x,y)|^2 dy dx. \quad \blacksquare$$

3.3.2/ PROOF OF THEOREM 4

For any $x \in [x_n, x_{n+1})$, the formula $\psi^{h\pm}(x,\xi)$ can be rewritten as

$$\psi^{h\pm}(x,\xi) = (\alpha^{\pm}(\xi) - 1)) \sum_{j \in J_n^{\pm}} u_{j+\frac{1}{2}} \frac{e^{-\theta^{\pm}(\xi)(x-x_j)}}{-\theta^{\pm}(\xi)},$$

where $\alpha^{\pm}(\xi) = e^{-\theta^{\pm}(\xi)h}$ and $J_n^{\pm}$ are defined as in Section 1.1.3.5 of Chapter 1. Therefore, for any $x \in [x_n, x_{n+1})$,

$$z^{N,h\pm}(x) = \pm h \sum_{j \in J_n^{\pm}} u_{j+\frac{1}{2}} \int_{\mathbb{R}} \frac{\alpha^{\pm}(\xi) - 1}{h} \frac{e^{-\theta^{\pm}(\xi)(x-x_j)}}{-\theta^{\pm}(\xi)} \mu^{N\pm}(x,\xi) d\xi.$$

Moreover, for any $x \in [x_n, x_{n+1})$,

$$\begin{aligned} e_{\hbar}^{N,h}(x) &= \left| z^{N,h+}(x) - z_{\hbar}^{N,h+}(x) + z^{N,h-}(x) - z_{\hbar}^{N,h-}(x) \right| \\ &\leq h \sum_{j \in J_n^{+}} |u_{j+\frac{1}{2}}| e^{N,h+}(x, x_j) + h \sum_{j \in J_n^{-}} |u_{j+\frac{1}{2}}| e^{N,h-}(x, x_j), \end{aligned}$$

where $e^{N,h\pm}$ are defined in Remark 13. To evaluate these sub-error estimate (3.30), we apply Theorem 2 for the functions $g^{N,h\pm}(x,\xi,t)$ as in Remark 13, namely

$$e^{N,h\pm}(x,t) \leq f^{\pm}(x,t) \times K_{h,\alpha}(\hbar). \tag{3.33}$$

Returning to the proof of this theorem, we consider the error of quadrature in ξ in each norm.

i) Using the L^1-norm : For any $x \in [x_n, x_{n+1})$, we have

$$e_{\hbar}^{N,h}(x) \leq \max_{y \in [0,1]} |u(y)| \times \left(\sum_{j \in J_n^{+}} h e^{N,h+}(x, x_{j+1}) + \sum_{j \in J_n^{-}} h e^{N,h-}(x, x_j) \right).$$

Therefore, the error estimate in ξ using the L^1-norm is

$$\begin{aligned} \|e_{\hbar}^{N,h}\|_{L^1(0,1)} &:= \int_0^1 e_{\hbar}^{N,h}(x) dx = \sum_{n=1}^{H} \int_{x_n}^{x_{n+1}} e_{\hbar}^{N,h}(x) dx \\ &\leq \max_{y \in [0,1]} |u(y)| \sum_{n=1}^{H} \int_{x_n}^{x_{n+1}} \left(\sum_{j \in J_n^{+}} h e^{N,h+}(x, x_j) + \sum_{j \in J_n^{-}} h e^{N,h-}(x, x_j) \right) dx. \end{aligned} \tag{3.34}$$

Thus, combining (3.33) and (3.32),

$$\|e_{\hbar}^{N,h}\|_{L^1(0,1)} \leq \max_{y \in [0,1]} |u(y)| \times F_{\hbar}(a, \theta_h, \alpha) \times K_{h,\alpha}(\hbar). \tag{3.35}$$

ii) Using L^2-norm : Using another inequality, we have another error estimate

$$\begin{aligned} e_{\hbar}^{N,h}(x)^2 &= \left| z^{N,h+}(x) - z_{\hbar}^{N,h+}(x) + z^{N,h-}(x) - z_{\hbar}^{N,h-}(x) \right|^2 \\ &\leq h^2 \sum_{k=1}^{H} |u_{k+\frac{1}{2}}|^2 \left(\sum_{j \in J_n^{+}} e^{N,h+}(x, x_{j+1})^2 + \sum_{j \in J_n^{-}} e^{N,h-}(x, x_j)^2 \right), \end{aligned}$$

So the error estimate in ξ using L^2-norm is

$$\begin{aligned}\|e_\hbar^{N,h}\|^2_{L^2(0,1)} &:= \int_0^1 e_\hbar^{N,h}(x)^2 dx \\ &= \sum_{n=1}^{H}\int_{x_n}^{x_{n+1}} e_\hbar^{N,h}(x)^2 dx \\ &\leq h\sum_{k=1}^{H}|u_{k+\frac{1}{2}}|^2 \sum_{n=1}^{H}\int_{x_n}^{x_{n+1}}\left(h\sum_{j\in J_n^+} e^{N,h+}(x,x_j)^2 + h\sum_{j\in J_{n+1}^-} e^{N,h-}(x,x_j)^2\right)dx.\end{aligned} \tag{3.36}$$

Furthermore,

$$h\sum_{k=1}^{H}|u_{k+\frac{1}{2}}|^2 \leq \max_{y\in[0,1]}|u(y)|^2.$$

Thus

$$\|e_\hbar^{N,h}\|_{L^2(0,1)} \leq \max_{y\in[0,1]}|u(y)|^2 \times G_\hbar(a,\theta_h,\alpha)\times K_{h,\alpha}(\hbar).\blacksquare$$

3.4/ NUMERICAL SIMULATIONS

We recall the model problem used in Section 1.2 of Chapter 1 : the operator P is solution to the Lyapunov equation,

$$\frac{d^2}{dx^2}Pu + P\frac{d^2}{dx^2}u = Qu \tag{3.37}$$

in $\omega = (0,1)$, for all u vanishing at the boundary, and where Q is a $L^2(\omega)$-bounded linear kernel operator. This problem originates in optimal filtering or control theory of the heat equation, $\frac{\partial T}{\partial t} - \frac{\partial^2 T}{\partial x^2} = q$ in ω with Dirichlet boundary conditions. We recall that the kernel p is the unique solution to the boundary value problem $\Delta p = q$ in the square Ω, $p = 0$ on the boundary and q is the kernel of Q.

We compare the relative error between the direct quadrature method,

$$Pu(x_n) = h\sum_{j=0}^{H} p(x_n,y_j)u(y_j), \text{ for all } n. \tag{3.38}$$

and the DR method with optimal parameters.

$$F_\hbar(a^*,\theta_h^*,\alpha^*) = \min_{\substack{a\in(0,\infty)\\ \theta_h\in(0,\infty)\\ \alpha\in(d_1,\pi/2-d_2)}} F_\hbar(a,\theta_h,\alpha). \tag{3.39}$$

The above Approximation of the Diffusive Realization (ADR) with the contour optimization is compared to a Direct Spectral method (DS) using the usual Legendre polynomials to estimate p and a trapezoidal rule for z. Experiments have been carried out for two kernels q, one kernel p and two inputs u. The kernels are a two-dimensional Gaussian density,

$$q_1(x,y) = \frac{C}{\sigma^2 2\pi}e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}},$$

with $\sigma = 0.2$, $x_0 = y_0 = 0.4$, $C = 10$, an oscillating function,

$$q_2(x, y) = 2a^2\pi^2 \sin(b\pi x) \sin(a\pi y),$$

with $a = 3$ and $b = 5$, and a function p_3 satisfying to $\Delta p_3 = q_3$,

$$\begin{aligned} p_3(x, y) &= x(x-1)(2\sqrt{10}x - \sqrt{10} + 2\sqrt{6}) \\ &\quad \times \left(20e^{3y-3} - 24e^{2y-2} + (8 - 2w_1)e^{y-1} + w_0 + w_1\right), \end{aligned} \tag{3.40}$$

with $w_0 = \frac{10-12e+2e^3}{e^2-e^3}, w_1 = \frac{10-12e+4e^2-2e^3}{e^2-e^3}$. Figure 3.2 shows these kernels in $\mathbb{R}^3$. The inputs are both oscillating functions,

$$u_1(y) = 3y \ \sin(6\pi y)^2 \text{ and } u_2(y) = 15 \sin(4\pi y^4).$$

Figure 3.3 shows these inputs in $\mathbb{R}^2$. The function p_3 is chosen since the error on μ^+ is equal to 0 and the error on μ^- is small enough with only 5×5 polynomials. The errors are evaluated using a reference solution z^{ref} computed with the DS method with 30×30 polynomials (except the case p_3 with only 5×5 polynomials) and with inputs discretized with 1,024 nodes. Three parameters of discretization are used, the number N of Legendre polynomials in each x- and y-directions to discretize the kernel p, the number H of discretization points x_ℓ of the input, and the number M of quadrature nodes along the contours. The contour parameters are chosen as the solution of the optimization problem (3.39). The discrete outputs z_ℓ are evaluated at the same nodes as the input. The error function is evaluated in the discrete ℓ^2-norm evaluated at the nodes x_ℓ of the input mesh

$$e = \frac{\left(\sum_{\ell=1}^{H} (z^{ref}(x_\ell) - z_\ell)^2\right)^{1/2}}{\left(\sum_{\ell=1}^{H} (z^{ref}(x_\ell))^2\right)^{1/2}}. \tag{3.41}$$

Four values $N = 10, 15, 20$ and 25 have been tested when the other parameters are very large, $H = 1,024$ and $M = 100$. The DS method yields errors varying from 10^{-6} to 10^{-12} for q_1 and from 10^{-2} to 10^{-15} for q_2. For the ADR they vary from 10^{-5} to 10^{-6} for q_1 and from 10^{-1} to 10^{-3} for q_2 when N varies from 10 to 20, and then increases. This lack of convergence may come from the operations of extension of $p^{\pm}$ which exhibits oscillations and also from the very high values of coefficients of the Legendre polynomials yielding large truncation errors in the Laplace domain. In the following experiment, we consider that $N = 20$ is the optimal value for the ADR. The input p_3 yields errors 10^{-10} with the value $N = 5$ for both the DS method and the ADR, so we didn't test more with N.

To test the effect of the number H of discretization points in the input, experiments have been carried out with $H = 16, 32, 64, 128$ and 256 while $N = 20$ ($N = 5$ for p_3) and $M = 100$. For both q_1 and p_3 (resp. q_2), the minimal error of 10^{-6} (resp. 10^{-3}) is reached for the ADR with $H = 256$ (resp. $H = 128$). So, for H varying between 16 and 256 (resp. 128) the errors of the ADR and of the DS method change from 10^{-3} to 10^{-6} and from 10^{-4} to 10^{-9} (resp. from 10^{-2} to 10^{-3} and from 10^{-2} to 10^{-6}). In all cases, we observe a better decay of the error for the DS method with respect to H than for the ADR. The error of the DS method is with infinite order, whereas for the ADR method the error follows the one of a quadrature method.

Three values $M = 20$, 40 and 60 have been compared when $N = 20$ and $H = 1,024$. The ADR method has errors varying between 10^{-3} to 10^{-6} for q_1, p_3 and between 10^{-2} to 10^{-3} for q_2. This confirms a very fast decay of the error with respect to the number of

quadrature points in the contours. For the ADR method, which involves complex numbers, a significant gain in terms of computation cost only exists if the number M can be as small as discussed in Chapter 2. With an optimization of the contours to solve the optimization problem (3.39), for $M = 10$, errors in the range of 10^{-3} for q_1, p_3 and of 10^{-2} for q_2 have been reached with the ADR method.

Figure 3.4 (resp. 3.5, 3.6) shows a contour plot of the relative error with $q_1(x, y)$ and $u_1(y)$ in the (θ_h, α) (resp. (a, θ_h), (a, α))-parameter plane. Similarly, Figure 3.7 (resp. 3.8, 3.9) shows a contour plot of the relative error with $q_2(x, y)$ and $u_1(y)$ in the (θ_h, α) (resp. (a, θ_h), (a, α))-parameter plane. Figure 3.10 (resp. 3.11, 3.12) shows a contour plot of the relative error with $p_3(x, y)$ and $u_1(y)$ in the (θ_h, α) (resp. (a, θ_h), (a, α))-parameter plane. Darker shades represent higher accuracy. The cross represents the solution to the optimization problem (3.39) with $d_1 = d_2 = 0.5$, $h = 0.02$. These figures show that the optimal parameters of the contour are very effective.

In total, the new extension of $p^{\pm}$ yields a viable numerical method that might be used when a limited precision is required, as, for example, applications in real-time distributed control. Limitations of the ADR method have been well identified.

3.5/ OTHER APPROACHES OF ERROR ESTIMATES

In this section, we discuss other approaches to mainly evaluate the error of quadrature in ξ. Subsection 3.5.1 only shows the discretization error based on an assumption that the diffusive symbols belong to $L^2(\mathbb{R})$ and are analytic in the strip D_d. Subsection 3.5.2 presents the global error based on an assumption that the diffusive symbols are bounded. In Subsection 3.5.3, another approach based on the balancing idea of Weideman and Trefethen provides an error estimate and its application to find optimal parameters of the contours.

3.5.1/ DISCRETIZATION ERROR ESTIMATE IN ξ

3.5.1.1/ STATEMENT OF THE PROBLEM AND RESULTS

We recall the diffusive realization of $z^{\pm}$ defined as in (1.10)

$$z^{\pm}(x) = \int_{\mathbb{R}} \mu^{\pm}(x, \xi)\psi^{\pm}(x, \xi)\, d\xi,$$

Ignoring the error from approximating the kernel by a spectral method, the diffusive realization thus are

$$z^{N\pm}(x) \quad = \quad \int_{\mathbb{R}} \mu^{N\pm}(x, \xi)\psi^{\pm}(x, \xi) d\xi. \tag{3.42}$$

where $\mu^{N\pm}$ is defined as in (1.11). These integrals are approximated by the trapezoidal rule

$$z_{\hbar}^{N\pm}(x) = \hbar \sum_{k=-\infty}^{\infty} \mu^{N\pm}(x, \xi_k)\psi^{\pm}(x, \xi_k). \tag{3.43}$$

In this subsection, our problem is to mainly evaluate the error of quadrature in ξ, namely

$$\|E_\hbar^\pm\|_{L^1(0,1)} = \int_0^1 E_\hbar^\pm(x)\,dx \text{ with } E_\hbar^\pm(x) = |z^{N\pm}(x) - z_\hbar^{N\pm}(x)|.$$

To estimate this error, we improve the following condition.

Assumption 2 :

For all $x \in [0,1]$,

(i) the mappings $\xi \mapsto \mu^{N\pm}(x, \xi + id)$ and $\xi \mapsto \psi^\pm(x, \xi + id)$ are in $L^2(\mathbb{R})$.

(ii) the function $w \mapsto \mu^{N\pm}(x, \xi + iw)\psi^\pm(x, \xi + iw)$ are analytic in the strip $-d_1 < w < d_2$ for some $d_1 > 0$ and $d_2 > 0$.

Thanks to Theorem 2 and Assumption 2, we have

$$E_\hbar^\pm(x) \quad \leq \quad \sum_{\ell=1}^{2} \frac{\int_{\mathbb{R}} |g^\pm(x, \xi + i(-1)^\ell d_\ell)| d\xi}{e^{2\pi d_\ell/\hbar} - 1}, \tag{3.44}$$

where

$$g^\pm(x,\xi) \quad = \quad \mu^{N\pm}(x,\xi)\psi^\pm(x,\xi).$$

We got the error estimate of quadrature in ξ.

Theorem 5 :

Assume that $u \in L^2(0,1)$, $\mathfrak{p}^\pm \in L^2(0,1)$, and the hereafter expressions $\mathcal{I}^\pm$, $\mathcal{J}^\pm$ are bounded. Furthermore, if the assumption 2 is fulfilled then

$$\|E_\hbar^\pm\|_{L^1(0,1)} \quad \leq \quad \sum_{\ell=1}^{2} \frac{\|u\|_{L^2(0,1)}}{e^{2\pi d_\ell/\hbar} - 1} \|\mathfrak{p}^\pm\|_{L^2(0,1)}^{1/2} \mathcal{I}^\pm(\theta_h^\pm, \alpha^\pm, (-1)^\ell d_\ell)^{1/2} \mathcal{J}^\pm(\theta_h^\pm, \alpha^\pm, (-1)^\ell d_\ell)^{1/2},$$

where

$$\mathfrak{p}^\pm(x) \quad = \quad \sum_{\ell=0}^{N} \widehat{p_\ell^\pm}^2(x), \text{ and } \widehat{p_\ell^\pm} \text{ are defined in (1.24) ,} \tag{3.45}$$

$$\mathcal{I}^\pm(\theta_h^\pm, \alpha^\pm, d) \quad = \quad \int_{\mathbb{R}} \frac{2\Theta^\pm(\xi,d) - 1 + e^{-2\Theta^\pm(\xi,d)}}{4\Theta^\pm(\xi,d)^2}\, d\xi, \tag{3.46}$$

$$\begin{aligned} \mathcal{J}^\pm(\theta_h^\pm, \alpha^\pm, d) \quad &= \quad \frac{1}{4\pi^2} \int_{\mathbb{R}} |\theta^{\pm\prime}(\xi + id)|^2 \left(\int_0^1 \left(\sum_{\ell=0}^{N} \left| \zeta_\ell^\pm(x, -\theta^\pm(\xi + id)) \right|^2 \right)^2 dx \right)^{1/2} d\xi, \\ \text{and } \Theta^\pm(\xi, d) \quad &= \quad \Re(\theta^\pm(\xi + id)) = \theta_h^\pm[\sin(d + \alpha^\pm)\cosh\xi - 1]. \end{aligned} \tag{3.47}$$

3.5.1.2/ Proof of Theorem 5

We start from (3.44)

$$\|E_\hbar^\pm\|_{L^1(0,1)} \leq \sum_{\ell=1}^{2} \frac{1}{e^{2\pi d_\ell/\hbar}-1} \int_0^1 \int_{\mathbb{R}} |g^\pm(x,\xi+i(-1)^\ell d_\ell)| d\xi\, dx.$$

Assumption 2 is fulfilled, so

$$\left(\int_0^1 \int_{\mathbb{R}} |g^\pm(x,\xi+id)| d\xi dx\right)^2 \leq \int_0^1 \int_{\mathbb{R}} |\mu^{N\pm}(x,\xi+id)|^2\, d\xi dx \int_0^1 \int_{\mathbb{R}} |\psi^\pm(x,\xi+id)|^2\, d\xi dx.$$

But,

$$\begin{aligned} |\psi^+(x,\xi+id)|^2 &= \left|\int_0^x e^{-\theta^+(\xi)(x-y)} u(y) dy\right|^2 \\ &\leq e^{-2\Theta^+(\xi,d)x} \int_0^x e^{2\Theta^+(\xi,d)y}\, dy \times \|u\|^2_{L^2(0,x)} \\ &\leq \frac{1-e^{-2\Theta^+(\xi,d)x}}{2\Theta^+(\xi,d)} \times \|u\|^2_{L^2(0,1)}, \end{aligned}$$

thus

$$\begin{aligned} &\int_0^1 \int_{\mathbb{R}} |\psi^+(x,\xi+id)|^2\, d\xi dx \\ \leq\ & \|u\|^2_{L^2(0,1)} \times \int_{\mathbb{R}} \frac{1}{2\Theta^+(\xi,d)} \left(1 - \int_0^1 e^{-2\Theta^+(\xi,d)x}\, dx\right) d\xi \\ \leq\ & \|u\|^2_{L^2(0,1)} \times \int_{\mathbb{R}} \frac{2\Theta^+(\xi,d) - 1 + e^{-2\Theta^+(\xi,d)}}{4\Theta^+(\xi,d)^2} d\xi. \end{aligned}$$

The same calculation for ψ^-

$$\begin{aligned} |\psi^-(x,\xi+id)|^2 &= \left|\int_x^1 e^{\theta^-(\xi)(x-y)} u(y) dy\right|^2 \\ &\leq \int_x^1 e^{2\Theta^-(\xi,d)(x-y)}\, dy \times \|u\|^2_{L^2(x,1)} \\ &\leq \frac{1-e^{-2\Theta^-(\xi,d)(1-x)}}{2\Theta^-(\xi,d)} \times \|u\|^2_{L^2(0,1)}, \end{aligned}$$

thus

$$\begin{aligned} &\int_0^1 \int_{\mathbb{R}} |\psi^-(x,\xi+id)|^2\, d\xi dx \\ \leq\ & \|u\|^2_{L^2(0,1)} \times \int_{\mathbb{R}} \frac{1}{2\Theta^-(\xi,d)} \left(1 - \int_0^1 e^{-2\Theta^-(\xi,d)(1-x)}\, dx\right) d\xi \\ \leq\ & \|u\|^2_{L^2(0,1)} \times \int_{\mathbb{R}} \frac{2\Theta^-(\xi,d) - 1 + e^{-2\Theta^-(\xi,d)}}{4\Theta^-(\xi,d)^2}\, d\xi. \end{aligned}$$

Therefore,

$$\int_0^1 \int_{\mathbb{R}} |\psi^\pm(x,\xi+id)|^2\, d\xi dx \leq \mathcal{I}^\pm(\theta_h^\pm, \alpha^\pm, d) \times \|u\|^2_{L^2(0,1)}. \tag{3.48}$$

Using the decomposition of $\mu^{N\pm}$,

$$\int_0^1 \int_{\mathbb{R}} |\mu^{N\pm}(x,\xi+id)|^2 \, d\xi dx = \frac{1}{4\pi^2} \int_0^1 \int_{\mathbb{R}} \Big| \sum_{\ell=0}^{N} \widehat{p}_\ell^{\pm}(x)\theta^{\pm\prime}(\xi+id)\zeta_\ell^{\pm}(x,-\theta^{\pm}(\xi+id)) \Big|^2 \, d\xi dx$$

$$\leq \frac{1}{4\pi^2} \int_{\mathbb{R}} |\theta^{\pm\prime}(\xi+id)|^2 \int_0^1 \Big| \sum_{\ell=0}^{N} \widehat{p}_\ell^{\pm}(x)\zeta_\ell^{\pm}(x,-\theta^{\pm}(\xi+id)) \Big|^2 \, dx d\xi. \tag{3.49}$$

Since

$$\Big| \sum_{\ell=0}^{N} \widehat{p}_\ell^{\pm}(x)\zeta_\ell^{\pm}(x,-\theta^{\pm}(\xi+id)) \Big|^2 \leq \sum_{\ell=0}^{N} \widehat{p}_\ell^{\pm 2}(x) \sum_{\ell=0}^{N} \Big| \zeta_\ell^{\pm}(x,-\theta^{\pm}(\xi+id)) \Big|^2, \tag{3.50}$$

so

$$\int_0^1 \Big| \sum_{\ell=0}^{N} \widehat{p}_\ell^{\pm}(x)\zeta_\ell^{\pm}(x,-\theta^{\pm}(\xi+id)) \Big|^2 dx \leq \int_0^1 \mathfrak{p}^{\pm}(x) \sum_{\ell=0}^{N} \Big| \zeta_\ell^{\pm}(x,-\theta^{\pm}(\xi+id)) \Big|^2 dx$$

$$\leq \|\mathfrak{p}^{\pm}\|_{L^2(0,1)} \left(\int_0^1 \left(\sum_{\ell=0}^{N} \Big| \zeta_\ell^{\pm}(x,-\theta^{\pm}(\xi+id)) \Big|^2 \right)^2 dx \right)^{1/2}. \tag{3.51}$$

Combining the above results, we have the proof. ■

Conclusion : This approach has presented a theoretical discretization error estimate for quadrature in ξ. Although, this error has an exponential decay, its expressions is very complex. Moreover, the truncation error hasn't yet considered sufficiently. Thus it is difficult to apply in seeking optimal parameters of the contour.

3.5.2/ ERROR ESTIMATE BY APPLYING THE RESULTS OF LOPEZ [20]

Another way to evaluate error estimate is proposed in this subsection. The idea is to start from the relationship between DR approximation and inversion of Laplace transform which was presented in Subsubsection 1.1.3.5. Namely, the approximation of the DR can be written like a linear combination of inverse Laplace transforms. Therefore, the results of Lopez concerning the inversion of Laplace transform can be applied for each term in that linear combination.

3.5.2.1/ STATEMENT OF THE PROBLEMS AND RESULTS

Firstly, we recall the diffusive realization of $z^{N,h\pm}$ in (1.33)

$$z^{N,h\pm}(x_n) = \pm \sum_{j\in J_n^{\pm}} u_{j+\frac{1}{2}} \int_{\mathbb{R}} \left(\frac{e^{-\theta^{\pm}(\xi)(\pm t_{n,j})}}{-\theta^{\pm}(\xi)} - \frac{e^{-\theta^{\pm}(\xi)(\pm t_{n,j+1})}}{-\theta^{\pm}(\xi)} \right) \mu^{N\pm}(x_n,\xi) d\xi.$$

Each integral is approximated by the trapezoidal rule, so we get the approximation

$$z_\hbar^{N,h\pm}(x_n) = \pm \sum_{j\in J_n^{\pm}} u_{j+\frac{1}{2}} \hbar \sum_{k=-M}^{M} \left(e^{-\theta^{\pm}(\xi_k)(\pm t_{n,j})} - e^{-\theta^{\pm}(\xi_k)(\pm t_{n,j+1})} \right) \frac{\mu^{N\pm}(x_n,\xi_k)}{-\theta^{\pm}(\xi_k)}.$$

So the error estimate at x_n are defined by

$$e_{\hbar}^{N,h\pm}(x_n) \quad = \quad |z^{N,h\pm}(x_n) - z_{\hbar}^{N,h\pm}(x_n)|. \tag{3.52}$$

We notice that $\pm t_{n,j}, \pm t_{n,j+1} > 0$ for all $j \in J_n^{\pm}$. So we define

$$g_0^{N,h\pm}(x,\xi,t) = e^{-\theta^{\pm}(\xi)|x-t|}\frac{\mu^{N\pm}(x,\xi)}{-\theta^{\pm}(\xi)}. \tag{3.53}$$

Thus $z^{N,h\pm}(x_n)$ can be rewritten by

$$z^{N,h\pm}(x_n) = \pm \sum_{j\in J_n^{\pm}} u_{j+\frac{1}{2}} \int_{\mathbb{R}} \left(g_0^{N,h\pm}(x_n,\xi,x_j) - g_0^{N,h\pm}(x_n,\xi,x_{j+1})\right) d\xi.$$

Then we will obtain an accurate estimate for the quadrature error

$$e_0^{N,h\pm}(x,t) = \Big| \int_{\mathbb{R}} g_0^{N,h\pm}(x,\xi,t)d\xi - \hbar \sum_{k=-M}^{M} g_0^{N,h\pm}(x,\xi_k,t)\Big|, \tag{3.54}$$

of truncated trapezoidal rule. This estimate is split into the discretization error

$$E_D(x, g_0^{N,h\pm}, t) = \Big| \int_{\mathbb{R}} g_0^{N,h\pm}(x,\xi,t)d\xi - \hbar \sum_{k=-\infty}^{\infty} g_0^{N,h\pm}(x,\xi_k,t)\Big|, \tag{3.55}$$

and the truncation error

$$E_T(x, g_0^{N,h\pm}, t) = \Big|\hbar \sum_{|k|>M}^{\infty} g_0^{N,h\pm}(x,\xi_k,t)\Big|. \tag{3.56}$$

Assumption 3 :

For all $x \in [0,1]$, the function $\mu^{N\pm}(x,\xi)$ are bounded on $\mathbb{R}$, i.e., there exists a constant $C_x > 0$ such that

$$|\mu^{N\pm}(x,\xi)| \le C_x. \tag{3.57}$$

Theorem 6 below provides a estimate for $e_0^{N,h\pm}(x,t)$.

Theorem 6 :

For all $x,t \in (0,1)$ and $x \neq t$, if Assumption 3 is fulfilled then

$$\begin{aligned} e_0^{N,h\pm}(x,t) \quad \le \quad & \frac{2C_x e^{\theta_{\hbar}^{\pm}|x-t|}}{\pi(1-\sin(d_2+\alpha^{\pm}))} L(\theta_{\hbar}^{\pm}|x-t|\sin(\alpha^{\pm}-d_1)) \times \\ & \left(\frac{1}{e^{\frac{2\pi d}{\hbar}}-1} + \frac{1}{e^{\theta_{\hbar}^{\pm}|x-t|\sin\alpha^{\pm}}\cosh(M\hbar)}\right), \end{aligned} \tag{3.58}$$

where $d = \min\{d_1, d_2\}$.

Theorem 7 :

For all $x_n \in (0,1)$, if Assumption 3 is fulfilled then

$$e_\hbar^{N,h\pm}(x_n) \leq 2C_2|J_n^\pm| \max_{j\in J_n^\pm} e_0^{N,h\pm}(x_n, x_{j+\frac{1\mp1}{2}}), \tag{3.59}$$

where $C_2 = \max_{x\in[0,1]} |u(x)|$, $|J_n^+| = n$ and $|J_n^-| = H - n$.

3.5.2.2/ PROOF OF THEOREM 6

We note that $-\theta^\pm(\xi)$ are the hyperbolic contours, so

$$|g_0^{N,h+}(x,\xi+iy,t)| \leq \frac{e^{\theta_h^+|x-t|}}{2\pi}\frac{e^{-\theta_h^+|x-t|\sin(y+\alpha^+)\cosh\xi}}{\cosh\xi - \sin(y+\alpha^+)}C_x,$$

then

$$|g_0^{N,h\pm}(x,\xi,t)| \leq \frac{C_x e^{\theta_h^\pm|x-t|}}{2\pi(1-\sin\alpha^\pm)} e^{-\theta_h^\pm|x-t|\sin\alpha^\pm\cosh\xi}. \tag{3.60}$$

We apply the result (2) of Lemma 3 and we get the truncation error

$$\begin{aligned} |E_T(x, g_0^{N,h\pm}, t)| &\leq \hbar\sum_{|k|>M}^{\infty} |g_0^{N,h\pm}(x,\xi_k,t)| \leq 2\int_{M\hbar}^{\infty} |g_0^{N,h\pm}(x,\xi,t)|d\xi \\ &\leq \frac{2C_x e^{\theta_h^\pm|x-t|}(1+L(\theta_h^\pm|x-t|\sin\alpha^\pm))}{2\pi(1-\sin\alpha^\pm)} \times e^{-\theta_h^\pm|x-t|\sin\alpha^\pm\cosh(M\hbar)}. \end{aligned} \tag{3.61}$$

We also have

$$\begin{aligned} |g_0^{N,h\pm}(x,\xi+iy,t)| &\leq \frac{C_x e^{\theta_h^\pm|x-t|}e^{-\theta_h^\pm|x-t|\sin(y+\alpha^\pm)\cosh\xi}}{2\pi(1-\sin(y+\alpha^\pm))},\ y\in[-d_1,d_2] \\ &\leq \frac{C_x e^{\theta_h^\pm|x-t|}e^{-\theta_h^\pm|x-t|\sin(\alpha^\pm-d_1)\cosh\xi}}{2\pi(1-\sin(d_2+\alpha^\pm))}. \end{aligned} \tag{3.62}$$

Moreover, applying the result (1) of Lemma 3 and get

$$\begin{aligned} N(x, g_0^{N,h\pm}, D_d) &= 4\int_0^\infty |g_0^{N,h\pm}(x,\xi+iy,t)|d\xi \\ &\leq 4\frac{C_x e^{\theta_h^\pm|x-t|}L(\theta_h^\pm|x-t|\sin(\alpha^\pm-d_1))}{2\pi(1-\sin(d_2+\alpha^\pm))}. \end{aligned} \tag{3.63}$$

Thus

$$\begin{aligned} e_0^{N,h\pm}(x,t) \leq \frac{N(x,g_0^{N,h\pm},D_d)}{e^{\frac{2\pi d}{\hbar}}-1} + |E_T(x,g^\pm,\hbar)| \leq \frac{2C_x e^{\theta_h^\pm|x-t|}}{\pi(1-\sin(d_2+\alpha^\pm))} \times \\ \left(\frac{L(\theta_h^\pm|x-t|\sin(\alpha^\pm-d_1))}{e^{\frac{2\pi d}{\hbar}}-1} + \frac{1+L(\theta_h^\pm|x-t|\sin\alpha^\pm)}{2e^{\theta_h^\pm|x-t|\sin\alpha^\pm\cosh(M\hbar)}}\right), \end{aligned}$$

since L decreases and $L(\gamma) \geq 1$ so

$$\frac{1+L(\theta_h^\pm|x-t|\sin\alpha^\pm)}{2} \leq L(\theta_h^\pm|x-t|\sin(\alpha^\pm-d_1)).$$

Therefore

$$e_0^{N,h\pm}(x,t) \leq \frac{2C_x e^{\theta_h^{\pm}|x-t|}}{\pi(1-\sin(d_2+\alpha^{\pm}))} L(\theta_h^{\pm}|x-t|\sin(\alpha^{\pm}-d_1))\times \left(\frac{1}{e^{\frac{2\pi d}{\hbar}}-1}+\frac{1}{e^{\theta_h^{\pm}|x-t|\sin\alpha^{\pm}\cosh(M\hbar)}}\right).\blacksquare$$

3.5.2.3/ PROOF OF THEOREM 7

Applying the result of Theorem 6, we have

$$\begin{aligned} e_{\hbar}^{N,h\pm}(x_n) &\leq \sum_{j\in J_n^{\pm}} |u_{j+\frac{1}{2}}|\left(e_0^{N,h\pm}(x_n,x_j)+e_0^{N,h\pm}(x_n,x_{j+1})\right) \\ &\leq \left(\max_{j\in J_n^{\pm}}|u_{j+\frac{1}{2}}|\right)\left(\sum_{j\in J_n^{\pm}} e_0^{N,h\pm}(x_n,x_j)+\sum_{j\in J_n^{\pm}} e_0^{N,h\pm}(x_n,x_{j+1})\right) \\ &\leq 2C_2\sum_{j\in J_n^{\pm}} e_0^{N,h\pm}(x_n,x_{j+\frac{1\mp1}{2}}) \\ &= 2C_2|J_n^{\pm}|\max_{j\in J_n^{\pm}} e_0^{N,h\pm}(x_n,x_{j+\frac{1\mp1}{2}}).\blacksquare \end{aligned}$$

Conclusion : From Theorem 6 and Theorem 7, we see that the error estimate is decreasing exponentially. However, this is only evaluated at a given $x_n \in (0,1)$. Therefore, applying this estimate to find optimal parameters for all x with a contour is impossible. Opposite, using many contours can not achieve real-time computation. We suggest another approach to estimate the error so that it helps find optimal parameters of only one contour.

3.5.3/ ERROR ESTIMATE BY USING BALANCING IDEAS

The idea of this approach is the balance of errors. Firstly, the balance of the discretization error and the truncation error to get optimal parameters of the contour. After that, the balance of the quadrature error in x and the quadrature error in ξ.

From (3.59), we can generalize

$$\max_{x\in[0,1]} e_{\hbar}^{N,h\pm}(x) \leq \frac{2C_2}{h}\max_{\substack{x\in[0,1]\\ |x-t|\in[h,1]}} e_0^{N,h\pm}(x,t). \tag{3.64}$$

Thanks to Theorem 2, we estimate

$$e_0^{N,h\pm}(x,t) \leq \sum_{\ell=1,2}\frac{\int_{\mathbb{R}}|g_0^{N,h\pm}(x,\xi+i(-1)^{\ell}d_{\ell},t)|d\xi}{e^{2\pi d_{\ell}/\hbar}-1}+\hbar\sum_{|k|>M}^{+\infty}|g_0^{N,h\pm}(x,\xi_k,t)|,$$

We pose

$$\begin{aligned} f_{\ell}^{h+}(\theta_h^+,\alpha^+,\xi,x,y) &= \frac{e^{\theta_h^+}e^{-\theta_h^+ h\sin(\alpha^+-d_1)\cosh\xi}}{\cosh\xi-\sin(\alpha^++d_2)}\left|\frac{1-e^{-\theta^+(\xi+iy)h}}{h}\right| \\ &\quad\times\left|\sum_{s=0}^{\ell}\frac{e^{s(x-1)}d^+(\ell,s)[\cosh^2\xi-\sin^2(\alpha^+-d_1)]^{1/2}}{\theta_h^+(1+\sin(i\xi-y-\alpha^+))+s}\right|. \end{aligned}$$

and

$$f_\ell^{h-}(\theta_h^-,\alpha^-,\xi,x,y) = \frac{e^{\theta_h^-}e^{-\theta_h^- h\sin(\alpha^- - d_1)\cosh\xi}}{\cosh\xi - \sin(\alpha^- + d_2)}\Big|\frac{1-e^{-\theta^-(\xi+iy)h}}{h}\Big| \times\Big|\sum_{s=0}^{\ell}\frac{e^{-sx}d^-(\ell,s)[\cosh^2\xi - \sin^2(\alpha^- - d_1)]^{1/2}}{\theta_h^-(1+\sin(i\xi - y - \alpha^-)) + s}\Big|.$$

Theorem 8 :

For all $x \in [0,1]$, $y \in [-d_1, d_2]$, $\ell = 0,1,...,N$, assume that the mapping $\xi \mapsto f_\ell^{h\pm}(\theta_h^\pm,\alpha^\pm,\xi,x,y)$ are in $L^2(\mathbb{R})$, then the sub-error estimate

$$\max_{\substack{x\in[0,1]\\ |x-t|\in[h,1]}} e_0^{N,h\pm}(x,t) \le \frac{1}{2\pi}\sum_{\ell=0}^{N} w_\ell^\pm \times \left[2\max_{\substack{x\in[0,1]\\ y\in[-d_1,d_2]}}\frac{\int_{\mathbb{R}} f_\ell^{h\pm}(\theta_h^{*\pm},\alpha^{*\pm},\xi,x,y)d\xi}{e^{2\pi d/\hbar}-1} + \max_{x\in[0,1]}\hbar\sum_{|k|>M}^{\infty} f_\ell^{h\pm}(\theta_h^{*\pm},\alpha^{*\pm},\xi_k,x,0)\right]. \tag{3.65}$$

where $w_\ell^\pm = \max_{x\in[0,1]}|\widehat{p_\ell^\pm}(x)|$, $d = \min\{d_1,d_2\}$ and $a^{*\pm},\theta_h^{*\pm},\alpha^{*\pm}$ are solution to

$$D^{*\pm} = \min_{\substack{a\in(0,\infty)\\ \theta_h^\pm\in(0,+\infty)\\ \alpha^\pm\in(d_1,\pi/2-d_2)}} D^\pm(a,\alpha^\pm,\theta_h^\pm), \tag{3.66}$$

where $a = M\hbar$,

$$D^\pm(a,\alpha^\pm,\theta_h^\pm) := \sum_{\ell=0}^{N} w_\ell^{\pm 2}\times \left[2\max_{\substack{x\in[0,1]\\ y\in[-d_1,d_2]}}\frac{\int_{\mathbb{R}} f_\ell^{h\pm}(\theta_h^\pm,\alpha^\pm,\xi,x,y)d\xi}{e^{2\pi d/\hbar}-1} - \max_{x\in[0,1]}\hbar\sum_{|k|>M}^{\infty} f_\ell^{h\pm}(\theta_h^\pm,\alpha^\pm,\xi_k,x,0)\right]^2.$$

Proof of Theorem 8

We note that the condition (3.19) implies $0 < \alpha^\pm - d_1 < \alpha^\pm < \alpha^\pm + d_2 < \frac{\pi}{2}$. This infers $0 < \sin(\alpha^\pm - d_1) < \sin(\alpha^\pm) < \sin(\alpha^\pm + d_2) < 1$. Moreover, $|x-t| \in [h,1]$, so $e^{\theta_h^\pm|x-t|} \le e^{\theta_h^\pm}$ and $e^{-\theta_h^\pm\sin(\alpha^\pm+d_2)\cosh\xi} \le e^{-\theta_h^\pm|x-t|\sin(y+\alpha^\pm)\cosh\xi} \le e^{-\theta_h^\pm h\sin(\alpha^\pm-d_1)\cosh\xi}$. Therefore, we have

$$\begin{aligned}
|g_0^{N,h+}(x,\xi+iy,t)| &\le \frac{1}{2\pi}\sum_{\ell=0}^{N}|\widehat{p_\ell^+}(x)|\frac{e^{\theta_h^+|x-t|(1-\sin(y+\alpha^+)\cosh\xi)}}{\theta_h^+(\cosh\xi-\sin(y+\alpha^+))}\Big|\frac{1-e^{-\theta^+(\xi+iy)h}}{h}\Big| \\
&\quad\times\Big|\sum_{s=0}^{\ell}\frac{e^{s(x-1)}d^+(\ell,s)\theta_h^+[\cosh^2\xi-\sin^2(y+\alpha^+)]^{1/2}}{\theta_h^+(1+\sin(i\xi-y-\alpha^+))+s}\Big|, \\
&\le \frac{1}{2\pi}\sum_{\ell=0}^{N}|\widehat{p_\ell^+}(x)|\frac{e^{\theta_h^+}e^{-\theta_h^+ h\sin(\alpha^+-d_1)\cosh\xi}}{\cosh\xi-\sin(\alpha^++d_2)}\Big|\frac{1-e^{-\theta^+(\xi+iy)h}}{h}\Big| \\
&\quad\times\Big|\sum_{s=0}^{\ell}\frac{e^{s(x-1)}d^+(\ell,s)\theta_h^+[\cosh^2\xi-\sin^2(\alpha^+-d_1)]^{1/2}}{\theta_h^+(1+\sin(i\xi-y-\alpha^+))+s}\Big|,
\end{aligned}$$

and

$$|g_0^{N,h-}(x,\xi+iy,t)| \leq \frac{1}{2\pi}\sum_{\ell=0}^{N}|\widehat{p_\ell^-}(x)|\frac{e^{\theta_h^-}e^{-\theta_h^- h\sin(\alpha^- - d_1)\cosh\xi}}{\cosh\xi-\sin(\alpha^-+d_2)}\Big|\frac{1-e^{-\theta^-(\xi+iy)h}}{h}\Big| \times\Big|\sum_{s=0}^{\ell}\frac{e^{-sx}d^-(\ell,s)\theta_h^-[\cosh^2\xi-\sin^2(\alpha^- - d_1)]^{1/2}}{\theta_h^-(1+\sin(i\xi-y-\alpha^-))+s}\Big|.$$

So the sub-error estimate

$$e_0^{N,h\pm}(x,t) \leq \frac{1}{2\pi}\sum_{\ell=0}^{N}|\widehat{p_\ell^\pm}(x)|\times \left(2\max_{y\in[-d_1,d_2]}\frac{\int_{\mathbb{R}}f_\ell^{h\pm}(\theta_h^\pm,\alpha^\pm,\xi,x,y)d\xi}{e^{2\pi d/\hbar}-1}+\hbar\sum_{|k|>M}^{\infty}f_\ell^{h\pm}(\theta_h^\pm,\alpha^\pm,\xi_k,x,0)\right). \tag{3.67}$$

Thus we have

$$\max_{\substack{x\in[0,1]\\|x-t|\in[h,1]}}e_0^{N,h\pm}(x,t)\leq\frac{1}{2\pi}\sum_{\ell=0}^{N}\max_{x\in[0,1]}|\widehat{p_\ell^\pm}(x)|\times \left[2\max_{\substack{x\in[0,1]\\y\in[-d_1,d_2]}}\frac{\int_{\mathbb{R}}f_\ell^{h\pm}(\theta_h^\pm,\alpha^\pm,\xi,x,y)d\xi}{e^{2\pi d/\hbar}-1}+\max_{x\in[0,1]}\hbar\sum_{|k|>M}^{\infty}f_\ell^{h\pm}(\theta_h^\pm,\alpha^\pm,\xi_k,x,0)\right]. \tag{3.68}$$

We balance this sub-error by finding

$$D^{*\pm}=\min_{\substack{a\in(0,\infty)\\\theta_h^\pm\in(0,+\infty)\\\alpha^\pm\in(d_1,\pi/2-d_2)}}D^\pm(a,\alpha^\pm,\theta_h^\pm), \tag{3.69}$$

where

$$D^\pm(a,\alpha^\pm,\theta_h^\pm):=\sum_{\ell=0}^{N}w_\ell^{\pm 2}\times \left[2\max_{\substack{x\in[0,1]\\y\in[-d_1,d_2]}}\frac{\int_{\mathbb{R}}f_\ell^{h\pm}(\theta_h^\pm,\alpha^\pm,\xi,x,y)d\xi}{e^{2\pi d/\hbar}-1}-\max_{x\in[0,1]}\hbar\sum_{|k|>M}^{\infty}f_\ell^{h\pm}(\theta_h^\pm,\alpha^\pm,\xi_k,x,0)\right]^2.$$

So we have the proof. ■

We define the L^∞-norm of the global error

$$\|e\|_{L^\infty(0,1)}=\max_{x\in[0,1]}e(x), \tag{3.70}$$

where $e(x)$ is defined right after the formula (3.26). We have the global error estimate in L^∞-norm

$$\|e\|_{L^\infty(0,1)} \leq \max_{x\in[0,1]}\sum_{\gamma=+,-}\left(e^{h\gamma}(x)+e^{N,h\gamma}(x)+e_\hbar^{N,h\gamma}(x)\right). \tag{3.71}$$

We ignore the error estimate $e^{N,h\pm}(x)$ from approximating the kernel by a spectral method.

Theorem 9 :

$$\|e\|_{L^\infty(0,1)} \leq C_1 h^* + \frac{C_2}{\pi h^*} \max\{F^{h^{*+}}(a^{*+}, \alpha^{*+}, \theta_h^{*+}), F^{h^{*-}}(a^{*-}, \alpha^{*-}, \theta_h^{*-})\},$$

where h^* is solution to

$$E^* := \min_{h\in(0,1)} E(h), \tag{3.72}$$

$$E(h) := \left(C_1 h - \frac{C_2}{\pi h} \max\{F^{h+}(a^{*+}, \alpha^{*+}, \theta_h^{*+}), F^{h+}(a^{*-}, \alpha^{*-}, \theta_h^{*-})\}\right)^2, \tag{3.73}$$

and

$$F^{h\pm}(a^\pm, \alpha^\pm, \theta_h^\pm)$$
$$= \sum_{\ell=0}^{N} w_\ell^\pm \left[2 \max_{\substack{x\in[0,1]\\ y\in[-d_1,d_2]}} \frac{\int_{\mathbb{R}} f_\ell^{h\pm}(\theta_h^\pm, \alpha^\pm, \xi, x, y) d\xi}{e^{2\pi d/\hbar} - 1} + \max_{x\in[0,1]} \hbar \sum_{|k|>M}^{\infty} f_\ell^{h\pm}(\theta_h^\pm, \alpha^\pm, \xi_k, x, 0) \right],$$

where $a^{*\pm}, \alpha^{*\pm}, \theta_h^{*\pm}$ are solution to (3.69) and $C_1 = \max_{x\in[0,1]} \|p(x,.)\|_{L^1(0,1)} \max_{y\in[0,1]} |u'(y)|$.

Proof of Theorem 9
From (3.64) and Theorem 8, we have

$$\max_{x\in[0,1]} e_\hbar^{N,h} \leq \frac{C_2}{\pi h} \left(F^{h+}(a^{*+}, \alpha^{*+}, \theta_h^{*+}) + F^{h+}(a^{*-}, \alpha^{*-}, \theta_h^{*-})\right) \tag{3.74}$$

Moreover, we also have

$$\begin{aligned} \max_{x\in[0,1]} e^h &= \max_{x\in[0,1]} \left| \int_0^1 p(x,y)(u(y) - \widetilde{u}(y)) dy \right| \\ &\leq \max_{x\in[0,1]} \left(h \times \max_{y\in[0,1]} |u'(y)| \times \int_0^1 |p(x,y)| dy \right) \\ &\leq h \times \max_{y\in[0,1]} |u'(y)| \times \max_{x\in[0,1]} \|p(x,.)\|_{L^1(0,1)}. \end{aligned}$$

So we balance e^h and $e_\hbar^{N,h}$ by finding E^* as in 3.72. ■

Remark 14 :

There is another approach without using the balance idea, that is, the optimal parameters of the contour can be computed directly from

$$\max_{\substack{x\in[0,1]\\ |x-t|\in[h,1]}} e_0^{N,h\pm}(x,t) \leq \frac{1}{2\pi} \min_{\substack{a^\pm\in(0,\infty)\\ \theta_\hbar^\pm\in(0,\infty)\\ \alpha^\pm\in(-d_1,\pi/2-d_2)}} F^{h\pm}(a^\pm, \alpha^\pm, \theta_h^\pm). \tag{3.75}$$

Conclusion : This method is only effective if the weights $w_\ell^\pm$ are small. This can be explained as follows. Since we used the strict inequality (3.67) to evaluate the norm of $g^{N,h\pm}$, this implies that the error will be large if $|\widehat{p_\ell^\pm}(x)|$ are large. To avoid this we still keep the

formula (3.27), then we continue the ideas optimization of the contours. We also note that the signs of $p_\ell(x)$ change and the signs of coefficients of Legendre polynomials are alternate. However, this approach can be applied in the case when the weights w are small. Moreover, the error estimate (3.72) has no more meaning in application since we don't know the function $u(y)$. The drawbacks of this approach are the computation time is so long (few hours).

3.6/ CONCLUSION

We have presented a contour optimization based on a theoretical error estimate which reduces computations. The approach has presented through a simple example of a Lyapunov equation arising in optimal control theory from the one-dimensional heat equation. However, the special features of this example are not taken into account and we do not expect further limitations in broader applications. In order to have a global view, other approaches are considered.

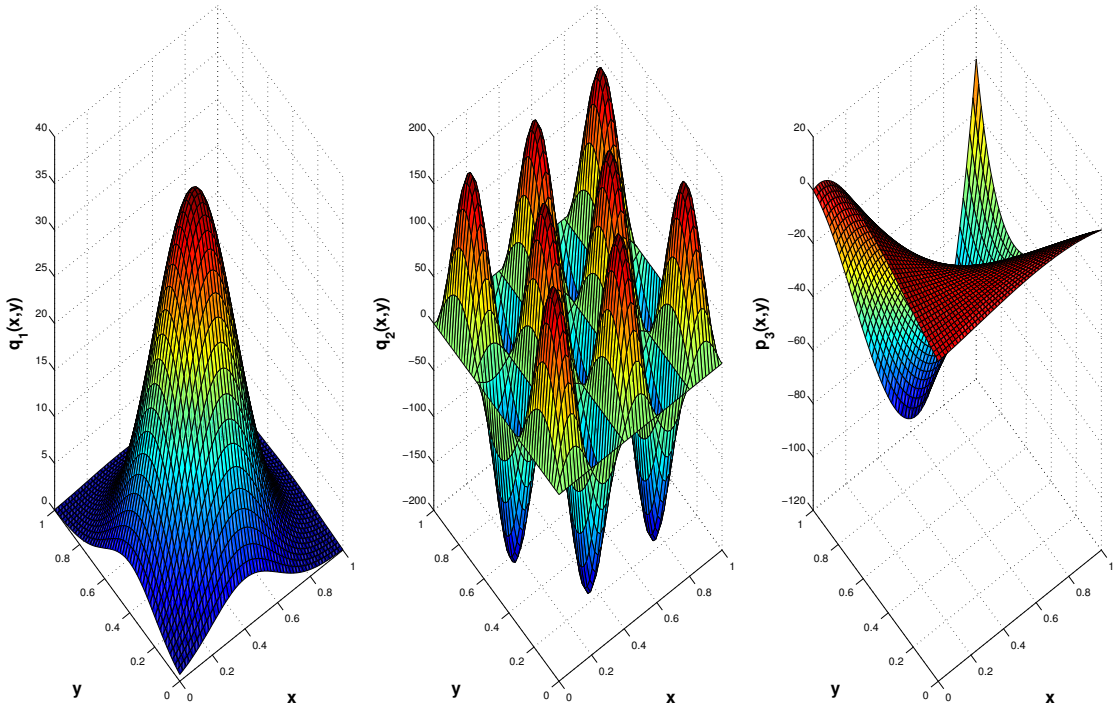


FIGURE 3.2 – Profiles $x - y$ of the kernels q_1, q_2, and q_3.

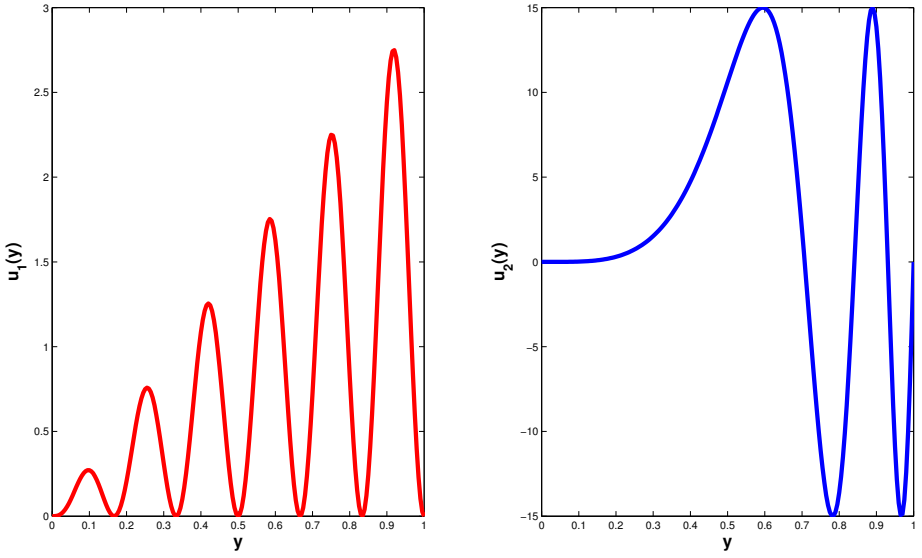


FIGURE 3.3 – Profiles of the inputs u_1 and u_2.

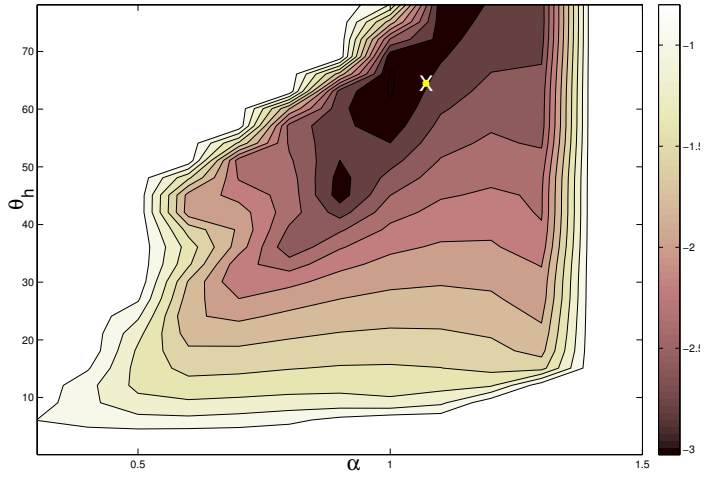


FIGURE 3.4 – Level lines of the logarithm $\log_{10}\|e\|_{L^2(0,1)}$ of the error function for $q_1(x,y)$ and $u_1(y)$ at a fixed $a = 1$. The cross represents the solution to the optimization problem $\min_{\alpha,\theta_h} F(a,\theta_h,\alpha)$ with $d_1 = d_2 = 0.5$, $h = 0.02$.

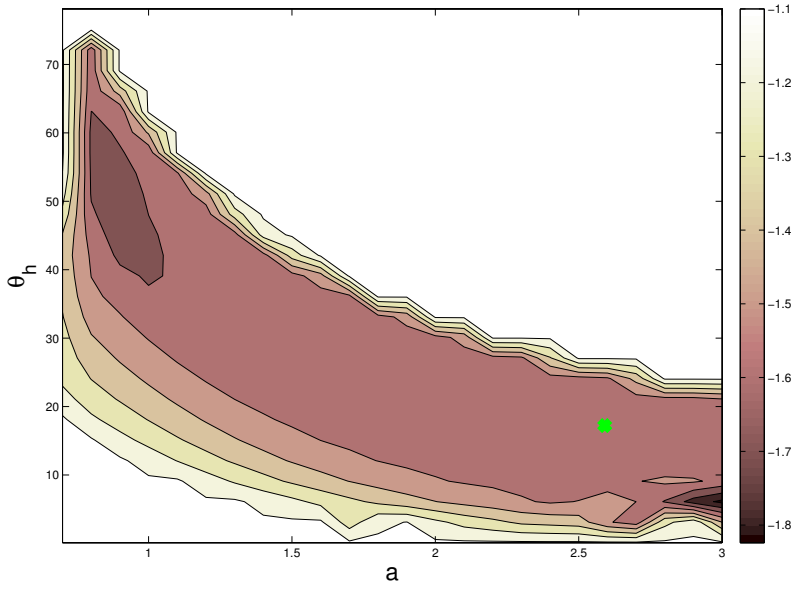


FIGURE 3.5 – Level lines of the logarithm $\log_{10}\|e\|_{L^2(0,1)}$ of the error function for $q_1(x,y)$ and $u_1(y)$ at a fixed $\alpha = 0.7849$. The cross represents the solution to the optimization problem (3.39) with $d_1 = d_2 = 0.5$, $h = 0.02$.

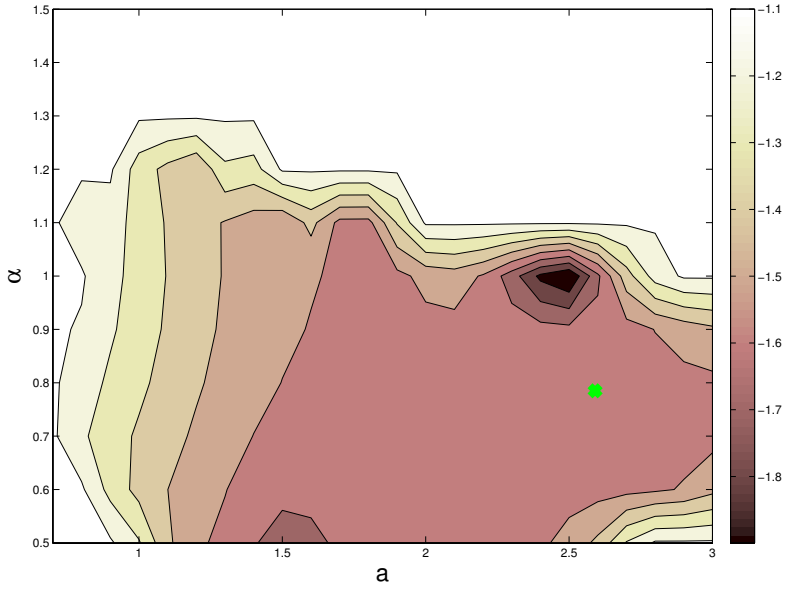


FIGURE 3.6 – Level lines of the logarithm $\log_{10} \|e\|_{L^2(0,1)}$ of the error function for $q_1(x, y)$ and $u_1(y)$ at a fixed $\theta_h = 17.2223$. The cross represents the solution to the optimization problem (3.39) with $d_1 = d_2 = 0.5$, $h = 0.02$.

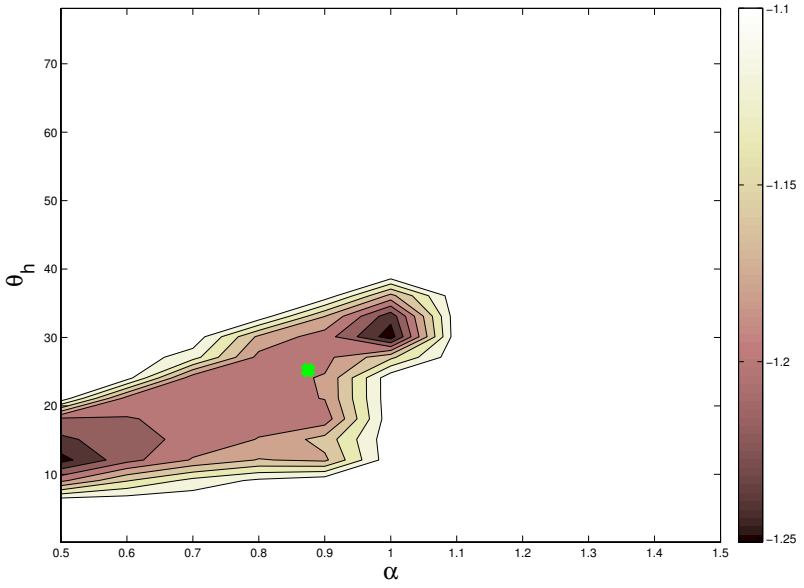


FIGURE 3.7 – Level lines of the logarithm $\log_{10} \|e\|_{L^2(0,1)}$ of the error function for $q_2(x, y)$ and $u_1(y)$ at a fixed $a = 1.9348$. The cross represents the solution to the optimization problem (3.39) with $d_1 = d_2 = 0.5$, $h = 0.02$.

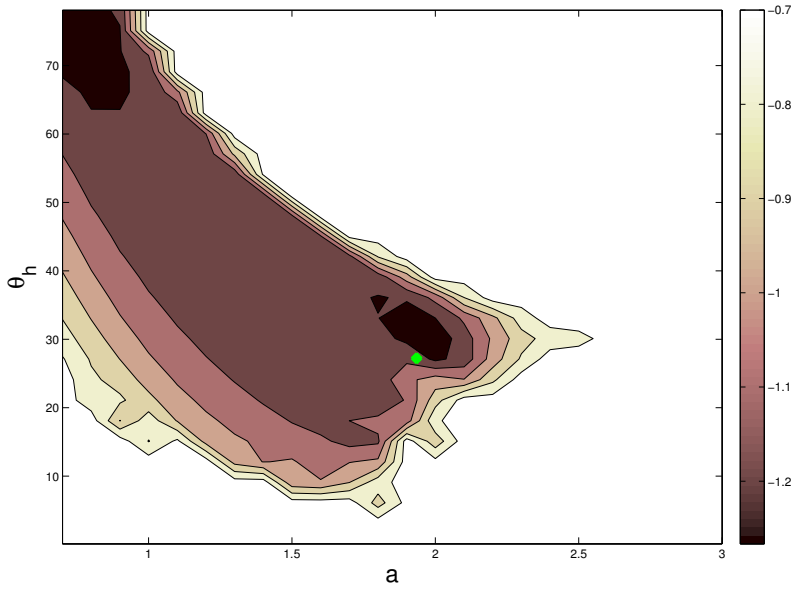


FIGURE 3.8 – Level lines of the logarithm $\log_{10} \|e\|_{L^2(0,1)}$ of the error function for $q_2(x, y)$ and $u_1(y)$ at a fixed $\alpha = 0.8747$. The cross represents the solution to the optimization problem (3.39) with $d_1 = d_2 = 0.5$, $h = 0.02$.

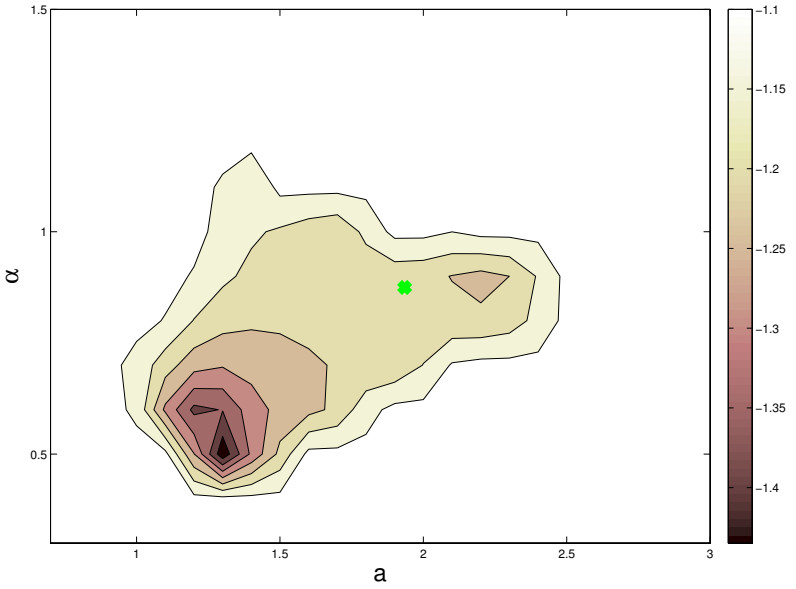


FIGURE 3.9 – Level lines of the logarithm $\log_{10} \|e\|_{L^2(0,1)}$ of the error function for $q_2(x, y)$ and $u_1(y)$ at a fixed $\theta_h = 25.2206$. The cross represents the solution to the optimization problem (3.39) with $d_1 = d_2 = 0.5$, $h = 0.02$.

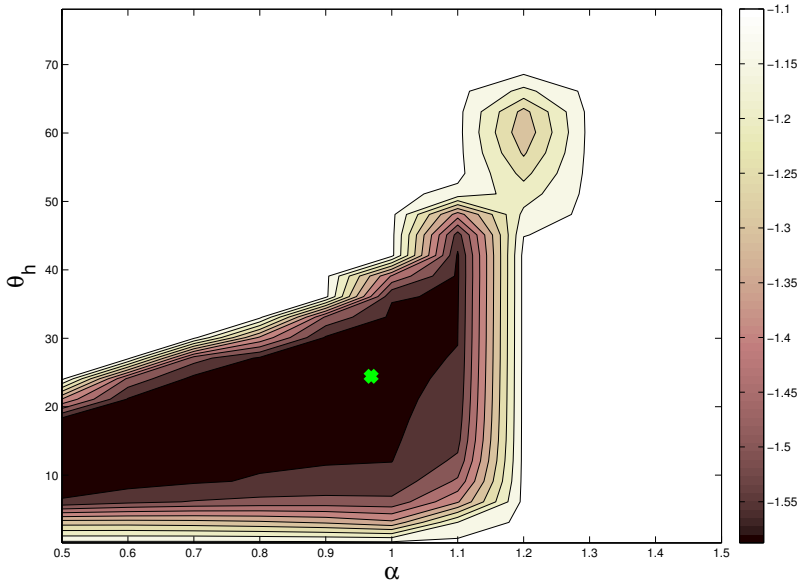


FIGURE 3.10 – Level lines of the logarithm $\log_{10}\|e\|_{L^2(0,1)}$ of the error function for $p_3(x,y)$ and $u_1(y)$ at a fixed $a = 2.2881$. The cross represents the solution to the optimization problem (3.39) with $d_1 = d_2 = 0.5$, $h = 0.02$.

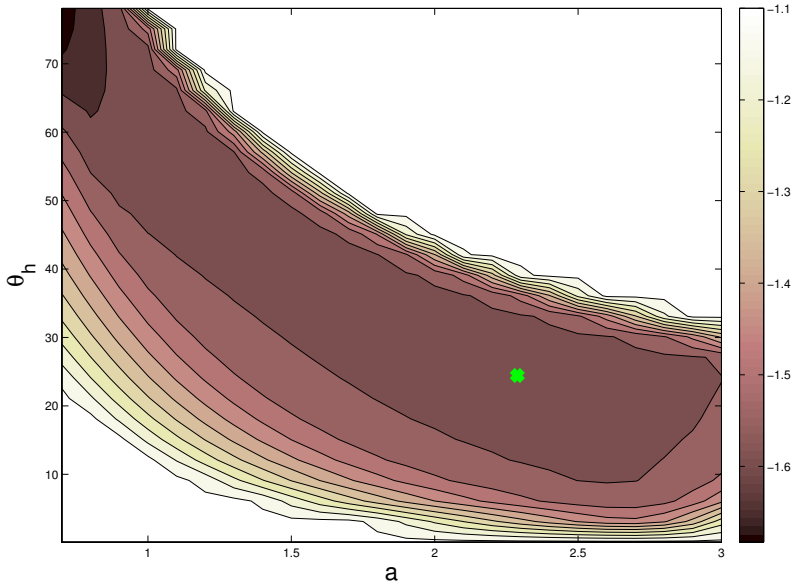


FIGURE 3.11 – Level lines of the logarithm $\log_{10}\|e\|_{L^2(0,1)}$ of the error function for $q_1(x,y)$ and $u_1(y)$ at a fixed $\alpha = 0.9688$. The cross represents the solution to the optimization problem (3.39) with $d_1 = d_2 = 0.5$, $h = 0.02$.

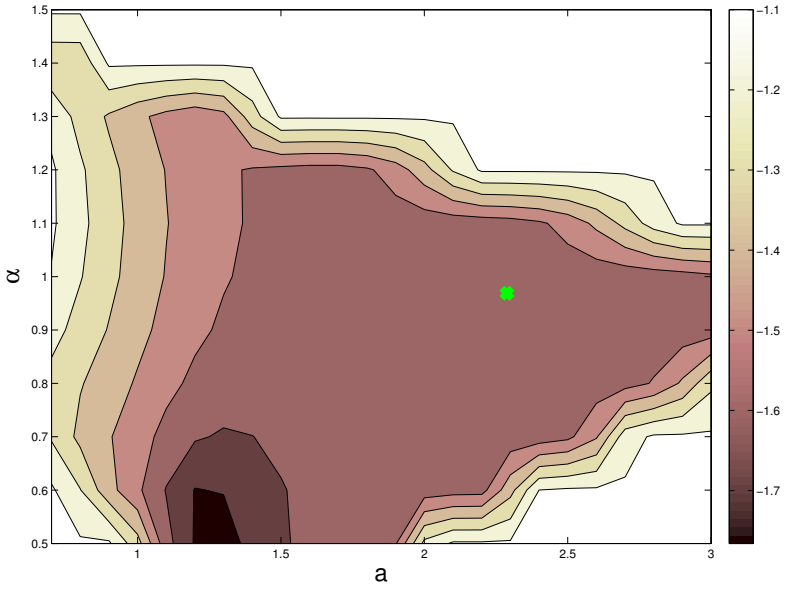


FIGURE 3.12 – Level lines of the logarithm $\log_{10} \|e\|_{L^2(0,1)}$ of the error function for $q_1(x, y)$ and $u_1(y)$ at a fixed $\theta_h = 24.4390$. The cross represents the solution to the optimization problem (3.39) with $d_1 = d_2 = 0.5$, $h = 0.02$.

4

CONCLUSION AND PERSPECTIVE

The diffusive realization of operators was mainly applied to operators with analytically known kernels. From the references in the field, it is known to be a very efficient method requiring little computation for real time realizations since small numbers of contour quadrature points are generally enough to yield good approximations. In Lenczner et al. [16] a mathematical framework was introduced to derive diffusive realization of operators solution to linear operator partial differential equations in one-dimensional domains. Only the premises of a numerical method were produced with very few numerical tests. Here, a more complete numerical method is discussed and its performances are investigated. In particular, it uses another method for analytic extension. The approach is presented through a simple example of a Lyapunov equation arising in optimal control theory from the one-dimensional heat equation. However, the special features of this example are not taken into account and we do not expect further limitations in broader applications.

Furthermore, we have presented a contour optimization based on a theoretical error estimate which reduces computations. We recognized that the DR approximation are a linear combination of inverse Laplace transforms, so the error estimate can be archived by applying the Stenger's results in the framework of the numerical inversion of the Laplace transform. Some comparisons of the relative error based on the choice of parameters of the contour of DR method and the direct method have presented. The contour optimization is shown to be effective.

Now the advantages of this method are its computation only requiring local inputs, parameters of the contour optimized. The online computation can be reformulated into a prefix sum which is very suitable for parallel computation. Therefore, combining these features opens a new direction for developing embedded real-time computation for distributed systems on distributed architectures. Namely, in view of real-time applications, we have discussed the cost of this method for a parallel implementation throughout estimating the numbers of operators and the computation time of both the DR method on three topologies and the direct method on the line topology.

We propose some different perspectives :

1. Developing this method in higher dimensional domains such as 2D.
2. Application DR to the non-integer power of a partial differential operator such as the square root of an operator, that is, $P^2 = Q$ with the kernels p and q satisfying

$$\int_0^1 \int_0^1 p(x,y)p(y,z)u(z)dzdy = \int_0^1 q(x,y)u(y)dy, \text{ for all } u \in H_0^1(\omega),$$

and the inversion of an operator, that is, $Q = P^{-1}$ with the kernels p and q satisfying

$$\int_0^1 q(z,x) \int_0^1 p(x,y)u(y)dydx = u(z), \text{ for all } u \in H_0^1(\omega).$$

3. Application DR to Riccati equations issued from distributed optimal control. Namely, we will consider an operator is the unique self-adjoint non-negative solution of the operational Riccati equation

$$(A^*P + PA - PBS^{-1}B^*P + C^*C)z = 0,$$

for all $z \in D(A)$-a dense domain of an infinitesimal generator A of a continuous semigroup on a some separable Hilbert space. A^* is the adjoint of the unbounded operator A. B^* is the adjoint of a bounded control operator B. C^* is the adjoint of an observation operator C. $S = Id$ is the weight operator. We note that the Lyapunov equation is the special case of the Riccati equation when $B = 0$.

4. Application DR to engineering problems as noise reduction, etc ...

5. Implementation on real distributed architectures as FPGAs. Youssef has successfully implemented the parallel and the pipeline architectures in a Xilinx Spartan3A XILINX [2010] comprising 200 kgates, 16 multipliers with 18-bit input data and 36-bit output data, 16 RAM blocks of 16 kbits each, and a 100 MHz clock. However, he only implemented for a vector of data $(u_n)_n$ with $N = 8$ components, for a quadrature formula with $M = 4$ nodes. We intended to implement with the larger number of components and nodes.

BIBLIOGRAPHIE

[1] A. H. Bhrawy and M. M. Al-Shomrani. A shifted legendre spectral method for fractional-order multi-point boundary value problems. *Advances in Difference Equations*, 2012(1) :1–19, 2012.

[2] P. Bidan, T. Lebey, G. Montseny, C. Neascu, and J. Saint-Michel. Transient voltage distribution in inverter fed motor windings : Experimental study and modeling. *IEEE Transactions on Power Electronics*, 16(1) :92–100, 2001.

[3] P. Bidan, T. Lebey, and C. Neacsu. Development of a new off-line test procedure for low voltage rotating machines fed by adjustable speed drives (ASD). *IEEE Transactions on Dielectrics and Electrical Insulation*, 10(1) :168–175, 2003.

[4] G. E. Blelloch. *Vector models for data-parallel computing*, volume 356. MIT press Cambridge, 1990.

[5] C. G. Canuto, Y. Hussaini, A. Quarteroni, and T.A. Zang. *Spectral Methods : Fundamentals in Single Domains*. Scientific Computation. Springer, 2010.

[6] E. Cuesta, C. Lubich, and C. Palencia. Convolution quadrature time discretization of fractional diffusion-wave equations. *Math. Comp.*, 75(254) :673–696, 2006.

[7] H.Q. Do, M. Lenczner, R. Couturier, and Y. Yakoubi. Diffusive realization of a lyapunov equation solution, and parallel algorithms implementation. In *CT15, Siam conference on Control and its Applications*, pages 68–75, Paris, France, jul 2015. SIAM.

[8] A. Grama. *Introduction to parallel computing*. Pearson Education, 2003.

[9] T. Hélie and D. Matignon. Diffusive representations for the analysis and simulation of flared acoustic pipes with visco-thermal losses. *Math. Models Methods Appl. Sci.*, 16(4) :503–536, 2006.

[10] T. Hélie, D. Matignon, and R. Mignot. Criterion design for optimizing low-cost approximations of infinite-dimensional systems : towards efficient real-time simulation. *Int. J. Tomogr. Stat.*, 7(F07) :13–18, 2007.

[11] P. Henrici. *Applied and Computational Complex Analysis : Vol. : 2. : Special Functions : Integral Transforms : Asymptotics : Continued Fractions*. John Wiley & Sons, 1977.

[12] D. K. Iakovidis, D. E. Maroulis, and D. G. Bariamis. Fpga architecture for fast parallel computation of co-occurrence matrices. *Microprocessors and Microsystems*, 31(2) :160–165, 2007.

[13] L. Laudebat, P. Bidan, and G. Montseny. Modeling and optimal identification of pseudodifferential electrical dynamics by means of diffusive representation - Part 1 : Modeling. *IEEE Transactions on Circuits and Systems I-Regular Papers*, 51(9) :1801–1813, 2004.

[14] F. T. Leighton. *Introduction to parallel algorithms and architectures*. Morgan Kaufmann San Francisco, 1992.

[15] M. Lenczner and G. Montseny. Diffusive realization of operator solutions of certain operational partial differential equations. *C. R. Math. Acad. Sci. Paris*, 341(12) :737–740, 2005.

[16] M. Lenczner, G. Montseny, and Y. Yakoubi. Diffusive realizations for solutions of some opreator equations : the one-dimensional case. *Mathematics of Computation*, 81(277) :319–344, Jan 2012.

[17] D. Levadoux and G. Montseny. Diffusive realization of the impedance operator on circular boundary for 2D wave equation. In *Mathematical and numerical aspects of wave propagation—WAVES 2003*, pages 136–141. Springer, Berlin, 2003.

[18] M. López-Fernández, C. Lubich, C. Palencia, and A. Schädle. Fast Runge-Kutta approximation of inhomogeneous parabolic equations. *Numer. Math.*, 102(2) :277–291, 2005.

[19] M. López-Fernández, C. Lubich, and A. Schädle. Adaptive, fast, and oblivious convolution in evolution equations with memory. *SIAM J. Sci. Comput.*, 30(2) :1015–1037, 2008.

[20] M. López-Fernández and C. Palencia. On the numerical inversion of the Laplace transform of certain holomorphic mappings. *Appl. Numer. Math.*, 51(2-3) :289–303, 2004.

[21] C. Lubich and A. Schädle. Fast convolution for nonreflecting boundary conditions. *SIAM J. Sci. Comput.*, 24(1) :161–182, 2002.

[22] D. Matignon and C. Prieur. Asymptotic stability of linear conservative systems when coupled with diffusive systems. *ESAIM Control Optim. Calc. Var.*, 11(3) :487–507, 2005.

[23] M. A. Mazidi and J. G. Mazidi. *The 8051 microcontroller and embedded systems*, volume 1. Prentice hall Upper Saddle River, NJ, USA :, 2000.

[24] M. A. McEvoy and N. Correll. Materials that couple sensing, actuation, computation, and communication. *Science*, 347(6228) :1261689, 2015.

[25] G. Montseny. Simple approach to approximation and dynamical realization of pseudodifferential time operators such as fractional ones. *IEEE Transactions on Circuits and Systems II-Express Briefs*, 51(11) :613–618, 2004.

[26] G. Montseny. *Représentation diffusive*. Hermès-Sciences, 2005.

[27] B. Parhami. *Introduction to parallel processing : algorithms and architectures*, volume 1. Springer, 1999.

[28] J. H. Reif. *Synthesis of parallel algorithms*. Morgan Kaufmann Publishers Inc., 1993.

[29] A. Schädle, M. López-Fernández, and C. Lubich. Fast and oblivious convolution quadrature. *SIAM J. Sci. Comput.*, 28(2) :421–438, 2006.

[30] P. Spasov. *Microcontroller Technology : The 68HC11*. Prentice-Hall, Inc., 1998.

[31] F. Stenger. Integration formulae based on the trapezoidal formula. *IMA Journal of Applied Mathematics*, 12(1) :103–114, 1973.

[32] F. Stenger. Approximations via Whittaker's cardinal function. *Journal of Approximation Theory*, 17(3) :222–240, 1976.

[33] F. Stenger. Numerical Methods based on Whittaker Cardinal, or Sinc Functions. *SIAM Review*, 23(2) :165–224, 1981.

[34] A. Talbot. The accurate numerical inversion of Laplace transforms. *J. Inst. Math. Appl.*, 23(1) :97–120, 1979.

[35] J. A. C. Weideman and L. N. Trefethen. Parabolic and hyperbolic contours for computing the Bromwich integral. *Math. Comp.*, 76(259) :1341–1356, 2007.

[36] Y. Yakoubi. *Deux Méthodes d'Approximation pour un Contrôle Optimal Semi-Décentralisé pour des Systèmes Distribués*. PhD thesis, Université de Franche-Comté, 2010.

[37] Y. Yakoubi, M. Lenczner, G. Goavec-Merou, R. Couturier, J.M. Friedt, et al. Diffusive realization of a lyapunov equation solution, and its fpga implementation. In *IFAG Conference*, pages 5477–5488, 2011.

TABLE DES FIGURES

LISTE DES TABLES

RÉSUMÉ

Cette thèse est consacrée à la conception d'algorithmes dans la perspective d'implantation de contrôles distribués temps réel implémentés sur un ensemble d'unités de calcul dont chacune ne peut communiquer qu'avec ses voisins directs et est associée à plusieurs capteurs et actionneurs. La solution étudiée dans cette thèse est fondée sur la théorie de la réalisation diffusive (RD) introduite pour des opérateurs temporels causaux définis sur la demi-droite réelle.

Le premier chapitre est consacré au rappel du cadre théorique établi par M. Lenczner et Y. Yakoubi pour la réalisation d'opérateurs linéaires solutions d'équations opérationnelles sur des domaines bornés. La méthode numérique qu'ils ont introduite souffre d'un certain nombre de limitations conduisant à des résultats numériques entachés d'erreurs importantes. La faiblesse principale de leur approche réside dans l'utilisation du prolongement analytique hors de l'intervalle (-1,1) des polynômes de Legendre utilisés pour l'approximation du noyau de l'opérateur.

La solution proposée consiste à définir un changement de variable supplémentaire qui garantit que le prolongement analytique du noyau est confiné à l'intervalle (-1,1) et conduit à une réduction drastique de l'erreur. Cette nouvelle approche est appliquée à une équation de Lyapunov rencontrée dans le cadre du contrôle optimal de l'équation de la chaleur.

Le deuxième chapitre analyse les coûts en temps de calcul d'implémentations sur différentes architectures parallèles. Trois architectures sont considérées : une ligne, un hypercube et un arbre binaire. Les temps de calcul sont évalués pour chacune de ces topologies et sont comparés à ceux d'une méthode directe mise en œuvre sur une architecture en ligne.

Finalement au troisième chapitre et pour la première fois des estimations d'erreur de la RD sont établies. Elles servent de base à l'optimisation des paramètres et donc à la minimisation du coût du calcul ce qui est déterminant dans la perspective d'applications temps-réel. Différentes approches sont considérées, mais celle qui est préférée consiste en la décomposition de la représentation diffusive en une combinaison linéaire d'inverses de transformations de Laplace et en l'utilisation d'estimations d'erreur qui s'y rapportent. Cette approche est validée par des expérimentations numériques.

Mots-clés : Réalisation diffusive, opérateur de calcul parallèle, contrôl distribué, calcul architecture, topology parallèle.

Document réalisé avec LaTeX et :
le style LaTeX pour Thèse de Doctorat créé par S. Galland — http://www.multiagent.fr/ThesisStyle
la collection de paquets `tex-upmethodology` — http://www.arakhne.org/tex-upmethodology/

Abstract:

In this thesis, we study the design and the implementation of generic control algorithms for systems composed of distributed MEMS in a semi-centralized context, i.e., only the direct neighbours can communicate. The control is computed in real time by a network of independent microcontrollers, each of them being associated with a couple of sensors and actuators. So the challenge is to introduce the possibility of communications between neighbour microcontrollers in order to implement control laws based on communications between neighbours. We focus on distributed control applications of the diffusive application where real-time implementations are an essential feature. To address this problem, the diffusive realization (DR) theory and its approximation are used. The advantage of DR is to deal with nonlocal problems encountered in many physical situations. Firstly, we recall the diffusive realization theory presented in the PhD thesis of Yakoubi [36]. However, his results were rather inaccurate. More precisely, the DR theory requires an analytic extension of the kernel p. This leads to non-uniformly bounded extensions with respect to the number of polynomials which causes high numerical errors. Therefore, in this thesis we apply an additional change of variables to the kernel p and its extension so that they become defined in an extension domain which eliminates an important source of error. The DR theory and its approximation are illustrated on a Lyapunov equation arising from the optimal control theory of the heat equation.
Then we discuss expected gains if the method is implemented on different parallel computer topologies based on corresponding designed networks. The envisioned applications are for real-time distributed control on distributed computing architectures. Namely, we present results for three parallel topologies: line topology, hypercube topology, binary topology. We also compare the computation times between our algorithm with these topologies and a direct spectral method with a line topology. These implementations considered in [7] offer a suitable parallel topology which is a good choice for real-time, embedded, massive, and low-cost computation.
Finally, as in Weideman and Trefethen [35], but with a significantly different theoretical approach, we provide, an error estimate of the DR and use it in a contour optimization method. Namely, we establish that the approximation of the DR can be decomposed into a linear combination of inverse Laplace transforms. Then the error estimates from Theorem 4.1 of Stenger [32] are applied to the evaluation of the discretization error of each term in that combination. Moreover, the method has been extensively tested and some significant results are reported.

Keywords: Diffusive realization, parallel computing operator, distributed control, computing architectures, parallel topology.

SPIM

■ École doctorale SPIM 16 route de Gray F - 25030 Besançon cedex

■ tél. +33 (0)3 81 66 66 02 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

UFC UNIVERSITÉ DE FRANCHE-COMTÉ