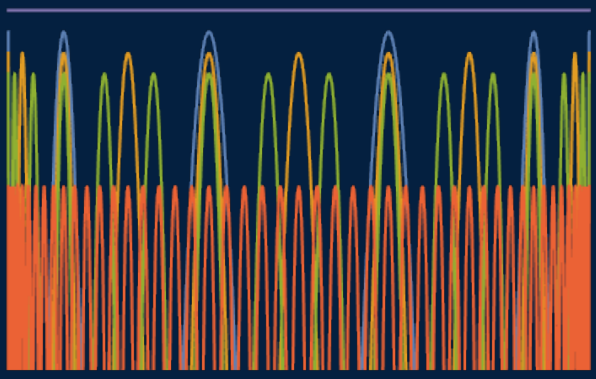
Improved Numerical Methods for Elliptic Problems in Astrophysics



Jose Adsuara Fuster

Doctorat en Física
Departament d'Astronomia i Astrofísica
Universitat de València

TESI DOCTORAL
2017

VNIVERSITAT DE VALÈNCIA

Improved Numerical Methods for Elliptic Problems in Astrophysics

Jose Adsuara Fuster

Doctorat en Física
Departament d'Astronomia i Astrofísica
Universitat de València

TESI DOCTORAL

Directors:
Miguel Ángel Aloy Torás
Pablo Cerdá Durán

Maig 2017

Miguel Ángel Aloy Torás
Professor Titular del *Departament d'Astronomia i Astrofísica* de la *Universitat de València*

i

Pablo Cerdá Durán
Investigador *Ramón y Cajal* del *Departament d'Astronomia i Astrofísica* de la *Universitat de València*

CERTIFIQUEN:

Que la present memòria, titulada *Improved Numerical Methods for Elliptic Problems in Astrophysics*, ha estat realitzada baix llur direcció en el Departament d'Astronomia i Astrofísica de la Universitat de València per Jose Adsuara Fuster i constituïx la seua Tesi Doctoral per optar al grau de Doctor en Física. I perquè conste, signen el present certificat en València a 30 de maig de 2017.

Signat: Miguel Ángel Aloy Torás

Signat: Pablo Cerdá Durán

A Teresa, constructora de somnis.
A Jose, Teresa i Mireia; contagiant alegria.

Agradecimientos

En primer lugar, dar las gracias a mis directores. Esta tesis es lo que es, en muchos aspectos, gracias a ellos.

Muchísimas gracias a mi director de tesis el profesor Miguel Ángel Aloy por muchas razones. Por permitirme vivir [1] durante estos últimos años de uno de mis sueños que a estas alturas de la vida me parecía ya imposible: la investigación. Por guiarme. Por darme también esa libertad necesaria a la hora de explorar nueva ideas. Por confiar en mi. Por ser un trabajador incansable, de esos pocos que se ponen a tu lado a trabajar siempre que hace falta. Por no rendirse nunca.

Gracias también a mi director el doctor Pablo Cerdá Durán. Por estar siempre disponible a la hora de atender cualquier duda, del tipo que sea, y por su acertada orientación. Por dejar a un lado lo que esta haciendo y escucharme. Por adaptar códigos a lo que haga falta.

Mi agradecimiento a la doctora Isabel Cordero Carrión. Por ese espíritu matemático en este mundo de físicos, velando y cuidando los mas mínimos detalles de cada una de las cosas que hace. Por su inestimable colaboración. Por su apoyo.

Mil gracias también al catedrático José María Ibáñez, cuyo voto de confianza inicial me permitió entrar en este gran grupo de investigación en el que ahora estoy. Por su sentimiento paternal.

Y a los profesores Juan Antonio Morales, Joan Josep Ferrando, Diego Sáez y Antonio Marquina, a los tuve mientras estudiaba matemáticas. Por ponerme en contacto con el Departament d'Astronomia i Astrofísica. Al resto del departamento por su ayuda siempre que ha hecho falta.

Al doctor Petar Mimica, por ese comentario sobre el artículo original gracias al cual arrancó todo este trabajo.

A Feli y Manel, por su inestimable ayuda administrativa.

[1] Agradecer el apoyo del European Research Council (Starting Independent Research Grant CAMAP-259276); y parcialmente de las ayudas AYA2013-40979-P, AYA2015-66899-C2-1-P, PROMETEO-II-2014-069, SAF2013-49284-EXP y Consolider SyeC net TIN2014-52608-REDC

Agradecer los recursos computacionales, la experiencia técnica y la asistencia proporcionada por el Servei d'Informàtica de la Universitat de València. También al Centre de Supercomputació de Barcelona.

A todos los que hemos convivido a lo largo de este camino. Por esas comidas y ratos siempre tan divertidos. Por hacerme sentir mas joven. A los que están: Alex, Jesús, Nico, Nico, Tomek, Isa, Adri. Y a los que estuvieron: Vas, Rebecca. Y como no, a nuestro entrañable Juan Antonio Chistoso.

Por nuestro café a media mañana; charlas en el despacho: Carlos, Pablo, Petar, Martin, Miguel Ángel, Jens; Carmen, Sergio.

A la familia, por estar siempre ahí, dispuesta a ayudar. A mis amigos.

Y en especial a mi familia. A mis hijos, por los que todo esto vale la pena. Y a mi mujer, sin la cual nada de esto habría sido posible. Por permitirme soñar. Por creer en mi. Por quererme.

Abstract

Elliptic partial differential equations appear in a wide variety of areas of mathematics, physics and engineering. They are of particular interest in Astrophysics and appear, e.g., when computing the gravitational potential, in the solution of the Grad-Shafranov equation for force-free magnetospheres, to impose divergence free constraints in the numerical integration of MHD equations or when solving the constraint equations in General Relativity. Typically, elliptic equations must be solved numerically, which sets an ever growing demand for efficient and highly parallel algorithms to tackle their computational solution.

The Scheduled Relaxation Jacobi is a promising class of methods, atypical for combining simplicity and efficiency, that has been recently introduced for solving linear Poisson-like elliptic equations. It is an extension of the classical Jacobi iterative method to solve linear systems of equations ($Au = b$). It also inherits its robustness. Its methodology relies on computing the appropriate parameters of a multilevel approach with the goal of minimizing the number of iterations needed to cut down the residuals below specified tolerances.

The efficiency in the reduction of the residual increases with the number of levels employed in the algorithm. Applying the original methodology to compute the algorithm parameters with more than 5 levels notably hinders obtaining optimal schemes, as the mixed (non-linear) algebraic-differential system of equations from which they result become notably stiff.

On one hand, we present a new methodology for obtaining the parameters of Scheduled Relaxation Jacobi schemes that overcomes the limitations of the original algorithm and provides parameters for these schemes with up to 15 levels and resolutions of up to 2^{15} points per dimension, allowing for acceleration factors larger than several hundreds with respect to the Jacobi method for typical resolutions and, in some high resolution cases, close to 1000. Most of the success in finding these optimal schemes with more than 10 levels is based on an analytic reduction of the complexity of the previously mentioned system of equations.

Furthermore, we extend the original algorithm to apply it to certain systems of non-linear elliptic equations.

On the other hand, in a typical Scheduled Relaxation Jacobi scheme, the former set of factors is employed in cycles of M consecutive iterations until a prescribed tolerance is reached. We present the analytic form for the optimal set of relaxation factors for the case in which all of them are strictly different, and find that the resulting algorithm is equivalent to a non-stationary generalized Richardson's method where the matrix of the system of equations is preconditioned multiplying it by $D = \mathrm{diag(A)}$. Our method to estimate the weights has the advantage that the explicit computation of the maximum and minimum eigenvalues of the matrix A (or the corresponding iteration matrix of the underlying weighted Jacobi scheme) is replaced by the (much easier) calculation of the maximum and minimum frequencies derived from a von Neumann analysis of the continuous elliptic operator. This set of weights is also the optimal one for the general problem, resulting in the fastest convergence of all possible Scheduled Relaxation Jacobi schemes for a given grid structure. We refer to it as the Chebyshev-Jacobi method. The amplification factor of the method can be found analytically and allows for the exact estimation of the number of iterations needed to achieve a desired tolerance. We also show that with the set of weights computed for the optimal SRJ scheme for a fixed cycle size it is possible to estimate numerically the optimal value of the parameter ω in the successive overrelaxation method in some cases.

We demonstrate with practical examples, with application in Astrophysics, that our method also works very well for Poisson-like problems in which a high-order discretization of the Laplacian operator is employed (e.g., a 9− or 17−points discretization). This is of interest since the former discretizations do not yield consistently ordered A matrices and, hence, the theory of Young cannot be used to predict the optimal value of the SOR parameter. Furthermore, the optimal SRJ schemes deduced here are advantageous over existing SOR implementations for high-order discretizations of the Laplacian operator in as much as they do not need to resort to multi-coloring schemes for their parallel implementation.

We present the implementation of the Chebyshev-Jacobi method using a purely MPI implementation, an openMP / MPI hybrid implementation over both shared memory machines and distributed memory machines. They show ideal speed up. We also show how to reach a remarkable speed up when solving elliptic partial differential equations with finite differences thanks to the joint use

of the Chebyshev-Jacobi method with high order discretizations and its parallel implementation over GPUs.

Finally, we have tried to apply our methods beyond the realm of Astrophysics with limited sucess though. Specially, we have addressed the problem of finding normal modes of human eyeballs. This problem is ready for being solved with an improved variant of the methodology here presented. The improvement consists on extending the calculation of the optimal set of parameters to non positive-definite matrices. Our ideas on how to proceed in this field are sketched in the outlook of this thesis.

Publications

The following peer-reviewed publications and proceedings reports have been published during the duration of the doctoral fellowship:

- **JE Adsuara**, I Cordero-Carrión, P Cerdá-Durán, MA Aloy,
  Improvements in the Scheduled Relaxation Jacobi method,
  JMC. Díaz Moreno, C García Vázquez, J Medina Moreno, F Ortegón Gallego, F Pérez Martínez, MV Redondo Neble, JR Rodríguez Galván (Eds.),
  Proceedings of the XXIV Congress on Differential Equations and Applications/XIV Congress on Applied Mathematics, Cádiz, June 8-12, 2015,
  Servicio de Publicaciones de la Universidad de Cádiz (2015)

- *Scheduled relaxation Jacobi method: improvements and applications*,
  JE Adsuara, I Cordero-Carrión, P Cerdá-Durán, MA Aloy,
  Journal of Computational Physics 321, 369-413 (2016)

- *Evaluation of the repeatability of a swept-source ocular biometer for measuring ocular biometric parameters*,
  Teresa Ferrer-Blasco, Alberto Domínguez-Vicent, José J Esteve-Taboada, Miguel A Aloy, **Jose E Adsuara**, Robert Montés-Micó,
  Graefe's Archive for Clinical and Experimental Ophthalmology, 1-7 (2016)

- *On the equivalence between the Scheduled Relaxation Jacobi method and Richardson's non-stationary method*,
  JE Adsuara, I Cordero-Carrión, P Cerdá-Durán, V Mewes, MA Aloy,
  Journal of Computational Physics 332, 446-460 (2017)

- *Estimation of the Mechanical Properties of the Eye through the Study of its Vibrational Modes*,
  MA Aloy, **JE Adsuara**, P Cerdá-Durán, M Obergaulinger, JJ Esteve-Taboada, T Ferrer-Blasco, R Montés-Micó,
  Submitted to PLOS ONE

- *Human eye normal vibrational modes for different axial lengths using linear elasticity*,
  JE Adsuara, P Cerdá-Durán, M Obergaulinger, P Mimica, JJ Esteve-Taboada, T Ferrer-Blasco, R Montés-Micó, MA Aloy,
  Submitted to Graefe's Archive for Clinical and Experimental Ophthalmology

- *Speeding up a few orders of magnitude the Jacobi method: high order Chebyshev-Jacobi over GPUs*,
  JE Adsuara, MA Aloy, , P Cerdá-Durán, I Cordero-Carrión
  Submitted to Journal of Computational Physics (2017)

Contents

Nomenclature

Acronyms / Abbreviations

AMR	Adaptative Mesh Refinement
BH	Black Hole
BVP	Boundary Value Problem
CFC	Conformal Flatness Condition
CJM	Chebyshev-Jacobi Method
CO	Consistently Ordered
CPU	Central Process Unit
ePDE	Elliptic Partial Differencial Equations
GPU	Graphics Processing Unit
GRB	Gamma-Ray Burst
GS	Grad-Shafranov
ICM	Intra-Cluster Medium
IVP	Initial Value Problem
lGRB	Long Gamma-Ray Bursts
MHD	Magnetohydrodynamics
MIMD	Multiple Instruction Multiple Data
MISD	Multiple Instruction Single Data
MPI	Message Passing Interface

NCO	Non-Consistently Ordered
NS	Neutron Star
PDE	Partial Differential Equations
RAM	Random Access Memory
RHD	Relativistic HydroDynamics
SGI	Silicon Graphics
sGRB	Short Gamma-Ray Bursts
SIMD	Single Instruction Multiple Data
SISD	Single Instruction Single Data
SOR	Successive Overrelaxtion
SRJ	Scheduled Relaxation Jacobi

Part I

Introduction

Chapter 1

Elliptic problems in astrophysics

1.1 Introduction

Reality is change. Change over time, and change with respect to space. The concept of derivative arises in this context, as the rate of change of one variable with respect to another. That is why partial differential equations (PDEs) appear everywhere in physics.

From a practical point of view, we may wonder why do we need to solve PDEs. As illustrated in the excellent monograph of [Otway 2015], in general, a solution of a PDE may reproduce three different types of behaviors: it can propagate as a wave packet, it can diffuse as heat, or it can oscillate going nowhere at all. Many physical phenomena can be cast combinations of these three possibilities. Thus, solving PDEs may well serve the purpose of modeling mathematically the local nature of physical phenomena. The different types of equations, namely, hyperbolic, parabolic, or elliptic, are ultimately correlated to which of the three aforementioned behaviors their solution represent, i.e., to wether the physical phenomenon propagates, diffuses, or oscillates. A rigorous exposition of this classification is given in [Courant and Hilbert 1962].

PDEs are the appropriate mathematical language for modeling many phenomena. Among the many famous and interesting PDEs, just in the case of linear single partial differential equations, there are plenty of examples (see e.g. Evans 2010, Salsa 2016): Laplace's equation, Helmholtz's or eigenvalue equation, linear transport equation, Liouville's equation, heat or diffusion equation,

Schrödinger's equation, Kolmogorov's equation, Fokker-Planck equation, wave equation, telegraph equation, general wave equation, Airy's equation, Beam equation, etc. And along with this, we also have nonlinear PDEs or systems of both linear and nonlinear PDEs. We refer the reader to [Zwillinger 1997] for an extensive listing and deeper study.

The research focuses on particular types of PDEs, since there is no general theory about the solvability of any of them. This situation naturally arises from the many different contexts where PDEs are found. Such variety induces that the properties of each specific type of PDE and of their respective solutions be extremely diverse. We focus this thesis on the efficient numerical solution of a particular case of PDEs, elliptic PDEs (ePDEs), within a specific area of knowledge, Astrophysics. In this chapter we review some basic mathematical concepts related with ePDEs and provide some of specific examples in Astrophysics we aim to address in this thesis or in future applications of the methods here developed. In the following chapter we will focus on the different solution strategies to solve numerically ePDEs.

1.2 Elliptic partial differential equations in astrophysics

We are thus interested in ePDEs. The name elliptic is inherited from the analogy, in terms of equations and their classification, between conic sections and *quasilineal second-order* PDEs [Hoffman 1992], which are respectively

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0, \tag{1.1}$$

$$Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G, \tag{1.2}$$

where A, B, C, D, E, F and G are all functions of the coordinates. These equations are classified according to the sign of the *discriminant*, $B^2 - 4AC$. An equation of this type is elliptic when the discriminant is strictly negative. This classification can be extended to the n-dimensional case, where we can define the operator

$$L := \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\frac{\partial^2}{\partial x_i \partial x_j} + \sum_{i=1}^{n} b_i \frac{\partial}{\partial x_i} + c, \tag{1.3}$$

where a_{ij}, $i, j = 1, \ldots, n$, are the elements (real functions of the coordinates) of a matrix $\mathcal{A} = (a_{ij})$, and b_i and c are also functions. The operator L is said to be elliptic if the eigenvalues of $\mathcal{A}$ are all positive or all negative.

Among the most relevant equations of elliptic type are *Laplace's* equation

$$\Delta u = 0, \tag{1.4}$$

and *Poisson's* equation

$$\Delta u = f, \tag{1.5}$$

where $u = u(x)$ with $x \in U \subset \mathbb{R}^n$ and $f : U \to \mathbb{R}$. Δ is the *Laplace operator*, which is a functional, i.e., a function which transforms functions in functions. Its definition is the divergence of the gradient:

$$\Delta u := \nabla \cdot (\nabla u), \tag{1.6}$$

which in orthonormal coordinates simply reads as:

$$\Delta u := \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2} u. \tag{1.7}$$

Laplace's equation describes, e.g., steady, irrotational flows, electrostatic potential in the absence of charge, equilibrium temperature distribution in a medium. It is almost ubiquitous in mathematical physics (see, e.g., [Feynman, Leighton, and Sands 1966], Chap. 12)

A function u satisfying Eq. (1.4) is called *harmonic* function, while a function satisfying Eq. (1.5) is called *superharmonic* (*subharmonic*) if $f \leq 0$ ($f \geq 0$). Solutions of the general second-order ePDE $Lu = 0$ are similar in many regards to harmonic functions [Salsa 2016].

The differential equations presented so far feature an infinite number of solutions. With the aim of obtaining a unique solution we must impose suitable conditions on the boundary ∂U of U. For instance, the inviscid flow past a sphere is determined by the boundary conditions on the sphere ($\mathbf{u} \cdot \mathbf{n} = 0$; $\mathbf{n}$ being the unit vector normal to the sphere boundary), as well as at infinity (e.g., $\mathbf{u}$ =const.).

Provided its physical origin, ePDEs result from boundary value problems (BVPs), in which at every point inside the domain of interest the solution depends of the boundary of the domain [Bodenheimer et al. 2006]. We may find three basic types of boundary conditions (or combinations thereof) depending on what is prescribed on the boundary ∂U:

1. *Dirichlet*: u takes a prescribed value.
2. *Neumann*: the normal derivative $\partial u/\partial n = \mathbf{n} \cdot \nabla u$ is prescribed.
3. *Robin*: a combination of u and its normal derivative (e.g., $\alpha u + \beta \partial u/\partial n$ is specified).

The aforementioned classification of a PDE is intimately related to the *characteristics* of the PDE [Hoffman 1992]. Characteristics are $(n-1)$-dimensional hypersurfaces in n-dimensional hyperspace (e.g. curves in the plane, in the $n = 2$ case). They are the preferred paths of information propagation. In the case of complex characteristics, there are no paths, along which physical information propagates. In this way, each point of the domain influences and is influenced by the rest of the points of the domain. Therefore, ePDEs typically arise when we face the solution of *equilibrium problems*, i.e., *steady-state problems* in *closed domains*. These are problems in which the time evolution of the system is either neglected, irrelevant or simply that certain differential relations must hold among the variables of the problem (e.g., the constraint equations in General Relativity). Neglecting the time evolution can be a very restrictive assumption on the class of problems under consideration, since many interesting realistic phenomena occurring in nature are typically dynamic and also involve wave propagation problems. However, the numerical fulfillment of constraint equations is a mathematical requisite to set a well posed problem. In other words, it is not a limitation of the modeling but rather a physically motivated relationship.

In account of the mean-value theorem for the Laplace's equation (see, e.g., Evans 2010, § 2.2.2), the Laplacian measures the variation of the value of a function in a point with respect to the average value in its neighborhood; the greater the difference, the greater the value of the Laplacian [Styer 2015]. So, as a generalization of the one-dimensional second derivatives, the Laplacian measures the average convexity (harmonic functions, for example, are functions of null mean convexity) [Vázquez 2014].

Although Laplace and Poisson equations possess analytic solutions in a limited number of simple cases, we usually need a numerical solution when elliptic problems are considered. The problems that we are going to consider belong to the realm of Astrophysics. We are interested in problems such as gamma-ray bursts (GRBs), core collapse supernovae, the merger of compact objects (neutron stars or black holes), magnetars, magnetized accretion discs around spinning black holes, galaxies clusters, galaxy formation and evolution, etc.

In physics we need to design experiments that contrast the new theories or models proposed. However, many of the physical conditions present in astrophysical scenarios are not reproducible in any laboratory. Simulation allow us to build models of a system and to do virtual experiments [Zingale 2017].

We present next a list of elliptic problems that people have encountered in Astrophysics. The list is not intended to be exhaustive, but it is representative

of the work developed in the group, within which the research for this thesis has been developed. We start with the areas of astrophysics closer to our research field and then we gradually move to other areas. We also review the numerical methods that have been used traditionally in each case.

1.2.1 Gravitational field

Stars are one of the main objects of study of Astrophysics. The stars can be modeled as self-gravitating fluid spheres. The equations of *hydrodynamics* describe how the fluid evolves in time. If we consider viscosity, radiation transfer or magnetic fields, we need the appropriate variants of the former equations: *Navier-Stokes* equations, *radiation hydrodynamics* equations or *magnetohydrodynamics* (MHD) equations, respectively [Bodenheimer et al. 2006].

Hydrodynamics equations in a self-gravitating object are coupled, in Newtonian gravity, to the Poisson equation for the gravitational field. During the time evolution of the fluid the gravitational field has to be recalculated at each time-step, which itself affects the dynamics of the fluid. Therefore, we have a gravity-fluid tandem that we have to solve numerically. Computational time in multidimensional simulations can very easily be dominated by the numerical solution of the Poisson equation (specially in 3D). It is therefore critical to use efficient algorithms for the elliptic part of the problem.

Ultimately gravity is described by the theory of general relativity. However, if the object we are modelling is not very compact, it is sufficient to consider the Newtonian limit, which results in the Poisson equation. As the compactness of the studied object increases, the relativistic effects begin to be noticed. In many scenarios, small post-Newtonian corrections are sufficient (see, e.g., Misner, Thorne, and Wheeler 1973). However, for extremely compact objects (neutron stars or black holes) we need to solve Einstein's field equations. Still in this case it is necessary to solve systems of ePDEs. The precise set of ePDEs to solve depends on the chosen formulation of the Einstein's equations. For instance, in the Baumgarte-Shapiro-Shibata-Nakamura (BSSN) formulation [Baumgarte and Shapiro 1999, Shibata and Nakamura 1995], it is necessary to enforce the Hamiltonian and momentum constraints (at least initially to provide initial date). In fully constrained formulations [Bonazzola et al. 2004], this is also the case, though the system of ePDEs may be modified.

In general, stars are an example in which gravity can be regarded as Newtonian. An example of numerical simulations involving a star in hydrodynamical equilibrium, and that will be described in more depth in Chap. 6, is the prop-

agation of GRB-jets through the envelope of a massive star. In this case, it is necessary to consider the gravitational potential in order to preserve the hydrostatic equilibrium of the envelope as the jet propagates. In this scenario it is sufficient to solve a Poisson equation whose source term is the distribution of mass in the space. Evidently, in this case we seek a numerical solution of the Poisson equation, since we do not know an analytical expression of this distribution, which is the result of the hydrodynamic simulation.

In the case of core-collapse supernova explosions, we deal with more compact objects than in the previous scenario, and relativistic corrections are needed. Supernova codes typically solve the gravitational potential in the Newtonian limit or using a pseudo-Newtonian potential that contains some relativistic effects. In any case, the resulting ePDE is the Poisson equation. Some examples of codes using this approach are PROMETHEUS (a tridimensional hydrodynamics code; Fryxell, Müller, and Arnett 1989, Keil 1997, Plewa and Müller 1999), VERTEX (which is based on PROMETHEUS hydrodynamics part and also includes Boltzmann neutrino transport Rampp and Janka 2002), AMRA [Plewa and Müller 2001] P-HOTB [Janka and Mueller 1996, Kifonidis et al. 2003] and AENUS [Obergaulinger et al. 2006, Obergaulinger, Janka, and Aloy 2014]. All of these codes use the method proposed in [Müller and Steinmetz 1995] for solving the Poisson equation. Employing the theory of Green functions, the solution of the equation is written in integral form. After that, the integrand is expanded into spherical harmonics. Finally, they truncate this series and only add a few of the first terms.

Ultimately, the best approach is to consider General Relativity. To be able to deal with curved four-dimensional space-times, it is most popular to use the $3+1$ formalism, which slices the space-time in a family of non-intersecting 3-dimensional space-like hypersurfaces, separated from each other by an elapsed time. In this formalism, the Einstein equations are written as a system of evolution and constraint equations. For the case of moderately high gravity, such as in core collapse supernovae, it is possible to use the conformal flatness condition (CFC) approximation [Wilson, Mathews, and Marronetti 1996]. Numerically, the major advantage of this approximation is that the $3+1$ metric equations are reduced to five coupled nonlinear ePDEs [Dimmelmeier, Font, and Müller 2001], which, due to its relative simplicity, makes their implementation more stable than in the standard formulations of the field equations commonly used in numerical relativity [Cerdá-Durán et al. 2005]. This approach has been followed in the CoCoNuT code, a general relativistic hydrodynamics code with dynamical space-time evolution [Dimmelmeier, Font, and Müller 2002a, Dimmelmeier, Font, and

Müller 2002b]. Several options have been followed historically to solve this elliptic system, being the spectral methods, using the library LORENE [*LORENE - Langage Objet pour la RElativité NumériquE*], the current implementation. The elliptic system of the CFC approach has been improved by reformulating the system of equations to solve [XCFC, Cordero-Carrión et al. 2009]. The improved system contains ten decoupled Poisson equations, two of which are nonlinear, which have to be resolved in a hierarchical way [Cordero-Carrión et al. 2009]. This hierarchy of elliptic equations improves the uniqueness properties of the solution and allows the numerical simulation of the formation of black holes, which was not possible with the previous system.

One of the most promising sources of gravitational waves is the coalescence of inspiraling compact binaries. Here, as above, we also need to work on general relativity. Although the most popular approach is to use hyperbolic formulations of Einstein's equations (e.g. BSSN) in which no elliptic equations shall be solved throughout the evolution (but the elliptic constraints must be solved to provide the initial data for the Cauchy problem), there have been attempts to use formulations based in elliptic equations, such as the CFC approximation. Example applications to binary neutron star mergers can be found in [Shibata, Baumgarte, and Shapiro 1998, Bauswein, Oechslin, and Janka 2010]. In the second case multigrid methods were used to solve the elliptic equations of the CFC method.

The equations of hydrodynamics can also be used for cosmological simulations, in which an elliptic equation for the gravitational potential is needed. The source term depends on the scale factor of a flat background, the Hubble constant and a density contrast between the mass density and the background mass density. This is necessary for the two existing approaches for solving numerically them: grid based or meshless codes. In [Quilis, Ibanez, and Saez 1996, Quilis 2004], for example, the equations governing the evolution of cosmological inhomogeneities form a hyperbolic system of conservation laws. Adaptative mesh refinement (AMR) is used to follow the gas dynamics while dark matter is computed by means of an n-body computation. In the coarsest mesh, which is cubic, [Quilis 2004] the gravitational field equations employing fast Fourier transforms. For each of the patches that is generated by the refinement procedure, the finite-difference discretization of the Poisson-like equations result in a linear system, which is numerically solved by means of successive overrelaxation (SOR) method. The reasons to employ different solution methods on the base level of the AMR hierarchy and on any other refined level is because in this case the SOR method is more convenient since, not only the shape of the domains is no longer regular,

but also because it is possible to use the previous values of the potential of the unrefined mesh as initial guess for the iterative procedure. The latter is very convenient to reduce the number of iterations until the desired tolerance, since the change of the solution on consecutive timesteps is small.

1.2.2 Initial data

A number of the numerical simulations involving fluid dynamics can be formulated as initial value problems (IVPs), whose governing equations are not elliptic but hyperbolic. Nevertheless, IVP need to begin their time marching strategy from initial data which are properly specified. Often the generation of initial data requires the solution of elliptic problems. This is the case when, e.g., we need to begin from equilibrium configurations of isolated stars, eventually including rotation and magnetic fields, in Newtonian gravity or full general relativity. Also the generation of the stationary spacetime in numerical relativity simulations requires the solution of involved ePDEs. Two complementary reviews on the topic are [Stergioulas 1998] and [Cook 2000b].

To more specifically exemplify the previously mentioned problems, let us first consider the case in which we aim to begin our models from a stellar configuration in hydrostatic equilibrium. In the relativistic case, if matter is also to be in equilibrium, then the four-velocity of the matter must be a linear combination of the two Killing vectors of its stationary spacetime. Having specified an equation of state, the structure of the star is computed by solving four components of Einstein's gravitational field equations. The first attempt to obtain an equilibrium configuration of relativistic rigidly rotating fluid is due to [Bonazzola and Schneider 1974]. Since then, several schemes have been put into practice. The method proposed by [Komatsu, Eriguchi, and Hachisu 1989a, Komatsu, Eriguchi, and Hachisu 1989b] consists in three equations of elliptic type that are solved using Green functions. In [Bonazzola et al. 1993, Bonazzola, Gourgoulhon, and Marck 1998] the field equations are derived in the $3+1$ formulation and all four equations describing the gravitational field are of elliptic type. The equations are solved using spectral methods, expanding all the variables in terms of trigonometric functions, in both the angular and radial directions. Then a fast Fourier transform is used to obtain the spectral coefficients and, hence, the solution.

Elliptic equations can also be used in the construction of homogeneous stellar models in Newtonian hydrodynamics. For instance, [Schöbel and Ansorg 2003] make use of pseudo-spectral methods for finding the rotational structure of stars

in hydrostatic equilibrium resulting from the balance between gravity, pressure gradients and centrifugal force [Roxburgh 2004, Roxburgh 2006].

Other equilibrium figures, different from the spheroidal ones, can be found in different astrophysical scenarios. In particular, Dyson [Dyson 1892, Dyson 1893] is able to give a fourth-order expansion of uniformly rotating, homogeneous and axisymmetric rings in Newtonian gravity. These Dyson rings are extended to Einstenian gravity in [Ansorg, Kleinwächter, and Meinel 2003], and are computed numerically with spectral methods. They might be relevant in two different astrophysical situations: as a result of stellar core-collapse in the case of high angular momentum; or they could be present in central regions of galaxies.

In the case of coalescing binary compact objects, initial data is also needed, which is the result of solving the constraint equations in General relativity, which are of elliptic type. Recent developments in the field can be found at [Dietrich et al. 2017, Ruchlin et al. 2017] and references therein. According to [Dietrich et al. 2017], who focus on neutron star, there are a number of well-developed codes for computing binary neutron star initial data, most notably the spectral code LORENE [*LORENE - Langage Objet pour la RElativité NumériquE*], the Princeton group's multigrid solver [East, Ramazanoglu, and Pretorius 2012], BAM's multigrid solver [Moldenhauer et al. 2014], the COCAL code [Tsokaros, Uryu, and Rezzolla 2015], the SpEC code's spectral solver [Foucart et al. 2008], and spectral code SGRID [Buonanno et al. 2011]. In [Ruchlin et al. 2017], devoted to black holes, and in which they compute initial data of binary black holes with high spins and boosts, allows us to reach the origin of these calculations in [Pfeiffer, Cook, and Teukolsky 2002, Pfeiffer et al. 2003] and [Gourgoulhon, Grandclément, and Bonazzola 2002, Grandclément, Gourgoulhon, and Bonazzola 2002]. They are applying spectral methods.

In the case of black hole initial data, we need to consider Cauchy initial data that represent one or more black holes in an asymptotically flat spacetime. Stationary spacetimes possesses both temporal and angular Killing vectors[1]. The majority of these will be either vacuum solutions or solutions of the Einstein-Maxwell equations. With no matter to support the gravitational field, these spacetimes usually have a non-trivial topology [Cook 2000b].

1.2.3 Magnetostatic configurations

So far only gravity has appeared as a fundamental force in astrophysical simulations. However, there are situations where electromagnetism also plays a

[1]Note that binary black hole initial data spacetimes are usually quasi-stationary, with an approximate helicoidal Killing vector field.

determining role. In fact, we can have magnetic fields in any of the previous areas like GRBs, supernovae, stellar core collapse, etc. As we have already indicated above, to work with magnetized plasmas we use the MHD equations. In MHD applications two basic problems can be cast in therms of ePDEs: the enforcement of the divergence free condition for the magnetic field and the generation of magnetostatic configurations in astrophysical systems. The former problems will be addressed in Sec. 1.2.4. In this section we specifically consider the special class of problems consisting in generating equilibrium magnetic field configurations in the magnetosphere of compact objects.

When we study magnetospheres of both neutron stars and black holes, under spherical symmetry conditions, the equation governing their structure is the general relativistic Grad-Shafranov (GS) equation [Lüst and Schlüter 1954, Chandrasekhar 1956, Chandrasekhar and Prendergast 1956, Shafranov 1958a, Grad and Rubin 1958], which needs to be solved numerically. The GS equation is an elliptic equation. As an example, in the case of neutron stars, we have [Torres-Forné et al. 2016], which studies pulsars and magnetars; and in the case of black holes we have [Uzdensky 2004] or [Contopoulos, Kazanas, and Papadopoulos 2013], which studies the force-free magnetosphere of accretion disk and rotating black hole. In the first case they use the cyclic reduction algorithm while in the second case they use the SOR algorithm. More recently, Akgün et al. 2016 have solved the GS equation to obtain the equilibria configurations of force-free twisted magnetospheres around neutron stars. These authors employ a direct method (the Thomas algorithm; Thomas 1949) to solve the block tridiagonal system resulting from the finite-difference discretization of the GS equation.

1.2.4 Projection methods

The numerical solution of physical problems in computational physics often involves solving elliptic-hyperbolic PDEs when (physical) constraints need to be enforced during the evolution of the system. A popular example is the so-called *projection scheme* used to enforce the zero divergence of the magnetic field constraint in ideal MHD [Brackbill and Barnes 1980]. The accurate solution of the equations of ideal MHD requires that magnetic fields be solenoidal, i.e., divergence free. Indeed, Maxwell equations guarantee that divergence free initial magnetic field data remain divergence free throughout the evolution. However, even if initially the magnetic field configuration is solenoidal, the numerical time evolution of the solution will introduce magnetic monopoles in the computational domain unless specific numerical techniques are employed [Brackbill and Barnes

1980, Dai and Woodward 1998]. Various numerical techniques have been devised to ensure the computed magnetic field is maintain divergence-free throughout the temporal evolution [Tóth 2000, Feng and Zhang 2016]. Projection schemes obtain the same order of accuracy as the underlying base schemes [Tóth 2000, Feng and Zhang 2016], which means that high-order finite differences are desirable when solving ePDEs associated with projection schemes in combination with high-order methods in resolving the hyperbolic evolution equations. We are interested here in those techniques, which convert the problem of preserving the divergence of the magnetic field into an elliptic problem. The underlying idea is to project the numerical solution provided by the base numerical scheme onto the subspace of zero divergence solutions. For that one makes use of a linear operator, and the projected solution is used in the next time step (instead of the originally computed by the base scheme).

The Helmholtz's theorem of vector calculus states that any sufficiently smooth, rapidly decaying vector field $\boldsymbol{X}$ in three dimensions can be split as the sum of a solenoidal (divergence-free) vector field $\boldsymbol{X}_{\rm sol}$ and an irrotational (curl-free) vector field $\boldsymbol{X}_{\rm irr}$. So, the vector field can be written as $\boldsymbol{X} = \nabla \times \boldsymbol{A} + \nabla\Phi$ because a solenoidal vector field may be derived from a vector potential $\boldsymbol{A}$ and an irrotational vector field from a scalar potential Φ.

If we take the divergence in both sides, we have $\nabla\boldsymbol{X} = \nabla\cdot(\nabla\times\boldsymbol{A}) + \nabla\cdot(\nabla\Phi)$. Taking into account that the divergence of the rotational is zero and that the divergence of the gradient is the Laplacian, we have the Poisson equation $\Delta\Phi = \nabla\boldsymbol{X}$. As the vector field $\boldsymbol{X}$ is known, the above equation can be solved for the scalar function Φ and the divergence-free part of $\boldsymbol{X}$ can be extracted simply doing $\boldsymbol{X}_{\rm sol} = \boldsymbol{X} - \nabla\Phi$.

We outline that projection methods have also been used in incompressible fluid dynamics [Chorin 1969, Bell, Colella, and Glaz 1989, Almgren, Bell, and Szymczak 1996], where the $\nabla\cdot\mathbf{v} = 0$ constraint ($\mathbf{v}$ is the velocity field) must be enforced. Examples of the application of projection methods to Astrophysics can be found in problems where incompressible turbulence plays a paramount role. For instance, many galaxy cluster properties are influenced by significant gas motions characterizing their intra-cluster medium (ICM). These motions may lead to strong ICM turbulent motions. The ICM turbulence, defined as the rotational of the gas velocity field, can be decomposed as explained above. Both components can contribute to the acceleration of relativistic particles in the ICM. However, whereas the compressive component generates shock waves, which play a crucial role in the ICM thermalization, the solenoidal component

mainly contributes to the amplification of ICM magnetic fields [Vazza et al. 2017].

A final example of the usage of projection methods arises in the numerical integration of the Einstein equations, where the construction of initial data involves solving the so-called constraint equations, a set of elliptic PDEs (see e.g. [Cook 2000a] for a detailed review). A popular way of evolving the resulting initial data is via the hyperbolic BSSN [Baumgarte and Shapiro 1999, Shibata and Nakamura 1995] scheme.

1.2.5 Other examples

In the previous sections we have oulined a few astrophysical problems that we are interested in applying the new methods of solution of ePDEs we develop in this thesis. To complete the landscape of potential applications of our new methods, we foresee that in those cases in which boundary conditions need to be specify in complicated hypersurfaces, our methods can be useful (see, e.g., Sec. 5.2). We mention a few of such cases:

- The governing equations for ideal MHD flow in a nozzle [Chu 1962] altern between elliptic and hyperbolic type three times: at the sonic point, where the fluid velocity equals the sound speed $c_{\rm s}$, at the Alfven point, where the fluid velocity equals the Alfven speed $c_{\rm a}$, and when the flow crosses the fast magnetosonic point, being the fast magnetosonic speed $c_{\rm s}c_{\rm a}/\sqrt{(c_{\rm s}^2+c_{\rm a}^2)}$. The application of this scenario in Astrophysics has its prototype in the twin-exhaust model, originally due to Blandford and Rees 1974 and applied to relativistic MHD jet in, e.g., Vlahakis and Königl 2004.

- The equations for small-amplitude electromagnetic wave propagation in zero- temperature plasma are elliptic at certain wave frequencies [e.g. Grossmann and Weitzner 1984, Otway 2008].

- Within the Hartle–Hawking model [Hartle and Hawking 1983], the Laplace–Beltrami equation on a region devoided of matter of space-time would have been of elliptic type in the early universe.

- In a flat space-time the wave equation, is elliptic for certain values of the radial coordinate, in a reference frame rotating with constant angular velocity [Stewart 2001].

- In the context of binary black hole spacetimes with a helical Killing vector field it is possible to formulate a helically reduced wave equation on a torus which is elliptic for certain values of the radial coordinate [Klein 2004].

Chapter 2

Introduction to iterative methods

2.1 Introduction

In this chapter, we make a basic overview of the strategies for the numerical solution of elliptic systems of partial differential equations (PDEs). Our goal is not to make an exhaustive review of the different solution methodologies, but rather to place in its correct context the new methods we have developed. These new methods belong to the class of finite-difference methods. Thus, in this chapter, we will restrict ourselves to the aforementioned class. As we shall see, there are two families of methods that can be employed: direct and iterative. We will focus on the iterative methods, which is the realm in which the methods we have developed fall (see [Arándiga, Donat, and Mulet 2000, Burden and Faires 2001, Briggs, Henson, and McCormick 2000] for more details).

2.2 Numerical resolution of elliptic PDEs

Let's see how we convert the problem of finding a solution of an elliptic PDE in finding the solution of a system of linear equations. Firstly, the PDE that we are considering will have a domain of definition. What we are going to do is to define a mesh of points in this domain, so that we try to find the solution not at any point in the domain but only at these discrete set of points. Then we replace the differentials of the equations by difference quotients using the points of the grid, employing the so-called finite differences approximation.

To fix ideas, we consider the Poisson equation in two spatial dimensions (2D) as a handy reference case (highly relevant for the ulterior applications we aim to). Using Cartesian coordinates the aforementioned equation reads:

$$\Delta u(x,y) := \frac{\partial^2}{\partial x^2}u(x,y) + \frac{\partial^2}{\partial y^2}u(x,y) = b(x,y), \tag{2.1}$$

over the domain $\Omega = \{(x,y)|x_{\min} < x < x_{\max}, y_{\min} < y < y_{\max}\}$. To properly specify the problem, we must provide suitable boundary conditions, e.g., $u(x,y) = w(x,y)$ for $(x,y) \in \partial\Omega$. Using a second-order central difference approximation for the second derivatives, we can express the Poisson equation in the arbitrary points of the mesh (x_i, y_j) as:

$$\frac{u(x_{i+1},y_j) - 2u(x_i,y_j) + u(x_{i-1},y_j)}{h^2} + \frac{u(x_i,y_{j+1}) - 2u(x_i,y_j) + u(x_i,y_{j-1})}{k^2} = b(x_i,y_j) + \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\epsilon_i, y_j) + \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, \eta_j), \tag{2.2}$$

where we have partitioned the interval $[x_{\min}, x_{\max}]$ in p equal parts of size $h = (x_{\min} - x_{\max})/p$ and the interval $[y_{\min}, y_{\max}]$ in q equal parts of size $k = (y_{\min} - y_{\max})/q$. Integer indices i and j may take any values in $\{0, 1, \ldots, p\}$, and $\{0, 1, \ldots, q\}$, respectively; $x_i = ih$, and $y_j = jk$. Finally, $\epsilon_i \in (x_{i-1}, x_{i+1})$, $\eta_j \in (y_{j-1}, y_{j+1})$. We represent this discretized domain as $\Omega^{h,k}$.

Assembling altogether the points of the grid and employing the boundary information when necessary[1], we build up a linear system of the form

$$Au = b, \tag{2.3}$$

where the components of the vector u are the values of $u(x_i, y_j)$, for all $i = 1, ..., p-1$ and $j = 1, ...q-1$, thus, $A \in \mathcal{M}_{l,l}(\mathbb{R})$ and $u, b \in \mathbb{R}^l$ where $l = (p-1)(q-1)$. Solving it, we obtain an approximation to the solution with an error $O(h^2 + k^2)$.

Due to the compact stencil of the finite difference representation of the derivatives we have used in Eq. (2.2) the resulting linear system is sparse. A useful way of writing the sparse matrix A is employing the so-called "stencil" representation. For the previous example, the stencil representation of A reads

$$A = \begin{bmatrix} & \frac{1}{k^2} & \\ \frac{1}{h^2} & \frac{-2}{h^2} + \frac{-2}{k^2} & \frac{1}{h^2} \\ & \frac{1}{k^2} & \end{bmatrix} \tag{2.4}$$

[1] In this simple example, the type of boundaries are the so-called Dirichlet boundary conditions, where the value of the solution is set at the boundary. There are other types very often used as e.g., Neumann or Robin boundary conditions.

These systems, besides being sparse, are symmetric and positive (semi)-definite (depending on the boundary conditions), and, if properly ordered, banded. All these three characteristics are fundamental to choose a particular algorithm for the numerical solution of the linear system at hand.

2.3 Solving linear systems

If $\det(A) \neq 0$, the square linear system of Eq. (2.3) has a unique solution. However, when $l \gg 1$, it is necessary to employ optimized algorithms to solve the linear system efficiently from the computational cost point of view. These numerical methods fall into two basic classes: *direct methods* and *iterative methods*. An excellent guide to both types of methods can be found in [Gene H. Golub 2012]. Here we give a shallow overview of both classes of methods.

A direct method solves "exactly" (ideally up to machine precision) the linear sytem. Most direct methods are based on the Gaussian elimination procedure in some of their steps. In other steps of a direct method, matrix A is factorized into simpler matrices. Likely, the most popular factorization is the so-called LU decomposition (along with its more general cases PLU, and $PLUQ$ decompositions), which have efficient implementations in some particular cases as the Crout-Doolittle algorithm or the Choleski decomposition (the latter reserved for positive definite A-matrices). There are other methods usually based on the fast Fourier transform or the method of cyclic reduction which are very efficient for particular types of matrices A (e.g., in the case A is a circulant matrix).

Iterative methods, whose classical prototypes are the methods of Jacobi, Gauss-Seidel, SOR, Conjugate Gradient and Chebyshev, start with an initial guess of the solution. Then, they construct a succession of approximate solutions that converge to the exact solution of the system at a reasonable rate. Each new approximation is achieved by a "small" correction step to the previous one. These classical methods are very easy to implement, can be applied to more general linear systems, and are more appropriate for large sparse matrices, as is our case. In practical terms, the succession of approximate solutions stops after a prescribed error goal is reached. Two appreciated advantages of iterative methods over direct methods are that they do not destroy the sparse structure of the system and that at every iteration we have an approximation to the solution. On the contrary, convergence to the solution might be slow, and it is only guaranteed for certain types of matrices A. Since the new algorithm we

have developed to solve elliptic systems of PDEs belongs to the class of iterative methods, we go deeper into their properties in the following sections.

2.4 Iterative methods

In order to build an iterative scheme, we aim to convert our linear system Eq. (2.3) in another equivalent system of the form

$$u = Tu + c. \tag{2.5}$$

We perform an affine decomposition of the matrix of the system $A = M - N$, expressing it as a difference between a "simpler" matrix M, the so-called *preconditioner*, and the rest of A, so that the system Eq. (2.3) is rewritten as follows:

$$Mu = Nu + b, \tag{2.6}$$

which multiplied by M^{-1} in both sides becomes

$$u = M^{-1}Nu + M^{-1}b. \tag{2.7}$$

Now, Eq. (2.7) is of the form Eq. (2.5) identifying $T = M^{-1}N$ and $c = M^{-1}b$.

The next step is to solve Eq. (2.3) iteratively. We denote with u the exact solution of the system. We start with a finite initial guess u^0 and construct the succession of approximated solutions $\{u^k\}$ that satisfy

$$u^{n+1} = M^{-1}\big(Nu^n + b\big) \tag{2.8}$$

We would like that the succession of approximations $\{u^n\}$ converges to the exact solution u, i.e., that $\{u^n\} \to u$ when $n \to \infty$. But this is not guaranteed unless matrices M and N satisfy certain conditions, which we address below.

We define the error ϵ^n at the nth iterative step as:

$$\epsilon^n = u^n - u\,. \tag{2.9}$$

Since both approximated solutions and the exact solution fulfill Eq. (2.7), the *iteration matrix*[2] $G = M^{-1}N$ relates the error between two consecutive iterations:

$$\epsilon^n = G\epsilon^{n-1}. \tag{2.10}$$

Applying inductively Eq. (2.10), between any given iteration and the initial error ϵ^0 we obtain

$$\epsilon^n = G^n\epsilon^0. \tag{2.11}$$

[2]Also know as the *amplification matrix*

With this, it is evident that

$$\{u^n\} \to u \iff \{||\epsilon^n||\} \to 0 \iff \{||G^n\epsilon^0||\} \to 0, \forall\epsilon^0 \iff ||G^n|| \to 0, \text{as } n \to \infty \tag{2.12}$$

where $||.||$ denotes either a vector or a matrix norm depending on the argument it encloses. The latter condition on matrix G, requires

$$||G|| < 1 \tag{2.13}$$

in some matricial norm. Condition Eq. (2.13) is equivalent to require that

$$\varrho(G) < 1, \tag{2.14}$$

where ϱ is the *spectral radius* of a matrix, i.e., the maximum of the absolute values of its eigenvalues. Thus, G must have a spectral radius smaller than the unity in order for the iterative method to converge. The intuitive idea behind all the above reasoning is simple. The error is multiplied by the iteration matrix at each step. We want the error to become negligibly small as the number of iterations grows. This is only possible if the spectral radius of the matrix is smaller than 1.

We point out that conditions Eq. (2.13) or Eq. (2.14) may be very restrictive when applied to generic ill-conditioned linear problems (e.g., as those arising from optimization; Gould 2000). In such cases, preconditioning techniques are typically employed. However, the class of problems to which we aim to apply iterative methods, i.e., resulting from the discretization of ePDEs are typically rather well behaved.

2.4.1 Jacobi and Gauss-Seidel

If we choose $M = D$, where D is formed with the diagonal elements of A and zeros elsewhere, the resulting algorithm is the *Jacobi* method [Jacobi 1845]. This choice of the matrix M is driven by the fact that the preconditioner, shall be as simple as possible to easily compute its inverse. With this choice, and knowing that every matrix A can be written as $A = L + D + U$, where L is the strictly lower triangular part of A, D is the diagonal, as we have already said, and U is the strictly upper triangular part of A, we have that the Jacobi iteration reads

$$u^{n+1} = D^{-1}\big[b - (L + U)u^n\big] \tag{2.15}$$

which, in components, is

$$u_i^{n+1} = \frac{1}{a_{i,i}}\left[b_k - \sum_{k=1}^{i-1} a_{i,k}u_k^n - \sum_{k=i+1}^{l} a_{i,k}u_k^n\right] \tag{2.16}$$

for $i \in [1, l]$. Furthermore, if the matrix A is sparse, as we obtain by discretizing elliptic PDEs, the sums in Eq. (2.16) have very few terms. For example, in the case of Eq. (2.2), and after taking into account that all the unknowns are stacked in a one-dimensional vector in the previous expression, we simply have:

$$u_{i,j}^{n+1} = \frac{1}{\frac{-2}{h} + \frac{-2}{k}} \left[b_{i,j} - \frac{u_{i-1,j}^n + u_{i+1,j}^n}{h^2} - \frac{u_{i,j-1}^n + u_{i,j+1}^n}{k^2} \right] \tag{2.17}$$

with $i \in [1, p-1]$ and $j \in [1, q-1]$

The *Gauss-Seidel* method is a simple modification to the Jacobi method. As it stands in Eq. (2.15), to compute the approximate solution at the iteration $n+1$, u^{n+1}, we only need the solution in the previous iteration, u^n. However, by the time we compute the i-th component of u^{n+1}, namely, u_i^{n+1}, we have already calculated all components from u_0^{n+1} to u_{i-1}^{n+1}. We can take advantage of the already computed components of the solution in $n+1$ to rewrite, Eq. (2.17) as

$$u_i^{n+1} = \frac{1}{a_{i,i}} \left[b_i - \sum_{k=1}^{i-1} a_{i,k} u_k^{n+1} - \sum_{k=i+1}^{l} a_{i,k} u_k^n \right], \tag{2.18}$$

which, in matrix form, is

$$u^{n+1} = (L + D)^{-1} \left[b - U u^n \right]. \tag{2.19}$$

Both, Jacobi and Gauss-Seidel methods are known to converge if matrix A is diagonal-dominant, very often the case in linear systems stemming from the discretization of elliptic systems of PDEs.

Summarizing, these classical methods are iterative algorithms to solve systems of linear equations. Due to their simplicity and their convergence properties, they are popular choices. However, their slow rate of convergence, compared to other iterative methods make them poor choices to solve linear systems.

2.4.2 Weighted Jacobi

The weighted Jacobi method (also known as simultaneous overrelaxation method [Young and (Auth.) 1971]) is a variant of the Jacobi method where the afine decomposition of matrix A is $A = M_\omega - N_\omega$, and $M_\omega = D/\omega$, where $\omega \in \mathbb{R}$ is chosen to minimize the spectral radius of the iteration matrix $G_\omega = M_\omega^{-1} N_\omega = (1-\omega)I - \omega D^{-1}(L+U)$ (L, U and D matrices are the same as in the Jacobi method; Sec. 2.4.1. Thus, in matrix form the weighted Jacobi method reads

$$u^{n+1} = \omega D^{-1} \left[b - (L+U) u^n \right] + (1-\omega) u^n. \tag{2.20}$$

In the case of the weighted Jacobi scheme applied to the Laplace equation, as an example of a system with sparse matrix, condition Eq. (2.14) on the iteration matrix can be approximated to a condition over the so-called *amplification factor*:

$$G = 1 - \omega\kappa, \tag{2.21}$$

which is obtained from the stability analysis of Von Neumann. The κ is a function of the wave numbers. It will be different depending on the equation we use, its discretization, the type of border, etc. This type of expressions will also play an important role in the Chap. 3.

2.4.3 Successive overrelaxation (SOR)

There is a simple yet important modification that can be made both to the Jacobi and to the Gauss-Seidel methods. Suppose that we have calculated the new iterate as shown above for each of the methods. Let us consider this value, which we now denote by $\tilde{u}^{n+1}$, as an intermediate value. Then, the new iterate is given by a weighted average with the previous iterate u^n as follows:

$$u^{n+1} = \omega\tilde{u}^{n+1} + (1-\omega)u^n \tag{2.22}$$

where $\omega \in \mathbb{R}$ is the weight that may be chosen. This generates a family of iterative methods depending on the value of ω. Applied to the case of Jacobi, we obtain the *weighted Jacobi* obtained in the previous section. For Gauss-Seidel, we obtain the SOR method. The preconditioner in this method is $M = \omega L + D$. Note that if we take $\omega = 1$ we recover the classical Gauss-Seidel methods.

2.4.4 Scheduled Relaxation Jacobi and Chebyshev-Jacobi method

The classical iterative schemes shown in the previous sections are robust, but their low convergence rates makes them impractical for computationally intensive applications. The only exception to this assertion comes from the SOR method. SOR can have excellent convergence properties if one knows the optimal weight for the matrix A at hand. However, its convergence rate is very sensitive to the exact value of ω used. Unfortunately, in many cases, the optimal value of ω is unknown.

The scheduled relaxation Jacobi method [Yang and Mittal 2014], SRJ hereafter, is an extension of the classical Jacobi method, which increases the rate of convergence in the case of linear problems that arise in the finite difference

discretization of elliptic equations. It consists on executing a series of weighted Jacobi steps with carefully chosen values for the weights (possibly a different weight for each successive iteration) in a sequence. The Chebyshev-Jacobi method (CJM), is the optimum of all SRJ methods for a given problem. In the CJM all the weighted Jacobi steps possess a different and unique weight. The CJM has proben to perform better than any SRJ algorithm in Adsuara et al. 2016.

2.5 Organisation of the thesis

Part II of this thesis is devoted to the SRJ method. In Chap. 3 we present the SRJ method for linear systems and propose some improvements. As we shall see, the number of weights plays an important role in the convergence properties of the numerical scheme, being faster for a larger number of weights. In Chap. 4 we obtain the optimal SRJ schemes of all possible ones and show that this method, the Chebyshev-Jacobi method (CJM hereafter), is equivalent to a Generalized Richardson iterative scheme. We present a procedure to obtain analytically the weights for linear systems arising from the finite-difference discretization of elliptic systems of PDEs.

In Part III we present numerical applications of the SRJ method, with interest in in the field of astrophysics. Chap. 5 presents a series of problems, and compares the performance of the new method with other methods in the literature. In Chap. 6 we explore the performance of the method in parallel architectures, namely using the OpenMP/MPI models for multi-core machines and an implementation for Graphics Processing Units (GPUs). Chap. 7 is devoted to an application outside the the field of astrophysics, in which we model normal vibrations of the human eye.

In Part IV we present our conclusions and briefly outlook future prospects of the methods presented in this thesis.

Part II

From Scheduled Relaxation Jacobi to Chebyshev-Jacobi

Chapter 3

Scheduled Relaxation Jacobi method: improvements and applications

The results of this chapter have been originally published in:

Improvements in the Scheduled Relaxation Jacobi method,
JE Adsuara, I Cordero-Carrión, P Cerdá-Durán, MA Aloy,
Proceedings of the XXIV Congress on Differential Equations and Applications/XIV Congress on Applied Mathematics, Cádiz, June 8-12, 2015, Servicio de Publicaciones de la Universidad de Cádiz (2015)

Scheduled relaxation Jacobi method: improvements and applications,
JE Adsuara, I Cordero-Carrión, P Cerdá-Durán, MA Aloy,
Journal of Computational Physics 321, 369-413 (2016)

The text in the following sections corresponds to an edited version of the aforementioned publications.

3.1 Introduction

As we have motivated in the previous chapter, we aim at solving PDEs associated with elliptic problems with interest in astrophysics. We focus here in iterative methods. One of the simplest and most studied iterative schemes is the so

called Jacobi method [Jacobi 1845, Richardson 1911], whose main drawback is its poor convergence rate. In oder to improve the efficiency of the Jacobi method, many alternatives have been considered. A popular possibility is the use of preconditioners [Juncosa and Mulliken 1960, Gunawardena, Jain, and Snyder 1991, Noutsos and Tzoumas 2006] applied to linear systems, that make the associated Jacobi and Gauss-Seidel methods converge asymptotically faster than the unpreconditioned ones. Indeed, the method we improve on here, can be adapted as a preconditioner for other methods (e.g., the conjugate gradient method). Very widespread is the use of multigrid methods [e.g., Trottenberg, Oosterlee, and Schüller 2001] that, in many cases, provide the solution with $\mathcal{O}(N)$ operations, or that can even be employed as preconditioners. Relaxation algorithms [originally introduced in Richardson 1911], improve the performance of the Jacobi method by considering modifications of the Gauss-Seidel algorithm that include a weight, for instance, successive overrelaxation (SOR) methods [Young 1954a].

Along this line, [Yang and Mittal 2014, YM14 henceforth] have recently presented a significant acceleration (of the order of 100) over the Jacobi algorithm, employing the Scheduled Relaxation Jacobi method. The SRJ method is a generalization of the weighted Jacobi method which adds an overrelaxation factor to the classical Jacobi in a similar fashion to the SOR method. This generalization includes a number P of different levels, in each of which, the overrelaxation (or underrelaxation) parameter or weight is tuned to achieve a significant reduction of the number of iterations, thus leading to a faster convergence rate. The optimal set of weights depends on the actual discretization of the problem at hand. Although the method greatly improves the convergence rate with respect to the original Jacobi, the schemes presented by YM14, optimal up to $P = 5$ and resolutions of up to 512 points per spatial dimension, are still not competitive with other methods used currently in the field (e.g., spectral methods [Bjorstad and Widlund 1986], or multigrid methods as commented above). The main advantage of the SRJ method over other alternatives to solve numerically ePDEs is its simplicity and the straightforward parallelization, since SRJ methods preserve the insensitivity of the Jacobi method to domain decomposition (in contrast, e.g., to multigrid methods).

The structure of this chapter is as follows. We begin giving an overview of the SRJ method (Sect. 3.2) and describing the original methodology for obtaining optimal schemes. In Sect. 3.3 we present basic improvements on the original work by YM14. Then, we present in Sect. 3.3.4, some analytical work which reduces the number of unknowns to solve for to $\mathcal{O}(P)$ (instead of $\mathcal{O}(P^2)$ as in YM14 and

Sect. 3.3). In Sect. 3.4 we show a comparison of the new method to compute optimal parameters for SRJ schemes with that of YM14.[1] Furthermore, we test the SRJ methods in a case study, namely a Poisson equation with Dirichlet boundary conditions (Sect. 3.5) that has analytic solution, and show that optimal SRJ parameters computed for resolutions close to that of the problem at hand can bring two orders of magnitude smaller number of iterations than the Jacobi method to solve such the problem.

3.2 SRJ schemes

Now we recap the most salient results obtained by YM14 and set the notation for the rest of the section.

First of all, if we define $\omega_i J$ as a single step in a weighted Jacobi iteration using the weight ω_i $(i = 1, \ldots, P)$, then the SRJ method can be cast as a successive application of elementary relaxation steps of the form

$$\overbrace{\overbrace{\omega_1 J \ldots \omega_1 J}^{q_1} \underbrace{\omega_2 J \ldots \omega_2 J}_{q_2} \ldots \overbrace{\omega_P J \ldots \omega_P J}^{q_P}}^{M} \overbrace{\omega_1 J \ldots \omega_1 J}^{q_1} \ldots ,$$

where the largest weight, ω_1 is applied q_1 times, and each of the remaining and progressively smaller weights ω_i $(i = 1, \ldots, P)$ is applied q_i times, respectively. A single cycle of the scheme ends after M elementary steps, where $M := \sum_{i=1}^{P} q_i$. In order to reach a prescribed tolerance goal, we need to repeat a number times the basic M-cycle of the SRJ method. Both, a vector of weights and a vector with the number of times we use each weight, define each optimal scheme. We emphasize that, from the point of view of the implementation, the only difference with the traditional weighted Jacobi is that, instead of having a fixed weight, SRJ schemes of P-levels require the computation of P weights.

In order to simplify the notation, we define $\boldsymbol{\omega} := (\omega_1, \ldots, \omega_P)$, with $\omega_i > \omega_{i+1}$ and $\boldsymbol{q} := (q_1, \ldots, q_P)$ which is in one-to-one correspondence with the previous $\boldsymbol{\omega}$. Also, we define $\boldsymbol{\beta} := (\beta_1, \ldots, \beta_P)$, where $\beta_i = q_i/M$ is the fraction of the iteration counts that a given weight ω_i is repeated in an M-cycle, with $\beta_P := 1 - \sum_{i=1}^{P-1} \beta_i$.

The basic idea of the SRJ schemes is finding the optimal values for $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$ that minimize the total number of iterations to reach a prescribed tolerance for a given number of points (i.e., numerical resolution) N.

[1] ASCII files containing the optimal parameters for the SRJ algorithms shown here with P values between 6 and 15 can be found at `http://www.uv.es/camap/SRJ.html`.

3.3 Finding the optimal parameters

Below we explain how to compute the optimal values of $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$ for a fixed value of P, following the prescription of YM14 and rewrite some parts of the YM14 algorithm to make them amenable for extension to a larger number of levels and resolutions.

3.3.1 Convergence analysis and optimization problem

We perform a convergence analysis of the method in order to obtain a number of restrictions that the parameters of the SRJ scheme must fulfill. As a model problem, we use the Laplace equation with homogeneous Neumann boundary conditions in two spatial dimensions, in Cartesian coordinates and over a domain with unitary size:

$$\begin{cases} \dfrac{\partial^2}{\partial x^2}u(x,y) + \dfrac{\partial^2}{\partial y^2}u(x,y) = 0, & (x,y) \in (0,1)\times(0,1) \\ \dfrac{\partial}{\partial x}\bigg|_{x=0} u(x,y) = 0,\ \dfrac{\partial}{\partial x}\bigg|_{x=1} u(x,y) = 0, & y \in (0,1) \\ \dfrac{\partial}{\partial y}\bigg|_{y=0} u(x,y) = 0,\ \dfrac{\partial}{\partial y}\bigg|_{y=1} u(x,y) = 0, & x \in (0,1). \end{cases} \tag{3.1}$$

We note that the system resulting from discretization of Eq. (3.1) is not invertible. There are inifinite possible solutions due to the choice of Newmann boundary conditions. Actually, u =constant is a solution of Eq. (3.1). The reason to employ this problem is because it was first considered by YM14, and we aimed to compare our algorithm with theirs as closely as possible. Subsequent test problems do not fix all boundary conditions to be of Newmann type and, therefore, they will have a unique solution. In this particular problem, we employ as initial data random values for u.

Let us consider a 2nd-order central-difference discretization of Eq. (3.1) on a uniform grid consisting of $N_x \times N_y$ zones, and define $N = \max(N_x, N_y)$. Then, we apply the Jacobi method with a relaxation parameter ω, so that the following iterative scheme results:

$$u_{i,j}^{n+1} = (1-\omega)u_{i,j}^n + \frac{\omega}{4}(u_{i,j-1}^n + u_{i,j+1}^n + u_{i-1,j}^n + u_{i+1,j}^n) \tag{3.2}$$

$$= u_{i,j}^n + \frac{\omega}{4}(u_{i,j-1}^n + u_{i,j+1}^n + u_{i-1,j}^n + u_{i+1,j}^n - 4u_{ij}^n), \tag{3.3}$$

where n is the index of iteration.

At this point, we perform a von Neumann stability analysis for obtaining the amplification factor,

$$G_\omega(\kappa) = (1 - \omega\kappa), \tag{3.4}$$

where κ is a function of the wave-numbers in each dimension. For the problem at hand (see App. C),

$$\kappa(k_x, k_y) = \sin^2\left(\frac{k_x}{2N_x}\right) + \sin^2\left(\frac{k_y}{2N_y}\right). \tag{3.5}$$

G_ω expresses how much the error can grow up from one iteration to the next one using the relaxation Jacobi method. Thus, if a single relaxation step is performed, we require $|G_\omega| < 1$ to ensure convergence. However, in an SRJ scheme, we perform a series of M-cycles (Sect. 3.2). Hence, even if on an elementary step of the algorithm one may violate the condition $|G_\omega| < 1$ (which may happen, e.g., if such step is an overrelaxation of the Jacobi method), the condition for convergence shall be obtained for the composition of M elementary amplification factors. As we apply Eq. (3.3) M-times but with P different weights ω_i, the following composition of amplifications factors is obtained:

$$\overbrace{\overbrace{G_{\omega_1}\dots G_{\omega_1}}^{q_1}\underbrace{G_{\omega_2}\dots G_{\omega_2}}_{q_2}\dots\overbrace{G_{\omega_P}\dots G_{\omega_P}}^{q_P}}^{M} = \prod_{i=1}^{P} {G_{\omega_i}}^{q_i}. \tag{3.6}$$

Finally, it is not important how many times we use each of the weights q_i but their relative frequency of use during an M-cycle, which is defined by β_i. Therefore, following YM14, one can define the per-iteration amplification factor function as a geometric mean of the modulus of the cycle amplification factor (Eq. 3.6):

$$\Gamma(\kappa) = \prod_{i=1}^{P} |1 - \omega_i\kappa|^{\beta_i}. \tag{3.7}$$

The previous transformation is very convenient to find deterministic optimal parameters for the SRJ schemes, since it avoids working with a Diophantine equation (Eq. 3.6), because $q_i \in \mathbb{N}$, while $\beta_i \in \mathbb{R}$.

From the definition of $\Gamma(\kappa)$, it is evident that larger values of the per-iteration amplification factor yield a larger number of iterations for the algorithm to converge. Thus, the optimal values for the SRJ parameters are obtained by looking for the extrema of $\Gamma(\kappa)$ in $[\kappa_{\min}, \kappa_{\max}]$, which is the interval bounding

the allowed values of κ, namely

$$\kappa_{\min} = \sin^2\left(\frac{\pi}{2N}\right), \quad \kappa_{\max} = 2\,, \tag{3.8}$$

and then, to minimize *globally* these extrema, so that the error per iteration decreases as much as possible. This sets our (*min-max*) optimization problem.

We explicitly point out that the value of $\kappa_{\min}$ depends on the type of boundary conditions of the problem, on the discretization of the elliptic operator and on the dimensionality of the problem. This is not the case for $\kappa_{\max}$, which equals 2 independent on the boundary conditions and dimensionality of the problem, though it depends on the discretization of the elliptic operator. For later reference, we write the explicit form of $\kappa_{\min}$ as a function of the number of dimensions, d, for a Cartesian discretization of the elliptic operator and Neumann boundary conditions:

$$\kappa_{\min}^{(\mathrm{d})} = \frac{2}{d}\sin^2\left(\frac{\pi}{2N^{(\mathrm{d})}}\right). \tag{3.9}$$

Obviously, we recover Eq. (3.8) setting $d = 2$. For practical purposes, it is possible to obtain the optimal SRJ parameters for any value of d once we know the optimal parameters in 2D. This is done by computing the effective number of points in 2D, $N_{\mathrm{eff}}^{(2)}$, corresponding to a given problem size $N^{(\mathrm{d})}$ in d-dimensions through the relation:

$$N_{\mathrm{eff}}^{(2)} = \frac{\pi}{2\arcsin\left(\sqrt{\frac{2}{d}}\sin\left(\frac{\pi}{2N^{(\mathrm{d})}}\right)\right)} \simeq N^{(\mathrm{d})}\sqrt{\frac{d}{2}}, \tag{3.10}$$

where the approximated result holds for large values of $N^{(\mathrm{d})}$.

Finally, as stated above, the values of $\kappa_{\min}$ change depending on whether Neumann or Dirichlet boundary conditions are considered, and so the optimal parameters change. Fortunately, there is a simple way to obtain the optimal parameters in case of Dirichlet boundary conditions from the 2D optimal parameters computed for Neumann boundary conditions, namely

$$N_{\mathrm{eff}}^{(2)} = \frac{\pi}{2\arcsin\left(\sqrt{\frac{2}{d}\sum_{i=1}^{d}\sin^2\left(\frac{\pi}{2N_{i,\mathrm{Dirichlet}}^{(\mathrm{d})}}\right)}\right)}$$

$$\simeq \frac{\sqrt{\frac{d}{2}}}{\sqrt{\sum_{i=1}^{d} \frac{1}{(N^{(\mathrm{d})}_{i,\mathrm{Dirichlet}})^2}}}, \tag{3.11}$$

where the approximated result holds when the number of points in each dimension is sufficiently large.

3.3.2 The non-linear system

In the optimization problem stated previously we need to compute the location of the extrema $\Gamma(\kappa)$, $\boldsymbol{\kappa}$ hereafter. From the optimization process one must also obtain the rest of the parameters of the SRJ scheme, namely, $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$. Thus, we need to solve a system $\mathcal{S}(\boldsymbol{\omega}, \boldsymbol{\beta}, \boldsymbol{\kappa})$ to determine all these unknowns.

For the evaluation of $\boldsymbol{\kappa}$, we must take into account that the location of the maxima can be either at the edges of the domain, $\kappa_0 := \kappa_{\min}$ and $\kappa_P := \kappa_{\max}$ (set by Eq. 3.8), or in other $P-1$ internal values κ_i $(i = 1, \ldots, P-1)$ determined (each of them) by the following condition:

$$\frac{\partial}{\partial \kappa} \log \Gamma(\kappa) = \sum_{i=1}^{P} \frac{\beta_i \omega_i}{1 - \kappa \omega_i} = 0. \tag{3.12}$$

From the solutions of Eq. (3.12), we obtain the $P-1$ different $\kappa_i = \kappa_i(\boldsymbol{\omega}, \boldsymbol{\beta})$, which allows us to reduce the number of unknowns of the system $\mathcal{S}(\boldsymbol{\omega}, \boldsymbol{\beta}, \boldsymbol{\kappa})$ from $3P-2$ $(\omega_1, \ldots, \omega_P, \beta_1, \ldots, \beta_{P-1}, \kappa_1, \ldots, \kappa_{P-1})$ to $2P-1$. Hence, we need to also obtain $2P-1$ equations to find a unique solution of the system.

For obtaining the set of the first P equations for the system, and following YM14, we equalize all the maxima[2]:

$$\Gamma(\kappa_0) = \Gamma(\kappa_i), \;\; i = 1, \ldots, P. \tag{3.13}$$

Furthermore, if we assume that $\boldsymbol{\omega} = \boldsymbol{\omega}(\boldsymbol{\beta})$, and therefore $\boldsymbol{\kappa} = \boldsymbol{\kappa}(\boldsymbol{\beta})$, a second set of $P-1$ equations can be obtained from the minimization of $\Gamma(\kappa_0)$:

$$\frac{\partial}{\partial \beta_j} \Gamma(\kappa_0) = 0, \; j = 1, \ldots P-1. \tag{3.14}$$

Thereby, our system is now $S(\boldsymbol{\omega}, \boldsymbol{\beta}, \frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\beta}})$, since the differentiation in Eq. (3.14) introduces $P(P-1)$ new ancillary variables, namely $\frac{\partial \omega_i}{\partial \beta_j}$; $i = 1, \ldots, P$, $j =$

[2]Suppose that the global maximum of Γ only is reached for κ_j. Then, $\partial \Gamma(\kappa_j)/\partial \omega_j < 0$ and it is possible to increase the value of ω_j a little. This leads to contradiction, since $\Gamma(\kappa_j)$ would be maximum global and strictly greater than the other local maxima, but with $\Gamma(\kappa_j)$ smaller. Thus, the local maximums that reach the global maximum must be at least two. Following the same idea, we come to equalize all of them.

$1, \ldots, P-1$. The final set of $P(P-1)$ additional equations to account for this extra ancillary variables results from applying the same condition as in Eq. (3.14) to the remaining values of κ_i, deduced from Eqs. (3.13) and (3.14):

$$0 = \frac{\partial}{\partial \beta_j}\Gamma(\kappa_i),\ i = 1, \ldots P,\ j = 1 \ldots P-1. \tag{3.15}$$

We note that we have set up the problem taking $\boldsymbol{\beta}$ as the reference variables, assuming that $\boldsymbol{\omega}$ were, in fact, functions of these reference variables, i.e., $\boldsymbol{\omega} = \boldsymbol{\omega}(\boldsymbol{\beta})$. However, this is not the only possibility. We have also considered, alternatively, $\boldsymbol{\omega}$ as reference variables and $\boldsymbol{\beta}$ as functions $\boldsymbol{\beta}(\boldsymbol{\omega})$, in which case, the $P-1$ equations equivalent to those of Eq. (3.15) read:

$$0 = \frac{\partial}{\partial \omega_j}\Gamma(\kappa_i),\ i = 1, \ldots P,\ j = 1 \ldots P-1. \tag{3.16}$$

We have solved the new resulting system employing the same methodology sketched above, and found that the solution is the same regardless of the set of reference variables employed, as it should be.

3.3.3 Basic improvements on the original SRJ algorithm

In order to make the SRJ method competitive with other existing algorithms to solve ePDEs, we must find the optimal parameters of SRJ schemes with a *sufficiently* large number of levels. Furthermore, since the optimal parameters depend on the resolution of the discretization used to solve a given problem, we also need to compute optimal parameters for a range of numerical resolutions larger than in YM14. Here we present some improvements on the SRJ algorithm, which allow us to solve the system from $P = 6$ up to $P = 10$ and for resolutions up to 2^{15} (see also [Adsuara et al. 2015], ACCA15 hereafter). Additional improvements based on analytical results will be commented in Sect. 3.3.4

Firstly, the stiffness of $S(\boldsymbol{\omega}, \boldsymbol{\beta}, \frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\beta}})$ increases with the number of levels P, which prevented YM14 to compute optimal SRJ schemes for $P > 5$. We have been able to reduce the complexity of the numerical solution by manipulating parts of them algebraically. On the one hand, we have hidden the part of the non-linear system involving the $\boldsymbol{\kappa}$ unknowns by solving for them symbolically and using those symbolic placeholders later when solving numerically for $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$. On the other hand, and after the previous manipulation, we have seen in Sect. 3.3.2 that we need to solve numerically a non-linear system of $P^2 + P - 1$ equations with the same number of unknowns. We aim to rewrite this system $S(\boldsymbol{\omega}, \boldsymbol{\beta}, \frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\beta}})$ as $S(\boldsymbol{\omega}, \boldsymbol{\beta})$, which requires obtaining $\frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\beta}}$ as a function of $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$. We also compute the solutions of this linear subsystem symbolically. These

manipulations of the original system of equations permitted us to compute the optimal SRJ parameters for a larger number of levels (up to $P = 10$) than in YM14.

Secondly, in order to increase the number of points, for the numerical solution of the system, since the accuracy of the results for large values of P critically depends on having sufficiently large precision, we needed Mathematica's ability to perform arbitrary precision arithmetic. We ended up using up to twenty four digits in the representation of the numbers in some cases.

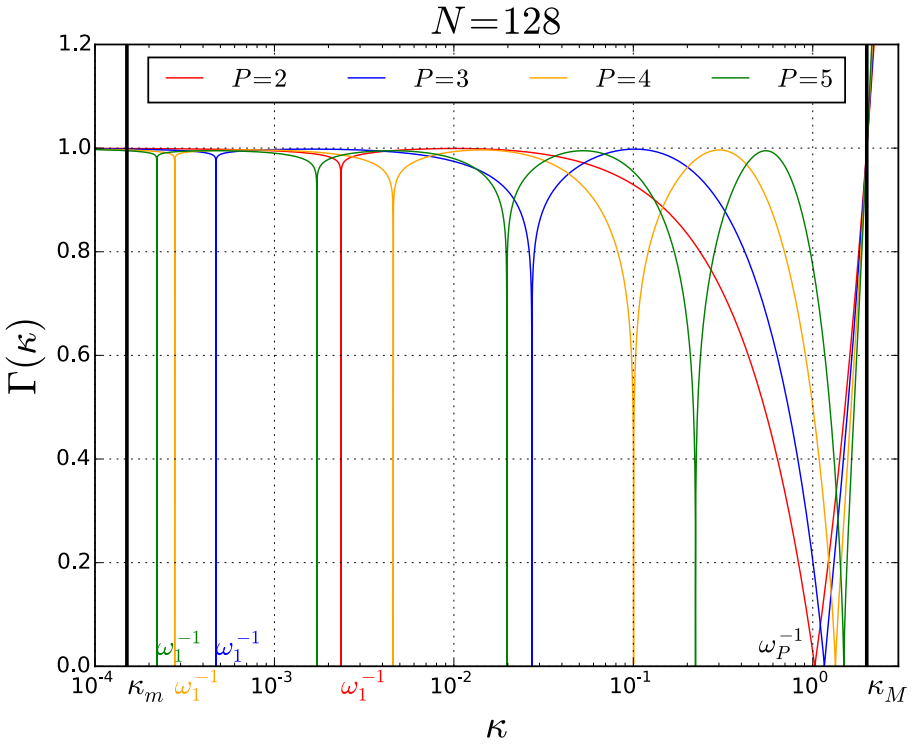


Figure 3.1 $\Gamma(\kappa)$ functions corresponding to different values of the numbers of levels P for $N = 128$. The inverse of the minimal weight, ω_1^{-1}, and the inverse of the maximal one, ω_P^{-1} moves towards $\kappa_{\min}$ and $\kappa_{\max}$, respectively, when P increases. The rest of the weights are distributed roughly logarithmically equally spaced inside (ω_1, ω_P).

Finally, the non-linear system we have to solve is very sensitive to the initial values that we guess for the unknowns. We developed a systematic way of setting the initial guesses from the values obtained from lower levels. The initial guesses used now are an improved version of what we used in ACCA15. As we can see in Fig. 3.1, when we increase the number of levels P, the values of ω_1^{-1} and ω_P^{-1} move towards $\kappa_{\min}$ and $\kappa_{\max}$, respectively. We can also observe that the

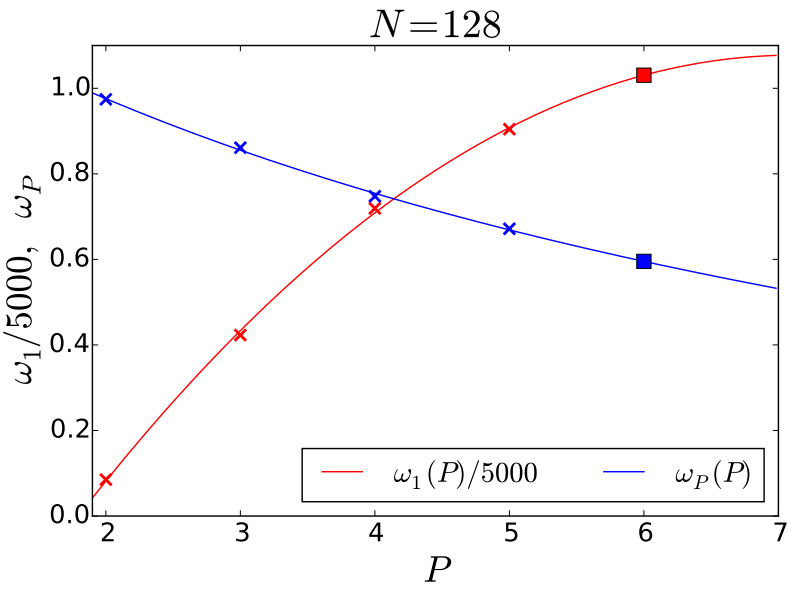


Figure 3.2 Discrete values of ω_1 and of ω_P for the $P = 6$, $N = 128$ scheme, together with both fits of their respective values as a function of P to conics as described in Sect. 3.3.3.

inverse of the rest of the weights of a scheme are roughly logarithmically equally spaced between the values ω_1^{-1} and ω_P^{-1}. Hence, defining $\tilde{R} = \omega_1/\omega_P$, we use as approximate location of the initial guesses for the inverse of any weight for an SRJ scheme of P levels the following expressions:

$$\{\omega_1^{-1}, \omega_1^{-1}\tilde{R}^{\frac{1}{2(P-1)}}, \omega_1^{-1}\tilde{R}^{\frac{3}{2(P-1)}}, \ldots, \omega_1^{-1}\tilde{R}^{\frac{P-4}{2(P-1)}}, \omega_1^{-1}\tilde{R}^{\frac{P-1}{2(P-1)}}, \omega_1^{-1}\tilde{R}^{\frac{P+2}{2(P-1)}}, \omega_1^{-1}\tilde{R}^{\frac{P+4}{2(P-1)}}, \ldots, \omega_1^{-1}\tilde{R}^{\frac{2P-3}{2(P-1)}}, \omega_1^{-1}\tilde{R} = \omega_P^{-1}\} \tag{3.17}$$

when P is odd, and

$$\{\omega_1^{-1}, \omega_1^{-1}\tilde{R}^{\frac{1}{2(P-1)}}, \omega_1^{-1}\tilde{R}^{\frac{3}{2(P-1)}}, \ldots, \omega_1^{-1}\tilde{R}^{\frac{P-5}{2(P-1)}}, \omega_1^{-1}\tilde{R}^{\frac{P-2}{2(P-1)}}, \ldots, \omega_1^{-1}\tilde{R}^{\frac{P}{2(P-1)}}, \ldots, \omega_1^{-1}\tilde{R}^{\frac{P+3}{2(P-1)}}, \omega_1^{-1}\tilde{R}^{\frac{P+5}{2(P-1)}}, \ldots, \omega_1^{-1}\tilde{R}^{\frac{2P-3}{2(P-1)}}, \omega_P^{-1}\} \tag{3.18}$$

when P is even.

Looking at Eqs. (3.17) and (3.18), the initial (guess) values of the weights for a new SRJ scheme with an additional level ($P' = P + 1$) can be built providing suitable estimates of the smallest and largest weights, which are obtained by fitting to two conics the values of the smallest and largest weights computed for SRJ schemes with $P - 3$ to P levels, and then extrapolating the result. For instance, in Fig. 3.2 we show the values of ω_1 and ω_P as a function of P with

red and blue symbols, respectively. If we want to obtain the initial values of ω_1 and ω_P for $P = 6$, we fit the values of ω_1 and, separately, those of ω_P for $P = 2, \ldots, 5$ to a hyperbola $\omega = \frac{a}{bP-c} + d$ or a parabola $\omega = aP^2 + bP + c$ depending on the flatness of the points, an infer the value for our P (in Fig. 3.2, the fit functions are plot with continuous lines, and the extrapolated values of ω_1 and ω_P for $P = 6$ with squares).

As we shall see in Sect. 3.3.4 and improving on the procedure outlined in this section, we do not need to provide initial values of $\boldsymbol{\beta}$, since they can be obtained analytically from the values of $\boldsymbol{\omega}$.

3.3.4 Advanced analytical improvements

So far, following basically the same procedure as in YM14, we have obtained optimal SRJ algorithms with up to $P = 10$ levels and multiple numerical resolutions. However, the limitations of the methodology of YM14 to compute optimal parameters for multilevel SRJ schemes prevents to develop algorithms with more than 10 levels. In this section, we will show a new methodology to evaluate the parameters of optimal SRJ schemes with up to $P = 15$ levels and resolutions of up to 2^{15} points per spatial dimension of the problem, which in some cases may yield accelerations of order 10^3 with respect to the Jacobi method. For a straightforward use of the newly developed SRJ schemes, we provide the readers with a comprehensive set of tables for different SRJ schemes and different resolutions in appendix B

Here we prove two important theorems, which tell us how to calculate analytically the ancillary variables $\frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\beta}}$ and the parameters $\boldsymbol{\beta}$ of the SRJ schemes in terms of the $\boldsymbol{\omega}$ and $\boldsymbol{\kappa}$ variables. Let us start with some technical results we need for the proof of these theorems. Notice that in all products appearing from now on, each index of the product refers only to expressions containing that particular index.

Lemma 1. *Let A and B be two matrices defined as $A := (a_{ij}) = \left(\frac{\kappa_i \, \beta_j}{1-\kappa_i \, \omega_j}\right)$ and $B := (b_{ij}) = \left(\frac{1-\omega_j/\omega_P}{1-\kappa_i \, \omega_j}\right)$, $i, j = 1, \ldots, P-1$, respectively. The inverse matrices, $A^{-1} = \tilde{A} = (\tilde{a}_{ij})$ and $B^{-1} = \tilde{B} = (\tilde{b}_{ij})$, are given by:*

$$\tilde{a}_{ij} = \prod_{\substack{k=1 \\ k \neq j}}^{P} \prod_{\substack{l=1 \\ l \neq i}}^{P} \frac{(1-\kappa_j \, \omega_i)(1-\kappa_k \, \omega_i)(1-\kappa_j \, \omega_l)}{\beta_i \, \kappa_j (\kappa_k - \kappa_j)(\omega_l - \omega_i)}, \tag{3.19}$$

$$\tilde{b}_{ij} = \frac{\omega_P\,(1-\kappa_j\,\omega_i)}{(1-\kappa_j\,\omega_P)} \prod_{\substack{k=1\\k\neq j}}^{P-1} \prod_{\substack{l=1\\l\neq i}}^{P} \frac{(1-\kappa_k\,\omega_i)(1-\kappa_j\,\omega_l)}{(\kappa_k-\kappa_j)(\omega_l-\omega_i)}. \tag{3.20}$$

Proof. We just need to check that $\sum_{j=1}^{P} \tilde{a}_{ij} a_{jm} = \sum_{j=1}^{P} \tilde{b}_{ij} b_{jm} = \delta_{im}$. For convenience, we define $\kappa_P := \kappa_{\max}$.

We will start checking that

$$\sum_{j=1}^{P} \frac{1}{(-1+\kappa_j\,\omega_m)} \left(\prod_{\substack{k=1\\k\neq j}}^{P} \prod_{\substack{l=1\\l\neq i}}^{P} \frac{(-1+\kappa_j\,\omega_l)}{(\kappa_j-\kappa_k)} \right) = \delta_{im} \left(\prod_{k=1}^{P} \prod_{\substack{l=1\\l\neq i}}^{P} \frac{(\omega_i-\omega_l)}{(-1+\kappa_k\,\omega_i)} \right). \tag{3.21}$$

We consider first the case where $i \neq m$. In general, taking into account that all the κ_i are strictly different, for a polynomial $F(x)$, with $\deg F(x) < P-1$, we can do a partial fraction decomposition of the following form:

$$\frac{F(x)}{\prod_{j=1}^{P-1}(x-\kappa_j)} = \sum_{j=1}^{P-1} \frac{F(\kappa_j)}{(x-\kappa_j)} \left(\prod_{\substack{k=1\\k\neq j}}^{P-1} \frac{1}{(\kappa_j-\kappa_k)} \right). \tag{3.22}$$

Considering $F(x) = \prod_{\substack{l=1\\l\neq i,m}}^{P} (-1+x\,\omega_l)$ in the above expression, and evaluating at $x=\kappa_P$, we get the desired expression.

We consider now the remaining case $i = m$. For convenience, we define $\kappa_0 = 1/\omega_i$, that satisfies $\kappa_0 \neq \kappa_j, j = 1,\ldots,P-1$. For a polynomial $F(x)$, with $\deg F(x) < P$, we can do a partial fraction decomposition of the following form:

$$\frac{F(x)}{\prod_{j=0}^{P-1}(x-\kappa_j)} = \sum_{j=0}^{P-1} \frac{F(\kappa_j)}{(x-\kappa_j)} \left(\prod_{\substack{k=0\\k\neq j}}^{P-1} \frac{1}{(\kappa_j-\kappa_k)} \right). \tag{3.23}$$

Considering $F(x) = \prod_{\substack{l=1\\l\neq i}}^{P}(-1+x\,\omega_l)$ in the above expression, and evaluating at $x=\kappa_P$, we get the desired result.

We use this equality to check the expression for the inverse matrix A^{-1} (as well as B^{-1} below):

$$\sum_{j=1}^{P} \tilde{a}_{ij} a_{jm} = \frac{\beta_m}{\beta_i} \sum_{j=1}^{P} \frac{(1-\kappa_j\,\omega_i)}{(1-\kappa_j\,\omega_m)} \prod_{\substack{k=1\\k\neq j}}^{P} \prod_{\substack{l=1\\l\neq i}}^{P} \frac{(-1+\kappa_k\,\omega_i)(-1+\kappa_j\,\omega_l)}{(\kappa_j-\kappa_k)(\omega_i-\omega_l)}$$

$$= \frac{\beta_m}{\beta_i} \left(\prod_{k=1}^{P} \prod_{\substack{l=1\\l\neq i}}^{P} \frac{(-1+\kappa_k\,\omega_i)}{(\omega_i-\omega_l)} \right) \sum_{j=1}^{P} \frac{1}{(-1+\kappa_j\,\omega_m)} \prod_{\substack{k=1\\k\neq j}}^{P} \prod_{\substack{l=1\\l\neq i}}^{P} \frac{(-1+\kappa_j\,\omega_l)}{(\kappa_j-\kappa_k)}$$

$$= \frac{\beta_m}{\beta_i} \left(\prod_{k=1}^{P} \prod_{\substack{l=1 \\ l\neq i}}^{P} \frac{(-1+\kappa_k\,\omega_i)}{(\omega_i-\omega_l)} \right) \left(\prod_{k=1}^{P} \prod_{\substack{l=1 \\ l\neq i}}^{P} \frac{(\omega_i-\omega_l)}{(-1+\kappa_k\omega_i)} \right) \delta_{im} = \delta_{im}. \tag{3.24}$$

Finally, we check the expression for the inverse matrix B^{-1}:

$$\sum_{j=1}^{P} \tilde{b}_{ij} b_{jm}$$

$$= \frac{(\omega_m-\omega_P)}{(\omega_i-\omega_P)} \left(\prod_{k=1}^{P-1} \prod_{\substack{l=1 \\ l\neq i}}^{P-1} \frac{(-1+\kappa_k\,\omega_i)}{(\omega_i-\omega_l)} \right) \sum_{j=1}^{P} \frac{-1}{(-1+\kappa_j\,\omega_m)} \prod_{\substack{k=1 \\ k\neq j}}^{P-1} \prod_{\substack{l=1 \\ l\neq i}}^{P-1} \frac{(-1+\kappa_j\,\omega_l)}{(\kappa_j-\kappa_k)}$$

$$= \frac{\omega_m-\omega_P}{\omega_i-\omega_P}\delta_{im} = \delta_{im}. \tag{3.25}$$

□

Theorem 1.

$$\frac{\partial}{\partial\beta_q}\omega_i = \sum_{j=1}^{P} \prod_{\substack{k=1 \\ k\neq j}}^{P} \prod_{\substack{l=1 \\ l\neq i}}^{P} \frac{(1-\kappa_j\omega_i)}{\beta_i\kappa_j} \frac{(1-\kappa_k\omega_i)}{(\kappa_k-\kappa_j)} \frac{(1-\kappa_j\omega_l)}{(\omega_l-\omega_i)} \log\left|\frac{1-\kappa_j\omega_q}{1-\kappa_j\omega_P}\right|. \tag{3.26}$$

Proof. We have that κ_j with $j = 1, \ldots, P-1$ are the roots where the local extrema are located. We will also use the already defined κ_P. Taking into account the Eqs. (3.14) and (3.15), for a fixed value of $l, 1 \leq l \leq P-1$, we construct the linear system $A\,\frac{\partial\boldsymbol{\omega}}{\partial\boldsymbol{\beta}} = \boldsymbol{f}$, which in components reads:

$$[a_{ij}] \cdot \begin{bmatrix} \frac{\partial}{\partial\beta_l}\omega_1 \\ \frac{\partial}{\partial\beta_l}\omega_2 \\ \vdots \\ \frac{\partial}{\partial\beta_l}\omega_P \end{bmatrix} = \begin{bmatrix} \log|1-\kappa_1\omega_l/1-\kappa_1\omega_P| \\ \log|1-\kappa_2\omega_l/1-\kappa_2\omega_P| \\ \vdots \\ \log|1-\kappa_P\omega_l/1-\kappa_P\omega_P| \end{bmatrix}, \tag{3.27}$$

where a_{ij} is defined in Lemma 1. We can solve for the ancillary variables $\frac{\partial\omega_i}{\partial\beta_j}$, $i = 1, \ldots, P$, $j = 1, \ldots, P-1$, analytically, just inverting the matrix A:

$$\begin{bmatrix} \frac{\partial}{\partial\beta_l}\omega_1 \\ \frac{\partial}{\partial\beta_l}\omega_2 \\ \vdots \\ \frac{\partial}{\partial\beta_l}\omega_P \end{bmatrix} = [a_{ij}]^{-1} \cdot \begin{bmatrix} \log|1-\kappa_1\omega_l/1-\kappa_1\omega_P| \\ \log|1-\kappa_2\omega_l/1-\kappa_2\omega_P| \\ \vdots \\ \log|1-\kappa_P\omega_l/1-\kappa_P\omega_P| \end{bmatrix} \tag{3.28}$$

Using Lemma 1 to get the expression of the inverse matrix and doing the corresponding matrix product, we obtain Eq. (3.26). Notice that we obtain the same results when we consider $\kappa_{\min}$ instead of κ_P. □

Theorem 2.

$$\beta_i = \prod_{k=1}^{P-1} \prod_{\substack{l=1 \\ l\neq i}}^{P} (1-\kappa_k \omega_i) \frac{\omega_l}{(\omega_l - \omega_i)}. \tag{3.29}$$

Proof. We consider now the $P-1$ equations resulting from Eq. (3.12) when κ is replaced by $\kappa_i, i = 1, \ldots, P-1$. We write $\beta_P = \beta_P(\beta_j)$, and obtain:

$$\sum_{j=1}^{P-1} \frac{(1-\omega_j/\omega_P)}{(1-\kappa_i\,\omega_j)} \beta_j = 1,\ i = 1, \ldots, P-1, \tag{3.30}$$

These equations can be rewritten in matrix form, $B\boldsymbol{\beta} = \boldsymbol{g}$, which in components reads:

$$[b_{ij}] \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{P-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \tag{3.31}$$

where b_{ij} is defined in Lemma 1. Using Lemma 1 to get the expression of the inverse matrix and doing the corresponding matrix product, we obtain:

$$\begin{aligned} \beta_i &= \omega_P \sum_{j=1}^{P-1} \frac{(1-\kappa_j\,\omega_i)}{(1-\kappa_j\,\omega_P)} \prod_{\substack{k=1 \\ k\neq j}}^{P-1} \prod_{\substack{l=1 \\ l\neq i}}^{P} \frac{(-1+\kappa_k\,\omega_i)(-1+\kappa_j\,\omega_l)}{(\kappa_j-\kappa_k)(\omega_i-\omega_l)} \\ &= \omega_P \left(\prod_{k=1}^{P-1} \prod_{\substack{l=1 \\ l\neq i}}^{P} \frac{(-1+\kappa_k\,\omega_i)}{(\omega_i-\omega_l)} \right) \sum_{j=1}^{P-1} \prod_{\substack{k=1 \\ k\neq j}}^{P-1} \prod_{\substack{l=1 \\ l\neq i}}^{P-1} \frac{(-1+\kappa_j\,\omega_l)}{(\kappa_j-\kappa_k)}. \end{aligned} \tag{3.32}$$

In particular, using the simplification proven at the beginning of Lemma 1 for $i = m$, changing P by $P-1$, and setting $\omega_i = 0$ just formally, we also get that:

$$\sum_{j=1}^{P-1} \left(\prod_{\substack{k=1 \\ k\neq j}}^{P-1} \prod_{\substack{l=1 \\ l\neq i}}^{P-1} \frac{(-1+\kappa_j\,\omega_l)}{(\kappa_j-\kappa_k)} \right) = \prod_{\substack{l=1 \\ l\neq i}}^{P-1} \omega_l. \tag{3.33}$$

Using this expression,

$$\beta_i = \prod_{k=1}^{P-1} \prod_{\substack{l=1 \\ l\neq i}}^{P} \frac{(-1+\kappa_k\,\omega_i)\,\omega_l}{(\omega_i-\omega_l)}. \tag{3.34}$$

□

3.3.5 Advantages of the rewritten system

In the most general case, as we have commented above, we need to solve a system $S(\boldsymbol{\omega}, \boldsymbol{\beta}, \boldsymbol{\kappa}, \frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\beta}})$. With the theorems recently introduced in Sect. 3.3.4, which basically express $\frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\beta}}$ as $\frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\beta}}(\boldsymbol{\omega}, \boldsymbol{\kappa})$ and $\boldsymbol{\beta}$ also as $\boldsymbol{\beta}(\boldsymbol{\omega}, \boldsymbol{\kappa})$, and substituting them into Eq. (3.13) and Eq. (3.14), the non-linear, algebraic-differential system reduces to a purely algebraic system of the form $S(\boldsymbol{\omega}, \boldsymbol{\kappa})$.

We have developed a code that implements everything commented in Sect. 3.3.3 and that automatically constructs the system and solves it for any value of P. This program is written in Mathematica (a pseudocode of which can be found in Alg. 1) and combines both symbolic with numerical calculations.

Data: Γ, $\boldsymbol{N}$
Result: $\boldsymbol{\omega}$, $\boldsymbol{\beta}$

$(\kappa_{\min}, \kappa_{\max}) \leftarrow$ `computeKappas(`$\boldsymbol{N}$`)` `// Eqs.` (3.8), (3.9)

for $j = 1, \ldots, P-1$ **do**
  $\mathrm{d}\Gamma\beta$`[`j`]` $\leftarrow \frac{\partial}{\partial \beta_j}\Gamma$
end
for $i = 1, \ldots, P$ **do**
  `equMax[`i`]` $\leftarrow \Gamma(\kappa_0) == \Gamma(\kappa_i)$ `// Eq.` (3.13)
end
for $i = 0, \ldots, P$ **do**
  for $j = 1, \ldots, P-1$ **do**
    `equDer[`i`]` $\leftarrow \mathrm{d}\Gamma\beta$`[`$j$`](`$\kappa_i$`)` $== 0$ `// Eqs.` (3.14), (3.15)
  end
end
for $i = 1, \ldots, P$ **do**
  for $q = 1, \ldots, P-1$ **do**
    $\mathrm{d}\omega\mathrm{d}\beta$`[`$i$`][`$q$`]` $\leftarrow$ `symbolicAncillaryVariables()` `// Eq.` (3.26)
  end
end
for $i = 1, \ldots, P-1$ **do**
  β`[`i`]` $\leftarrow$ `symbolicBetas()` `// Eq.` (3.29)
end

$(\boldsymbol{\omega}_g, \boldsymbol{\kappa}_g) \leftarrow$ `computeInitialGuesses()` `// Eqs.` (3.17), (3.18), (3.36)

$(\boldsymbol{\omega}, \boldsymbol{\kappa}) \leftarrow$ `findScheme(equMax[],equDer[],`$\mathrm{d}\omega\mathrm{d}\beta$`[],`$\beta$`[],`$\boldsymbol{\omega}_g, \boldsymbol{\kappa}_g$`)`

$\boldsymbol{\beta} \leftarrow$ `computeBetas(`$\boldsymbol{\omega}, \boldsymbol{\kappa}$`,`$\beta$`[])` `// Eq.` (3.29)

return $\boldsymbol{\omega}$, $\boldsymbol{\beta}$

Algorithm 1: Pseudocode for computing optimal schemes.

We obtain two major benefits from the new system of equations to be solved:

1. For any given number of levels P, we reduce by orders of magnitude the computing time in the generation of the symbolic system that we need to solve. Furthermore, the analytic reduction of the system presented above allows us to reduce its dimensionality, by reducing the number of equations and unknowns from $P^2 + P - 1$ to P. The consequence of such reduction is that the number of operations needed to solve the system as well as the number of initial guesses we must provide to begin its iterative solution decreases drastically. Thereby, the numerical stiffness of the original system decreases, since it is fundamentally brought by the difficulty in finding good initial guesses for all the variables involved in its solution. In Sect. 3.3.3, we built $S(\boldsymbol{\omega}, \boldsymbol{\beta})$ and needed a lot of time for the symbolic calculations involved in obtaining, on the one hand, $\frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\beta}}(\boldsymbol{\omega}, \boldsymbol{\beta})$, solving the corresponding linear subsystem (Eq. 3.31) and, on the other hand, $\boldsymbol{\kappa}(\boldsymbol{\omega}, \boldsymbol{\beta})$. With the new methodology, employing Eqs. (3.26) and (3.29), we exchange the role of $\boldsymbol{\beta}$ and $\boldsymbol{\kappa}$, since we have analytic formulae to express the ancillary variables and $\boldsymbol{\beta}$ as functions of $\boldsymbol{\omega}$ and $\boldsymbol{\kappa}$, respectively. With the consequent reduction in the calculation time, the computational time to solve the remaining equations becomes negligible.

2. The solution of the non-linear system to obtain the optimal SRJ parameters needs a suitable methodology to compute their initial values (Sect. 3.3.3). In Sect. 3.3.3, we had to provide initial values for $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$. With the newly derived theorems, we only need to provide initial guesses for $\boldsymbol{\omega}$, since employing Eq. (3.29) $\boldsymbol{\beta} = \boldsymbol{\beta}(\boldsymbol{\omega}, \boldsymbol{\kappa})$, and $\boldsymbol{\kappa}$ satisfies

$$\kappa_i \in \left(\frac{1}{\omega_i}, \frac{1}{\omega_{i+1}} \right). \tag{3.35}$$

From the plots of Γ (see, e.g., Fig. 3.1), we can see that each maximum κ_i is roughly placed at:

$$\kappa_i \approx \frac{1}{\omega_i} + \frac{\frac{1}{\omega_{i+1}} - \frac{1}{\omega_i}}{3}, \tag{3.36}$$

which are the values that we will use as initial guesses.

With these two improvements, we have reduced by four orders of magnitude the total computational time for finding the parameters of an optimal scheme. For example, for $P = 10$, it was necessary to spend a calculation time in Mathematica of the order of one week with the methodology employed by Sect. 3.3.3. In contrast, with the improvements reported here, we can accomplish the same task in tens of seconds. While, in practice, in Sect. 3.3.3 we were

limited (due to the computing time) to SRJ schemes with $P \leq 10$, now we can tackle larger number of levels.

3.4 Results

Now we first calibrate our new method comparing the parameters of our SRJ schemes with those of YM14 for $P \leq 5$ (Sect. 3.4.1). Later (Sect. 3.4.2), we present new optimal schemes computed employing the new methodology sketched previously, up to $P = 15$.

3.4.1 Calibration of the method

To calibrate the new methodology, we have recomputed the optimal parameters for SRJ schemes with $P \leq 5$ and found that our results are the same as those obtained by YM14, when we use the same number of points per dimension N on the same model problem (Eq. 3.1).

Following the ideas of YM14, the performance of any SRJ scheme with respect to the Jacobi method can be quantified estimating the *convergence performance index*, ρ,

$$\rho := \sum_{i=1}^{P} \omega_i \beta_i \tag{3.37}$$

which we have calculated for each SRJ method we have computed, and checked that it approaches its theoretical value when we solve numerically (Eq. 3.1). We point out that the value of ρ depends on the dimensionality of the problem since the value of $\kappa_{\min}$ does (see, Eq. 3.9).

YM14 showed that the optimal parameters computed for coarser grids can be used for finer ones. Nevertheless, minimizing the gaps between different values of N is important because the acceleration of the convergence with respect to the Jacobi method may not be the largest possible unless we compute the optimal SRJ parameters corresponding to a given problem size. Thus, we have completed the tables presented by YM14 minimizing the possible gaps between resolutions. Furthermore, we have computed the optimal SRJ parameters for a number of intermediate values of N in Tab. B.2, where we also show the value of ρ. In order to verify the correct behaviour of the schemes computed, we monitor in Fig. 3.3 the evolution of the difference between two consecutive approximations of the solution for the model problem specified in Eq. (3.1),

$$r_{ij}^{n} = u_{ij}^{n} - u_{ij}^{n-1}, \tag{3.38}$$

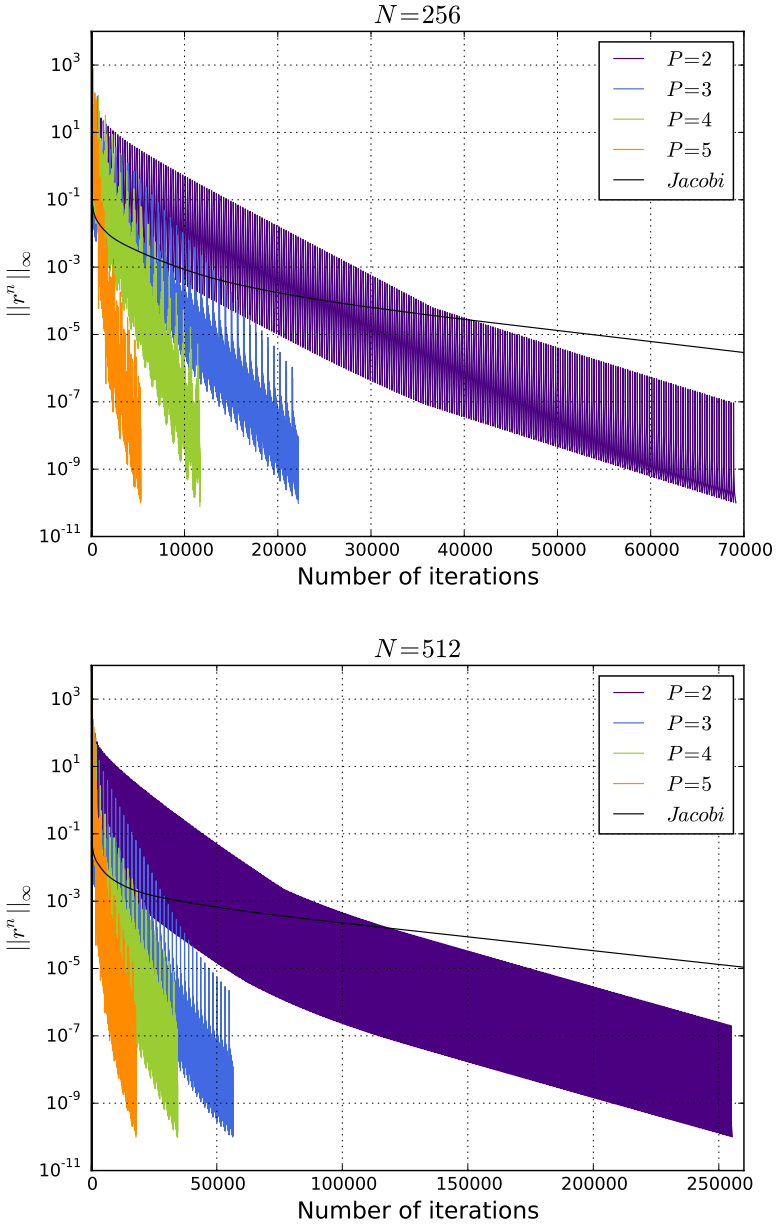


Figure 3.3 Comparison of the evolution of the difference between consecutive approximate solutions ($||r^n||_\infty$) of Eq. (3.3) of SRJ schemes from $P = 2$ to $P = 5$ for a grid with $N = 256$ and with $N = 512$ zones per dimension. We also include the evolution of the residual for the Jacobi method as a reference.

using element-wise norms and operations, that in the bidimensional case would be, for example,

$$||r^n||_\infty = \max_{ij} r_{ij}^n, \tag{3.39}$$

as a function of the number of iterations n for SRJ schemes having all the values of P given by YM14. In the same figure, we also include the residual evolution for the Jacobi method (black line). As expected, the number of iterations to reach the prescribed tolerance decreases[3] as P increases. For all the schemes shown in Fig. 3.3, where the number of points is set to $N = 256$, we have obtained the expected theoretical value of ρ.

3.4.2 New SRJ optimal schemes

After verifying that we recover the optimal parameters computed in YM14, we have improved on their results computing the optimal values of SRJ schemes with $P > 5$. In App. B we provide the Tables B.3 to B.19, corresponding to the optimal parameters for SRJ schemes with $P = 6, \ldots, 13$ and various resolutions for the Laplace problem (Eq. 3.1). In Tables B.21, and B.23, we show the optimal solution parameters for $P = 14$ and $P = 15$.

We encountered that finding optimal parameters at low resolution is increasingly more difficult as the number of levels increases. Indeed, as we can see, for $P = 15$ the minimum value of N we have been able to compute is 64. The reason for the inability of the proposed method to find optimal parameters for low N and large P is that larger values of P imply that the results are extremely sensitive to tiny changes in the smaller wave numbers (i.e., to the values of κ_i close to $\kappa_{\min}$), and small numerical errors prevent a full evaluation of the solution of the non-linear system $\mathcal{S}$, unless the (guessed) initial values are very close to the optimal ones.

We remark that thanks to the improvements done (Sect. 3.3.3) and specially with the analytic solution of a part of the unknowns of the system (Sect. 3.3.4), not only the optimal solution is achievable, but also it is reachable with a moderate computational cost: employing Mathematica on a standard workstation, the computational time of the optimal parameters ranges from tenths of a second for the $P = 6$ scheme to tens of seconds for $P = 15$.

In Fig. 3.4 we show the evolution of the residual for some of the new optimal SRJ schemes solving the model problem used throughout the section. These new schemes show a progressively larger efficiency as P grows. A good proxy for the

[3] If not explicitly mentioned, in all the cases considered here the absolute tolerance is fixed so that $||r^n||_\infty < 10^{-10}$.

performance of the method is the convergence performance index, which grows with the number of levels. We achieve a reduction in the time of computation to solve the problem because of the reduction in the number of iterations to reach convergence. This reduction is roughly proportional to $P\log_{10}(P+1)$. However, the rate of reduction of the error is non monotonic. For large values of P (namely, $P \geq 12$), a direct inspection of Fig. 3.4 shows a faster decline of the residual once any given SRJ method reduces its residual below the one corresponding to the Jacobi method (in this case, this happens after about 4.500 iterations).

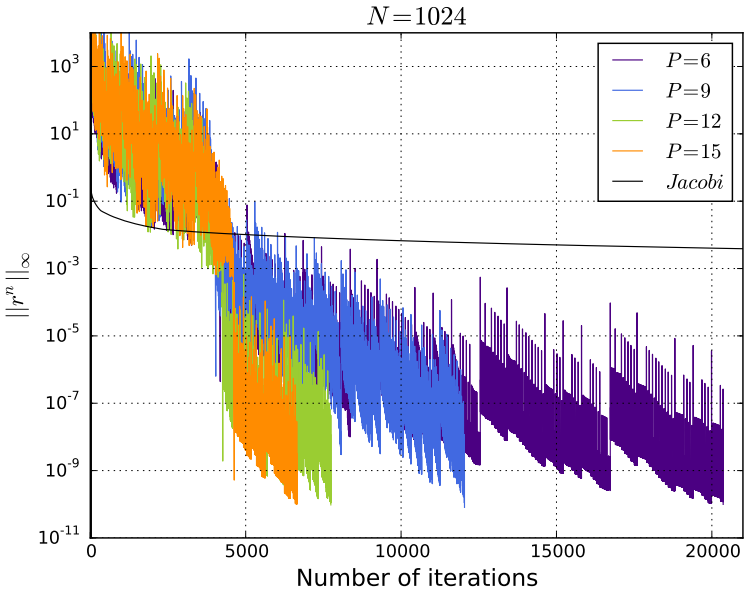


Figure 3.4 Comparison of the residual evolution for the optimal SRJ schemes with $P = 6, 9, 12$ and 15 and $N = 1024$ points per dimension. For reference, we also include the evolution of the residual for the Jacobi method (black line).

3.4.3 Obtention of the integer parameters q

There is a step in the practical implementation of SRJ methods that may impact on the performance of the resulting algorithm, measured by the number of iterations needed to reduce the residual below a prescribed tolerance. Once the solution has been found and we know the *real* values of $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$, one must obtain the *integer* values of $\boldsymbol{q}$. The conversion to integer begins by defining $\bar{\boldsymbol{\beta}} := \frac{\boldsymbol{\beta}}{\beta_1}$, so that $\beta_1 = q_1 = 1$. For the conversion to integer of the rest of

the $\bar{\beta}_i$ $(i = 2, \ldots, P)$, we have tested several possibilities, including the floor $\boldsymbol{q} = \lfloor \bar{\boldsymbol{\beta}} \rfloor$, rounding $\bar{\boldsymbol{\beta}}$ to the nearest integer, taking the ceiling function $\boldsymbol{q} = \lceil \bar{\boldsymbol{\beta}} \rceil$, or combinations of the former alternatives, since it is possible to apply different recipes for every β_i/β_1. Each of these alternatives may yield a different number of iterations to reach convergence (see below). After computing the integer values of $\boldsymbol{q}$, a key point to account for is that the $\Gamma(\kappa)$ function must remain below 1, since otherwise our method diverges. In Fig. 3.5 (upper panel) we observe that the amplification factor per M-cycle may change by more than 10%, for values of κ close to $\kappa_{\max}$, depending on the method adopted to convert $\boldsymbol{\beta}$ to integer.

While the number of levels is small, the differences among the distinct conversions from real to integer do not change much either the number of iterations or the convergence rate of the resulting scheme. However, when P increases, there can be non-negligible changes in the total number of iterations to reduce the residual of our model equation below a prescribed tolerance. In the lower panel of the Fig. 3.5 we show the evolution of the residual as a function of the iteration number for two different choices of the integer conversion of $\boldsymbol{\beta}$ into $\boldsymbol{q}$ in the case $P = 6$ (the optimal parameters of which can be found on Tab. B.4). We note that there is a difference of more than 1200 iterations ($\sim 25\%$) between the distinct integer conversions. Unfortunately, changing the number of levels, the same recipes for converting reals to integers yield efficiencies of the methods that do not display a clear trend. Fortunately, increasing the number of levels by one unit results in a reduction of the number of iterations to reach convergence which is larger than that resulting from any manipulation of the integer values of q_i in an SRJ scheme with a given P. Hence, in the following, the results we will provide are obtained by taking simply $q_i = \lfloor \beta_i/\beta_1 \rfloor$.

3.5 Numerical example: Poisson equation with Dirichlet boundary conditions

So far we have considered only the application of SRJ schemes to the solution of the Laplace equation with homogeneous Neumann boundary conditions (Eq. 3.1). This was also the case in YM14. Here we consider a case study consisting on solving a Poisson equation in two dimensions endowed with Dirichlet boundary conditions. The exact problem setting reads

$$\frac{\partial^2}{\partial x^2} u(x,y) + \frac{\partial^2}{\partial y^2} u(x,y) = -e^{xy}(x^2 + y^2), \qquad (x,y) \in (0,1) \times (0,1),$$

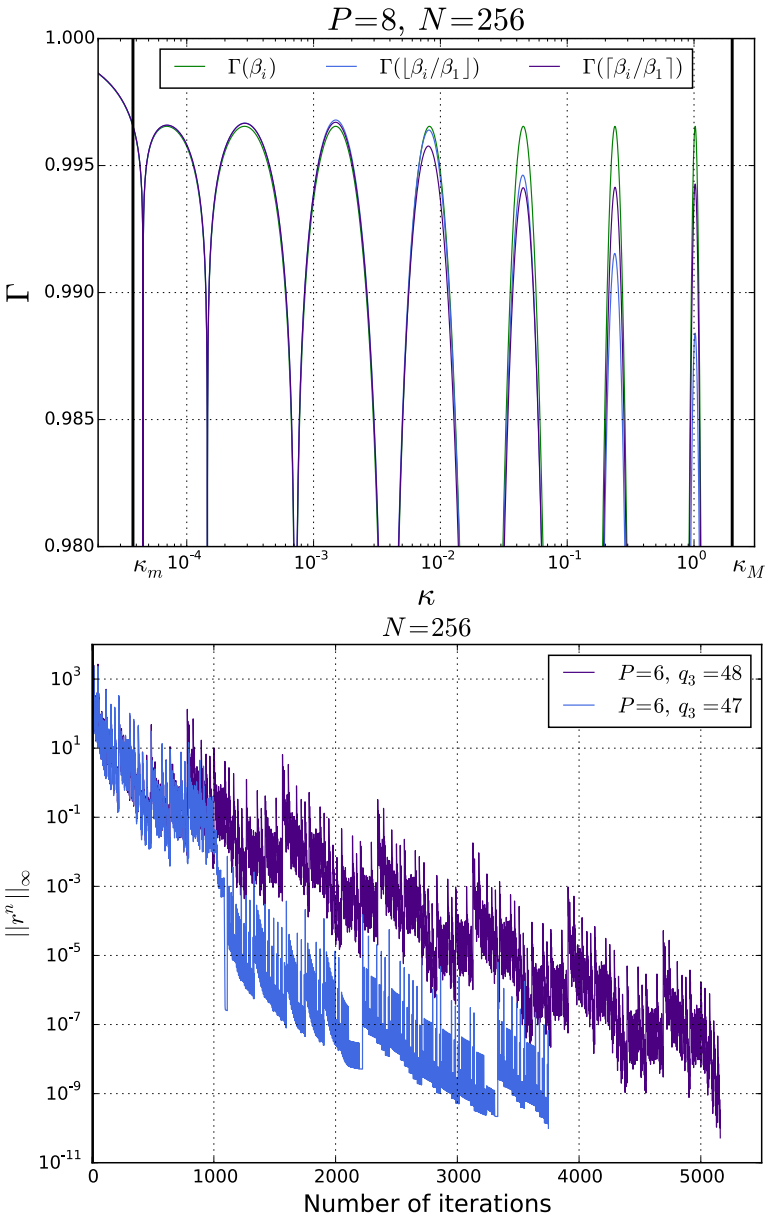


Figure 3.5 Comparison of the evolution of the residual of Eq. (3.3) of two SRJ schemes having $P = 6$ and $N = 256$ zones per dimension. Top: mean amplification factor per M-cycle. Bottom: evolution of the residual as a function of the number of iterations. The different variants of the $P = 6$ SRJ method are displayed with different color lines showing the dependence of the performance of the method on the conversion of the real values of the optimal solution for β_i to the integer values $q_i = \lfloor \beta_i/\beta_1 \rfloor$ (blue) and $q_i = \lceil \beta_i/\beta_1 \rceil$ (violet).

$$\begin{aligned} u(0,y) &= -1,\ u(1,y) = -e^{y}, \qquad y \in [0,1], \\ u(x,0) &= -1,\ u(x,1) = -e^{x}, \qquad x \in [0,1], \end{aligned} \tag{3.40}$$

which has analytic solution:

$$u(x,y) = -e^{xy}. \tag{3.41}$$

This kind of problem will help us to assess whether the change in the boundary conditions affects the efficiency of an SRJ scheme. Imposing Dirichlet boundary conditions is typically less challenging than dealing with Neumann ones, since Dirichlet boundary conditions change the value of $\kappa_{\max}$ so that (YM14)

$$\kappa_{m,\text{Dirichlet}} = \sin^2\left(\frac{\pi}{2N_x}\right) + \sin^2\left(\frac{\pi}{2N_y}\right), \tag{3.42}$$

to be compared with Eq. (3.8). Hence, the optimal SRJ values obtained for a given N and Neumann boundary conditions do not exactly coincide with those optimal in problems involving Dirichlet boundary conditions, hence we must follow the recipe provided in Eq. (3.11). Furthermore, since in Eq. (3.41) we are considering a Poisson equation, we can test whether the presence of source terms modifies the performance of SRJ methods.

For the case studied we choose a discretization consisting on $N_x \times N_y = 585 \times 280$ uniform numerical zones. Although here we do not have Neumann boundary conditions, we will use the optimal values of the SRJ scheme for this case in order to show that even though this choice is not optimal, still it substantially speeds up the solution of the problem with respect to Jacobi. In this case, we have that $N = 585$. If we apply the SRJ scheme with $P = 10$, we must look for the $\boldsymbol{\omega}$ and the $\boldsymbol{\beta}$ parameters in Tab. B.12. In this case, the table does not provide an entry for $N = 585$, but as YM14 point out, we can chose as (non-optimal) parameters for the SRJ scheme those corresponding to a smaller resolution.[4] In our case, the closest resolution that matches this criterion in Tab. B.12 is that corresponding to the row with $N = 550$.

We use the simplest way to obtain the q_i from the β_i ensuring convergence, namely $q_i = \left\lfloor \frac{\beta_i}{\beta_1} \right\rfloor$ with $1 \leq i \leq P$, resulting in

$$\boldsymbol{q} = \{1, 1, 3, 9, 21, 49, 116, 268, 587, 1014\},\ M = 2069. \tag{3.43}$$

This means that we will use $\omega_1 = 106105$ and $\omega_2 = 40577.2$ once per M-cycle, $\omega_3 = 10230.6$ three times per M-cycle, etc. In practice, it is necessary to distribute the largest over-relaxation steps over the whole M-cycle to prevent the occurrence of overflows. YM14 provide a Matlab script that generates a

[4] It is, however, possible to compute the optimal values for $N = 585$ employing our algorithm.

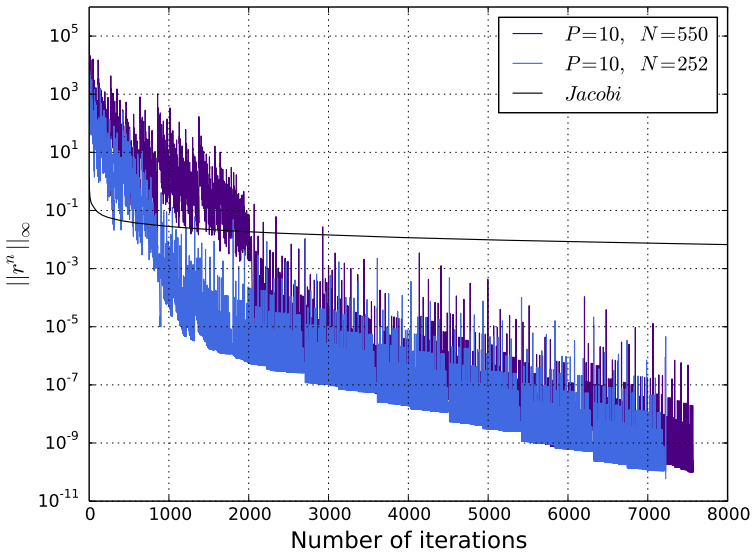


Figure 3.6 Evolution of the residual as a function of the iteration number for different SRJ schemes. With violet line we show the case in which the parameters of an SRJ scheme with $P = 10$, $N = 550$ and Neumann boundary conditions are chosen to compute the solution of the problem stated in Eq. (3.41), on a grid of $N_x \times N_y = 585 \times 280$ and *Dirichlet* boundary conditions. With blue line we show the case in which the parameters of an SRJ scheme with $P = 10$, $N = 252$ and Neumann boundary conditions are used. The latter case corresponds to the closest value of N to the effective resolution $N_{\text{eff}}^{(2)} = 252.56$ that shall be used when Dirichlet boundary conditions (instead of Neumann ones) are used. For comparison, we also display the evolution of the residual for the Jacobi method.

schedule for the distribution of ω_i on the M-cycle that guarantees the absence of overflows. We find that an even distribution of the over-relaxations over the entire M-cycle is sufficient in order to avoid overflows.

In Fig. 3.6 we plot the evolution of the residual as a function of the number of iterations for a SRJ scheme with $P = 10, N = 550$ (instead of $N = 585$), as well as the residual evolution employing the Jacobi method for the solution of Eq. (3.41). This example shows that even picking an SRJ scheme whose parameters are non-optimal for the problem size at hand ($N = 585$ in this case), for the presence of source terms and for the kind of boundary conditions specified (Dirichlet for the problem at hand), we can largely speed up the convergence with respect to the Jacobi method. Theoretically, for the optimal $P = 10$, $N = 550$ SRJ method, an acceleration of the order of $\rho = 125.85$ with respect to the Jacobi method is expected, something confirmed with our numerical results (Fig. 3.6).

Finally, we briefly illustrate how to find the optimal scheme for Dirichlet boundary conditions (which are, indeed, the ones with which the problem at hand is set). According to Eq. (3.11), $N_{\text{eff}}^{(2)} = 252.56$. Therefore, we compute the optimal parameters for the SRJ scheme with $N = 252$, obtaining:

$$\boldsymbol{\omega} = \{22912.5, 10310.7, 3128.79, 836.313, 216.061, 55.4614, \\ 14.3482, 3.85729, 1.19337, 0.556912\} \tag{3.44}$$

$$\mathbf{q} = \{1, 1, 3, 6, 13, 29, 62, 131, 256, 401\}. \tag{3.45}$$

In Fig. 3.6 we can see that the results are slightly better than for the (non-optimal) set of parameters with $P = 10$, $N = 550$ and Neumann boundary conditions, since in the case with $N = 252$, the accuracy goal is reached with $\sim 5\%$ less iterations. However, this small deviation (motivated by the choice of non-optimal parameters) can be overcome in, at least two ways when selecting a non-optimal scheme for the problem at hand. First, by noting that the difference in the number of iterations can be reduced by employing different methods for the conversion from real to integer of the values of $\mathbf{q}$ (see Sect. 3.4.3). Second, by increasing the number of levels of the SRJ scheme. This is another reason to add to the relevance of the work presented here, since increasing P tends to yield schemes that converge in less iterations (basically at no extra computational cost).

Chapter 4

On the equivalence between the Scheduled Relaxation Jacobi method and Richardson's non-stationary method

The results of this chapter have been originally published in:

On the equivalence between the Scheduled Relaxation Jacobi method and Richardson's non-stationary method,
JE Adsuara, I Cordero-Carrión, P Cerdá-Durán, V Mewes, MA Aloy,
Journal of Computational Physics 332, 446-460 (2017)

The text in the following sections corresponds to an edited version of the aforementioned publication.

4.1 Introduction

The SRJ method can be expressed for a linear system $Au = b$ as

$$u^{n+1} = u^n + \omega_n D^{-1}(b - Au^n), \tag{4.1}$$

where D is the diagonal of the matrix A. If we consider a set of P different relaxation factors, $\omega_n,\ n = 1, \ldots, P$, such that $\omega_n > \omega_{n+1}$ and we apply each

relaxation factor q_n times, the *total amplification factor* after $M := \sum_{n=1}^{P} q_n$ iterations is

$$G_M(\kappa) = \prod_{n=1}^{P} (1 - \omega_n \kappa)^{q_n}, \tag{4.2}$$

which is an estimation of the reduction of the residual during one cycle (M iterations). As we have shown in the previous chapter, in the former expression κ is a function of the wave-numbers obtained from a von Neumann analysis of the system of linear equations resulting from the discretization of the original elliptical problem by finite differences (for more details see [Yang and Mittal 2014]). Yang & Mittal [Yang and Mittal 2014] argued that, for a fixed number P of different weights, there is an optimal choice of the weights ω_n and repetition numbers q_n that minimizes the maximum *per-iteration amplification factor*, $\Gamma_M(\kappa) = |G_M(\kappa)|^{1/M}$, in the interval $\kappa \in [\kappa_{\min}, \kappa_{\max}]$ and therefore also the number of iterations needed for convergence. The boundaries of the interval in κ correspond to the minimum and the maximum weight numbers allowed by the discretization mesh and boundary conditions used to solve the elliptic problem under consideration.

In the aforementioned paper, [Yang and Mittal 2014] computed numerically the optimal weights for $P \leq 5$ and we have extended the calculations up to $P = 15$ (see previous chapter and ACCA15). The main properties of the SRJ, obtained by [Yang and Mittal 2014] and confirmed by us, are the following:

1. Within the range of P studied, increasing the number of weights P improves the rate of convergence.

2. The resulting SRJ schemes converge significantly faster than the classical Jacobi method by factors exceeding 100 in the methods presented by [Yang and Mittal 2014] and ~ 1000 in those presented in Sect. 3.3.4. Increasing grid sizes, i.e. decreasing $\kappa_{\min}$, results in larger acceleration factors.

3. The optimal schemes found use each of the weights multiple times, resulting in a total number of iterations M per cycle significantly larger than P, e.g. for $P = 2$, [Yang and Mittal 2014] found an optimal scheme with $M = 16$ for the smallest grid size they considered ($N = 16$), while for larger grids M notably increases (e.g., $M = 1173$ for $N = 1024$).

The optimization procedure outlined by [Yang and Mittal 2014] has a caveat though. Even if the amplification factor were to reduce monotonically by increasing P, for sufficiently high values of P, the number of iterations per cycle M may be comparable to the total number of iterations needed to solve

a particular problem for a prescribed tolerance. At this point, using a method with higher P, and thus higher M, would increase the number of iterations to converge, even if the $\Gamma(\kappa)$ is nominally smaller. With this limitation in mind we outline a procedure to obtain optimal SRJ schemes, minimizing the total number of iterations needed to reduce the residual by an amount sufficient to reach convergence or, equivalently, to minimize $|G_M(\kappa)|$. Note that the total number of iterations can be chosen to be equal to M without loss of generality, i.e. one cycle of M iterations is needed to reach convergence. To follow this procedure one should find the optimal scheme for fixed values of M, and then choose M such that the maximum value of $|G_M(\kappa)|$ is similar to the residual reduction needed to solve a particular problem. The first step, the minimization problem, is in general difficult to solve, since fixing M gives an enormous freedom in the choice of the number of weights P, which can range from 1 to M. However, the numerical results of [Yang and Mittal 2014] and ours presented in Sect. 3.3.4, seem to suggest that in general increasing the number of weights P will always lead to better convergence rates. This leads us to conjecture that the optimal SRJ scheme, for fixed M, is the one with $P = M$, i.e. all weights are different and each weight is used once per cycle, $q_i = 1$, $(i = 1, \ldots, M)$. In terms of the total amplification factor $G_M(\kappa)$, it is quite reasonable to think that if one maximizes the number of different roots by choosing $P = M$, the resulting function is, on average, closer to zero than in methods with smaller number of roots, $P < M$, and one might therefore expect smaller maxima for the optimal set of coefficients. One of the aims of this work is to compute the optimal coefficients for this particular case and demonstrate that $P = M$ is indeed the optimal case.

Another goal of this section is to show the performance of optimal SRJ methods compared with optimal SOR algorithms applied to a number of different discretizations of the Laplacian operator in two-dimensional (2D) applications (Sect. 4.3). We will show that optimal SRJ methods applied to high-order discretizations of the Laplacian, which yield iteration matrices that cannot be consistently ordered, perform very similarly to optimal SOR schemes (when an optimal SOR weight can be computed). We will further discuss that the trivial parallelization of the SRJ methods outbalances the slightly better scalar performance of SOR in some cases (Sect. 4.4). Also, we will show that the optimal weight of the SOR method can be suitably approximated by functions related to the geometric mean of the set of weights obtained for optimal SRJ schemes. This is of particular relevance when the iteration matrix is non-consistently ordered

and hence, the analytic calculation of the optimal SOR weight is extremely intricate.

4.2 Optimal $P = M$ SRJ scheme

Let us consider a SRJ method with $P = M$ and hence $q_n = 1$, $(n = 1, \ldots, M)$. For this particular choice, the amplification factor $G_M(\kappa)$ is a polynomial of degree M in κ with M different roots. In this case, the set of weights ω_n that minimizes the value of the maximum of $|G_M(\kappa)|$, given by Eq. (4.2), in the interval $\kappa \in [\kappa_{\min}, \kappa_{\max}]$, $0 < \kappa_{\min} \leq \kappa_{\max}$ [1], can be determined by the definition of the amplification factor

$$G_M(0) = 1, \tag{4.3}$$

and by the following M conditions[2]:

$$G_M(\kappa_n) = -G_M(\kappa_{n+1}), \quad n = 0, \ldots, M-1, \tag{4.4}$$

where $\kappa_0 = \kappa_{\min}$, $\kappa_M = \kappa_{\max}$, and κ_n, $n = 1, \ldots, M-1$ are the relative extrema of the function $G_M(\kappa)$. To simplify further we rescale κ as follows:

$$\tilde{\kappa} = 2\frac{\kappa - \kappa_{\min}}{\kappa_{\max} - \kappa_{\min}} - 1. \tag{4.5}$$

As a function of $\tilde{\kappa}$ the amplification factor is $\tilde{G}_M(\tilde{\kappa}) = G_M(\kappa(\tilde{\kappa}))$. In the resulting interval, $\tilde{\kappa} \in [-1, 1]$, there is a unique polynomial of degree M such that the absolute value of $\tilde{G}_M(\tilde{\kappa})$ at the extrema $\tilde{\kappa}_i$ is the same (fulfilling Eqs. (4.4)) and such that $\tilde{G}_M(\tilde{\kappa}(0)) = 1$. This polynomial is proportional to the Chebyshev polynomial of first kind of degree M, $T_M(\kappa)$, which can be defined through the identity $T_M(\cos\theta) = \cos(M\,\theta)$. This polynomial satisfies that

$$|T_M(-1)| = |T_M(\tilde{\kappa}_n)| = |T_M(+1)| = 1, \quad n = 1, \ldots, M-1, \tag{4.6}$$

with $\tilde{\kappa}_i$ being the local extrema of $T_M(\tilde{\kappa})$ in $[-1, 1]$. The constant of proportionality can be determined from Eq. (4.3), and the amplification factor reads in this case

$$\tilde{G}_M(\tilde{\kappa}) = \frac{T_M(\tilde{\kappa})}{T_M(\tilde{\kappa}(0))} \quad ; \quad \tilde{\kappa}(0) = -\frac{(1 + \kappa_{\min}/\kappa_{\max})}{(1 - \kappa_{\min}/\kappa_{\max})} < -1. \tag{4.7}$$

[1] In this work, $\kappa_{\min}$ and $\kappa_{\max}$ are assumed to be strictly positive as the discretization of an elliptic problem leads to a matrix A that is positive definite. In problems where it is not, a simple option is to work with the matrix $A^T A$ and the equivalent system $A^T A u = A^T b$.

[2] These conditions result from the solution of a global min-max optimization problem over $G_M(\kappa)$ or, equivalently, over $\Gamma_M(\kappa)$ (see Appendix B of [Yang and Mittal 2014]).

This result is equivalent to Markoff's theorem[3]. Note that the value of $\tilde{\kappa}(0)$ does not depend on the actual values of $\kappa_{\min}$ and $\kappa_{\max}$, but only on the ratio $\kappa_{\min}/\kappa_{\max}$. The roots and local extrema of the polynomial $T_M(\tilde{\kappa})$ are located, respectively, at

$$\tilde{\omega}_n^{-1} = -\cos\left(\pi\frac{2n-1}{2M}\right), \quad n = 1, \ldots, M, \tag{4.8}$$

$$\tilde{\kappa}_n = \cos\left(\pi\frac{n}{M}\right), \quad n = 1, \ldots, M-1, \tag{4.9}$$

which coincide with those of $\tilde{G}_M(\tilde{\kappa})$. Therefore, the set of weights

$$\omega_n = 2\left[\kappa_{\max} + \kappa_{\min} - (\kappa_{\max} - \kappa_{\min})\cos\left(\pi\frac{2n-1}{2M}\right)\right]^{-1}, \quad n = 1, \ldots, M, \tag{4.10}$$

corresponds to the optimal SRJ method for $P = M$.

We have found with the simple analysis done that the optimal SRJ scheme when $P = M$ is fixed turns out to be closely related to a Chebyshev iteration or Chebyshev semi-iteration for the solution of systems of linear equations (see, for instance, [Gutknecht and Rallin 2002] for a review). This is especially easy to realize if we consider the original formulation of this kind of methods, which appeared in the literature as special implementations of the non-stationary or semi-iterative Richardson's method (RM, hereafter; see, e.g., [Young 1953, Frank 1960] for generic systems of linear equations, or [Shortley 1953] for the application to boundary-value problems). [Yang and Mittal 2014] argued that, for a uniform grid, Eq. D.2.4 is identical to that of the RM [Richardson 1911]. There is, nevertheless, a minor difference between Eq. D.2.4 of the SRJ method and the RM as it has been traditionally written [Young 1954b], that using our notation would be $u^{n+1} = u^n + \hat{\omega}_n(b - Au^n)$, which gives the obvious relation $\hat{\omega}_n = \omega_n d^{-1}$, in the case in which all elements in D are the same and equal to d. We note that this difference disappears in more modern formulations of the RM (e.g., [Opfer and Schober 1984]), in which the RM is also written as a fixed point iteration of the form $u^{n+1} = Tu^n + c$, with $T = I - B^{-1}A$, $c = B^{-1}b$ and B any non-singular matrix. Differently from the RM in its definition by Young [Young 1954b], our method in the case $M = 1$ would fall in the category of stationary Generalized Richardson's (GRF) methods according to the textbook of Young [Young 1971, chap. 3]. GRF methods are defined by the updating formula

$$u^{n+1} = u^n + \mathcal{P}(Au^n - b) \tag{4.11}$$

[3]For an accessible proof of the original theorem [Markoff 1916] , see Young's textbook [Young 1971], Theorem 9-3.1.

where $\mathcal{P}$ is any non singular matrix (in our case, $\mathcal{P} = -\omega_n D^{-1}$). In the original work of Richardson [Richardson 1911], all the values of $\hat{\omega}_n^{-1}$ were set either equal or evenly distributed in $[a, b]$, where a and b are, respectively, lower and upper bounds to the minimum and maximum eigenvalues, λ_i, of the matrix A (optimally, $a = \min(\lambda_i)$, $b = \max(\lambda_i)$). If a single weight is used throughout the iteration procedure, a convenient choice is $\hat{\omega} = 2/(b + a)$.[4]

Yang & Mittal [Yang and Mittal 2014] state that the SRJ approach to maximizing convergence is fundamentally different from that of the stationary RM. They argue that the RM aims to reduce $\Gamma(\kappa)$ uniformly over the range $[\kappa_{\min}, \kappa_{\max}]$ by generating equally spaced nodes of Γ in this interval, while SRJ methods set a min-max problem whose goal is to minimize $|\Gamma|_{\max}$.[5] As a result, SRJ methods require computing a set of weights yielding two differences with respect to the non-stationary RM in its original formulation [Yang and Mittal 2014]:

1. the nodes in the SRJ method are not evenly distributed in the range $[\kappa_{\min}, \kappa_{\max}]$;

2. optimal SRJ schemes naturally have many repetitions of the same relaxation factor whereas RM generated distinct values of $\hat{\omega}_n$ in each iteration of a cycle.

From these two main differences, [Yang and Mittal 2014] conclude that while optimal SRJ schemes actually gain in convergence rate over the Jacobi method as grids get larger, the convergence rate gain for Richardson's procedure (in its original formulation) never produces acceleration factors larger than 5 with respect to the Jacobi method. This result was supported by Young in his Ph.D. thesis [Young 1950, p. 4], but on the basis of employing orderings of the weights which did pile-up roundoff errors, preventing a faster method convergence (see point 2 below).

The difference outlined in point 1 above is non existent for GRF methods, where the eigenvalues of A are not necessarily evenly distributed in the spectral range of matrix A (i.e., in the interval $[a, b]$). We note that Young [Young 1953] attempted to chose the $\hat{\omega}_n$ parameters of the RM to be the reciprocals of the

[4] In the case of SRJ schemes with $P = M$, it is easy to demonstrate (see A.2) that the harmonic mean of the weights ω_n very approximately equals the value of the inverse weight of the stationary RM ($2d^{-1}/(\kappa_{\max} + \kappa_{\min}) \simeq 2/(b + a)$).

[5] We note that this argument does not hold in the implementation of the non-stationary RM method made by Young [Young 1953], since in this case one also attempts to minimize $|\Gamma|_{\max}$.

roots of the corresponding Chebyshev polynomials in $[a, b]$, which resulted in a method that is *almost the same* as ours, but with two differences:

First, we do not need to compute the maximum and minimum eigenvalues of the matrix A; instead, we compute $\kappa_{\max}$ and $\kappa_{\min}$, which are related to the maximum and minimum frequencies that can be developed on the grid of choice employing an straightforward von Neumann analysis. Indeed, this procedure to estimate the maximum and minimum frequencies for the elliptic operators (e.g., the Laplacian) in the continuum limit allows applying it to matrices that are not necessarily consistently ordered, like, e.g., the ones resulting from the 9-point discretization of the Laplacian [Adams, LeVeque, and Young 1988]. In Sect. 4.4 we show how our method can be straightforwardly prescribed in this case and other more involved (high-order) discretizations of the Laplacian.

Second, in Young's method [Young 1953] the two-term recurrence relation given to solve $Au = b$ turned out to be unstable. Young found that the reason for the instability was the build up of roundoff errors in the evaluation of the amplification factor (Eq. 4.2), which resulted as a consequence of the fact that many of the values of ω_n can be much larger than one. Somewhat unsuccessfully, Young [Young 1953] tried different orderings of the sequence of weights ω_n, and concluded that, though they ameliorated the problem for small values of M, did not cure it when M was sufficiently large. Later, Young [Young 1954b, Young 1956] examines a number of orderings and concluded that some gave better results than others. However, the key problem of existence of orderings for which RM defines a stable numerical algorithm amenable to a practical implementation was not shown until the work of Anderssen & Golub [Anderssen and Golub 1972]. These authors showed that employing the ordering developed by Lebedev & Finogenov [Lebedev and Finogenov 1971] for the iteration parameters in the Chebyshev cyclic iteration method, the RM devised by Young [Young 1953] was stable against the pile-up of round-off errors. However, Anderssen & Golub [Anderssen and Golub 1972] left open the question of whether other orderings are possible. In our case, numerical stability is brought about by the ordering of the weights in the iteration procedure. This ordering is directly inherited from the SRJ schemes of [Yang and Mittal 2014], and notably differs from the prescriptions given for two- or three-term iteration relations in Chebyshev semi-iterations [Gutknecht and Rallin 2002] and from those suggested by [Young 1953]. Indeed, the ordering we use differs from that of [Lebedev and Finogenov 1971, Nikolaev and Samarskii 1972, Lebedev and Finogenov 2002] (see A.1). Thus, though we do not have a theoretical proof for it, we empirically confirm that other alternative orderings work.

Taking advantage of the analysis made by [Young 1953], we point out that the average rate of convergence of the method in a cycle of M iterations is

$$R_M = \frac{1}{M} \log |T_M(\tilde{\kappa}(0))|, \tag{4.12}$$

and it is trivial to prove that for $\kappa \in [\kappa_{\min}, \kappa_{\max}]$

$$G_M(\kappa) \leq \left| \frac{1}{T_M(\tilde{\kappa}(0))} \right| < 1, \tag{4.13}$$

providing a simple way to compute an upper bound for the amplification factor for the optimal scheme. This condition also guarantees the convergence of the optimal SRJ method. Therefore, if we aim to reduce the initial residual of the method by a factor σ, we have to select a sufficiently large M such that

$$\sigma \geq |T_M(\tilde{\kappa}(0))|^{-1} \tag{4.14}$$

It only remains to demonstrate that the optimal SRJ scheme with $P = M$ is also the optimal SRJ scheme for any $P \leq M$. Markoff's theorem states that for any polynomial $Q(x)$ of degree smaller or equal to M, such that $\exists x_0 \in \mathbb{R}, x_0 < -1$, with $Q(x_0) = 1$, and $Q(x) \neq T_M(x)/T_M(x_0)$, then

$$\max |Q(x)| > \max \left| \frac{T_M(x)}{T_M(x_0)} \right| \quad \forall x \in [-1, 1]. \tag{4.15}$$

This theorem implies that any other polynomial of order $P \leq M$, different from Eq. (4.7), is a poorer choice as amplification factor. The first implication is that $G_M(\tilde{\kappa}(0)) < G_{M-1}(\tilde{\kappa}(0))$, i.e., increasing M decreases monotonically the amplification factor $G_M(\kappa)$. As a consequence, the per iteration amplification factor $\Gamma_M(\kappa)$ also decreases by increasing M. The second consequence is that the case $P < M$ results in an amplification factor with larger extrema than the optimal $P = M$ case, and hence proves that our numerical scheme leads to the optimal set of weights for any SRJ method with M steps. This confirms our intuition that adding additional roots to the polynomial would decrease the value of its maxima, resulting in faster numerical methods. Though the SRJ algorithm with $P = M$ we have presented here turns out to be nearly equivalent to the non-stationary RM of Young [Young 1953], in order to single it out as the optimum among the SRJ schemes, we will refer to it as the *Chebyshev-Jacobi* method (CJM) henceforth.

Finally, we plot in Fig. 4.1 the per-iteration amplification factor, $\Gamma_M(\kappa)$, for different values of M. It is evident from the plot that all the maxima are of equal height, and that the maxima decrease as M increases.

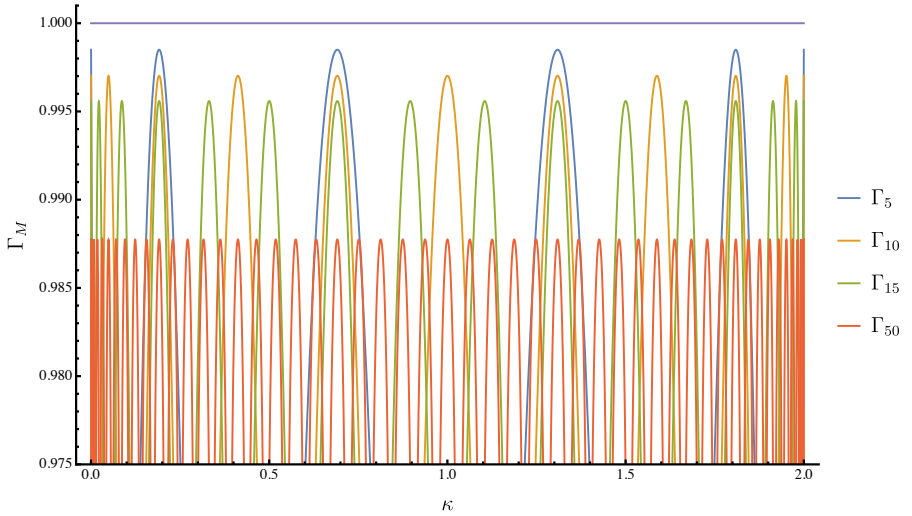


Figure 4.1 Plot $\Gamma_M(\kappa)$ for the following different values of M (5, 10, 15, and 50) for a bidimensional mesh of 128×128 points. One can see that all the extrema are equal. The plot also shows that the higher the value of M, the lower the local maxima of Γ_M.

4.3 Numerical examples: Laplace equation

In order to assess the performance of the new optimal set of schemes devised, we resort to the same prototype numerical example considered in [Yang and Mittal 2014], namely, the solution of the Laplace equation with homogeneous Neumann boundary conditions in two spatial dimensions, in Cartesian coordinates and over a domain with unitary size:

$$\begin{cases} \dfrac{\partial^2}{\partial x^2}u(x,y) + \dfrac{\partial^2}{\partial y^2}u(x,y) = 0, & (x,y) \in (0,1)\times(0,1) \\ \dfrac{\partial}{\partial x}u(x,y)\Big|_{x=0} = \dfrac{\partial}{\partial x}u(x,y)\Big|_{x=1} = 0, & y \in (0,1) \\ \dfrac{\partial}{\partial y}u(x,y)\Big|_{y=0} = \dfrac{\partial}{\partial y}u(x,y)\Big|_{y=1} = 0, & x \in (0,1). \end{cases} \tag{4.16}$$

We consider a spatial discretization of the Laplacian operator employing a second-order, 5-point formula

$$\Delta u_{ij} = \frac{1}{h^2}\Big[u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}\Big], \tag{4.17}$$

where we are assuming that the grid spacing, h, is the same along the x and y directions. In all examples presented in this work, we will use initial random data to initialize our computations. To compare the performance of different

numerical schemes we monitor the evolution of the difference between two consecutive approximations of the solution for the model problem specified in Eq. (4.16),

$$||r^n||_\infty = \max_{ij} |u_{ij}^n - u_{ij}^{n-1}|, \tag{4.18}$$

where u_{ij}^n is the numerical approximation computed after n iterations at the grid point (x_i, y_j).

In Fig. 4.2 (top), we compare the evolution of the residual as a function of the number of iterations for several SRJ schemes, as well as for the new schemes developed here. The violet line corresponds to the best SRJ scheme presented in [Yang and Mittal 2014] for the solution of the problem set above and a spatial grid of $N_x \times N_y = 256 \times 256$ uniform zones, i.e. the SRJ scheme with $P = 5$ and $M = 780$. Comparing with the new CJM for $P = M = 780$ (orange line in Fig. 4.2 top), it is evident that the new scheme reduces the number of iterations to reach the prescribed tolerance ($||r^n||_\infty \leq 10^{-10}$ in this example) by about a factor of 5. We also include in Fig. 4.2 (top; green line) the residual evolution corresponding to the best SRJ optimal algorithm developed by [Adsuara et al. 2015] for the proposed resolution, namely, the scheme with $P = 15$ levels and $M = 1160$. It is obvious that even the CJM with $P = M = 780$ reduces the residual faster than the $P = 15$ SRJ scheme. However, since the $P = 15$ SRJ scheme requires a larger value of M than in the case of $P = 5$, for a fair comparison, we also include in Fig. 4.2 (top; blue line) the CJM with $P = M = 1160$. The latter is the best performing scheme, though the difference between the two new CJM with different values of P is very small (in Fig. 4.2 the blue and orange lines practically overlap).

A positive property of the new algorithm presented in Sect. 4.2 is its predictability, i.e., the easiness to estimate the size of the M-cycle in order to reduce the tolerance by a prescribed amount (Eq. 4.14). Indeed, it is not necessary to monitor the evolution of the residual in every iteration (as in many other non-stationary methods akin to the Richardson's method -e.g., in the gradient method-), with the obvious reduction in computational load per iteration that this implies. In Fig. 4.2 (bottom) we show that our algorithm performs as expected, reducing the initial residual by factors of larger than 10^6, 10^8 and 10^{10} in a single cycle consisting of $P = 1939$, 2470 and 3000 iterations, respectively, since for the problem at hand we have $\kappa_{\min} = \sin\left(\frac{\pi}{2\times 256}\right)^2 = 3.76491 \times 10^{-5}$, $\kappa_{\max} = 2$, and thus, $\tilde{\kappa}(0) = -1.00004$. We note that in Fig. 4.2 (bottom) we have normalized the residual (Eq. 4.18 to its value on the first iteration. This allows for an easier comparison of different methods.

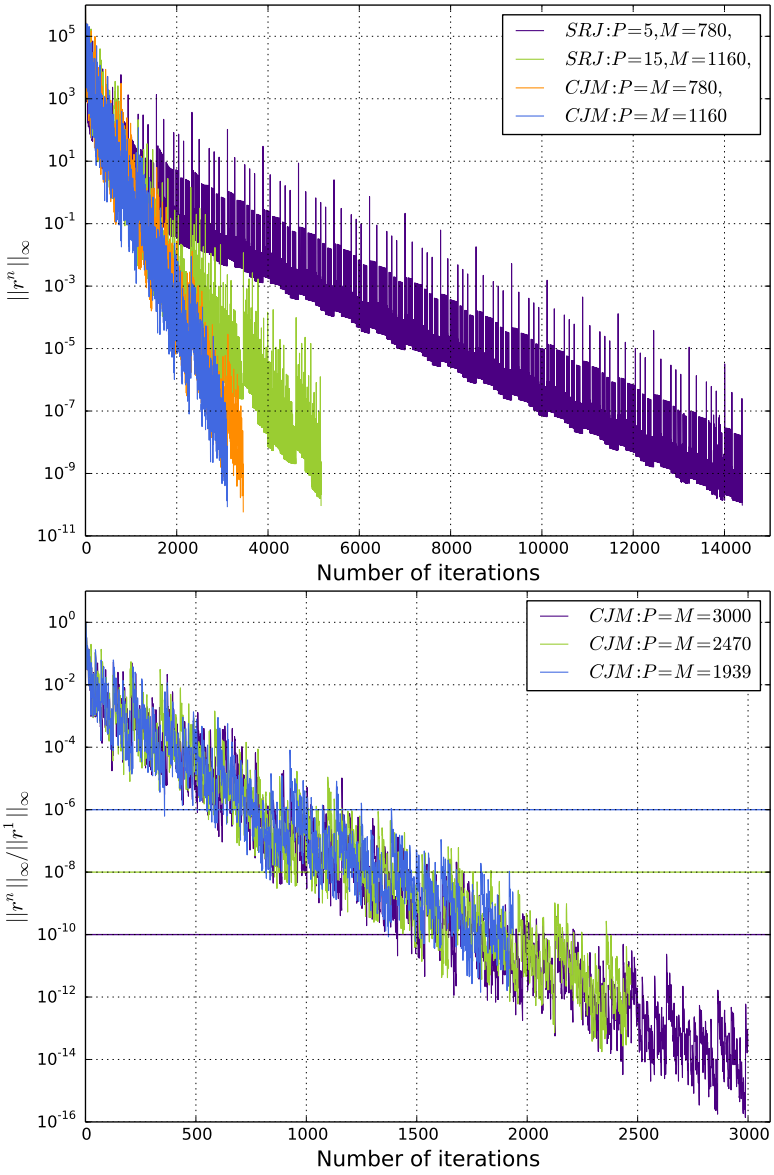


Figure 4.2 Top: Evolution of the residual $||r^n||_\infty$, defined in Eq. 4.18, as a function of the number of iterations for the problem set in Eq. 4.16 and a Cartesian grid of 256×256 uniform zones. The different color lines correspond to different schemes (see legends). We can observe that the reduction of the residual is faster in the new Chebyshev-Jacobi schemes than in the corresponding SRJ schemes with the same value of M. Bottom: We show three examples where we computed the optimal value of the M for reaching the desired residual in one cycle. The cases $P = 1939$, 2470 and 3000 correspond to schemes that (theoretically) should reduce the initial residual by factors $\simeq 10^6$, 10^8 and 10^{10}.

In this simple example the upper bound for the residual obtained from Eq. (4.14) is very rough and clearly overestimates the number of iterations to reduce the residual below the prescribed values. In more complex problems this will not necessarily be the case as we will show in the following, more demanding example.

4.4 CJM for non-consistently ordered matrices: high-order discretization of the Laplacian operator in 2D with 9 and 17 points

Here we consider the case of non-consistently ordered (NCO) matrices. For this class of matrices Young's theory does not apply. This makes it difficult to apply standard methods such as SOR, because the relation between optimal parameter and the spectral radius of the Jacobi iteration matrix is unknown. Here, we will investigate two of these cases, namely a 9-point and 17-point discretization of the Laplacian in 2D.

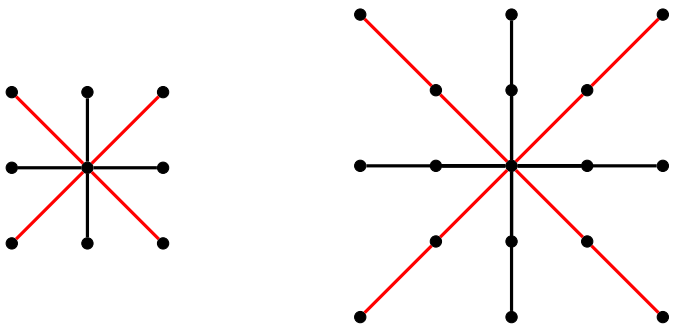


Figure 4.3 Schematic representation of the 9- and 17-point stencils. The black and red lines correspond to the standard stencil S_+ and rotated stencil $S_\times$, respectively. See main text for details.

One way of obtaining this type of discretizations is doing a convex combination between the discretization of the Laplacian operator using the standard stencil, S_+, with its discretization in a rotated stencil, $S_\times$, being $\alpha \geq 1/2$ (see Fig. 4.3):

$$\alpha S_+ + (1-\alpha) S_\times \,. \tag{4.19}$$

Writing α as a rational number a/b, the resulting 9-points discretized Laplacian is

$$\Delta u_{ij} = \frac{1}{2bh^2}\Big[2au_{i-1,j} + 2au_{i+1,j} + 2au_{i,j-1} + 2au_{i,j+1} \\ +(b-a)u_{i-1,j-1} + (b-a)u_{i+1,j+1} + (b-a)u_{i-1,j+1} + (b-a)u_{i+1,j-1} \\ -4(a+b)u_{i,j}\Big], \tag{4.20}$$

where, for simplicity, we assume that the grid spacing, h, is the same in the $x-$ and $y-$directions. From this general form, we can recover the standard 5-points discretization simply taking $a = b = 1$. In the same way, we can recover the 9-points discretization of the Laplacian studied in [Adams, LeVeque, and Young 1988] by imposing $a = 2$ and $b = 3$:

$$\Delta u_{ij} = \frac{1}{6h^2}\Big[4u_{i-1,j} + 4u_{i+1,j} + 4u_{i,j-1} + 4u_{i,j+1} \\ +u_{i-1,j-1} + u_{i+1,j+1} + u_{i-1,j+1} + u_{i+1,j-1} - 20u_{i,j}\Big]. \tag{4.21}$$

From the von Neumann stability analysis of Eq. (4.16), we obtain the following expression of the amplification factor for the Laplacian discretization of Eq. (4.20)

$$G_M = 1 - \omega\Big[\frac{2a}{a+b}\sin^2\frac{k_x\Delta x}{2} + \frac{2a}{a+b}\sin^2\frac{k_y\Delta y}{2} \\ +\frac{b-a}{a+b}\big[1 - \cos(k_x\Delta x)\cos(k_y\Delta y)\big]\Big] \tag{4.22}$$

For $\alpha = a = b = 1$, we recover the expression of the amplification factor shown in [Yang and Mittal 2014, Adsuara et al. 2015]. It is easy to check that when $a = 2$ and $b = 3$, Eq. (4.22) reduces to

$$G_M = 1 - \frac{\omega}{5}\Big[4\sin^2\frac{k_x\Delta x}{2} + 4\sin^2\frac{k_y\Delta y}{2} + 1 - \cos(k_x\Delta x)\cos(k_y\Delta y)\Big]. \tag{4.23}$$

The factor multiplying ω in the previous expression is related to the weights of any SRJ scheme and singularly with the CJM. As a function of the wave number κ, the minimum amplification factor results for $k_x = k_y = \pi/L$, while the maximum amplification factor is attained for $k_x = \pi/\Delta x$ and $k_y = \pi/\Delta y$, with respective wave numbers $\kappa_{\min}$ and $\kappa_{\max}$, whose expressions are

$$\kappa_{\min} = \frac{4}{5}\sin^2\frac{\pi}{2N_x} + \frac{4}{5}\sin^2\frac{\pi}{2N_y} + \frac{1}{5}\Big[1 - \cos\frac{\pi}{Nx}\cos\frac{\pi}{N_y}\Big], \tag{4.24}$$

$$\kappa_{\max} = \frac{8}{5}. \tag{4.25}$$

It can be shown that the 9-point discretization of the Laplacian provides a fourth-order accurate method for the Poisson equation when the source term is smooth [LeVeque 2007].

Next, we consider the case of a 17-point discretization of the Laplacian. From the general form of Eq. (4.19), again writing $\alpha = a/b$ one obtains

$$\begin{aligned}\Delta u_{ij} = \frac{1}{24bh^2}\Big[\\ -2au_{i-2,j} + 32au_{i-1,j} + 32au_{i+1,j} - 2au_{i+2,j} \\ -2au_{i,j-2} + 32au_{i,j-1} + 32au_{i,j+1} - 2au_{i,j+2} \\ -(b-a)u_{i-2,j-2} + 16(b-a)u_{i-1,j-1} + 16(b-a)u_{i+1,j+1} - (b-a)u_{i+2,j+2} \\ -(b-a)u_{i-2,j+2} + 16(b-a)u_{i-1,j+1} + 16(b-a)u_{i+1,j-1} - (b-a)u_{i+2,j-2} \\ -60(a+b)u_{i,j}\Big]. \end{aligned} \tag{4.26}$$

The standard 9-point discretization of the Laplacian is recovered for $a = b = 1$ in Eq. (4.26). Performing the von Neumann stability analysis for Eq. (4.16), we obtain the following expression of the amplification factor for the Laplacian discretization of Eq. (4.26)

$$\begin{aligned}G_M = 1 - \omega\frac{1}{15(a+b)}\Big[-2a\big(\sin^2(k_x\Delta x) + \sin^2(k_y\Delta y)\big) + \\ 32a\left(\sin^2\left(\frac{k_x\Delta x}{2}\right) + \sin^2\left(\frac{k_y\Delta y}{2}\right)\right) - \\ (b-a)\big([1-\cos(2k_x\Delta x)\cos(2k_y\Delta y)] - 16[1-\cos(k_x\Delta x)\cos(k_y\Delta y)]\big)\Big], \end{aligned} \tag{4.27}$$

and,therefore, taking into account the minimum and maximum wave numbers as in the previous case, the extremal values of κ are:

$$\begin{aligned}\kappa_{\min} = \frac{1}{15(a+b)}\Big[-2a\left(\sin^2\frac{\pi}{N_x} + \sin^2\frac{\pi}{N_y}\right) + 32a\left(\sin^2\frac{\pi}{2N_x} + \sin^2\frac{\pi}{2N_y}\right) \\ -(b-a)\left([1-\cos\frac{2\pi}{N_x}\cos\frac{2\pi}{N_y}] - 16[1-\cos\frac{\pi}{N_y}\cos\frac{\pi}{N_y}]\right)\Big], \end{aligned} \tag{4.28}$$

$$\kappa_{\max} = \frac{64a}{15(a+b)}\,. \tag{4.29}$$

Let us consider the particular case $a = 1$ and $b = 2$. For the Laplacian discretization (4.26), we have

$$\begin{aligned}\Delta u_{ij} = \frac{1}{48h^2}\Big[-2u_{i-2,j} + 32u_{i-1,j} + 32u_{i+1,j} - 2u_{i+2,j} \\ -2u_{i,j-2} + 32u_{i,j-1} + 32u_{i,j+1} - 2u_{i,j+2}\end{aligned}$$

$$-u_{i-2,j-2} + 16u_{i-1,j-1} + 16u_{i+1,j+1} - u_{i+2,j+2}$$
$$-u_{i-2,j+2} + 16u_{i-1,j+1} + 16u_{i+1,j-1} - u_{i+2,j-2} - 180u_{i,j}\Big] \tag{4.30}$$

and the expressions for $\kappa_{\min}$ and $\kappa_{\max}$ of Eqs. (4.28) and (4.29) reduce to

$$\kappa_{\min} = \frac{1}{45}\Big[-2\Big(\sin^2\frac{\pi}{N_x} + \sin^2\frac{\pi}{N_y}\Big) + 32\Big(\sin^2\frac{\pi}{2N_x} + \sin^2\frac{\pi}{2N_y}\Big)$$
$$-[1 - \cos\frac{2\pi}{N_x}\cos\frac{2\pi}{N_y}] + 16[1 - \cos\frac{\pi}{N_y}\cos\frac{\pi}{N_y}]\Big] \tag{4.31}$$

$$\kappa_{\max} = \frac{64}{45} \tag{4.32}$$

Next, we numerically test the performance of the CJM for the two high-order discretizations of the Laplacian operator we have discussed above. To do so, we numerically solve the following problem:

$$\Delta u = -(x^2 + y^2)e^{xy}, \tag{4.33}$$

in the unit square with appropriate Dirichlet boundary conditions. The boundaries are specified easily in this case, since there exists an analytic solution for the problem at hand that we can compute at the edges of the computational domain. The analytic solution reads

$$u(x, y) = -e^{xy}. \tag{4.34}$$

In Figs. 4.4 and 4.5 we show the residual evolution obtained when solving problem (4.33) with different high-order discretizations of the Laplacian. In the top panel of Fig. 4.4 we use the classical 5-points discrete approximation for the Laplacian (Eq. (4.17)). It is evident that our method almost reaches the performance of the optimal SOR [LeVeque and Trefethen 1988]. In fact, as we prove in appendix A.2, this optimal weight for the SOR method coincides, up to first order with the geometrical mean of the weights obtained with our optimal scheme. In the bottom panels of Figs. 4.4 and 4.5 we display the evolution of the residual when solving the same problem but using the 9-point discretization of the Laplacian proposed by [Adams, LeVeque, and Young 1988] (Eq. (4.21)). In the bottom panel of Fig. 4.4, we use a mesh with 128 points in each dimension, while in the bottom panel of Fig. 4.5 we use 256 points per dimension. In both cases, the performance is comparable with the optimal SOR whose weight is calculated in [Adams, LeVeque, and Young 1988]. Finally, the upper panel of Fig. 4.5 shows the number of iterations when solving the same problem, but using a 64^2 grid and our 17-points Laplacian (Eq. (4.26)), with the optimal CJM obtained with the $\kappa_{\min}$ and $\kappa_{\max}$ of Eqs. (4.28) and (4.29) (i.e., in the case

$a = 1,\ b = 2$, which gives equal weight to all points in the neighborhood). In this case, the optimal weight of the SOR is unknown, so we compute the numerical solution for several values of the SOR weight. Remarkably, the CJM scheme compares fairly well with SOR.

Last but not least, we are interested in the parallel implementation of these schemes. It is known that in the case of the standard 5-points discretization of the Laplacian, one needs to implement a red-black coloring strategy for the efficient parallel implementation of SOR. In the case of the 9-points discretization of the Laplacian, [Adams, LeVeque, and Young 1988] points out that one needs four colors for a parallel implementation. Furthermore, the ordering strategy with more than two colors is not unique. Adams [Adams, LeVeque, and Young 1988] find 72 different four-color orderings, which lead to different convergence rates. In contrast, our CJM scheme (as any SRJ scheme) is trivially parallelizable since there is no need for a coloring strategy and, consequently, it possesses a unique convergence rate. We find that the tiny performance difference between the SOR method, when applied to problems where the optimal weight is unknown, and the CJM is outbalanced by the simplicity in the parallelization of the latter.

An example for the use of a 13-point stencil for the Laplacian when using a projection scheme in incompressible fluid flows can be found in [Almgren et al. 2013]. Similarly, constraint fulfilling initial data for the numerical integration in time needs to be constructed with the same spatial accuracy as the one employed in the finite difference scheme used to solve the hyperbolic evolution equations. In numerical relativity simulations, it is customary to use a fourth-order Runge-Kutta time integration, which requires at least fourth order finite differencing in spatial derivatives (see [Centrella et al. 2010] for a review). In the latter case, it is convenient to construct also the initial data employing a fourth-order method to solve the ePDEs involved.

Furthermore, to discuss advantages arising from higher order discretizations, let us consider Eq. (4.33) once more. As we know the analytic solution to our problem (Eq. (4.34)), we can monitor the real error at each iteration in the computation of our numerical solution.[6] In Fig. 4.6 we show that a significantly higher number of grid points is needed when employing lower-order discretization stencils in order to achieve approximately the same error (i.e. a solution of the same quality). With a mesh of only 32×32 points we reach the sought accuracy goal employing a dicretization of the Lagrangian with a 17-points stencil. To achieve the same accuracy with our 9-point stencil discretization, about 128×128

[6] In actual computations, where we do not know the real error, we monitor the residual that shall be proportional to the real error.

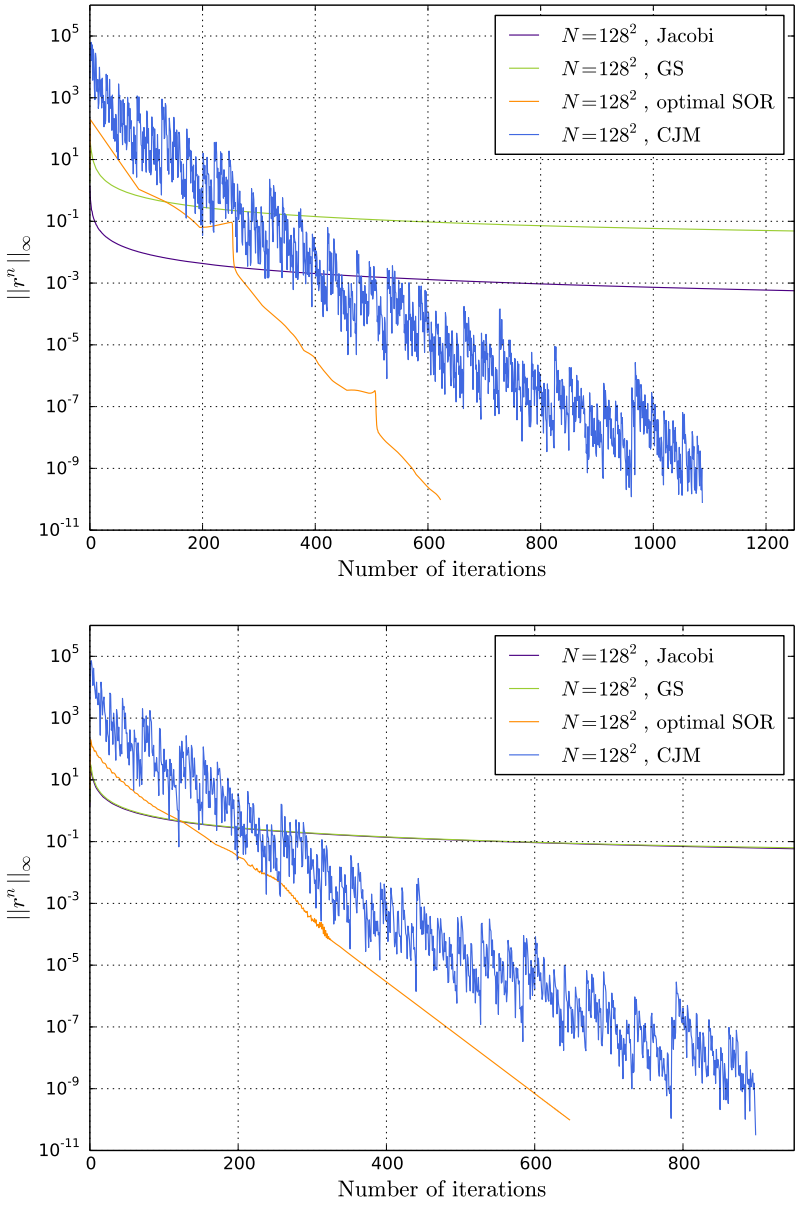


Figure 4.4 Evolution of the residual for the solution of the Poisson equation (4.33) in 2D, with 5-points discrete Laplacian (Eq. (4.17); top panel) and 9-points (Eq. (4.21); bottom panel) for different iterative methods and for the resolutions indicated in the legends. Note that the bottom panel corresponds to a problem set up with $N_x = N_y = 128$.

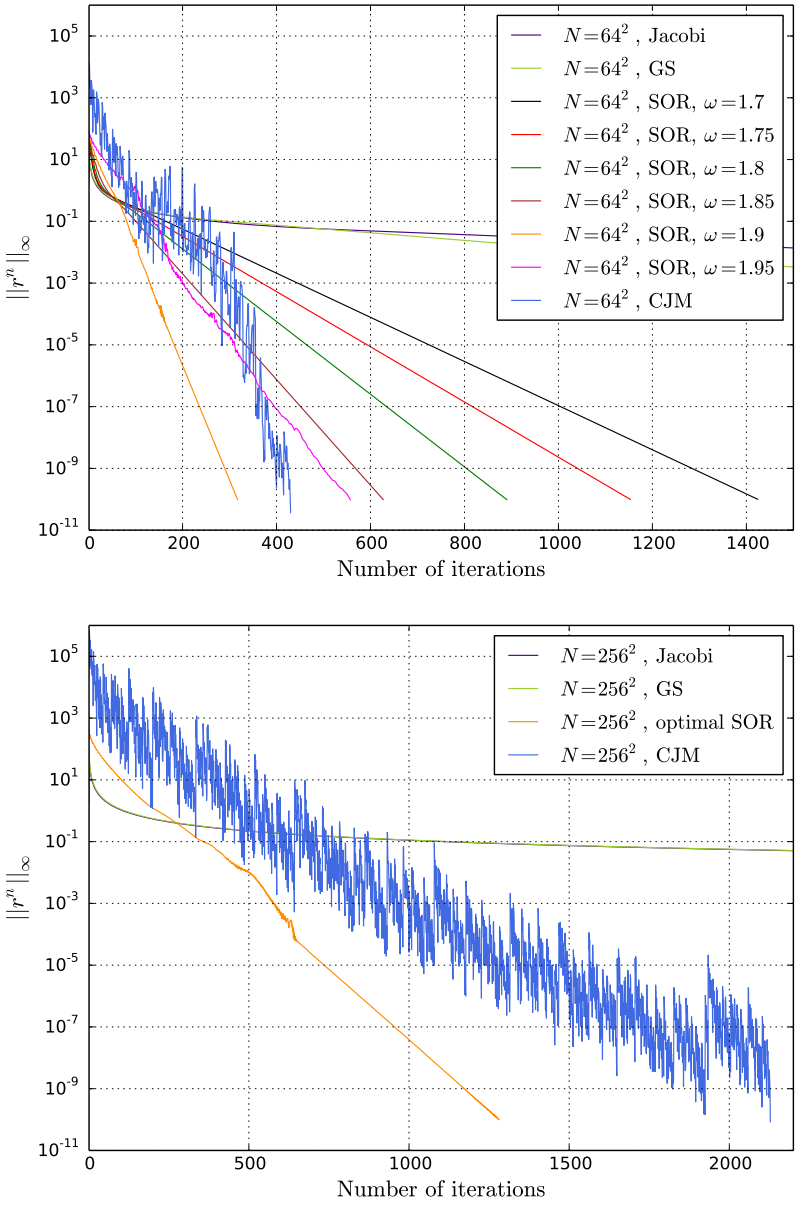


Figure 4.5 Evolution of the residual for the solution of the Poisson equation (4.33) in 2D, 17-points Laplacian (Eq. (4.26); top panel) and with 9-points (Eq. (4.21); bottom panel) for different iterative methods and for the resolutions indicated in the legends. Note that the bottom panel corresponds to a problem set up with$N_x = N_y = 256$ points.

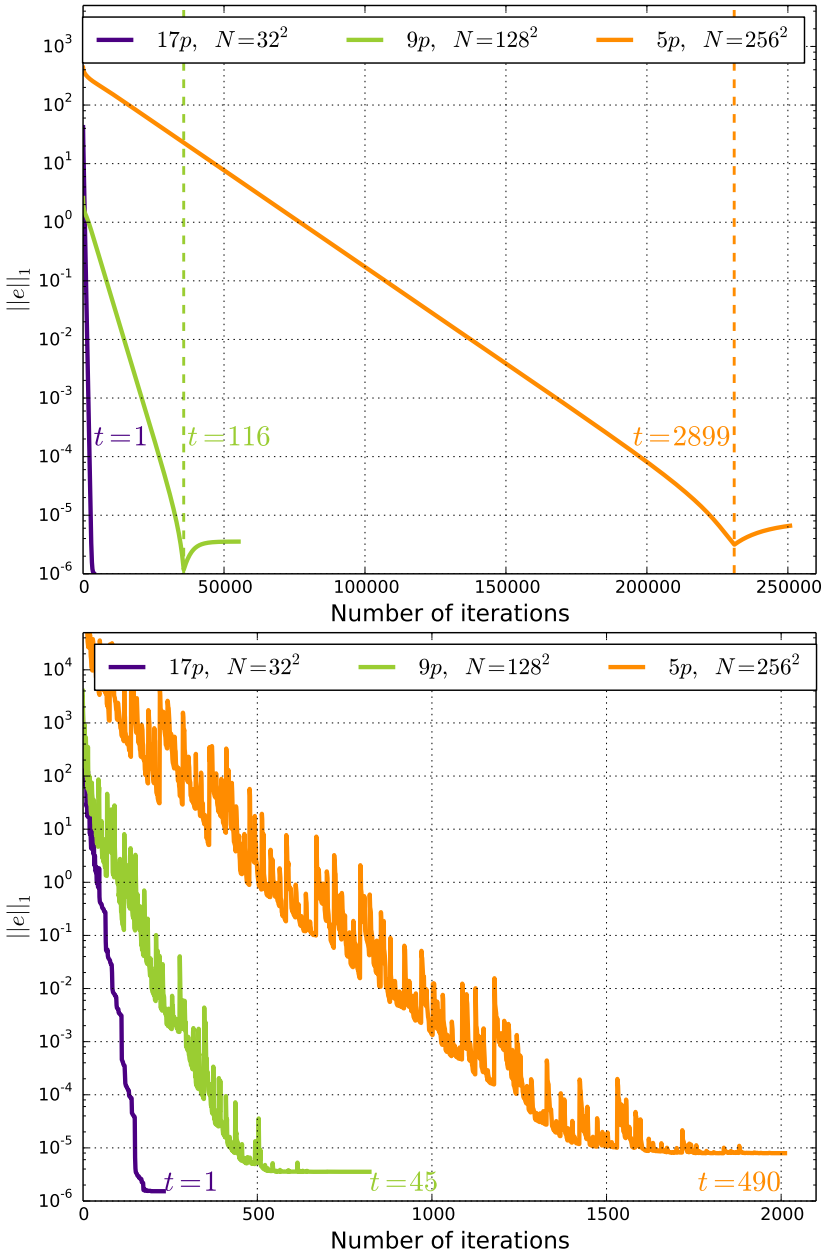


Figure 4.6 Norm-1 error of the numerical solution with respect to the analytic solution (Eq. 4.34) for the Jacobi method (top panel) and CJM (bottom panel) with different resolutions and orders of discretization of the Laplacian in Eq. (4.33). The numerical solution is evolved until the norm-1 error reaches, approximately, 10^{-6}. We also annotate besides each of the lines the time needed to run the model at hand normalized to the run time of the fastest case.

grid zones are needed, resulting in approximately 4 to 10 times more iterations than with the 17-points stencil when using the CJM or the Jacobi schemes, respectively. In the case of the standard second-order 5-point stencil, the grid should contain more than 256×256 points and the number of iterations increases by about 60 times when applying the Jacobi method, and 10 times in CJM with respect to the number required when using the maximum order stencil. Although each step of the iterative algorithm performs more operations for higher order discretizations, this penalty is negligible compared to the considerable reduction in the number of iterations, which in turn translates into a huge decrease in the actual calculation time (see the labels of Fig. 4.6). Therefore, we have shown that not only the number of iterations increases when employing low-order discretizations of the Laplacian, but also that the computational time needed to arrive to a prescribed norm-1 error goal is also substantially larger.

As a final point we note that these NCO matrices lead to more compact stencils which effectively reduce the communications in parallelizations with distributed memory (message passing paradigm). We will turn back to this idea in Chap. 6

Part III

Applying the SRJ and the CJM methods in astrophysics and beyond

Chapter 5

Sequential applications in astrophysics

The results of this chapter have been originally published in:

Scheduled relaxation Jacobi method: improvements and applications,
JE Adsuara, I Cordero-Carrión, P Cerdá-Durán, MA Aloy,
Journal of Computational Physics 321, 369-413 (2016)

In this chapter, we present numerical results of the methods discussed in the previous chapters applied to the resolution of some problems of interest in astrophysics. We limit ourselves here to a sequential implementation of the method. Results in of a parallel implementation are presented in the next chapter.

5.1 Poisson equation in spherical coordinates

The Poisson equation appears, among others, in problems involving gravity, either Newtonian or some approximations to General Relativity, and electrostatics. In numerical simulations, e.g., in Astrophysics and Cosmology, the computation of the gravitational potential is usually coupled to a hyperbolic set of equations describing the dynamics of the fluid, e.g., the Euler equations. In those cases the Poisson equation is solved on each time step (or every several time steps) of the evolution of the hyperbolic part. Our aim is to test the efficiency of the SRJ and

CJM compared with other methods currently used by the scientific community. In simulations of stellar interiors, spherical coordinates are a popular choice of coordinates, so we adopt them for our test. To mimic typical astrophysical scenarios we have chosen a test in which the source has compact support and boundary conditions are applied at radial infinity.

The Poisson equation in spherical coordinates (r, θ, φ) reads

$$\frac{\partial^2 u}{\partial r^2} + \frac{2}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} + \frac{\cot\theta}{r^2}\frac{\partial u}{\partial \theta} + \frac{1}{r^2 \sin\theta}\frac{\partial^2 u}{\partial \varphi^2} = s, \tag{5.1}$$

being u and s functions of (r, θ, φ). For our test we choose the source term to be the series

$$s(r,\theta,\varphi) = \begin{cases} -\sum_{n=0}^{n_{\max}} \sum_{m=-m_{\max}}^{m_{\max}} a_{2n}\, k_{2n}^2\, j_{2n}(k_{2n} r)\, Y_{2n}^{m,c}(\theta,\varphi), & \text{for } r \le 1 \\ 0, & \text{for } r > 1, \end{cases} \tag{5.2}$$

being j_l the spherical Bessel functions of the first kind and $Y_l^{m,c}$ the real part of the spherical harmonics. Only even parity terms, $l = 2n$ are considered. k_l is the first root of the spherical Bessel function of order l, such that $s(1, \theta, \varphi) = 0$. We chose $a_l = 1/2^l$, such that the series is convergent. We impose homogeneous Neumann boundary conditions at $r = 0$, $\theta = 0$ and $\theta = \pi$, and periodic boundary conditions in the φ direction. If we impose homogeneous Dirichlet conditions at radial infinity ($r \to \infty$), the solution of this elliptic problem is

$$u(r,\theta,\varphi) = \begin{cases} \sum_{n=0}^{n_{\max}} \sum_{m=-m_{\max}}^{m_{\max}} \left(a_{2n}\, j_{2n}(k_{2n} r) + b_{2n} r^{2n}\right)\, Y_{2n}^{m,c}(\theta,\varphi), & \text{for } r \le 1 \\ \sum_{n=0}^{n_{\max}} \sum_{m=-m_{\max}}^{m_{\max}} \frac{c_{2n}}{r^{2n+1}}\, Y_{2n}^{m,c}(\theta,\varphi), & \text{for } r > 1, \end{cases} \tag{5.3}$$

where the coefficients c_l and b_l can be computed imposing continuity of u and its first derivatives at $r = 1$, resulting in

$$b_l = c_l = -\frac{a_l}{2l+1} \partial_r j_l(k_l r)|_{r=1} = -\frac{a_l}{2l+1}\left[l\, j_l(k_l) - k_l\, j_{l+1}(k_l)\right] \tag{5.4}$$

Since our interest is to assess the performance of the SRJ and CJM methods under the conditions which are found on real applications, we solve this equation numerically in the domain $r \in [0, 1]$, $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi]$, i.e. only in the region where the sources are non zero, and apply Dirichlet boundary conditions at $r = 1$, using the analytical solution given by Eq. (5.3).

We emphasize that this problem set up includes boundary conditions of mixed type (Neumann and Dirichlet) and, hence, none of the schemes whose optimal parameters have been tabulated in this chapter is strictly optimal. However, as we shall see, even in such conditions, the new schemes presented in this chapter, in particular the CJM, will be competitive with other alternatives in the literature.

We set up three versions of the test with different dimensionality. In the 3D test, we choose $n_{\max} = \infty$ (in practice, we sum only a sufficiently large number of terms to obtain an accurate enough value of the infinite sum; see below) and $m_{\max} = 2n$ and solve the equation in the domain $r \in [0, 1]$, $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi]$, discretized in an equidistant grid of size $N \times N \times N$ points. In the 2D case we consider axisymmetry, i.e., no φ dependence in u and s. We choose $n_{\max} = \infty$ and $m_{\max} = 0$ and solve in the domain $r \in [0, 1]$ and $\theta \in [0, \pi]$, discretized in a grid with $N \times N$ points. In the 1D case we consider spherical symmetry, i.e. no θ or φ dependence in u and s. We choose $n_{\max} = 0$, $m_{\max} = 0$ and solve in the domain $r \in [0, 1]$ with N points. Since the series in Eqs. (5.2) and (5.3) are convergent, we compute them numerically by adding terms until the last significant digit does not change. We use a second-order finite difference discretization for Eq. (5.1) and one ghost cell in each coordinate direction to impose boundary conditions. For convenience, we multiply Eq. (5.1) by r^2 in the discretized version.

As an example, we present explicitly the discretization of the 1D problem. The 2D and 3D discretizations are analogous to what is described here. We use a staggered grid with ghost cells, $r_i = (i - 1/2)\Delta r$ with $i = 0, ..., N + 1$, where $\Delta r = 1/N$. Points $i = 0$ and $i = N + 1$ are ghost cells used only for the purpose of imposing boundary conditions. Using second-order centered differences to approximate the second partial derivatives and imposing spherical symmetry ($\partial_\theta = \partial_\varphi = 0$) Eq. (5.1), multiplied by r^2, can be discretized as

$$r_i^2 \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta r^2} + 2r_i \frac{u_{i+1} - u_{i-1}}{2\Delta r} = r_i^2 s_i, \quad (i = 1, ..., N) \tag{5.5}$$

where sub-index i indicates a function evaluated at r_i. The values of u_0 and u_{N+1} are set as boundary conditions. The resulting linear system of N equations with N unknowns, u_i, $i = 1, ..., N$, can be written in matrix form

$$\sum_{j=1}^{N} \mathcal{A}_{ij} u_j = r_i^2 s_i, \quad (i = 1, ..., N), \tag{5.6}$$

being $\mathcal{A}_{ij}$ the elements of the coefficient matrix, which in the 1D case is a $N \times N$ tridiagonal matrix

$$\begin{aligned} \mathcal{A}_{ii-1} &= (r_i - 2\Delta r)\,\frac{r_i}{\Delta r^2} \\ \mathcal{A}_{ii} &= \frac{-2r_i^2}{\Delta r^2} \\ \mathcal{A}_{ii+1} &= (r_i + 2\Delta r)\,\frac{r_i}{\Delta r^2} \\ \mathcal{A}_{ij} &= 0, \quad \text{otherwise.} \end{aligned} \tag{5.7}$$

Note that this matrix is diagonally dominant by rows and columns except for the first two rows and the first column. If the r^2 factor were not present, the matrix would not be diagonally dominant by columns and the convergence of the iterative methods could not be guaranteed. Once the boundary conditions are applied, the coefficient matrix is effectively modified. Wether the resulting effective matrix is diagonally dominant or not depends crucially on how the boundary conditions are applied.

We impose Dirichlet boundary conditions at the outer boundary by setting $u_{N+1} = u_{\text{analytic}}(r_{N+1})$, being the analytic solution that is given by Eq. (5.3). In this case the equation at $i = N$ results

$$\mathcal{A}_{NN-1}u_{N-1} + \mathcal{A}_{NN}u_N = r_N^2 s_N - \mathcal{A}_{NN+1}u_{\text{analytic}}(r_{N+1}), \tag{5.8}$$

being $\mathcal{A}_{NN+1}$ an extension of the coefficient matrix used for practical purposes. At the inner boundary we impose homogeneous Neumann conditions, i.e. $\partial_r u|_{r_0} = 0$. Standard numerical techniques to deal with this kind of boundary conditions (see e.g. [Smith 1985], p. 76-77), expand the Laplacian operator around $r = 0$, resulting in

$$\partial_{rr}u + 2/r\,\partial_r u = 3\partial_{rr}u + \mathcal{O}(\Delta r^2). \tag{5.9}$$

The second-order discretization of this equation at the first radial cell yields

$$\Delta u|_{r_1} \approx 3\frac{u_2 - 2u_1 + u_0}{\Delta r^2}, \tag{5.10}$$

and a second-order discretization of the boundary condition results in

$$\frac{u_1 - u_0}{\Delta r} = 0, \tag{5.11}$$

i.e. $u_1 = u_0$, which results in

$$\Delta u|_{r_1} \approx 3\frac{u_2 - u_1}{\Delta r^2}. \tag{5.12}$$

Using this prescription for the discretization of the Laplacian operator at the first radial cell, ensures that the matrix is diagonally dominant by rows and columns.

An additional simplification can be made by noticing that, as long as the source s is regular at $r = 0$, Eq. (5.12) implies that $u_1 = u_2 + \mathcal{O}(\Delta r^2)$. Therefore, for a second-order method, one can assume $u_0 = u_1 = u_2$, as an alternative prescription to solve the problem of the diagonal dominance. Since this prescription not only fixes the value of the ghost cell, u_0, but also the value u_1, this condition reduces the dimensionality of the linear system by 1. The resulting effective coefficient matrix $\hat{\mathcal{A}}_{ij}$ is a tridiagonal matrix of size $(N-1) \times (N-1)$, with indices $i, j = 2, ...N$. The elements read

$$\begin{aligned}
\hat{\mathcal{A}}_{22} &= -\left(r_2 + 2\Delta r\right) \frac{r_2}{\Delta r^2} = -\frac{21}{4} \\
\hat{\mathcal{A}}_{23} &= \left(r_2 + 2\Delta r\right) \frac{r_2}{\Delta r^2} = +\frac{21}{4} \\
\hat{\mathcal{A}}_{ij} &= \mathcal{A}_{ij}, \quad \text{otherwise.}
\end{aligned} \tag{5.13}$$

The new matrix is diagonally dominant by rows and columns, which guarantees the convergence of Jacobi-based iterative methods. In practice, the effective coefficient matrix, $\hat{\mathcal{A}}_{ij}$, is not used in the iterative methods directly. Instead we use a coefficient matrix $\mathcal{A}_{ij}$ extended to the ghost cells and we set at each iteration the values at the ghost cells (u_0 and u_{N+1}) and at u_1 according to prescription given above. This procedure is equivalent to using the effective coefficient matrix, but it eases the implementation of the algorithm.

We have tested both prescriptions to make the matrix diagonally dominant, namely that of Eq. (5.12) and the additional simplification in Eq. (5.13). We have found that, although both prescriptions result in convergent methods, the number of iterations needed to converge is systematically $\sim 30\%$ smaller with the second prescription, for all the iterative methods tested. Therefore, we provide here only results for the second (faster) prescription.

We perform series of calculations for different values of the number of points N. For each calculation we use the SRJ and CJM coefficients computed in Part. (II) matching the corresponding value of N. In the SRJ case, for each series we perform calculations using coefficients computed with different values of P and the optimal M. In the CJM case, we take the optimal value of M to achieve a reduction of the residual equal to the required tolerance, in the same way as it was described in the examples of Part. (II). The convergence criterion

is that the L_∞-norm of the residual, defined as

$$||r||_\infty = \max_{i=1,\ldots,N} \left| \sum_{j=1}^{N} \mathcal{A}_{ij} u_j - r_i^2 s_i \right|, \tag{5.14}$$

is smaller than the tolerance. Note that this criterion differs from previous subsections, Eqs. (3.38) and (3.39), in a factor ω_i. The tolerance goal is set to $10^{-5}/N^2$, which depends on the number of points. Since we use a second-order discretization, this scaling in the tolerance ensures that the difference between the numerical and the analytical solution is dominated by finite differencing errors and at the same time avoids unnecessary iterations in the low resolution calculations. This prescription for the tolerance mimics the tolerance choice that is used under realistic conditions and renders a fairer comparison in the computational cost between different resolutions.

For comparison we also perform calculations using other iterative methods: Jacobi, Gauss-Seidel and SOR (weight equal to 1.9). For each case involving iterative methods we perform two calculations: the *ab initio calculation* in which the solution is initialized to zero, and the *realistic calculation* in which the solution is initialized to $u_{\rm analytic}(1 + ran(-0.5, 0.5)/N)$, being $ran(-0.5, 0.5)$ a random number in the interval $[-0.5, 0.5]$. The *realistic calculation* tries to mimic the conditions encountered in many numerical simulations in which an elliptic equation (or a system of PDEs) is solved coupled with evolutionary (hyperbolic) PDEs,[1] which are typically solved using explicit methods, whose time step is limited by the Courant-Friedrichs-Lewy (CFL) condition. This means that the change in the source of the Poisson equation between subsequent calculations is $\mathcal{O}(\Delta x)$; therefore, if the solution of the previous time step is used for the iteration in the elliptic solver, this should differ only $\mathcal{O}(\Delta x) \sim 1/N$, from the solution.

In addition to iterative methods, we perform the calculations using a direct inversion method and spectral, methods the later ones quite used in astrophysical applications. In the *direct inversion method*, we compute the LU factorization of the matrix associated with the coefficients of the discretized version of the equation by performing Gaussian elimination using partial pivoting with row interchanges. We use the implementation in the `dgbtrf` routine of the `LAPACK` library [*LAPACK - Linear Algebra PACKage*], which allows for efficient storage of the matrix coefficient in bands. Once the LU decomposition is known, we solve the system of linear equations using the `dgbtrs` routine. Since this method is non-iterative, its computational cost does not depend on the initial

[1] For instance, this is the case of the fluid equations.

value. However, this approach has advantages when used repeatedly, coupled to evolution equations (e.g., for a fluid). Most of the computational cost of this method is due to the LU decomposition, but once it has been performed, solving the linear system for different values of the sources is computationally less intensive. Therefore, we consider the computational cost of the whole process, LU decomposition and solution of the system, in the *ab initio calculations*, while only the solution of the linear system, assuming the LU decomposition is given, in the *realistic calculations*.

For the spectral solver we use the `LORENE` library [*LORENE - Langage Objet pour la RElativité NumériquE*]. To provide results that are comparable to all other numerical methods used in the present work we use the following procedure: first we evaluate the source, s, at the *finite differences grid* used in all other numerical methods; then, the source is interpolated to the collocation points in the *spectral grid*, which do not coincide with the *finite differences grid*; next the solution is computed by means of the `LORENE` library; finally the function is evaluated at the cells of the *finite differences grid*. The details of the procedure are described in Dimmelmeier et al. 2005. The accuracy of the numerical method is dominated by the second-order finite differences discretization error associated with the *finite differences grid*, provided sufficient number of collocation points are used in the *spectral grid*. We have tested that it is sufficient to use $N/2$ collocation points per dimension to fulfill this requirement. When using the spectral solver, there is no difference in the computational cost in *ab initio* or *realistic calculations*.

We have performed the calculations using a 3.4 GHz Intel core i7 and 16 GB of memory. We have measured the computational time for each method timing exclusively the part of the code involved in the computation and not the allocation and initialization of variables. Figures 5.1–5.3 show the dependence of the computational time for 1D, 2D and 3D tests in the *ab initio calculations* and the *realistic calculations* setups. As expected, for any dimensionality, the SRJ and CJM methods render a significant speed up with respect to other iterative methods, due to the smaller number of iterations needed. Only SOR method has comparable computational time for low resolutions ($N < 100$). The computational time for SRJ and CJM methods scales approximately as N^{d+1}, being d the dimensionality of the test, i.e. the number of iterations is proportional to N. In comparison, the computational costs of other iterative methods (Jacobi, Gauss-Seidel, SOR), scale approximately as N^{d+2}, i.e. the number of iterations needed scales as N^2. This factor N improvement of SRJ and CJM with respect other iterative methods ensures that the method will

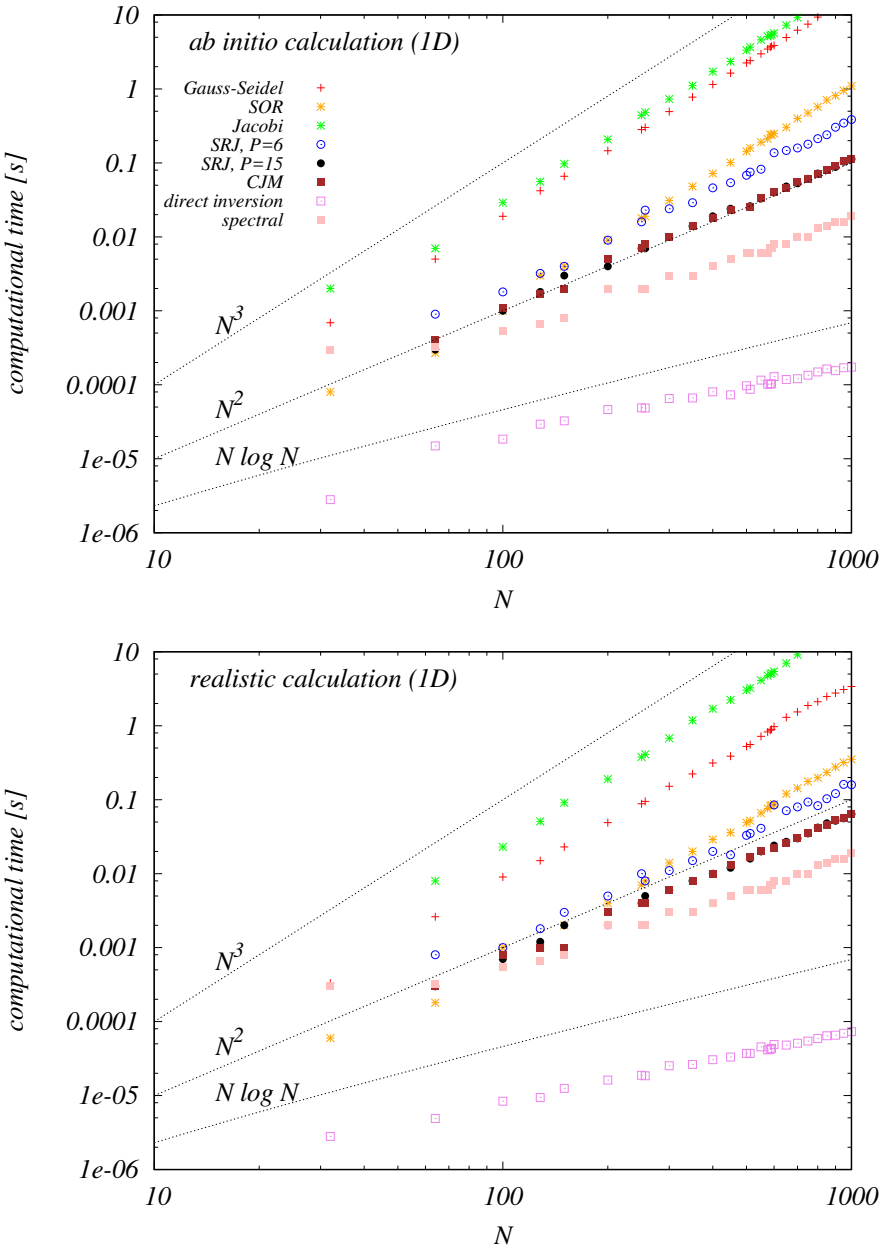


Figure 5.1 Computational cost of the solution of the Poisson equation in spherical coordinates, depending on the size of the problem N, using different numerical methods, including SRJ for the minimum ($P = 6$), maximum ($P = 15$) and CJM set of coefficients computed in Part. (II). The 1D test case is show for show *ab initio* (top) and *realistic calculations* (bottom), respectively. Note that points for $P = 15$ and CJM lay on top of each other in many cases.

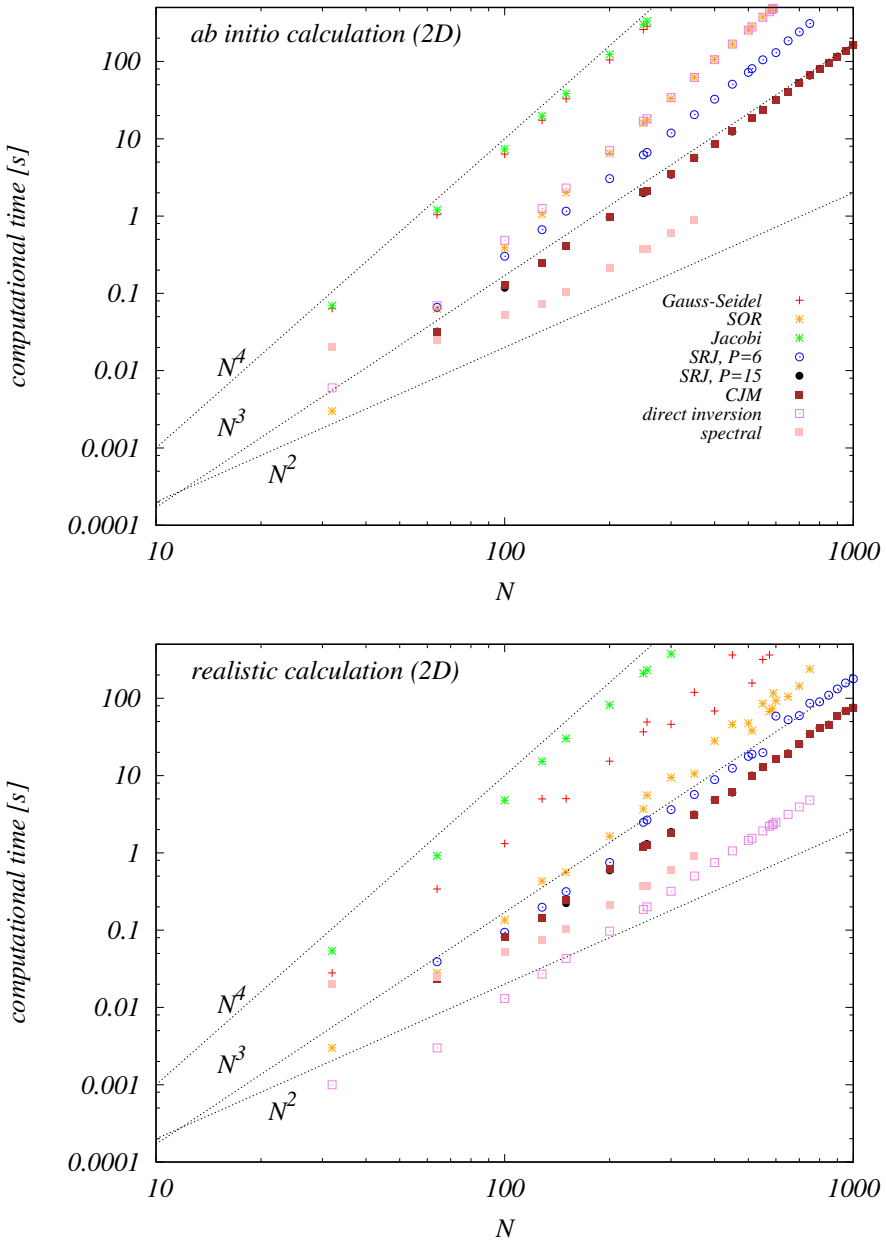


Figure 5.2 Same as Fig. 5.1 but for the 2D test case.

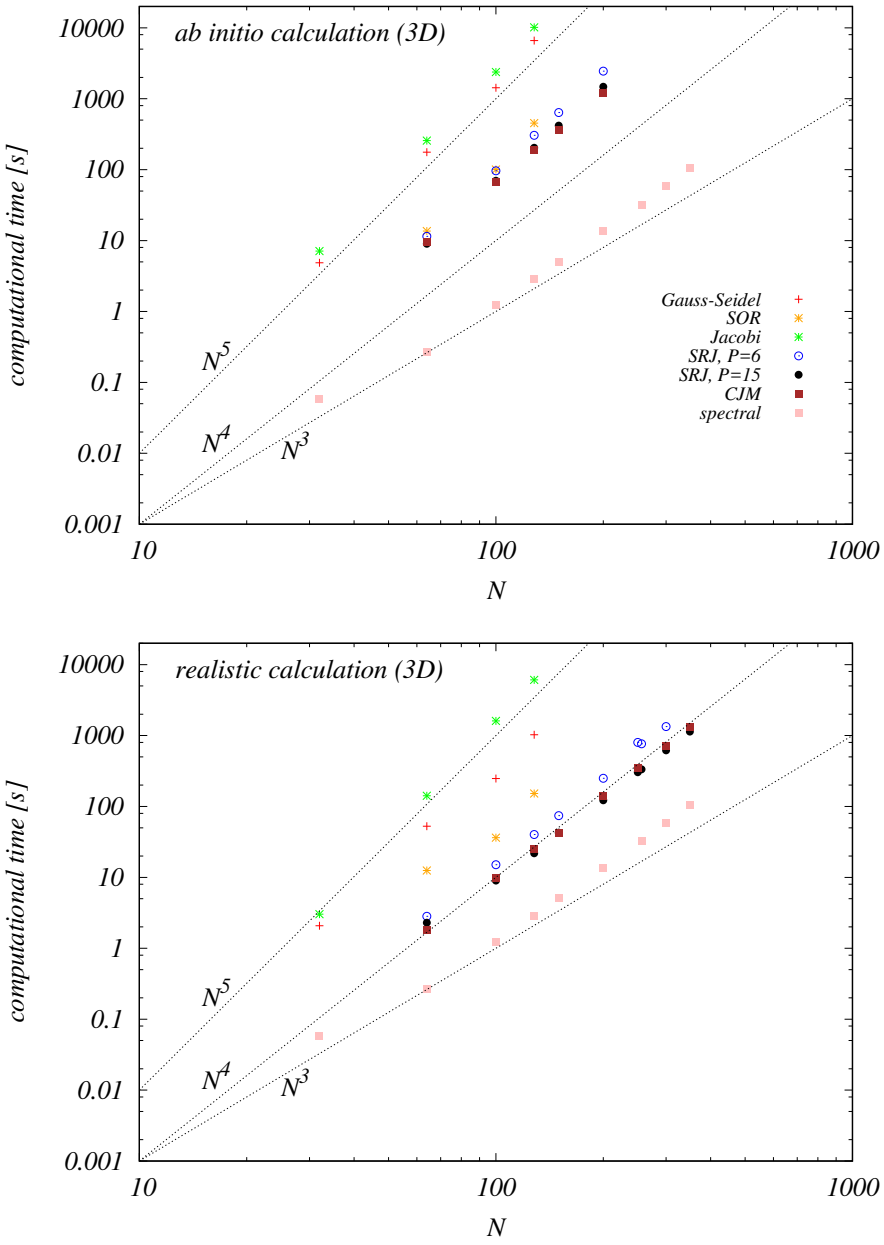


Figure 5.3 Same as Fig. 5.1 but for the 3D test case.

always be less costly for sufficient high resolution. In all cases, CJM shows a very similar behaviour to the SRJ method with $P = 15$. Compared to non-iterative methods the results depend on the dimensionality of the test.

For the 1D test (Fig. 5.1), both spectral and the direct inversion method are significantly faster than SRJ and CJM. The computational cost of both methods are close to $N \log N$. Therefore, we conclude that SRJ and CJM methods are not competitive for 1D problems, even when *realistic* conditions are considered.

In 2D (Fig. 5.2), the computational cost of the direct inversion method for *ab initio calculations* increases significantly, scaling as N^4, because the associated matrix is not tridiagonal anymore, as in the 1D case, but is a banded matrix of band size $2N + 1$. Hence, the direct inversion method is more costly than SRJ and CJM for resolutions $N > 100$. However, in the *realistic calculation*, in which the LU decomposition is not performed, the direct inversion method is still the fastest, with a computational time scaling as N^3 (the same as SRJ and CJM) but with lower computational cost. Due to limitations of the `LORENE` library, we were not able to perform multidimensional computations using spectral methods for $N > 350$. Compared to iterative methods, spectral methods are about a factor 2 faster than SRJ ($P = 15$) and CJM in *realistic calculations* and become comparable for $N < 100$. It seems fair to say that spectral methods perform better for *ab initio calculations*, since in this case, SRJ and CJM methods (see $P = 6$ and $P = 15$ in Figure 5.1) scale as N^3. Therefore, we conclude that for 2D calculations SRJ and CJM is a competitive method, when compared with spectral methods. Although the direct inversion method is the fastest in the range of values of N selected for our tests, we expect that this advantage will disappear when going to larger number of points; the memory needed for the direct inversion method scales as N^3 (due to the explicit use of the banded structure of the matrix) in comparison with N^2 of all other methods (iterative and spectral). This strongly limits the size of the problem to be solved without using parallelization.

In 3D (Fig. 5.3) all computations are significantly more costly, so we limit our tests to what is achievable within ~ 1 hour of computation time. For the SRJ and CJM methods tested this is $N \leq 200$ in *ab initio calculations* and $N < 400$ or *realistic calculations* (note that for spectral methods, we have run also cases with larger values of N). For $N > 100$ the computational cost of spectral methods is a factor ≈ 10 lower than a SRJ methods with $P = 15$ and CJM, in the *realistic calculation*. Using the SRJ parameters for $P = 15$ or the optimal CJM and an effective number of points per dimension as given by Eq. (3.10), the SRJ/CJM method become $\sim 20\%$ faster, so that it is "only" ≈ 8

slower than the spectral method. The conclusion is that spectral methods still seem to have advantages over SRJ and CJM methods, for the 3D test presented. However, both spectral and SRJ/CJM methods scale approximately as N^4 in 3D. Due to the large amount of memory needed for the direct inversion method, which scales as N^5, we did not present any such calculation for the 3D case. In practice, this limitation makes the direct inversion method unfeasible for computations in a single CPU. The performance of iterative methods in parallel architectures is explored in Chap. 6.

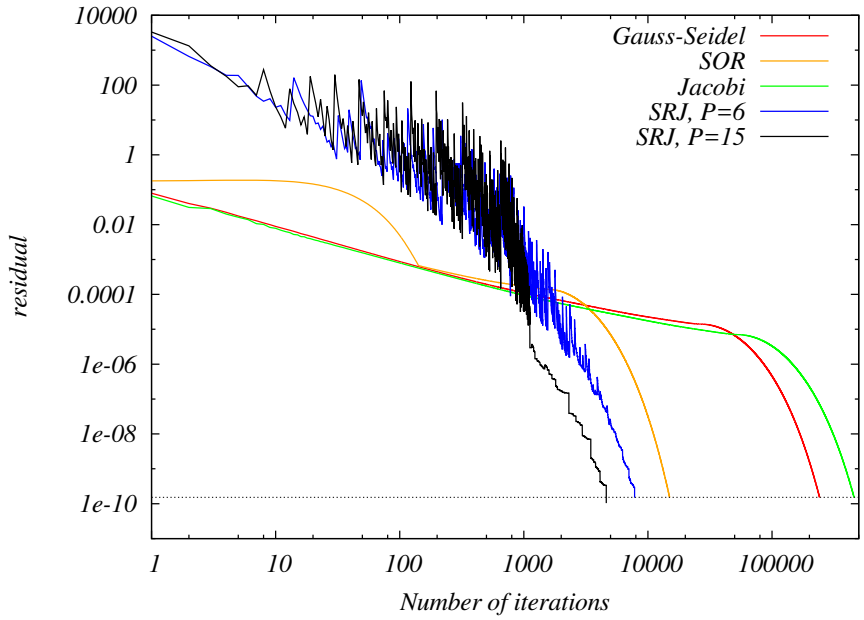


Figure 5.4 Example of the evolution of the residual during the iterative procedure, for several of the iterative methods used in this work. The dashed-black horizontal line corresponds to the tolerance goal.

Figure 5.4 shows the evolution of the tolerance, computed using Eq. (5.14), for a 2D simulation with $N = 256$, and the initial conditions of the *realistic calculation*, using different iterative methods (CJM is not plotted, but it shows a very similar behaviour to the SRJ method with $P = 15$). Note, that the residual at the first iteration is significantly larger than in Jacobi, due to the use of a weight with large value in this iteration ($\omega_1 = 19127$ and 25234 for $P = 6$ and 15, respectively). This is compensated by a faster average convergence in cycles of M iterations as expected from SRJ methods, being $M = 781$ and 1154 for $P = 6$ and 15, respectively.

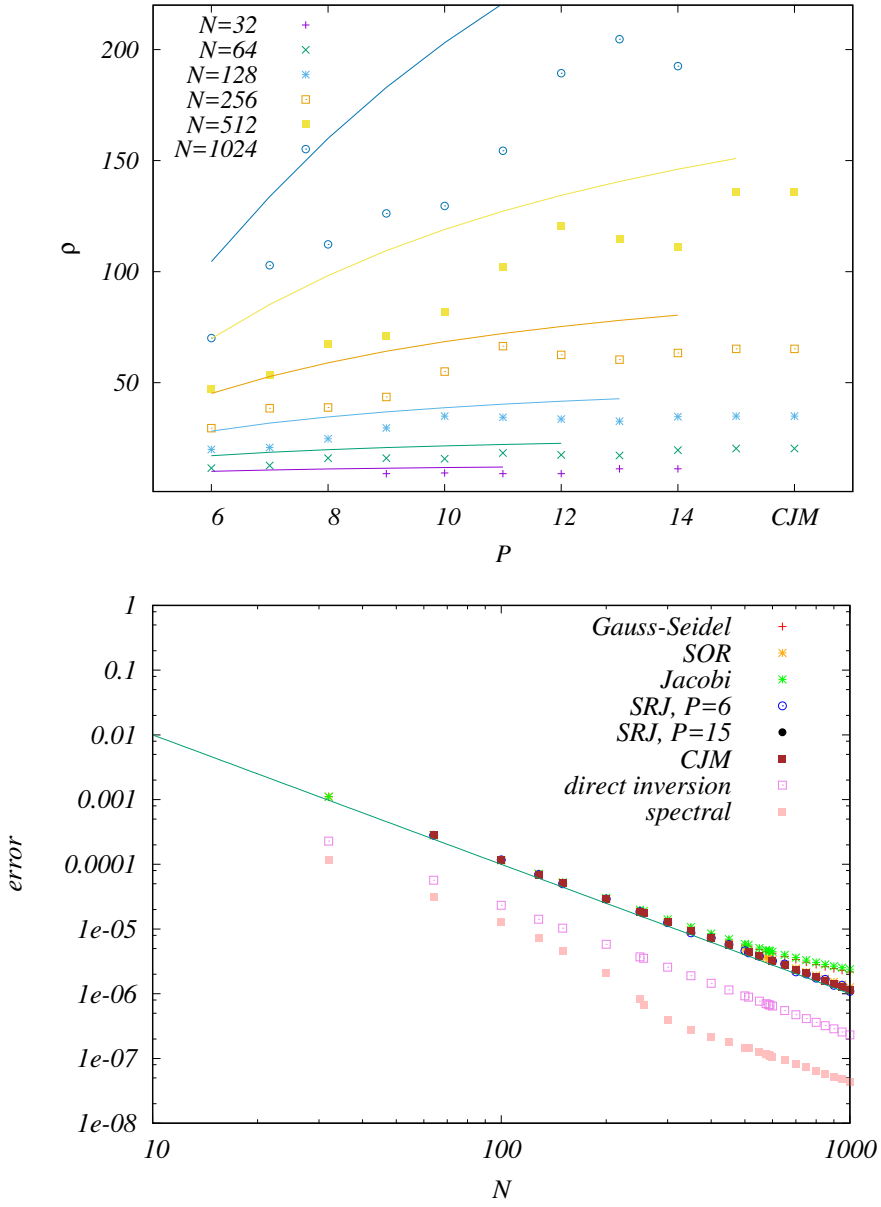


Figure 5.5 Detailed analysis of the solution of the Poisson equation for 1D. The top panel shows the dependence of the numerically estimated value of ρ on P, for several values of N ranging from 32 to 1024. For CJM the value of P exceeds by far those used in SRJ methods and is simply marked as "CJM" in the scale. Solid lines of the same color represent the theoretical estimate of ρ, for each case. The bottom panel shows the error in the solution as a function of N, computed as the L_∞-norm of the difference between the numerical and the analytical solution. The solid black line represents $1/N^2$, which is an estimation of the expected finite difference error.

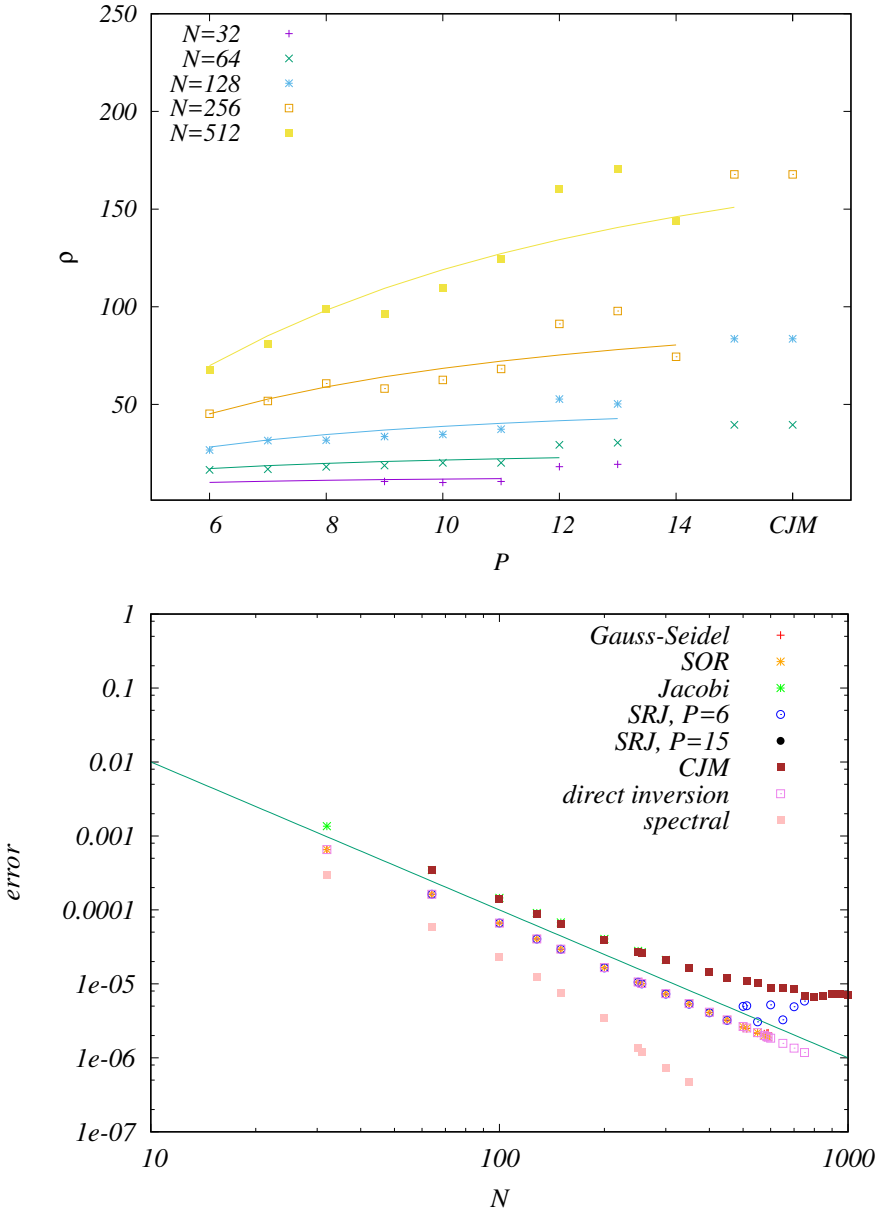


Figure 5.6 Same as Fig. 5.5 for the Poisson equation in 2D.

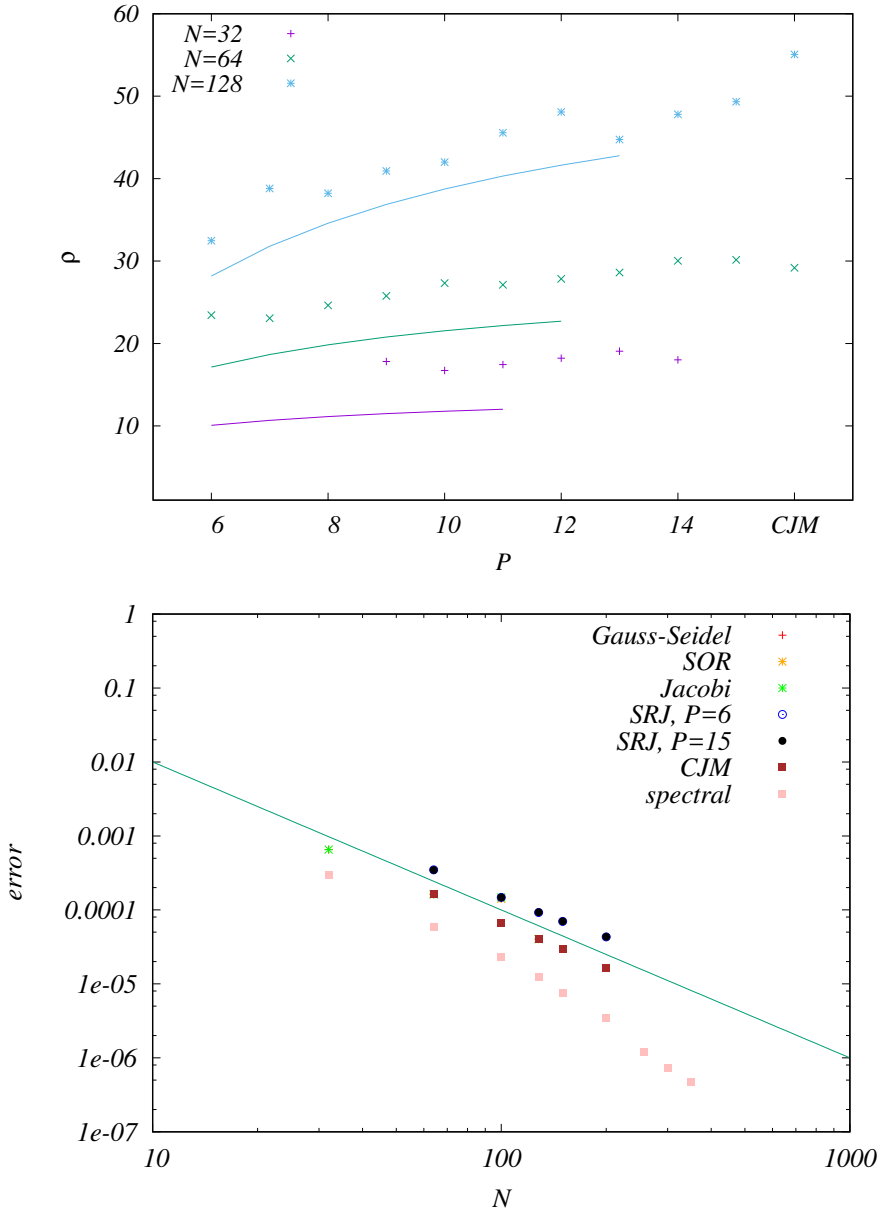


Figure 5.7 Same as Fig. 5.5 for the Poisson equation in 3D.

Finally, we have estimated numerically the value of ρ (Eq. 3.37) for different SRJ and CJM weights, to be compared with the theoretical predictions. For this purpose we compute the ratio of number of iterations needed with Jacobi and a given SRJ/CJM method, using the same tolerance and resolution, N. The upper panels of Figs. 5.5–5.7 show the dependence of ρ on P for several values of N, computed using the set of *ab initio calculations*. For CJM the value of P exceeds by far those used in SRJ methods (typically several times N) and is simply marked as "CJM" in the scale. Regardless of the dimensionality, in all calculations the numerical values of ρ are close to the theoretical predictions (solid lines). In the 1D test problems (Fig. 5.5) there is a tendency of the theoretical values to overestimate the numerically computed value. This trend is exacerbated for large values of N (namely, $N > 512$). In 3D the situation is reversed (Fig. 5.7), and the theoretical value of ρ falls below the numerical one. To explain these differences, we shall consider that the optimal weights depend on the dimensionality of the problem, since κ_m does also depend on dimensionality (see Eq. 3.9). As the optimization of the weights has been performed for the 2D case, it is not surprising to find such discrepancies when using the same weights and the same value of ρ in a problem with different dimensionality. Indeed, we have repeated some of the 3D and 1D test problems employing the optimal SRJ and CJM parameters corresponding to the effective number of points set according to Eq. (3.10), and found that (i) the SRJ and CJM scheme runs is $\sim 20\%$ less iterations and, (ii) for this effective number of points, the theoretical convergence performance index, computed with the dimensionality corrections mentioned above Eq. (3.37), becomes an upper bound for the numerical values of ρ. Adding to this arguments, we also note that the discretization of the Laplacian operator in spherical coordinates may also change slightly the optimal weights. Finally, another factor that explains the discrepancies is that the boundary conditions of this problem are mixed (as commented above), and the optimal weights are computed for purely Neumann boundary conditions.

We find that for 2D applications increasing P from 6 to 15 yields an increase in ρ of $\sim 2-3$ for the largest resolutions considered here (Figs. 5.5–5.7; upper panels). CJM yields similar results to the SRJ method with $P = 15$ in all cases. In 3D, the increment of ρ is expected to be similar (we do not show examples with larger values of N due to the long duration of the tests and the fact that such numerical grids are typically computed with parallel algorithms, which we do not discuss here; but see Chap. 6). Hence, it is worthwhile employing SRJ schemes with a larger number of levels than those originally proposed in YM14, specially considering that there is no extra complexity in the algorithm

implementation for any $P \geq 2$, once the weights for large values of P are known. The use of CJM is even more convenient because, although it did not showed a significant improvement over the SRJ method with $P = 15$, the computation of the coefficients is significantly faster and can be easily adapted for any grid size.

The lower panels of Figs. 5.5–5.7; show the error in the solution as a function of N, computed as the L_∞-norm of the difference between the numerical and the analytical solution. In all cases the error is dominated by the finite difference error associated to the discretization of the elliptic operator, which, for a second-order method, is expected to be $\sim \mathcal{O}((\Delta x)^2) \sim \mathcal{O}(1/N^2)$. This is a symptom that our prescription for the tolerance is yielding converged numerical solutions, in iterative methods. It also shows that the number of spectral grid points used is sufficient for such calculations.

We have also tried different discretizations of the equation and the boundary conditions, although not as systematically as the presented case. In general, using discretizations which lead to non-diagonally dominant coefficient matrices, increases the number of iterations to converge or, in some cases, they do not converge at all. The Jacobi method is the most sensitive to this, while all other iterative methods (Gauss-Seidel, SOR, SRJ and CJM) seem less affected by this issue. As an example, if just $u_0 = u_1$ is used for the inner boundary condition (consistent with Eq. (5.11)), the Jacobi method needs about 5 times more iterations in 1D, while all other iterative methods remain almost unaltered (only SOR shows differences for $N \leq 64$). This is an indication that the new method is not only faster than well-known iterative methods but can also be more robust than some of them.

5.2 Grad-Shafranov equation in spherical coordinates

The Grad-Shafranov (GS) equation [Grad and Hogan 1970, Shafranov 1958b] describes equilibrium solutions in ideal magnetohydrodynamics for a two dimensional plasma. It is of interest in studying the plasma in magnetic confinement fusion (e.g. Tokamaks), the solar corona and neutron star magnetospheres, among others.

In spherical coordinates (r, θ, φ) the magnetic field of an axisymmetric $(\partial_\varphi(.) = 0)$ plasma configuration can be expressed as

$$\boldsymbol{B}(r,\theta) = \boldsymbol{\nabla} \times \boldsymbol{A} = \frac{1}{r\sin\theta}\boldsymbol{\nabla}\Psi(r,\theta) \times \hat{\boldsymbol{e}}_{\boldsymbol{\varphi}} + \frac{F(r,\theta)}{r\sin\theta}\hat{\boldsymbol{e}}_{\boldsymbol{\varphi}}, \tag{5.15}$$

where $\boldsymbol{A}$ is the vector potential and $\hat{\boldsymbol{e}}_{\boldsymbol{\varphi}}$ is the unit vector in the φ direction. The flux function, $\Psi \equiv r \sin\theta A_\varphi$, is constant along magnetic field lines and is a measure of the poloidal magnetic field strength. The toroidal function, $F \equiv B_\varphi r \sin\theta$, is a measure of the toroidal field strength. Using Ampere's law, $\boldsymbol{J} = \boldsymbol{\nabla} \times \boldsymbol{B}$, being $\boldsymbol{J}$ the electric current, the flux function can be linked to the toroidal current as

$$\Delta^* \Psi \equiv \partial_{rr}\Psi + \frac{1}{r^2}\partial_{\theta\theta}\Psi - \frac{\cot\theta}{r^2}\partial_\theta \Psi = -J_\varphi r \sin\theta, \tag{5.16}$$

where Δ^* is the GS elliptic operator. For simplicity we consider here the case in which the inertia of the fluid can be neglected (magnetically dominated). In this case, if we impose force balance, $\boldsymbol{J} \times \boldsymbol{B} = 0$, the toroidal function depends on the flux function, $F(\Psi)$. As a result Eq. (5.16) leads to the GS equation

$$\Delta^* \Psi = -F(\Psi)F'(\Psi). \tag{5.17}$$

Not neglecting the inertia of the fluid leads to additional pressure terms, which are not considered here. A popular choice for the toroidal function is $F(\Psi) = C\Psi$, being C a constant. In this case the GS equation results in

$$\Delta^* \Psi + C^2 \Psi = 0, \tag{5.18}$$

which is a suitable elliptic problem to be solved with SRJ and CJM methods. Equation (5.18) resembles the Helmholtz differential equation in that it contains a Laplacian-like operator and a linear term in Ψ. Therefore, this test will show the ability of SRJ and CJM methods to handle more complicated elliptic operators. In addition we use this test to demonstrate the ability of iterative methods to handle boundary conditions imposed at arbitrarily shaped boundaries.

We compute the solution of Eq. (5.18) for two sets of boundary conditions, in the numerical domain $r \in [1, 10]$ and $\theta \in [0, \pi]$. In all cases we impose homogeneous Dirichlet conditions at $\theta = 0$ and $\theta = \pi$. In *test A* we impose Dirichlet boundary conditions at $r = 1$ and $r = 10$ with $\Psi = \sin^2\theta / r$. In the case $C = 0$, the solution for this test is a dipolar field. As the value of C is increased the solution results in a twisted dipole.

In *test B* we solve the GS equation in part of the domain, the region defined by

$$\begin{aligned} &r < \left(4.5\sin^2\theta + 2.5\sin^2(2\theta)\right)\left(1 - 0.4\cos(3\theta) + 0.3\cos(5\theta) + 0.05\sin(25\theta)\right) \\ &\& \\ &(r\sin\theta - 4)^2 + (r\cos\theta - 1.6)^2 < 1, \end{aligned} \tag{5.19}$$

Table 5.1 Number of iterations and computational time used by the SRJ and CJM methods with $N = 300$ and $P = 14$ to solve the GS equation, depending on the value of C.

test A	SRJ, $P = 14$		CJM	
C	iterations	computational time [s]	iterations	computational time [s]
0	5350	5.72	3400	3.67
0.01	5350	5.73	3400	3.78
0.1	5350	5.72	3400	3.70
0.2	4660	4.89	3400	3.76
0.3	7750	8.14	4620	4.98
0.31	9830	10.87	5850	6.27
0.32	13540	14.51	8020	8.64
0.33	21390	22.92	12550	13.44
0.34	49080	51.91	28740	30.35
test B	SRJ, $P = 14$		CJM	
C	iterations	computational time [s]	iterations	computational time [s]
0	3450	2.02	3400	2.12
0.01	3450	2.01	3400	2.08
0.1	3450	2.02	3400	2.08
0.5	3600	2.10	3400	2.07
1.0	3550	2.07	3400	2.07
1.3	3410	1.98	3400	2.07
1.4	3660	2.72	3400	2.06
1.45	4980	2.88	3400	2.09
1.47	11620	6.75	6710	4.00

inside the aforementioned numerical domain. The boundary of this region intersects the sphere $r = 1$ at $\theta_1 = 0.3037$ and $\theta_2 = 2.8903$. At $r = 1$ we impose Dirichlet boundary conditions with $\Psi = \sin((\theta - \theta_1)/(\theta_2 - \theta_1)\pi)^2$, and homogeneous Dirichlet conditions at the remaining boundaries. Imposing boundary conditions in arbitrarily shaped boundaries is straightforward when using iterative methods such as SRJ and CJM; we set $\Psi = 0$ everywhere outside the region (D.9) and apply the SRJ and CJM iteration only inside this region using a mask.

In both tests we use a second-order discretization of the GS equation and a numerical resolution of 300×300 equispaced grid points covering the numerical domain. We solve the equations using the SRJ method with weights corresponding to $N = 300$ and $P = 14$ and CJM. In both tests we initialize Ψ to zero in the whole domain. Table 5.1 shows the number of iterations and computational

Table 5.2 Number of iterations used by different methods to solve the GS equation (*test A*), for several values of C. In parenthesis the ratio of the number of iterations using Jacobi to the number of iterations, i.e. an estimation of the value of ρ.

C	0	0.2	0.32
CJM	3400 (72)	3400 (91)	8020 (144)
SRJ, $P = 14$	5350 (46)	4660 (67)	13540 (85)
SRJ, $P = 6$	9820 (25)	9180 (34)	22380 (52)
SOR	41050 (6.0)	57650 (5.4)	228100 (5.1)
Gauss-Seidel	132940 (1.9)	187560 (1.7)	788290 (1.5)
Jacobi	246260 (1)	310490 (1)	1153610 (1)

time to obtain a numerical solution with residual bellow 10^{-12}, depending on the value of C used. We have employed the same convergence criterion as in subsection 5.1. The results of the SRJ method in Table 5.1 are comparable to those presented in [Adsuara et al. 2016] and the number of iterations are identical to that work. The computational time differs because, although it was run in the same machine, there have been some optimisations in the implementation of the algorithm that makes it faster. We prefer to show here the results with the optimized code that can be directly compared with the results for CJM. We observe that the CJM is systematically faster than the SRJ ($P = 14$), specially in the case of large values of C. This is specially significative in the test B, in which the number of iterations is basically independent of C until is very close to its maximum value.

Table 5.2 compares SRJ method with $P = 14$ and CJM with the other iterative methods presented in the previous section, for the *test A*. In all cases the CJM is the fastest method, by a factor comparable to those obtained in Sect. 5.1. Again, the fact that using $P = 6$ results in about twice as many iterations for solving the problem as when employing $P = 14$ shows the advantage of employing SRJ schemes with the largest available number of levels.

The upper panels of Fig. (5.8) show the results for *test A*, for three different values of C, for the case of SRJ with $P = 14$ (results for CJM are similar). For the case $C = 0$, the analytical solution is $\Psi = \sin^2\theta/r$. In this case the maximum difference between the analytical and the numerical result, in absolute value, is 8.5×10^{-5}, which is consistent with the second-order discretization ($9/N^2 = 10^{-4}$). For $C = 0.1$ a toroidal component appears, but the flux function, Ψ, remains essentially the same. For higher values of C there is a tendency of the magnetic field lines to become more inflated, to support the increased magnetic tension due to the high magnetic field. In this regime the number of

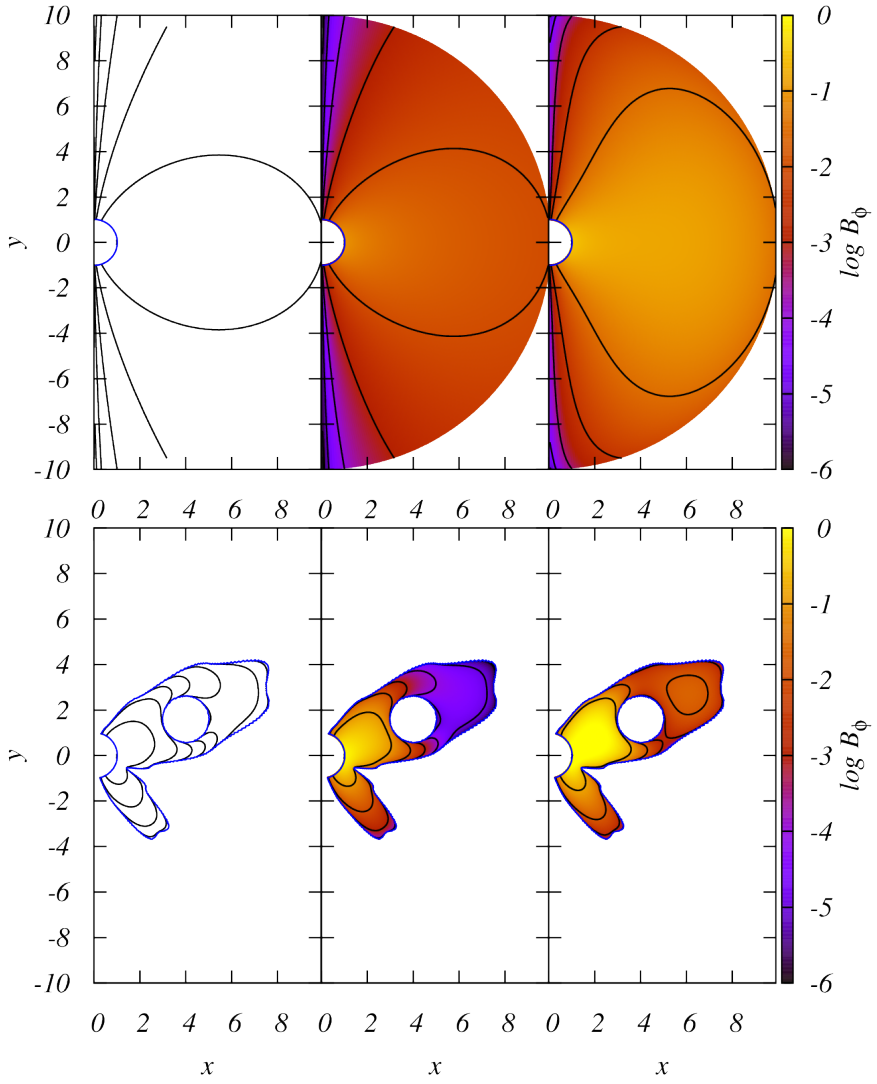


Figure 5.8 Numerical solution of the GS equation for *test A* (upper panels) and *test B* (lower panels) for different values of the constant C, using SRJ method with $P = 14$. From left to right $C = 0$, 0.1 and 0.34 (upper panels) and $C = 0$, 1.0 and 1.47 (lower panels). Isocontours of $\log \Psi$ (solid black lines), which coincide with magnetic field lines, are plotted in increments of 1. Colors show $\log B_\varphi$. For convenience we plot the (x, y) plane, being $x \equiv r \sin\theta$ and $y \equiv r \cos\theta$. Blue lines in lower panels show the boundary of the region in which the GS equation is solved.

iterations needed in the SRJ and CJM methods increases. We were not able to obtain solutions for values larger than $C = 0.34$. This is not a problem of the numerical method itself, since other methods (Jacobi, Gauss-Seidel, SOR) show similar behavior. The value $C \approx 0.35$ corresponds to an eigenvalue of the GS operator. For this case the matrix associated to the discretization of the GS equation is singular and hence it cannot be inverted. This is causing the convergence problems near this point.

Lower panels of Fig. (5.8) show the results for *test B*, for three different values of C, for the case of SRJ with $P = 14$ (results for CJM are similar). This case behaves qualitatively similar to *test A* but with more complicated geometry. The case $C = 0$ shows no toroidal field, which appears as C is increased. For $C = 1.0$ the flux function is still similar to that of the untwisted case, albeit slightly deformed. For $C = 1.47$, the maximum value that we were able to achieve, the topology of the field has changed, showing a region of close magnetic field lines in the upper right part of the domain. As in *test A*, the difficulty to achieve convergence for larger values of C is related to the presence of an eigenvalue of the GS operator. Note that the solution is everywhere smooth, and magnetic field lines (black lines) are tangent to the domain boundary (blue curve) as expected (except for $r = 1$ where non-zero Dirichlet boundary conditions are applied).

In general the SRJ and CJM methods shows reasonable rates of convergence and computational time to solve the problem with high accuracy, despite of the complicated boundary conditions. This renders a method which can be used in real applications of the GS equation with a good trade of excellent performance and ease of implementation.

Chapter 6

Parallel applications in astrophysics

The results of this chapter have been originally published in (or are submitted to):

On the equivalence between the Scheduled Relaxation Jacobi method and Richardson's non-stationary method,
JE Adsuara, I Cordero-Carrión, P Cerdá-Durán, V Mewes, MA Aloy,
Journal of Computational Physics 332, 446-460 (2017)

Speeding up a few orders of magnitude the Jacobi method: high-order Chebyshev-Jacobi over GPUs,
JE Adsuara, MA Aloy, , P Cerdá-Durán, I Cordero-Carrión
Submitted to Journal of Computational Physics (2017)

Although we have already mentioned several times throughout the thesis the enormous potential that seems to have our method to operate in parallel, we have not touched in detail this aspect of the algorithm so far. In this chapter we present the implementation of the CJM using a purely MPI implementation, an openMP / MPI hybrid implementation and using Graphics Processing Units (GPUs).

6.1 Some ideas about parallelism

According to the Flynn taxonomy for computer architectures [Flynn 1972], the von Neumann model [Neumann 1945], which is the main architecture used in modern computers, falls, basically, into the SISD (Single Instruction, Single Data stream) category. The processors in this category are able to perform actions sequentially. The two categories of the taxonomy that extend the model to parallel computing are SIMD (Single Instruction, Multiple Data stream) and MIMD (Multiple Instruction, Multiple Data stream). Although the SIMD architectures were the first to develop, they fell into disuse when the more general MIMD reached their full development in both shared memory and distributed memory machines.

Basically all supercomputers and clusters fall into the MIMD category. One of the most widely used technologies for parallel programming on shared memory machines is openMP. For distributed memory, where message passing is required, the reference technology for parallel applications is MPI.

Nevertheless, GPUs have resurrected the SIMD architecture. The graphics cards have been improving the processing of images. Their main advantage with respect to CPUs is that they apply the same graphic operation to many pixels at the same time, implementing vectorial operations. In this evolution, modern GPUs have abstracted the concepts of image and graphic operation to mesh and functions over its nodes, resulting in a very efficient vectorial processor for scientific calculation.

6.2 Purely MPI implementation: long gamma-ray bursts

6.2.1 Astrophysical scenario and equations

The gamma-ray bursts (GRBs) are one of the most energetic phenomena in the universe. The actual phenomenology allows for two different types of GRBs regarding its timescale: long (lGRB) and short (sGRB) [Kouveliotou et al. 1993]. Each of these two classes is linked to a different progenitor type: long-lasting bursts are indicative of compact stellar progenitors with enough energy in form of mass or angular momentum to drive GRB jets; short timescales suggest more compact systems such as the merger of two neutron stars (NSs) or of a neutron star and a black hole (BH).

Nowadays, one paradigm that explains the origin of most lGRBs is the collapsar model [Woosley 1993, MacFadyen and Woosley 1999]. In this model, a stellar mass BH results from the collapse of the massive core of the progenitor star. The BH is surrounded by a thick accretion torus, the accretion of which feeds both the BH and, depending on the dominant mechanism to tap the energy of the central engine, an ultrarelativistic jet.

The MRGENESIS code [Aloy et al. 1999, Leismann et al. 2005, Mimica et al. 2009] is a tool that allows the study of progenitors of GRBs. In order to do that, the code solves the equations of relativistic hydrodynamics (RHD). Assuming axisymmetric jets and flat space, MRGENESIS can use 2D spherical coordinates (polar coordinates) in Minkowski space-time. In the case of lGRBs, gravitational effects may become relevant so it is necessary to solve the fluid evolution coupled to the gravitational potential generated by a self-gravitating fluid. This gravitational potential is necessary, e.g., to balance pressure gradients, especially at the stellar surface, in order to reach hydrostatic equilibrium. We have included in MRGENESIS the computation of a Newtonian potential Φ to take into account self-gravity.

The hyperbolic system of conservation laws governing the motion of a relativistic fluid is (in natural units ($c = G = 1$) spherical coordinates)

$$\frac{\partial}{\partial t}\boldsymbol{U} + \frac{1}{r^2}\frac{\partial}{\partial r}(r^2\boldsymbol{F}) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}\boldsymbol{G} = \boldsymbol{S}, \tag{6.1}$$

where $\boldsymbol{U}$, $\boldsymbol{F}$ and $\boldsymbol{G}$ are vectors of conserved quantities and of fluxes in the radial and angular directions defined as follows:

$$\boldsymbol{U} = (D, S^r, S^\theta, \tau)^T, \tag{6.2}$$

$$\boldsymbol{F} = (Dv^r, S^r v^r + p, S^\theta v^r, (\tau + p)v^r)^T. \tag{6.3}$$

$$\boldsymbol{G} = (Dv^\theta, S^r v^\theta, S^\theta v^\theta + p, (\tau + p)v^\theta)^T. \tag{6.4}$$

The aforementioned conserved quantities are the relativistic mass density D, the momentum density $\boldsymbol{S} = (S^r, S^\theta)$, and the energy density τ, which are all measured in the Eulerian frame. Also the velocity $\boldsymbol{v} = (v^r, rv^\theta)$ is measured in this frame. These conserved quantities are related to quantities in the co-moving frame, the primitive variables, which are the rest-mass density ρ, the pressure p, and the four-velocity u^μ ($\mu = 0, \ldots, 3$) , through

$$D = \rho W, \tag{6.5}$$

$$\boldsymbol{S} = \rho h W^2 \boldsymbol{v}, \tag{6.6}$$

$$\tau = \rho h W^2 - p - D, \tag{6.7}$$

$$(v^r, v^\theta) = \frac{1}{W}(u^1, \frac{u^2}{r}), \tag{6.8}$$

where W is the Lorentz factor, h is the enthalpy, and

$$W = u^0 = \frac{1}{\sqrt{1 - \boldsymbol{v}^2}} \tag{6.9}$$

$$h = 1 + \varepsilon + p/\rho \tag{6.10}$$

being ε the specific enthalpy. Finally, in absence of physical sources (e.g. gravity), the source term $\boldsymbol{S}$ only contains geometrical terms due to the 2D spherical coordinates:

$$\boldsymbol{S} = \begin{pmatrix} 0 \\ \frac{1}{r}\left(2p + S^\theta v^\theta\right) \\ \frac{1}{r}\left(\frac{\cos\theta}{\sin\theta}p - S^\theta v^r\right) \\ 0 \end{pmatrix}. \tag{6.11}$$

In our implementation of the self-gravity of the fluid, as a first approximation and although the code is relativistic, we have chosen a Newtonian potential with some relativistic corrections (for example, we employ as source of the Poisson equation $\rho_{\text{eff}} := \rho h W^2 - p$ instead of ρ). As we keep the metric constant, the contribution of this potential only appears as an additional term $\boldsymbol{S}_{\text{pot}}$ in the source of the Eq. (6.1), i.e., $\boldsymbol{S}_{\text{new}} = \boldsymbol{S} + \boldsymbol{S}_{\text{pot}}$, where S refers to Eq. (6.11) and

$$\boldsymbol{S}_{\text{pot}} = \begin{pmatrix} 0 \\ -(\rho h W^2 - p)\frac{\partial}{\partial r}\Phi \\ -\frac{1}{r}(\rho h W^2 + \tilde{S}^r \tilde{v}^r)\frac{\partial}{\partial \theta}\Phi \\ -\rho h W^2 \left(\tilde{v}^r \frac{\partial}{\partial r}\Phi + \frac{\tilde{v}^\theta}{r}\frac{\partial}{\partial \theta}\Phi\right) \end{pmatrix}. \tag{6.12}$$

The Poisson's equation $\Delta\Phi = 4\pi\rho_{\text{eff}}$ defines the behavior of the potential Φ and its dependence on the mass distribution. We solve the Poisson's equation in spherical coordinates using the CJM.

The aim of the RHD simulations which are instrumental here [1] is not to study in detail the central engine of the GRB, which is explicitly excluded from our computational domain, but to study the propagation of a jet from well outside of the stellar core to its surface. Therefore, we excise the innermost region $r < R_0$ and model the jet engine as an energy and momentum input at

[1] In this thesis we are only concerned with the solution of the elliptic Poisson equation necessary to keep the stellar structure in hydrodynamic equilibrium. More details can be found in Chapter 4 of Cuesta 2017.

$r = R_0$. Our computational domain extends radially from an inner boundary at R_0 ($\gtrsim 10^9$ cm) to R_f, typically located well outside the star. To better impose boundary conditions at the innermost excised region, we do not solve directly the potential $\Phi(r,\theta)$ but a modified potential $\bar{\Phi}(r,\theta) = \Phi(r,\theta) + M_0/r$, where M_0 is the excised mass below R_0. Neumann boundary conditions for the potential are imposed at R_0, $\partial_r\bar{\Phi}|_{R_0} = 0$, and Dirichlet boundary conditions at the radial outermost end of the mesh, $\bar{\Phi}_{R_f} = -M_T/R_f$. The total mass M_T within the grid excludes the excised mass M_0. Once $\bar{\Phi}$ is calculated, we subtract in the radial direction the quantity M_0/r to recover the real potential Φ.

In the same way as in Sec. 5.1, we do not only need to calculate the potential once but it is necessary to recompute it during the time integration algorithm for the hyperbolic part governing the fluid evolution equations. In principle, it should be necessary to compute the gravitational potential at every time step of the hyperbolic part. However, in practice it is possible to recompute it less often to achieve a satisfactory accuracy in the overall calculation of a model. As a guide, we recompute the gravitational potential every n_r time steps, n_r being the number of cells in the radial direction. This guarantees that the update occurs within the light crossing time of the numerical grid ($\sim n_r\Delta t$). Finally, between two consecutive calculations a readjustment in the inner mass is needed due to the mass flux across R_0.

In the test presented here, we do not consider the whole simulation but we present results on the performance of the newly developed methods for one of these recalculations of the gravitational potential, under conditions which are comparable to those encountered in the full simulation.

6.2.2 Results

Our aim is to assess the performance of the CJM, similarly to previous tests presented in this thesis, in a real astrophysical application. In particular, we are interested in the performance of a parallel implementation of the method. The results presented in this section use a MPI parallelization implementation of the algorithm.

In this test, we use the classical 5-points stencil in polar coordinates shown in figure Fig. 6.1. The problem, as we have presented above, includes boundary conditions of mixed type. Therefore, the optimal scheme presented in Sec. 4 is not necessarily optimal for this case. However, as we shall shown below, it is sufficiently well behaved to be useful for these computations.

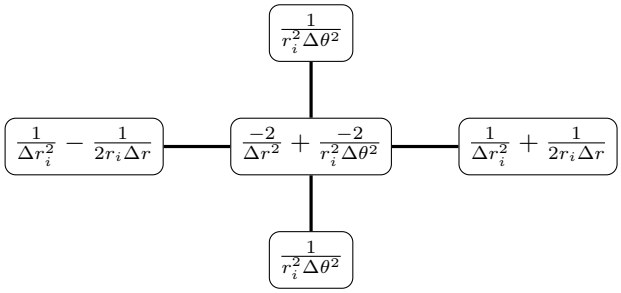


Figure 6.1 Schematic representation of the classical 5-points discretizations of the spherical Laplacian in two spatial dimensions. Each box denotes a neighboring point of a given one (i, j), represented by the central box. The value enclosed by each box corresponds to the coefficient of the discretizations. See Chap. 3 for details.

We have done the tests on two machines, one with distributed memory (CT), the type of machine where MPI codes are usually executed, and one with shared memory (SV). The CT machine is a cluster with 2016 cores. It consists of 504 nodes with 4 cores (2 sockets dual core) PowerPC 970MP working at 2.3GHz and 8 GB per node of RAM. Communication between them is done through a Myrinet network, which has a bandwidth of 2 Gb full duplex. The SV is a supercomputer with 384 cores and 2 TB of RAM memory. It is an Altix UltraViolet 1000 made by Silicon Graphics (SGI). It has 32 blades dual socket Xeon X7542 hexacore Nehalem working at 2,67 GHz. It has 64 GB per blade and 32 GB per socket, however all the memory is shared by all the processes and the communication between blades works over the NUMAlink (5) protocol of SGI. The communications are expected to be faster in SV than in CT. However, in SV the computing time may be affected by the load of the system, while in CT we have exclusive use of the processors, as long as we use multiples of 4, i.e. occupying entire blades.

We have performed series of simulations for three different resolutions. In the CT machine, we have a low-resolution model, with a mesh of size $n_r \times n_\theta = 1024 \times 64$, and a high resolution model, with 16384×256. In the SV machine the size of the mesh is 15000×270.

In the MPI parallelization model, we make a partition of the full domain into subdomains, which are then distributed among processors. Each processor communicates the values at the boundaries of its corresponding mesh to its neighbors. The efficiency of the parallelization scheme, does not only depend on the number of subdomains created, but also on the shape of these domains.

To test the dependency of the parallelization efficiency on the number of processors and the shape of the subdomains, we perform simulations using MPI

partitions with all possible common divisors of the two dimensions without exceeding the maximum of available number of cores in the machine or in the execution queue. We define the number of processors as N_{CPU}, and the number of subdivisions in the $r-$ and $\theta-$ direction as N_r and N_θ, respectively, such that $N_{\mathrm{CPU}} = N_r N_\theta$. In the case of CT, both resolutions tested are powers of two. Therefore, we perform simulations with partitions of $N_r \times N_\theta \in D_{\mathrm{CT_r}} \times D_{\mathrm{CT}_\theta}$, being $D_{\mathrm{CT_r}} = \{2^i : i = 1, \ldots, 8\}$ and $D_{\mathrm{CT}_\theta} = \{2^i : i = 1, \ldots, 6\}$ and $N_{\mathrm{CPU}} \leq 256$, which is the maximum number of cores available in the execution queue. They are chosen in this way to ensure that half of the combinations are multiples of four and therefore we will fill whole blades, which is the best choice from the point of view of efficiency, in the CT machine. The other half of possible partitions will help us evaluate the impact of sharing blades with other processes. In the case of SV, the chosen partitions are $N_r \times N_\theta \in D_{\mathrm{SV_r}} \times D_{\mathrm{SV}_\theta}$, being $D_{\mathrm{SV_r}} = \{1, 2, 3, 5, 6, 10, 15, 30, 60\}$ and $D_{\mathrm{SV}_\theta} = \{1, 2, 3, 5, 6, 10, 15, 30, 45\}$. In this case we use, at most, with 100 cores, i.e., $N_{\mathrm{cpu}} < 100$. As an example, the partition $(16, 4)$, which is among the valid ones in CT, involves N_{CPU} =64 cores. For the high-resolution mesh, we typically partition the radial direction in 16 domains and the angular direction in 4 domains, so each core is dealing with a mesh of size 1024×64.

Our approach is to solve the problem in the whole grid with an iterative method (CJM), as if it were a single grid but distributing the load of each subdomain among processors. This forces us to communicate the values at the boundaries of the subdomains in each step of the iteration procedure. This procedure involves an intense communication of MPI messages among processors, which may potentially limit the performance and scalability of the scheme. One of the objectives of this test is to check if this is the case. There could be alternative approaches to avoid such intense communication, e.g. by performing the iteration only in the subdomains and communicating the boundaries less often using, e.g., a block Jacobi method. These alternatives, that should be studied in the future, are not considered here for the sake of simplicity.

In the same way we did in the sequential case, for each case involving the CJM we perform two calculations: *ab initio calculations*, that mimic what occurs at the start of the simulation, and *realistic calculation*, more faithful to what really happens when we recalculate the metric throughout the simulation (see Sec. 5.1 for details).

In addition to the CJM and for comparison purposes, we also perform the calculations using an alternative method that was already implemented in the code. The alternative method (*direct inversion method* hereafter) solves the

elliptic problem in each subdomain using a direct method (LU decomposition as described on Sec. 5.1) and then performs a block iteration over the subdomains until convergence at the domain boundaries is achieved. For the block iteration we use P. L. Lions' method [Lions 1990], which is a generalisation of the block Jacobi method but using Robin boundary conditions at the boundaries between domains.

We focus in two aspects of the simulations: the execution time and the speed up. To compute the execution time, we only consider the part of the code involved in the computation of the solution, without including memory allocation or initialization, which is a measure of the absolute speed of the method. To determine the speed up, we compute the ratio of the execution time with a certain number of cores to the execution time with a single core (sequential execution). The speed up gives us information about the relative speed of the method with respect to the number of processors, which is interesting to determine the scalability of the algorithm.

In the Fig. 6.2, we show the execution time for each of the two methods, to solve the problem at hand. The time for all the possible partitions mentioned above is displayed. The plot shows the results for the CT machine with the high-resolution mesh and using the *realistic calculation* setup. Other resolutions and computations in the SV machine behave in a qualitatively similar way. Comparing the direct inversion method with the CJM, the first is much more sensitive to the way the partition of the subdomain is done, while the second is rather insensitive. In the direct inversion method, more elongated subdomains give smaller execution times. This is because the LU decomposition algorithm used, scales as $\sim n_r^{(\mathrm{s})}(n_\theta^{(\mathrm{s})})^2$ (for $n_r^{(\mathrm{s})} > n_\theta^{(\mathrm{s})}$), where $n_r^{(\mathrm{s})} = n_r/N_r$ and $n_\theta^{(\mathrm{s})} = n_\theta/N_\theta$ are the radial and angular extent of the grid at each subdomain, respectively. Hereafter and unless it is indicated otherwise, we plot only results for the best possible partition.

In Figs. 6.3, 6.4 and 6.5, we present the results for the two methods in the CT machine and the SV machine, respectively, for the different resolutions used, and the different setups (*ab initio* and *realistic* calculation).

In the CT machine (Fig. 6.3) and for the low-resolution simulations, it is not efficient to use more than approximately 16 processors. Beyond that number of processors the execution time increases in either method due to the "insufficient" computational load per subdomain. For the high resolution simulation (Fig. 6.4), it is advantageous to use the method up to 128 processors. In this machine, the direct inversion method is always faster than the CJM. However, we must take into account two details that soften this apparent superiority. Firstly, the best

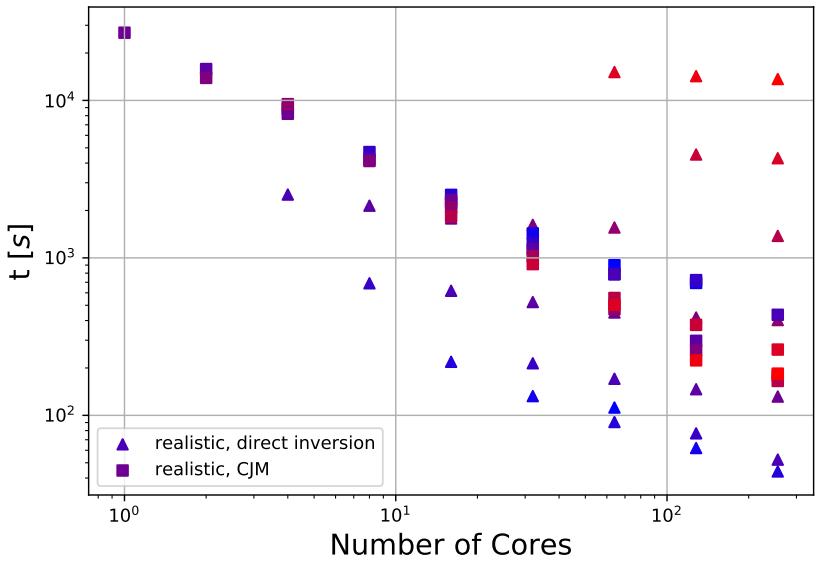


Figure 6.2 Execution time using the direct inversion method (triangles) and the CJM (squares). The color goes from red to blue, passing through purple. Red indicates partitions with more cells in the angular direction than in the radial direction. Blue indicates partitions with more cells in the radial direction than in the angular direction. Purple indicates partitions close to the square shape. A resolution of 16385×256 cells (high-resolution model) is used in the CT machine. The execution time of the direct inversion method is much more sensitive to the shape of the subdomains than the CJM.

LU times are being achieved with partitions with very extreme aspect ratio, in which $N_r = 1$. This would be fine if the elliptic solver was not coupled to a hydrodynamic code. However, the same partition used in the elliptic solver has to be used in the hydrodynamics part, to avoid excessive transfer of data among processors, and $N_r = 1$ is not the optimal partition for the hydrodynamics part of the code. The optimal partition, i.e. that minimizing the total execution time of both the hydrodynamics and the elliptic par, is likely to be different to the one presented here and will render execution times closer to CJM. Secondly, we have extrapolated the execution times for larger number of processors, by means of a least squares fit in both methods, and we obtain that, for the high resolution simulation, from 3000 processors onwards, the execution time using the CJM is expected to be lower than the direct inversion method. These results are qualitatively similar in both *ab initio* and the *realistic* calculations.

The results for the SV machine (only high resolution simulation, see Fig. 6.5) are similar. We do not appreciate the saturation of the execution time with the

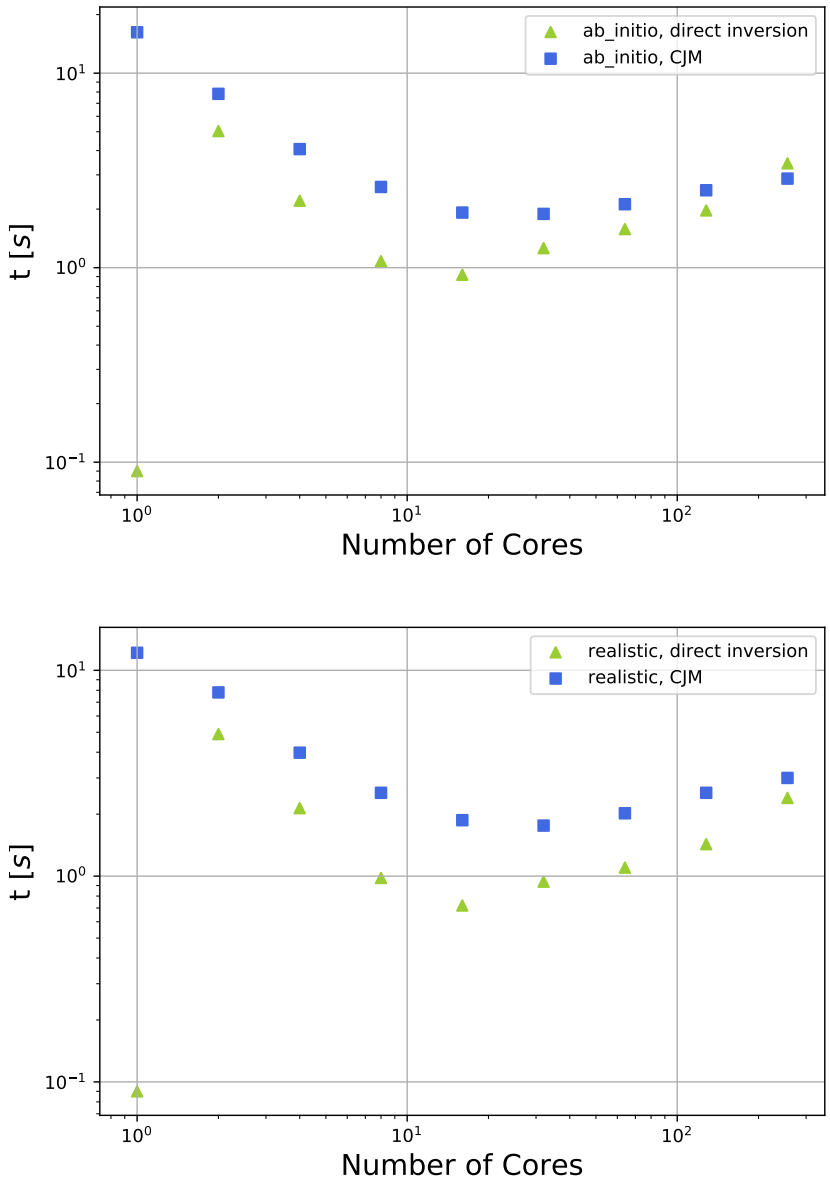


Figure 6.3 Execution time on the CT machine depending on the size of the problem $n_r \times n_\theta$, using the two methods: direct inversion and CJM. Upper and lower panels show *ab initio* and *realistic calculation*, respectively. Both panels correspond to the the low-resolution test.

number of processors for the CJM method, which shows a qualitatively similar behavior in both *ab initio* and *realistic* calculations. Although the direct method

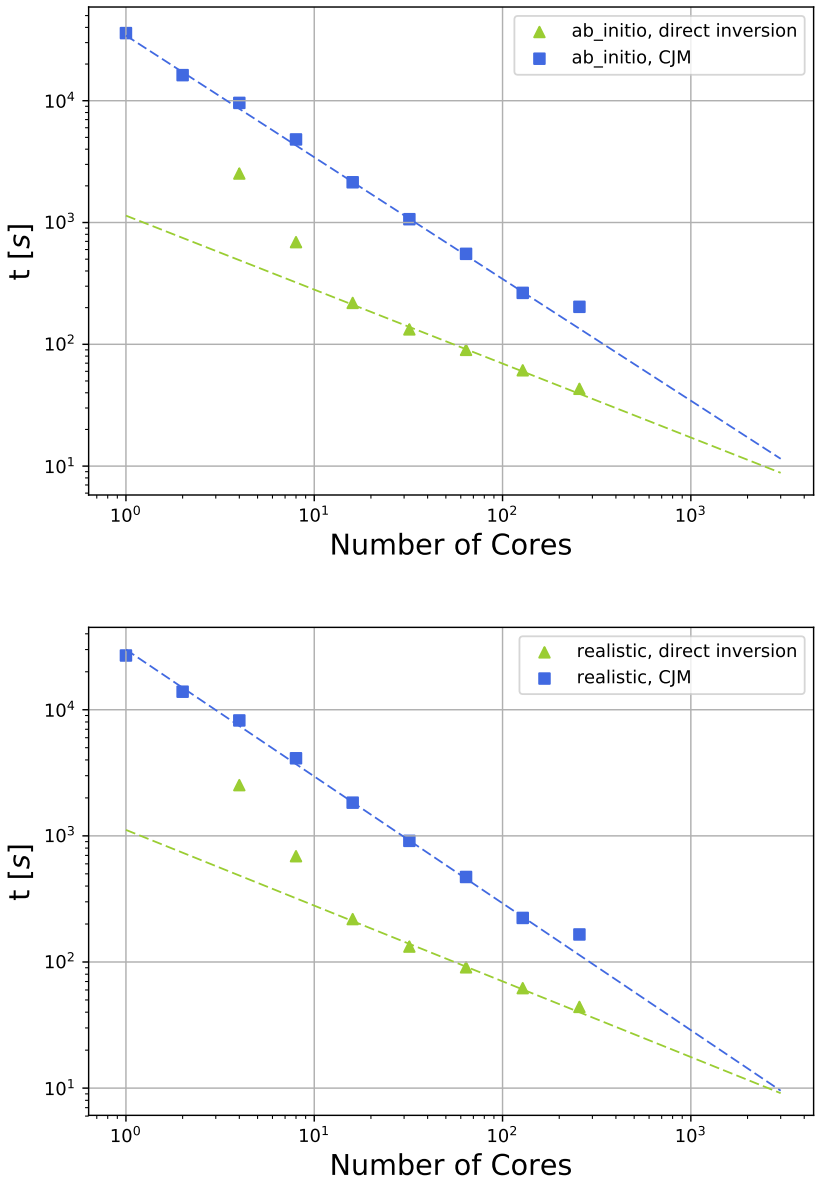


Figure 6.4 Same as Fig. 6.3 but for the high-resolution test.

is faster in some cases, this is more an exception than the rule, since in general it shows a large dispersion. This is an artefact of how the partitions are done. As commented above, for the direct inversion method the optimal partition is

that with $N_r = 1$, i.e $N_r \times N_\theta = 1 \times N_{\rm CPU}$. This implies that the optimal case can only be used if the number of angular grid points, n_θ, is divisible by $N_{\rm CPU}$, which is not always the case. The best results in the plot are those in which this division is possible. In the rest of the cases the partition is not optimal because $N_r > 1$. For example, if we want to use 4 processors, we can not use the partition $(1, 4)$ because 4 does not divide 270. In the latter case, we shall resort to the partition $(2, 2)$, which does not have the optimal aspect ratio and, hence, exhibits a considerably longer execution time. We conclude that, although the direct inversion method showed the best performance under the optimal conditions, the CJM is more flexible and tolerates better the changes in the shape of the subdomains.

Finally, we consider the speed up of our method in the CT machine for the low- and high-resolution test (Figs. 6.6 and 6.7), respectively. The ideal case would be a speed up equal to $N_{\rm cpu}$ (orange line). Similarly as with the execution time, the speed up saturates at about 16 and 128 cores for the low- and high-resolution calculation, respectively. In all cases, the speedup of the CJM method is better (higher) than that of the direct inversion method. In the CJM the execution time saturates for a larger number of processors and, when saturated, the decline is less abrupt. This behaviour occurs both in low- and high-resolution simulations.

6.3 OpenMP/MPI hybrid code: static field in 3D

Here we test the CJM and the predictability of the residual evolution in a three-dimensional (3D) elliptic equation with a source term. For this test, we use infrastructure provided by the Einstein Toolkit `Einstein Toolkit`, Löffler et al. 2012. The actual calculation consists on finding the static field of a uniformly charged sphere of radius R in 3D Cartesian coordinates subject to Dirichlet boundary conditions. For that we solve the Poisson equation:

$$\Delta\phi(x, y, z) = -4\pi\rho_{\rm e}, \tag{6.13}$$

where $\rho_{\rm e} = 3Q/(4\pi R^3)$ and Q is the charge of the sphere. We solve the elliptic equation (D.22) with a standard second-order accurate 7-point stencil in a

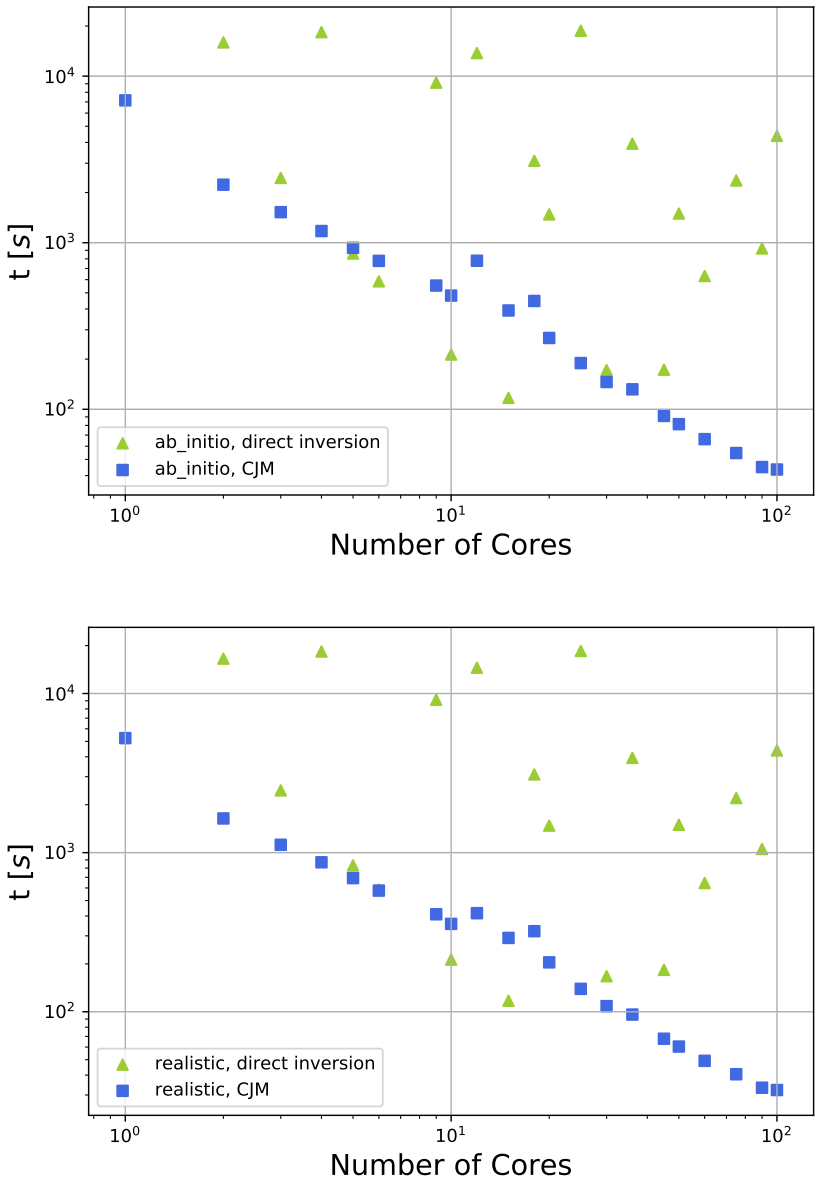


Figure 6.5 Execution time on the SV machine for a high resolution simulation (15000×270) using the two methods: direct inversion and CJM. Top and bottom panels show *ab initio* and *realistic calculations*, respectively. The dispersion in the execution times for the direct inversion is caused by the impossibility of choosing the optimal partition depending in the number of processors.

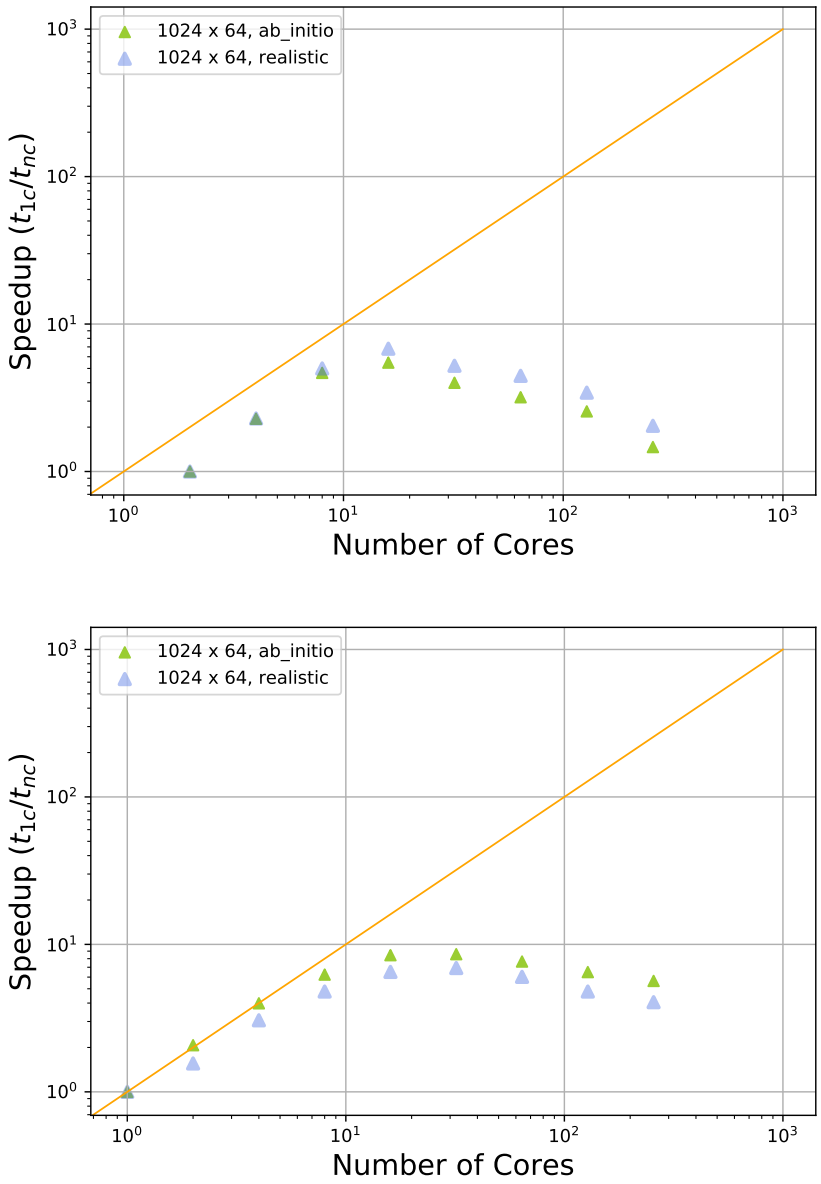


Figure 6.6 Speedup of the two methods considered on the CT machine for the low-resolution test. Upper and lower panels show the direct inversion method and CJM, respectively. Both *ab initio* and *realistic* are shown.

uniform mesh with grid spacings $h_x = h_y = h_z = h$

$$\Delta u_{ijk} = \frac{1}{h^2}\Big[u_{i-1,jk} + u_{i+1,jk} + u_{i,j-1,k} + u_{i,j+1,k} + u_{i,j,k-1} + u_{ij,k+1} - 6u_{ijk}\Big]. \tag{6.14}$$

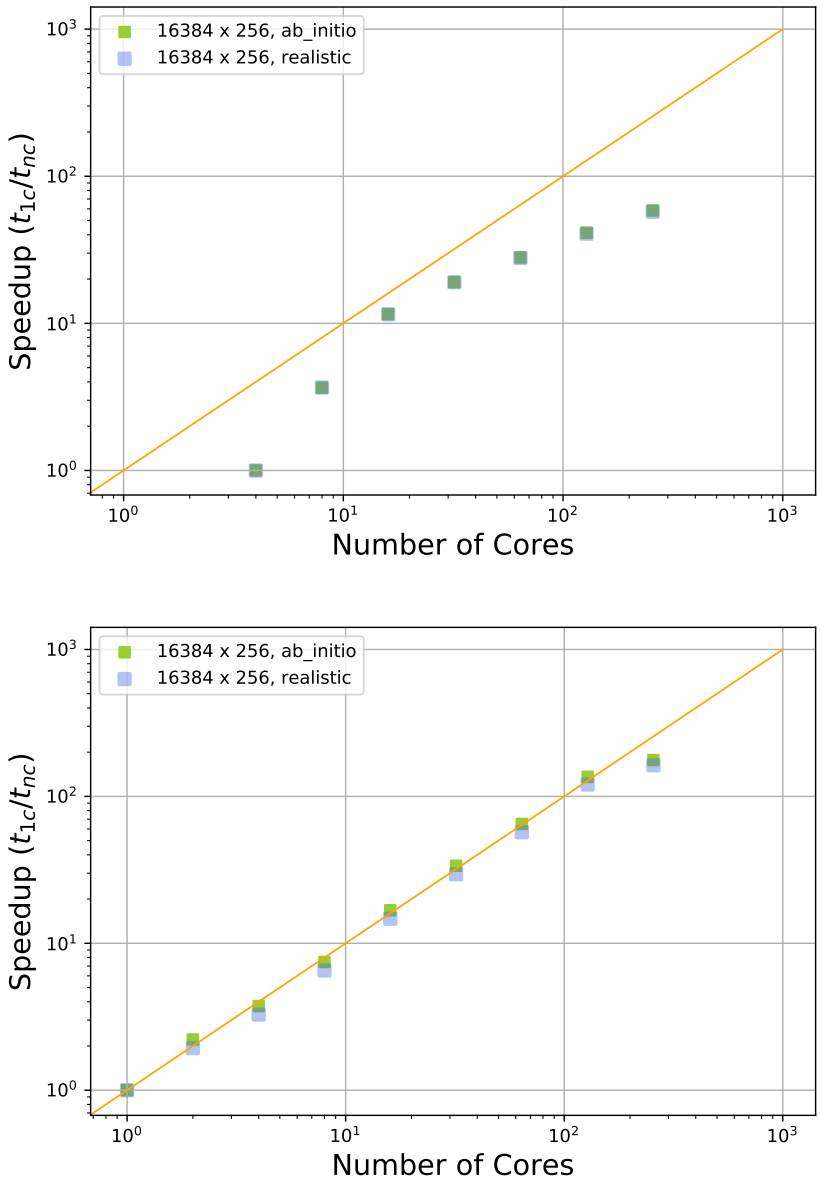


Figure 6.7 Same as Fig. 6.6 but for the high-resolution test.

We consider two different grid sizes with $N_x = N_y = N_z = N = 128$ and $N_x = N_y = N_z = N = 256$ points and the following iterative methods: Jacobi, Gauss-Seidel (SOR with $\omega = 1$), SOR with the optimal relaxation factor

$\omega_{\rm opt} = 2/(1 + \sin(\pi/N))$, and CJM with the optimal sequence of weights for a given resolution. The results for the two grid resolutions are shown in Fig. 6.8. Both SOR and CJM (slightly less than twice the number of iterations of SOR) are more than an order of magnitude faster than the Jacobi and Gauss-Seidel methods. While the CJM is not as fast as SOR when using the optimal relaxation factor $\omega_{\rm opt}$, we note here two arguments that should favor the use of the CJM over SOR: Firstly, Young's theory of relating $\omega_{\rm opt}$ to the spectral radius of the Jacobi iteration matrix $\rho(J)$ via $\omega_{\rm opt} = 2/(1 + \sqrt{1 - \rho(J)^2})$ only applies when the original matrix of the linear system $Au = b$ is consistently ordered. Secondly, the CJM is trivially parallelized, while SOR requires multicolor schemes for a successful parallelization, as we have shown in Sec. 4.4 presenting results for 9-point and 17-point Laplacians in 2D.

Next, we solve equation (D.22) subject to reflection symmetry (homogeneous Neumann boundary conditions) at the $x = 0$, $y = 0$, $z = 0$ planes (so-called octant symmetry) with $N_x = N_y = N_z = N = 64$ points, using the same iterative methods as before. For the CJM, we choose the same sequence of weights as those we used for the full 3D domain using $N = 128$ points. Because of the boundary conditions used to impose octant symmetry, the resulting matrix A is non-consistently ordered and hence there is no analytic expression to calculate $\omega_{\rm opt}$ for SOR; in this case we test a sequence of values of ω to empirically estimate its optimal value for the given problem. The residuals of the different iterative methods are shown in Fig. 6.13. The CJM now performs better than SOR for any ω we have tested. Furthermore, as seen in the plot, SOR is very sensitive to the exact value of ω that is chosen, as it is well known. The CJM is free of this need to estimate and choose a sensitive parameter.

6.4 Speeding up a few orders of magnitude the Jacobi method: high-order Chebyshev-Jacobi over GPUs

In this section we show how to reach a remarkable speed up when solving elliptic partial differential equations with finite differences thanks to the joint use of the Chebyshev-Jacobi method with high-order discretizations and its parallel implementation over GPUs.

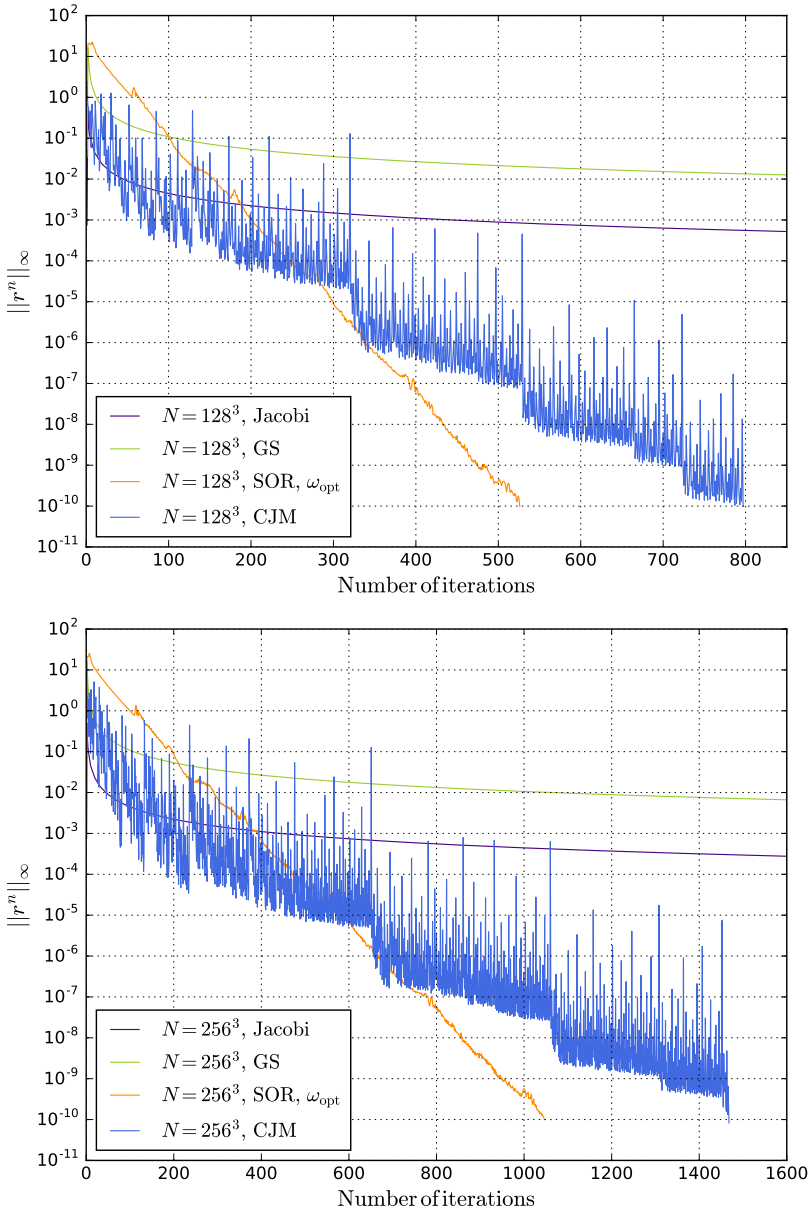


Figure 6.8 Evolution of the residual for the solution of the Poisson equation (D.22) in 3D, with $N = 128$ (upper panel) and $N = 256$ (bottom panel) for different iterative methods. The apparent faster convergence of Jacobi relative to Gauss-Seidel is due to the shown small interval of iterations which was chosen to highlight the convergence behavior of CJM relative to SOR. Gauss-Seidel actually reaches the desired tolerance faster than Jacobi in all examples, as expected.

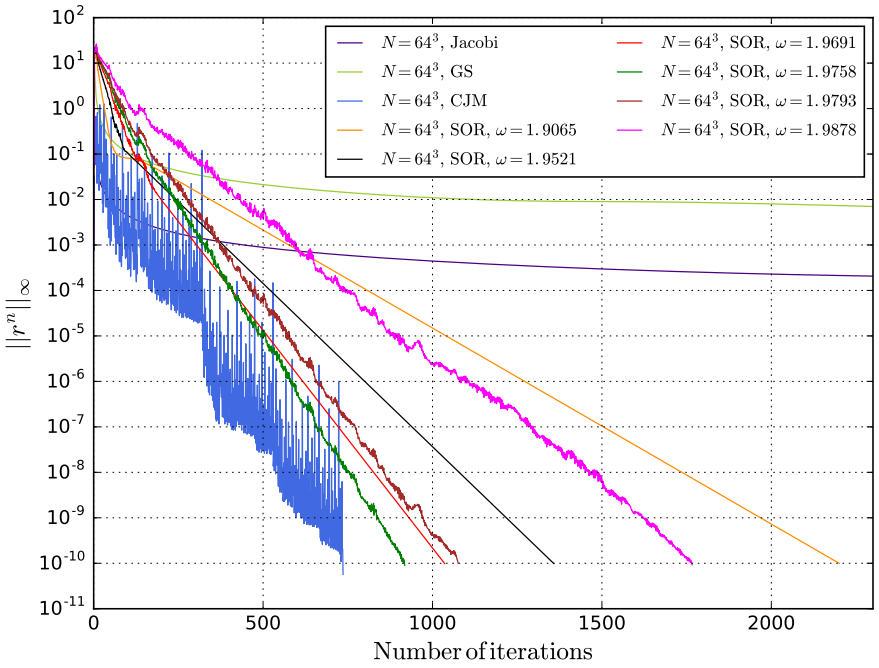


Figure 6.9 Same as Fig. 6.8 but for the Poisson equation (D.22) in 3D using octant symmetry, with $N = 64$ points per dimension.

6.4.1 The CJM with a high-order discretization of the Laplacian

In Chap. 4 we presented two parametrized discretizations of the Laplacian in 2D, which can achieve more than second-order accuracy. Here we use one discretization of each of these two families. From the first family, we use a 9-point discretization corresponding to a value of the parameter of $\alpha = \frac{2}{3}$ (Chap. 4), whose Laplacian is discretized as in Eq. (4.21) and whose $\kappa^{(9)}_{\rm min}$ and $\kappa^{(9)}_{\rm max}$ values[2], needed for obtaining the scheme and related with the size of the mesh, are given by Eq. (4.25) The grid assumed in this discretization is uniform, so that $L_x/N_x = L_y/N_y := h$, L_x and L_y being the sizes along the $x-$ and $y-$ directions, respectively.

[2]The superscript (9) refers to the number of points in which the Laplacian operator is discretized. Later in the text we will use also subscript (17) to refer to the values of $\kappa_{\rm min}$ and $\kappa_{\rm max}$ used in the 17-point discretization of the Lagrangian.

From the second family of discretizations, we use a 17-point discretization with $\alpha = \frac{2}{3}$ (Eq.(4.26)), whose discrete representation of the Laplacian reads

$$\begin{aligned}\Delta u_{ij} = \frac{1}{72h^2}\Big[&-4u_{i-2,j} + 64u_{i-1,j} + 64u_{i+1,j} - 4u_{i+2,j} \\ &-4u_{i,j-2} + 64u_{i,j-1} + 64u_{i,j+1} - 4u_{i,j+2} \\ &-u_{i-2,j-2} + 16u_{i-1,j-1} + 16u_{i+1,j+1} - u_{i+2,j+2} \\ &-u_{i-2,j+2} + 16u_{i-1,j+1} + 16u_{i+1,j-1} - u_{i+2,j-2} - 300u_{i,j}\Big]. \end{aligned} \tag{6.15}$$

and whose corresponding $\kappa_{\min}$ and $\kappa_{\max}$ values are

$$\begin{aligned}\kappa_{\min}^{(17)} = &-\frac{4}{75}\left[\sin^2\frac{\pi}{N_x} + \sin^2\frac{\pi}{N_y}\right] + \frac{64}{75}\left[\sin^2\frac{\pi}{2N_x} + \sin^2\frac{\pi}{2N_y}\right] \\ &-\frac{1}{75}\left[\sin^2\left(\frac{\pi}{N_x} + \frac{\pi}{N_y}\right) + \sin^2\left(\frac{\pi}{N_x} - \frac{\pi}{N_y}\right)\right] \qquad (6.16) \\ &+\frac{16}{75}\left[\sin^2\left(\frac{\pi}{2N_x} + \frac{\pi}{2N_y}\right) + \sin^2\left(\frac{\pi}{2N_x} - \frac{\pi}{2N_y}\right)\right] \qquad (6.17)\end{aligned}$$

$$\kappa_{\max}^{(17)} = \frac{128}{75}. \tag{6.18}$$

In Figure 6.10 we show an schematic view of the two stencils for the two different discretizations along with the numerical coefficients corresponding to each node. Note that the value of the discrete Laplacian operator evaluated at the central node is a weighted sum over all the nodes schematically represented in each of the figure panels. The purpose of these schemes is showing that given a central point, the Laplacian discretization is fully specified by providing a list of M numerical coefficients enclosed in all the surrounding nodes.

6.4.2 GPUs

Modern GPUs have become extremely powerful processing units, which can accelerate enormously the computation of heavy mathematical operations. However, for that to happen, the algorithms must be properly vectorized and the pipeline of arithmetic operations and data transfers from/to the main memory must be optimized. The CJM is perfectly suited for its porting to GPUs. In every iteration of the method the approximate solution is stored in each node and we must apply the same set of elementary arithmetic operations to all of them.

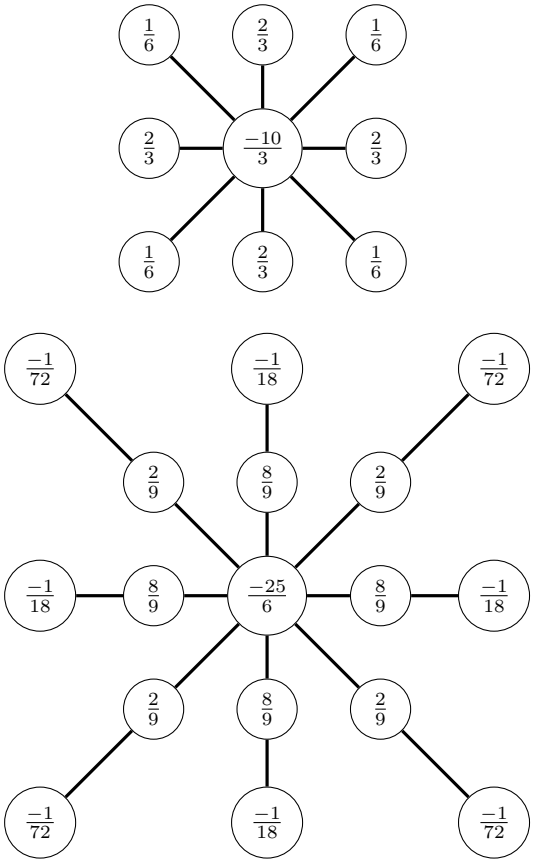


Figure 6.10 Schematic representation of the 9-points and 17-point discretizations of the Laplacian in two spatial dimensions, corresponding to a value $\alpha = \frac{2}{3}$. Each circle corresponds to a neighboring point of a given one (i, j), represented by the central circle. The value enclosed by each circumference corresponds to the coefficient of the discretizations shown in Eqs. 4.21 and 6.15 multiplied by h^2. See Chap. 4 for details.

6.4.3 CUDA devices

We present in this subsection the specific devices we used. We aim to test more than a single GPU model in order to properly assess the scaling properties of our implementation of the CJM, which is based on the CUDA *CUDA* technology of NVDIA. CUDA devices can be characterized by two parameters (among others): the Compute Capability (CC) and the number of Streaming Multiprocessors (SMs). The CC of a device is represented by a version number. It comprises a major revision number X and a minor revision number Y, and it is denoted with the format "X.Y". Devices with the same major revision number belong to the same core architecture.

CPU	Device	Architecture	CC	SMs
Opteron (Kp)	Tesla K40c	Kepler	3.5	15
Intel (Mx)	GeForce GTX Titan X	Maxwell	5.2	24

Table 6.1 Properties of the two GPU architectures in which the CJM has been tested. We note that the memory on board (12 Gb) is the same for both GPUs.

We have used two CUDA devices: a Tesla K40c and a GeForce GTX Titan X, with CC values of 3.5 and 5.2, respectively. We note that the major revision numbers are different. They correspond to two distinct architectures, Kepler and Maxwell, with major revision numbers 3 and 5, respectively. These GPU devices are connected to two different computers, having CPUs Dual-Core AMD Opteron 2222 working at a frequency of 3 GHz and Intel 7-4820K with a working frequency of 3.70 GHz, respectively. We will refer to the CPUs of these systems as Kp and Mw, respectively. In our first device we have 15 SMs, while in the second one we have 24 SMs. We have decided to use both types of hardware because, although the Maxwell microarchitecture is more recent and, a priori, better in many technical aspects, it seems that double precision arithmetics (required for practical applications of the CJM) works better on Kepler than on Maxwell. Table 6.1 summarizes the relevant properties of the two architectures employed in this section.

6.4.4 Code

In order to execute the Jacobi method and the CJM over GPUs, we have developed two CUDA kernels, i.e., sections of code that instead of running on the CPU run on the GPU device. We need to transfer the data that the kernel will use from the memory of the host to that of the device. Once the results are computed, they are transferred back from the device to the host. The 12GB of memory of our devices is large enough to allow us to transfer the whole data structure of the problem to the GPU memory in one transfer at the beginning of the calculation. Once the problem has been solved to the desired accuracy (fully on the GPU device), we recover back the solution in the CPU also with a single data transfer. We have executed the same code in the two considered GPU architectures.

The code is of stencil type, i.e., each node in the grid requires of knowing the values of the variable, whose solution we seek, at a certain number of neighbors as Fig. 6.10 schematically shows for the cases in which the Laplacian operator is discretized in 9 or 17 points. In the most general case we have

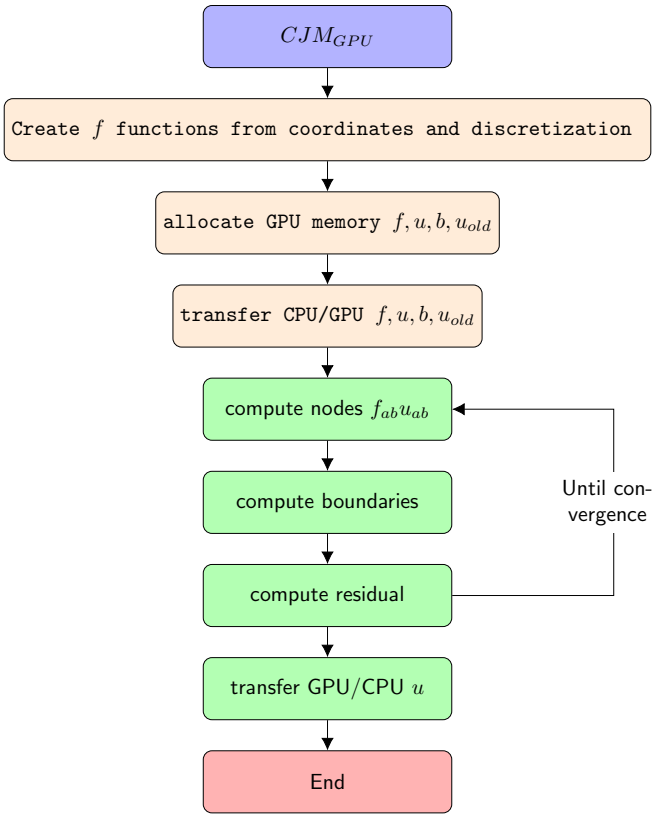


Figure 6.11 Flow chart of the code. b stands for the source term (if any) of the elliptic partial differential equation. Green background rectangles correspond to execution blocks in the GPU device. The rest of the execution blocks take place on the host computer.

$f_{\mathrm{NW}}^{(x_{1,i},x_{2,j},\Delta x_{1,i},\Delta x_{2_j})}$	$f_{\mathrm{N}}^{(x_{1,i},x_{2,j},\Delta x_{1,i},\Delta x_{2,j})}$	$f_{\mathrm{NE}}^{(x_{1,i},x_{2,j},\Delta x_{1,i},\Delta x_{2,j})}$
$f_{\mathrm{W}}^{(x_{1,i},x_{2,j},\Delta x_{1,i},\Delta x_{2,j})}$	$f_{\mathrm{C}}^{(x_{1,i},x_{2,j},\Delta x_{1,i},\Delta x_{2,j})}$	$f_{\mathrm{E}}^{(x_{1,i},x_{2,j},\Delta x_{1,i},\Delta x_{2,j})}$
$f_{\mathrm{SW}}^{(x_{1,i},x_{2,j},\Delta x_{1,i},\Delta x_{2,j})}$	$f_{\mathrm{S}}^{(x_{1,i},x_{2,j},\Delta x_{1,i},\Delta x_{2,j})}$	$f_{\mathrm{SE}}^{(x_{1,i},x_{2,j},\Delta x_{1,i},\Delta x_{2,j})}$

f_{NW2}	f_{NWN}	f_{N2}	f_{NEN}	f_{NE2}
f_{NWW}	f_{NW}	f_{N}	f_{NE}	f_{NEE}
f_{W2}	f_{W}	f_{C}	f_{E}	f_{E2}
f_{SWW}	f_{SW}	f_{S}	f_{SE}	f_{SEE}
f_{SW2}	f_{SWS}	f_{S2}	f_{SES}	f_{SE2}

Table 6.2 Notation for the functions of neighbors at a distance one and two.

implemented, both the central node, identified with the integer indices (i, j), where $1 \leq i \leq N_x$ and $1 \leq j \leq N_y$ and each of its (at most) 24 neighbors spanned by the discretization of the Laplacian can have different numerical factors weighting their contributions (see, e.g., the values of such coefficients enclosed in the circles shown in Fig. 6.10, which correspond to the simplest case of a two-dimensional problem in Cartesian coordinates). These numerical factors may change as a function of the position of the central node of the discretization in the grid. We refer to each of these numerical factors with the terminology sketched in Tab. 6.2. The numerical factor corresponding to the central node at a position (i, j) is annotated by $f_{\mathrm{C}}^{(x_{1,i}, x_{2,j}, \Delta x_{1,i}, \Delta x_{2,j})}$, as it may depend on the general coordinates $(x_{1,i}, x_{2,j})$ and on the grid spacing between consecutive nodes around the central node $(\Delta x_{1,i}, \Delta x_{2,j})$, since we allow for non-uniform meshes. The rest of the numerical factors are named using the cardinal points and numbers (see Tab. 6.2). To improve the efficiency and to deal more easily with the boundary conditions, in the cases in which the Laplacian is discretized in up to 9 points, we have created a different data structure for this case (upper part of Tab. 6.2), which we differentiate from the most generic case represented by the lower part of Tab. 6.2.

As a result of this, the code is totally generic regarding discretization and coordinates: the choice of coordinates determines the discretization of the Laplacian operator, and this discretization determines the mask of the functions (see Table 6.3). In addition, although the kernels are quite generic, the code allows particular instantiations to the different stencils in a simple and optimal way (without saving repeated values, without operating the zeros, etc.).

Our implementation also takes advantage of memory hierarchy on the GPU device, since the kernel uses both the Shared Memory and the Registers on board of the CUDA device. Employing the CUDA Occupancy Calculator (free tool provided by Nvidia), we have tuned a number of parameters of the developed kernel in order to maximize the occupancy of the GPU devices. After some experimentation, we find that a 100% occupancy results employing 256 threads per block and 2048 bytes of shared memory per block independent of the compute capability of the device. The number of registers per thread has been chosen to be 37 (32) for a device with CC = 3.5 (CC = 5.2).

6.4.5 Test problem

We employ a very simple setting to calibrate the new implementation of our algorithm for GPUs. For that we solve a simple Poisson problem with a source

Cartesian coordinates		
0	$\dfrac{1}{\Delta y^2}$	0
$\dfrac{1}{\Delta x^2}$	$\dfrac{-2}{\Delta x^2}+\dfrac{-2}{\Delta y^2}$	$\dfrac{1}{\Delta x^2}$
0	$\dfrac{1}{\Delta y^2}$	0

Polar coordinates		
0	$\dfrac{1}{r_i^2\Delta\theta^2}$	0
$\dfrac{1}{\Delta r_i^2}-\dfrac{1}{2r_i\Delta r}$	$\dfrac{-2}{\Delta r^2}+\dfrac{-2}{r_i^2\Delta\theta^2}$	$\dfrac{1}{\Delta r_i^2}+\dfrac{1}{2r_i\Delta r}$
0	$\dfrac{1}{r_i^2\Delta\theta^2}$	0

Bipolar coordinates		
0	$\dfrac{\cosh\nu_j-\cos\mu_i}{a^2\Delta\nu^2}$	0
$\dfrac{\cosh\nu_j-\cos\mu_i}{a^2\Delta\mu^2}$	$\dfrac{2(\cosh\mu_i-\cos\nu_j)}{a^2\Delta\mu^2}+\dfrac{2(\cosh\mu_i-\cos\nu_j)}{a^2\Delta\nu^2}$	$\dfrac{\cosh\nu_j-\cos\mu_i}{a^2\Delta\mu^2}$
0	$\dfrac{\cosh\nu_j-\cos\mu_i}{a^2\Delta\nu^2}$	0

Table 6.3 Values of the variables of first table in Table 6.2 for some specific cases. In all cases we assume uniform meshes. The upper, middle and lower tables correspond to the standard 5-points discretization of the Laplacian operator in Cartesian, polar and bipolar coordinates, respectively.

term, whose solution is analytically known (see, Chap. 4; Eq. (4.34)), with appropriate Dirichlet boundary conditions. The unit square domain $(x, y) \in [0, 1]^2$ is discretized in $N_x \times N_y = 1024 \times 1024$ nodes, where the numerical solution is computed. The boundaries are easily specified in this case, since there exists an analytic solution for the problem at hand (Eq. (4.34)), which can be used to compute the boundaries at the edges of the computational domain. In next sections, we will take advantage of the knowledge of the analytic solution to compute the reduction of the error with resolution.

6.4.6 Times and ratios

We have solved the test problem of Sec. 6.4.5, until reaching a prescribed tolerance, using the classical 5-points discretization of the Laplacian, and also the two discretizations of 9-points and 17-points introduced in Sec. 6.4.1. We employ a resolution dependence tolerance, which decreases as N^{-2}. For reference, we have solved the problem with the CJM as well as with the classical Jacobi method. Finally, as we have mentioned before in Sec. 6.4.3, we have used two different microarchitectures for the implementations, which work over GPUs.

Figure 6.12 shows the computational time (in seconds) as a function of the number of points per dimension N. For simplicity, we make our measurements in uniform computational meshes satisfying $N_x = N_y = N$, so that the grid resolution in any of the two spatial directions is $h = 1/N$. For low resolutions most of the time in the GPU implementations is consumed by the transfer of data between the host and the device. This time to transfer data is larger than the time required by the method itself. In the different panels of Fig. 6.12 this effect shows up as a plateau region in the curves corresponding to GPU implementations of, specially, the CJM. The plateau extends up to a certain turnover value N_{to} (depending on the method and on the architecture of the device), above which we observe that the slopes of the lines stabilize in a very similar way to the one of the sequential execution, but almost two orders of magnitude below it. For instance, in the upper left panel, of Fig. 6.12 this flat region extends up to $N_{\text{to}} \lesssim 300$. The latency of the data transfers is more obvious in the Kepler architecture (corresponding to a CC of 3.5) than in the case of Maxwell. The turnover in the former device happens at $N_{\text{to}} \sim 300$ for the CJM run over GPUs, while it is located at $N_{\text{to}} \sim 256$ for the latter device (see Fig. 6.12 upper left panel). For the Jacobi method implemented on GPUs, the aforementioned plateau does not exist. The reason for it is twofold. First, the amount of data transfers between the host and the CUDA device is slightly

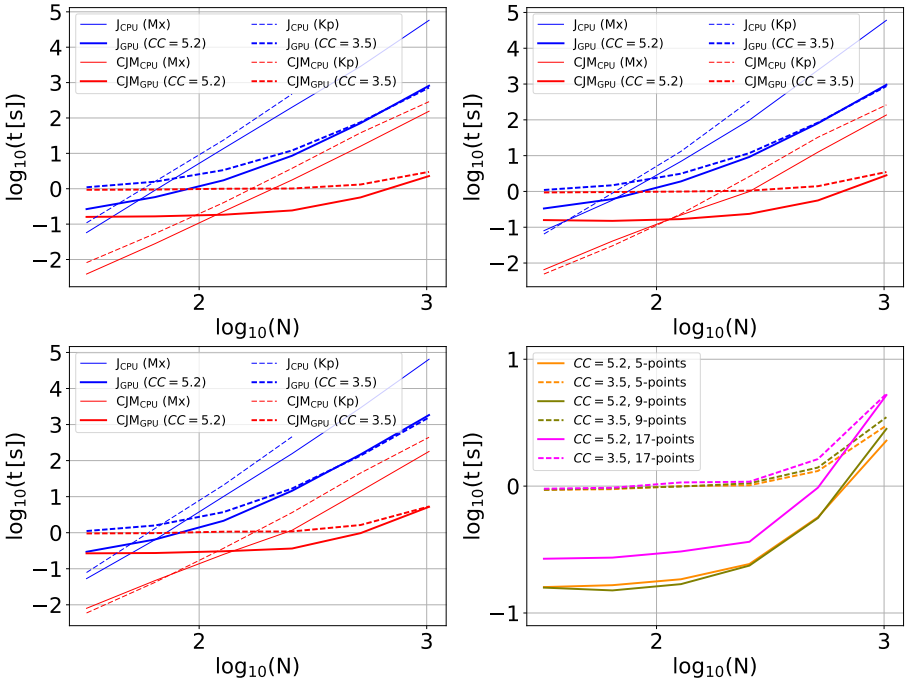


Figure 6.12 Both top and bottom left plots show the time necessary to reach a prescribed tolerance for different methods and implementations. We use a 5-points, 9-points and 17-points stencils in top left, top right and bottom left plots, respectively. Blue (red) lines refer to the Jacobi (Chebyshev-Jacobi) method. Thin (thick) lines correspond to serial code (GPU implementation). Solid (dashed) lines are associated to a Compute Capability equals to 5.2 (3.4). Bottom right plot shows the times of the Chebyshev-Jacobi method over GPUs.

smaller in the latter method. This extra data required by the CJM are the M weights needed to perform the M iterations of a complete computational cycle. Second, the Jacobi method is computationally more intensive than the CJM, since the number of operations per grid node is comparable in both methods, but the number of iterations to reach the tolerance goal is much larger in the Jacobi method than in CJM. As a result, the Jacobi method is relatively more costly with respect to the data transfer. We therefore conclude that a minimum mesh size is needed so that the data transfer time between the CPU and the GPU, and viceversa, is negligible in comparison to the computing time. Furthermore, we note that below certain critical mesh size it is not even advantageous using the CUDA devices, since the CPU implementation of the methods at hand runs faster. This is the case, e.g., of the Jacobi method when using the 5-points discretization of the Laplacian displayed in Fig. 6.12 (upper left panel): where the thin solid (dashed) blue line, corresponding to the CPU executions, exhibit

an smaller computational time than the corresponding GPU runs, indicated with thick solid (dashed) lines. Both lines (thin and thick corresponding to the same method) cross at a "critical" value $N_c \lesssim 64$. This effect is exacerbated in the test involving the CJM, where the CPU executions are faster than the corresponding GPU ones for $N_c \lesssim 128$. Noteworthy, comparing the upper left panel of Fig. 6.12 with the upper right and bottom left panels of the same figure, the effectiveness of the GPU implementation of the CJM over the corresponding CPU does not depend on the stencil of the discretization of the Laplacian.

From the bottom right plot, where only the Chebyshev-Jacobi method on GPUs is plotted, we observe that the qualitative behavior is quite similar for all the different stencils. Comparing results obtained in the two CUDA devices with the same discretization of the Laplacian, it is evident that the architecture of Maxwell (CC = 5.2) is faster than that of Kepler. However, the gap between both architectures reduces as the resolution increases. Actually, the difference in execution time between both devices displays a trend to reduce as the number of points in the discretization of the Laplacian grows, particularly beyond the value of N_{to} for each test. This is remarkably visible for the 17-points discretization of the Laplacian (pink lines of Fig. 6.12 bottom right panel), since at the highest resolution of our tests ($N = 1024$), the computational time is the same in both devices. This happens in spite of the fact that Maxwell architecture is faster regarding both transfers and executions. However, Kepler architecture is better suited for double precision computations, which we employ in practical applications of the elliptic solver at hand.

In Figure 6.13 (top panel) we plot the ratios between the times of different methods and implementations. These ratios provide an estimation of the acceleration factor by which the GPU implementation of either the CJM or the Jacobi method is faster (or slower) than any other of the methods. The figure also displays the dependence of the speed up of the GPU versions on the three types of stencil. We observe that the larger number of points the stencil has, the smaller is the speed up. The improvement over the CPU implementation is larger for the Jacobi method than for the CJM (note that the purple solid line lies above any of the blue lines in top panel of Fig. 6.13). However, this larger relative speed up in the Jacobi method, with the smallest number of discretization points of the Laplacian, reduces with either increasing values of N, or when considering higher order discretizations of the Laplacian. In the latter case, the speed up of the CJM over GPUs equals that of the corresponding Jacobi method employing a 17-points discretization of the Laplacian when $N = 1024$. In fact, extrapolating the results to even higher resolutions (not included in the

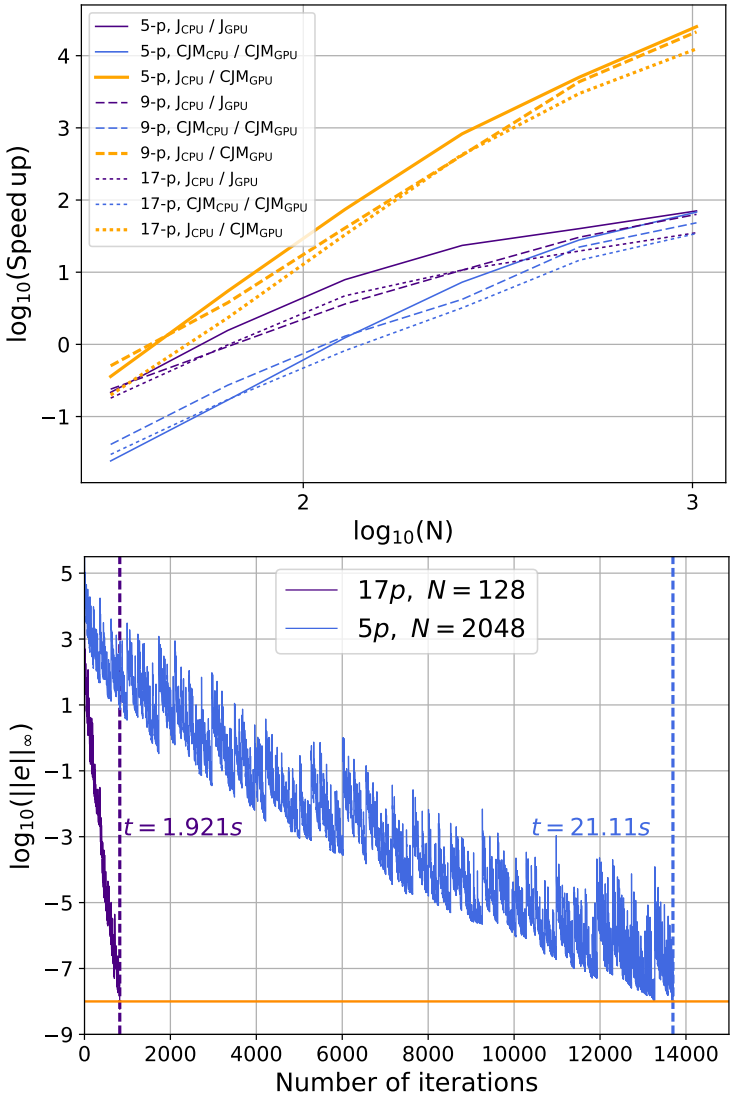


Figure 6.13 Top: Ratio of computational time in a CPU execution to the computational time in an analogous GPU execution as a function of the number of points per dimension. Solid lines correspond to the 5-points stencil, dashed lines to the 9-points stencil and dotted lines to the 17-points stencil. Blue and purple lines represent the ratios between the times of the sequential versus the parallel implementations of the Jacobi and Chebyshev-Jacobi methods. Orange lines represent the ratio between the time of the slowest method, the sequential implementation of the Jacobi method, and the fastest one, the CJM in parallel over GPUs. Bottom: Real error versus number of iterations for a 17-points stencil with a mesh of $N = 128$ points (purple line) and a classical 5-points stencil with a mesh of $N = 2048$ points (blue line), until reaching a real error of 10^{-8} (orange horizontal line). The vertical dashed lines indicate the final number of iterations. The computational times of each run are also labeled in the plot.

5-points	j	j_GPU	cj	cj_GPU
j	1	–	–	–
j_GPU	71	1	–	–
cj	371	5	1	–
cj_GPU	25235	357	68	1
9-points	**j**	**j_GPU**	**cj**	**cj_GPU**
j	1	–	–	–
j_GPU	63	1	–	–
cj	441	7	1	–
cj_GPU	21360	337	48	1
17-points	**j**	**j_GPU**	**cj**	**cj_GPU**
j	1	–	–	–
j_GPU	35	1	–	–
cj	361	10	1	–
cj_GPU	12420	352	34	1

Table 6.4 Speed up factors of each of the methods listed in the columns with respect to the methods annotated in the rows in which the stencil of the Laplacian discretization is provided. All the values correspond to the test problem of Sec. 6.4.5 evaluated on a grid with $N = 1024$ points per dimension.

plot) we foresee that the speed up factor of the CJM over its corresponding sequential CPU implementation will be better than that of the Jacobi method. From the ratio between the CJM on GPUs, and the slowest one, the classical Jacobi method in sequential, it can be appreciated the difference of several orders of magnitude in the speed up for high resolutions (see orange lines in top panel of Fig. 6.13).

In addition to Fig. 6.13, we list the values of the ratios for the particular case of a mesh with $N = 1024$ points per dimension in Table 6.4. In this table we systematically include the speed up factors of each of the methods listed in the columns with respect to the methods annotated in the rows in which the stencil of the Laplacian discretization is provided.

Finally, as we have already shown in Chap. 4, it proves convenient to use a higher order discretization of the operator although it involves larger stencils. In the latter case the computational time due to the fact of having more points to obtain a solution of the same quality is negligible compared to the reduction in time due to the smaller number of iterations needed. To illustrate this point, we use the CJM on GPUs (our method of choice). We understand that having solutions of the same quality means reaching the same "real" error level. By real error we denote the difference, in infinity norm, between the obtained numerical solution and the analytical one. Since we have the analytical solution (Eq. (4.34))

of the test problem (Eq. (4.33)), we can use the real error instead of a tolerance as the stopping criterion. In Fig. 6.13 we plot the real error versus number of iterations using, on one hand, a 17-points stencil with a mesh of $N = 128$ points and, on the other hand, a classical 5-points stencil with a mesh of $N = 2048$ points, until reaching a prescribed value of the real error (10^{-8}); in addition to the smaller resolution needed for the higher order method (17-points stencil), we have a reduction of one order of magnitude both in the number of iterations and in the computational time.

Chapter 7

Beyond astrophysics: modelling the human eye

The results of this chapter have been originally published in (or are submitted to):

Evaluation of the repeatability of a swept-source ocular biometer for measuring ocular biometric parameters,
T Ferrer-Blasco, A Domínguez-Vicent, JJ Esteve-Taboada, MA Aloy, JE Adsuara, R Montés-Micó,
Graefe's Archive for Clinical and Experimental Ophthalmology, 1-7 (2016)

Estimation of the Mechanical Properties of the Eye through the Study of its Vibrational Modes,
MA Aloy, JE Adsuara, P Cerdá-Durán, M Obergaulinger, JJ Esteve-Taboada, T Ferrer-Blasco, R Montés-Micó,
Submitted to PLOS ONE

Human eye normal vibrational modes for different axial lengths using linear elasticity,
JE Adsuara, P Cerdá-Durán, M Obergaulinger, P Mimica, JJ Esteve-Taboada, T Ferrer-Blasco, R Montés-Micó, MA Aloy,
Submitted to Graefe's Archive for Clinical and Experimental Ophthalmology

The text in the following sections corresponds to an edited version of the aforementioned publications.

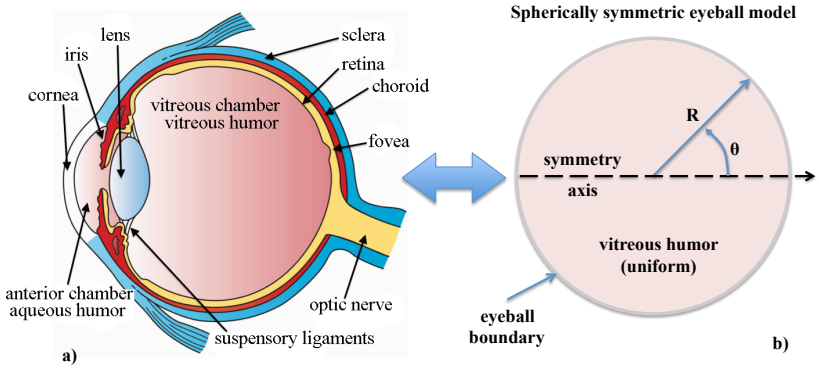


Figure 7.1 Simplified mechanical model of the eyeball. Left: transversal cut of the human eye with the different structural parts annotated in it (source: Holly Fischer, Wikipedia). Right: spherically symmetric, homogeneous and isotropic eyeball model employed in this work.

During the elaboration of the present thesis it has arisen a parallel project devoted to the application of elliptic equations in an area very different from astrophysics. We have applied our numerical schemes to model oscillations in a human eye, which has applications in optometry and biomechanics. We discuss below its most relevant aspects: we construct a model to obtain oscillation frequencies, we investigate the dependence on the parameters of the system and we present our results of the collaboration with the Department of Optics at the University of Valencia to perform measurements in human eyes, which will serve to improve the model in the future.

7.1 Introduction

The eye is a complex organ consisting of several functional and mutually interacting parts [Ethier, Johnson, and Ruberti 2004]. The most important ones are the cornea, lens, vitreous, sclera and retina (Fig. 7.1). Measuring the eye's mechanical properties *in vivo* and with minimally invasive techniques can be the key for individualized solutions to a number of eye pathologies. Here we show that these properties are related to the normal vibrational modes of the eyeball, i.e., to the periodic variations of matter inside of the eyeball resulting from perturbations with respect to its equilibrium state.

We model the eyeball as a *spherical*, *homogeneous* and *isotropic elastic* solid ball with axial symmetry (Fig. 7.1). While assuming that the eyeball is axially symmetric is very well justified, the assumptions of homogeneity and isotropy are certainly not the most accurate possible. However, these assumptions serve

for the primary purpose of reducing the dependence of the constitutive equation only to two elastic constants or moduli of the eye material: the Young's modulus E, and the Poisson ratio σ [1]. In this simplified framework, we will compute, first analytically and afterwards numerically, the eigenfrequencies of the model attempting to grasp the essential mechanics of an average human eye.

7.2 Analytic normal modes

In linear elasticity, the equation of motion for an homogeneous isotropic elastic solid is given by the *Navier-Cauchy* equation [Love 1944], which can be written either in vector form

$$(\lambda + 2\mu)[\nabla(\nabla \cdot \boldsymbol{u})] - \mu[\nabla \times (\nabla \times \boldsymbol{u})] + \boldsymbol{F} = \rho \ddot{\boldsymbol{u}}, \tag{7.1}$$

or, component-wise, as:

$$\mu \nabla^2 u_i + (\lambda + \mu)\vartheta_{,i} + F_i = \rho \ddot{u}_i, \tag{7.2}$$

where u_i are the *displacements* with respect to an equilibrium position, $\vartheta = \nabla \cdot \boldsymbol{u}$ is the *dilatation*, double dotted quantities denote the second time derivatives (∂^2_{tt}) of such quantities, ∇^2 is the Laplacian operator, F_i denote the body forces, and μ and λ are the *Lamé constants*. The Lamé constants are related with the Young's modulus, σ, and the Poisson ratio, E, by the following expressions:

$$\sigma = \frac{\lambda}{2(\lambda + \mu)}, \qquad E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu} \tag{7.3}$$

We start assuming oscillatory solutions of the form

$$u_i = u'_i \cos(pt + \xi), \tag{7.4}$$

where p is the angular frequency of the perturbation, u'_i are functions independent of the time t and the constant ξ is independent of the coordinates x_i and only modifies the phase of the vibration. Plugging the test solutions Eq. (7.4) into Eq. (7.2), one obtains in the absence of body forces ($F_i = 0$):

$$\mu \Delta u_i + (\lambda + \mu)\vartheta_{,i} + \rho p^2 u_i = 0, \tag{7.5}$$

which can be rewritten in the form of an eigenvalue problem

$$-\frac{\mu}{\rho}\Delta u_i - \frac{\lambda + \mu}{\rho}\vartheta_{,i} = p^2 u_i. \tag{7.6}$$

[1] The Young's modulus is a measure of the stiffness of a solid material. It defines the relationship between stress (force per unit area) and strain (proportional deformation) in a material. The Poisson ratio is the amount of transversal expansion divided by the amount of axial compression, for small values of these changes.

where the allowed frequencies of vibration and their corresponding displacements (i.e., distortions of the underlying homogeneous and isotropic structure) correspond to the eigenvalues and eigenvectors of the Navier-Cauchy equation for given boundary conditions (imposed at the eyeball surface; Fig. 7.1).

There are exact solutions of Eq. (7.6) for solid elastic bodies in which simple boundary conditions are imposed. One example is a sphere with homogeneous Neumann boundary conditions applied at its surface. Noteworthy, these solutions have been adapted from its original forms to the study of the oscillations of spherical detectors of Graviational Waves. Under the assumption of axisymmetry and expressing their results in terms of spherical solid harmonics ω_n, ϕ_{n+1} and χ_n, Rue [Rue 1996] obtained, based on Love [Love 1944], analytic solutions for the shapes of these vibrations,

$$u_i = \sum_n \Bigg[-\frac{1}{h^2}\frac{\partial}{\partial x_i}\Big\{\omega_n \psi_n(hr)\Big\} + \psi_n(\kappa r)\Big(\epsilon_{ijk} x_j \frac{\partial}{\partial x_k}\chi_n + \frac{\partial}{\partial x_i}\phi_{n+1}\Big) - \frac{n+1}{n+2}\psi_{n+2}(\kappa r)\kappa^2 r^{2n+5}\frac{\partial}{\partial x_i}\frac{\phi_{n+1}}{r^{2n+3}}\Bigg] \tag{7.7}$$

as well as for the vibrational frequencies of spherical bodies. Here, ϵ_{ijk} is the *Levi-Civita symbol* and

$$\psi_n(x) = \left(\frac{1}{x}\frac{\partial}{\partial x}\right)^n \frac{\sin x}{x}, \tag{7.8}$$

In axially symmetric systems, the eigenmodes can be classified in two groups: toroidal and spheroidal. Toroidal modes only involve motion about the symmetry axis sketched in Fig. 7.1. Toroidal modes are incompressible since they do not change the volume of the eyeball. In this first class of vibrations ω_n and ϕ_n vanish ($\omega_n = \phi_n = 0$), and the frequency, from which we compute the eigenvalues of the system, is given by:

$$p_n = 0 \quad \text{with} \quad p_n = (n-1)\psi_n(\kappa a) + \kappa a \psi_n'(\kappa a) \tag{7.9}$$

Spheroidal modes, implying displacements of the eyeball material in both radial and/or angular directions, are compressible. In this second class of vibrations χ_n vanish ($\chi_n = 0$). The frequency equation is given by:

$$b_n c_n - a_n d_n = 0 \tag{7.10}$$

with:

$$a_n = \frac{1}{(2n+1)h^2}[\kappa^2 a^2 \psi_n(ha) + 2(n-1)\psi_{n-1}(ha)], \tag{7.11}$$

$$b_n = -\frac{1}{2n+1}\Big[\frac{\kappa^2}{h^2}\psi_n(ha) + \frac{2(n+2)}{ha}\psi_n'(ha)\Big], \tag{7.12}$$

$$c_n = \kappa^2 a^2 \psi_n(\kappa a) + 2(n-1)\psi_{n-1}(\kappa a), \tag{7.13}$$

$$d_n = \kappa^2 \frac{n}{n+1}\Big[\psi_n(\kappa a) + \frac{2(n+2)}{\kappa a}\psi'_n(\kappa a)\Big]. \tag{7.14}$$

7.3 Numerical code

We have developed a code that solves the eigenvalue problem set by the Navier-Cauchy equation discretizing the eyeball sphere on a two-dimensional grid of nodes in spherical coordinates ($0 \le r \le R$, $0 \le \theta \le \pi$). As a first step, we have assumed the elastic moduli to be uniform throughout the spatial grid. However, there is no restriction to implement elastic moduli that depend on the location in the eyeball. This is important because it may enable us to improve the degree of realism of our model for the vibrational modes of the eye, in particular, by using different elastic moduli for the sclera, the cornea, the lens, and the vitreous humour.

First, we will rewrite Eq. (7.6) for the numerical part with the analytical solutions available in the previous subsection. In spherical coordinates, and under the assumption of axisymmetry, i.e., neglecting the φ-dependence, the displacements can be written as

$$u_i = (u_r(r,\theta), u_\theta(r,\theta), u_\varphi(r,\theta)),$$

and satisfy the following three equations:

$$-\frac{\mu}{\rho}\Delta u_r - \frac{\lambda+\mu}{\rho}u_{j,jr} = p^2 u_r \tag{7.15}$$

$$-\frac{\mu}{\rho}\Delta u_\theta - \frac{\lambda+\mu}{\rho}u_{j,j\theta} = p^2 u_\theta \tag{7.16}$$

$$-\frac{\mu}{\rho}\Delta u_\varphi - \frac{\lambda+\mu}{\rho}u_{j,j\varphi} = p^2 u_\varphi. \tag{7.17}$$

Under the assumption of axisymmetry, all the derivatives with respect to φ vanish and Eq. (7.17) decouples from the other two Equations (7.15) and (7.16). We obtain the toroidal modes from the latter equation:

$$-\frac{\mu}{\rho}\Delta u_\varphi = p^2 u_\varphi. \tag{7.18}$$

with homogeneous Neumann boundary conditions, $u_{\varphi,r} = 0$. The spheroidal modes result from Equations (7.15) and (7.16):

$$-\frac{\mu}{\rho}\Delta u_r - \frac{\lambda+\mu}{\rho}\partial_r\left[\frac{2}{r}u_r + \partial_r u_r + \frac{1}{r}\partial_\theta u_\theta + \frac{\cot\theta}{r}u_\theta\right] = p^2 u_r \tag{7.19}$$

$$-\frac{\mu}{\rho}\Delta u_\theta - \frac{\lambda+\mu}{\rho}\frac{1}{r}\partial_\theta\left[\frac{2}{r}u_r + \partial_r u_r + \frac{1}{r}\partial_\theta u_\theta + \frac{\cot\theta}{r}u_\theta\right] = p^2 u_\theta \tag{7.20}$$

also with homogeneous Neumann boundary conditions, $u_{r,r} = 0; u_{\theta,r} = 0$. Equations (7.19) and (7.20) can be cast as an eigenvalue equation, $Lu = \lambda u$, with the vectorial operator

$$Lu = -\frac{\mu}{\rho}\Delta u_i - \frac{\lambda+\mu}{\rho}u_{j,ji} \tag{7.21}$$

with $i = r, \theta$, i.e., the equations are coupled due to the axisymmetry hypothesis. If we explicitly insert spherical coordinates, then:

$$Lu = \left(-\frac{\mu}{\rho}\Delta u_r - \frac{\lambda+\mu}{\rho}u_{j,jr}, -\frac{\mu}{\rho}\Delta u_\theta - \frac{\lambda+\mu}{\rho}u_{j,j\theta} \right) \tag{7.22}$$

and, for clarity, in matrix form we have:

$$Lu = \begin{bmatrix} a_{rr} & a_{r\theta} \\ a_{\theta r} & a_{\theta\theta} \end{bmatrix} \begin{bmatrix} u_r \\ u_\theta \end{bmatrix} \tag{7.23}$$

where expansion of the operator yields:

$$a_{rr} = -\frac{\lambda+2\mu}{\rho}\partial_{rr} - \frac{2(\lambda+2\mu)}{\rho r}\partial_r - \frac{\mu}{\rho r^2}\partial_{\theta\theta} - \frac{\mu\cot\theta}{\rho r^2}\partial_\theta + \frac{2(\lambda+2\mu)}{\rho r^2} \tag{7.24}$$

$$a_{r\theta} = -\frac{(\lambda+\mu)\cot\theta}{\rho r}\partial_r + \frac{\lambda+3\mu}{\rho r^2}\partial_\theta - \frac{\lambda+\mu}{\rho r}\partial_{r\theta} + \frac{\cot\theta(\lambda+3\mu)}{\rho r^2} \tag{7.25}$$

$$a_{\theta r} = \frac{2(\lambda+2\mu)}{\rho r^2}\partial_\theta - \frac{\lambda+\mu}{\rho r}\partial_{r\theta} \tag{7.26}$$

$$a_{\theta\theta} = -\frac{\mu}{\rho}\partial_{rr} + -\frac{2\mu}{\rho r}\partial_r - \frac{\lambda+2\mu}{\rho r^2}\partial_{\theta\theta} - \frac{(\lambda+2\mu)\cot\theta}{\rho r^2}\partial_\theta + \frac{(\lambda+2\mu)\csc^2\theta}{\rho r^2} \tag{7.27}$$

To begin with, we focus on toroidal modes. We compute in a first step the eigenvalues (vibrational frequencies), and in a second step the eigenfunctions (normal displacements). For the eigenvalues we simply compute the zeros of the characteristic polynomial. In practice, working in logarithmic space is advantageous because it reduces the magnitude of coefficients of the polynomial. Knowing the family of eigenvalues, we compute the kernel for each one of them. We substitute each eigenvalue into the corresponding equation, Eq. (7.18) or Equations (7.19)-(7.20), obtaining an elliptic equation. With this, we subtract the eigenvalue from the diagonal of the matrix produced by the discretization of the elliptic operator, and proceed further solving the corresponding system of equations by direct numerical inversion of the matrix of the system. As the rank of this matrix cannot be complete, we will obtain the compatible but indeterminate solution as a function of one of the variables for the toroidal case.

The computation of the characteristic polynomial and the solution of the elliptic equation, have been performed numerically by means of the LU decom-

position (LAPACK implementation). In the resolution of the elliptic problem needed for the computation of the eigenvectors it is not possible to use the CJM. The reason is that the matrix of the system is not positive definite. In chapter 9 we discuss a variant of the CJM method that could be used in such cases.

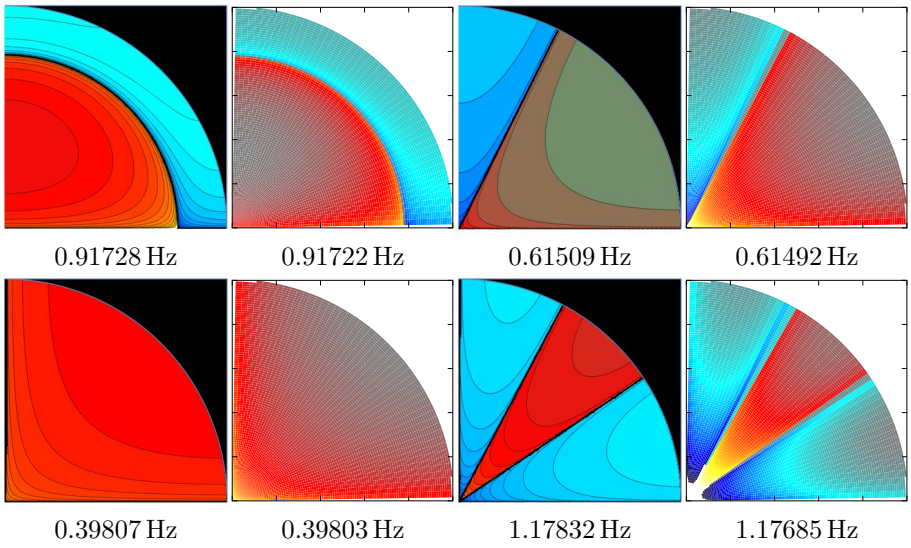


Figure 7.2 Comparison between the analytic (panels with black background) and numerical (white background) solutions of vibrational patterns. Because of the symmetries, only one quadrant of the full equatorial plane of an spherical body is shown. Modes of odd and even parities are displayed in the upper and lower panels, respectively. In this case, we are using 100 points in the radial direction and 50 in the angular one. We can also observe a good agreement in their corresponding frequencies (listed below each panel), that improves as we increase the resolution.

We calibrate the code by comparing the frequencies computed with our numerical code and the corresponding analytic values at a density of $\rho = 1\,\mathrm{kg\,m^{-3}}$, elastic moduli of $E = 2.5\,\mathrm{Pa}$, $\sigma = 0.25$ and a radius of the sphere of $R = 1\,\mathrm{m}$. Note that these values do not correspond to a typical human eye. They are employed for numerical convenience.

As shown in Fig. 7.2, we get a good agreement in the toroidal ($\varphi-$) case, both in the vibrational patterns and in their corresponding frequencies, demonstrating the ability of the numerical code to recover the analytic values. We point out that agreement improves with a finer mesh encompassing the eyeball (in Fig. 7.2 we employ a relatively coarse grid of 100×50 points in the $r \times \phi$ directions). A similar analysis has been done for modes where the displacements of the material happen only in the $r-$ and $\theta-$directions (spheroidal modes). The conclusion of

Table 7.1 Frequencies of selected normal modes of the eye. Table containing the frequencies (measured in Hertz) of selected toroidal modes (T) computed with our numerical code for an average human eyeball. The set of material parameters employed to obtain these values are $R = 0.0125\,\mathrm{m}$, $\rho = 1000\,\mathrm{kg\,m^{-3}}$, $E = 0.2985\,\mathrm{MPa}$, and $\sigma = 0.49$. Toroidal modes with $n = 0$ are forbidden since they require driving external forces (assumed non existing in this model).

T		l		
		1	2	3
n	1	734.455	318.707	492.481
	2	1158.97	909.362	1076.14
	3	1570.32	1339.88	1514.10

both calibration experiments is that our numerical procedure to compute the eigenfrequencies of the system and their displacements is accurate enough.

7.4 Application of the method to a typical human eye

The exact eigenfrequency values are sensitive to the imposed boundary conditions. We assume that the surface of the eye (either the sclera or the cornea) is free to oscillate when suitable perturbations are inflicted to the eyeball. Here, we consider a set of "standard" eye parameters. We adopt $R = 0.0125\,\mathrm{m}$, $\rho = 1000\,\mathrm{kg\,m^{-3}}$ for the eyeball typical radius and average density, respectively. Mean values for the corneal and scleral Poisson ratio, σ, are in the range $0.42-0.47$ [Uchio et al. 1999]. We take $\sigma = 0.49$, slightly above the average to account for the incompressible character of the vitreous humour. As the eigenfrequencies are roughly proportional to $\sigma^{-1/2}$, their predicted values are basically insensitive to this parameter in the typical ranges measured for constituents of the human eye. There is a large scattering in the values of the Young's modulus, E, of different parts of the eye [Hirneiss et al. 2011]. We employ a typical value $E = 0.2985\,\mathrm{MPa}$. The eigenfrequencies exhibit a weak dependence with the value of the Young's modulus, $\propto E^{1/2}$. Since the largest values reported for the Young's modulus are $E_{\max} \simeq 20\,\mathrm{MPa}$, at most a factor of a few increase in the computed frequencies is possible.

In Fig. 7.3 we show six different patterns of toroidal vibrational modes at the lowest frequencies in our simplified model of the eye that correspond to the same transversal cut as shown in Fig. 7.1. The different patterns are identified by a set of two integer numbers n and l that denote the number of nodes the solution has in the radial and in the $\theta-$angular direction, respectively. Each pair of values (n, l) has a unique characteristic frequency. The upper left panel of

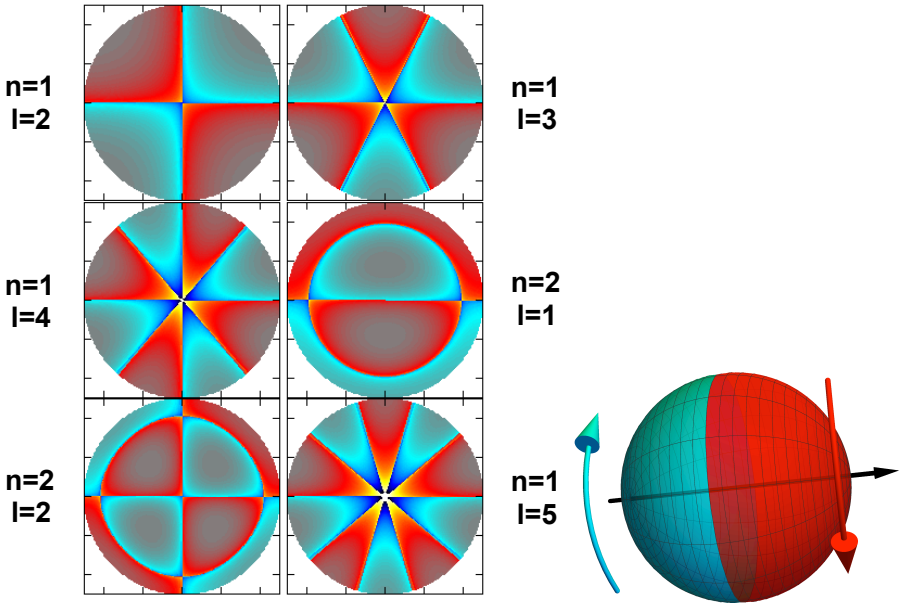


Figure 7.3 *Left:* Six different patterns of toroidal vibration at the lowest frequencies in our model of the eye that correspond to the same transversal cut as shown in Fig. 7.1. Light and dark blue (red and yellow) shades indicated a motion towards (away from) the reader and normal to the drawn plane. *Left panels:* eigenfunctions with even parity in l: $(n = 1, l = 2)$ vibrating at 319 Hz, $(1, 4)$ at 648 Hz and $(2, 2)$ at 909 Hz. *Right panels:* eigenfunctions with odd parity: $(1, 3)$ at 492 Hz, $(2, 1)$ at 1159 Hz and $(1, 5)$ at 798 Hz. *Right:* Three-dimensional representation of the toroidal mode $n = 1$ and $l = 2$ (corresponding to the upper left panel. The arrows indicate the direction of the motion about the symmetry axis of the system (showed with a black arrow).

Fig. 7.3 corresponds to matter rotating (counter-rotating) about the symmetry axis in the northern (southern) hemisphere (see Fig. 7.3 for a three-dimensional representation of the mode $(1, 2)$). There is a number of normal mode frequencies falling in the range ~ 100 Hz to ~ 10 MHz (Tab. 7.1). Modes with frequencies of a few hundreds of Hz have periods of oscillation much shorter than other quasi-periodic variations of the eyeball volume triggered by phasic processes like respiration and pulse.

7.5 Frequencies as a function of the Axial Length

The purpose of this section is to assess how the frequency of the normal vibration modes of the human eye varies for different axial lengths (AL).

AL of the human eye is the axial distance between the anterior corneal surface and the retinal pigment epithelium [Hitzenberger 1991]. The main morphological

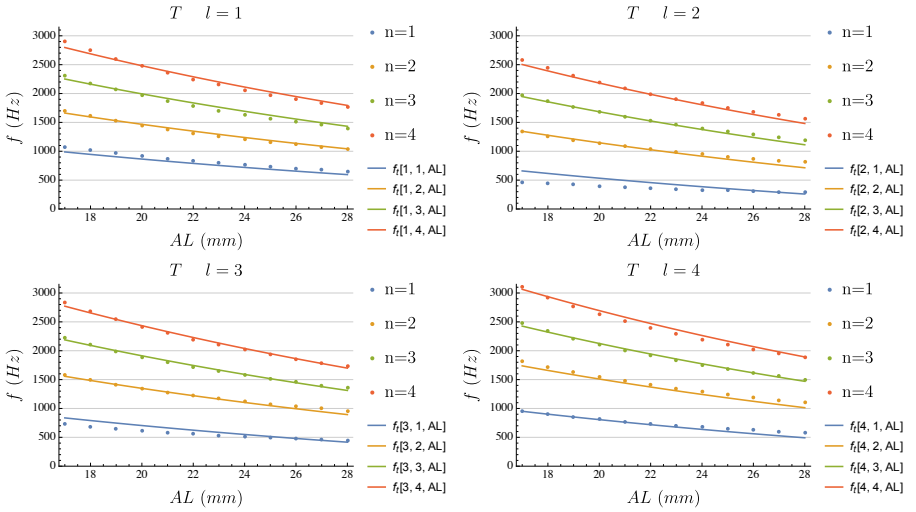


Figure 7.4 Frequency values for the first toroidal vibrational modes of the eye (T). The corresponding value for l is indicated in each panel. The obtained frequencies are shown in colored dots, depending on the values for $n = [1, 4]$ and $AL = [17, 28]$ mm. Fitted analytical functions are shown in solid lines following the same color coding. It is easy to infer an inverse relationship between the AL and the f values.

difference between a myopic, an emmetropic, and a hyperopic eye is the axial length [Llorente et al. 2004]. Besides, AL value is changing from birth to the adult stage, following the human body growth. In newborns, the average value is about 17 mm in diameter [Hussain and Shahid 2014]. In the infant stage (between $3 - 8$ years old), the eye grows to a length of $19 - 21$ mm. During childhood and adolescence, and towards the adult stage, the human eye continues growing gradually to a final size of about, depending on the final refractive state, $23 - 27$ mm [Vaughan and Asbury 2004, Llorente et al. 2004].

Fig. 7.4 shows the plots with the frequency values for the first toroidal (T) vibrational modes of the eye. The corresponding vibrational mode is indicated in each panel. Frequency values are shown for $AL = [17, 28]$ mm. Dots of different colors are used to distinguish modes with different values of n (from $n = 1$ to 4). The numerical values of the discrete points of Fig. 7.4 are shown in Tab. 7.2.

A simple inspection of Fig. 7.4 suggests that there exists a one-to-one map relating and frequency with the AL. We note that frequency decreases monotonically as AL is increased for each fixed value of n. Thus, we have fit the data with the following multidimensional analytical function,

$$f_t(l, n, AL) = \cosh(0.0769803 l^{1.45408}) \cosh(8.3869 n^{0.0763511}) \exp(-0.0373661 AL) \\ -97.8468 l^3 + 821.721 l^2 - 2131.71 l + 1230.2. \qquad (7.28)$$

Table 7.2 Values of the frequencies that have been numerically obtained for the first vibrational toroidal and spheroidal modes (all frequencies are measured in Hz). Values for $l = 1$ and $l = 2$ in the upper left and right tables, respectively, and for $l = 3$ and $l = 4$ in the lower left and right ones.

AL	$n = 1$	$n = 2$	$n = 3$	$n = 4$
17	1080.08	1704.37	2309.29	2907.44
18	1020.08	1609.68	2180.99	2745.91
19	966.389	1524.96	2066.20	2601.39
20	918.069	1448.71	1962.89	2471.32
21	874.352	1379.73	1869.42	2353.64
22	834.608	1317.01	1784.45	2246.66
23	798.321	1259.75	1706.86	2148.97
24	765.058	1207.26	1635.75	2059.43
25	734.455	1158.97	1570.32	1977.06
26	706.207	1114.40	1509.92	1901.02
27	680.051	1073.12	1454.00	1830.61
28	655.764	1034.80	1402.07	1765.23

AL	$n = 1$	$n = 2$	$n = 3$	$n = 4$
17	468.687	1337.30	1970.42	2580.81
18	442.649	1263.00	1860.95	2437.43
19	419.351	1196.53	1763.00	2309.15
20	398.384	1136.70	1674.85	2193.69
21	379.413	1082.57	1595.10	2089.23
22	362.167	1033.37	1522.59	1994.26
23	346.421	988.437	1456.39	1907.55
24	331.986	947.252	1395.71	1828.07
25	318.707	909.362	1339.88	1754.95
26	306.449	874.386	1288.35	1687.45
27	295.099	842.002	1240.63	1624.95
28	284.560	811.930	1196.32	1566.92

AL	$n = 1$	$n = 2$	$n = 3$	$n = 4$
17	724.237	1582.56	2226.62	2843.85
18	684.002	1494.64	2102.92	2685.86
19	648.002	1415.98	1992.24	2544.50
20	615.602	1345.18	1892.62	2417.27
21	586.287	1281.12	1802.50	2302.16
22	559.638	1222.89	1720.57	2197.52
23	535.306	1169.72	1645.76	2101.97
24	513.002	1120.98	1577.19	2014.39
25	492.481	1076.14	1514.10	1933.82
26	473.540	1034.75	1455.86	1859.44
27	456.001	996.428	1401.94	1790.57
28	439.716	960.842	1351.87	1726.62

AL	$n = 1$	$n = 2$	$n = 3$	$n = 4$
17	954.721	1820.11	2475.69	3100.41
18	901.681	1718.99	2338.15	2928.16
19	854.224	1628.52	2215.09	2774.05
20	811.513	1547.09	2104.34	2635.35
21	772.870	1473.42	2004.13	2509.85
22	737.739	1406.44	1913.03	2395.77
23	705.664	1345.30	1829.86	2291.61
24	676.261	1289.24	1753.62	2196.12
25	649.211	1237.67	1683.47	2108.28
26	624.241	1190.07	1618.72	2027.19
27	601.121	1145.99	1558.77	1952.11
28	579.652	1105.06	1503.10	1882.39

This function can be used to interpolate the eyeball vibrational frequencies within the range of axial lengths in which a human eyeball is expected to fall, without the need of solving a costly and computationally involved eigenvalue problem.

The values obtained for this fitted analytical function are shown in Fig. 7.4 using solid lines of the same colour than that for the discrete frequency values (obtained numerically with the methodology described in Sec. 7.3). As can be seen, the analytical function f_t provides a good fit to the data, with typical errors smaller than 1% and maximum errors $\lesssim 15\%$.

7.6 Model improvements

We have developed a very simple model to obtain the vibrational modes of the eyeball. The final goal of this project is to find whether the vibrational modes of the eyeball structure are related not only to the size or structure of the eye, but also to the health condition of potential patiens. A first step towards that aim is to obtain physiological data from actual individuals that can be fed into our models. In particular, the most basic data we need to build our model are the dimensions of the eyeball and of its internal constituents. Working together with optometry speciallists, we have obtained data from a sample of volunteer people about the eyeball biometric parameters. For that, we have employed a new swept-source optical biometer (the IOLMaster 700 swept-source optical

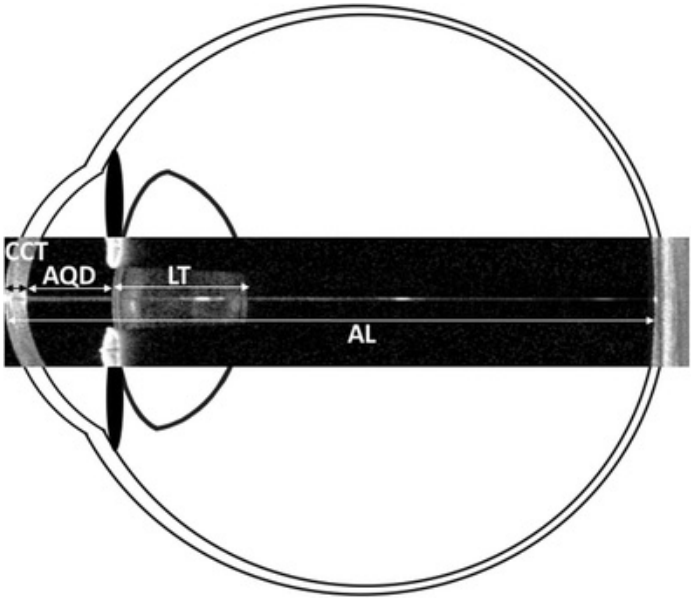


Figure 7.5 Example of a SS-OCT biometry optical B-scan. CCT central corneal thickness, AQD aqueous depth, LT lens thickness, AL axial length.

biometer; Carl Zeiss Meditec, Jena, Germany). A typical output of the biometer can be seen in Fig. 7.5, where also the main sizes of the foremost structural elements of the eye are marked.

Table 7.3 Mean values, ranges, and repeatability metrics of the parameters evaluated. The acronyms employed mean: "SD" standard deviation, "Min" minimum value, "Max" maximum value, "Sw" within subject standard deviation, "AL" axial length, ACD" anterior chamber depth, "CCT" central corneal thickness, "LT" lens thickness, "WTW" white-to-white distance, "$K1$" and "$K2$" keratometry values.

Parameter	Mean ± SD [Min - Max]	Sw	Repeatability limit
AL [mm]	24.07 ± 1.27 [$20.42 - 27.07$]	0.01	0.03
ACD [mm]	3.60 ± 0.29 [$2.93 - 4.22$]	0.03	0.07
CCT [μm]	552 ± 32 [$489 - 610$]	2	4
LT [mm]	3.71 ± 0.31 [$3.23 - 4.34$]	0.04	0.11
WTW [mm]	12.13 ± 0.43 [$11.18 - 12.94$]	0.10	0.20
$K1$ [mm]	7.75 ± 0.28 [$7.15 - 8.25$]	0.02	0.06
$K2$ [mm]	7.65 ± 0.29 [$6.82 - 8.15$]	0.02	0.05

Beyond the possibility of obtaining directly measured biometric parameters, we have participated in the evaluation of the repeatability of the IOLMaster 700 optical biometer. Thirty subjects with healthy and phakic eyes (eyes containing a natural lens) have been included in this study, and only one eye per participant has been analysed. Each eye has been measured five times with the optical biometer and the results are summed up in Tab. 7.3. In the table we provide the

mean values for all the structural sizes that the biometer can measure as well as its standard deviations and the range of the measured quantities. The last two columns correspond to statistic estimators which are of optometric use, but not necessary for our model.

In the previous sections, we have presented a modeling of the eye based upon the assumption that the eyeball is spherically symmetric and can be modeled employing an average value for the elastic moduli. Certainly, a more accurate modelling of the eye structure can be done. Our model may easily be improved by, e.g., assigning different elastic properties to different parts of the eye. Indeed, it is possible to assign different elastic properties on a point-by-point basis, to account for the heterogeneity of the various eye constituents. And to add this structure, we need to know their measurements. Also, lifting up the assumption of spherical symmetry is trivial, since the decomposition of the vibrational modes into toroidal and spheroidal is still valid if we assume axial symmetry (an excellent approximation for the eyeball physiology).

Summarizing, we have presented a novel way of performing the analysis of the normal modes of an idealized human eye. For that, we have imported the analytical results developed in a number of areas of Physics, more precisely in the field of Gravitational Wave Physics. We have developed a simplified, spherically symmetric eyeball model, for which there exist analytic solutions for the eigenfrequencies for simple boundary conditions. We have used these solutions to calibrate our new finite-difference numerical code. Also, we have studied how the frequencies vary according to AL, and we have measured actual biometric data from human volunteers to improve our eye model.

Part IV

Conclusions and outlook

Chapter 8

Conclusions

We provide next a detailed, individual discussion of our results presented in the two chapters of Part II and the three chapters of Part III.

8.1 Scheduled Relaxation Jacobi method: improvements and applications

In Chap. 3, building upon the results of YM14, we have devised a new method for obtaining the optimal parameters for SRJ schemes applied to the numerical solution of ePDEs. Our new method reduces the complexity of the non-linear system of equations from which optimal parameters are computed.

We have shown that the new multilevel SRJ schemes keep improving the convergence performance index of the scheme, which means that increasing the value of P we obtain ever larger acceleration factors with respect to the Jacobi method. We report acceleration factors of a few hundreds and, in some cases, more than 1000 with respect to the Jacobi method if a sufficiently large number of points per dimension (namely, $N > 16000$) and number of levels are considered. For multidimensional applications increasing P from 5 (original maximum number of levels in YM14) to 15 yields a decrease in the computational cost by factors $\sim 2-3$ for the largest resolutions considered here. Since even larger resolutions result in correspondingly larger gains, we note that the benefit of employing SRJ algorithms with $P = 15$ will be really advantageous in three-dimensional supercomputing applications. In such cases it is worthwhile employing SRJ schemes with a larger number of levels than those originally proposed in YM14, specially considering that there is no extra

complexity in the algorithm implementation for any $P \geq 2$ once the weights for large values of P are known. One advantage of SRJ algorithms is that the optimal set of coefficients for a given number of points per dimension, N_1, may be reused for computational grids with larger a number of points per dimension, N_2 (YM14). The set of SRJ parameters thus employed is not optimal for the grid with the larger number of points per dimension though. However, the application of the SRJ parameter set to the grid with N_2 points per dimension will still speed up the iterative solution with respect to the base Jacobi scheme, even considerably if N_2 is not much larger than N_1. Thus, in App. B, we have provided a comprehensive set of tables with all the necessary optimal coefficients for a dense set of different numbers of points per dimension.

Mainly due to the fact that we have derived analytic solutions for part of the unknowns, our new method reduces the stiffness of the non-linear system of equations from which optimal parameters are computed, allowing us to obtain new SRJ methods for up to $P = 15$ and arbitrarily large number of points per dimension N.

From this number of levels, new problems arise, which hinder the computation of optimal coefficients at relatively low number of discretization points. These problems are related to the fact that for large values of P the solution to the problem are very sensitive to tiny changes in the smaller wave numbers, and small numerical errors prevent the successful evaluation of the solution of even the simplified system of non-linear equations resulting from the algebraic simplifications we have shown here. These problems were solved with the development of the Chebyshev-Jacobi method presented in Chap. 4

8.2 On the equivalence between the Scheduled Relaxation Jacobi method and Richardson's non-stationary method

In Chap. 4 we have obtained the optimal coefficients for the SRJ method to solve linear systems arising in the finite difference discretization of elliptic problems in the case $P = M$, i.e., using each weight only once per cycle. We call the resulting method Chebyshev-Jacobi (CJM). We have proven that these are the optimal coefficients for the general case, where we fix P but allow for repetitions of the coefficients ($P \leq M$). Furthermore, we have provided a simple estimate to compute the optimal value of M to reduce the initial residual by a prescribed factor.

We have tested the performance of the CJM with one simple example in 2 dimensions, showing that the analytically derived amplification factors can be obtained in practice. When comparing the optimal $P = M$ set of coefficients with those in YM14 and in Chap. 3, the CJM always gives better results, i.e., it achieves a larger reduction of the residual for the same number of iterations M. Additionally, the new coefficients can be computed analytically, as a function of M, $\kappa_{\max}$, and $\kappa_{\min}$, which avoids the numerical resolution of the minimization problem involved in Chap. 3 on the SRJ. The result is a numerical method that is easy to implement, and where all necessary coefficients can easily be calculated given the grid size, boundary conditions and tolerance of the elliptic problem at hand *before* the actual iteration procedure is even started.

Following the same philosophy that inspired the development of SRJ methods, we have found that the case $P = M$ results in an iterative method nearly equivalent to the non-stationary Richardson method as implemented by Young [Young 1953]. More specifically, in the implementation of Young, the coefficients ω_n are taken to be the reciprocals of the roots of the corresponding Chebyshev polynomials in the interval bounding the spectrum of eigenvalues of the matrix (A) of the linear system. Furthermore, inspired by the same ideas as in the original SRJ methods, the actual minimum and maximum eigenvalues of A do not need to be explicitly computed. Instead, we resort to a (much simpler) von Neumann analysis of the linear system, which yields the values of the $\kappa_{\min}$ and $\kappa_{\max}$ that *replace* the values of the minimum and maximum eigenvalues of A.[1] The key to our success in the practical implementation of the Chebyshev-Jacobi methods stems from a suitable ordering (or scheduling) of the weights ω_n in the algorithm. Though other orderings have also been shown to work, our choice clearly limits the growth of round-off errors when the number of iterations is large. This ordering is inherited from the SRJ schemes.

We have also tested the performance of the CJM for more than second-order discretizations of the elliptic Laplacian operator. These cases are especially involved since the matrix of iteration cannot be consistently ordered. Thus, Young's theory cannot be employed to find the value of the optimal weight of a SOR scheme applied to the resulting problems. For the particular case of the 9-points discretization of the Laplacian, even though the iteration matrix cannot be consistently ordered, Adams [Adams, LeVeque, and Young 1988] found the optimal weight for the corresponding SOR scheme in a rather involved

[1] We note that many other iterative schemes rely on a dynamic choice of the relaxation parameter or on dynamic preconditioning techniques and may be applied to matrices that come from discretizations of physical problems over generic grids (see e.g. [Saad and Schultz 1985, Saad 2003, Saad 1985, Dehghan and Hajarian 2011, Antuono and Colicchio 2016]).

derivation. Comparing the results for the numerical solution of a simple Poisson-like problem of the SOR method derived by Adams and the CJM we obtain here, for the same 9-points discretization of the Laplacian, it is evident that both methods perform quite similarly (though the optimal SOR scheme is still slightly better). However, the SOR method requires a multi-coloring parallelization strategy with up to 72 four-color orderings (each with different performance), when applied to the 9-points discretizations of the Laplacian operator. The parallelization strategy is even more intricate when a 17-points discretization of the Laplacian is used. In contrast, CJM methods are trivially parallelizable and do not require any multi-coloring strategy. Thus, we conclude that the slightly smaller performance difference between the CJM and the SOR method in sequential applications is easily outbalanced in parallel implementations of the former method. Furthermore, we have shown that employing higher-order discretizations of the Laplacian operator is very advantageous to reduce both the number of iterations and the computational time needed to reach a preestablished real error goal (i.e., the true error one makes comparing the exact solution of a problem with the numerical approximation of it). Given the stencil increase needed to implement a 17-points discretization of the Laplacian, we infer that a parallel implementation of this method may require a very modest increment in the number of zones transferred as internal boundaries among different computational subdomains. Hence, applying high-order discretizations of the Laplacian is ideally suited for problems that combine the solution of elliptic and hyperbolic systems of coupled equations (e.g., as in the case of Euler-Poisson systems dealing with self-gravitating fluids).

8.3 Sequential applications in astrophysics

In Chap. 5 we have applied the SRJ method and the CJM to the resolution of some problems of interest in astrophysics. We have limited ourselves to a sequential implementation of the method. Currently, we have reached acceleration factors that have made that the SRJ methods become competitive (depending on the dimensionality of the problem and its size) with, e.g., spectral methods for the solution of some ePDEs. The comparison has been done for the solution of the Poisson equation in spherical coordinates in 1D, 2D and 3D.

We find that for 1D Poisson-like problems, the fastest method of solution, of the ones tested, is the direct inversion method implemented in `LAPACK`. This happens because, in realistic applications in which the Poisson equation has to be solved multiple times, the LU decomposition of the matrix solver, where most

of the computational work is done, needs to be performed once, and thus it can be stored for the rest of the computations.

In 2D, the best performing method depends on whether our initial guess is close to the actual solution or far off. In realistic applications, where ePDEs are coupled to systems of hyperbolic PDEs, the solution from a previous time iteration does not change significantly over the course of a single timestep. In such conditions, the `LAPACK` libraries are the best performing. However, spectral methods are advantageous if, in 2D, the initial values are far from the actual solution of the problem. We further note that in realistic coupled systems, and for a relatively large number of points per dimension ($N > 500$), the SRJ methods are competitive with spectral ones.

In 3D applications, we find that the total computational cost of SRJ methods scales as N^4, i.e, as in the case of spectral methods. Considering that (i) applying direct inversion methods to 3D problems may be unfeasible because of memory restrictions, and that (ii) SRJ methods can be parallelized straightforwardly (much more easily than, e.g., spectral or multigrid methods), we foresee that they are a competitive alternative for the solution of elliptic problems in supercomputing applications and in 3D.

Finally, we outline that the easy implementation of complex boundary conditions in SRJ and CJM is also an advantage with respect to other existing alternatives. This fact has been demonstrated solving the Grad-Shafranov equation on a very involved domain and including mixed-type boundary conditions. Indeed, the versatility of the CJM to deal with arbitrary boundaries mades it suitable to tackle realistic problems in Astrophysics.

8.4 Parallel applications in astrophysics

In previous chapters, we have delineated the potencial degree of parallelism of the CJM method as a major advantage over other competing algorithms to solve linear systems of equations resulting from the discretization of elliptic systems of partial differential equations. Building upon the basic Jacobi method, the parallel implementation of the CJM is as simple as that of the former method. In Chap. 6 we have materialized our previous claims and have presented an implementation of the CJM using a purely MPI implementation, a tridimensional implementation using openMP / MPI hybrid implementation and using GPUs.

We have tested the MPI ported CJM algorithm in two different MIMD architectures, one with distributed memory and one with shared memory. We have compared both times and scalability of the CJM with respect direct inversion

solving an astrophysical problem in spherical coordinates. The parallelisation strategy has been to split the grid in domains and distribute the workload among processors. The iterative problem is then solved in the whole grid and domain boundaries are communicated at each iteration step. We have seen, extrapolating, that for the high resolution simulation, from 3000 processors onwards, the execution time using the CJM is expected to be lower than the direct inversion method in both ab initio and the realistic calculations. The speedup for CJM is ideal, which does not happen with direct inversion. Using an LU-based direct version method was problematic at high resolutions when the number of partitions were small in some dimension, because the matrix no longer fit in memory.

The solution of the Poisson equation to find out the electric potential distribution created by a sphere with uniform charge has been used as a test-bed in 3D. We have taken advantage of an existing platform for numerical relativity. We have used a hybrid OpenMP/MPI implementation of the CJM. The simulations have maintained an ideal speedup up to 64 cores with $N = 256$ in 3D. Both SOR and CJM are more than an order of magnitude faster than the Jacobi and Gauss-Seidel methods. Nonetheless, the formula for calculating ω_{opt} for SOR only applies when the original matrix of the linear system is consistently ordered; and CJM is trivially parallelized, while SOR requires multicolor schemes for a successful parallelization. The case of octant symmetry leads to an NCO matrix and, hence, there is no analytic expression to calculate ω_{opt} for SOR. We test a sequence of values of ω to empirically estimate its optimal value for the given problem. In the latter case, the CJM performs better than SOR for the set of tested values of ω.

We have also tested the CUDA ported Jacobi and CJM algorithms in two different GPU architectures. The differences in actual computing time reduce significantly as either the grid size increases or the number of points employed in the discretization of the Laplacian grows. We find that it is possible to speed up by several orders of magnitude the classical Jacobi method thanks to the use of the parallel implementation of the CJM on GPUs.

Moreover, we have illustrated the benefits from using the parallel implementation of the CJM over GPUs in combination with a high-order discretization of the Laplacian operator in a test problem. We conclude that it is always advantageous employing high-order discretizations of the elliptic operator since they requiere less iterations and less computational time to reach the same real error goal. This conclusion is independent of the parallel implementation of any of the methods we have tested in this work. However, the combination of

high-order discretization of the elliptic operators and the CJM implemented on GPUs results in an extremely powerful method for practical applications.

8.5 Beyond astrophysics: modelling the human eye

We have presented an analysis of the normal modes of an idealized human eye. For this purpose we have imported the analytical results developed in a number of areas of Physics, more precisely in the field of gravitational wave physics.

We have shown that beyond the mechanical characterization of the eyeball components, the normal vibrational modes of the eye could be involved in physiological processes like, e.g., the accommodation. We have developed a simplified, spherically symmetric eyeball model, for which there exist analytic solutions for the eigenfrequencies for simple boundary conditions. We have used these solutions to calibrate our new finite-difference numerical code. Also, we have studied the dependence of the frequencies with respect to the axial length. These results have been contrasted with actually measured biometric data of a selected sample of patients with a twofold goal. On the one hand, calibrating the model with real data and, on the other hand, feeding back the data on the eyeball model. The latter aim is still on-going.

The normal vibrational modes of the eye could be involved in some physiological processes, e.g. the accommodation. Accommodation occurs through changes in the shape and thickness of the crystalline lens. The thickness and the curvature of the lens increase, causing an increase in the eye's optical power. Since it is a muscle-induced activity, accommodation is a highly fluctuant and dynamic process. These fluctuations are related to the fluctuations in ocular aberrations, and occur with corresponding frequencies [Campbell, Robson, and Westheimer 1959, Charman and Heron 1988, Dubra 2004]. The microfluctuations of accommodation play an important role in the variability of the optical quality of the eye. There are two main components of the accommodation response: a low frequency component ($< 0.5\,\mathrm{Hz}$), which corresponds to the drift in the accommodation response, and a peak at higher frequency, in the $1 - 2\,\mathrm{Hz}$ band [Campbell, Robson, and Westheimer 1959, Charman and Heron 1988]. The vibrational eyeball modes we have considered –having the lowest frequencies– seem to happen on timescales of a few milliseconds. The exact way in which the normal eyeball modes are correlated with the accommodation process is beyond the scope of this thesis. However, we anticipate that to tackle such study

one would need to improve our current model by, at least, differentiating in the eyeball model the cornea-sclera, the vitreous humour and the lens. Towards this direction we will conduct our future research.

Chapter 9

Outlook

During the development of the thesis there have been emerging different promising ideas that, due to lack of time, have been set aside for the future. Here we discuss some of these ideas.

Limits in the performance in parallel applications. We would like to bring to the limit, both in the number of processors and in the mesh size, the parallel implementations discussed in this thesis, to assess its performance in this regime. For example, in our application to compute the gravitational potential in progenitors of long GRB simulations, we would like to verify the scalability up to thousands of processors. For this purpose we will need to use sufficiently large mesh sizes to have enough work load per processor. This test will have to be run in high performance computing facilities, since they are not possible in our local machines. In the case of the GPUs, we plan to increase the mesh size to fill the available memory in the GPU and see if the same trend that we observed in in thesis continues in terms of processing times. On the other hand, it is evident that for very large problem sizes, the elliptic problems we aim to address in, e.g. Euler-Poisson systems, will need to be split across several GPU devices running in parallel. We will test this additional parallelization degree once adequate computational resources become available.

Solving XCFC equations using CJM with massive parallelism. In the future we plan to use the methods developed in this thesis to solve the Einstein field equations in the XCFC formulation. In this formulation we have a hierarchy of ten decoupled elliptic equations. A parallel implementation could be of interest

for several astrophysical scenarios, including core collapse supernovae and the evolution of isolated neutron stars.

The goal would be to test the different implementations that we have presented in this thesis (MPI, openMP / MPI and GPUs) to study their performance in real application of three-dimensional hydrodynamics simulations.

We would also like to study the behaviour of the scheme, applied to the XCFC case, in other types of coordinates, different to the spherical coordinates used in the past. For instance, we would like to test spherical coordinates with a compactified radial coordinate extending to radial infinity, using different compactification functions. Likewise, we also aim to test our new method employing Cartesian and bispherical coordinates, either with or without compactification in the XCFC formulation.

Optimal weight, ω, for SOR method with NCO matrices. One of the classical methods for solving linear systems of equations, especially those systems obtained in the numerical solution of PDEs, is the SOR method. Although its rate of convergence can be very good, compared to other methods presented in this thesis, it depends critically on the value of the relaxation factor ω used by the algorithm. Moreover, it is generally difficult to obtain this value. We are developing a methodology to find the optimal weight for the SOR iteration based on the knowledge learnt from Chebyshev-Jacobi schemes. Preliminary results show that it is possible to compute the optimal value of ω straightforwardly, not only for the series of CO matrices with known solution presented in this thesis, but also for some cases of NCO matrices. In particular, we obtain an expression, $\omega_{opt}(N) = 32/\left(4+3\sqrt{1-\cos\frac{\pi}{N}}\right)^2$, for the case of the 9-points discretization of the Laplacian operator, employing $\alpha = 2/3$. This result agrees numerically with one computed by LeVeque and Trefethen 1988, namely $\omega_{opt}(N) = 2 - \frac{2.116\pi}{N}$ N being the numerical resolution). We are currently working on the results for the 17-points discretization of the Laplacian and for more general cases. These more general cases, corresponding to high-order discretizations of the Laplacian operator, are relatively easy to obtain in the framework devised in this thesis. Such framework has been formulated in terms of the parameter α, which accounts for how to combine points at different distances from a given node in the discrete mesh where the numerical solution is sought.

25-points discretizacion of the Laplacian. We are studying higher order schemes with complete 25-points discretizations of the Laplacian. One way of obtaining this type of discretizations is doing convex combinations between

the discretization of the Laplacian operator using different rotated stencils (see Fig. 9.1). Accounting for the previous comments, we can write the stencil of the Laplacian discretization as

$$\alpha S_+ + \beta S_\times + \gamma S_{\frac{\pi}{6}} + (1-\alpha-\beta-\gamma) S_{\frac{\pi}{3}} \,, \tag{9.1}$$

being $\alpha \geq 1/2$, which shall be regarded as a generalization of Eq. (4.19).

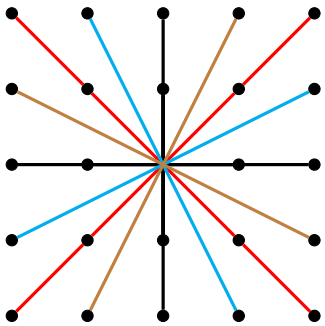


Figure 9.1 Schematic representation of the 25-point stencils. The colored lines correspond to the standard stencil S_+ (black) and rotated stencils $S_\times$ (red) , $S_{\frac{\pi}{6}}$ (cyan) and $S_{\frac{\pi}{3}}$ (brown), respectively. To be compared with Fig. 4.3.

Without loss of generality[1], we can write α, β and γ as we did in Sec. 4.4, i.e., as a rational numbers $\alpha := \frac{a}{d}$, $\beta := \frac{b}{d}$ and $\gamma := \frac{c}{d}$. Assuming, in order to obtain handy expressions in the von Neumann study of stability, that $c = \frac{d-a-b}{2}$, i.e., that we are receiving the same contribution from $S_{\frac{\pi}{6}}$ and from $S_{\frac{\pi}{3}}$, we obtain the following expression for $\kappa_{\min}$

$$\begin{aligned}
\kappa_{\min} = \frac{1}{150a + 75b + 12(d-a-b)} \Bigg\{ \\
-10a \left[\sin^2 \frac{\pi}{N_x} + \sin^2 \frac{\pi}{N_y} \right] + 160a \left[\sin^2 \frac{\pi}{2N_x} + \sin^2 \frac{\pi}{2N_y} \right] \\
-5b \left[\sin^2 \left(\frac{\pi}{N_x} + \frac{\pi}{N_y} \right) + \sin^2 \left(\frac{\pi}{N_x} - \frac{\pi}{N_y} \right) \right) \Bigg] \\
+80b \left[\sin^2 \left(\frac{\pi}{2N_x} + \frac{\pi}{2N_y} \right) + \sin^2 \left(\frac{\pi}{2N_x} - \frac{\pi}{2N_y} \right) \right] \\
+6(d-a-b) \left[\sin^2 \left(\frac{\pi}{N_x} + \frac{\pi}{2N_y} \right) + \sin^2 \left(\frac{\pi}{N_x} - \frac{\pi}{2N_y} \right) \right.
\end{aligned}$$

[1]Note that the general case $\alpha := \frac{a'}{d'}$, $\beta := \frac{b'}{e'}$ and $\gamma := \frac{c'}{f'}$ always can be reduced to the previous case simply computing the least common multiple of the denominators $d := [d', e', f']$. Then $\alpha = \frac{\frac{a'}{d'} d}{d}$, $\beta := \frac{\frac{b'}{e'} d}{d}$ and $\gamma := \frac{\frac{c'}{f'} d}{d}$.

$$+\sin^2\left(\frac{\pi}{2N_x}+\frac{\pi}{N_y}\right)+\sin^2\left(\frac{\pi}{2N_x}-\frac{\pi}{N_y}\right)\Big]\Big\}$$

and for $\kappa_{\max}$

$$\kappa_{\max}=\frac{320a+24(d-a-b)}{150a+75b+12(d-a-b)}$$

The framework to combine stencils here provided seems to be useful for high-order extensions of the Laplacian discretization. As the implementation we have of the method on GPUs is of stencil type, our infrastructure is already prepared to handle such large discretizations. However, in MIMD architectures, we will have to developed further our computational routines to incorporate larger discretization stencils.

CJM for non positive-definite matrices. As we mentioned when finding the eigenvalues in the modes of vibration of the eye, the CJM cannot be applied to that case because we have a non positive-definite matrix. The problem with non positive-definite matrices is that their eigenvalues do not have a definite sign and, therefore, may cross zero. In [Anderssen and Golub 1972] there is an appendix with a translation of a work of Lebedev that explains how to apply iterative methods to operators, whose eigenvalue spectrum spans several intervals, possibly of different sign. We have found a possible way to combine the former strategy with our CJM. It is necessary to remember that we do not work with eigenvalues but with an approximation to them obtained from a von Neumann analysis of stability. In our particular case, the problem consists in finding two intervals that exclude zero and, at the same time, include the maximum negative eigenvalue and the minimum positive one. With this, and after matching the lengths of the two intervals, we have $[a_1, a_2]$ in the negative part and $[a_3, a_4]$ in the positive part, Lebedev explains how to compose a parabola[2] $Q_2(t) = t(t-2c)$, with $c = (a_2 + a_3)/2$, with the transformed Chebyshev polynomial that we use for obtaining the weights of our CJM schemes $\tilde{G}_M(\tilde{\kappa})$. Thereby, conceptually, we have that the weights are the inverse of the zeros of the polynomial function $P_{2M}(t) = \tilde{G}_M(Q_2(t))$, with M being the length of the cycle we want to use. This scheme will have a behavior similar to the CJM but avoiding the problem that the operator has a part of the spectrum with negative sign. We are testing it currently. We point out that if our attempts of generalization are successful, we may have paved the way to employ the CJM as a generic solver for any linear system, regardless it comes from the discretization of ePDEs or not.

[2]In our case, since we have only two intervals, the construction of Lebedev shall result in a second-order polynomial.

Understanding the good behavior of the CJM for non-Cartesian operators. We have used our scheme in numerous tests involving non-Cartesian coordinates. Despite of the fact that the von Neumann analysis that we have used is, strictly speaking, only valid in Cartesian coordinates, we have observed a good performance in the solution of the elliptic problems at hand. A possible step forward would be trying to apply the von Neumann stability analysis directly to these operators and hoping to understand this behaviour. We are currently working in the analysis of the Laplacian operator in spherical polar coordinates.

Relaxation scheme for multigrid algorithms Multigrid schemes find themselves among the most popular and efficient iterative solvers for linear systems. In this schemes one needs to apply some relaxation algorithm in order to reach the solution. Usually the Jacobi method is used in this relaxation step. We are currently working in a multigrid implementation that uses CJM as relaxation algorithm, and we plan to compare its performance with respect to the Jacobi method.

Beyond these previous items, in which we have already some ideas to develop, there are a number of other questions, which we may also address in future work. For instance:

- The methodology developed in this thesis has been applied to finite difference methods. It would be very interesting to consider its potential extension to other methods of solving ePDEs, e.g., finite element methods or discontinuous Galerkin schemes.
- Both SRJ and CJ methods have been applied to regular (cuadrangular) meshes. As was already pointed by YM14, the method can be applied also to non-uniform cuadrangular meshes. Thus, it would be viable to apply it in AMR grids. Besides that, it will be certainly useful considering extensions of the method to more general grids, namely general unstructured (simplex) meshes (e.g., triangles and tetrahedra).

As we have seen in the last paragraphs, there is ample room for improvement and application of SRJ schemes and, specially of the CJM. We point out that the methods here described suffer from the same principle problem as the SOR method. They require the calculation of the optimal parameters for each problem size and discretization operator. However, in all applications we have considered so far, the evaluation of the optimal parameters has been doable and the computational optimization worth. We take this fact as hint pointing

towards the relevance of the methodology developed in this thesis, as well as an stimulus to continue our research in this promising line and in its applications in Astrophysics. Many of these applications are ongoing and will be timely published elsewhere.

Part V

Appendices

Appendix A

About the weights

A.1 Ordering of the weights

As we point out in Sect. 4.2, the ordering of the weights ω_n is the key to avoid the pile up of roundoff errors. In this appendix, we show that the ordering provided by Yang and Mittal 2014 for SRJ schemes, and that we also use for the optimal $P = M$ schemes, differs from the one suggested by other authors.

Lebedev & Finogenov Lebedev and Finogenov 1971 provided orderings for the cases in which the number of weights is a power of 2. Translated to our notation, we shall have $M = 2^r$, $r = 0, 1, \ldots$. In such a case, let the ordering of the set $(\omega_1, \omega_2, \ldots, \omega_M)$ as obtained from Eq. (4.10), be mapped with the vector of indices $(1, 2, \ldots, M)$. Let us consider an integer permutation of the vector of indices of order M, $\Xi_M := (j_1, j_2, \ldots, j_M)$, where $(1 \leq j_k \leq M, j_i \neq j_k)$, which are constructed according to the following recurrence relation:

$$\Xi_{2^0} = \Xi_1 := (1) \quad \text{and} \quad \Xi_{2^{r-1}} := (j_1, j_2, \ldots, j_{2^{r-1}}) \tag{A.1}$$

$$\Xi_{2^r} = \Xi_M := (j_1, 2^r + 1 - j_1, j_2, 2^r + 1 - j_2, \ldots, j_{2^{r-1}}, 2^r + 1 - j_{2^{r-1}}) \tag{A.2}$$

In particular, we have,

$$\begin{aligned}
\Xi_2 &= (1, 2), \\
\Xi_4 &= (1, 4, 2, 3), \\
\Xi_8 &= (1, 8, 4, 5, 2, 7, 3, 6), \\
\Xi_{16} &= (1, 16, 8, 9, 4, 13, 5, 12, 2, 15, 7, 10, 3, 14, 6, 11).
\end{aligned}$$

In contrast, we can obtain different SRJ schemes, and correspondingly, different orderings, for the same number of weights, because of the later depend on the number of points employed in the discretization (see Eq. 4.10). Furthermore, the ordering also depends on the tolerance goal, σ (which sets the value of M; Eq. 4.14). Next we list some of the orderings we can obtain for different discretizations (annotated in parenthesis in the form $N_x \times N_y$) and values of σ:

$$\begin{aligned}
\Xi_2^{\mathrm{SRJ}} &= (1,2), \\
\Xi_4^{\mathrm{SRJ}} &= (1,4,3,2), \\
\Xi_8^{\mathrm{SRJ}} &= (1,8,5,2,3,7,4,6) \text{ for } (4\times 4, \sigma = 0.01), \\
&\quad\ (1,8,5,3,6,2,7,4) \text{ for } (8\times 8, \sigma = 0.15), \\
\Xi_{16}^{\mathrm{SRJ}} &= (1,15,9,2,12,3,4,13,5,6,7,8,10,11,14,16) \text{ for } (4\times 4, \sigma = 2\times 10^{-5}), \\
&\quad\ (1,16,9,6,12,3,14,7,10,4,13,5,15,8,2,11) \text{ for } (8\times 8, \sigma = 6\times 10^{-3}).
\end{aligned}$$

which obviously differ from the orderings Ξ_j for $j \geq 4$.

We note that Nikolaev and Samarskii 1972 provided also orderings for arbitrary values of M, which coincide with those of Lebedev & Finogenov Lebedev and Finogenov 1971 when M is a power of 2 (i.e., $M = 2^r$). Finally, more recently, Lebedev & Finogenov Lebedev and Finogenov 2002 have extended their previous work to a larger number of cases (e.g., $M = 2^r 3^s$) and applied also to Chebyshev iterative methods. We remark that the SRJ ordering of the weights can be applied to arbitrary values of M.

A.2 Properties of the weights

In this appendix we show some algebraic properties of the weights of the CJM. The first one is that the harmonic mean of the weights equals the average of the maximum and minimum weight numbers:

Theorem 3. *Let ω_i be the weights given by Eq. (4.10). Then it holds that*

$$\frac{1}{M}\sum_{i=1}^{M} \omega_i^{-1} = \frac{\kappa_{\max} + \kappa_{\min}}{2}. \tag{A.3}$$

Proof:

$$\frac{1}{M}\sum_{i=1}^{M} \omega_i^{-1} = \frac{(\kappa_{\max} + \kappa_{\min})}{2} - \frac{(\kappa_{\max} - \kappa_{\min})}{2M}\sum_{i=1}^{M} \cos\left(\frac{\pi(i-1/2)}{M}\right). \tag{A.4}$$

Let $j \in [1, M/2]$. Since

$$\cos\left(\frac{\pi(j-1/2)}{M}\right) = -\cos\left(\frac{\pi((M-j+1)-1/2)}{M}\right), \tag{A.5}$$

all the terms in the summation cancel out, except the central one in case M is odd. In this last case, $M = 2n+1$, and the only remaining term is $\cos\left(\frac{\pi(n+1/2)}{2n+1}\right) = \cos\left(\frac{\pi}{2}\right) = 0$. In general, the summation reads

$$\frac{1}{M}\sum_{i=1}^{M}\omega_i^{-1} = \frac{\kappa_{\max}+\kappa_{\min}}{2}. \tag{A.6}$$

Corollari: Since the relation between the weights of the stationary RM and the CJM is $\hat{\omega} = \omega d^{-1}$, where $D = \mathrm{diag}(A)$, having all its elements equal to d, and since $\hat{\omega} = 2/(a+b)$, where $a = \min(\lambda_i)$ and $b = \max(\lambda_i)$, being λ_i the eigenvalues of matrix A, it turns out that

$$\frac{2d^{-1}}{\kappa_{\max}+\kappa_{\min}} = \frac{2}{a+b} = \hat{\omega}. \tag{A.7}$$

Theorem 4. *Let ω_i be the weights given by Eq. (4.10). Then it holds that*

$$\lim_{n\to+\infty}\left[\prod_{i=1}^{n}\omega_i^{-1}\right]^{1/n} = \left(\frac{\sqrt{\kappa_{\max}}+\sqrt{\kappa_{\min}}}{2}\right)^2. \tag{A.8}$$

Proof:

Let us define $A = (\kappa_{\max}+\kappa_{\min})/2$ and $B = (\kappa_{\max}-\kappa_{\min})/2$. It is well known that the Chebyshev polynomials of first kind of degree n, $T_n(x)$, satisfy the following recurrence relation:

$$T_0(x) = 1;\;\; T_1(x) = x;\;\; T_n(x) = 2x\,T_n(x) - T_{n-1}(x), n > 2. \tag{A.9}$$

From this property, it is easy to check that the leading coefficient of $T_n(x)$ is 2^{n-1}. Taking into account the leading coefficient and the roots of $T_n(x)$ from Eq.(4.8), we get that

$$T_n(x) = 2^{n-1}\prod_{k=1}^{n}\left\{x - \cos\left[\frac{(2k-1)\pi}{2n}\right]\right\}. \tag{A.10}$$

Therefore,

$$\begin{aligned}\prod_{i=1}^{n}\omega_i^{-1} &= \left[A - B\cos\left(\frac{\pi}{2n}\right)\right]\left[A - B\cos\left(\frac{3\pi}{2n}\right)\right]\ldots\left[A - B\cos\left(\frac{(2n-1)\pi}{2n}\right)\right] \\ &= \frac{B^n}{2^{n-1}}T_n\left(\frac{A}{B}\right)\end{aligned}$$

$$= \frac{B^n}{2^n}\left[\left(\frac{A}{B}-\sqrt{\frac{A^2}{B^2}-1}\right)^n+\left(\frac{A}{B}+\sqrt{\frac{A^2}{B^2}-1}\right)^n\right], \tag{A.11}$$

where last equality uses the explicit expression of $T_n(x)$ for $x = A/B > 1$. From this equality, we can bound the geometrical mean of the inverse of the weights ω_i:

$$\frac{B}{2}\left(\frac{A}{B}+\sqrt{\frac{A^2}{B^2}-1}\right) \leq \left(\prod_{i=1}^{n}\omega_i^{-1}\right)^{1/n} \leq \frac{B}{2^{(n-1)/n}}\left(\frac{A}{B}+\sqrt{\frac{A^2}{B^2}-1}\right). \tag{A.12}$$

Taking limits for $n \to \infty$, we obtain that

$$\lim_{n\to+\infty}\left[\prod_{i=1}^{n}\omega_i^{-1}\right]^{1/n} = \frac{1}{2}\left(A+\sqrt{A^2-B^2}\right) = \left(\frac{\sqrt{\kappa_{\max}}+\sqrt{\kappa_{\min}}}{2}\right)^2. \tag{A.13}$$

Appendix B

Compendium of parameters of optimal SRJ schemes

Table B.1 Parameters $\boldsymbol{w}$, $\boldsymbol{\beta}$ and the estimation of the convergence performance index $\rho = \sum_{i=1}^{P} \omega_i \beta_i$ of the $P = 2$, $P = 3$ and $P = 4$ schemes for a number of values of N.

N		$P = 2$	$P = 3$	$P = 4$
100	$\boldsymbol{\omega}$	{321.074, 0.968096}	{1420.73, 30.0648, 0.845599}	{2308.12, 162.259, 8.50839, 0.732499}
	$\boldsymbol{\beta}$	{0.00993673, 0.990063}	{0.00502828, 0.0729552, 0.922017}	{0.00430412, 0.0245487, 0.158309, 0.812838}
	ρ	4.15	10.12	15.86
150	$\boldsymbol{\omega}$	{509.976, 0.977667}	{2724.66, 41.8246, 0.870558}	{4707.62, 259.325, 10.9382, 0.756243}
	$\boldsymbol{\beta}$	{0.0064850, 0.99352}	{0.00304593, 0.0574955, 0.939459}	{0.00268244, 0.0182254, 0.138417, 0.840676}
	ρ	4.28	11.52	19.50
200	$\boldsymbol{\omega}$	{704.099, 0.982735}	{4295., 52.6521, 0.886485}	{7968.04, 368.694, 13.2257, 0.774085}
	$\boldsymbol{\beta}$	{0.00478657, 0.995213}	{0.00211898, 0.048302, 0.949579}	{0.00185082, 0.0144313, 0.124148, 0.85957}
	ρ	4.35	12.49	22.38
250	$\boldsymbol{\omega}$	{901.84, 0.985888}	{6090.23, 62.8089, 0.897814}	{11871.8, 481.378, 15.1867, 0.786453}
	$\boldsymbol{\beta}$	{0.00378101, 0.996219}	{0.00159331, 0.0420853, 0.956321}	{0.00139269, 0.0120297, 0.114584, 0.871993}
	ρ	4.39	13.21	24.75
300	$\boldsymbol{\omega}$	{1102.34, 0.988045}	{8082.34, 72.4478, 0.906414}	{16301., 591.753, 17.0536, 0.797245}
	$\boldsymbol{\beta}$	{0.00311809, 0.996882}	{0.00125948, 0.0375472, 0.961193}	{0.00110797, 0.0104108, 0.106471, 0.88201}
	ρ	4.42	13.77	26.74
350	$\boldsymbol{\omega}$	{1305.06, 0.989617}	{10250.9, 81.6684, 0.913233}	{21362.1, 707.502, 18.7691, 0.805757}
	$\boldsymbol{\beta}$	{0.00264906, 0.997351}	{0.001031, 0.0340609, 0.964908}	{0.000908394, 0.0091685, 0.100232, 0.889691}
	ρ	4.44	14.23	28.49
400	$\boldsymbol{\omega}$	{1509.63, 0.990814}	{12580.2, 90.5404, 0.918815}	{27421.7, 850.177, 20.3972, 0.812165}
	$\boldsymbol{\beta}$	{0.00230021, 0.9977}	{0.000866032, 0.0312827, 0.967851}	{0.000751992, 0.00802258, 0.0955775, 0.895648}
	ρ	4.46	14.62	30.12
450	$\boldsymbol{\omega}$	{1715.8, 0.991758}	{15057.7, 99.1151, 0.923493}	{35453.1, 1161.24, 24.1452, 0.825463}
	$\boldsymbol{\beta}$	{0.00203086, 0.997969}	{0.000742062, 0.0290068, 0.970251}	{0.000612978, 0.00627071, 0.0853787, 0.907738}
	ρ	4.47	14.94	31.82
500	$\boldsymbol{\omega}$	{1923.36, 0.992522}	{17673.1, 107.432, 0.927487}	{41329., 1177.71, 25.2645, 0.831242}
	$\boldsymbol{\beta}$	{0.00181679, 0.998183}	{0.000645947, 0.0271018, 0.972252}	{0.000546643, 0.00623392, 0.0815398, 0.91168}
	ρ	4.49	15.23	32.75

Table B.2 Parameters $\boldsymbol{w}$, $\boldsymbol{\beta}$ and the estimation of the convergence performance index $\rho = \sum_{i=1}^{P} \omega_i \beta_i$ of the $P = 5$ schemes for a number of values of N.

N		$P = 5$
100	$\boldsymbol{\omega}$	$\{2846.74, 411.781, 40.0941, 3.97003, 0.659793\}$
	$\boldsymbol{\beta}$	$\{0.00395334, 0.0134445, 0.0549429, 0.22302, 0.70464\}$
	ρ	20.34
150	$\boldsymbol{\omega}$	$\{6083.43, 723.916, 58.9841, 4.88096, 0.679269\}$
	$\boldsymbol{\beta}$	$\{0.00248163, 0.00967631, 0.04472, 0.205462, 0.73766\}$
	ρ	26.24
200	$\boldsymbol{\omega}$	$\{10402.8, 1077.5, 77.4789, 5.6526, 0.693256\}$
	$\boldsymbol{\beta}$	$\{0.00176797, 0.00760734, 0.038404, 0.192846, 0.759375\}$
	ρ	31.18
250	$\boldsymbol{\omega}$	$\{15750.6, 1464.91, 95.6673, 6.33405, 0.704153\}$
	$\boldsymbol{\beta}$	$\{0.00135255, 0.00628676, 0.0340103, 0.183094, 0.775256\}$
	ρ	35.47
300	$\boldsymbol{\omega}$	$\{22085.5, 1881.21, 113.603, 6.95081, 0.713063\}$
	$\boldsymbol{\beta}$	$\{0.00108339, 0.00536588, 0.0307308, 0.175201, 0.787619\}$
	ρ	39.29
350	$\boldsymbol{\omega}$	$\{29373.9, 2322.91, 131.323, 7.51817, 0.720588\}$
	$\boldsymbol{\beta}$	$\{0.000896175, 0.00468493, 0.028165, 0.168605, 0.797649\}$
	ρ	42.75
400	$\boldsymbol{\omega}$	$\{37587.8, 2787.39, 148.854, 8.04621, 0.727091\}$
	$\boldsymbol{\beta}$	$\{0.000759202, 0.00415986, 0.0260888, 0.162962, 0.806030\}$
	ρ	45.91
450	$\boldsymbol{\omega}$	$\{46703.1, 3272.60, 166.218, 8.54196, 0.732811\}$
	$\boldsymbol{\beta}$	$\{0.000655107, 0.00374204, 0.0243656, 0.158048, 0.813189\}$
	ρ	48.84
500	$\boldsymbol{\omega}$	$\{56698.8, 3776.87, 183.430, 9.01057, 0.737910\}$
	$\boldsymbol{\beta}$	$\{0.000573622, 0.00340130, 0.0229066, 0.153708, 0.819411\}$
	ρ	51.56

Table B.3 Parameters for optimized $P = 6$ SRJ schemes for various values of N ($N = 100 + 50k$, $k = 0, \ldots, 18$)..

N	Optimal Scheme Parameters	ρ
100	$\boldsymbol{\omega} = \{3177.91, 734.924, 110.636, 15.7187, 2.41695, 0.614642\}$ $\boldsymbol{\beta} = \{0.00367219, 0.00906128, 0.0276578, 0.0863039, 0.261008, 0.612297\}$	23.75
150	$\boldsymbol{\omega} = \{6903.64, 1362.54, 176.155, 21.6587, 2.85787, 0.629548\}$ $\boldsymbol{\beta} = \{0.0023493, 0.00645108, 0.0217415, 0.0745986, 0.249258, 0.645601\}$	31.57
200	$\boldsymbol{\omega} = \{11953.5, 2106.17, 244.75, 27.1876, 3.22218, 0.640444\}$ $\boldsymbol{\beta} = \{0.00169767, 0.00503682, 0.018218, 0.0669113, 0.240064, 0.668072\}$	38.38
250	$\boldsymbol{\omega} = \{18283.1, 2948.76, 315.677, 32.4263, 3.53814, 0.649052\}$ $\boldsymbol{\beta} = \{0.00131373, 0.00414205, 0.01583, 0.0613214, 0.232565, 0.684828\}$	44.49
300	$\boldsymbol{\omega} = \{25858.1, 3878.89, 388.49, 37.4429, 3.82008, 0.656173\}$ $\boldsymbol{\beta} = \{0.00106245, 0.00352207, 0.0140831, 0.0570002, 0.226254, 0.698078\}$	50.06
350	$\boldsymbol{\omega} = \{34650.4, 4888.21, 462.888, 42.2805, 4.07647, 0.662248\}$ $\boldsymbol{\beta} = \{0.000886166, 0.00306586, 0.0127387, 0.0535196, 0.220818, 0.708972\}$	55.22
400	$\boldsymbol{\omega} = \{44636.2, 5970.3, 538.656, 46.9691, 4.31277, 0.667545\}$ $\boldsymbol{\beta} = \{0.000756216, 0.00271546, 0.0116658, 0.0506321, 0.216052, 0.718178\}$	60.04
450	$\boldsymbol{\omega} = \{55794.8, 7119.99, 615.629, 51.5305, 4.53277, 0.67224\}$ $\boldsymbol{\beta} = \{0.000656794, 0.00243753, 0.0107858, 0.0481827, 0.211815, 0.726123\}$	64.57
500	$\boldsymbol{\omega} = \{68108.3, 8333.03, 693.68, 55.9812, 4.7392, 0.676457\}$ $\boldsymbol{\beta} = \{0.000578495, 0.0022115, 0.0100486, 0.0460682, 0.208004, 0.733089\}$	68.86
550	$\boldsymbol{\omega} = \{81560.3, 9605.84, 772.706, 60.3341, 4.93412, 0.680283\}$ $\boldsymbol{\beta} = \{0.000515383, 0.00202394, 0.0094204, 0.044217, 0.204545, 0.739278\}$	72.94
600	$\boldsymbol{\omega} = \{96136.1, 10935.4, 852.621, 64.5997, 5.1191, 0.683785\}$ $\boldsymbol{\beta} = \{0.000463536, 0.00186573, 0.00887743, 0.0425775, 0.201382, 0.744834\}$	76.82
650	$\boldsymbol{\omega} = \{111822., 12318.9, 933.355, 68.7865, 5.29542, 0.687012\}$ $\boldsymbol{\beta} = \{0.000420261, 0.00173043, 0.00840264, 0.0411115, 0.198469, 0.749866\}$	80.55
700	$\boldsymbol{\omega} = \{128607., 13754.1, 1014.85, 72.9015, 5.46409, 0.690005\}$ $\boldsymbol{\beta} = \{0.000383651, 0.00161335, 0.00798331, 0.0397896, 0.195772, 0.754458\}$	84.12
750	$\boldsymbol{\omega} = \{146478., 15239., 1097.05, 76.9509, 5.62594, 0.692794\}$ $\boldsymbol{\beta} = \{0.000352319, 0.00151104, 0.00760979, 0.0385893, 0.193262, 0.758676\}$	87.56
800	$\boldsymbol{\omega} = \{165426., 16771.6, 1179.9, 80.9397, 5.78167, 0.695405\}$ $\boldsymbol{\beta} = \{0.000325232, 0.00142084, 0.00727458, 0.0374926, 0.190916, 0.76257\}$	90.88
850	$\boldsymbol{\omega} = \{185440., 18350.2, 1263.38, 84.8724, 5.93187, 0.69786\}$ $\boldsymbol{\beta} = \{0.000301608, 0.00134071, 0.00697179, 0.0364852, 0.188715, 0.766185\}$	94.09
900	$\boldsymbol{\omega} = \{206511., 19973.4, 1347.45, 88.7529, 6.07703, 0.700175\}$ $\boldsymbol{\beta} = \{0.000280844, 0.00126905, 0.00669671, 0.0355554, 0.186644, 0.769554\}$	97.20
950	$\boldsymbol{\omega} = \{228631., 21639.7, 1432.07, 92.5845, 6.21758, 0.702366\}$ $\boldsymbol{\beta} = \{0.000262466, 0.00120458, 0.0064455, 0.0346935, 0.184687, 0.772707\}$	100.21
1000	$\boldsymbol{\omega} = \{251790., 23347.8, 1517.22, 96.3704, 6.3539, 0.704445\}$ $\boldsymbol{\beta} = \{0.000246098, 0.00114626, 0.00621504, 0.0338915, 0.182835, 0.775666\}$	103.13

Table B.4 Parameters for optimized $P = 6$ SRJ schemes for various values of N ($N = 2^k$ $k = 5, \ldots, 15$).

N	Optimal Scheme Parameters	ρ
32	$\boldsymbol{\omega} = \{354.762, 126.248, 29.5834, 6.40284, 1.53177, 0.576323\}$ $\boldsymbol{\beta} = \{0.0119532, 0.0222174, 0.051387, 0.123168, 0.282127, 0.509147\}$	10.08
64	$\boldsymbol{\omega} = \{1349.25, 370.523, 66.1485, 11.0483, 2.01538, 0.59895\}$ $\boldsymbol{\beta} = \{0.0059103, 0.0130086, 0.0356183, 0.100213, 0.271745, 0.573505\}$	17.15
128	$\boldsymbol{\omega} = \{5098.06, 1070.77, 146.886, 19.1063, 2.67591, 0.62365\}$ $\boldsymbol{\beta} = \{0.00280225, 0.00737736, 0.0239167, 0.0790606, 0.254045, 0.632799\}$	28.28
256	$\boldsymbol{\omega} = \{19127., 3055.94, 324.322, 33.039, 3.57356, 0.649974\}$ $\boldsymbol{\beta} = \{0.00127813, 0.00405608, 0.0155927, 0.0607468, 0.231752, 0.686574\}$	45.18
512	$\boldsymbol{\omega} = \{71233.5, 8633.18, 712.561, 57.0344, 4.78697, 0.677408\}$ $\boldsymbol{\beta} = \{0.000562137, 0.00216338, 0.00988896, 0.0456021, 0.207144, 0.734639\}$	69.86
1024	$\boldsymbol{\omega} = \{263274.200, 24182.2023, 1558.26459, 98.1721442, 6.41792734, 0.70540635\}$ $\boldsymbol{\beta} = \{0.000238864, 0.00112020, 0.00611101, 0.0335258, 0.181980, 0.777025\}$	104.5
2048	$\boldsymbol{\omega} = \{965411.762, 67235.3401, 3391.89306, 168.347866, 8.59713962, 0.73342290\}$ $\boldsymbol{\beta} = \{0.0000982338, 0.000563635, 0.00368554, 0.0241846, 0.157526, 0.813942\}$	151.3
4096	$\boldsymbol{\omega} = \{3511588.84, 185687.226, 7348.16859, 287.430514, 11.4912775, 0.76094204\}$ $\boldsymbol{\beta} = \{0.0000391657, 0.000275898, 0.00217265, 0.0171466, 0.134626, 0.845740\}$	211.8
8192	$\boldsymbol{\omega} = \{12668072.6, 509620.165, 15841.4365, 488.419035, 15.3117934, 0.787502901\}$ $\boldsymbol{\beta} = \{0.0000151650, 0.000131569, 0.00125389, 0.0119658, 0.113783, 0.872850\}$	287.3
16384	$\boldsymbol{\omega} = \{45320546.3, 1390363.37, 33981.7837, 825.831823, 20.3259724, 0.81271835\}$ $\boldsymbol{\beta} = \{0.00000571292, 0.0000612205, 0.000709569, 0.00823073, 0.0952379, 0.895755\}$	377.6
32768	$\boldsymbol{\omega} = \{160790516, 3771703.63, 72530.6624, 1389.31261, 26.8709119, 0.83628657\}$ $\boldsymbol{\beta} = \{0.00000209794, 0.0000278446, 0.000394356, 0.00558784, 0.0790411, 0.914947\}$	481.6

Table B.5 Parameters for optimized $P = 7$ SRJ schemes for various values of N ($N = 100 + 50k$, $k = 0, \ldots, 18$).

N	Optimal Scheme Parameters	ρ
100	$\boldsymbol{\omega} = \{3392.32, 1082.34, 224.848, 42.594, 8.09726, 1.71942, 0.585658\}$ $\boldsymbol{\beta} = \{0.00342091, 0.0068221, 0.0169684, 0.0438593, 0.113205, 0.27876, 0.536965\}$	26.38
150	$\boldsymbol{\omega} = \{7440.09, 2083.58, 378.528, 63.3295, 10.6285, 1.97391, 0.59717\}$ $\boldsymbol{\beta} = \{0.00222099, 0.00482464, 0.0130515, 0.0364807, 0.101845, 0.272802, 0.568775\}$	35.79
200	$\boldsymbol{\omega} = \{12975.6, 3307.79, 547.016, 83.8798, 12.8988, 2.18024, 0.605662\}$ $\boldsymbol{\beta} = \{0.00162297, 0.00375204, 0.0107739, 0.0318451, 0.0940257, 0.267403, 0.590577\}$	44.19
250	$\boldsymbol{\omega} = \{19962.8, 4727.89, 727.329, 104.286, 14.9921, 2.35673, 0.612421\}$ $\boldsymbol{\beta} = \{0.00126735, 0.00307737, 0.00925585, 0.0285774, 0.0881481, 0.262644, 0.60703\}$	51.87
300	$\boldsymbol{\omega} = \{28373.8, 6325.18, 917.591, 124.573, 16.9537, 2.51252, 0.618048\}$ $\boldsymbol{\beta} = \{0.00103282, 0.00261183, 0.00815942, 0.0261112, 0.0834859, 0.258429, 0.62017\}$	59.01
350	$\boldsymbol{\omega} = \{38185.5, 8085.74, 1116.5, 144.759, 18.8121, 2.65293, 0.622874\}$ $\boldsymbol{\beta} = \{0.000867201, 0.00227035, 0.00732423, 0.0241631, 0.0796508, 0.254661, 0.631063\}$	65.71
400	$\boldsymbol{\omega} = \{49378.2, 9998.61, 1323.09, 164.856, 20.586, 2.78137, 0.627102\}$ $\boldsymbol{\beta} = \{0.000744401, 0.00200871, 0.00666341, 0.0225732, 0.0764118, 0.251261, 0.640337\}$	72.05
450	$\boldsymbol{\omega} = \{61935., 12054.9, 1536.61, 184.873, 22.2891, 2.90016, 0.630867\}$ $\boldsymbol{\beta} = \{0.000649961, 0.0018016, 0.00612545, 0.0212434, 0.0736213, 0.248167, 0.648391\}$	78.08
500	$\boldsymbol{\omega} = \{75840.7, 14247.2, 1756.46, 204.819, 23.9316, 3.01099, 0.634261\}$ $\boldsymbol{\beta} = \{0.000575235, 0.00163343, 0.00567765, 0.0201097, 0.0711791, 0.24533, 0.655495\}$	83.85
550	$\boldsymbol{\omega} = \{91082., 16569.2, 1982.16, 224.699, 25.5213, 3.11509, 0.637353\}$ $\boldsymbol{\beta} = \{0.000514743, 0.00149409, 0.00529822, 0.0191281, 0.0690147, 0.242712, 0.661838\}$	89.38
600	$\boldsymbol{\omega} = \{107646., 19015.4, 2213.29, 244.52, 27.0643, 3.21343, 0.640191\}$ $\boldsymbol{\beta} = \{0.000464851, 0.00137668, 0.00497197, 0.0182675, 0.0670763, 0.240282, 0.667561\}$	94.70
650	$\boldsymbol{\omega} = \{125523., 21581., 2449.49, 264.284, 28.5658, 3.30677, 0.642816\}$ $\boldsymbol{\beta} = \{0.000423052, 0.00127638, 0.00468803, 0.0175049, 0.0653251, 0.238015, 0.672767\}$	99.84
700	$\boldsymbol{\omega} = \{144700., 24261.7, 2690.48, 283.996, 30.0298, 3.39572, 0.645256\}$ $\boldsymbol{\beta} = \{0.000387569, 0.00118966, 0.00443832, 0.0168231, 0.0637313, 0.235892, 0.677538\}$	104.8
750	$\boldsymbol{\omega} = \{165170., 27053.7, 2935.97, 303.66, 31.4598, 3.48076, 0.647538\}$ $\boldsymbol{\beta} = \{0.000357102, 0.00111394, 0.00421676, 0.0162087, 0.0622713, 0.233896, 0.681936\}$	109.6
800	$\boldsymbol{\omega} = \{186923., 29953.6, 3185.75, 323.277, 32.8588, 3.56231, 0.649679\}$ $\boldsymbol{\beta} = \{0.000330683, 0.00104722, 0.00401865, 0.0156514, 0.0609266, 0.232012, 0.686013\}$	114.3
850	$\boldsymbol{\omega} = \{209951., 32958.3, 3439.6, 342.851, 34.2293, 3.64072, 0.651697\}$ $\boldsymbol{\beta} = \{0.000307575, 0.000987994, 0.00384032, 0.015143, 0.0596819, 0.23023, 0.689809\}$	118.9
900	$\boldsymbol{\omega} = \{234245., 36064.8, 3697.33, 362.383, 35.5733, 3.71629, 0.653606\}$ $\boldsymbol{\beta} = \{0.000287208, 0.000935052, 0.00367881, 0.0146766, 0.0585247, 0.228538, 0.693359\}$	123.3
950	$\boldsymbol{\omega} = \{259798., 39270.7, 3958.79, 381.876, 36.893, 3.78925, 0.655415\}$ $\boldsymbol{\beta} = \{0.000269134, 0.000887441, 0.00353176, 0.0142469, 0.0574448, 0.226929, 0.696691\}$	127.6
1000	$\boldsymbol{\omega} = \{286604., 42573.3, 4223.81, 401.331, 38.1899, 3.85983, 0.657136\}$ $\boldsymbol{\beta} = \{0.000252996, 0.000844392, 0.00339723, 0.0138493, 0.0564334, 0.225395, 0.699828\}$	131.9

Table B.6 Parameters for optimized $P = 7$ SRJ schemes for various values of N ($N = 2^k$ $k = 5, \ldots, 15$).

N	Optimal Scheme Parameters	ρ
32	$\boldsymbol{\omega} = \{370.035, 167.331, 51.1952, 13.9321, 3.80777, 1.18727, 0.556551\}$ $\boldsymbol{\beta} = \{0.0107542, 0.0171537, 0.0336988, 0.0699421, 0.144888, 0.282064, 0.441499\}$	10.68
64	$\boldsymbol{\omega} = \{1426.38, 523.554, 126.345, 27.5077, 6.01245, 1.48211, 0.57366\}$ $\boldsymbol{\beta} = \{0.005425, 0.00988035, 0.0224061, 0.0531545, 0.125891, 0.28267, 0.500573\}$	18.67
128	$\boldsymbol{\omega} = \{5473.26, 1613.5, 308.842, 54.2321, 9.55437, 1.86957, 0.592601\}$ $\boldsymbol{\beta} = \{0.00263361, 0.00553096, 0.0144785, 0.0392473, 0.106251, 0.275379, 0.55648\}$	31.80
256	$\boldsymbol{\omega} = \{20897.4, 4910.51, 749.664, 106.726, 15.2337, 2.37639, 0.613148\}$ $\boldsymbol{\beta} = \{0.00123422, 0.00301272, 0.0091062, 0.0282465, 0.0875342, 0.262111, 0.608755\}$	52.76
512	$\boldsymbol{\omega} = \{79377.3, 14792.9, 1810.11, 209.596, 24.3177, 3.03654, 0.635029\}$ $\boldsymbol{\beta} = \{0.00055958, 0.00159767, 0.00558101, 0.0198616, 0.0706365, 0.244683, 0.65708\}$	85.20
1024	$\boldsymbol{\omega} = \{299914.156, 44192.3571, 4352.25238, 410.656961, 38.8047154, 3.89292506, 0.65793356\}$ $\boldsymbol{\beta} = \{0.000245852, 0.000825159, 0.00333663, 0.0136687, 0.0559701, 0.224683, 0.701271\}$	133.8
2048	$\boldsymbol{\omega} = \{1127025.97, 131100.588, 10426.4645, 802.498148, 61.8325621, 4.99924687, 0.68152912\}$ $\boldsymbol{\beta} = \{0.000104810, 0.000415215, 0.00194787, 0.00921977, 0.0436271, 0.203420, 0.741265\}$	204.5
4096	$\boldsymbol{\omega} = \{4211571.43, 386588.015, 24893.3205, 1563.73805, 98.3060734, 6.42233602, 0.70547174\}$ $\boldsymbol{\beta} = \{0.0000434063, 0.000203656, 0.00111148, 0.00610263, 0.0335008, 0.181922, 0.777116\}$	303.8
8192	$\boldsymbol{\omega} = \{15648209.9, 1133886.05, 59234.7606, 3037.61946, 155.857928, 8.24483034, 0.72942127\}$ $\boldsymbol{\beta} = \{0.0000174827, 0.0000974264, 0.000620536, 0.00396819, 0.0253729, 0.160955, 0.808968\}$	438.7
16384	$\boldsymbol{\omega} = \{57801287.0, 3309510.67, 140475.554, 5881.12587, 246.314088, 10.5684196, 0.75305435\}$ $\boldsymbol{\beta} = \{0.00000685565, 0.0000454934, 0.000339312, 0.00253741, 0.0189738, 0.141061, 0.837036\}$	616.2
32768	$\boldsymbol{\omega} = \{212234180, 9615316.86, 331986.888, 11346.7836, 387.921369, 13.5177566, 0.77607577\}$ $\boldsymbol{\beta} = \{0.00000262045, 0.0000207546, 0.000181910, 0.00159715, 0.0140222, 0.122590, 0.861585\}$	842.0

Table B.7 Parameters for optimized $P = 8$ SRJ schemes for various values of N ($N = 100 + 50k$, $k = 0, \ldots, 18$).

N	Optimal Scheme Parameters	ρ
100	$\boldsymbol{\omega} = \{3537.92, 1419.79, 377.85, 89.614, 20.816, 4.96187, 1.3481, 0.566216\}$ $\boldsymbol{\beta} = \{0.00319391, 0.00549334, 0.0117303, 0.0263687, 0.0597575, 0.134042, 0.283454, 0.47596\}$	28.46
150	$\boldsymbol{\omega} = \{7806.72, 2810.62, 663.524, 140.897, 29.3973, 6.27138, 1.51276, 0.575281\}$ $\boldsymbol{\beta} = \{0.00209747, 0.00386747, 0.00888556, 0.0213663, 0.0517106, 0.124107, 0.282297, 0.505669\}$	39.17
200	$\boldsymbol{\omega} = \{13677.4, 4551.55, 987.73, 194.11, 37.56, 7.41325, 1.64438, 0.582004\}$ $\boldsymbol{\beta} = \{0.00154611, 0.00299977, 0.00725956, 0.018313, 0.0464484, 0.116968, 0.280231, 0.526234\}$	48.90
250	$\boldsymbol{\omega} = \{21120.6, 6606.74, 1343.7, 248.789, 45.4233, 8.44441, 1.75577, 0.587377\}$ $\boldsymbol{\beta} = \{0.00121592, 0.00245618, 0.00618861, 0.0162041, 0.0426301, 0.111443, 0.277985, 0.541877\}$	57.94
300	$\boldsymbol{\omega} = \{30113.4, 8950.92, 1727.05, 304.653, 53.0547, 9.39499, 1.85328, 0.591865\}$ $\boldsymbol{\beta} = \{0.000996887, 0.00208219, 0.00542207, 0.0146373, 0.0396817, 0.106964, 0.275766, 0.554449\}$	66.43
350	$\boldsymbol{\omega} = \{40636.8, 11564.9, 2134.67, 361.517, 60.4978, 10.2833, 1.94055, 0.595727\}$ $\boldsymbol{\beta} = \{0.000841422, 0.00180845, 0.00484238, 0.0134154, 0.0373083, 0.103214, 0.273641, 0.564929\}$	74.48
400	$\boldsymbol{\omega} = \{52674.7, 14433.5, 2564.23, 419.246, 67.7827, 11.1214, 2.01992, 0.599121\}$ $\boldsymbol{\beta} = \{0.000725636, 0.00159908, 0.00438648, 0.0124289, 0.0353398, 0.1, 0.271626, 0.573894\}$	82.16
450	$\boldsymbol{\omega} = \{66212.9, 17544.1, 3013.86, 477.737, 74.9318, 11.918, 2.09294, 0.60215\}$ $\boldsymbol{\beta} = \{0.000636233, 0.00143358, 0.00401723, 0.0116114, 0.03367, 0.0971967, 0.269722, 0.581713\}$	89.53
500	$\boldsymbol{\omega} = \{81238.9, 20886.1, 3482.07, 536.911, 81.9618, 12.6794, 2.16076, 0.60489\}$ $\boldsymbol{\beta} = \{0.000565234, 0.00129935, 0.00371125, 0.0109201, 0.03223, 0.0947158, 0.267924, 0.588635\}$	96.62
550	$\boldsymbol{\omega} = \{97741.4, 24450.5, 3967.62, 596.704, 88.8863, 13.4104, 2.22421, 0.607385\}$ $\boldsymbol{\beta} = \{0.000507567, 0.00118823, 0.003453, 0.010326, 0.0309663, 0.0924954, 0.266225, 0.594839\}$	103.5
600	$\boldsymbol{\omega} = \{115710., 28229.4, 4469.46, 657.063, 95.716, 14.1148, 2.28393, 0.609684\}$ $\boldsymbol{\beta} = \{0.000459858, 0.00109468, 0.00323173, 0.00980844, 0.0298481, 0.0904893, 0.264614, 0.600453\}$	110.1
650	$\boldsymbol{\omega} = \{135136., 32215.7, 4986.68, 717.944, 102.46, 14.7956, 2.34042, 0.611813\}$ $\boldsymbol{\beta} = \{0.000419774, 0.00101481, 0.00303977, 0.0093526, 0.0288479, 0.0886624, 0.263086, 0.605577\}$	116.5
700	$\boldsymbol{\omega} = \{156011., 36403.2, 5518.5, 779.308, 109.125, 15.4554, 2.39408, 0.613796\}$ $\boldsymbol{\beta} = \{0.000385654, 0.000945799, 0.00287144, 0.00894728, 0.0279458, 0.0869875, 0.261633, 0.610284\}$	122.8
750	$\boldsymbol{\omega} = \{178325., 40786.3, 6064.21, 841.124, 115.718, 16.0962, 2.44524, 0.615653\}$ $\boldsymbol{\beta} = \{0.000356284, 0.000885566, 0.00272248, 0.00858393, 0.0271264, 0.0854428, 0.260247, 0.614635\}$	128.9
800	$\boldsymbol{\omega} = \{202073., 45359.8, 6623.2, 903.364, 122.244, 16.7198, 2.49417, 0.617399\}$ $\boldsymbol{\beta} = \{0.000330755, 0.000832525, 0.00258962, 0.00825589, 0.0263774, 0.0840111, 0.258925, 0.618678\}$	134.9
850	$\boldsymbol{\omega} = \{227246., 50119.2, 7194.93, 966.001, 128.708, 17.3276, 2.5411, 0.619047\}$ $\boldsymbol{\beta} = \{0.000308374, 0.000785453, 0.00247029, 0.00795791, 0.0256892, 0.082678, 0.257659, 0.622452\}$	140.7
900	$\boldsymbol{\omega} = \{253839., 55060., 7778.89, 1029.01, 135.114, 17.921, 2.58622, 0.620607\}$ $\boldsymbol{\beta} = \{0.000288606, 0.00074339, 0.00236245, 0.00768573, 0.0250537, 0.0814319, 0.256446, 0.625988\}$	146.4
950	$\boldsymbol{\omega} = \{281844., 60178.5, 8374.63, 1092.39, 141.465, 18.501, 2.6297, 0.622089\}$ $\boldsymbol{\beta} = \{0.000271028, 0.000705575, 0.00226446, 0.00743591, 0.0244645, 0.0802629, 0.255282, 0.629313\}$	151.9
1000	$\boldsymbol{\omega} = \{311256., 65470.9, 8981.74, 1156.09, 147.766, 19.0687, 2.67166, 0.6235\}$ $\boldsymbol{\beta} = \{0.000255303, 0.000671391, 0.00217498, 0.0072056, 0.0239159, 0.0791628, 0.254163, 0.632451\}$	157.4

Table B.8 Parameters for optimized $P = 8$ SRJ schemes for various values of N ($N = 2^k$ $k = 5, \ldots, 15$).

N	Optimal Scheme Parameters	ρ
32	$\boldsymbol{\omega} = \{380.243, 203.177, 76.1463, 24.9346, 7.91926, 2.60779, 0.993193, 0.543507\}$ $\boldsymbol{\beta} = \{0.00977378, 0.0140492, 0.0244443, 0.0454149, 0.0855457, 0.15853, 0.273683, 0.388559\}$	11.14
64	$\boldsymbol{\omega} = \{1478.41, 666.114, 202.545, 54.3776, 14.2439, 3.8448, 1.19182, 0.55682\}$ $\boldsymbol{\beta} = \{0.0050073, 0.00800348, 0.0157692, 0.0329114, 0.0693924, 0.144485, 0.282072, 0.44236\}$	19.84
128	$\boldsymbol{\omega} = \{5729.04, 2152.82, 532.541, 118.056, 25.6833, 5.72098, 1.44559, 0.571677\}$ $\boldsymbol{\beta} = \{0.00247575, 0.004441, 0.00991522, 0.0232218, 0.0547725, 0.128023, 0.283006, 0.494145\}$	34.60
256	$\boldsymbol{\omega} = \{22118.3, 6873.21, 1388.33, 255.434, 46.3503, 8.5623, 1.76811, 0.587956\}$ $\boldsymbol{\beta} = \{0.00118505, 0.00240419, 0.00608363, 0.0159926, 0.0422379, 0.110858, 0.277715, 0.543523\}$	58.98
512	$\boldsymbol{\omega} = \{83648.1, 21109.1, 3504.48, 539.515, 82.2596, 12.7104, 2.1634, 0.604989\}$ $\boldsymbol{\beta} = \{0.000565345, 0.00129282, 0.00369896, 0.0108933, 0.0321732, 0.094622, 0.267859, 0.588895\}$	98.21
1024	$\boldsymbol{\omega} = \{325872.429, 68072.0251, 9277.08426, 1186.79068, 150.772525, 19.3370248, 2.69130897, 0.62415405\}$ $\boldsymbol{\beta} = \{0.000248331, 0.000656122, 0.00213472, 0.00710128, 0.0236657, 0.0786571, 0.253641, 0.633896\}$	160.0
2048	$\boldsymbol{\omega} = \{1243390.54, 211839.541, 23840.6659, 2549.96741, 271.441418, 29.0896995, 3.33872799, 0.64369876\}$ $\boldsymbol{\beta} = \{0.000109012, 0.000330956, 0.00122248, 0.00459014, 0.0172536, 0.0647443, 0.237249, 0.674501\}$	254.3
4096	$\boldsymbol{\omega} = \{4724941.11, 655366.979, 61084.1980, 5467.76799, 487.846845, 43.7324841, 4.15010426, 0.66392578\}$ $\boldsymbol{\beta} = \{0.0000465981, 0.000163126, 0.000685047, 0.00291125, 0.0123798, 0.0525884, 0.219311, 0.711915\}$	394.6
8192	$\boldsymbol{\omega} = \{17880305.2, 2017253.16, 156092.848, 11699.7068, 875.030230, 65.6587728, 5.16366699, 0.68460941\}$ $\boldsymbol{\beta} = \{0.0000194120, 0.0000785878, 0.000375889, 0.00181326, 0.00875016, 0.0421983, 0.200638, 0.746127\}$	438.7
16384	$\boldsymbol{\omega} = \{67375460.0, 6181549.54, 397871.724, 24979.4457, 1566.01561, 98.3988686, 6.42559672, 0.70552038\}$ $\boldsymbol{\beta} = \{0.00000788702, 0.0000370175, 0.000202091, 0.00110994, 0.00609737, 0.0334824, 0.181878, 0.777185\}$	882.9
32768	$\boldsymbol{\omega} = \{252775864, 18866153.6, 1011634.78, 53208.1901, 2795.89696, 147.142217, 7.99143284, 0.72643283\}$ $\boldsymbol{\beta} = \{.000000312768, 0.0000170557, 0.000106532, 0.000668220, 0.00419188, 0.0262904, 0.163531, 0.805192\}$	1273

Table B.9 Comparison of the parameters obtained by YM14 with their quasi-optimal (QO) method and our optimal (O) method. We present the schemes in the same order as in YM14. In the last column we also show the convergence performance index ($\rho := \sum_{i=1}^{P} \omega_i \beta_i$).

P	N	Type	Optimal / quasi-optimal β-prescribed scheme parameters	ρ
8	512	O	$\boldsymbol{\omega} = \{83648.1, 21109.1, 3504.48, 539.515, 82.2596, 12.7104, 2.1634, 0.604989\}$ $\boldsymbol{\beta} = \{0.000565345, 0.00129282, 0.00369896, 0.0108933, 0.0321732, 0.094622, 0.267859, 0.588895\}$	98.21
		QO	$\boldsymbol{\omega} = \{91299, 25979, 3862.1, 549.90, 80.217, 11.992, 1.9595, 0.59145\}$ $\boldsymbol{\beta} = \{0.000411523, 0.00123457, 0.0037037, 0.0111111, 0.0333333, 0.1, 0.3, 0.550206\}$	94.84
6	512	O	$\boldsymbol{\omega} = \{71233.5, 8633.18, 712.561, 57.0344, 4.78697, 0.677408\}$ $\boldsymbol{\beta} = \{0.000562137, 0.00216338, 0.00988896, 0.0456021, 0.207144, 0.734639\}$	69.86
		QO	$\boldsymbol{\omega} = \{83242, 14099, 1334.1, 126.45, 12.193, 0.79246\}$ $\boldsymbol{\beta} = \{0.000351494, 0.00140598, 0.0056239, 0.0224956, 0.0899824, 0.880141\}$	61.22
5	1024	O	$\boldsymbol{\omega} = \{210607., 9955.65, 357.116, 12.9282, 0.771947\}$ $\boldsymbol{\beta} = \{0.000227265, 0.00174271, 0.0148278, 0.125833, 0.85737\}$	72.80
		QO	$\boldsymbol{\omega} = \{178919, 8024.1, 349.03, 15.9047, 0.799909\}$ $\boldsymbol{\beta} = \{0.000286779, 0.00200746, 0.0140522, 0.0983654, 0.885288\}$	74.60
7	1024	O	$\boldsymbol{\omega} = \{299914.156, 44192.3571, 4352.25238, 410.656961, 38.8047154, 3.89292506, 0.65793356\}$ $\boldsymbol{\beta} = \{0.000245852, 0.000825159, 0.00333663, 0.0136687, 0.0559701, 0.224683, 0.701271\}$	133.8
		QO	$\boldsymbol{\omega} = \{300015, 47617, 4738.4, 428.51, 39.410, 3.9103, 0.65823\}$ $\boldsymbol{\beta} = \{0.000246063, 0.000738189, 0.00319882, 0.0135335, 0.0558563, 0.224656, 0.701772\}$	133.5

Table B.10 Parameters for optimized $P = 9$ SRJ schemes for various values of N ($N = 100 + 50k$, $k = 0, \ldots, 18$).

N	Optimal Scheme Parameters	ρ
100	$\boldsymbol{\omega} = \{3640.86, 1729.24, 559.604, 158.917, 43.5786, 11.9645, 3.41756, 1.12661, 0.552629\}$ $\boldsymbol{\beta} = \{0.00298948, 0.00462251, 0.0087721, 0.0176949, 0.0362139, 0.0740779, 0.149046, 0.280346, 0.426238\}$	30.12
150	$\boldsymbol{\omega} = \{8067.04, 3497.09, 1016.04, 260.97, 65.0344, 16.2237, 4.19023, 1.24183, 0.559916\}$ $\boldsymbol{\beta} = \{0.00198105, 0.00324426, 0.00656828, 0.0140648, 0.0304787, 0.0660202, 0.141034, 0.282856, 0.453753\}$	41.92
200	$\boldsymbol{\omega} = \{14177.3, 5750.32, 1548.34, 370.696, 86.3811, 20.1478, 4.84929, 1.33292, 0.565337\}$ $\boldsymbol{\beta} = \{0.0014704, 0.00251182, 0.00532492, 0.0118951, 0.0268474, 0.0605723, 0.135008, 0.283441, 0.472929\}$	52.78
250	$\boldsymbol{\omega} = \{21947.5, 8446.56, 2144.68, 486.462, 107.643, 23.8396, 5.43484, 1.4094, 0.56968\}$ $\boldsymbol{\beta} = \{0.0011629, 0.00205427, 0.00451336, 0.0104186, 0.0242715, 0.0565249, 0.130206, 0.283258, 0.48759\}$	62.97
300	$\boldsymbol{\omega} = \{31358.5, 11555.4, 2797.26, 607.253, 128.836, 27.3558, 5.9677, 1.4759, 0.573317\}$ $\boldsymbol{\beta} = \{0.000957974, 0.0017401, 0.00393644, 0.009334, 0.0223169, 0.0533419, 0.126229, 0.282718, 0.499425\}$	72.62
350	$\boldsymbol{\omega} = \{42394.6, 15053.5, 3500.43, 732.375, 149.969, 30.7322, 6.46034, 1.53511, 0.576452\}$ $\boldsymbol{\beta} = \{0.000811947, 0.00151052, 0.00350254, 0.00849594, 0.0207656, 0.050741, 0.122843, 0.282002, 0.509327\}$	81.84
400	$\boldsymbol{\omega} = \{55042.3, 18922., 4249.87, 861.316, 171.05, 33.993, 6.92094, 1.5887, 0.579211\}$ $\boldsymbol{\beta} = \{0.000702809, 0.00133513, 0.00316284, 0.00782458, 0.0194944, 0.0485563, 0.119902, 0.281197, 0.517825\}$	90.69
450	$\boldsymbol{\omega} = \{69289.9, 23145.3, 5042.15, 993.684, 192.085, 37.1561, 7.35522, 1.63782, 0.581677\}$ $\boldsymbol{\beta} = \{0.000618273, 0.00119662, 0.00288877, 0.00727198, 0.0184271, 0.0466826, 0.117306, 0.28035, 0.525258\}$	99.23
500	$\boldsymbol{\omega} = \{85126.9, 27710.2, 5874.43, 1129.16, 213.078, 40.2346, 7.76736, 1.68326, 0.58391\}$ $\boldsymbol{\beta} = \{0.000550948, 0.00108438, 0.00266241, 0.00680743, 0.0175139, 0.0450492, 0.114986, 0.279488, 0.531857\}$	107.49
550	$\boldsymbol{\omega} = \{102544., 32605.2, 6744.37, 1267.5, 234.034, 43.2388, 8.16055, 1.72565, 0.58595\}$ $\boldsymbol{\beta} = \{0.000496122, 0.000991529, 0.00247193, 0.00641024, 0.0167209, 0.0436064, 0.112892, 0.278627, 0.537784\}$	115.52
600	$\boldsymbol{\omega} = \{121532., 37820.1, 7649.95, 1408.49, 254.954, 46.1771, 8.53726, 1.76542, 0.587829\}$ $\boldsymbol{\beta} = \{0.000450652, 0.000913401, 0.00230916, 0.0060659, 0.0160234, 0.0423181, 0.110985, 0.277774, 0.54316\}$	123.32
650	$\boldsymbol{\omega} = \{142084., 43346.1, 8589.42, 1551.94, 275.842, 49.0561, 8.89946, 1.80294, 0.589571\}$ $\boldsymbol{\beta} = \{0.000412363, 0.000846729, 0.00216828, 0.0057639, 0.0154037, 0.0411574, 0.109236, 0.276936, 0.548076\}$	130.93
700	$\boldsymbol{\omega} = \{164193., 49175.1, 9561.25, 1697.7, 296.7, 51.8815, 9.24875, 1.8385, 0.591197\}$ $\boldsymbol{\beta} = \{0.000379703, 0.00078915, 0.002045, 0.00549643, 0.0148481, 0.0401035, 0.107622, 0.276115, 0.552601\}$	138.35
750	$\boldsymbol{\omega} = \{187851., 55299.9, 10564.1, 1845.64, 317.53, 54.6579, 9.58648, 1.87232, 0.59272\}$ $\boldsymbol{\beta} = \{0.000351533, 0.00073891, 0.00193613, 0.00525752, 0.0143463, 0.0391402, 0.106125, 0.275312, 0.556792\}$	145.61
800	$\boldsymbol{\omega} = \{213052., 61713.8, 11596.7, 1995.64, 338.333, 57.3892, 9.91376, 1.9046, 0.594153\}$ $\boldsymbol{\beta} = \{0.000327001, 0.000694683, 0.0018392, 0.00504256, 0.0138901, 0.0382548, 0.10473, 0.27453, 0.560692\}$	152.72
850	$\boldsymbol{\omega} = \{239791., 68410.7, 12658.1, 2147.59, 359.111, 60.0791, 10.2315, 1.9355, 0.595507\}$ $\boldsymbol{\beta} = \{0.000305456, 0.000655443, 0.00175229, 0.0048479, 0.0134729, 0.0374366, 0.103425, 0.273767, 0.564338\}$	159.69
900	$\boldsymbol{\omega} = \{268061., 75385.1, 13747.1, 2301.4, 379.865, 62.7304, 10.5406, 1.96515, 0.59679\}$ $\boldsymbol{\beta} = \{0.000286394, 0.000620388, 0.00167386, 0.00467063, 0.0130894, 0.0366774, 0.102199, 0.273024, 0.56776\}$	166.53
950	$\boldsymbol{\omega} = \{297859., 82631.8, 14863., 2456.98, 400.597, 65.3458, 10.8416, 1.99367, 0.59801\}$ $\boldsymbol{\beta} = \{0.000269416, 0.000588879, 0.0016027, 0.00450836, 0.0127354, 0.0359699, 0.101043, 0.272299, 0.570983\}$	173.24
1000	$\boldsymbol{\omega} = \{329177., 90146., 16004.9, 2614.27, 421.306, 67.9276, 11.1353, 2.02115, 0.599172\}$ $\boldsymbol{\beta} = \{0.000254204, 0.000560401, 0.00153782, 0.00435916, 0.0124072, 0.0353083, 0.0999517, 0.271594, 0.574027\}$	179.84

Table B.11 Parameters for optimized $P = 9$ SRJ schemes for various values of N ($N = 2^k$ $k = 5, \ldots, 13$).

N	Optimal Scheme Parameters	ρ
32	$\boldsymbol{\alpha} = \{387.382, 233.468, 102.424, 38.9705, 14.1267, 5.1286, 1.96283, 0.872543, 0.534479\}$ $\boldsymbol{\beta} = \{0.00895557, 0.0119599, 0.0189643, 0.0322828, 0.0562016, 0.0977771, 0.165994, 0.261519, 0.346345\}$	11.50
64	$\boldsymbol{\alpha} = \{1515.04, 792.702, 289.003, 91.8543, 28.038, 8.56994, 2.7402, 1.0158, 0.545103\}$ $\boldsymbol{\beta} = \{0.00464454, 0.00676243, 0.0119546, 0.0225735, 0.0433789, 0.0833122, 0.156892, 0.275129, 0.395353\}$	20.78
128	$\boldsymbol{\alpha} = \{5910.34, 2656.28, 804.934, 214.987, 55.6093, 14.4, 3.8679, 1.195, 0.557016\}$ $\boldsymbol{\beta} = \{0.00232987, 0.00372963, 0.007362, 0.0154007, 0.0326349, 0.0691242, 0.144247, 0.282116, 0.443056\}$	36.87
256	$\boldsymbol{\alpha} = \{22990.5, 8798.35, 2220.15, 500.707, 110.19, 24.27, 5.50127, 1.41784, 0.570149\}$ $\boldsymbol{\beta} = \{0.00113407, 0.00201056, 0.00443415, 0.0102715, 0.0240097, 0.0561044, 0.129691, 0.283207, 0.489137\}$	64.15
512	$\boldsymbol{\alpha} = \{89163.2, 28855.3, 6079.87, 1162.11, 218.111, 40.962, 7.86335, 1.6937, 0.584415\}$ $\boldsymbol{\beta} = \{0.000536789, 0.00106054, 0.00261378, 0.00670659, 0.0173137, 0.044687, 0.114465, 0.279281, 0.533335\}$	109.44
1024	$\boldsymbol{\alpha} = \{344749., 93846.5, 16562.1, 2690.35, 431.24, 69.1555, 11.2738, 2.034, 0.599711\}$ $\boldsymbol{\beta} = \{0.000247452, 0.000547682, 0.00150865, 0.00429171, 0.0122579, 0.0350055, 0.0994484, 0.271262, 0.575431\}$	182.96
2048	$\boldsymbol{\alpha} = \{1328880, 303078., 44937.7, 6215.79, 851.582, 116.707, 16.1894, 2.45258, 0.615917\}$ $\boldsymbol{\beta} = \{0.000111223, 0.000276977, 0.000853022, 0.00269416, 0.00853108, 0.0270113, 0.0852253, 0.260049, 0.615248\}$	299.64
4096	$\boldsymbol{\alpha} = \{5106340, 972991., 121553., 14336.3, 1679.42, 196.78, 23.2619, 2.96607, 0.632898\}$ $\boldsymbol{\beta} = \{0.0000487855, 0.000137195, 0.000472679, 0.00166043, 0.00584223, 0.0205547, 0.0721511, 0.246475, 0.652658\}$	480.54
8192	$\boldsymbol{\alpha} = \{19559500, 3107886, 327956., 33012.6, 3307.31, 331.387, 33.4177, 3.59437, 0.650509\}$ $\boldsymbol{\beta} = \{0.0000208973, 0.0000665666, 0.000256798, 0.00100536, 0.00393987, 0.0154394, 0.0604121, 0.231281, 0.687578\}$	754.47

Table B.12 Parameters for optimized $P = 10$ SRJ schemes for various values of N ($N = 100 + 50k$, $k = 0, \ldots, 18$).

N	Optimal Scheme Parameters	ρ
100	$\boldsymbol{\omega} = \{3716.11, 2003.92, 758.907, 249.869, 78.5973, 24.4799, 7.71415, 2.55553, 0.983449, 0.542795\}$ $\boldsymbol{\beta} = \{0.0028058, 0.00400978, 0.00692995, 0.0128148, 0.0241718, 0.0457763, 0.0862794, 0.159105, 0.272849, 0.385258\}$	31.49
150	$\boldsymbol{\omega} = \{8257.94, 4120.6, 1415.79, 425.028, 122.54, 35.0395, 10.1187, 3.05606, 1.06871, 0.548763\}$ $\boldsymbol{\beta} = \{0.00187294, 0.00280821, 0.00514163, 0.0100348, 0.019919, 0.0396644, 0.0786806, 0.153149, 0.278034, 0.410696\}$	44.20
200	$\boldsymbol{\omega} = \{14544.8, 6857.12, 2199.03, 618.829, 167.848, 45.1996, 12.278, 3.47555, 1.13556, 0.553212\}$ $\boldsymbol{\beta} = \{0.00139791, 0.00217152, 0.00414295, 0.00839998, 0.0172885, 0.0356771, 0.0733849, 0.148408, 0.280624, 0.428506\}$	56.02
250	$\boldsymbol{\omega} = \{22556.4, 10166.9, 3091.02, 827.681, 214.189, 55.0715, 14.2712, 3.84341, 1.19131, 0.556783\}$ $\boldsymbol{\beta} = \{0.00111059, 0.00177456, 0.00349571, 0.00730005, 0.0154531, 0.0327874, 0.0693676, 0.1445, 0.282038, 0.442173\}$	67.18
300	$\boldsymbol{\omega} = \{32276.8, 14016.1, 4079.92, 1049.31, 261.365, 64.719, 16.1413, 4.17472, 1.23956, 0.559776\}$ $\boldsymbol{\beta} = \{0.000918413, 0.00150238, 0.00303809, 0.00649915, 0.014078, 0.0305574, 0.0661576, 0.141186, 0.282825, 0.453239\}$	77.83
350	$\boldsymbol{\omega} = \{43692.7, 18378.4, 5157.02, 1282.12, 309.244, 74.1832, 17.9147, 4.47839, 1.28233, 0.56236\}$ $\boldsymbol{\beta} = \{0.000781034, 0.0013037, 0.00269539, 0.00588469, 0.012998, 0.0287629, 0.0635007, 0.138314, 0.283239, 0.462521\}$	88.06
400	$\boldsymbol{\omega} = \{56792.9, 23232.2, 6315.6, 1524.93, 357.729, 83.4929, 19.6092, 4.76023, 1.3209, 0.564637\}$ $\boldsymbol{\beta} = \{0.000678071, 0.00115204, 0.00242805, 0.00539537, 0.0121206, 0.0272748, 0.0612448, 0.135783, 0.283418, 0.470505\}$	97.94
450	$\boldsymbol{\omega} = \{71567.4, 28559.9, 7550.26, 1776.81, 406.749, 92.6693, 21.2376, 5.02426, 1.35614, 0.566674\}$ $\boldsymbol{\beta} = \{0.000598118, 0.00103237, 0.00221302, 0.00499467, 0.0113895, 0.0260125, 0.0592921, 0.133524, 0.283441, 0.477503\}$	107.50
500	$\boldsymbol{\omega} = \{88007.4, 34346., 8856.52, 2037.02, 456.248, 101.729, 22.8094, 5.27342, 1.38866, 0.568519\}$ $\boldsymbol{\beta} = \{0.000534297, 0.000935438, 0.00203589, 0.00465934, 0.0107682, 0.0249227, 0.0575758, 0.131484, 0.283358, 0.483727\}$	116.80
550	$\boldsymbol{\omega} = \{106105., 40577.2, 10230.6, 2304.96, 506.181, 110.684, 24.3319, 5.5099, 1.4189, 0.570207\}$ $\boldsymbol{\beta} = \{0.000482215, 0.000855288, 0.00188718, 0.00437377, 0.0102318, 0.0239683, 0.0560489, 0.129626, 0.2832, 0.489327\}$	125.85
600	$\boldsymbol{\omega} = \{125853., 47241.7, 11669.3, 2580.09, 556.511, 119.545, 25.8109, 5.73541, 1.44721, 0.571763\}$ $\boldsymbol{\beta} = \{0.000438937, 0.000787878, 0.00176037, 0.00412708, 0.00976257, 0.0231228, 0.0546765, 0.127921, 0.282988, 0.494414\}$	134.69
650	$\boldsymbol{\omega} = \{147244., 54329.2, 13169.7, 2861.99, 607.205, 128.321, 27.2511, 5.95132, 1.47387, 0.573207\}$ $\boldsymbol{\beta} = \{0.000402428, 0.000730372, 0.00165081, 0.00391144, 0.00934766, 0.0223663, 0.0534326, 0.126347, 0.282739, 0.499072\}$	143.33
700	$\boldsymbol{\omega} = \{170274., 61830., 14729.5, 3150.28, 658.239, 137.019, 28.6564, 6.15872, 1.49908, 0.574555\}$ $\boldsymbol{\beta} = \{0.000371232, 0.000680723, 0.00155511, 0.00372101, 0.00897738, 0.0216838, 0.052297, 0.124886, 0.282461, 0.503366\}$	151.80
750	$\boldsymbol{\omega} = \{194935., 69735.6, 16346.3, 3444.62, 709.588, 145.645, 30.0301, 6.35853, 1.52302, 0.575818\}$ $\boldsymbol{\beta} = \{0.000344283, 0.000637414, 0.00147072, 0.00355139, 0.00864431, 0.0210638, 0.0512538, 0.123523, 0.282164, 0.507348\}$	160.09
800	$\boldsymbol{\omega} = \{221222., 78038.2, 18018.1, 3744.71, 761.234, 154.205, 31.3748, 6.55149, 1.54582, 0.577009\}$ $\boldsymbol{\beta} = \{0.000320779, 0.000599295, 0.00139569, 0.00339916, 0.00834266, 0.0204969, 0.0502903, 0.122246, 0.281852, 0.511058\}$	168.23
850	$\boldsymbol{\omega} = \{249131., 86730.6, 19743.3, 4050.3, 813.158, 162.702, 32.693, 6.73825, 1.56762, 0.578133\}$ $\boldsymbol{\beta} = \{0.000300108, 0.000565482, 0.0013285, 0.00326163, 0.0080678, 0.0199759, 0.0493961, 0.121045, 0.281529, 0.51453\}$	176.23
900	$\boldsymbol{\omega} = \{278657., 95806., 21520.1, 4361.15, 865.346, 171.141, 33.9867, 6.91935, 1.5885, 0.5792\}$ $\boldsymbol{\beta} = \{0.000281793, 0.00053528, 0.00126795, 0.00313666, 0.00781602, 0.0194947, 0.0485629, 0.119913, 0.2812, 0.517792\}$	184.10
950	$\boldsymbol{\omega} = \{309795., 105258., 23347., 4677.05, 917.783, 179.526, 35.2576, 7.09526, 1.60855, 0.580214\}$ $\boldsymbol{\beta} = \{0.000265459, 0.000508136, 0.00121307, 0.00302252, 0.00758428, 0.0190484, 0.0477835, 0.118841, 0.280866, 0.520867\}$	191.84
1000	$\boldsymbol{\omega} = \{342541., 115082., 25222.8, 4997.8, 970.458, 187.859, 36.5072, 7.2664, 1.62786, 0.581182\}$ $\boldsymbol{\beta} = \{0.000250807, 0.000483607, 0.00116308, 0.00291777, 0.00737009, 0.0186328, 0.0470519, 0.117825, 0.280529, 0.523776\}$	199.47

Table B.13 Parameters for optimized $P = 10$ SRJ schemes for various values of N ($N = 2^k\, k = 5, \ldots, 13$).

N	Optimal Scheme Parameters	ρ
32	$\boldsymbol{\omega} = \{392.56, 258.705, 128.513, 55.3119, 22.4376, 8.96645, 3.64502, 1.57769, 0.792114, 0.527982\}$ $\boldsymbol{\beta} = \{0.0082622, 0.0104568, 0.0154228, 0.0244627, 0.0399088, 0.0655551, 0.106845, 0.169025, 0.248, 0.312061\}$	11.78
64	$\boldsymbol{\omega} = \{1541.73, 902.275, 380.269, 138.845, 48.1557, 16.5052, 5.73583, 2.10711, 0.90061, 0.536644\}$ $\boldsymbol{\beta} = \{0.00432745, 0.00588253, 0.00954679, 0.0166301, 0.0296487, 0.0531258, 0.094633, 0.164386, 0.264966, 0.356854\}$	21.54
128	$\boldsymbol{\omega} = \{6043.13, 3109.46, 1109.83, 345.372, 103.01, 30.452, 9.09794, 2.84852, 1.03416, 0.546386\}$ $\boldsymbol{\beta} = \{0.00219628, 0.00323088, 0.00578319, 0.0110516, 0.021504, 0.0419877, 0.081641, 0.155591, 0.276224, 0.40079\}$	38.74
256	$\boldsymbol{\omega} = \{23632.9, 10600.9, 3204.77, 853.638, 219.81, 56.2401, 14.5015, 3.88488, 1.19745, 0.557168\}$ $\boldsymbol{\beta} = \{0.00108359, 0.00173667, 0.00343275, 0.00719109, 0.015268, 0.0324906, 0.0689461, 0.144074, 0.282159, 0.443618\}$	68.49
512	$\boldsymbol{\omega} = \{92199.9, 35801.3, 9180.24, 2100.65, 468.194, 103.887, 23.179, 5.33126, 1.39611, 0.568938\}$ $\boldsymbol{\beta} = \{0.000520856, 0.000914849, 0.0019979, 0.00458673, 0.0106325, 0.0246824, 0.0571936, 0.131022, 0.283326, 0.485123\}$	118.99
1024	$\boldsymbol{\omega} = \{358829., 119927., 26140., 5153.43, 995.822, 191.842, 37.1, 7.34695, 1.63687, 0.58163\}$ $\boldsymbol{\beta} = \{0.000244298, 0.000472653, 0.00114063, 0.00287047, 0.00727288, 0.0184432, 0.0467163, 0.117355, 0.280367, 0.525117\}$	203.09
2048	$\boldsymbol{\omega} = \{1393090, 398936., 74096.7, 12614.5, 2115.46, 354.041, 59.4078, 10.1522, 1.92781, 0.595172\}$ $\boldsymbol{\beta} = \{0.000111934, 0.000239591, 0.000638994, 0.00176393, 0.00489192, 0.0135738, 0.0376372, 0.103747, 0.273958, 0.563438\}$	340.42
4096	$\boldsymbol{\omega} = \{5395040, 1319170, 209311., 30823.9, 4488.88, 652.834, 95.1119, 14.0506, 2.27851, 0.609477\}$ $\boldsymbol{\beta} = \{0.0000501434, 0.000119186, 0.000351366, 0.00106505, 0.00323843, 0.00984977, 0.0299458, 0.0906682, 0.264761, 0.599951\}$	560.18
8192	$\boldsymbol{\omega} = \{20841177, 4339863, 589668., 75210.5, 9514.64, 1202.61, 152.183, 19.4605, 2.70028, 0.624451\}$ $\boldsymbol{\beta} = \{0.0000219770, 0.0000581897, 0.000189695, 0.000632223, 0.00211144, 0.00705278, 0.0235524, 0.0784280, 0.253403, 0.634551\}$	904.73

Table B.14 Parameters for optimized $P = 11$ SRJ schemes for various values of N ($N = 100 + 50k$, $k = 0, \ldots, 18$).

N	Optimal Scheme Parameters	ρ
100	$\boldsymbol{\omega} = \{3772.69, 2243.38, 965.764, 359.834, 127.002, 44.0668, 15.2981, 5.41117, 2.02855, 0.885269, 0.535461\}$ $\boldsymbol{\beta} = \{0.00264078, 0.0035551, 0.00569791, 0.00980872, 0.0173037, 0.0307448, 0.054615, 0.0962556, 0.165235, 0.263076, 0.351068\}$	32.64
150	$\boldsymbol{\omega} = \{8401.79, 4674.24, 1842.62, 630.247, 205.246, 65.8794, 21.1553, 6.90331, 2.37588, 0.951043, 0.540431\}$ $\boldsymbol{\beta} = \{0.00177338, 0.00248623, 0.0041962, 0.00758878, 0.0140228, 0.0260629, 0.048432, 0.0894173, 0.161257, 0.27015, 0.374614\}$	46.12
200	$\boldsymbol{\omega} = \{14822.2, 7852.9, 2907.6, 936.628, 288.313, 87.62, 26.639, 8.21547, 2.66282, 1.00226, 0.544141\}$ $\boldsymbol{\beta} = \{0.00132965, 0.00192098, 0.00336441, 0.0063002, 0.0120299, 0.023085, 0.0442925, 0.0845072, 0.157812, 0.274205, 0.391154\}$	58.75
250	$\boldsymbol{\omega} = \{23016.7, 11730.6, 4137.15, 1272.65, 375.138, 109.305, 31.8602, 9.40806, 2.9118, 1.04478, 0.547122\}$ $\boldsymbol{\beta} = \{0.00106031, 0.00156902, 0.0028284, 0.00544114, 0.0106571, 0.0209649, 0.0412369, 0.0807077, 0.15484, 0.276815, 0.403879\}$	70.76
300	$\boldsymbol{\omega} = \{32971.7, 16271.6, 5515.11, 1634.24, 465.066, 130.944, 36.8811, 10.5132, 3.13414, 1.08142, 0.549623\}$ $\boldsymbol{\beta} = \{0.000879604, 0.00132793, 0.00245107, 0.00482002, 0.00963873, 0.0193512, 0.0388452, 0.0776262, 0.152245, 0.27861, 0.414205\}$	82.28
350	$\boldsymbol{\omega} = \{44676.1, 21447.6, 7029.52, 2018.49, 557.653, 152.544, 41.7411, 11.5505, 3.3365, 1.11379, 0.551784\}$ $\boldsymbol{\beta} = \{0.000750095, 0.00115207, 0.00216949, 0.00434619, 0.00884531, 0.018067, 0.0368981, 0.0750455, 0.149947, 0.279896, 0.422883\}$	93.38
400	$\boldsymbol{\omega} = \{58120.2, 27235.7, 8671., 2423.21, 652.571, 174.109, 46.4672, 12.533, 3.52319, 1.1429, 0.553689\}$ $\boldsymbol{\beta} = \{0.000652808, 0.00101792, 0.00195046, 0.00397066, 0.00820512, 0.017012, 0.0352674, 0.0728327, 0.147888, 0.280843, 0.430359\}$	104.15
450	$\boldsymbol{\omega} = \{73295.5, 33616.3, 10432., 2846.69, 749.573, 195.645, 51.0794, 13.47, 3.69717, 1.16942, 0.555395\}$ $\boldsymbol{\beta} = \{0.000577107, 0.000912097, 0.00177473, 0.00366439, 0.00767482, 0.0161243, 0.0338724, 0.0709012, 0.146025, 0.281552, 0.436922\}$	114.61
500	$\boldsymbol{\omega} = \{90194.6, 40572.9, 12306.1, 3287.54, 848.462, 217.153, 55.5924, 14.3682, 3.86057, 1.19384, 0.556941\}$ $\boldsymbol{\beta} = \{0.000516568, 0.000826427, 0.00163028, 0.003409, 0.00722646, 0.0153633, 0.0326589, 0.0691913, 0.144324, 0.282088, 0.442766\}$	124.81
550	$\boldsymbol{\omega} = \{108811., 48090.9, 14288., 3744.58, 949.075, 238.635, 60.018, 15.233, 4.01502, 1.21651, 0.558356\}$ $\boldsymbol{\beta} = \{0.000467079, 0.000755611, 0.00150923, 0.0031922, 0.00684112, 0.0147012, 0.031589, 0.0676602, 0.142759, 0.282494, 0.448031\}$	134.77
600	$\boldsymbol{\omega} = \{129138., 56157.2, 16373.2, 4216.85, 1051.28, 260.095, 64.3657, 16.0685, 4.16174, 1.23769, 0.559662\}$ $\boldsymbol{\beta} = \{0.000425891, 0.000696065, 0.00140618, 0.00300546, 0.00650546, 0.0141179, 0.0306352, 0.0662761, 0.141312, 0.2828, 0.452819\}$	144.52
650	$\boldsymbol{\omega} = \{151170., 64760.4, 18557.4, 4703.5, 1154.97, 281.534, 68.6429, 16.8779, 4.30172, 1.2576, 0.560873\}$ $\boldsymbol{\beta} = \{0.000391093, 0.000645281, 0.00131728, 0.00284264, 0.00620977, 0.0135987, 0.0297771, 0.065015, 0.139966, 0.283028, 0.457208\}$	154.08
700	$\boldsymbol{\omega} = \{174902., 73889.9, 20837.1, 5203.8, 1260.04, 302.953, 72.8562, 17.664, 4.43575, 1.27639, 0.562005\}$ $\boldsymbol{\beta} = \{0.000361319, 0.000601444, 0.00123973, 0.00269919, 0.0059468, 0.0131326, 0.0289991, 0.0638582, 0.138708, 0.283195, 0.461258\}$	163.46
750	$\boldsymbol{\omega} = \{200329., 83536.2, 23209.1, 5717.11, 1366.41, 324.353, 77.011, 18.4291, 4.56449, 1.29422, 0.563067\}$ $\boldsymbol{\beta} = \{0.000335564, 0.00056321, 0.00117144, 0.0025717, 0.00571101, 0.012711, 0.0282888, 0.0627909, 0.137528, 0.283312, 0.465017\}$	172.68
800	$\boldsymbol{\omega} = \{227446., 93690.5, 25670.5, 6242.84, 1474., 345.736, 81.1118, 19.175, 4.68847, 1.31117, 0.564067\}$ $\boldsymbol{\beta} = \{0.000313074, 0.000529564, 0.00111079, 0.00245751, 0.0054981, 0.0123272, 0.0276365, 0.0618011, 0.136416, 0.283388, 0.468522\}$	181.75
850	$\boldsymbol{\omega} = \{256250., 104345., 28218.7, 6780.48, 1582.76, 367.102, 85.1626, 19.9034, 4.80816, 1.32735, 0.565012\}$ $\boldsymbol{\beta} = \{0.000293271, 0.000499722, 0.00105653, 0.00235455, 0.00530463, 0.0119758, 0.0270345, 0.060879, 0.135366, 0.283431, 0.471805\}$	190.67
900	$\boldsymbol{\omega} = \{286735., 115491., 30851.3, 7329.59, 1692.62, 388.453, 89.167, 20.6157, 4.92393, 1.34284, 0.565909\}$ $\boldsymbol{\beta} = \{0.000275706, 0.000473069, 0.00100769, 0.00226114, 0.00512788, 0.0116523, 0.0264763, 0.0600167, 0.13437, 0.283447, 0.474892\}$	199.47
950	$\boldsymbol{\omega} = \{318899., 127124., 33566., 7889.72, 1803.54, 409.788, 93.128, 21.3131, 5.03613, 1.3577, 0.566763\}$ $\boldsymbol{\beta} = \{0.000260025, 0.000449118, 0.000963456, 0.00217597, 0.00496559, 0.0113534, 0.0259566, 0.0592074, 0.133425, 0.283439, 0.477805\}$	208.14
1000	$\boldsymbol{\omega} = \{352738., 139235., 36360.9, 8460.51, 1915.46, 431.11, 97.0482, 21.9966, 5.14503, 1.37198, 0.567577\}$ $\boldsymbol{\beta} = \{0.000245944, 0.000427475, 0.000923201, 0.00209793, 0.00481594, 0.0110759, 0.025471, 0.0584454, 0.132524, 0.283412, 0.480561\}$	216.69

Table B.15 Parameters for optimized $P = 11$ SRJ schemes for various values of N ($N = 2^k$ $k = 5, \ldots, 12$).

N	Optimal Scheme Parameters	ρ
32	$\boldsymbol{\omega} = \{396.431, 279.642, 153.434, 73.1608, 32.6594, 14.2231, 6.19822, 2.7735, 1.32935, 0.735639, 0.523157\}$ $\boldsymbol{\beta} = \{0.00766724, 0.00932051, 0.0129802, 0.0194268, 0.0300202, 0.0469263, 0.0733318, 0.113197, 0.169003, 0.234379, 0.283748\}$	12.02
64	$\boldsymbol{\omega} = \{1561.74, 995.92, 472.015, 193.505, 74.7492, 28.3009, 10.7213, 4.15116, 1.71187, 0.820855, 0.530347\}$ $\boldsymbol{\beta} = \{0.00404857, 0.00522535, 0.00791843, 0.012907, 0.0216291, 0.0365655, 0.0618008, 0.103457, 0.168173, 0.2534, 0.324875\}$	22.17
128	$\boldsymbol{\omega} = \{6143.1, 3508.98, 1431.82, 506.33, 170.137, 56.292, 18.6338, 6.27437, 2.23256, 0.92445, 0.538451\}$ $\boldsymbol{\beta} = \{0.00207453, 0.00286193, 0.00473326, 0.00839676, 0.0152373, 0.0278259, 0.0508049, 0.0921114, 0.162945, 0.267575, 0.365434\}$	40.31
256	$\boldsymbol{\omega} = \{24118.5, 12241.2, 4294.93, 1314.76, 385.775, 111.904, 32.4724, 9.54483, 2.93972, 1.04945, 0.547444\}$ $\boldsymbol{\beta} = \{0.00103495, 0.00153545, 0.00277641, 0.00535641, 0.0105195, 0.0207491, 0.0409204, 0.0803054, 0.154511, 0.277065, 0.405226\}$	72.17
512	$\boldsymbol{\omega} = \{94506.1, 42326.5, 12772.1, 3395.79, 872.457, 222.311, 56.6621, 14.5787, 3.89841, 1.19943, 0.557292\}$ $\boldsymbol{\beta} = \{0.000503804, 0.000808234, 0.00159933, 0.00335382, 0.00712881, 0.0151963, 0.0323902, 0.0688089, 0.143937, 0.282196, 0.444078\}$	127.22
1024	$\boldsymbol{\omega} = \{369575., 145216., 37730.4, 8738.16, 1969.52, 441.339, 98.9161, 22.3201, 5.19622, 1.37865, 0.567955\}$ $\boldsymbol{\beta} = \{0.000239683, 0.00041781, 0.000905131, 0.00206274, 0.00474813, 0.0109496, 0.0252489, 0.058095, 0.132107, 0.283393, 0.481833\}$	220.76
2048	$\boldsymbol{\omega} = \{1442350, 494839., 110897., 22426.1, 4440.04, 875.522, 172.665, 34.2146, 6.95091, 1.59211, 0.579383\}$ $\boldsymbol{\beta} = \{0.00011157, 0.000212226, 0.000503422, 0.001247, 0.00311134, 0.00777184, 0.0194126, 0.0484209, 0.119719, 0.281141, 0.518349\}$	376.94
4096	$\boldsymbol{\omega} = \{5617760, 1676280, 324683., 57443.8, 9998.5, 1735.51, 301.272, 52.4724, 9.32049, 1.84572, 0.591523\}$ $\boldsymbol{\beta} = \{0.0000508577, 0.000105956, 0.00027523, 0.000741393, 0.00200742, 0.00543916, 0.0147371, 0.0398943, 0.1073, 0.275945, 0.553504\}$	633.15

Table B.16 Parameters for optimized $P = 12$ SRJ schemes for various values of N ($N = 100 + 50k$, $k = 0, \ldots, 18$).

N	Optimal Scheme Parameters	ρ
100	$\boldsymbol{\omega} = \{3816.09, 2449.48, 1171.54, 484.552, 188.591, 71.7552, 27.1524, 10.3398, 4.03462, 1.67921, 0.814946, 0.530037\}$ $\boldsymbol{\beta} = \{0.00249324, 0.0032053, 0.00483143, 0.0078347, 0.0130695, 0.0220179, 0.0371777, 0.0626105, 0.104333, 0.168776, 0.25022, 0.32343\}$	33.60
150	$\boldsymbol{\omega} = \{8512.57, 5158.9, 2278.59, 870.502, 314.372, 111.324, 39.2426, 13.9067, 5.03399, 1.93522, 0.867388, 0.534161\}$ $\boldsymbol{\beta} = \{0.00168233, 0.00223889, 0.00353484, 0.00599785, 0.0104405, 0.0183241, 0.0322191, 0.0565309, 0.0983065, 0.166362, 0.259643, 0.344719\}$	47.75
200	$\boldsymbol{\omega} = \{15036.2, 8734.56, 3644.21, 1316.62, 451.186, 151.917, 50.9495, 17.1663, 5.89563, 2.1434, 0.907955, 0.537264\}$ $\boldsymbol{\beta} = \{0.0012661, 0.00172882, 0.0028221, 0.00494455, 0.00886996, 0.0160265, 0.0290014, 0.0523866, 0.0938883, 0.164052, 0.265144, 0.35987\}$	61.09
250	$\boldsymbol{\omega} = \{23372.3, 13127.2, 5239.13, 1813.2, 596.796, 193.289, 62.3814, 20.2163, 6.66795, 2.32215, 0.941482, 0.539767\}$ $\boldsymbol{\beta} = \{0.00101274, 0.00141158, 0.00236512, 0.00424802, 0.0078001, 0.0144149, 0.0266745, 0.0492834, 0.0904131, 0.161946, 0.268825, 0.371606\}$	73.83
300	$\boldsymbol{\omega} = \{33509.2, 18300.1, 7043.04, 2353.87, 749.797, 235.292, 73.5996, 23.1092, 7.37598, 2.48054, 0.97028, 0.541873\}$ $\boldsymbol{\beta} = \{0.000842345, 0.00119444, 0.00204463, 0.00374746, 0.00701309, 0.0132018, 0.0248808, 0.0468262, 0.0875581, 0.160039, 0.271482, 0.381171\}$	86.10
350	$\boldsymbol{\omega} = \{45437.3, 24224.4, 9040.67, 2934.05, 909.215, 277.826, 84.6431, 25.8779, 8.03469, 2.62379, 0.995654, 0.543695\}$ $\boldsymbol{\beta} = \{0.000719967, 0.00103614, 0.00180618, 0.00336745, 0.00640404, 0.012245, 0.0234383, 0.0448065, 0.0851418, 0.158305, 0.273495, 0.389234\}$	97.97
400	$\boldsymbol{\omega} = \{59148.4, 30876.1, 11220., 3550.26, 1074.33, 320.818, 95.5393, 28.5444, 8.65395, 2.75524, 1.01842, 0.545305\}$ $\boldsymbol{\beta} = \{0.000627865, 0.000915427, 0.00162115, 0.00306749, 0.00591535, 0.0114649, 0.0222425, 0.0431011, 0.0830517, 0.156721, 0.275074, 0.396198\}$	109.52
450	$\boldsymbol{\omega} = \{74635.1, 38235.1, 13571.2, 4199.71, 1244.59, 364.215, 106.308, 31.1249, 9.24068, 2.87718, 1.03912, 0.546747\}$ $\boldsymbol{\beta} = \{0.000556078, 0.000820244, 0.001473, 0.00282369, 0.00551249, 0.0108128, 0.0212283, 0.0416316, 0.0812134, 0.155264, 0.276343, 0.402322\}$	120.77
500	$\boldsymbol{\omega} = \{91891., 46283.8, 16086.1, 4880.15, 1419.54, 407.973, 116.965, 33.6311, 9.79994, 2.99126, 1.05814, 0.548056\}$ $\boldsymbol{\beta} = \{0.00049858, 0.000743205, 0.00135144, 0.002621, 0.00517332, 0.0102569, 0.0203527, 0.0403451, 0.0795753, 0.153916, 0.277382, 0.407785\}$	131.77
550	$\boldsymbol{\omega} = \{110910., 55006.9, 18757.6, 5589.69, 1598.81, 452.059, 127.522, 36.0724, 10.3356, 3.09868, 1.07577, 0.549254\}$ $\boldsymbol{\beta} = \{0.000451511, 0.000679537, 0.00124974, 0.00244939, 0.0048829, 0.00977557, 0.0195859, 0.0392045, 0.0780998, 0.152663, 0.278245, 0.412713\}$	142.53
600	$\boldsymbol{\omega} = \{131688., 64390.8, 21579.6, 6326.71, 1782.12, 496.444, 137.99, 38.4561, 10.8506, 3.20041, 1.09222, 0.55036\}$ $\boldsymbol{\beta} = \{0.000412285, 0.000626012, 0.00116328, 0.00230192, 0.00463078, 0.0093535, 0.0189066, 0.0381824, 0.076759, 0.151492, 0.278972, 0.4172\}$	153.09
650	$\boldsymbol{\omega} = \{154218., 74423.2, 24546.8, 7089.83, 1969.18, 541.105, 148.375, 40.7882, 11.3475, 3.29717, 1.10766, 0.551388\}$ $\boldsymbol{\beta} = \{0.000379105, 0.000580369, 0.00108879, 0.00217362, 0.00440935, 0.0089794, 0.0182987, 0.0372586, 0.0755316, 0.150394, 0.279588, 0.421318\}$	163.47
700	$\boldsymbol{\omega} = \{178498., 85093., 27654.3, 7877.82, 2159.79, 586.021, 158.686, 43.0737, 11.8282, 3.38957, 1.12222, 0.552348\}$ $\boldsymbol{\beta} = \{0.000350682, 0.000540974, 0.00102389, 0.0020608, 0.00421299, 0.00864481, 0.0177503, 0.0364172, 0.0744008, 0.149361, 0.280116, 0.425121\}$	173.67
750	$\boldsymbol{\omega} = \{204521., 96390., 30898., 8689.62, 2353.73, 631.175, 168.927, 45.3167, 12.2943, 3.4781, 1.13601, 0.553249\}$ $\boldsymbol{\beta} = \{0.00032607, 0.000506619, 0.000966783, 0.00196071, 0.00403737, 0.00834321, 0.017252, 0.035646, 0.0733534, 0.148384, 0.28057, 0.428654\}$	183.70
800	$\boldsymbol{\omega} = \{232286., 108305., 34273.9, 9524.26, 2550.85, 676.553, 179.104, 47.5208, 12.7472, 3.56315, 1.14911, 0.554098\}$ $\boldsymbol{\beta} = \{0.000304555, 0.000476389, 0.00091612, 0.00187122, 0.00387917, 0.00806952, 0.0167964, 0.0349352, 0.0723784, 0.147459, 0.280963, 0.431951\}$	193.59
850	$\boldsymbol{\omega} = \{261787., 120829., 37778.5, 10380.9, 2750.99, 722.141, 189.221, 49.689, 13.188, 3.64508, 1.16161, 0.554902\}$ $\boldsymbol{\beta} = \{0.000285593, 0.000449578, 0.00087084, 0.00179064, 0.00373575, 0.00781967, 0.0163775, 0.0342768, 0.0714671, 0.14658, 0.281305, 0.435042\}$	203.34
900	$\boldsymbol{\omega} = \{293021., 133954., 41408.7, 11258.7, 2954.01, 767.927, 199.282, 51.824, 13.6178, 3.72416, 1.17355, 0.555664\}$ $\boldsymbol{\beta} = \{0.000268759, 0.000425635, 0.000830107, 0.00171765, 0.00360498, 0.0075904, 0.0159906, 0.0336644, 0.0706121, 0.145743, 0.281603, 0.43795\}$	212.97
950	$\boldsymbol{\omega} = \{325986., 147673., 45161.3, 12157., 3159.78, 813.901, 209.291, 53.928, 14.0374, 3.80065, 1.185, 0.556389\}$ $\boldsymbol{\beta} = \{0.000253717, 0.000404119, 0.000793252, 0.00165119, 0.00348515, 0.00737903, 0.0156317, 0.0330924, 0.0698072, 0.144944, 0.281863, 0.440695\}$	222.47
1000	$\boldsymbol{\omega} = \{360677., 161978., 49033.7, 13075.1, 3368.2, 860.054, 219.25, 56.0032, 14.4476, 3.87476, 1.19599, 0.557081\}$ $\boldsymbol{\beta} = \{0.000240198, 0.000384678, 0.000759733, 0.00159036, 0.00337486, 0.00718335, 0.0152974, 0.0325564, 0.0690473, 0.14418, 0.28209, 0.443295\}$	231.85

Table B.17 Parameters for optimized $P = 12$ SRJ schemes for various values of N ($N = 2^k\ k = 5, \ldots, 12$).

N	Optimal Scheme Parameters	ρ
32	$\boldsymbol{\omega} = \{399.282, 296.323, 175.654, 90.9514, 43.9229, 20.5641, 9.54942, 4.47911, 2.17305, 1.13535, 0.692165, 0.520193\}$ $\boldsymbol{\beta} = \{0.00717978, 0.00847231, 0.0112881, 0.0161292, 0.0238518, 0.0357931, 0.0539181, 0.0808575, 0.119387, 0.170519, 0.20623, 0.266374\}$	12.22
64	$\boldsymbol{\omega} = \{1576.99, 1074.79, 560.296, 253.075, 107.095, 44.1519, 18.083, 7.46292, 3.16734, 1.43996, 0.763014, 0.525858\}$ $\boldsymbol{\beta} = \{0.00380547, 0.0047207, 0.00676987, 0.0104438, 0.0166204, 0.0267627, 0.0432181, 0.0695623, 0.110575, 0.170246, 0.236935, 0.300341\}$	22.71
128	$\boldsymbol{\omega} = \{6219.99, 3856.45, 1757.41, 692.507, 257.487, 93.7674, 33.9796, 12.3838, 4.61567, 1.8302, 0.846238, 0.532512\}$ $\boldsymbol{\beta} = \{0.0019642, 0.00257829, 0.00399715, 0.00666322, 0.0114074, 0.0197032, 0.0340994, 0.0588784, 0.100697, 0.167428, 0.256209, 0.336374\}$	41.64
256	$\boldsymbol{\omega} = \{24493.8, 13707.4, 5444.91, 1875.85, 614.788, 198.299, 63.738, 20.5709, 6.75595, 2.34211, 0.945155, 0.540038\}$ $\boldsymbol{\beta} = \{0.000988845, 0.00138134, 0.0023209, 0.00417958, 0.00769344, 0.0142519, 0.0264357, 0.0489597, 0.0900422, 0.161707, 0.269188, 0.372852\}$	75.32
512	$\boldsymbol{\omega} = \{96295., 48316.3, 16713.2, 5047.85, 1462.18, 418.525, 119.508, 34.2226, 9.93052, 3.0176, 1.06249, 0.548353\}$ $\boldsymbol{\beta} = \{0.000486446, 0.000726847, 0.00132543, 0.00257728, 0.00509962, 0.0101352, 0.0201597, 0.0400592, 0.0792075, 0.153607, 0.277603, 0.409012\}$	134.37
1024	$\boldsymbol{\omega} = \{377942., 169051., 50934.3, 13522.7, 3469.15, 882.268, 224.013, 56.9895, 14.6414, 3.90954, 1.20112, 0.557403\}$ $\boldsymbol{\beta} = \{0.000234184, 0.000375997, 0.000744695, 0.00156296, 0.00332495, 0.00709444, 0.0151449, 0.0323108, 0.0686972, 0.143825, 0.282189, 0.444496\}$	236.32
2048	$\boldsymbol{\omega} = \{1480870, 587626., 154364., 36112.1, 8217.37, 1858.2, 419.746, 94.9399, 21.6287, 5.08649, 1.36434, 0.567144\}$ $\boldsymbol{\beta} = \{0.000110457, 0.00019134, 0.000411749, 0.000932607, 0.00213412, 0.00489319, 0.0112226, 0.0257294, 0.0588522, 0.133007, 0.283417, 0.479098\}$	409.64
4096	$\boldsymbol{\omega} = \{5792620, 2030850, 465804., 96218.1, 19441.1, 3910.7, 786.112, 158.157, 31.985, 6.63805, 1.55596, 0.577534\}$ $\boldsymbol{\beta} = \{0.0000510879, 0.0000958256, 0.000224079, 0.000547792, 0.00134943, 0.0033284, 0.00821094, 0.0202514, 0.0498712, 0.121686, 0.281701, 0.512683\}$	699.67

Table B.18 Parameters for optimized $P = 13$ SRJ schemes for various values of N ($N = 100 + 50k$, $k = 0, \ldots, 18$).

N	Optimal Scheme Parameters	ρ
100	$\boldsymbol{\omega} = \{3850.42, 2628.04, 1372.85, 621.295, 263.269, 108.54, 44.3191, 18.1012, 7.47022, 3.1732, 1.46178, 0.770791, 0.52618\}$ $\boldsymbol{\beta} = \{0.00235853, 0.00292322, 0.00418701, 0.00645168, 0.0102578, 0.0165109, 0.0266858, 0.0431288, 0.0694093, 0.110458, 0.161845, 0.243877, 0.301907\}$	34.43
150	$\boldsymbol{\omega} = \{8600.11, 5583., 2713.17, 1140.26, 450.396, 173.646, 66.4069, 25.4013, 9.80185, 3.88069, 1.65106, 0.810599, 0.529505\}$ $\boldsymbol{\beta} = \{0.0015982, 0.00204078, 0.00304774, 0.00489728, 0.00810045, 0.0135406, 0.0227101, 0.0380865, 0.06365, 0.105277, 0.164788, 0.25184, 0.320423\}$	49.15
200	$\boldsymbol{\omega} = \{15205.3, 9511.83, 4388.27, 1750.36, 658.133, 242.067, 88.403, 32.2912, 11.8866, 4.4795, 1.80488, 0.84212, 0.532062\}$ $\boldsymbol{\beta} = \{0.00120674, 0.00157554, 0.00242509, 0.00401446, 0.00682843, 0.0117242, 0.0201874, 0.0347575, 0.059662, 0.101442, 0.165082, 0.257141, 0.333953\}$	63.10
250	$\boldsymbol{\omega} = \{23653.2, 14365.9, 6363.85, 2438.1, 882.631, 313.079, 110.345, 38.8978, 13.808, 5.00982, 1.93667, 0.868438, 0.534148\}$ $\boldsymbol{\beta} = \{0.000967836, 0.00128633, 0.00202738, 0.00343434, 0.00596933, 0.0104642, 0.0183895, 0.0323153, 0.0566335, 0.0983815, 0.164575, 0.260976, 0.34458\}$	76.48
300	$\boldsymbol{\omega} = \{33933.9, 20108.6, 8615.78, 3194.46, 1121.42, 386.232, 132.247, 45.289, 15.6094, 5.4916, 2.05313, 0.891162, 0.535915\}$ $\boldsymbol{\beta} = \{0.000806822, 0.00108844, 0.00174925, 0.00301942, 0.00534147, 0.00952376, 0.0170188, 0.0304106, 0.0542077, 0.0958339, 0.163779, 0.263906, 0.353314\}$	89.41
350	$\boldsymbol{\omega} = \{46039., 26711., 11125.8, 4012.89, 1372.77, 461.214, 154.117, 51.5071, 17.3169, 5.93653, 2.15818, 0.911239, 0.537452\}$ $\boldsymbol{\beta} = \{0.000690964, 0.000944197, 0.00154278, 0.00270562, 0.00485811, 0.00878712, 0.0159262, 0.0288638, 0.0521946, 0.0936541, 0.162881, 0.266231, 0.36072\}$	101.97
400	$\boldsymbol{\omega} = \{59961.3, 34148.9, 13879.2, 4888.34, 1635.37, 537.796, 175.961, 57.581, 18.9481, 6.35222, 2.25433, 0.929279, 0.538812\}$ $\boldsymbol{\beta} = \{0.000603625, 0.000834233, 0.00138288, 0.00245871, 0.00447198, 0.00818994, 0.0150272, 0.0275709, 0.050481, 0.091751, 0.161958, 0.268127, 0.367144\}$	114.20
450	$\boldsymbol{\omega} = \{75694.4, 42401.8, 16864.2, 5816.77, 1908.21, 615.806, 197.782, 63.5318, 20.5156, 6.74396, 2.34329, 0.945698, 0.540035\}$ $\boldsymbol{\beta} = \{0.000535451, 0.000747535, 0.00125506, 0.00225859, 0.00415487, 0.00769319, 0.0142696, 0.0264665, 0.0489941, 0.0900639, 0.161042, 0.269708, 0.372811\}$	126.15
500	$\boldsymbol{\omega} = \{93232.9, 51452., 20070.7, 6794.87, 2190.47, 695.105, 219.583, 69.3753, 22.0284, 7.11557, 2.42632, 0.960792, 0.541147\}$ $\boldsymbol{\beta} = \{0.000480773, 0.000677371, 0.00115032, 0.0020926, 0.00388878, 0.0072716, 0.0136193, 0.025507, 0.0476845, 0.0885501, 0.16015, 0.271048, 0.377879\}$	137.85
550	$\boldsymbol{\omega} = \{112572., 61283.9, 23490., 7819.84, 2481.47, 775.581, 241.365, 75.124, 23.4937, 7.46998, 2.50434, 0.974781, 0.542167\}$ $\boldsymbol{\beta} = \{0.000435959, 0.000619389, 0.00106281, 0.00195237, 0.00366162, 0.00690802, 0.0130527, 0.024662, 0.0465171, 0.0871783, 0.159289, 0.2722, 0.382461\}$	149.33
600	$\boldsymbol{\omega} = \{133706., 71883.4, 27114.6, 8889.31, 2780.65, 857.141, 263.131, 80.7878, 24.9172, 7.80944, 2.57806, 0.987834, 0.543109\}$ $\boldsymbol{\beta} = \{0.00039857, 0.000570647, 0.000988491, 0.00183209, 0.00346494, 0.00659031, 0.012553, 0.0239094, 0.0454659, 0.085925, 0.158461, 0.273201, 0.38664\}$	160.61
650	$\boldsymbol{\omega} = \{156632., 83238., 30937.8, 10001.2, 3087.51, 939.707, 284.882, 86.3749, 26.3034, 8.13575, 2.64805, 1.00008, 0.543985\}$ $\boldsymbol{\beta} = \{0.000366911, 0.000529085, 0.000924528, 0.00172762, 0.00329263, 0.00630962, 0.0121077, 0.0232328, 0.0445115, 0.0847721, 0.157665, 0.274079, 0.390481\}$	171.71
700	$\boldsymbol{\omega} = \{181345., 95336.2, 34953.7, 11153.7, 3401.64, 1023.21, 306.618, 91.8922, 27.6561, 8.45039, 2.71475, 1.01163, 0.544804\}$ $\boldsymbol{\beta} = \{0.000339764, 0.000493215, 0.000868846, 0.00163592, 0.00314016, 0.00605932, 0.0117075, 0.0226198, 0.0436389, 0.0837054, 0.156902, 0.274856, 0.394033\}$	182.64
750	$\boldsymbol{\omega} = \{207842., 108167., 39157., 12345.1, 3722.65, 1107.6, 328.341, 97.3453, 28.9783, 8.75456, 2.77854, 1.02256, 0.545574\}$ $\boldsymbol{\beta} = \{0.000316234, 0.000461934, 0.000819899, 0.00155467, 0.00300409, 0.00583432, 0.0113452, 0.0220606, 0.042836, 0.0827132, 0.15617, 0.275548, 0.397336\}$	193.41
800	$\boldsymbol{\omega} = \{236120., 121722., 43542.7, 13574.1, 4050.24, 1192.81, 350.051, 102.739, 30.2727, 9.04927, 2.83973, 1.03294, 0.5463\}$ $\boldsymbol{\beta} = \{0.000295649, 0.00043441, 0.000776506, 0.00148213, 0.00288174, 0.00563066, 0.011015, 0.0215473, 0.0420935, 0.0817863, 0.155467, 0.276167, 0.400423\}$	204.03
850	$\boldsymbol{\omega} = \{266174., 135991., 48106.5, 14839.3, 4384.1, 1278.81, 371.75, 108.078, 31.5415, 9.33538, 2.89858, 1.04284, 0.546987\}$ $\boldsymbol{\beta} = \{0.000277491, 0.000409999, 0.000737751, 0.00141689, 0.00277102, 0.00544519, 0.0107124, 0.0210739, 0.0414035, 0.0809169, 0.154791, 0.276725, 0.403319\}$	214.53
900	$\boldsymbol{\omega} = \{298003., 150967., 52844.3, 16139.5, 4723.96, 1365.56, 393.437, 113.366, 32.7868, 9.61362, 2.95529, 1.0523, 0.54764\}$ $\boldsymbol{\beta} = \{0.000261358, 0.000388199, 0.000702912, 0.00135787, 0.00267024, 0.00527538, 0.0104337, 0.0206352, 0.0407597, 0.0800987, 0.154141, 0.27723, 0.406046\}$	224.89
950	$\boldsymbol{\omega} = \{331603., 166640., 57752.4, 17473.7, 5069.61, 1453.01, 415.114, 118.606, 34.0103, 9.88464, 3.01007, 1.06136, 0.548261\}$ $\boldsymbol{\beta} = \{0.000246931, 0.00036861, 0.000671408, 0.00130418, 0.00257804, 0.00511916, 0.0101759, 0.0202269, 0.0401568, 0.0793262, 0.153514, 0.277689, 0.408623\}$	235.13
1000	$\boldsymbol{\omega} = \{366972., 183005., 62827.2, 18840.9, 5420.8, 1541.14, 436.781, 123.801, 35.2134, 10.149, 3.06307, 1.07006, 0.548854\}$ $\boldsymbol{\beta} = \{0.000233955, 0.000350909, 0.000642773, 0.0012551, 0.0024933, 0.00497482, 0.00993636, 0.0198456, 0.0395904, 0.0785949, 0.152911, 0.278107, 0.411065\}$	245.27

Table B.19 Parameters for optimized $P = 13$ SRJ schemes for various values of N ($N = 2^k$ $k = 5, \ldots, 12$).

N	Optimal Scheme Parameters	ρ
32	$\boldsymbol{\omega} = \{401.471, 309.92, 195.459, 108.339, 55.9022, 27.8438, 13.6785, 6.72758, 3.36452, 1.75033, 1.05207, 0.686717, 0.518762\}$ $\boldsymbol{\beta} = \{0.00675791, 0.0077945, 0.0100255, 0.0137902, 0.0196578, 0.0285024, 0.0415987, 0.0607023, 0.088016, 0.126799, 0.117918, 0.222136, 0.256303\}$	12.38
64	$\boldsymbol{\omega} = \{1589.04, 1142.32, 644.729, 316.512, 145.007, 64.4121, 28.2867, 12.4267, 5.52727, 2.53974, 1.28491, 0.733286, 0.522941\}$ $\boldsymbol{\beta} = \{0.00358591, 0.00431074, 0.00590608, 0.00869324, 0.0132327, 0.0204319, 0.0317095, 0.0492101, 0.0759855, 0.11602, 0.153206, 0.23494, 0.282768\}$	23.16
128	$\boldsymbol{\omega} = \{6280.77, 4159.32, 2079.6, 899.633, 365.213, 144.533, 56.7035, 22.2513, 8.81404, 3.58658, 1.57341, 0.794381, 0.528164\}$ $\boldsymbol{\beta} = \{0.00186278, 0.00235054, 0.00345302, 0.00545819, 0.00888982, 0.0146421, 0.0242045, 0.0400093, 0.0658841, 0.107333, 0.16405, 0.248777, 0.313085\}$	42.78
256	$\boldsymbol{\omega} = \{24790.3, 15008.7, 6619.9, 2525.38, 910.571, 321.752, 112.975, 39.6752, 14.0299, 5.06992, 1.95136, 0.871332, 0.534375\}$ $\boldsymbol{\beta} = \{0.0009945278, 0.00125876, 0.00198896, 0.00337751, 0.00588404, 0.0103375, 0.0182063, 0.0320629, 0.0563154, 0.0980522, 0.164489, 0.261368, 0.345714\}$	78.06
512	$\boldsymbol{\omega} = \{97710.2, 53741., 20872.2, 7036.69, 2259.54, 714.315, 224.812, 70.7632, 22.3842, 7.2021, 2.44547, 0.964243, 0.5414\}$ $\boldsymbol{\beta} = \{0.000469226, 0.000662473, 0.00112793, 0.00205685, 0.00383108, 0.00717958, 0.0134764, 0.0252947, 0.0473925, 0.0882089, 0.159941, 0.27134, 0.379019\}$	140.62
1024	$\boldsymbol{\omega} = \{384576., 191103., 65321.5, 19508.6, 5591.29, 1583.67, 447.177, 126.28, 35.7841, 10.2736, 3.08792, 1.07412, 0.549129\}$ $\boldsymbol{\beta} = \{0.000228181, 0.000343005, 0.000629932, 0.001233, 0.002455, 0.0049033, 0.00982729, 0.0196712, 0.0393302, 0.0782572, 0.152629, 0.278294, 0.412193\}$	250.09
2048	$\boldsymbol{\omega} = \{1511490, 675363., 203229., 53892.9, 13810.2, 3507.87, 889.156, 225.393, 57.2711, 14.697, 3.92023, 1.20273, 0.557498\}$ $\boldsymbol{\beta} = \{0.000108834, 0.000174858, 0.000346582, 0.000727906, 0.00154954, 0.00330852, 0.00706863, 0.0151018, 0.0322407, 0.0685947, 0.143668, 0.282261, 0.444849\}$	438.96
4096	$\boldsymbol{\omega} = \{5932130, 2373560, 629341., 148493., 34065.2, 7764.3, 1767.21, 402.245, 91.7049, 21.0624, 4.99621, 1.35246, 0.566463\}$ $\boldsymbol{\beta} = \{0.0000509606, 0.0000878108, 0.000187895, 0.000423376, 0.000964004, 0.00219946, 0.0050204, 0.011458, 0.0261403, 0.0594939, 0.133745, 0.283445, 0.476784\}$	760.22

Table B.20 Parameters for optimized $P = 14$ SRJ schemes for various values of N ($N = 100 + 50k$, $k = 0, \ldots, 18$).

N	Optimal Scheme Parameters	ρ
100	$\boldsymbol{\omega} = \{3878.55, 2785.67, 1570.02, 769.498, 351.868, 155.898, 68.1753, 29.7098, 12.9941, 5.75755, 2.75287, 1.36341, 0.743467, 0.523835\}$ $\boldsymbol{\beta} = \{0.00223066, 0.00268294, 0.00367875, 0.00541949, 0.00825729, 0.012765, 0.019848, 0.0309124, 0.0481036, 0.0748715, 0.0959847, 0.170428, 0.2368, 0.288017\}$	35.12
150	$\boldsymbol{\omega} = \{8671.64, 5959.56, 3144.72, 1438.03, 615.285, 255.886, 105.238, 43.1569, 17.7519, 7.38476, 3.24126, 1.48992, 0.772445, 0.526323\}$ $\boldsymbol{\beta} = \{0.00151803, 0.00187383, 0.00266836, 0.00408745, 0.00646311, 0.0103505, 0.0166552, 0.0268341, 0.0431958, 0.0694114, 0.100573, 0.169923, 0.243817, 0.302629\}$	50.34
200	$\boldsymbol{\omega} = \{15342.9, 10203.7, 5132.76, 2234.12, 912.112, 362.768, 142.862, 56.1217, 22.1054, 8.79573, 3.65467, 1.59758, 0.796262, 0.528322\}$ $\boldsymbol{\beta} = \{0.00115001, 0.00144748, 0.00211891, 0.00333763, 0.00541838, 0.00889808, 0.0146726, 0.0242198, 0.0399456, 0.0657302, 0.101006, 0.169306, 0.248859, 0.313891\}$	64.84
250	$\boldsymbol{\omega} = \{23881.3, 15470.9, 7496.09, 3140.75, 1236.83, 475.305, 181.025, 68.7946, 26.2065, 10.0762, 4.02155, 1.69204, 0.816583, 0.529997\}$ $\boldsymbol{\beta} = \{0.000924757, 0.00118231, 0.00176872, 0.00284695, 0.00471686, 0.00789814, 0.0132733, 0.0223266, 0.0375256, 0.0629164, 0.100286, 0.168614, 0.252701, 0.323018\}$	78.79
300	$\boldsymbol{\omega} = \{34278.1, 21725.7, 10207.4, 4146.18, 1585.72, 592.623, 219.656, 81.2544, 30.1232, 11.2643, 4.35503, 1.7766, 0.834356, 0.531439\}$ $\boldsymbol{\beta} = \{0.000772581, 0.00100079, 0.00152417, 0.002497, 0.00420622, 0.00715567, 0.0122136, 0.020863, 0.0356119, 0.0606376, 0.0992048, 0.167894, 0.25575, 0.330669\}$	92.3
350	$\boldsymbol{\omega} = \{46526., 28939.3, 13245.9, 5241.76, 1956.08, 714.096, 258.693, 93.545, 33.8951, 12.3814, 4.66283, 1.85345, 0.850189, 0.532707\}$ $\boldsymbol{\beta} = \{0.000662864, 0.000868418, 0.00134284, 0.00223296, 0.00381437, 0.00657652, 0.0113733, 0.0196825, 0.0340395, 0.0587264, 0.0980248, 0.16717, 0.258242, 0.337243\}$	105.45
400	$\boldsymbol{\omega} = \{60618.7, 37088., 16594.7, 6420.75, 2345.86, 839.25, 298.085, 105.695, 37.5482, 13.4414, 4.95006, 1.92411, 0.864496, 0.533839\}$ $\boldsymbol{\beta} = \{0.000580015, 0.000767473, 0.00120256, 0.00202561, 0.00350221, 0.00610869, 0.010685, 0.0187016, 0.0327121, 0.0570842, 0.0968455, 0.166458, 0.260328, 0.343\}$	118.29
450	$\boldsymbol{\omega} = \{76550.6, 46151.6, 20240.1, 7677.7, 2753.45, 967.72, 337.793, 117.724, 41.1009, 14.4541, 5.22033, 1.98968, 0.877567, 0.534862\}$ $\boldsymbol{\beta} = \{0.000515249, 0.000687865, 0.00109052, 0.00185784, 0.00324648, 0.00572076, 0.0101073, 0.017868, 0.0315686, 0.0556475, 0.0957059, 0.165764, 0.262105, 0.348115\}$	130.87
500	$\boldsymbol{\omega} = \{94317., 56112.2, 24170.1, 9008.13, 3177.53, 1099.21, 377.785, 129.647, 44.5668, 15.4266, 5.47629, 2.05098, 0.889618, 0.535797\}$ $\boldsymbol{\beta} = \{0.000463236, 0.000623426, 0.000998804, 0.00171891, 0.00303236, 0.00539248, 0.00961327, 0.0171471, 0.0305679, 0.054373, 0.0946203, 0.165092, 0.263642, 0.352715\}$	143.20
550	$\boldsymbol{\omega} = \{113913., 66954.2, 28374.8, 10408.2, 3617.01, 1233.48, 418.035, 141.476, 47.9566, 16.3643, 5.71997, 2.10865, 0.900811, 0.536657\}$ $\boldsymbol{\beta} = \{0.000420555, 0.000570165, 0.00092222, 0.0016017, 0.00284993, 0.00511011, 0.00918421, 0.0165148, 0.0296809, 0.0532295, 0.0935924, 0.164444, 0.264987, 0.356893\}$	155.31
600	$\boldsymbol{\omega} = \{135336., 78663.8, 32845.3, 11874.7, 4070.95, 1370.32, 458.522, 153.221, 51.2786, 17.2714, 5.95295, 2.16317, 0.91127, 0.537454\}$ $\boldsymbol{\beta} = \{0.00038908, 0.000525385, 0.000857231, 0.0015013, 0.00269227, 0.00486396, 0.00880696, 0.015954, 0.0288864, 0.0521941, 0.0926211, 0.163818, 0.266176, 0.360718\}$	167.24
650	$\boldsymbol{\omega} = \{158580., 91228.1, 37573.6, 13404.7, 4538.56, 1509.57, 499.227, 164.888, 54.5397, 18.1514, 6.17652, 2.21494, 0.921097, 0.538198\}$ $\boldsymbol{\beta} = \{0.000354693, 0.000487197, 0.000801333, 0.0014142, 0.00255438, 0.00464697, 0.00847177, 0.0154516, 0.0281686, 0.0512493, 0.0917036, 0.163215, 0.267237, 0.364244\}$	178.98
700	$\boldsymbol{\omega} = \{183644., 104636., 42552.7, 14995.7, 5019.15, 1651.07, 540.135, 176.484, 57.7456, 19.0071, 6.39171, 2.26428, 0.930369, 0.538894\}$ $\boldsymbol{\beta} = \{0.000328761, 0.000454234, 0.000752702, 0.00133783, 0.00243256, 0.00445388, 0.00817133, 0.0149979, 0.0275151, 0.0503814, 0.090836, 0.162634, 0.268191, 0.367514\}$	190.56
750	$\boldsymbol{\omega} = \{210523., 118876., 47776.4, 16645.5, 5512.09, 1794.7, 581.231, 188.016, 60.9012, 19.8408, 6.59939, 2.31147, 0.939153, 0.53955\}$ $\boldsymbol{\beta} = \{0.000306266, 0.000425485, 0.000709977, 0.00127025, 0.00232401, 0.00428065, 0.00789997, 0.0145854, 0.0269165, 0.0495796, 0.0900146, 0.162073, 0.269053, 0.370562\}$	201.99
800	$\boldsymbol{\omega} = \{239214., 133939., 53238.8, 18352.1, 6016.86, 1940.35, 622.504, 199.487, 64.0103, 20.6544, 6.80028, 2.35671, 0.947503, 0.540169\}$ $\boldsymbol{\beta} = \{0.000286569, 0.000400187, 0.00067212, 0.00120995, 0.00222654, 0.00412413, 0.00765326, 0.0142079, 0.026365, 0.0488352, 0.0892357, 0.161532, 0.269837, 0.373415\}$	213.28
850	$\boldsymbol{\omega} = \{269715., 149816., 58934.6, 20113.6, 6532.95, 2087.9, 663.944, 210.901, 67.0766, 21.4498, 6.99499, 2.40021, 0.955464, 0.540756\}$ $\boldsymbol{\beta} = \{0.000269181, 0.000377748, 0.000638326, 0.00115578, 0.00213845, 0.00398185, 0.00742767, 0.0138606, 0.0258544, 0.048141, 0.0884956, 0.161009, 0.270553, 0.376097\}$	224.44
900	$\boldsymbol{\omega} = \{302024., 166499., 64858.9, 21928.4, 7059.93, 2237.28, 705.541, 222.262, 70.1032, 22.2283, 7.18405, 2.44212, 0.963074, 0.541314\}$ $\boldsymbol{\beta} = \{0.000253722, 0.000357707, 0.00060796, 0.00110682, 0.00205836, 0.00385178, 0.00722033, 0.0135396, 0.0253796, 0.0474912, 0.0877914, 0.160504, 0.271211, 0.378626\}$	235.47
950	$\boldsymbol{\omega} = \{336137., 183978., 71007.3, 23794.9, 7597.41, 2388.4, 747.287, 233.574, 73.0926, 22.9911, 7.36793, 2.48257, 0.970367, 0.541846\}$ $\boldsymbol{\beta} = \{0.000239888, 0.000339698, 0.000580514, 0.00106231, 0.00198518, 0.00373232, 0.00702889, 0.0132417, 0.0249364, 0.0468808, 0.0871199, 0.160016, 0.271817, 0.38102\}$	246.38
1000	$\boldsymbol{\omega} = \{372052., 202248., 77375.6, 25711.8, 8145.01, 2541.19, 789.175, 244.838, 76.0472, 23.7394, 7.54701, 2.52169, 0.97737, 0.542354\}$ $\boldsymbol{\beta} = \{0.000227438, 0.000323423, 0.000555578, 0.00102166, 0.001918, 0.0036221, 0.0068514, 0.0129641, 0.0245213, 0.0463056, 0.0864787, 0.159543, 0.272378, 0.38329\}$	257.19

Table B.21 Parameters for optimized $P = 14$ SRJ schemes for various values of N ($N = 2^k\ k = 5, \ldots, 12$).

N	Optimal Scheme Parameters	ρ
32	$\boldsymbol{\omega} = \{402.8, 318.533, 208.846, 120.929, 65.1531, 33.8031, 17.2446, 8.7701, 4.50164, 2.36779, 1.58821, 1.06413, 0.669773, 0.517215\}$	12.47
	$\boldsymbol{\beta} = \{0.00647086, 0.0073607, 0.00926188, 0.0124338, 0.0173072, 0.0245363, 0.035068, 0.0502174, 0.0717549, 0.104058, 0.0350767, 0.168979, 0.212666, 0.24481\}$	
64	$\boldsymbol{\omega} = \{1598.72, 1200.1, 724.027, 381.957, 187.598, 88.992, 41.5807, 19.3493, 9.04533, 4.29382, 2.29786, 1.25078, 0.716642, 0.521476\}$	23.51
	$\boldsymbol{\beta} = \{0.00338329, 0.0039658, 0.00523063, 0.00739484, 0.0108296, 0.0161218, 0.0241681, 0.0363079, 0.0545086, 0.0822675, 0.082198, 0.170812, 0.229322, 0.27349\}$	
128	$\boldsymbol{\omega} = \{6330.52, 4427.82, 2398.11, 1126.86, 494.844, 210.953, 88.8667, 37.3202, 15.7238, 6.70433, 3.03854, 1.43707, 0.760478, 0.525303\}$	43.75
	$\boldsymbol{\beta} = \{0.00176623, 0.00215768, 0.00302688, 0.00456605, 0.00711566, 0.0112388, 0.0178429, 0.0283674, 0.0450598, 0.0714887, 0.0994468, 0.170158, 0.241043, 0.296722\}$	
256	$\boldsymbol{\omega} = \{25031., 16170., 7803.54, 3256.38, 1277.48, 489.145, 185.638, 70.2999, 26.6852, 10.2232, 4.06315, 1.70266, 0.818836, 0.530181\}$	80.43
	$\boldsymbol{\beta} = \{0.00090345 7, 0.00115703, 0.00173491, 0.00279897, 0.00464739, 0.00779789, 0.0131313, 0.0221232, 0.0372734, 0.0626189, 0.100168, 0.168529, 0.253103, 0.324005\}$	
512	$\boldsymbol{\omega} = \{98853.4, 58634.4, 25154.6, 9337.93, 3281.64, 1131.19, 387.422, 132.494, 45.3869, 15.6547, 5.53583, 2.06513, 0.892377, 0.536009\}$	146.12
	$\boldsymbol{\beta} = \{0.00045224 43, 0.0006097 42, 0.0009791 98, 0.0016890 1, 0.0029859 8, 0.0053209 5, 0.0095049 3, 0.016988, 0.0303456, 0.0540876, 0.0943683, 0.164935, 0.26398, 0.353754\}$	
1024	$\boldsymbol{\omega} = \{389930., 211296., 80509.3, 26649.4, 8411.36, 2615.1, 809.329, 250.229, 77.4537, 24.0938, 7.63137, 2.54003, 0.980636, 0.542591\}$	262.34
	$\boldsymbol{\beta} = \{0.00022189 4, 0.0003161 55, 0.0005443 98, 0.0010033 6, 0.0018876 6, 0.0035721 5, 0.0067706 8, 0.012837 4, 0.0243311, 0.0460409, 0.086181, 0.159321, 0.272632, 0.38434\}$	
2048	$\boldsymbol{\omega} = \{1536240., 757099., 256177., 75783.7, 21526.3, 6043.96, 1691.56, 473.123, 132.43, 37.1993, 10.5906, 3.1509, 1.08412, 0.549805\}$	465.33
	$\boldsymbol{\beta} = \{0.00010686 7, 0.0001614 73, 0.0002983 41, 0.0005872 41, 0.0011754 6, 0.0023628 3, 0.0047545 3, 0.0095689 4, 0.0192541, 0.038699, 0.077294, 0.152063, 0.278729, 0.414945\}$	
4096	$\boldsymbol{\omega} = \{6045072, 2698661, 811234., 214918., 55022.7, 13963.2, 3535.64, 894.898, 226.617, 57.5285, 14.7527, 3.93026, 1.20411, 0.557585\}$	815.34
	$\boldsymbol{\beta} = \{0.00005057 14, 0.0000812954, 0.0001612 34, 0.0003388 26, 0.0007216 77, 0.0015417 4, 0.0032958 0, 0.0070462 4, 0.0150625, 0.0321750, 0.0684563, 0.143616, 0.282283, 0.445170\}$	

Table B.22 Parameters for optimized $P = 15$ SRJ schemes for various values of N ($N = 100 + 50k$, $k = 0, \ldots, 18$).

N	Optimal Scheme Parameters	ρ
100	$\boldsymbol{\omega} = \{3900.7, 2917.56, 1750.24, 917.535, 447.788, 211.033, 97.8831, 45.1243, 20.8015, 9.63811, 5.19063, 2.69523, 1.30463, 0.729676, 0.52263\}$ $\boldsymbol{\beta} = \{0.00211354, 0.00248261, 0.00328491, 0.00466027, 0.00684846, 0.0102307, 0.015395, 0.0232377, 0.0351381, 0.0539546, 0.0440768, 0.115928, 0.169266, 0.232834, 0.28055\}$	35.62
150	$\boldsymbol{\omega} = \{8731.9, 6299.82, 3574.97, 1764.89, 812.568, 362.251, 159.248, 69.6404, 30.449, 13.37, 6.43047, 3.027, 1.40229, 0.752546, 0.524622\}$ $\boldsymbol{\beta} = \{0.00143811, 0.00172513, 0.00235617, 0.00345693, 0.00524685, 0.00808252, 0.0125283, 0.0194675, 0.0302848, 0.0474906, 0.0542321, 0.11255, 0.169529, 0.238971, 0.29264\}$	51.31
200	$\boldsymbol{\omega} = \{15459.6, 10836.2, 5887.55, 2775.66, 1222.6, 522.574, 220.57, 92.6754, 38.9339, 16.42, 7.3844, 3.28605, 1.47625, 0.769405, 0.526065\}$ $\boldsymbol{\beta} = \{0.00109279, 0.0013333, 0.00186699, 0.0028111, 0.00437317, 0.00689647, 0.010935, 0.017374, 0.0276256, 0.0441322, 0.0577854, 0.110203, 0.169452, 0.243027, 0.301093\}$	66.29
250	$\boldsymbol{\omega} = \{24074.2, 16482.5, 8649.12, 3932.14, 1672.98, 691.895, 282.873, 115.18, 46.8951, 19.1619, 8.22785, 3.52027, 1.54157, 0.783992, 0.527297\}$ $\boldsymbol{\beta} = \{0.00088128, 0.00109027, 0.00155754, 0.00239368, 0.00379676, 0.006099, 0.00984518, 0.0159203, 0.0257581, 0.0418003, 0.0589403, 0.108288, 0.169245, 0.246258, 0.308126\}$	80.75
300	$\boldsymbol{\omega} = \{34568., 23204.9, 11831.2, 5221.34, 2159.63, 869.493, 346.353, 137.457, 54.5501, 21.7213, 9.00657, 3.73816, 1.60113, 0.797059, 0.528388\}$ $\boldsymbol{\beta} = \{0.000738052, 0.000923817, 0.00134184, 0.00209712, 0.00337956, 0.00551145, 0.00902828, 0.0148121, 0.0243112, 0.0399833, 0.0591017, 0.106635, 0.168961, 0.248953, 0.314222\}$	94.79
350	$\boldsymbol{\omega} = \{46934.7, 30975.9, 15411.5, 6633.06, 2679.17, 1054.58, 410.977, 159.617, 61.9913, 24.1524, 9.7385, 3.94279, 1.65611, 0.808933, 0.52937\}$ $\boldsymbol{\beta} = \{0.00063454, 0.000802316, 0.00118197, 0.00187368, 0.00306013, 0.00505457, 0.00838329, 0.0139235, 0.0231325, 0.0384872, 0.058814, 0.105172, 0.168631, 0.251248, 0.319602\}$	108.48
400	$\boldsymbol{\omega} = \{61168.7, 39772.5, 19371.5, 8159.09, 3228.87, 1246.47, 476.666, 181.708, 69.2673, 26.4843, 10.4333, 4.13611, 1.70724, 0.819823, 0.530261\}$ $\boldsymbol{\beta} = \{0.000556209, 0.000709559, 0.00105828, 0.00169833, 0.00280596, 0.0046861, 0.00785612, 0.0131873, 0.0221419, 0.0372151, 0.0583153, 0.103854, 0.168272, 0.253232, 0.324411\}$	121.87
450	$\boldsymbol{\omega} = \{77265.3, 49575., 23696., 9792.69, 3806.5, 1444.59, 543.333, 203.752, 76.408, 28.7354, 11.0971, 4.3196, 1.7551, 0.829887, 0.531078\}$ $\boldsymbol{\beta} = \{0.000494861, 0.000636339, 0.000959489, 0.00155652, 0.00259789, 0.0043809, 0.00741432, 0.0125628, 0.0212909, 0.0361099, 0.0577191, 0.102656, 0.167899, 0.25497, 0.328752\}$	135.00
500	$\boldsymbol{\omega} = \{95220., 60366.1, 28371.8, 11528.2, 4410.2, 1648.44, 610.9, 225.763, 83.4336, 30.9182, 11.7345, 4.49446, 1.80013, 0.839249, 0.531833\}$ $\boldsymbol{\beta} = \{0.000445513, 0.00057702, 0.000878606, 0.00143911, 0.00242379, 0.00412284, 0.00703687, 0.0120236, 0.0205478, 0.0351345, 0.0570832, 0.101557, 0.167517, 0.256507, 0.332706\}$	147.89
550	$\boldsymbol{\omega} = \{115029., 72130.4, 33387.7, 13360.8, 5038.38, 1857.63, 679.299, 247.749, 90.3589, 33.0419, 12.3487, 4.66169, 1.8427, 0.848005, 0.532533\}$ $\boldsymbol{\beta} = \{0.00040496, 0.000527957, 0.000811066, 0.0013401, 0.00227553, 0.00390105, 0.00670944, 0.0115513, 0.0198904, 0.034263, 0.0564379, 0.100541, 0.167134, 0.25788, 0.336333\}$	160.58
600	$\boldsymbol{\omega} = \{136689., 84854.1, 38733.6, 15286.4, 5689.71, 2071.79, 748.469, 269.715, 97.1953, 35.1137, 12.9426, 4.82212, 1.88311, 0.856234, 0.533187\}$ $\boldsymbol{\beta} = \{0.000371046, 0.000486682, 0.000753751, 0.00125532, 0.00214748, 0.00370786, 0.00642182, 0.0111329, 0.0193026, 0.0334767, 0.0557995, 0.099597, 0.166751, 0.259115, 0.339681\}$	173.08
650	$\boldsymbol{\omega} = \{160196., 98524.8, 44400.5, 17301.1, 6363., 2290.64, 818.358, 291.664, 103.952, 37.1391, 13.5182, 4.97644, 1.92159, 0.863999, 0.5338\}$ $\boldsymbol{\beta} = \{0.000342267, 0.000451462, 0.000704454, 0.00118179, 0.00203555, 0.00353769, 0.00616654, 0.0107586, 0.0187724, 0.0327614, 0.0551766, 0.0987156, 0.166373, 0.260234, 0.342789\}$	185.41
700	$\boldsymbol{\omega} = \{185548., 113131., 50380.6, 19401.8, 7057.24, 2513.9, 888.92, 313.598, 110.635, 39.1228, 14.0774, 5.12526, 1.95835, 0.871356, 0.534377\}$ $\boldsymbol{\beta} = \{0.000317542, 0.000421048, 0.000661567, 0.00111734, 0.00193672, 0.00338637, 0.00593793, 0.010421, 0.0182905, 0.0321064, 0.0545736, 0.0978891, 0.165999, 0.261253, 0.345689\}$	197.58
750	$\boldsymbol{\omega} = \{212741., 128662., 56666.5, 21585.6, 7771.5, 2741.35, 960.116, 335.519, 117.253, 41.0685, 14.6216, 5.26907, 1.99358, 0.878346, 0.534923\}$ $\boldsymbol{\beta} = \{0.000296072, 0.000394511, 0.000623891, 0.00106032, 0.0018487, 0.00325072, 0.00573167, 0.0101143, 0.0178497, 0.0315029, 0.0539927, 0.0971112, 0.165631, 0.262187, 0.348405\}$	209.60
800	$\boldsymbol{\omega} = \{241773., 145109., 63251.7, 23849.8, 8504.98, 2972.78, 1031.91, 357.429, 123.809, 42.9795, 15.1522, 5.40832, 2.0274, 0.88501, 0.53544\}$ $\boldsymbol{\beta} = \{0.000277257, 0.00037115, 0.000590511, 0.00100947, 0.00176971, 0.00312826, 0.00554433, 0.00983409, 0.0174443, 0.030944, 0.0534343, 0.0963765, 0.16527, 0.263047, 0.350959\}$	221.48
850	$\boldsymbol{\omega} = \{272642., 162462., 70130., 26192., 9256.94, 3208.01, 1104.27, 379.329, 130.308, 44.8586, 15.6702, 5.54338, 2.05996, 0.891377, 0.535932\}$ $\boldsymbol{\beta} = \{0.000260633, 0.000350423, 0.000560715, 0.000963807, 0.00169836, 0.00301701, 0.00537318, 0.0095766, 0.0170695, 0.0304242, 0.0528984, 0.0956807, 0.164917, 0.263841, 0.353369\}$	233.24
900	$\boldsymbol{\omega} = \{305344., 180712., 77296., 28610.2, 10026.7, 3446.87, 1177.17, 401.219, 136.753, 46.708, 16.1767, 5.67458, 2.09136, 0.897476, 0.536401\}$ $\boldsymbol{\beta} = \{0.000245842, 0.000331905, 0.000533945, 0.000922543, 0.00163353, 0.00291539, 0.00521603, 0.00933889, 0.0167216, 0.0299387, 0.0523842, 0.0950198, 0.16457, 0.264579, 0.355649\}$	244.87
950	$\boldsymbol{\omega} = \{339879., 199853., 84744.3, 31102.1, 10813.7, 3689.21, 1250.58, 423.1, 143.15, 48.5298, 16.6725, 5.80222, 2.12169, 0.903331, 0.53685\}$ $\boldsymbol{\beta} = \{0.000232596, 0.00031526, 0.000509752, 0.000885049, 0.00157432, 0.0028213, 0.00507108, 0.0091185, 0.0163973, 0.0294837, 0.0518908, 0.0943906, 0.16423, 0.265265, 0.357814\}$	256.39
1000	$\boldsymbol{\omega} = \{376243., 219876., 92470.3, 33666.1, 11617.4, 3934.9, 1324.47, 444.973, 149.499, 50.326, 17.1582, 5.92656, 2.15105, 0.908961, 0.537279\}$ $\boldsymbol{\beta} = \{0.000220668, 0.000300214, 0.000487773, 0.000850813, 0.00152, 0.00273615, 0.00493681, 0.0089134, 0.016094, 0.0290559, 0.0514173, 0.0937903, 0.163898, 0.265906, 0.359873\}$	267.81

Table B.23 Parameters for optimized $P = 15$ SRJ schemes for various values of N ($N = 2^k\ k = 6, \ldots, 11$).

N	Optimal Scheme Parameters	ρ
64	$\boldsymbol{\omega}$ = {1604.55, 1236.6, 777.72, 429.57, 220.699, 109.208, 53.1653, 25.7023, 12.4395, 6.06839, 3.77684, 2.26342, 1.17188, 0.697364, 0.519746}	23.72
	$\boldsymbol{\beta}$ = {0.00324844, 0.00375019, 0.00483085, 0.00665688, 0.00950942, 0.0138266, 0.0202681, 0.0298105, 0.043917 2, 0.0661899, 0.0257826, 0.120006, 0.167699, 0.222552, 0.261952}	
128	$\boldsymbol{\omega}$ = {6371.83, 4666.19, 2709.05, 1370.56, 646.134, 294.622, 132.367, 59.1371, 26.4159, 11.8529, 5.94244, 2.89717, 1.36449, 0.743778, 0.523863}	44.51
	$\boldsymbol{\beta}$ = {0.00167178, 0.00198802, 0.00268004, 0.00387851, 0.00580983, 0.00883865, 0.0135361, 0.0207864, 0.031965, 0.0496738, 0.0510383, 0.113836, 0.169489, 0.236713, 0.288096}	
256	$\boldsymbol{\omega}$ = {25234.4, 17233., 9009.26, 4080.06, 1729.54, 712.786, 290.428, 117.862, 47.8274, 19.4772, 8.32423, 3.5472, 1.54899, 0.785632, 0.527435}	82.46
	$\boldsymbol{\beta}$ = {0.00086125 3, 0.0010671, 0.00152771, 0.00235298, 0.00373991, 0.00601951, 0.0097354 1, 0.0157724, 0.0255664, 0.0415604, 0.0589968, 0.108078, 0.169214, 0.246607, 0.308902}	
512	$\boldsymbol{\omega}$ = {99805.2, 63101.3, 29545., 11959.4, 4558.78, 1698.18, 627.242, 231.042, 85.1043, 31.433, 11.8839, 4.53525, 1.81056, 0.841402, 0.532005}	150.96
	$\boldsymbol{\beta}$ = {0.00043557 3, 0.0005644 18, 0.0008613 16, 0.0014138 6, 0.0023860 9, 0.0040666 4, 0.0069541 7, 0.0119047, 0.0203829, 0.0349166, 0.0569284, 0.101306, 0.167425, 0.256851, 0.333604}	
1024	$\boldsymbol{\omega}$ = {394347., 229799., 96276., 34921.9, 12008.9, 4053.99, 1360.11, 455.47, 152.531, 51.1795, 17.388, 5.98513, 2.16481, 0.91159, 0.537479}	273.25
	$\boldsymbol{\beta}$ = {0.00021535 4, 0.0002934 94, 0.0004779 2, 0.0008354 1, 0.0014954 7, 0.0026972, 0.0048757 9, 0.0088198 6, 0.0159551, 0.0288594, 0.0511968, 0.0935117, 0.163741, 0.266199, 0.360827}	
2048	$\boldsymbol{\omega}$ = {1556575., 832736., 312142., 101721., 31639.4, 9698.49, 2959.69, 902.095, 274.961, 83.9203, 25.799, 8.03399, 2.62374, 0.99542, 0.543653}	489.09
	$\boldsymbol{\beta}$ = {0.00010464 3, 0.0001502 77, 0.0002612 82, 0.0004859 59, 0.0009220 98, 0.0017595, 0.0033623 8, 0.0064287 4, 0.0122909, 0.0234935, 0.0445452, 0.0851626, 0.158288, 0.273725, 0.38902}	

Appendix C

Von Neumann stability analysis

In oder to provide an example of the steps needed to compute the expressions for $\kappa_{\min}$ and $\kappa_{\max}$ in either SRJ schemes or in the CJM, we show in this appendix a prototype case, the discretization of the Laplace equation $\Delta u = 0$. In the following, we We consider a second order central discretization of the previous equation on a mesh with $N_x \times N_y$ nodes uniformly spaced, so that $\Delta x = L_x/N_x$ and $\Delta y = L_y/N_y$, L_x and L_y being the length of each of the dimensions. We obtain

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta y^2} = 0. \tag{C.1}$$

Isolating the value of $u_{i,j}$ one obtains the following iterative scheme:

$$u_{i,j}^{n+1} = \frac{1}{2\Delta x^2 + 2\Delta y^2}\left[\Delta y^2 u_{i-1,j}^n + \Delta y^2 u_{i+1,j}^n + \Delta x^2 u_{i,j-1}^n + \Delta x^2 u_{i,j+1}^n\right]. \tag{C.2}$$

Next we write the iteration step as a weighted Jacobi combination of the value of $u_{i,j}$ in the previous iteration ($u_{i,j}^n$) and the expression provided in Eq. (C.2) employing a weight, ω, to be specified later:

$$u_{i,j}^{n+1} = (1-\omega)u_{i,j}^n + \frac{\omega}{2\Delta x^2 + 2\Delta y^2}\left[\Delta y^2 u_{i-1,j}^n + \Delta y^2 u_{i+1,j}^n + \Delta x^2 u_{i,j-1}^n + \Delta x^2 u_{i,j+1}^n\right], \tag{C.3}$$

which, assuming $\Delta x = \Delta y$ (in order to simplify the algebra), yields

$$u_{i,j}^{n+1} = u_{i,j}^n + \frac{\omega}{4}(u_{i-1,j}^n + u_{i+1,j}^n + u_{i,j-1}^n + u_{i,j+1}^n - 4u_{i,j}^n). \tag{C.4}$$

Since the exact solution $u^n_{i,j}$ must satisfy the discretized equation exactly, the error $\epsilon^n_{i,j}$ must also satisfy the finite difference equation (C.4):

$$\epsilon^{n+1}_{i,j} = \epsilon^n_{i,j} + \frac{\omega}{4}(\epsilon^n_{i-1,j} + \epsilon^n_{i+1,j} + \epsilon^n_{i,j-1} + \epsilon^n_{i,j+1} - 4\epsilon^n_{i,j}). \tag{C.5}$$

We aim to compute the amplification of the error from one iteration step to the next one, i.e.:

$$G = \frac{\epsilon^{n+1}_{i,j}}{\epsilon^n_{i,j}}. \tag{C.6}$$

For that we expand the total error in a finite Fourier series

$$\epsilon(x,y) = \sum_{m=1}^{N_x}\sum_{n=1}^{N_y} A_{mn} e^{ik_m x} e^{ik_n y} \tag{C.7}$$

and replace those expressions in the iterative scheme. Writing explicitly the error terms appearing in Eq. (C.5) in terms of the elements of the finite Fourier basis into which we are expressing the total error we have

$$\begin{aligned}
\epsilon^n_{i,j} &= e^{ik_x x} e^{ik_y y} \\
\epsilon^n_{i-1,j} &= e^{ik_x(x-\Delta x)} e^{ik_y y} = e^{ik_x x} e^{ik_y y} e^{-ik_x \Delta x} \\
\epsilon^n_{i+1,j} &= e^{ik_x(x+\Delta x)} e^{ik_y y} = e^{ik_x x} e^{ik_y y} e^{ik_x \Delta x} \\
\epsilon^n_{i,j-1} &= e^{ik_x x} e^{ik_y(y-\Delta y)} = e^{ik_x x} e^{ik_y y} e^{-ik_x \Delta y} \\
\epsilon^n_{i,j+1} &= e^{ik_x x} e^{ik_y(y-\Delta y)} = e^{ik_x x} e^{ik_y y} e^{ik_x \Delta y}
\end{aligned} \tag{C.8}$$

After replacing the previous expressions in Eq. (C.5), we obtain

$$G = 1 + \frac{\omega}{4}(e^{-ik_x\Delta x} + e^{ik_x\Delta x} + e^{-ik_y\Delta y} + e^{ik_y\Delta y} - 4) \tag{C.9}$$

To compute the values of $\kappa_{\min}$ and $\kappa_{\max}$ we shall write the amplification factor (Eq. (**??**)) in the form

$$G = 1 - \omega\kappa(k_x, k_y). \tag{C.10}$$

After some algebra we get:

$$G = 1 - \omega\left[1 - \frac{1}{2}\frac{e^{-ik_x\Delta x} + e^{ik_x\Delta x}}{2} - \frac{1}{2}\frac{e^{-ik_y\Delta y} + e^{ik_y\Delta y}}{2}\right]. \tag{C.11}$$

Rewriting the expressions using trigonometric functions and identifying the $\kappa(k_x, k_y)$ comparing Eqs. (C.10) and (C.11), we have:

$$\kappa(k_x, k_y) = \frac{2 - \cos k_x\Delta x - \cos k_y\Delta y}{2}, \tag{C.12}$$

or equivalently:

$$\kappa(k_x, k_y) = \sin^2 \frac{k_x \Delta x}{2} + \sin^2 \frac{k_y \Delta y}{2}. \tag{C.13}$$

This generic expression of κ must be bounded accounting for the types of boundary conditions we have in our grid. For instance, if we deal with Dirichlet boundaries we cannot have one-dimensional Fourier modes and we shall obtain the bounds of Eq. (C.13) looking for the sum of each of the individual maxima obtained for purely one-dimensional problems in both the x and y dimensions. For Neumann boundaries one dimensional Fourier modes are allowed, and bounding Eq. (C.13) is much simpler. Restricting the analysis to this later case, we express the wave numbers $k_x = \pi m / L_x$ with $m = 1, 2, \ldots, N_x$ and $k_y = \pi r / L_y$ with $r = 1, 2, \ldots, N_y$. It is evident that the maximum of Eq. (C.13) holds for $\sin^2 \frac{k_x \Delta x}{2} = \sin^2 \frac{k_y \Delta y}{2} = 1$, yielding $\kappa_{\max} = 2$. On the other hand, the smallest possible value Eq. (C.13) is $\kappa_{\min} = \sin^2 \frac{\pi}{2N}$, where $N = \max\{N_x, N_y\}$.

Appendix D

Resum

D.1 Objectius

Les ePDEs (elliptic partial differential equations, en anglés) apareixen en una àmplia varietat d'àrees de les matemàtiques, la física i l'enginyeria. Són de particular interés en Astrofísica on apareixen, per exemple, quan es calcula el potencial gravitacional, en la solució de l'equació de Grad-Shafranov per magnetosferes lliures de forces, o d'imposar lligadures de divergència zero en la integració numèrica de les equacions MHD (magnetohydrodynamics, en anglés). En general, les ePDEs s'han de resoldre numèricament, establint una demanda cada vegada més gran d'algoritmes eficients i altament paral·lels per abordar la seua resolució computacional.

El SRJ (scheduled relaxation Jacobi, en anglés) pertany a una prometedora classe de mètodes, atípic per la combinació de senzillesa i eficàcia, que s'ha introduït recentment per resoldre ePDEs lineals de tipus Poisson. És una extensió del mètode iteratiu clàssic de Jacobi utilitzat per resoldre sistemes d'equacions lineals del tipus $Au = b$. Hereta, d'entre altres, la seua robustesa. La seua metodologia es basa en el càlcul d'uns paràmetres apropiats per a una aproximació multinivell amb l'objectiu de minimitzar el nombre d'iteracions necessàries per a reduir el residual per davall d'una tolerància especificada.

L'eficiència en la reducció del residual augmenta amb el nombre de nivells emprats en l'algoritme. Tanmateix, l'aplicació de la metodologia original per calcular els paràmetres d'estos esquemes SRJ òptims més enllà de 5 nivells és enormement dificultosa. Això és degut majoritàriament a la presència d'un sistema mixt algebraic-diferencial (no lineal) d'equacions el qual es torna cada vegada més rígid a mesura que augmenta el nombre de nivells.

Dit això, el que volem fer és:

- D'una banda, volem trobar una nova metodologia per a l'obtenció dels paràmetres dels esquemes òptims de l'algoritme SRJ que supere les limitacions de la metodologia original i proporcionar els paràmetres per a estos esquemes amb un nombre elevat de nivells, fóra bo fins a 15, i per a resolucions de fins a 2^{15} punts per dimensió. Això donarà lloc a factors d'acceleració de diversos centenars respecte del mètode de Jacobi en el cas de resolucions típiques i de milers en alguns casos amb altes resolucions. La major part de l'èxit en la recerca d'estos esquemes òptims amb més de 10 nivells es basarà en una reducció analítica de la complexitat del sistema d'equacions abans esmentat. A més, fóra bo estendre l'algoritme original per aplicar-lo a certs sistemes d'equacions el·líptiques no lineals.

- D'altra banda, en un esquema típic SRJ, s'empra l'anterior conjunt de paràmetres en cicles de M iteracions consecutives fins que s'arriba a la tolerància prescrita. Volem presentar la forma analítica del conjunt òptim de factors de relaxació per al cas en què tots ells són estrictament diferents, i veure que l'algoritme resultant és equivalent al mètode no estacionari de Richardson generalitzat, en el que es precondiciona la matriu del sistema d'equacions multiplicant per $D = \mathrm{diag(A)}$. El nostre mètode per estimar els pesos té l'avantatge que el càlcul explícit dels valors propis mínim i màxim de la matriu A (o la matriu d'iteració corresponent de l'esquema de Jacobi amb pes subjacent) es substituïx pel càlcul (molt més fàcil) de les freqüències mínima i màxima derivades de l'anàlisi d'estabilitat de von Neumann de l'operador el·líptic continu. Este conjunt de pesos també és l'òptim per al problema general, la qual cosa ens dóna la convergència més ràpida de tots els possibles esquemes SRJ per una estructura de malla donada. Ens referirem a ell com el mètode de Chebyshev-Jacobi. El factor d'amplificació del mètode es pot trobar analíticament i permet l'estimació exacta del nombre d'iteracions necessàries per a assolir la tolerància desitjada. També es vol mostrar que a partir del conjunt de pesos calculats per l'esquema SRJ òptim per a una mida de cicle fix és possible calcular numèricament el valor òptim del paràmetre ω del mètode SOR (successive overrelaxation, en anglés) en alguns casos.

- Volem demostrar amb exemples pràctics, d'aplicació en Astrofísica, que el nostre mètode també funciona molt bé per als problemes de tipus Poisson en els que es fa servir una discretització d'alt ordre de l'operador Laplacià (per exemple, discretitzacions de 9− o 17− punts). Això té molt d'interés,

ja que estes discretitzacions no produïxen matrius CO (consistently ordered, en anglés) i, per tant, la teoria de Young no es pot utilitzar per calcular el valor òptim del paràmetre de relaxació òptim de SOR. D'altra banda, els esquemes SRJ òptims deduïts ací són avantatjoses respecte a les implementacions existents per SOR pel que fa a discretitzacions d'alt ordre de l'operador Laplacià en la mesura que no cal recórrer als esquemes multicolor per a la seua execució en paral·lel.

- Presentar el mètode de Chebyshev-Jacobi fent servir una implementació purament MPI i una implementació híbrida OpenMP/MPI, ambdues sobre màquines de memòria compartida i de memòria distribuïda. Mostrar el seu rendiment i com escalen. També mostrar com arribar a velocitats de convergència notables amb execucions en paral·lel sobre GPUs quan la resolució d'equacions en derivades parcials el·líptiques amb diferències finites es fa utilitzant de manera conjunta el mètode de Chebyshev-Jacobi i les discretitzacions d'alt ordre.

- Finalment, tractar d'aplicar els nostres mètodes més enllà de l'àmbit de l'Astrofísica. En particular, abordar el problema de trobar els modes normals de vibració de l'ull humà. Este problema es pot resoldre amb una variant millorada de la metodologia que ací es presenta. La millora consistix a estendre el càlcul del conjunt òptim de paràmetres al cas de matrius no definides positives. Les nostres idees sobre com procedir en este camp s'esbossen en el treball futur d'esta tesi.

D.2 Metodologia

D.2.1 Problemes el·líptics en astrofísica

La realitat és canvi. Canvis en el temps, i canvis a l'espai. El concepte de derivada apareix en este context, com la taxa de variació d'una variable respecte d'una altra. És per això que les PDEs (partial differential equations, en anglés) apareixen per tot arreu en la física.

Des d'un punt de vista pràctic, ens preguntem per què necessitem resoldre PDEs. Com s'il·lustra en l'excel·lent monografia de Otway [2015] una solució d'una PDE pot reproduir, en general, tres tipus diferents de comportaments: pot propagar-se com si fos un paquet d'ones, pot difondre's com ho fa la calor, o pot oscil·lar sense anar enlloc. I resulta que molts fenòmens físics són combinacions d'estos tres comportaments. Per tant, resoldre PDEs pot servir

per a modelar matemàticament la naturalesa local dels fenòmens físics. Les solucions dels diferents tipus de PDEs, és a dir, les solucions de les equacions de tipus hiperbòlic, de tipus parabòlic i de tipus el·líptic, estan relacionades amb un dels tres comportaments anteriors, o siga, estan relacionades amb fenòmens de propagació, de difusió, o d'oscil·lació. Amb tot això, podem dir, i amb raó, que les PDEs són el llenguatge matemàtic apropiat per a modelar molts fenòmens. Entre les moltes PDEs famoses i interessants, sols en el cas d'equacions diferencials parcials lineals, tenim un munt d'exemples: (veure per exemple Evans [2010] o Salsa [2016]: l'equació de Laplace, l'equació de Helmholtz o dels valors propis, l'equació de transport lineal, l'equació de Louville, l'equació de la calor o de difusió, l'equació de Schrödinger, l'equació de Kolmogorov, l'equació de Fokker-Planck, l'equació d'ona, l'equació del telègraf, l'equació d'ona generalitzada, l'equació d'Airy, l'equació de Beam, etc. I no hem dit res de les PDEs no lineals ni dels sistemes de PDEs lineals o no lineals. Remetem el lector a Zwillinger [1997] per a una llista més extensa i un estudi més profund del tema. La investigació es centra en tipus concrets de PDEs, ja que no existix una teoria general sobre la resolució de qualsevol tipus d'estes. En la present tesi ens centrem en la resolució numèrica, de manera eficient, d'un tipus particular de PDEs, les ePDEs. A més, o farem dins d'una àrea específica del coneixement: l'Astrofísica.

En este capítol es revisen alguns conceptes matemàtics bàsics relacionats amb les ePDEs i proporcionem alguns dels exemples específics dins l'àrea de l'Astrofísica que tractarem d'alguna manera en esta tesi o en futures aplicacions dels mètodes desenvolupats al llarg de la mateixa. En el següent capítol ens centrarem en les diferents estratègies de solució per resoldre numèricament les ePDEs.

D.2.2 Introducció als mètodes iteratius

En este capítol de la tesi es fa una descripció bàsica de les estratègies per a la resolució numèrica de sistemes d'equacions en derivades parcials. El nostre objectiu no és fer una revisió exhaustiva de les diferents metodologies de resolució, sinó més aviat presentar el necessari per a posar en el context correcte els nous mètodes que hem desenvolupat. Com veurem més endavant, hi ha dues famílies de mètodes que es poden emprar: mètodes directes i mètodes iteratius. Ens centrarem en els mètodes iteratius, que és l'àmbit on estan els mètodes que s'han desenvolupat.

D.2.3 El mètode SRJ: millores i aplicacions

Com ja hem dit amb anterioritat, el nostre objectiu és resoldre PDEs associades amb problemes el·líptics amb interés en astrofísica. Ens centrem en els mètodes iteratius. Un dels esquemes iteratius més simples i més estudiats és el mètode de Jacobi [Jacobi 1845, Richardson 1911]. El principal inconvenient que presenta és la seua baixa velocitat de convergència. Per tal de millorar l'eficiència del mètode de Jacobi, s'han considerat moltes alternatives. Una possibilitat molt popular és l'ús de precondicionadors aplicats als sistemes lineals, que fan que els mètodes associats de Jacobi i Gauss-Seidel convergisquen de manera més ràpida asimptòticament que quan no estan precondicionats. De fet, el mètode que millorem ací, podrà ser emprat també com un precondicionador per altres mètodes, com per exemple, el mètode del gradient conjugat; o també del mètode multigrid, molt estés en l'actualitat i que pot trobar la solució en ordre $O(n)$. Els algoritmes de relaxació milloren el rendiment del mètode de Jacobi considerant modificacions de l'algorisme de Gauss-Seidel incloent un pes, com per exemple es fa en el mètode SOR [Young 1954a].

Seguint esta direcció, YM14 han presentat recentment el mètode SRJ, el qual presenta una acceleració considerable (de l'ordre de 100) sobre l'algoritme de Jacobi. El mètode SRJ és una generalització del mètode de Jacobi amb pes, el qual incorpora un factor de sobrerelaxació al Jacobi clàssic d'una manera similar a com ho fa al mètode SOR. Esta generalització inclou un nombre P de nivells diferents, en cada un dels quals, el paràmetre de sobrerelaxació (o subrelaxació) està calculat per tal d'aconseguir una reducció significativa del nombre d'iteracions, la qual cosa conduïx a una taxa de convergència molt més ràpida. El conjunt òptim de pesos depén de la discretització particular del problema en qüestió. Tot i que el mètode millora en gran mesura la taxa de convergència pel que fa al Jacobi, els esquemes presentats per YM14 arriben fins $P = 5$ i a resolucions de fins a 512 punts per dimensió espacial, de manera que encara no són competitius amb altres mètodes utilitzats actualment en problemes reals d'astrofísica (per exemple, mètodes espectrals, o mètodes multigrid com s'ha comentat més amunt). L'avantatge principal del mètode SRJ sobre altres alternatives per resoldre ePDEs numèricament és la seua simplicitat i la seua paral·lelització directa, ja que els mètodes SRJ preserven la insensibilitat del mètode de Jacobi a la descomposició en dominis (en contrast, per exemple, als mètodes multigrid).

D.2.4 Equivalència entre el SRJ i el mètode de Richardson no estacionari

El mètode SRJ pot expressar-se, per a un sistema lineal $Au = b$, com

$$u^{n+1} = u^n + \omega_n D^{-1}(b - Au^n),$$

on D és la diagonal de la matriu A. Si considerem un conjunt de P factors de relaxació diferents, $\omega_n,\ n = 1, \ldots, P$, de tal manera que $\omega_n > \omega_{n+1}$ i on cada factor de relaxació s'aplica q_n vegades, el *factor d'amplificació total* després de $M := \sum_{n=1}^{P} q_n$ iteracions és

$$G_M(\kappa) = \prod_{n=1}^{P} (1 - \omega_n \kappa)^{q_n},$$

que és una estimació de la reducció del residual durant un cicle (M iteracions). Com hem mostrat en el capítol anterior, l'expressió anterior de κ és una funció dels números d'ona obtinguda a partir d'una anàlisi d'estabilitat de von Neumann del sistema d'equacions lineals que resulten de la discretització del problema el·líptic original utilitzant diferències finites (per a més detalls veure YM14). YM14 van argumentar que, per a un nombre P fix de diferents pesos, hi ha una elecció òptima dels pesos ω_n i números de repetició q_n, la quantitat de vegades que s'utilitza cada factor ω_n, que minimitza el factor d'amplificació màxim per iteració, $\Gamma_M(\kappa) = |G_M(\kappa)|^{1/M}$, en l'interval $\kappa \in [\kappa_{\min}, \kappa_{\max}]$ i per tant també el nombre d'iteracions necessàries per a la convergència. Els límits de l'interval en κ corresponen al mínim i màxim número d'ona permesos per la discretització de la malla i per les condicions de contorn utilitzades per resoldre el problema el·líptic que tenim entre mans. En YM14 es calculen numèricament els pesos òptims per a $P \leq 5$ i s'han estés fins $P = 15$ (vegeu el capítol anterior i ACCA15).

Les principals propietats del SRJ, obtingudes per YM14 i confirmades per nosaltres, són les següents:

1. Dins del rang de valors de P estudiats, augmentant el nombre de pesos P millora la velocitat de convergència.

2. Els esquemes SRJ resultants convergixen significativament més ràpid que el mètode de Jacobi clàssic en un factor més gran que 100 en els mètodes presentats per YM14 i ~ 1000 en els nous esquemes presentats per nosaltres en el capítol anterior. Augmentar el tamany de la malla, que equival a disminuir el valor de $\kappa_{\min}$, porta a factors d'acceleració més grans.

3. Els esquemes òptims trobats utilitzen cada un dels pesos diverses vegades, resultant en un nombre total d'iteracions M per cicle significativament més gran que P, per exemple per $P = 2$, YM14 troba un esquema òptim amb $M = 16$ per la mida de malla més xicotet considerat ($N = 16$), mentre que per a les malles grans M augmenta considerablement (per exemple, $M = 1173$ per $N = 1024$).

El procediment d'optimització indicat per YM14 té un problema, però. Encara que el factor d'amplificació es reduís de forma monòtona mitjançant l'augment de P, per a valors prou alts de P el nombre d'iteracions per cicle M pot ser comparable amb el nombre total d'iteracions necessaries per a resoldre un problema particular fins a una tolerància prescrita. En este punt, utilitzant un mètode amb més P, i per tant major M, augmentaria el nombre d'iteracions per convergir, fins i tot si el $\Gamma(\kappa)$ és nominalment més petit. Amb esta limitació en el cap descrivim un procediment per obtenir esquemes SRJ òptims, minimitzant el nombre total d'iteracions necessari per a reduir el residual una quantitat suficient per a arribar a convergir o, equivalentment, per reduir al mínim $|G_M(\kappa)|$. De fet, el nombre total d'iteracions pot ser triat perquè siga igual a M sense pèrdua de generalitat, és a dir, que calga un cicle de M iteracions complet per assolir la convergència. Per seguir este procediment s'ha de trobar l'esquema òptim per a valors fixos de M, i després triar M tal que el valor màxim de $|G_M(\kappa)|$ siga similar a la reducció del residual necessari per resoldre un problema particular. El primer pas, el problema de minimització, és difícil de resoldre en general, ja que en fixar M tenim una enorme llibertat en l'elecció del nombre de pesos P, que pot oscil·lar entre 1 a M. No obstant això, els resultats numèrics obtinguts fins ara semblen suggerir que, en general, l'augment del nombre de pesos P sempre conduirà a una millor taxa de convergència. Això ens porta a conjecturar que l'esquema SRJ òptim, per a una M fixa, és el que té $P = M$, és a dir, tots els pesos són diferents i cada pes s'utilitza sols una vegada per cicle, $q_i = 1$, $(i = 1, \ldots, M)$. En termes del factor d'amplificació total $G_M(\kappa)$, és bastant raonable pensar que si es maximitza el nombre de diferents arrels triant $P = M$, la funció resultant és, de mitjana, més a prop de zero que en els mètodes amb menor nombre d'arrels, $P < M$. Per tant, es podria esperar màxims més xicotet per al conjunt òptim de coeficients. Un dels objectius d'este treball és calcular els coeficients per a este cas particular i demostrar que $P = M$ és, de fet, el cas òptim. Li direm al mètode CJM (Chebyshev-Jacobi method, en anglés).

Un altre objectiu d'esta secció és mostrar el funcionament del SRJ en comparació amb l'algoritme SOR òptim aplicat a un nombre diferent de discretitzacions

de l'operador Laplacià de dues dimensions (2D). Demostrarem que els mètodes òptims SRJ aplicats a discretitzacions d'alt ordre del Laplacià, per matrius d'iteració CO, funcionen de manera molt similar als esquemes òptims SOR (quan el pes òptim de SOR pot ser calculat). Discutim, a més, que la paral·lelització trivial dels mètodes SRJ equilibra lleugerament el millor rendiment de SOR en alguns casos. A més, demostrarem que el pes òptim del mètode SOR pot ser calculat de manera apropiada com una funció de la mitjana geomètrica del conjunt de pesos obtinguts per als esquemes de SRJ òptims. Això és de particular rellevància quan la matriu d'iteració és NCO (non-consistently ordered, en anglés), i per tant el càlcul analític del pes òptim de SOR és extremadament complex.

D.2.5 Aplicacions seqüencials en astrofísica

En este capítol es presenten els resultats numèrics dels mètodes descrits en els capítols anteriors aplicats a la resolució d'alguns problemes d'interés en astrofísica. Ens limitem ací a una aplicació seqüencial dels mètodes. Els resultats d'implementacions en paral·lel es presenten en el capítol següent.

Per una banda, es resolen l'equació de Poisson en coordenades esfèriques:

$$\frac{\partial^2 u}{\partial r^2} + \frac{2}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} + \frac{\cot\theta}{r^2}\frac{\partial u}{\partial \theta} + \frac{1}{r^2 \sin\theta}\frac{\partial^2 u}{\partial \varphi^2} = s, \tag{D.1}$$

essent u i s funcions de (r, θ, φ). Per als nostres tests, escollim que el terme font siga

$$s(r,\theta,\varphi) = \begin{cases} -\displaystyle\sum_{n=0}^{n_{\max}} \sum_{m=-m_{\max}}^{m_{\max}} a_{2n}\, k_{2n}^2\, j_{2n}(k_{2n} r)\, Y_{2n}^{m,c}(\theta,\varphi), & \text{per } r \le 1 \\ 0, & \text{per } r > 1, \end{cases} \tag{D.2}$$

essent j_l les funcions de Bessel esfèriques de primer tipus i $Y_l^{m,c}$ la part real dels harmònics esfèrics. Sols considerem els termes parells, $l = 2n$. k_l és la primera arrel de la funció de Bessel esfèrica d'ordre l, de manera que $s(1,\theta,\varphi) = 0$. Escollim $a_l = 1/2^l$, de manera que la sèrie siga convergent. Imposem condicions de frontera homogènies de tipus Neumann en $r = 0$, $\theta = 0$ i $\theta = \pi$, i condicions de frontera periòdiques en la direcció φ. Si imposem condicions de frontera homogènies de tipus Dirichlet en l'infinit radial ($r \to \infty$), la solució a este

problema el·líptic és

$$u(r,\theta,\varphi)=\begin{cases}\displaystyle\sum_{n=0}^{n_{\max}}\sum_{m=-m_{\max}}^{m_{\max}}\left(a_{2n}\,j_{2n}(k_{2n}r)+b_{2n}r^{2n}\right)\,Y_{2n}^{m,c}(\theta,\varphi), & \text{per } r\le 1\\ \displaystyle\sum_{n=0}^{n_{\max}}\sum_{m=-m_{\max}}^{m_{\max}}\frac{c_{2n}}{r^{2n+1}}\,Y_{2n}^{m,c}(\theta,\varphi), & \text{per } r>1,\end{cases} \tag{D.3}$$

on els coeficients c_l i b_l poden ser calculats imposant continuïtat de u i de les seues primeres derivades en $r=1$, resultant

$$b_l=c_l=-\frac{a_l}{2l+1}\partial_r j_l(k_l r)|_{r=1}=-\frac{a_l}{2l+1}\left[l\,j_l(k_l)-k_l\,j_{l+1}(k_l)\right] \tag{D.4}$$

Com que ens interessa la part de rendiment, resolem l'equació de manera numèrica en el domini $r\in[0,1]$, $\theta\in[0,\pi]$ and $\varphi\in[0,2\pi]$.

Per una altra banda, també resolem l'equació de Grad-Safranov. Esta equació descriu solucions d'equilibri en magnetohidrodinàmica ideal en plasmes bidimensionals. Té interés en l'estudi de plasma de fusió confinat magnèticament (per exemple Tokamaks), de la corona solar i magnetoesferes d'estrelles de neutrons, entre d'altres.

En coordenades esfèriques (r,θ,φ) el camp magnètic d'una configuració de plasma axisimètrica ($\partial_\varphi=0$) pot expressar-se com:

$$\boldsymbol{B}(r,\theta)=\boldsymbol{\nabla}\times\boldsymbol{A}=\frac{1}{r\sin\theta}\boldsymbol{\nabla}\Psi(r,\theta)\times\hat{\boldsymbol{e}}_{\boldsymbol{\varphi}}+\frac{F(r,\theta)}{r\sin\theta}\hat{\boldsymbol{e}}_{\boldsymbol{\varphi}}, \tag{D.5}$$

on $\boldsymbol{A}$ és el potencial vector i $\hat{\boldsymbol{e}}_{\boldsymbol{\varphi}}$ és el vector unitari en la direcció φ. La funció flux, $\Psi\equiv r\sin\theta A_\varphi$, és constant al llarg de les línies de camp magnètic i és una mesura de la força del camp magnètic poloidal. La funció toroidal, $F\equiv B_\varphi r\sin\theta$, és una mesura de la força del camp toroidal. Utilitzant la llei d'Ampere, $\boldsymbol{J}=\boldsymbol{\nabla}\times\boldsymbol{B}$, essent $\boldsymbol{J}$ el corrent elèctric, la funció flux pot ser lligada a el corrent toroidal com

$$\Delta^*\Psi\equiv\partial_{rr}\Psi+\frac{1}{r^2}\partial_{\theta\theta}\Psi-\frac{\cot\theta}{r^2}\partial_\theta\Psi=-J_\varphi r\sin\theta, \tag{D.6}$$

on Δ^* és l'operador el·líptic GS. Per simplicitat considerarem ací el cas en el que la inèrcia del fluid pot ser menyspreada (dominat magnèticament). En este cas, si imposem un balanç de forces, $\boldsymbol{J}\times\boldsymbol{B}=0$, la funció toroidal depén de la funció flux, $F(\Psi)$. Com a resultat, Eq. (D.6) ens porta cap a l'equació GS:

$$\Delta^*\Psi=-F(\Psi)F'(\Psi). \tag{D.7}$$

El no menysprear la inèrcia del fluid porta a termes de pressió adicionals que ací no considerem. Una elecció popular per a la funció toroidal es $F(\Psi)=C\Psi$,

essent C una constant. En este cas l'equació GS queda

$$\Delta^* \Psi + C^2 \Psi = 0, \tag{D.8}$$

que és un problema el·líptic adequat per a ser resolt mitjançant el mètode SRJ i el CJM. L'equació (D.8) es pareix a l'equació diferencial de Helmholtz doncs conté un operador de tipus Laplacià i un terme lineal en Ψ. Així doncs, este test mostrarà l'habilitat de SRJ i de CJM per treballar amb operadors el·líptics més complicats. A més, l'utilitzarem per a demostrar l'habilitat dels mètodes iteratius per a treballar en condicions de frontera imposades en fronteres de forma arbitrària.

Calculem la solució de la Eq. (5.18) per a dos conjunts de condicions frontera, en el domini numèric $r \in [1, 10]$ i $\theta \in [0, \pi]$. En tots els casos imposem condicions tipus Dirichlet homogènies en $\theta = 0$ i $\theta = \pi$. En el *test A* imposem condicions de frontera tipus Dirichlet en $r = 1$ i $r = 10$ amb $\Psi = \sin^2\theta / r$. En el cas $C = 0$, la solució per a este test és un camp dipolar. A mesura que el valor de C augmenta, la solució resultant és un dipol retorçat.

En el *test B* resolem l'equació GS en part del domini, en l'interior de la regió definida per

$$\begin{aligned} & r < \left(4.5 \sin^2\theta + 2.5 \sin^2(2\theta)\right) \left(1 - 0.4\cos(3\theta) + 0.3\cos(5\theta) + 0.05\sin(25\theta)\right) \\ & \& \\ & (r\sin\theta - 4)^2 + (r\cos\theta - 1.6)^2 < 1. \end{aligned} \tag{D.9}$$

D.2.6 Aplicacions paral·leles en astrofísica

Tot i que s'esmenta diverses vegades al llarg de la tesi l'enorme potencial que sembla tenir el nostre mètode per operar en paral·lel, no hem entrat en detall en este aspecte de l'algoritme fins ara. En este capítol presentem l'aplicació del CJM fent servir una implementació purament MPI, unaimplementació híbrida OpenMP/MPI i l'ús de GPUs.

Es resol el problema del potencial per a progenitors de lRGBs (long gamma-ray bursts, en anglés). En este cas, els efectes gravitatoris comencen a ser rellevants i és per això que cal resoldre el fluid acoblat a un potencial gravitatori generat pel mateix fluid autogravitant. Este potencial gravitatori és necessari per a contrarestar els gradients de pressió, de manera especial en la superfície estel·lar, i d'eixa forma arribar a l'equilibri hidrostàtic. Hem inclòs en MRGENESIS el càlcul d'un potencial newtonià per a tractar l'autogravetat.

El sistema hiperbòlic de lleis conservatives que governen el moviment relativista del fluid és (en unitats naturals i coordenades esfèriques)

$$\frac{\partial}{\partial t}\boldsymbol{U} + \frac{1}{r^2}\frac{\partial}{\partial r}(r^2\boldsymbol{F}) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}\boldsymbol{G} = \boldsymbol{S}, \tag{D.10}$$

on $\boldsymbol{U}$, $\boldsymbol{F}$ i $\boldsymbol{G}$ són vectors de quantitats conservades i de fluxos en les direccions radial i angular definides de la següent manera:

$$\boldsymbol{U} = (D, S^r, S^\theta, \tau)^T, \tag{D.11}$$

$$\boldsymbol{F} = (Dv^r, S^r v^r + p, S^\theta v^r, (\tau + p)v^r)^T. \tag{D.12}$$

$$\boldsymbol{G} = (Dv^\theta, S^r v^\theta, S^\theta v^\theta + p, (\tau + p)v^\theta)^T. \tag{D.13}$$

Les quantitats conservades que acabem de nomenar són la densitat de massa relativista D, la densitat de moment $\boldsymbol{S} = (S^r, S^\theta)$, i la densitat d'energia τ, essent totes mesurades en un marc de referència eulerià. També la velocitat $\boldsymbol{v} = (v^r, rv^\theta)$ es mesura en este sistema. Estes quantitats conservades estan relacionades amb quantitats en el marc de referència comòbil, les variables primitives, que són la densitat de massa en repòs, ρ, la pressió p, i la quadrivelocitat u^μ ($\mu = 0, \ldots, 3$), mitjançant

$$D = \rho W, \tag{D.14}$$

$$\boldsymbol{S} = \rho h W^2 \boldsymbol{v}, \tag{D.15}$$

$$\tau = \rho h W^2 - p - D, \tag{D.16}$$

$$(v^r, v^\theta) = \frac{1}{W}(u^1, \frac{u^2}{r}), \tag{D.17}$$

on W és el factor de Lorentz, h és l'entalpia, i

$$W = u^0 = \frac{1}{\sqrt{1 - \boldsymbol{v}^2}} \tag{D.18}$$

$$h = 1 + \varepsilon + p/\rho \tag{D.19}$$

essent ε l'entalpia específica. Finalment, en absència de fonts físiques (per exemple la gravetat), el terme font $\boldsymbol{S}$ sols conté termes geomètrics a causa de les coordenades esfèriques bidimensionals:

$$\boldsymbol{S} = \begin{pmatrix} 0 \\ \frac{1}{r}\left(2p + S^\theta v^\theta\right) \\ \frac{1}{r}\left(\frac{\cos\theta}{\sin\theta}p - S^\theta v^r\right) \\ 0 \end{pmatrix}. \tag{D.20}$$

En la nostra implementació del fluid autogravitant, com a primera aproximació i encara que el codi siga relativista, hem escollit un potencial newtonià amb certes correccions relativistes (per exemple, utilitzem com a terme font de la Poisson l'equació $\rho_{\text{eff}} := \rho h W^2 - p$ en lloc de ρ). Com que mantenim la mètrica constant, la contribució d'este potencial sols apareix com un terme addicional $\boldsymbol{S}_{\text{pot}}$ en la font de la Eq. (D.10), o siga, $\boldsymbol{S}_{\text{new}} = \boldsymbol{S} + \boldsymbol{S}_{\text{pot}}$, on S fa referència a l'Eq. (D.20) i

$$\boldsymbol{S}_{\text{pot}} = \begin{pmatrix} 0 \\ -(\rho h W^2 - p)\dfrac{\partial}{\partial r}\Phi \\ -\dfrac{1}{r}(\rho h W^2 + \tilde{S}^r \tilde{v}^r)\dfrac{\partial}{\partial \theta}\Phi \\ -\rho h W^2 \left(\tilde{v}^r \dfrac{\partial}{\partial r}\Phi + \dfrac{\tilde{v}^\theta}{r}\dfrac{\partial}{\partial \theta}\Phi \right) \end{pmatrix}. \tag{D.21}$$

L'equació de Poisson $\Delta\Phi = 4\pi\rho_{\text{eff}}$ definix el comportament del potencial Φ i la seua dependència respecte de la distribució de massa. Resolem l'equació de Poisson en coordenades esfèriques utilitzant el CJM.

L'objectiu de les simulacions RHD (relativistic hydrodynamics, en anglés), que són instrumentals ací [1], no és estudiar en detall la maquinària central dels GRB, que s'exclou de forma explícita del nostre domini computacional, sinó l'estudi de la propagació dels dolls des de ben a fora del nucli estel·lar fins a la seua superfície. Amb això, escindim la regió més interna $r < R_0$ i modelem el motor del doll com una entrada d'energia i de moment en $r = R_0$. Els nostres dominis computacionals s'estenen radialment des d'una frontera interna en R_0 ($\gtrsim 10^9$ cm) fins R_f, típicament localitzada fora de l'estrella. Per imposar millor les condicions de frontera en la part més interna de la zona d'escissió, no resolem directament el potencial $\Phi(r,\theta)$ sinó un potencial modificat $\bar{\Phi}(r,\theta) = \Phi(r,\theta) + M_0/r$, on M_0 és la massa escindida per davall de R_0. Imposem condicions de frontera de tipus Neumann per al potencial en R_0, $\partial_r \bar{\Phi}|_{R_0} = 0$, i condicions de frontera de tipus Dirichlet en la part radial mes a fora de la malla, $\bar{\Phi}_{R_f} = -M_T/R_f$. La massa total M_T dins la malla no inclou la massa escindida M_0. Una vegada calculem $\bar{\Phi}$, restem en la direcció radial la quantitat M_0/r per recuperar el potencial real Φ.

De la mateixa manera que en l'apartat anterior, no sols necessitem calcular el potencial una sola vegada sinó que necessitem recalcular-lo al llarg de l'algoritme per a la integració en temps de la part hiperbòlica que governa les equacions d'evolució del fluid. En principi, seria necessari calcular el potencial gravitatori a

[1] Més detalls en el capítol 4 de Cuesta 2017

cada pas de temps de la part hiperbòlica. Tanmateix, en la pràctica en possible recalcular-lo amb menys freqüència per a aconseguir una precisió satisfactòria al llarg de tota la computació del model. Com a guia, recalculem el potencial gravitatori cada n_r passos de temps, essent n_r el nombre de cel·les en la direcció radial. Açò garantix que l'actualització té lloc dins del temps que tarda la llum a anar de part a part de la malla numèrica ($\sim n_r \Delta t$). Finalment, entre dos càlculs consecutius es fa un reajustament de la massa interna necessari a causa del flux de massa a través de R_0.

En el test que presentem ací, no considerem la simulació completa sinó els resultats de rendiment del nou mètode desenvolupat per un dels recàlculs del potencial gravitatori, baix condicions que són comparables a les que trobaríem en la simulació completa.

També calculem per trobar el camp estacionari d'una esfera carregada uniforme de radi R en coordenades cartesianes en 3D utilitzant condicions de frontera de tipus Dirichlet. Així doncs, resolem l'equació de Poisson:

$$\Delta\phi(x, y, z) = -4\pi\rho_{\mathrm{e}}, \tag{D.22}$$

on $\rho_{\mathrm{e}} = 3Q/(4\pi R^3)$ i Q és la carrega de l'esfera.

D.2.7 Més enllà de l'astrofísica: modelant l'ull humà

Durant l'elaboració de la tesi ha sorgit un projecte paral·lel dedicat a l'aplicació de les equacions el·líptiques en una àrea molt diferent de l'astrofísica. Hem utilitzat els nostres esquemes numèrics en la modelització de les oscil·lacions de l'ull humà, amb aplicacions en optometria i biomecànica. Es construïx un model per obtenir les freqüències d'oscil·lació, s'investiga la dependència dels paràmetres del sistema i es presenten els nostres resultats de la col·laboració amb el Departament d'òptica de la Universitat de València de la realització de mesures en ulls humans que serviran per a millorar el nostre model en el futur.

L'ull és un òrgan complex que consta de diverses parts funcionals amb interaccions mútues. Les més importants són la còrnia, el cristal·lí, el vitri, l'escleròtica i la retina. Mesurar les propietats mecàniques de l'ull *in vivo* i amb tècniques mínimament invasives pot ser la clau per a solucions individualitzades a una sèrie de patologies oculars. Ací mostrem que estes propietats estan relacionades amb els modes de vibració normals del globus de l'ull, és a dir, a les variacions periòdiques de la matèria a l'interior del globus ocular resultant de pertorbacions pel que fa al seu estat d'equilibri.

Es modela el globus ocular com una bola sòlida elàstica esfèrica, homogènia i isòtropa amb simetria axial. Si bé el cas que el globus ocular té simetria

axial és justificat, els supòsits d'homogeneïtat i isotropia no són la millor opció. No obstant això, estos supòsits servixen amb l'objectiu principal de reduir la dependència de l'equació constitutiva només a dues constants elàstiques o mòduls del material de l'ull: el mòdul de Young E, i el coeficient de Poisson σ. En este marc simplificat, calcularem, en primer lloc analíticament i després numèricament, les freqüències pròpies del model per intentar comprendre els mecanismes essencials d'un ull humà mitjà.

D.3 Conclusions

Oferim a continuació una discussió individualitzada en detall dels resultats presentats tant en els dos capítols de la Part II com en els tres capítols de la Part III.

D.3.1 SRJ: millores i aplicacions

En este capítol, basant-nos en els resultats de YM14, hem ideat un nou mètode per a l'obtenció dels paràmetres òptims per als esquemes SRJ aplicats a la resolució numèrica d'ePDEs. El nou mètode reduïx la complexitat del sistema no lineal d'equacions a partir de la qual es calculen els paràmetres òptims.

Hem demostrat que els nous esquemes multinivell SRJ seguixen millorant la velocitat de convergència de l'esquema, el que significa que augmentant el valor de P obtenim factors d'acceleració cada vegada més grans respecte al mètode de Jacobi. Es presenten factors d'acceleració d'uns pocs centenars i, en alguns casos, de més de 1.000 respecte al mètode de Jacobi quan es considere tant un nombre prou gran de punts per dimensió (és a dir, $N > 16.000$) com de nivells. Per aplicacions multidimensionals, augmentant el nombre P de 5 (el valor màxim de nivells en YM14) fins a 15 produïx una disminució en el cost computacional d'un factor $\sim 2-3$ per a les resolucions més altes considerades ací. Encara que per altes resolucions obtenim resultats millors, observem que el benefici d'emprar algoritmes SRJ amb $P = 15$ serà molt avantatjós en aplicacions en tres dimensions funcionant en supercomputadors. En estos casos, cal utilitzar esquemes SRJ amb un major nombre de nivells que els proposats originalment en YM14, i més si tenim en compte que no hi ha complexitat addicional a la implementació de l'algoritme per a qualsevol $P \geq 2$ una vegada coneguts els pesos per a valors grans de P. Un dels avantatges dels algoritmes SRJ és que el conjunt òptim de coeficients per a un nombre donat de punts per dimensió, N_1, pot ser reutilitzat per a les malles computacionals amb un nombre més gran

de punts per dimensió, N_2 (YM14). El conjunt de paràmetres SRJ utilitzats d'esta manera no és òptim per a la malla amb el nombre més gran de punts per dimensió. No obstant això, l'aplicació del conjunt de paràmetres SRJ a la malla amb N_2 punts per dimensió accelerarà la solució iterativa respecte de l'esquema bàsic de Jacobi, fins i tot considerablement si N_2 no és molt més gran que N_1. Hem proporcionat un ampli conjunt de taules amb tots els coeficients òptims necessaris per a un conjunt dens de nombres diferents de punts per dimensió.

Degut principalment al fet que hem derivat solucions analítiques per a una part de les incògnites, el nostre nou mètode reduïx la rigidesa del sistema no lineal d'equacions d'on calculem els paràmetres òptims, i això ens permet d'obtenir nous mètodes SRJ per a un màxim de $P = 15$ i un nombre arbitràriament gran de punts per dimensió N.

A partir d'este nombre de nivells sorgixen nous problemes que dificulten el càlcul de coeficients òptims per a un nombre baix de punts en la discretització. Estos problemes estan relacionats amb el fet que per a valors grans de P la solució del problema és molt sensibles a petits canvis en els números d'ona més xicotets, i els xicotets errors numèrics eviten l'èxit a l'hora d'avaluar la solució fins i tot el sistema d'equacions no lineals que resulta de la simplificació algebraica que hem mostrat ací. Estos problemes són resolts amb el desenvolupament del mètode Chebyshev-Jacobi presentar a continuació.

D.3.2 Cap a l'equivalència entre el mètode SRJ i el mètode no estacionari de Richardson

En este capítol s'han obtingut els coeficients òptims per al mètode SRJ per resoldre sistemes lineals provinents de la discretització en diferències finites de problemes el·líptics en el cas $P = M$, és a dir, usant cada pes només una vegada per cicle. Anomenem al mètode resultant Chebyshev-Jacobi (CJM). Hem demostrat que estos són els coeficients òptims per al cas general, on fixem P però permetem repeticions dels coeficients ($P \leq M$). A més, hem proporcionat una manera simple d'estimar el valor òptim de M per reduir el residual inicial un factor prescrit.

Hem provat el rendiment del CJM amb un exemple senzill en 2 dimensions, el qual ens mostra que el factor d'amplificació derivat analíticament es pot obtenir a la pràctica. En comparar el rendiment del conjunt de coeficients òptims per a $P = M$ amb els de YM14 i els del capítol anterior, el CJM sempre dóna millors resultats, és a dir, s'aconseguix una major reducció del residual per al mateix nombre d'iteracions M. A més, els nous coeficients es poden calcular

analíticament, com una funció de M, $\kappa_{\min}$ i $\kappa_{\max}$, el que evita la resolució numèrica del problema de minimització necessari en el cas del SRJ. El resultat és un mètode numèric que és fàcil d'implementar, i on tots els coeficients necessaris són fàcilment calculables tenint en compte la mida de la malla, les condicions de contorn i la tolerància del problema el·líptic que volem resoldre abans que el procediment d'iteració haja començat.

Seguint la mateixa filosofia que va inspirar el desenvolupament de mètodes SRJ, hem trobat que el cas $P = M$ resultat en un mètode iteratiu gairebé equivalent al mètode de Richardson no estacionari tal com el va implementar Young [1953]. Més específicament, en la implementació de Young, els coeficients ω_n es prenen de manera que siguen els recíprocs de les arrels del corresponent polinomi de Chebyshev en l'interval que limita l'espectre de valors propis de la matriu (A) del sistema lineal. A més, inspirat en les mateixes idees que les originals del mètode SRJ, els valors propis mínims i màxims de A no necessiten ser explícitament calculats. En lloc d'això, es recorre a una anàlisi (molt més simple) d'estabilitat de von Neumann del sistema lineal, que dóna els valors de $\kappa_{\min}$ i $\kappa_{\max}$ que substituixen els valors propis mínim i màxims de la matriu A^2. La clau del nostre èxit en la implementació pràctica del CJM deriva d'un ordre adequat en la utilització dels pesos ω_n en l'algoritme. Encara que altres ordenacions també s'ha demostrat que funcionen, la nostra elecció limita clarament el creixement d'errors d'arrodoniment quan el nombre d'iteracions és gran. Esta ordenació s'hereta dels esquemes SRJ.

També hem provat el rendiment del CJM per a discretitzacions de més de segon ordre de l'operador Laplacià el·líptic. Estos casos són especialment complicats, ja que la matriu d'iteració no és CO. Per tant, la teoria de Young no pot ser emprada per tal de trobar el valor del pes òptim per a un esquema SOR que es vulga aplicar al problema resultant. Per al cas particular de la discretització de 9 punts del Laplacià, tot i que la matriu d'iteració no és CO, es van trobar el pes òptim per a l'esquema SOR corresponent utilitzant una derivació bastant complicada. En la comparació dels resultats per a la solució numèrica d'un problema simple de tipus Poisson, el mètode SOR derivat per Adams i el CJM obtingut per nosaltres, per a la mateixa discretització de 9 punts del Laplacià, tots dos mètodes es comporten de manera similar. (Encara que l'esquema SOR òptim és encara una poc millor). No obstant això, el mètode SOR requerix una estratègia de paral·lelització multicolor d'entre

[2]Observem que molts altres esquemes iteratius es basen en una selecció dinàmica del paràmetre de relaxació o en tècniques de precondicionament dinàmiques i poden ser aplicades a matrius que vénen de discretitzacions de problemes físics sobre malles més genèrics.

les 72 possibles amb quatre colors (cadascuna amb diferent rendiment), que s'aplica a la discretització de 9 punts de l'operador Laplacià. L'estratègia de paral·lelització encara és més complicada quan s'utilitza una discretització del Laplacià de 17 punts. En contrast, el CJM és trivialment paral·lelitzable i no requerixen cap estratègia multicolor. Per tant, arribem a la conclusió que el rendiment un poc millor del SOR respecte del CJM en aplicacions seqüencials desapareix pràcticament en les implementacions paral·leles del mètode. A més, també hem demostrat que la discretització d'ordre superior de l'operador de Laplace és molt avantatjosa tant en el nombre d'iteracions com en el temps computacional necessari per a assolir un error real preestablert (és a dir, el veritable error que trobem en comparar la solució exacta d'un problema amb la seua aproximació numèrica). Donat l'augment del nombre de punts necessaris per a implementar una discretització de 17 punts de l'operador de Laplace, inferim que una implementació paral·lela d'este mètode pot requerir un modest increment en el nombre de zones transferides en les fronteres internes entre els subdominis computacionals diferents. Amb tot, l'aplicació de discretitzacions d'alt ordre del Laplacià és ideal per als problemes que combinen la resolució dels sistemes acoblats el·líptics-hiperbòlics (per exemple, en el cas dels sistemes d'Euler-Poisson que modelen fluids autogravitants).

D.3.3 Aplicacions seqüencials en l'astrofísica

En este capítol hem aplicat el mètode SRJ i el CJM a la resolució d'alguns problemes d'interés en astrofísica. Ens limitem ací a una implementació seqüencial del mètode. Ja hem aconseguit arribar a factors d'acceleració que fan que els mètodes SRJ siguen competitius (en funció de la dimensionalitat del problema i de la seua grandària) amb, per exemple, mètodes espectrals per a la resolució d'algunes ePDEs. La comparació s'ha fet per a la resolució de l'equació de Poisson en coordenades esfèriques en 1D, 2D i 3D.

Trobem que per problemes 1D de tipus Poisson, el mètode més ràpid de resolució, dels provats, és el mètode d'inversió directa implementat a `LAPACK`. Això succeïx perquè, en aplicacions realistes en les quals l'equació de Poisson ha de ser resolta múltiples vegades, la descomposició LU de la matriu, on es troba la majoria del treball de còmput, es fa una sola vegada i es pot emmagatzemar per a la resta dels càlculs.

En 2D, el mètode que funciona millor depén de si el nostre valor inicial està a prop de la solució real o lluny d'ella. En aplicacions realistes, on les ePDEs s'acoblen a sistemes de PDEs hiperbòliques, la solució en la iteració de temps

anterior no canvia significativament durant el curs d'un sol pas de temps. En estes condicions, les llibreries `LAPACK` són les que donen un rendiment més alt. No obstant això, els mètodes espectrals són avantatjosos si, en 2D, els valors inicials estan lluny de la solució real del problema. Assenyalem a més que en sistemes acoblats realistes, i per un nombre relativament gran de punts per dimensió ($N > 500$), els mètodes SRJ són competitius amb els espectrals.

En aplicacions en 3D, ens trobem que el cost computacional total de mètodes SRJ escalen com N^4, és a dir, com en el cas dels mètodes espectrals. Tenint en compte que (i) l'aplicació de mètodes d'inversió directes a problemes en 3D pot ser inviable a causa de les restriccions de memòria, i que (ii) els mètodes SRJ poden ser paral·lelitzats molt ràpidament (molt més fàcilment que, per exemple, espectrals o mètodes multigrid), preveiem que són una alternativa competitiva per a la solució de problemes el·líptics en aplicacions de supercomputació en 3D.

Finalment, descrivim que la fàcil implementació de condicions de frontera molt complicades en SRJ i CJM és també un avantatge respecte a altres alternatives existents. Este fet s'ha demostrat a la resolució de l'equació de Grad-Shafranov en un domini molt intricat i incloent condicions de frontera mixtes. De fet, la versatilitat del CJM per fer front a fronteres arbitràries fa que siguen adequats per fer front a problemes reals en Astrofísica.

D.3.4 Aplicacions paral·leles en astrofísica

En els capítols anteriors hem delineat el potencial grau de paral·lelisme de CJM com un important avantatge sobre altres algoritmes que competixen en la resolució de sistemes d'equacions lineals resultants de la discretització dels sistemes el·líptiques d'equacions en derivades parcials. Basant-se en el mètode bàsic de Jacobi, la implementació paral·lela del CJM és tan senzilla com la del mètode anterior. Ara materialitzem les nostres reivindicacions anteriors i presentem una implementació del CJM sobre MPI únicament, una implementació tridimensional híbrida sobre OpenMP/MPI i una implementació utilitzant GPUs.

Hem provat l'algoritme de CJM sobre MPI portat a dues diferents arquitectures MIMD, una amb memòria distribuïda i l'altra amb memòria compartida. Hem comparat tant l'escalabilitat com el temps d'execució del CJM respecte de la inversió directa per resoldre un problema d'astrofísica en coordenades esfèriques. L'estratègia de paral·lelització ha estat dividir la malla en dominis i distribuir la càrrega de treball entre processadors. El problema es resol llavors iterativament en tota la malla i les fronteres dels dominis són comunicades en cada pas d'iteració. Hem vist, extrapolant, que per a les simulacions d'alta

resolució, a partir de 3000 processadors en amunt, el temps d'execució utilitzat pel CJM s'espera que siga menor que el mètode d'inversió directa tant en càlculs *ab initio* com en càlculs realistes. A més, trobem que el speedup del CJM és ideal, cosa que no passa amb la inversió directa. La utilització d'una inversió directa basada en el mètode LU és problemàtica a altes resolucions quan el nombre de particions són xicotets en alguna dimensió, perquè la matriu ja no cap a la memòria.

La solució de l'equació de Poisson per a esbrinar la distribució del potencial elèctric creat per una esfera amb càrrega uniforme s'ha utilitzat com un banc de proves en 3D. Hem aprofitat l'existència d'una plataforma per a relativitat numèrica. Hem utilitzat una implementació híbrida OpenMP/MPI del CJM. Les simulacions han mantingut un speedup ideal fins a 64 nuclis i amb $N = 256$ en 3D. Tant SOR com CJM són més d'un ordre de magnitud més ràpids que els mètodes de Jacobi i Gauss-Seidel. No obstant això, la fórmula per calcular el ω_{opt} per SOR només s'aplica quan la matriu original del sistema lineal és CO; a més, CJM es paral·lelitza trivialment, mentre SOR requerix esquemes multicolors per a una paral·lelització exitosa. El cas de simetria octant conduïx a una matriu NCO i, per tant, no hi ha una expressió analítica pel càlcul de ω_{opt} per a SOR. Hem provat, en este cas, una seqüència de valors de ω de manera empírica per estimar el seu valor òptim per al problema donat. En última instància, el CJM es comporta millor que SOR per al conjunt de valors provats de ω.

També hem provat les versions portades dels algoritmes Jacobi i CJM sobre CUDA en dues arquitectures diferents GPU. Les diferències en temps de càlcul real es reduïxen significativament, ja siga en augmentar la mida de la malla o el nombre de punts utilitzats en la discretització de l'operador de l'operador Laplacià. Ens trobem amb què és possible accelerar diversos ordres de magnitud el mètode clàssic de Jacobi gràcies a l'ús de les implementacions paral·leles de CJM sobre GPUs.

D'altra banda, hem il·lustrat els beneficis de l'ús de la implementació paral·lela del CJM sobre GPUs de manera combinada amb una discretització d'alt ordre de l'operador Laplacià utilitzant un problema de prova. Hem arribat a la conclusió que és sempre avantatjosa la utilització de discretitzacions d'alt ordre de l'operador el·líptic, ja que requerix menys iteracions i menys temps de càlcul per a aconseguir el mateix objectiu d'error real. Esta conclusió és independent de l'aplicació paral·lela de qualsevol dels mètodes que hem provat en este treball. No obstant això, la combinació de discretitzacions d'alt ordre

dels operadors el·líptics i la implementació del CJM sobre GPUs es traduïx en un mètode extremadament poderós per a aplicacions pràctiques.

D.3.5 Més enllà de l'astrofísica: modelant l'ull humà

Hem presentat una anàlisi dels modes normals d'un ull humà idealitzat. Amb esta finalitat, hem importat els resultats analítics desenvolupats en altres àrees de física, més precisament, en el camp de la física d'ones gravitacionals.

Hem demostrat que més enllà de la caracterització mecànica de les components del globus ocular, els modes de vibració normals de l'ull podrien estar involucrats en processos fisiològics com, per exemple, l'acomodació. Hem desenvolupat un model simplificat de globus ocular, amb simetria esfèrica, per al que hi ha solucions analítiques per a les freqüències pròpies amb condicions de contorn senzilles. Hem utilitzat estes solucions per calibrar el nostre nou codi numèric de diferències finites. A més, s'ha estudiat la dependència de les freqüències pel que fa a la longitud axial. Estos resultats han estat contrastats amb dades biomètriques mesurades realment a una mostra de pacients seleccionats amb un doble objectiu. D'una banda, calibrar el model amb dades reals i, d'altra banda, retroalimentar les dades sobre el model de globus ocular. L'últim objectiu està encara en curs.

Els modes de vibració normal de l'ull podrien estar implicats en alguns processos fisiològics, per exemple, l'acomodació. L'acomodació ocorre a través de canvis en la forma i gruix de la lent cristal·lina. El gruix i la curvatura de la lent augmenta, causant un augment de la potència òptica de l'ull. Atés que és una activitat induïda per múscul, l'acomodació és un procés altament fluctuant i dinàmic. Estes fluctuacions estan relacionades amb les fluctuacions de les aberracions oculars, i es produeixen a unes determinades freqüències. Les microfluctuacions d'acomodació poden tenir un paper important en la variabilitat de la qualitat òptica de l'ull. Hi ha dos components principals de la resposta acomodativa: una component de baixa freqüència ($< 0,5$ Hz), que correspon a la deriva en la resposta d'allotjament, i un pic a major freqüència, en la banda $1 - 2$ Hz. Els modes de vibració del globus ocular que nosaltres hem considerat ocórren a escales de temps d'uns pocs mil·lisegons. La manera exacta en què els modes normals de vibració del globus ocular es correlacionen amb el procés d'acomodació està més enllà de l'abast d'aquesta tesi. No obstant això, preveiem que per fer front a este estudi es necessari millorar el nostre model actual, almenys, la diferenciació en el model de globus ocular de la còrnia-escleròtica,

del humor vitri i de la lent. En aquesta direcció realitzarem la nostra futura recerca.

Bibliography

[1] L. M. Adams, R.-J. LeVeque, and D. Young. "Analysis of the SOR Iteration for the 9-Point Laplacian." In: *SIAM Journal on Numerical Analysis* 25.5 (1988), pp. 1156–1180. DOI: 10.1137/0725066. URL: http://dx.doi.org/10.1137/0725066.

[2] J. E. Adsuara et al. "Improvements in the Scheduled Relaxation Jacobi method." In: *Proceedings of the XXIV Congress on Differential Equations and Applications / XIV Congress on Applied Mathematics Cádiz, June 8-12, 2015.* Ed. by C. Díaz Moreno, J. M. and Díaz Moreno, J. C. and García Vázquez, C. and Medina Moreno, J. and Ortegón Gallego, F. Pérez Martínez, M. V. Redondo Neble, and J. R. Rodríguez Galván. Servicio de Publicaciones de la Universidad de Cádiz. 2015. ISBN: 978-84-9828-527-7.

[3] J. Adsuara et al. "Scheduled Relaxation Jacobi method: Improvements and applications." In: *Journal of Computational Physics* 321 (2016), pp. 369 –413. ISSN: 0021-9991. DOI: http://dx.doi.org/10.1016/j.jcp.2016.05.053. URL: http://www.sciencedirect.com/science/article/pii/S002199911630198X.

[4] T. Akgün et al. "The force-free twisted magnetosphere of a neutron star." In: *Mon. Not. R. Astron. Soc.* 462 (Oct. 2016), pp. 1894–1909. DOI: 10.1093/mnras/stw1762. arXiv: 1605.02253 [astro-ph.HE].

[5] A. S. Almgren, J. B. Bell, and W. G. Szymczak. "A numerical method for the incompressible Navier-Stokes equations based on an approximate projection." In: *SIAM Journal on Scientific Computing* 17.2 (1996), pp. 358–369.

[6] A. S. Almgren et al. "On the use of higher-order projection methods for incompressible turbulent flow." In: *SIAM Journal on Scientific Computing* 35.1 (2013), B25–B42.

[7] M. A. Aloy et al. "GENESIS: A High-Resolution Code for Three-dimensional Relativistic Hydrodynamics." In: *Astrophys. J. Suppl. Ser.* 122 (May 1999), pp. 151–166. DOI: 10.1086/313214.

[8] R. S. Anderssen and G. H. Golub. *Richardson's non-stationary matrix iterative procedure.* Tech. rep. STAN-CS-72-304. Computer Science Dept., Stanford Univ., 1972, 76. URL: http://i.stanford.edu/pub/cstr/reports/cs/tr/72/304/CS-TR-72-304.pdf.

[9] M. Ansorg, A. Kleinwächter, and R. Meinel. "Relativistic Dyson Rings and Their Black Hole Limit." In: *Astrophys. J. Lett.* 582 (Jan. 2003), pp. L87–L90. DOI: 10.1086/367632. eprint: gr-qc/0211040.

[10] M. Antuono and G. Colicchio. "Delayed Over-Relaxation for iterative methods." In: *Journal of Computational Physics* 321 (2016), pp. 892–907.

[11] F. Arándiga, R. Donat, and P. Mulet. *Metodes numerics per a l'algebra lineal*. Universitat de Valencia, 2000. ISBN: 84-370-4390-5.

[12] T. W. Baumgarte and S. L. Shapiro. "Numerical integration of Einstein's field equations." In: *Physical Review D* 59.2, 024007 (Jan. 1999), p. 024007. DOI: 10.1103/PhysRevD.59.024007. eprint: gr-qc/9810065.

[13] A. Bauswein, R. Oechslin, and H.-T. Janka. "Discriminating strange star mergers from neutron star mergers by gravitational-wave measurements." In: *Phys. Rev. D* 81.2, 024012 (Jan. 2010), p. 024012. DOI: 10.1103/PhysRevD.81.024012. arXiv: 0910.5169 [astro-ph.SR].

[14] J. B. Bell, P. Colella, and H. M. Glaz. "A Second Order Projection Method for the Incompressible Navier-Stokes Equations." In: *Journal of Computational Physics* 85 (Dec. 1989), pp. 257–283. DOI: 10.1016/0021-9991(89)90151-4.

[15] P. E. Bjorstad and O. B. Widlund. "Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures." In: *SIAM Journal on Numerical Analysis* 23 (Dec. 1986), pp. 1097–1120. DOI: 10.1137/0723075.

[16] R. D. Blandford and M. J. Rees. "A 'twin-exhaust' model for double radio sources." In: *Mon. Not. R. Astron. Soc.* 169 (Dec. 1974), pp. 395–415. DOI: 10.1093/mnras/169.3.395.

[17] P. Bodenheimer et al. *Numerical Methods in Astrophysics: An Introduction (Series in Astronomy and Astrophysics)*. CRC Press, 2006. ISBN: 9780750308830. URL: http://amazon.com/o/ASIN/0750308834/.

[18] S. Bonazzola, E. Gourgoulhon, and J.-A. Marck. "Numerical approach for high precision 3D relativistic star models." In: *Phys. Rev. D* 58.10, 104020 (Nov. 1998), p. 104020. DOI: 10.1103/PhysRevD.58.104020. eprint: astro-ph/9803086.

[19] S. Bonazzola and J. Schneider. "An Exact Study of Rigidly and Rapidly Rotating Stars in General Relativity with Application to the Crab Pulsar." In: *Astrophys. J.* 191 (July 1974), pp. 273–290. DOI: 10.1086/152965.

[20] S. Bonazzola et al. "Axisymmetric rotating relativistic bodies: A new numerical approach for 'exact' solutions." In: *Astron. Astrophys.* 278 (Nov. 1993), pp. 421–443.

[21] S. Bonazzola et al. "Constrained scheme for the Einstein equations based on the Dirac gauge and spherical coordinates." In: *Phys. Rev. D* 70 (10 2004), p. 104007. DOI: 10.1103/PhysRevD.70.104007. URL: https://link.aps.org/doi/10.1103/PhysRevD.70.104007.

[22] J. U. Brackbill and D. C. Barnes. "The effect of nonzero product of magnetic gradient and B on the numerical solution of the magnetohydrodynamic equations." In: *Journal of Computational Physics* 35 (May 1980), pp. 426–430. DOI: 10.1016/0021-9991(80)90079-0.

[23] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial (2nd Ed.)* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000. ISBN: 0-89871-462-1.

[24] A. Buonanno et al. “Reducing orbital eccentricity of precessing black-hole binaries.” In: *Phys. Rev. D* 83 (10 2011), p. 104034. DOI: 10.1103/PhysRevD.83.104034. URL: https://link.aps.org/doi/10.1103/PhysRevD.83.104034.

[25] R. L. Burden and J. D. Faires. *Numerical analysis*. 2001.

[26] F. W. Campbell, J. G. Robson, and G. Westheimer. “Fluctuations of accommodation under steady viewing conditions.” In: *The Journal of Physiology* 145.3 (Mar. 1959), pp. 579–594. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1356964/.

[27] J. Centrella et al. “Black-hole binaries, gravitational waves, and numerical relativity.” In: *Reviews of Modern Physics* 82 (Oct. 2010), pp. 3069–3119. DOI: 10.1103/RevModPhys.82.3069. arXiv: 1010.5260 [gr-qc].

[28] P. Cerdá-Durán et al. “CFC+: improved dynamics and gravitational waveforms from relativistic core collapse simulations.” In: *Astron. Astrophys.* 439 (Sept. 2005), pp. 1033–1055. DOI: 10.1051/0004-6361:20042602. eprint: astro-ph/0412611.

[29] S. Chandrasekhar. “Axisymmetric Magnetic Fields and Fluid Motions.” In: *Astrophys. J.* 124 (July 1956), p. 232. DOI: 10.1086/146217.

[30] S. Chandrasekhar and K. H. Prendergast. “The Equilibrium of Magnetic Stars.” In: *Proceedings of the National Academy of Science* 42 (Jan. 1956), pp. 5–9. DOI: 10.1073/pnas.42.1.5.

[31] W. N. Charman and G. Heron. “Fluctuations in accommodation: a review.” In: *Ophthalmic and Physiological Optics* 8.2 (1988), pp. 153–164. ISSN: 1475-1313. DOI: 10.1111/j.1475-1313.1988.tb01031.x. URL: http://dx.doi.org/10.1111/j.1475-1313.1988.tb01031.x.

[32] A. J. Chorin. “On the convergence of discrete approximations to the Navier-Stokes equations.” In: *Math. Comp.* 23 (1969), pp. 341–353. DOI: 10.1090/S0025-5718-1969-0242393-5.

[33] C. K. Chu. “Magnetohydrodynamic Nozzle Flow with Three Transitions.” In: *Physics of Fluids* 5 (May 1962), pp. 550–559. DOI: 10.1063/1.1706656.

[34] I. Contopoulos, D. Kazanas, and D. B. Papadopoulos. “The Force-Free Magnetosphere of a Rotating Black Hole.” In: *The Astrophysical Journal* 765.2 (2013), p. 113. DOI: 10.1088/0004-637x/765/2/113.

[35] G. Cook. “Initial Data for Numerical Relativity.” In: *Living Reviews in Relativity* 3 (Dec. 2000). DOI: 10.12942/lrr-2000-5. eprint: gr-qc/0007085.

[36] G. B. Cook. “Initial Data for Numerical Relativity.” In: *Living Reviews in Relativity* 3, 5 (Nov. 2000), p. 5. DOI: 10.12942/lrr-2000-5. eprint: gr-qc/0007085.

[37] I. Cordero-Carrión et al. “Improved constrained scheme for the Einstein equations: An approach to the uniqueness issue.” In: *Phys. Rev. D* 79.2, 024017 (Jan. 2009), p. 024017. DOI: 10.1103/PhysRevD.79.024017. arXiv: 0809.2325 [gr-qc].

[38] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Vol. 2. Wiley-Interscience, New York, 1962.

[39] *CUDA*. URL: http://www.nvidia.com/object/cuda_home_new.html.

[40] C. Cuesta. PhD thesis. Valencia, Spain: Universitat de València, 2017.

[41] W. Dai and P. R. Woodward. "On the Divergence-free Condition and Conservation Laws in Numerical Simulations for Supersonic Magnetohydrodynamical Flows." In: *The Astrophysical Journal* 494.1 (1998), p. 317. URL: http://stacks.iop.org/0004-637X/494/i=1/a=317.

[42] M. Dehghan and M. Hajarian. "Improving preconditioned SOR-type iterative methods for L-matrices." In: *Int. J. Numer. Methods Biomed. Eng.* 27.5 (2011), pp. 774–784.

[43] T. Dietrich et al. "Gravitational waves and mass ejecta from binary neutron star mergers: Effect of the stars' rotation." In: *Phys. Rev. D* 95.4, 044045 (Feb. 2017), p. 044045. DOI: 10.1103/PhysRevD.95.044045. arXiv: 1611.07367 [gr-qc].

[44] H. Dimmelmeier, J. A. Font, and E. Müller. "Gravitational Waves from Relativistic Rotational Core Collapse." In: *Astrophys. J. Lett.* 560 (Oct. 2001), pp. L163–L166. DOI: 10.1086/324406. eprint: astro-ph/0103088.

[45] H. Dimmelmeier, J. A. Font, and E. Müller. "Relativistic simulations of rotational core collapse I. Methods, initial models, and code tests." In: *Astron. Astrophys.* 388 (June 2002), pp. 917–935. DOI: 10.1051/0004-6361:20020563. eprint: astro-ph/0204288.

[46] H. Dimmelmeier, J. A. Font, and E. Müller. "Relativistic simulations of rotational core collapse II. Collapse dynamics and gravitational radiation." In: *Astron. Astrophys.* 393 (Oct. 2002), pp. 523–542. DOI: 10.1051/0004-6361:20021053. eprint: astro-ph/0204289.

[47] H. Dimmelmeier et al. "Combining spectral and shock-capturing methods: A new numerical approach for 3D relativistic core collapse simulations." In: *Phys. Rev. D* 71.6, 064023 (Mar. 2005), p. 064023. DOI: 10.1103/PhysRevD.71.064023. eprint: astro-ph/0407174.

[48] A. Dubra. "A Shearing Interferometer for the Evaluation of Human Tear Film Topography." PhD thesis. London: Imperial College, 2004.

[49] F. W. Dyson. "The Potential of an Anchor Ring. [Abstract]." In: *Proceedings of the Royal Society of London Series I* 51 (1892), pp. 448–452.

[50] F. W. Dyson. "The Potential of an Anchor Ring. Part II." In: *Philosophical Transactions of the Royal Society of London Series A* 184 (1893), pp. 1041–1106. DOI: 10.1098/rsta.1893.0020.

[51] W. E. East, F. M. Ramazanoglu, and F. Pretorius. "Conformal thin-sandwich solver for generic initial data." In: *Phys. Rev. D* 86 (10 2012), p. 104053. DOI: 10.1103/PhysRevD.86.104053. URL: https://link.aps.org/doi/10.1103/PhysRevD.86.104053.

[52] C. R. Ethier, M. Johnson, and J. Ruberti. "Ocular Biomechanics and Biotransport." In: *Annual Review of Biomedical Engineering* 6.1 (2004). PMID: 15255770, pp. 249–273. DOI: 10.1146/annurev.bioeng.6.040803.140055. eprint: http://dx.doi.org/10.1146/annurev.bioeng.6.040803.140055. URL: http://dx.doi.org/10.1146/annurev.bioeng.6.040803.140055.

[53] L. Evans. *Partial Differential Equations (Graduate Studies in Mathematics)*. American Mathematical Society. 2nd edition. ebenda, 2010. ISBN: 9780821849743. URL: http://books.google.es/books?id=TwFInwEACAAJ.

[54] X. Feng and M. Zhang. “A comparative study of divergence cleaning methods of magnetic field in the solar coronal numerical simulation.” In: *Frontiers in Astronomy and Space Sciences* 3, 6 (Mar. 2016), p. 6. DOI: 10.3389/fspas.2016.00006.

[55] R. Feynman, R. Leighton, and M. Sands. *The Feynman Lectures in Physics*. Vol. II. Addison-Wesley, 1966. ISBN: 9780465024940.

[56] M. J. Flynn. “Some Computer Organizations and Their Effectiveness.” In: *IEEE Trans. Computers* 21.9 (1972), pp. 948–960.

[57] F. Foucart et al. “Initial data for black hole–neutron star binaries: A flexible, high-accuracy spectral method.” In: *Phys. Rev. D* 77 (12 2008), p. 124051. DOI: 10.1103/PhysRevD.77.124051. URL: https://link.aps.org/doi/10.1103/PhysRevD.77.124051.

[58] W. L. Frank. “Solution of Linear Systems by Richardson’s Method.” In: *J. ACM* 7.3 (July 1960), pp. 274–286. ISSN: 0004-5411. DOI: 10.1145/321033.321041. URL: http://doi.acm.org/10.1145/321033.321041.

[59] B. A. W. D. Fryxell, E. Müller, and Arnett. “Hydrodynamics and Nuclear Burning.” In: *preprint MPA-449*. Max Plank Institut für Astrophysics (Garching). 1989.

[60] C. F. V. L. Gene H. Golub. *Matrix Computations*. 4th. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2012. ISBN: 1421407949,9781421407944.

[61] N. I. M. Gould. “Iterative methods for ill-conditioned linear systems from optimization.” In: *Nonlinear Optimization and Related Topics*. Ed. by G. D. Pillo and F. Giannessi. Boston, MA: Springer US, 2000, pp. 123–141. ISBN: 978-1-4757-3226-9. DOI: 10.1007/978-1-4757-3226-9_7. URL: http://dx.doi.org/10.1007/978-1-4757-3226-9_7.

[62] E. Gourgoulhon, P. Grandclément, and S. Bonazzola. “Binary black holes in circular orbits. I. A global spacetime approach.” In: *Phys. Rev. D* 65.4, 044020 (Feb. 2002), p. 044020. DOI: 10.1103/PhysRevD.65.044020. eprint: gr-qc/0106015.

[63] H. Grad and J. Hogan. “Classical Diffusion in a Tokomak.” In: *Physical Review Letters* 24 (June 1970), pp. 1337–1340. DOI: 10.1103/PhysRevLett.24.1337.

[64] H. Grad and H. Rubin. “” In: *Proc. Second United Nations Conf. Peaceful Uses of Atomic Energy (Geneva)*. Vol. 31. 1958, p. 190.

[65] P. Grandclément, E. Gourgoulhon, and S. Bonazzola. “Binary black holes in circular orbits. II. Numerical methods and first results.” In: *Phys. Rev. D* 65.4, 044021 (Feb. 2002), p. 044021. DOI: 10.1103/PhysRevD.65.044021. eprint: gr-qc/0106016.

[66] W. Grossmann and H. Weitzner. “A reformulation of lower-hybrid wave propagation and absorption.” In: *Physics of Fluids* 27 (July 1984), pp. 1699–1703. DOI: 10.1063/1.864824.

[67] A. D. Gunawardena, S. K. Jain, and L. Snyder. “Modified iterative methods for consistent linear systems.” In: *Linear Algebra and its Applications* 154 (Aug. 1991), pp. 123–143. DOI: http://dx.doi.org/10.1016/0024-3795(91)90376-8.

[68] M. H. Gutknecht and S. Rallin. "The Chebyshev iteration revisited." In: *Parallel Computing* 28.2 (2002), pp. 263 –283. ISSN: 0167-8191. DOI: {10.1016/S0167-8191(01)00139-9}. URL: http://www.sciencedirect.com/science/article/pii/S0167819101001399.

[69] J. B. Hartle and S. W. Hawking. "Wave function of the Universe." In: *Phys. Rev. D* 28 (Dec. 1983), pp. 2960–2975. DOI: 10.1103/PhysRevD.28.2960.

[70] C. Hirneiss et al. "Corneal biomechanics measured with the ocular response analyser in patients with unilateral open-angle glaucoma." In: *Acta Ophthalmologica* 89.2 (2011), e189–e192. ISSN: 1755-3768. DOI: 10.1111/j.1755-3768.2010.02093.x.

[71] C. K. Hitzenberger. "Optical measurement of the axial eye length by laser Doppler interferometry." In: *Invest. Ophthalmol. Vis. Sci.* 32.3 (1991), pp. 616 –24.

[72] J. D. Hoffman. *Numerical methods for engineers and scientists*. New York, NY: McGraw-Hill, 1992.

[73] R. N. Hussain and G. Shahid F. Woodruff. "Axial length in apparently normal pediatric eyes." In: *Eur. J. Ophthalmol.* 24.1 (2014), pp. 120 –123.

[74] C. Jacobi. "Über eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden linären Gleichungen." In: *Astronomische Nachrichten* 22.20 (1845), pp. 297 –306. DOI: 10.1002/asna.18450222002.

[75] H.-T. Janka and E. Mueller. "Neutrino heating, convection, and the mechanism of Type-II supernova explosions." In: *Astron. Astrophys.* 306 (Feb. 1996), p. 167.

[76] M. L. Juncosa and T. W. Mulliken. "On the increase of convergence rates of relaxation procedures for elliptic partial differential equations." In: *J. Assoc. Comput. Math.* 7 (1960), pp. 29–36.

[77] W. Keil. PhD thesis. München, Germany: Technische Universität München, 1997.

[78] K. Kifonidis et al. "Non-spherical core collapse supernovae. I. Neutrino-driven convection, Rayleigh-Taylor instabilities, and the formation and propagation of metal clumps." In: *Astron. Astrophys.* 408 (Sept. 2003), pp. 621–649. DOI: 10.1051/0004-6361:20030863. eprint: astro-ph/0302239.

[79] C. Klein. "Binary black hole spacetimes with a helical Killing vector." In: *Phys. Rev. D* 70.12, 124026 (Dec. 2004), p. 124026. DOI: 10.1103/PhysRevD.70.124026. eprint: gr-qc/0410095.

[80] H. Komatsu, Y. Eriguchi, and I. Hachisu. "Rapidly rotating general relativistic stars. I - Numerical method and its application to uniformly rotating polytropes." In: *Mon. Not. R. Astron. Soc.* 237 (Mar. 1989), pp. 355–379. DOI: 10.1093/mnras/237.2.355.

[81] H. Komatsu, Y. Eriguchi, and I. Hachisu. "Rapidly rotating general relativistic stars. II - Differentially rotating polytropes." In: *Mon. Not. R. Astron. Soc.* 239 (July 1989), pp. 153–171. DOI: 10.1093/mnras/239.1.153.

[82] C. Kouveliotou et al. "Identification of two classes of gamma-ray bursts." In: *Astrophys. J. Lett.* 413 (Aug. 1993), pp. L101–L104. DOI: 10.1086/186969.

[83] *LAPACK - Linear Algebra PACKage*. URL: http://www.netlib.org/lapack/.

[84] V. I. Lebedev and S. A. Finogenov. "On construction of the stable permutations of parameters for the Chebyshev iterative methods. Part I." In: *Russ. J. Numer. Anal. Math. Modelling* 17.5 (2002), pp. 437 –456.

[85] V. I. Lebedev and S. A. Finogenov. "On the order of choice of the iteration parameters in the Chebyshev cyclic iteration method." In: *Zur. Vych. Mat. i Mat. Fiz.* 11.1 (1971). English translation in Anderssen, R. S. & Golub G. H., "Richardson's non-stationary matrix iterative procedure", Rep. STAN-CS-72-304, Computer Science Dept., Stanford Univ. (1972), pp. 425 –438.

[86] T. Leismann et al. "Relativistic MHD simulations of extragalactic jets." In: *Astron. Astrophys.* 436 (June 2005), pp. 503–526. DOI: 10.1051/0004-6361:20042520.

[87] R.-J. LeVeque and L. Trefethen. "Fourier Analysis of the SOR Iteration." In: *IMA Journal of Numerical Analysis* 8.3 (1988), pp. 273–279. DOI: 10.1093/imanum/8.3.273.

[88] R. J. LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. Vol. 98. Siam, 2007.

[89] P. Lions. "On the Schwarz alternating method. III: a variant for nonoverlapping subdomains." In: *Third international symposium on domain decomposition methods for partial differential equations, Houston (Texas), march 20-22, 1989*. Ed. by T. F. Chan et al. 1990, pp. 21, 43, 44, 47, 66, 71.

[90] L. Llorente et al. "Myopic versus hyperopic eyes: axial length, corneal shape and optical aberrations." In: *Journal of Vision* 4.4 (2004), pp. 288 –298.

[91] F. Löffler et al. "The Einstein Toolkit: a community computational infrastructure for relativistic astrophysics." In: *Classical and Quantum Gravity* 29.11, 115001 (June 2012), p. 115001. DOI: 10.1088/0264-9381/29/11/115001. arXiv: 1111.3344 [gr-qc].

[92] *LORENE - Langage Objet pour la RElativité NumériquE*. URL: http://www.lorene.obspm.fr/.

[93] A. Love. *A Treatise on the Mathematical Theory of Elasticity, 4th Edition*. Cambridge University Press, 1944.

[94] R. Lüst and A. Schlüter. "Kraftfreie Magnetfelder. Mit 4 Textabbildungen." In: *Zeit. f. Astroph.* 34 (1954), p. 263.

[95] A. I. MacFadyen and S. E. Woosley. "Collapsars: Gamma-Ray Bursts and Explosions in "Failed Supernovae"." In: *Astrophys. J.* 524 (Oct. 1999), pp. 262–289.

[96] W Markoff. "Über Poynome, die in einem gegeben Intervalle möglichst wenig von Null abweichen." In: *Math. Ann.* 77 (1916), pp. 213–258.

[97] P. Mimica et al. "Spectral Evolution of Superluminal Components in Parsec-Scale Jets." In: *Astrophys. J.* 696 (May 2009), pp. 1142–1163. DOI: 10.1088/0004-637X/696/2/1142.

[98] C. W. Misner, K. S. Thorne, and J. A. Wheeler. *Gravitation*. W. H. Freeman and Co., 1973. ISBN: 0-7167-0344-0.

[99] N. Moldenhauer et al. "Initial data for binary neutron stars with adjustable eccentricity." In: *Phys. Rev. D* 90 (8 2014), p. 084043. DOI: 10.1103/PhysRevD.90.084043. URL: https://link.aps.org/doi/10.1103/PhysRevD.90.084043.

[100] E. Müller and M. Steinmetz. "Simulating self-gravitating hydrodynamic flows." In: *Computer Physics Communications* 89 (Aug. 1995), pp. 45–58. DOI: 10.1016/0010-4655(94)00185-5. eprint: astro-ph/9402070.

[101] J. v. Neumann. *First Draft of a Report on the EDVAC*. Tech. rep. 1945.

[102] E. S. Nikolaev and A. A. Samarskii. "Selection of the iterative parameters in Richarson's method." In: *Zur. Vych. Mat. i Mat. Fiz.* 12.4 (1972). English translation by J. Berry, pp. 960 –973.

[103] D. Noutsos and M. Tzoumas. "On optimal improvements of classical iterative schemes for Z-matrices." In: *Journal of Computational and Applied Mathematics* 188 (Apr. 2006), pp. 89–106.

[104] M. Obergaulinger, H.-T. Janka, and M. A. Aloy. "Magnetic field amplification and magnetically supported explosions of collapsing, non-rotating stellar cores." In: *Mon. Not. R. Astron. Soc.* 445 (Dec. 2014), pp. 3169–3199. DOI: 10.1093/mnras/stu1969. arXiv: 1405.7466 [astro-ph.SR].

[105] M. Obergaulinger et al. "Axisymmetric simulations of magnetorotational core collapse: approximate inclusion of general relativistic effects." In: *Astron. Astrophys.* 457 (Oct. 2006), pp. 209–222. DOI: 10.1051/0004-6361:20064982. eprint: astro-ph/0602187.

[106] G. Opfer and G. Schober. "Richardson's iteration for nonsymmetric matrices." In: *Linear Algebra and its Applications* 58 (1984), pp. 343 –361. ISSN: 0024-3795. DOI: 10.1016/0024-3795(84)90219-2. URL: http://www.sciencedirect.com/science/article/pii/0024379584902192.

[107] T. H. Otway. *Elliptic–Hyperbolic Partial Differential Equations: A Mini-Course in Geometric and Quasilinear Methods*. 1st ed. Springer Briefs in Mathematics. Springer International Publishing, 2015. ISBN: 978-3-319-19760-9. DOI: 10.1007/978-3-319-19761-6.

[108] T. H. Otway. "Energy inequalities for a model of wave propagation in cold plasma." In: *Publ. Mat.* 52.1 (2008), pp. 195–234. URL: http://projecteuclid.org/euclid.pm/1197908703.

[109] H. P. Pfeiffer, G. B. Cook, and S. A. Teukolsky. "Comparing initial-data sets for binary black holes." In: *Phys. Rev. D* 66.2, 024047 (July 2002), p. 024047. DOI: 10.1103/PhysRevD.66.024047. eprint: gr-qc/0203085.

[110] H. P. Pfeiffer et al. "A multidomain spectral method for solving elliptic equations." In: *Computer Physics Communications* 152 (May 2003), pp. 253–273. DOI: 10.1016/S0010-4655(02)00847-0. eprint: gr-qc/0202096.

[111] T. Plewa and E. Müller. "AMRA: An Adaptive Mesh Refinement hydrodynamic code for astrophysics." In: *Computer Physics Communications* 138 (Aug. 2001), pp. 101–127. DOI: 10.1016/S0010-4655(01)00199-0. eprint: astro-ph/0010626.

[112] T. Plewa and E. Müller. "The consistent multi-fluid advection method." In: *Astron. Astrophys.* 342 (Feb. 1999), pp. 179–191. eprint: astro-ph/9807241.

[113] V. Quilis. "A new multidimensional adaptive mesh refinement hydro + gravity cosmological code." In: *Mon. Not. R. Astron. Soc.* 352 (Aug. 2004), pp. 1426–1438. DOI: 10.1111/j.1365-2966.2004.08040.x. eprint: astro-ph/0405389.

[114] V. Quilis, J. M. A. Ibanez, and D. Saez. "A Multidimensional Hydrodynamic Code for Structure Evolution in Cosmology." In: *Astrophys. J.* 469 (Sept. 1996), p. 11. DOI: 10.1086/177753. eprint: astro-ph/9604037.

[115] M. Rampp and H.-T. Janka. "Radiation hydrodynamics with neutrinos. Variable Eddington factor method for core-collapse supernova simulations." In: *Astron. Astrophys.* 396 (Dec. 2002), pp. 361–392. DOI: 10.1051/0004-6361:20021398. eprint: astro-ph/0203101.

[116] L. F. Richardson. "The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam." In: *Royal Society of London Philosophical Transactions Series A* 210 (1911), pp. 307–357. DOI: 10.1098/rsta.1911.0009.

[117] I. W. Roxburgh. "2-dimensional models of rapidly rotating stars I. Uniformly rotating zero age main sequence stars." In: *Astron. Astrophys.* 428 (Dec. 2004), pp. 171–179. DOI: 10.1051/0004-6361:20041202.

[118] I. W. Roxburgh. "2-dimensional models of rapidly rotating stars. II. Hydrostatic and acoustic models with Omega=Omega(r,thetas)." In: *Astron. Astrophys.* 454 (Aug. 2006), pp. 883–888. DOI: 10.1051/0004-6361:20065109.

[119] I. Ruchlin et al. "Puncture initial data for black-hole binaries with high spins and high boosts." In: *Phys. Rev. D* 95.2, 024033 (Jan. 2017), p. 024033. DOI: 10.1103/PhysRevD.95.024033.

[120] J. Rue. "On the Normal Modes of Freely Vibrating Elastic Objects of Various Shapes." MA thesis. The Netherlands: University of Amsterdam, 1996.

[121] Y. Saad. "A Flexible Inner-Outer Preconditioned GMRES Algorithm." In: *SIAM J. Sci. Comput.* 14.2 (1985), pp. 461–469.

[122] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003. URL: http://epubs.siam.org/doi/abs/10.1137/1.9780898718003.

[123] Y. Saad and M. Schultz. "Conjugate gradient-like algorithms for solving nonsymmetric linear systems." In: *Mathematics of Computation* 44 (1985), pp. 417–424.

[124] S. Salsa. *Partial Differential Equations in Action: From Modelling to Theory*. 3rd ed. UNITEXT 99. Springer International Publishing, 2016. ISBN: 978-3-319-31237-8,978-3-319-31238-5.

[125] K. Schöbel and M. Ansorg. "Maximal mass of uniformly rotating homogeneous stars in Einsteinian gravity." In: *Astron. Astrophys.* 405 (July 2003), pp. 405–408. DOI: 10.1051/0004-6361:20030634. eprint: astro-ph/0301618.

[126] V. D. Shafranov. "On Magnetohydrodynamical Equilibrium Configurations." In: *Soviet Journal of Experimental and Theoretical Physics* 6 (1958), p. 545.

[127] V. D. Shafranov. "On Magnetohydrodynamical Equilibrium Configurations." In: *Soviet Journal of Experimental and Theoretical Physics* 6 (1958), p. 545.

[128] M. Shibata, T. W. Baumgarte, and S. L. Shapiro. "Stability of coalescing binary stars against gravitational collapse: Hydrodynamical simulations." In: *Phys. Rev. D* 58.2, 023002 (July 1998), p. 023002. DOI: 10.1103/PhysRevD.58.023002. eprint: gr-qc/9805026.

[129] M. Shibata and T. Nakamura. "Evolution of three-dimensional gravitational waves: Harmonic slicing case." In: *Physical Review D* 52 (Nov. 1995), pp. 5428–5444. DOI: 10.1103/PhysRevD.52.5428.

[130] G. Shortley. "Use of Tschebyscheff-Polynomial Operators in the Numerical Solution of Boundary-Value Problems." In: *Journal of Applied Physics* 24.4 (1953), pp. 392–396. DOI: {10.1063/1.1721292}.

[131] G. Smith. *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford applied mathematics and computing science series. Clarendon Press, 1985. ISBN: 9780198596509. URL: http://books.google.es/books?id=TwFInwEACAAJ.

[132] N. Stergioulas. "Rotating Stars in Relativity." In: *Living Reviews in Relativity* 1, 8 (June 1998), p. 8. DOI: 10.12942/lrr-1998-8. eprint: gr-qc/9805012.

[133] J. M. Stewart. "Signature change, mixed problems and numerical relativity." In: *Classical and Quantum Gravity* 18 (Dec. 2001), pp. 4983–4995. DOI: 10.1088/0264-9381/18/23/301.

[134] D. F. Styer. "The geometrical significance of the Laplacian." In: *American Journal of Physics* 83.12 (2015), pp. 992–997. DOI: 10.1119/1.4935133. eprint: http://dx.doi.org/10.1119/1.4935133. URL: http://dx.doi.org/10.1119/1.4935133.

[135] L. H. Thomas. *Elliptic Problems in Linear Differential Equations over a Network*. Watson Sci. Comput. Lab Report. Columbia University, 1949.

[136] A. Torres-Forné et al. "Are pulsars born with a hidden magnetic field?" In: *Mon. Not. R. Astron. Soc.* 456 (Mar. 2016), pp. 3813–3826. DOI: 10.1093/mnras/stv2926. arXiv: 1511.03823 [astro-ph.SR].

[137] G. Tóth. "The nabla B=0 Constraint in Shock-Capturing Magnetohydrodynamics Codes." In: *Journal of Computational Physics* 161 (July 2000), pp. 605–652. DOI: 10.1006/jcph.2000.6519.

[138] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001. ISBN: 0-12-701070-X.

[139] A. Tsokaros, K. Uryu, and L. Rezzolla. "New code for quasiequilibrium initial data of binary neutron stars: Corotating, irrotational, and slowly spinning systems." In: *Phys. Rev. D* 91 (10 2015), p. 104030. DOI: 10.1103/PhysRevD.91.104030. URL: https://link.aps.org/doi/10.1103/PhysRevD.91.104030.

[140] `Einstein Toolkit`. http://einsteintoolkit.org/.

[141] E. Uchio et al. “Simulation model of an eyeball based on finite element analysis on a supercomputer.” In: *British Journal of Ophthalmology* 83.10 (1999), pp. 1106–1111. DOI: 10.1136/bjo.83.10.1106. eprint: http://bjo.bmj.com/content/83/10/1106.full.pdf+html. URL: http://bjo.bmj.com/content/83/10/1106.abstract.

[142] D. A. Uzdensky. “Force-Free Magnetosphere of an Accretion Disk - Black Hole System. I. Schwarzschild Geometry.” In: *The Astrophysical Journal* 603.2 (2004), 652–662. DOI: 10.1086/381543.

[143] D. Vaughan and T. Asbury. *Vaughan & Asbury General Ophtahlmology*. Lange Medical Books / McGraw-Hill, 2004.

[144] J. L. Vázquez. *Senderos de la ciencia : del operador Laplaciano a los procesos difusivos no lineales*. Real Academia de Ciencias Exactas, Físicas y Naturales, 2014.

[145] F. Vazza et al. “Turbulence and vorticity in Galaxy clusters generated by structure formation.” In: *Mon. Not. R. Astron. Soc.* 464 (Jan. 2017), pp. 210–230. DOI: 10.1093/mnras/stw2351. arXiv: 1609.03558.

[146] N. Vlahakis and A. Königl. “Magnetic Driving of Relativistic Outflows in Active Galactic Nuclei. I. Interpretation of Parsec-Scale Accelerations.” In: *Astrophys. J.* 605 (Apr. 2004), pp. 656–661. DOI: 10.1086/382670. eprint: astro-ph/0310747.

[147] J. R. Wilson, G. J. Mathews, and P. Marronetti. “Relativistic numerical model for close neutron-star binaries.” In: *Phys. Rev. D* 54 (July 1996), pp. 1317–1331. DOI: 10.1103/PhysRevD.54.1317. eprint: gr-qc/9601017.

[148] S. E. Woosley. “Gamma-ray bursts from stellar mass accretion disks around black holes.” In: *Astrophys. J.* 405 (Mar. 1993), pp. 273–277. DOI: 10.1086/172359.

[149] X. I. Yang and R. Mittal. “Acceleration of the Jacobi iterative method by factors exceeding 100 using scheduled relaxation.” In: *Journal of Computational Physics* 274 (2014), pp. 695 –708. ISSN: 0021-9991. DOI: http://dx.doi.org/10.1016/j.jcp.2014.06.010. URL: http://www.sciencedirect.com/science/article/pii/S0021999114004173.

[150] D. Young. “Iterative Methods for Solving Partial Difference Equations of Elliptic Type.” PhD thesis. Cambridge MA, USA: Harvard University, Mathematics Department, May 1950.

[151] D. Young. “Iterative methods for solving partial difference equations of elliptic type.” In: *Trans. Amer. Math. Soc.* 76 (1954), pp. 92–111. ISSN: 0002-9947.

[152] D. Young. “On Richardson’s Method for Solving Linear Systems with Positive Definite Matrices.” In: *Journal of Mathematics and Physics* 32.1 (1953), pp. 243–255. ISSN: 1467-9590. DOI: 10.1002/sapm1953321243. URL: http://dx.doi.org/10.1002/sapm1953321243.

[153] D. Young. “On Richardson’s method for solving linear systems with positive definite matrices.” In: *J. Math. Phys.* 3 (1954), pp. 243–255.

[154] D. Young. “On the solution of linear systems by iteration.” In: *Proc. Sixth Symp. in Appl. Math.* Vol. 6. Amer. Math. Soc. 1956, pp. 283–298.

[155] D. M. Young. *Iterative Solution of Large Linear Systems*. Dover Ed. Dover Books on Mathematics. Dover Publications, 1971. ISBN: 0486425487,9780486425481.

[156] D. M. Young and W. R. (Auth.) *Iterative Solution of Large Linear Systems*. Computer Science and Applied Mathematics. Elsevier Inc, Academic Press, 1971. ISBN: 978-0-12-773050-9,0127730508.

[157] M. Zingale. *Introduction to Computational Astrophysical Hydrodynamics*. 2017.

[158] D. Zwillinger. *Handbook of differential equations (3rd Ed.)* 1997.