**Manuel Ricardo Fonseca Costa**
Engenheiro do Ambiente

# DisPar Methods and Their Implementation on a Heterogeneous PC Cluster

Dissertação para obtenção do Grau de Doutor em Engenharia do Ambiente

Orientador: António da Nóbrega de Sousa Câmara
Professor Associado com Agregação da
Faculdade de Ciências e Tecnologia da
Universidade Nova de Lisboa

Júri:

Presidente: Prof. Doutora Isabel Maria Spencer Vieira Martins
Arguentes: Prof. Doutor Hai Xiang Lin
Doutor André Bustorff Fortunato

Vogais: Prof. Doutor Daniel Peter Loucks
Prof. Doutor António da Nóbrega de Sousa Câmara
Prof. Doutor José Alberto Cardoso e Cunha
Prof. Doutor António Pedro de Nobre Carmona Rodrigues

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

**Abril, 2003**

**DisPar Methods and Their Implementation on a Heterogeneous PC Cluster**

# Sumário

Esta dissertação avalia duas áreas cruciais da simulação de advecção-difusão.

A primeira parte é dedicada a estudos numéricos. Foi comprovado que existe uma relação directa entre os momentos de deslocamento de uma partícula de poluente e os erros de truncatura. Esta relação criou os fundamentos teóricos para criar uma nova família de métodos numéricos, DisPar.

Foram introduzidos e avaliados três métodos. O primeiro é um método semi-Lagrangeano 2D baseado nos momentos de deslocamento de uma partícula para malhas regulares, DisPar-k. Com este método é possível controlar explicitamente o erro de truncatura desejado. O segundo método também se baseia nos momentos de deslocamento de uma partícula, sendo, contudo, desenvolvido para malhas uniformes não regulares, DisParV. Este método também apresentou uma forte robustez numérica. Ao contrário dos métodos DisPar-K e DisParV, o terceiro segue uma aproximação Eulereana com três regiões de destino da partícula. O método foi desenvolvido de forma a manter um perfil de concentração inicial homogéneo independentemente dos parâmetros usados. A comparação com o método DisPar-k em situações não lineares realçou as fortes limitações associadas aos métodos de advecção-difusão em cenários reais.

A segunda parte da tese é dedicada à implementação destes métodos num Cluster de PCs heterogéneo. Para o fazer, foi desenvolvido um novo esquema de partição, AORDA. A aplicação, Scalable DisPar, foi implementada com a plataforma da Microsoft .Net, tendo sido totalmente escrita em C#. A aplicação foi testada no estuário do Tejo que se localiza perto de Lisboa, Portugal.

Para superar os problemas de balanceamento de cargas provocados pelas marés, foram implementados diversos esquemas de partição: “Scatter Partitioning”, balanceamento dinâmico de cargas e uma mistura de ambos. Pelos testes elaborados, foi possível verificar que o número de máquinas vizinhas se apresentou como o mais limitativo em termos de escalabilidade, mesmo utilizando comunicações assíncronas. As ferramentas utilizadas para as comunicações foram a principal causa deste fenómeno. Aparentemente, o Microsoft .Net

remoting 1.0 não funciona de forma apropriada nos ambientes de concorrência criados pelas comunicações assíncronas. Este facto não permitiu a obtenção de conclusões acerca dos níveis relativos de escalabilidade das diferentes estratégias de partição utilizadas. No entanto, é fortemente sugerido que a melhor estratégia irá ser “Scatter Partitioning” associada a balanceamento dinâmico de cargas e a comunicações assíncronas. A técnica de “Scatter Partitioning” mitiga os problemas de desbalanceamentos de cargas provocados pelas marés. Por outro lado, o balanceamento dinâmico será essencialmente activado no inicio da simulação para corrigir possíveis problemas nas previsões dos poderes de cada processador.

# Abstract

This thesis assesses two main areas of the advection-diffusion simulation.

The first part is dedicated to the numerical studies. It has been proved that there is a direct relation between pollutant particle displacement moments and truncation errors. This relation raised the theoretical foundations to create a new family of numerical methods, DisPar.

Three methods have been introduced and appraised. The first is a 2D semi-Lagrangian method based on particle displacement moments for regular grids, DisPar-k. With this method one can explicitly control the desired truncation error. The second method is also based on particle displacement moments but it is targeted to regular/non-uniform grids, DisParV. The method has also shown a strong numerical capacity. Unlike DisPar-k and DisParV, the third method is a Eulerian approximation for three particle destination units. The method was developed so that an initial concentration profile will be kept homogeneous independently of the used parameters. The comparison with DisPar-k in non-linear situations has emphasized the strong shortcomings associated with numerical methods for advection-diffusion in real scenarios.

The second part of the dissertation is dedicated to the implementation of these methods in a heterogeneous PC Cluster. To do so, a new partitioning method has been developed, AORDA. The application, Scalable DisPar, was implemented with the Microsoft .Net framework and was totally written in C#. The application was tested on the Tagus Estuary, near Lisbon (Portugal).

To overcome the load imbalances caused by tides scatter partitioning was implemented, dynamic load balancing and a mix of both. By the tests made, it was possible to verify that the number of neighboring machines was the main factor affecting the application scalability, even with asynchronous communications. The tools used for communications mainly caused this. Microsoft .Net remoting 1.0 does not seem to properly work in environments with concurrency associated with the asynchronous communications. This did not allow taking conclusions about the relative efficiency between the partitioning strategies used. However, it is strongly suggested that the best approach will be to scatter partitioning with dynamic load balancing and with asynchronous communications. Scatter partitioning mitigates

load imbalances caused by tides and dynamic load balancing is basically trigged at the begging of the simulation to correct possible problems in processor power predictions.

To my parents and Ana

# Acknowledgments

My involvement in this wok started in 1996 when I joined, led by Professor António Câmara, the Virtual Tejo project. Several people taking part in it shared my interest in numerical modeling. Professor António Carmona Rodrigues played an important role teaching us several numerical concepts and incusing his passion for this kind of studies. Also Edmundo Nobre gave relevant contribution initiating me in code development.

In the Virtual Tejo project we implemented Paulo Castro's numerical models for advection-diffusion based on cellular automata. His work inspired us in developing what we now call a model based on particle displacement moments. In fact, the DisPar family of numerical methods was created in 1997 when I made a first version of those models with João Serpa. The model resulted from a training work ordered by Professor Câmara. As usual, he compelled us to innovate!

After finishing our under-graduation courses, both Serpa and me kept on spending days and nights dealing with particles, transport and numerical methods. One year afterwards the first version was ready for conference and sent to Professor Câmara for possible corrections. The answer was: “The paper will be good for the Journal of Hydraulic Engineering better than for a conference!”. In the meantime, André Fortunato and Anabela Oliveira would give us a quite relevant theoretical support.

In 2002 I worked for about three months in the University of Delft (Faculty of Information Technology and Systems). Professor Arnold Heemink accepted me as a visiting student and gave me the opportunity to acquire a new knowledge in partitioning methods for parallel computing. I worked daily with Professor Hai Xiang Lin in the development of the AORDA algorithm and in the Scalable DisPar application. Hai Xiang, a co-author, is a supervisor of the last part of this dissertation.

Many thanks to Sofia Simões for helping me in English corrections, Gualter Baptista for his contribution in the Scalable DisPar GUI development and to all the other referred companions and friends. I also want to acknowledge to my wife Ana, to my parents and stepfather for their support and advice. Thank you very much.

The author also wants to acknowledge Fundação para a Ciência e Tecnologia (FCT) from the Portuguese Ministry of Science and Technology by supporting the project “Simulação Distribuída e Visualização de Novas Formulações Numéricas em Qualidade da Água” under research contract POCTI/MGS/33998/2000.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

## *1.1 Statement of the Problem*

The amount of environmental catastrophes that happened worldwide throughout the years has strongly contributed to change the public opinion about the importance of the environment. Besides this, medical studies establishing relations between pollution and some diseases like cancer have also contributed to the global awareness of the importance that the environment has in our life.

The public opinion awareness has been fostering the need for the comprehension of how the natural environment behaves. Simulation is one of the key issues to understand small portions of it. Within this big field of study, the simulation of water quality in rivers, estuaries and coastal zones is of the most importance. For instance, the simulation of how pollutants behave in a water body is essential to decide where to locate a waste water discharge facility. Therefore, the theoretical foundations to support these studies are vital for accurate predictions.

The movement of pollutants in a water body is a diffusive process described by the advection-diffusion equation. In its turn, advection-diffusion is a stochastic process, which can be considered to be a Markov process. Considering that the pollutant particle displacement is independent from other particles, it is quite reasonable to assume this displacement as a Markov jump ([28]; [59]; [17]; [25]).

The resolution of the advection-diffusion differential equation in complex systems is hampered since it does not have an analytical solution, almost always requiring to be approximated (for introductory reading, [21]; [65]; [10]; [29]). Many numerical methods have been developed along the last decades. Nonetheless, most of these models rely on a pure mathematical approximation to the differential form of the advection-diffusion equation, disregarding the true nature of this type of processes.

Lagrangian methods, also known as particle tracking methods are the only methods explicitly assuming the Markov nature of the displacement of a particle ([28]; [59]; [17]). This type of methods simulates the behavior of each individual

particle by decomposing the pollutant mass into discrete units. The particle is displaced according to a specific probability distribution, which is usually assumed to be Gaussian. From a theoretical point of view, these methods do not have spatial error and they do not generate instabilities.

However, in real simulations, particle-tracking methods can hide problems related with non-linearities due to their apparent stability. This type of problems is usually related with bathymetric discretizations and/or inconsistencies with hydrodynamic models. These problems also occur in Eulerian-Lagrangian and Eulerian formulations. In the first family of numerical methods instabilities can be produced thus jeopardizing the accuracy of the results [45]. On the other hand, Eulerian models usually disguise these problems with an apparent stability by changing the physics of the simulated process.

Another problem hampering simulation is related with computation. The computational cost associated with 2D and 3D advection-diffusion models is rather significant which makes the simulation of many different scenarios unviable. This drawback is aggravated by the continuing need to recur to finer grids in order to capture effects at smaller temporal and spatial scales [64]. Distributed computing is one possible way to overcome these computational requirements.

The use of PC clusters is quite well known by the High Performance Computing (HPC) community, and currently it is the most accepted approach used to build low-price super-computers (for example, [6] and [7]). The cluster can be composed by dedicated computers, desktop machines or even by a mix of both. At least for universities, with computers connected by high-speed networks, and with small budgets, this is a very attractive approach to build a super-computer.

The distributed simulation of both hydrodynamic and transport in shallow-waters usually resorts to domain decomposition techniques by scattering a portion or portions of the computational domain among the available machines ([36]; [52]; [64]; [13]; [14]; [12]). However, the effect of tides can be very significant on the computational domain creating serious problems of load balancing. Scatter partitioning is one possible way to mitigate this type of problems [52], as well as dynamic load balancing ([12]; [14]).

Any partitioning strategy targeting desktop PCs must be prepared to deal with heterogeneous computational capacities. From a practical perspective, it is

almost impossible to work with heterogeneous PC Clusters without using dynamic load balancing. The true power of each machine is unknown, which implies that this value must be initially guessed and afterwards corrected by redefining the computational domain assignments. Regardless of dynamic load balancing, the use of scatter partitioning can be found to be useful to minimize the probability of load imbalances caused by the effect of tides. So, why not to mix both scatter partitioning and dynamic load balancing?

## *1.2 Research Objectives*

The research work described in this dissertation will address two distinct areas concerning the simulation of advection-diffusion in shallow-waters. The first part will focus on the numerical issues of the advection-diffusion processes. New numerical relations for both Eulerian and Eulerian-Lagrangian formulations based on Markov processes will be presented. These relations will be used as a tool to develop the DisPar family of numerical methods. The second part of this thesis deals with the computational implementation of such numerical methods on a PC cluster. Furthermore, its scalability is tested on the Tagus Estuary located near Lisbon, Portugal. The following research objectives were pursued:

### 1.2.1 Part I (Chapter 2, 3, 4 and 5)

1 – Development of a mathematical relation between truncation errors and Markov particle displacement moments for any explicit numerical formulation based on nodes.

2 – Based on the relation proved in 1, development of a new semi-Lagrangian method for 1D and 2D advection diffusion processes for regular grids.

3 – Development of a semi-Lagrangian method for 1D and 2D advection-diffusion processes also based on particle displacement moments for regular grids with different sizes over the x direction and over the y direction.

4 – Illustrate the importance of non-linearities in advection-diffusion.

### 1.2.2 Part II (Chapter 6)

5 – Development of a new partitioning scheme for heterogeneous PC Clusters.

6 – Implementation of the developed partitioning method described in 5, into a scalable application with the developed numerical methods. The application must also allow scatter partitioning, dynamic load balancing and a mix of both.

7 – Test of the application scalability on the Tagus Estuary, which significantly suffers from the effect of tides. This test is followed by another test of the developed partitioning method with simple partitioning strategies with dynamic load balancing, and its comparison with both scatter partitioning and a mix of scatter partitioning and dynamic load balancing.

## *1.3 Outline of the dissertation*

This thesis is composed by two distinct components relative to advection-diffusion processes. The first part (Chapters 2, 3, 4, 5) introduces several numerical methods for advection-diffusion processes based on particle displacement moments. The appendix 10 presents some mathematical developments that were made to prove some of the stated results or simply to assess the formulations for some specific situations. The second part (chapter 6) deals with the computational implementation of the developed numerical methods on a heterogeneous PC cluster.

Chapter 2 demonstrates that there is a relation between errors associated with numerical methods for advection-diffusion and the Markov particle displacement moments. For simplicity, this relation was solely proved for explicit node based numerical formulations for linear situations.

Chapter 3 introduces a 2D explicit semi-Lagrangian method for regular grids based on Gaussian particle displacement moments obtained by the relation between the advection-diffusion and the Fokker-Planck equation. The developed method consists in dividing the Gaussian distribution in a user specified number of discrete probabilities, which are evaluated as function of the particle displacement moments. These numerical probabilities are used as coefficients to calculate mass transfers between domain nodes. The method is assessed in theoretical situations by comparing the numerical results with known analytical solutions and on a practical situation by applying it to the Tagus estuary.

In chapter 4, a more general version of the numerical method developed in the chapter 3 is presented. Instead of nodes, the method makes an explicit use of

volumes allowing to dealing with non-uniform grids in the 1D situations and regular/non-uniform in the 2D case.

Chapter 5 aims to show the shortcomings associated with the numerical methods for advection-diffusion transport in non-linear situations. A simplified Eulerian formulation for three particle destination units is developed, so that the concentration profile is kept homogeneous if the initial concentration is uniform in all domain and boundaries. By comparing its results with the model developed in chapter 2 it will be possible to realize what usually happens in concentration based models in non-linear situations. Both models are appraised in theoretical situations and in a 1D practical situation.

Finally, in chapter 6 some techniques for the distributed simulation of advection-diffusion in shallow waters are presented. A new developed adaptive partitioning scheme for heterogeneous clusters is developed. It has the ability to deal with variable computational powers, load imbalances caused by the effect of tides, and possible wrong guesses for each machine power. The computational implementation, Scalable DisPar, was made with the Microsoft .NET framework and was totally written in C#. To appraise the theoretical developments, Scalable DisPar was tested in the Tagus estuary. Several situations were tested comparing scatter partitioning, dynamic load balancing and mixing both strategies.

# 2 Numerical methods for advection-diffusion seen through the eyes of Markov particles

## 2.1 *Introduction*

The numerical simulation of the transport of dissolved substances in natural systems has become an increasingly important tool used, for example, in water quality management and environmental impact assessment of engineered facilities. The resolution of such models has been based on the deterministic solution of the advection diffusion equation, through the use of Eulerian Models – EMs ([22];[33] and some general books like, for example, [29]; [10]) and Eulerian-Lagrangian Models – ELMs ([30]; [44]; [4]; [45]; [46]; [39]; [40]; [41]; [42]; [43]; [34]). However, these methods do not make explicit use of stochastic concepts, which can be seen as a disadvantage in the comprehension of physical processes involving randomness. In fact, the transport of a particle in a turbulent water body involves a degree of complexity so large that only its statistical behavior can be measured. In some systems, this stochastic nature associated to particle motion encouraged the development of several numerical formulations in statistical physics. Some of these systems - e.g. Brownian motion, birth-death processes or noise in electronic systems [25] - follow the principles of Markov processes.

Particle-tracking methods are another numerical technique used for the simulation of advection-diffusion and are quite appealing due to their strong numerical power. These methods have been applied to the pollutant simulation in both ground waters [59] and in surface water bodies such as rivers or estuaries ([28]; [17]; [5]). One of the main advantages of particle tracking methods derives from the direct use of stochastic concepts, by explicitly assuming that the motion of a particle in a water body is a Markov process.

Thus, a new approach to evaluate numerical errors associated with numerical methods for advection-diffusion is developed by directly using the concept of a Markov particle. As it will be proved, the direct use of stochastic concepts to analyze numerical formulations can bring a physical meaning to the associated numerical errors. It is expected that such approach will create an open field for the development of new Eulerian and Eulerian-Lagrangian numerical

methods explicitly based on the consequences of a Markov assumption for the pollutant particle jumps.

This chapter is organized as follows. Section 2.2 represents a mixture of bibliographic review about numerical methods for advection-diffusion processes with the assessment of the relation between them and Markov particles. Finally, section 2.3 outlines the theoretical foundations necessary to analyze numerical methods by explicitly using the underlying advection-diffusion stochastic concepts.

## *2.2 Background*

The mathematical foundations of the numerical studies developed in this thesis do not follow the traditional strategies to solve the advection-diffusion equation. Therefore, the bibliographic review about this issue will not follow the traditional analyses about numerical methods, either. The bibliographic review about numerical methods will be integrated with the description of the relation between the advection-diffusion and the Fokker-Planck equations. By doing so, the author hopes to simplify the jump between traditional methods (finite differences and finite elements) and the use of stochastic concepts in numerical formulations.

### 2.2.1 Numerical methods for the advection-diffusion equation

Transport of substances in shallow waters is very often described by the depth integrated advection-diffusion equation, expressed as it follows:

$$\frac{\partial(hC)}{\partial t} = -\frac{\partial(hu_x C)}{\partial x} - \frac{\partial(hu_y C)}{\partial y} + \frac{\partial}{\partial x}\left(D_x h \frac{\partial C}{\partial x}\right) + \frac{\partial}{\partial x}\left(D_y h \frac{\partial C}{\partial y}\right) \tag{2.1}$$

where h = water depth; C= concentration; $u_x$ = flow velocity over the x-axis; $u_y$ = flow velocity over the y-axis; $D_x$ = dispersion over x; $D_y$ = dispersion over y.

In practical situations, the analytical solution for equation (2.1) is unknown creating the need to numerically approximate it. Eulerian (EM) discretizations are the simplest and oldest family of numerical methods for the advection-diffusion equation. EMs can discretize the computational domain by either dividing it in uniform or unstructured grids. The simplest versions for this type of approaches are well reported in the following books: [10], [29] and [65]. The main advantage associated with this type of methods is given by their computational

implementation simplicity for regular grids. However, they don't track the water movement, which can create physical inconsistencies given that the computational units are always influenced by the same neighboring nodes (or volumes).

Besides this shortcoming, this type of models usually has numerical dispersion when strong non-linearities exist in any of the associated parameters like the bathymetry (this issue will be assessed on chapter 5). With this type of problems, the parameterization made for a spatial resolution does not necessarily work if applied to a more refined or even less refined spatial grid. The dispersion values obtained by a calibration process represent a mix of physics with numerical dispersion associated with the grid resolution/ numerical method used. This type of shortcomings can jeopardize the associated results, giving rise to confusions on the estimation of dispersion parameters if different spatial scales or different numerical methods are to be applied. Unfortunately, it was not possible to find literature doing any kind of comparisons between different methods for the same parameters.

Eulerian-Lagrangian models (ELM) represent another family of numerical methods for the advection-diffusion equation and are usually more robust than Eulerian approximations to the transport equation. The component associated with the drift term is treated more closely to the advection-diffusion physics. Flow motion is followed by the Lagrangian component defining how computational units influence other units.

In most of the situations, ELMs use points different from the grid nodes to evaluate mass transfers between domain nodes or volumes. These temporary nodes are used to interpolate to the grid ones. The interpolation process usually creates mass conservation problems as it can be seen, for instance, in [4] and [47].

In the literature it is also possible to find another type of Eulerian-Lagrangian models which does not do interpolations (for example, [41]). They always use the grid nodes for the evaluation of mass transfers and are called semi-Lagrangian methods.

ELMs are powerful; however authors usually avoid going on with profound studies about the associated numerical dispersion. It is possible to find authors with several published papers without any kind of formal mathematical analyses

([40], [41], [42], [43] and [66]). Therefore, and as it was stated and exemplified by [53], there appears to be a certain amount of misunderstanding in Eulerian-Lagrangian methods for the advection-diffusion equation. Some numerical methods only perform well if they use large time steps, introducing numerical dispersion if large number of time steps is to be used.

As it is possible to verify, by analyzing the numerical methods described in the literature many problems can be found, still existing some confusion about the relation between a numerical formulation and its consequences on the simulated process. Some numerical parameters like the Courant or Peclet numbers exist, establishing a bridge between numerical problems and physical concepts. However, none of the numerical approaches to the advection-diffusion equation makes an explicit use of the stochastic nature associated with the analyzed physical processes. Therefore, some stochastic concepts for Markov particles will now be outlined and later on used to set up a mathematical relation between numerical errors and the stochastic concepts associated with the advection-diffusion equation.

### 2.2.2 Transport problems and the Fokker-Planck equation

A Markov process is defined as a stochastic process that has the following property:

$$P\left(x_n,t_n|x_1,t_1;\ldots;x_{n-1},t_{n-1}\right)=P\left(x_n,t_n|x_{n-1},t_{n-1}\right) \tag{2.2}$$

where $P(x_n,t_n|x_1,t_1;\ldots;x_{n-1},t_{n-1})$ represents the transition probability of a particle to be in position $x_n$ at time $t_n$ if it was in all the positions $x$ at some specific time (i.e. if it was in $x_1$ at time $t_1$ and … and in $x_{n-1}$ at time $t_{n-1}$); $P(x_n,t_n|x_{n-1},t_{n-1})$ represents the transition probability conditioned only by the particle spatial position at the previous time. Thus, in a Markov process the transition probability is solely dependent on the previous spatial position.

The motion of a particle obeying this condition can be expressed by Chapman-Kolmogorov equation, which expresses the fact that a particle initially positioned in $x_1$ at time $t_1$ will get to position $x_3$ at time $t_3$ via any middle position $x_2$ at time $t_2$ [61]:

$$P\left(x_3,t_3\mid x_1,t_1\right)=\int P\left(x_3,t_3\mid x_2,t_2\right)P\left(x_2,t_2\mid x_1,t_1\right)dx_2 \tag{2.3}$$

This equation is an identity, which must be obeyed by the transition probability of any Markov process. For this type of processes it is also possible to verify that if a particle was initially in $x_1$ at time $t_1$, the probability to be in $x_2$ at time $t_2$ can be given by:

$$P(x_2,t_2) = \int P(x_2,t_2 \mid x_1,t_1)P(x_1,t_1)dx_1 \tag{2.4}$$

Master equation represents a differential form of the Chapman-Kolmogorov equation and can be expressed through the Kramers-Moyal expansion [50]:

$$\frac{\partial P(x,t)}{\partial t} = \sum_{r=1} \frac{(-1)^r}{r!} \frac{\partial^r}{\partial x^r} \left( \frac{\langle x^r \rangle_\tau}{\tau} P(x,t) \right) \tag{2.5}$$

where $P(x,t)$ represents the probability of a particle to be in $x$ at time $t$; $\langle x^r \rangle_\tau$ = particle displacement expectation associated with the temporal period of time $\tau$; $\tau$ = infinitesimal time associated with the particle displacement probability $P(x_n,t+\tau|x_1,t)$. This expression was meant to transition probabilities and, for that case, $P$ represents a conditional probability [50]. As it can be seen, two different time scales can be found in the master equation. The infinitesimal scale defining the temporal variation of P(x,t) and $\tau$ represents the lapse of time spent by the particle's Markov jump from $x_1$ to $x_n$.

The Fokker-Planck equation is a special case of the master equation expressed with the Kramers-Moyal expansion (2.5) for Markov processes whose individual jumps are small. In this situation, Fokker-Planck can either represent an approximation to this type of Markov processes or represent a process in which all terms >2 vanish from the master equation (2.5). The Fokker-Planck equation is therefore given by the following expression:

$$\frac{\partial P(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left( \frac{\langle x \rangle_\tau}{\tau} P(x,t) \right) + \frac{1}{2}\frac{\partial^2}{\partial x^2}\left( \frac{\langle x^2 \rangle_\tau}{\tau} P(x,t) \right), \tau \to 0 \tag{2.6}$$

However, the Fokker-Planck formal version is applied to transition probabilities [50] and is written as:

$$\frac{\partial P\left(x,t+\tau \mid x_0,t\right)}{\partial t} = \\ = -\frac{\partial}{\partial x}\left(\frac{\langle x\rangle_\tau}{\tau}P\left(x,t+\tau \mid x_0,t\right)\right)+\frac{\partial^2}{\partial x^2}\left(\frac{\langle x^2\rangle_\tau}{\tau}P\left(x,t+\tau \mid x_0,t\right)\right),\tau\to 0 \tag{2.7}$$

Many authors have used analogies between equation (2.7) and the advection-diffusion equation to produce the necessary parameters for particle tracking models. This analogy can be found in advection diffusion problems like the simulation of pollutants in a water body ([28];[17]; [57]; [58]) or the transport simulation in groundwater [59]. To illustrate the broadly described and used relation between this two equations, the Fokker-Planck equation for the (2.6) like form will now derived from the transport equation.

The 2D depth integrated advection-diffusion equation for fluxes of mass can be rewritten according to [17] as:

$$\begin{aligned}&\frac{\partial(Ch)}{\partial t} = \\ &-\frac{\partial}{\partial x}\left[\left(u_x+\frac{D_{xx}}{h}\frac{\partial h}{\partial x}+\frac{\partial D_{xx}}{\partial x}+\frac{D_{xy}}{h}\frac{\partial h}{\partial y}+\frac{\partial D_{xy}}{\partial y}\right)Ch\right]- \\ &-\frac{\partial}{\partial y}\left[\left(u_y+\frac{D_{yy}}{h}\frac{\partial h}{\partial y}+\frac{\partial D_{yy}}{\partial y}+\frac{D_{yx}}{h}\frac{\partial h}{\partial x}+\frac{\partial D_{yx}}{\partial x}\right)Ch\right]+ \\ &+\frac{\partial^2}{\partial x^2}\left(D_{xx}Ch\right)+\frac{\partial^2}{\partial y^2}\left(D_{yy}Ch\right)+ \\ &+\frac{\partial^2}{\partial x\partial y}\left(D_{xy}Ch\right)+\frac{\partial^2}{\partial x\partial y}\left(D_{yx}Ch\right)\end{aligned} \tag{2.8}$$

In practical situations, the diagonal coefficients do not have much expression and are usually avoided in the transport simulations. Thereby, in the special case where the coordinate system is aligned with the principal axes of dispersion ($D_{xy}$=$D_{yx}$=0), the transport equation can be simplified as:

$$\begin{aligned}&\frac{\partial(Ch)}{\partial t}+ \\ &+\frac{\partial}{\partial x}\left[\left(u_x+\frac{\partial D_x}{\partial x}+\frac{D_x}{h}\frac{\partial h}{\partial x}\right)Ch\right]- \\ &+\frac{\partial}{\partial y}\left[\left(u_y+\frac{\partial D_y}{\partial y}+\frac{D_y}{h}\frac{\partial h}{\partial y}\right)Ch\right]+ \\ &-\frac{\partial^2}{\partial x^2}\left(D_xCh\right)-\frac{\partial^2}{\partial y^2}\left(D_yCh\right)=0\end{aligned} \tag{2.9}$$

Assuming that the concentration in (x,y) is associated with the volume dx×dy×h, equation (2.9) can be rewritten as fluxes of mass by replacing C by the associated mass divided by the correspondent volume, as:

$$\begin{aligned}&\frac{\partial\left(M(x,y,t)\right)}{\partial t}+\\&+\frac{\partial}{\partial x}\left[\left(u_x+\frac{\partial D_x}{\partial x}+\frac{D_x}{h}\frac{\partial h}{\partial x}\right)M(x,y,t)\right]-\\&+\frac{\partial}{\partial y}\left[\left(u_y+\frac{\partial D_y}{\partial y}+\frac{D_y}{h}\frac{\partial h}{\partial y}\right)M(x,y,t)\right]+\\&-\frac{\partial^2}{\partial x^2}\left(D_xM(x,y,t)\right)-\frac{\partial^2}{\partial y^2}\left(D_yM(x,y,t)\right)=0\end{aligned} \tag{2.10}$$

where M(x,y,t) = mass associated with point(x,y) at time t.

Assuming that the concentration represents the sum of mass associated with a finite number of individual particles divided by the water volume and that all particles have the same mass, M(x,y,t) can be represented by the product of P(x,y,t) by the total particle mass as:

$$M(x,y,t)=P(x,y,t)\sum M_{Particle} \tag{2.11}$$

By replacing (2.11) in (2.10) the following expression is obtained:

$$\begin{aligned}&\frac{\partial\left(P(x,y,t)\sum M_{Particle}\right)}{\partial t}+\\&+\frac{\partial}{\partial x}\left[\left(u_x+\frac{\partial D_x}{\partial x}+\frac{D_x}{h}\frac{\partial h}{\partial x}\right)P(x,y,t)\sum M_{Particle}\right]-\\&+\frac{\partial}{\partial y}\left[\left(u_y+\frac{\partial D_y}{\partial y}+\frac{D_y}{h}\frac{\partial h}{\partial y}\right)P(x,y,t)\sum M_{Particle}\right]+\\&-\frac{\partial^2}{\partial x^2}\left(D_xP(x,y,t)\sum M_{Particle}\right)-\frac{\partial^2}{\partial y^2}\left(D_yP(x,y,t)\sum M_{Particle}\right)=0\end{aligned} \tag{2.12}$$

The sum of the mass of all particles is constant and therefore can be removed from the inside of the P(x,y,t) derivatives, as:

$$\begin{aligned}&\sum M_{Particle}\frac{\partial\left(P(x,y,t)\right)}{\partial t}+\\&+\sum M_{Particle}\frac{\partial}{\partial x}\left[\left(u_x+\frac{\partial D_x}{\partial x}+\frac{D_x}{h}\frac{\partial h}{\partial x}\right)P(x,y,t)\right]-\\&+\sum M_{Particle}\frac{\partial}{\partial y}\left[\left(u_y+\frac{\partial D_y}{\partial y}+\frac{D_y}{h}\frac{\partial h}{\partial y}\right)P(x,y,t)\right]+\\&-\sum M_{Particle}\frac{\partial^2}{\partial x^2}\left(D_xP(x,y,t)\right)-\sum M_{Particle}\frac{\partial^2}{\partial y^2}\left(D_yP(x,y,t)\right)=0\end{aligned} \tag{2.13}$$

The sum can now be removed from (2.13), producing the 2D Fokker-Planck equation for the probability of a Markov particle to be in (x,y) at time t, as:

$$\begin{aligned}&\frac{\partial\left(P(x,y,t)\right)}{\partial t}= \\ &-\frac{\partial}{\partial x}\left[\frac{\left(u_x+\frac{\partial D_x}{\partial x}+\frac{D_x}{h}\frac{\partial h}{\partial x}\right)\tau}{\tau}P(x,y,t)\right]- \\ &-\frac{\partial}{\partial y}\left[\frac{\left(u_y+\frac{\partial D_y}{\partial y}+\frac{D_y}{h}\frac{\partial h}{\partial y}\right)\tau}{\tau}P(x,y,t)\right]+ \\ &+\frac{1}{2}\frac{\partial^2}{\partial x^2}\left(\frac{2D_x\tau}{\tau}P(x,y,t)\right)+\frac{1}{2}\frac{\partial^2}{\partial y^2}\left(\frac{2D_y\tau}{\tau}P(x,y,t)\right)\end{aligned} \tag{2.14}$$

By this analogy between the advection-diffusion and the Fokker-Planck equations it is possible to obtain the particle displacement average and second order moment over both horizontal axes as:

$$\langle x\rangle_\tau=\left(u_x+\frac{\partial D_x}{\partial x}+\frac{D_x}{h}\frac{\partial h}{\partial x}\right)\tau \tag{2.15}$$

$$\left\langle x^2\right\rangle_\tau=2D_x\tau \tag{2.16}$$

$$\langle y\rangle_\tau=\left(u_y+\frac{\partial D_y}{\partial y}+\frac{D_y}{h}\frac{\partial h}{\partial y}\right)\tau \tag{2.17}$$

$$\left\langle y^2\right\rangle_\tau=2D_y\tau \tag{2.18}$$

This result is equal to the one obtained by [28] and [17]. In both articles, the authors arrive to the same results for these four expectations and in both cases the analogy is made for advection-diffusion and formal Fokker-Planck equations. However, in our opinion, the similarity between the advection-diffusion equation and the one obtained as a consequence of a Markov process, is more natural if the Fokker-Planck equation is expressed for P(x,y,t) and not ($P(x,y,t+\tau|x_0,y_0,t)$.

### 2.2.3 Particle tracking methods

Particle tracking methods represent a successful approach to the advection-diffusion simulation by their simplicity and by their numerical power. In this type of models, mass is transported as discrete particles according to the flow

motion and to the turbulence associated with the simulated physical process. Most of these models assume the particle displacements to be a Markov process, consistent with the advection-diffusion equation ([28]; [59] and [17]). Therefore, the necessary parameters for the particle displacement are obtained by establishing a relation between the transport equation and the formal version of the Fokker-Planck one as it was described in the previous section. To simulate each individual particle motion, an Ito assumption is made and the displacement is expressed as:

$$dX = \left( u_x + \frac{\partial D_x}{\partial x} + \frac{D_x}{h}\frac{\partial h}{\partial x} \right) \Delta t + \sqrt{2 D_x \Delta t} Z_{n1} \tag{2.19}$$

$$dY = \left( u_y + \frac{\partial D_y}{\partial y} + \frac{D_y}{h}\frac{\partial h}{\partial y} \right) \Delta t + \sqrt{2 D_y \Delta t} Z_{n2} \tag{2.20}$$

where dX=particle displacement over the x direction for a $\Delta$t time jump; dY= particle displacement over the y direction for a $\Delta$t time jump; Zn=[Zn1,Zn2]T = vector of two independent random numbers with zero mean and unit variance;

Some authors consider that Z can have any distribution if those two expectations are respected ([59] and [17]). However, if Z is non-Gaussian and all parameters are constant, the probability $P(x,y,t+\Delta t|x_0,y_0,t)$ will not be given by the product of two independent Gaussian distributions, which means that some error is superfluously introduced. By the central limit theorem, it is known that for large number of time steps, a non-Gaussian distribution for the Markov jumps will converge to a final 2D Gaussian function for P(x,y,t). If the final result is dependent on the number of time steps, some temporal error is introduced in the numerical formulation. For linear situations, the Gaussian distribution for the particle displacement is the only one in which the final result is always the same independently from the number of time steps used. Therefore, Gaussian distribution is the most reasonable function for the Z vector.

The success of particle-tracking methods essentially comes from their numerical power. Over a theoretical perspective, they do not have spatial error given that they are able, by their nature, to perfectly represent the Gaussian distribution. The only parameter discretized is the time for the Markov jump $\tau$. Besides this issue, this type of models does not have problems with strong particle concentration gradients given that they have a discrete nature. However, the stability associated to particle models, can disguise inconsistencies associated

with the underlying hydrodynamic models and/or with the representation of the bathymetry. Sometimes, many particles are needed in order to correctly estimate particle concentrations, like in salt intrusions in an estuary, drastically increasing the associated computational costs relative to Eulerian or Eulatian-Lagrangian models.

One of the main advantages of particle-tracking methods comes from their proximity to the underlying physics. The numerical formulation is not an abstract mathematical approximation to the advection-diffusion process, like it is typically done in finite differences approaches. For all this, a question can be raised. Is it possible to explicitly mix numerical concepts for advection-diffusion processes with its stochastic nature? The answer to this question will guide us to the next section.

## 2.3 Relation between truncation errors and particle displacement moments

Errors associated with a numerical method are usually assessed for linear situations and are obtained by decomposing into Taylor series relative to some point all the state variable values associated with different nodes. To evaluate the numerical error, all terms from the differential expression to be discretized are transferred to one of the two sides of the equation, leaving zero on the other. The difference between the Taylor decomposition and the differential form represents the numerical error associated with the numerical formulation. For linear advection-diffusion processes represented by the Fokker-Planck equation, errors are usually expressed by the extra coefficients associated with the different spatial derivatives of P as:

$$\frac{\partial P}{\partial t} + u\frac{\partial P}{\partial x} - D\frac{\partial^2 P}{\partial x^2} = \sum_{r=0}^{\infty} G_r \frac{\partial^r P}{\partial x^r} \tag{2.21}$$

where $G_r$ = error associated with the spatial derivative of P of order r.

For example, $G_2$ is usually defined as numerical dispersion and probably is one of the most important errors to be avoided by numerical approximations. However, what is the physical meaning of all these errors? In advection-diffusion, it is assumed that the particle displacement Markov nature is carried out in a continuous space. If this same continuous process is represented in a discrete

space, is it possible to define any relation with the differential equation discretization process?

Therefore, a new approach to assess truncation errors associated with some numerical formulation to the Fokker-Planck equation is to be developed in this section. For simplicity, it will be exclusively developed for explicit formulations based on nodes.

Assuming that the descritization of a continuous Markov process corresponds to remove the particle continuous freedom and condition its motion to the domain nodes, any explicit approximation to the Fokker-Planck equation should follow this equation:

$$P(x,t+\Delta t)=\sum_{r}P(x,t+\Delta t \mid x-\Delta x_r,t)P(x-\Delta x_r,t) \tag{2.22}$$

where r = computational domain node; P(x,t) = numerical probability for a particle to be in node x at time t; P(x, t+$\Delta$t|x-$\Delta$x,t) = numerical probability for the particle positioned in node x-$\Delta$x at time t to move to node x at time t+$\Delta$t.

The relation between truncation errors and particle displacement moments will now be demonstrated by decomposing into Taylor series relative to point (x,t) both sides of the equation (2.22). By doing this it will be possible to calculate a generic expression for the error associated with the P spatial derivative of order r ($G_r$).

### 2.3.1 Right-hand side

Let $\Psi$ be the matrix of probabilities,

$$\psi=\left[P(x-\Delta x_1,t)\quad P(x-\Delta x_2,t)\quad\cdots\qquad\qquad P(x-\Delta x_r,t)\right]_{(r)} \tag{2.23}$$

$$W=\begin{bmatrix}P(x,t+\Delta t \mid x-\Delta x_1,t)\\P(x,t+\Delta t \mid x-\Delta x_2,t)\\\vdots\\P(x,t+\Delta t \mid x-\Delta x_n,t)\end{bmatrix}_{(r)} \tag{2.24}$$

thus, it is possible to express equation (2.22) in matrix notation as function of $\Psi$ and W:

$$P(x,t+\Delta t)=\psi W \tag{2.25}$$

With linear conditions the probability for particle displacement will only depend on the distance between the origin node and the destination one, and therefore it is possible to establish the generic equality:

$$P\left(x,t+\Delta t \mid x-\Delta x,t\right)=P\left(x+\Delta x,t+\Delta t \mid x,t\right) \tag{2.26}$$

which means that W can be rewritten as:

$$W=\begin{bmatrix} P\left(x,t+\Delta t \mid x-\Delta x_1,t\right) \\ P\left(x,t+\Delta t \mid x-\Delta x_2,t\right) \\ \vdots \\ P\left(x,t+\Delta t \mid x-\Delta x_n,t\right) \end{bmatrix}_{(r)} = \begin{bmatrix} P\left(x+\Delta x_1,t+\Delta t \mid x,t\right) \\ P\left(x+\Delta x_2,t+\Delta t \mid x,t\right) \\ \vdots \\ P\left(x+\Delta x_n,t+\Delta t \mid x,t\right) \end{bmatrix}_{(r)} \tag{2.27}$$

So, W matrix is equivalent to the matrix associated with the particle displacement distribution if the particle was initially in x at time t with the possibility to move to $x+\Delta x_1$, …, $x+\Delta x_r$ after a time step $\Delta t$. These numerical probabilities can be expressed as function of the displacement moments. To do so, two new matrices are defined:

Let

$$S=\begin{bmatrix} \left(\Delta x_1\right)^0 & \left(\Delta x_2\right)^0 & \cdots & \left(\Delta x_r\right)^0 \\ \left(\Delta x_1\right)^1 & \left(\Delta x_2\right)^1 & \cdots & \left(\Delta x_r\right)^1 \\ \vdots & \vdots & \ddots & \vdots \\ \left(\Delta x_1\right)^{r-2} & \left(\Delta x_2\right)^{r-2} & \cdots & \left(\Delta x_r\right)^{r-2} \\ \left(\Delta x_1\right)^{r-1} & \left(\Delta x_2\right)^{r-1} & \cdots & \left(\Delta x_r\right)^{r-1} \end{bmatrix}_{(r)(r)} \tag{2.28}$$

$$E=\begin{bmatrix} \left\langle x^0 \right\rangle_{Met} \\ \left\langle x^1 \right\rangle_{Met} \\ \vdots \\ \left\langle x^{r-1} \right\rangle_{Met} \end{bmatrix}_{(r)} \tag{2.29}$$

where S = matrix with the particle displacement moments coefficients relative to x; E = numerical expectations relative to x.

The numerical expectations centered in x can be expressed as function of W as:

$$E=SW \tag{2.30}$$

which means that W matrix expressed as function of the numerical particle displacement moments is given by:

$$W = S^{-1}E \tag{2.31}$$

Replacing (2.31) in (2.25) this new expression is obtained:

$$P(x,t+\Delta t) = \psi S^{-1}E \tag{2.32}$$

Now, all $\Psi$ terms will be developed into Taylor series relative to point (x,t) and truncated after the r[nd] spatial derivative. To perform this decomposition, one can consider the following matrices:

$$\eta_x = \left[ \frac{\partial^0 P}{\partial x^0}(x,t) \quad \frac{\partial^1 P}{\partial x^1}(x,t) \quad \ldots \quad \frac{\partial^{r-1} P}{\partial x^{r-1}}(x,t) \right]_{(r)} \tag{2.33}$$

where $\eta$x represents the first 2k+1 spatial derivative orders, including the zero order;

the coefficient matrix L:

$$L = \begin{bmatrix} \frac{1}{0!} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{1!} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{1}{(2k-1)!} & 0 \\ 0 & 0 & \cdots & 0 & \frac{1}{(2k)!} \end{bmatrix}_{r\times r} \tag{2.34}$$

and Z matrix is expressed as:

$$Z = \begin{bmatrix} (-\Delta x_1)^0 & (-\Delta x_2)^0 & \cdots & (-\Delta x_r)^0 \\ (-\Delta x_1)^1 & (-\Delta x_2)^1 & \cdots & (-\Delta x_r)^1 \\ \vdots & \vdots & \ddots & \vdots \\ (-\Delta x_1)^{r-2} & (-\Delta x_2)^{r-2} & \cdots & (-\Delta x_r)^{r-2} \\ (-\Delta x_1)^{r-1} & (-\Delta x_2)^{r-1} & \cdots & (-\Delta x_r)^{r-1} \end{bmatrix}_{(r)(r)} \tag{2.35}$$

Thus, the $\psi$ matrix can now be written as:

$$\psi = \eta_x LZ \tag{2.36}$$

Replacing $\psi$ in (2.32), it is possible to write it as:

$$P(x,t+\Delta t) = \eta_x LZS^{-1}E \tag{2.37}$$

Hence, and by theorem 4 expressed in Appendix I, it is possible to write the equation (2.37) as follows:

$$P(x,t+\Delta t) = \sum_{r=0}^{\infty} \frac{(-1)^r}{r!} \langle x^r \rangle_{Met} \frac{\partial^r P}{\partial x^r}(x,t) \tag{2.38}$$

### 2.3.2 Left-hand side

To decompose into Taylor series P(x,t+$\Delta$t) relative to (x,t), and expressing this decomposition as function of the spatial derivatives, we will assume that P(x,t) can be represented by the linear Fokker-Planck equation:

$$\frac{\partial P(x,t)}{\partial t} = -u \frac{\partial P(x,t)}{\partial x} + D \frac{\partial^2 P(x,t)}{\partial x^2} \tag{2.39}$$

Let Rj be the matrix

$$R_j = \begin{bmatrix} 0 & \cdots & \binom{j}{0} D^0 (-u)^{j-0} & \cdots & \binom{j}{j} D^j (-u)^0 & \cdots & 0 \end{bmatrix}^T_{(r)} \tag{2.40}$$

where the first line is referenced by 0 and the nonzero terms begin at line j and end at line 2j. Rj's general term belonging to line v can be expressed as:

$$\begin{cases} R_{V,j} = \binom{j}{v-j} D^{v-j} (-u)^{j-(v-j)} & v \in [j, 2j] \\ R_{V,j} = 0 & v \notin [j, 2j] \end{cases} \tag{2.41}$$

The conversion from temporal to spatial derivatives is proved in theorem 3 demonstration from Appendix I and its general expression, written in a matrix format, can be expressed as:

$$\frac{\partial^j P}{\partial t^j} = \eta_x R_j \tag{2.42}$$

Let $\eta_t$ be the matrix of P temporal derivatives:

$$\eta_t = \left[ \frac{\partial^0 P}{\partial t^0}(x,t) \quad \frac{\partial^1 P}{\partial t^1}(x,t) \quad \cdots \quad \frac{\partial^{r-2} P}{\partial t^{r-2}}(x,t) \quad \frac{\partial^{r-1} P}{\partial t^{r-1}}(x,t) \right]_{(r)} \tag{2.43}$$

and T the matrix:

$$T = \begin{bmatrix} \Delta t^0 \\ \Delta t^1 \\ \vdots \\ \Delta t^{r-2} \\ \Delta t^{r-1} \end{bmatrix}_{(r)} \tag{2.44}$$

The P(x,t+Δt) development into Taylor Series truncated after the 2k term and relative to point (x,t) leads to:

$$P(x,t+\Delta t) = \left[ \frac{\partial^0 P}{\partial t^0}(x,t) \quad \frac{\partial^1 P}{\partial t^1}(x,t) \quad \cdots \quad \frac{\partial^{r-1} P}{\partial t^{r-1}}(x,t) \right]_{(r)} LT \tag{2.45}$$

Replacing the derivatives in expression (2.45) using expression (2.42):

$$P(x,t+\Delta t) = \eta_x \left[ R_0 \quad R_1 \quad \cdots \quad R_{r-1} \right]_{(r)(r)} LT \tag{2.46}$$

Now, it is necessary to evaluate the number of nonzero terms present in a R matrix line (i.e. the matrix with all sub-matrices Rj). To accomplish that, one must look at Rj's expression and verify that the first nonzero term begins at j. This means that the last nonzero value in line v will be in column v, which is the first from this column.

Assuming that ρ represents the amount of terms from line v not equal to zero the first entry can be given by v-(ρ-1). Therefore, so that a line v entry from matrix R may be different from zero, it must obey the condition: v-(ρ-1) ≤ v ≤ 2(v-(ρ-1)), which means that: ρ ≥ 1 and ρ ≤(v+2)/2. The first condition is universal and the second one imposes that the number of nonzero terms in line v is given by ρ=(v+2)/2 if v is even and ρ=(v+1)/2 if v is odd.

Thus line v obtained from the product RLT can now be represented by:

$$(RLT)_v = \sum_{j=v-(\rho-1)}^{v} \frac{1}{j!} \Delta t^j \binom{j}{v-j} D^{v-j} (-u)^{j-(v-j)} \tag{2.47}$$

This expression can be rewritten with the sum starting in zero and as function of 2DΔt and uΔt as:

$$\left(RLT\right)_v = \sum_{j=0}^{(\rho-1)} \frac{(-1)^{v-2(\rho-1)+2j}}{\left((\rho-1)-j\right)!\left(v-2(\rho-1)+2j\right)!} \frac{1}{2^{(\rho-1)-j}} \times \\ \times \left(2D\Delta t\right)^{(\rho-1)-j} \left(u\Delta t\right)^{v-2(\rho-1)+2j} \tag{2.48}$$

If the sum is expressed in reverse order, equation (2.48) can be yielded as:

$$\left(RLT\right)_v = (-1)^v \sum_{j=0}^{(\rho-1)} \frac{1}{(j)!(v-2j)!2^j} \left(2D\Delta t\right)^j \left(u\Delta t\right)^{v-2j} \tag{2.49}$$

From the appendix 10, it is possible to verify, that the v line from matrix RTL will be equal to the product of (-1)v/v! by the Gaussian expectation of order v with average uΔt and variance of 2DΔt:

$$\left(RLT\right)_v = \frac{(-1)^v}{v!} \sum_{j=0}^{(\rho-1)} \frac{v!}{(j)!(v-2j)!2^j} \left(2D\Delta t\right)^j \left(u\Delta t\right)^{v-2j} = \frac{(-1)^v}{v!} \left\langle x^v \right\rangle_{Gauss} \tag{2.50}$$

Therefore, P(x,t+Δt) can now be expressed as:

$$P\left(x,t+\Delta t\right) = \sum_{r=0} \frac{(-1)^r}{r!} \left\langle x^r \right\rangle_{Gauss} \frac{\partial P}{\partial x}\left(x,t\right) \tag{2.51}$$

### 2.3.3 Truncation errors

After decomposing both sides of equation (2.22) in Taylor series relative to point (x,t), the relation between analytical and numerical particle displacement moments can be expressed as:

$$P\left(x,t+\Delta t\right) = \sum_{r=0}^{\infty} \frac{(-1)^r}{r!} \left\langle x^r \right\rangle_{Gauss} \frac{\partial^r P}{\partial x^r}\left(x,t\right) = \sum_{r=0}^{\infty} \frac{(-1)^r}{r!} \left\langle x^r \right\rangle_{Met} \frac{\partial^r P}{\partial x^r}\left(x,t\right) \tag{2.52}$$

Removing from the left-hand side sum the first three terms, the following relation can be defined:

$$\begin{aligned} & P\left(x,t+\Delta t\right) = \sum_{r=0}^{\infty} \frac{(-1)^r}{r!} \left\langle x^r \right\rangle_{Gauss} \frac{\partial^r P}{\partial x^r}\left(x,t\right) = \\ & = P\left(x,t\right) + \left( -\frac{\left\langle x \right\rangle_{Gauss}}{\Delta t} \frac{\partial P}{\partial x}\left(x,t\right) + \frac{1}{2} \frac{\left\langle x^2 \right\rangle_{Gauss}}{\Delta t} \frac{\partial^2 P}{\partial^2 x}\left(x,t\right) \right) \Delta t \\ & + \sum_{r=3}^{\infty} \frac{(-1)^r}{r!} \left\langle x^r \right\rangle_{Gauss} \frac{\partial^r P}{\partial x^r}\left(x,t\right) = \\ & = P\left(x,t\right) + \frac{\partial P}{\partial t}\left(x,t\right)\Delta t + \sum_{r=3}^{\infty} \frac{(-1)^r}{r!} \left\langle x^r \right\rangle_{Gauss} \frac{\partial^r P}{\partial x^r}\left(x,t\right) \end{aligned} \tag{2.53}$$

It is possible to verify that the two terms multiplied by $\Delta t$ are equivalent to P temporal derivative, hence:

$$\frac{\partial P}{\partial t}(x,t) = -\frac{P(x,t)}{\Delta t} - \sum_{r=3}^{\infty} \frac{(-1)^r}{r!} \frac{\langle x^r \rangle_{Gauss}}{\Delta t} \frac{\partial^r P}{\partial x^r}(x,t) + \\ + \sum_{r=0}^{\infty} \frac{(-1)^r}{r!} \frac{\langle x^r \rangle_{Met}}{\Delta t} \frac{\partial^r P}{\partial x^r}(x,t) \tag{2.54}$$

If the two spatial derivatives associated with the linear Fokker-Planck equation are added to both sides of the equation,

$$\frac{\partial P}{\partial t}(x,t) + \frac{\langle x \rangle_{Gauss}}{\Delta t} \frac{\partial P}{\partial x}(x,t) - \frac{1}{2} \frac{\langle x^2 \rangle_{Gauss}}{\Delta t} \frac{\partial^2 P}{\partial^2 x}(x,t) = \\ \frac{\langle x \rangle_{Gauss}}{\Delta t} \frac{\partial P}{\partial x}(x,t) - \frac{1}{2} \frac{\langle x^2 \rangle_{Gauss}}{\Delta t} \frac{\partial^2 P}{\partial^2 x}(x,t) + \\ - \frac{P(x,t)}{\Delta t} - \sum_{r=3}^{\infty} \frac{(-1)^r}{r!} \frac{\langle x^r \rangle_{Gauss}}{\Delta t} \frac{\partial^r P}{\partial x^r}(x,t) + \sum_{r=0}^{\infty} \frac{(-1)^r}{r!} \frac{\langle x^r \rangle_{Met}}{\Delta t} \frac{\partial^r P}{\partial x^r}(x,t) \tag{2.55}$$

it is possible to define the following relation:

$$\frac{\partial P}{\partial t} + u \frac{\partial P}{\partial x} - D \frac{\partial^2 P}{\partial x^2} = \sum_{r=0}^{\infty} \frac{(-1)^r}{r!} \frac{\langle x^r \rangle_{Met} - \langle x^r \rangle_{Gauss}}{\Delta t} \frac{\partial^r P}{\partial x^r} \tag{2.56}$$

By matching equations (2.56) and (2.21) one can verify that $G_r$ is given by the difference between the moment associated with the numerical method and the analytical Gaussian moment for the particle displacement, such that:

$$G_r = \frac{(-1)^r}{r!} \frac{\langle x^r \rangle_{Met} - \langle x^r \rangle_{Gauss}}{\Delta t} \tag{2.57}$$

If the observed expectation calculated for a particle according to the numerical method used differs from the Gaussian one, this discrepancy represents the numerical error attached to the spatial derivatives of order r.

#### 2.3.3.1 $G_r$ calculation example

To better illustrate how to calculate $G_r$, an example will be given for the well known Forward Time Centered Space model (FTCS) applied to the Fokker-Planck equation (see, for example, [29] for details). FTCS discretizes the Fokker-Planck as:

$$\frac{P(i,n+1)-P(i,n)}{\Delta t} = -u\frac{P(i+1,n)-P(i-1,n)}{2\Delta x} + D\frac{P(i+1,n)-2P(i,n)+P(i-1,n)}{\Delta x^2} \tag{2.58}$$

FTCS uses two neighboring nodes and the node itself to build the numerical method, which means that equation (2.22) can be simplified as:

$$P(i,n+1) = \begin{bmatrix} P(i+1,n) & P(i,n) & P(i-1,n) \end{bmatrix} \times \begin{bmatrix} P(i,n+1 \mid i+1,n) \\ P(i,n+1 \mid i,n) \\ P(i,n+1 \mid i-1,n) \end{bmatrix} \tag{2.59}$$

Expression (2.58) can be rewritten in matrix notation as:

$$P(i,n+1) = \begin{bmatrix} P(i+1,n) & P(i,n) & P(i-1,n) \end{bmatrix} \times \begin{bmatrix} \frac{D\Delta t}{\Delta x^2} - \frac{1}{2}\frac{u\Delta t}{\Delta x} \\ 1 - \frac{2D\Delta t}{\Delta x^2} \\ \frac{D\Delta t}{\Delta x^2} + \frac{1}{2}\frac{u\Delta t}{\Delta x} \end{bmatrix} \tag{2.60}$$

By the relation (2.26), it is possible to redefine (2.59) as:

$$P(i,n+1) = \begin{bmatrix} P(i+1,n) & P(i,n) & P(i-1,n) \end{bmatrix} \times \begin{bmatrix} P(i-1,n+1 \mid i,n) \\ P(i,n+1 \mid i,n) \\ P(i+1,n+1 \mid i,n) \end{bmatrix} \tag{2.61}$$

If the numerical approximation is seen by the eyes of the discretized Markov displacement, the three numerical probabilities for the particle displacement can be given by matching equations (2.59), (2.60) and (2.61). Hence, the displacement probabilities are evaluated respectively as:

$$P(i-1,n+1 \mid i,n) = \frac{D\Delta t}{\Delta x^2} - \frac{1}{2}\frac{u\Delta t}{\Delta x} \tag{2.62}$$

$$P(i,n+1 \mid i,n) = 1 - \frac{2D\Delta t}{\Delta x^2} \tag{2.63}$$

$$P(i+1,n+1 \mid i,n) = \frac{D\Delta t}{\Delta x^2} + \frac{1}{2}\frac{u\Delta t}{\Delta x} \tag{2.64}$$

The numerical error associated with FTCS can be partially analyzed by calculating the three first expectations respectively of order 0, 1 and 2. For simplicity, the independent variable particle position is expressed in node notation and is centered in the node i. These three particle displacement moments are therefore respectively obtained as:

$$\left\langle (x_{node} - i)^0 \right\rangle_{Met} = \\ = (-1)^0 \times \left( \frac{D\Delta t}{\Delta x^2} - \frac{1}{2}\frac{u\Delta t}{\Delta x} \right) + 0^0 \times \left( 1 - \frac{2D\Delta t}{\Delta x^2} \right) + (1)^0 \times \left( \frac{D\Delta t}{\Delta x^2} + \frac{1}{2}\frac{u\Delta t}{\Delta x} \right) = 1 \tag{2.65}$$

$$\left\langle (x_{node} - i)^1 \right\rangle_{Met} = \\ = (-1)^1 \times \left( \frac{D\Delta t}{\Delta x^2} - \frac{1}{2}\frac{u\Delta t}{\Delta x} \right) + 0^1 \times \left( 1 - \frac{2D\Delta t}{\Delta x^2} \right) + (1)^1 \times \left( \frac{D\Delta t}{\Delta x^2} + \frac{1}{2}\frac{u\Delta t}{\Delta x} \right) = \frac{u\Delta t}{\Delta x} \tag{2.66}$$

$$\left\langle (x_{node} - i)^2 \right\rangle_{Met} = \\ = (-1)^2 \times \left( \frac{D\Delta t}{\Delta x^2} - \frac{1}{2}\frac{u\Delta t}{\Delta x} \right) + 0^2 \times \left( 1 - \frac{2D\Delta t}{\Delta x^2} \right) + (1)^2 \times \left( \frac{D\Delta t}{\Delta x^2} + \frac{1}{2}\frac{u\Delta t}{\Delta x} \right) = \\ = \frac{2D\Delta t}{\Delta x^2} \tag{2.67}$$

where $x_{node}$ = particle position in node notation.

To find $G_0$, $G_1$ and $G_2$ it is necessary to calculate the dimensionless expected Gaussian moments, which can be done by using equation (A.7), like:

$$\left\langle (x_{node} - i)^0 \right\rangle_{Gauss} = 1 \tag{2.68}$$

$$\left\langle (x_{node} - i)^1 \right\rangle_{Gauss} = \frac{u\Delta t}{\Delta x} \tag{2.69}$$

$$\left\langle (x_{node} - i)^2 \right\rangle_{Gauss} = \left( \frac{u\Delta t}{\Delta x} \right)^2 + \frac{2D\Delta t}{\Delta x^2} \tag{2.70}$$

It is now possible to verify that the two FTCS first expectations (equations (2.65) and (2.66)) are respectively equal to the two Gaussian ones (equations (2.68) and (2.69)):

$$\left\langle (x_{node} - i)^0 \right\rangle_{Met} - \left\langle (x_{node} - i)^0 \right\rangle_{Gauss} = 0 \tag{2.71}$$

$$\left\langle (x_{node} - i)^1 \right\rangle_{Met} - \left\langle (x_{node} - i)^1 \right\rangle_{Gauss} = 0 \tag{2.72}$$

Mass conservation is guaranteed by the 0$^{th}$ order moment and particle displacement average respects the Gaussian one. On the other hand, the FTCS second order moment is different from the Gaussian expected moment, which means that the numerical formulation has error associated with the second spatial derivative.

$$\left\langle (x_{node} - i)^2 \right\rangle_{Met} - \left\langle (x_{node} - i)^2 \right\rangle_{Gauss} = -\left( \frac{u\Delta t}{\Delta x} \right)^2 \tag{2.73}$$

Now, replacing the result expressed in equation (2.73) in expression (2.57), it is possible to verify that:

$$G_2 = \frac{(-1)^2}{2!}\frac{\left(\left\langle (x_{node}-i)^2 \right\rangle_{Met} - \left\langle (x_{node}-i)^2 \right\rangle_{Gauss}\right)\Delta x^2}{\Delta t} = \frac{(-1)^2}{2!}\frac{\left(-(u\Delta t)^2\right)}{\Delta t} = \\ = -\frac{1}{2}u^2\Delta t \tag{2.74}$$

This error is equal to the one obtained by the formal decomposition in Taylor series and is usually called numerical dispersion ([65]; [29] and [10]). This error corresponds to decrease the Gaussian variance and, if $\Delta$t is high, FTCS simulates a wrong problem.

For all the previous exposed, it is possible verify that **there exist a direct relation between particle displacement moments and numerical errors**. Numerical errors represent enlargements or decrements in the displacement moments. This relation is found to be very useful by giving a physical meaning to all errors associated with the extra terms in the spatial derivatives. The importance of particle displacement moments will be strengthen in the next chapter by developing a new numerical method for the deterministic simulation of advection diffusion based on Gaussian expectations.

## *2.4 Conclusions*

In this chapter a direct relation between particle displacement moments and truncation errors associated with numerical methods for the linear Fokker-Planck equation (or linear advection-diffusion) is proved. This relation attributes a physical meaning to any numerical error of any order. This was made possible by assuming that any discretization of the Fokker-Planck equation corresponds to the removal of the continuous spatial freedom of the associated Markov particle jump.

Therefore, it was found that numerical errors associated with the spatial derivative of order r are given by the difference between the moment of order r associated with the numerical formulation and the Gaussian moment of the same order. If both moments are equal, the method has no error associated with the correspondent spatial derivative. Thereby, to evaluate the numerical error associated with some method, it is solely necessary to calculate the displacement

moments associated with the numerical particle displacement probabilities. The results will be equal to the ones obtained by decomposing into Taylor series all the state variable node values relative to some point.

For all this, the explicit use of stochastic concepts to build and analyze numerical methods for the simulation of advection-diffusion processes was found to be quite useful. It provides an open field for the development of new ideas to build numerical methods on and it attributes a physical meaning to any kind of numerical error.

If the error associated with a formulation is given by the difference between the observed and the Gaussian moments, why not to build a model within this last expectations are forced? The answer to this question will be given in the next chapter by developing a new numerical method exactly based on this relation.

# 3 DisPar-k - Numerical method for the advection-diffusion simulation based on particle displacement moments

## *3.1 Introduction*

Eulerian-Lagrangian methods constitute the most robust family of numerical methods for the solution of the advection-diffusion equation. By tracking the flow motion, these methods are not limited by the time step as it happens to Eulerian formulations. However, this advantage does not give any information about the numerical error associated with a specific formulation. In fact, and as it was pointed out by [53], many Eulerian-Lagrangian formulations introduce numerical dispersion, which represents one of the main shortcomings associated with numerical approximations to the transport equation. According to this author, some models will only produce accurate results if large time steps are to be applied. Otherwise this error increases with the number of time steps used. Besides these shortcomings, ELMs can lose mass due to the interpolations made, jeopardizing their usage in non-conservative models.

To keep positivity in ELMs some filters can be used [46]. However, these techniques usually solve the problem by introducing numerical errors such as numerical dispersion.

As it was proven in chapter 2, the explicit use of stochastic concepts for the numerical approximation of advection-diffusion processes can bring several numerical advantages. Taking into consideration the direct relation between truncation errors and the Gaussian particle displacement moments, a new numerical method for advection-diffusion processes is developed in this chapter, DisPar-k ([19]; [20]). The underling ideas of particle displacement distributions expressed as function of the average and variance can be observed in [9].

The method consists in discretizing the continuous particle displacement distribution, which is assumed to be Gaussian, according to the grid nodes in a user-specified number of discrete numerical probabilities. As it was mentioned in

chapter 2, if the particle displacement distribution is a Gaussian with an average and variance obtained by the relation between the Fokker-Planck and advection-diffusion equations, all other moments can be obtained based on these two parameters. A user-specified assemblage of consecutive moments is utilized to evaluate the same number of numerical probabilities for the particle displacement for each node and for each time step. The moments used are chosen by assuming that the first spatial orders from equation (2.56) are the most important to be respected. For example, if the particle displacement distribution is to be discretized in 5 units the moments to be respected will start in 0 (which represents mass conservation) up to an expectation of the 4th order. To avoid problems related with the Courant number, the selection of nodes to discretize the Gaussian distribution is made according to the average. Thereby, the method assumes a semi-Lagrangian nature since one node will influence a group of other domain nodes according to the flow motion. By solely using the grid nodes, the method avoids problems related with mass conservation typically caused by interpolations and/or integrations between domain nodes ([4]; [47]). Finally, the numerical probabilities for the particle displacement are used as coefficients to transfer mass between domain nodes.

This chapter is organized as it follows. In section 3.2, the concept used to build the numerical method is first outlined for the 1D situation. The section ends by describing the 2D model and associated boundaries. Section 3.3 assesses the model accuracy by firstly evaluating the model with known analytical solutions in both 1D and 2D situations. Finally, the section ends by testing the numerical formulation on the Tagus estuary, which is located near Lisbon, Portugal.

## *3.2 Concept*

### 3.2.1 1D Concept

As it was mentioned in the previous chapter, advection-diffusion processes represent a process described by the Fokker-Planck equation. This last equation is a consequence of a Markov process for a specific situation where the particle displacement moments of order >2 vanish from the Kramers-Moyal expansion. Like particle tracking methods applied to advection-diffusion methods [28], the numerical method to be developed in this chapter is based on the displacement

distribution for a particle following a Markov process. The method assumes a Gaussian distribution for the particle displacement with an average and second order moment given by:

$$\langle x \rangle = \left( u_x + \frac{\partial D_x}{\partial x} + \frac{D_x}{h}\frac{\partial h}{\partial x} \right) dt \tag{3.1}$$

$$\left\langle x^2 \right\rangle = 2D_x dt \tag{3.2}$$

where $u_x$ = flow velocity over x; $D_x$ = Fickian dispersion over x; h water depth.

The numerical method consists in dividing the continuous particle displacement distribution into $2k_{x+1}$ discrete probabilities according to the grid nodes. The selection of nodes to be used in the Gaussian distribution discretization is based on the particle displacement average and the descretization center is given by the average integer part. Around this central node, which is defined by its index as $\beta_{xi}$, there are respectively $k_x$ consecutive nodes to upstream and $k_x$ consecutive nodes to downstream (Figure 3.1). By doing this, the method assumes a semi-Lagrangian nature, avoiding problems related to the Courant number. Particle displacement average is obtained by simply discretizing the time as:

$$\langle x_i \rangle = \left( u_i + \frac{\partial D_i}{\partial x} + \frac{D_i}{h}\frac{\partial h}{\partial x} \right) \Delta t \tag{3.3}$$

This type of approximation can introduce several problems in non-linear situations, since all the particle displacement average parameters can change over space and time. An Eulerian approximation to this parameter is quite limited, jeopardizing the advantages of the semi-Lagrangian nature. However, many techniques can be found to improve average tracking (for example, [66]). Since this is not a goal of the current work, the method will be formulated based on the simple Eulerian approximation to the particle displacement evaluation.

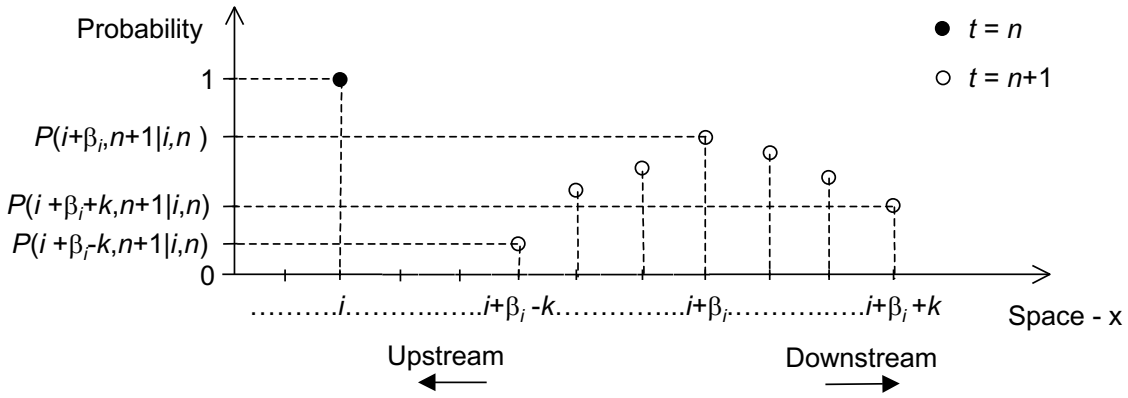



Figure 3.1 – Particle displacement distribution discretization in 2k+1 numerical probabilities

Another problem is now raised in the Gaussian distribution descretization process. For small time steps it is possible to verify that the particle displacement variance is equal to:

$$\sigma^2(x) = \langle x^2 \rangle - \langle x \rangle^2 = 2D_x dt - \left( \left( u_x + \frac{\partial D_x}{\partial x} + \frac{D_x}{h}\frac{\partial h}{\partial x} \right) dt \right)^2 = 2D_x dt - O\left(dt^2\right) \tag{3.4}$$

The variance is equal to the second order moment if the time step is infinitesimal and hence:

$$\sigma^2(x) = 2D_x dt \tag{3.5}$$

Either using the second order moment or using the variance discretizations can evaluate the Gaussian distribution. Which introduces less numerical error if an Eulerian approximation is to be applied? $\sigma^2=2D_x\Delta t$ or $\langle x^2\rangle=2D_x\Delta t$? This can be answered by analyzing the Fokker-Planck solution for linear conditions (2.39), in which is possible to verify that the particle displacement distribution is Gaussian with average $u\Delta t$ and variance $2D\Delta t$. To respect this analytical solution, variance is descretized as:

$$\sigma_i^2(i) = 2D_i \Delta t \tag{3.6}$$

where i is the particle initial position (i.e. position at time n).

As it was proved on the appendix 1, theorem 2, it is possible to express any Gaussian expectation as function of average and variance. The moments of a certain order can be either obtained by simply expressing them as function of the two previous ones like,

$$\left\langle x^{v+2} \right\rangle = \langle x \rangle \left\langle x^{v+1} \right\rangle + (v+1)\sigma^2(x)\left\langle x^{v} \right\rangle \tag{3.7}$$

or by using this more elaborated relation

$$\left\langle x^{v}\right\rangle = \sum_{j=0}^{\rho-1} \frac{v!}{2^{j} j!(v-2j)!} \left(\sigma^{2}(x)\right)^{j} \langle x\rangle^{v-2j} \tag{3.8}$$

where $\rho=(v+2)/2$ if v is even or $\rho=(v+1)/2$ if v is odd.

$W_i$ can represent the $2k_x+1$ probabilities associated to this discretization process in matrix notation, like

$$W_i = \begin{bmatrix} P\left(i+\beta_{x_i}-k, n+1 \middle| i,n\right) \\ P\left(i+\beta_{x_i}-k+1, n+1 \middle| i,n\right) \\ \vdots \\ P\left(i+\beta_{x_i}+k-1, n+1 \middle| i,n\right) \\ P\left(i+\beta_{x_i}+k, n+1 \middle| i,n\right) \end{bmatrix}_{(2k_x+1)} \tag{3.9}$$

where P(x,n+1|i,n) = numerical probability associated to the particle displacement between node i to node x (i.e. probability that a particle located in i at t=n, will move to x at t=n+1).

The evaluation of these $2k_x+1$ probabilities is to be done by forcing the Gaussian expectations of $2k_x+1$ different orders and by expressing the displacement moments as function of the numerical probabilities according to the grid nodes, as:

$$E_i = MW_i \tag{3.10}$$

where M is the square matrix, with $(2k_x+1)x(2k_x+1)$ elements, given by:

$$M = \begin{bmatrix} -k^{0} & (-k+1)^{0} & \cdots & 0^{0} & \cdots & (k-1)^{0} & k^{0} \\ -k^{1} & (-k+1)^{1} & \cdots & 0^{1} & \cdots & (k-1)^{1} & k^{1} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \\ -k^{2k-1} & (-k+1)^{2k-1} & \cdots & 0^{2k-1} & \cdots & (k-1)^{2k-1} & k^{2k-1} \\ -k^{2k} & (-k+1)^{2k} & \cdots & 0^{2k} & \cdots & (k-1)^{2k} & k^{2k} \end{bmatrix}_{(2k_x+1)(2k_x+1)} \tag{3.11}$$

and $E_i$ is the expectation matrix with 2k+1 elements, as

$$E_i = \begin{bmatrix} \left\langle \left(x-\beta_{x_i}\right)^0 \right\rangle \\ \left\langle \left(x-\beta_{x_i}\right)^i \right\rangle \\ \vdots \\ \left\langle \left(x-\beta_{x_i}\right)^{2k-1} \right\rangle \\ \left\langle \left(x-\beta_{x_i}\right)^{2k} \right\rangle \end{bmatrix}_{(2k_x+1)} \tag{3.12}$$

As it is possible to verify, $E_i$ first entry, 0th order moment, stats that the sum of all probabilities is 1 and therefore mass conservation is also imposed.

Finally, the numerical probabilities are calculated by solving the following algebraic system of equations:

$$W = M_i^{-1} E_i \tag{3.13}$$

Positivity is not guaranteed by solving the system, creating therefore the possibility to obtain negative probabilities, which might sound to something deceitful through the eyes of the probability definition. In fact, in many situations, the only way to guarantee that the particle displacement moments are respected is to define negative numerical probabilities. Over a numerical perspective, this is no more than a precision problem without any meaning on the advection-diffusion simulation results. However, if the simulated parameter is non-conservative, positivity becomes an important issue to be considered, raising several problems concerning the advection-diffusion accuracy. Is it possible to use high accurate numerical formulations for the transport simulation tolerant to the non-conservative simulation requirements?

Although the stochastic nature of the advection-diffusion process was explicitly used to build the numerical method, its application is deterministic by using the discrete probabilities as coefficients to transfer mass between domain nodes. Thereby, instead of individual particles, the model's state variable is represented by the pollutant mass. As it was mentioned in the chapter 2, Gaussian moments are just crucial to achieve the desire numerical error in the simulation of advection-diffusion processes.

Finally, from a practical perspective it is important to note that the system (3.13) is a Vandermonde system and therefore very good algorithms carrying on very low computational prices with tiny round off are available in the literature (for

example, [26] and [48]). Bearing in mind computational costs, it is desirable to calculate the particle displacement moments as function of the two previous orders (3.7) instead of the exclusive relation between the moment of a certain order and the average and variance (3.8). The number of mathematical operations will be drastically smaller if several consecutive moments are to be calculated.

### 3.2.2 2D Concept

The Markov nature postulation made for advection-diffusion processes, assumes beforehand the independence of the particle motion between the two horizontal axes. This independence, made it possible to develop the 2D version of DisPar-k, treating the probabilities for the particle displacement independently from each other and thereby the 2D distribution is obtained by just multiplying both probabilities. Again, this approach follows the same strategy of particle tracking models where the particle displacement is independently treated for each axis ([28] and [17]). The motion over both directions is a consequence of the independent movement over the x-axis and the y-axis. Therefore, the particle displacement over the y-axis also has a Gaussian distribution for the particle displacement with an average and variance respectively given by:

$$\langle y \rangle = \left( u_y + \frac{\partial D_y}{\partial y} + \frac{D_y}{h}\frac{\partial h}{\partial y} \right) dt \tag{3.14}$$

$$\sigma^2(y) = 2D_y dt \tag{3.15}$$

where $u_y$ = flow velocity over the y-axis; $D_y$ = dispersion coefficient (or Fickian coefficient) over the y-axis.

Using a simple Eulerian approach carries out the temporal discretization process, exactly as it was done for the x-axis.

$$\langle y \rangle = \left( u_y + \frac{\partial D_y}{\partial y} + \frac{D_y}{h}\frac{\partial h}{\partial y} \right) \Delta t \tag{3.16}$$

$$\sigma^2(y) = 2D_y \Delta t \tag{3.17}$$

The 2D displacement distribution represents a discretization of a 2D Gaussian distribution with an average and variance respectively given by equations (3.3) and (3.6) and (3.14) and (3.17). The generic numerical approximation to this distribution is given by the probability for a particle to be in the x,y node at time n+1 if it was in node i,j at time n as. The product of both

probabilities for the displacements expresses this probability over the two horizontal axes as:

$$P(x,y,n+1 \mid i,j,n) = P(x,n+1 \mid i,j,n) \times P(y,n+1 \mid i,j,n) \tag{3.18}$$

where $P(x,n+1 \mid i,j,n)$ = probability for the particle displacement over the x direction if the particle was in i,j node at time n; where $P(y,n+1 \mid i,j,n)$ = probability for the particle displacement over the y direction if the particle was initially in i,j node at time n.

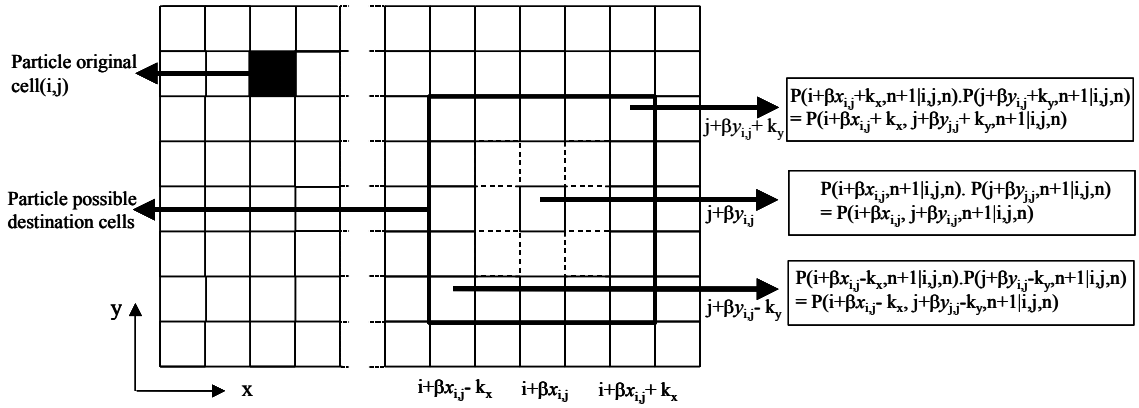



Figure 3.2 – destination rectangle associated with the 2D particle displacement distribution

## 3.2.3 Boundaries

### 3.2.3.1 Open Boundaries

For the simulation of open boundaries, a virtual grid is created with cells of the same size of the domain ones (Figure 3.3). Mass transfers for this virtual region are just removed from the computational domain.

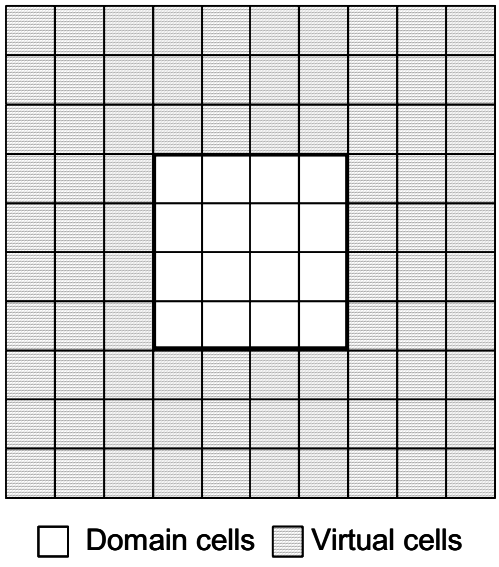



Figure 3.3 – DisPar-k creates virtual regions to deal with open boundaries

#### 3.2.3.2 Land Boundaries

Land volumes are treated as reflecting boundaries. If the particle destination rectangle intercepts this type of volumes, the mass to be transferred to them will remain in the source volume. However, the presence of this type of boundaries, represent a violation to the assumption that the particle displacement distribution is Gaussian and that the particle motion over the x-axis is independent of the y-axis. A physical barrier conditions the particle displacement over these two axes.

## *3.3 Tests*

### 3.3.1 1D theoretical tests

The accuracy of the developed method was tested by two well-known problems. The first problem, a linear situation, is a transport with the initial condition of a Gaussian profile, which has an average of $x_0$ and a standard deviation of $d_0$. The boundary conditions imposed are $C(0,t)=C(\infty, 0)=0$ and the analytical solution for this problem is given by:

$$C(x,t)=\frac{d_0}{\sqrt{d_0+2Dt}}\exp\left[-\frac{\left(x-x_0-ut\right)^2}{2d_0^2+4Dt}\right] \tag{3.19}$$

The second problem is a conservative transport of continuous injection where $u$ and $D$ have spatial variability. A methodology provided by [72] is used to obtain this analytical solution where the following initial and boundary conditions are imposed: $C(x,0)=0$ for $x>x'$, $C(x',t)=C_0$ *for* $x\leq x'$ and $C(\infty,t)=0$. The velocity field and the diffusion coefficient vary respectively linearly and quadratically with distance, i.e. $u(x)=u_0x$ and $D(x)=D_0x^2$ and the section area is constant. So, the analytical solution becomes:

$$C(x,t)=\frac{C_0}{2}\left[\begin{array}{l}\dfrac{x'}{x}\,erfc\left(\dfrac{\ln\left(x/x'\right)t\left(u_0+D_0\right)}{2\sqrt{D_0t}}\right)+\\ +\exp\left(\dfrac{u_0\ln\left(x/x'\right)}{D_0}\right)erfc\left(\dfrac{\ln\left(x/x'\right)+t\left(u_0+D_0\right)}{2\sqrt{D_0t}}\right)\end{array}\right] \tag{3.20}$$

Two tests were done for the linear case with the Gaussian profile [3] and a third one was carried out for the continuous injection with non-linear conditions. Each of these tests has two models, showing their growing power in prediction capability. The values used in each test are summarized in Table 3-1.

**Table 3-1– Parameters and conditions adopted in the tests**

| | Test 1 (Linear) | Test 2 (Linear) | Test 3 (Non-Linear) |
|---|---|---|---|
| $\Delta t$ | 24 | 9.6 | 0.05 |
| $\Delta x$ | 200 | 200 | 0.1 |
| Total points | 64 | 64 | 66 |
| $x'$ | 0 | 0 | |
| $u(x)$ | 10 | 50 | 1x, $u_0 = 0.1$ |
| $D(x)$ | 0 | 2500 | $0.003x^2$, $D_0$=0.003 |
| Time step number | 25 | 10 | 40 |
| Initial Condition $C(x,0)$ | Gauss hill, $x_0$=2000, $d_0$=264 | Gauss hill, $x_0$=2000, $d_0$=264 | 0 |
| $C(0,t)$ | 0 | 0 | 1 |
| $C((s\text{-}1)\ \Delta x, t)$ | 0 | 0 | 0.062 |
| Max. Courant ($u\Delta t/\Delta x$) | 1.2 | 2.4 | 3.8 |
| Max Dispersion coef. ($2D\Delta t/\Delta x^2$) | 0 | 1.2 | 1.7 |
| Max. Peclet number ($2E_i(x)/V_i(x)$) | $\infty$ | 4 | 33.5 |

The first test was done to show the importance the number of nodes used ($2k$+1) has on the accuracy of the results on a pure advection situation. Observing Figure 3.4, it is possible to verify that the model with $k$ = 6 produces more accurate results than the one with $k$ = 1. This result corresponds to what was theoretically predicted, since the increase of nodes reduces the spatial error, which is the most important one introduced by the drift term. Increasing the Courant number is not a restriction since the spatial error will depend exclusively on the fractional part of the particle displacement average.

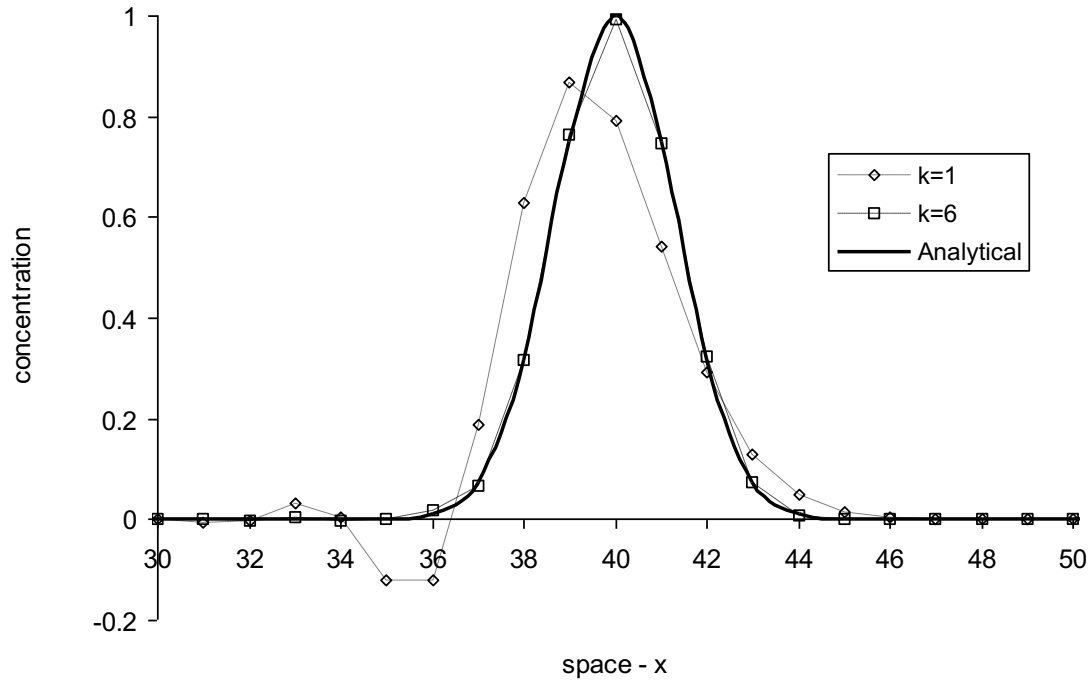



Figure 3.4 – Results from the DisPar-k in a pure advection situation (test 1)

On the other hand, the diffusion term is really dependent on the time step, meaning that temporal discretization can represent the most important issue in terms of accuracy. However, by increasing the number of nodes, this problem is expected to disappear as it can be seen in the second test (Figure 3.5).

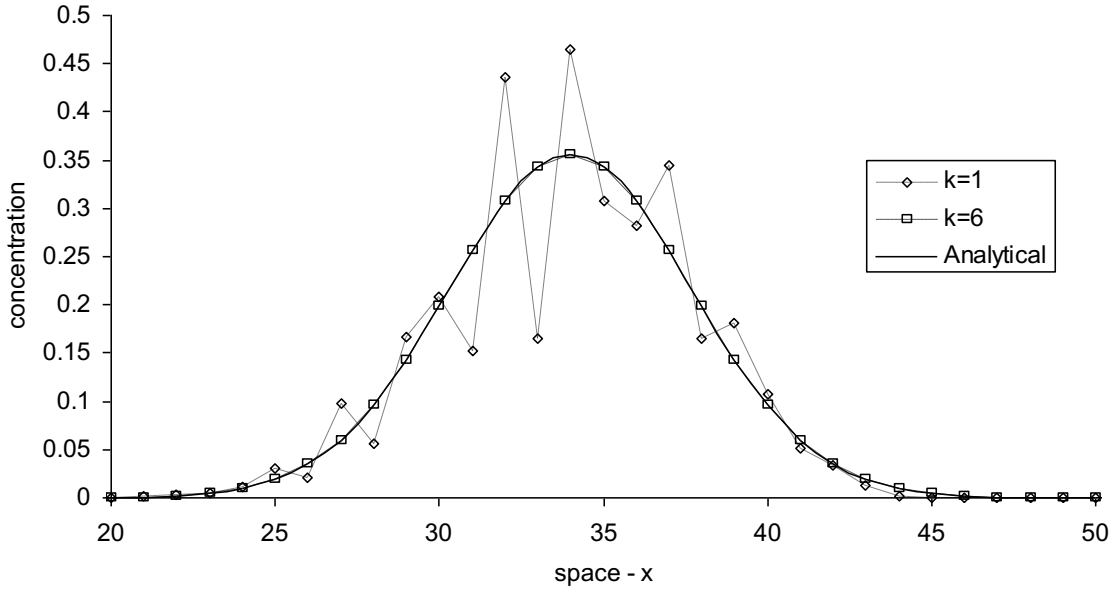



Figure 3.5 - Results from the DisPar-k in a diffusive-dominated situation (test 2)

A test closer to reality will be done now to better evaluate the formulation. For the boundary treatment it was considered that $\beta+k-1$ nodes to each side of the upstream and downstream boundaries influence the domain. This means that there are $2(\beta+k-1)$ hypothetical nodes with possible influence on the computational domain according to the boundary parameters. The values used in these possible mass origins are equal to the corresponding boundary and they were treated exactly in the same way as the domain nodes.

The highest Peclet number can be found in the upstream node decreasing progressively to downstream. The results near this advection-dominated region are accurate in both models, reflecting the DisPar-k power to treat the advective term. However, downstream, it is possible to verify the instability produced by the three-node model. As it happens on the second test, the temporal error introduced by the diffusion term is extremely visible in this part of the computational domain. Once again the increase of the number of nodes used to compute the model at each time solved the problem and the results produced are remarkably accurate.

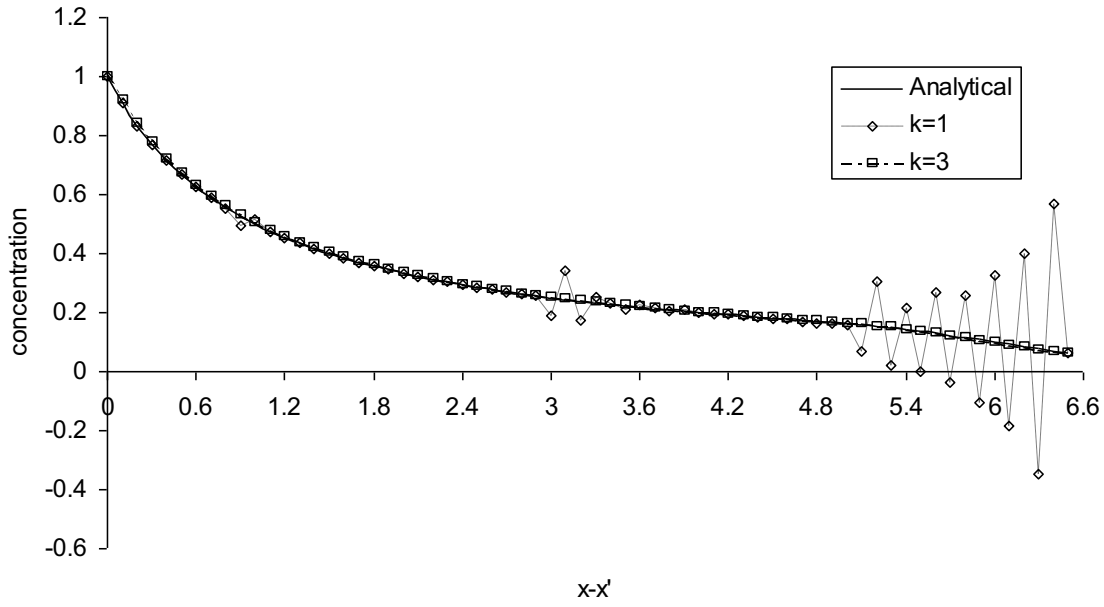



Figure 3.6 - Results from the DisPar-k in a non-linear situation (test 3)

## 3.3.2 2D tests

### 3.3.2.1 Rotating field test

DisPar 2D formulation behavior is tested in a steady rotating field at an angular velocity ($\omega$) of $2\pi/100$, without dispersion. The initial condition is a Gaussian plume centered on x = 30 and y = 20, with a standard deviation of 3 and

a maximum value of 1. The grid is uniform and the central point is x = 0 and y = 0, with $\Delta x = \Delta y = 1$, $u_{xi,j} = j.\omega$ and $u_{yi,j} = - i.\omega$. The value of $k_x$ is equal to $k_y$ and the total simulation time is equal to 100, which corresponds to one turn of rotation. Two different $\Delta$ts (0.5 and 0.05) were applied, leading to maximum Courant numbers of 0.94 and 0.09. In Figure 3.7 it is possible to observe that the increase in the particle destination cells ($[2k_x+1]\times[2k_y+1]$) and the $\Delta t$ decrease lead to an improvement in the results since the Gaussian plume is better represented. It is also possible to identify the $k_x$ and $k_y$ needed to obtain the minimum peak error for a specific $\Delta t$, since the increase in the number of destination cells up to 25 (i.e. $k_x = k_y = 2$) significantly reduces this error (Figure 3.8). For higher $k_x$ and $k_y$ values, this error is essentially temporal, which implies a decrease in $\Delta t$ to obtain better results. The maximum negative concentration cannot be considered residual only for the simulation with 9 destination cells (Figure 3.8).

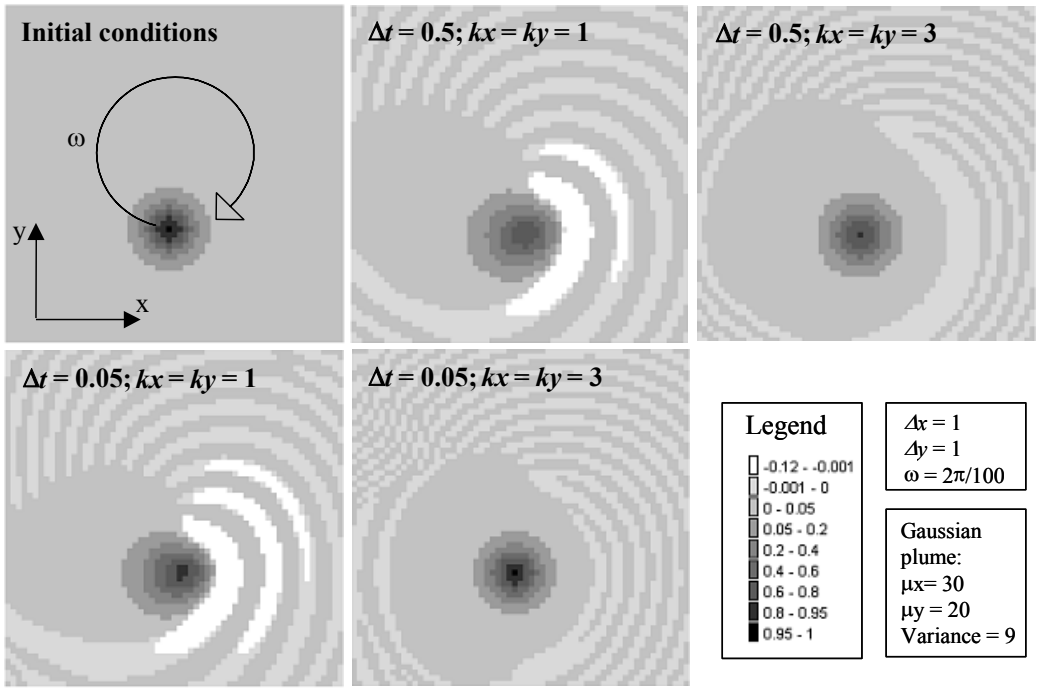



Figure 3.7 –One turn of rotation, with different $\Delta$ts and number of destination cells

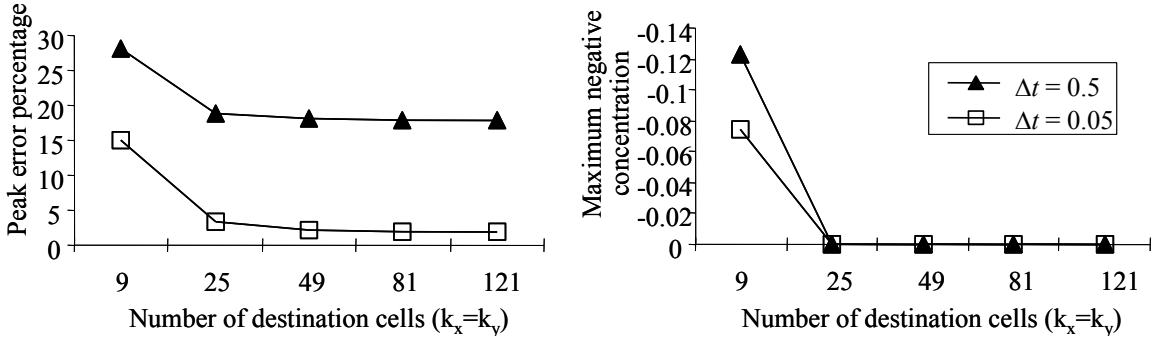



Figure 3.8 – Peak error percentage and Maximum negative concentration

### 3.3.2.2 Tagus estuary

In this section, the model is applied to the Tagus Estuary, so that its behavior may be better evaluated in a complex flow system. The hydrodynamic data was interpolated from a finite element model with an unstructured grid [23]. The computational domain was discretized in 500×589 cells with $\Delta x = \Delta y = 100$ m and its geographical representation can be seen in Figure 3.9. Six tests were carried out to assess the importance of $\Delta t$, $k_x$ and $k_y$ in DisPar-k results. The first three tests had a $\Delta t = 600$s and the particle destination square was composed respectively of (2×1+1) × (2×1+1),(2×3+1)×(2×3+1) and (2×5+1)×(2×5+1) nodes. The second set of tests was obtained with the same three particle destination squares, but with a shorter temporal resolution ($\Delta t = 120$s). All the tests were obtained for pure advection ($D_x = D_y = 0$ ms$^{-2}$), and the total simulation time was 17 hours. The initial condition is a Gaussian plume with a standard deviation of about 447 m (Figure 3.9).

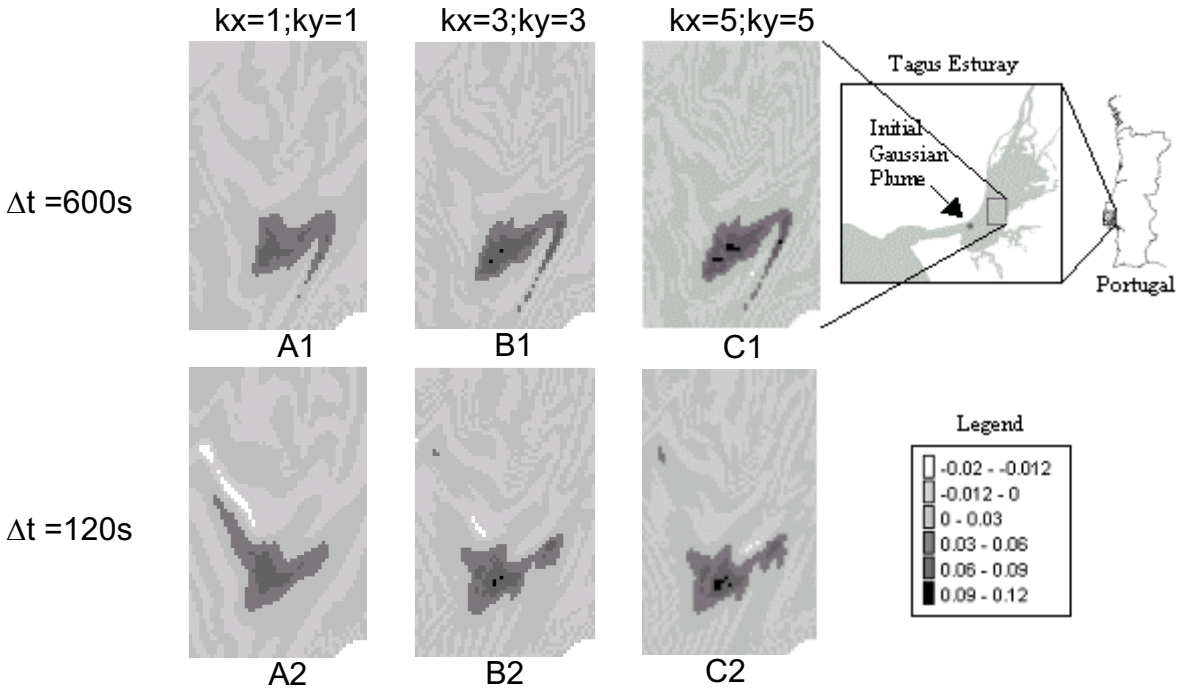



Figure 3.9 – Results for the Tagus estuary with two different time steps (600s and 120s) and three different particle destination rectangles (1×1, 3×3, 5×5)

From Figure 3.9 it is possible to observe that the increase of temporal resolution has changed the plume in the three particle destination squares (A, B, C) with special emphasis on the first one. Negative values (-0.02 to –0.012) are much more expressive in situation A2 since temporal error is no longer disguising spatial error. As was theoretically predicted, this last error was reduced by

increasing the particle destination cells (B2 and C2), making the plume much more definite. In tests B and C the results showed some physical incoherence since the plume peak has increased to values ranging from 0.09 (initial peak value) up to 0.12.

## 3.4 Conclusions

This chapter introduces DisPar-k, a numerical method based on the displacement moments of a Markov particle. The method has a semi-Lagrangian nature to avoid problems associated to the Courant number. The results obtained in both theoretical and practical situations are in agreement with the relation defined in chapter 2 between truncation errors and particle displacement moments.

Besides the semi-Lagrangian numerical possibilities and associated numerical power to hold problems related with the time step, it is possible to define the desired numerical accuracy. However, in practical situations, more robust techniques must be used to work with large time steps to improve the particle displacement average evaluation. The explicit nature of the model creates another problem with the dispersion component since it is mostly dominated by temporal error. To hold large time steps and relatively high dispersion coefficients, the particle displacement distribution must be discretized in many units, which can lead to round off errors or prohibitive computational costs. Another shortcoming associated with this model is the fact that it can only be applied to uniform meshes, removing versatility in practical applications. The strategies used to overcome the first limitation are well known by the Eulerian-Lagrangian community, but how to work with an explicit formulation that simultaneously has high time steps and dispersion? How to work with non-regular grids? In chapter 4 the answer to these two questions will be given by developing another formulation based on particle displacement moments for regular/non-uniform grids.

# 4 Particle Distribution Model applied to non-uniform/regular grid

## *4.1 Introduction*

DisPar-k is a powerful method for the simulation of advection-diffusion, but it is limited to simple regular grids. The model is very powerful to deal with advection, yet the dispersion component cannot be so well treated if high time steps are to be used. To overcome these two limitations a more general model called DisParV is developed in this chapter [11]. DisParV is designed to deal with non-uniform grids for 1D simulation and to work with regular/non-uniform grids in the 2D case. This was made possible by explicitly using the concept of volumes instead of nodes as it was done in DisPar-k. Besides the development of a more versatile model, with this chapter it is intended to give the necessary foundations for the consideration of unstructured grids in any dimension. As a first approach to volumes, it is assumed that the particle is uniformly distributed in each domain unit simplifying, though, the underlying mathematical treatments.

In DisParV the particle displacement distribution is also discretized in a user specified number of units. However, unlike with DisPar-k, in this model the Gaussian discretization process can be done in non-regular units. This way, it is possible to overcome the shortcomings associated with dispersion through the aggregation of volumes. This approach permits the creation of an explicit numerical method capable of dealing with high time steps without any stability problems. However it should be noted that this last statement is solely theoretically formulated, since it was not possible to perform particular tests, due to the author's time constraints.

This chapter starts with the introduction of the DisParV concept, describing how the correspondent system of equations can be rewritten as a Vandermonde system in order to improve its computational solution (Section 4.2). Section 4.2.3 presents some comparisons with known analytical solutions. Section 4.4 outlines some ideas on how to build an ever-stable model without numerical dispersion.

Finally, section 5 points out some future directions toward the management of unstructured grids.

## *4.2 DisParV CONCEPT*

### 4.2.1 One-dimensional concept

As DisPar-k, DisParV is a numerical formulation for the simulation of advection-diffusion processes based on the particle displacement moments. DisParV also assumes that the displacement distribution is Gaussian with an average and variance defined by establishing a relation between Fokker-Planck and (2.14) transport equations (2.9).

$$\langle x \rangle = \left( u_x + \frac{\partial D_x}{\partial x} + \frac{D_x}{h}\frac{\partial h}{\partial x} \right) dt \tag{4.1}$$

$$\sigma^2(x) = \left\langle \left( x - \langle x \rangle \right)^2 \right\rangle = 2D_x dt \tag{4.2}$$

where $\langle x \rangle$ = particle displacement average over the x-axis; $\sigma 2[x]$ = particle displacement variance over the x-axis; $u_x$ = velocity over the x-axis; $D_x$ = dispersion coefficient over the x-axis; h = water depth.

The method divides the Gaussian distribution into a specified number of discrete units with a uniform distribution inside each one. These discrete units are selected according to the grid shape and their choice is made following a semi-Lagrangian approach. The central unit of the possible destination regions is selected according to the physical parameters associated with the particle initial position and to the numerical parameter time step ($\Delta t$). Around the central unit of the possible particle destination volumes, the method assumes that there are respectively k volumes to the upstream and downstream sides. The particle displacement distribution is therefore divided in 2k+1 units, with the middle unit associated with the Gaussian peak. In DisParV, the selection of the particle destination central volume for a particle initially located in volume i (V[i]) is based on the average displacement after a time step of a particle located in $x_i$ (V[i] upstream position). The volume where this average value falls represents the central volume and is defined by its index as $\beta_i$ (Figure 4.1).

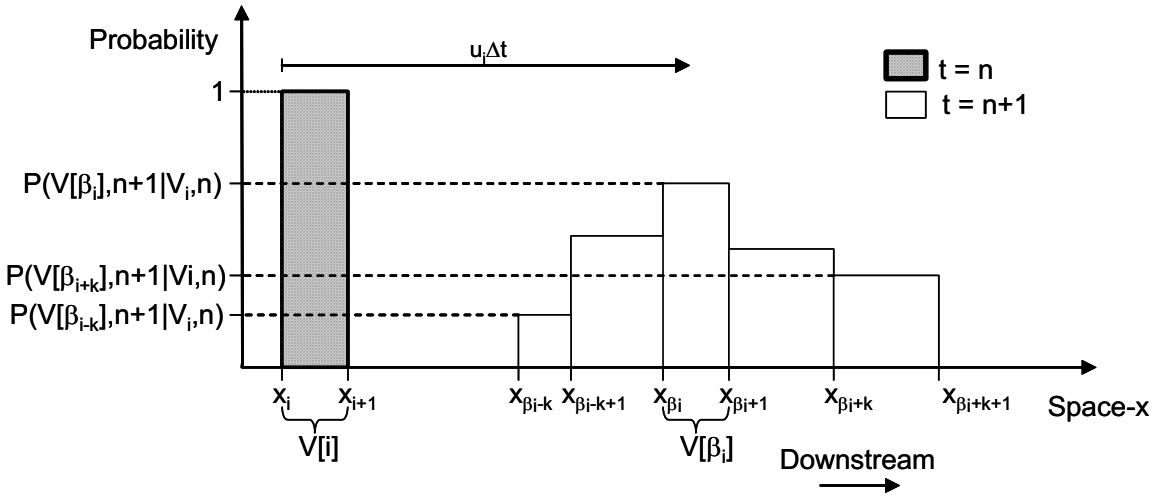



Figure 4.1 - Possible events for a particle displacement after a $\Delta t$

In this formulation, there are two independent variables, the particle position in V[i], $\chi_i$ ($\chi_i \in [x_i,x_i+1]$), and the displacement after a time step caused by the transport parameters for a particle inside volume V[i]($\delta x_i$). As it was mentioned, DisParV assumes a uniform distribution for $\chi_i$ and, therefore, its moment of order a can be yielded as:

$$\left\langle \left(\chi_i\right)^a \right\rangle = \frac{x_{i+1}^{a+1} - x_i^{a+1}}{\left(a+1\right)\left(x_{i+1} - x_i\right)}, \qquad \chi_i \in \left[x_i, x_{i+1}\right] \tag{4.3}$$

On the other hand, $\delta x$ assumes a Gaussian distribution, allowing the evaluation of all particle displacement moments based on the average and variance knowledge [20]:

$$\left\langle \left(\delta x_i\right)^a \right\rangle = \sum_{m=0}^{\rho-1} \frac{a!}{2^m\, m!\left(a-2m\right)!} \left\langle \left(\delta x_i - \left\langle \delta x_i \right\rangle\right)^2 \right\rangle^m \left\langle \delta x_i \right\rangle^{a-2m} \tag{4.4}$$

where $\rho$=(a+2)/2 if a is even or $\rho$=(a+1)/2 if a is odd.

To evaluate the particle displacement probabilities, it is necessary to formulate a relation between numerical probabilities and the displacement moments of ($\chi_i$+$\delta x_i$), to guarantee that the particle will always have a uniform distribution inside volumes. This can be achieved by respecting the following system:

$$
\begin{cases}
\displaystyle\sum_{r=-k}^{k}\left(\frac{x^{1}_{\beta_i+r+1}-x^{1}_{\beta_i+r}}{(1)\left(x_{\beta_i+r+1}-x_{\beta_i+r}\right)}P\left(V\left[\beta_i+r\right],n+1\middle|V\left[i\right],n\right)\right)=\left\langle\left(\chi_i+\delta x_i\right)^{0}\right\rangle \\
\displaystyle\sum_{r=-k}^{k}\left(\frac{x^{2}_{\beta_i+r+1}-x^{2}_{\beta_i+r}}{(2)\left(x_{\beta_i+r+1}-x_{\beta_i+r}\right)}P\left(V\left[\beta_i+r\right],n+1\middle|V\left[i\right],n\right)\right)=\left\langle\left(\chi_i+\delta x_i\right)^{1}\right\rangle \\
\vdots \\
\displaystyle\sum_{r=-k}^{k}\left(\frac{x^{2k}_{\beta_i+r+1}-x^{2k}_{\beta_i+r}}{(2k)\left(x_{\beta_i+r+1}-x_{\beta_i+r}\right)}P\left(V\left[\beta_i+r\right],n+1\middle|V\left[i\right],n\right)\right)=\left\langle\left(\chi_i+\delta x_i\right)^{2k-1}\right\rangle \\
\displaystyle\sum_{r=-k}^{k}\left(\frac{x^{2k+1}_{\beta_i+r+1}-x^{2k+1}_{\beta_i+r}}{(2k+1)\left(x_{\beta_i+r+1}-x_{\beta_i+r}\right)}P\left(V\left[\beta_i+r\right],n+1\middle|V\left[i\right],n\right)\right)=\left\langle\left(\chi_i+\delta x_i\right)^{2k}\right\rangle
\end{cases}
\tag{4.5}
$$

To evaluate the 2k+1 probabilities the system (4.5) is solved and probabilities are expressed as function of the particle displacement moments. These numerical probabilities are used as coefficients for mass transfers between domain volumes.

In the appendix 10 it is proved that if the computational domain has a constant $\Delta x$, this formulation produces exactly the same probabilities as DisPar-k if k and $\beta$ are the same in both methods. As it was proved in chapter 3, the spatial error decreases for linear situations as the number of particle destination units increases, which means that this will also happen with DisParV for this specific situation. This relation for constant $\Delta x$ suggests that DisParV spatial accuracy will also grow with k for non-uniform grids and it will be strengthen afterwards by a comparison between numerical results and analytical solutions.

To simplify the computational implementation, expression (4.5) is transformed into a Vardermonde system. This system is well known and algorithms with very good performance carrying on low round-off errors can be found, for example, in [26] or [48]. The generic expectation of $\chi_i+\delta_i$ of order j can be rewritten, such that the x divisors are put in evidence as:

$$
\left\langle\left(\chi_i+\delta x_i\right)^{j}\right\rangle=\sum_{r=-k}^{k}\left(\frac{x^{j+1}_{\beta_i+r+1}-x^{j+1}_{\beta_i+r}}{(j+1)\left(x_{\beta_i+r+1}-x_{\beta_i+r}\right)}P\left(V\left[\beta_i+r\right],n+1\,|\,V\left[i\right],n\right)\right) \tag{4.6}
$$

$$\left\langle \left(\chi_i + \delta x_i\right)^j \right\rangle = \frac{P\left(V\left[\beta_i - k\right], n+1 \mid V\left[i\right], n\right)}{\left(j+1\right)\left(x_{\beta_i - k} - x_{\beta_i - k+1}\right)} x_{\beta_i - k} +$$
$$+\left( \frac{P\left(V\left[\beta_i - k+1\right], n+1 \mid V\left[i\right], n\right)}{\left(j+1\right)\left(x_{\beta_i - k+1} - x_{\beta_i - k+2}\right)} - \frac{P\left(V\left[\beta_i - k\right], n+1 \mid V\left[i\right], n\right)}{\left(j+1\right)\left(x_{\beta_i - k} - x_{\beta_i - k+1}\right)} \right) x_{\beta_i - k+1} +$$
$$+\cdots+ \tag{4.7}$$
$$\left( \frac{P\left(V\left[\beta_i + k\right], n+1 \mid V\left[i\right], n\right)}{\left(j+1\right)\left(x_{\beta_i + k} - x_{\beta_i + k+1}\right)} - \frac{P\left(V\left[\beta_i + k-1\right], n+1 \mid V\left[i\right], n\right)}{\left(j+1\right)\left(x_{\beta_i + k-1} - x_{\beta_i + k}\right)} \right) x_{\beta_i + k} -$$
$$-\frac{P\left(V\left[\beta_i + k\right], n+1 \mid V\left[i\right], n\right)}{\left(j+1\right)\left(x_{\beta_i + k} - x_{\beta_i + k+1}\right)} x_{\beta_i + k+1}$$

To redefine system (4.5) as function of the independent variable x, one must do it by multiplying both sides of equation (4.7) by (j+1) and by introducing a new dimension. So, the system of equations (4.5) can be written as a Vandermonde system in matrix notation according to the following matrices:

$$W_{V[i]} = \begin{bmatrix} \dfrac{P\left(V\left[\beta_i - k\right], n+1 \middle| V\left[i\right], n\right)}{x_{\beta_i - k} - x_{\beta_i - k+1}} \\ \dfrac{P\left(V\left[\beta_i - k+1\right], n+1 \middle| V\left[i\right], n\right)}{x_{\beta_i - k+1} - x_{\beta_i - k+2}} - \dfrac{P\left(V_{[\beta_i - k]}, n+1 \middle| V\left[i\right], n\right)}{x_{\beta_i - k} - x_{\beta_i - k+1}} \\ \vdots \\ \dfrac{P\left(V\left[\beta_i + k\right], n+1 \middle| V\left[i\right], n\right)}{x_{\beta_i + k} - x_{\beta_i + k+1}} - \dfrac{P\left(V\left[\beta_i + k-1\right], n+1 \middle| V\left[i\right], n\right)}{x_{\beta_i + k-1} - x_{\beta_i + k}} \\ -\dfrac{P\left(V\left[\beta_i + k\right], n+1 \middle| V\left[i\right], n\right)}{x_{\beta_i + k} - x_{\beta_i + k+1}} \end{bmatrix}_{((2k+1)+1)} \tag{4.8}$$

$$M = \begin{bmatrix} \left(x_{\beta_i - k}\right)^0 & \cdots & \left(x_{\beta_i}\right)^0 & \cdots & \left(x_{\beta_i + k}\right)^0 & \left(x_{\beta_i + k+1}\right)^0 \\ \left(x_{\beta_i - k}\right)^1 & \cdots & \left(x_{\beta_i}\right)^1 & \cdots & \left(x_{\beta_i + k}\right)^1 & \left(x_{\beta_i + k+1}\right)^1 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \\ \vdots & \ddots & \ddots & \ddots & \vdots & \\ \left(x_{\beta_i - k}\right)^{2k} & \cdots & \left(x_{\beta_i}\right)^{2k} & \cdots & \left(x_{\beta_i + k}\right)^{2k} & \left(x_{\beta_i + k+1}\right)^{2k} \\ \left(x_{\beta_i - k}\right)^{2k+1} & \cdots & \left(x_{\beta_i}\right)^{2k+1} & \cdots & \left(x_{\beta_i + k}\right)^{2k+1} & \left(x_{\beta_i + k+1}\right)^{2k+1} \end{bmatrix}_{((2k+1)+1)((2k+1)+1)} \tag{4.9}$$

$$E = \begin{bmatrix} 0 \\ 1\left\langle \left(\chi_i + \delta x_i\right)^0 \right\rangle \\ \vdots \\ 2k\left\langle \left(\chi_i + \delta x_i\right)^{2k-1} \right\rangle \\ (2k+1)\left\langle \left(\chi_i + \delta x_i\right)^{2k} \right\rangle \end{bmatrix} \tag{4.10}$$

as

$$MW_{V[i]} = E_{V[i]} \tag{4.11}$$

It is possible to verify that the first line from this system is just auxiliary, not representing any particle displacement moment. As it was mentioned, this system will have one dimension more, but its computational solution will be worthwhile by the associated mathematical knowledge.

The matrix holding the displacement probabilities is expressed as function of $\langle\chi_i\rangle$ and $\langle\delta x_i\rangle$ in accordance to the system:

$$W_{V[i]} = M^{-1}E_{V[i]} \tag{4.12}$$

To solve this system, one must take into consideration that $\chi_i$ and $\delta x$ are independent variables and therefore it is possible to express any expectation of $\chi_i + \delta x_i$ as function of $\langle\chi_i\rangle$ and $\langle\delta x_i\rangle$ by the binomial theorem application

$$\left\langle \left(\chi_i + \delta x_i\right)^a \right\rangle = \sum_{r=0}^{a} \binom{a}{r} \left\langle \left(\chi_i\right)^r \right\rangle . \left\langle \left(\delta x_i\right)^{a-r} \right\rangle \tag{4.13}$$

From a practical perspective, instead of using (4.4) to calculate the particle displacement moments, it is possible calculate Gaussian expectations as function of the previous calculated moments and the known average and variance as:

$$\left\langle \delta x_i^{j+2} \right\rangle = \left\langle \delta x_i \right\rangle \left\langle \delta x_i^{j+1} \right\rangle + (j+1)\sigma^2\left[\delta x_i\right]\left\langle \delta x_i^{j} \right\rangle \tag{4.14}$$

By using this relation, the number of computational operations to obtain the particle displacement moments decreases, increasing the computational performance.

Finally, after the system is solved, P(V[$\beta_i$+k],n+1| V[i],n) or P(V[$\beta_i$-k],n+1| V[i],n) should be the first probabilities to be calculated. For example, if the first obtained probability is the one associated with the last $W_{V[i]}$ entry, P(V[$\beta_i$+k],n+1|

V[i],n), the process follows further on by replacing the calculated probability in the previous $W_{V[i]}$ entry. This will be repeatedly done until no unknown probability is left.

### 4.2.2 Two-Dimensional concept

The 2-D DisParV concept is based on the 1-D DisParV scheme applied independently to each dimension, as it was presented in the previous section. Particle displacement moments over the y-axis and particle initial position moments can be obtained for the second spatial dimension (y axis) as it was done for the x-axis, by also assuming a Gaussian distribution for the particle displacement with an average and variance like:

$$\langle y \rangle = \left( y + \frac{\partial D_y}{\partial y} + \frac{D_y}{h}\frac{\partial h}{\partial y} \right) dt \tag{4.15}$$

$$\sigma^2(y) = 2D_y dt \tag{4.16}$$

where $\langle y \rangle$ = particle displacement average over the y-axis; $\sigma 2[y]$ = particle displacement variance over the y-axis; $u_y$ = velocity over the y-axis; $D_y$ = dispersion coefficient over the y-axis.

The product of the independent probabilities produces the 2-D probability distribution for a particle displacement. Thus, the probability for a particle to move from volume (i, j) to (x, y) over the time step, $P(V[x,y]\ ,\ n+1 \mid V[i,j]\ ,\ n)$, is equal to the product of $P(V[x],\ n+1 \mid V[i,j],\ n)$ and $P(V[y],\ n+1 \mid V[i,\ j]\ ,\ n)$. The region for the particle possible destination has $(2k_x+1)\times(2k_y+1)$ volumes, as can be observed in Figure 4.2:

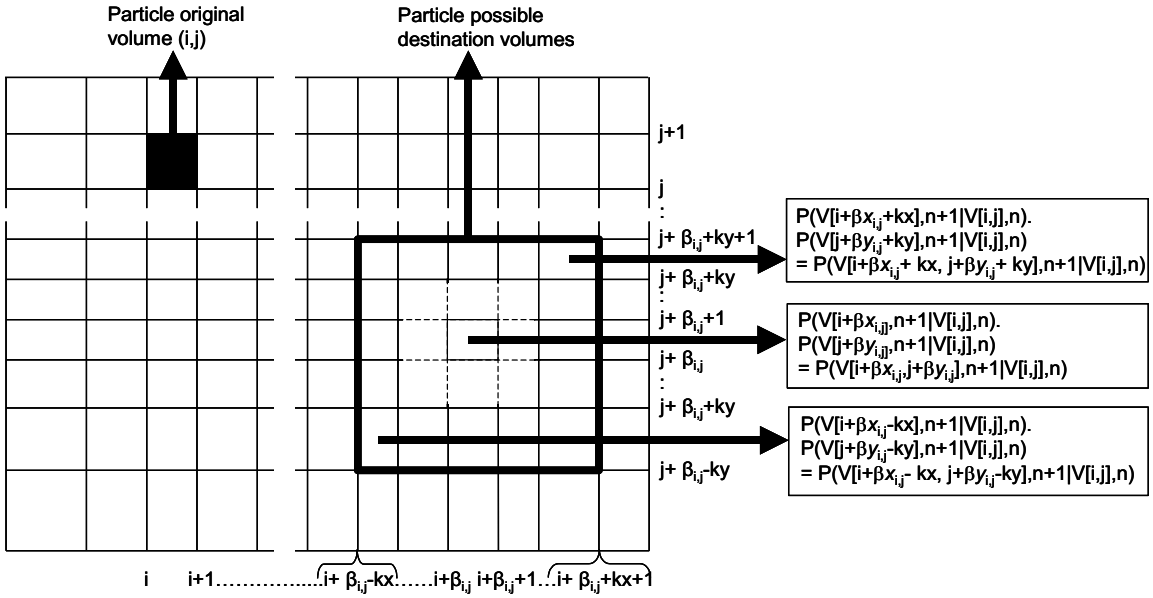



Figure 4.2 - Possible events for a particle in a time step

The mass transfers between volumes over a time step is directly evaluated, so that the mass transfer from volume (i,j) to volume (x,y) is simply given by the product of volume (i,j) particle mass at time n by $P(V[x,y]\ ,\ n+1 \mid V[i,\ j]\ ,\ n)$.

## 4.2.3 Boundaries

### 4.2.3.1 Open Boundaries

For the simulation of open boundaries, a virtual grid is created according to the computational domain edge columns and rows. Volumes on the computational domain left side are assumed to have a $\Delta x$ value equal to the leftmost column and volumes on the right side will assume the $\Delta x$ values of the right most column. The volumes outside the y-axis extremes are also defined based on the same principles, respectively assuming the $\Delta y$ values of the top most and bottom most rows (Figure 4.3). Mass transfers for this virtual region are just removed from the computational domain.

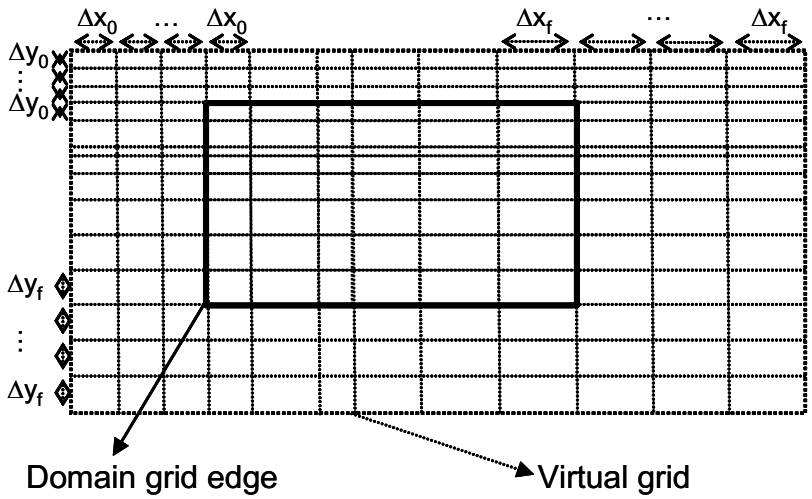



Figure 4.3 – Virtual grid used to calculate particle displacement probabilities outside domain

#### 4.2.3.2 Land Boundaries

Land volumes are treated exactly in the same way as in the previous chapter. Therefore, if the particle destination rectangle intercepts a land volume, the mass to be transferred to it will remain in the source volume.

## *4.3 1-D Gaussian plume tests*

The accuracy of DisParV was tested by two linear situations with the initial condition of a Gaussian profile. In the first case the space was dived in a regular grid ($\Delta x = 200$; $u = 10$; $D = 0$; $\Delta t = 24$; number of volumes = 64; time step number = 25; initial Gauss hill, average = 2000 and standard deviation = 264). In the second situation, space was divided into a mesh where $\Delta x$ varies linearly from 4 (most upstream volume) to 1 (central volume) and then it changes symmetrically towards downstream ($u=0.7$; $D = 0$; $\Delta t = 1$; number of volumes = 71; time step number = 100; initial Gauss hill, average = 45 and standard deviation = 4). Observing Figure 4.4 and Figure 4.5 it is possible to verify in both tests the model accuracy increase as k changes from 1 to 6.

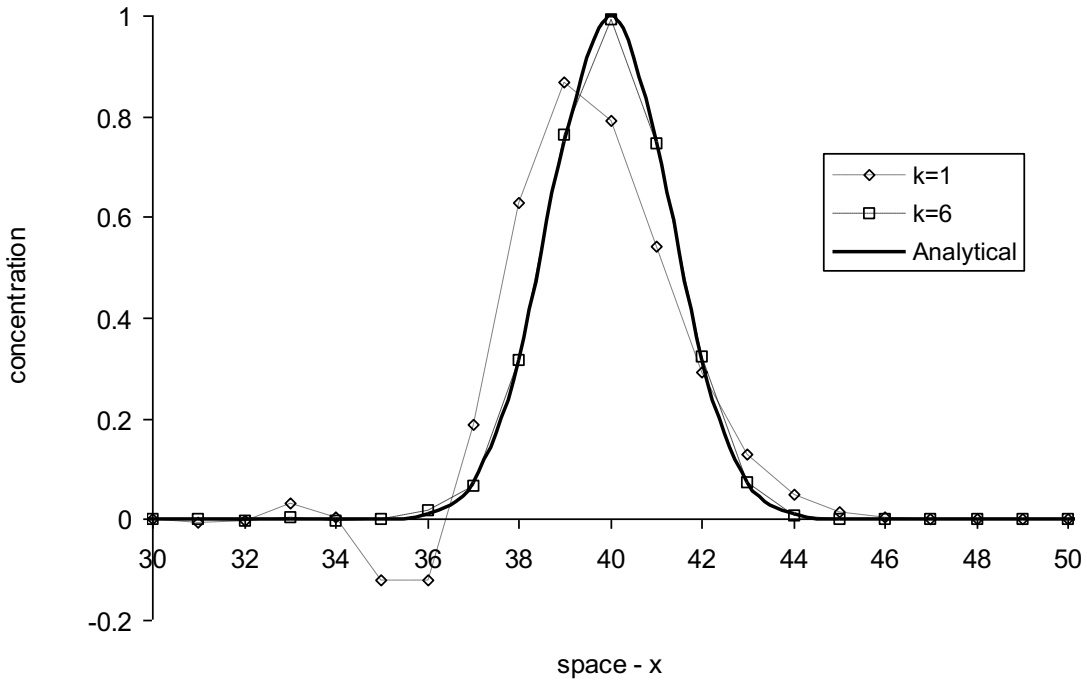



Figure 4.4 - Uniform grid test

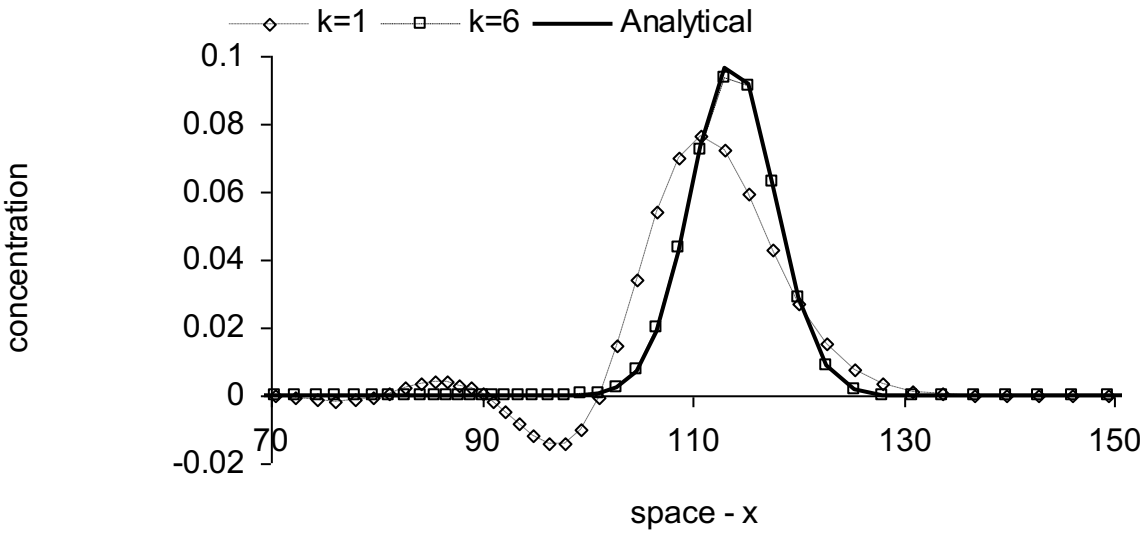



Figure 4.5 - Non-uniform grid test

## *4.4 Volume aggregation as a tool to deal with high dispersion coefficients*

As it was previously mentioned, DisParV is powerful to treat the advection component, yet some limitations can be found if high values of dispersion are to be used. Instabilities caused by the dispersion component, happen when the Gaussian distribution has 1 or 2% of its area outside the destination region. If the

formulation is kept explicit, this type of problems can only be solved by increasing the number of units discretizing the particle displacement distribution or by aggregating domain volumes. The first approach can create problems with round off errors by the size of the system to be solved or can have prohibitive computational costs to correct this type of errors. On the other hand, the second approach can solve the problem with a discretization of the Gaussian distribution by resorting to only three volumes. For example, if the time step used $\Delta t_1$ guarantees that the Gaussian distribution will be almost all inside the three volumes closest to the particle displacement average, the model can be applied without aggregation (Figure 4.6). On the other hand, if the time step implies that the Gaussian curve has a higher standard deviation like in Figure 4.7, volumes $V[\beta_i-1]$ and $V[\beta_i+1]$ can be given by respectively aggregating two domain units in each volume. By doing this, it will be guaranteed that the Gaussian distribution will be almost all inside the destination region.

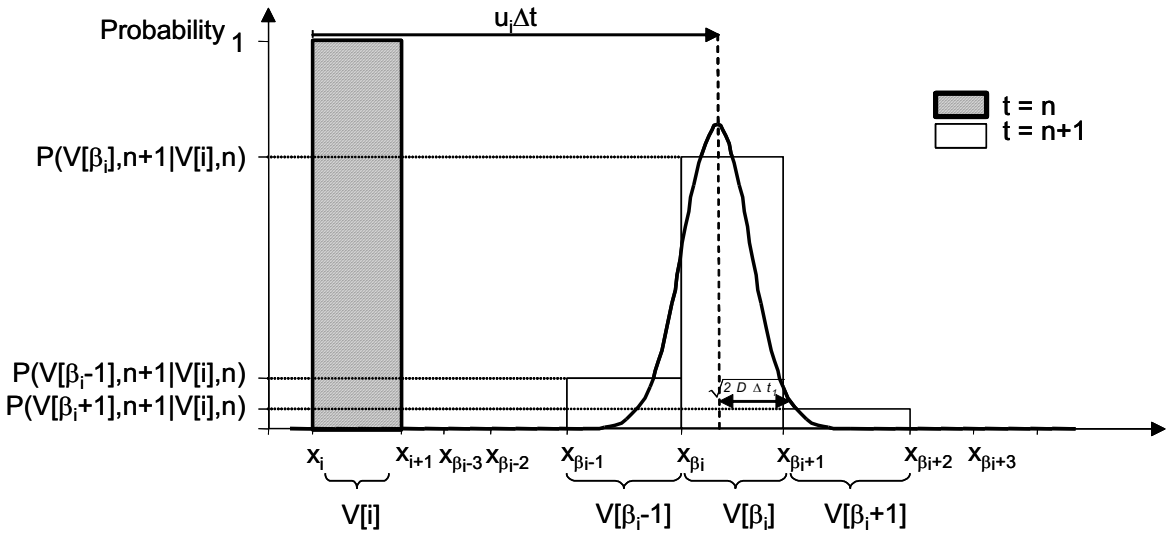



Figure 4.6 –Particle displacement distribution for a $\Delta t_1$ discretization in three volumes

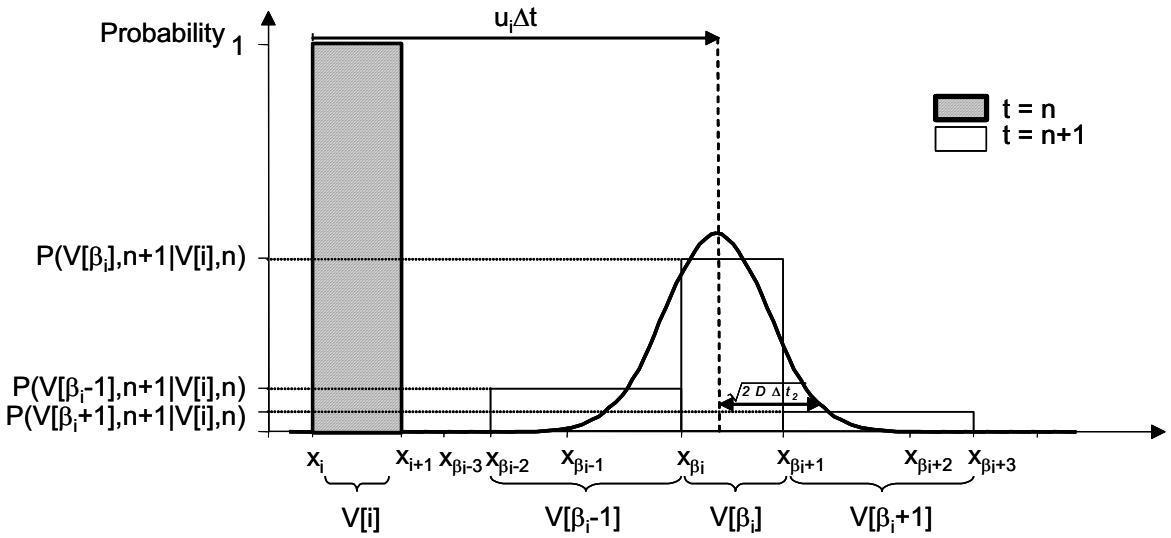



Figure 4.7 –Particle displacement distribution for a $\Delta t_2$ discretization in three volumes. The first and last volumes are obtained by respectively aggregating two grid volumes.

However, this is only an idea to be studied in future developments. To implement this strategy, some mathematical studies must be done to understand the relation between what will be the minimum area of the Gaussian distribution to be inside the destination region before wiggles appearing. If this relation is found and a higher order method to track the particle displacement average is used, it will be possible to build an unconditionally stable model without resorting to numerical inconsistencies, like the introduction of numerical dispersion. If this model shows some instability, it could be only caused by problems outside the numerical formulation itself. Therefore, must of the possible expected instabilities are expected to be a consequence of hydrodynamic errors or an effect of the non-linearities associated rough bathymetric representations or even a consequence of both [45].

## *4.5 Conclusion*

This chapter introduces DisParV, a numerical formulation for advection-diffusion based on the displacement moments of a Markov particle. The model consists in a more general version of DisPar-k, since it allows working with regular/non-uniform grids. It was proven that if DisParV is applied to uniform grids it produces exactly the same numerical probabilities for the particle displacement as DisPar-k. Considering this relation, the relation presented in chapter 2 an the comparisons between numerical results and analytical solutions, it is strongly suggested that this formulation will also correct the truncation error up to the order

given by the number of units used in the Gaussian discretization process. Similarly to DisPar-k, the dispersion component in DisParV represents the main shortcoming associated with the formulation. However, by aggregating several spatial units this problem will probably be overcome without the need of any implicit formulation.

This model has created the theoretical foundations toward unstructured grids with any number of spatial dimensions. Therefore, it is expected that DisParV establish basis for new developments in the most flexible type of grids. In future developments, other distributions for the particle presence inside each volume can also be considered independently of the grid shape. This type of approaches somehow follows the approach used in finite-elements, with the main difference that the internal shapes will be given by the particle displacement moments.

# 5 Numerical dispersion caused by non-linearities: Eulerian example with three particle destination units

## *5.1 Introduction*

Many studies for linear conditions have been done to appraise numerical errors in either Eulerian or Eulerian-Lagrangian methods. When a new model to solve the advection-diffusion equation is developed, its numerical power is assessed by the associated truncation error and/or by comparing it with known analytical solutions. However, most of these analyzes are carried out for linear situations leaving the problems related to non-linearities to be appraised in some near future. For years, the lack of studies in this area has been fostering much confusion about the meaning of many practical results, since they live soaked up by numerical errors associated with non-linearities (i.e. linearity is relative to the advection-diffusion parameters). In many cases, parameterization will only work for that specific numerical method, completely failing if another formulation is to be applied with the same physical parameters. A well-known model is QUAL 2E [8]. This model was built to be always stable regardless of the physical parameters used by changing the physics of the simulated phenomena.

For all this, the first DisPar method [15] will be introduced and compared with DisPar-k in non-linear situations. The former DisPar method was built so that a homogeneous concentration profile can be kept unaltered independently of the spatial variability associated with the advection-diffusion parameters. By comparing it with DisPar-k in non-linear situations, it is expected to obtain a better perspective over numerical dispersion and non-linear applications. DisPar is part of this work, despite the fact that in this chapter it is shown a bit differently from the published paper [15].

This chapter is organized as follows. Section 5.2 introduces the DisPar method. In section 5.3 it is proved that for small cells and for small Markov jumps, the particle displacement average and variance converge to the ones obtained by

comparing the Fokker-Planck and advection-diffusion equations. Finally, the chapter ends (section 5.4) with a comparison between DisPar and DisPar-k both applied to the same non-linear situations.

## 5.2 Eulerian DisPar model for three particle destination cells

In this section an Eulerian version of DisPar models for three particle destination units is to be done. To be in accordance with name given in [15], the model developed in this chapter is solely called DisPar. The model was built to guarantee that no mass imbalance exists independently from the variability associated with the advection-diffusion parameters. To do so, advection and dispersion are separately treated by respectively analyzing each average and each variance. Since the average and variance of two independent processes are both given by the sum of each independent process, it is possible to define the following relation:

$$\left\langle x_{adv} + x_{disp} \right\rangle = \left\langle x_{adv} \right\rangle + \left\langle x_{disp} \right\rangle \tag{5.1}$$

$$\sigma^2 \left( x_{adv} + x_{disp} \right) = \sigma^2 \left( x_{adv} \right) + \sigma^2 \left( x_{disp} \right) \tag{5.2}$$

where $x_{adv}$ = particle position influenced by the flow motion; $x_{disp}$ = particle position influenced by the turbulence.

After calculating the four particle displacement statistical parameters, the Eulerian numerical probabilities for the particle displacement will be given by (Figure 5.1):

$$P\left(i-1,n+1 \mid i,n\right) = \frac{1}{2}\left( \sigma_i^2 \left( x_{adv} + x_{disp} \right) + \left( \left\langle x_{adv} + x_{disp} \right\rangle_i \right)^2 - \left\langle x_{adv} + x_{disp} \right\rangle_i \right) \tag{5.3}$$

$$P\left(i,n+1 \mid i,n\right) = 1 - \sigma_i^2 \left( x_{adv} + x_{disp} \right) - \left( \left\langle x_{adv} + x_{disp} \right\rangle_i \right)^2 \tag{5.4}$$

$$P\left(i+1,n+1 \mid i,n\right) = \frac{1}{2}\left( \sigma_i^2 \left( x_{adv} + x_{disp} \right) + \left( \left\langle x_{adv} + x_{disp} \right\rangle_i \right)^2 + \left\langle x_{adv} + x_{disp} \right\rangle_i \right) \tag{5.5}$$

where $\langle x \rangle_i$ = particle displacement average associated with cell i and is centered in the origin cell; $\sigma_i^2(x)$ = particle displacement variance associated with cell i;

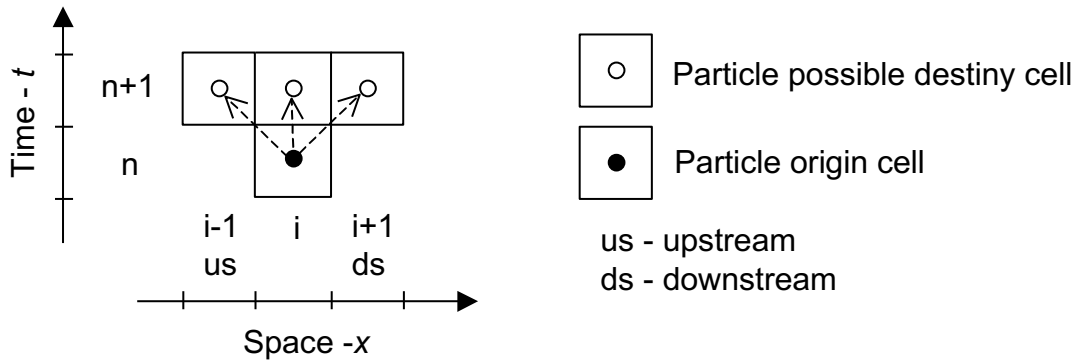



Figure 5.1 – Eulerian scheme for the particle displacement distribution discretized in units

### 5.2.1 Advective Displacement Average and Variance

The average displacement of a particle due to advection in one time step is simply the product of velocity ($u_i$) by $\Delta t$. In a discrete space with constant cell length ($\Delta x$), the particle spatial movement can become dimensionless when written as $u_i \Delta t/\Delta x$:

$$\langle x_{adv} \rangle_i = \frac{u_i \Delta t}{\Delta x} \tag{5.6}$$

Since the advection component is deterministic by definition, its variance is zero and thus:

$$\sigma_i^2 \left( x_{adv} \right) = 0 \tag{5.7}$$

### 5.2.2 Dispersive Displacement Average and Variance

The dispersion process is basically a consequence of the non-resolved advective water movements. These movements can only be represented statistically and the traditional parameter representing them is known as the dispersion coefficient or Fickian coefficient (D).

Mass conservation implies that, in a time step, the average masses of water that move by dispersion from cell i into cell i-1, and from cell i-1 into cell i are equal, i.e.:

$$Q\left(i-1,n+1 \mid i,n\right) = Q\left(i,n+1 \mid i-1,n\right) \tag{5.8}$$

$$Q\left(i+1,n+1 \mid i,n\right) = Q\left(i,n+1 \mid i+1,n\right) \tag{5.9}$$

where $Q_i^{disp\_us}$ and $Q_{i-1}^{disp\_ds}$ are the average flow moving from cell i into cell i-1 and from i-1 into i, respectively.

Hence, the flow $Q_i^{disp\_us}$ ($Q_i^{disp\_ds}$) must be a function of both Fickian coefficients $D_i$ and $D_{i-1}$ ($D_i$ and $D_{i+1}$), i.e., both Fickian coefficients reflect the quantity of water transferred between neighboring cells. Dividing each coefficient by the corresponding cell length and multiplying it by the section area can evaluate these flows:

$$Q(i-1,n+1 \mid i,n) = \frac{1}{2}\left(\frac{A_i D_i}{\Delta x_i} + \frac{A_{i-1} D_{i-1}}{\Delta x_{i-1}}\right) \tag{5.10}$$

$$Q(i+1,n+1 \mid i,n) = \frac{1}{2}\left(\frac{A_i D_i}{\Delta x_i} + \frac{A_{i+1} D_{i+1}}{\Delta x_{i+1}}\right) \tag{5.11}$$

where $A_{i-1}$, $A_i$ and $A_{i+1}$ correspond to the section areas associated with cells i-1, i and i+1, respectively.

For constant cell length the average dispersion velocities in cell i can be given by:

$$d(i-1,n+1 \mid i,n) = \frac{1}{2}\frac{(A_i D_i + A_{i-1} D_{i-1})}{A_i \Delta x} \tag{5.12}$$

$$d(i+1,n+1 \mid i,n) = \frac{1}{2}\frac{(A_i D_i + A_{i+1} D_{i+1})}{A_i \Delta x} \tag{5.13}$$

where $d_i^{disp\_us}$ = cell i upstream average dispersion velocity and $d_i^{disp\_ds}$ = cell i downstream average dispersion velocity.

Assuming that a particle is uniformly distributed in cell i, it has the same probability to be transported with the blocks of water that move into cell i-1 and with those that move into cell i+1. This means that the average dimensionless velocity can represent the particle dispersion probability:

$$P^{disp}(i-1,n+1 \mid i,n) = d(i-1,n+1 \mid i,n)\frac{\Delta t}{\Delta x_i} \tag{5.14}$$

$$P^{disp}(i+1,n+1 \mid i,n) = d(i+1,n+1 \mid i,n)\frac{\Delta t}{\Delta x_i} \tag{5.15}$$

where $P_i^{disp}\left(x,n+1\,|\,x',n\right)$= probability that the particle will move from cell x' into cell x due to dispersion.

Introducing (5.12) and (5.13) respectively in (5.14) and (5.15), the probabilities can be rewritten, for constant cell length as:

$$P^{disp}\left(i-1,n+1\,|\,i,n\right)=\frac{A_{i-1}D_{i-1}+A_jD_j}{2A_i\Delta x}\frac{\Delta t}{\Delta x} \tag{5.16}$$

$$P^{disp}\left(i+1,n+1\,|\,i,n\right)=\frac{A_{i+1}D_{i+1}+A_jD_j}{2A_i\Delta x}\frac{\Delta t}{\Delta x} \tag{5.17}$$

These probabilities can now be used to obtain the dispersive displacement average and variance, which are respectively given as:

$$\left\langle x_{disp}\right\rangle_i=-P^{disp}\left(i-1,n+1\,|\,i,n\right)+P^{disp}\left(i+1,n+1\,|\,i,n\right) \tag{5.18}$$

$$\left\langle x_{disp}\right\rangle_i=\left(\frac{A_{i+1}D_{i+1}-A_{i-1}D_{i-1}}{2A_i\Delta x}\right)\frac{\Delta t}{\Delta x} \tag{5.19}$$

$$\sigma_i^2\left(x_{disp}\right)=\left\langle x_{disp}^2\right\rangle_i-\left(\left\langle x_{disp}\right\rangle_i\right)^2 \tag{5.20}$$

$$\sigma_i^2\left(x_{disp}\right)=\left(\frac{A_{i+1}D_{i+1}+2A_iD_i+A_{i-1}D_{i-1}}{2A_i\Delta x}\right)\frac{\Delta t}{\Delta x}-\left[\left(\frac{A_{i+1}D_{i+1}-A_{i-1}D_{i-1}}{2A_i\Delta x}\right)\frac{\Delta t}{\Delta x}\right]^2 \tag{5.21}$$

### 5.2.3 Total Displacement Average and Variance

The total average expression (5.1) can now be written, using (5.6) and (5.19), as:

$$\left\langle x_{adv}+x_{disp}\right\rangle_i=\left(\frac{A_{i+1}D_{i+1}-A_{i-1}D_{i-1}}{2A_i\Delta x}+u\right)\frac{\Delta t}{\Delta x} \tag{5.22}$$

Similarly, the total variance expression (2) becomes:

$$\sigma_i^2\left(x_{adv}+x_{disp}\right)=\left(\frac{\left(A_{i+1}D_{i+1}+2A_iD_i+A_{i-1}D_{i-1}\right)}{2A_i\Delta x}\right)\frac{\Delta t}{\Delta x}-\left[\left(\frac{A_{i+1}D_{i+1}-A_{i-1}D_{i-1}}{2A_i\Delta x}\right)\frac{\Delta t}{\Delta x}\right]^2 \tag{5.23}$$

As it is possible to verify, the method introduces numerical dispersion, since particle displacement variance is different from $2D\Delta t$. To keep the concentration homogeneous independently from the spatial variability of the parameters, it is necessary to explicitly introduce numerical dispersion. It is also possible to notice

that the average approximation is given by a simple central differences discretization of ∂(AD)/ ∂x.

### 5.2.4 Probability Distribution for Particle Displacement

Now it is possible to obtain the probability expressions by replacing in (5.3), (5.4) and (5.5) the particle displacement total average and variance, obtained respectively in expressions (5.22) and (5.23), as follows:

$$P(i-1,n+1\mid i,n)=\frac{1}{2}\left(\begin{array}{l}\left(\frac{\left(A_{i+1}D_{i+1}+2A_iD_i+A_{i-1}D_{i-1}\right)}{2A_i}\right)\frac{\Delta t}{\Delta x^2}+\\ \frac{\left(A_{i+1}D_{i+1}-A_{i-1}D_{i-1}\right)}{A_i}\frac{\Delta t}{\Delta x^2}\left(\frac{u_i\Delta t}{\Delta x}-\frac{1}{2}\right)+\frac{u_i\Delta t}{\Delta x}\left(\frac{u_i\Delta t}{\Delta x}-1\right)\end{array}\right) \tag{5.24}$$

$$P(i,n+1\mid i,n)=1-\left[\begin{array}{l}\left(\frac{\left(A_{i+1}D_{i+1}+2A_iD_i+A_{i-1}D_{i-1}\right)}{2A_i}\right)\frac{\Delta t}{\Delta x^2}+\\ \frac{\left(A_{i+1}D_{i+1}-A_{i-1}D_{i-1}\right)}{A_i}\frac{\Delta t}{\Delta x^2}\frac{u_i\Delta t}{\Delta x}+\left(\frac{u_i\Delta t}{\Delta x}\right)^2\end{array}\right] \tag{5.25}$$

$$P(i+1,n+1\mid i,n)=\frac{1}{2}\left(\begin{array}{l}\left(\frac{\left(A_{i+1}D_{i+1}+2A_iD_i+A_{i-1}D_{i-1}\right)}{2A_i}\right)\frac{\Delta t}{\Delta x^2}+\\ \frac{\left(A_{i+1}D_{i+1}-A_{i-1}D_{i-1}\right)}{A_i}\frac{\Delta t}{\Delta x^2}\left(\frac{u_i\Delta t}{\Delta x}+\frac{1}{2}\right)+\frac{u_i\Delta t}{\Delta x}\left(\frac{u_i\Delta t}{\Delta x}+1\right)\end{array}\right) \tag{5.26}$$

Considering that these probabilities can be applied to any existing particles in the same cell and that mass is given by the sum of all the particles, it is possible to use these probabilities as a deterministic mass transfer prediction between neighboring cells. For example, the mass removed from cell i into cell i+1, over a time interval, is obtained by the product of P(i+1,n+1|i,n) by the cell i particle mass in time n ($M_i^n$). The particle mass that remains in cell i, in a time step, is equal to the product of P(i,n+1|i,n) by $M_i^n$.

### 5.2.5 State equation

To better illustrate DisPar, its state variable will be given by the concentration of particles as in a typical Eulerian discretization of the advection-diffusion equation. The model will therefore be expressed so that cell i particle mass at time n+1 ($M_i^{n+1}$) is obtained as it is shown in Figure 5.2. The variable

$M_i^{n+1}$ corresponds to the sum of the particle mass that remains in cell i (product of P(i,n+1|i,n) by $M_i^n$) with the particle mass removed from the two neighboring cells into cell i. These mass transfers are given by P(i,n+1|i-1,n) $M_{i-1}^n$ and P(i,n+1|i+1,n) $M_{i+1}^n$ respectively:

$$M_i^{n+1} = P(i,n+1 \mid i-1,n) M_{i-1}^n + P(i,n+1 \mid i,n) M_i^n + P(i,n+1 \mid i+1,n) M_{i+1}^n \tag{5.27}$$

where $M_{i-1}^n$ ($M_i^n$)($M_{i+1}^n$) = particle mass in time n, in cell i-1(i)(i+1).

As $\Delta x$ is constant, the cell i particle concentration in time n+1 ($C_i^n$) can be obtained by:

$$\begin{aligned} C_i^{n+1} &= P(i,n+1 \mid i-1,n) \frac{A_{i-1}^n}{A_i^{n+1}} C_{i-1}^n + \frac{A_i^n}{A_i^{n+1}} P(i,n+1 \mid i,n) C_i^n + \\ &+ P(i,n+1 \mid i-1,n) \frac{A_{i+1}^n}{A_i^{n+1}} C_{i+1}^n \end{aligned} \tag{5.28}$$

where $C_{i-1}^n$ ($C_i^n$)($C_{i+1}^n$) = concentration in time n, in cell i-1(i)(i+1). Expression (5.28) corresponds to the DisPar model state equation and the coefficients u, D and A present in the probabilities expressions P(i,n+1|i-1,n), P(i,n+1|i,n) and P(i,n+1|i+1,n).

The DisPar state equation (5.28) is found to be similar to a finite difference explicit scheme, if one considers each cell center as a node in an Eulerian spatial grid. Therefore, it is possible to expect advantages and shortcomings like these classes of models.

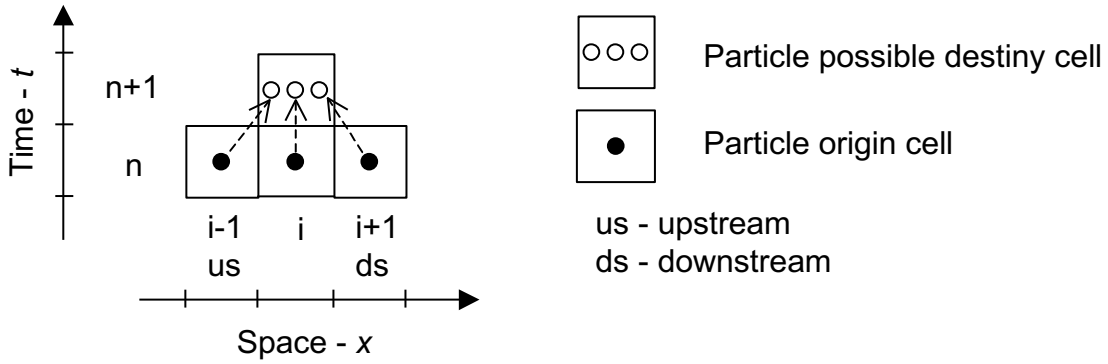



Figure 5.2 – With the Eulerian scheme cell, i is always influenced by the two neighboring cells and its own at the previous time

## 5.3 Convergence analysis

Analyzing the particle displacement total average and total variance convergence one can expect results equal to the traditional particle tracking models running in infinitesimal temporal and spatial conditions. To obtain the total average measured in distance units ($\langle x_{adv} + x_{disp} \rangle$), $\langle x_{adv} + x_{disp} \rangle_i$ must be multiplied by $\Delta x$. To obtain the total variance measured in the square of distance ($\sigma^2(x)$) it is necessary to multiply $\sigma_i^2(x)$ by $\Delta x^2$.

In the spatial convergence situation ($\Delta x \rightarrow 0$) and for small particle jumps ($\Delta t \rightarrow 0$) if one assumes that the AD spatial derivative exists in the entire domain, the central differences in space can be written by definition as:

$$\frac{\left(A_{i+1}D_{i+1} - A_{i-1}D_{i-1}\right)}{2dx} = \frac{\partial(AD)}{\partial x} \tag{5.29}$$

This means that the total average expression can be written as:

$$\langle x_{adv} + x_{disp} \rangle_{\Delta t \rightarrow 0, \Delta x \rightarrow 0} = \left( \frac{1}{A} \frac{\partial(AD)}{\partial x} + u \right) dt = \left( \frac{\partial D}{\partial x} + \frac{D}{A} \frac{\partial A}{\partial x} + u \right) dt \tag{5.30}$$

where $\langle x_{adv} + x_{disp} \rangle$ = total average measured in distance for any point.

In the limit situation it is possible to develop functions $A_{i+1}D_{i+1}$ and $A_{i-1}D_{i-1}$ in Taylor series relative to point (x=i, t=n). Their sum can be written as:

$$A_{i+1}D_{i+1} + A_{i-1}D_{i-1} = 2A_iD_i + \frac{2}{2!}\frac{\partial(AD)}{\partial x}dx^2 + \frac{2}{4!}\frac{\partial(AD)}{\partial x}dx^4 + \ldots \tag{5.31}$$

but since dx is an infinitesimal value, the sum of these two functions is:

$$A_{i+1}D_{i+1} + A_{i-1}D_{i-1} = 2A_iD_i \tag{5.32}$$

In the convergence situation, the variance can also be rewritten as:

$$\sigma^2 \left( x_{adv} + x_{disp} \right)_{,\Delta t \rightarrow 0, \Delta x \rightarrow 0} = (2D_i)dt - \left( \frac{1}{A_i} \frac{\partial(AD)}{\partial x} dt \right)^2 \tag{5.33}$$

but since the second term is one order higher than the first one ($dt >> dt^2$):

$$2Ddt >> \left( \frac{1}{A_i} \frac{\partial(AD)}{\partial x} \right)^2 dt^2 \tag{5.34}$$

the total variance converges to the Fickian one:

$$\sigma^2\left(x_{adv}+x_{disp}\right)_{,\Delta t\to 0,\Delta x\to 0}=2Ddt \tag{5.35}$$

where $\sigma^2\left(x_{adv}+x_{disp}\right)$ = total variance measured in distance for any point.

## 5.4 Non-linear water depth tests

### 5.4.1 Theoretical tests

To appraise the importance of non-linearities three tests are present for a spatial variable water depth, where *u* is equal to 0 and *D* is constant for all spatial points. In these conditions, a uniform initial concentration field should maintain the same values over any simulation time. The three tests represent three different water depth profiles: the first is a function that represents a physical discontinuity (if x <= 53 then y = 2; if x > 53 then y = 6, Figure 5.1), where the derivative significantly changes. The second situation corresponds to a continuum function (y = $\sqrt[3]{}$(x-50)+5, Figure 5.4) with an impossible derivative at a specific point (x = 50). The last situation is a 4$^{th}$ order polynomial (Figure 5.5), derivable at all points. In all the three situations $\Delta x$ = 1, *D* = 0.01, *k* = 1, the total simulation time is equal to 100 and the boundaries do not influence the results at the regions presented in the figures. The water depth spatial derivatives are approximated with a centered difference, since higher orders in the derivative calculation do not improve the results.

As it was already pointed out, the conditions described above imply the theoretical conservation of the initial concentration field. If these concentration values change, it is possible to understand the influence of the water depth spatial variability in the DisPar-k numerical errors.

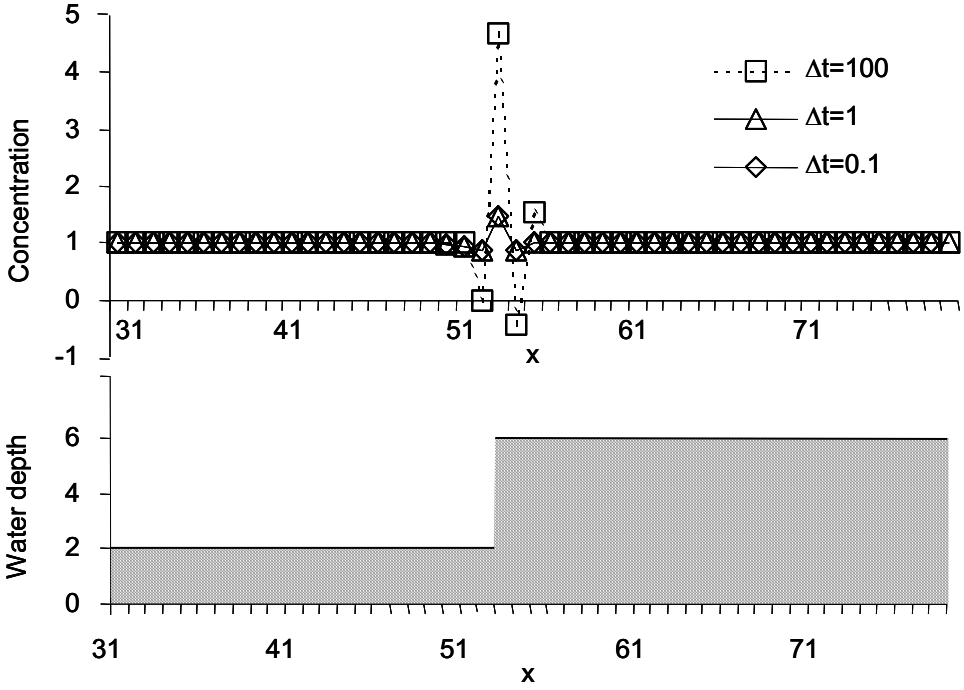



Figure 5.3 - Results for water depth function representing a physical discontinuity

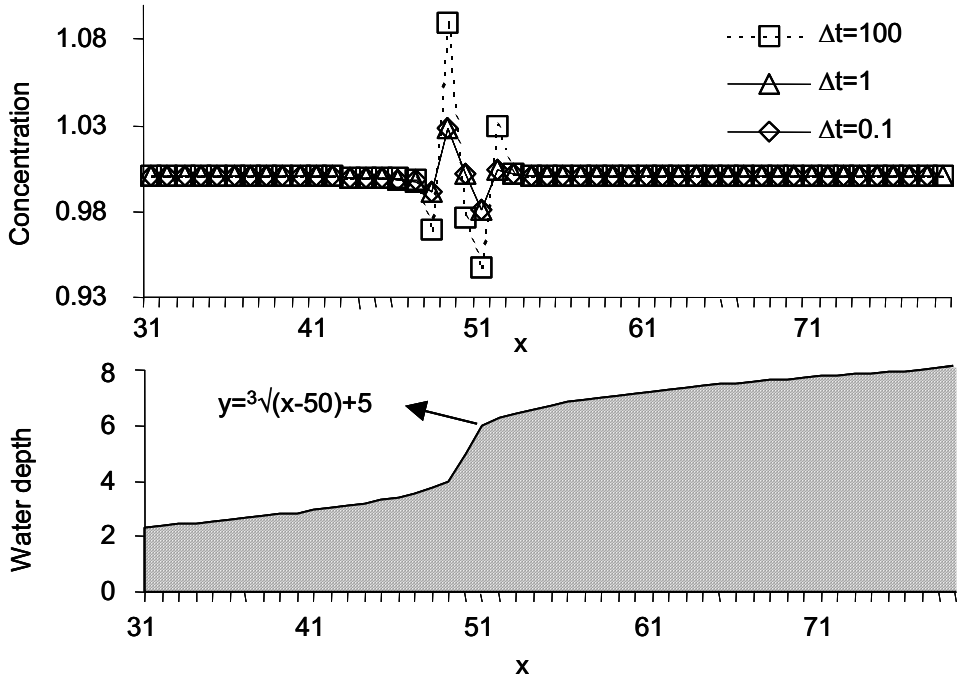



Figure 5.4 – Results for the continuum water height function with a non-derivable point

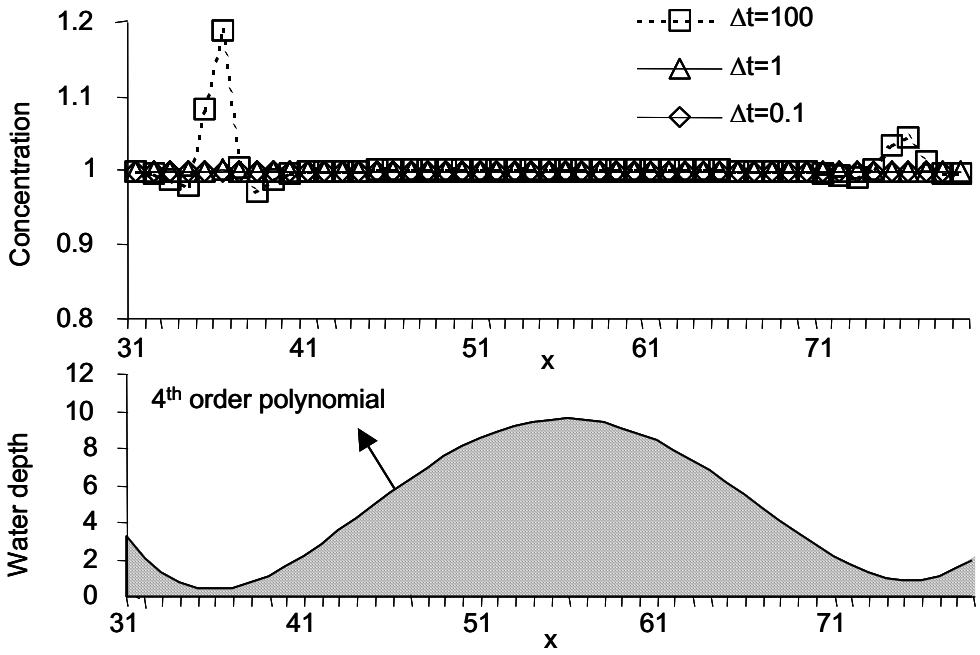



Figure 5.5 – Results for the continuum water height function with all points derivable

The two first situations (Figure 5.3 and Figure 5.4) show that concentration changes are not only dependent on temporal error, but also on the water depth profile, since the time step decrease from 1 to 0,1 does not produce any improvement in the results. As it was previously mentioned, this problem is handled in DisPar due to a specific balancing for the dispersion flow (i.e. the dispersion flow from point *i* to *i*+1 is equal to the dispersion flow from point *i*+1 to *i*), which is achieved by a numerical dispersion introduction in the second order term. In DisPar-k, the average and variance imposed respectively in expressions (5.22) and (5.23) lead to the sort of errors presented in Figure 5.3 and Figure 5.4 when discontinuities are presented in the water depth data. In random walk particle tracking models ([28] and [17]) these problems are also expected since the advective term includes the water depth spatial derivative. From a practical point of view, one can conclude the need of another spatial dimension, in this case the vertical dimension, in order to model the transport process correctly.

In Figure 5.5 it is possible to verify that the water depth profile with the polynomial function, which is always derivable, does not provoke the spatial error presented in the other situations. Therefore, the decrease in the time step is sufficient to obtain accurate results.

### 5.4.2 Application with real data

In this section two tests will be carried out using a hydrodynamic model with real data in a steady state situation. By doing so, it will be possible to test the numerical formulations in a practical case and therefore compare what was theoretically predicted in the previous sections with what will happen in practical cases.

The first test aims to evaluate possible mass imbalances in the transport model. Thus, a uniform concentration field will be applied to the entire domain and it will be evaluated after some time. The goal of the second test is to appraise and reinforce the idea that numerical dispersion must be added in the model formulation, so as to correct mass imbalances caused by discontinuities. For that purpose, a comparison between DisPar and DisPar-k will be made by an instantaneous spill of mass. Both tests will run with a small $\Delta t$ since the goal is to show problems due to the discontinuities in the particle displacement average derivatives.

As it was done in the previous section, each derivative from the average term (equation 5) was calculated by a centered differences scheme. In the first test the parameters used were $\Delta t$ = 0.01 s and time step number=100 and in the second one $\Delta t$ = 1 s and time step number= 1000. The number of destination cells was 11 ($k$=5) for both tests.

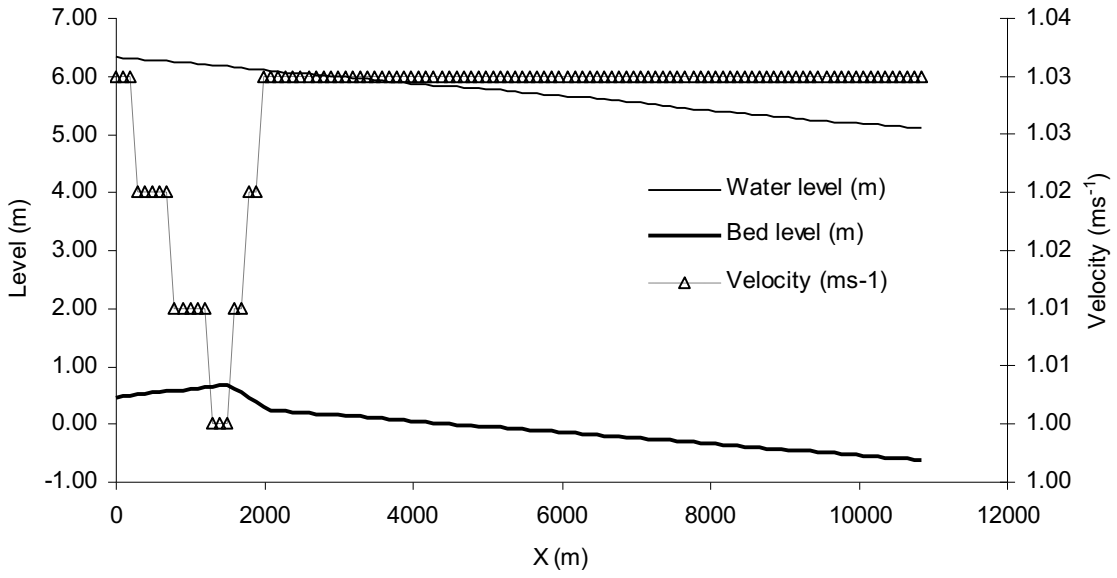



Figure 5.6 - River Waal profile (water level, bed level and velocity)

The case study was applied to a Dutch Rhine branch called the River Waal. The Waal part in study is located between 900 km and 910 km relative to the Rhine datum. The hydrodynamic results were obtained from SOBEK, a computational 1-D river model developed by the Delft Hydraulics and the Institute of Inland Water Management and Waste Water Treatment (RIZA) of the Dutch government. The results were obtained with a dominant discharge of 1600 $m^3s^{-1}$ with a constant $\Delta x$ of 99.58 m and can be seen in Figure 5.6. The data used for the model calibration was recollected in the years of 1995/96. The hydrodynamic simulation was performed with constant section width of 271.00 m except for section 15, where the value was 298.00 m.

The dispersion term was calculated using the well-known Fischer's formula [21]:

$$D = 0.011 \frac{U^2 B^2}{H U^*} \tag{5.36}$$

where $D$ = dispersion coefficient, $U$ = velocity, $B$ = width, $H$ = mean depth and $U^*$ = is the shear velocity ($U^*=(gHS)^{0.5}$); $g$ = acceleration due to gravity; $S$ = channel slope).

As it was explained in the previous section, DisPar-k is very sensitive to non-continuous derivatives in the average term (3.1), which happens with the dispersion derivative. To assess its importance the dispersion variability is assuaged by redefining each point value as the average of its own and the two neighbors. In Figure 5.7 it is possible to observe that the dispersion peak decreases from 2401 $m^2s^{-1}$ to 1722 $m^2s^{-1}$, which is a fall of almost 30%. The smoothed dispersion variability is clearly much softer.

The first test with constant concentration undoubtedly shows the mass transfer imbalances in the region where the parameters have more spatial variability (Figure 5.8). The second test (with smoothed dispersion) also shows this type of unsteadiness, but in a much thinner scale (Figure 5.8), which shows the importance of non-continuities in this type of models.

As it was strengthened in the previous section, this unsteadiness can only be disguised by introducing numerical error in the particle displacement variance (i.e changing the Fickian variance imposed on equation (3.5)). The spill of mass test (Figure 5.9) clearly shows this issue, since DisPar has a higher peak than the

two tests with DisPar-k. These differences in the tests occur in the small imbalances near the peak of the distribution where the dispersion coefficient was not smoothed.

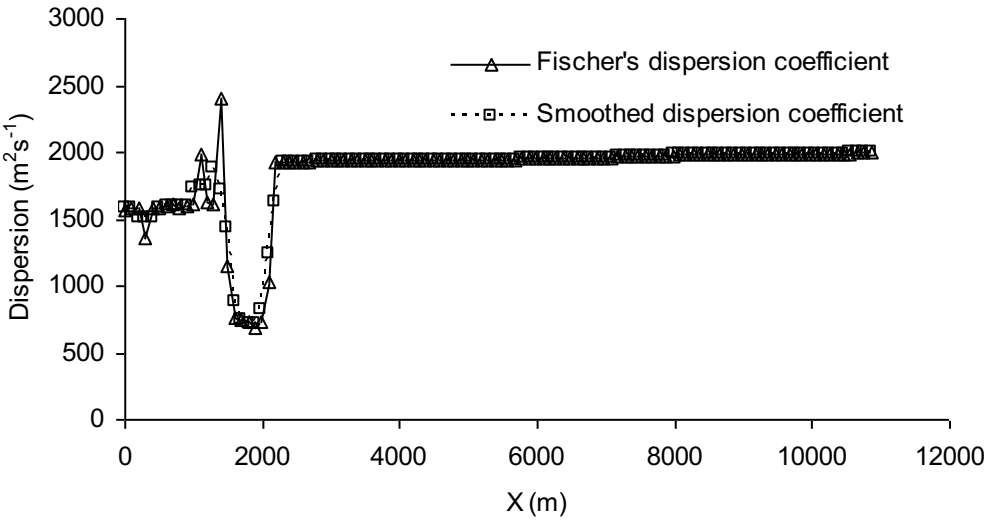



Figure 5.7 – Dispersion coefficient profile for two situations: directly obtained from expression 48; averaged dispersion

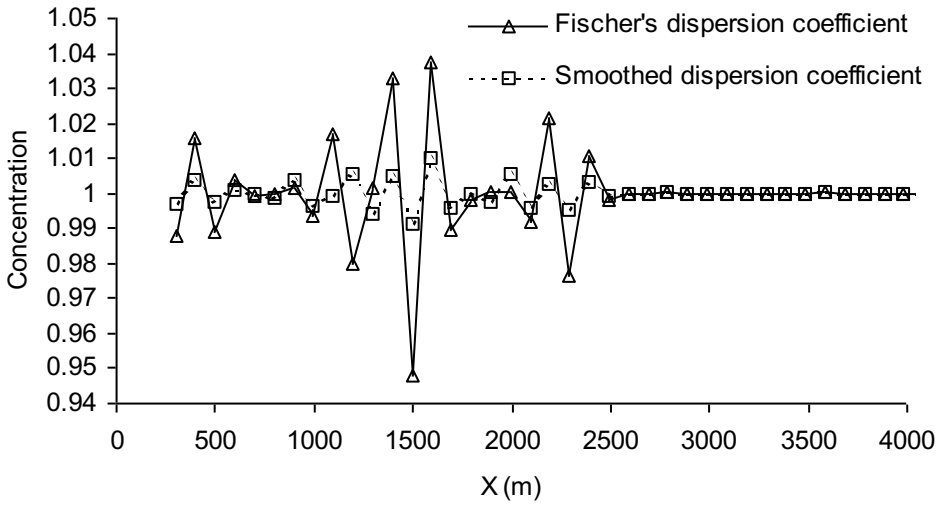



Figure 5.8 – Results obtained with an initial concentration of 1 in the entire domain ($\Delta t$=0.01; time steps=100)

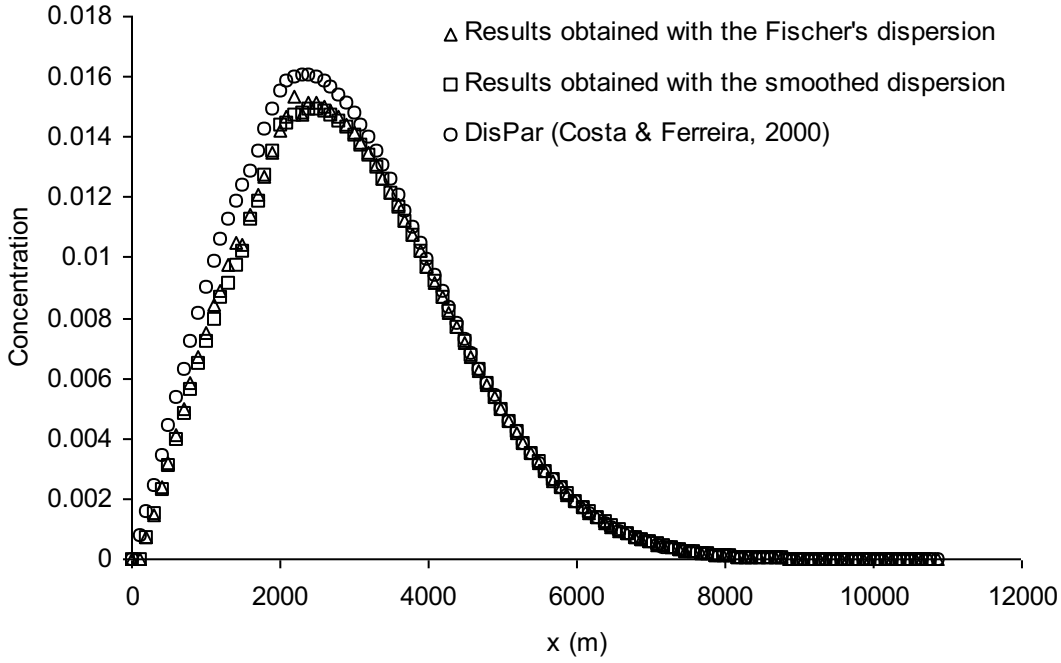



Figure 5.9 – Results obtained for a spill of mass in cell 11 ($\Delta t$=1; time steps=1000)

## *5.5 Conclusion*

In this chapter it was developed an Eulerian model for three particle destination cells to maintain homogeneous concentrations regardless of the variability associated with the parameters if an homogeneous profile is to be applied. The model is similar to a typical finite differences approximation to the transport equation, however with a different advection treatment. The model introduces numerical dispersion for the situations where non-linearities exist in the water depth or in the dispersion coefficient profiles. By comparing this formulation with DisPar-k in these non-linear situations, it is possible to verify that if an initial homogeneous concentration profile is applied to a non-continuous water depth shape without flow motion, the concentration can be only kept the same if the particle displacement variance is different from the Fickian one. This type of problems, demonstrates the strong shortcomings associated with advection-diffusion simulations. To solve it (or mitigate it), one must carefully take into consideration the number of dimensions to be used. Even with a reasonable number of dimensions, bathymetric representations are probably one of the most important aspects of a good model, given that strong non-linearities associated with rough discretizations can create serious problems in terms of numerical dispersion.

Nevertheless, the model developed in this chapter, represents a straightforward formulation to be used by non-conservative models as the ecological ones. It is relatively simple to guarantee positivity, with a relative strong power to treat the drift component. It is known that the model will always have numerical dispersion if the dispersion and/or the water depth profile change in time and/or in space. For this reason, to mitigate this shortcoming, a detailed spatial resolution must be used.

# 6 Techniques for distributed simulation of Advection-Diffusion in Shallow Waters

## *6.1 Introduction*

Pollutant transport models in coastal seas and estuaries are crucial for water quality management and to support environmental impact assessments. The solution of these models has been based on the direct resolution of the transport equation, by means of Eulerian models, Eulerian-Lagrangian and Semi-Lagrangian. DisPar family represents a class of Semi-Lagrangian methods, founding its formulation on the comparison between transport and Fokker-Planck equations ([11]; [15]; [19] and [20]). This last equation represents a diffusive process for Markov particles, only requiring the average and the second order moment to express the particle displacement distribution over space and time. DisPar-k assumes this distribution to be Gaussian and uses it to transfer mass between nodes, cells or volumes, by discretizing the continuous distribution into a user specified number of units. Since in this distribution any moment can be expressed as function of the known average and variance, each probability is evaluated as function of the particle displacement moments. The authors proved that the spatial numerical error is proportional to the number of units used in the particle displacement distribution discretization. Yet, in practical situations, one of the main sources of errors derives from problems associated to the bathymetry representation. Eulerian models usually disguise this type of problems by introducing numerical dispersion and Eulerian-Lagragian and Semi-Lagrangian methods have local mass errors near complex boundaries and in areas of steep bathymetric gradients [45]. The accumulation of this type of errors over time creates stability problems to the models and jeopardizes the accuracy of the results. The only way to truly avoid these shortcomings is to increase spatial resolution, with all the associated computational costs. Although this is a relatively easy task for the engineer, most of today's computers are not yet powerful enough to support the desired spatial resolution. For all this, a computational implementation resorting to High Performance Computing - HPC ([1]; [6]; [7]; [18])

- is a possible way to be a step ahead not only in the simulation speed capabilities, but also on the numerical accuracy of the results.

HPC is becoming increasingly popular among the scientific community as it moves out of the traditional centers for super computing due to the growing use of PC clusters. Either dedicated computers or desktop computers working in a collaborative environment can form this type of super computers. Places like universities have an incredible number of PCs connected through a local area network, fostering the use of collaborative computing as a strategy toward low cost super computing. However, the heterogeneity associated with this type of clusters requires the use of more complex and sophisticated algorithms in order to achieve the desired scalability (for generic purposes about HPC read for example, [1] and [18] or HPC integrated with Geographic Information Systems read [27]).

Domain decomposition is the technique most often used in the parallel simulation of both hydrodynamic and transport in shallow waters ([36]; [38]; [64]; [52]; [70]; [63]; [12]; [13] and [14]). Many applications have also been applied to ground water simulation (like, [68]; [60]; [62]) and climate simulation ([32]; [37]; [51]; [69]; [55]).

Several different methods for domain decomposition can be found in the literature like the simple row wise and column wise partitioning or the more elaborated Orthogonal Recursive Bisection (ORB). However, the computational domain associated to a model applied to shallow waters can drastically change throughout the simulation ([36]; [64]; [52]; [12]). The load variation associated with tides can create serious load imbalances leaving several processors idle for long periods of time. Scatter partitioning is one of the possible ways to correct these load imbalances, given that one machine will be assigned to different spatial regions and therefore the probability to receive all type of cells increases [52]. Dynamic Load Balancing (DLB) is another possibility to rectify load imbalances by exchanging loads in runtime between neighboring sub-domains [12].

In this paper a dynamic load balancing method based on the orthogonal recursive bisection method is developed which is capable of dealing with heterogeneous clusters by assigning partitions to processors in accordance with the expected actual processing power of the processors. Although the method guarantees correct load assignments, the uncertainty associated to each processor power prediction can be reasonably high. To correct this type of

problems and also possible load imbalances caused by tides, an adaptive orthogonal recursive dissection algorithm (AORDA) has been developed. As it was pointed out, scattered partitioning is another strategy to mitigate load imbalances, raising several questions like, which is the best? Do they achieve better scalability in parallel performance if jointly used? We have implemented all these individual strategies as well as their combination into a powerful software tool for the advection-diffusion simulation based on DisPar methods. Microsoft .NET framework and its tools for remote communications were used to implement this distributed application by following an Object Oriented (OO) approach. To allow for an evaluation of those theoretical issues, the DisPar method is applied to the Tagus estuary near Lisbon, Portugal.

The paper is organized as follows. Section 6.2 outlines possible partitioning strategies. Section 6.3 introduces AORDA, which is an adaptive partitioning method for heterogeneous. Section 6.4 describes the OO design used to implement the distributed application (Scalable DisPar) on the Microsoft .NET framework. Finally, Section 6.5 analyses the results obtained on the Tagus estuary.

## *6.2 Partitioning strategies*

### 6.2.1 Static partitioning

Domain decomposition has often been applied as strategy in parallelizing computer simulation applications. It is based on partitioning the domain into equal sub-domains. Each processor is assigned with one sub-domain. This strategy works well for static uniform grid and static (actual) processing power. For dynamically changing loads of the grid points in the computational domain, scattered partitioning techniques are often used ([24]; [54]; [52]; [16]).

In a scattered partitioning scheme, given P processors, the domain is first partitioned into F*P equal sub-grids (F >1 and is an integer), then each processor gets F sub-grids which are scattered over the domain. By scattering these assignments, the load associated with processors is evenly distributed. Therefore a relative balance of the load is achieved despite the changing calculation load (see e.g.[52]). A main drawback of the scattered partitioning method is that both

the number of messages and the volume of communications increase linearly with F for synchronous communications.

It has been shown that maximum load imbalance decreases with the increase of the partitioning frequency [52],

$$Max_{p\in 1,\ldots,P}\left(L_P - \overline{L_P}\right) < \frac{C}{F^{\alpha}},\ F \geq 1 \qquad (6.1)$$

where Lp is the workload associated with processor p;  denotes the average processor workload; F is the partitioning frequency; C is a constant marginally dependent on F and $\alpha$ is a constant associated with the L function variation. $\alpha$ becomes smaller when workload changes rapidly over the grid.

The relation (6.1) gives an upper bound of the load imbalance. It also shows that the workload function L has a significant impact on the effect of scattered partitioning. If the load L does not change much over the grid ($\alpha$ is large), scattered partitioning will be highly effective and, on the other hand, if the computational domain is drastically affected by tides ($\alpha$ is small) F must be very large in order to achieve an acceptable level of load balance. Besides reducing load imbalances, the increase of F enables a more efficient domain assignment because templates containing only land cells can be removed and the remaining templates can be adjusted to the shape of the domain (Figure 6.1).

However, the increase of F implies that the surface between neighboring partitions also increases, which means that the communication cost can become very high or even prohibitive for F>>1. Moreover, the computational domain has a limited number of computational units (i.e., it has 500*500 cells, 1000*1000…), so the value of F is also limited.

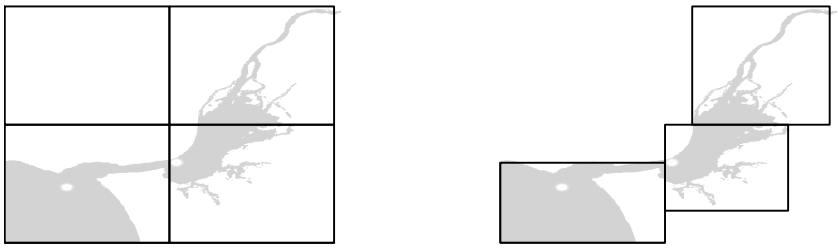


Figure 6.1 – Illustration of scattered partitioning in 4 new templates. The left top template is only composed by land cells and thereby is eliminated and the other three are adjusted to the domain shape

In the literature, the scattered partitioning methods usually assume a homogeneous parallel system where all processors have the same (actual)

processing power. In our situation, the reality is that not only the parallel cluster can be heterogeneous, but also the actual processing power changes in time due to some processors become heavily loaded with many jobs and some others become less loaded. This means that each partition may vary in size (according to the actual processing power) and that from time to time we need to change the partition (according to the changed actual processing power). In combination with the dynamic character of the calculation load of the grid points this makes the task of designing an efficient partitioning method a very complex and challenging one. To the knowledge of the authors, no method has been presented literature so far.

### 6.2.2 Dynamic Load Balancing

Efficient load balancing requires finding the appropriate granularity of partitioning so that each processor is allocated with a workload proportional to its performance. If knowledge about the actual speeds of the machines in a heterogeneous cluster does not exist, this task can represent a pure game of luck.

The most obvious and simplest way to guess the machine's power is through the CPU clock speed, and allocates partitions proportionally to the predicted load and CPU clock speed. However, even if the processors are of the same type (for example, Intel PII or PIII), they can have different main boards, different types of memories and/or different types of hard disks, increasing the number of factors affecting the overall performance. Besides these hardware factors, the dynamic changing nature of computational load induced in this case by tides is another factor influencing load balancing. Scattered partitioning and dynamic load balancing can be used to solve this type of problems by correcting imbalances at runtime. Loads will then be exchanged between machines in order to minimize the differences between processors.

#### 6.2.2.1 Dynamic load balancing strategies

[71] classified the strategies for dynamic load balancing (DLB) as either local or distributed based on the information they use to make load-balancing decisions. They can also be classified as either centralized if the location for load balancing decisions is just taken by one processor or distributed if the load balancer is scattered among the processors.

Global Centralized DLB (GCDLB) – In GCDLB, load balancer lives in the master process. In this scheme slaves send their computational time or any other

information to the load balancer. After calculating the new distribution, load balancer sends instructions to all salves for the execution of load transfers.

Global Distributed DLB (GDDLB) – in this scheme, load balancer is replicated on slaves. The profile information is sent to all processors, avoiding the need for the load balancer to send out instructions, as that information is available to all processors.

Local Centralized DLB (LCDLB) – There is one centralized processor responsible by handling several different groups. Once it receives information from one group, it sends an order for the execution of load exchanges within that group before proceeding to the other groups.

Local Distributed DLB (LDDLB) – Load balancer is replicated on all slaves. The profile information exclusively exchanged between members of a group. This approach is efficient in terms of communications, although no convergence is guaranteed.

#### 6.2.2.2 DLB in Computational Fluid Dynamics

In the literature, successful application of DLB has been reported. It is found to be a very attractive approach to overcome load imbalances and thereby achieve high performance. In the following we discuss three categories of different applications using DLB: particle models, variable grid methods and fix grid methods with variable load.

##### *6.2.2.2.1 Particle models*

Particle tracking methods represent a technique often used in CFD either by the physical nature of the simulated process or by their associated numerical power in Lagrangian methods. Most DLB applications in the literature are based on domain decomposition where the domain is subdivided into small sub-domains each containing an equal amount of workload (e.g. an equal number of particles) ([56]; [35]; [49]).

[56] described a LCDLB method for parallel simulations of molecular dynamics using short-range forces. The partitioning method used is an orthogonal recursive method based on the aggregation of particles in cells. Each template is defined according to those cells so that the number of particles assigned to each processor is approximately equal. The dynamic load balancing approach was

hierarchically designed in such a way that one processor will communicate with a maximum of one processor. By doing this, the author say it is possible to avoid communication bottlenecks typically associated with global load balancers. However, the load transfers are sequentially processed as many times as the number of levels, which means that some particles have to travel through more than one machine during the load balancing process if the application is implemented on a cluster. Therefore, when applied to clusters, the method decreases the probability for network bottlenecks, although creating possible unnecessary communications between neighboring domains.

#### *6.2.2.2.2 Variable grids*

The mesh is locally refined or even rebuilt at every rebalancing step. Many applications and architectures can be found in the literature for the dynamic mesh partitioning and refinement in unstructured grids (for example, [67] and [30]). However, this type of applications represents something far beyond CFD. [2] developed an application which reflects very well the type of problems associated with simulation and re-meshing in CFD. They show a distributed approach for the simulation of blood flow modeled as flows with dynamic interfaces. To solve the problems caused by the interfacial motion, they used an Eulerian-Lagrangian approach, creating a new mesh at every time step. One interesting aspect of this paper is given by the fact that dynamic meshing can represent less than 5% of the overall computation time, which clearly shows the huge computational needs associated with this type of models.

#### *6.2.2.2.3 Fix grid with variable load*

The load associated with each fixed computational unit can vary over time. Hydrodynamic and/or pollutant transport simulations in shallow waters is one of the most obvious situations of this class of models. However, almost nothing exists in the literature using dynamic load balancing as a tool for the simulation of either hydrodynamic or transport in shallow waters. [12] show this in a previous paper, which also deals with a parallel simulation of water quality in an estuary. Domain decomposition was achieved by a column wise partitioning and DLB was carried out by a LDDLB strategy, with the system being rebalanced by exchanging a small number of columns between neighboring sub-domains. However, as expected in a local distributed strategy, this application does not converge properly as the time

step decreases. Besides this issue, any column wise or row wise approach has problems with scalability, it limits the parallel efficiency as the number of processors increases.

## *6.3 AORDA*

In this section a partitioning scheme for heterogeneous clusters is developed, creating the possibility to assign a subsection of the computational domain proportionally to the processing power of each processor. Because of the dynamic characters of a heterogeneous cluster environment, the workload distribution across the processors and consequently the partitioning method must be able to deal with not only the variation in the clock speeds of the processors in the cluster, but also the changes in the actual processing power of each processor due to the CPU-loads during the DisPar-k simulation. A method capable to deal with the dynamics of a heterogeneous cluster will be introduced in this section. It is called **Adaptive Orthogonal Recursive Dissection Algorithm** (AORDA). The heuristic of the static algorithm is explained (ORDA) and illustrated. It is also explained how to adapt DisPar-k to the partitioning algorithm. Finally, the section ends with the Adaptive ORDA (AORDA) explanation and illustration.

### 6.3.1 The ORDA Algorithm: static partitioning

The goal of the partitioning scheme is to subdivide the initial template, such that the time for processor i in processing sub-grid i, $G_i$, is approximately equal for all processors ($t(G_1,1) \approx t(G_2,2) \approx \ldots \approx t(G_p,p)$). The scheme is an orthogonal recursive method with recursive bisection and it has the choice between bi-section and tri-section when the number of processors is $\leq 3$. Thereby, two new subsets are defined if the template has more than 3 machines, so that the total power associated with each one is approximately equal. The sub-region split is proportional to the total power present in each subset, and is alternately performed on the column wise and row wise directions. When finally a subsection has 2 or 3 machines the recursive thread is finished by respectively doing a division in 2 or 3 new regions on the row/column wise direction.

#### 6.3.1.1 Heuristic of the ORDA algorithm

Consider a 2-dimensional rectangular area discretized with equal distance grid lines. Without loss of generality we assume a rectangular grid of [1, Xmax] x [1, Ymax], with Xmax and Ymax are the total number of the grid lines along the horizontal and vertical direction respectively. Such a rectangular grid is called a template. In the simpler case a template is equal to the entire computational domain. Later on we divide the computational domain into a number of templates, and each of such templates is partitioned into a number of sub-grids equal to the total number of processors in the cluster. The partitioning scheme proceeds as follows: first the grid is subdivided into two sub-grids row wise along the Y-direction (or column wise along the X-direction). Next each of the sub-grid is partitioned along the X-direction (or Y-direction) into two smaller sub-grids. This process repeats recursively with the partition along alternate directions until the required number of partition is obtained.

In the following the partitioning algorithm is described. $\#P_i$ denotes the number of processors in the subset i of processors. $G_1$ and $G_2$ denotes the two sub-grid of a (sub)grid G, and $L(G_i)$ is the amount of computational work associated with $G_i$. Furthermore, $S_1$ and $S_2$ denote the total estimated actual processing power (i.e., processing speed) of the processors in subset $P_1$ and $P_2$ respectively. The computation time for processing a sub-grid Gi by a set of processors $P_i$ is $t(G_i,S_i)$. The detailed steps of the partitioning heuristic are sketched as follows.

1º

if the last direction was column wise, then divide the current template in the row wise direction;

2º

if #Pi=2 --- divide the template in 2 regions in the row wise direction so that the area assigned to each processor is proportional to the individual power

$$t(G_1,1) \approx t(G_2,2) \Leftrightarrow \frac{L(G_1)}{S_1} \approx \frac{L(G_2)}{S_2} \tag{6.2}$$

Since partitioning is carried out on the row wise direction, to guarantee that both computational times are approximately equal a value on the y-axis direction, Y*, must be found (Figure 6.2).

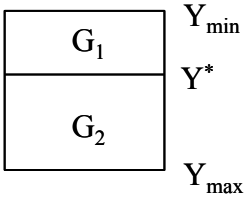



Figure 6.2- partitioning in the row wise direction into two new templates

For example, if the domain is just composed by water cells with an homogeneous load, the total load associated with each region can be defined by the area $G_1$+$G_2$ and therefore Y* can be expressed as:

$$\frac{L(G_1)}{S_1} = \frac{L(G_2)}{S_2} \Leftrightarrow \frac{(X_{\max} - X_{\min})(Y^* - Y_{\min})}{S_1} = \\ = \frac{(X_{\max} - X_{\min})(Y_{\max} - Y^*)}{S_2} \Leftrightarrow Y^* = \frac{Y_{Max}S_1 + Y_{Min}S_2}{S_1 + S_2} \quad (6.3)$$

if #Pi= 3 —- this step is performed with the same logic from the previous one and therefore the template is divided in 3 regions in the row wise direction so that the area assigned to each processor is proportional to the individual processing power.

$$t(G_1,1) \approx t(G_2,2) \approx t(G_3,3) \Leftrightarrow \frac{L(G_1)}{S_1} \approx \frac{L(G_2)}{S_2} \approx \frac{L(G_3)}{S_3} \quad (6.4)$$

To accomplish this, two values on the y-axis direction must be found, $Y^*_1$ and $Y^*_2$ (Figure 6.3)

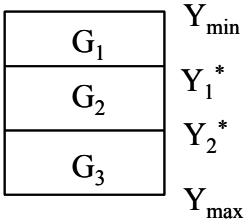



Figure 6.3 - partitioning in the row wise direction into three new templates

if #Pi>3 – find two subsets (Set1 and Set2) from the available processors {Pi} so that the estimated actual processing power assigned to each of the two subsets is approximately equal (Set1 » Set2). The algorithm used to execute this distribution is explained on the Appendix II. After that, the template is divided such that the area assigned to each set is proportional to the total power present in

each group. By doing this, the computational time taken by each set of computers will be approximately equal.

$$t(G_1,1,..,z) \approx t(G_2,z+1,..,p) \Leftrightarrow \frac{L(G_1)}{\sum_{i=0}^{z} S_i} \approx \frac{L(G_2)}{\sum_{i=z+1}^{p} S_i} \quad (6.5)$$

where $t(G_1,\{1,\ldots,z\})$ = the computational time spent by processors 1, …, z on the simulation associated with the spatial region $G_1$.

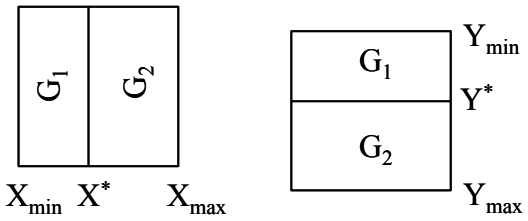



Figure 6.4 – The partition process is repeated recursively with directions alternately column wise or row wise which is set at step 1

#### 6.3.1.2 Load Prediction

A fundamental aspect concerning the algorithm is the load function definition. The two most obvious and simplest approaches are the load definition based on the area of the template and the definition based on the number of water cells present in the template spatial region. From an empirical perspective, it is easy to claim that the number of cells will improve the load prediction given that land cells do not have any special computations besides an if condition. However, as mentioned before, in practical situations these predictions are subject to computational changes caused by the cycle of tides with regions getting dry during an expressive period of the simulated time. Notwithstanding this issue, it was considered that the load function associated with a spatial region L(G) is defined by the number of water cells (L(G) = number of water cells). As it will be seen on the next sections, this type of problems can be mitigated using scatter partitioning and/or dynamic load balancing. Therefore, to define each Y*, $Y_1$*, $Y_2$* or X* the split or splits is/are accomplished so that the assignment of the processors in each template is proportional to the number of water cells and to the power of the processors.

#### 6.3.1.3 Example with 7 machines: static partitioning

To better illustrate the algorithm, an example is outlined in this section. So, supposing that we have an heterogeneous cluster composed by 2 Intel® PII with

450 MHz, 1 Intel® Celeron 500 MHz, 1 Intel® PIII 866 MHz, 1 Intel® PIII 1000 MHz and 1 AMD® 650 MHz. To predict the machinery power we will use the clock speed as a guess for the true power of each processor.

$$[450 \quad 450 \quad 500 \quad 866 \quad 866 \quad 1000 \quad 650] \tag{6.6}$$

The number of processors is >3, which means we have to find two subsets from the available processors such that the total power of each is approximately equal. To do so, we will use the algorithm defined on the Appendix II. According to the algorithm, the first step is to sort the processors in ascendant order rewriting though the previous matrix (6.6) as:

$$[450 \quad 450 \quad 500 \quad 650 \quad 866 \quad 866 \quad 1000] \tag{6.7}$$

Now we will start defining the subsets by assigning the fastest processor (1000 MHz) with Set1 and the second fastest processor (866 MHz) with Set2. Secondly, we will assign the slowest processor with Set1 (450 MHz) and the second slowest processor with Set2 (450 MHz). The process jumps again to the fastest processor ranking by assigning the third processor on the podium (866 MHz) with the Set1 and the one with the forth place (650 MHz) will join Set2. At this point, there is only one processor left (500 MHz), which will be assigned to the subset with less total power. Set1 has a total power of 2316 MHz and Set2 has 1966 MHz, which makes Set2 the nominated to receive the spare processor, totalizing a computational power of 2466 MHz with four computational units.

After defining the first two subsets we will split the initial template, which represents the total computational domain, in the column wise direction such that the number of cells in each spatial region is proportional to both total powers (48%:52%). $Set_1$ is associated with the left side of the template and $Set_2$ is associated with the right side (Figure 6.5 b)). At this point, we do have two new templates, one with three processors and other with four, and therefore it is again necessary to independently apply the partitioning scheme to each template.

The template on the left side has three processors:

$$[450 \quad 866 \quad 1000] \tag{6.8}$$

and so it is dived in 3 new regions in the row wise direction bearing in mind that each processor will receive a number of cells approximately proportional to each processor power (20%:37%:43%)(Figure 6.5 c)).

On the other hand, the template on the right has 4 machines, which means it is necessary to call again the algorithm from Appendix II.

$$[866 \quad 450 \quad 650 \quad 500] \tag{6.9}$$

The matrix is sorted creating this new one

$$[450 \quad 500 \quad 650 \quad 866] \tag{6.10}$$

The definition of the subsets is $Set_1$\{866 MHz,450 MHz \} and $Set_2$\{650 MHz,500 MHz \} and the direction of the domain split is now on the row wise direction. $Set_1$ is assigned to the top of the template and $Set_2$ is assigned to the bottom with the relation 53%:47% (Figure 6.5 c)). Again, the partitioning scheme is applied independently to each new template and given that they have both 2 associated processors they will be both split in two on the row wise direction proportionally to the power of the processors (66%:34% and 57%:43%) and the number of water cells (Figure 6.5 d)). At this stage the partitioning scheme ends given that all threads have arrived to templates with 2 or 3 processors.

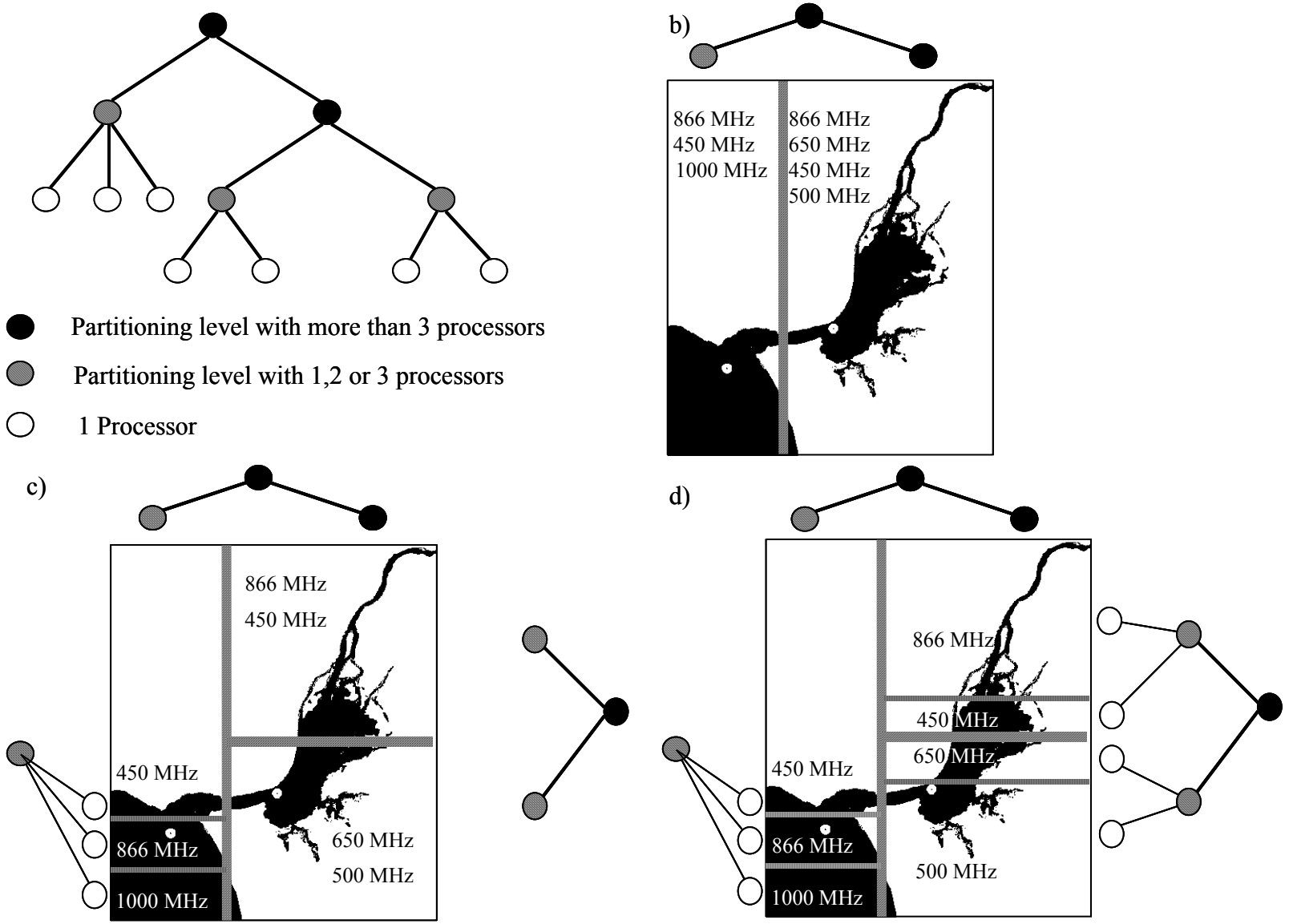



Figure 6.5 – Partitioning stages for a cluster composed by 7 heterogeneous machines

#### 6.3.1.4 DisPar-k and the partitioning scheme

##### *6.3.1.4.1 Sub-domain extra regions for internal calculations*

After partitioning, the computational domain is divided into smaller sub-domains. In the case of PC clusters each sub-domain is assigned to an available PC. To execute the simulation exactly in the same way as the sequential version, some extra information across the boundaries of the sub-domain is necessary for the calculation in each sub-domain. For example, to guarantee a 4th order approximation for water depth derivative over the X-direction, it is necessary to use water depth values from 5 cells, 2 on the left side, 2 on the right side and the cell own value. To do these calculations in cells at the sub-domain periphery, two extra columns are required on the left and right sides of the sub-domain boundary (Figure 6.6 a).

Besides the derivatives, it also needs the destination region water depth for each cell at each time step, given that if a particle destination cell is land, the particle is reflected and will remain in the original cell. Again, the replication of the parameters for temporal series to produce hydrodynamic data is found to be a straightforward strategy. Thereby, this approach implies that some information is replicated and also implies that the total amount of computations will grow with the size of the extra information.

It is important to remember that the DisPar-k particle destination rectangle position can fluctuate with the cycle of tides as a semi-Lagrangian nature consequence. Courant number can lead to a reasonable displacement of the destination rectangle relative to the initial cell as the time step increases. However, DisPar-k uses an explicit (integration) formulation, thus it is not reasonable to use very large time steps. Therefore, if, for example, $k_x=k_y=2$ it is imposed that the sub-domain will always use 2 extra lines for each of its four sides. To deal with a maximum Courant number of 1 an extra line is needed in order to have the information of all destination cells (Figure 6.6 b)).

Finally, the extra region size is defined by the maximum extremes of both areas. This extra region is applied to the parameters with spatial

derivatives (water depth and both dispersion coefficients) and also to both velocities to allow any treatment based on neighboring values (Figure 6.7).

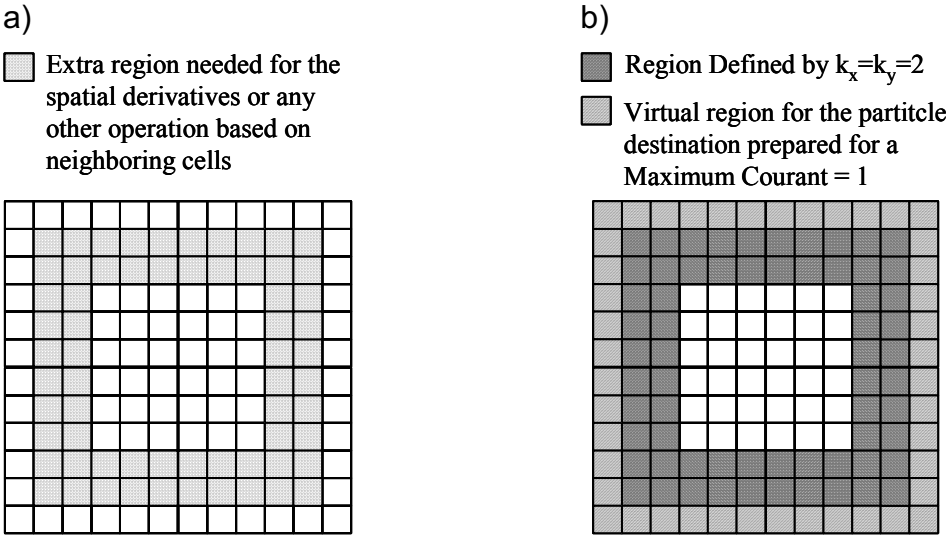



Figure 6.6 – Extra region needed for the discrete particle displacement distribution evaluation

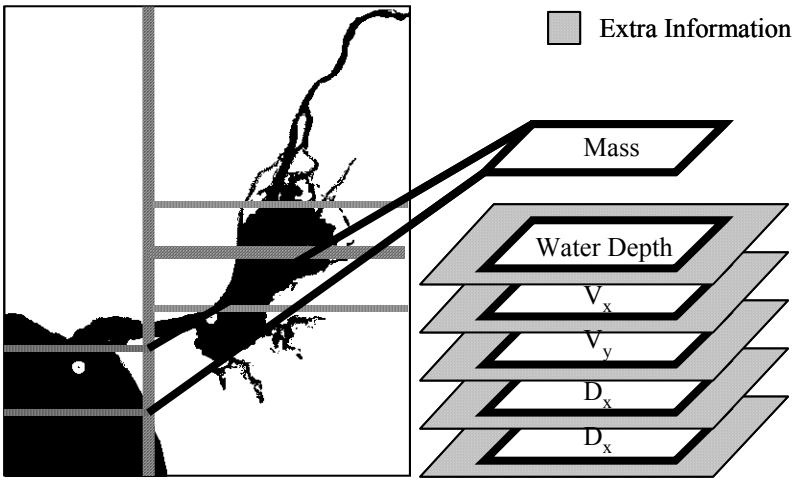



Figure 6.7- Extra spatial information per sub-domain

### *6.3.1.4.2 Neighboring sub-domain definition*

The semi-Lagrangian nature of the model associated with variable velocities throughout the simulation creates another problem related to the neighboring regions. How to determine who is a neighbor of whom, if neighbors of a sub-domain can change during time? Again, we are sure that the region defined by $k_x$ and $k_y$ is always required to create a virtual region for the situation where the Courant number>1 on cells from the sub-domain boundary. For example, if the particle destination rectangle has 9 cells ($k_x$=$k_y$=1) to support Courant values of 2 without loosing mass, the region to

search for neighbors must be composed by a column and a row from respectively $k_x$ and $k_y$ and by a virtual region with two rows and two columns (Figure 6.8). Any sub-domain intercepting this region composed by three columns and three rows is considered as possible neighbor. In the example from Figure 6.8 it is possible to verify that processor $P_A$ will have three neighbors ($P_B$, $P_C$ and $P_D$) even if they do not touch it like it happens with processor $P_D$.

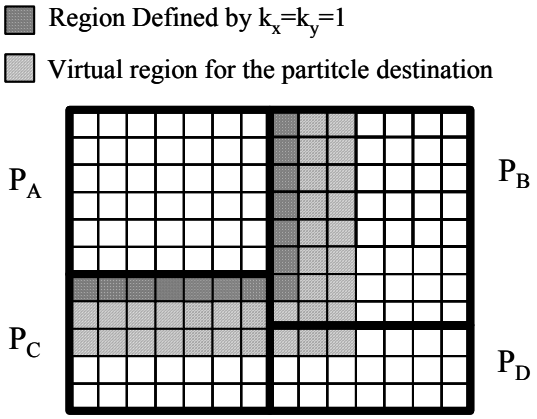



Figure 6.8 – Partitioning example for 4 processors and consequent number of neighboring processors

### 6.3.2 The Adaptive ORDA Algorithm: adaptive partitioning

The dynamic load balancing approach follows exactly the same principles from the domain decomposition strategy, by redesigning partitions in agreement with the different recursive levels. This approach is therefore similar to what was done by [56] for particle tracking models. However, this paper uses a global strategy (GDDLB) given that it was considered to be more appropriate for PC clusters. The probability for a network bottleneck increases with a centralized load balancer, although a smaller volume of communication is reached. Communications will not be replicated, as it would happen if the hierarchical design of [56] was used. It is also possible to find similar strategies in some parallel algorithms for dynamic mesh partitioning in unstructured grids [30].

#### 6.3.2.1 Heuristic of the AORDA algorithm

In the DLB scheme, it is guaranteed that each previously defined template will always have the same neighboring processors. This redefinition is also made based on the new domain information and estimation of each

processor's power (Si'), which is now based on the number of water cells processed between rebalancing trials per time unit:

$$S_i' = \frac{L(G_i, n)}{t_i^n} \tag{6.11}$$

where $L(G_i,n)$ = number of water cells present in $G_i$ on the last simulated time of the $n^{th}$ rebalance; $t_i^n$ is the computational time employed by processor i on the simulation.

It is easy to figure out that this prediction has several shortcomings, such as the computational domain can be continuously changing between two rebalances. If the number of time steps is high and/or the time step is expressive, such that the computational domain crosses several phases, the last domain information does not correctly predict L. L is estimated based on a specific time despite the different tide stages between rebalances.

Besides this issue and as it was mentioned in a previous section, the load associated with each cell is not homogeneous. Near land boundaries, numerical error is explicitly introduced to stabilize the model by removing both spatial derivatives from the average term of particle displacement (section 3.2.3.2). This removal reduces the computational cost associated with cells in these conditions, creating heterogeneity in the computational load. To minimize the probability for an excessive load exchange, the exchange is kept to be no more than 10% of the source sub-domain's total area. In the end, the differences among the meshes will be smoothed out after a number of rebalance steps.

#### 6.3.2.2 Example with 7 machines: adaptive partitioning

In order to illustrate this dynamic load balancing approach, we will go on with the example for the cluster with 7 processors (section 6.3.1.3) by rebalancing it according to some illustrative situation.

In this example it is assumed that all computers are logged-off and that CPUs are most of the time idle, however with the exception of one the two Intel PII 450 MHz, which has a permanent usage of 25% while being used to listening to music in MP3 format. The system is rebalanced only if the verified load imbalance value exceeds 0.1. After the first 30 simulated time

steps, load balancer requests computational times from each of the 7 slaves and detects a load imbalance value of 0.11. This value exceeds the threshold of the acceptable level of load imbalance, and therefore the dynamic load balancing process is invoked. In the next step the load balancer requests from all slaves their water depth matrix, which will be used to update the domain information. The process proceeds by calculating each processor power Si' based on this new information and the computation time (Table 6-1). From Table 6-1 it can be seen that the machine processing MP3 music has an observed processor power smaller than the ratio between the CPU clock speeds. This is because the processor is being shared by another task in addition to the simulation. On the other hand, the machine with the Celeron processor performs a bit worse than was expected by just looking at the CPU clock speeds. This can be either caused by the CPU type or by differences in other hardware components.

The AORDA repartitioning begins at level 1 from the top by dividing the processors into two sets of equal processing power and assigning approximately an equal amount workload to each set. The first partitioning level is respectively composed by processors {866 MHz, 450 MHz, 1000 MHz} and {866 MHz, 650 MHz, 450 MHz, 500 MHz} and the machine processing music was the one assigned to the left template. The total capacity to process water cells on the left templates is 21.6 water cells/millisecond and on the right template is 24.0 water cells/millisecond (Table 6-1), resulting in a ratio between the two templates of 47% : 53%. We assume now that the right template has 1% of total relative power more than the initial situation at the previous rebalancing step. In order to guarantee that the assigned number of cells is proportional to this new ratio of powers, the new ideal position for the domain split is 20 columns to the left relative to the initial partitioning. Taking into consideration that this value represents 4% of the number of columns present in the left template, the threshold of 10% is not exceeded and therefore it is processed without any restriction (Figure 6.9 a)). The next step is to partition the two new sub-templates again, according to the ratio 14%:40%:46% on the left template and 54%:46% on the right one (Figure 6.9 c)). For the left side template the partitioning reaches the end. For

the right template, the partitioning at this level results in two sub-templates each with two machines (Figure 6.9 c)). Finally, when the new partitioning has been determined, exchanging columns or rows of cells between processors can transfer workload. Details of the transfer will be described in the section on implementation.

**Table 6-1 – Observed computational powers**

| Machine | Rel. Power (CPU speed) | Obs. Power S' (Cells/millisecond) | Obs. Rel. Power |
|---|---|---|---|
| Intel ® PII 450 MHz processing MP3 music | 0.45 | 3.0 | 0.30 |
| Intel ® PII 450 MHz | 0.45 | 4.3 | 0.43 |
| Intel ® Celeron 500 MHZ | 0.50 | 4.5 | 0.40 |
| Intel ® PIII 866 MHz | 0.87 | 8.6 | 0.86 |
| Intel ® PIII 866 MHz | 0.87 | 8.6 | 0.86 |
| Intel ® PIII 1000 MHz | 1.00 | 10.0 | 1.00 |
| AMD ® 650 MHz | 0.65 | 6.6 | 0.66 |

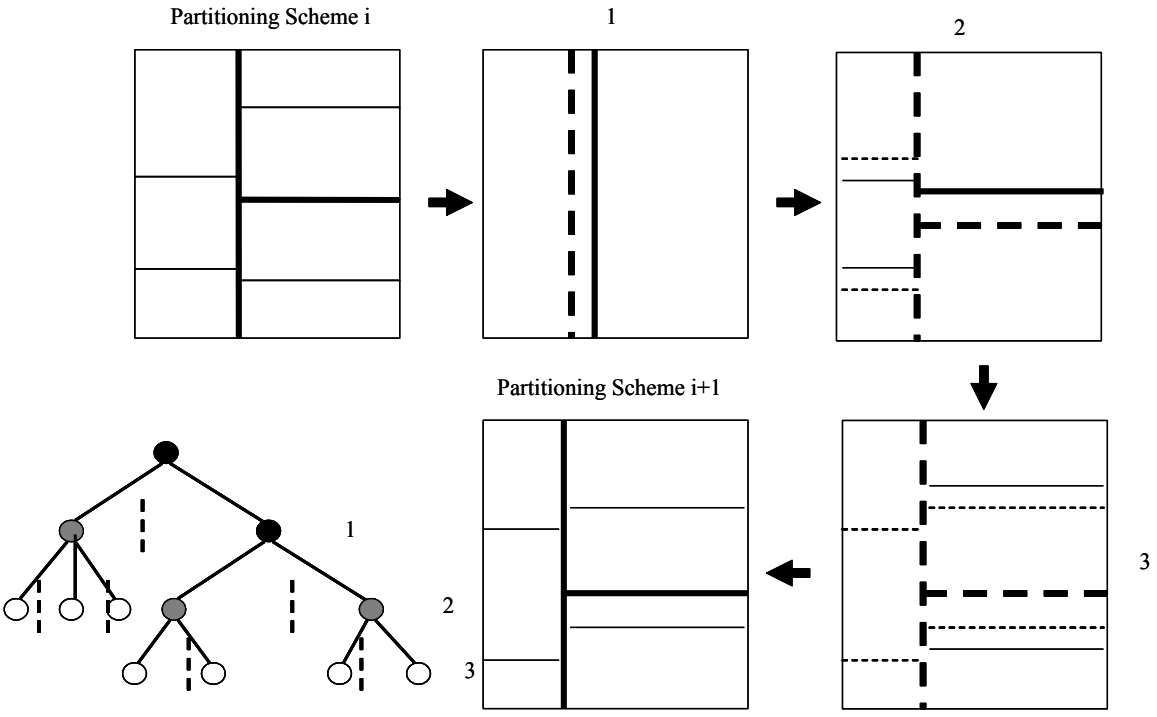



Figure 6.9 – Repartitioning according to the new estimative of the power of the processors and to the new domain shape, which is changing with tides

## *6.4 Implementation*

Scalable DisPar was written following an object-oriented approach. All the main entities composing the distributed scheme were implemented as a class family in which it is possible to highlight the following ones: master,

slave, advection-diffusion simulation, hydrodynamic synthesis and, finally, partitioning. Each family has a primary abstract class holding specific features to the type of tasks it is responsible for and it defines several abstract methods to be implemented by final classes so that any implementation always has to follow the same layout. With the exception of partitioning, all families have a second level composed by two classes to respectively hold different type of meshes and a third one composed by three classes to deal with the three possible dimensions respectively. The application uses the Microsoft .NET framework and was entirely written in C# (CSharp). In the following, first a brief introduction to the framework and used tools will be given, followed by explanation of the classes in each family. A section explaining how all these components make up the distributed application concludes it.

### 6.4.1 Microsoft® .NET framework

Like Java, Microsoft .Net framework also has an intermediate language (IL), which is used to interoperate between components. Instead of binary standards, like the Component Object Model (COM), .Net uses this IL to provide interoperability between components written in different languages. Any .Net compliant language (C++.Net, Visual Basic.Net, C#, J#, Pascal, …) compiles to the IL and therefore any language can use any component independently from the used language. When the code is executed for the first time, the framework compiles the IL code to machine code, optimizing it to the machine where the code is being executed.

Code developed with a language compiler that targets the runtime is called managed code and therefore the framework guaranties that all errors will be properly reported through the use of exceptions. To do so, some performance has to be traded especially if the code intensively reads and writes to arrays. Verification is performed by looking at the extremes of the array at runtime, throwing an exception if boundaries are crossed over. However, memory access violations represent a nightmare in sequential computation and a truly torture in distributed computation. Memory access

violation issues and powerful exception handling tools invited us though to join the managed world.

Assemblies are a fundamental part of programming with the .NET Framework, since they represent a piece of code like a dll, defining and implementing classes. Because of its functionality in communication, remote assemblies played a crucial role in the development of this distributed application. This family of components allows objects to interact with one another, across application domains either locally or remotely. .Net remoting family of assemblies provides a number of services like communication channels for transporting messages to and from remote applications. These messages can use either binary encoding through sockets or XML through the Simple Object Access Protocol (SOAP). If an object is used as a remote object, it must be derived from MarshalByRefObject, like it happens to the main class of each developed family of components. When a client activates a remote object it receives a proxy and all the subsequent operations will be correctly redirected to enable the remoting infrastructure to intercept and forward calls appropriately.

### 6.4.2 Hydrodynamic synthesis

In the current application hydrodynamic data is not simulated, rather it is obtained through temporal series applied to each node, cell or volume. The first class of hydrodynamic synthesis is named HydrodynamicSynthesis, followed by the classes HydroForRegularGrids and HydroForUnstructuredGrids, and classes holding specific features related to the number of dimensions compose the third level in the hierarchy (Figure 6.10). TwoDHydroForRegularGrids implements the necessary methods to compute each value of water depth, Vx or Vy and also methods for exchanging load with neighboring sub-domains. These last methods are crucial for dynamic load balancing, they are responsible for reallocating memory or coping all the data associated with a spatial region. The final class implements the necessary methods for obtaining hydrodynamic parameters by opening files or by using any other means.

ImgTwoDHydroForRegularGrids gets the parameters by opening an Idrisi Img file for each parameter.

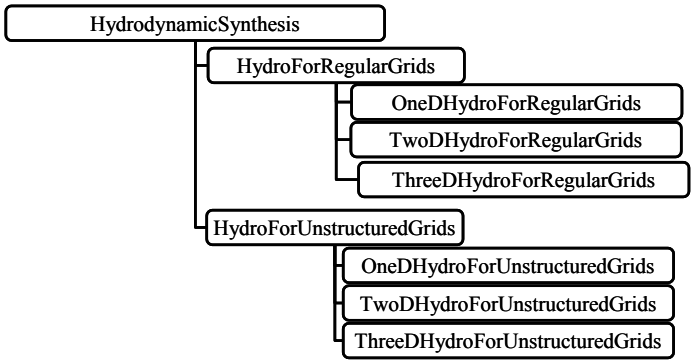



**Figure 6.10 - Family of classes responsible by the hydrodynamic synthesis production**

### 6.4.3 Advection-Diffusion simulation

This family has a DisPar numerical formulation. The first class (SubDomainTransport) defines several abstract methods like the method performing load balancing in the case of a LDDLB strategy or methods defining the sea or river boundary conditions by forcing concentration values in some cells, nodes or volumes. The second level has two classes to deal with structured (RegularSDTransport) and unstructured (UnstructuredSDTransport) meshes respectively. The third has one class per each possible dimensionality (1D, 2D or 3D).

In the following we use the case of a 2D regular grid to further illustrate the implementation details. The class for 2D models in regular grids is TwoDRegSDTransport. This class implements all the abstract methods except only two to be implemented by the final classes corresponding to a numerical method such as DisParV [11], DisPar-k [20] or the 2D version of DisPar-k, [19] (Figure 6.12). Similar to the case of hydrodynamic synthesis, TwoDRegSDTransport has several methods for exchanging loads between neighboring domains.

Only six parameters {Eij[x], Eij[y], Vij[x], Vij[y], bij(x), bij(y)} are needed per cell for the computation in any explicit version of DisPar in two dimensions and therefore the methods to be implemented by a final class has the structure shown in Figure 6.11.

/// <summary>

```
        /// Calculates both particle displacement variances over XX and YY for cell(x0, y0)
        /// </summary>
        /// <param name="x0">column index</param>
        /// <param name="y0">row index</param>
        /// <param name="VarianceX">Calculated variance over XX</param>
        /// <param name="VarianceY">Calculated variance over YY</param>
        protected override void GetBothXYVariances(int x0, int y0,out double VarianceX,out double VarianceY)
        {
            //Implementation
        }

        /// <summary>
        /// Calculates both particle displacement averages and betas respectively over XX and YY for cell(x0, y0)
        /// </summary>
        /// <param name="x0">column index</param>
        /// <param name="y0">row index</param>
        /// <param name=" AverageX">Calculated average over XX relative to betaX</param>
        /// <param name=" AverageY">Calculated average over YY relative to betaY</param>
        /// <param name="BetaX">Calculated betax</param>
        /// <param name="BetaY">Calculated betay</param>
        protected override void GetBothXYAverages(int x0, int y0,out double AverageX,out double AverageY,out int BetaX,out int BetaY)
        {
```

```
        //Implementation
    }
```

Figure 6.11 – Outline of the two methods responsible by the numerical method implementation (C# syntax)

By doing this, the software engineer implementing a numerical method does not have to know anything about the underlying mechanism for distributed computation. In fact, the software developer can have a very modest knowledge about software development and implement his/her DisPar model with all the inherited power.

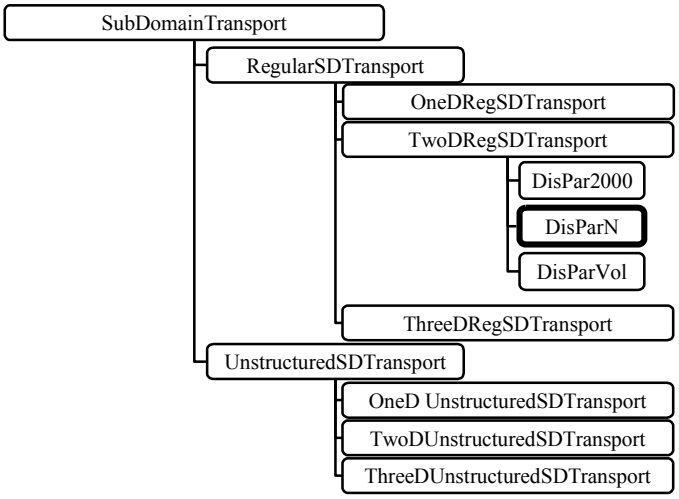



Figure 6.12 – Family of classes responsible by the simulation of advection-diffusion

### 6.4.4 Slave

Like the name suggests, slave family is to be used as the slave from master/slave design. The first class in the hierarchy (Slave) implements several methods such as the ones necessary to pack and send data to neighboring domains, receive data from neighboring domains and its correspondent synchronization. Besides these methods, it also has generic properties as the last computation time or the time spent waiting for neighboring data before going on with the simulation.

A SubDomainTransport final class like DisPar-k implements the transport simulation and slave creates as many instances as the number of associated partitions. The simulation process is managed by sequentially calling methods from each sub-domain simulation class (Figure 6.13). To simulate one time step, the numerical method is first applied to the regions

influencing neighboring sub-domains (first loop from Figure 6.13). The simulation process proceeds further on by packing data and asynchronously sending it to the neighboring machines. As data is being sent, the numerical method is applied to the cells having no influence in the neighboring sub-domains (second loop from Figure 6.13). The next step is a test to verify whether an error has occurred in the process of sending data to the neighboring machines, and it blocks the simulation execution until all neighboring data are sent out. On the receiving side, data sent by neighboring sub-domains is updated into the sub-domains and the process can be blocked again until all neighboring data have arrived. Finally, the simulation process ends by overriding cells with the forcing conditions (last loop from Figure 6.13).

```
/// <summary>
/// Simulation will be applied to all partitions present in this machine. Only one time step will be simulated.
/// </summary>
public void Run()
{
    //Apply the numerical method to the regions influencing neighboring
    //sub-domains
    for(int i = 0; i<countPartitions; i++)
            subDomainInstance[i].DoBoundaryShipments();

    //Pack all data previously calculated per neighboring machine and
    //asynchronously send it
    DoBoundaryShipmentsToAllNeighboringMachines();

    //Apply the numerical method to the remaining cells
```

```
    for(int i = 0; i<countPartitions; i++)
        subDomainInstance[i].Run();

    //Test if there was any problem sending data to the neighboring
//domains
TestAsynchronousResults();

//Update each partition with sent data
ReceiveAll();

//Override cells with forcing conditions
for(int i = 0; i<countPartitions; i++)
    subDomainInstance[i].OverrideCellsWithForceConditions();

}
```

Figure 6.13 – Sequence of operations for a time step associated with each slave

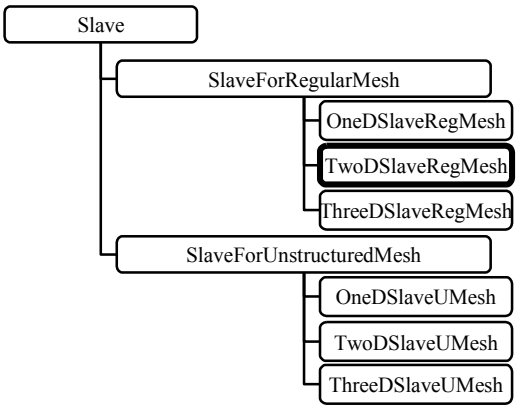



Figure 6.14 – Slave family of classes

### 6.4.5 Partitioning

The partitioning family was designed to support geometric partitioning with either one or two dimensions. In the case of 1D partitioning two classes

were implemented for respectively column wise and row wise partitioning [12]. On the other hand, AORDA is of the type of two-dimensional partitioning (TwoDPartitioning), and a specific class (AORDA) has been implemented for this.

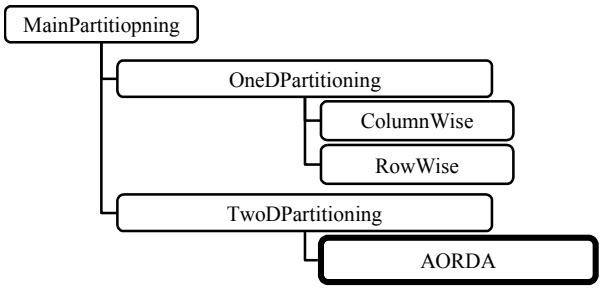



Figure 6.15 –Partitioning family of classes

### 6.4.6 Master

Master family represents the master from the Master/Slave scheme. It has the task to manage the simulation process. The first class defines several abstract methods such as the method to perform load balancing in a GDDLB strategy and has several properties such as the computation cost of the dynamic load balancing. The functionality of the Master can be better explained if embedded in the overall application and therefore it will be further explained in the next section.

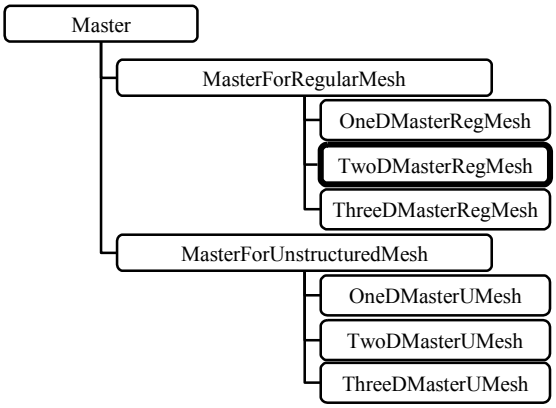



Figure 6.16 – Master family of classes

### 6.4.7 Application

The master/slave middle tier component based approach is managed by an end user application, which can be either a typical desktop application or a web page. Every time a new simulation is started, this application sends the transport dll and all its dependencies to the machine which will host the

master (Figure 6.17). When the dll arrives at the machine it is compiled just in time (JIT) by the framework and according to what was specified by the end user it broadcasts the dll again to the machines running the slaves (Figure 6.17). After JIT compilation, slave component is registered for the .Net remoting mechanism and Master receives proxies to all of them. The next step is partitioning using the specified partitioning class. After the partitioning is determined, the relation of neighbors is known and the list of neighboring partitions is broadcast to all slaves. Finally, the master broadcasts the initial parameters to the slaves and the distributed simulation is started.

Each slave performs the calculation for the sub-domain assigned to it. If for some reason one of the machines takes more time to do the calculation of one time step simulation, all the neighboring ones will be blocked waiting for its data.

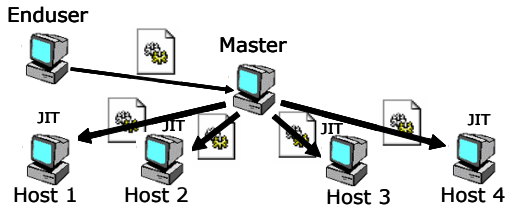



Figure 6.17 – Code marshaling between the end user application and the master and between master and slaves

In order to optimize the load balance, a dynamic load balancer (DLB) is invoked after some simulation time steps. Our software is designed to be flexible to work with different DLB strategies [12], here we will only describe the DLB method in association with a GCDLB strategy. In the GCDLB strategy, the master periodically requests the computation time from slaves according to some user specified criteria. These requests can be either processed with a constant frequency (every x time steps) or processed with variable steps depending on the last load imbalance value. If the load imbalance exceeds a specified threshold, load balancing frequency is increased by some value. On the other hand, if the measured load imbalance is smaller than the threshold, the load balancing frequency is decreased by some value. A minimum and a maximum value for the frequency are set. Load balancing is only invoked when the measured load imbalance exceeds some user specified value. By using this approach, the rebalancing

frequency can be automatically adjusted according to the level of load variation in the system without any knowledge of the simulated domain.

If DLB was invoked and if some difference exists between old and new meshes, load exchanges are carried out according to the differences between the two meshes and are processed in three different stages (Figure 6.18). The first stage is to adapt the sub-domains to the new regions by allocating the necessary memory (Figure 6.18 b)), followed by the copy of the intercepted regions (Figure 6.18c)) and, finally, all sub-domains acquire their final shape by removing spare regions (Figure 6.18 d) and e)). Again, the neighboring sub-domains must be determined and broadcast to slaves.

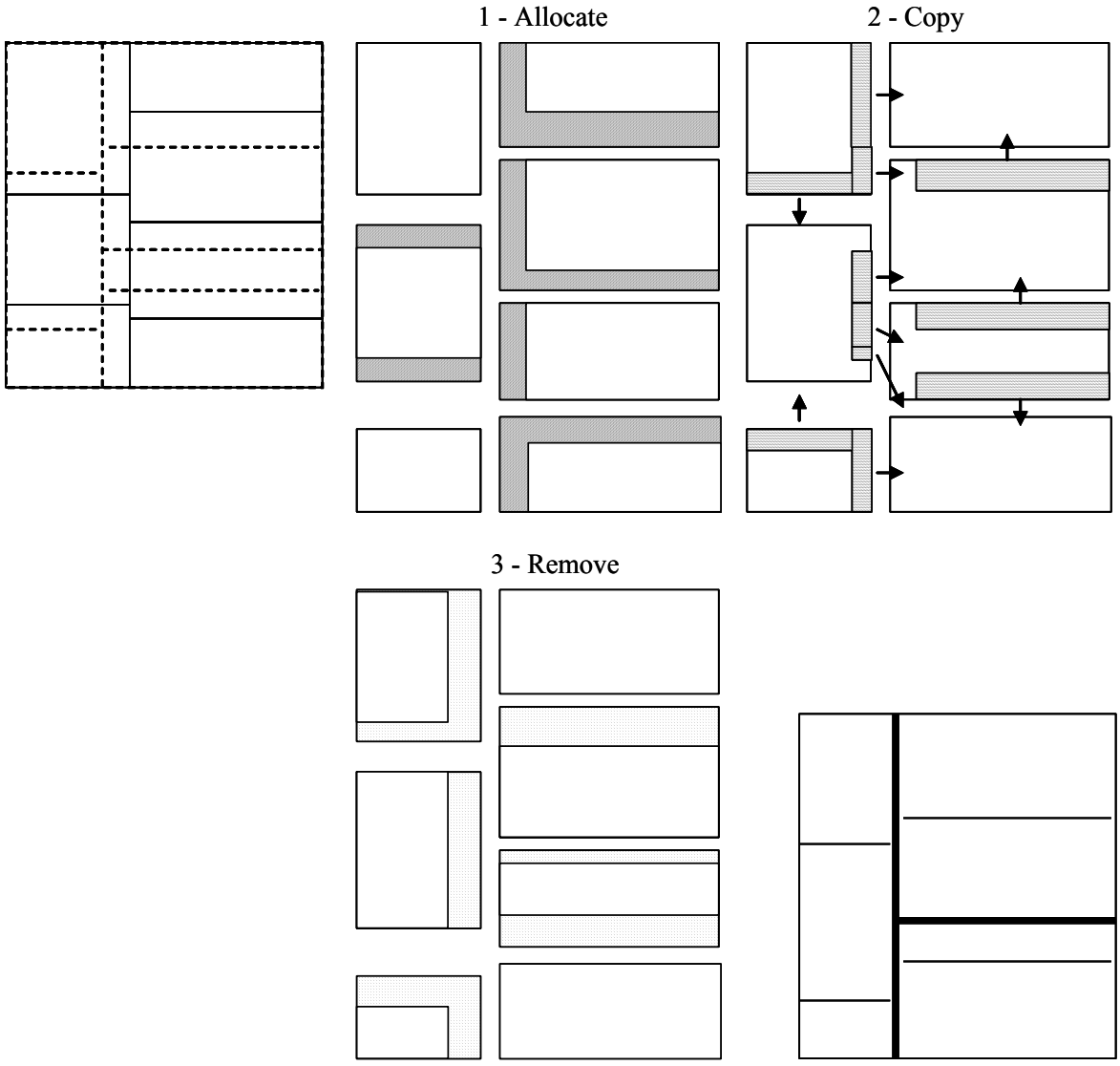



Figure 6.18 – The three operations (allocate, copy and remove) executed to repartitioning according to the new partitioning design

## 6.5 Results for the Tagus Estuary

In this section Scalable DisPar will be applied to the Tagus estuary for two different problem sizes. The performance and scalability of Scalable DisPar will be evaluated through these experiments. The first test problem comprises 500×586 cells (which corresponds to a spatial discretization of Δx=Δy=100m) and the second one is almost 2 times larger with 834×981 cells (which corresponds to a discretization of Δx=Δy=60m). Tagus estuary is quite strongly affected by tides with a significant part of the total area getting completely dry at low tide.

Figure 6.19 shows two typical profiles in low and high tides. The total simulated time was about 33 hours, comprising roughly two and a half tides. All results are obtained using a destination rectangle composed by 7*7 cells. It is already a huge task to compare the different parallel strategies, so we decide not to investigate the effect of the size of the particle destination rectangles.

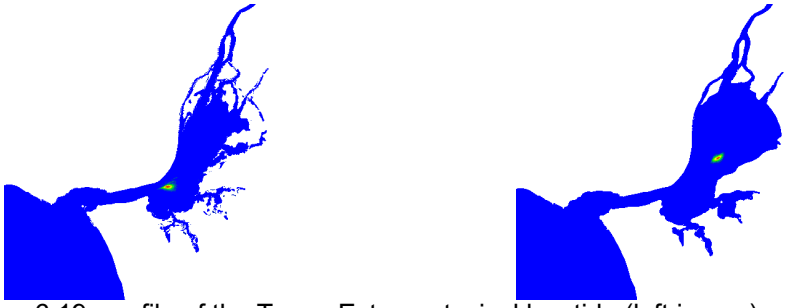


Figure 6.19- profile of the Tagus Estuary: typical low tide (left image) and typical high tide (right image)

The computers present in the room for students from the Department of Environmental Engineering Sciences from Faculty of Sciences and Technology/Universidade Nova de Lisboa were used to build up the PC cluster. There are 16 heterogeneous PCs, ranging from Intel PII 350 MHz up to Intel PIII 450 MHz, Intel Celeron 450 MHz and AMD 800MHz connected through Ethernet by a 3COM switch with 100 Mbs. All the PCs have 382 MB of RAM with the exception of 2 with only 256 MB. The power of each PC was evaluated by measuring the total computation time taken by each machine simulating 36 hours with the computational grid of 500×586 cells, at a time step Δt = 60s it amounts to 2000 time steps. Two templates Figure 6.20 are

used to cover the entire domain. With this domain decomposition, the unnecessary memory usage by cells just composed by land can be minimized and the simulation is also faster than with a single template. Each machine's relative power is expressed by normalizing the measured computation times for the grid with Δx=Δy=100m by the time of the fastest machine (91 minutes). For example, a cluster composed by the faster machine and another PC which toke 182 minutes, has a power of 1 + 0.5 = 1.5.

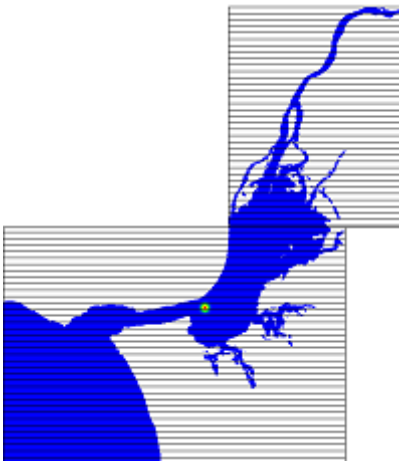


Figure 6.20 – Domain decomposition used for the relative power estimation (1×2)

The required amount of memory by the hydrodynamic harmonic synthesis parameters for the simulations with Dx=Dy=60m is larger than the memory of a single machine, therefore all results are obtained and normalized relative to the simulation time using a cluster of 2 machines. This cluster comprises one machine with relative power 1 and one with 0.56. The domain was decomposed in two templates, which means that each machine had two scattered sub-domains and a total power of 1.56. The measured computation time was 189 minutes.

The DLB component is invoked with a variable time interval depending on the measured load imbalance. The minimum number of time steps per rebalancing trial is 20, and the maximum is 60. The interval is decreased by 10 time steps when a load imbalance > 0.1 is measured and increased by 10 time steps otherwise.

For both grid sizes, scattered partitioning was obtained by decomposing the computational domain into a fixed grid with 1×1 - template

with 1 column and 1 row, 1×2 – template with 1 column and 2 rows, etc. This strategy implies that sometimes the number of rows (or columns) associated with one template is smaller than the number of templates in the row wise direction (or column wise) required by the AORDA scheme, which means that given a fixed grid some decompositions cannot be used.

All the results presented in this paper were obtained during the night to minimize the probability to cross active users with simulations. The experiences made with active users, namely during classes, showed that it is very difficult to win performance by increasing the cluster size with or without DLB. In fact, increasing the size of the cluster increases the probability that at least one machine blocks the simulation. It is important to note that the simulations were always run with a lower priority level to minimize any disturbance to the other users, even for the simulations run during the night. It should be pointed out that Scalable DisPar can only work with desktop clusters with other active users if it is 100% fault tolerant. The observed number of reboots made by users is very high and since the Scalable DisPar application was not prepared for this type of situations, the simulation is then completely disturbed.

In order to compare the performance of different load balancing strategies, parallel simulations are carried out with the following load balancing schemes: 1. Static partitioning with one template for the entire domain; 2. Static, two levels of scattered partitioning with two or three templates; 3. Dynamic load balancing with DLB in AORDA; 4. Dynamic load balancing, a combination of DLB and scattered partitioning with two or three templates.

### 6.5.1 Load Imbalance evaluation

For the evaluation of load imbalance the following dimensionless definition is used (see e.g. [52]), it is defined as the maximum difference between the computation time of any processor p and the average computation time:

$$\text{Load imbalance} = Max_{p\in 1,\cdots,P} \frac{T_p - \overline{T_p}}{\overline{T}} \tag{6.12}$$

where Tp = computational time associated with processor p; $\overline{T_p}$ = average computational time.

### 6.5.2 Computational time

The total computational time associated with a simulation can be decomposed as:

Tot. Comp. Time = Activation + Simulation + Average Waiting Time +

+ DLB + Request results (6.13)

where Activation = time spent activating the simulation; Simulation = time spent on the simulation itself; Average Waiting Time = average time that slaves spend idle waiting to receive neighboring data; DLB = Time spent by the repartitioning design; Request results = time spent actualizing master with all the information from slaves.

In this paper the average time that the slaves spend waiting for neighboring data is dimensionless expressed as:

$$Idle = \frac{Average \quad Waiting \quad Time}{Total - (Activation + DLB)} \times 100\% \tag{6.14}$$

where Idle = the average time that the cluster stays idle during simulation process excluding the time taken to rebalance the system if DLB is to be applied.

The used clusters are always small and therefore the observed costs associated with requesting results from slaves by master are very small compared with the total computation time and therefore they will be not be considered.

### 6.5.3 Static Load Balancing

As it is possible to observe in Figure 6.21, the total computational time associated with each simulation is almost twice larger for the grid 834×981 than for the 500×589 cells. However, the communication cost relative to the total computation time is much higher for the smaller grid. In both situations for all partitioning strategies, the verified efficiency was far distant from ideal values especially for the 500×589 grid (Figure 6.22 a)). Figure 6.22 shows

that the efficiency tends to decrease independently from the number of templates used. Clusters composed by more than 12 machines tend to keep the levels of efficiency stable for both domain sizes. By looking to Figure 6.23, it can be observed that 8 was the maximum number of neighboring machines for the three tested situations. It can also be verified that, for the same cluster size, the number of neighboring machines tends to increase by one neighbor when the computational domain is scattered in more than 1 template.

The number of neighboring machines is the dominant factor affecting the efficiency. There exists a very clear relation between the average idleness that the machines spend on waiting to receive neighboring data and the number of neighbors (Figure 6.24). This is true for both grid sizes, where the main difference is the relative percentage. For the smaller grid, the average idleness is much higher relative to the simulation time. The importance of the number of neighboring machines is emphasized by the fact of relative independence between efficiency and the level of load imbalance for all simulation experiments (Figure 6.25) and by the very clear relation between efficiency and communication cost (Figure 6.26). However, for the larger grid, there exists a sharp tendency for an efficiency decrease as the load imbalance increases independently from the number of templates (Figure 6.25 b)).

a) b)

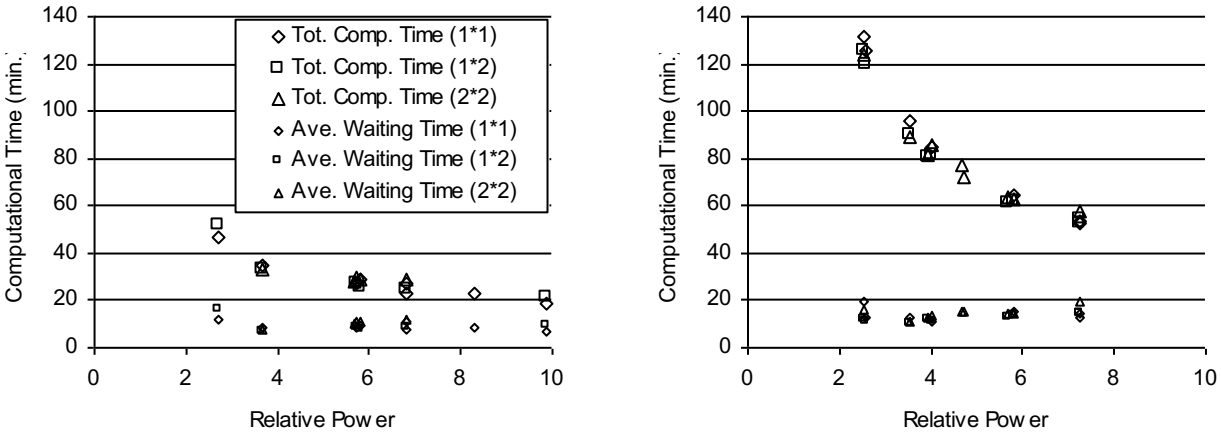
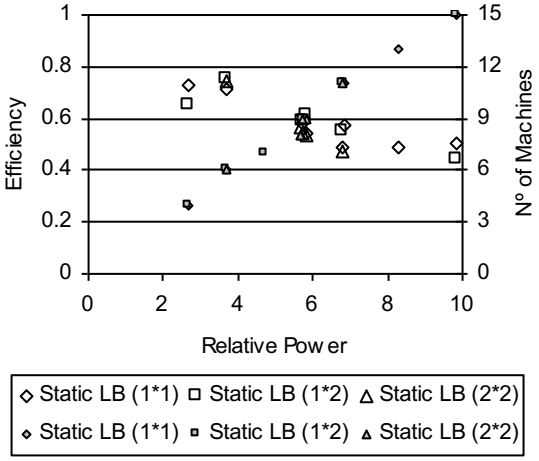



Figure 6.21 – Computational times versus the cluster relative power for a) 500×589 grid and for b) 834×981 grid

a)




b)

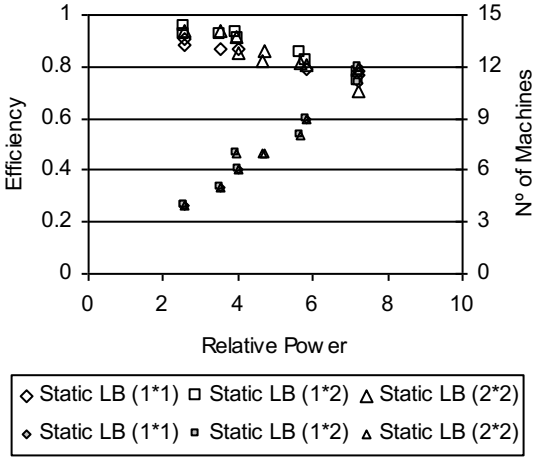
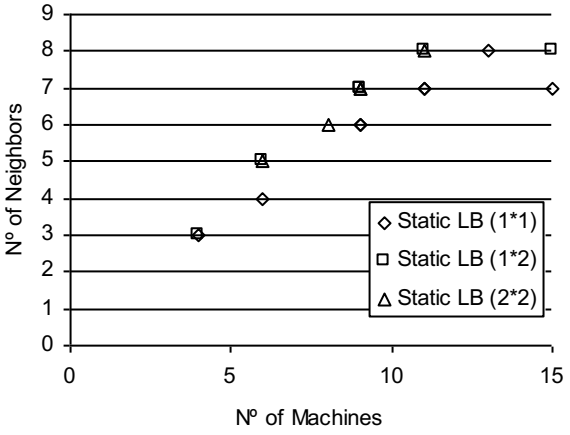



Figure 6.22- Application efficiency versus relative computational power for a) 500×589 grid and for b) 834×981 grid

a)




b)

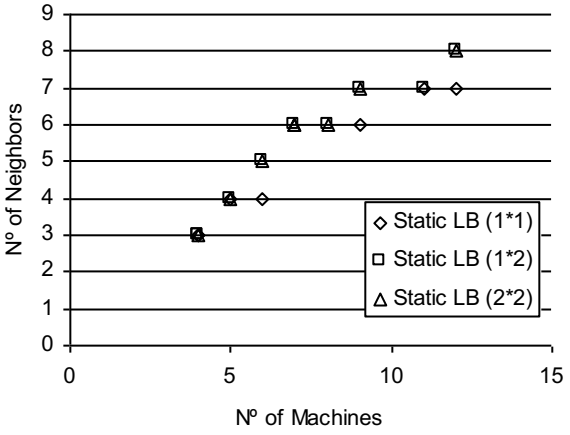



Figure 6.23 - Number of neighbors per number of machines composing the cluster for a) 500×589 grid and for b) 834×981 grid

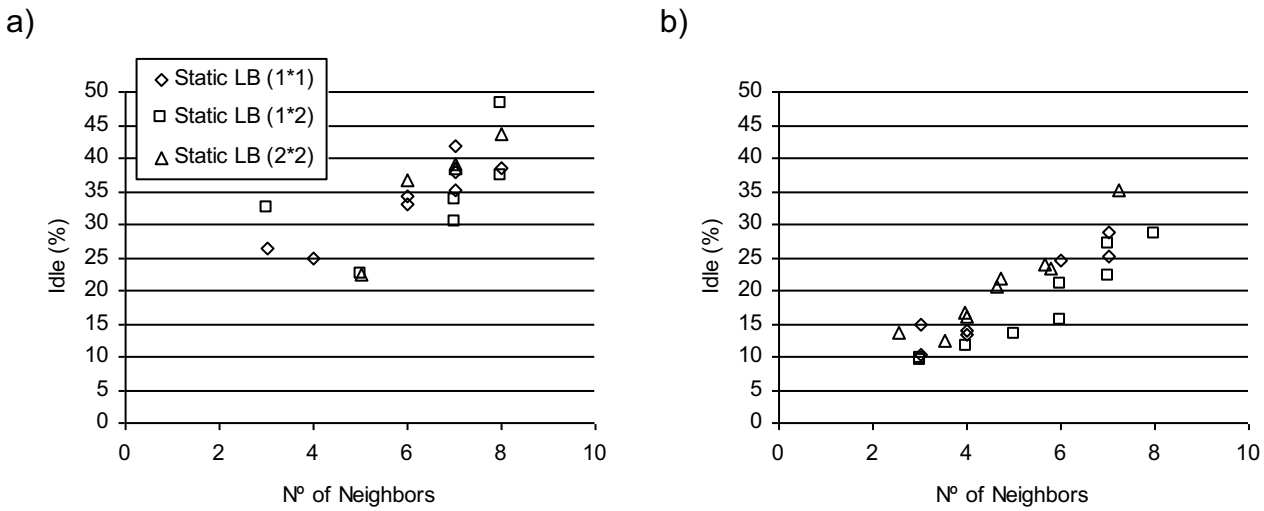



Figure 6.24 - Average idle time per the maximum number of neighboring machines for a) 500×589 grid and for b) 834×981 grid

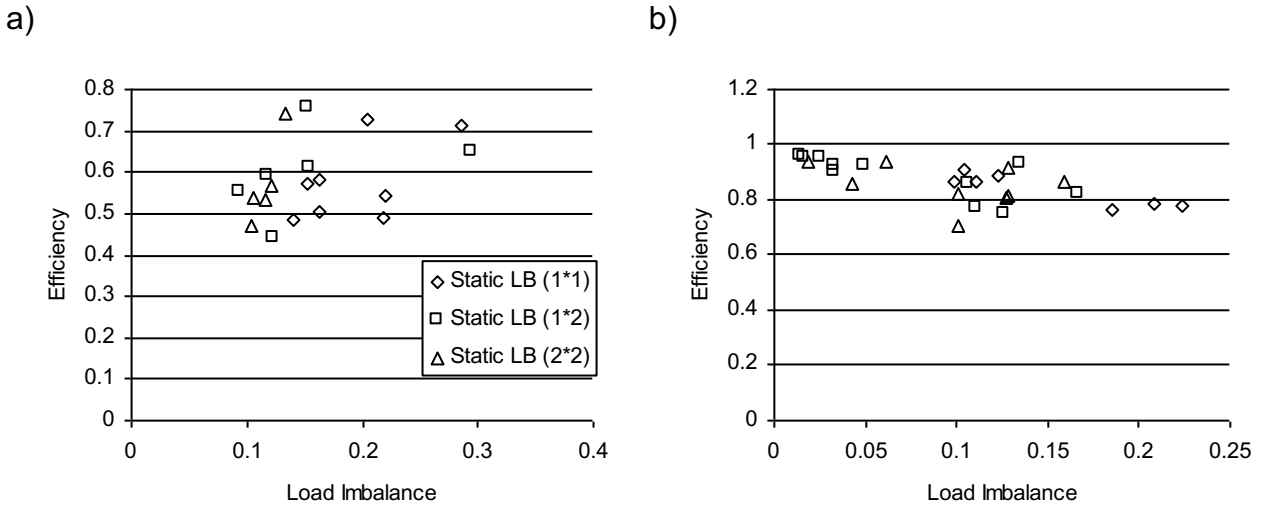



Figure 6.25 – Efficiency versus load imbalance for a) 500×589 grid and for b) 834×981 grid

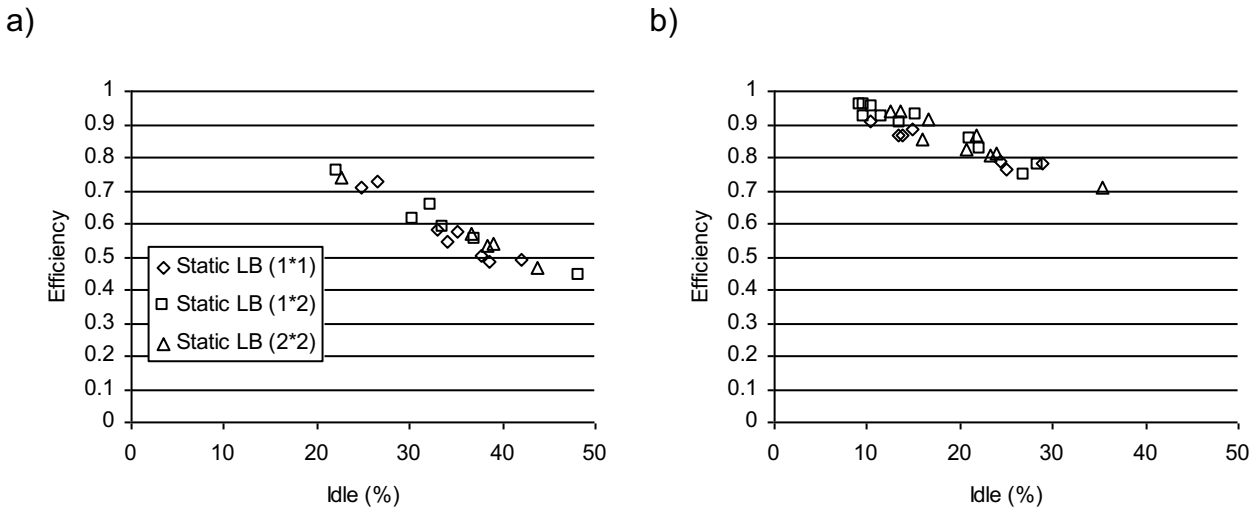



Figure 6.26 – Efficiency versus average idle time relative to the total computational time for a) 500×589 grid and for b) 834×981 grid

### 6.5.4 Load Imbalance and Frequency

The influence of the frequency in scattered partitioning on the load balance is evaluated by dividing the computational domain in 1×1, 1×2, 2×2, 2×3 and 3×3 subdomains. These experiments will be run for two different grids and for two cluster sizes of 4 and 7 machines. Figure 6.27 shows that scattered partitioning without any kind of optimization can produce very small templates, like the one on the top of the downstream region of the Tagus estuary.

As can be observed in Figure 6.28, the load imbalance decreases as the frequency increases confirming the conclusion by [52], however, the load imbalance starts growing again as a consequence of the errors associated with the AORDA applied to small templates. It can be verified that the number of machines has influence on the effectiveness of scattered partitioning. More machines imply more error associated with small templates and therefore load imbalance is higher for the cluster with 7 machines for both grid sizes. From the above discussions, we conclude that scattered partitioning by itself does not solve the problem of load imbalance, given that some templates can be very small if no optimization is used.

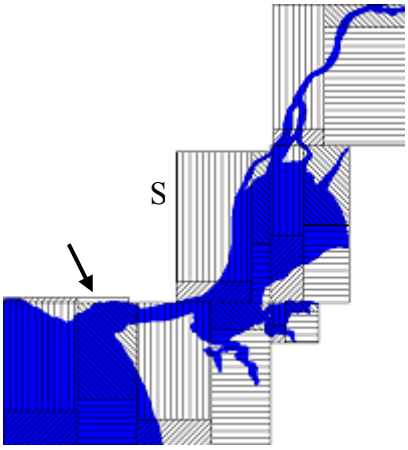



Figure 6.27 – Scatter partitioning for 3×3 templates and for 4 machines. Some of them were solely composed by land cells and therefore were removed and the others were adjusted to Tagus Estuary shape.

a) b)

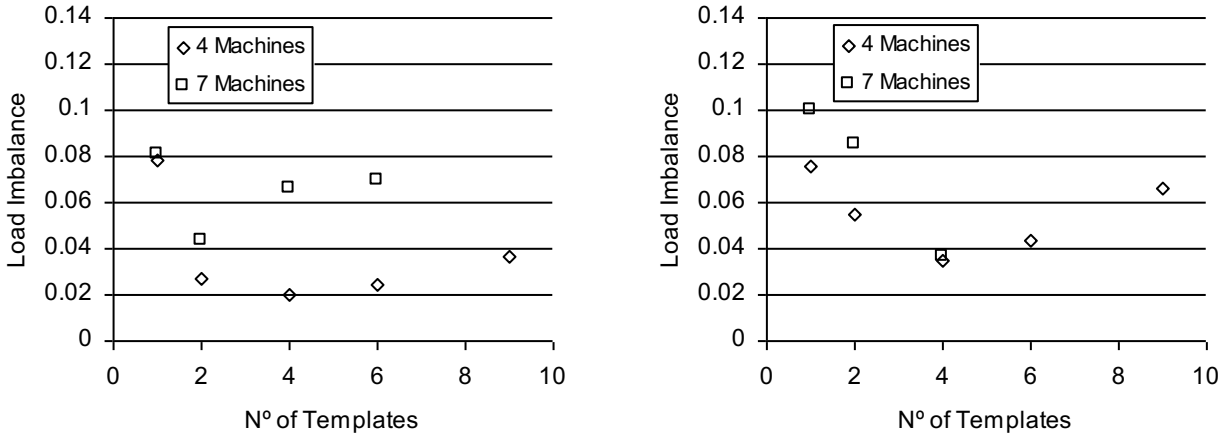



Figure 6.28 – Load imbalance versus number of templates for a) 500×589 grid and for b) 834×981 grid

### 6.5.5 Scatter partitioning mixed with DLB

Like the static load balancing situation, AORDA with DLB activated also shows a very expressive communication cost CommCost, which is much more expressive in the smaller computational grid (Figure 6.29). Again, there was no expressive difference between using scatter or simple partitioning strategies. The efficiency also showed a similar behavior to the static load balancing tests (Figure 6.30), as well as the maximum number of neighboring machines per number of machines composing the cluster (Figure 6.31). Again, it was possible to verify a clear relation between number of neighbors and average time that a machine spends waiting for the neighboring data (Figure 6.32). Usually the DLB cost is higher for the 1×1 partitioning strategy

since the system must be rebalanced more often than with the other two (1×2 and 2×2). However, the 1×2 and 2×2 grids for the same cluster size tended to have one neighbor more than 1×1, which increases the associated communication cost (Figure 6.31).

Like the static load balancing situation, AORDA with DLB activated also shows a very significant communication cost CommCost, which is much higher for the smaller computational grid (Figure 6.29). Again, there was no significant difference between using scattered or simple partitioning strategies. Also the efficiency showed a similar behavior for the static load balancing tests (Figure 6.30), and there is no significant difference in the maximum number of neighboring machines per number of machines composing the cluster (Figure 6.31). Again, the relation between number of neighbors and average time that a machine spends waiting for the neighboring data can be clearly observed (Figure 6.32). Usually the DLB cost is higher for the 1×1 partitioning strategy since the system must be rebalanced more times than with the other two partitioning (1×2 and 2×2). However, the 1×2 and 2×2 partitioning for the same cluster size tended to have one neighbor more than 1×1, which increases the associated communication cost (Figure 6.31).

a) b)

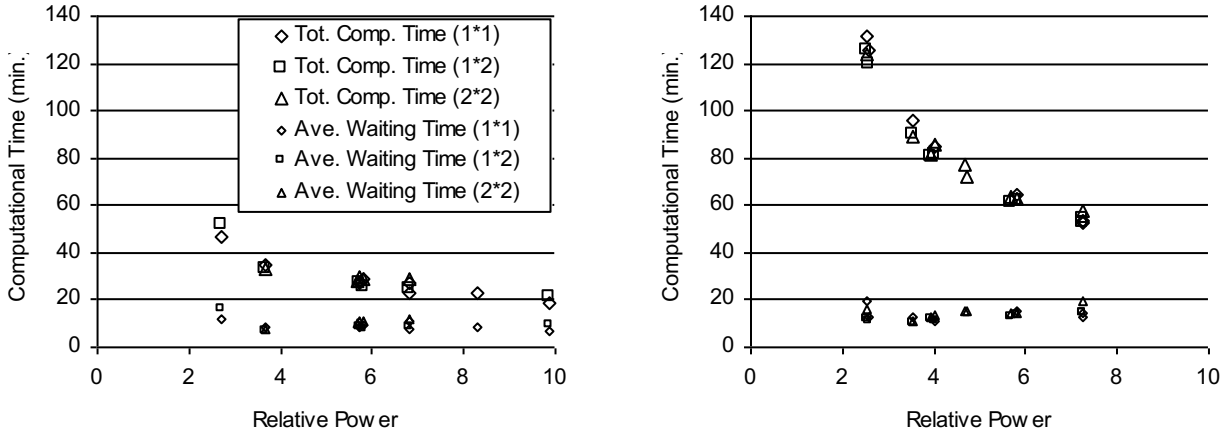
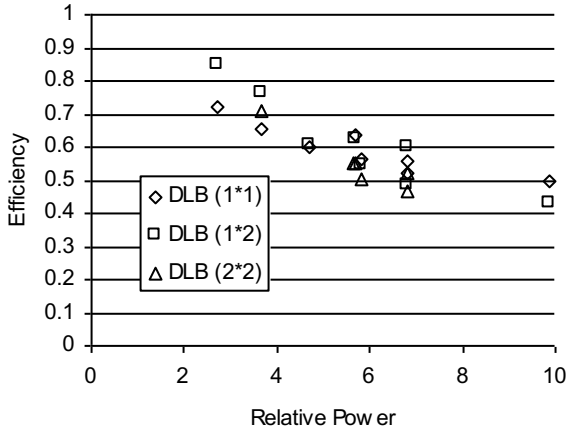



Figure 6.29 - Computational times versus the cluster relative power for a) 500×589 grid and for b) 834×981 grid

a)




b)

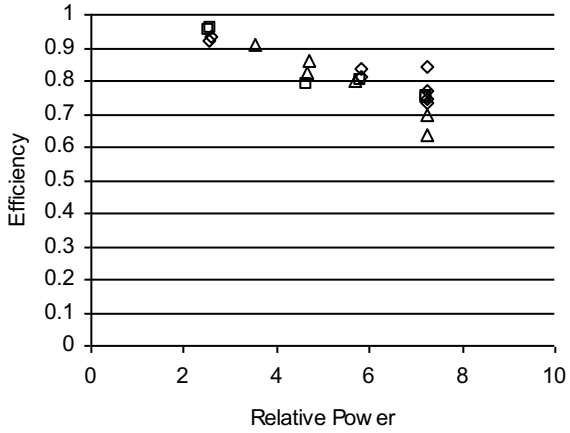
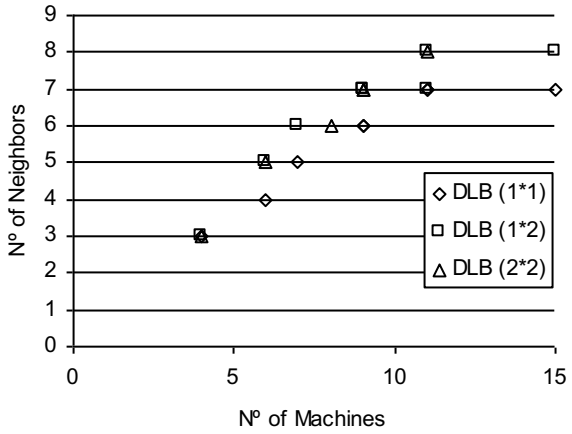



Figure 6.30 - Application efficiency versus relative computational power for a) 500×589 grid and for b) 834×981 grid

a)




b)

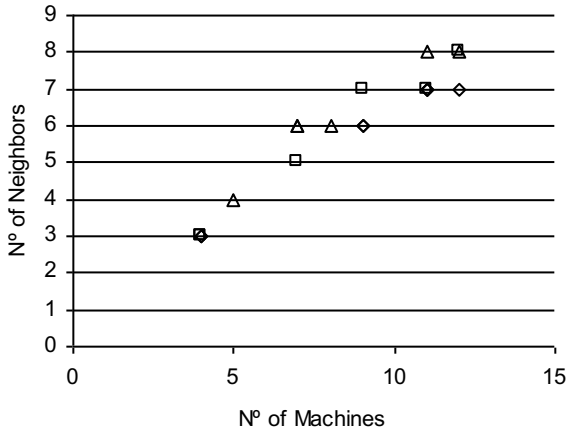



Figure 6.31 - Number of neighbors per number of machines composing the cluster for a) 500×589 grid and for b) 834×981 grid

a) b)

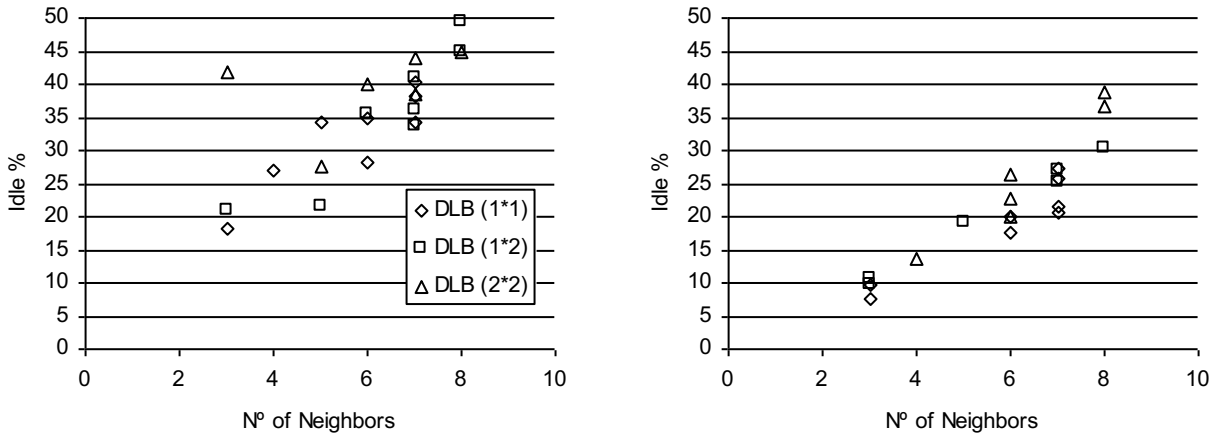



Figure 6.32 - Average idle time per the maximum number of neighboring machines for a) 500×589 grid and for b) 834×981 grid

a) b)

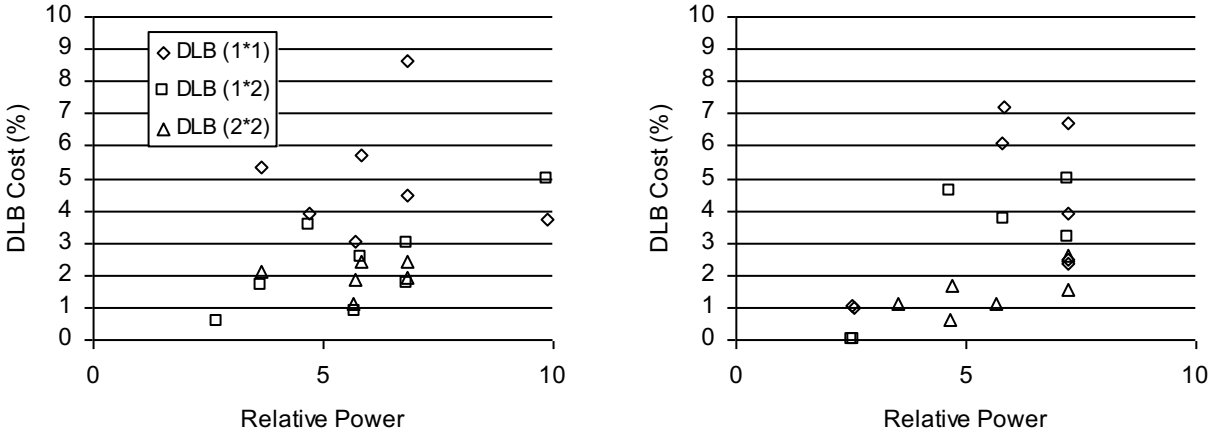
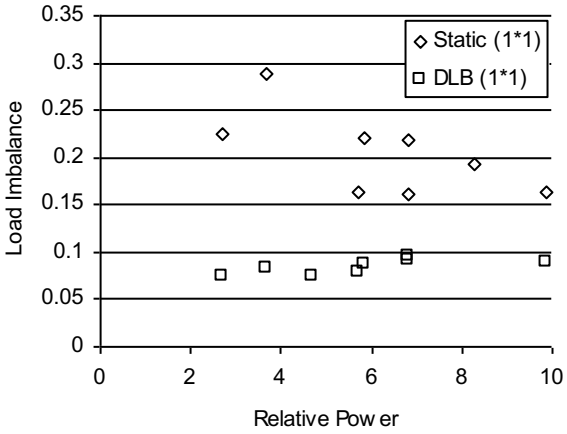



Figure 6.33 – Dynamic Load-Balancing cost (DLB) relative to the total computational time associated with a) 500×589 grid and b) 834×981 grid

As expected, for the same number of templates, DLB improved load imbalance compared to the tests with static load balancing (Figure 6.34, Figure 6.35 and Figure 6.36). The cost associated with DLB is relatively low for the 1×2 and 2×2, so we expect that when the throughput associated with .Net remote communications improves this will probably be the best strategy to be applied. DLB will be called only to correct bad machine power predictions or to execute small adjustments during the simulation process with a relatively low cost.

a)




b)

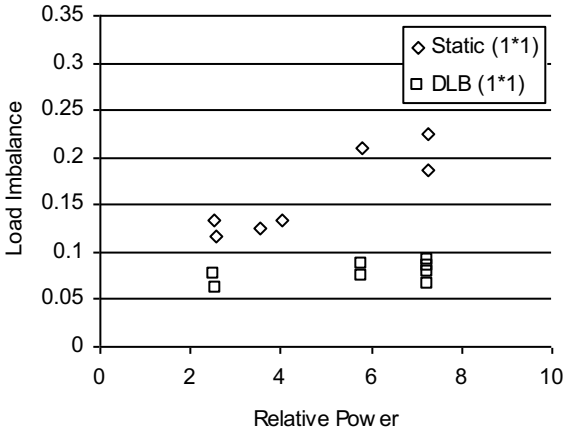
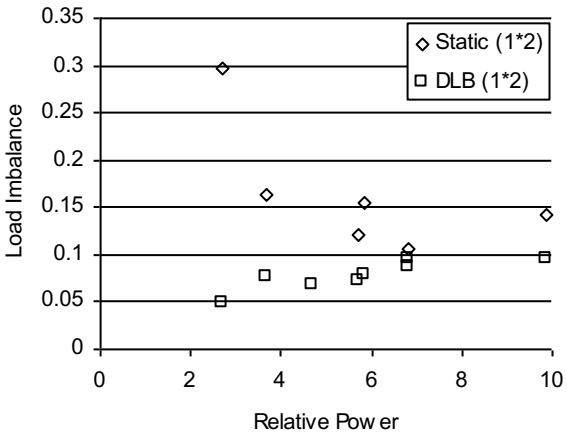



Figure 6.34 : Load Imbalance for both static and DLB for 1×1 template associated with a) 500×589 grid and b) 834×981 grid

a)




b)

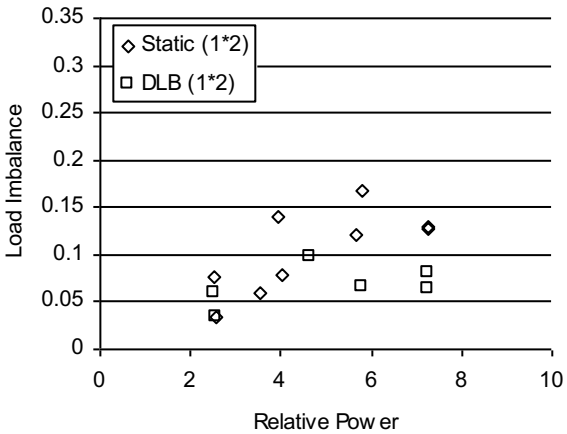



Figure 6.35 - Load Imbalance for both static and DLB for 1×2 template associated with a) 500×589 grid and b) 834×981 grid

a) b)

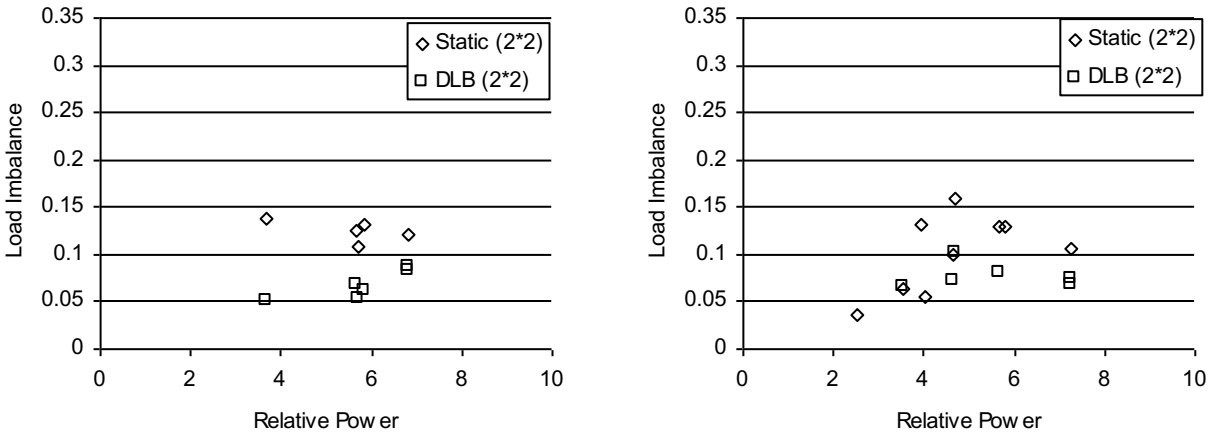



Figure 6.36 - Load Imbalance for both static and DLB for 2×2 template associated with a) 500×589 grid and b) 834×981 grid

## 6.6 Conclusions

This chapter describes and assesses several techniques for the distributed simulation of advection-diffusion based on DisPar methods for shallow waters on a heterogeneous PC cluster and introduces AORDA, an adaptive partitioning method for heterogeneous clusters. The developed software to implement these techniques (Scalable DisPar) follows a 100% component based approach and was implemented using the Microsoft .Net framework version 1.0.

Experimental results revealed that the performance of both static and dynamic repartitioning strategies is completely dominated by the number of neighboring machines. This was mainly caused by the heuristics for remote procedure calls under multithreaded environments in the current Microsoft .NET framework version 1.0. Even with asynchronous calls the communication cost was still the primary factor that caused the not so good performance and scalability. The communication cost is a dominant factor not only for the simulations for the grid with 500×586 cells, but also for the larger grid with 834×981 cells. In both cases the communication cost associated with the exchange of mass between neighboring sub-domains at each time step, grew linearly with the number of neighboring machines. However, as it can be verified in all situations tested, 8 was the maximum number of

neighboring machines for a cluster with at least 15 PCs, which means that the application will probably scale.

Scattered partitioning applied by solely dividing the computational domain in 1×2, 2×2, … partitions has proved to be very efficient combined with 2 or 3 templates after which load imbalance tends to increase again. This is mainly caused by the shape of the Tagus estuary and by the simple scatter strategy used. Therefore, since the computational domain is discrete and taking into consideration that the AORDA method is based on the domain shape, scatter partitioning will be only efficient to mitigate load imbalances if scatter partitioning is applied taking into consideration each template size. Nevertheless, and as it was pointed out by [52], the communication costs can jeopardize scatter partitioning efficiency, which can only be mitigated to a certain level by asynchronous communications.

For the Tagus estuary, the communication cost associated with dynamic load balancing showed a low price relative to the total computational cost, which might express the effectiveness of this strategy. The shortcoming associated with communications did not allow us to take any conclusion based on the computational results relative to the efficiency provided by static scatter partitioning and adaptive scatter partitioning. Nevertheless, adaptive scatter partitioning with 2 or 3 templates will be probably the most efficient strategy if the Microsoft .NET remoting throughput improves. Dynamic load balancing in this situation will be basically trigged to correct possible bad processor power predictions or to execute small corrections during the simulation.

The use of an object-oriented design to build up Scalable DisPar was found to be very important for providing the possibility to integrate software developers with different type of knowledge. New numerical methods based on particle displacement moments (DisPar family) can be built without any knowledge about distributed computing by solely adding two methods.

Finally, it can be concluded that the use of new software tools like the Microsoft .NET framework in this type of applications had proved to be quite cumbersome. Not being an open source tool, it is sometimes difficult to understand problems like the observed problem in throughput. The

productivity provided by very user-friendly compilers and powerful mechanisms for debugging is completely pushed back by this type of problems.

# 7 Conclusions

In this chapter we will conclude the dissertation by giving an overview of the main results that have been presented in both the two parts that constitute the research work. These are organized around the following two main disciplines: numerical formulations for advection-diffusion models based on Markov processes and their implementation on a heterogeneous PC Cluster. The chapter ends with the work that remains to be done.

## *7.1 Developed work*

### 7.1.1 Part I – Chapters 2, 3, 4 and 5

In Chapter 2 it has been shown that there is a direct relation between particle displacement moments for a Gaussian distribution and the truncation error associated with any explicit formulation for advection-diffusion processes.

This was made possible by assuming that any explicit numerical formulation for the Fokker-Planck equation has a direct relation with a discrete numeric particle displacement distribution. If the formulation is decomposed into Tayler Series relative to some point and expressed as function of the spatial derivatives, the coefficients represent a difference between displacement moments. One of the moments is from a Gaussian distribution with an average and variance equal to the Fokker-Planck parameters (i.e., water velocity and Fickian variance). The other moment is obtained from the distribution for the particle jump associated with the numerical formulation itself. Therefore, the coefficient associated with the spatial derivative of order r is function of the difference between the Gaussian and numerical moments also of order r. If the moments of order r are equal, the formulation has no error associated with the $r^{th}$ spatial derivative.

The relation established for the Fokker-Planck equation is perfectly valid for the transport equation expressed as mass concentration. Therefore,

particle displacement moments are just vital for the study of any numeric formulation for the transport or Fokker-Planck equations.

This relation has brought a physical meaning to any error of any order associated with a numerical formulation. According to the central limit theorem, one knows that as the number of time steps increases, the solution for the particle displacement converges to the Gaussian distribution. This relation emphasizes the importance that the introduction of numerical dispersion can have on the final results. If the formulation produces a second order particle displacement moment different from the Gaussian one, the method will be conditionally convergent. It will be only convergent if the difference is function of the time step.

It is also possible to verify that if the numerical formulation produces a negative value in a displacement moment of even order, a physical inconsistency is produced. By definition an even order must be always positive. This is a very similar problem to Courant numbers larger than 1 in Eulerian formulations restricted by this parameter.

The importance of particle displacement moments is shown in Chapter 3 by developing a semi-Lagrangian method, DisPar-k, based on Gaussian moments. Unlike most of the methods for the advection-diffusion equation, DisPar-k discretizes the Gaussian particle displacement distribution into a user specified number of units. The Gaussian parameters are obtained exactly in the same way as the traditional particle tracking methods, by establishing a relation between the Fokker-Planck and the transport equations.

The results obtained in theoretical tests are in agreement with the theoretical developments made in Chapter 2. Although these developments have been done for linear situations, the test made for the Tagus estuary was also consistent with the theory. The accuracy increases with the number of units used in the discretization of the particle displacement distribution.

On a computational point of view, DisPar-k is quite efficient since the system to be solved is the well-known Vandermonde one, which has very good algorithms already developed by the mathematical community (for

example, [26]). Besides this fact, the model is very easy to implement in 2D or even 3D situations. The 2D or 3D numerical distribution is simply obtained by the product of the independent numerical probabilities over each direction.

However, as it was illustrated by the theoretical tests, the dispersion represents the main shortcoming of DisPar-k. To keep the model without wiggles and stable, sometimes it is necessary to drastically increase the number of units associated with the Gaussian discretization process. As this number of discrete units increase, the system to be solved also increases. This creates a problem with round off even for a Vandermonde system.

Chapter 3 had introduced a more powerful version of DisPar-k explicitly based on volumes (DisParV). This was made possible by analyzing two independent variables: particle initial position and particle displacement caused by the physical parameters. Besides giving more versatility, this type of approach can be used to aggregate domain volumes overcoming the DisPar-k shortcoming associated high values of dispersion. However, it was only possible to formulate this possibility, and no mathematical developments were made to support it.

Although the mathematical developments made for DisParV assessment were quite modest, we are strongly convinced that the foundations toward unstructured grids were launched. By using the two independent variables it will be possible to develop numerical formulations based on particle displacement moments for unstructured grids.

In Chapter 5 the importance of nonlinearities in advection diffusion models is analyzed. Similarly to particle tracking methods, if a non-continuous profile exists in one of the parameters, local mass imbalances are expected to happen. The only way to disguise this type of problems is by changing the physical parameters. This is what is frequently done in Eulerian formulations.

This type of situations is usually associated with an incorrect number of dimensions or rough discretization processes. The presence of local mass imbalances is basically a protest. It hints that the underlying physics was not

respected! On the other hand, if the continuity presupposition is accomplished, the model works as expected.

In our opinion, the study of nonlinearities is vital for the production of calibrated and validated results. For example, if a model is calibrated with some spatial resolution, will it be possible to validate it with other numerical method? Will it be possible to validate it with a thinner spatial resolution? Therefore, we believe that more attention must be paid to the problem of nonlinearities. According to the author's knowledge, no theoretical studies exist for transport models.

### 7.1.2 Part II – Chapter 6

The second part of this thesis (Chapter 6) was dedicated to the computational implementation of the developed numerical methods. This chapter introduces a new partitioning scheme for heterogeneous PC clusters, the AORDA method. It also presents Scalable DisPar, which is a software package of components for the distributed simulation of DisPar models. The software supports several partitioning methods like the AORDA one. Therefore, Scalable DisPar is prepared to dynamically rebalance loads. Besides this feature, it is also prepared to use scatter partitioning.

The tests made in the Tagus Estuary with simple AORDA, static scatter partitioning and a scatter partitioning with dynamic load balancing were not conclusive. Even with asynchronous communications, the number of neighboring domains was found to be the main cause in the loss of efficiency. The throughput provided by the Microsoft .NET framework 1.0 was found to be quite limitative for this type of applications.

The communication cost associated with the exchange of mass between neighboring sub-domains at each time step, grew linearly with the number of neighboring machines. However, it was possible to verify that, for a cluster composed by a maximum of 15 machines, all tested situations had a maximum of 8 neighboring machines. This suggests that the application will probably scale.

Static scatter partitioning has proved to be very efficient by progressively decreasing load imbalances with 2 and 3 templates. For more

templates load imbalances tend to increase again. This was mainly caused by the simple scatter partitioning strategy used. The templates were found by simply diving the computational domain in, for example, 1 by 2 templates, 2 by 2 templates and so on. Since the AORDA method is based on the domain shape, this type of approach can produce very small templates and therefore reduce its capacity to correctly split them.

For the Tagus estuary, the communication cost associated with dynamic load balancing turned out to be very small compared to the total computational time. This fact probably expresses the effectiveness of the adaptive component. As it was mentioned, the communication cost made unviable the possibility to take conclusions about the efficiency provided by each partitioning strategy. However, it is strongly suggested that an adaptive scatter partitioning with 2 or 3 templates with asynchronous communications, will be the most efficient approach. Dynamic load balancing will be used to correct possible bad processor power predictions.

## *7.2 Work to be done*

The relation between particle displacement moments and truncation errors produced the theoretical foundations to develop models with complete control over the associated numerical error.

Yet, many more mathematical studies must be carried out in order to extend this possibility to unstructured grids. This has left the following question to be answered. How to develop a method for this type of meshes based on particle displacement moments?

Besides the issues related to the type of grids, it is also possible to suggest that the study of non-linearities in advection-diffusion models is one of the key issues for the success of this type of models. Therefore, the study of grid generation for advection-diffusion models must receive an expressive attention in future studies. How to generate a grid which will minimize the numerical dispersion associated with the advection-diffusion model?

The relation between particle displacement moments and truncation errors for implicit formulations was not assessed in this thesis. Of course, this

creates a new question. Is there any relation between particle displacement moments and truncation errors for implicit formulations?

The problems found in the used version of the Microsoft .NET framework, did not allow us to take conclusions about the relative parallel efficiency of the developed partitioning strategies. Therefore, a key issue to be done in the future is to understand how to solve the problems with concurrency of .NET remoting components. After that, a new set of results for the parallel efficiency must be done. Will scatter partitioning with dynamic load balancing be the best strategy as it was suggested?

# 8 References


[1] Andrews, G.R. (2000). Foundations of Multithreaded, Parallel and Distributed Programming. Addison Wesley.

[2] Antaki, J.F; Blelloch, G.E.; Ghattas, O.; Malcevic, I.; Miller, G.L and Walkington, N.J. (2000). “A Parallel Dynamic-Mesh Lagrangian Method For Simulation of Flows With Dynamic Interfaces”. Proceedings of Super Computing 2000.

[3] Baptista A.M., E.E. Adams and P. Gresho, (1995). “Benchmarks for the Transport Equation: The Convection-Diffusion Forum and Beyond.” Quantitative Skill Assessment for Coastal Ocean Models, Lynch & Davies, eds., AGU Coastal and Estuarine Studies, 47, 241-268.

[4] Baptista, A.M., (1987). “Solution of Advection-Dominated Transport by Eulerian-Lagrangian Methods using the Backward Method of Characteristics. Ph.D. dissertation (Massachusetts Institute of Technology, Cambridge, M.A.)

[5] Boogaard, H., Hoogkamer, M. and Heemink, A. (1993). “Parameter Identification in Particle Models.” Stochastic Hydrology and Hydraulics 7, 109-130.

[6] Buyya, R. (1999). High Performance Cluster Computing Volume 1 Architectures and Systems. Prentice Hall PTR.

[7] Buyya, R. (1999). High Performance Cluster Computing Volume 2 Programming and Applications. Prentice Hall PTR.

[8] Camara, A. and Randall, C. (1984) “The QUAL II Model”, Journal of Environmental Engineering, ASCE, 110, 5, 993-995.

[9] Castro, P. (1996). “Dynamic Water Quality modeling using cellular automata”, Ph.D. Thesis, Virginia Tech, Blacksburg, Virginia.

[10] Chapra, S. (1997). Surface Water Quality Modeling. McGraw-Hill International Editions.

[11] Costa M, Ferreira J. S. (2002). “Particle Distribution Model applied to non-uniform/regular grid” In Proceedings XIV International Conference on Computational Methods in Water Resources, Delft, The Netherlands, 1275-1282.

[12] Costa M., Ferreira J.S., Baptista G., Lobo F., Nobre E. and Câmara A.(2002) “Component based application for 2D advection-diffusion distributed simulation” In Proceedings Hydroinformatics 2002, Cardiff, UK, 1527-1532.

[13] Costa M., Ferreira J.S., Lobo F., Nobre E., Câmara A. (2001).” Scalable DisPar: uma aplicação distribuída e interoperável para simulações 2D de Advecção-Difusão” In A Hidroinformática em Portugal 2001. (In Portuguese)

[14] Costa M., Ferreira J.S., Lobo F., Nobre E., Câmara A. (2002). “Scalable DisPar: A component based application for 2D advection-diffusion distributed simulation” In AI, Simulation and Planning In High Autonomy Systems, Lisboa, Portugal, 214-218.

[15] Costa, M. & Ferreira, J.S., (2000). “Discrete Particle Distribution Model for Advection-Diffusion Transport.” Journal of Hydraulic Engineering, ASCE. 126 (7), 525-532.

[16] D.M. Nicol and J.H. Saltz (1990). “An analysis of scatter decomposition”. IEEE Transactions on Computers, 11(39), 1337-1345.

[17] Dimou K. & Adams E.(1993). “A Random-Walk Particle Tracking Model for Well-Mixed Estuaries and Coastal Waters.” Estuarine, Coastal and Shelf Science, 37, 99-110.

[18] Dowd, K and Severance, C. (1998). High Performance Computing. O’REILLY.

[19] Ferreira J.S. & Costa M. (2001). “Two-Dimensional Advection-Diffusion Model based on Markov Processes.” In 3rd International Symposium on Environmental Hydraulics. December 5-8, 2001, Arizona State University, USA.

[20] Ferreira, J.S. & Costa, M. (2002). “A Deterministic Advection-Diffusion Model based on Markov Processes”. Journal of Hydraulic Engineering, Special Issue on Stochastic Hydraulics, ASCE. Vol. 128, 4, 399-411.

[21] Fischer H.B., List E.J., Koh R.C.Y., Imberger J. & Brooks N.H. (1979). Mixing in Inland and Coastal Waters. Academic, New York.

[22] Forrest, M. H.; Preissmann A. (1977). “Accurate Calculation of Transport in Two Dimensions”. Journal of The Hydraulics Division, ASCE, Vol. 103, 11, 1259-1277.

[23] Fortunato, A.B. and A. Oliveira (2000). On the Representation of Bathymetry by Unstructured Grids. In: L.R. Bentley et al. (Eds.), Computational Methods in Water Resources XIII, Balkema, Vol. 2, 889-896.

[24] Fox G.; Johnson, M.A; Lyzenga, G.A.; Otto, S. W.; Salmon; J. K. and Walker, D.W. (1988). Solving Problems on Concurrent Processors, Vol.1, Prentice-Hall International Editions.

[25] Gardiner, C. W. (1985). Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences, second edition. Springer Verlag.

[26] Golub, Gene H. & Van Loan, Charles F. (1996). Matrix Computations. The Johns Hopkins University Press.

[27] Healey, R.; Dowers, S; Gittings, B. and Mineter, M. (1998). Parallel Processing Algorithms for GIS. Taylor&Francis.

[28] Heemink, A. (1990). “Stochastic Modelling of Dispersion in Shallow Water.” Stochastic Hydrology and Hydraulics, 4, 161-174.

[29] Hoffman, J. (1992). Numerical Methods for Engineers and Scientists. McGraw-Hill International Editions.

[30] Holly, F.O. and Preissmann, A. (1977). “Accurate calculation of transport in two dimensions”. Journal of the Hydraulics Division, ASCE, Vol. 103, 11, 1259-1277.

[31] Jones, M.T. and Plassmann, P.E. (1994). “Parallel Algorithms for the Adaptive Refinement and Partitioning of Unstructured Meshes”. In Proceedings Scalable High Performance Computing Conference 94, IEEE, 478-485.

[32] Lee, P.C.S.; Zaveri, R.A.; Easter, R.C. and Peters, L.K (1998). “Technical note On the parallelization of a global climate-chemistry modeling system”. Atmospheric Environment, Pergamon, 33, 675-681.

[33] Li, C.W.; Yu, T.S. (1994). “Conservative Characteristics-Based Schemes for Mass Transport”. Journal of Hydraulic Engineering, ASCE, Vol. 120, 9, 1089-1099.

[34] Lin B.; Falconer, R.A. (1997). “Tidal Flow and Transport Modeling Using Ultimate Quickest Scheme”. Journal of Hydraulic Engineering, ASCE, Vol. 123, 4, 303-314.

[35] Lin, H.X.; Heemink, A.W. and Stijnen, J.W. (1999). “Parallel Simulation of Transport Phenomena With The Particle Model Simpar”. Proceedings of the International Conference on Simulation and Scientific Computing.

[36] Lin, H.X.; ten Cate, H.H.; Dekker, L.; Heemink, A.W.; Roest, M.R.T.; Vollebregt, E.A.H.; van Stijn, Th.L. and Berlamont, J.B.(1995). “Parallel simulation of 3-D flow and transport models within the NOWESP project”. Simulation Practice and Theory, 3, 257-271.

[37] Loft, R.D.; Thomas, S.J. and Dennis, J.M. (2001). “Terascale spectral element dynamical core for atmospheric general circulation models”. In Super Computing. 2001, ACM.

[38] Luong, P.; Breshears, C.P. and Ly L.N. (2001). “Costal Ocean Modeling of the U.S. West Coast with Multiblock Grid and Dual-Level Parallelism. In Proceedings of Super Computing, 2001.

[39] Manson, J.R.; Wallis, S.G. (1995). “An accurate numerical algorithm for Advective Transport”. Communications in Numerical Methods in Engineering, Vol. 11, 1039-1045, John Wiley & Sons, Ltd.

[40] Manson, J.R.; Wallis, S.G. (1998). “Accurate Simulation of Transport Processes in Two-Dimensional Shear Flow”. Communications in Numerical Methods in Engineering, Vol. 14, 863-869, John Wiley & Sons, Ltd.

[41] Manson, J.R.; Wallis, S.G. (2000). “A conservative semi-Lagrangian fate and transport model for fluvial systems – I. Theoretical development”. Water Resources, Pergamon, Vol. 34, 15, 3769-3777.

[42] Manson, J.R.; Wallis, S.G. (2001). “A conservative semi-Lagrangian transport model for rivers with transient storage zones”. Water Resources Research, American Geophysical Union, Vol. 37, 12, 3321-3329.

[43] Manson, J.R.; Wallis, S.G.; Wang, D. (2000). “A conservative semi-Lagrangian fate and transport model for fluvial systems – I. Numerical testing and practical applications”. Water Resources, Pergamon, Vol. 34, 15, 3769-3777.

[44] Neuman, S.P. (1984). Adaptative Eulerian-Lagrangian Finite Element Method for Advection-Dispersion, International Journal of Numerical Methods in Fluids, 20, 321-337.

[45] Oliveira A., Fortunato A.B. & Baptista A.M. (2000). “Mass Balance in Eulerian-Lagrangian Transport Simulations in Estuaries” Journal of Hydraulic Engineering. 126 (8), 605-614.

[46] Oliveira, A & Fortunato, A.B. (2002). “Toward an Oscillation-Free, Mass Conservative, Eulerian-Lagrangian Transport Model”. Journal of Computational Physics. 183, 142-164.

[47] Oliveira, A. &. Baptista A.M. (1995). “A comparison of integration and interpolation Eulerian-Lagrangian methods.” International Journal of Numerical Methods in Fluids, 21(3),183-204.

[48] Press, W.H; Teukolsky, S.A.; Vetterling, W.T. and Flannery, B.P. (1992). Numerical Receips in C Second Edition. Cambridge University Press.

[49] Qiang, J.; Ryne, R.D. and Habit, S. (2000). “Self-Consistent Langevin Simulation of Coulomb Collisions in Charged-Particle Beams”. Proceeding of SuperComputing 2000, ACM.

[50] Risken, H.(1989). The Fokker-Plank Equation – Methods of Solution and Application, Second Edition. Springer Verlag.

[51] Roe, K.; Stevens, D. and McCord, C. (2001). “High resolution weather modeling for improved fire Management”. In Super Computing 2001, ACM.

[52] Roest, M (1997). “Partitioning for Parallel Finite Difference Computations in Coastal Water Simulation”. Ph.D. Thesis, Delft University of Technology, Netherlands.

[53] Russell, T.F. (2002). "Numerical dispersion in Eulerian-Lagrangian methods". In Proceeding of Computational Methods in Water Resources, edited by Hassanizadeh, S.M.; Schotting, R.J.; Gray, W.G. and Pinder, G.F., 963-970.

[54] Salmon, J. (1988). “A mathematical analysis of the scattered decomposition”. In Proceedings of the 4th Conference on Hypercubes, Concurrent Computers and Applications, Vol. 1, 239-240.

[55] Shingu, S.; Takahara, H; Fuchigami, H.; Yamada, M.; Tsuda, Y.; Ohfuchi, W. Sasaki, Y.; Kobayashi, K.; Hagiwara, T.; Habata, S.; Yokokawa, M; Itoh, H. and Otsuka, K. (2002). “A 26.58 Tflops global atmospheric simulation with the spectral transform method on the Earth Simulator”. In Super Computing, 2002, ACM.

[56] Srinivasan, S.G.; Ashok, I.; Jônsson, H.; Kaloji, G.; Zahorjan, J.(1997). “Dynamic-Domain-decomposition parallel molecular dynamics”. Computer Physics Communications, 102, 44-58.

[57] Stijnen, J. W (2002). “Numerical Methods for Stochastic Environmental Models”. Ph.D. Dissertation, Delft University of Technology, Netherlands.

[58] Stijnen, J. W, Heemink, A.W. and Lin, H.X. (2001). “Higher order Numerical Methods for Pollutant Transport”. In 3rd International Symposium on Environmental Hydraulics (Tempe, Arizona, USA).

[59] Tompson, A.F.B. & Grelhar, L.W (1990). “Numerical Simulation of Solute Transport in Three-Dimensional, Randomly Heterogeneous Porous Media” Water Resources Research, 26 (10), 2541-2562.

[60] Tsai, W.F.; Shen, C.Y.; Fu, H.H. and Kou, C.C. (1999). “Study of parallel computation for Ground-Water solute transport”. Journal of Hydrologic Engineering, ASCE, 4(1), 49-56.

[61] Van Kampen, N. G. (1992). Stochastic Processes in Physics and Chemistry, Second Edition. North-Holland Publications.

[62] Vittoli, C.; Wilders, P.; Manzini, M. and Fotia, G. (1998). “Distributed parallel computation of 2D miscible transport with multi-domain implicit time integration”. Simulation Practice and Theory, Elsevier, 6, 71-88.

[63] Vollebregt, E.A.H and Roest, M.R.T. (2002). “Parallel Shallow Water Simulation for Operational use”. In Parallel Computation Fluid Dynamics – Practice and Theory, Edited by P. Wilders, A. Ecer, J. Periaux, N. Satofuka and P. Fox, Elsevier Science.

[64] Vollebregt, E.A.H. (1997). Parallel Software Development Techniques for Shallow Water Models. Ph.D. Thesis, Technical University of Delft, Netherlands.

[65] Vreugdenhil, C.B (1989). Computational Hydraulics An Introduction. Springer-Verlag.

[66] Wallis S.G., Neelz S. & Manson J.R. (2002) “Solute transport modelling in unsteady river flows using DISCUS”. Proceedings of 14th International Conference on Computational Methods in Water Resources, Delft, The Netherlands, June 23-28, pp 1701-1708.

[67] Walshaw, C. and Cross, M. (1999) “Dynamic Mesh Partitioning & Load-Balancing for Parallel Computational Mechanics Codes”. In The 3rd Euro-

Conference on Parallel and Distributed Computing for Computational Mechanics, Weimar, Germany.

[68] Wu, Y.S.; Zhang, K.; Ding, C.; Pruess, K.; Elmroth, E. and Bodvarsson, G.S. (2002). “As efficient parallel-computing method for modeling nonisothermal multiphase flow and multicomponent transport in porous and fractured media”. Advances in Water Resources, 25, 243-261.

[69] Yokokawa, M; Itakura, K.; Uno, A; Ishihara, T. and Kaneda, Y. “16.4 Tflops direct numerical simulation of turbulence by Fourier Spectral Method on the Earth Simulator”. In Super Computing, 2002, ACM.

[70] Yu, C.S.; Chang, H.Y; Tcheng, S.C.and Huang, K.C. (2000). “Modeling tidal flow currents in the Taiwan strait on parallel computers” In High Performance Computing 2000, Grand Challenges in Computer Simulation, Edited by Adrian Tentner, Washington, US.

[71] Zaki, Mohammed J., Parthasarathy Srinivasan, Li W. (1999). Customized Dynamic Load Balancing. In High Performance Cluster Computing: Architectures and Systems, Volume 1, Ed. By Rajkumar Buyya. Prentice-Hall, New Jersey.

[72] Zoppou, C. & Knight J.H. (1997) “Analytical Solution for Advection and Advection-Diffusion Equations with Spatially Variable Coefficients.” Journal of Hydraulic Engineering. 123 (2), 144-148.

# 9 Notation

C – Concentration

h – water depth

D – dispersion coefficient

$P(x_2,t_2|x_1,t_1)$ – probability of a particle to move to $x_2$ at time $t_2$ if it was in $x_1$ at time $t_1$

P(x,t) – probability for a particle to be in x at time t

$\langle x^r \rangle$ - expectation of x of order r

$\sigma^2(x)$ – variance of x

$2k_x+1$ – number of units used in the Gaussian distribution discretization

$G_r$ - error associated with the spatial derivative of P of order r

$\Delta t$ – time step

$\Delta x$ – spatial resolution over the x direction

$\Delta y$ – spatial resolution over the y direction

$\Psi$ - matrix of probabilities

W – matrix with transition probabilities

$R_j$ – matrix with products of D by u

S - matrix with the particle displacement moments coefficients relative to x

E - numerical expectations relative to x

L – diagonal matrix with the inverse of factorial values

$\eta_x$ – matrix with spatial derivatives of P of different orders

$\eta_t$ – matrix with temporal derivatives of P of different orders

V[i] – spatial volume with index i

$\beta_i$ – integer part of the particle displacement average for regular grids and volume where this average value falls for non-uniform grids

$\chi i$ – particle position in volume v[i]

$\delta x_i$ – displacement after a time step caused by the transport parameters

$x_{adv}$ - particle position influenced by the flow motion

$x_{disp}$ - particle position influenced by the turbulence

Q – flow motion

$M_i$ – mass present in cell, volume or node i

$L_p$ is the workload associated with processor p

F - partitioning frequency

$t(G_i,S_i)$ - computation time for processing a sub-grid Gi by a set of processors Pi

$S_i$ - total estimated actual processing power

# 10 Appendix I – Mathematical studies for the Numerical Developments

This Appendix shows some developments, which made it possible to prove the stated results on the chapter 2.

These developments include the formulation and demonstration of 4 theorems.

## *10.1 Gaussian Distribution studies*

For any distribution it is possible to express its moments of order n as follows:

$$\left\langle \left( x^{v} \right) \right\rangle = \left\langle \left( x - \langle x \rangle + \langle x \rangle \right)^{v} \right\rangle \tag{A.1}$$

Decomposing expression (A.1) according to the binomial theorem yields a new expression as:

$$\left\langle x^{v} \right\rangle = \sum_{j=0}^{v} \binom{v}{j} \left\langle \left( x - \langle x \rangle \right)^{j} \right\rangle \langle x \rangle^{v-j} \tag{A.2}$$

All odd terms from E[(x-E(x))j] are zero which means that expression (A.2) can be rewritten as:

$$\left\langle x^{v} \right\rangle = \sum_{j=0}^{\rho-1} \binom{v}{2j} \left\langle \left( x - \langle x \rangle \right)^{2j} \right\rangle \langle x \rangle^{v-2j} \tag{A.3}$$

where $\rho$=(v+2)/2 if n is even or $\rho$=(v+1)/2 if n is odd.

To get the Gaussian moment of order n expressed only as function of average and variance two theorems will be formulated and shown.

### 10.1.1 Theorem 1

If x is a random variable with Gaussian distribution it is possible to establish the following relationship:

$$\left\langle \left(x - E(x)\right)^{2j} \right\rangle = \frac{(2j)!}{2^j\, j!}\left(\sigma^2(x)\right)^j \tag{A.4}$$

Demonstration

The Gaussian moment of order v is written by definition as:

$$\left\langle x^v \right\rangle = \int x^v \frac{1}{\sqrt{2\sigma^2(x)\pi}} exp\left( -\frac{\left(x - \langle x \rangle\right)^2}{2\sigma^2(x)} \right) dx \tag{A.5}$$

Integrating this expression by parts it is possible to express the moment of order v+2 as function of the two earlier ones and thus:

$$\left\langle x^{v+2} \right\rangle = \langle x \rangle \left\langle x^{v+1} \right\rangle + (v+1)\sigma^2(x)\left\langle x^v \right\rangle \tag{A.6}$$

To demonstrate the equality (A.6) let us assume, for example, that v is zero in expression (A.6) and replace the independent variable x by a new one centered on average. In this case the first product of the right-hand side is always zero, since the independent variable is of odd order and is centered on average. Now, using expression (A.4) to get both expectations on both sides of equation (A.6) it is possible to verify that both are in fact equal, which means that equation (A.3) is true for that case and therefore for all the others, proving the theorem by induction.

### 10.1.2 Theorem 2

If x is a random variable with Gaussian distribution the Gaussian moment of order n can be yielded as function of average and variance replacing expression (A.4) in the expression (A.3):

$$\left\langle x^v \right\rangle = \sum_{j=0}^{\rho-1} \frac{v!}{2^j\, j!(v-2j)!}\left(\sigma^2(x)\right)^j \langle x \rangle^{v-2j} \tag{A.7}$$

Demonstration

To demonstrate this theorem it is possible to use again the expectation relationship (A.6). Thus, to perform the demonstration by induction let us assume that v=1 and use expression (A.7) to get the right-hand side as function of average and variance. The result obtained is formally equal to the result produced by expression (A.7) for the third order

moment. An even order can also be applied to the left-hand side of equation (A.6) and verify that both sides are equal. Thus, it was proved by induction that E(xn) can be expressed as function of average and variance like in expression (A.7).

## *10.2 Fokker-Planck equation studies*

### 10.2.1 Theorem 3

The linear Fokker-Plank equation can be expressed as

$$\frac{\partial P}{\partial t} = -u\frac{\partial P}{\partial x} + D\frac{\partial^2 P}{\partial x^2} \tag{A.8}$$

which means that the temporal derivative of order v converted to spatial derivatives can be expressed as:

$$\frac{\partial^v P}{\partial t^v} = \sum_{j=0}^{v}\binom{v}{j} D^j (-u)^{v-j}\,\frac{\partial^{v+j} P}{\partial x^{v+j}} \text{ , v>0} \tag{A.9}$$

Demonstration

To demonstrate this theorem the derivative of order v+1 will be obtained from the one of order v and therefore:

$$\frac{\partial}{\partial t}\left(\frac{\partial^v P}{\partial t^v}\right) = \frac{\partial}{\partial t}\left[\sum_{j=0}^{v}\binom{v}{j} D^j (-u)^{v-j}\,\frac{\partial^{v+j} P}{\partial x^{v+j}}\right] \tag{A.10}$$

Calculating this derivative the expression (A.10) can be yielded as

$$\frac{\partial^{v+1} P}{\partial t^{v+1}} = -u\frac{\partial}{\partial x}\left(\frac{\partial^v P}{\partial t^v}\right) + D^2\,\frac{\partial^2}{\partial x^2}\left(\frac{\partial^v P}{\partial t^v}\right) \tag{A.11}$$

For example, if v is equal to 1 the expression (A.11) can be written as

$$\frac{\partial^2 P}{\partial t^2} = u^2\frac{\partial^2 P}{\partial x^2} - 2uD^2\,\frac{\partial^3 P}{\partial x^3} + D\frac{\partial^4 P}{\partial x^4} \tag{A.12}$$

which is true, proving the theorem.

### 10.2.2 Theorem 4

Let $\lambda$ be the diagonal matrix:

$$\lambda = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & -1 & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & -1 & 0 \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}_{(2k+1)\times(2k+1)} \quad \text{(A.13)}$$

If Z and S are the matrices presented in Truncation Error Analysis section then:

$$\lambda = ZS^{-1} \quad \text{(A.14)}$$

Demonstration

Let B be the matrix

$$B = \lambda S \quad \text{(A.15)}$$

Multiplying the column j from matrix S by the line i from matrix λ, it is possible to write the entry bij from matrix B as

$$b_{ij} = \lambda_{ii} s_{ij} = (-1)^{j-1} s_{ij} = z_{ij} \quad \text{(A.16)}$$

where λii= diagonal entry from matrix λ.

Entry zij is equal to bij, and so Z=B. Writing this equality as:

$$\lambda S = Z \quad \text{(A.17)}$$

and multiplying both sides of the equation by S-1, it is possible to verify that

$$\lambda = ZS^{-1} \quad \text{(A.18)}$$

proving the theorem.

## *10.3 Instantaneous mass spill with linear conditions*

In this section the DisPar-k expression as a function of time and space for three particle destination units was developed for an instantaneous mass spill with linear conditions and for k=1. This expression could be useful in theoretical studies avoiding the dynamic simulation.

Since the DisPar model works in a Random Walk principle, the development of its expression will be based on the particle displacement distribution. So, considering parameters u, D and A constant, probabilities $P_i^{us}$ and $P_i^{ds}$ can be written respectively as $P^{us}$ and $P^{ds}$.

Assuming k as the number of times a particle moves to the right and |z| (considering that z is negative) the number of times a particle moves to the left after n time steps, (k+|z|) is the event "success of dispersion" in n time successive trials. This event has the following probability:

$$\left(P^{us}\right)^{|z|}\left(1-\left(P^{us}+P^{ds}\right)\right)^{n-\left(k+|z|\right)}\left(P^{ds}\right)^{k} \tag{A.19}$$

The number of possible ways or combinations through which the particle moves to the right, which can occur k times in n trials, is:

$$\binom{n}{k}=\frac{n!}{k!(n-k)!} \tag{A.20}$$

The number of times the particle will not move to the right is n-k. So the number of combinations by which the particle moves to the left, which can occur |z| times in n-k trials, is:

$$\binom{n-k}{|z|}=\frac{(n-k)!}{|z|!(n-k-|z|)!} \tag{A.21}$$

The possible number of paths that the particle follows moving k times to the right and |z| times to the left can be given by the product of both combinations:

$$\binom{n}{k}\binom{n-k}{|z|}=\binom{n}{|z|}\binom{n-|z|}{k}=\frac{n!}{k!|z|!\left(n-\left(k+|z|\right)\right)!} \tag{A.22}$$

So, the probability that the particle will move k times to the right and |z| times to the left independently of the order it happens, after n time steps, is:

$$P(k,z)=\binom{n}{k}\binom{n-k}{|z|}\left(P^{us}\right)^{|z|}\left(1-\left(P^{us}+P^{ds}\right)\right)^{n-(k+|z|)}\left(P^{ds}\right)^{k} \tag{A.23}$$

where k and z are the independent variables.

If a particle is found in a cell after n time steps, this is the result of the routes it followed independently of the order they occurred. For instance, if it is found in the cell on the right of the initial one after three time steps, this means that either the particle moved twice to the right and once to the left or that it moved once to the right and never to the left.

**Table 10-1. Possible routes followed by a particle found in cell i after n time steps**

| Route (r)<br>(1) | K<br>(2) | Z<br>(3) |
|---|---|---|
| 0 | i+0 | 0 |
| 1 | i+1 | -1 |
| … | … | … |
| m - 2 | i + m – 2 | -(m – 2) |
| m - 1 | i + m – 1 | -(m – 1) |

So, if a particle is found in cell i on the right of the original cell after n time steps, the possible routes it followed can be seen in table 1. Parameter m corresponds to the number of possible routes (the same logic can be applied if i is negative) and r is the possible route followed by a particle.

The number of successes of the dispersion (k+|z|) has to be smaller than n and so:

$$|i|+m-1+|-(m-1)|\le n \Rightarrow m\le \frac{n-|i|+2}{2} \tag{A.24}$$

Considering that m is integer, the maximum value for this variable fulfilling the relation described above can be expressed as:

$$m=\frac{n-|i|+2}{2}\text{, if (n-| i| ) is even}$$

$$m=\frac{n-|i|+1}{2}\text{, if (n-| i| ) is odd} \tag{A.25}$$

All the possible paths (r) followed by the particle found in cell i are impossible to occur at the same time. Thus, the probability that a particle will be found in this cell after n time steps can be given by the sum of all the probabilities of each path. The expression can be written as:

$$P(X=i)=\sum_{r=0}^{m-1}\binom{n}{|i|+r}\binom{n-(|i|+r)}{r}\left(P^{us}\right)^{r+\frac{|i|-i}{2}}\left(1-\left(P^{us}+P^{ds}\right)\right)^{n-(2r+|i|)}\left(P^{ds}\right)^{r+\frac{i+|i|}{2}} \quad \text{(A.26)}$$

where the independent variable X represents cell i (with i $\in${-n, …, 0, …, n}).

Terms (| i| -i)/2 and (| i| +i)/2 are used to make it possible to work with i positive and negative values. (i.e. when i is positive the first term is equal to zero and the second is i, but when i is negative the first is equal to i and the second is zero).

If, at the origin, all the particles are grouped at time zero, the number of particles in position i$\Delta$x at a subsequent time n$\Delta$t will be proportional to the probability that an individual particle is at i$\Delta$x. Thus, considering that cell 0 has an initial concentration C0, the DisPar model can be written for an instantaneous mass injection as:

$$C_i^n=C_0\sum_{r=0}^{m-1}\binom{n}{|i|+r}\binom{n-(|i|+r)}{r}\left(P^{us}\right)^{r+\frac{|i|-i}{2}}\left(1-\left(P^{us}+P^{ds}\right)\right)^{n-(2r+|i|)}\left(P^{ds}\right)^{r+\frac{i+|i|}{2}} \quad \text{(A.27)}$$

where $C_i^n$ = particle concentration at position i$\Delta$x in time n$\Delta$t.

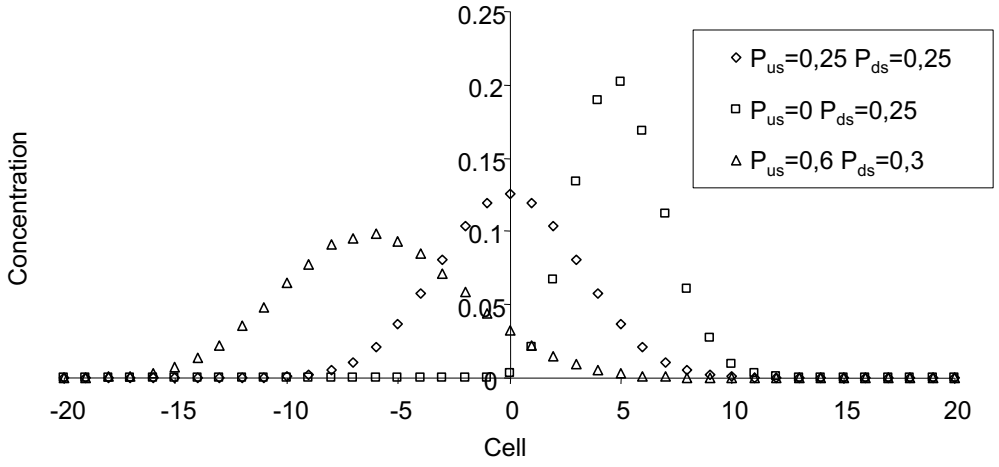



Figure 10.1 - shows some results obtained by the expression (A.27) for 20 time trials with C0=1.

## *10.4 Comparison between DisPar-k and DisParV for constant cell length*

In this section it is proved the evenness between the DisPar formulation for nodes (DisPar-k) and volumes (DisParV) with constant $\Delta x$. Although this equality assumption is rather natural, it is only possible to unconditionally claim it if mathematically proved. The demonstration will be performed by showing the equivalence between both matrices of numerical probabilities for the same number of destination units. DisPar-k nodes are centered on each DisPar-k volume and it is assumed that the particle initial position volume and the particle initial position node are coincident, as well as the particle destination volumes and nodes.

DisParV

Assuming an uniform grid with a constant $\Delta x$ ($x_{i+1}-x_i = \Delta x$, $i \in \{0,1, \ldots\}$) and a uniform distribution for a particle inside V[i], $i \in \{0,1, \ldots\}$, according to the DisParV formulation (Figure 10.2) the expectation matrix (A.29) for the particle displacement is given by (A.31).

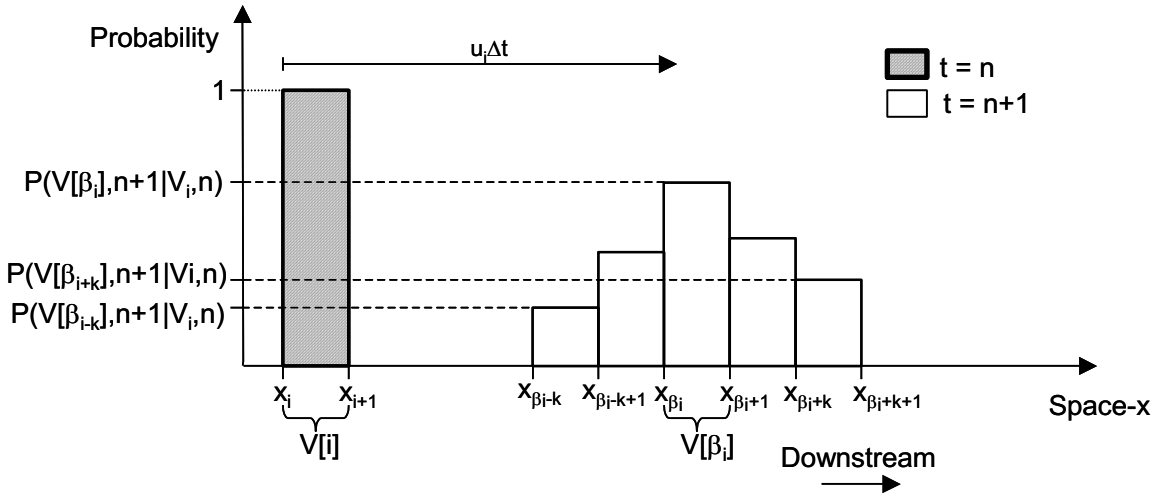



Figure 10.2 – DisParV particle displacement distribution discretization for a regular grid

$$M = \begin{bmatrix} \left\langle \left(\chi_{\beta_i - k}\right)^0 \right\rangle & \cdots & \left\langle \left(\chi_{\beta_i + k - 1}\right)^0 \right\rangle & \left\langle \left(\chi_{\beta_i + k}\right)^0 \right\rangle \\ \vdots & & \vdots & \vdots \\ \left\langle \left(\chi_{\beta_i - k}\right)^{2k-1} \right\rangle & \cdots & \left\langle \left(\chi_{\beta_i + k + 1}\right)^{2k-1} \right\rangle & \left\langle \left(\chi_{\beta_i + k}\right)^{2k-1} \right\rangle \\ \left\langle \left(\chi_{\beta_i - k}\right)^{2k} \right\rangle & \cdots & \left\langle \left(\chi_{\beta_i + k - 1}\right)^{2k} \right\rangle & \left\langle \left(\chi_{\beta_i + k}\right)^{2k} \right\rangle \end{bmatrix}_{(2k+1)(2k+1)} \quad \text{(A.28)}$$

$$E_{V_i} = \begin{bmatrix} \left\langle \left(\chi_i + \delta x_i\right)^0 \right\rangle \\ \vdots \\ \left\langle \left(\chi_i + \delta x_i\right)^{2k-1} \right\rangle \\ \left\langle \left(\chi_i + \delta x_i\right)^{2k} \right\rangle \end{bmatrix}_{(2k+1)} \quad \text{(A.29)}$$

$$P = \begin{bmatrix} P\left(V\left[\beta_i - k\right], n+1 \mid V\left[i\right], n\right) \\ \vdots \\ P\left(V\left[\beta_i + k - 1\right], n+1 \mid V\left[i\right], n\right) \\ P\left(V\left[\beta_i + k\right], n+1 \mid V\left[i\right], n\right) \end{bmatrix}_{(2k+1)} \quad \text{(A.30)}$$

$$E_{V_i} = MP \quad \text{(A.31)}$$

DisPar-k

In the formulation based on nodes each discrete unit is centered in the middle of the DisParV volumes (Figure 10.3). The expectation matrix with the particle displacement moments from the 0 to the 2kth order is given by equation(A.35).

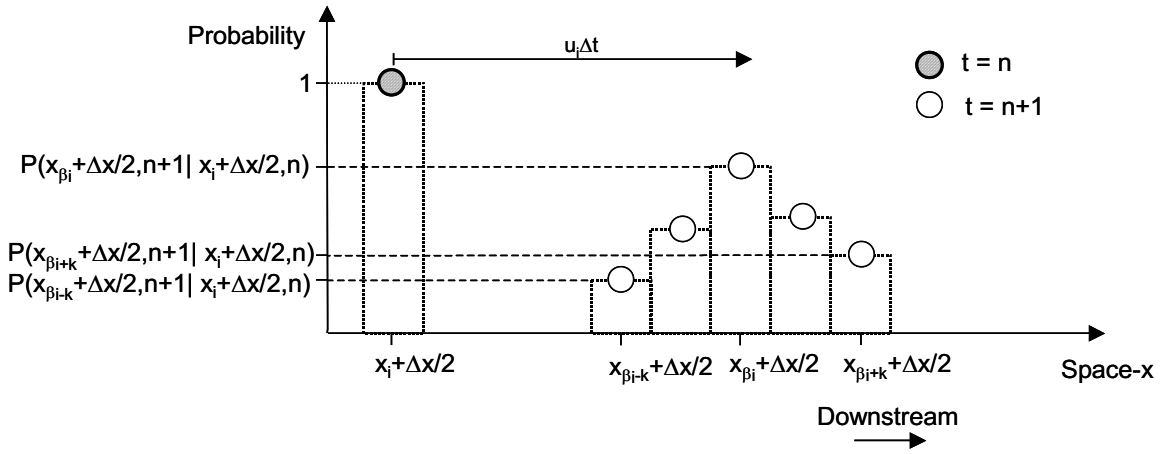



Figure 10.3 – DisPar-k particle displacement distribution discretization

$$M' = \begin{bmatrix} \left(x_{\beta_i-k}+\frac{\Delta x}{2}\right)^0 & \cdots & \left(x_{\beta_i+k-1}+\frac{\Delta x}{2}\right)^0 & \left(x_{\beta_i+k}+\frac{\Delta x}{2}\right)^0 \\ \vdots & & \vdots & \vdots \\ \left(x_{\beta_i-k}+\frac{\Delta x}{2}\right)^{2k-1} & \cdots & \left(x_{\beta_i+k-1}+\frac{\Delta x}{2}\right)^{2k-1} & \left(x_{\beta_i+k}+\frac{\Delta x}{2}\right)^{2k-1} \\ \left(x_{\beta_i-k}+\frac{\Delta x}{2}\right)^{2k} & \cdots & \left(x_{\beta_i+k-1}+\frac{\Delta x}{2}\right)^{2k} & \left(x_{\beta_i+k}+\frac{\Delta x}{2}\right)^{2k} \end{bmatrix}_{(2k+1)(2k+1)} \tag{A.32}$$

$$P' = \begin{bmatrix} P\left(x_{\beta_i-k}+\frac{\Delta x}{2},n+1\,|\,x_i+\frac{\Delta x}{2},n\right) \\ \vdots \\ P\left(x_{\beta_i+k-1}+\frac{\Delta x}{2},n+1\,|\,x_i+\frac{\Delta x}{2},n\right) \\ P\left(x_{\beta_i+k}+\frac{\Delta x}{2},n+1\,|\,x_i+\frac{\Delta x}{2},n\right) \end{bmatrix}_{(2k+1)} \tag{A.33}$$

$$E_i = \begin{bmatrix} \left\langle \left( x_i + \frac{\Delta x}{2} + \delta x \right)^0 \right\rangle \\ \vdots \\ \left\langle \left( x_i + \frac{\Delta x}{2} + \delta x \right)^{2k-1} \right\rangle \\ \left\langle \left( x_i + \frac{\Delta x}{2} + \delta x \right)^{2k} \right\rangle \end{bmatrix}_{(2k+1)} \tag{A.34}$$

$$E_i = M'P' \tag{A.35}$$

To simplify the mathematical treatment the particle position is centered on the initial node (xi+$\Delta$x/2) and therefore the system can be rewritten as:

$$M'' = \begin{bmatrix} \left(x_{\beta_i - k} - x_i\right)^0 & \cdots & \left(x_{\beta_i + k - 1} - x_i\right)^0 & \left(x_{\beta_i + k} - x_i\right)^0 \\ \vdots & & \vdots & \vdots \\ \left(x_{\beta_i - k} - x_i\right)^{2k-1} & \cdots & \left(x_{\beta_i + k - 1} - x_i\right)^{2k-1} & \left(x_{\beta_i + k} - x_i\right)^{2k-1} \\ \left(x_{\beta_i - k} - x_i\right)^{2k} & \cdots & \left(x_{\beta_i + k - 1} - x_i\right)^{2k} & \left(x_{\beta_i + k} - x_i\right)^{2k} \end{bmatrix}_{(2k+1)(2k+1)} \tag{A.36}$$

$$P'' = \begin{bmatrix} P\left(x_{\beta_i - k} - x_i, n+1 \mid 0, n\right) \\ \vdots \\ P\left(x_{\beta_i + k - 1} - x_i, n+1 \mid 0, n\right) \\ P\left(x_{\beta_i + k} - x_i, n+1 \mid 0, n\right) \end{bmatrix}_{(2k+1)} \tag{A.37}$$

$$E'_i = \begin{bmatrix} \left\langle \delta x^0 \right\rangle \\ \vdots \\ \left\langle \delta x^{2k-1} \right\rangle \\ \left\langle \delta x^{2k} \right\rangle \end{bmatrix}_{(2k+1)} \tag{A.38}$$

$$M''P'' = E'_i \tag{A.39}$$

Theorem

If S is the matrix

$$S=\begin{bmatrix} \binom{0}{0}\left\langle \chi_i^{0-0}\right\rangle & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots \\ \binom{2k-1}{0}\left\langle \chi_i^{(2k-1)-0}\right\rangle & \cdots & \binom{2k-1}{2k-1}\left\langle \chi_i^{(2k-1)-(2k-1)}\right\rangle & 0 \\ \binom{2k}{0}\left\langle \chi_i^{2k-0}\right\rangle & \cdots & \binom{2k}{2k-1}\left\langle \chi_i^{2k-(2k-1)}\right\rangle & \binom{2k}{2k}\left\langle \chi_i^{2k-2k}\right\rangle \end{bmatrix}_{(2k+1)(2k+1)} \tag{A.40}$$

then it is equal to the product

$$S = MM''^{-1} \tag{A.41}$$

Demonstration:

To demonstrate the theorem it will be proved the evenness between the product of SM'' and M by showing that all the entries are equal. The demonstration follows through by a simple algebra transformation expressing S as function of M and M''.

Let Z be the matrix

$$Z = SM'' \tag{A.42}$$

Matrix Z entry from row d and column j (zdj) can be written as:

$$z_{dj}=\sum_{r=0}^{d-1}\left[\binom{d-1}{r}\frac{x_{i+1}^{d-r}-x_i^{d-r}}{(d-r)\Delta x}\left(x_{\beta_i-k+j-1}-x_i\right)^r\right] \tag{A.43}$$

The entry d,j from matrix M is defined by the uniform expectations as:

$$m_{dj}=\left\langle \chi_{\beta_i-k+j-1}^{d-1}\right\rangle \tag{A.44}$$

Since the moment of a constant value is equal to the value itself, it is possible to write:

$$\left\langle \left(x_{\beta_i-k+j-1}-x_i\right)^r\right\rangle=\left(x_{\beta_i-k+j-1}-x_i\right)^r \tag{A.45}$$

By this relation and taking into consideration that

$$\frac{x_{i+1}^{d-r} - x_i^{d-r}}{(d-r)\Delta x} = \left\langle \chi_i^{d-r} \right\rangle \tag{A.46}$$

the matrix Z entry can be rewritten as:

$$z_{dj} = \sum_{r=0}^{d-1} \left[ \binom{d-1}{r} \left\langle \chi_i^{d-r} \right\rangle \left\langle \left( x_{\beta_i - k + j - 1} - x_i \right)^r \right\rangle \right] \tag{A.47}$$

This expression represents the application of the binomial theorem to the expectation of (χi+(xβi-k+j-1-xi))d-1 and therefore zdj can be again rewritten as:

$$z_{dj} = \left\langle \left( \chi_i + \left( x_{\beta_i - k + j - 1} - x_i \right) \right)^{d-1} \right\rangle = \left\langle \chi_{\beta_i - k + j - 1}^{d-1} \right\rangle = m_{dj} \tag{A.48}$$

Taking into consideration that (xβi-k+j-1-xi) is a constant, its sum with χi is equivalent to displace the associated x range [xi,xi+1] to [xβi-k+j,xβi-k+j-1]. χ distribution is not changed, which means the expectation of (χi+(xβi-k+j-1-xi))d-1 is equal to the expectation of (χβi-k+j-1)d-1 and consequently both entries from matrix Z and M are equal. By multiplying both sides of the equality Z=M by M''-1 it is possible to express S as

$$Z = M \Rightarrow SM'' = M \Rightarrow S = MM''^{-1} \tag{A.49}$$

proving the theorem.

Theorem

If the previous defined conditions for the DisParV and DisPar-k formulations are met, both matrices of numerical probabilities are equal for any k≥1.

$P = P''$, k≥1 (A.50)

Demonstration:

The product SE'i can be expressed as

$$SE'_i = \begin{bmatrix} \sum_{r=0}^{0} \binom{0}{0} \left\langle \chi_i^{0-r} \right\rangle \left\langle \delta x_i^r \right\rangle \\ \vdots \\ \sum_{r=0}^{2k-1} \binom{2k-1}{0} \left\langle \chi_i^{(2k-1)-r} \right\rangle \left\langle \delta x_i^r \right\rangle \\ \sum_{r=0}^{2k-1} \binom{2k}{0} \left\langle \chi_i^{2k-r} \right\rangle \left\langle \delta x_i^r \right\rangle \end{bmatrix}_{(2k+1)(2k+1)} \quad \text{(A.51)}$$

Each entry from this product represents the application of the binomial theorem to the expectation of ($\chi$i+$\delta$xi). If the matrix is rewritten bearing this relation in mind, it is possible to verify that:

$$SE'_i = E_{V_i} \quad \text{(A.52)}$$

By the previous theorem, S can be rewritten according to (A.41) and by multiplying both sides of the equation by M-1, it is possible to express both matrices of probabilities as:

$$MM''^{-1} E'_i = E_{V_i} \Rightarrow M^{-1} E_{V_i} = M''^{-1} E'_i \Rightarrow P = P'' \quad \text{(A.53)}$$

proving the theorem.

## *10.5 Positivity analyses for the DisPar method*

If each probability in this scheme respects the definition (i.e. lies between 0 and 1), then the positivity and stability are guaranteed.

There is only an upper limit to the space step and it results from the condition applied to $P_i^{us}$ expression (5.24):

$$P(i-1, n+1 \mid i, n) \geq 0 \Rightarrow \Delta x \max_i = \frac{\left(A_{i-1} D_{i-1} + A_i D_i\right)}{A_i u_i} \quad \text{(A.54)}$$

where $\Delta$xmaxi = $\Delta$x maximum value allowed to cell i.

If there is no spatial variation of A and D, this $\Delta$x restriction represents the traditional criteria adopted in explicit schemes for the Peclet number ($u\Delta x/D \leq 2$). Below this limit, there is no lower restriction for the time step,

which has the following upper limit, resulting from the condition applied to $P_i^0$ expression (5.25):

$$P(i,n+1\,|\,i,n) \geq 0 \Rightarrow$$

$$\Rightarrow \begin{cases} \Delta t\max_i = \dfrac{\Delta x\left(0,5a_i - \sqrt{0,25a_i^2 + 4b_i u_i A_i \Delta x + 4u_i^2 A_i^2 \Delta x^2}\right)}{2u\left(-b_i - u_i A_i \Delta x\right)}, \Delta x \neq -\dfrac{b_i}{A_i u_i} \wedge u_i \neq 0 \\ \Delta t\max_i = \dfrac{2A_i \Delta x^2}{a_i}, \Delta x = -\dfrac{b_i}{A_i u_i} \vee u_i = 0 \end{cases} \tag{A.55}$$

where $\Delta$tmaxi = $\Delta$t maximal value allowed for cell i; ai = Ai+1Di+1+2AiDi+Ai-1Di-1 and bi = Ai+1Di+1-Ai-1Di-1

# 11 Appendix II - Computational developments

## *11.1 Definition of two subsets with approximately the same power from an array with processor powers*

This appendix describes the algorithm used to find two subsets with approximately the same total power from an array of values indicating processor powers. The algorithm is a possible approach used to split the processors so that they can be assigned to a new sub-region proportionally to their total computational power. To accomplish this, the first step is to sort the array with the processor powers. Secondly, the fastest processor is assigned to the first element of Set1, the second most powerful processor is assigned to the first element of Set2. After this, the assignment is made for the slowest processors such that the slowest processor is associated with Set1 and the second slowest processor is associated with Set2. Now, the cycle starts again on the antepenultimate fastest processor and on the antepenultimate slowest processors, exactly as it was done previously. This approach will continue in loop alternately assigning the fastest and the slowest processors to the same set. The loop ends when there is no more processors available or when there is one left. If the last case is verified, the processor left will join the set with less total power.

A simplified version of the algorithm will be shown to better demonstrate how it works.

```
static void SplitProcessors(
    in Processor [] processorsIn, //List with the processors to obtain the
  //subsets
    out Processor [] set1,        //Set1 array of processors
    out Processor [] set2)        //Set2 array of processors
```

```
{
//Copy the array with the list of processors
Processor[] processors = processorsIn.Clone();
//Sort the processors according to their power
Array.Sort(processors);
//Allocate temporary arrays
Processors[] set1Tmp = new Processors[processors.Length/2+1];
Processors[] set2Tmp = new Processors[processors.Length/2+1];
int count1 = 0;
int count2 = 0;
while(processors.Length-(count1+count2)!=1)
{
    if(count1 is even)
    {
        set1Tmp[count1] = processors[(processors.Length-1)-
        count1];
        set2Tmp[count2] = processors[(processors.Length-1)-
        count1-1];
    }
    else
    {
        set1Tmp[count1] = processors[(count2-1)];
    set2Tmp[count2] = processors[(count2-1)+1]
    }

    //Increment both counters
    count1++;
```

```
count2++;
}

//If the total number of processors in the array of processors is
//even
//the processor left by the previous loop will be assigned to the set
//with less total power
if(processors.Length!=count1+count2)
{
        int pIndex;
        if(count1 is even)
                pIndex = processors.Length-1-count1;
        else
                pIndex = count2-1;

        if(Sum(set2Tmp)< Sum(set1Tmp))
        {
                set2Tmp[count2] = processors[pIndex];
                count2++;
        }
        else
        {
                set1Tmp[count1] = processors[pIndex];
                count1++;
        }
}
```

```
    //Create the output sets based on the assigned processors
    set1 = new Processor[count1];
    set2 = new Processor[count2];
    //Copy the temporary data to the output sets
    Array.Copy(set1Tmp,0,set1,count1);
    Array.Copy(set2Tmp,0,set2,count2);
}
```