National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Parallel CFD & Optimization Unit

Compression of multidimensional flow fields using Incremental Proper Generalized Decomposition & Molecular Dynamics computations and optimization of coarse-grained Ionic Liquid lubricants

Diploma Thesis

Kliafas Leonidas

Academic Advisor:
Kyriakos C. Giannakoglou, Professor NTUA

Industrial Advisor:
Dr. Konstantinos Gkagkas, Expert Toyota Motor Europe

Athens, 2019

Acknowledgements

First and foremost, I would like to express my gratitude to my professor Kyriakos C. Giannakoglou for our collaboration in such an interesting project. The remarks and the advice he gave me were always very helpful and enlightening. I would also like to thank him for recommending me for the internship position in TME and his assistance throughout this period.

I would also like to acknowledge the team of Advanced Material Research division of TME and especially thank K. Gkagkas, M. Di Gennaro and G. Toraman for the fine collaboration.

Then, I would like to thank all the members of the PCopt/NTUA and especially Dr. X. Trompoukis, Dr. V. Asouti and D. Kapsoulis for always be willing to provide help and knowledge.

Last but not least, I would like to thank my family, my friends and Ifigeneia for every kind of support they provide. Their endless confidence in me always made me feel strong and sure.

National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Parallel CFD & Optimization Unit

Compression of multidimensional flow fields using Incremental Proper Generalized Decomposition & Molecular Dynamics computations and optimization of coarse-grained Ionic Liquid lubricants

Diploma Thesis
by **Kliafas Leonidas**

Supervisors: K. C. Giannakoglou, Professor NTUA,
Dr. Konstantinos Gkagkas, Expert TME

Athens, 2019

Abstract

This diploma thesis comprises two parts. In the first part, the Incremental Proper Generalized Decomposition method is developed and used for compressing the solution fields of multidimensional (including unsteady) fluid mechanics problems. This part was conducted at National Technical University of Athens. In the second part, Ionic Liquids are studied and proposed for improved lubrication. This part was conducted during an 8-month internship at the premises of Toyota Motor Europe (TME) in Zaventem, Belgium.

In the first part, the Incremental Proper Generalized Decomposition (iPGD) approximation method (developed by the PCopt/NTUA) is developed, improved with respect to its current version and programmed. Effort is put on the acceleration of the method and the improvement of accuracy. The presented mathematical formulations are accompanied by numerical examples for validation. Also, 3D and 4D flow problems (one dimension may refer to time) are solved and then the solution fields are compressed by iPGD.

This method was initially developed to be coupled to a gradient-based optimization software. For unsteady problems, the adjoint equations are integrated backwards in time and in order to be solved, the primal solution is necessary at the time step of the integration. Storing the solution field of the primal problem at every time step increases storage requirements. iPGD is developed to support the adjoint solver by compressing the solution field of the primal problem at every time step, at the moment this is computed, thus reducing the store requirements of the optimization.

In the second part, Ionic Liquids (IL) proposed for lubricants are analyzed. Molecular Dynamics computations are carried out to compute IL properties relevant to lubrication, such as the dynamic viscosity of the fluid and the force carrying capability of a liquid confined and compressed by two walls. Then, these properties are optimized by changing the molecular design of liquids. Evolutionary Algorithms are involved to optimize the lubrication properties in which the molecular dynamics algorithms acted as the evaluation tool. A brief presentation of the theoretical background is also included.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης

Ο Σταδιακός Ιδιο-Γενικευμένος Διαχωρισμός για την συμπίεση πολυδιάστατων πεδίων ροής & Υπολογισμοί Μοριακής Δυναμικής και βελτιστοποίηση χονδρόκοκκων μοντέλων Ιονικών Ρευστών για λίπανση

Διπλωματική Εργασία
του **Κλιάφα Λεωνίδα**

Επιβλέποντες: Κυριάκος Χ. Γιαννάκογλου , Καθηγητής ΕΜΠ,
Δρ. Κωνσταντίνος Γκάγκας, TME

Αθήνα, 2019

Περίληψη

Η διπλωματική αυτή εργασία αποτελείται από δύο τμήματα. Το πρώτο αφορά στη μέθοδο του Σταδιακού Ιδιο-Γενικευμένου Διαχωρισμού και στην ικανότητα της να συμπιέζει πολυδιάστατα ροϊκά πεδία (συμπεριλαμβανομένων και των χρονικά μη-μόνιμων). Το μέρος αυτό πραγματοποιήθηκε στο Εθνικό Μετσόβιο Πολυτεχνείο, στην Αθήνα. Το δεύτερο τμήμα σχετίζεται με τη χρήση των Ιονικών Ρευστών ως μέσων λίπανσης βελτιστοποιώντας συγκεκριμένες ιδιότητές τους. Το μέρος αυτό συνοψίζει οκτάμηνη πρακτική εργασία στις εγκαταστάσεις της Toyota Motor Europe (TME), στο Zaventem του Βελγίου.

Στο πρώτο μέρος, μελετάται, βελτιώνεται και προγραμματίζεται η μέθοδος του Σταδιακού Ιδιο-Γενικευμένου Διαχωρισμού (σΙΓΔ). Επαναδιατυπώνεται το μαθηματικό υπόβαθρο της μεθόδου και αναπτύσσονται νέοι αλγόριθμοι με σκοπό το παραγώμενο λογισμικό να επιτυγχάνει προσεγγίσεις μεγαλύτερης ακρίβειας, αλλά κυρίως να μειωθεί δραστικά ο χρόνος υλοποίησης της συμπίεσης, σε σχέση με προηγούμενες εκδοχές της μεθόδου. Η παρουσίαση του μαθηματικού μοντέλου συνοδεύεται από αριθμητικές εφαρμογές για επαλήθευση των παραπάνω ισχυρισμών. Ακόμη, επιλύονται 3Δ και 4Δ προβλήματα ρευστών (η μία διάσταση είναι χρονική) και τα ροϊκά πεδία που προκύπτουν συμπιέζονται από τη μέθοδο του σΙΓΔ.

Η μέθοδος σΙΓΔ αναπτύχθηκε με σκοπό να χρησιμοποιηθεί στη συμπίεση των πεδίων λύσης των εξισώσεων που διέπουν το πρωτεύον πρόβλημα σε μία μέθοδο βελτιστοποίησης βασισμένη στην κλίση της συνάρτησης στόχου. Σε προβλήματα μη-μόνιμων ροών, το συζυγές πρόβλημα ολοκληρώνεται ανάποδα στον χρόνο, απαιτεί όμως σε κάθε χρονική στιγμή το πεδίο λύσης του πρωτεύοντος προβλήματος. Έτσι, αντί να αποθηκευτεί (κοστοβόρα) όλο αυτό το πεδίο, σε κάθε χρονική στιγμή αποθηκεύεται μόνο η τρέχουσα λύση μαζί με την ήδη συμπιεσθείσα προηγούμενη και έτσι μειώνονται οι αποθηκευτικές απαιτήσεις.

Στο δεύτερο μέρος αυτής της διπλωματικής εργασίας, μελετώνται και προτείνονται τα ιονικά ρευστά ως λιπαντικά μέσα. Η δυναμική συνεκτικότητα του ρευστού καθώς και η ικανότητά του να μεταφέρει φορτία από στερεά τοιχώματα υπολογίζονται μέσω κώδικα Υπολογιστικής Μοριακής Δυναμικής. Στη συνέχεια, επιδιώκεται η βελτιστοποίηση των μεγεθών αυτών με την αλλαγή της μοριακής δομής του ρευστού μέσω εξελικτικών αλγορίθμων. Προηγείται μια σύντομη παρουσίαση του θεωρητικού υπόβαθρου της Υπολογιστικής Μοριακής Δυναμικής.

Acronyms

NTUA	National Technical University of Athens
PCopt	Parallel CFD & Optimization unit
TME	Toyota Motor Europe

CFD	Computational Fluid Dynamics
PGD	Proper Generalized Decomposition
iPGD	Incremental Proper Generalized Decomposition
PDE	Partial Differential Equation
ODE	Ordinary Differential Equations

MD	Molecular Dynamics
NVE	constant Number, Volume, Energy
NVT	constant Number, Volume, Temperature
NPT	constant Number, Pressure, Temperature
EA	Evolutionary Algorithms

Contents

Part I

Compression of multidimensional flow fields using Incremental Proper Generalized Decomposition

Chapter 1

Introduction

1.1 Purpose

In gradient-based optimizations for unsteady flow problems, adjoint equations are integrated backwards in time and, during this phase, the solution field of the primal problem is needed at each time step. At a first glance, two extreme options exist: either fully store the solution time-series of the primal problem for use while solving the adjoint equations or to store nothing and recompute it at every time step. The first option highly increases the storage requirements and for large-scale unsteady flows becomes prohibitive. The second one has zero storage requirements but is totally inefficient since recomputing increases the computational time a lot.

In order to avoid both extreme options, a nice idea is to use an approximative compression technique after solving the primal problem and to store only the approximation of the solution field. **Proper Generalized Decomposition (PGD)** is a high-accurate approximation method to compress a field that is already available. So, full storage capacity is required until PGD takes over. But if this storage capacity was available, there is, practically, no reason for compressing the field. For this reason, the PCopt/NTUA developed the **Incremental PGD (iPGD)** that compresses unsteady fields incrementally by handling each new instantaneous solution and the previously compressed (through iPGD) time-series. The solution time-series of the primal problem is enriched by a time-marching technique, so after an instantaneous field is computed, the iPGD compresses this field, stores the approximation and deletes the original.

The iPGD was first presented and extensively studied in the Diploma Thesis of Vasilis Papageorgiou (ref. [2], [3]). There, it was presented how it can compress 1D and 2D unsteady fields, but also its low speed was highlighted. In this Diploma Thesis, iPGD is extented for compressing 3D unsteady fields, its accuracy is improved and, the most important, it is noticeably accelerated.

1.2 Proper Generalized Decomposition

The main idea of PGD (and iPGD) is that any function $u = u\left(x_1, x_2, ..., x_q\right)$ of q independent variables x_i, $i = 1, 2, ..., q$, can be approximated by a M-term sum of products of q 1D functions as:

$$u\left(x_1, x_2, ..., x_q\right) \approx \sum_{\mu=1}^{M} {X_1}^{\mu}(x_1){X_2}^{\mu}(x_2)...{X_q}^{\mu}(x_q) \tag{1.1}$$

The fact that a multidimensional field is approximated by 1D functions is called decomposition. Every 1D function ${X_i}^{\mu}(x_i)$ is called basis function or (just) basis.

A field like u, in order to be fully defined, assuming that every independent variable x_i is discretized in n_i terms, demands $Q_1 = n_1 n_2 ... n_q$ values. On the contrary, the PGD of the same field demands only $Q_2 = M\left(n_1 + n_2 + ... + n_q\right)$ values. In the next chapters, Q_1 and Q_2 are compared to highlight savings in storage capacity.

The PGD method is used for two main tasks. Either for approximating unknown fields that are the solution of a PDE, or for the post-compression of already defined fields in order to save storage. The second case is presented in section 2.1 in detail.

In the first case, PGD is used as a solver of non-linear equations, due to the fact that in real applications the discretization of the computational space demands considerable storage. In short, the requested field $u\left(x_1, x_2, ..., x_q\right)$ in a PDE is replaced by the PGD approximation $\sum\limits_{\mu=1}^{M} {X_1}^{\mu}(x_1){X_2}^{\mu}(x_2)...{X_q}^{\mu}(x_q)$, and the equation is solved separately for each 1D bases ${X_i}^{\mu}(x_i)$. Thus, the difficult procedure of solving one PDE is reduced to the easy procedure of solving q ODE. Instead of the expensive solution, PGD produces a much cheaper (and enough accurate) approximation. This case is not part of this diploma thesis and will not be discussed further.

The Incremental PGD of an unsteady field is:

$$u\left(x, t\right) \approx \sum_{\mu=1}^{M} X^{\mu}(x) T^{\mu}(t) \tag{1.2}$$

where x is the space (1D/2D/3D) and t is the time. The compression of this field is performed incrementally; the bases are computed (renewed) for every certain moment just after the field at this moment is solved. This method is very efficient especially for fields that result from a time-marching computational technique.

Following chapters include a detailed presentation of the PGD for post-compression of 2D, 3D and 4D fields (section 2.1), the mathematical formulation of iPGD for the compression of 2D, 3D and 4D fields accompanied by numerical examples (section 2.2), programming tips for accelerating the algorithm (chapter 3) and applications of iPGD on unsteady fluid mechanics problems(chapter 4).

Chapter 2

The Proper Generalized Decomposition

In this chapter, the standard PGD is presented in order to prepare the reader for its Incremental variant. After that, there is a detailed presentation of the iPGD for multidimensional approximations followed by numerical examples.

2.1 The standard PGD

During this section, PGD for compressing 2D, 3D and 4D fields is presented in detail. In real applications these dimensions may refer to space and/or time.

2.1.1 Post-compression of a 2D field

Having for compression the 2D field $u(\xi, \eta)$. The ξ and η are the space/parametric or time coordinates of nodes, where the discretized field is defined. The field is approximated by a finite sum of M terms. Every μ-th term is the product of the $\phi^{\mu}(\xi)$ and $\theta^{\mu}(\eta)$ bases. So, the approximation becomes:

$$u(\xi, \eta) \approx \sum_{\mu=1}^{M} \phi^{\mu}(\xi)\theta^{\mu}(\eta) \tag{2.1}$$

Initially, the sum of the equation (2.1) is consisted only of one term and is enriched gradually to the appropriate M by consecutive steps. Concerning M, it can either be defined a priori or result during the procedure. The second case is when the sum is enriched until the approximation residual (or its variation as M is increasing) is less than a predefined thresshold-value.

Termination Criteria of the Enrichment Process

In most of the cases, knowing the appropriate value of M beforehand is impossible, so the contribution of the new-added term must be compared with a predefined threshold-value.

This value can either be the importance of the new-added term to all previous, as:

$$\mathcal{E}(m) = \frac{\left\| \int_\xi \phi^m(\xi) d\xi \int_\eta \theta^m(\eta) d\eta \right\|}{\left\| \sum_{\mu=1}^{m-1} \int_\xi \phi^\mu(\xi) d\xi \int_\eta \theta^\mu(\eta) d\eta \right\|} \tag{2.2}$$

or the difference of the last two terms divided by the first term, that is the greater (see ref. [2]), as:

$$\mathcal{E}(m) = \frac{\left\| \int_\xi \phi^m(\xi) d\xi \int_\eta \theta^m(\eta) d\eta - \int_\xi \phi^{m-1}(\xi) d\xi \int_\eta \theta^{m-1}(\eta) d\eta \right\|}{\left\| \int_\xi \phi^1(\xi) d\xi \int_\eta \theta^1(\eta) d\eta \right\|} \tag{2.3}$$

The user is free to choose the most efficient termination criterion for each case.

Progressive Construction of the Separated Representation

The terms of the approximation sum are computed gradually, thus, in order to highlight the construction process, it will be assumed that the $m-1$ (where $m \geq 1$) terms are already known and the bases of the m-th step ($\phi^m(\xi)$ and $\theta^m(\eta)$) are to be computed. The approximation of m terms reads:

$$u^m_{\text{PGD}}(\xi, \eta) \;=\; u^{m-1}_{\text{PGD}}(\xi, \eta) + \phi^m(\xi)\, \theta^m(\eta) \tag{2.4}$$

where $u^{m-1}_{\text{PGD}}(\xi, \eta) \;=\; \sum_{\mu=1}^{m-1} \phi^\mu(\xi)\, \theta^\mu(\eta)$ is the known up-to-now approximation.

The two bases are coupled (they are in product) so an iterative process is necessary to solve them, after replacing $u^m_{\text{PGD}}(\xi, \eta)$ by the known $u(\xi, \eta)$. This internal iterative procedure has to do with computing the bases $\phi^m(\xi)$ and $\theta^m(\eta)$ and must not be confused with the iterative construction process (cycle).

An iterative technique like the Alternating Direction Strategy is a rational choice for the above computation. In this technique, for every step p of the internal iterative process the function $\phi^m_p(\xi)$ is computed by the already known function $\theta^m_{p-1}(\eta)$ and then the $\theta^m_p(\eta)$ is computed by $\phi^m_p(\xi)$. The whole process starts after giving to the function $\theta^m_0(\eta)$ an arbitrary initial value.

There are two ways to terminate these iterations. It is up to user's experience to choose between them. The first way is to predefine the maximum number of iterations. By doing that, the user can control the time needed but there is the risk the two functions are not (fully) converged or converged very quickly and unnecessary iterations performed. The second way is to stop the iterations when a termination criterion is met. Such a criterion checks if the change of the value of a basis every last two iterations is smaller than a threshold-value. This criterion has the form:

$$\int_{\xi} \left(\phi_p^m(\xi) - \phi_{p-1}^m(\xi)\right)^2 d\xi < \epsilon \tag{2.5}$$

and

$$\int_{\eta} \left(\theta_p^m(\eta) - \theta_{p-1}^m(\eta)\right)^2 d\eta < \epsilon \tag{2.6}$$

where ϵ is a predefined positive threshold-value. By this way, the basis functions will converge but this may make the algorithm very slow. More often a combination of these two ways is used; a criterion is imposed and the maximum number of iterations is predefined.

When the internal iterations are over, the values of $\phi_p^m(\xi)$ and $\theta_p^m(\eta)$ are stored as $\phi^m(\xi)$ and $\theta^m(\eta)$ and the construction process is going on with the next term $(m+1)$. The construction process ends when M terms are added to the approximation sum.

Alternating Direction Strategy

In order to present the Alternating Direction Strategy in more detail it is assumed that the process is in the p-th internal iteration of the m-th construction cycle. This process is performed in two steps. In each step one basis is computed.

- **Step 1st:** Computation of $\phi_p^m(\xi)$

The approximation of the field u so far reads:

$$u(\xi,\eta) \approx \sum_{\mu=1}^{m-1} \phi^{\mu}(\xi)\,\theta^{\mu}(\eta) + \phi_p^m(\xi)\,\theta_{p-1}^m(\eta) \tag{2.7}$$

Remember that the field u is available, the compressed field $\sum_{\mu=1}^{m-1}\phi^{\mu}(\xi)\,\theta^{\mu}(\eta)$ has been found and the basis θ_{p-1}^m was computed in the previous step of the internal iterative procedure, so the only unknown quantity in equation (2.7) is the value of ϕ_p^m. Galerkin's projection method is used in order to compute this value.

According to that, all terms of equation (2.7) are transferred to the left-hand-side, they are multiplied by Galerkin weight function u^* and integrated in both directions

(ξ and η). If $\theta^m_{p-1}(\eta)$ is used as weight function, equation (2.7) becomes:

$$\int_\xi\int_\eta \theta^m_{p-1}(\eta)\, u(\xi,\eta) d\eta d\xi - \int_\xi\int_\eta \theta^m_{p-1}(\eta) \sum_{\mu=1}^{m-1} \phi^\mu(\xi)\,\theta^\mu(\eta) d\eta d\xi - \int_\xi\int_\eta (\theta^m_{p-1}(\eta))^2 \phi^m_p(\xi) d\eta d\xi = 0 \tag{2.8}$$

Because most of the terms of equation (2.8) are one-dimensional and all functions of η are known (from previous step $p-1$), the integrals in the η-direction can be rearranged to contain only functions of η, while the integrals in the ξ-direction to contain all the left-hand-side of the equation. After that, equation (2.8) becomes:

$$\int_\xi \left(\int_\eta \theta^m_{p-1}(\eta)\, u(\xi,\eta) d\eta - \sum_{\mu=1}^{m-1} \phi^\mu(\xi) \int_\eta \theta^m_{p-1}(\eta)\theta^\mu(\eta) d\eta - \phi^m_p(\xi) \int_\eta (\theta^m_{p-1}(\eta))^2 d\eta \right) d\xi = 0 \tag{2.9}$$

The integral in the ξ-direction can be deleted because the integrated quantity was initially equal to zero (see eq. 2.7, after all terms are moved to the left-hand-side). The only unknown quantity in this equation is $\phi^m_p(\xi)$ and can be computed as:

$$\phi^m_p(\xi) = \frac{\int_\eta u(\xi,\eta)\theta^m_{p-1}(\eta) d\eta - \sum_{\mu=1}^{m-1} \phi^\mu(\xi) \int_\eta \theta^m_{p-1}(\eta)\theta^\mu(\eta) d\eta}{\int_\eta (\theta^m_{p-1}(\eta))^2 d\eta} \tag{2.10}$$

- **Step 2nd:** Computation of $\theta^m_p(\eta)$

After computing $\phi^m_p(\xi)$, the approximation of the field can be written:

$$u(\xi,\eta) \approx \sum_{\mu=1}^{m-1} \phi^\mu(\xi)\,\theta^\mu(\eta) + \phi^m_p(\xi)\,\theta^m_p(\eta) \tag{2.11}$$

and the unknown quantity now is $\theta^m_p(\eta)$. Same as before, performing Galerkin projection method for the equation (2.11), with ϕ^m_p as weight function and after all terms are moved to the left-hand-side and integrated, equation (2.11) becomes:

$$\int_\eta\int_\xi \phi^m_p(\xi)\, u(\xi,\eta) d\xi d\eta - \int_\eta\int_\xi \phi^m_p(\xi) \sum_{\mu=1}^{m-1} \theta^\mu(\eta)\, \phi^\mu(\xi) d\xi d\eta - \int_\eta\int_\xi (\phi^m_p(\xi))^2 \theta^m_p(\eta) d\xi d\eta = 0 \tag{2.12}$$

Corresponding to the previous step, equation (2.12) can be rearranged and the integrals in the ξ-direction can be computed separately from the η-direction because all functions of ξ are known from the previous step. The integral on the η-direction that contains all the quantities of the left-hand-side can be deleted because the integrated quantity was initially equal to zero. The remaining equation is solved for $\theta_p^m(\eta)$ which is the only unknown function. Thus, equation (2.12) becomes:

$$\theta_p^m(\eta) = \frac{\int_\xi u(\xi,\eta)\phi_p^m(\xi)d\xi - \sum_{\mu=1}^{m-1}\theta^\mu(\eta)\int_\xi \phi_p^m(\xi)\phi^\mu(\xi)d\xi}{\int_\xi (\phi_p^m(\xi))^2 d\xi} \tag{2.13}$$

The above process is repeated until the two bases converge and then the sum of equation (2.4) is enriched with one more term increasing the approximation accuracy.

A flowchart of the above algorithm is presented in figure 2.1 in order to provide the reader with a better understanding of the PGD compression. In this flowchart, the rounded rectangle, the rectangle, the trapezium and the rhombus refers to start/stop, process, input/output value and decision, respectively.

This subsection is based on ref. [1] and [2]. The PGD method also extends to the compression of 3D and 4D fields. These dimensions may be spatial and/or temporal. Only the equations that compute the bases are listed below since the method is implemented in the exact same way.

2.1.2 Post-compression of a 3D field

Let the 3D field to be compressed is $u(\alpha,\beta,\gamma)$. One possible case is that α, β and γ are all spatial variables. They correspond, for example, to the three directions of a structured grid on the nodes of which the field u is defined. Alternatively, α and β could refer to space (e.g. a 2D structured grid), with γ representing time. In this case, PGD method compresses the solution of a 2D unsteady flow problem. The approximation of a field like this is:

$$u(\alpha,\beta,\gamma) \approx \sum_{\mu=1}^{M} A^\mu(\alpha)B^\mu(\beta)\Gamma^\mu(\gamma) \tag{2.14}$$

Same as the 2D field, initially the sum (2.14) is consisted only of one term and gradually is enriched. Suppose that the enrichment process is at m-th step and the internal process at iteration p, the basis functions $A^m(\alpha)$, $B^m(\beta)$ and $\Gamma^m(\gamma)$ are computed by the following equations:

$$A_p^m = \frac{\int_\gamma\int_\beta uB_{p-1}^m\Gamma_{p-1}^m d\beta d\gamma - \sum_{\mu=1}^{m-1} A^\mu \int_\beta B_{p-1}^m B^\mu d\beta \int_\gamma \Gamma_{p-1}^m \Gamma^\mu d\gamma}{\int_\beta \left(B_{p-1}^m\right)^2 d\beta \int_\gamma \left(\Gamma_{p-1}^m\right)^2 d\gamma} \tag{2.15}$$

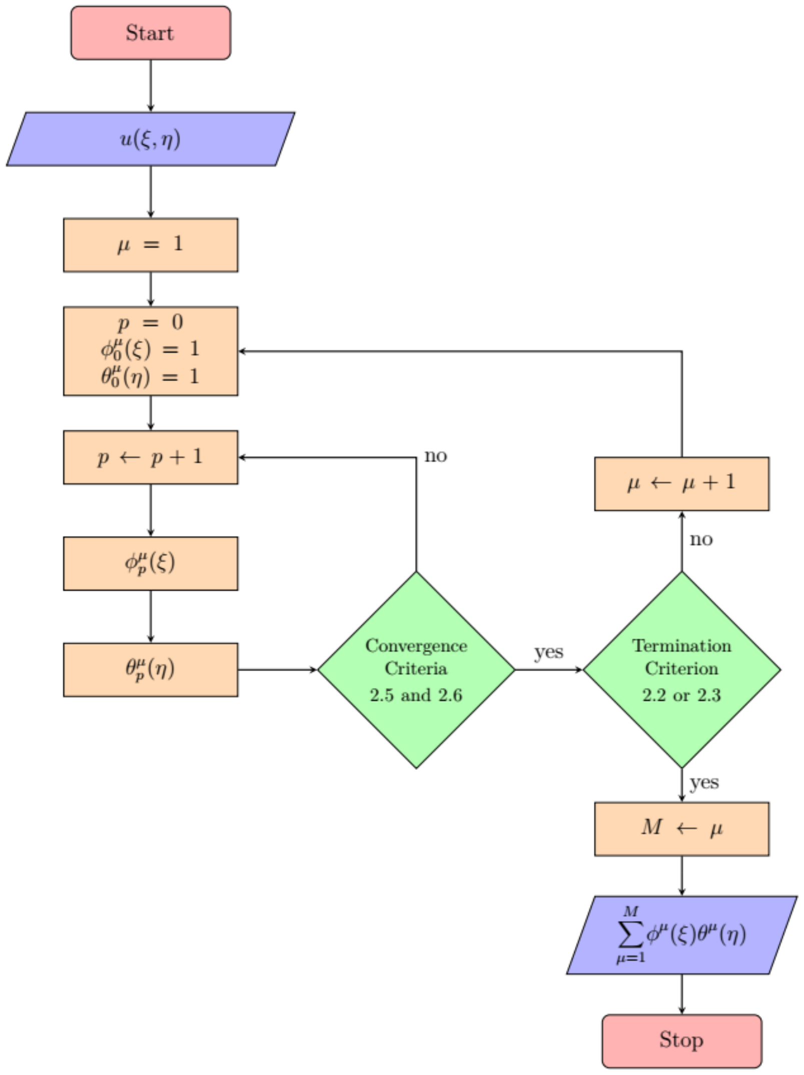


Figure 2.1: *Flowchart of the PGD algorithm.*

$$B_p^m = \frac{\int_\gamma\int_\alpha uA_{p-1}^m\Gamma_{p-1}^m d\alpha d\gamma - \sum_{\mu=1}^{m-1} B^\mu \int_\alpha A_{p-1}^m A^\mu d\alpha \int_\gamma \Gamma_{p-1}^m \Gamma^\mu d\gamma}{\int_\alpha \left(A_{p-1}^m\right)^2 d\alpha \int_\gamma \left(\Gamma_{p-1}^m\right)^2 d\gamma} \tag{2.16}$$

$$\Gamma_p^m = \frac{\int_\beta\int_\alpha uA_{p-1}^m B_{p-1}^m d\alpha d\beta - \sum_{\mu=1}^{m-1} \Gamma^\mu \int_\alpha A_{p-1}^m A^\mu d\alpha \int_\beta B_{p-1}^m B^\mu d\beta}{\int_\alpha \left(A_{p-1}^m\right)^2 d\alpha \int_\beta \left(B_{p-1}^m\right)^2 d\beta} \tag{2.17}$$

In the above relations, for brevity, functions u, A, B and Γ denote $u(\alpha,\beta,\gamma)$, $A(\alpha)$, $B(\beta)$ and $\Gamma(\gamma)$, respectively. The meaning of the indicators does not change.

2.1.3 Post-compression of a 4D field

Having for compression the 4D field $u(\alpha,\beta,\gamma,\zeta)$. In fluid mechanics problems the variables α,β and γ are the three dimensions of space (e.g. the directions of a structured grid), while ζ is the time variable. The PGD of this field is:

$$u(\alpha,\beta,\gamma,\zeta) \approx \sum_{\mu=1}^{M} A^{\mu}(\alpha)B^{\mu}(\beta)\Gamma^{\mu}(\gamma)Z^{\mu}(\zeta) \tag{2.18}$$

Same as above, initially the sum (2.18) is consisted only of one term and gradually is enriched. Suppose that this process is at m-th step and p-th internal iteration , the bases $A^m(\alpha)$, $B^m(\beta)$, $\Gamma^m(\gamma)$ and $Z^m(\zeta)$ are computed by the following equations:

$$A_p^m = \frac{\int_\zeta\int_\gamma\int_\beta uB_{p-1}^m\Gamma_{p-1}^m Z_{p-1}^m d\beta d\gamma d\zeta - \sum_{\mu=1}^{m-1} A^\mu \int_\beta B_{p-1}^m B^\mu d\beta \int_\gamma \Gamma_{p-1}^m\Gamma^\mu d\gamma \int_\zeta Z_{p-1}^m Z^\mu d\zeta}{\int_\beta \left(B_{p-1}^m\right)^2 d\beta \int_\gamma \left(\Gamma_{p-1}^m\right)^2 d\gamma \int_\zeta \left(Z_{p-1}^m\right)^2 d\zeta} \tag{2.19}$$

$$B_p^m = \frac{\int_\zeta\int_\gamma\int_\alpha uA_{p-1}^m\Gamma_{p-1}^m Z_{p-1}^m d\alpha d\gamma d\zeta - \sum_{\mu=1}^{m-1} B^\mu \int_\alpha A_{p-1}^m A^\mu d\alpha \int_\gamma \Gamma_{p-1}^m\Gamma^\mu d\gamma \int_\zeta Z_{p-1}^m Z^\mu d\zeta}{\int_\alpha \left(A_{p-1}^m\right)^2 d\alpha \int_\gamma \left(\Gamma_{p-1}^m\right)^2 d\gamma \int_\zeta \left(Z_{p-1}^m\right)^2 d\zeta} \tag{2.20}$$

$$\Gamma_p^m = \frac{\int_\zeta\int_\beta\int_\alpha uA_{p-1}^m B_{p-1}^m Z_{p-1}^m d\alpha d\beta d\zeta - \sum_{\mu=1}^{m-1} \Gamma^\mu \int_\alpha A_{p-1}^m A^\mu d\alpha \int_\beta B_{p-1}^m B^\mu d\beta \int_\zeta Z_{p-1}^m Z^\mu d\zeta}{\int_\alpha \left(A_{p-1}^m\right)^2 d\alpha \int_\beta \left(B_{p-1}^m\right)^2 d\beta \int_\zeta \left(Z_{p-1}^m\right)^2 d\zeta} \tag{2.21}$$

$$Z_p^m = \frac{\int_\gamma\int_\beta\int_\alpha uA_{p-1}^m B_{p-1}^m \Gamma_{p-1}^m d\alpha d\beta d\gamma - \sum_{\mu=1}^{m-1} Z^\mu \int_\alpha A_{p-1}^m A^\mu d\alpha \int_\beta B_{p-1}^m B^\mu d\beta \int_\gamma \Gamma_{p-1}^m \Gamma^\mu d\gamma}{\int_\alpha \left(A_{p-1}^m\right)^2 d\alpha \int_\beta \left(B_{p-1}^m\right)^2 d\beta \int_\gamma \left(\Gamma_{p-1}^m\right)^2 d\gamma} \tag{2.22}$$

where, the functions u, A, B, Γ and Z denote $u(\alpha,\beta,\gamma,\zeta)$, $A(\alpha)$, $B(\beta)$, $\Gamma(\gamma)$ and $Z(\zeta)$ respectively.

2.1.4 Verification of standard PGD method treating it as a minimization problem

The previous relations for computing the basis functions can be proved by an alternative procedure. The following is a demonstration of this approximation only for the case of 2D fields (just for space-saving reasons), emphasizing that the procedure followed is exactly the same for fields of more than two dimensions.

Suppose knowing the 2D field $u(\xi, \eta)$. It is approximated as:

$$u(\xi,\eta) \approx \sum_{\mu=1}^{M} \phi^{\mu}(\xi)\theta^{\mu}(\eta) \tag{2.23}$$

Initially, the sum of equation (2.23) contains only one term and is gradually enriched through successive steps to the appropriate M. Suppose the enrichment process has completed $m-1$ steps and $\phi^m(\xi)$ and $\theta^m(\eta)$ are to be computed. To approximate $u(\xi, \eta)$ as precisely as possible, the approximation error (deviation) must be minimized. Such a deviation between the real field and the approximation is defined as:

$$E = \frac{1}{2}\int_{\xi}\int_{\eta}\left[\sum_{\mu=1}^{m} \phi^{\mu}(\xi)\theta^{\mu}(\eta) - u(\xi,\eta)\right]^2 d\eta d\xi \tag{2.24}$$

Quantity E from equation (2.24) will be minimum if its partial derivatives (with respect to the functions $\phi^m(\xi)$ and $\theta^m(\eta)$) are equal to zero. That is:

$$\frac{\partial E}{\partial \phi^m(\xi)} = 0 \Leftrightarrow$$

$$\int_{\eta}\left[\sum_{\mu=1}^{m} \phi^{\mu}(\xi)\theta^{\mu}(\eta) - u(\xi,\eta)\right]\theta^m(\eta)d\eta = 0 \Leftrightarrow$$

$$\phi^m(\xi)\int_{\eta}\theta^m(\eta)\theta^m(\eta)d\eta + \sum_{\mu=1}^{m-1}\phi^{\mu}(\xi)\int_{\eta}\theta^{\mu}(\eta)\theta^m(\eta)d\eta - \int_{\eta}u(\xi,\eta)\theta^m(\eta)d\eta = 0 \Leftrightarrow$$

$$\phi^m(\xi) = \frac{\displaystyle\int_{\eta}u(\xi,\eta)\theta^m(\eta)d\eta - \sum_{\mu=1}^{m-1}\phi^{\mu}(\xi)\int_{\eta}\theta^m(\eta)\theta^{\mu}(\eta)d\eta}{\displaystyle\int_{\eta}(\theta^m(\eta))^2 d\eta} \tag{2.25}$$

and

$$\frac{\partial E}{\partial \theta^m(\eta)} = 0 \Leftrightarrow$$

$$\int_{\xi}\left[\sum_{\mu=1}^{m} \phi^{\mu}(\xi)\theta^{\mu}(\eta) - u(\xi,\eta)\right]\phi^m(\xi)d\xi = 0 \Leftrightarrow$$

$$\theta^m(\eta)\int_\xi \phi^m(\xi)\phi^m(\xi)d\xi + \sum_{\mu=1}^{m-1}\theta^\mu(\eta)\int_\xi \phi^\mu(\xi)\phi^m(\xi)d\xi - \int_\xi u(\xi,\eta)\phi^m(\xi)d\xi = 0 \Leftrightarrow$$

$$\theta^m(\eta) = \frac{\displaystyle\int_\xi u(\xi,\eta)\phi^m(\xi)d\xi - \sum_{\mu=1}^{m-1}\theta^\mu(\eta)\int_\xi \phi^m(\xi)\phi^\mu(\xi)d\xi}{\displaystyle\int_\xi (\phi^m(\xi))^2 d\xi} \tag{2.26}$$

Indeed, the two couples (2.10), (2.13) and (2.25), (2.26) are the same. Also here, the requested bases are coupled, so an iterative process is necessary to solve them.

2.2 The Incremental PGD

The PGD method, presented above, can approximate multidimensional fields with high precision, significantly reducing storage requirements. The major disadvantage, however, is that the entire field must be available. Solution to this problem is the Incremental PGD (iPGD) proposed by the PCopt/NTUA and initially presented by Vasilis Papageorgiou (ref. [2] and [3]). This approximative compression method can compress unsteady fields in an incremental way - in each construction-step it handles the field of only one time step and not the whole time-series. The following is a detailed presentation of the method for 2D, 3D and 4D fields in space-time.

2.2.1 Compression of an unsteady 1D field

Having for compression the unsteady field $U(x,t)$. This will be approximated by a M-terms sum of products of the 1D bases $X(x)$ and $T(t)$ as:

$$U(x,t) \approx \sum_{\mu=1}^{M} X^\mu(x)T^\mu(t) \tag{2.27}$$

It must be emphasized that in iPGD M can only be predefined by the user. The spatial coordinate is discretized in I nodes ($i = 1,..,I$), while the total time is divided into discrete steps ($k = 1,..,K,..$), where by K denotes the current timestep. After discretization, equation (2.27) for every node i and every time step k is:

$$U_{i,k} \approx \sum_{\mu=1}^{M} X_i^\mu\ T_k^\mu \tag{2.28}$$

The approximation accuracy can be quantified by the total approximation error:

$$E = \frac{1}{2}\sum_{i=1}^{I}\sum_{k=1}^{K+1}\left[\sum_{\mu=1}^{M} X_i^\mu\ T_k^\mu - U_{i,k}\right]^2 \tag{2.29}$$

The smaller the error, the better the approximation. Up to this point the analysis is not different from the proof of the standard PGD treated as a minimization problem, where the above error is minimized by setting its derivatives to zero and, then, the basis functions $X(x)$ and $T(t)$ are computed using the original field $U(x,t)$.

From this point on, iPGD differs from the standard PGD. The fact that the field only at the new time step $U_{i,K+1}$ is known (the field at all previous moments is not stored and at all next moments is not obtained yet) makes it necessary to redefine the approximation error. Instead of the field $U_{i,k}$ $(k = 1,..K)$, this error is handling the M couples of bases computed in previous time step and denoted by $\widetilde{X}$ and $\widetilde{T}$.

At time step $K+1$, every basis needs to be redefined, while new bases T_{K+1}^m (where $m = 1,...,M$) has to be added in order to take into account new change in time. For that, the total approximation error from equation (2.29) is redefined as:

$$E = \frac{1}{2}\sum_{i=1}^{I}\left[\sum_{\mu=1}^{m} X_i^\mu T_{K+1}^\mu - U_{i,K+1}\right]^2 + \frac{w}{2}\sum_{i=1}^{I}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{m} X_i^\mu T_k^\mu - \sum_{\mu=1}^{M}\widetilde{X}_i^\mu \widetilde{T}_k^\mu\right]^2 \tag{2.30}$$

where $U_{i,k}^{iPGD} = \sum_{\mu=1}^{M}\widetilde{X}_i^\mu\widetilde{T}_k^\mu$, $k = 1,..,K$, is the previous approximation of $U_{i,k}$.

Now, with the formulation (2.30), the total error takes into account both the solution of the next time step and all the previous ones by their up-to-now approximation. w is a user-defined weight factor that gives a relative "flexibility" to the extent that the bases will change from one time step to the next. Its appropriate value depends largely on how the morphology of the field changes through time.

The need to minimize the error (2.30) indicates the procedure to follow. In order to calculate the unknown bases X_i^m, T_k^m and T_{K+1}^m, the partial derivatives with respect to each of these three bases must be set to zero. That is:

$$\frac{\partial E}{\partial X_i^m} = 0 \Leftrightarrow$$

$$\left[\sum_{\mu=1}^{m} X_i^\mu T_{K+1}^\mu - U_{i,K+1}\right] T_{K+1}^m + w\sum_{k=1}^{K}\left[\sum_{\mu=1}^{m} X_i^\mu T_k^\mu - \sum_{\mu=1}^{M}\widetilde{X}_i^\mu\widetilde{T}_k^\mu\right] T_k^m = 0 \Leftrightarrow$$

$$X_i^m = \frac{\left[U_{i,K+1} - \sum\limits_{\mu=1}^{m-1} X_i^\mu T_{K+1}^\mu\right] T_{K+1}^m + w\sum\limits_{k=1}^{K}\left[\sum\limits_{\mu=1}^{M}\widetilde{X}_i^\mu \widetilde{T}_k^\mu - \sum\limits_{\mu=1}^{m-1} X_i^\mu T_k^\mu\right] T_k^m}{\left(T_{K+1}^m\right)^2 + w\sum\limits_{k=1}^{K}\left(T_k^m\right)^2} \tag{2.31}$$

$$\frac{\partial E}{\partial T_k^m} = 0 \Leftrightarrow$$

$$w\sum_{i=1}^{I}\left[\sum_{\mu=1}^{m} X_i^\mu T_k^\mu - \sum_{\mu=1}^{M}\widetilde{X}_i^\mu \widetilde{T}_k^\mu\right] X_i^m = 0 \Leftrightarrow$$

$$T_k^m = \frac{\sum\limits_{i=1}^{I}\left[\sum\limits_{\mu=1}^{M}\widetilde{X}_i^\mu \widetilde{T}_k^\mu - \sum\limits_{\mu=1}^{m-1} X_i^\mu T_k^\mu\right] X_i^m}{\sum\limits_{i=1}^{I}\left(X_i^m\right)^2} \tag{2.32}$$

$$\frac{\partial E}{\partial T_{K+1}^m} = 0 \Leftrightarrow$$

$$\sum_{i=1}^{I}\left[\sum_{\mu=1}^{m} X_i^\mu T_{K+1}^\mu - U_{i,K+1}\right] X_i^m = 0 \Leftrightarrow$$

$$T_{K+1}^m = \frac{\sum\limits_{i=1}^{I}\left[U_{i,K+1} - \sum\limits_{\mu=1}^{m-1} X_i^\mu T_{K+1}^\mu\right] X_i^m}{\sum\limits_{i=1}^{I}\left(X_i^m\right)^2} \tag{2.33}$$

The sums of equations (2.31)-(2.33) have to be rearranged to accelerate the computation. Some identities of summation are used to rearrange them. These are:

$$\sum_i (CA_i) = C\sum_i (A_i) \tag{2.34a}$$

$$\sum_i \left(A_i^1 + A_i^2 + ... + A_i^\nu\right) = \sum_i \left(A_i^1\right) + \sum_i \left(A_i^2\right) + ... + \sum_i \left(A_i^\nu\right) \tag{2.34b}$$

$$\sum_{i_1}\sum_{i_2}...\sum_{i_\nu}\left(A_{i_1}^1 A_{i_2}^2 ... A_{i_\nu}^\nu\right) = \sum_{i_1}\left(A_{i_1}^1\right)\sum_{i_2}\left(A_{i_2}^2\right)...\sum_{i_\nu}\left(A_{i_\nu}^\nu\right) \tag{2.34c}$$

where C is a constant and A, A^j, for $j = 1, ..., \nu$, are functions each depending on the variable in the position of the lower pointer. So, equations (2.31)-(2.33) become:

$$X_i^m = \frac{U_{i,K+1}T_{K+1}^m + w\sum_{\mu=1}^{M}\left[\widetilde{X}_i^\mu\sum_{k=1}^{K}\widetilde{T}_k^\mu T_k^m\right] - \sum_{\mu=1}^{m-1}X_i^\mu\left[T_{K+1}^\mu T_{K+1}^m + \sum_{k=1}^{K}T_k^\mu T_k^m\right]}{\left(T_{K+1}^m\right)^2 + w\sum_{k=1}^{K}\left(T_k^m\right)^2} \tag{2.35}$$

$$T_{K+1}^m = \frac{\sum_{i=1}^{I}U_{i,K+1}X_i^m - \sum_{\mu=1}^{m-1}\left[T_{K+1}^\mu\sum_{i=1}^{I}X_i^\mu X_i^m\right]}{\sum_{i=1}^{I}\left(X_i^m\right)^2} \tag{2.36}$$

$$T_k^m = \frac{\sum_{\mu=1}^{M}\left[\widetilde{T}_k^\mu\sum_{i=1}^{I}\widetilde{X}_i^\mu X_i^m\right] - \sum_{\mu=1}^{m-1}\left[T_k^\mu\sum_{i=1}^{I}X_i^\mu X_i^m\right]}{\sum_{i=1}^{I}\left(X_i^m\right)^2} \tag{2.37}$$

In equations (2.35)-(2.37), the bases appear in coupled form, so an iterative internal procedure is required. In order to start it, the bases could be initialized as:

$$X_i^\mu = U_{i,1}, \qquad \mu = 1,..,M \tag{2.38a}$$

$$T_1^\mu = \begin{cases} 1, & \mu = 1 \\ 0, & \mu = 2,..,M \end{cases} \tag{2.38b}$$

or:

$$T_1^\mu = 1, \qquad \mu = 1,..,M \tag{2.39a}$$

$$X_i^\mu = \begin{cases} U_{i,1}, & \mu = 1 \\ 0, & \mu = 2,..,M \end{cases} \tag{2.39b}$$

In both cases, it is ensured that, at the initial time step, the field is reproduced precisely. For each new time step, the bases X_i^m and T_k^m are redefined while the basis for the next time step T_{K+1}^m has to be computed. The whole procedure is over when even the last time step is compressed.

A flowchart is used for the illustration of this algorithm, but in order to be more generic it is presented in the next subsection for the 2D unsteady field.

In order to prove and demonstrate the reliability of iPGD, the approximation must be compared with the original field. This is achieved through the total relative error:

$$\mathcal{E}(M) = \frac{\sum_{i=1}^{I}\sum_{k=1}^{K}\left(U_{i,k} - \sum_{\mu=1}^{M} X_i^\mu T_k^\mu\right)^2}{\sum_{i=1}^{I}\sum_{k=1}^{K} U_{i,k}^2} \tag{2.40}$$

It is obvious that the value of this error depends directly on the number of terms M of the approximation sum. As mentioned, storing the original field in real applications is non-affordable and some times even impossible (that is why iPGD was developed), but in the context of this work, however, the storage of the original field is necessary in order to compute the total error and to highlight the possibilities of iPGD.

Numerical Example

In this example, iPGD is used to compress the following 1D unsteady field:

$$u(x,t) = \left[\sin\left(x^3 - 1\right) - 1\right]^2 \log\left(tx^2 + \frac{\pi}{2}\right)\sin\left(\frac{4\pi}{T_{int}} t\right) \tag{2.41}$$

The total length, related with the spatial variable x is equal to 1 and divided in $I = 81$ nodes. The studied time interval is $T_{int} = 0.01$ and divided in $K = 200$ time steps. The presence of t within the logarithm indicates that (2.41) is non-periodic. For a periodic field, the time interval could well be chosen equal to one period.

Figure 2.2 shows the relation between the approximation error 2.40 and the value of M. As M is increasing the error is decreasing up to a minimum. After that, using more terms does not increase the accuracy but wastes storage capacity.

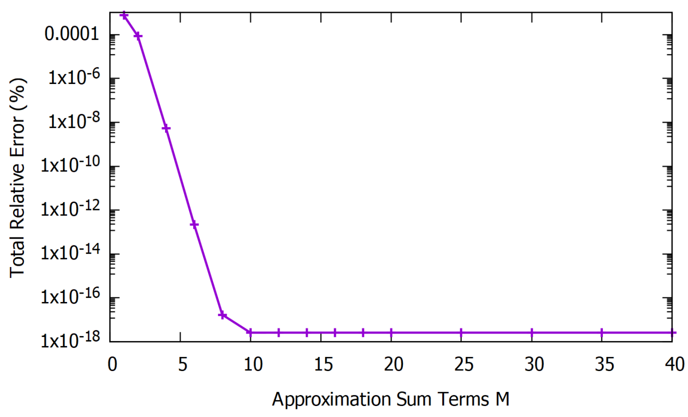


Figure 2.2: *Compression of 1D unsteady field* (2.41) *by iPGD. The relation between total relative error (%) with the number of terms of the approximation sum.*

Figure 2.2 makes clear that, in this application, even for $M = 2$ the error is very small (less than 0.0001%), while for greater M it is practically zero. As mentioned in section 1.2, the definition of the original field requires:

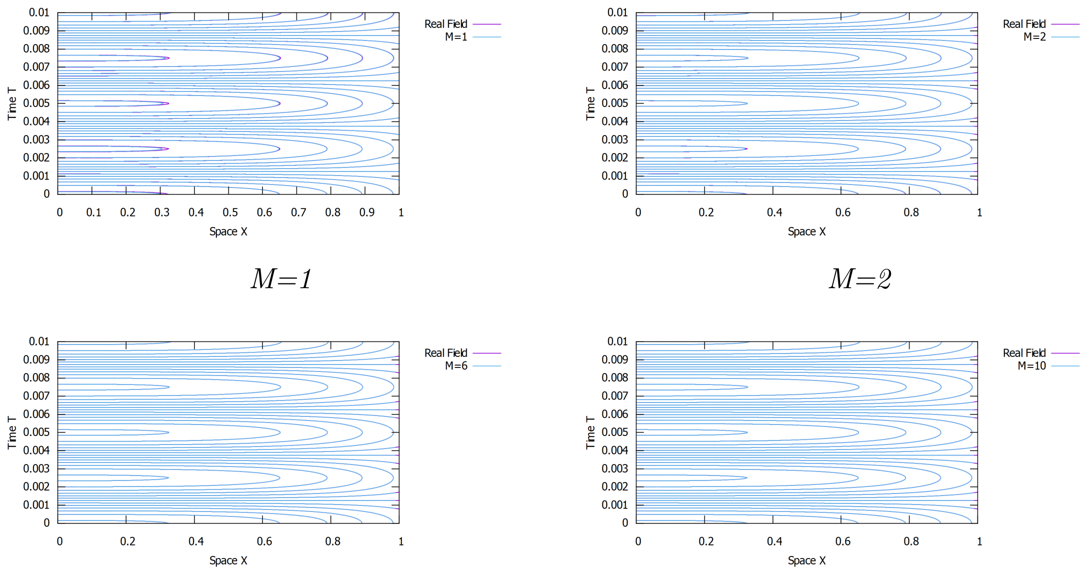
$$Q_1 = I \cdot K = 81 \cdot 200 = 16200$$

values, while the compressed one requires:

$$Q_2 = M \cdot (I + K) = 281M$$

values. For $M = 2$, the required storage capacity is decreased to the $3,47\%$ of Q_1, while even for $M = 4$ (where the error is $10^{-8}\%$) it is decreased to the 7% of Q_1.

In order to visualize the increase of accuracy when increasing M, figures 2.3 show the iso-u curves of the original field and the approximation for $M = 1$, 2, 4, and 10.



M=1 *M=2*

M=6 *M=10*

Figure 2.3: *Compression of 1D unsteady field* (2.41) *by iPGD. Iso-u curves of the original field and the iPGD approximations constructed for M equal to 1, 2, 6 and 10.*

Figures 2.3 highlight the ability of iPGD to perfectly approximate the 2.41 field even for $M = 2$. For the above construction of the compressed field, the weight coefficient was chosen equal to 1, the maximum number of the internal iterations equal to 20 and for the stopping criterion of these iterations a threshold value equal to 10^{-14} is used. Unless otherwise stated, these parameters are used in all examples.

2.2.2 Compression of an unsteady 2D field

The following analysis is an extension of the above, so it appears more concise. Definitions are omitted as they were given in detail above. Supposing $U(x, y, t)$ is a 2D unsteady field to be compressed by iPGD. The spatial coordinate x is divided in I discrete nodes ($i = 1, .., I$) and the spatial coordinate y in J discrete nodes ($j = 1, .., J$). The total time is divided in discrete time steps ($k = 1, .., K, ..$), where K is the current one. The approximation of field $U(x, y, t)$ in discrete form is:

$$U_{i,j,k} \approx \sum_{\mu=1}^{M} X_i^\mu \; Y_j^\mu \; T_k^\mu \tag{2.42}$$

The total error that takes into account both the exact field at the next time step and all the previous time-series by their up-to-now approximation, is defined as:

$$\begin{aligned} E = \frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu T_{K+1}^\mu - U_{i,j,K+1}\right]^2 \\ + \frac{w}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu T_k^\mu - \sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu\right]^2 \end{aligned} \tag{2.43}$$

where $\sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu$, for $k = 1, .., K$, is the up-to-now approximation.

Quantity E from equation (2.43) will be minimum if its partial derivatives (with respect to the bases X_i^m, Y_j^m, T_k^m and T_{K+1}^m) are equal to zero. That is:

$$\frac{\partial E}{\partial X_i^m} = 0 \Leftrightarrow$$

$$\sum_{j=1}^{J}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu T_{K+1}^\mu - U_{i,j,K+1}\right] Y_j^m T_{K+1}^m + w\sum_{j=1}^{J}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu T_k^\mu - \sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu\right] Y_j^m T_k^m = 0 \Leftrightarrow$$

$$X_i^m = \frac{T_{K+1}^m \sum_{j=1}^{J}\left[U_{i,j,K+1} - \sum_{\mu=1}^{m-1} X_i^\mu Y_j^\mu T_{K+1}^\mu\right] Y_j^m + w\sum_{j=1}^{J}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu - \sum_{\mu=1}^{m-1} X_i^\mu Y_j^\mu T_K^\mu\right] Y_j^m T_k^m}{\left(T_{K+1}^m\right)^2 \sum_{j=1}^{J}\left(Y_j^m\right)^2 + w\sum_{j=1}^{J}\sum_{k=1}^{K}\left(Y_j^m\right)^2 \left(T_k^m\right)^2} \tag{2.44}$$

$$\frac{\partial E}{\partial Y_j^m} = 0 \Leftrightarrow$$

$$\sum_{i=1}^{I}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu T_{K+1}^\mu - U_{i,j,K+1}\right] X_i^m T_{K+1}^m + w\sum_{i=1}^{I}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu T_k^\mu - \sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu\right] X_i^m T_k^m = 0 \Leftrightarrow$$

$$Y_j^m = \frac{T_{K+1}^m \sum_{i=1}^{I}\left[U_{i,j,K+1} - \sum_{\mu=1}^{m-1} X_i^\mu Y_j^\mu T_{K+1}^\mu\right] X_i^m + w\sum_{i=1}^{I}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu - \sum_{\mu=1}^{m-1} X_i^\mu Y_j^\mu T_K^\mu\right] X_i^m T_k^m}{\left(T_{K+1}^m\right)^2 \sum_{i=1}^{I}\left(X_i^m\right)^2 + w\sum_{i=1}^{I}\sum_{k=1}^{K}\left(X_i^m\right)^2\left(T_k^m\right)^2} \tag{2.45}$$

$$\frac{\partial E}{\partial T_k^m} = 0 \Leftrightarrow$$

$$w\sum_{i=1}^{I}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu T_k^\mu - \sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu\right] X_i^m Y_j^m = 0 \Leftrightarrow$$

$$T_k^m = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu - \sum_{\mu=1}^{m-1} X_i^\mu Y_j^\mu T_K^\mu\right] X_i^m Y_j^m}{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(X_i^m\right)^2\left(Y_j^m\right)^2} \tag{2.46}$$

$$\frac{\partial E}{\partial T_{K+1}^m} = 0 \Leftrightarrow$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu T_{K+1}^\mu - U_{i,j,K+1}\right] X_i^m Y_j^m = 0 \Leftrightarrow$$

$$T_{K+1}^m = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\left[U_{i,j,K+1} - \sum_{\mu=1}^{m-1} X_i^\mu Y_j^\mu T_{K+1}^\mu\right] X_i^m Y_j^m}{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(X_i^m\right)^2\left(Y_j^m\right)^2} \tag{2.47}$$

Equations (2.44)-(2.47) contain many nested sums of different variables. Computational algorithms that contain these kind of equations tend to be very slow when solving large-scale fields. This problem was emphasized by V. Papageorgiou (ref. [2]) and is the main challenge in the present diploma thesis. The computational time is reduced if equations (2.44)-(2.47) are rearranged by the use of the identities of summation (2.44)-(2.47). Thus, equations (2.44)-(2.47) are transformed to:

$$X_i^m = \frac{T_{K+1}^m \sum_{j=1}^{J} U_{i,j,K+1} Y_j^m + w \sum_{\mu=1}^{M} \left[\widetilde{X}_i^\mu \sum_{j=1}^{J} \left(\widetilde{Y}_j^\mu Y_j^m \right) \sum_{k=1}^{K} \left(\widetilde{T}_k^\mu T_k^m \right) \right]}{\left(T_{K+1}^m \right)^2 \sum_{j=1}^{J} \left(Y_j^m \right)^2 + w \sum_{j=1}^{J} \left(Y_j^m \right)^2 \sum_{k=1}^{K} \left(T_k^m \right)^2}$$

$$- \frac{\sum_{\mu=1}^{m-1} \left\{ X_i^\mu \left[T_{K+1}^\mu T_{K+1}^m + w \sum_{k=1}^{K} \left(T_k^\mu T_k^m \right) \right] \sum_{j=1}^{J} \left(Y_j^\mu Y_j^m \right) \right\}}{\left(T_{K+1}^m \right)^2 \sum_{j=1}^{J} \left(Y_j^m \right)^2 + w \sum_{j=1}^{J} \left(Y_j^m \right)^2 \sum_{k=1}^{K} \left(T_k^m \right)^2} \tag{2.48}$$

$$T_{K+1}^m = \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} U_{i,j,K+1} X_i^m Y_j^m - \sum_{\mu=1}^{m-1} \left[T_{K+1}^\mu \sum_{i=1}^{I} \left(X_i^\mu X_i^m \right) \sum_{j=1}^{J} \left(Y_j^\mu Y_j^m \right) \right]}{\sum_{i=1}^{I} \left(X_i^m \right)^2 \sum_{j=1}^{J} \left(Y_j^m \right)^2} \tag{2.49}$$

$$Y_j^m = \frac{T_{K+1}^m \sum_{i=1}^{I} U_{i,j,K+1} X_i^m + w \sum_{\mu=1}^{M} \left[\widetilde{Y}_i^\mu \sum_{i=1}^{I} \left(\widetilde{X}_i^\mu X_i^m \right) \sum_{k=1}^{K} \left(\widetilde{T}_k^\mu T_k^m \right) \right]}{\left(T_{K+1}^m \right)^2 \sum_{i=1}^{I} \left(X_i^m \right)^2 + w \sum_{i=1}^{I} \left(X_i^m \right)^2 \sum_{k=1}^{K} \left(T_k^m \right)^2}$$

$$- \frac{\sum_{\mu=1}^{m-1} \left\{ Y_j^\mu \left[T_{K+1}^\mu T_{K+1}^m + w \sum_{k=1}^{K} \left(T_k^\mu T_k^m \right) \right] \sum_{i=1}^{I} \left(X_i^\mu X_i^m \right) \right\}}{\left(T_{K+1}^m \right)^2 \sum_{i=1}^{I} \left(X_i^m \right)^2 + w \sum_{i=1}^{I} \left(X_i^m \right)^2 \sum_{k=1}^{K} \left(T_k^m \right)^2} \tag{2.50}$$

$$T_k^m = \frac{\sum_{\mu=1}^{M} \left[\widetilde{T}_k^\mu \sum_{i=1}^{I} \left(\widetilde{X}_i^\mu X_i^m \right) \sum_{j=1}^{J} \left(\widetilde{Y}_j^\mu Y_j^m \right) \right] - \sum_{\mu=1}^{m-1} \left[T_k^\mu \sum_{i=1}^{I} \left(X_i^\mu X_i^m \right) \sum_{j=1}^{J} \left(Y_j^\mu Y_j^m \right) \right]}{\sum_{i=1}^{I} \left(X_i^m \right)^2 \sum_{j=1}^{J} \left(Y_j^m \right)^2} \tag{2.51}$$

In equations (2.48)-(2.51) the requested bases are coupled, so an iterative procedure is necessary. In order to start these iterations, it is proposed to initialize the time-related bases T_1^μ as 1 and approximate the field of this first time step as a steady field via the standard PGD method (not the incremental). That means:

$$U'(x,y) \equiv U(x,y,t=1)$$

$$U'(x,y) \approx \sum_{\mu=1}^{M} X^{\mu}(x) Y^{\mu}(y)$$

So, the bases X_i^m (for $i = 1,..,I$) , Y_j^m (for $j = 1,..,J$) and T_1^{μ} are computed from:

$$X_i^m = \frac{\sum_{j=1}^{J} U_{i,j,1} Y_j^m by_j - \sum_{\mu=1}^{m-1} X_i^{\mu} \sum_{j=1}^{J} Y_j^m Y_j^{\mu} by_j}{\sum_{j=1}^{J} \left(Y_j^m\right)^2 by_j}, \qquad \mu = 1,..,M \tag{2.52a}$$

$$Y_j^m = \frac{\sum_{i=1}^{I} U_{i,j,1} X_i^m bx_i - \sum_{\mu=1}^{m-1} Y_j^{\mu} \sum_{i=1}^{I} X_i^m X_i^{\mu} bx_i}{\sum_{i=1}^{I} \left(X_i^m\right)^2 bx_i}, \qquad \mu = 1,..,M \tag{2.52b}$$

$$T_1^{\mu} = 1, \qquad \mu = 1,..,M \tag{2.52c}$$

From equations (2.52), the first two are the discretized form of equations (2.10) and (2.13). The discretization technique used here for the integrals is the Trapezoidal Integration method (ref. 5), so the following integration factors appear:

$$bx_i = \begin{cases} \frac{1}{2}, & i = 1 \\ 1, & i = 2,..,I-1 \\ \frac{1}{2}, & i = I \end{cases} \tag{2.53a}$$

$$by_j = \begin{cases} \frac{1}{2}, & j = 1 \\ 1, & j = 2,..,J-1 \\ \frac{1}{2}, & j = J \end{cases} \tag{2.53b}$$

The accuracy of iPGD is quantified by the total relative error (%):

$$\mathcal{E}(M) = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} \left(U_{i,j,k} - \sum_{\mu=1}^{M} X_i^{\mu} Y_j^{\mu} T_k^{\mu}\right)^2}{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} U_{i,j,k}^2} \tag{2.54}$$

Figure 2.4 shows the PGD algorithm for the initialization of X_i^{μ}, Y_j^{μ} and T_1^{μ} from equations 2.52. Figure 2.5 shows the iPGD algorithm presented above. In this flowchart, the initialization algorithm is represented by a process box (rectangle) labeled as "PGD for Initialization" that contains the main body of the first flowchart.

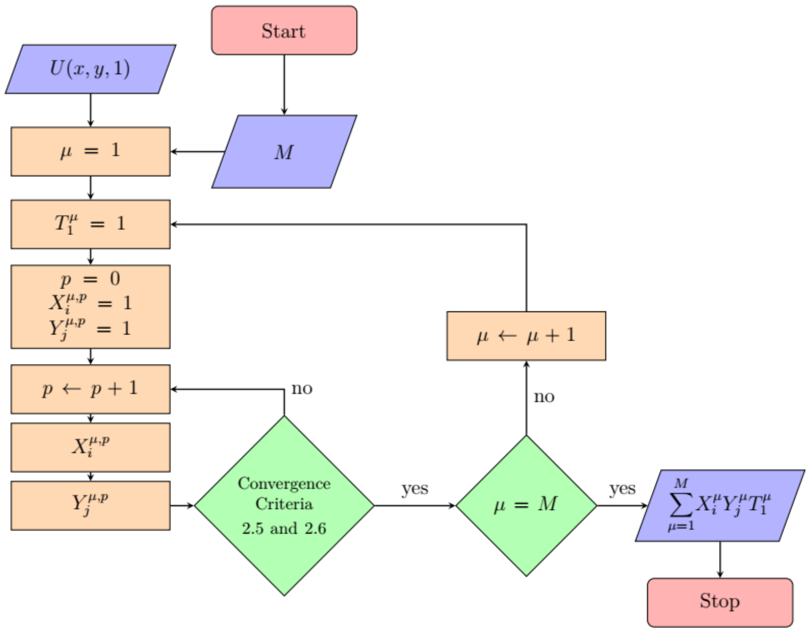


Figure 2.4: *Flowchart of the PGD algorithm for the initialization of the iPGD bases.*

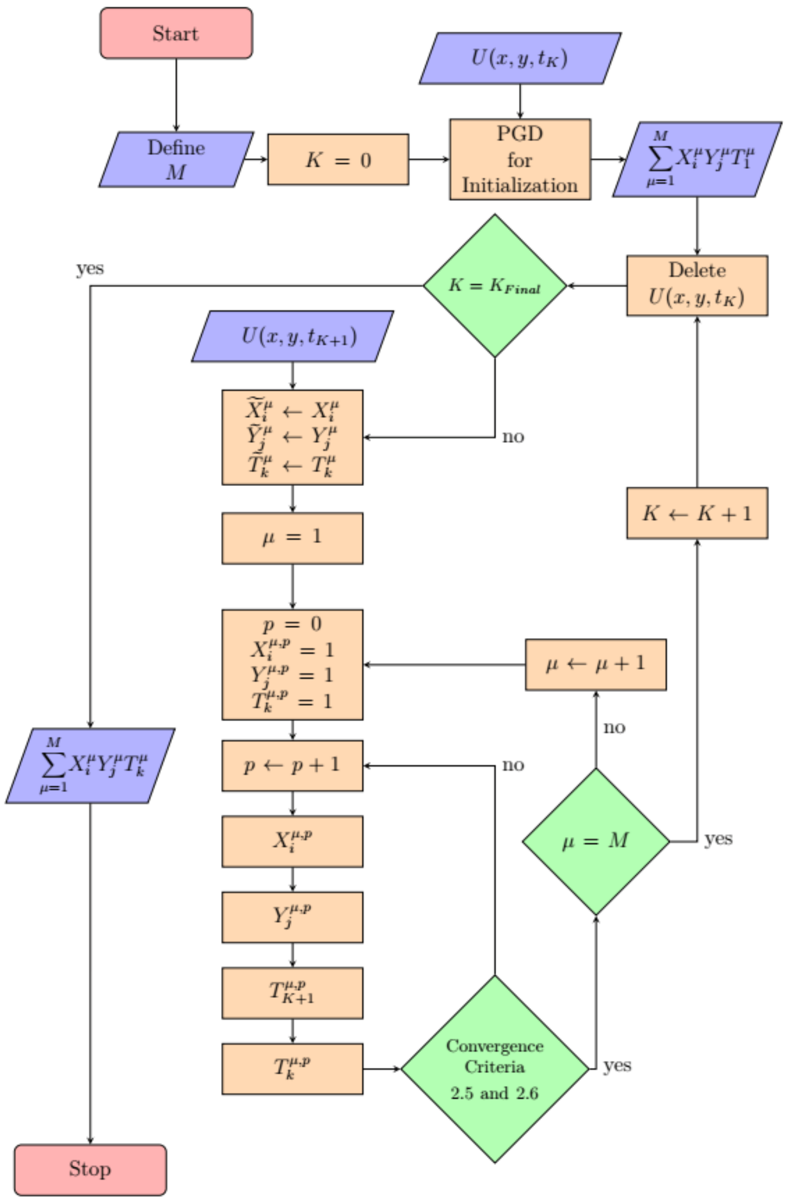


Figure 2.5: *Flowchart of the iPGD algorithm.*

Numerical example

The 2D unsteady field $u(x,y,t)$ to be compressed is the following:

$$u(x,y,t) = \frac{\sin\left(\sqrt{10\frac{t}{T_{int}}x^2 + y^2}\right)}{\sqrt{x^2+y^2}} \quad , \tag{2.55}$$

where x, y are spatial variables and t is the time. The total length in both x and y directions is equal to 1 and both lengths are divided in $I = 21$ and $J = 21$ nodes respectively. The total time is equal to $T_{int} = 0.01$ and divided in 50 time steps.

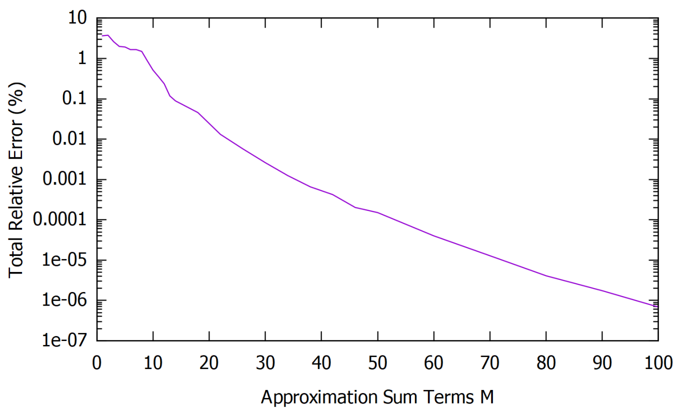


Figure 2.6: *Compression of 2D unsteady field* (2.55) *by iPGD. The relation between total relative error (%) (eq. 2.54) with the number of terms of the approximation sum.*

Figure 2.6 monitors the total error (2.54) when the terms of the approximation sum are increasing. At least 14 terms are required in order for the approximation error to become less than 0.1%. For this number of terms, the required storage capacity is 5.8% of the original field u. In order to have an error around 1% ($M = 9$) the required storage capacity is only 3.8% of the full storage.

In appendix A, the field (2.55) is compared with the iPGD approximations for various M numbers. The main conclusion from figures A.1 - A.4 is that iPGD can accurately approximate this field even with 20 terms, having achieved a percent reduction of the required storage capacity equal to:

$$\frac{Q_1 - Q_2}{Q_1} = \frac{I \cdot J \cdot K - M(I + J + K)}{I \cdot J \cdot K} = \frac{21 \cdot 21 \cdot 50 - 20(21 + 21 + 50)}{21 \cdot 21 \cdot 50} = 92\%$$

At this point, the dependence of the accuracy to the weight coefficient w of equations (2.48) and (2.50) must be studied. For this purpose, the total relative error 2.54 is computed for various approximation terms M and for various w of different orders of magnitude. The results are given in figure 2.7 where each curve represents the relation between the error and M for specific weight coefficient w. The weight coefficients used here are 0.001, 0.01, 0.1, 1, 10, 100 and 1000.

In figure 2.7, the curve of $w = 1$ seems the most accurate and the curves $w = 0.1$ and $w = 10$ are very close to the first and have the same behaviour. Curves of smaller weight coefficient ($w = 0.01$ and $w = 0.001$) have different behaviour. For small number of terms ($M < 40$) the accuracy is almost one order of magnitude worst than the $w = 1$ curve, but for $M > 40$ the accuracy is improved and almost coincides the $w = 1$ curve. Using a so small weight coefficient (as $w = 0.001$) is like neglecting the up-to-now approximation, and the redefinition of the X_i^μ and Y_j^μ bases is based only on approximating the field of the next time step, but it cannot be achieved

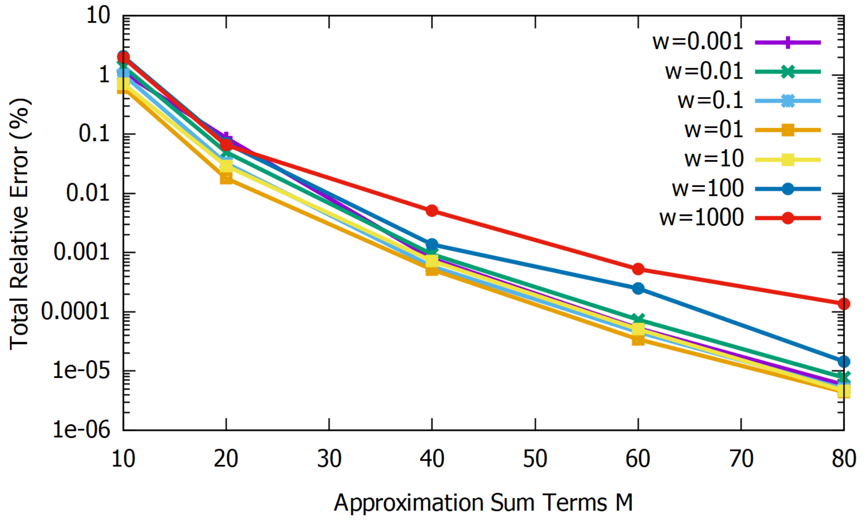


Figure 2.7: *Compression of 2D unsteady field* (2.55) *by iPGD.The total relative error (%) 2.54 as a function of number of approximation terms M. Every curve corresponds to a different weight coefficient w.*

accurately for small M. The accuracy of $w = 100$ and $w = 1000$ curves is bad and at large values of M this accuracy gap increases. It seems that the behaviour of $w > 1$ and $0 < w < 1$ are opposite. The best behaviour of $0 < w < 1$ curves is at large M, while the best behaviour of $w > 1$ curves is at small M. As mentioned, the most accurate curve is for $w = 1$. Denoting the two terms of the approximation error 2.7 (approximation based on the up-to-now approach and approximation based on the field of the next time step) equal weight seems the most rational strategy.

2.2.3 The improvement in the iPGD method

The iPGD method was initially presented and extensively studied in V. Papageorgiou's diploma thesis (ref. [2] and [3]). There, the equations for computing the bases were formed, the importance of every term of the approximation sum was studied and the ability of this method to construct approximations of high accuracy was validated through numerical examples. Apart from all these, a great disadvantage was highlighted - the algorithms are very slow when compressing large-scale unsteady fields. The purpose of this subsection is to highlight the improvement on iPGD done in the present thesis. The greater part of it is related with the acceleration of the method, but also accuracy was improved. For short, in this section, any reference to Papageorgiou's diploma thesis will be referred to hereinafter as VPDT and any reference to the present diploma thesis will be referred as LKDT.

In VPDT, when compressing an unsteady 2D field, the bases X_i^m, Y_j^m, T_k^m and T_{K+1}^m

were computed by equations:

$$X_i^m = \frac{T_{K+1}^m \sum_{j=1}^{J} Y_j^m U_{i,j,K+1} + w\widetilde{X}_i^m \sum_{k=1}^{K}\sum_{j=1}^{J} \widetilde{Y}_j^m Y_j^m \widetilde{T}_k^m T_k^m - Q1_x - Q2_x}{(T_{K+1}^m)^2 \sum_{j=1}^{J} (Y_j^m)^2 + w \sum_{k=1}^{K}\sum_{j=1}^{J} (Y_j^m)^2 (T_k^m)^2} \tag{2.56}$$

where $Q1_x = T_{K+1}^m \sum_{\mu=1}^{m-1} \left[\left(\sum_{j=1}^{J} (Y_j^\mu Y_j^m) \right) X_i^\mu T_{K+1}^\mu \right]$

and $Q2_x = w \sum_{k=1}^{K}\sum_{j=1}^{J} \left[\sum_{\mu=1}^{m-1} \left(X_i^\mu Y_j^\mu T_k^\mu - \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu \right) Y_j^m T_k^m \right]$

$$Y_i^m = \frac{T_{K+1}^m \sum_{i=1}^{I} X_i^m U_{i,j,K+1} + w\widetilde{Y}_j^m \sum_{k=1}^{K}\sum_{i=1}^{I} \widetilde{X}_i^m X_i^m \widetilde{T}_k^m T_k^m - Q1_y - Q2_y}{(T_{K+1}^m)^2 \sum_{i=1}^{I} (X_i^m)^2 + w \sum_{k=1}^{K}\sum_{i=1}^{I} (X_i^m)^2 (T_k^m)^2} \tag{2.57}$$

where $Q1_y = T_{K+1}^m \sum_{\mu=1}^{m-1} \left[\left(\sum_{i=1}^{I} (X_i^\mu X_i^m) \right) Y_j^\mu T_{K+1}^\mu \right]$

and $Q2_y = w \sum_{k=1}^{K}\sum_{i=1}^{I} \left[\sum_{\mu=1}^{m-1} \left(X_i^\mu Y_j^\mu T_k^\mu - \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu \right) X_i^m T_k^m \right]$

$$T_k^m = \frac{\widetilde{T}_k^m \sum_{i=1}^{I}\sum_{j=1}^{J} \widetilde{X}_i^m X_i^m \widetilde{Y}_j^m Y_j^m - \sum_{i=1}^{I}\sum_{j=1}^{J} \left(X_i^\mu Y_j^\mu T_k^\mu - \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu \right) X_i^m Y_j^m}{\sum_{i=1}^{I}\sum_{j=1}^{J} (X_i^m)^2 (Y_j^m)^2} \tag{2.58}$$

$$T_{K+1}^m = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} X_i^m Y_J^m U_{i,j,K+1} - \sum_{\mu=1}^{m-1} \left[T_{K+1}^\mu \sum_{i=1}^{I}\sum_{j=1}^{J} X_i^\mu \, X_i^m \, Y_j^m \, Y_j^\mu \right]}{\sum_{i=1}^{I}\sum_{j=1}^{J} (X_i^m)^2 (Y_j^m)^2} \tag{2.59}$$

The unsteady 2D field of the previous numerical example (2.55) is compressed by iPGD using equations (2.56) - (2.59). This approximation is compared with those of equations (2.48) - (2.49). For both versions, the computational time is monitored and presented in figure 2.8b for M, while figure 2.8a compares the total relative error (%) between the two versions. In these figures VPDT (or ΔΕΒΠ) refers to equations (2.56) - (2.59) and LKDT (or ΔΕΛΚ) refers to equations (2.48) - (2.49).

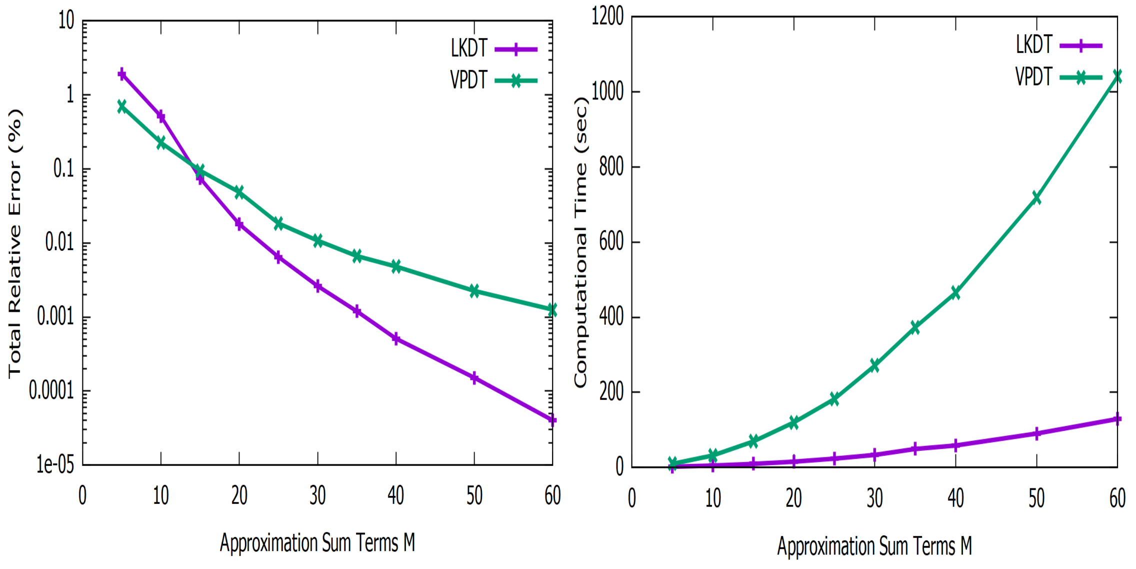
Additionally, figures 2.9 - 2.11 show the iso-u curves of the original field and of the approximations of both iPGD versions. The improvement of the method is concluded by all figures 2.8 and 2.9 - 2.11. For example, for $M = 60$, figures 2.8 show that LKDT version needs 12% of the time that VPDT needs and the error of LKDT is 31 times smaller than the one of VPDT version. The form of these curves indicate that the difference could be even larger for more summation terms M.

Two are the reasons that the LKDT version of iPGD algorithms is faster. The first one is that the equations of this version do not contain multiple summations due to the use of summation identities (2.34) (in contrast with equations 2.56 - 2.59). This is an advantage, because in programming multiple summation means loops nested inside other loops and this produces very slow algorithms. The second reason is that the equations of LKDT version are solved in specific steps that accelerate the algorithm even more. These steps are presented in details in Chapter 3.

The reason that equations 2.56 - 2.59 produce less accurate approximations can be found in their birth process (see ref. [2]). In both versions, during the process of computing the m-th bases for the new time step $(K+1)$, in the equation of the approximation error the real field at all previous time steps is replaced by the up to that time approximation. That is the equation (2.43) and is repeated here:

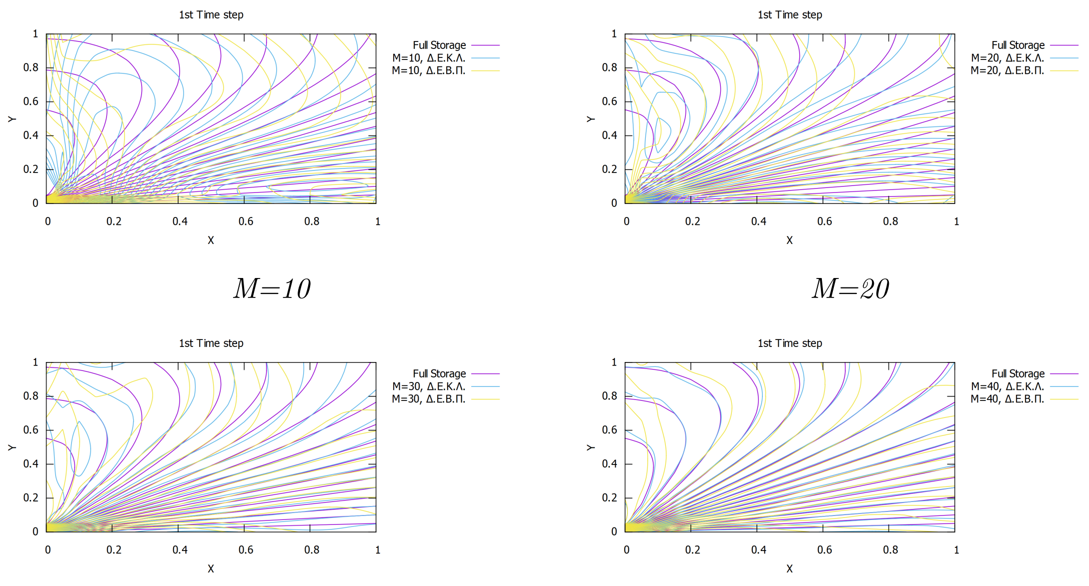
$$E = \frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu T_{K+1}^\mu - U_{i,j,K+1}\right]^2 + \frac{w}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu T_k^\mu - \underbrace{\sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu}_{\text{up to } K\text{th time step approach}}\right]^2 \tag{2.60}$$

But equations 2.56 - 2.59 instead of using all the M terms of the approximation sum, exploit only the m first. This means that, for these equations, it is assumed that $U_{i,j,k} \approx \sum_{\mu=1}^{m} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu$, for $k = 1,..K$ (and not $U_{i,j,k} \approx \sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu$) and thus, relations 2.56 - 2.59 ignore the contribution of $\sum_{\mu=m+1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{T}_k^\mu$ terms.



(a) *Error* **(b)** *Time*

Figure 2.8: *Compression of 2D unsteady field* (2.55) *by iPGD. The total relative error (%) (eq. 2.54) (left) and the computational time (right) of iPGD for VPDT version (eqs. 2.56 - 2.59) and LKDT version (eqs. 2.48 - 2.49). x-axis the terms of the approximation sum.*



M=10 *M=20*

M=30 *M=40*

Figure 2.9: *Compression of 2D unsteady field* (2.55) *by iPGD. Iso-u curves of the real field* (2.55) *and the approximations by the VPDT version of iPGD (eqs. 2.56 - 2.59) and the LKDT version of iPGD (eqs. 2.48 - 2.49) at the* 1*-st time step for M=10,20,30,40.*

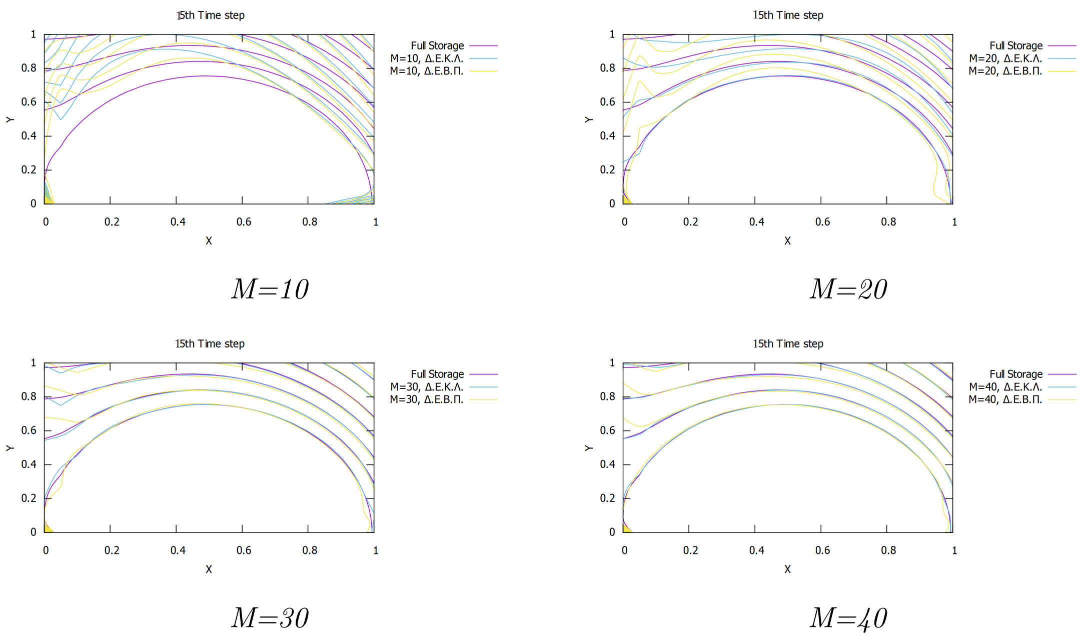
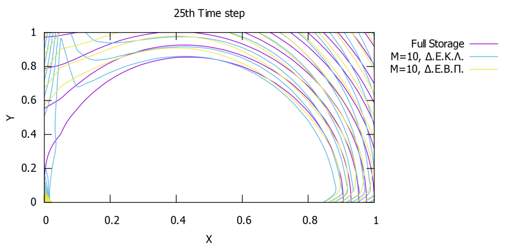
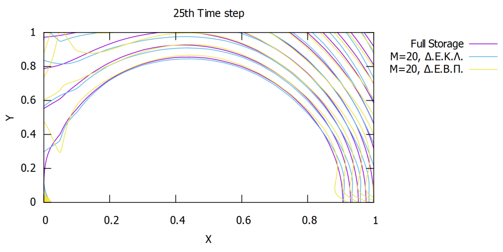


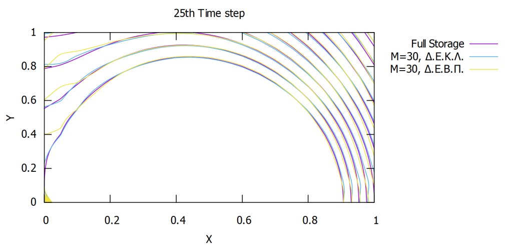
Figure 2.10: *Compression of 2D unsteady field* (2.55) *by iPGD. Iso-u curves of the real field* (2.55) *and the approximations by the VPDT version of iPGD (eqs. 2.56 - 2.59) and the LKDT version of iPGD (eqs. 2.48 - 2.49) at the* 15*-th time step for M=10,20,30,40.*



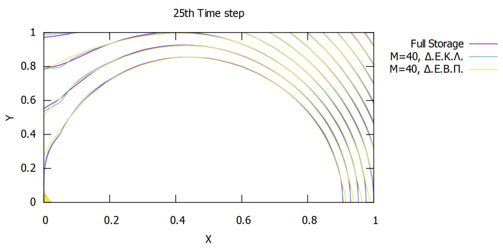
M=10



M=20



M=30



M=40

Figure 2.11: *Compression of 2D unsteady field* (2.55) *by iPGD. Iso-u curves of the real field* (2.55) *and the approximations by the VPDT version of iPGD (eqs. 2.56 - 2.59) and the LKDT version of iPGD (eqs. 2.48 - 2.49) at the* 25*-th time step for M=10,20,30,40.*

2.2.4 Compression of an unsteady 3D field

The extension for unsteady 3D fields of the previous analysis is concisely presented in this subsection. $U(x, y, z, t)$ is the field to be compressed. The spatial variables x, y and z are discretized in I, J and L nodes respectively, where $i = 1, .., I$, $j = 1, .., J$ and $l = 1, .., L$. The total time is divided in discrete time steps ($k = 1, .., K, ..$), where K is the current one. The approximation of $U(x, y, z, t)$ in discrete form is:

$$U_{i,j,l,k} \approx \sum_{\mu=1}^{M} X_i^{\mu}\ Y_j^{\mu}\ Z_l^{\mu}\ T_k^{\mu} \tag{2.61}$$

The total error taking into account both the solution of the next time step $K + 1$ and the time-series of all the previous time steps by their up-to-now approximation is defined as:

$$E = \frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{l=1}^{L}\left[\sum_{\mu=1}^{m} X_i^{\mu} Y_j^{\mu} Z_l^{\mu} T_{K+1}^{\mu} - U_{i,j,l,K+1}\right]^2 + \frac{w}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{l=1}^{L}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{m} X_i^{\mu} Y_j^{\mu} Z_l^{\mu} T_k^{\mu} - \sum_{\mu=1}^{M} \widetilde{X}_i^{\mu} \widetilde{Y}_j^{\mu} \widetilde{Z}_l^{\mu} \widetilde{T}_k^{\mu}\right]^2 \quad (2.62)$$

where: $U_{i,j,l,k}^{iPGD} \approx \sum\limits_{\mu=1}^{M} \widetilde{X}_i^{\mu} \widetilde{Y}_j^{\mu} \widetilde{Z}_l^{\mu} \widetilde{T}_k^{\mu}$, for $k = 1, .., K$, is the up-to-now approximation of the $U_{i,j,l,k}$ time series.

In order to compute the bases X_i^m, Y_j^m, Z_l^m, T_k^m and T_{K+1}^m the partial derivatives of the error (2.62) (with respect to these bases) have to be equal to zero. Moreover, by using the familiar general identities of summation (2.34), the bases X_i^m, Y_j^m, Z_l^m, T_k^m and T_{K+1}^m are computed by:

$$\frac{\partial E}{\partial X_i^m} = 0 \Leftrightarrow$$

$$\sum_{j=1}^{J}\sum_{l=1}^{L}\left[\sum_{\mu=1}^{m} X_i^{\mu} Y_j^{\mu} Z_l^{\mu} T_{K+1}^{\mu} - U_{i,j,l,K+1}\right] Y_j^m Z_l^m T_{K+1}^m + w\sum_{j=1}^{J}\sum_{l=1}^{L}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{m} X_i^{\mu} Y_j^{\mu} Z_l^{\mu} T_k^{\mu} - \sum_{\mu=1}^{M} \widetilde{X}_i^{\mu} \widetilde{Y}_j^{\mu} \widetilde{Z}_l^{\mu} \widetilde{T}_k^{\mu}\right] Y_j^m Z_l^m T_k^m = 0 \Leftrightarrow$$

$$X_i^m = \frac{T_{K+1}^m \sum\limits_{j=1}^{J}\sum\limits_{l=1}^{L} U_{i,j,l,K+1} Y_j^m Z_l^m + w\sum\limits_{\mu=1}^{M}\left[\widetilde{X}_i^{\mu} \sum\limits_{j=1}^{J}\left(\widetilde{Y}_j^{\mu} Y_j^m\right)\sum\limits_{l=1}^{L}\left(\widetilde{Z}_l^{\mu} Z_l^m\right)\sum\limits_{k=1}^{K}\left(\widetilde{T}_k^{\mu} T_k^m\right)\right]}{\left(T_{K+1}^m\right)^2 \sum\limits_{j=1}^{J}\left(Y_j^m\right)^2 \sum\limits_{l=1}^{L}\left(Z_l^m\right)^2 + w\sum\limits_{j=1}^{J}\left(Y_j^m\right)^2\sum\limits_{l=1}^{L}\left(Z_l^m\right)^2\sum\limits_{k=1}^{K}\left(T_k^m\right)^2} - \frac{\sum\limits_{\mu=1}^{m-1}\left\{X_i^{\mu}\left[T_{K+1}^{\mu} T_{K+1}^m + w\sum\limits_{k=1}^{K}\left(T_k^{\mu} T_k^m\right)\right]\sum\limits_{j=1}^{J}\left(Y_j^{\mu} Y_j^m\right)\sum\limits_{l=1}^{L}\left(Z_l^{\mu} Z_l^m\right)\right\}}{\left(T_{K+1}^m\right)^2 \sum\limits_{j=1}^{J}\left(Y_j^m\right)^2 \sum\limits_{l=1}^{L}\left(Z_l^m\right)^2 + w\sum\limits_{j=1}^{J}\left(Y_j^m\right)^2\sum\limits_{l=1}^{L}\left(Z_l^m\right)^2\sum\limits_{k=1}^{K}\left(T_k^m\right)^2} \quad (2.63)$$

$$\frac{\partial E}{\partial Y_j^m} = 0 \Leftrightarrow$$

$$\sum_{i=1}^{I}\sum_{l=1}^{L}\left[\sum_{\mu=1}^{m} X_i^{\mu} Y_j^{\mu} Z_l^{\mu} T_{K+1}^{\mu} - U_{i,j,l,K+1}\right] X_i^m Z_l^m T_{K+1}^m + w\sum_{i=1}^{I}\sum_{l=1}^{L}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{m} X_i^{\mu} Y_j^{\mu} Z_l^{\mu} T_k^{\mu} - \sum_{\mu=1}^{M} \widetilde{X}_i^{\mu} \widetilde{Y}_j^{\mu} \widetilde{Z}_l^{\mu} \widetilde{T}_k^{\mu}\right] X_i^m Z_l^m T_k^m = 0 \Leftrightarrow$$

$$Y_i^m = \frac{T_{K+1}^m \sum_{i=1}^{I}\sum_{l=1}^{L} U_{i,j,l,K+1} X_i^m Z_l^m + w\sum_{\mu=1}^{M}\left[\widetilde{Y}_j^{\mu}\sum_{i=1}^{I}\left(\widetilde{X}_i^{\mu} X_i^m\right)\sum_{l=1}^{L}\left(\widetilde{Z}_l^{\mu} Z_l^m\right)\sum_{k=1}^{K}\left(\widetilde{T}_k^{\mu} T_k^m\right)\right]}{\left(T_{K+1}^m\right)^2 \sum_{i=1}^{I}\left(X_I^m\right)^2 \sum_{l=1}^{L}\left(Z_l^m\right)^2 + w\sum_{i=1}^{I}\left(X_i^m\right)^2\sum_{l=1}^{L}\left(Z_l^m\right)^2\sum_{k=1}^{K}\left(T_k^m\right)^2} - \frac{\sum_{\mu=1}^{m-1}\left\{Y_j^{\mu}\left[T_{K+1}^{\mu} T_{K+1}^m + w\sum_{k=1}^{K}\left(T_k^{\mu} T_k^m\right)\right]\sum_{i=1}^{I}\left(X_i^{\mu} X_i^m\right)\sum_{l=1}^{L}\left(Z_l^{\mu} Z_l^m\right)\right\}}{\left(T_{K+1}^m\right)^2 \sum_{i=1}^{I}\left(X_I^m\right)^2 \sum_{l=1}^{L}\left(Z_l^m\right)^2 + w\sum_{i=1}^{I}\left(X_i^m\right)^2\sum_{l=1}^{L}\left(Z_l^m\right)^2\sum_{k=1}^{K}\left(T_k^m\right)^2} \tag{2.64}$$

$$\frac{\partial E}{\partial Z_l^m} = 0 \Leftrightarrow$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{m} X_i^{\mu} Y_j^{\mu} Z_l^{\mu} T_{K+1}^{\mu} - U_{i,j,l,K+1}\right] X_i^m Y_j^m T_{K+1}^m + w\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{m} X_i^{\mu} Y_j^{\mu} Z_l^{\mu} T_k^{\mu} - \sum_{\mu=1}^{M} \widetilde{X}_i^{\mu} \widetilde{Y}_j^{\mu} \widetilde{Z}_l^{\mu} \widetilde{T}_k^{\mu}\right] X_i^m Z_l^m T_k^m = 0 \Leftrightarrow$$

$$Z_i^m = \frac{T_{K+1}^m \sum_{i=1}^{I}\sum_{j=1}^{J} U_{i,j,l,K+1} X_i^m Y_j^m + w\sum_{\mu=1}^{M}\left[\widetilde{Z}_l^{\mu}\sum_{i=1}^{I}\left(\widetilde{X}_i^{\mu} X_i^m\right)\sum_{j=1}^{J}\left(\widetilde{Y}_j^{\mu} Y_j^m\right)\sum_{k=1}^{K}\left(\widetilde{T}_k^{\mu} T_k^m\right)\right]}{\left(T_{K+1}^m\right)^2 \sum_{i=1}^{I}\left(X_I^m\right)^2 \sum_{l=1}^{L}\left(Y_j^m\right)^2 + w\sum_{i=1}^{I}\left(X_i^m\right)^2\sum_{j=1}^{J}\left(Y_j^m\right)^2\sum_{k=1}^{K}\left(T_k^m\right)^2} - \frac{\sum_{\mu=1}^{m-1}\left\{Z_l^{\mu}\left[T_{K+1}^{\mu} T_{K+1}^m + w\sum_{k=1}^{K}\left(T_k^{\mu} T_k^m\right)\right]\sum_{i=1}^{I}\left(X_i^{\mu} X_i^m\right)\sum_{j=1}^{J}\left(Y_j^{\mu} Y_j^m\right)\right\}}{\left(T_{K+1}^m\right)^2 \sum_{i=1}^{I}\left(X_I^m\right)^2 \sum_{l=1}^{L}\left(Y_j^m\right)^2 + w\sum_{i=1}^{I}\left(X_i^m\right)^2\sum_{j=1}^{J}\left(Y_j^m\right)^2\sum_{k=1}^{K}\left(T_k^m\right)^2} \tag{2.65}$$

$$\frac{\partial E}{\partial T_{K+1}^m} = 0 \Leftrightarrow$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{l=1}^{L}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu Z_l^\mu T_{K+1}^\mu - U_{i,j,l,K+1}\right] X_i^m Y_j^m Z_l^m = 0 \Leftrightarrow$$

$$T_{K+1}^m = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{l=1}^{L} U_{i,j,l,K+1} X_i^m Y_j^m Z_l^m - \sum_{\mu=1}^{m-1}\left[T_{K+1}^\mu \sum_{i=1}^{I}\left(X_i^\mu X_i^m\right)\sum_{j=1}^{J}\left(Y_j^\mu Y_j^m\right)\sum_{j=1}^{J}\left(Z_l^\mu Z_l^m\right)\right]}{\sum_{i=1}^{I}\left(X_i^m\right)^2 \sum_{j=1}^{J}\left(Y_j^m\right)^2 \sum_{l=1}^{L}\left(Z_l^m\right)^2} \tag{2.66}$$

$$\frac{\partial E}{\partial T_k^m} = 0 \Leftrightarrow$$

$$w\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{l=1}^{L}\left[\sum_{\mu=1}^{m} X_i^\mu Y_j^\mu Z_l^\mu T_k^\mu - \sum_{\mu=1}^{M} \widetilde{X}_i^\mu \widetilde{Y}_j^\mu \widetilde{Z}_l^\mu \widetilde{T}_k^\mu\right] X_i^m Y_j^m Z_l^m = 0 \Leftrightarrow$$

$$T_k^m = \frac{\sum_{\mu=1}^{M}\left[\widetilde{T}_k^\mu \sum_{i=1}^{I}\left(\widetilde{X}_i^\mu X_i^m\right)\sum_{j=1}^{J}\left(\widetilde{Y}_j^\mu Y_j^m\right)\sum_{l=1}^{L}\left(\widetilde{Z}_l^\mu Z_l^m\right)\right]}{\sum_{i=1}^{I}\left(X_i^m\right)^2 \sum_{j=1}^{J}\left(Y_j^m\right)^2 \sum_{l=1}^{L}\left(Z_l^m\right)^2} - \frac{\sum_{\mu=1}^{m-1}\left[T_k^\mu \sum_{i=1}^{I}\left(X_i^\mu X_i^m\right)\sum_{j=1}^{J}\left(Y_j^\mu Y_j^m\right)\sum_{l=1}^{Z}\left(Z_l^\mu Z_l^m\right)\right]}{\sum_{i=1}^{I}\left(X_i^m\right)^2 \sum_{j=1}^{J}\left(Y_j^m\right)^2 \sum_{l=1}^{L}\left(Z_l^m\right)^2} \tag{2.67}$$

In equations (2.63)-(2.67), the desired bases are coupled, so they can be solved by an internal iterative technique. The initialization of this technique is as follows: the field at the first time step is considered a steady field (the time variable is neglected) and approximated by the standard PGD, and the temporal bases are set equal to 1. This strategy is explained by:

$$U'(x,y,z) \equiv U(x,y,z,t=1)$$

$$U'(x,y,z) \approx \sum_{\mu=1}^{M} X^\mu(x) Y^\mu(y) Z^\mu(z)$$

and the bases X_i^μ, Y_j^μ, Z_l^μ and T_1^μ are calculated by:

$$X_i^m = \frac{\sum_{j=1}^{j}\sum_{l=1}^{L} U_{i,j,l,1} Y_j^m Z_l^m bz_l by_j - \sum_{\mu=1}^{m-1} X_i^\mu \sum_{j=1}^{J} Y_j^m Y_j^\mu by_j \sum_{l=1}^{L} Z_l^m Z_l^\mu bz_l}{\sum_{j=1}^{J} \left(Y_j^m\right)^2 by_j \sum_{l=1}^{L} \left(Z_l^m\right)^2 bz_l} \tag{2.68}$$

$$Y_j^m = \frac{\sum_{i=1}^{I}\sum_{l=1}^{L} U_{i,j,l,1} X_i^m Z_l^m bz_l bx_i - \sum_{\mu=1}^{m-1} Y_j^\mu \sum_{i=1}^{I} X_i^m X_i^\mu bx_i \sum_{l=1}^{L} Z_l^m Z_l^\mu bz_l}{\sum_{i=1}^{I} \left(X_i^m\right)^2 bx_i \sum_{l=1}^{L} \left(Z_l^m\right)^2 bz_l} \tag{2.69}$$

$$Z_l^m = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} U_{i,j,l,1} X_i^m Y_j^m by_j bx_i - \sum_{\mu=1}^{m-1} Z_l^\mu \sum_{i=1}^{I} X_i^m X_i^\mu bx_i \sum_{j=1}^{J} Y_j^m Y_j^\mu by_j}{\sum_{i=1}^{I} \left(X_i^m\right)^2 bx_i \sum_{j=1}^{J} \left(Y_j^m\right)^2 by_j} \tag{2.70}$$

$$T_1^m = 1 \tag{2.71}$$

From equations (2.68) - (2.71), the ones that compute the spatial bases X_i^μ, Y_j^μ and Z_l^μ are the discretized form of equations (2.15) - (2.17). The discretization technique used here for the integrals is the Trapezoidal Integration method (see ref. [5]), so the following integration factors appear:

$$bx_i = \begin{cases} \frac{1}{2}, & i = 1 \\ 1, & i = 2, .., I-1 \\ \frac{1}{2}, & i = I \end{cases} \tag{2.72a}$$

$$by_j = \begin{cases} \frac{1}{2}, & j = 1 \\ 1, & j = 2, .., J-1 \\ \frac{1}{2}, & j = J \end{cases} \tag{2.72b}$$

$$bz_l = \begin{cases} \frac{1}{2}, & l = 1 \\ 1, & l = 2, .., L-1 \\ \frac{1}{2}, & l = L \end{cases} \tag{2.72c}$$

Numerical example

The 3D unsteady field $u(x, y, z, t)$ is given by the function:

$$u(x, y, z, t) = \cos\left(6\pi \frac{t}{T_{int}}\right) \sin\left(2\pi \frac{t}{T_{int}} - 4\pi x\right) \cos\left(2\pi \frac{t}{T_{int}} - 1.6\pi y\right) \cos\left(2\pi \frac{t}{T_{int}} - 2.8\pi z\right) \tag{2.73}$$

Both x, y and z are spatial variables, their total length is equal to 1 and are discretized in $I = 10$, $J = 10$ and $L = 10$ nodes respectively. The studied time interval is equal to $T_{int} = 0.01$ and is divided in 40 time steps.

The approximation of the discrete form of field (2.73) is:

$$u_{i,j,l,k} \approx \sum_{\mu=1}^{M} X_i^\mu Y_j^\mu Z_l^\mu T_k^\mu \tag{2.74}$$

The approximation deviation is quantified by the total relative error (%), defined as:

$$\mathcal{E}(M) = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{l=1}^{L}\sum_{k=1}^{K}\left(U_{i,j,l,k} - \sum_{\mu=1}^{M} X_i^\mu Y_j^\mu Z_l^\mu T_k^\mu\right)^2}{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{l=1}^{L}\sum_{k=1}^{K} U_{i,j,l,k}^2} \tag{2.75}$$

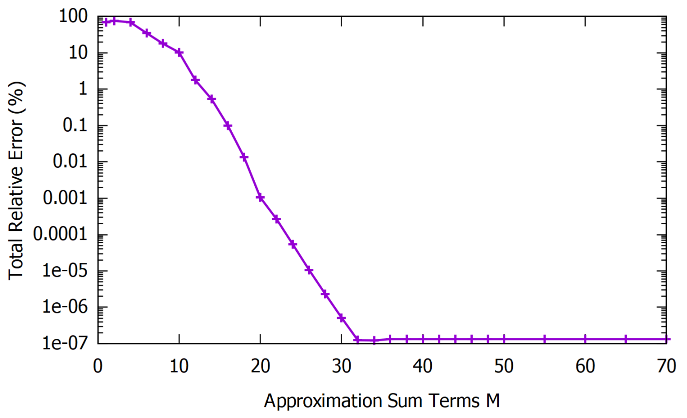


Figure 2.12: *Compression of 3D unsteady field* (2.73) *by iPGD. The relation between total relative error (%) (equation 2.75) with the number of terms M of the approximation sum.*

Obviously, the approximation accuracy is strongly depending on the number of terms M of the approximation sum. This dependence is presented in figure 2.12 where the accuracy is represented by the value of total relative error (%) (equation (2.75)). This figure shows that 16 terms are needed in order the error to be less than 1% and, for an error less than 0.001%, 20 terms are enough. For more than 32 terms the error has reached a minimum at $1 \cdot 10^{-7}\%$ and adding more terms does not decrease this error.

In Appendix B, figures B.1 - B.20 shows the iso-u curves of the original field and the iPGD approximations. This figures make it clear that for $M = 32$ the two groups of curves coincide, while, even for $M = 24$ (where the error is $1 \cdot 10^{-4}\%$) when they are not, they behave similarly.

Thus, for $M = 24$ the storage percent reduction is computed as:

$$\frac{Q_1 - Q_2}{Q_1} = \frac{I \cdot J \cdot L \cdot K - M(I + J + L + K)}{I \cdot J \cdot L \cdot K} = \frac{10 \cdot 10 \cdot 10 \cdot 40 - 24(10 + 10 + 10 + 40)}{10 \cdot 10 \cdot 10 \cdot 40} = 95.8\%$$

Main conclusion made after all these numerical examples is that the Incremental PGD can efficiently approximate multidimensional unsteady fields with insignificant storage requirements. In the next chapter, iPGD is used to compress the flow fields in fluid mechanics applications.

Chapter 3

Software Acceleration

In a previous diploma thesis on the same subject (ref. [2]), the major disadvantage of the iPGD method, in the code programmed by that time, was that it was very slow when used to approximate large-scale unsteady fields. Changes made to the code developed in the present diploma thesis made it run much faster. In the numerical example of the previous chapter, the improved method approximates the 2D unsteady field by a sum of 60 terms almost 10 times faster than the previous version.

The greatest part of this acceleration is due to the use of summation identities (relations 2.34). When these identities are applied in the iPGD equations, nested loops (that model the multiple sums) of the software are replaced by loops in row (that model products of simple sums).

In addition to the aforementioned modifications, software acceleration is also achieved with some additional in-code design, the most important of which are described below.

Suppose that the 3D unsteady field $u(x, y, z, t)$ is to be approximated and the K first time steps have already been compressed. Also, assume that in this approximation process the only thing defined at present is the original field at time step $K + 1$ ($U_{i,j,l,K+1}$) and the bases of the up-to-now approximation (k=1, ..., K) of the previous time series ($\widetilde{X}_i^\mu$, $\widetilde{Y}_j^\mu$, $\widetilde{Z}_l^\mu$ and $\widetilde{T}_k^\mu$, for all i, j, l and $\mu = 1, ..., M$). In the m-th cycle, bases X_i^m for $i = 1, ..., I$, Y_j^m for $j = 1, ..., J$, Z_l^m for $l = 1, ..., L$, T_k^m for $k = 1, ..., K$ and T_{K+1}^m will be found by solving iteratively equations (2.63) to (2.67). At this point, the important part is how these bases are computed inside only one iteration, so the iteration index (p) will be neglected. The computations during one iteration can be separated in 5 steps.

- **Step 1st:** Computation of basis X_i^m

Basis X_i^m, for $i = 1, ..., I$ is computed by the relation:

$$X_i^m = \frac{T_{K+1}^m \sum_{j=1}^{J} \sum_{l=1}^{L} U_{i,j,l,K+1} Y_j^m Z_l^m + w \sum_{\mu=1}^{M} \left[\widetilde{X}_i^\mu \sum_{j=1}^{J} \left(\widetilde{Y}_j^\mu Y_j^m \right) \sum_{l=1}^{L} \left(\widetilde{Z}_l^\mu Z_l^m \right) \sum_{k=1}^{K} \left(\widetilde{T}_k^\mu T_k^m \right) \right]}{\left(T_{K+1}^m\right)^2 \sum_{j=1}^{J} \left(Y_j^m\right)^2 \sum_{l=1}^{L} \left(Z_l^m\right)^2 + w \sum_{j=1}^{J} \left(Y_j^m\right)^2 \sum_{l=1}^{L} \left(Z_l^m\right)^2 \sum_{k=1}^{K} \left(T_k^m\right)^2}$$

$$- \frac{\sum_{\mu=1}^{m-1} \left\{ X_i^\mu \left[T_{K+1}^\mu T_{K+1}^m + w \sum_{k=1}^{K} \left(T_k^\mu T_k^m \right) \right] \sum_{j=1}^{J} \left(Y_j^\mu Y_j^m \right) \sum_{l=1}^{L} \left(Z_l^\mu Z_l^m \right) \right\}}{\left(T_{K+1}^m\right)^2 \sum_{j=1}^{J} \left(Y_j^m\right)^2 \sum_{l=1}^{L} \left(Z_l^m\right)^2 + w \sum_{j=1}^{J} \left(Y_j^m\right)^2 \sum_{l=1}^{L} \left(Z_l^m\right)^2 \sum_{k=1}^{K} \left(T_k^m\right)^2}$$

There is not any term in the denominator containing the X_i, so the terms $\sum_{j=1}^{J} \left(Y_j^m\right)^2$, $\sum_{l=1}^{L} \left(Z_l^m\right)^2$ and $\sum_{k=1}^{K} \left(T_k^m\right)^2$ are computed only once and used for every i. The numerator is computed separately for every i and, divided by the constant denominator, gives the corresponding X_i^m.

- **Step 2nd:** Computation of basis Y_i^m

Basis Y_j^m, for $j = 1, ..., J$ can be found by the relation:

$$Y_i^m = \frac{T_{K+1}^m \sum_{i=1}^{I} \sum_{l=1}^{L} U_{i,j,l,K+1} X_i^m Z_l^m + w \sum_{\mu=1}^{M} \left[\widetilde{Y}_j^\mu \sum_{i=1}^{I} \left(\widetilde{X}_i^\mu X_i^m \right) \sum_{l=1}^{L} \left(\widetilde{Z}_l^\mu Z_l^m \right) \sum_{k=1}^{K} \left(\widetilde{T}_k^\mu T_k^m \right) \right]}{\left(T_{K+1}^m\right)^2 \sum_{i=1}^{I} \left(X_I^m\right)^2 \sum_{l=1}^{L} \left(Z_l^m\right)^2 + w \sum_{i=1}^{I} \left(X_i^m\right)^2 \sum_{l=1}^{L} \left(Z_l^m\right)^2 \sum_{k=1}^{K} \left(T_k^m\right)^2}$$

$$- \frac{\sum_{\mu=1}^{m-1} \left\{ Y_j^\mu \left[T_{K+1}^\mu T_{K+1}^m + w \sum_{k=1}^{K} \left(T_k^\mu T_k^m \right) \right] \sum_{i=1}^{I} \left(X_i^\mu X_i^m \right) \sum_{l=1}^{L} \left(Z_l^\mu Z_l^m \right) \right\}}{\left(T_{K+1}^m\right)^2 \sum_{i=1}^{I} \left(X_I^m\right)^2 \sum_{l=1}^{L} \left(Z_l^m\right)^2 + w \sum_{i=1}^{I} \left(X_i^m\right)^2 \sum_{l=1}^{L} \left(Z_l^m\right)^2 \sum_{k=1}^{K} \left(T_k^m\right)^2}$$

Similarly, the denominator is not depending on Y_j and, moreover, the terms $\sum_{l=1}^{L} \left(Z_l^m\right)^2$ and $\sum_{k=1}^{K} \left(T_k^m\right)^2$ are defined in the previous step. So, in order to compute the denominator here only $\sum_{i=1}^{I} \left(X_i^m\right)^2$ needs to be computed, only once.

The numerator is computed for each and every j and, then, the corresponding Y_j^m

results by a division.

- **Step 3rd:** Computation of basis Z_l^m

Basis Z_l^m, for $l = 1, ..., L$ is computed by the relation:

$$Z_i^m = \frac{T_{K+1}^m \sum_{i=1}^{I} \sum_{j=1}^{J} U_{i,j,l,K+1} X_i^m Y_j^m + w \sum_{\mu=1}^{M} \left[\widetilde{Z}_l^\mu \sum_{i=1}^{I} \left(\widetilde{X}_i^\mu X_i^m \right) \sum_{j=1}^{J} \left(\widetilde{Y}_j^\mu Y_j^m \right) \sum_{k=1}^{K} \left(\widetilde{T}_k^\mu T_k^m \right) \right]}{\left(T_{K+1}^m \right)^2 \sum_{i=1}^{I} \left(X_I^m \right)^2 \sum_{l=1}^{L} \left(Y_j^m \right)^2 + w \sum_{i=1}^{I} \left(X_i^m \right)^2 \sum_{j=1}^{J} \left(Y_j^m \right)^2 \sum_{k=1}^{K} \left(T_k^m \right)^2}$$
$$- \frac{\sum_{\mu=1}^{m-1} \left\{ Z_l^\mu \left[T_{K+1}^\mu T_{K+1}^m + w \sum_{k=1}^{K} \left(T_k^\mu T_k^m \right) \right] \sum_{i=1}^{I} \left(X_i^\mu X_i^m \right) \sum_{j=1}^{J} \left(Y_j^\mu Y_j^m \right) \right\}}{\left(T_{K+1}^m \right)^2 \sum_{i=1}^{I} \left(X_I^m \right)^2 \sum_{l=1}^{L} \left(Y_j^m \right)^2 + w \sum_{i=1}^{I} \left(X_i^m \right)^2 \sum_{j=1}^{J} \left(Y_j^m \right)^2 \sum_{k=1}^{K} \left(T_k^m \right)^2}$$

Similarly, the denominator does not contain any Z_l term, so it can be computed only once and used for every $l = 1, ..., L$. Also, terms $\sum_{i=1}^{I} \left(X_i^m \right)^2$ and $\sum_{k=1}^{K} \left(T_k^m \right)^2$ are defined in the two previous steps, and in order to compute the denominator here only the term $\sum_{j=1}^{J} \left(Y_j^m \right)^2$ needs to be computed and just once. This term could not be considered already defined from the first step, because in the second step the basis Y_j^m is redefined. The numerator is computed for each and every l and after that every Z_l^m derives by division.

- **Step 4th:** Computation of T_{K+1}^m

The value of T_{K+1}^m is computed by the relation:

$$T_{K+1}^m = \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{l=1}^{L} U_{i,j,l,K+1} X_i^m Y_j^m Z_l^m - \sum_{\mu=1}^{m-1} \left[T_{K+1}^\mu \sum_{i=1}^{I} \left(X_i^\mu X_i^m \right) \sum_{j=1}^{J} \left(Y_j^\mu Y_j^m \right) \sum_{j=1}^{J} \left(Z_l^\mu Z_l^m \right) \right]}{\sum_{i=1}^{I} \left(X_i^m \right)^2 \sum_{j=1}^{J} \left(Y_j^m \right)^2 \sum_{l=1}^{L} \left(Z_l^m \right)^2}$$

Also in this step, terms $\sum_{i=1}^{I} \left(X_i^m \right)^2$ and $\sum_{j=1}^{J} \left(Y_j^m \right)^2$ are defined in the two previous steps and in order to compute the denominator here only the (unknown after the second step) term $\sum_{l=1}^{L} \left(Z_l^m \right)^2$ needs to be computed and just once.

Then, the terms of the numerator are computed and by division results the value of T_{K+1}^m.

- **Step 5th:** Computation of basis T_k^m

The basis T_k^m, for $k = 1, ..., K$ is computed by the relation:

$$T_k^m = \frac{\sum_{\mu=1}^{M} \left[\widetilde{T}_k^\mu \sum_{i=1}^{I} \left(\widetilde{X}_i^\mu X_i^m \right) \sum_{j=1}^{J} \left(\widetilde{Y}_j^\mu Y_j^m \right) \sum_{l=1}^{L} \left(\widetilde{Z}_l^\mu Z_l^m \right) \right]}{\sum_{i=1}^{I} (X_i^m)^2 \sum_{j=1}^{J} \left(Y_j^m \right)^2 \sum_{l=1}^{L} (Z_l^m)^2} - \frac{\sum_{\mu=1}^{m-1} \left[T_k^\mu \sum_{i=1}^{I} (X_i^\mu X_i^m) \sum_{j=1}^{J} \left(Y_j^\mu Y_j^m \right) \sum_{l=1}^{Z} (Z_l^\mu Z_l^m) \right]}{\sum_{i=1}^{I} (X_i^m)^2 \sum_{j=1}^{J} \left(Y_j^m \right)^2 \sum_{l=1}^{L} (Z_l^m)^2}$$

Here, the denominator is not only constant for every $k = 1, ..., K$ and independent of basis T_k, but is also defined in the previous step. So, in order to compute basis T_k^m only the computation of the numerators and the division with the already computed denominator is required.

It is worth noting that the above steps can also be implemented in standard PGD to accelerate it, whether it is a separate software or for the initialization of the iPGD construction.

Chapter 4

The Incremental PGD in Fluid Mechanics Problems

In this chapter, unsteady multidimensional fluid flows are solved and then the flow time-series are approximated (compressed) by the Incremental PGD method. The purpose of this chapter is to illustrate the use of this method in CFD problems and the accuracy of the corresponding approximations.

4.1 Compression of unsteady flow fields through a 2D S-shaped pipe

In this section, the flow through a 2D S-shaped pipe is solved and then approximated by iPGD method. A structured grid of $221 \times 111 = 24531$ nodes is used. Figure 4.1 shows the computational space and the structured grid.

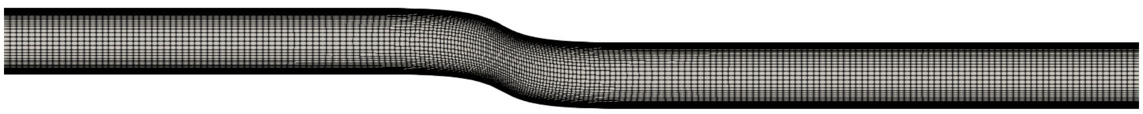


Figure 4.1: *Flow through a 2D S-shaped pipe. Computational space and the structured grid.*

The computational software used to compute the flow field is called PUMA (Parallel

Unstructured Multi-row Adjoint) and is developed by the PCopt/NTUA. PUMA is a GPU-enabled Reynolds-Averaged Navier-Stokes equations' solver using the Spalart-Allmaras turbulence model. The solver can use structured or unstructured meshes on which the governing equations are discretized by the finite volume technique with vertex-centered storage of the flow variables (see ref. [6]). PUMA computations are performed on a NVIDIA-SMI 346.46 GPU. The iPGD software enables only one CPU, the Intel(R) Xeon(TM) CPU 2.66GHz. The flow fields computed directly from PUMA are ρ, ρu, ρv, ρw and ρE, where ρ is the density, u, v and w are the velocities in x, y and z directions respectively (in this case $w = 0$), and E is the total Energy. The velocity fields u, v, w can be found by division and the pressure field P by the simple:

$$\rho E = \frac{P}{\gamma - 1} + \frac{1}{2}\rho \left[\frac{(\rho u)^2}{\rho^2} + \frac{(\rho v)^2}{\rho^2} + \frac{(\rho w)^2}{\rho^2}\right] \tag{4.1}$$

In this case study, the compressible fluid chosen is air (perfect gas) with gas constant $R = 287\ J/kg \cdot grad$, heat capacity ratio $\gamma = 1.4$ and dynamic viscosity $\mu = 1.716 \times 10^{-5}\ kg/ms$. At the inlet of the pipe there is constant total pressure $P_{tot} = 101325$ Pascal and total temperature $T_{tot} = 288\ K$. The gas enters the pipe with zero angle in x-direction. This 2D flow inside the pipe is unsteady because of the time dependent pressure at the outlet of the pipe, which value is given in Pascal by the periodic function 4.2 for $A = 101000$ Pascal and studied time interval equal to one period $T_{int} = 0.01$ seconds.

$$P_{out}(t) = A + (101325 - A)\sin\left(2\pi\frac{t}{T_{int}}\right)\cos\left(4\pi\frac{t}{T_{int}}\right)\cos\left(\pi\frac{t}{T_{int}} + 0.2\pi\right) \tag{4.2}$$

This period T_{int} is divided in 21 equal steps. Figure 4.2 shows the values of outlet pressure, given by equation 4.2, for all the discrete 21 moments of a period.

The flow fields ρ, ρu, ρv and ρE are compressed by $M = 10$, 50, 70 and 100 summation terms. The total relative error (%) (eq. 2.54) is computed for every field and presented in figure 4.3. In this figure, it is shown that the approximation of ρ and ρE fields is performed much more accurately than approximating ρu and ρv.

In Appendix C, the following properties of the flow can be found (as coloured maps): density ρ, pressure P, velocity u in x-direction and velocity v in y-direction at the 1st, the 7th, the 13th and the 19th time step. In figures C.1 to C.16 the original field is compared with the iPGD approximations for $M = 10$, 50, 70 and 100. In these figures, it is clear that even for $M = 50$ the flow fields are very similar to the recorded data by CFD analysis, while for $M = 100$ approximated solutions are exactly the same with the reference data. For $M = 100$, the achieved storage percent reduction is:

$$\frac{Q_1 - Q_2}{Q_1} = \frac{I \cdot J \cdot K - M\,(I + J + K)}{I \cdot J \cdot K} = \frac{221 \cdot 111 \cdot 21 - 100\,(221 + 111 + 21)}{221 \cdot 111 \cdot 21} = 93.1\%$$

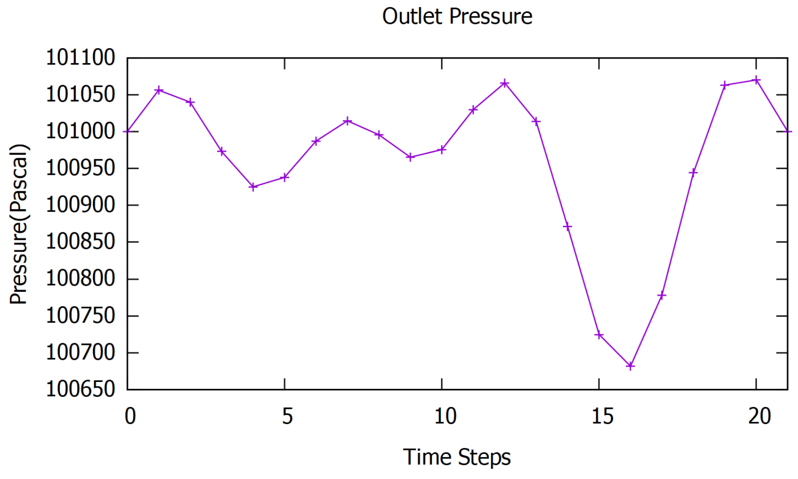


Figure 4.2: *Flow through a 2D S-shaped pipe. The outlet pressure, given by equation 4.2, for all the* 21 *moments of a time period.*

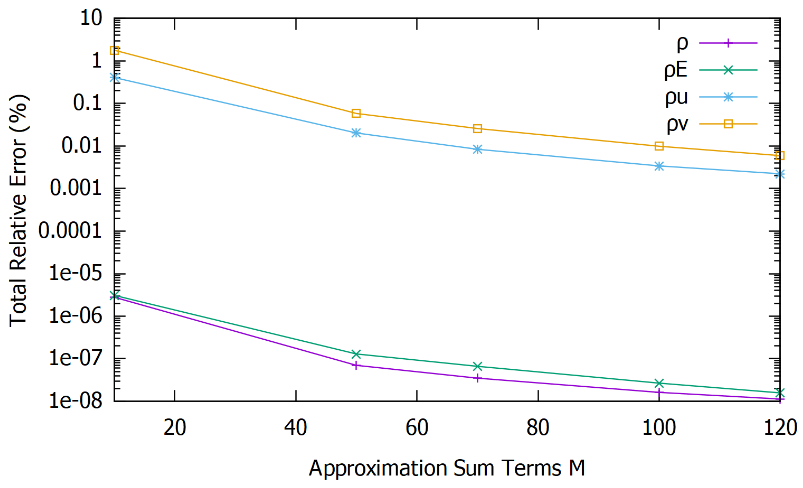


Figure 4.3: *Flow through a 2D S-shaped pipe. Total relative error (%) (eq. 2.54) of approximating the ρ, ρu, ρv and ρE for $M = 10,\ 50,\ 70$ and* 100.

4.2 Compression of unsteady flow fields through a 3D U-shaped pipe

In this section, the unsteady flow through a 3D U-shaped pipe (with square cross section) is computed and then the flow property fields are compressed by the iPGD. PUMA software is solving the fluid equations on the $233 \times 51 \times 51 = 606033$ nodes of the structured grid. Figure 4.4 shows the external surface of the computational volume and the structured grid on it.

The air of the S-bend case is flowing through this pipe with exactly the same properties. Conditions at the inlet of the pipe are given total pressure $P_{tot} = 101325$

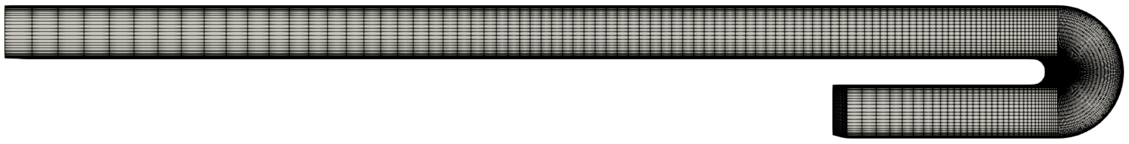


Figure 4.4: *Flow through a 3D U-shaped pipe. Computational space and structured grid of the U-shaped with square cross section pipe.*

Pascal and total temperature $T_{tot} = 288\ K$. The gas enters the pipe with zero angles to both directions. At the pipe outlet, pressure is defined in Pascal by the periodic function (4.3) for $A = 101000$ Pascal and period $T_{int} = 0.01$ seconds. This period is divided in 51 equal time steps.

$$P_{out}(t) = A - (101325 - A)\sqrt{\sin\left(2\pi \frac{t}{T_{int}}\right)} \cos\left(4\pi \frac{t}{T_{int}}\right) \cos\left(\pi \frac{t}{T_{int}} + 0.2\pi\right) \quad (4.3)$$

Figure 4.5 shows the value of the outlet pressure, given by equation (4.3) for a time period.

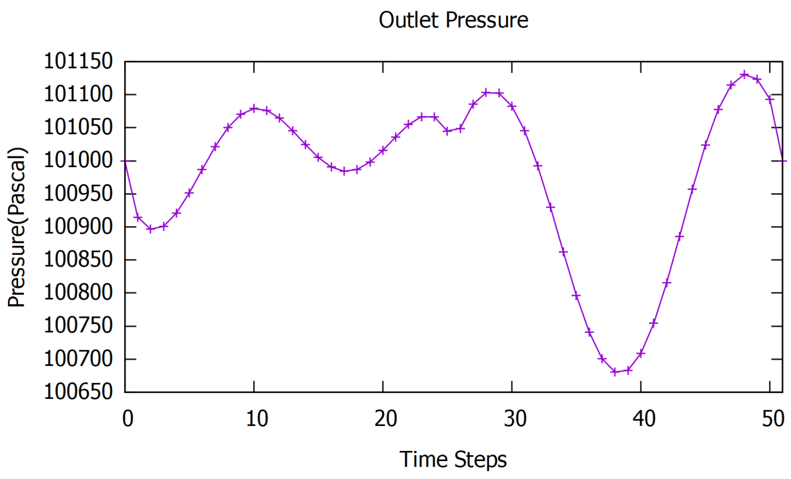


Figure 4.5: *Flow through a 3D U-shaped pipe. The outlet pressure, given by equation* (4.3) *during a time period.*

As already mentioned in previous chapters, the accuracy of iPGD approximations is strongly depending on the number of terms of the approximation sum. In figures 4.6, the approximation accuracy is quantified by the relative total error (%), defined by equation 2.75 and computed for various summation terms M. The compressed flow fields, the errors of which are presented in figures 4.6, are the properties taken directly from PUMA (ρ, ρu, ρv, ρw and ρE). Figures 4.6 shows that approximating fields ρ and ρE can be achieved accurately enough with small M. Even for $M = 10$, the total error on ρ is less than 0.002%, while the total error on ρE is less than

$6 \cdot 10^5\%$. On the contrary, ρu and ρv need at least $M = 50$ for a total error around 5% and for $M = 200$ (where the error curves are almost flat) the total error is around 1.3%. The ρw field is the most difficult to approximate. For $M = 50$, the total error is around 10.5% while even for $M = 220$ the error is not less than 2.9%.

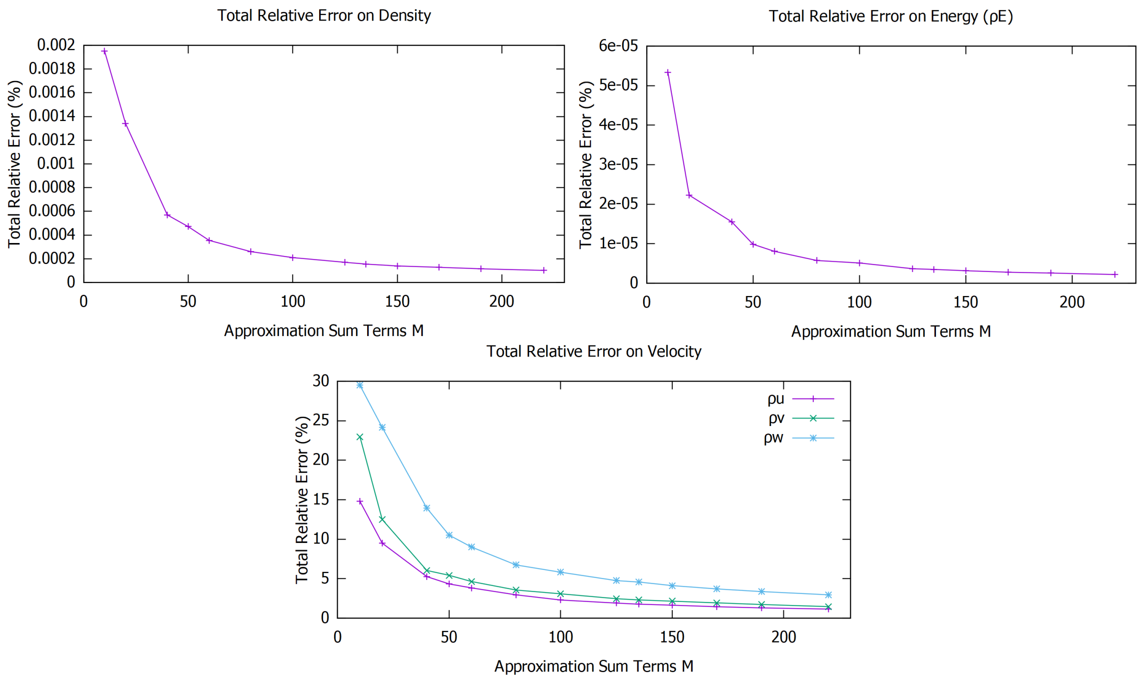


Figure 4.6: *Flow through a 3D U-shaped pipe. Total Relative Error, defined by eq. 2.75 for various approximation summation terms.*

Figures (D.1) to (D.16) in Appendix D present the five original flow fields (density ρ, pressure P and velocities u, v and w) and two corresponding approximations constructed by 135 and 200 terms. The case of $M = 135$ was chosen because compressing a field at one time step needs exactly the same computational time that PUMA needs to solve the fields of one time step (9 minutes). Figures 4.6 show that the accuracy of this case is very satisfying. This can be noticed also in the figures of the appendix D. The case of $M = 200$ was chosen because it represents the best possible approximation. From figures 4.6 it seems that (around that values of M) the curves are (almost) flat and this means that adding more terms will not further improve the accuracy. The figures in the appendix verify that the compressed fields of this case are the same with the original ones.

The storage percent reduction of the iPGD approaches are:
for the $M = 135$ case

$$\frac{Q_1 - Q_2}{Q_1} = \frac{I \cdot J \cdot L \cdot K - M\,(I + J + L + K)}{I \cdot J \cdot L \cdot K} =$$

$$= \frac{233 \cdot 51 \cdot 51 \cdot 51 - 135\,(233 + 51 + 51 + 51)}{233 \cdot 51 \cdot 51 \cdot 51} = 99,83\%$$

for the $M = 200$ case

$$\frac{Q_1 - Q_2}{Q_1} = \frac{I \cdot J \cdot L \cdot K - M\,(I + J + L + K)}{I \cdot J \cdot L \cdot K} =$$

$$= \frac{233 \cdot 51 \cdot 51 \cdot 51 - 200\,(233 + 51 + 51 + 51)}{233 \cdot 51 \cdot 51 \cdot 51} = 99,75\%$$

Practically, the storage requirements are eliminated in both cases.

The speed of approximating the fields at each time step, as well as the reduction of storage requirements, are decreasing functions of the used number of approximation sum terms M. On the contrary, the accuracy of the approach is an increasing function of M (see also figures 4.6), so it is of great importance to find where these three conflicting objectives are sufficiently met.

PUMA software needs 9 minutes to solve the ρ, u, v, w and P fields at every time step. The process of compressing one of these fields is not related with any of the others, so the approximations of the five fields at every time step can be constructed simultaneously. If these fields are compressed by the same M, their approximations will demand exactly the same time. So, if the five fields are compressed simultaneously (thus, on 5 CPUs), the time needed is of a single approximation, but the computational cost is five times higher. The flow fields at the ν-th time step can be compressed just after they are solved and simultaneously with the computation of the next fields at the $(\nu + 1)$-th time step.

In the approximation case of $M = 135$, compressing a field at a random time step takes exactly the same time with solving the flow fields of this moment (9 minutes). If computation and compression are performed simultaneously as described in the previous paragraph, the whole process needs the same time with approximating one time-series for all the 51 time steps plus the time of computing the flow fields at the 1st time step. This means that the approximation of all the flow problem solution will be available 9 minutes after the computation is over. In contrast with PUMA, iPGD software is not running in parallel, so even if it needs the same time, the computational cost is much lower. PUMA computations are performed on a GPU that contains a lot of processors and these iPGD compressions are performed on just five CPUs. Figures 4.6 shows that fields ρ and ρE can be approximated accurately enough with less terms. For example, for $M = 80$ terms, the compression of a field at a random time step needs 3 minutes and 25 seconds. During the 9 minutes of the computation, the fields ρ and ρE can be compressed the one after the other, on the same CPU, and thus reducing the computational cost (4 instead of 5 required CPUs).

Previous works on iPGD considered this method deterrently slow but in this chapter it was illustrated how it can approximate unsteady large-scale fields fast and accurately. Thus, iPGD may be considered as an efficient approximative compression method for large-scale unsteady flow solutions.

Chapter 5

Conclusions and Perspectives

To summarize, in this part of the Diploma Thesis, the mathematical formulation and the algorithms of the Incremental Proper Generalized Decomposition method were improved. Three are the main interventions done to this method. Firstly, the correction on the definition of approximation error (2.43), in order to use all the M, and not only m, terms of the up-to-now approximation. The second is the replacement of the several summations (nested loops) with products of simple sums (in-raw single loops) in respect to the identities of summation. The third is the fine design of the algorithm in order to make the least possible computations. Also, this method was extended to compress 3D unsteady problems. The main conclusions of this study are:

- iPGD is indeed improved. The 2D unsteady field 2.55 was compressed 31 times more accurately and 10 times faster than by the previous version of iPGD.
- It was shown in section 4.2 that iPGD can compress with high accuracy large-scale 3D unsteady flow fields minimizing the storage requirements to less than 0.5% of the initial. If computation and compression are performed simultaneously, the approximation is ready just 9 minutes after computation.

There are interesting things to be studied, that are not included in this diploma thesis. Some of them are:

- A valid way to choose the appropriate value of M. Remember that this must be defined a priori. An appropriate M produces accurate approximations with low storage requirements in acceptable computational time.
- Further acceleration of the method, likely by creating software that runs in parallel or enables GPUs.
- Direct coupling with the solver.
- A valid method to choose the appropriate weight factor w, maybe by the time-derivative of the function.

Part II

Molecular Dynamics computations and optimization of coarse-grained Ionic Liquid lubricants

Chapter 6

Introduction to Molecular Dynamics and Lubrication

The following part of this Diploma Thesis resumes a project conducted in the Advanced Material Research division of Toyota Motor Europe (TME). The topic is the use of Ionic Liquids as lubricants. Properties of lubricants are computed during MD simulations and then optimized by EA. In chapter 6, there is a brief presentation of the necessary theoretical background and in chapter 7, calculations and optimizations are conducted and the results are presented and discussed.

6.1 Lubrication

Lubrication is usually applied to a tribological system to mitigate friction and wear by interposing a substance providing reduced shear strength between two surfaces to help to carry the load. Lubrication regimes (see ref.[27]) will differ in accord with the conditions in practice. The three most common beyond them are:

- **Boundary lubrication** occurs when the solid surfaces are so close together that interactions between solid asperities dominate the contact. In this kind of lubrication, friction is increased.
- **Hydrodynamic lubrication** is also known as full-film or thick-film lubrication. The fluid film is thick enough to prevent contact between the solid parts and all the load can be carried by this film. It enables the coupled elements to function without any wear over a long operating period.
- **Mixed lubrication** is an intermediate regime where both previous mechanisms are functioning. There may be frequent solid contact, but some portion of the bearing surface remains supported by a partial hydrodynamic fluid film.

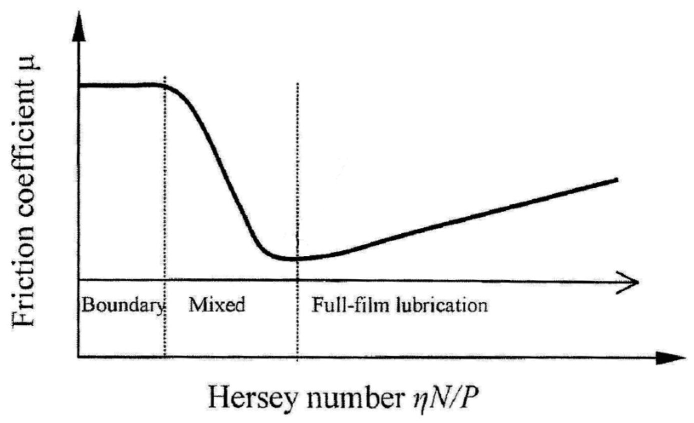


Figure 6.1: *Stribeck curve from ref. [7] and the three lubrication regimes.*

The above lubricant regimes are distinguished in the Stribeck curve. A typical Stribeck curve (ref. [7]) is presented in Figure (6.1). In this curve for a fluid-lubricated system, the friction coefficient (μ) is a function of the Hersey number:

$$\text{Hersey Number} = \eta N/P \quad ,$$

where η is the dynamic viscosity of the lubricant, N is the relative velocity between the parts and P is the normal load applied to the lubricant.

For the hydrodynamic region of figure 6.1, for constant load (P) and velocity (N), decreasing viscosity (η) minimizes friction. But if viscosity is over decreased, lubrication exits hydrodynamic region and friction is increased. So, a situation of minimum viscosity but inside the hydrodynamic region (in order not to have contact between the lubricated parts) is favourable.

Ionic liquids (IL) are neutral (in total) liquids consisted of positive and negative charged molecules called ions (cations and anions respectively).Nowadays, more and more often, IL are proposed as lubricants due to some beneficial properties such as negligible vapor pressure, non-flammability, high thermal-oxidative stability and reasonable viscosity-temperature behaviour (ref. [28]).

At this project, IL are proposed as lubricants for two reasons. The first is that viscosity can easily be tuned by changing the molecular design of the IL. This is how viscosity can be decreased in the hydrodynamic lubrication. The second reason is that when confined in very small gaps (of some Å) they create a strong alternating ion-layers formation capable to carry great amounts of load (ref. [30], [26] and [25]). This "thin-film" can be used as a protection between the two parts and prevent contact on the asperities (ref. [28]). In order to optimize lubricant behaviour of IL the viscosity must be minimized and the force-carrying capability must be maximized.

In order to minimize viscosity, molecular design must be changed. The computation technique that solves total properties by using information from the molecular level is Molecular Dynamics computation. The search for the optimal molecular design of the IL can be performed by Evolutionary Algorithms.

6.2 Molecular Dynamics (MD)

Molecular Dynamics (MD) simulation is a a time-marching computational technique based on Newtonian mechanics. The central idea is that for a given system of particles (atoms, molecules, etc.), Newton's second law ($\mathbf{F} = m\mathbf{a}$) is solved for every particle, and thus each new instantaneous position can be computed. More specifically, a given system is consisted of N particles. For every single particle i with mass m_i, situated at $\mathbf{x}_i$ position, $\mathbf{F}_i$ is the resultant force from every other particle in the system. The Newton's second law is written as:

$$\mathbf{F}_i = m_i \frac{d^2\mathbf{x}_i}{d^2t} \ . \tag{6.1}$$

The material of this section is based on ref. [9] and [10]. Especially the subsections Thermodynamic Ensembles, Controlling Temperature and Controlling Pressure are based also in ref. [11], ref. [12], [13], [14], [15], [16][17] and [18].

Interatomic pairwise potential

The most time-consuming part of almost all MD simulations is the computation of the atomic force of every particle. The atomic force of a particle i is the reaction (opposite) to the resultant of all the forces acting on the particle i from every other particle j. This resultant derives from a potential energy, and the atomic force reads:

$$F_i = -\frac{\vartheta U_i(x_i)}{\vartheta x_i} \ , \tag{6.2}$$

where the potential energy is not a real total potential but represents the addition of pairwise interparticle potentials between all particles as $U_i(x_i) = \sum_{j=1}^{N} u_{ij}$.

Regarding pairwise potentials, the most common and the one used in the present thesis is the Lennard-Jones $12-6$ potential in addition with long range Coulomb interactions. This potential (LJ-C potential) has the following form:

$$u_{ij} = \underbrace{4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right]}_{Lennard-Jones} + \underbrace{\frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}}}_{Coulomb} \ , \tag{6.3}$$

where r_{ij} is the distance between particle i and particle j, ε_{ij} is the depth of the pair-wise LJ potential curve (a measure of how strongly the two particles interact), σ_{ij} is the finite distance at which the inter-particle LJ potential is zero, q_i and q_j are the electric charges of the interacting particles and ε_0 is the vacuum permittivity. All the above terms are properties of the materials in the system and are known in advance, while $r_{ij} = |\mathbf{x}_i - \mathbf{x}_j|$ is computed at every time step. At any time step t, knowing the position x_i of a particle i, all the forces acting on it can be computed by equation 6.2. These forces are used in equation 6.1 to compute the new position x_i of the particle at $t + \Delta t$ time step.

The Lennard-Jones (LJ) potential is an empirical model widely used to describe molecular interactions between neutral particles. The attractive term is assumed to decay with distance as the inverse sixth power (dipole-dipole interactions) and the repulsive term is assumed to decay more rapidly (inverse twelfth power)(see Eq. 6.3). The attractive interactions are due to fluctuating dipoles (van der Waals interactions), and the repulsive interactions are due to overlap of the electron clouds (Pauli exclusion principle) which forces electrons into higher energy states. The form of a Lennard-Jones potential can be found in figure 6.2. In this figure, the Lennard-Jones potential between the Carbon atoms (C) of two Methane molecules (CH_4) is computed for $\varepsilon_{CC} = 0.066\ kcal \cdot mol^{-1}$ and $\sigma_{CC} = 3.5$ Å.

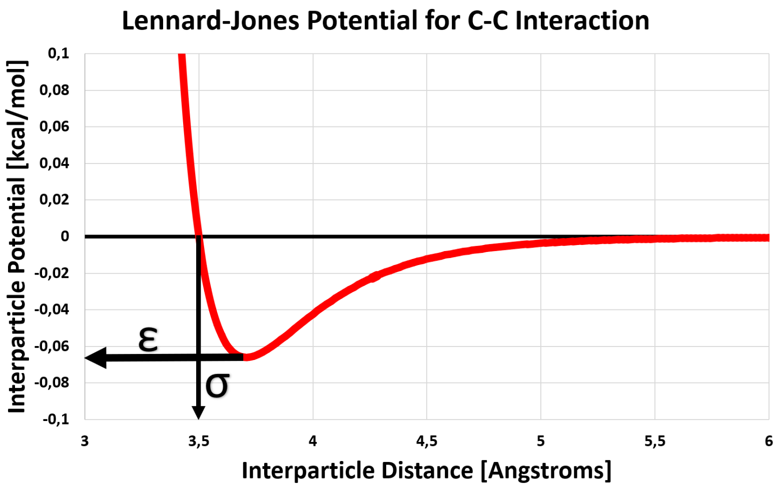


Figure 6.2: *The Lennard-Jones potential between the C atoms of two CH_4 molecules.*

For electrically charged particles, Coulomb's law is the experimental model describing the electrical pairwise interactions. The force of the interaction is attractive between opposite charged atoms and repulsive between like-signed charged atoms.

Verlet Algorithm

After computing all forces between all particles, next step is to compute their new positions. For this purpose, **Verlet Algorithm** uses the Taylor expansion of the position $\mathbf{x}_i$ of a particle i, around an arbitrary moment t, by a small time step Δt. So, the positions of a specific particle i, in time $t + \Delta t$ and $t - \Delta t$ are:

$$\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \frac{\partial \mathbf{x}_i(t)}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 \mathbf{x}_i(t)}{\partial^2 t}\Delta t^2 + \frac{1}{3!}\frac{\partial^3 \mathbf{x}_i(t)}{\partial^3 t}\Delta t^3 + O(\Delta t^4) \quad (6.4a)$$

$$\mathbf{x}_i(t-\Delta t) = \mathbf{x}_i(t) - \frac{\partial \mathbf{x}_i(t)}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 \mathbf{x}_i(t)}{\partial^2 t}\Delta t^2 - \frac{1}{3!}\frac{\partial^3 \mathbf{x}_i(t)}{\partial^3 t}\Delta t^3 + O(\Delta t^4) \quad (6.4b)$$

By adding equations 6.4 and replacing the second derivative of position by Newton's equation 6.1 derives:

$$\mathbf{x}_i(t+\Delta t) = 2\mathbf{x}_i(t) - \mathbf{x}_i(t-\Delta t) + \frac{\mathbf{F}_i}{m_i}\Delta t^2 + O(\Delta t^4) \quad (6.5)$$

In order to estimate the new position of the particle with an error of order Δt^4, only the present and the previous positions are needed.

Energy, Temperature, Pressure

The total energy of a MD system is the addition of potential and kinetic energies. The potential energy derives from the addition of the pairwise potential of all (N) particles and the kinetic energy from the velocities of all particles as:

$$E(t) = U(t) + K(t) = \underbrace{\sum_{i=1}^{N}\sum_{j>i}^{N} v_{ij}}_{Potential\ Energy} \underbrace{+ \frac{1}{2}\sum_{i=1}^{N} m_i \left(\frac{d\mathbf{x}_i}{dt}\right)^2}_{Kinetic\ Energy} \quad (6.6)$$

In Newtonian dynamics the total energy 6.6 is a conserved quantity. As time marches, energy flows back and forth between kinetic and potential energy, causing $K(t)$ and $U(t)$ to fluctuate while their sum remains fixed. In practice, there are some small fluctuations also in total energy caused by errors in time–integration. By monitoring the total energy over time evolution, useful information about the state of the system can be obtained. If total energy does not change over time, it means that the system has reached **equilibrium**. After equilibrium, the microscopic properties (positions, velocities, etc) still change, but the thermodynamic ones (temperature, pressure, heat capacity, etc) remain fixed or fluctuate over a fixed value. The computations of these properties should be performed after equilibrium.

Useful thermodynamic properties of a system are the temperature and pressure. These quantities are measured frequently in experiments and are, thus, important for comparison between experiment and computation. After computing the instantaneous kinetic energy, temperature derives from the equipartition theorem as:

$$T(t) = \frac{2K(t)}{3Nk_B} \ , \tag{6.7}$$

where N is the total number of atoms of the system and k_B is the Boltzmann constant. Pressure can be computed by the virial function of Clausius (ref. [31]) as:

$$P = \frac{Nk_BT}{V} + \frac{1}{3V}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{F}_i \tag{6.8}$$

Temperature and pressure derive from (6.7) and (6.8), after averaging over time.

Thermodynamic Ensembles

A **thermodynamic ensemble** is a collection of a large number of indistinguishable replicas of a studied system, which interact with each other, but are isolated from the rest of the universe. These replicas could be in different microscopic states, as determined by the positions and momenta of the constituent molecules, but the macroscopic state determined by the pressure, temperature and/or other thermodynamic variables are identical. Although a mechanical system certainly evolves over time, the ensemble does not necessarily evolve. A statistical ensemble that does not change over time, can said to be in statistical equilibrium. Some information about common thermodynamic ensembles is given below.

The **micro-canonical (NVE) ensemble** is used to represent the possible states of a mechanical system, in which the Number of particles, the Volume and the total Energy remain constant through time. The equations used in the micro-canonical ensemble have already been presented above. It should be mentioned that NVE is not corresponding to experimentally realistic situations because it is hard to perform measurements that satisfy exactly the requirement of fixed energy.

The **canonical (NVT) ensemble** is the statistical ensemble that represents the possible states of a mechanical system in thermal equilibrium with a large heat bath at a fixed temperature. Practically, this is achieved by introducing a thermostat to the system. The Number of particles, the Volume and the absolute Temperature are fixed during the MD. The system can exchange energy with the heat bath, so that the states of the system will differ in total energy.

In the **isothermal-isobaric (NPT) ensemble**, apart from fixed Number of particles and Temperature, Pressure is also fixed, and thus volume is dynamically changing during the simulation. This can be achieved by introducing a thermostat and a barostat in the system. This ensemble is useful when density must be measured.

In the study cases of the next chapter, MD computations are performed in the NVT and NPT ensembles. The MD software used for these computations is LAMMPS

(Large-scale Atomic/Molecular Massively Parallel Simulator, ref. [29]). In the following subsections there is a brief presentation of the equations solved by LAMMPS during a NVT and NPT ensemble.

Controlling Temperature

In the NVT ensemble, a thermostat determines the equations of motion of particles. In LAMMPS, a chain of M Nosé-Hoover thermostats is implemented. The equations of these thermostats are:

$$\frac{d\mathbf{x}_i}{dt} = \frac{\mathbf{p}_i}{m_i} \tag{6.9a}$$

$$\frac{d\mathbf{p}_i}{dt} = \mathbf{F}_i - \frac{p_{\xi_1}}{Q_1}\mathbf{p}_i \tag{6.9b}$$

$$\frac{d\xi_k}{dt} = \frac{p_{\xi_k}}{Q_k} \tag{6.9c}$$

$$\frac{dp_{\xi_1}}{dt} = \underbrace{\left(\sum_{i=1}^{N}\frac{\mathbf{p}_i^2}{m_i} - 3Nk_BT\right)}_{G_1} - \frac{p_{\xi_2}}{Q_2}p_{\xi_1} \tag{6.9d}$$

$$\frac{dp_{\xi_k}}{dt} = \underbrace{\left[\frac{p_{\xi_{k-1}}^2}{Q_{k-1}} - k_BT\right]}_{G_k} - \frac{p_{\xi_{k+1}}}{Q_{k+1}}p_{\xi_k} \tag{6.9e}$$

$$\frac{dp_{\xi_M}}{dt} = \underbrace{\left[\frac{p_{\xi_{M-1}}^2}{Q_{M-1}} - k_BT\right]}_{G_M} , \tag{6.9f}$$

where $\mathbf{x}_i$, $\mathbf{p}_i$, m_i are, respectively, the position, momentum and mass of particle i, ξ_k (for $k = 1, ..., M$) are the additional degrees of freedom, due to the thermostats, Q_k is a parameter with dimensions of $energy \cdot (time)^2$, determines the scale of temperature fluctuations and behaves as mass for the motion of ξ_k, p_{ξ_k} is the momentum of ξ_k and T is the temperature imposed by the chain of M thermostats. The G_k terms are considered as heat-bath 'forces'.

For the above equations of motion, the conserved quantity is:

$$E_{NVT} = E + \sum_{k=1}^{M}\frac{p_{\xi_k}^2}{2Q_k} + 3Nk_BT\xi_1 + \sum_{k=2}^{M}\xi_k \tag{6.10}$$

where E is the total energy of equation 6.6.

Controlling Pressure

During an isothermal-isobaric simulation, Newtonian equations of motion are replaced by Nosé-Hoover thermostat–barostat equations of motion. In this case, two chains of thermostats are implemented; one is $\{\eta_k, p_{\eta_k}, Q\}$ coupled to the particles and the other $\{\xi_k, p_{\xi_k}, Q'\}$ to the barostat $\{\epsilon, p_\epsilon, W\}$. The ϵ is an extra degree of freedom due to the barostat, defined as $\epsilon = ln(V/V_{t=0})$, p_ϵ is the momentum of ϵ and W has units $energy \cdot (time)^2$ and behaves as mass of the barostat. The equations of motion in this case are:

$$\frac{d\mathbf{x}_i}{dt} = \frac{\mathbf{p}_i}{m_i} + \frac{p_\epsilon}{W}\mathbf{x}_i \tag{6.11a}$$

$$\frac{d\mathbf{p}_i}{dt} = \mathbf{F}_i - \left(1 + \frac{1}{N}\right)\frac{p_\epsilon}{W}\mathbf{p}_i - \frac{p_{\xi_1}}{Q_1}\mathbf{p}_i \tag{6.11b}$$

$$\frac{dV}{dt} = 3V\frac{p_\epsilon}{W} \tag{6.11c}$$

$$\frac{dp_\epsilon}{dt} = 3V(P_{int} - P) + \frac{1}{N}\sum_{i=1}^{N}\frac{\mathbf{p}_i^2}{m_i} - \frac{p_{\xi_1}}{Q'_1}p_\epsilon \tag{6.11d}$$

$$\frac{d\eta_k}{dt} = \frac{p_{\eta_k}}{Q_k} \tag{6.11e}$$

$$\frac{dp_{\eta_k}}{dt} = G_k - \frac{p_{\eta_{k+1}}}{Q_{k+1}}p_{\eta_k} \tag{6.11f}$$

$$\frac{dp_{\eta_M}}{dt} = G_M \tag{6.11g}$$

$$\frac{d\xi_k}{dt} = \frac{p_{\xi_k}}{Q'_k} \tag{6.11h}$$

$$\frac{dp_{\xi_1}}{dt} = \underbrace{\frac{p_\epsilon^2}{W} - k_B T}_{G'_k} - \frac{p_{\xi_2}}{Q'_2}p_{\xi_k} \tag{6.11i}$$

$$\frac{dp_{\xi_k}}{dt} = \underbrace{\frac{p_{\xi_{k-1}}^2}{Q'_{k-1}} - k_B T}_{G'_k} - \frac{p_{\xi_{k+1}}}{Q'_{k+1}}p_{\xi_k} \tag{6.11j}$$

$$\frac{dp_{\xi_M}}{dt} = \underbrace{\frac{p_{\xi_{M-1}}^2}{Q'_{M-1}} - k_B T}_{G'_M}\ , \tag{6.11k}$$

where P_{int} is the instantaneous internal pressure and can be computed as:

$$P_{int} = \frac{1}{3V}\left[\sum_{i=1}^{N}\left(\frac{\mathbf{p}_i^2}{m_i} + \mathbf{x}_i \cdot \mathbf{F}_i\right) - 3V\frac{\partial U}{\partial V}\right] \tag{6.12}$$

In this case, the conserved quantity is:

$$E_{NPT} = E + \frac{p_\epsilon^2}{2W} + PV + \sum_{k=1}^{M} \left(\frac{p_{\eta_k}^2}{2Q_k} + \frac{p_{\xi_k}^2}{2Q'_k} \right) + 3Nk_BT\eta_1 + k_BT\sum_{k=2}^{M}\eta_k + k_BT\sum_{k=1}^{M}\xi_k \tag{6.13}$$

where E is the total energy of equation 6.6.

Boundary Conditions

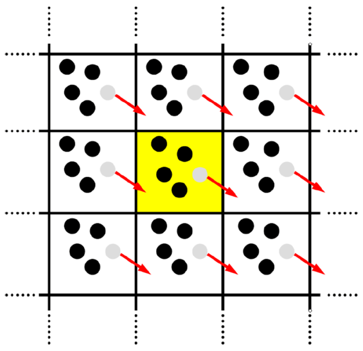


Figure 6.3: *2D Simulation box with periodic boundaries in all directions. Figure taken from ref. [19].*

In order to represent bulk liquid in MD simulations the computational volume must have **periodic boundaries**. This means that a particle can exit the box from a point of the one side of the boundary with a specific velocity and re-enter from the symmetric (with respect to the center of the box) point of the opposite side of the boundary with the same velocity (see Figure 6.3). It also means that a particle close to a border interact with its neighbours inside the box and, also, with fictitious particles outside of the box – replicas of atoms close to the opposite border and inside the box. When using periodic boundaries the original volume is called unit cell and all the others are called images of this unit cell.

In a case in which the liquid is confined between walls, it is more realistic to use **fixed boundaries** in order not to have interactions with replicas of the whole geometry. In most cases, it is convenient to use fixed boundaries only in the vertical-to-walls direction, and periodic to the other(s).

Chapter 7

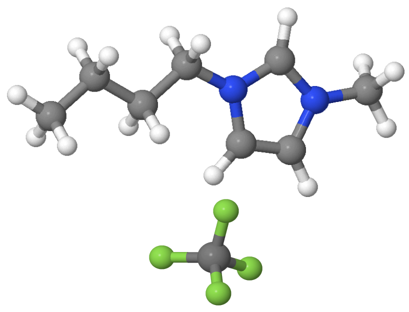
Calculations of Ionic Liquid lubricant properties

As already mentioned, in this diploma thesis Ionic Liquids (IL) are proposed as lubricants because of the strong ion-layers formation when confined and the molecular design freedom to tune viscosity. Before optimizing this properties, the MD algorithms that compute them must be verified. To do that, these algorithms are used for the properties of a specific ionic liquid with known properties. The chosen IL is the **1-Butyl-3-methylimidazolium tetrafluoroborate** ([BMIM$^+$][BF$_4$$^-$]) with chemical formula $C_8H_{15}BF_4N_2$ and molecular mass 226.04 $gr{\cdot}mol^{-1}$ and is presented in Figure 7.1a.

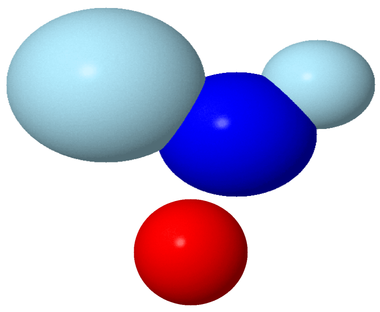
Coarse-Grain MD

When molecules consisting of many and various atoms are inserted into a system, the computation cost is higher. In order to avoid this a whole molecule (or parts of it) can be seen as one solid bead with its own properties. This model is called **coarse-grain model** (CG) and reduces a lot the computation effort in a simulation. Another benefit of coarse-graining is that reduces the number of possible degrees of freedom for the optimization that will follow. The procedure of obtaining the coarse-grain model from the all-atoms one is not part of this project. In the following cases, the [BMIM$^+$][BF$_4$$^-$] is used in its coarse-grain form, presented in Figure 7.1b. In this model the anion [BF$_4$$^-$] is represented by one bead A and the cation [BMIM$^+$] is represented by the rigid 3-bead molecule C2-C1-C3. The properties of the CG model are proposed by Merlet *et al.* ([20]) and are presented in Table 7.1.

In the next chapters, the main properties related with lubrication that are computed are viscosity and the load capacity of IL. At first, these properties are computed only for coarse-grained [BMIM$^+$][BF$_4$$^-$] and then the same algorithms are used in a procedure of search of custom IL with optimized lubricant behaviour.



(a) *All–Atoms Model*



(b) *Coarse–Grain Model*

Figure 7.1: *[BMIM$^+$][BF$_4^-$] molecule. Pictures drawn with Jmol package*

particle	x(Å)	y(Å)	z(Å)	M $(gr \cdot mol^{-1})$	σ_i (Å)	ϵ_i $(kcal \cdot mol^{-1})$	q_i (e)
A	0.000	0.000	0.000	86.81	4.51	0.773	−0.7800
C1	0.000	−0.527	1.365	67.07	4.38	0.611	0.4374
C2	0.000	1.641	2.987	15.04	3.41	0.086	0.1578
C3	0.000	0.187	−2.389	57.12	5.04	0.437	0.1848

Table 7.1: *Properties of coarse-grain particles of [BMIM][BF4] from (ref. [20])*

7.1 Case 1: Viscosity of bulk ionic liquid

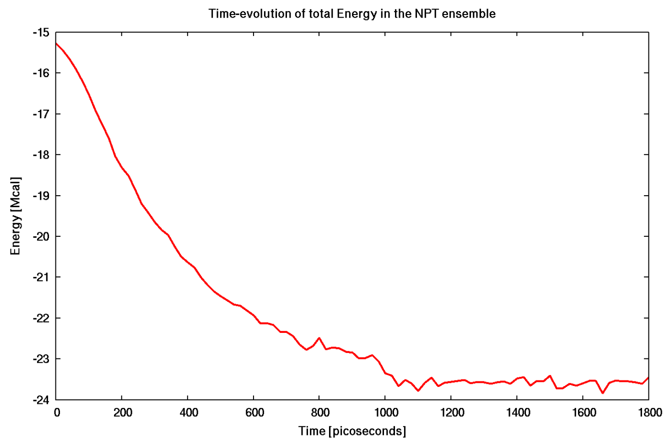
In this case, 3D MD simulation is used to compute density ρ and dynamic viscosity η of the bulk neutral ionic liquid [BMIM$^+$][BF$_4^-$] at four temperatures: 500, 400, 375 and 350 K. For all simulations, the time step for integrating equations of motion is 2 fs. 3 chains of Nosé-Hoover thermostats with a time constant of 10 ps are employed for both NVT and NPT ensembles. Additionally, only for the NPT ensemble the system is barostated at 1 atm with a time constant of 20 ps. Initially, the N=400 ion-pairs are inserted randomly in the simulation volume, and they are allowed to relax in the NPT ensemble for 1800 ps.

During an NPT relaxation, the volume of the unit cell is equal to the volume occupied by its particles, therefore density computations are easily performed. Thus, instantaneous density ρ can be defined as:

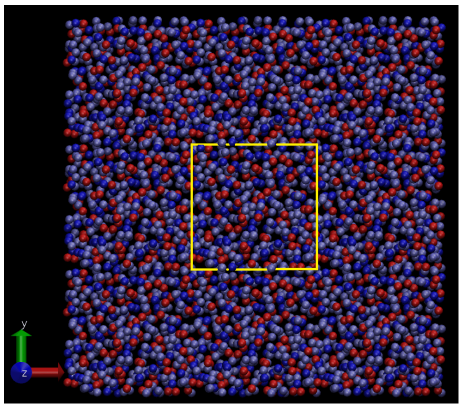
$$\rho = \frac{total\ mass}{volume} \tag{7.1}$$

where total mass is fixed and known in advance and volume derives from the equations of motion for NPT ensemble.

Of course, in order to obtain reliable results, density must be computed after equilibrium and must be averaged for a sufficiently long period of time. Figure 7.2a presents the time evolution of the total energy for a system of 400 [BMIM$^+$][BF$_4^-$] in the NPT ensemble. It is safe to assume that after 1300 ps the system has reached equilibrium and density computations can be performed. Figure 7.2b is a snapshot of the relaxed bulk IL in the xy-plane. The boundaries of the simulation box are coloured yellow in order to distinguish the unit shell from its images. The [BF$_4^-$] anions (bead A) are the red spheres and the rest are the beads of the [BMIM$^+$] cations.



(a) *Time-evolution of the total energy.*



(b) *Snapshot of the equilibrated system. In yellow the unit shell and around it its periodic images in the xy-plane. Red spheres represent the [BF$_4^-$] anions.*

Figure 7.2: *[BMIM$^+$][BF$_4^-$] relaxing in the NPT ensemble.*

The second property obtained in this case is the zero shear viscosity η of the bulk liquid. During viscosity calculations simulation box is fixed and the system is allowed to relax in an NVT ensemble.The dynamic viscosity (or just viscosity) η derives as the mean value of the three shear components η_{xy}, η_{yz} and η_{zx} calculated by the Green-Kubo formula as:

$$\eta_{\alpha\beta} = \frac{V}{k_B T} \int_0^\infty \langle \sigma_{\alpha\beta}(0)\sigma_{\alpha\beta}(\tau)\rangle d\tau \ , \tag{7.2}$$

where T, V are the temperature and the volume of the system and $\langle \sigma_{\alpha\beta}(0)\sigma_{\alpha\beta}(\tau)\rangle$ the autocorrelation function of the off-diagonal components of the stress tensor.

As already mentioned, every instantaneous value computed in MD must be averaged through a long period of time. But for Green-Kubo viscosity this is not enough, the averaging must also take place over different trajectories because the calculations have a lot of noise [21]. For this reason, the same case has been solved many times but from different initial positions of particles. In Figure 7.3, time evolution of viscosity is monitored. The different curves represent different trajectories (produced by different initial positions) of the same mentioned system at temperature of 350 K. It is clear that after 10 nanoseconds viscosity converges to a value within a fixed range. This fact makes the averaging through different trajectories necessary.

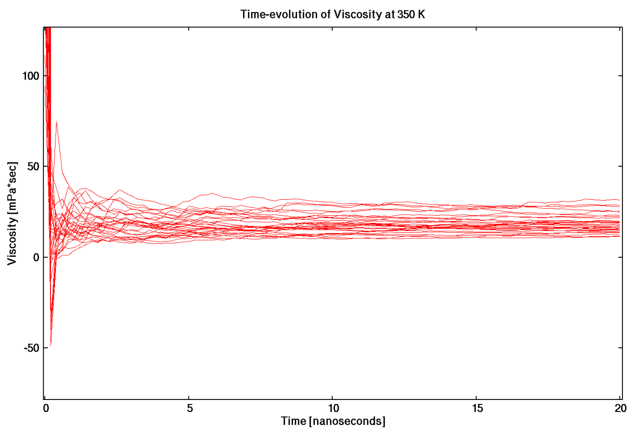
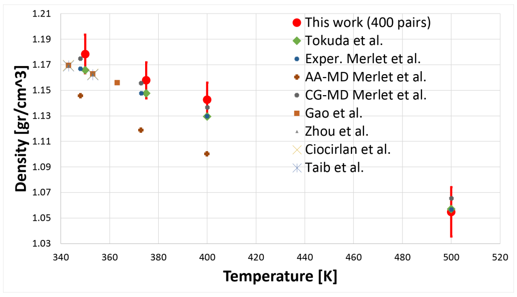


Figure 7.3: *Time-evolution of Green-Kubo viscosity measurements. Different curves represent the same thermodynamic case but with different particle trajectories.*

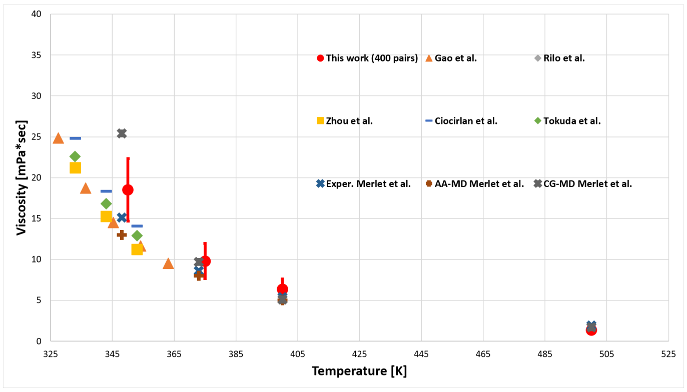
The reliability of these computations can be verified after comparing the results with corresponding experimental databases (ref. [22], [23], [24] and [20]). In Figures 7.4a and 7.4b, the obtained results from the present coarse-grain model and experimental ones from literature are indeed very close and behave similarly. In Figure 7.4a, most of the experimental points of density fall inside the range of the errorbars of the present calculations. Regarding Figure 7.4b, viscosity calculations at temperatures 500, 400 and 375 K are very accurate. The calculated value of viscosity at 350 K is corresponding better to experimental measurements at 340 K of Ciocirlan *et al.* and Gao *et al.*. At 350 K, the difference of this work with the corresponding experimental measurements is from 3 to 5 $mPa \cdot sec$. The point AA-MD Merlet *et al.* is an all-atom MD simulation and its results appear to be more accurate but the equations of motion in this case are solved for the 5 atoms of [BF_4^-] and the 25 atoms of [$BMIM^+$] than just for the 4 particles of the coarse-grain [$BMIM^+$][BF_4^-] and it is much more expensive.

Initially, the number of ion-pairs N is taken equal to 400 (as in Merlet *et al.*), but in order to accelerate the process (in order to use it with Evolutionary Algorithms) it is important to use the smallest N that produces reliable results. For this reason, five different systems for $N = 400$, 200, 140, 110 and 80 are simulated multiple times. The ion-pairs dependence of viscosity is presented in Figure 7.5b for the four temperatures. Every point represents an average over 50 results and every result is taken after relaxation in the NVT ensemble for 10 ns. The errorbars represent the standard deviation only of the red curve (110 ion-pairs). It is clear that only for $N = 80$ viscosity is not converged to the others. So, for every next case N is consider equal to 110 molecules unless mentioned otherwise. Similarly with the viscosity figure, in figure 7.5a the temperature-dependence of density is presented for various N ion-pairs, but there is not any important change of density results when the number of ion-pairs is decreasing.

The computational time needed by the MD described above (2 ns in NPT and 5

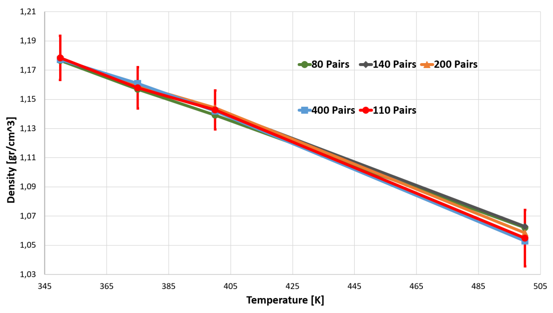


(a) *Density*

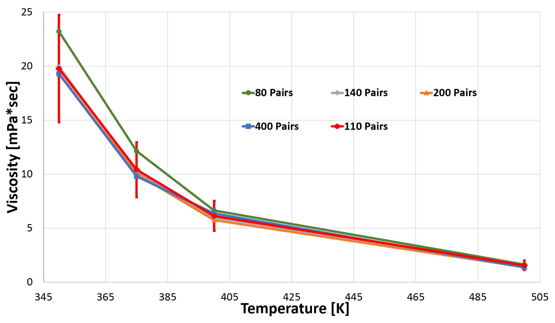


(b) *Viscosity*

Figure 7.4: *Density and viscosity of [BMIM$^+$][BF$_4^-$] over temperature. Compare of present results (red) with corresponding from (ref. [22], [23], [24] and [20]).*



(a) *Density*



(b) *Viscosity*

Figure 7.5: *Temperature-dependence of density and viscosity for $N = 80$, 110, 140, 200 and 400 ion-pairs.*

ns in NVT) is presented in Figure 7.6 for different N. In Figure 7.6 the black solid line represents the time of this simulation performed on 16 CPUs and the red line on 8 CPUs. Having available only 16 CPUs, two runs consuming 8 CPUs can run simultaneously so the red line represents also the time that these two runs need to finish, but if the run is consuming 16 CPUs, multiple runs can only be performed one by one. Thus, for two runs of 16 CPUs the time is double of the black solid line and this is exactly what the black dashed line represents. So, as long the black dashed curve of Figure 7.6 is above red, running on 8 processors is more efficient.

So, the conclusion of figures 7.5b and 7.6 is that 110 ion-pairs are enough for reliable results and the faster way is to run these simulations on 8 processors.

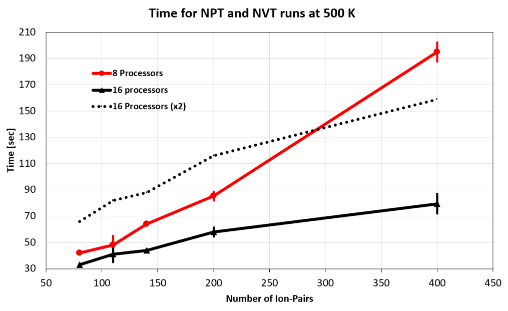


Figure 7.6: *Computational time of measuring density and viscosity at 500 K for various ion-pairs using 8 or 16 processors.*

The structure of the bulk liquid can be understood by looking at the pair radial distribution functions (RDF) $g_{\alpha\beta}(r)$. These curves are the probability of finding the particle β at distance r from α, with α taken as the origin of coordinates where α,β can be either anions or cations. The area around α is separated in discrete chunks and every chunk is represented by a discrete r. The RDFs are computed by equation 7.3 where δ is Kronecker's function and r_{ij} the distance between the i-th α and the j-th β. This radial distribution curves can be found in Figure 7.7 and refer to equilibrated [BMIM$^+$][BF$_4^-$] at 400 K. Every curve in this figure is an average over 40 and the maximum standard deviation (error) of this results is 0.018, so errorbars are neglected from this figure. These curves reveal an ordered structure. As expected, molecules of the opposite charge of a central one are found closer to it than molecules of the same charge. Also, in Cation-Cation curve there is not a specific peak but a wide flat area with maximum $g(r)$. This is happening because the [BMIM$^+$] cation is consisted by three particles and they occupy more than one consecutive chunks.

$$g_{\alpha\beta}(r) = \frac{1}{N_\alpha N_\beta} \sum_{i=1}^{N_\alpha} \sum_{j=1}^{N_\beta} \langle \delta \left(|\mathbf{r_{ij}}| - r \right) \rangle \tag{7.3}$$

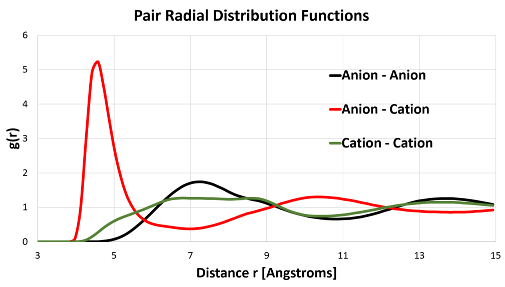


Figure 7.7: *Radial distribution functions of equilibrated [$BMIM^+$][BF_4^-] at 400 K.*

7.2 Minimization of viscosity

It is important for a lubricant to have low viscosity (in hydrodynamic lubrication) in order to reduce friction between two surfaces. The purpose of this section is to study how MD simulations can be used during optimization with Evolutionary Algorithms as an evaluation tool. Also, the importance of every possible design variable is studied, in order to reduce the degrees of freedom for the final optimization that will follow in next section.

Evolutionary Algorithms are population-based optimization methods that imitate the evolution of biological populations. The optimal solutions are these that are better adapted to the objectives. All the optimizations here are conducted in parallel by **EASY (Evolutionary Algorithm SYstem)**, an optimization software developed by the PCopt/NTUA. EASY is a high-fidelity optimization software that can deal single- or multi-objective problems subjected to constrains. It can reduce the needed evaluations by growing Radial Basis Function networks that imitates the evaluation tool. It also offers the choice of coupling the Evolutionary Algorithms with a Gradient-Based method in multiple levels of various fidelity.

As already mentioned, MD simulations and especially viscosity computations are very time-consuming because they require averaging over time and over different trajectories. Also, optimizations by evolutionary algorithms tend to converge slowly especially when the number of design variables is increased, because this number represents the dimensionality of the search space of the problem. For this purpose an initial study of the importance of each design variable is conducted. Multiple optimizations are conducted with a group of design variables as free parameters and all the others kept fixed.

In this optimizations, the objective function is the IL's viscosity at 500 K. The evaluation tool is the LAMMPS algorithm written for and used in section (7.1). All the design variables refer to IL properties and are grouped in four categories: LJ parameters, Geometry of Cation, Masses and Charges. The first eight design variables (first group) are the σ and ϵ parameters of Lennard-Jones Potential for beads A, C_1, C_2 and C_3. The second group contains the lengths from C_1 to C_2 (l_1) and from C_1 to C_3 (l_2) and the angle $C_2 - C_1 - C_3$ (ϕ). The Mass category contains the four CG particle masses m_A, m_{C_1}, m_{C_2} and m_{C_3}. The design variables in Charge group are only two. During optimization anion's and cation's charge are fixed in -0.78 e and 0.78 e and the design variables are defined as $\alpha_1 = q_{C_1}/0.78e$ and $\alpha_2 = q_{C_2}/0.78e$, where q_i is the charge of particle i. The charge of the third bead is defined as $q_{C_3} = 0.78e - q_{C_1} - q_{C_2}$. The evaluation tool also calculates density to use it as a constrain. This kind of constrain is imposed to make sure that the optimized design variables do not refer to a gas that obviously will have much lower viscosity. The imposed constrain is $\rho > 0.6\ gr/cm^3$. For each of these groups an optimization is conducted with the design variables of it as free variables and the variables of the other groups remaining fixed. The Table 7.3 contains all design variables, their range of definition and their order of discretization, where $i = A, C1, C2, C3$ and $j = 1, 2$.

	ϵ_i $(kcal \cdot mol^{-1})$	σ_i (Å)	l_j (Å)	ϕ (degrees)	α_j	m_i $(gr \cdot mol^{-1})$
min	0.001	1.0	1.0	5.0	−0.5	1.0
max	2.0	10.0	5.0	180.0	0.5	200.0
discrete step	0.016	0.07	0.125	0.7	0.008	1.55

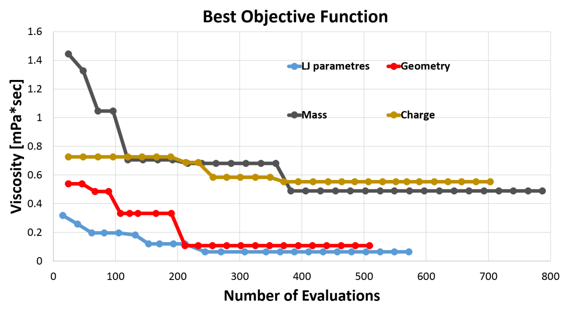
Table 7.2: *Deign variables and their definition range.*

The results are gathered in figure 7.8a. There, the value of the best objective function (minimum viscosity at 500 K) is monitored during the optimizations. The main conclusion is that changing the charge ratios or particle masses is not having as great impact in viscosity as changing LJ and geometry parameters so from now the design variables of these two groups (Charge and Mass) will be fixed. Another reason for this is the observation that some charge combinations produce candidates that their evaluation is very slow.

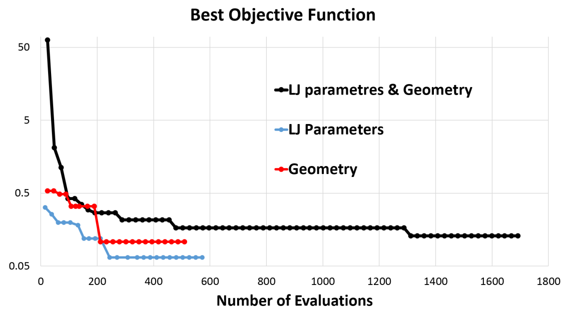
As already mentioned in section 7.1 reliable viscosity computations demand averaging over a range of different trajectories, but it will make optimization very expensive. For increasing the reliability of the results a new optimization is conducted, where viscosity of every candidate is computed in three runs. The difference between these three runs is in the initial molecular positions that consequently develop different molecular trajectories. Of course, three runs are not enough for a reliable statistical quantity and this is a compromise in order to keep low cost. The objective function is the same ($\min(\eta_{500})$) and the design variable set is:

$$\{\epsilon_A, \epsilon_{C1}, \epsilon_{C2}, \epsilon_{C3}, \sigma_A, \sigma_{C1}, \sigma_{C2}, \sigma_{C3}, l_1, l_2, \phi, \alpha_1, \alpha_2, m_A, m_{C1}, m_{C2}, m_{C3}\}$$

but only the first eleven are not fixed.

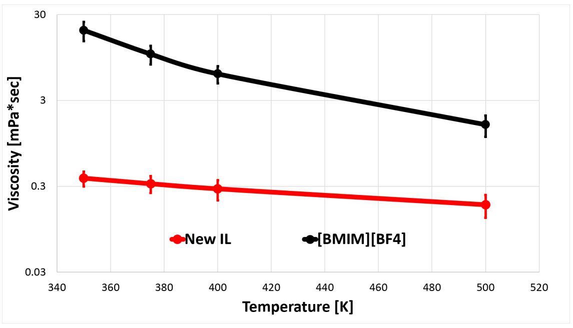


(a) *Minimization of viscosity with different free design variables groups.*

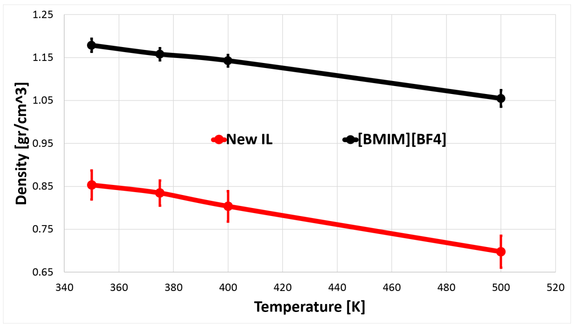


(b) *Minimization of viscosity. Converge of best objective function.*

The results of this optimization are shown in Figure 7.8b (y-axes in logarithmic scale). First thing noticed is that the best solution founded is worse than the ones found in Figure 7.8a (0.13>0.06) but this is not very strange. Viscosity is computed



(a) *Viscosity over temperature.*



(b) *Density over temperature*

Figure 7.9: *Comparison of the new IL with [$BMIM^+$][BF_4^-].*

only once for every candidate in optimizations of Figure 7.8a and thus the algorithm can easily get trapped in an extreme non-realistic solution. The best solution found is:

ϵ_A	ϵ_{C1}	ϵ_{C2}	ϵ_{C3}	σ_A	σ_{C1}	σ_{C2}	σ_{C3}	
0.026	0.088	0.245	0.543	7.38	2.06	3.48	1.14	
l_1	l_2	ϕ	α_1	α_2	m_A	m_{C1}	m_{C2}	m_{C3}
3.84	5.00	154	0.561	0.202	86.81	67.07	15.04	57.12

Table 7.3: *Parameters of optimized Ionic Liquid*

As already explained masses and charges are fixed so they are the same with [$BMIM^+$][BF_4^-] but the parameters of Table 7.3 represent a different Ionic Liquid. Of course, the properties of this new liquid must be computed in a more detailed way. For this purpose LAMMPS algorithm of chapter (7.1) is used again for measuring new IL's density and viscosity. The results are printed in figures 7.9 with the same properties of [$BMIM^+$][BF_4^-]. Y-axes of viscosity figure is in logarithmic scale for a better compare. Figure 7.9a shows that viscosity is minimized not only at 500 K (that is the objective function) but in all the temperature range. An other interesting thing to observe is that the gradient of the $\eta(T)$ curve is reduced. For the new IL it is found that $\eta_{500} = 0.18\ mPa \cdot sec$ and $\eta_{350} = 0.37$ so their ratio is $\eta_{350}/\eta_{500} = 2.1$ and the same ratio for [$BMIM^+$][BF_4^-] is equal to 12.6 (see section 7.1). The additional benefit of this fact is that it makes easier to design the function of an engine in a wide temperature range.

7.3 Case 2: Force of confined ionic liquid

In this case, 3D MD simulations are used in order to compute the force of the confined ionic liquid [BMIM$^+$][BF$_4$$^-$] between two metal walls of iron (Fe) and gradually compressed by the upper wall. The force computed here is from the liquid to the upper wall. In figures 7.11, snapshots of this case are presented. The y-direction is vertical to the page, z-direction is opposite to the move of the upper wall and x-direction is pointing from the center to the right of the volume. The boundaries are periodic only in the y-direction. The volume of the liquid and the walls is small enough compared with the total volume in order to have space for the squeezed out molecules. The particles of every wall are behaving as a single entity on which the external force or motion is imposed. The wall particles are situated on a BCC lattice (2 basis atoms, one at the corner and one at the center of the cube) of 2.87 Å length and their Lennard-Jones parametres are $\epsilon_{Fe} = 0.0022$ kcal/mol and $\sigma_{Fe} = 2.49$ Å. Every wall consists of 3 uncharged layers of atoms and one (neighbor to the liquid) of positive charge. The total charge of every charged wall-layer is equal with the charge of 15 cations, thus 11.7 e but the whole system is neutral.

Initially, the gap between the walls is 23 Å. The wall is compressing the liquid with a constant velocity of 38 m/s for 2.5 ps and then stays still and let it relax for 25 ps. This is happening multiple times until the final gap becomes equal to the diameter of an anion. These alternating compressions and relaxations are performed in a NVT ensemble of time constant of 100 fs with a time step of 0.01 fs for integrating equations of motion. During this run the normal force applied from the IL to the wall is computed by the following formula:

$$F = \frac{\sum_{j=1}^{N_w}\sum_{i=1}^{N_{IL}} F_{ij}}{N_w}\ , \tag{7.4}$$

where N_w is the number of particles of the upper wall, N_{IL} is the number of IL particles and F_{ij} the force on z-direction from the liquid particle i to the wall particle j and derives from Equation 6.3.

In Figure 7.10 this normal force is presented as a function of the gap between the two walls. The red line monitors the normal force during the run and the black points are the normal force computed at the end of every relaxation and are interpolated by the black visual-guide line. These black points represent the average of normal force over the last 50 fs of each relaxation. The black line is considered as the real force curve and is behaving similar with experimental measurements of ref. [30].

As expected when the ionic liquid is confined, its molecules are placed in layers of alternating charge (ref. [30], [26] and [25]). This is pointed in following Figures 7.11. By looking on the relaxed points in force curve at Figure 7.10, it seems that as the IL is compressed the force is increasing until some topical peaks and then is

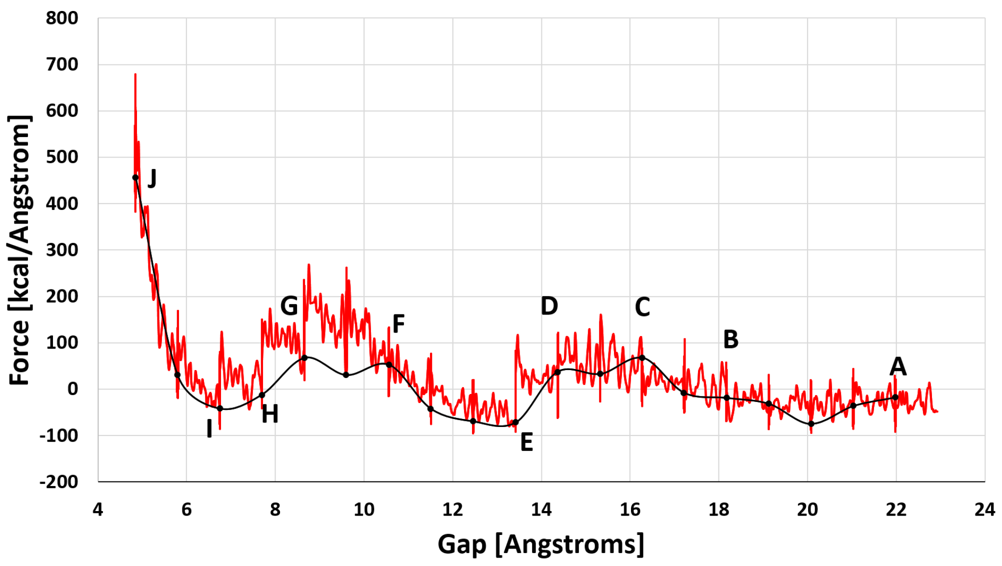
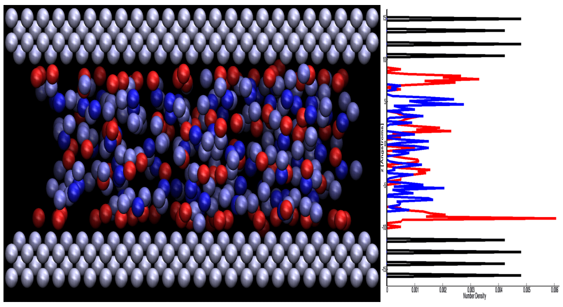
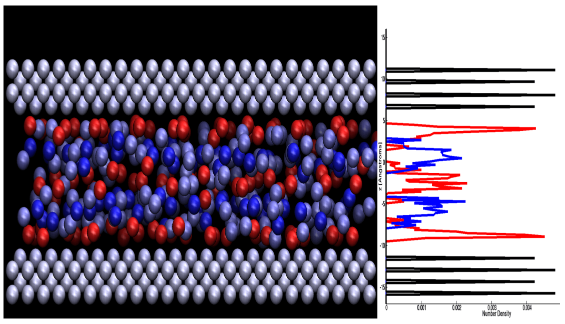


Figure 7.10: *Normal force of liquid to the wall during compression of the IL.*

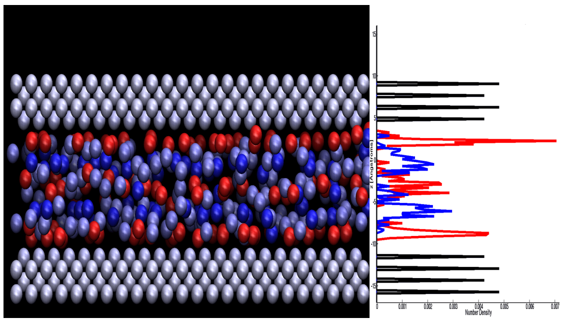
falling. This is happening because after these peaks some molecules are squeezed out, the number of ion layers is reduced and the force is falling and even becomes negative in certain regions. The negative normal force is happening because the ionic liquid is striving to reduce the gap due to adhesion phenomena(ref. [26]). After some compression this procedure is repeated. This fact can be confirmed by both snapshots and number density curves of figures 7.11. In these figures red beads and curves refer to anions, blue curves and both blue and light blue beads refer to cations. Black curves and grey uniformly placed beads refer to the walls. In these figures, the number density of wall particles is divided by 40 in order to fit well in figures. From both snapshots and number density curves the layering of $[BMIM^{+}][BF_4^{-}]$ molecules is distinguishable. In point A, there are four anion layers and three of cations. Then two layers are squeezed out, the force drops and before point B it starts increasing. At points B to D there are three anion layers and two of cations. After point D the force drops so two ion layers are squeezed out and in point E there are two anion layers and one of cations. The same layering condition is presented until point G. Until that time the latest number of layers is compressed but not squeezed out and thus force is increasing. After point G some ions are squeezed out and at point I there is clearly only one layer consisted by similar number of cations and anions. At final point J almost all cations are squeezed out and the one remaining layer is mostly consisted of anions. For engineering applications the steep rise of the normal force at small gaps can be beneficial for protecting against solid-solid contact and consequent wear. The last safe point that a lubricant should work for hydrodynamic lubrication is assumed to be point G. If the gap between the walls becomes less than that it is not ensured that the walls will not get in touch and for this not to happen the force on point G must be maximized.



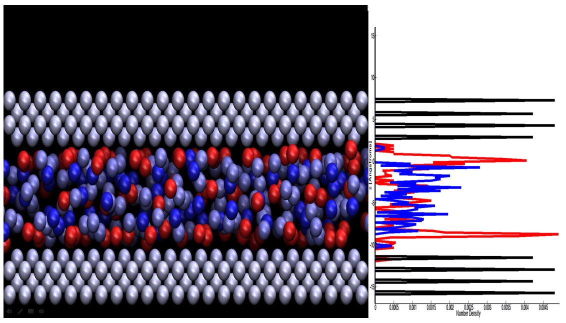
(a) *Point A*



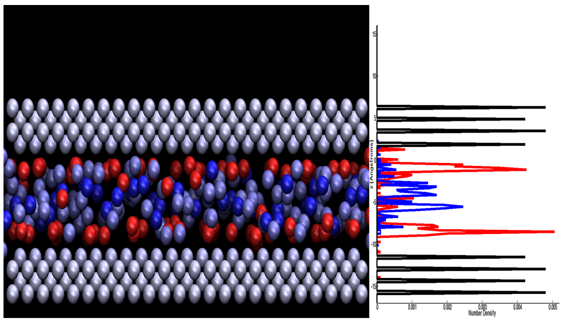
(b) *Point B*



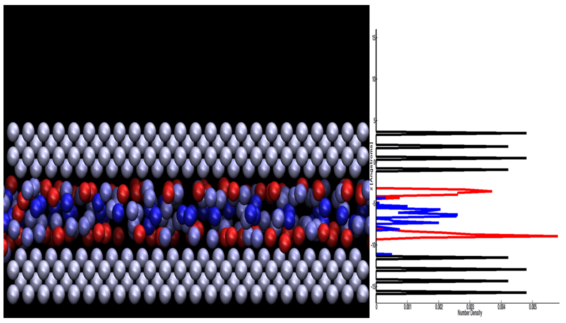
(c) *Point C*



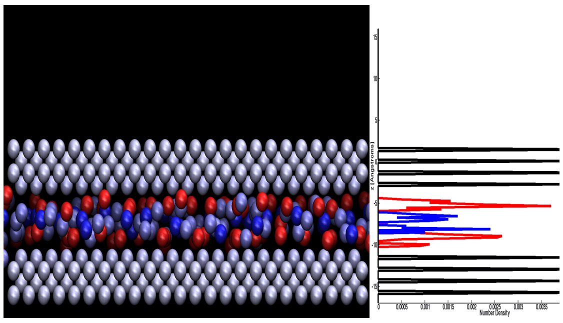
(d) *Point D*



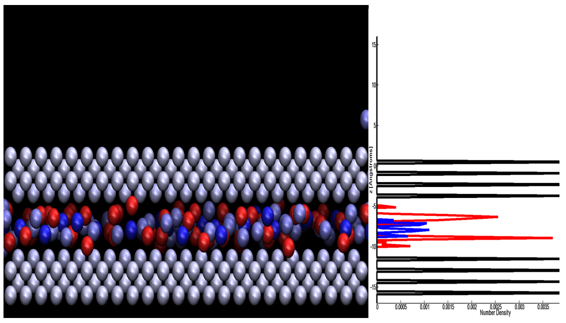
(e) *Point E*



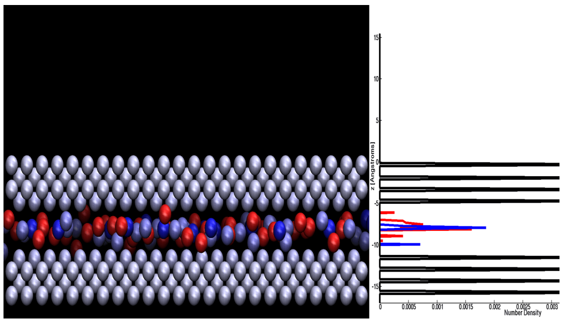
(f) *Point F*



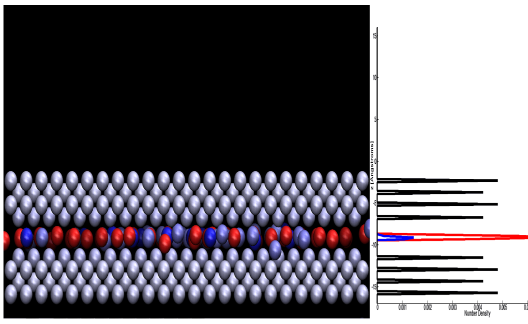
(g) *Point G*



(h) *Point H*



(i) *Point I*



(j) *Point J*

Figure 7.11: *Snapshots of the system at points of figure 7.10 accompanied by number density curve.*

7.4 Optimization

The purpose of this section is to find the CG IL with optimized lubricant properties by using the CG model of [BMIM$^+$][BF$_4^-$] as template.

A lubricant is introduced between two moving parts to reduce friction. For this purpose, in case of full-film lubrication its viscosity must be minimum and in case of thin-film lubrication it must be capable to carry very strong forces in order to avoid solid-solid contact. So two quantities are optimized here: viscosity of the thick film must be minimum and force capability of thin film must be maximum.

The full-film is modelled as the bulk liquid of Case 1 and the thin film as the confined liquid of Case 2. So the first objective function is directly defined as the viscosity of bulk IL at 500 K. For the thin film, it must be ensured that at least three ion-layers are always between the two iron plates, so the normal force at this gap (point G in Fig. 7.10 for [BMIM$^+$][BF$_4^-$], point "G" for every random candidate during optimization) must be maximized. EASY is only minimizing so the two objective functions are defined as: $min(\eta_{500})$ and $min(-F_{"G"})$.

Finding the point "G" automatically for every candidate is not so obvious. The upper wall is compressing (and relaxing) the liquid until the gap is equal to the diameter of one anion. This diameter is equal to $\sigma = 0.5(\sigma_A + \sigma_{Fe})$ (according to Lorentz-Berthelot mixing rules). When monitoring force, after point "G" (or G in figure 7.10), two layers are squeezed out and the force drops because of the produced vacuum. After some more compressions, the increase of normal force is very steep.

In order to find point "G" (last point with three ion-layers) the normal force curve (black point-line in Fig.7.4) must be read from the opposite direction that is created. Starting from the point of the smallest gap (point J in Fig.7.4), automatically the force of every point is compared with the force of the next point corresponding to bigger gap. As long as the first point of this couple is greater than the second, its force is set to zero otherwise this procedure stops. For this modified force curve, "G" is the total maximum point. This statement is in agree with experimental results (ref. [30]). Even in the case that the total maximum is somewhere else in the curve (for some irrational reason) the existence of at least three ion-layers is ensured.

	ϵ_i $(kcal \cdot mol^{-1})$	σ_i (Å)	l_j (Å)	ϕ (degrees)
min	0.001	1.0	1.0	5.0
max	2.0	6.0	6.0	180.0
discrete step	0.016	0.08	0.08	0.7

Table 7.4: *Design variables, their definition range and their discretization.*

The design variables are the same with the final optimization of Section 7.2. They are presented in Table 7.4 accompanied by their definition range and the order of their

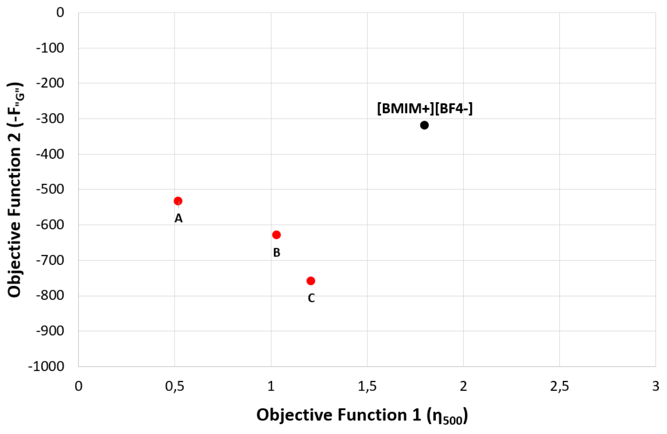


Figure 7.12: *The Pareto Front accompanied by the [BMIM$^+$][BF$_4^-$] point.*

discretization. Again, the bulk liquid density is calculated and used as constrain ($\rho \geqslant 0.6\ gr/cm^3$) in order not to end up with a vapour.

Figure 7.12 shows the Pareto front (red points) of this optimization and the initial liquid [BMIM$^+$][BF$_4^-$] (black point). The three new liquids are having almost half viscosity and more than double force from [BMIM$^+$][BF$_4^-$]. The design properties of these three optimal liquids are presented in table 7.5. These properties do not correspond to real atoms; the particles are CG beads that must be translated to real all-atom models but this is not part of this project. Chemistry will indicate which of these can be translated to a real IL, but the fact that the three optimal solutions are so different (see the angles in table 7.5) allows to be optimistic.

	ϵ_A ($kcal \cdot mol^{-1}$)	ϵ_{C1} ($kcal \cdot mol^{-1}$)	ϵ_{C2} ($kcal \cdot mol^{-1}$)	ϵ_{C3} ($kcal \cdot mol^{-1}$)
min	0.01	0.01	0.01	0.01
max	2.0	2.0	2.0	2.0
Best A	0.261	0.433	0.229	0.605
Best B	0.0.621	0.558	0.088	0.621
Best C	0.308	0.417	1.295	0.135
[BMIM$^+$][BF$_4^-$]	0.773	0.611	0.086	0.437

	σ_A (Å)	σ_{C1} (Å)	σ_{C2} (Å)	σ_{C3} (Å)	l_1 (Å)	l_2 (Å)	ϕ (degrees)
min	1.0	1.0	1.0	1.0	1.0	1.0	5.0
max	6.0	6.0	6.0	6.0	6.0	6.0	180.0
Best A	5.84	5.21	5.37	2.90	5.44	4.73	177.2
Best B	4.89	5.37	4.41	2.27	3.54	5.37	109.7
Best C	4.73	1.63	1.08	4.89	3.38	3.46	29.8
[BMIM$^+$][BF$_4^-$]	4.51	4.38	3.41	5.04	2.71	3.82	116.0

Table 7.5: *Parameters of optimized Ionic Liquid*

Chapter 8

Conclusions and Perspectives

Conclusions

To summarize, in this part of the Diploma Thesis viscosity and force of the CG [BMIM$^+$][BF$_4^-$] were computed and then this CG model is used as a template for the optimization of these quantities. The main conclusions of this study are:

- CG models used for computing properties of IL produce accurate results and reduce significantly the computational cost.
- By the coupling of MD and EA, three new IL are found with optimized lubrication properties.

Perspectives

The results of this internship, opens the way for new interesting studies on this subject. Some of them are:

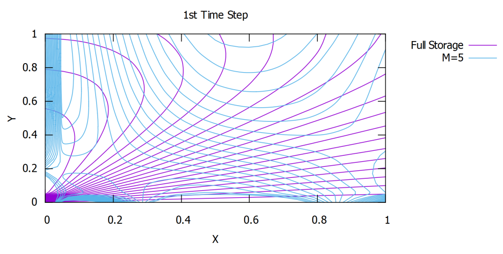
- The "translation" of the three new coarse-grain IL to their chemical formula in order to be studied more detailed and evaluated as possible lubricants.
- A more complicated study can be achieved. The molecular model of the cation could have more than three beads in order to model longer molecules. In the optimization procedure all masses and all charges (in respect of zero total charge) could be used as design variables for a better optimization. Of course, all these increase the computational cost of an evaluation and the optimization cost.

Appendix A

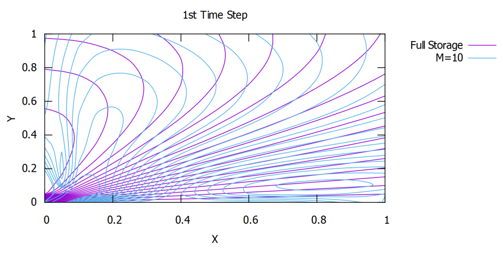
Appendix: Numerical Example of 2D unsteady field

Figures related to the numerical example of subsection 2.2.2 are presented here. These figures visualize the original field (eq. 2.55) and the compressed ones, in order to help the reader realize the high accuracy of the approximation method.

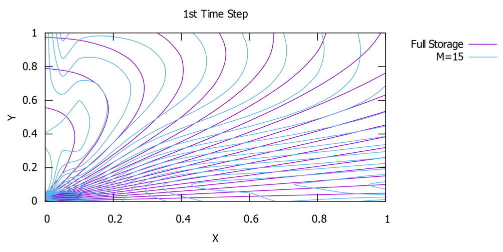
In the following figures, the iso-u curves of the original field and of the iPGD approximations (for $M = 5,\ 10,\ 15,\ 20,\ 25,\ 30$ and 40) are presented at the 1st, 15th, 25th and 45th time step.



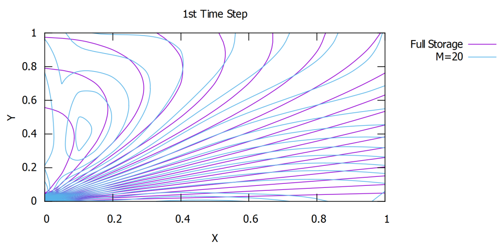
M=5



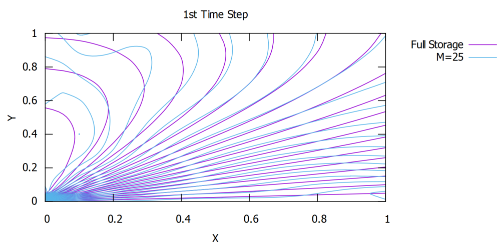
M=10



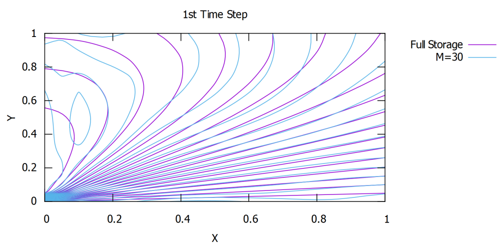
M=15



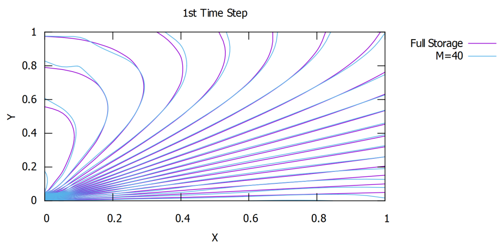
M=20



M=25

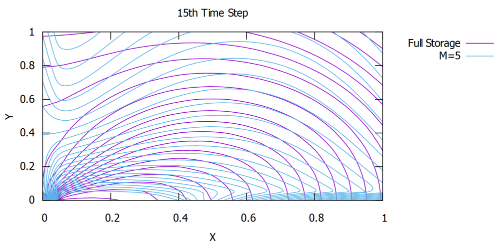


M=30

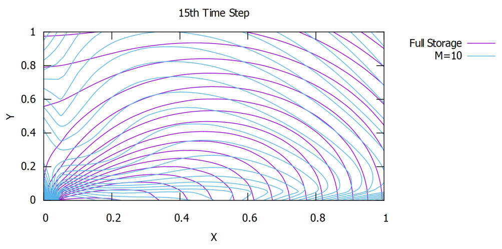


M=40

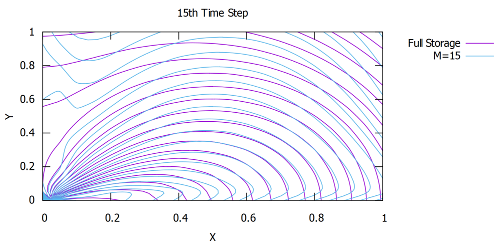
Figure A.1: *Numerical example - function 2.55. Iso-u curves at the* 1*st time step. Comparison between the original and the iPGD approximations for* $M = 5, 10, 15, 20, 25, 30$ *and* 40.



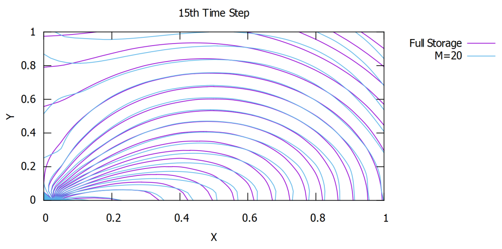
M=5



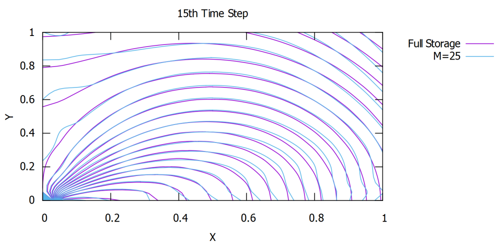
M=10



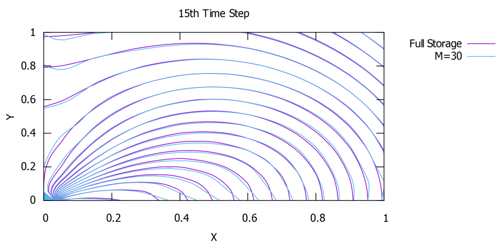
M=15



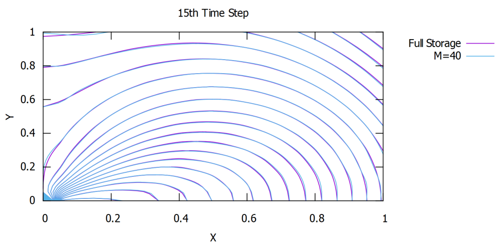
M=20



M=25

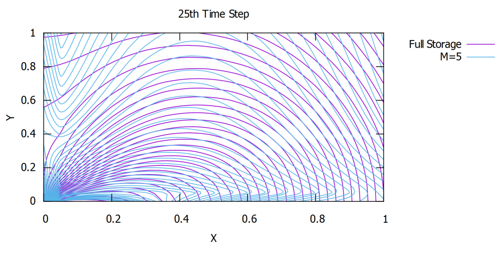


M=30

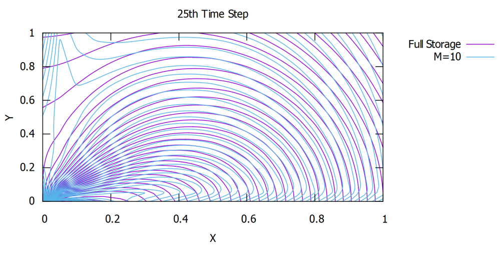


M=40

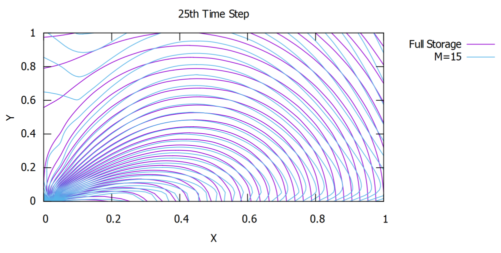
Figure A.2: *Numerical example - function 2.55. Iso-u curves at the* 15*th time step. Comparison between the original and the iPGD approximations for* $M = 5,\ 10,\ 15,\ 20,\ 25,\ 30$ *and* 40.



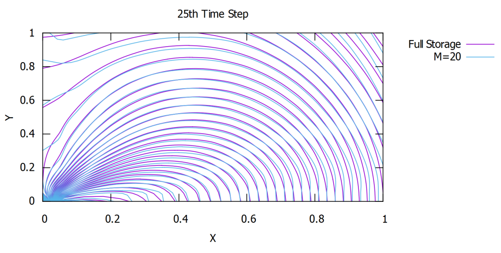
M=5



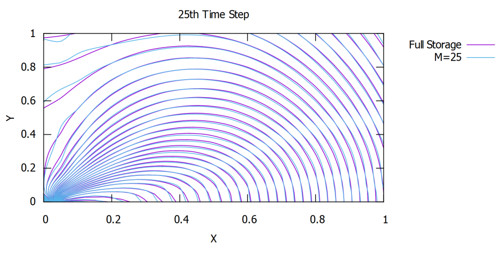
M=10



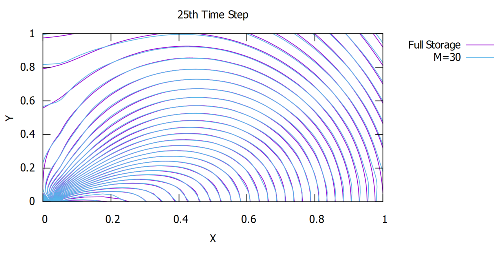
M=15



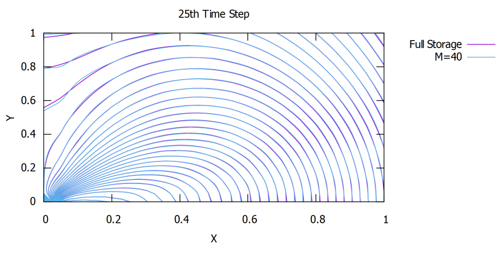
M=20



M=25

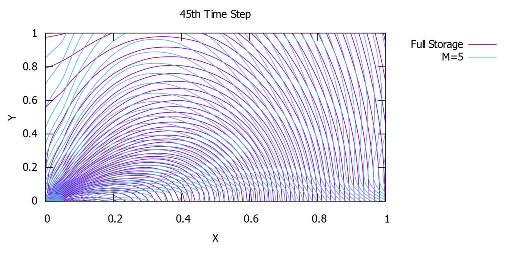


M=30

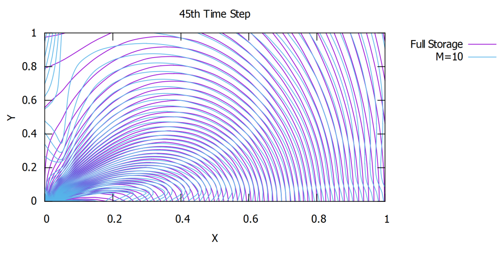


M=40

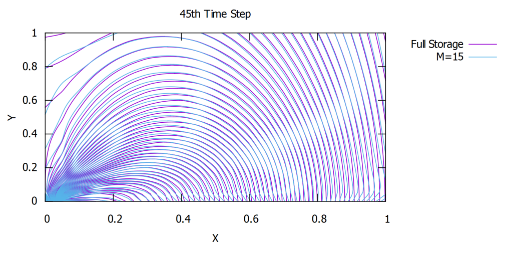
Figure A.3: *Numerical example - function 2.55. Iso-u curves at the* 25*th time step. Comparison between the original and the iPGD approximations for* $M = 5,\ 10,\ 15,\ 20,\ 25,\ 30$ *and* 40.



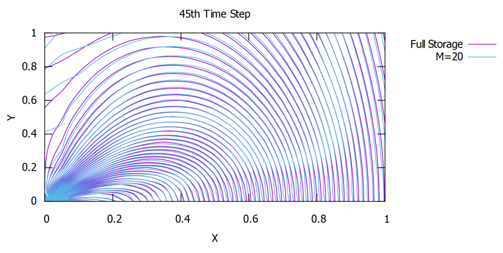
M=5



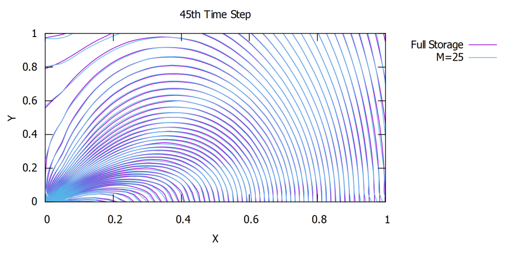
M=10



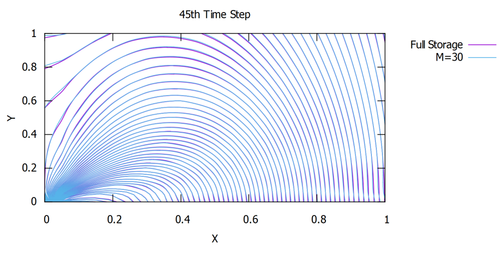
M=15



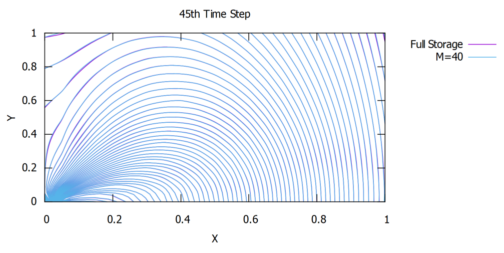
M=20



M=25



M=30



M=40

Figure A.4: *Numerical example - function 2.55. Iso-u curves at the* 45*th time step. Comparison between the original and the iPGD approximations for* $M = 5,\ 10,\ 15,\ 20,\ 25,\ 30$ *and* 40.

Appendix B

Appendix: Numerical Example of 3D unsteady field

In this Appendix are presented the figures of the numerical example of subsection 2.2.4 in order to highlight the efficiency of the iPGD method. The iso-u curves of the original field (eq. 2.73) and of the iPGD approaches (for approximation sum terms $M = 10,\ 16,\ 24$ and 32) are presented on the xy-plane at the 1st, 10th, 25th and 35th time step for the discrete values of z spatial variable ($z = 0.1$, $z = 0.3$, $z = 0.5$, $z = 0.7$ and $z = 0.9$).

Iso-u field for $l = 1$ and $k = 1$

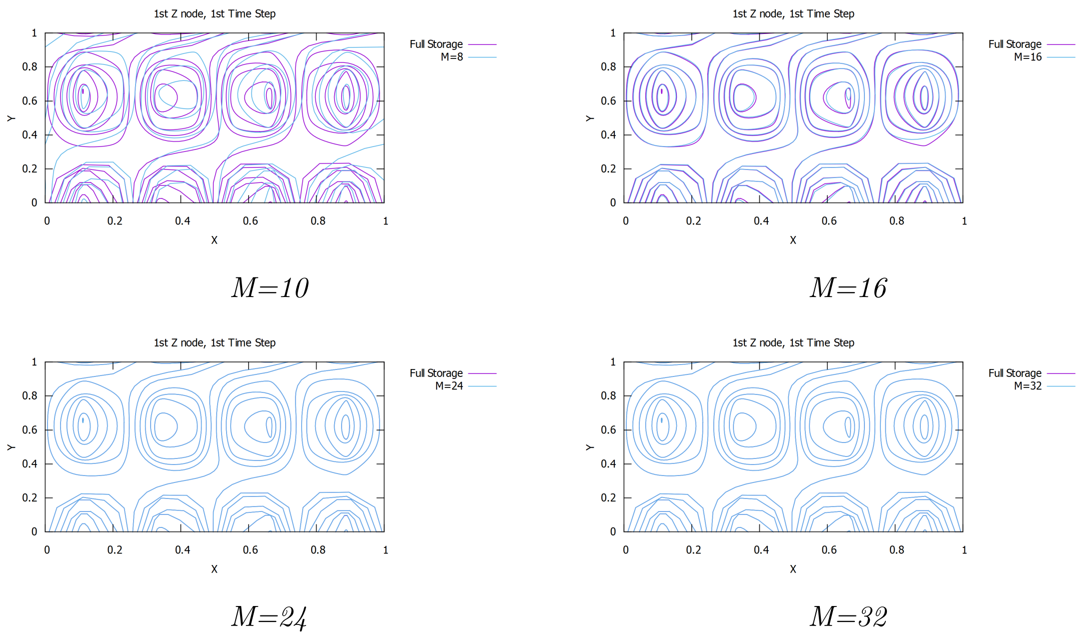


Figure B.1: *Numerical example - function* (2.73). *Iso-u curves at the* 1*st time step and* $l = 1$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 3$ and $k = 1$

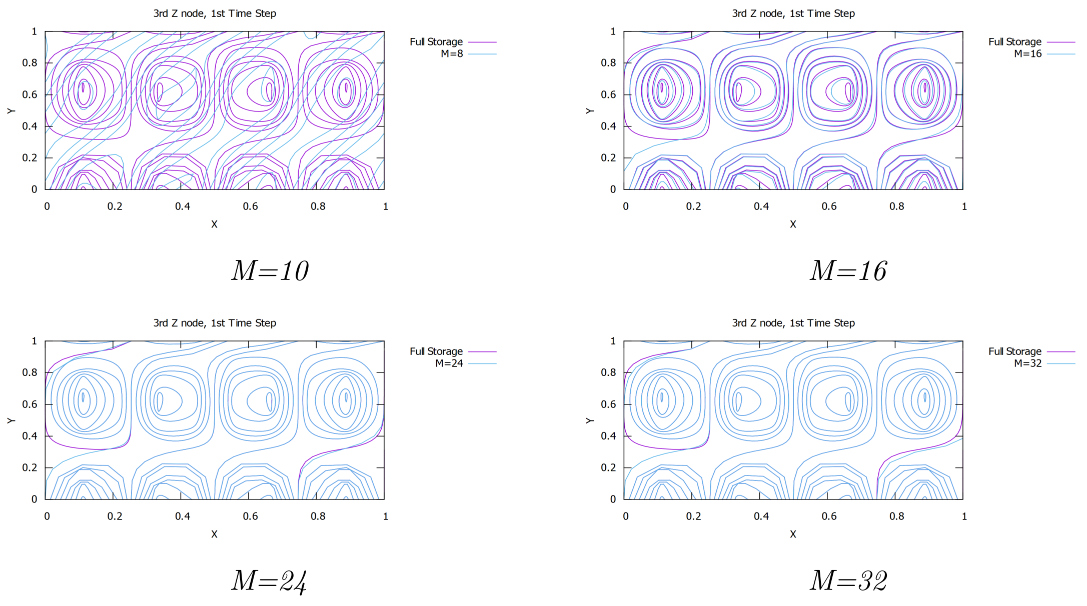


Figure B.2: *Numerical example - function* (2.73). *Iso-u curves at the* 1*st time step and* $l = 3$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 5$ and $k = 1$

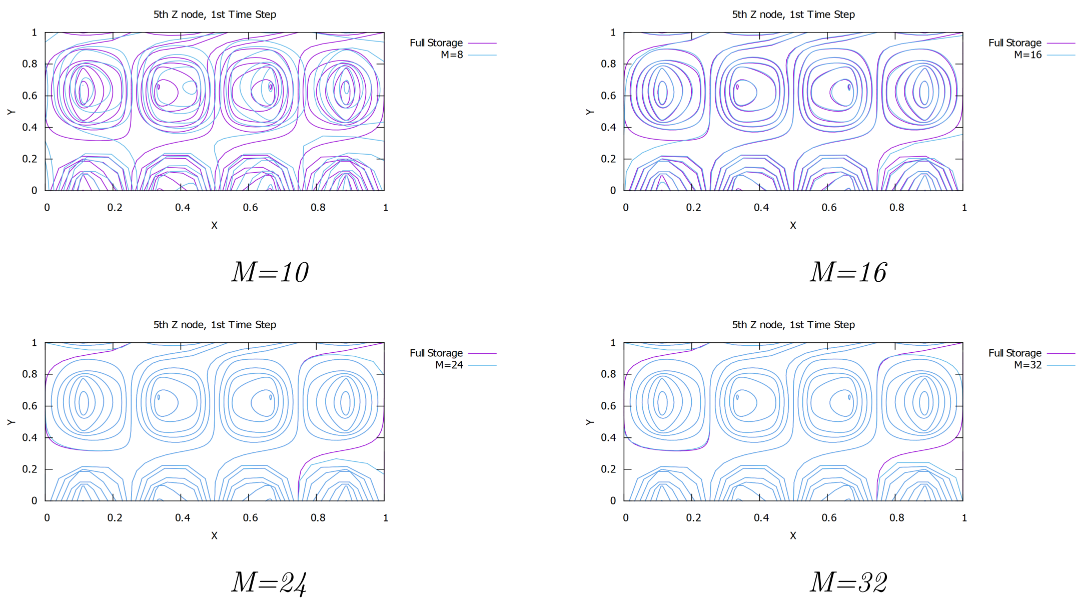


Figure B.3: *Numerical example - function* (2.73). *Iso-u curves at the* 1*st time step and* $l = 5$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 7$ and $k = 1$

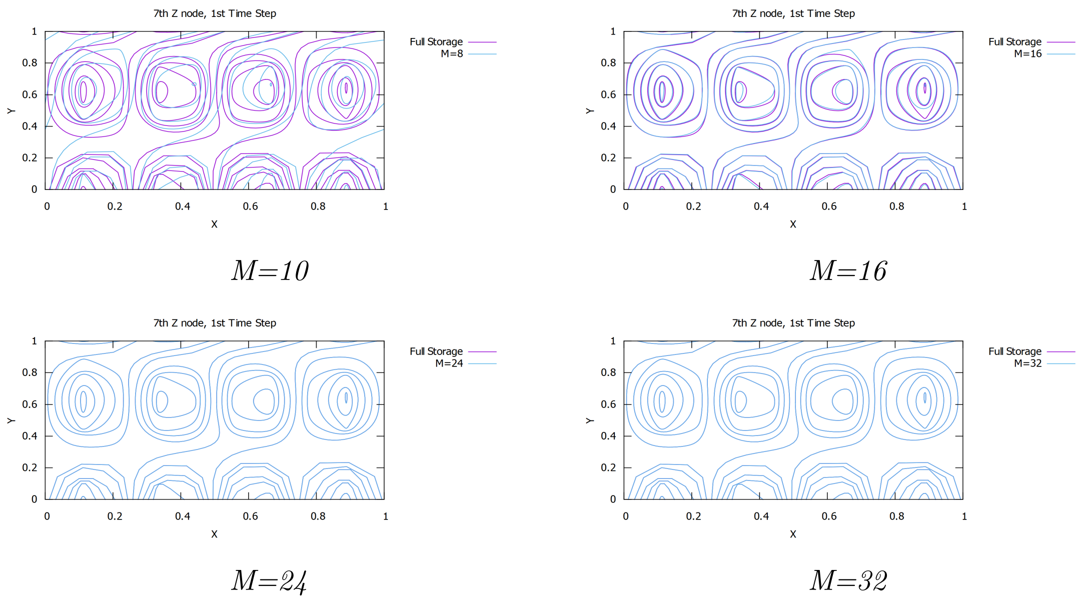


Figure B.4: *Numerical example - function* (2.73). *Iso-u curves at the* 1*st time step and* $l = 7$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 9$ and $k = 1$

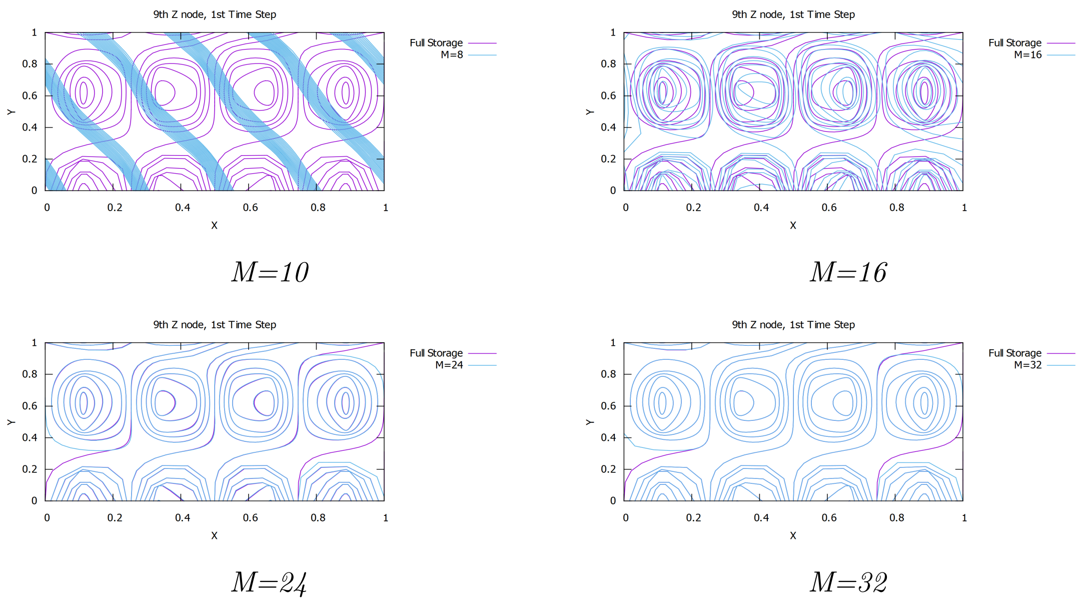


Figure B.5: *Numerical example - function* (2.73). *Iso-u curves at the* 1*st time step and* $l = 9$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 1$ and $k = 10$

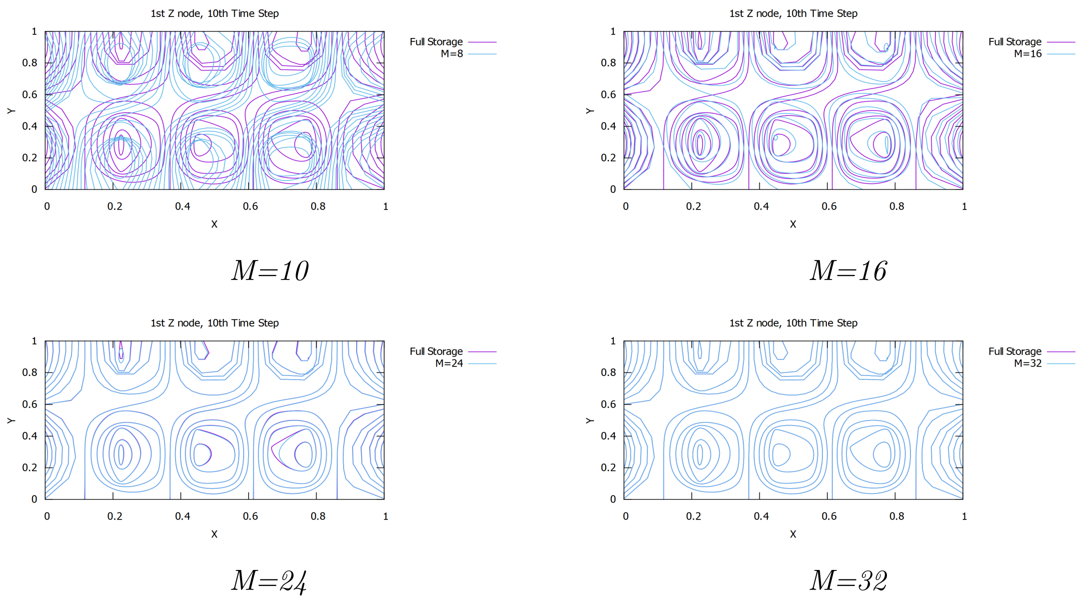
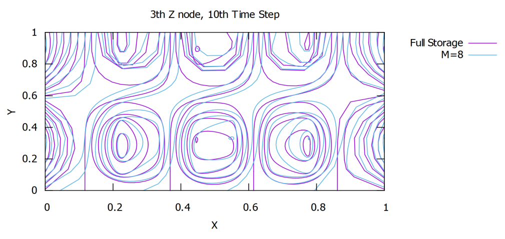
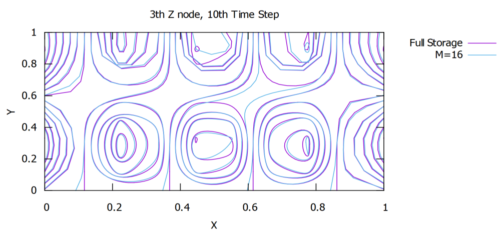


Figure B.6: *Numerical example - function* (2.73). *Iso-u curves at the* 10*th time step and* $l = 1$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

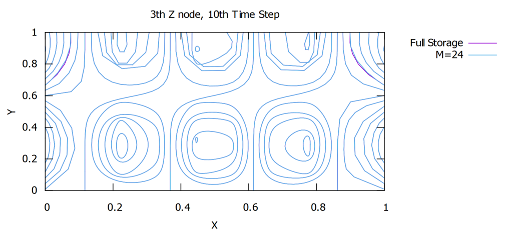
Iso-u field for $l = 3$ and $k = 10$



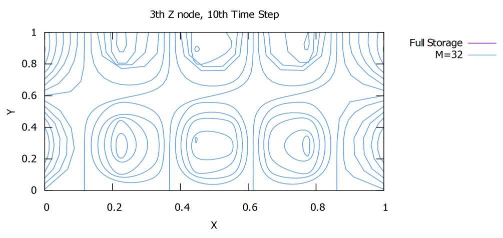
M=10



M=16



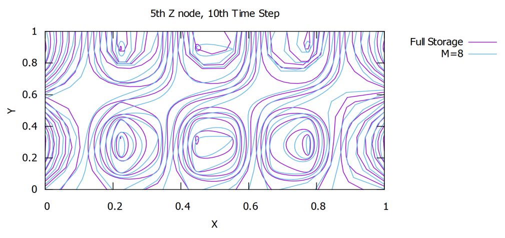
M=24



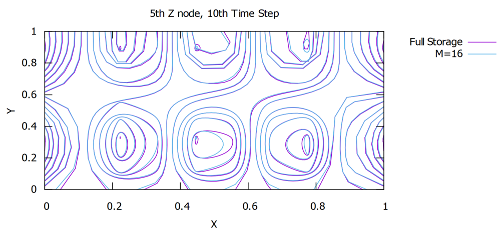
M=32

Figure B.7: *Numerical example - function* (2.73)*. Iso-u curves at the* 10*th time step and* $l = 3$*. Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32*.*

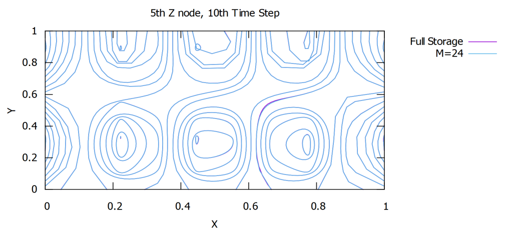
Iso-u field for $l = 5$ and $k = 10$



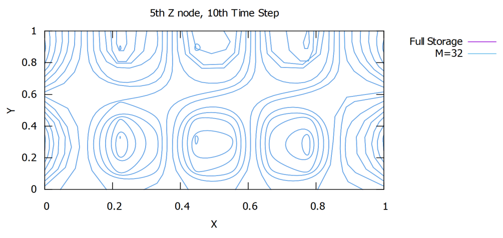
M=10



M=16



M=24



M=32

Figure B.8: *Numerical example - function* (2.73)*. Iso-u curves at the* 10*th time step and* $l = 5$*. Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32*.*

Iso-u field for $l = 7$ and $k = 10$

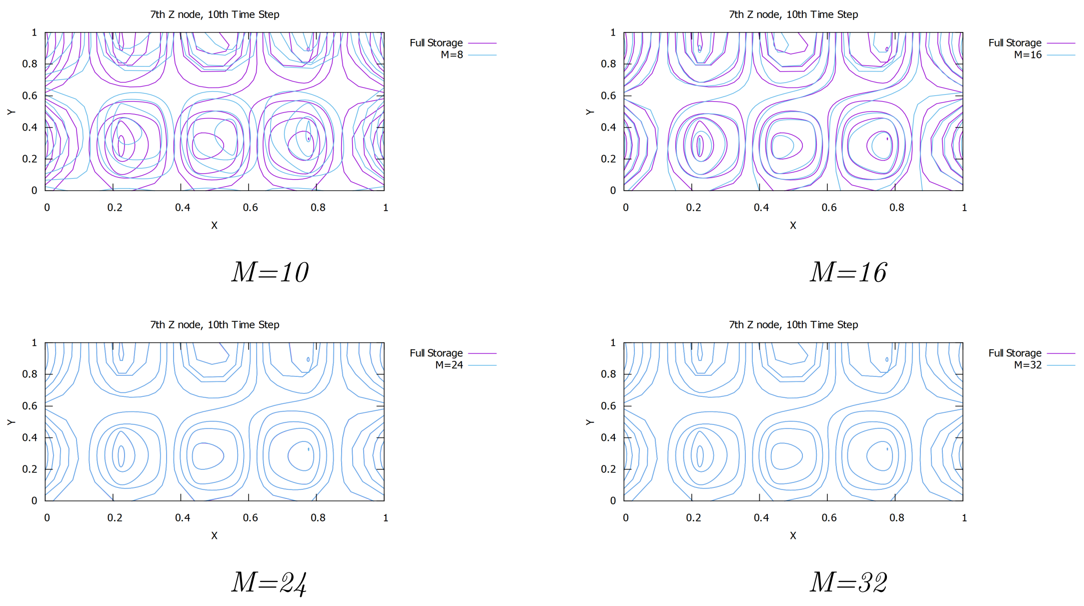


Figure B.9: *Numerical example - function* (2.73). *Iso-u curves at the* 10*th time step and* $l = 7$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 9$ and $k = 10$

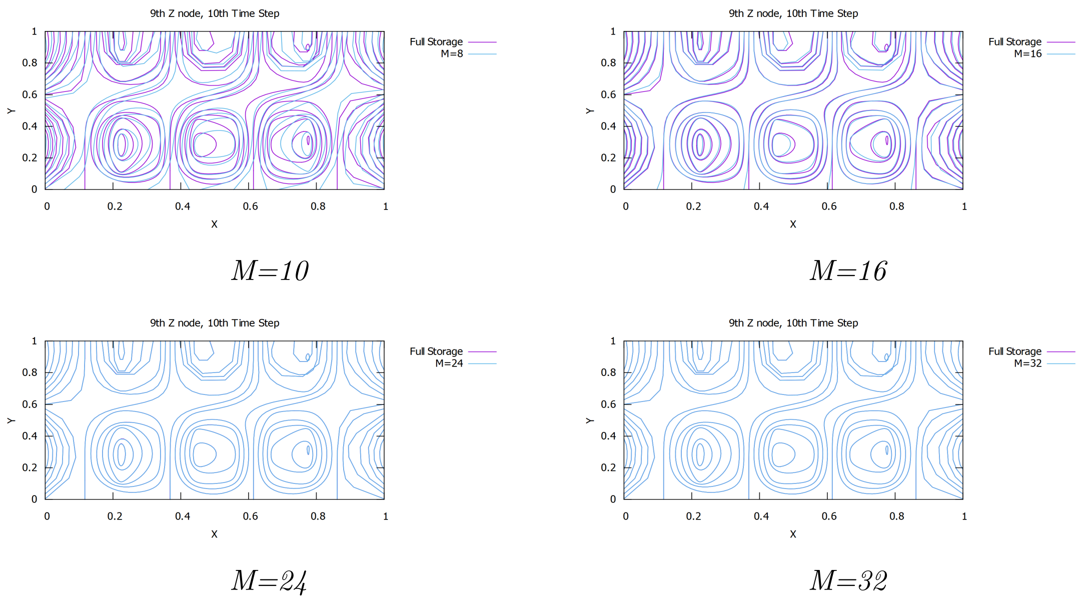


Figure B.10: *Numerical example - function* (2.73). *Iso-u curves at the* 10*th time step and* $l = 9$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 1$ and $k = 25$

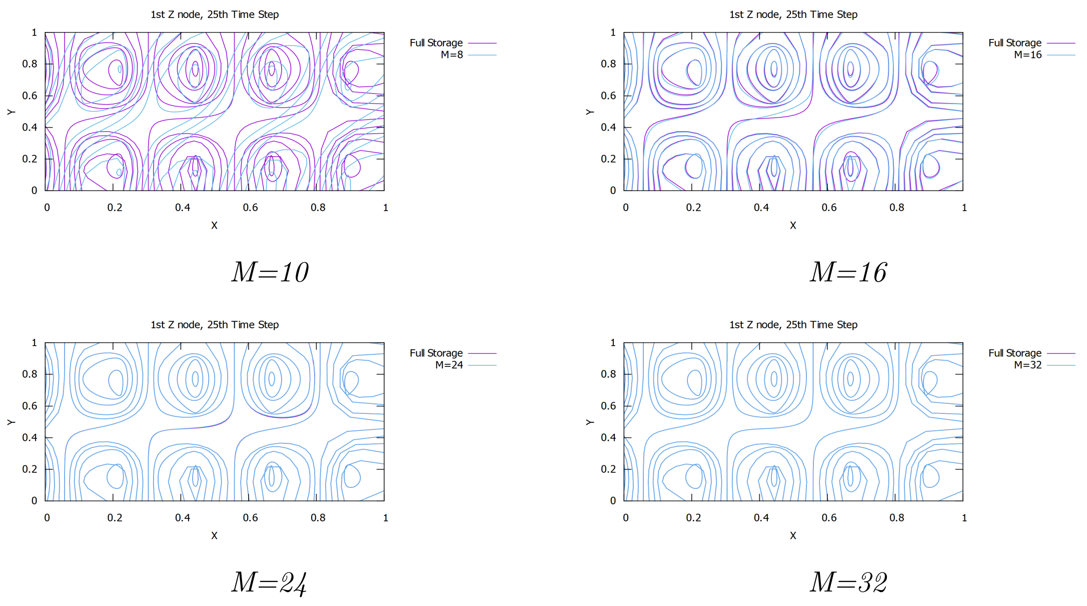


Figure B.11: *Numerical example - function* (2.73). *Iso-u curves at the* 25*th time step and* $l = 1$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 3$ and $k = 25$

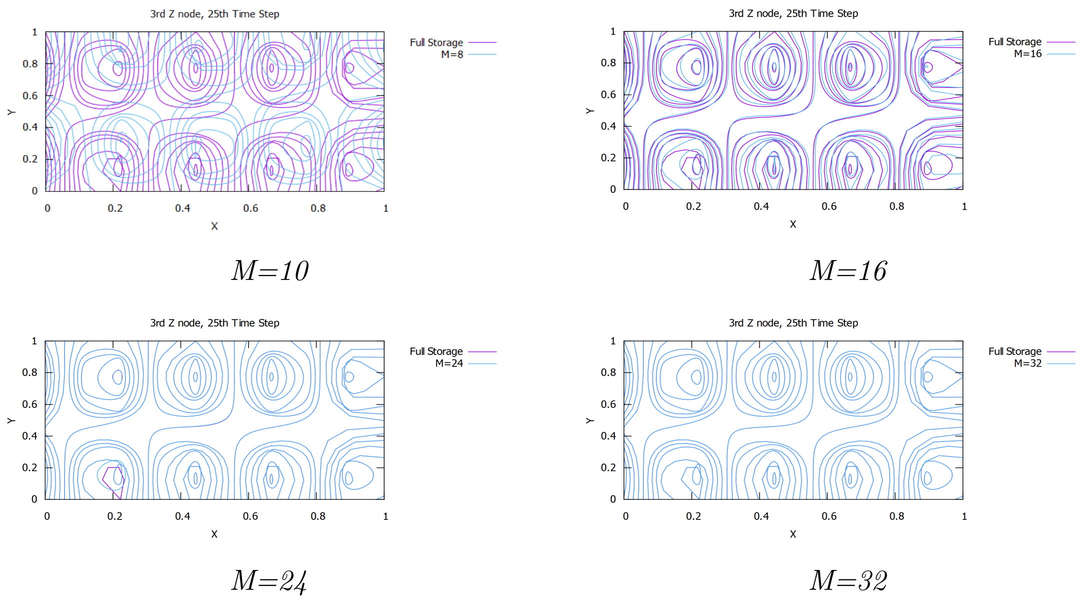
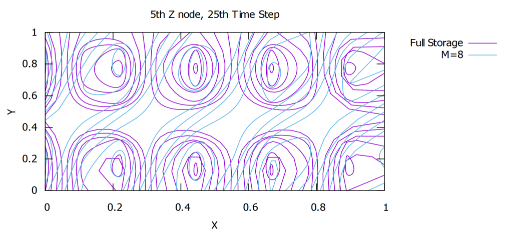
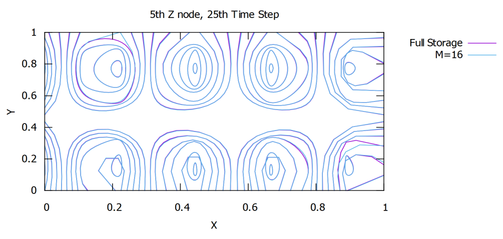


Figure B.12: *Numerical example - function* (2.73). *Iso-u curves at the* 25*th time step and* $l = 3$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

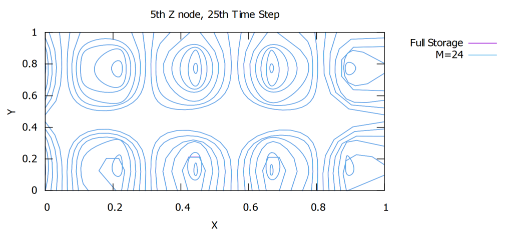
Iso-u field for $l = 5$ and $k = 25$



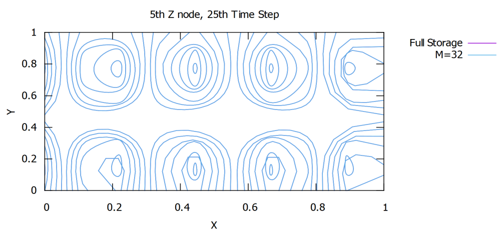
M=10



M=16



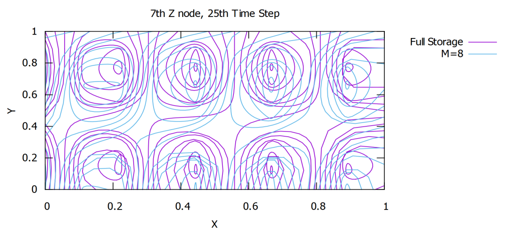
M=24



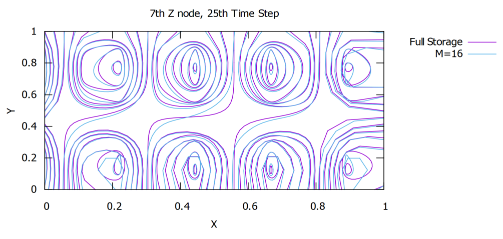
M=32

Figure B.13: *Numerical example - function* (2.73)*. Iso-u curves at the* 25*th time step and $l = 5$. Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32*.*

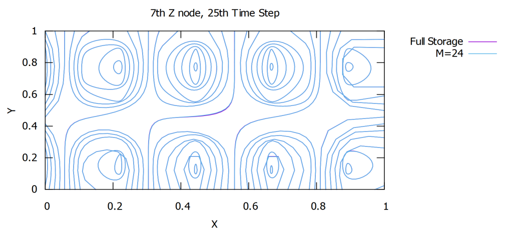
Iso-u field for $l = 7$ and $k = 25$



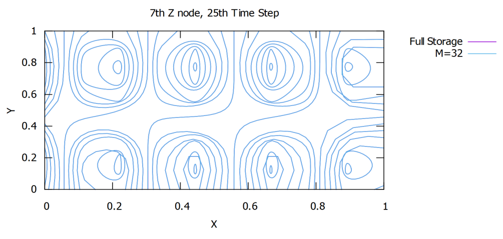
M=10



M=16



M=24



M=32

Figure B.14: *Numerical example - function* (2.73)*. Iso-u curves at the* 25*th time step and $l = 7$. Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32*.*

Iso-u field for $l = 9$ and $k = 25$

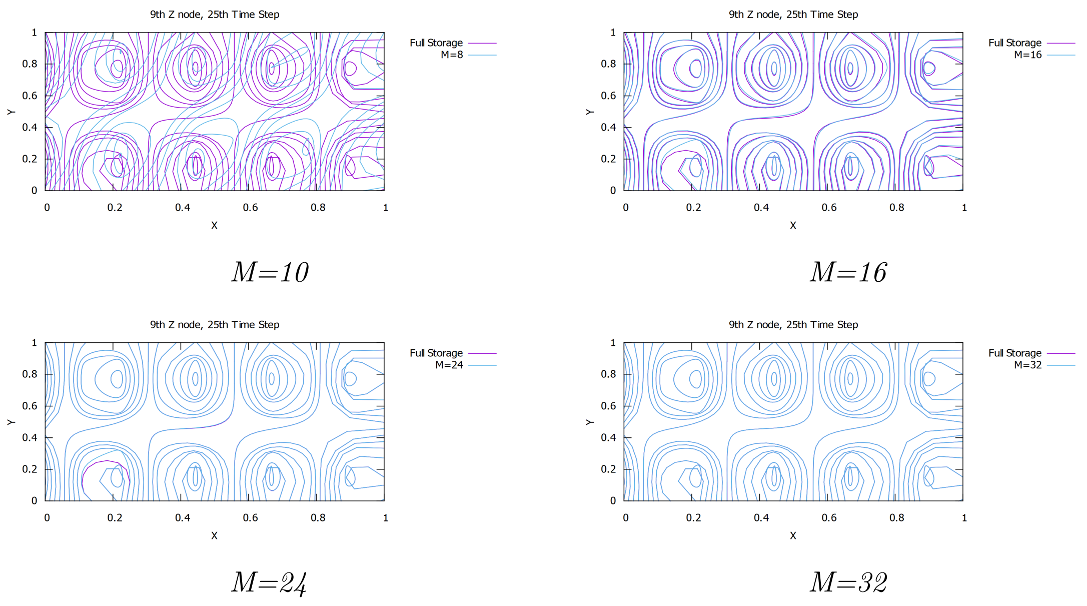


Figure B.15: *Numerical example - function* (2.73). *Iso-u curves at the* 25*th time step and* $l = 9$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 1$ and $k = 35$

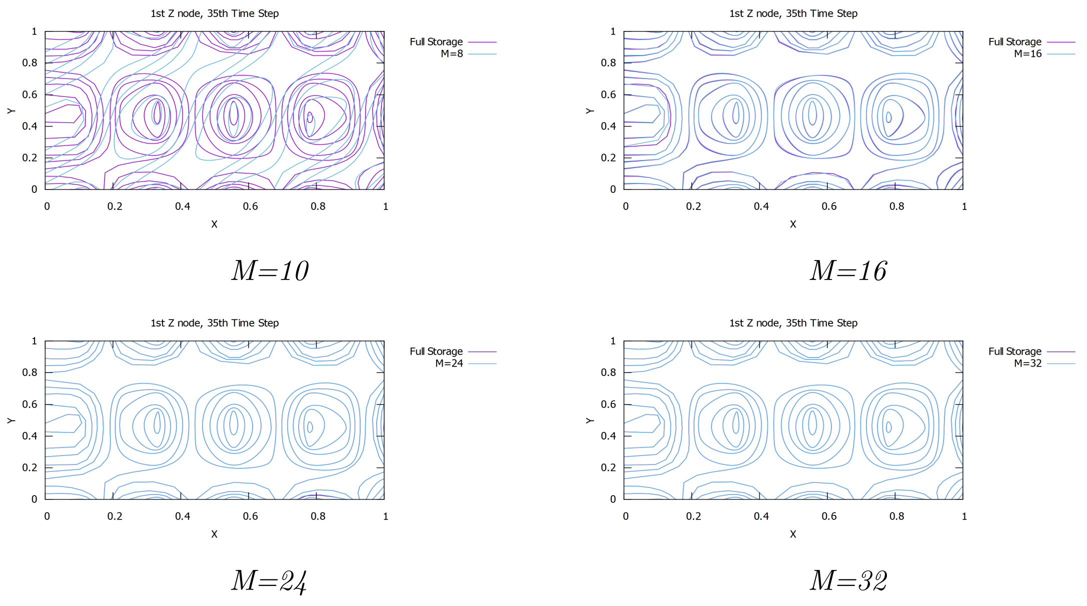


Figure B.16: *Numerical example - function* (2.73). *Iso-u curves at the* 35*th time step and* $l = 1$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 3$ for $k = 35$

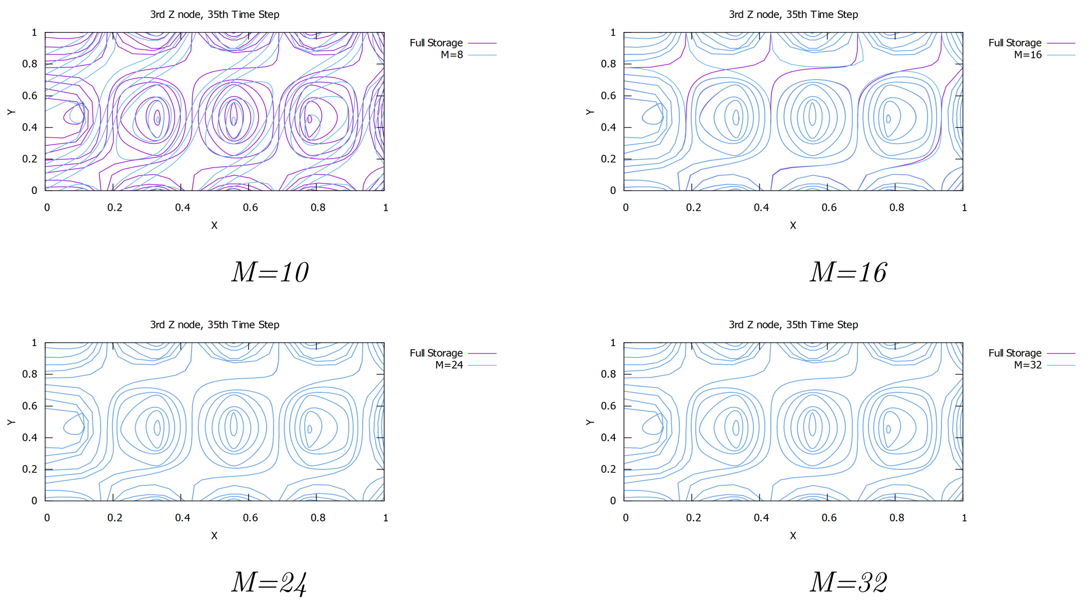


Figure B.17: *Numerical example - function* (2.73). *Iso-u curves at the* 35*th time step and* $l = 3$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 5$ and $k = 35$

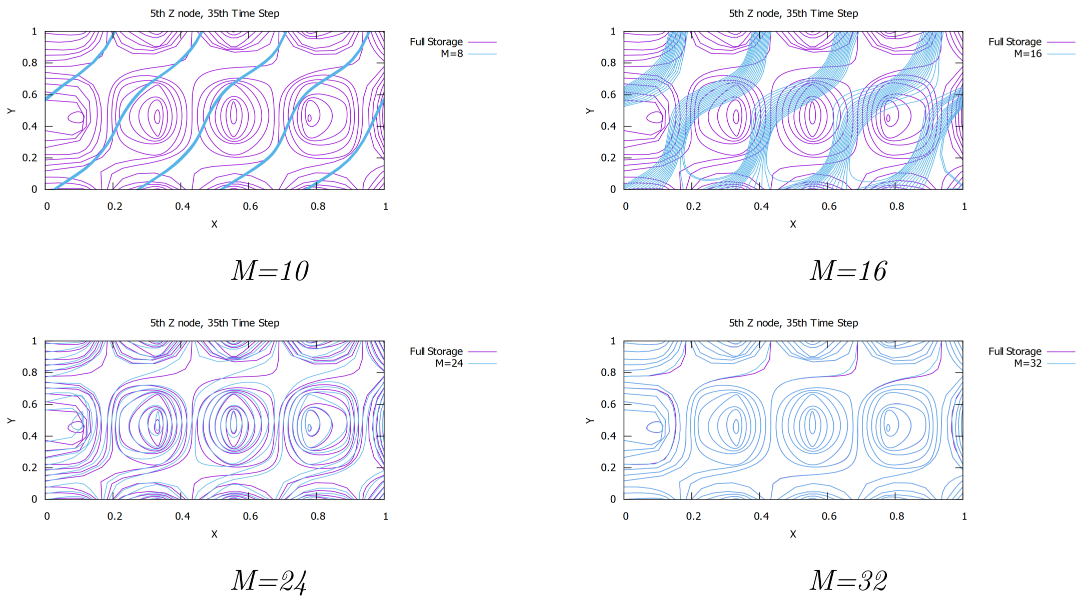


Figure B.18: *Numerical example - function* (2.73). *Iso-u curves at the* 35*th time step and* $l = 5$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 7$ and $k = 35$

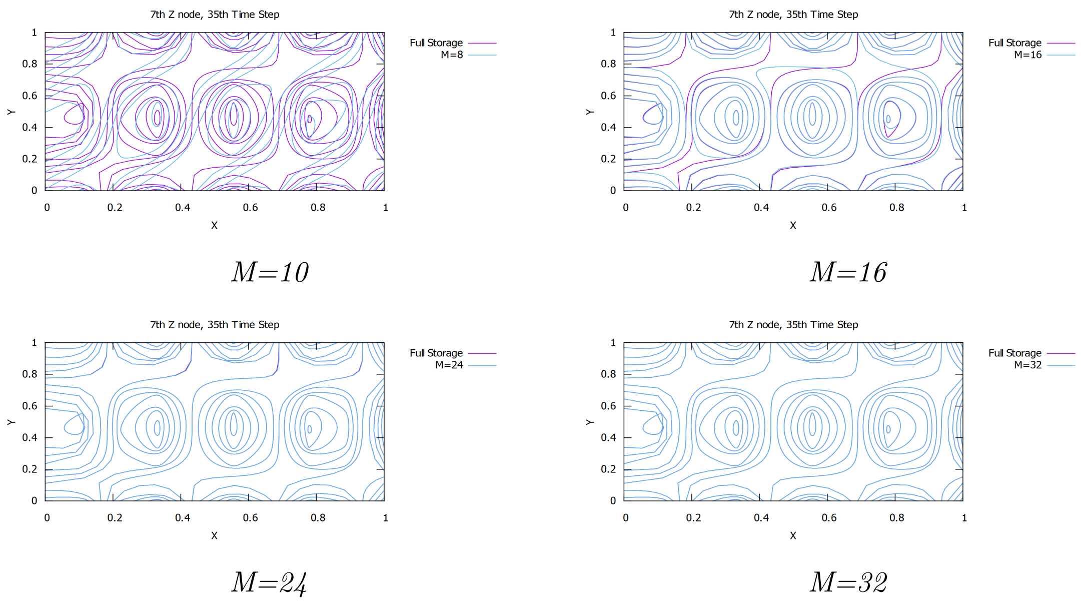


Figure B.19: *Numerical example - function* (2.73). *Iso-u curves at the* 35*th time step and* $l = 7$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

Iso-u field for $l = 9$ for $k = 35$

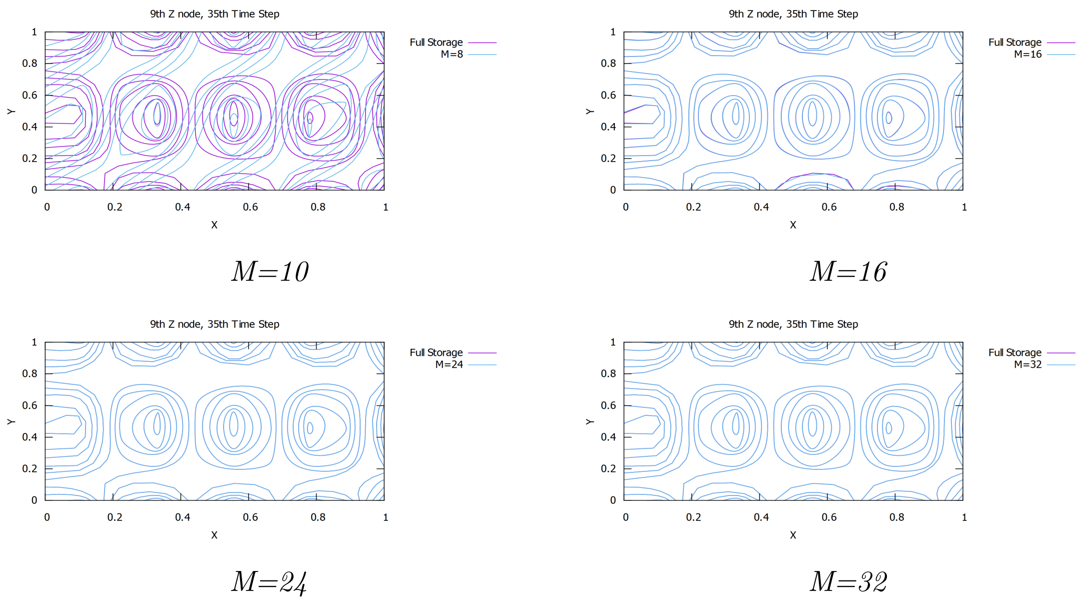


Figure B.20: *Numerical example - function* (2.73). *Iso-u curves at the* 35*th time step and* $l = 9$. *Comparison between the original and the iPGD approximations for* $M = 10,\ 16,\ 24$ *and* 32.

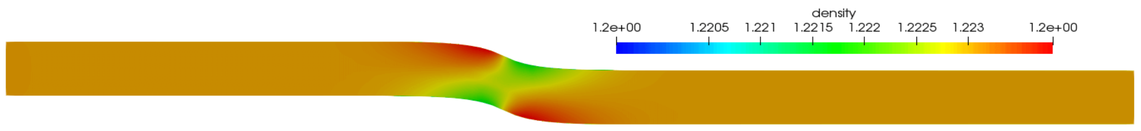
Appendix C

Appendix: Unsteady Flow through a 2D S-Shaped Pipe

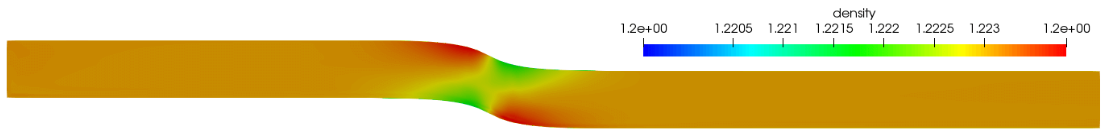
In this Appendix, the CFD results of density ρ, velocities u, v in x and y directions respectively are compared with their iPGD approximations for various summation terms (M = 10, 50, 70 and 100). The fields are plotted at the 1st, 7th, 13th and 19th time step.

The comparisons are plotted in 2D coloured maps. In these figures the values of the flow field are represented as a colour-scale in the xy-plane.

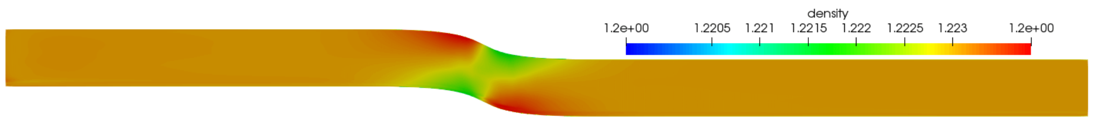
Density ρ at the 1st Time Step



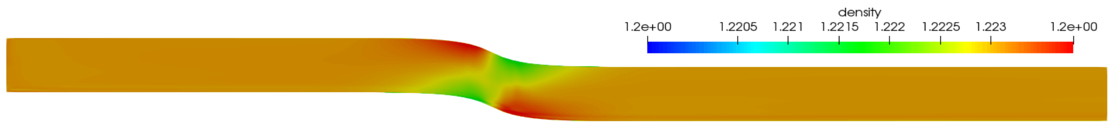
Full Storage



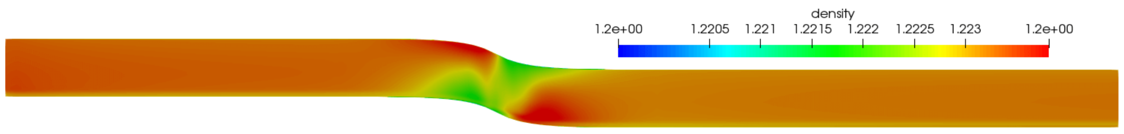
M=100



M=70



M=50



M=10

Figure C.1: *Flow through a 2D S-shaped pipe. Density at the* 1*st time step. Original field and iPGD approximations for* $M = 100, 70, 50$ *and* 10.

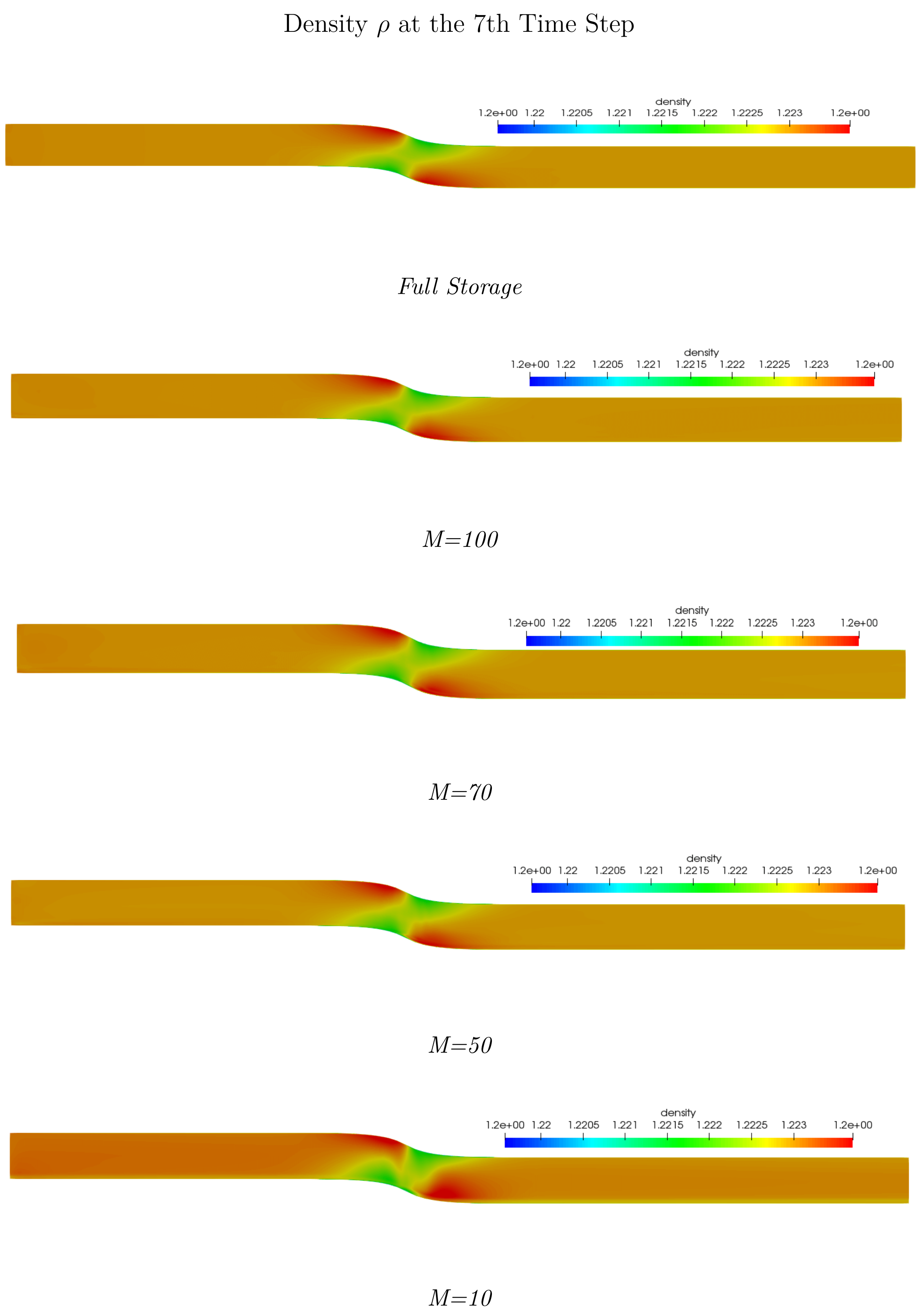
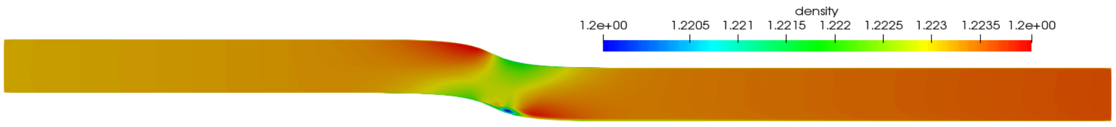
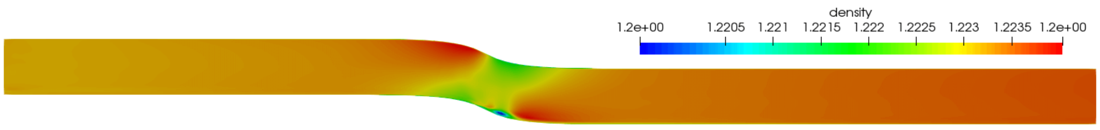


Figure C.2: *Flow through a 2D S-shaped pipe. Density at the 7th time step. CFD results and iPGD approximations for $M = 100, 70, 50$ and 10.*

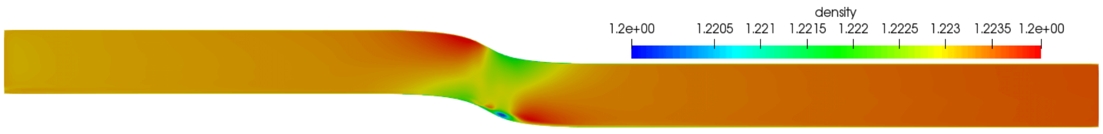
Density ρ at the 13th Time Step



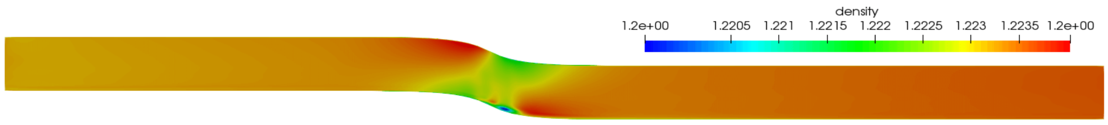
Full Storage



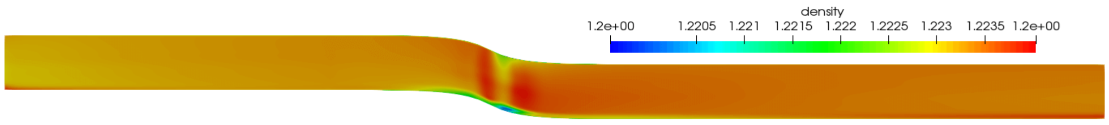
M=100



M=70



M=50



M=10

Figure C.3: *Flow through a 2D S-shaped pipe. Density at the 13th time step. CFD results and iPGD approximations for $M = 100, 70, 50$ and 10.*

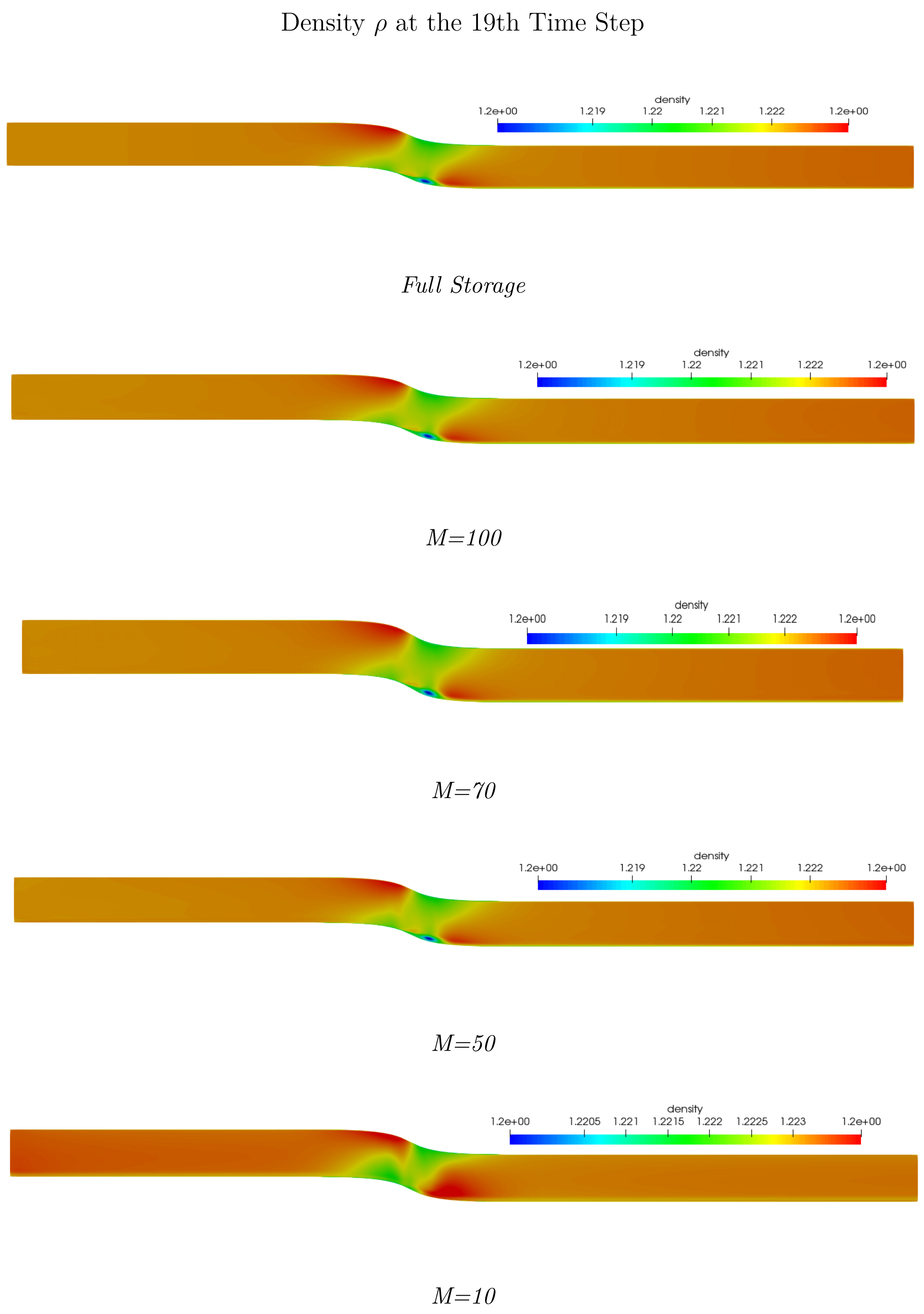
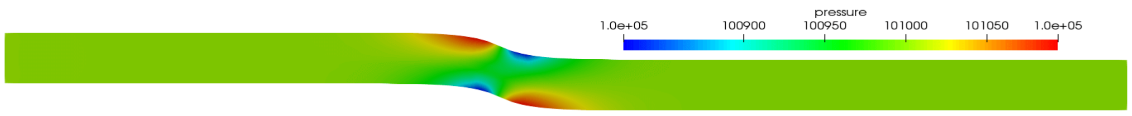
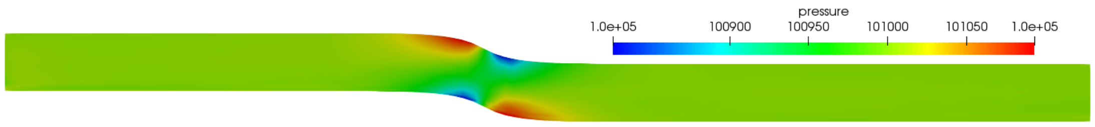


Figure C.4: *Flow through a 2D S-shaped pipe. Density at the 19th time step. CFD results and iPGD approximations for* $M = 100, 70, 50$ *and* 10.

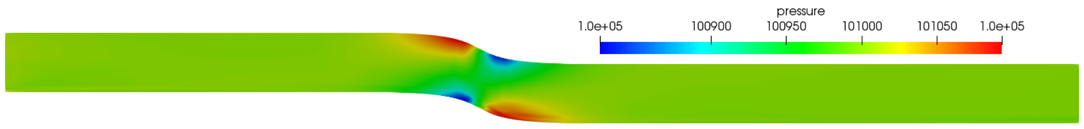
Pressure P at the 1st Time Step



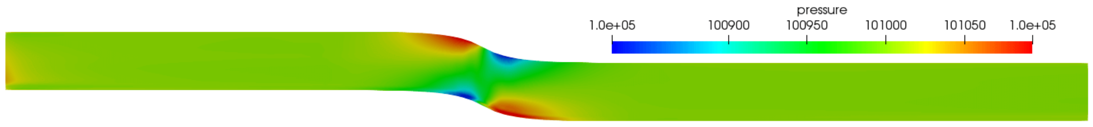
Full Storage



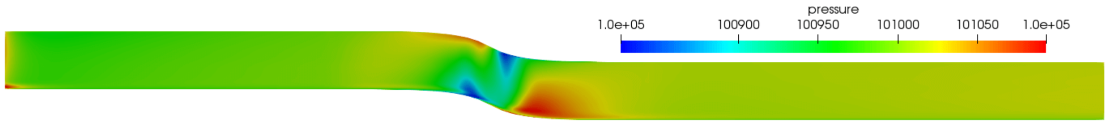
M=100



M=70



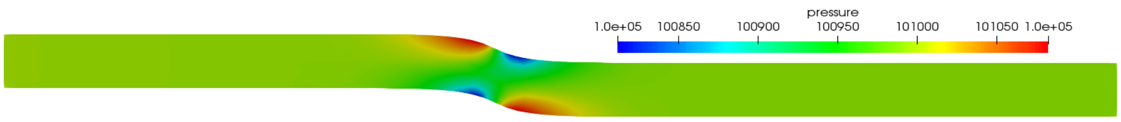
M=50



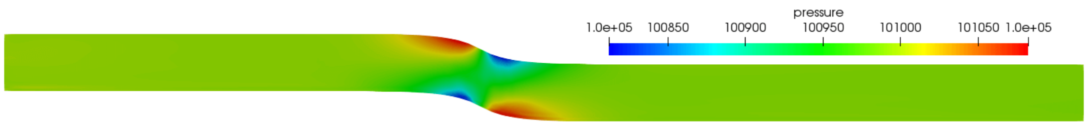
M=10

Figure C.5: *Flow through a 2D S-shaped pipe. Pressure at the* 1*st time step. CFD results and iPGD approximations for* $M = 100, 70, 50$ *and* 10.

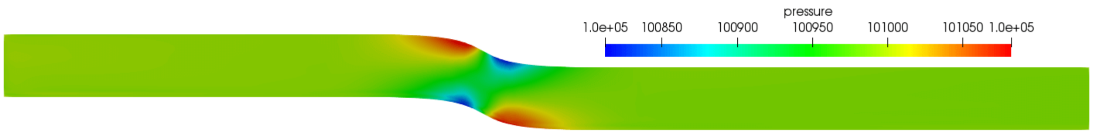
Pressure P at the 7th Time Step



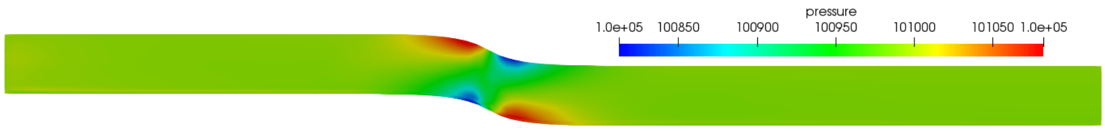
Full Storage



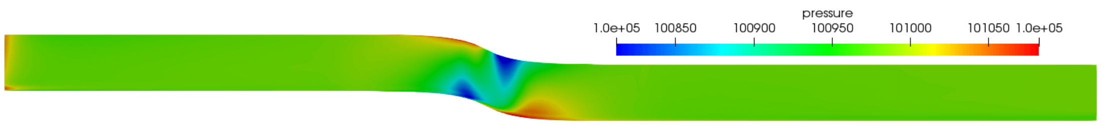
M=100



M=70



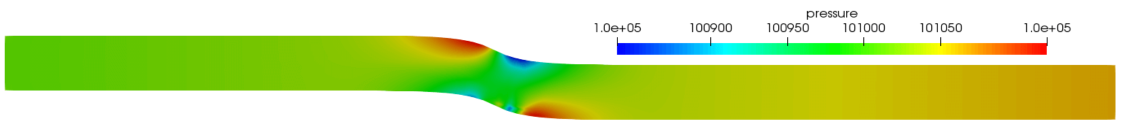
M=50



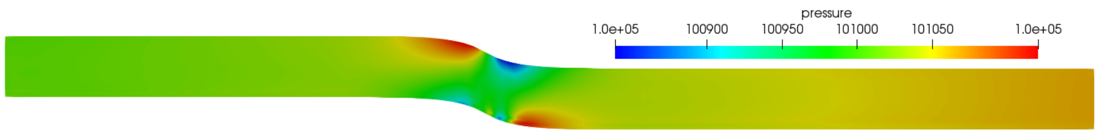
M=10

Figure C.6: *Flow through a 2D S-shaped pipe. Pressure at the 7th time step. CFD results and iPGD approaches for* 100*,* 70*,* 50 *and* 10 *summation terms.*

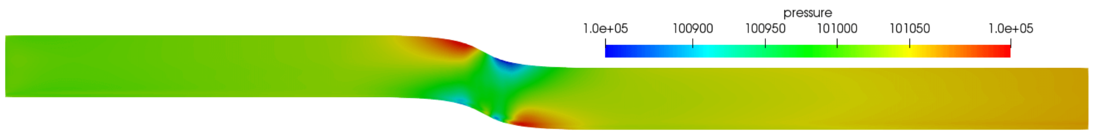
Pressure P at the 13th Time Step



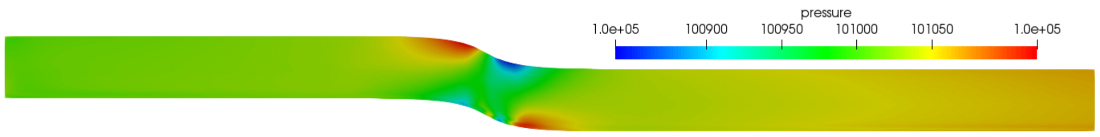
Full Storage



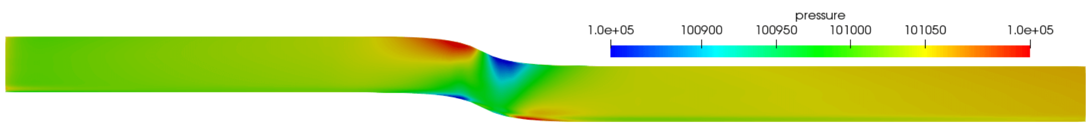
M=100



M=70



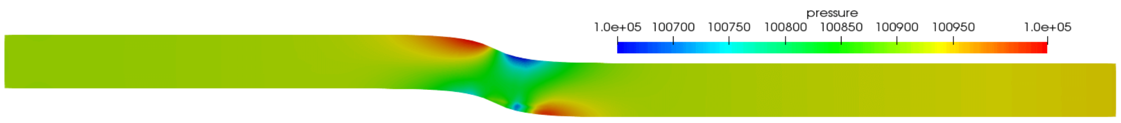
M=50



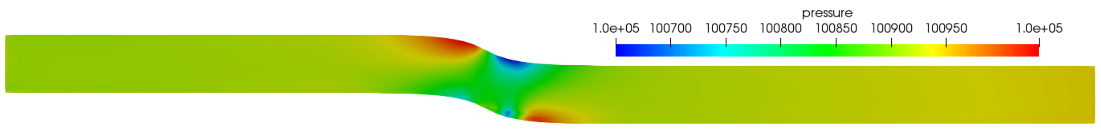
M=10

Figure C.7: *Flow through a 2D S-shaped pipe. Pressure at the* 13*th time step. CFD results and iPGD approximations for* $M = 100, 70, 50$ *and* 10.

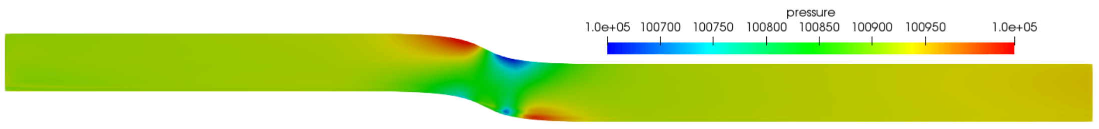
Pressure P at the 19th Time Step



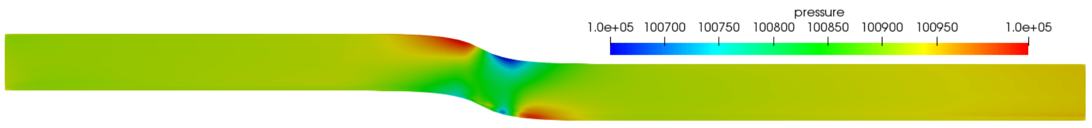
Full Storage



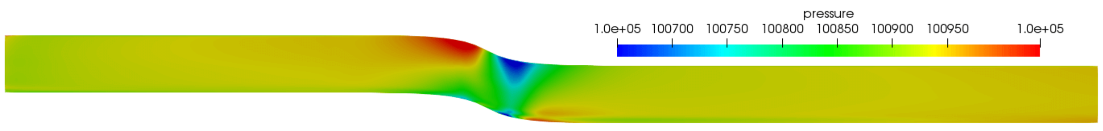
M=100



M=70



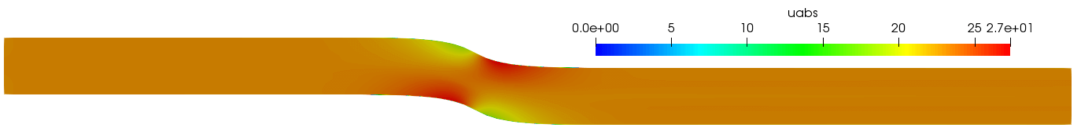
M=50



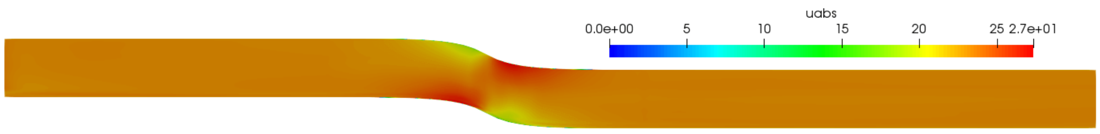
M=10

Figure C.8: *Flow through a 2D S-shaped pipe. Pressure at the* 19*th time step. CFD results and iPGD approximations for* $M = 100, 70, 50$ *and* 10*.*

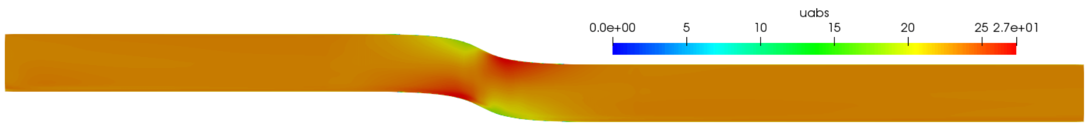
Velocity u at the 1st Time Step



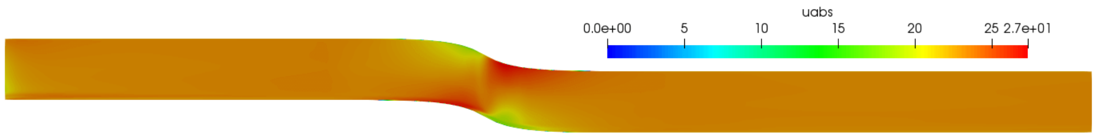
Full Storage



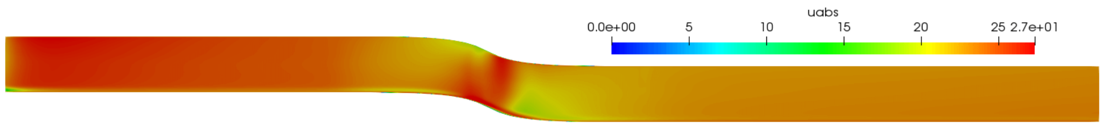
M=100



M=70



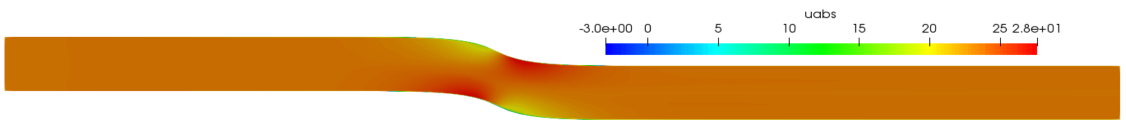
M=50



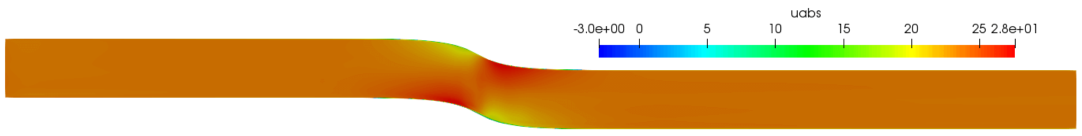
M=10

Figure C.9: *Flow through a 2D S-shaped pipe. Velocity u at the 1st time step. CFD results and iPGD approximations for $M = 100, 70, 50$ and 10.*

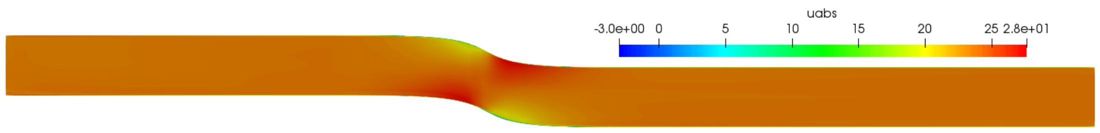
Velocity u at the 7th Time Step



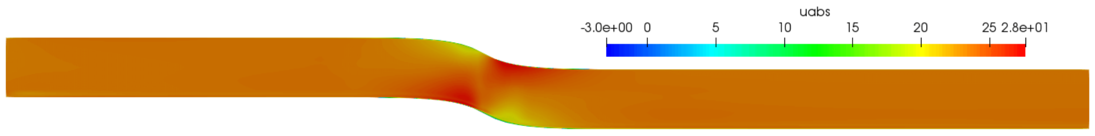
Full Storage



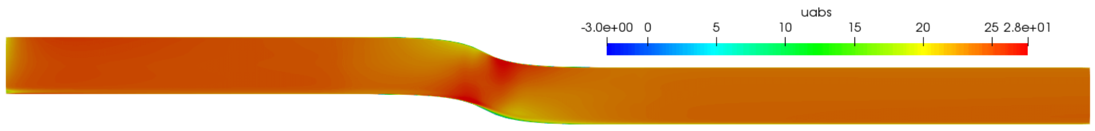
M=100



M=70



M=50



M=10

Figure C.10: *Flow through a 2D S-shaped pipe. Velocity u at the 7th time step. CFD results and iPGD approximations for $M = 100, 70, 50$ and 10.*

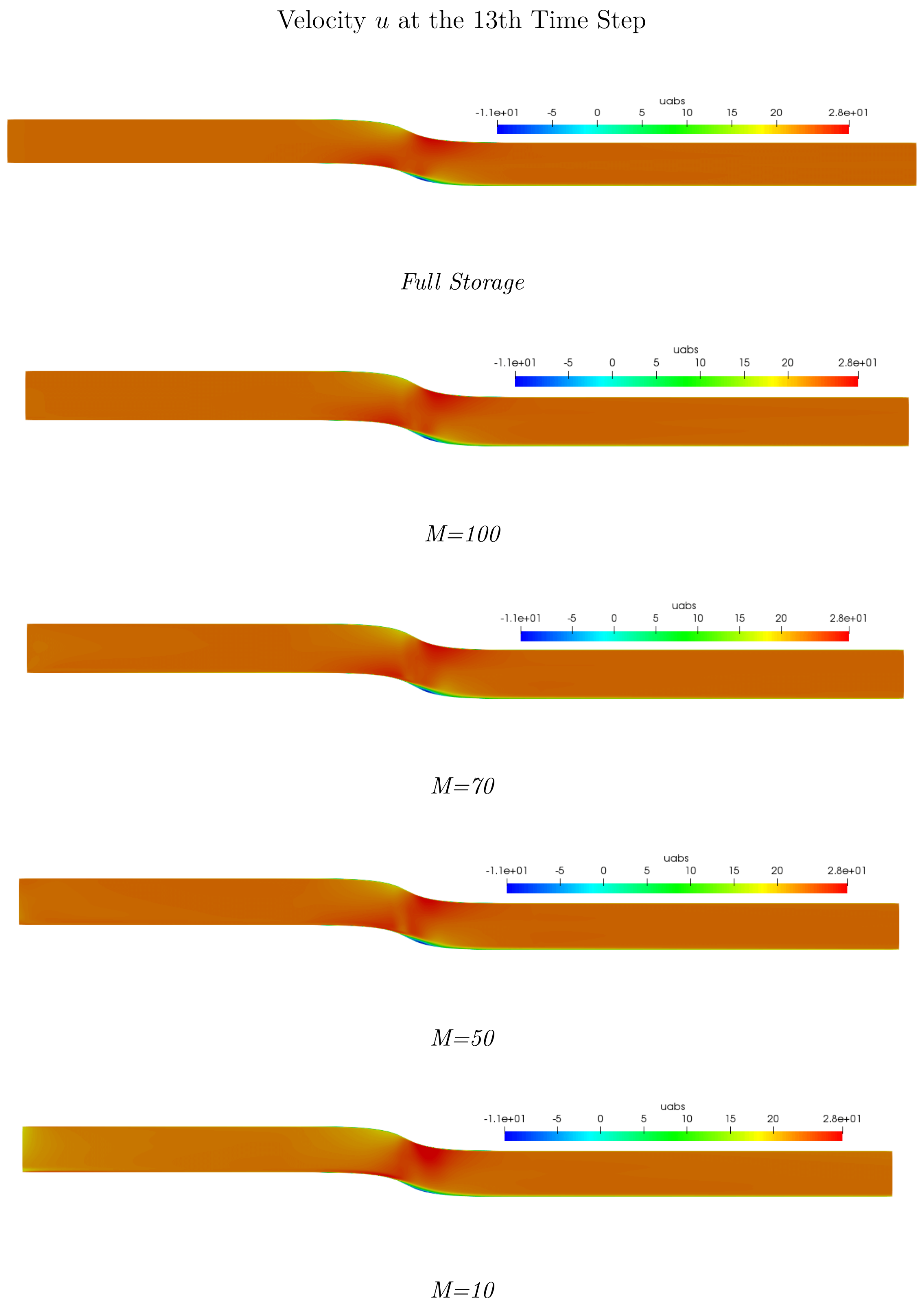


Figure C.11: *Flow through a 2D S-shaped pipe. Velocity u at the 13th time step. CFD results and iPGD approximations for $M = 100, 70, 50$ and 10.*

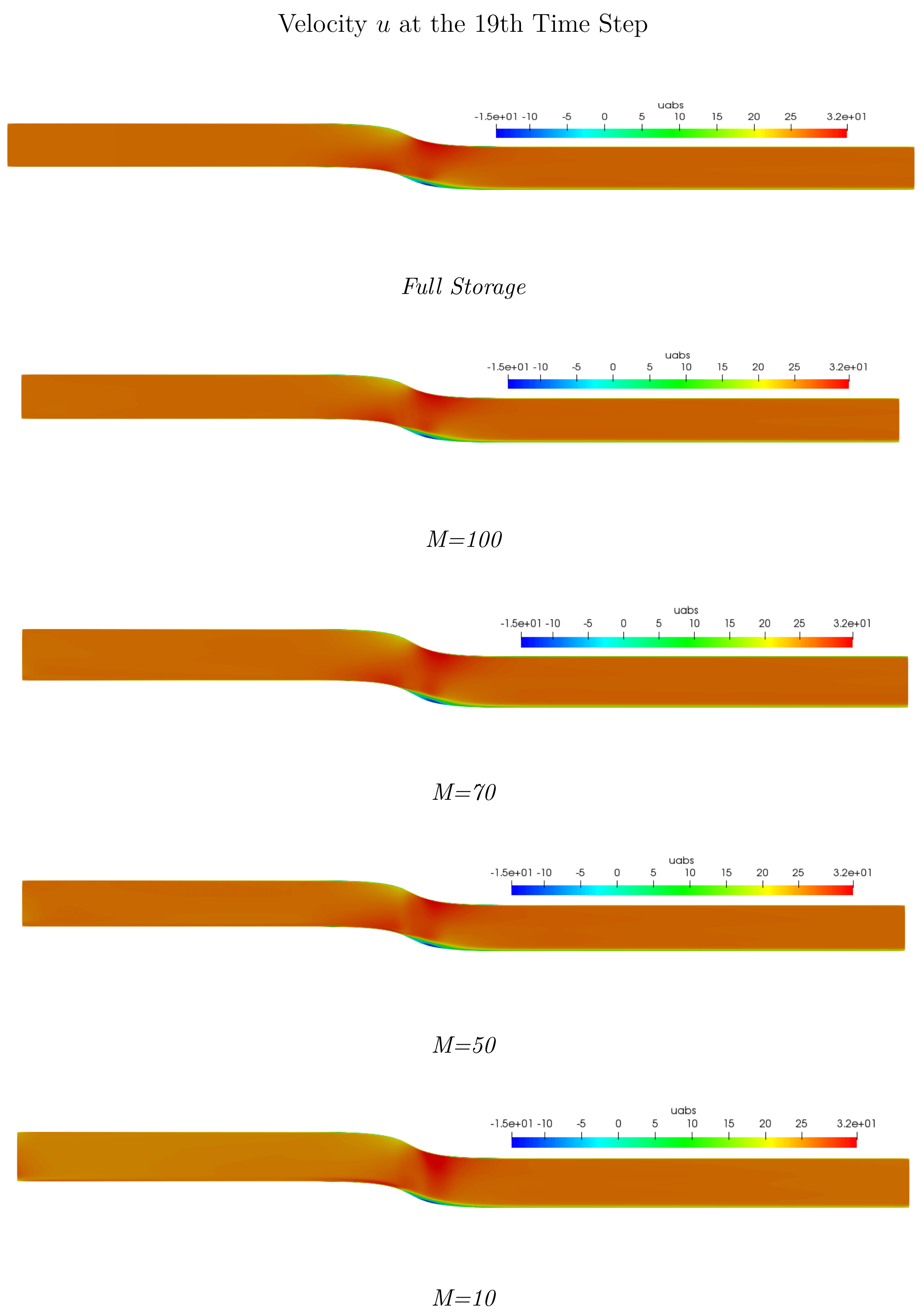
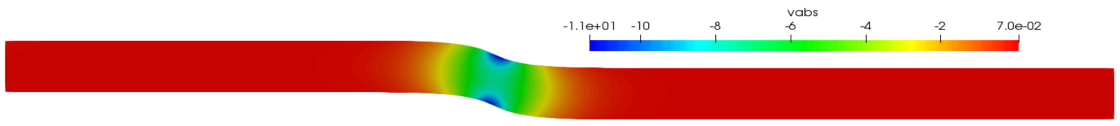
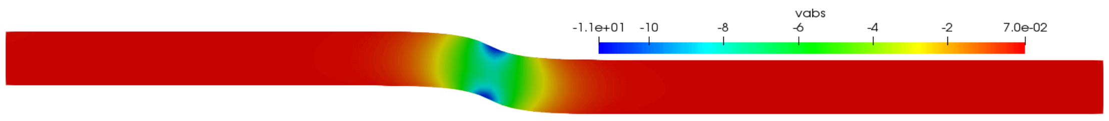


Figure C.12: *Flow through a 2D S-shaped pipe. Velocity u at the 19th time step. CFD results and iPGD approximations for $M = 100, 70, 50$ and 10.*

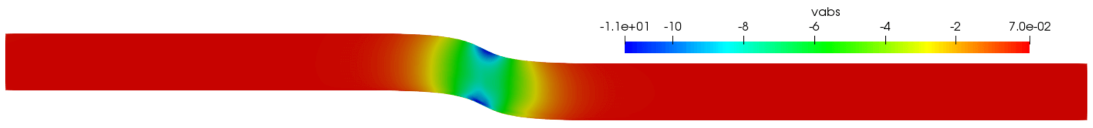
Velocity v at the 1st Time Step



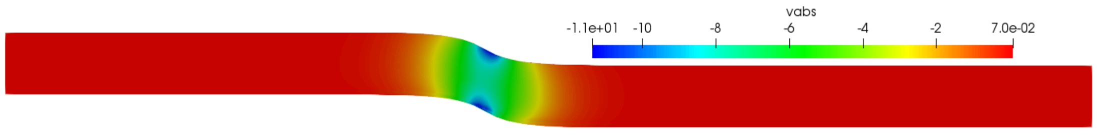
Full Storage



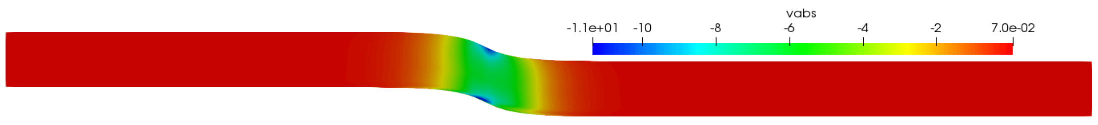
M=100



M=70



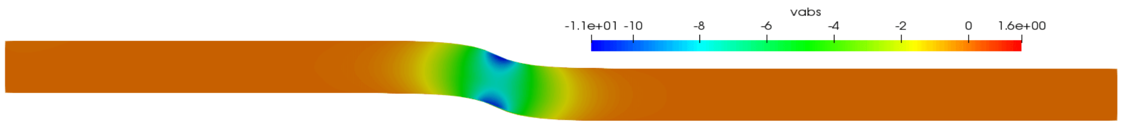
M=50



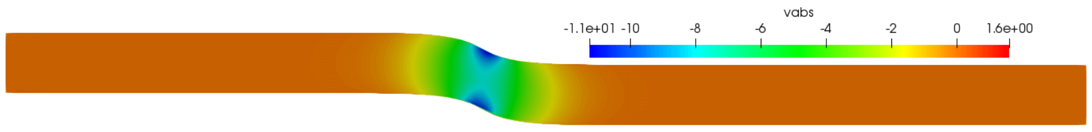
M=10

Figure C.13: *Flow through a 2D S-shaped pipe. Velocity v at the 1st time step. CFD results and iPGD approximations for $M = 100, 70, 50$ and 10.*

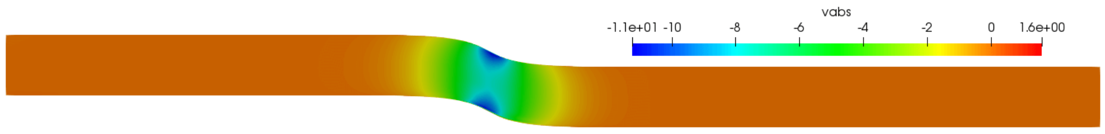
Velocity v at the 7th Time Step



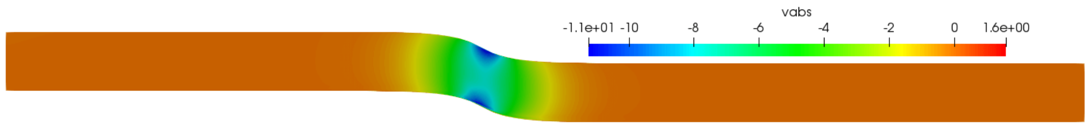
Full Storage



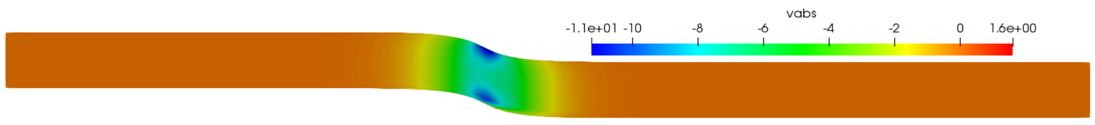
M=100



M=70



M=50



M=10

Figure C.14: *Flow through a 2D S-shaped pipe. Velocity v at the 7th time step. CFD results and iPGD approximations for $M = 100, 70, 50$ and 10.*

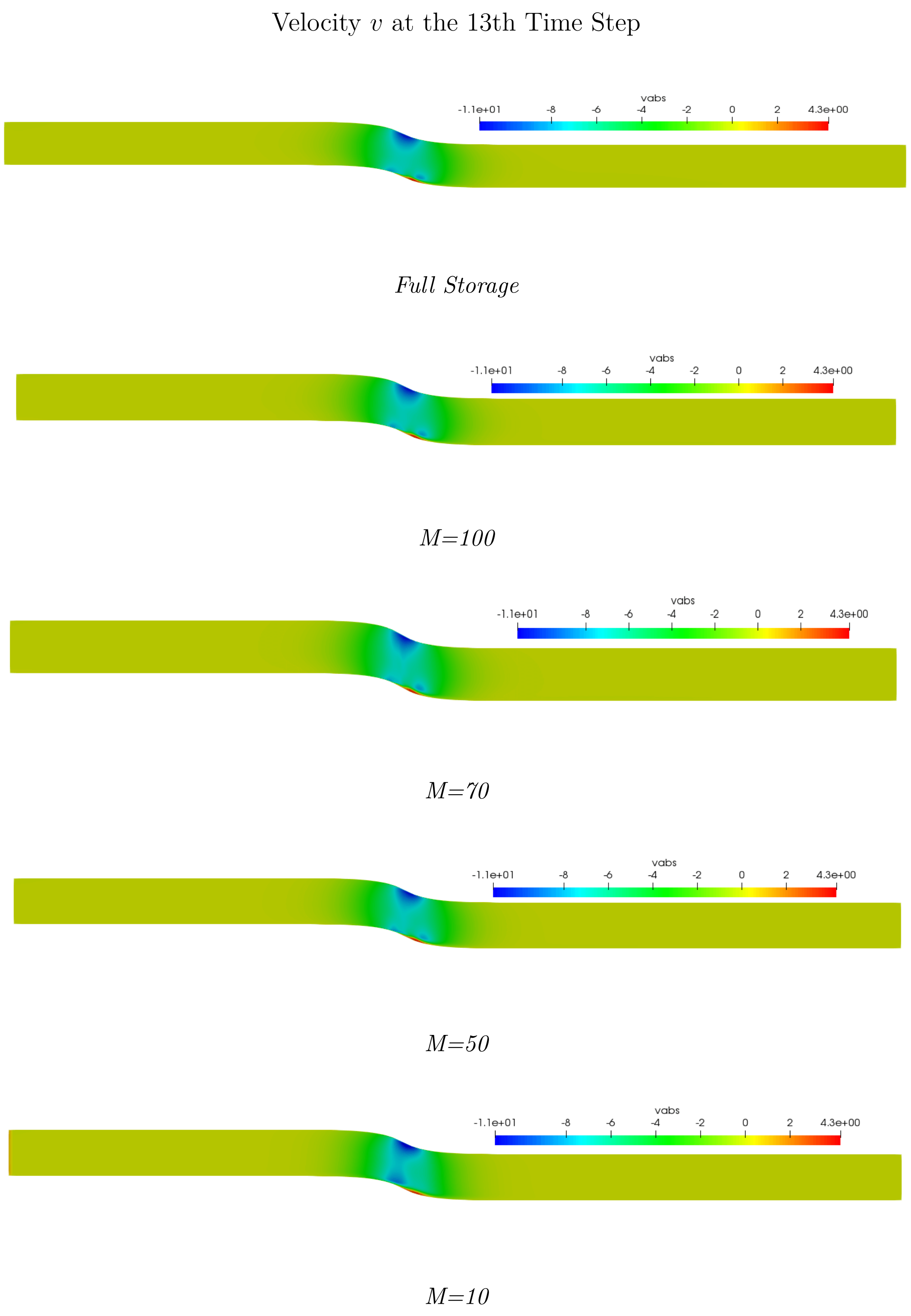


Figure C.15: *Flow through a 2D S-shaped pipe. Velocity v at the 13th time step. CFD results and iPGD approximations for $M = 100, 70, 50$ and 10.*

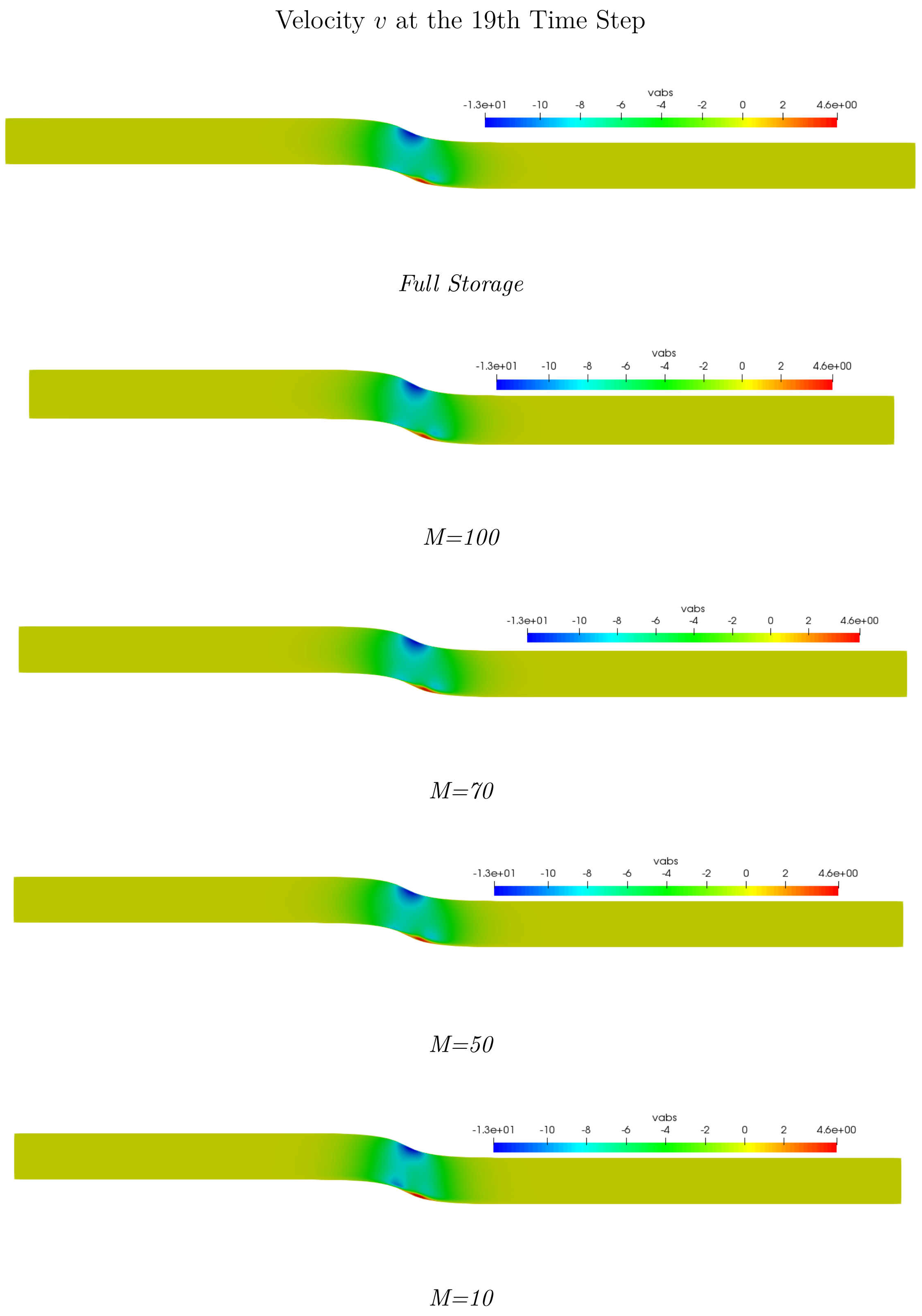


Figure C.16: *Flow through a 2D S-shaped pipe. Velocity v at the 19th time step. CFD results and iPGD approximations for $M = 100, 70, 50$ and 10.*

Appendix D

Appendix: Unsteady Flow through a 3D U-Shaped Pipe

For the flow in a 3D U-Shaped Pipe with square cross section, the flow fields of density ρ, pressure P and velocities u, v and w (in x, y and z directions respectively) are computed and then approximated by iPGD with 135 and 200 approximation sum terms.

In the following figures the CFD results are compared with the iPGD approximations. The visualization of the fields is achieved in 2D coloured maps. In this figures the colour scale represents the value of the field in xy-plane for specific time steps (1st and 25th) and specific value of the variable z (at 1/2 and 1/8 of its total). It is impossible to fully visualize such a large-scale 4D (3 spacial and 1 temporal) field on the paper, for this reason figures presented here are the ones with the most abrupt changes of their values, thus the most difficult to be approximated.

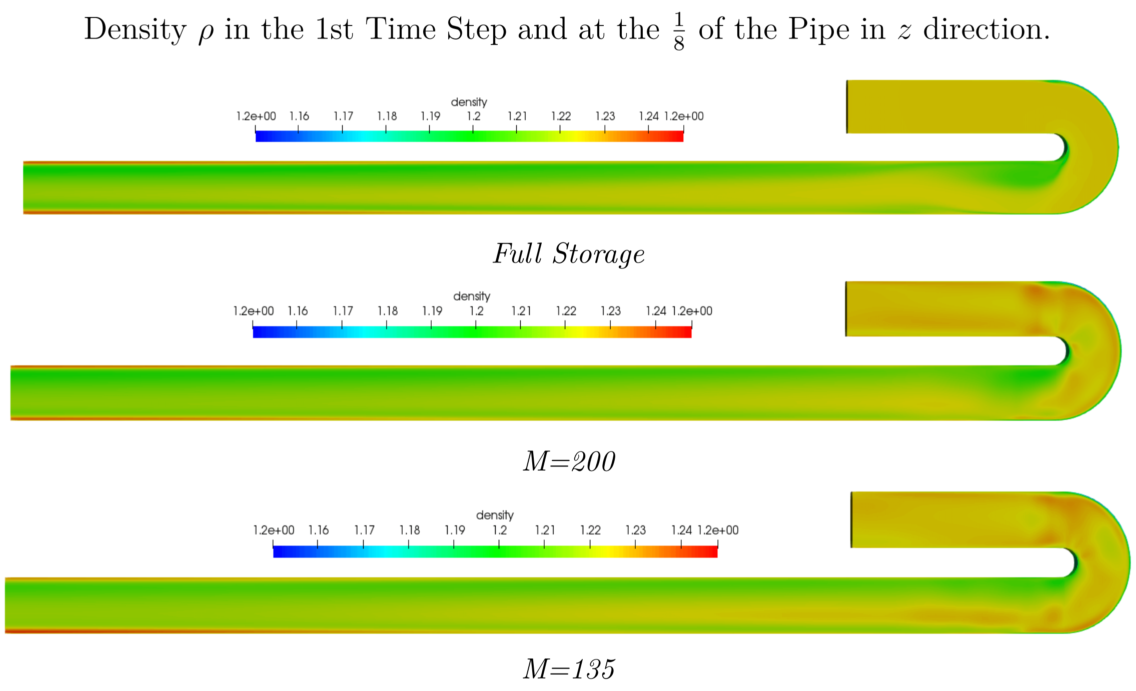


Figure D.1: *Flow through a 3D U-shaped pipe. Density for the* 1*st time step at the* $\frac{1}{8}$ *of the pipe in* z *direction. CFD results and iPGD approximations for* $M = 135$ *and* $M = 200$.

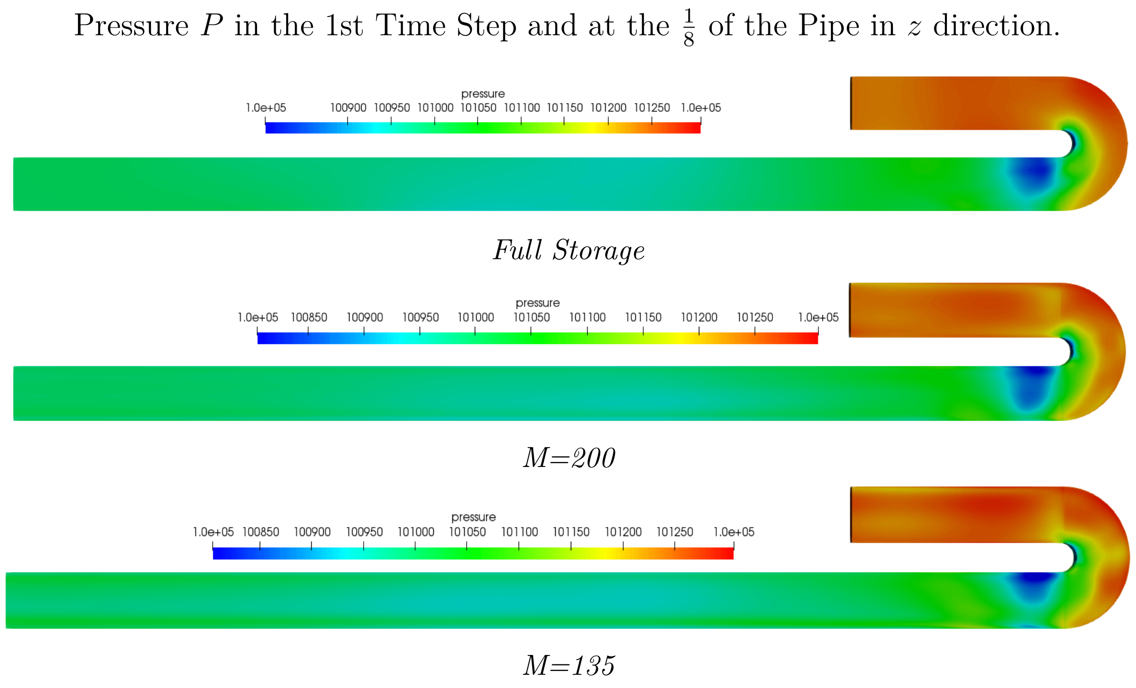


Figure D.2: *Flow through a 3D U-shaped pipe. Pressure for the* 1*st time step at the* $\frac{1}{8}$ *of the pipe in* z *direction. CFD results and iPGD approximations for* $M = 135$ *and* $M = 200$.

Velocity u in the 1st Time Step and at the $\frac{1}{8}$ of the Pipe in z direction.

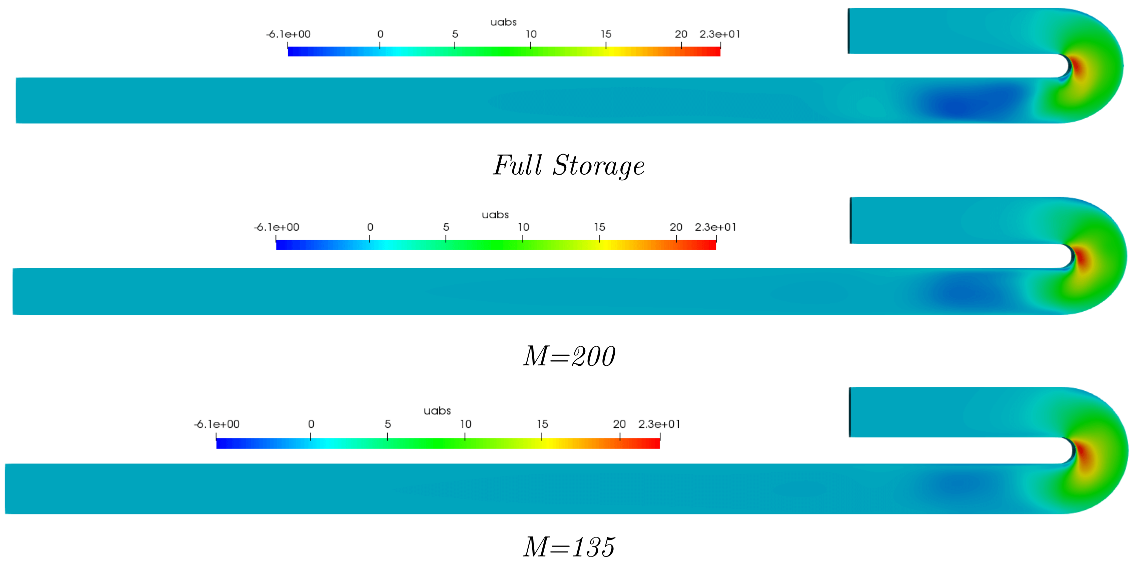


Figure D.3: *Flow through a 3D U-shaped pipe. Velocity u for the 1st time step at the $\frac{1}{8}$ of the pipe in z direction. CFD results and iPGD approximations for $M = 135$ and $M = 200$.*

Velocity v in the 1st Time Step and at the $\frac{1}{8}$ of the Pipe in z direction.

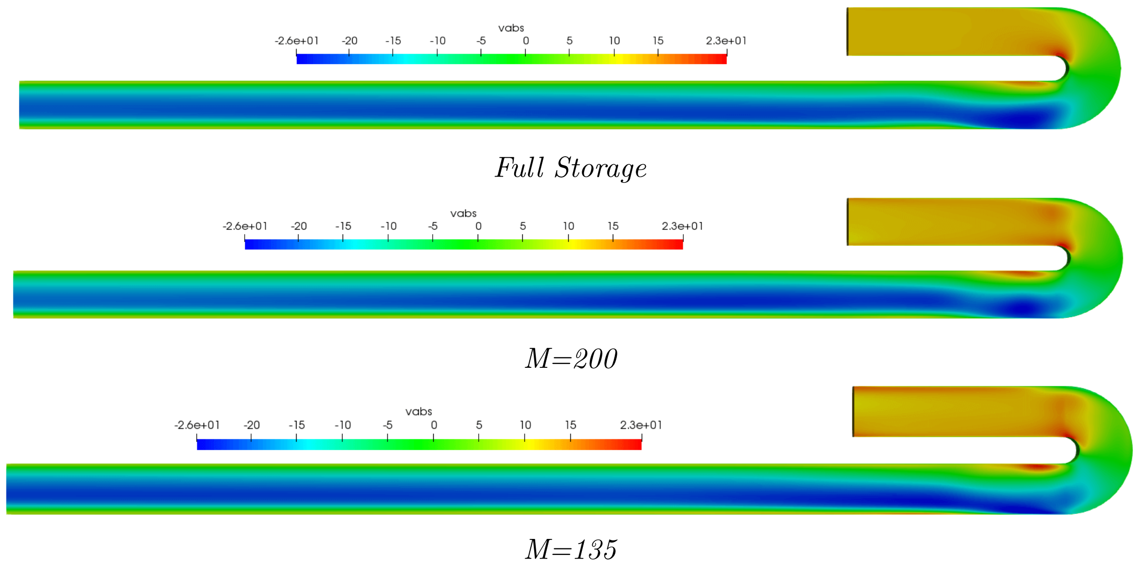


Figure D.4: *Flow through a 3D U-shaped pipe. Velocity v for the 1st time step at the $\frac{1}{8}$ of the pipe in z direction. CFD results and iPGD approximations for $M = 135$ and $M = 200$.*

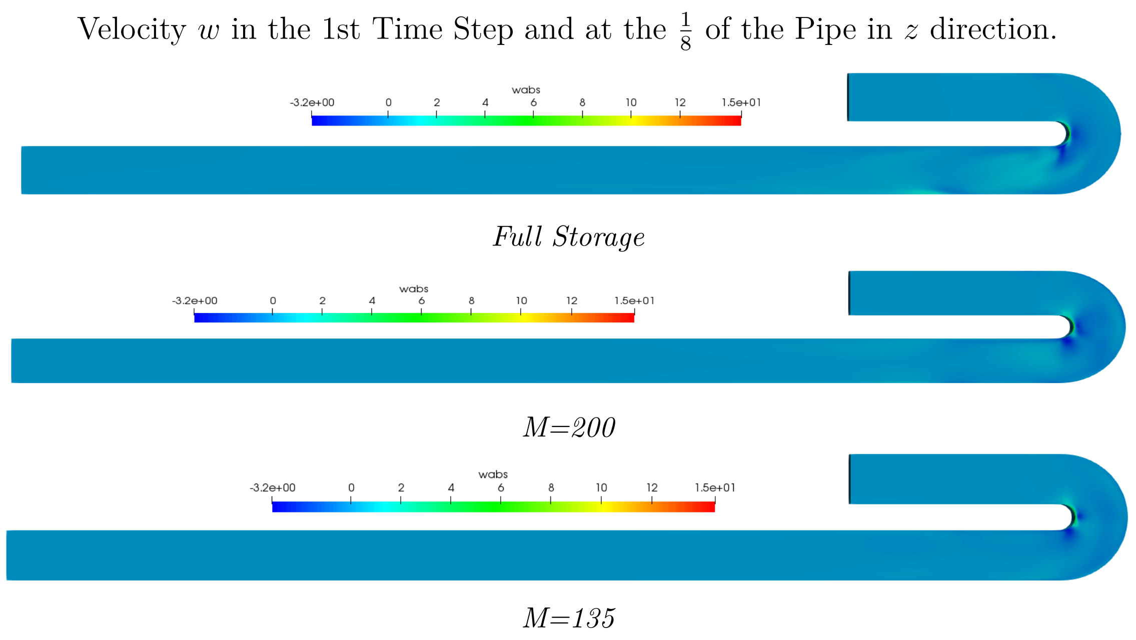


Figure D.5: *Flow through a 3D U-shaped pipe. Velocity w for the 1st time step at the $\frac{1}{8}$ of the pipe in z direction. CFD results and iPGD approximations for $M = 135$ and $M = 200$.*

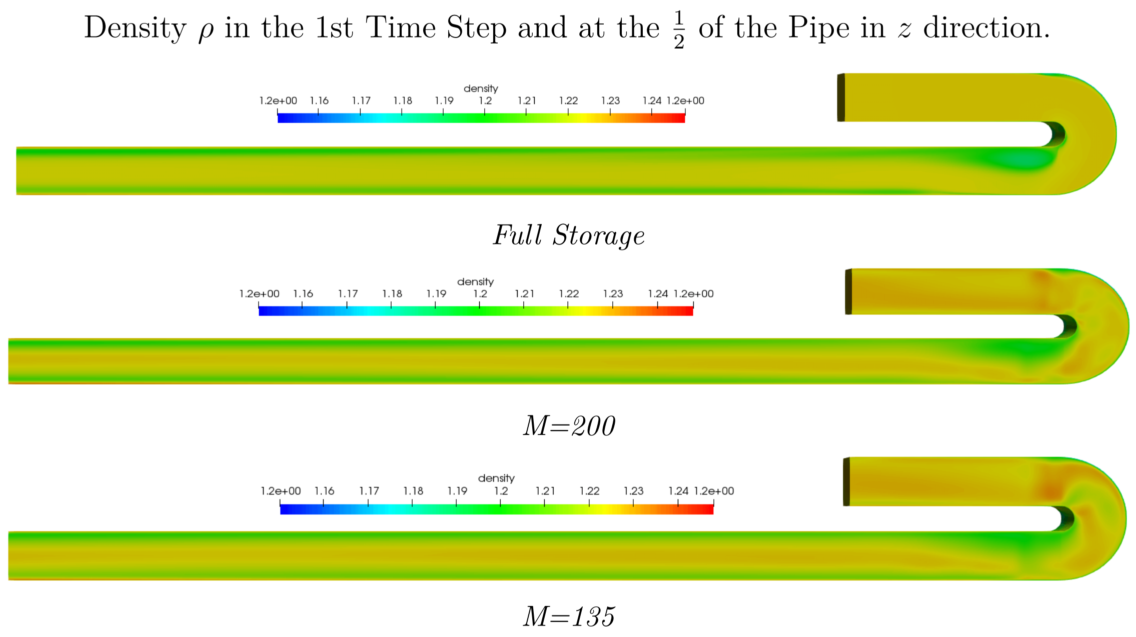


Figure D.6: *Flow through a 3D U-shaped pipe. Density for the 1st time step at the $\frac{1}{2}$ of the pipe in z direction. CFD results and iPGD approximations for $M = 135$ and $M = 200$.*

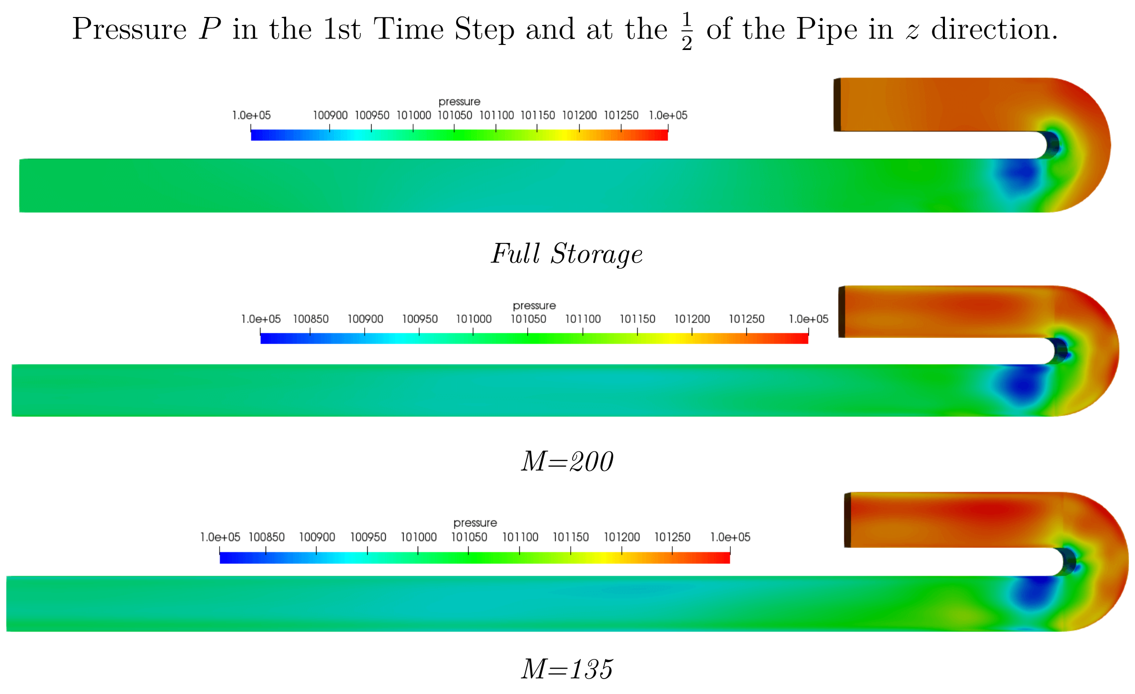


Figure D.7: *Flow through a 3D U-shaped pipe. Pressure for the* 1*st time step at the* $\frac{1}{2}$ *of the pipe in* z *direction. CFD results and iPGD approximations for* $M = 135$ *and* $M = 200$.

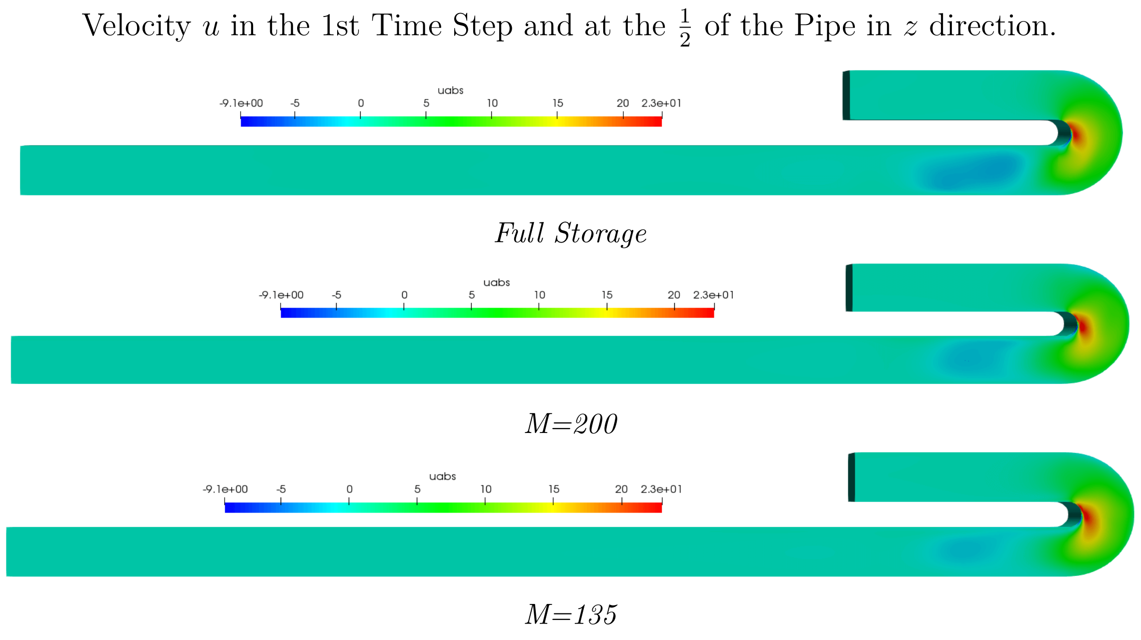


Figure D.8: *Flow through a 3D U-shaped pipe. Velocity u for the* 1*st time step at the* $\frac{1}{2}$ *of the pipe in* z *direction. CFD results and iPGD approximations for* $M = 135$ *and* $M = 200$.

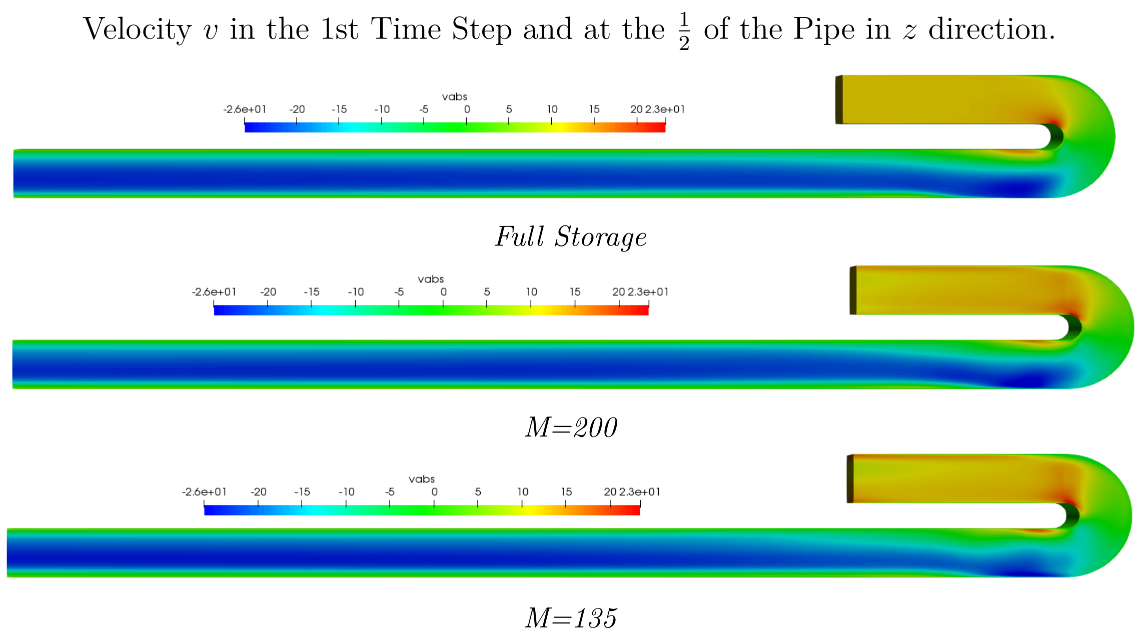


Figure D.9: *Flow through a 3D U-shaped pipe. Velocity v for the 1st time step at the $\frac{1}{2}$ of the pipe in z direction. CFD results and iPGD approximations for $M = 135$ and $M = 200$.*

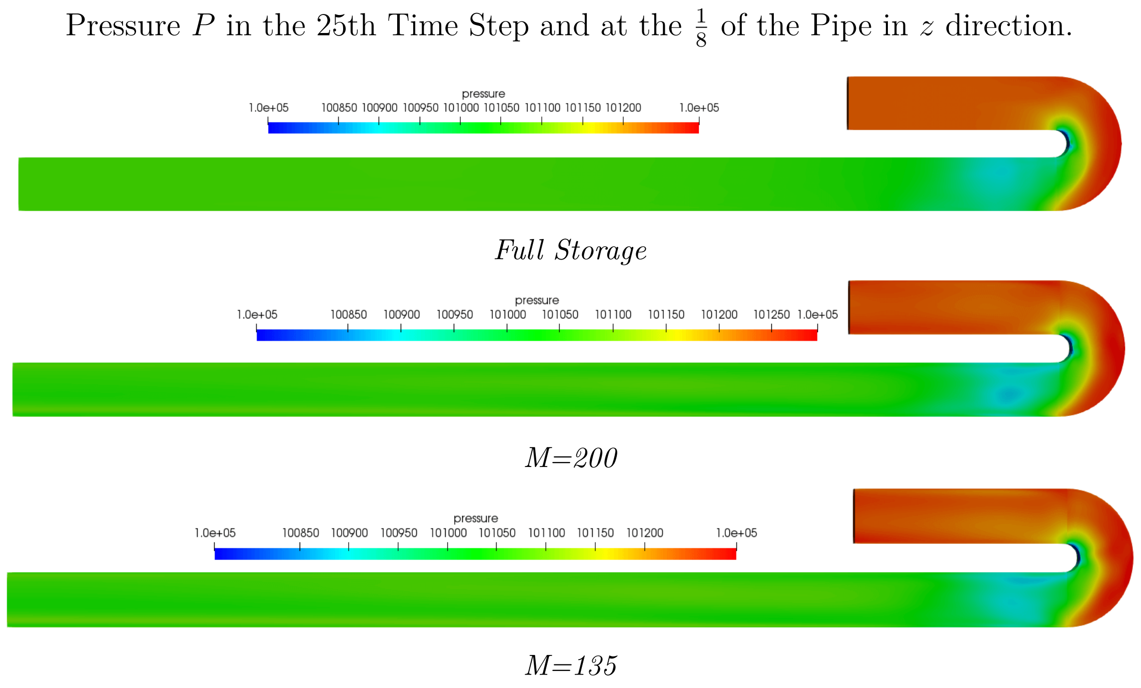


Figure D.10: *Flow through a 3D U-shaped pipe. Pressure for the 25th time step at the $\frac{1}{8}$ of the pipe in z direction. CFD results and iPGD approximations for $M = 135$ and $M = 200$.*

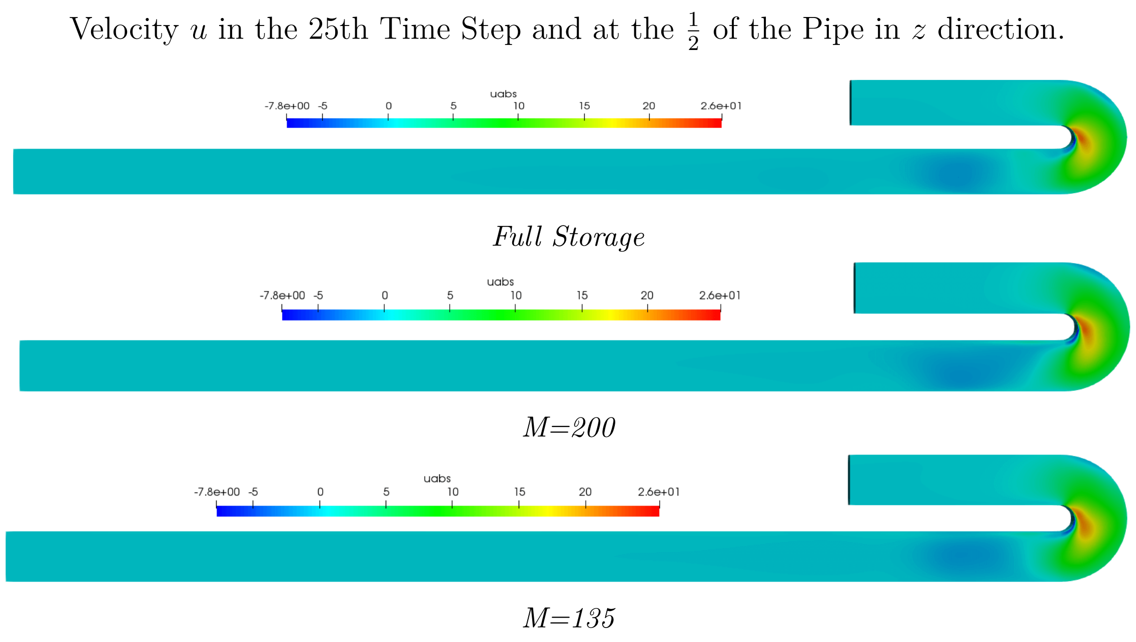


Figure D.11: *Flow through a 3D U-shaped pipe. Velocity u for the 25th time step at the $\frac{1}{8}$ of the pipe in z direction. CFD results and iPGD approximations for $M = 135$ and $M = 200$.*

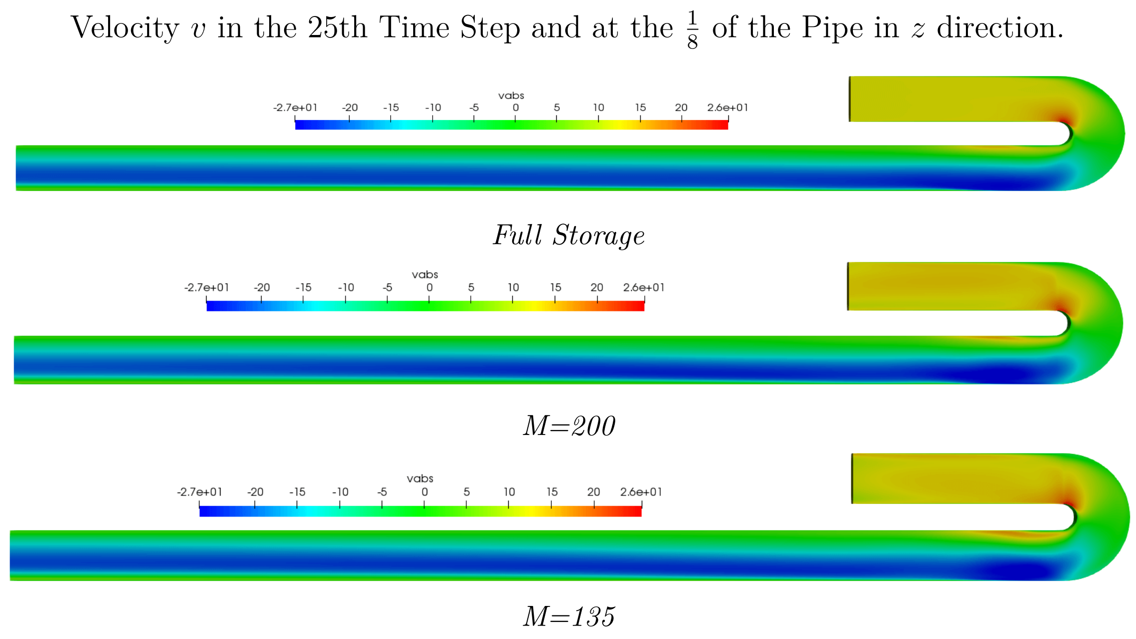


Figure D.12: *Flow through a 3D U-shaped pipe. Velocity v for the 25th time step at the $\frac{1}{8}$ of the pipe in z direction. CFD results and iPGD approximations for $M = 135$ and $M = 200$.*

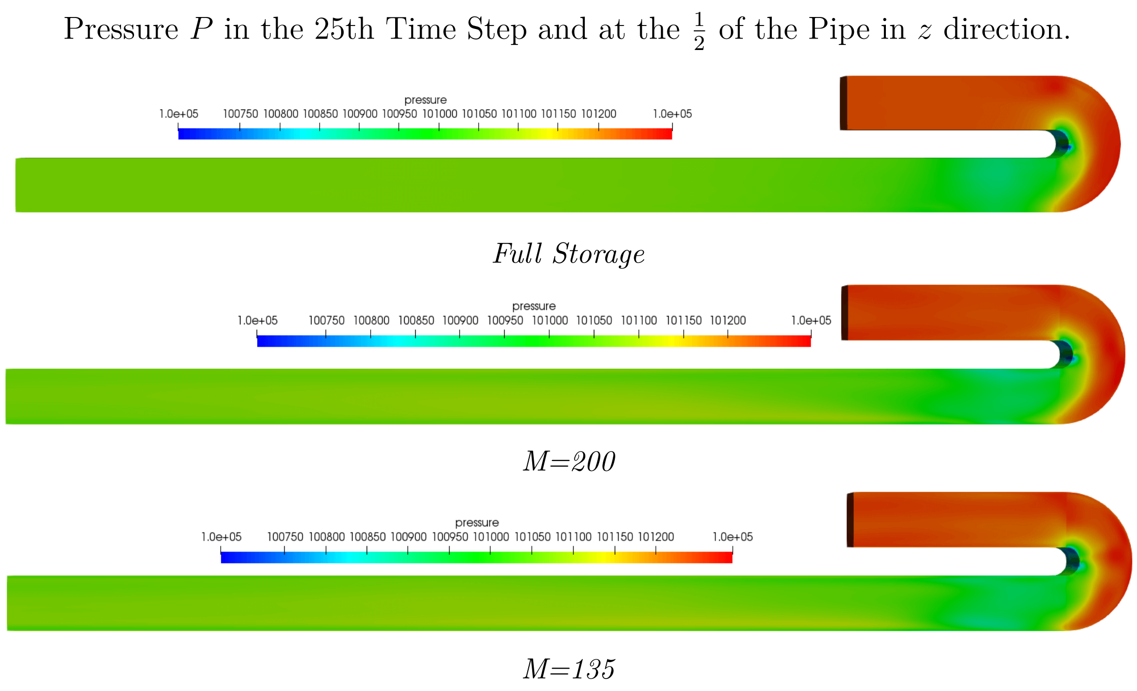

Pressure P in the 25th Time Step and at the $\frac{1}{2}$ of the Pipe in z direction.
pressure
1.0e+05 100750 100800 100850 100900 100950 101000 101050 101100 101150 101200 1.0e+05
Full Storage
M=200
M=135

Figure D.13: *Flow through a 3D U-shaped pipe. Pressure for the 25th time step at the* $\frac{1}{2}$ *of the pipe in z direction. CFD results and iPGD approximations for* $M = 135$ *and* $M = 200$.

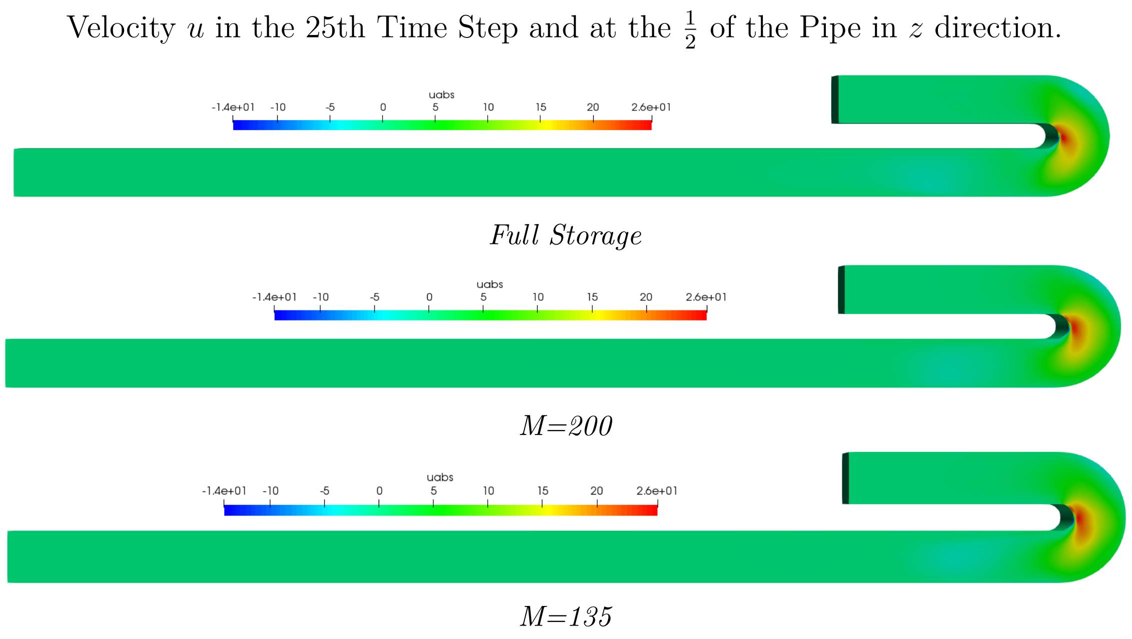
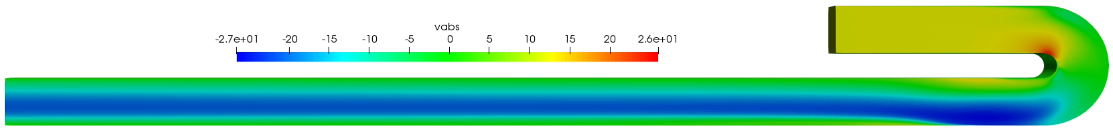
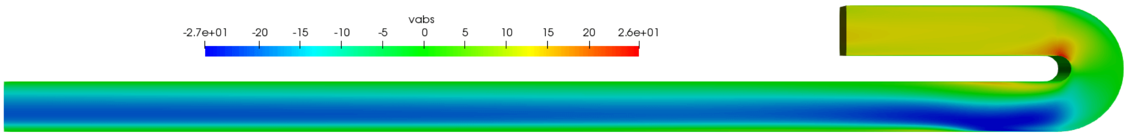

Velocity u in the 25th Time Step and at the $\frac{1}{2}$ of the Pipe in z direction.
uabs
-1.4e+01 -10 -5 0 5 10 15 20 2.6e+01
Full Storage
M=200
M=135

Figure D.14: *Flow through a 3D U-shaped pipe. Velocity u for the 25th time step at the* $\frac{1}{2}$ *of the pipe in z direction. CFD results and iPGD approximations for* $M = 135$ *and* $M = 200$.

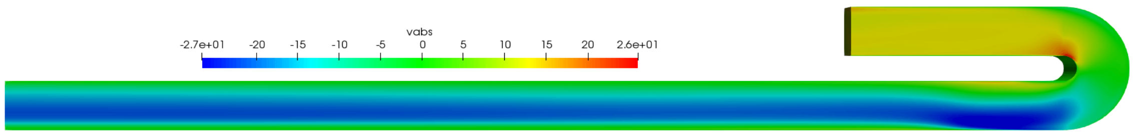
Velocity v in the 25th Time Step and at the $\frac{1}{2}$ of the Pipe in z direction.



Full Storage



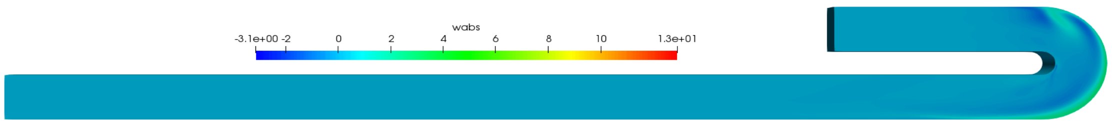
M=200



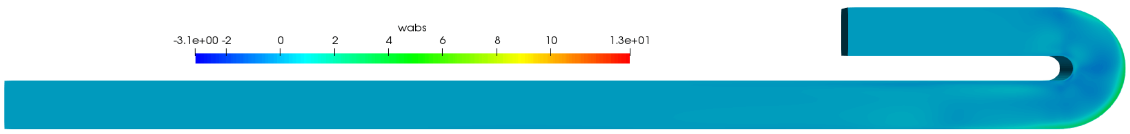
M=135

Figure D.15: *Flow through a 3D U-shaped pipe. Velocity v for the 25th time step at the $\frac{1}{2}$ of the pipe in z direction. CFD results and iPGD approximations for $M = 135$ and $M = 200$.*

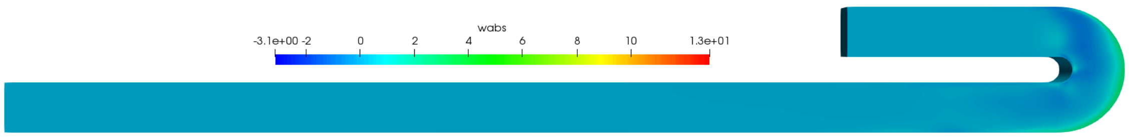
Velocity w in the 25th Time Step and at the $\frac{1}{2}$ of the Pipe in z direction.



Full Storage



M=200



M=135

Figure D.16: *Flow through a 3D U-shaped pipe. Velocity w for the 25th time step at the $\frac{1}{8}$ of the pipe in z direction. CFD results and iPGD approximations for $M = 135$ and $M = 200$.*

Bibliography

1. F. Chinesta, R. Keunings, and A. Leygue: *The Proper Generalized Decomposition for Advanced numerical Simulations, A Primer.* Springer International Publishing, Nantes, France, 2014

2. Papageorgiou, V.: *Programming of the Proper Generalized Decomposition Method for the Prediction and/or Compression of the Solution of 2D Steady and Unsteady PDEs. Applications in Adjoint-Based Optimization.* Diploma thesis, Laboratory of Thermal Turbomachines, NTUA, Athens, 2017.

3. V. Papageorgiou, K. Samouchos, and K.C.Giannakoglou: *The Unsteady Continuous Adjoint Method Assisted by the Proper Generalized Decomposition Method.* In *EUROGEN International Conference on Evolutionary and Deterministic Methods for Design Optimization and Control with Applications to Industrial and Societal Problems*, Madrid, Spain, Sep 13-15 2017

4. Giannakoglou, K.C.: *Optimization Methods in Aerodynamics.* Notes. Laboratory of Thermal Turbomachines, NTUA, Athens, 2006.

5. Giannakoglou, K.C.: *Numerical Analysis for Engineers.* Notes. Laboratory of Thermal Turbomachines, NTUA, Athens, 2003.

6. Kapsoulis, D. & Tsiakas, K. & Trompoukis, X. & Asouti, V. & Giannakoglou, K. (2017). *Evolutionary Multi-Objective Optimization Assisted by Metamodels, Kernel PCA and Multi-Criteria Decision Making Techniques with Applications in Aerodynamics.* Applied Soft Computing. 64.

7. He, T. & Zhu, D. & Wang, J. & Wang, Q. (2016). *Experimental and Numerical Investigations of the Stribeck Curves for Lubricated Counterformal Contacts.* Journal of Tribology. 139. 10.1115/1.4034051.

8. Mendonça, A.C.F. & Pádua, A.C.F. & Malfreyt, P. (2013). *Nonequilibrium Molecular Simulations of New Ionic Lubricants at Metallic Surfaces: Prediction of the Friction.* J. Chem. Theory Comput. 9(3), 1600–1610.

9. Frenkel, D. & Smit, B.: *Understanding molecular simulation : from algorithms to applications*, Academic Press, 1996.

10. D. C Rapaport *The art of molecular dynamics simulation*, Cambridge University Press, 2004

11. Gibbs, J. W. *Elementary Principles in Statistical Mechanics, developed with especial reference to the rational foundation of thermodynamics*, Charles Scribner's Sons, 1902, New York

12. Nosé, Sh. (1984). *A Molecular Dynamics Method for Simulations in the Canonical Ensemble*. Mol. Phys. 52. 255-268.

13. Nosé, Sh. (1984). *A Unified Formulation of the Constant Temperature Molecular Dynamics Method*. J Chem. Phys. 81. 511.

14. Hoover, W. (1985). *Canonical Dynamics: Equilibrium Phase-Space Distributions*. Phys. Rev. A. 31. 1695.

15. Shinoda, W. & Shiga, M & Mikami, M. (2004). *Rapid estimation of elastic constants by molecular dynamics simulation under constant stress*. Phys. Rev. B. 69. 134103

16. Martyna, G. & Tobias, D. J. & Klein, M. L. (1994). *Constant-Pressure Molecular-Dynamics Algorithms*. J. Chem. Phys. 101(5) .

17. Tuckerman, M. & Alejandre, J. & López-Rendón, R. & Jochim, A. L. & Martyna, G. (2006). *A Liouville-operator derived measure-preserving integrator for molecular dynamics simulations in the isothermal–isobaric ensemble*. J. Phys. A. 39. 5629.

18. Parrinello, MRA & Rahman, A. J. (1982). *Polymorphic Transitions in Single Crystals: A New Molecular Dynamics Method*. J. Appl. Phys. 52. 7182 - 7190.

19. Steinhauser, M. *Introduction to Molecular Dynamics Simulations: Applications in Hard and Soft Condensed Matter Physics*. 2012.

20. Merlet, C. & Salanne, M. & Rotenberg, B. (2012). *New Coarse-Grained Models of Imidazolium Ionic Liquids for Bulk and Interfacial Molecular Simulations*. J. Phys. Chem. C. 116. 7687-7693.

21. Zhang, Y. & Otani, A. & Maginn, E. (2015). *Reliable Viscosity Calculation from Equilibrium Molecular Dynamics Simulations: A Time Decomposition Method*. J. Chem. Theory Comput. 11. 150707115106005.

22. Tokuda, H. & Hayamizu, K. & Ishii, K. & Susan, Md & Watanabe, M. (2004). *Physicochemical Properties and Structures of Room Temperature Ionic Liquids. 1. Variation of Anionic Species*. J. Phys. Chem. B. 108(42).

23. Gao, J. & Wagner, N. (2016). *Non-ideal viscosity and excess molar volume of mixtures of 1-butyl-3-methylimidazolium tetrafluoroborate ([C4mim][BF4]) with water*. J. Mol. Liq. 223. 10.1016

24. Chaban, V. & Voroshylova, I. & Kalugin, O. (2011). *A new force field model for the simulation of transport properties of imidazolium-based ionic liquids.* Phys. chem. chem. phys. 13(17).7910-20.

25. Stankovic, I. & Dasic, M. & Gkagkas, K. (2019). *Molecular Dynamics Investigation of the Influence of the Shape of Cation on the Structure and Lubrication Properties of Ionic Liquids.* Phys. Chem. Chem. Phys. 21.

26. Dasic, M. & Gkagkas, K. & Stankovic, I. (2018). *Influence of Confinement on Flow and Lubrication Properties of a Salt Model Ionic Liquid Investigated with Molecular Dynamics* (published in EPJE doi: 10.1140/epje/i2018-11740-6).

27. P. A. Revvel: *Joint Replacement Technology*, Woodhead Publishing in Materials, 2008

28. Dupont, J. & de Souza, R. F. & Suarez, P. (2003). *Ionic Liquid (Molten Salt) Phase Organometallic Catalysis.* Chemical reviews. 102. 3667-92.

29. S. Plimpton, *Fast Parallel Algorithms for Short-Range Molecular Dynamics*, J Comp Phys, 117, 1-19 (1995)

30. Perkin, S. & Albrecht, T. & Klein, J. (2010) *Layering and Shear Properties of an Ionic Liquid, 1-ethyl-3-methylimidazolium Ethylsulfate, Confined to Nano-films between Mica Surfaces* Phys. Chem. Chem. Phys. 12, 1243– 1247

31. Clausius, R. (1870) *On a mechanical theorem applicable to heat*, The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, 40:265, 122-127