Published in Image Processing On Line on 2018–07–26.
Submitted on 2018–05–18, accepted on 2018–05–31.
ISSN 2105–1232 © 2018 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2018.228

An Analysis and Implementation of Multigrid Poisson Solvers With Verified Linear Complexity

J. Matías Di Martino[1] and Gabriele Facciolo[2]

[1] IFFI, Universidad de la República, 11300 Montevideo, Uruguay (matiasdm@fing.edu.uy)
[2] CMLA, ENS Cachan, CNRS, Université Paris-Saclay, 94235 Cachan, France (facciolo@cmla.ens-cachan.fr)

Abstract

The Poisson equation is the most studied partial differential equation, and it allows to formulate many useful image processing methods in an elegant and efficient mathematical framework. Using different variations of data terms and boundary conditions, Poisson-like problems can be developed, e.g. for local contrast enhancement, inpainting, or image seamless cloning among many other applications. Multigrid solvers are among the most efficient numerical solvers for discrete Poisson-like equations. However, their correct implementation relies on: (i) the proper definition of the discrete problem, (ii) the right choice of interpolation and restriction operators, and (iii) the adequate formulation of the problem across different scales and layers. In the present work we address these aspects, and we provide a mathematical and practical description of multigrid methods. In addition, we present an alternative to the extended formulation of Poisson equation proposed in 2011 by Mainberger et al. The proposed formulation of the problem suits better multigrid methods, in particular, because it has important mathematical properties that can be exploited to define the problem at different scales in a intuitive and natural way. In addition, common iterative solvers and Poisson-like problems are empirically analyzed and compared. For example, the complexity of problems is compared when the topology of Dirichlet boundary conditions changes in the interior of the regular domain of the image. The main contribution of this work is the development and detailed description of an implementation of a multigrid numerical solver which converges in linear time.

Source Code

The reviewed source code and documentation of a Matlab implementation for Multigrid Poisson solvers and the applications described in this work are available from the web page of this article[1]. Usage instructions are included in the `README.txt` file of the archive.

Keywords: multigrid; Poisson equation; Laplace equation; Poisson editing

[1] https://doi.org/10.5201/ipol.2018.228

1 Introduction

The Poisson equation is one of the simplest and more versatile elliptic PDEs. It is at the core of many powerful image processing methods [2, 9, 11, 13, 14, 17] and has been extensively studied for its practical applications and its interesting mathematical properties. In the past decade, the volume of digital data has grown exponentially, and large amounts of high resolution digital images are uploaded and processed everyday. This scenario is imposing new challenges to classical image processing methods which need to be adapted and their theory updated to meet modern computational requirements. In order to make Poisson-based algorithms practical, fast and efficient, numerical solvers must be developed. Multigrid methods are among the most reliable and efficient numerical solvers for Poisson-like problems and tend to outperform other iterative solvers as the size of the domain increases. Our main goal is to describe in detail and review this family of methods both in theory and practice. In addition, we extend the ideas presented by Mainberger et al. [10] where a codec for the compression of cartoon-like images was proposed. This codec is implemented by keeping only the edges of different objects in the image, which are then used to recover the remaining pixels by solving the Laplace equation. To decode edge-based compressed images, Mainberger et al. [10] proposed an efficient multigrid algorithm based on an extended formulation of the Laplace equation. Inspired in these ideas, we extend [10] to non-homogeneous problems, i.e. when a Poisson-like equation needs to be solved instead of a Laplace equation. Furthermore, the proposed formulation of the problem allows to impose arbitrary Dirichlet boundary conditions in arbitrary points of the domain. We show how this problem can be stated in a self-adjoint positive definite form, which guarantees the well-posedness of the presented multigrid numerical solvers. To complement the ideas here presented and to obtain a deeper understanding of multigrid, we highly recommend to complement the reading of this article with the book of Bringgs et al. [4].

2 Description of the Problem

Let us begin by analyzing the continuous formulation of the problem which gives us the intuition and tools that later are used to introduce its discrete counterpart.

Poisson equation on a continuous domain. Let Ω be a domain with piecewise C^1 boundary on the Euclidean plane $\mathbf{R}^2$, and let $f : \Omega \to \mathbf{R}$ and $g : \partial\Omega \to \mathbf{R}$ be two functions on Ω and $\partial\Omega$, respectively. The Poisson equation can be defined setting different kinds of boundary conditions, e.g., Dirichlet and Neumann are two canonical cases. When Dirichlet conditions are imposed g plays the role of setting the value of the solution on $\partial\Omega$, i.e.

$$\begin{cases} \Delta u = f & \text{on } \Omega \\ u = g & \text{on } \partial\Omega, \end{cases} \tag{1}$$

where $u : \Omega \to \mathbf{R}$ is the solution to the problem, and f the Laplacian datum. On the other hand, Neumann conditions can be imposed letting g be the value of the outward partial derivative of the solution on $\partial\Omega$ i.e.

$$\begin{cases} \Delta u = f & \text{on } \Omega \\ \dfrac{\partial u}{\partial \nu} = g & \text{on } \partial\Omega, \end{cases} \tag{2}$$

where ν denotes the outward unit normal of $\partial\Omega$.

The differential operator $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ corresponds to the Laplace operator, and it is arguably the simplest nontrivial second order operator. The existence and uniqueness of solutions to Equa-

tions (1) - (2), provided that the data terms are regular enough, is a standard result in linear PDE theory [5].

2.1 Discrete Model

The present work focuses on the analysis of the numerical solution of discrete approximations of the Poisson equation. In particular, we will focus on image-like data, i.e. we will assume that the domain of interest can be sampled as a regular rectangular grid

$$u(x, y) = u_{ij} \quad \text{for } x = hj,\ y = hi, \tag{3}$$

where h is the distance between samples (pixels) and the discrete indexes $(i, j) \in \Omega^h \stackrel{def}{=} \{1, \dots, H\} \times \{1, \dots, W\} \subset \mathbf{Z}^2$.

A discrete approximation for the Laplacian can be obtained through a second order Taylor expansion,

$$\begin{aligned}
u_{i(j+1)} &= u(x + h, y) = u(x, y) + u_x\, h + u_{xx}\, \frac{h^2}{2} + \mathcal{O}(h^3) \\
u_{i(j-1)} &= u(x - h, y) = u(x, y) - u_x\, h + u_{xx}\, \frac{h^2}{2} + \mathcal{O}(h^3) \\
u_{(i+1)j} &= u(x, y + h) = u(x, y) + u_y\, h + u_{yy}\, \frac{h^2}{2} + \mathcal{O}(h^3) \\
u_{(i-1)j} &= u(x, y - h) = u(x, y) - u_y\, h + u_{yy}\, \frac{h^2}{2} + \mathcal{O}(h^3) \\
\Rightarrow\ \Delta u(x, y) &\stackrel{h\to 0}{\approx} \Delta^h u_{ij} \stackrel{def}{=} \tfrac{1}{h^2}\left(-4u_{ij} + u_{(i+1)j} + u_{(i-1)j} + u_{i(j+1)} + u_{i(j-1)}\right),
\end{aligned} \tag{4}$$

which leads to the standard five-point Laplacian stencil. The previous definition requires the knowledge of the values of u outside Ω (e.g. when computing $\Delta^h u$ at the boundaries of Ω^h). In the present work, when a value outside the domain is required, the value of the nearest pixel inside Ω^h is used instead. This is an arbitrary choice that corresponds to implicitly impose vanishing Neumann conditions. It is also equivalent to considering the *graph Laplacian* [3] on the graph whose vertices are the points of Ω^h and whose edges are given by 4-connectivity.

A vanishing discrete Laplacian has an intuitive interpretation: if we regard the values of u as measuring some quantity, the fact that $\Delta u(p) = 0$ implies that the quantity at point p equals the average quantity on the neighbors of p (thus, $u(p)$ can be recovered by averaging the values of u on the neighbors of p). When $\Delta u(p) = 0$ for every p, we say that the quantity is *in equilibrium*.

The discrete Poisson equation on Ω^h can be expressed as

$$\Delta^h u(i, j) = f(i, j) \qquad (i, j) \in \Omega^h, \tag{5}$$

f imposes the Laplacian values of the solution, and Neumann boundary conditions at $\partial\Omega^h$ are implicitly met as described above. Notice that a super-index is included to make the distance between pixels explicit. This will become helpful to differentiate problems defined across different discrete grids (which is the core of multigrid analysis).

Equation (5) is the most standard way of defining the Poisson equation, however, in many practical applications [10, 13] it is necessary to impose additional conditions on the interior of the discrete set

Ω^h. This more general problem can be expressed as

$$\begin{cases} \Delta^h u(i,j) = f(i,j) & (i,j) \in \Omega^h \backslash \Theta^h \\ u(i,j) = g(i,j) & (i,j) \in \Theta^h, \end{cases} \tag{6}$$

where Θ^h is a subset of Ω^h indicating the positions of pixels to be set. As before $f : \Omega^h \backslash \Theta^h \to \mathbf{R}$ represents the Laplacian data, and $g : \Theta^h \to \mathbf{R}$ the values of the solution on Θ^h. Again, Neumann boundary conditions on the edges of the image are automatically enforced due to the chosen definition for the discrete Laplacian.

Equation (6) is linear in the set of unknowns $u(i,j)$, and therefore, it can be expressed as

$$A^h \, u^h = b^h. \tag{7}$$

Abusing of notation, let u^h now denote a $HW \times 1$ column vector obtained by stacking column wise the image $u(i,j)$, b^h is a column vector containing a mixture of data terms f and g, and the matrix A^h is a $HW \times HW$ sparse matrix obtained from Δ^h and Θ^h. Equation (7) can be expressed in different ways. In the following we describe three alternatives to define this linear problem: the *Reduced Formulation*, the *Extended Formulation*, and a *Self-Adjoint Extended Formulation*.

Reduced and Extended Formulation. Using the column representation introduced above, the Laplacian operator Δ^h can be implemented as the matrix multiplication

$$\mathcal{L}^h u^h = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & 1 & \cdots & 0 & 0 \\ 1 & -3 & 1 & 0 & \cdots & 1 & 0 \\ 1 & 1 & -4 & 1 & \cdots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & \cdots & 1 & -2 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{31} \\ \vdots \\ u_{HW} \end{pmatrix}. \tag{8}$$

Therefore Equation (6) can be expressed as

$$\left[(I - I_{\Theta^h})(-\mathcal{L}^h) + I_{\Theta^h}\right] u^h = (I - I_{\Theta^h})(-f^h) + I_{\Theta^h} g^h. \tag{9}$$

I denotes the $HW \times HW$ identity matrix, and I_{Θ^h} corresponds to a diagonal matrix where the element $I_{\Theta^h}(k,k) = 1$ if $k \in \Theta^h$ and 0 otherwise.

Equation (9) can be seen as the concatenation of the equations $-\Delta^h u^h(i,j) = -f^h(i,j)$ if $(i,j) \in \Omega^h \backslash \Theta^h$ and $u^h(i,j) = g^h(i,j)$ if $(i,j) \in \Theta^h$. The multiplication by -1 on the first set of equations is introduced to define a linear problem with nonnegative eigenvalues.

The reduced formulation is obtained when the number of unknowns is minimized by removing those associated to trivial conditions $u^h(k) = g^h(k)$ (and of course, modifying the data terms

accordingly [10]). Let us illustrate this with the following 3×3 toy example

$$f^h = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & - \\ f_{31} & - & f_{33} \end{pmatrix}, \quad g^h = \begin{pmatrix} - & - & - \\ - & - & g_{23} \\ - & g_{32} & - \end{pmatrix}, \quad u^h = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix},$$

$$I_{\Theta^h} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathcal{L}^h = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -3 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2 \end{pmatrix}. \tag{10}$$

Equation (10) illustrates a Poisson-like problem in which pixels $(2,3)$ and $(3,2)$ are set to the values g_{23} and g_{32} respectively, and a condition on the Laplacian of the remaining pixels is imposed. The concatenation of Equations $-\Delta^h u^h(i,j) = -f^h(i,j)$ for $(i,j) \in \Omega^h \backslash \Theta^h$ and $u^h(i,j) = g^h(i,j)$ for $(i,j) \in \Theta^h$ leads to an *Extended Formulation* linear problem

$$A^h_{ext} u^h_{ext} = b^h_{ext} \rightarrow \underbrace{\begin{pmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 3 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix}}_{\left[h^2(I-I_{\Theta^h})(-\mathcal{L}^h)+I_{\Theta^h}\right]} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{13} \\ u_{23} \\ u_{33} \end{pmatrix} = \underbrace{\begin{pmatrix} -h^2 f_{11} \\ -h^2 f_{21} \\ -h^2 f_{31} \\ -h^2 f_{12} \\ -h^2 f_{22} \\ g_{32} \\ -h^2 f_{13} \\ g_{23} \\ -h^2 f_{33} \end{pmatrix}}_{h^2(I-I_{\Theta^h})(-f^h)+I_{\Theta^h} g^h}. \tag{11}$$

This linear system can be reduced by eliminating the unknowns associated to trivial conditions, e.g. we can set $u_{32} = g_{32}$ and $u_{23} = g_{23}$ obtaining a more compact *Reduced Formulation*

$$A^h_{red} u^h_{red} = b^h_{red} \rightarrow \begin{pmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 3 & -1 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{13} \\ u_{33} \end{pmatrix} = \begin{pmatrix} -h^2 f_{11} \\ -h^2 f_{21} \\ -h^2 f_{31} - g_{32} \\ -h^2 f_{12} \\ -h^2 f_{22} - g_{32} - g_{23} \\ -h^2 f_{13} - g_{23} \\ -h^2 f_{33} - g_{32} - g_{23} \end{pmatrix}. \tag{12}$$

The Reduced Formulation is an effective and compact way of writing the linear problem associated with the discrete Poisson equation. It is also the more efficient in terms of memory usage (as only the non-trivial equations are preserved). However, the reduced formulation is not convenient when multigrid methods are considered. The main idea behind multigrid approaches consists of solving the numerical problem along a set of different layers. Each layer is associated with a specific spatial resolution of the problem. Regular grids can be sampled very naturally and represented in a pyramidal structure which facilitates the definition of multigrid solution. Because of this, it is

useful to keep the set of trivial equations (i.e. those associated to Dirichlet conditions) which leads to the Extended Formulation of the problem [10]. As we shall see, this regular representation of the domain can be exploited to substantially simplify the implementation of hierarchical representations of $\Omega^h \rightarrow \{\Omega^h, \Omega^{2h}, \ldots, \Omega^H\}$, which is crucial for multigrid numerical solvers.

Self-Adjoint Extended Formulation. The extended formulation provides a practical definition of the problem to implement hierarchical representations of the domain. However, the resulting matrix A^h_{ext} that characterizes the problem lacks some important properties as we shall see. In particular, to guarantee multigrid methods convergence, it is sufficient that the matrix A^h associated to the linear problem $A^h u^h = b^h$ be self-adjoint and positive definite (see for instance Appendix A and B). This condition can be written as

$$\begin{aligned} &\langle A^h\, u^h, v^h\rangle = \langle u^h, A^h\, v^h\rangle \quad \forall\ u^h,\ v^h\ ; \\ &\langle A^h\, u^h, u^h\rangle > 0 \qquad\qquad\quad u^h \neq 0. \end{aligned} \tag{13}$$

It is easy to see that (because of the lack of symmetry) the extended formulation of the problem does not satisfy the self-adjoint positive definite conditions detailed in Equation (13) (see, e.g. Equation (11)).

To obtain a self-adjoint extended formulation we start by defining $\tilde{g}(i,j) = g(i,j)$ for $(i,j) \in \Theta^h$ and 0 otherwise. Then we apply the change of variable $u' = u - \tilde{g}$ to obtain an equivalent system but with homogeneous boundary conditions

$$\begin{cases} -\Delta^h u(i,j) = -f(i,j) & (i,j) \in \Omega^h \backslash \Theta^h \\ u(i,j) = g(i,j) & (i,j) \in \Theta^h \end{cases} \rightarrow \begin{cases} -\Delta^h u' = -f(i,j) + \Delta\tilde{g} & (i,j) \in \Omega^h \backslash \Theta^h \\ u'(i,j) = 0 & (i,j) \in \Theta^h. \end{cases} \tag{14}$$

Then, as $u'(i,j) = 0$ for $(i,j) \in \Theta^h$, $\mathcal{L}^h u' \equiv \mathcal{L}^h (I - I_{\Theta^h}) u'$ for all the pixels in $\Omega \backslash \Theta^h$. Hence, Equation (14) can be rewritten as the sparse linear problem,

$$\left[(I - I_{\Theta^h})(-\mathcal{L}^h)(I - I_{\Theta^h}) + I_{\Theta^h}\right] u'^h = (I - I_{\Theta^h})(-f^h + \Delta\tilde{g}^h). \tag{15}$$

After solving this system of equations, the change of variable should be reverted as $u = u' + \tilde{g}$ recovering the proper solution to the original problem u.

Recalling our simple 3×3 illustrative example, the proposed self-adjoint extended formulation corresponds to

$$A^h_{sa-ext} u'^h = b^h_{sa-ext} \rightarrow \underbrace{\begin{pmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 3 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}}_{\left[h^2(I-I_{\Theta^h})(-\mathcal{L}^h)(I-I_{\Theta^h})+I_{\Theta^h}\right]} \begin{pmatrix} u'_{11} \\ u'_{21} \\ u'_{31} \\ u'_{12} \\ u'_{22} \\ u'_{32} \\ u'_{13} \\ u'_{23} \\ u'_{33} \end{pmatrix} = \underbrace{\begin{pmatrix} -h^2(f_{11} + \Delta\tilde{g}_{11}) \\ -h^2(f_{21} + \Delta\tilde{g}_{21}) \\ -h^2(f_{31} + \Delta\tilde{g}_{31}) \\ -h^2(f_{12} + \Delta\tilde{g}_{12}) \\ -h^2(f_{22} + \Delta\tilde{g}_{22}) \\ 0 \\ -h^2(f_{13} + \Delta\tilde{g}_{13}) \\ 0 \\ -h^2(f_{33} + \Delta\tilde{g}_{33}) \end{pmatrix}}_{h^2(I-I_{\Theta^h})(-f^h+\Delta\tilde{g}^h)}. \tag{16}$$

Proposition 1. *If $\Theta \neq \emptyset$, the Reduced, Extended and Self-Adjoint discrete problems have the same (unique) solution.*

It is easy to see that if a discrete function u is solution of one of the linear problems then it is also a solution of the others. To prove that the three formulations of the problem are equivalent, we must prove that a solution for each of these problems exists and is unique.

[6, **Theorem 1**]. *If the partitioned matrix A is block strictly diagonally dominant, or if A is block irreducible and block diagonally dominant with* $|A_{i,i}| > \sum_{j \neq i} |A_{i,j}|$ *for at least one i, then A is non-singular.*

When $\Theta \neq \emptyset$ the previous theorem applies for the three formulations of the problem, proving that A_{red}, A_{ext}, and A_{sa-ext} are non-singular. Hence, the linear problems associated to these matrices have a well defined unique solution. This proves that the solution for the discrete Poisson problem can be obtained by solving any of the previous formulations.

3 Multigrid Numerical Solvers

In this section we introduce several numerical solvers that can be used to solve the Self-Adjoint Extended formulation of the discrete Poisson Equation (6). We start by introducing the basic (one-grid) iterative solvers, then we analyze how to improve the performance of these solvers by applying them at two different scales. Finally, we generalize the ideas developed for two grids to multiple grids, which leads to several different multigrid approaches.

Notation. Let $Au = b$ denote a system of linear equations. We use the variable u to denote the exact solution of this system and v to denote an approximation (obtained by some numerical solver). By u_k/u_{ij} we denote the vector/matrix associated to the continuous 2D signal $u(x, y)$, if the discrete variable u is treated as a matrix or as a column vector will be clear from the context. To emphasize that a given variable is associated to a particular grid, e.g. Ω^h, we use the notation u^h, v^h. Assuming that the solution of the system $Au = b$ is unique, there are two important quantities to look at

$$e \stackrel{def}{=} u - v, \tag{17}$$

named as the *error* or *algebraic error*, and

$$r \stackrel{def}{=} b - Av, \tag{18}$$

known as the *residual*. In addition we denote as $\|u^h\|_h$ the discrete $\mathcal{L}^2$ norm defined on a uniform grid with spacing h as,

$$\|u^h\|_h = \left(h^2 \sum_k (u_k^h)^2 \right)^{1/2}. \tag{19}$$

The scaling factor h^2 is introduced to make the discrete norm an approximation of the continuous $\mathcal{L}^2$ norm.

3.1 Basic Iterative Solvers

Jacobi and Gauss-Seidel iterative methods are very simple and play an important role in multigrid solvers. These methods can be used to solve linear problems $Ax = b$ and convergence is guaranteed when the matrix A is diagonally dominant [7]. For all the discrete problems formulated, this condition is met when the set Θ^h is not empty. An approximate expression for the rate of convergence of these methods is provided in [7] for the canonical Poisson problem, i.e. $\mathcal{L}^h u = f$. In general, convergence

properties of iterative methods are closely related to the eigenvalues and eigenvectors of A (as we will illustrate in Section 4).

We denote by $v^{(t)}$ the approximation obtained at the iteration step t. Remember that v represents a given approximation to the solution of the linear system $Au = b$ with exact solution u. Let us start by splitting the matrix A in the form,

$$A = D + L + U, \tag{20}$$

where D is the diagonal of A, and L and U are the strictly lower and upper triangular parts of A respectively. Isolating the diagonal part of A, we have

$$Du = -(L + U)u + b, \tag{21}$$

which suggests the Jacobi iterative scheme

$$\begin{aligned} v^{(t+1)} &= -D^{-1}(L + U)v^{(t)} + D^{-1}b \\ &= R_J\, v^{(t)} + D^{-1}b, \end{aligned} \tag{22}$$

where $R_J = -D^{-1}(L + U)$ is defined as the Jacobi iteration matrix.

A very simple but powerful modification to the previous method (see Algorithm 1), consists in considering the Jacobi iteration as an intermediate update $v^{(t+0.5)}$ and in defining $v^{(t+1)}$ as a weighted update in the direction of $(v^{(t+0.5)} - v^{(t)})$, i.e.

$$\begin{aligned} v^{(t+0.5)} &= R_J\, v^{(t)} + D^{-1}b \\ v^{(t+1)} &= v^{(t)} + \omega \left(v^{(t+0.5)} - v^{(t)}\right) \\ &= [(1 - \omega)I + \omega R_J]v^{(t)} + \omega D^{-1}b \\ &= R_\omega\, v^{(t)} + \omega D^{-1}b. \end{aligned} \tag{23}$$

The coefficient $\omega \in \mathbf{R}$ is a weight factor that needs to be set. I denotes the identity matrix and $R_\omega = [(1 - \omega)I + \omega R_J]$ is the *weighted* or *damped Jacobi* iteration matrix. (Notice that by setting $\omega = 1$ we recover the standard Jacobi scheme.)

The Gauss-Seidel method is very similar to the Jacobi method, but it uses the values of the new approximation $v_{ij}^{(t+1)}$ as soon as it becomes available. This can be expressed using the matrix decomposition described above as

$$(D + L)u = -U\, u + b, \tag{24}$$

which suggests the Gauss-Seidel iterative scheme (see Algorithm 2)

$$\begin{aligned} v^{(t+1)} &= -(D + L)^{-1}Uv^{(t)} + (D + L)^{-1}b \\ &= R_G\, v^{(t)} + (D + L)^{-1}b. \end{aligned} \tag{25}$$

The matrix $R_G = -(D + L)^{-1}U$ represents the Gauss-Seidel iteration matrix.

Algorithm 1: `WeightedJacobi(`A`,`b`,`x^0`,`ω`,`n`)`

Input : A and b // System $Au = b$
Input : x^0 // Initial guess of the solution u
Input : ω // Weight. (For $\omega = 1$ we get Jacobi iterative method)
Input : n // Number of iteration steps
Output: x^1 // Computed next approximation of u

$L = \text{lowerTriangular}(A)$ // Strictly lower triangular part of A
$U = \text{upperTriangular}(A)$ // Strictly upper triangular part of A
$D = \text{diagonal}(A)$ // Diagonal part of A
for $k = 1 : n$ **do**
  $x^1 = (1 - \omega)x^0 + \omega(D \backslash (-(L + U)x^0 + b))$
  $x^0 = x^1$
return x^1

Algorithm 2: `GaussSeidel(`A`,`b`,`x^0`,`n`)`

Input : A and b // System $Au = b$
Input : x^0 // Initial guess of the solution u
Input : n // Number of iteration steps
Output: x^1 // Computed next approximation of u

$L = \text{lowerTriangular}(A)$ // Strictly lower triangular part of A
$U = \text{upperTriangular}(A)$ // Strictly upper triangular part of A
$D = \text{diagonal}(A)$ // Diagonal part of A
for $k = 1 : n$ **do**
  $x^1 = (D + L) \backslash (-Ux^0 + b)$
  $x^0 = x^1$
return x^1

Frequency response. To understand the key advantages of multigrid approaches, it is important to review how different iterative methods behave depending on the frequency components of the error. Let us introduce a simple example to illustrate this. Assume we want to solve the Discrete Poisson problem presented in Equation (6), with $\Omega^h = \{1, \cdots, N\} \times \{1, \cdots, N\}$, Θ^h being an empty set (no Dirichlet interior points are imposed), and with data term $f(i, j) = 0 \; \forall \, i, j$. This problem can be written as

$$A^h u = b, \tag{26}$$

where $A^h = \mathcal{L}^h$ and $b = [0, \cdots, 0]^T$. It is obvious that $u(i, j) = 0 \; \forall i, j$ is a solution of this problem. The question we want to address now is the following: if we start with a cosine function as initial guess

$$v_\nu^0(i, j) = \cos\left(\frac{\nu\pi}{N} i\right), \tag{27}$$

how many iterations T are needed to converge to the true solution $v^T \approx 0$? The answer to this question will provide us intuition to understand how the evolution of the error is conditioned by its Fourier content. Figure 1 illustrates how the algebraic error evolves, starting from a sinusoidal seed v^0, when the Jacobi iterative method is used. Images in the upper row illustrate the evolution of the computed solution v^t when the initialization contains a low frequency sinusoidal, while bottom images show the evolution for a high frequency cosine function (for the same iteration numbers). As

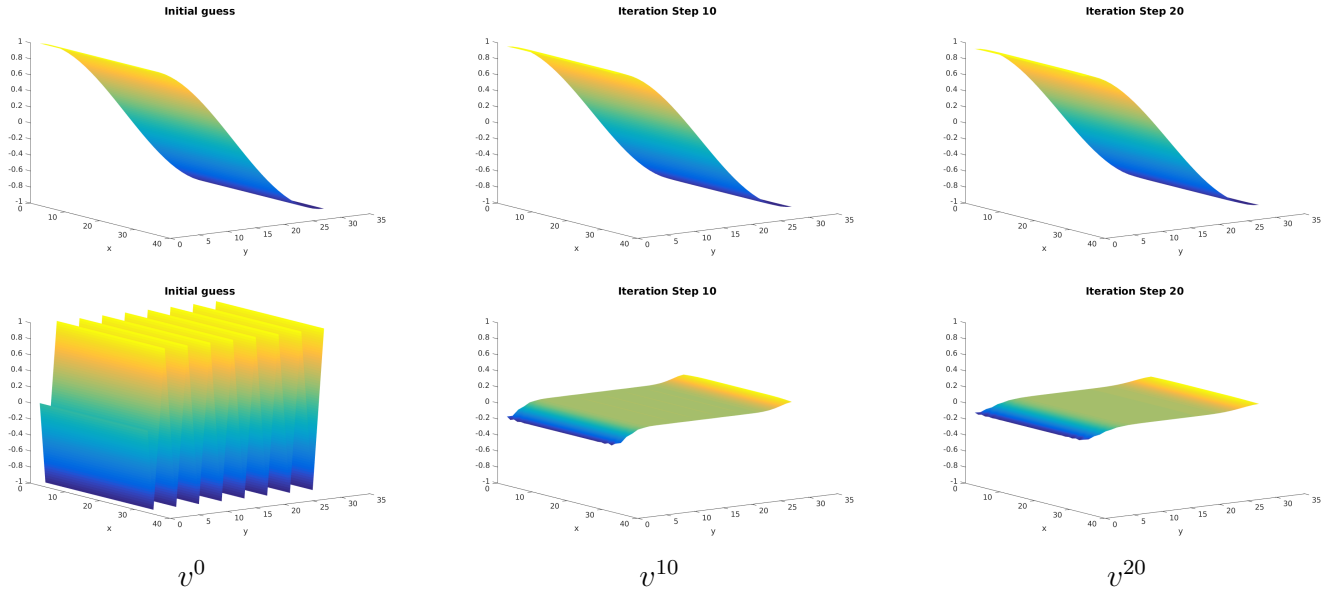

Figure 1: Evolution of the error for the homogeneous Poisson equation ($A = \mathcal{L}^h$, $b = 0$) when cosines of two different frequencies are used as seed v^0.

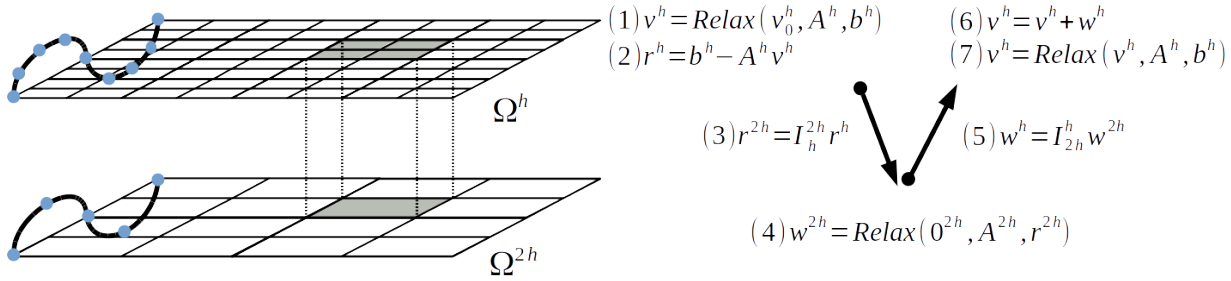

Figure 2: Diagram of a two grid scheme. Smooth components in a fine grid are observed as higher frequency components on coarser grids.

can be seen, simple iterative methods are very effective at reducing the high frequency components of the error, while several iteration steps are required to reduce low frequency components.

3.2 From One Grid to Two Grids

The key idea behind multigrid methods is to solve the problem using multiple discrete grids of different sizes. Then, a low frequency in the finer grid can be represented as a high frequency in a coarser grid, and hence, it can be effectively recovered using trivial iterative schemes (as we saw in the previous section).

A basic two-grid correction scheme (as illustrated in Figure 2) consists of seven main steps:

1. Perform n_1 iterations (of Gauss-Seidel or Jacobi iterative method) over the problem (9) $A^h u^h = b^h$ with a given initial seed v_0^h.

2. Compute the fine-grid residual $r^h = b^h - A^h v^h$.

3. Restrict the residual to the coarser grid $r^{2h} = I_h^{2h} r^h$.

4. Solve the problem $A^{2h} w^{2h} = r^{2h}$.

5. Interpolate the refined error to the fine grid $w^h = I_{2h}^h w^{2h}$.

6. Correct the fine grid approximation $v^h \leftarrow v^h + w^h$.

7. Perform n_2 iterations (of Gauss-Seidel or Jacobi iterative method) over the problem $A^h u^h = b^h$ starting from the refinement v^h computed in the previous step.

To implement the previous simple two grid correction scheme, we must define the interpolation/restriction operators I_{2h}^h / I_h^{2h} (Section 3.2.1) and we need to address how the problem in the coarse layer is defined: $A^{2h} w^{2h} = r^{2h}$ (Section 3.2.2). Before proceeding with these definitions, it is worth commenting why the seven steps described above are important for a successful definition of multigrid schemes. One of the key problems that need to be controlled when implementing multigrid methods is aliasing. As we described above, simple iterative methods are efficient only to reduce high frequency components of the error. Therefore, if spurious low frequency components are created because of aliasing, the implemented multigrid method will no longer be efficient. This explains why the residual r^h of the problem is transfered to the coarse grid (instead of v^h). The iterations performed in step 1, guarantee that the residual of the problem $A^h u^h = b^h$ does not contain high frequency components (as they were efficiently removed by Gauss-Seidel or Jacobi methods). Then it is safe to restrict the problem to the coarse grid $r^{2h} = I_h^{2h} r^h$ and no aliasing will be produced (as r^h is a smooth signal). Also step 7 can be explained with similar arguments. The reason why n_2 iterations of Gauss-Seidel (or Jacobi) method are performed after the solution in the fine grid is corrected, is to remove high frequency components of the error that may be introduced during the interpolation operation $w^h = I_{2h}^h w^{2h}$ (Appendix A provides a formal discussion on this subject).

3.2.1 Interpolation and Restriction Operators

To transfer information through the different grids it is necessary to define interpolation and restriction operators. The interpolation operator, denoted as I_{2h}^h, is a transformation $I_{2h}^h : \Omega^{2h} \to \Omega^h$ that interpolates a discrete function v^{2h} defined in the coarse grid into the fine grid $v^h = I_{2h}^h v^{2h}$. On the other hand, the restriction operator does the inverse transformation mapping a given discrete function in the fine grid to the coarse grid, i.e. $I_h^{2h} : \Omega^h \to \Omega^{2h}$ with $v^{2h} = I_h^{2h} v^h$.

The proper definition of interpolation and restriction operators is critical to achieve efficient multigrid implementations. In particular, there are three basic properties that need to be considered:

1. The order of the interpolation and the restriction operator should be higher or equal to the order of the differential equation we aim to solve (see [8] and Appendix A for a detailed discussion about the operators order).

2. The interpolation I_{2h}^h and the restriction I_h^{2h} should verify (see Appendix B and reference [4, pg. 185]) the condition
$$I_h^{2h} = c\,(I_{2h}^h)^T \qquad c \in \mathbf{R}. \tag{28}$$

3. Interpolation and restriction should be compatible with the boundary conditions at $\partial\Omega$.

A restriction operator I_h^{2h} can be expressed without lost of generality as

$$I_h^{2h}(v^h) = \tilde{I}_h^{2h}(K^h * v^h), \tag{29}$$

where $\tilde{I}_h^{2h}$ denotes a basic sub-sampling operation, i.e. $\tilde{I}_h^{2h}(v^h)(k) = v^{2h}(k) = v(k(2h)) = v^h(2k)$; and K^h is a smoothing kernel (see Appendix A and reference [8] for a more detailed discussion).

The basic sub-sampling operation just selects one over two samples on each axis of the discrete grid. For the case of a two dimensional discrete domain, we can arbitrarily start the sampling process

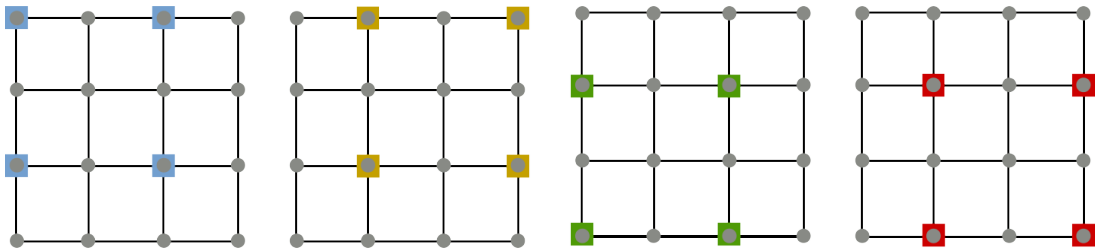

Figure 3: Illustration of the pixels selected by the sub-sampling operators (from left to right) $\tilde{I1}_h^{2h}$, $\tilde{I2}_h^{2h}$, $\tilde{I3}_h^{2h}$ and $\tilde{I4}_h^{2h}$.

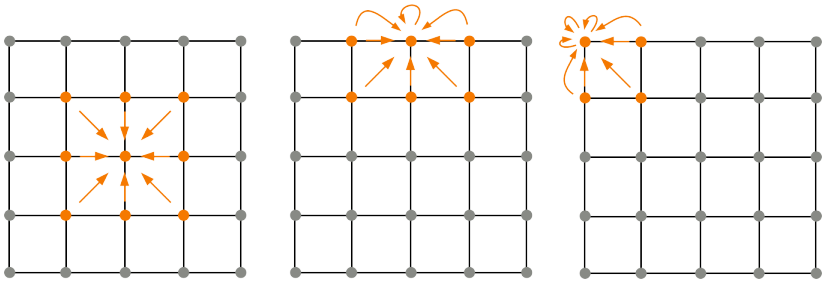

Figure 4: Illustration of the definition of the discrete convolution on the boundaries of the discrete domain.

at the pixel position $(1,1)$, $(1,2)$, $(2,1)$ and $(2,2)$ (referred as $\tilde{I1}_h^{2h}$, $\tilde{I2}_h^{2h}$, $\tilde{I3}_h^{2h}$ and $\tilde{I4}_h^{2h}$ respectively) as illustrated in Figure 3.

The Poisson equation consists of a second order partial differential equation, which implies that the restriction and interpolation should be at least second order operators [8]. To that end, the well known full-weighting stencil was used in the present work,

$$K^h = \frac{1}{4}\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}. \tag{30}$$

For each discrete position $(i,j) \in \Omega^h$, the discrete convolution $K^h * v^h$ requires the value of v^h at the eight neighbors $(i \pm 1, j \pm 1)$. This cannot be done at the boundary of the image domain, e.g., $i = 1$, $i = H$, $j = 1$ or $j = W$). In those cases, Neumann boundary conditions are applied which is equivalent to use the closest value of v^h inside the domain Ω^h as it is illustrated in Figure 4.

Observing that Equation (29) is linear in the elements of the matrix v^h, it can be expressed as $I_h^{2h} v^h$ where I_h^{2h} is a $HW \times HW$ sparse matrix, and v^h represents the column vector $HW \times 1$ obtained by writing the two dimensional discrete signal in lexicographical order. The definition of the sparse matrices $\tilde{Ii}_h^{2h}$ is straightforward. Algorithm 3 describes how to define the kernel convolution as a matrix multiplication $K * v^h = G\, v^h$ (incorporating the boundary condition at $\partial\Omega^h$). Then, the sparse restriction matrix can be defined as

$$I_h^{2h} = \frac{1}{4}\left(\tilde{I1}_h^{2h} + \tilde{I2}_h^{2h} + \tilde{I3}_h^{2h} + \tilde{I4}_h^{2h}\right) G, \tag{31}$$

and the interpolation as

$$I_{2h}^{h} = 4\left(I_h^{2h}\right)^T. \tag{32}$$

The multiplicative factor included in Equation (32) was set so that the energy (norm) of discrete functions remains invariant when restriction and projection are applied consecutively.

Algorithm 3: `ComputeG` (Matrix version of the convolution kernel)

Input : H // Height of the discrete domain Ω^h
Input : W // Width of the discrete domain Ω^h
Output: $HW \times HW$ matrix G // matrix form of the full-weighting convolution

```
// Prepare sparse matrix indices
sh = [H,W]
[py, px] = ind2sub( sh, [1:HW])          // get pixel indices for the output image
i = sub2ind(sh , py, px)                 // get rows indices of the operator (y,x) (identity)
G = sparse(HW, HW)                       // initialize

// Vector containing the offsets and weights of the stencil
alldxdy = [[-1,-1, 1/4]; [1, 0, 1/2]; [-1, 1, 1/4];
           [0, -1, 1/2]; [0, 0, 1];    [0, 1, 1/2];
           [1, -1, 1/4]; [-1, 0, 1/2]; [1, 1, 1/4]];

// Loop over the vector combining the weights of the 9 positions of the stencil
for t =1:9 do
    dy = alldxdy(t, 1)
    dx = alldxdy(t, 2)
    val = alldxdy(t, 3)

    // Get corresponding input columns mirroring at the boundaries ie
       max(min(ycoord, H), 1)
    j = sub2ind(sh, max(min(py + dy, H), 1), max(min(px + dx, W), 1))
    G = G + sparse(i, j, val, HW, HW)
G = 1/4 G                                // Normalization factor (See stencil def.  Eq. (30))
return G
```

The previous definitions chosen for the restriction and interpolation operators, presented in Equations (31) - (32), average the four different alternatives for defining the basic sub-sampling matrices $\tilde{I1}_h^{2h}$-$\tilde{I4}_h^{2h}$ illustrated in Figure 3. This is an arbitrary choice and one can also choose any of the basic sub-sampling operators. We found in practice that the average of the four basic sub-sampling strategies reduces boundary effects. In particular, it is easy to verify that the proposed definitions keep invariant any constant matrix. More precisely, given a constant $H \times W$ matrix C, the equation $C = I_{2h}^h I_h^{2h} C$ holds for every pixel (even those at the boundary of the image).

3.2.2 Definition of the Problem in the Coarse Grid

The fourth step of the two-grid approach described in Section 3.2 requires to solve the coarse problem $A^{2h}w^{2h} = r^{2h}$. Restriction and prolongation procedures were already addressed, which allows us to precisely define r^{2h} from r^h and vice-versa. However, a precise definition of A^{2h} is still necessary and will be addressed in the following.

The solution w^{2h} of equation $A^{2h}w^{2h} = r^{2h}$ should provide an update to the current approximation v^h such that the error (in the fine grid) is minimized, i.e.

$$w^{2h} = \arg\min_{\tilde{w}^{2h}} \|u^h - (v^h + I_{h2}^h \tilde{w}^{2h})\|. \tag{33}$$

Proposition 2. *Given the problem in a fine grid $A^h u^h = b^h$ and a current approximation of its solution v^h, the optimal update w^{2h} on a coarser grid can be obtained as the solution of the coarse linear problem $A^{2h}w^{2h} = b^{2h}$ where $A^{2h} = (I_{2h}^h)^T A^h I_{2h}^h$ and $b^{2h} = (I_{2h}^h)^T (b^h - A^h v^h)$.*

Algorithm 4: v_1 = MG(A,b,v_0,H,W,μ,n_1,n_2)

Input : A, b // Matrix and data term of the problem $Ax = b$
Input : v_0 // Initial guess (seed)
Input : H, W // Height and Width of the discrete domain Ω^h
Input : μ // Number of iterative calls per layer
Input : n_1 // Number of pre-smoothing steps
Input : n_2 // Number of post-smoothing steps
Output: v_1 // updated approximation of the solution

// (0) Initialization
$[I_{2h}^h,\ I_h^{2h}] = \text{ComputeRestrictionAndInterpolationOp}(H,W)$
// (1) Pre-smoothing Relaxation
$v_1 = \text{GaussSeidel}(A,b,v_0,n_1)$ // n_1 iterations of GaussSeidel algorithm
// (2) Compute the residual
$r^h = b - A\,v_1$
// (3) Restrict the residual to a coarser layer
$r^{2h} = I_h^{2h}\,r^h$
// (4) Solve the problem in the coarse layer
$w^{2h} = 0$ // initial guess in the coarse layer
$A^{2h} = I_h^{2h} A^h I_{2h}^h$ // Define the problem in the coarser layer
for i = *1:*μ **do**
  $w^{2h} \leftarrow \text{MG}(A^{2h},r^{2h},w^{2h},H/2,W/2,\mu)$
// (5) Interpolate the correction function to the fine grid
$w^h = I_{2h}^h w^{2h}$
// (6) Apply the correction obtained from the coarse layers
$v_1 = v_1 + w^h$
// (7) Post-smoothing relaxation
$v_1 = \text{GaussSeidel}(A,b,v_1,n_2)$ // Reduce high-freq distortion due to the interpolation
return v_1

Proposition 2 is proved in Appendix B using the important propertied described above: (i) that A^h is a self-adjoint matrix and (ii) that $I_{2h}^h \propto (I_h^{2h})^T$. This proposition is of crucial importance. Firstly, because it provides a precise definition of A^{2h} in terms of I_{2h}^h and A^h; and secondly, because it shows that the coarse version of the residual ($r^{2h} = (I_{2h}^h)^T(b^h - A^h v^h)$) plays the role of the data term in the coarse layer.

The two-grid scheme now is fully defined and can be implemented with no ambiguity, the next step is of course to generalize these ideas to multiple grids.

3.3 Multigrid Approach

Recalling the seven main steps that described a two-grid correction scheme, the fourth step consisted of solving the linear problem $A^{2h}w^{2h} = r^{2h}$ (in the coarse grid). This problem has exactly the same structure as the original problem (i.e. is a self-adjoint linear problem), thus, it can be solved using a two-grid approach. In other words, when the linear problem (at step 4) needs to be solved, a two-grid scheme can be recursively applied leading to an even coarser grid and so forth.

The previous ideas motivate the definition of a multigrid solver as described in Algorithm 4. The parameter μ sets the number of times the algorithm uses the coarser grids to improve the solution obtained in the current grid. Even though the parameter μ can be set to an arbitrary integer number,

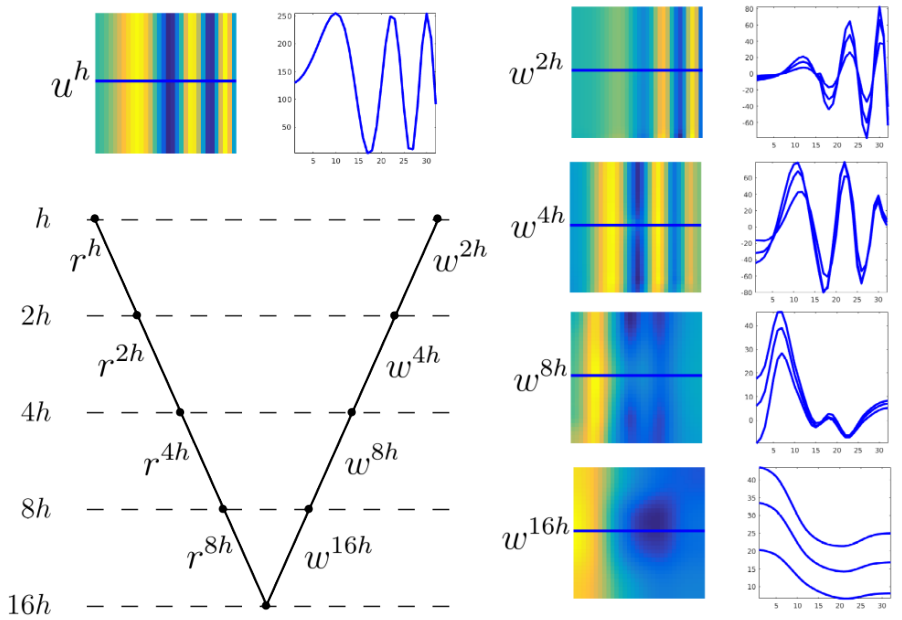

Figure 5: Illustration of V-scheme for a 32×32 discrete problem. On the left we see the ground truth solution and the diagram of the multigrid computations. On the right are shown the corrections $w^{2^i h}$ computed at each scale i. Note that different grids are able to capture different components of the solution.

$\mu = 1$ and $\mu = 2$ are the most common choices. The first case ($\mu = 1$) is called a V-Scheme and when $\mu = 2$ a W-scheme. Figure 5 illustrates the V-scheme with a simple example. The fine grid was defined as $\Omega^h = [1 \cdots 32] \times [1 \cdots 32]$ and the ground truth is $u^h(x, y) = (1 + sin(x^2/2W))$. The Laplacian of this function was calculated from the ground truth solution and Poisson equation solved (starting from $v_0(x, y) = 0$ for all $(x, y) \in \Omega^h$). This first illustrative example shows how different grids (scales) are able to capture different components of the solution. Then the combination of these orthogonal spectral components allows to recover the global solution in a efficient manner.

4 Experiments and Discussions

The previous section presented some basic iterative and multigrid numerical solvers that can be used to solve discrete Poisson problems. The goal of this section is to present a set of experiments that will help us understand interesting properties of the problems and methods explained above. In particular, we aim at answering: How different the solutions of different discrete formulations of the Poisson equation are? How does the topology of the set Θ^h impact on the efficiency of numerical solvers? And fundamentally, can multigrid methods converge to a solution in a time that is linear with the number of unknowns?

Changes in the solutions due to differences in the formulation. To illustrate some of the differences between the extended and self-adjoint extended formulation presented in Section 2.1 let us consider the following example. When photographers capture images of scenes with a very large illumination range, a common problem is to end up with photographies that have poor local contrast (either in the bright or dark portions of the image). For example, Figure 6(a) illustrates a challenging scene that presents a high luminance range which cannot be successfully captured by a camera sensor. To improve the local contrast in the dark portions of the image we define a Poisson-like problem by: setting the domain Θ as the set of bright pixels (b), the Laplacian term f as an amplified version of the original Laplacian (c), and we preserve the original values of the image

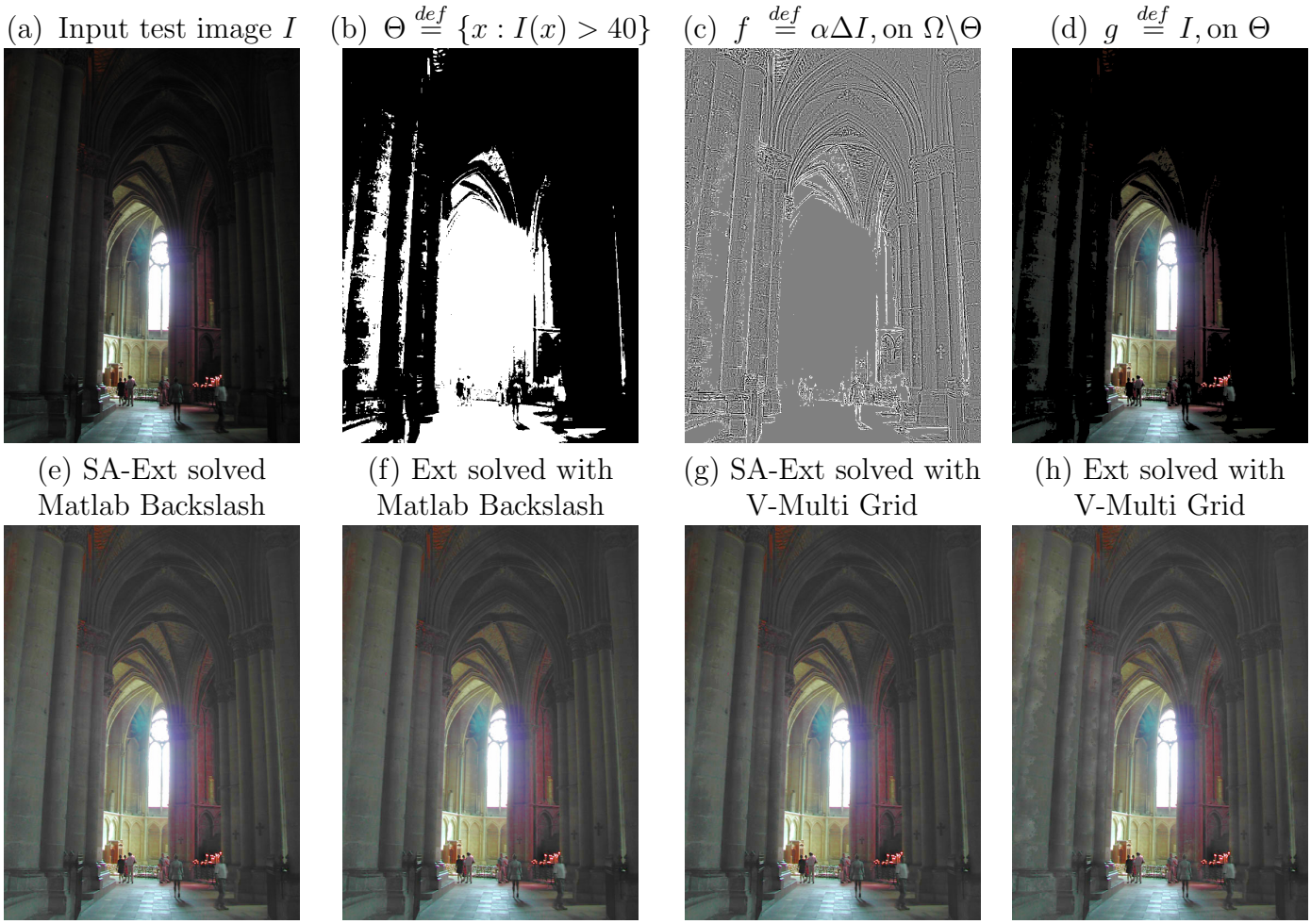

Figure 6: The first row illustrates: (a) A test input image, (b) the domain Θ, (c) the Laplacian term f, and finally the boundary Dirichlet conditions g. In this example, the Laplacian term was defined by amplifying the Laplacian on the dark areas of the original image by a factor $\alpha = 2$ (with the objective of amplifying the local contrast in those regions). The second row illustrates the solution obtained for Self-Adjoint Extended (e) and the Extended formulation (f) of the discrete Poisson problem defined above, as the numerical solver matlab backslash is used in this first two examples. Images (g) and (h) illustrate the solution to the same system of sparse linear equations but when a multigrid V-scheme iterative solver is used.

in the bright areas (d) (Dirichlet conditions). The second row of Figure 6 illustrates the solution obtained for the Self-Adjoint Extended (e-g) and the Extended formulation (f-h). Matlab backslash is used as the numerical solver in the first two examples (e-f). On the other hand, images (g) and (h) illustrate the solution to the same system of sparse linear equations but when a multigrid V-scheme iterative solver is used. The solutions (e), (f) and (g) are equivalent (we numerically checked that these images are identical). This was expected as the Self-Adjoint and the Extended discrete formulations are mathematically equivalent to each other. Therefore given an arbitrary numerical solver, *if it converges* to the solution there will be no difference between the Self-Adjoint and the Extended formulations. Despite the previous observation, for a single multigrid iteration different solutions are obtained when the Self-Adjoint or the Extended formulation are chosen. In particular we can see in this example (see the results (g) and (h)) that after a single V iteration, convergence is achieved only for the self-adjoint formulation of the problem (see Appendix B).

Impact of the topology of the boundary conditions Θ^h on the solver efficiency. As we showed in the previous example, Poisson like problems can be solved efficiently by solving a sparse

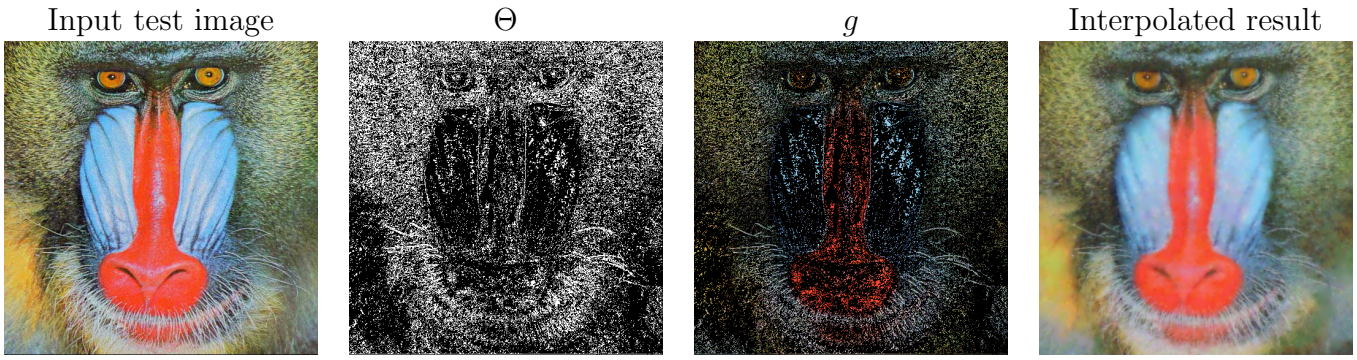

Figure 7: Interpolation example solving a homogeneous Poisson equation. In this illustrative example, approximately 30% of the pixels values were kept (in particular those associated to edges) and the interpolation obtained by solving the homogeneous Poisson equation.

linear system of equations $Au = b$. The effectiveness of different numerical methods depends on the structure of the matrix A. From now on, we will analyze the structure of the matrix of the problem associated to the Self-Adjoint Extended problem. As A is symmetric and positive definite, we can decompose it into a basis of orthonormal eigenvectors ϕ_k with associated (strictly) positive real values λ_k. The data term can be written as $b = \sum_k c_k \phi_k$ where c_k is a set of scalar constant coefficients. Then, it is easy to prove that u can be obtained as $u = \sum_k \frac{c_k}{\lambda_k} \phi_k$. When the canonical Poisson equation is considered, i.e. homogeneous Neumann/Dirichlet boundary conditions are imposed in the exterior of a regular rectangular domain, the eigenvectors ϕ_k become cosine/sine functions and the decomposition expressed above becomes a cosine/sine Fourier decomposition [4]. In addition, the smaller the eigenvalue λ_k, the more critical the weight of the k^{th} eigenvector $c_k/\lambda_k \phi_k$.

Let us illustrate the previous ideas with a simple example. Consider a test image I that needs to be compressed e.g. by keeping only those pixels associated to edges of the image (similar to the method proposed in [10]). To that end, we can define the set Θ as the set of pixels x_i for which the gradient is higher than a certain threshold t. The Dirichlet data term g may be set as the values of the image I at the given positions x_i, and finally, a null Laplacian data term $f = 0$ may be used for interpolation. Figure 7 shows an example of an input image, the pixels kept and the corresponding data term g. Numerically this problem can be stated as the Poisson-like system of equations (13) (with the data terms specified above).

As described in previous sections, the discrete formulation of the problem leads to a sparse linear system of equations $Au = b$. The structure of A depends on the shape of the set Θ, which determines the eigenvectors of A. For example, Figure 8 illustrates four different sets Θ and the corresponding eigenvectors associated to the smaller eigenvalues of the matrix A.

Figure 8 shows that when more anchorage points $g(x)$ for $x \in \Theta$ are provided, the eigenvectors of the problem become sharper (i.e. their Fourier transforms contain higher frequency content). This is a very important issue as the effectiveness of different iterative methods strongly depends on the frequency content of the error vector at each iteration step. Taking this into account, and recalling the properties in the Fourier domain of the iterative methods presented in Section 3 we can conclude that the more points of the solution we set, the more effective simple relaxation methods will become. Furthermore, two particular cases are of particular interest. First, when the set Θ is empty; in that case the problem cannot be uniquely solved as solutions are defined up to a global constant. This explains why in that case we obtain a null eigenvalue associated to a constant eigenvector (as illustrated in the first row of Figure 8). A second interesting case is when the value on a single pixel is set (i.e. $\Theta = \{x_1\}$). In this case, the solution is unique, while the eigenvectors of A associated to the smaller eigenvalues are smooth. This implies that setting a single pixel of the solution guarantees that the solution of the problem is unique while leading to the numerically more challenging problem.

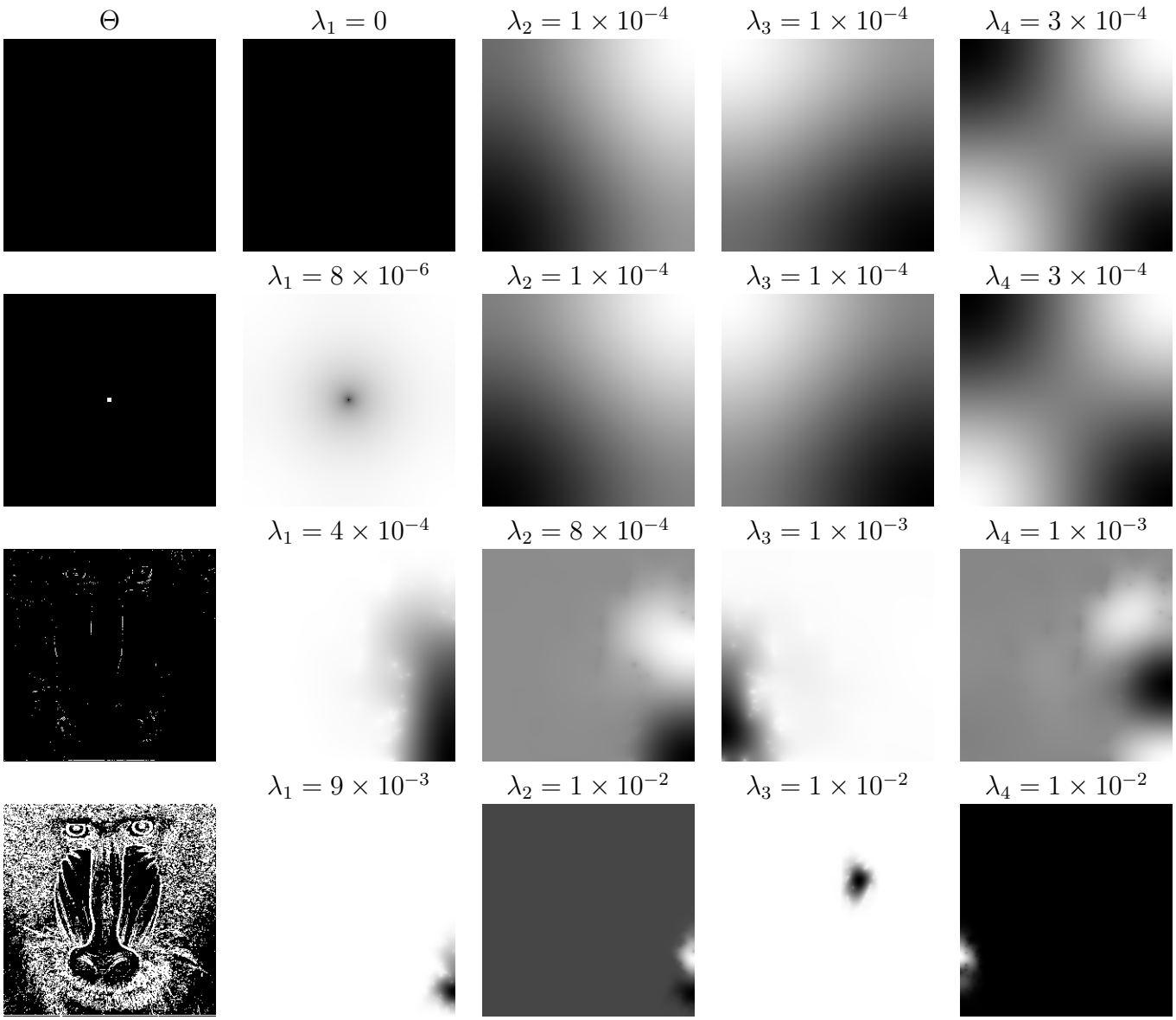

Figure 8: The first column shows different sets in which the value of the solution is provided, while the other four columns illustrate the four eigenvectors associated to the smaller eigenvalues of the matrix of the problem: A.

Evaluating multigrid convergence rate. In a third round of experiments, we compared multigrid iterative method with: Gauss-Seidel, Conjugate Gradient, Preconditioned Conjugate Gradient [1], Bi-Conjugate Gradient, Symmetric LQ [12], Generalized minimum residual [15], Transpose-free Quasi-minimal Residual [1], and Bi-Conjugate Gradient Stabilized [18] methods. We tested our own implementation of Multigrid, Gauss-Seidel and Conjugate-Gradient methods and used implementations available in Matlab[2] in the rest of the cases. Given a test input image I we solved the discrete Poisson problem given by Equation (13) with $f = I$, $g = \Delta^h I$ and Θ obtained by randomly sampling pixels of the discrete domain. Intuitively, this problem consists in recovering an image I from its partial derivatives ΔI given a set of image points on Θ.

Figure 9 shows the evolution of the norm of the error $\|u - v(t)\|$ along the iterations for the selected set of iterative solvers. As in the previous experiment, we observe that the less pixels the domain Θ sets, the more challenging the problem becomes. In addition, we observe that the multigrid approach achieves efficient convergence rates even in the extreme case that Θ contains a single pixel.

[2]Matlab version R2016b

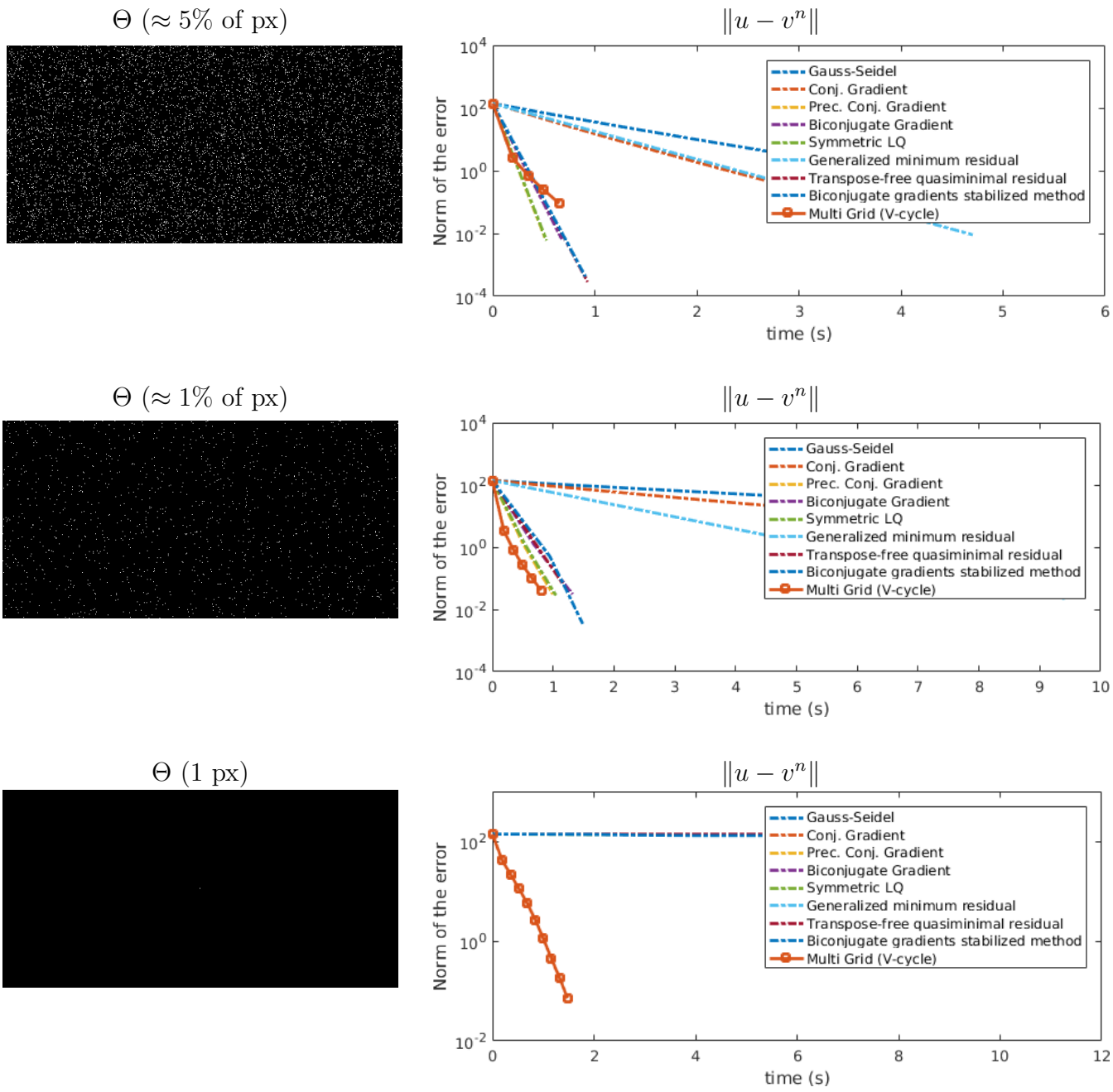

Figure 9: Evolution of the norm of the error as a function of time for different iterative methods and boundary conditions. The size of the input image was 256×512.

Complementing the previous experiment, we also tested how the time required for convergence depended on the size of the problem (i.e. the number of pixels in the domain Ω). Figure 10 shows the time required to converge to a solution with an absolute mean error below 1 gray level (input images are in the range $[0,\ 255]$). The resolution of input images was increased exponentially. Multigrid V and W schemes were tested and compared with Matlab "backslash"[3] routine. Matlab "backslash" solves large sparse linear problems in a smart and efficient manner by analyzing the matrix associated to the problem. Then, the actual solver executed is chosen to exploit the properties of the matrix A. In general, we observed that for modest problem sizes (i.e. less or equal to one million pixels) backslash was the fastest way to solve the problem. However, as the size of the domain increases,

[3]In order to have more meaningful comparisons, we executed Matlab in a single processor thread.

multigrid schemes become more competitive. Multigrid methods are definitely the right choice when the number of pixels is over 2 millions of pixels. In addition, Multigrid schemes can solve Poisson-like problems in linear time as can be observed in Figure 10. This is a key feature of multigrid approaches that makes them particularly useful to solve large PDE equations.

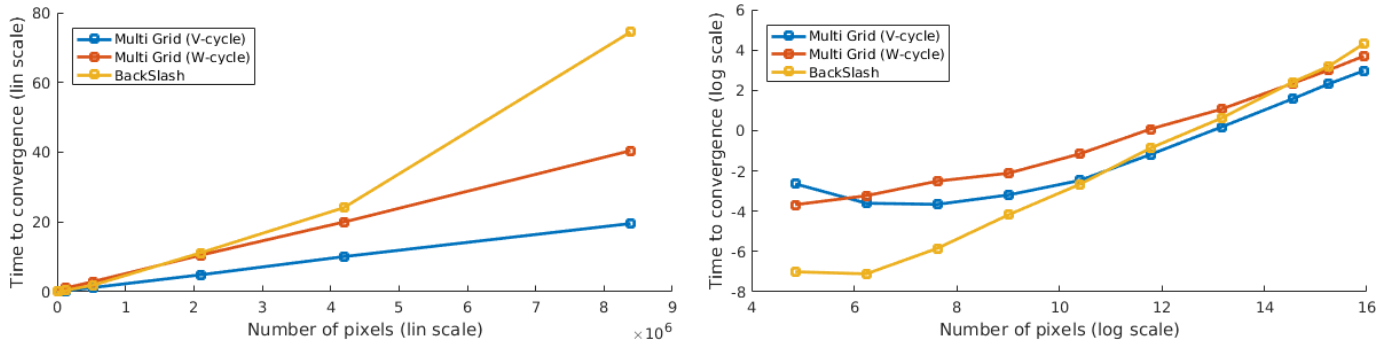

Figure 10: Execution time versus domain size (i.e. number of pixels). Left plot shows the time to convergence (in linear scale) for a V and W multigrid schemes compared with Matlab Backslash. Right plot illustrates the same results in a logarithmic scale.

Finally, we evaluated how multigrid parameters affects its performance. Figure 11 shows the time for convergence for different number of pre- and post-smoothing relaxation steps for both V and W schemes. As can be seen, a few post and pre relaxation steps provide efficient multigrid schemes. In particular, we found that zero pre-relaxation steps and one post-relaxation achieved the faster convergence.

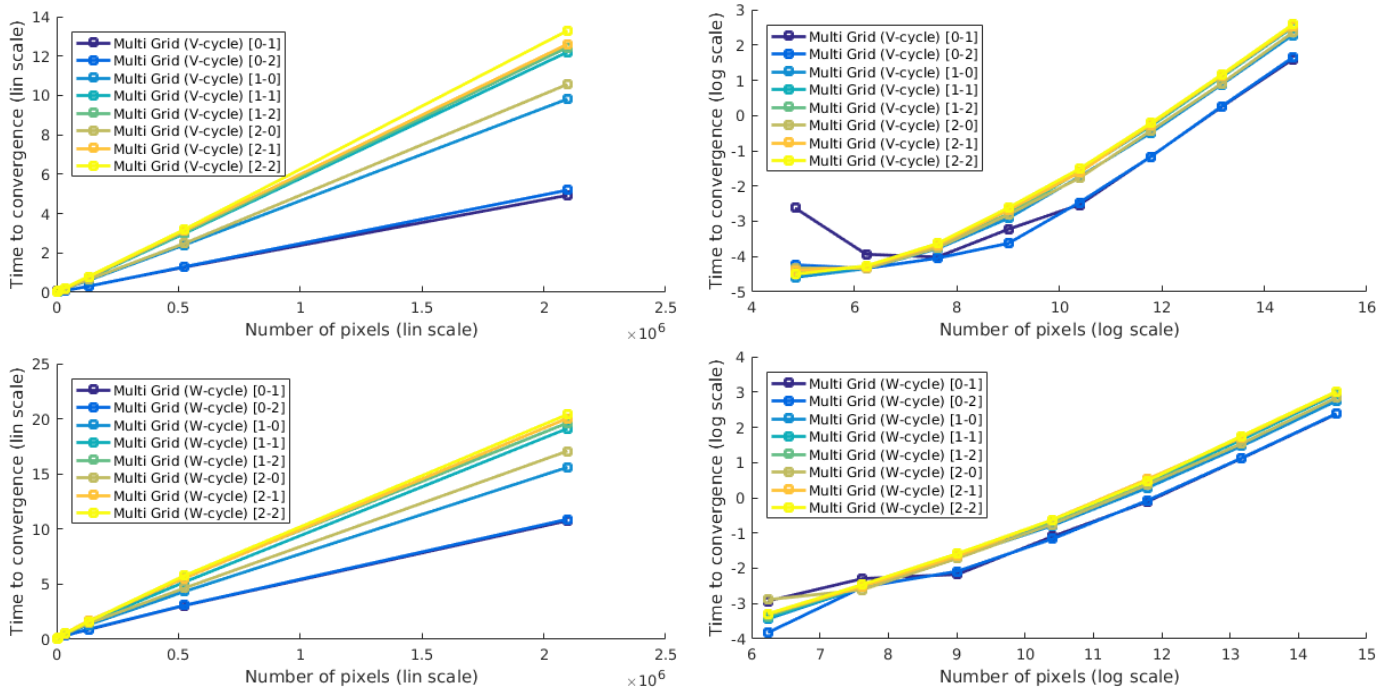

Figure 11: Execution time versus domain size for different values of pre- and post-relaxation steps. The first number in the legend of the plots corresponds to the number of pre-smoothing G-S relaxation steps and the second number to the number of post-smoothing relaxation steps. Results are shown in both linear and logarithmic scale.

5 Conclusions

We described in depth, analyzed and compared iterative numerical methods to solve Poisson-like problems. The focus of this work was on the mathematical properties of different formulations of the problem and how they impact multigrid ideas. In addition, some of the ideas presented by Mainberger et al. [10] were reviewed and extended. A discrete self-adjoint extended formulation of the Poisson equation was described, which allowed to impose arbitrary Dirichlet boundary conditions in arbitrary points of the domain. Moreover, we studied under which conditions the problem is well-posed and presented some experimental validation examples. The obtained results show how the characteristics of the set of boundary conditions affect the convergence properties of different iterative methods. Then, we compared iterative numerical solvers and studied the relation between the convergence time and the size of the grid. We verified that convergence of multigrid methods can be achieved in linear time. In addition, multigrid V and W schemes were tested with different parameters and we observed that very few pre- and post-smoothing relaxation steps provided optimal and stable convergence rates.

Acknowledgment

The authors would like to thank Jean-Michel Morel for fruitful discussions, Juan Francisco Garamendi for pointing us to useful references, and Enric Meinhardt-Llopis for his interesting suggestions and ideas which substantially improved this work. Work partly founded by CSIC and PEDECIBA (Uruguay), the Office of Naval research by grant N00014-17-1-2552 and ANR-DGA project ANR-12-ASTR-0035.

Image Credits

 NASA[4]

 Standard test image.

Appendix A: Restriction and Prolongation Operator Properties

Given a continuous function $u \in \mathcal{L}^2(\mathbf{R})$ the Fourier Transform can be defined as

$$\hat{u}(\nu) = \int_{-\infty}^{+\infty} u(x) e^{-i2\pi\nu x} dx. \tag{34}$$

From now on we will assume that u has a limited spectrum band. A continuous version of the discrete sampled function $u^h(k) \stackrel{def}{=} u(kh)$ can be written as the distribution,

$$\underline{u}(x) = u(x) \cdot \sum_{k \in \mathbf{Z}} \delta(x - kh), \tag{35}$$

where h corresponds to the sampling distance (equivalent to the pixel size for images).

[4] http://dragon.larc.nasa.gov/retinex/

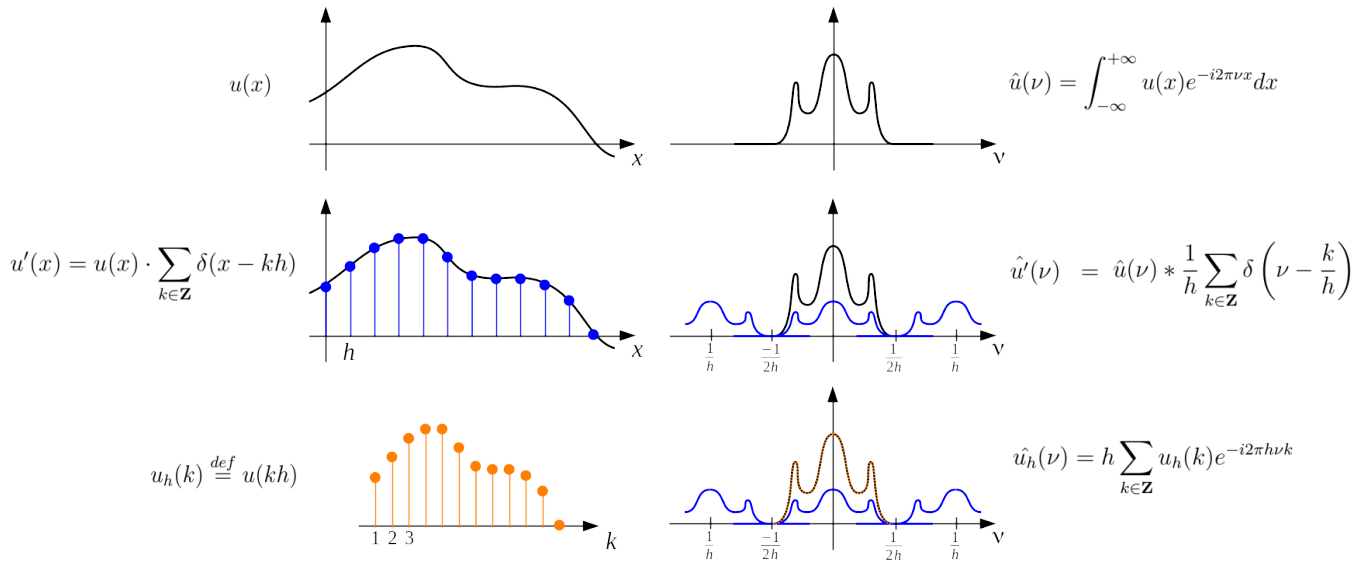

Figure 12: Illustration of the relation between the continuous and the sampled function in the spatial and frequency domain.

The Fourier transform of $\underline{u}(x)$ can be written using standard Fourier Transform properties [16] as

$$\begin{aligned} \hat{\underline{u}}(\nu) &= \hat{u}(\nu) * \widehat{\sum_{k\in\mathbf{Z}} \delta(x - kh)} \\ &= \hat{u}(\nu) * \frac{1}{h}\sum_{k\in\mathbf{Z}} \delta\left(\nu - \frac{k}{h}\right), \end{aligned} \tag{36}$$

which can also be related to the Fourier Transform of the discrete signal $u^h(k)$ by

$$\begin{aligned} \hat{\underline{u}}(\nu) &= \int_{-\infty}^{+\infty} u(x) \cdot \sum_{k\in\mathbf{Z}} \delta(x - kh)e^{-i2\pi\nu x}dx \\ &= \sum_{k\in\mathbf{Z}} \int_{-\infty}^{+\infty} u(x)\delta(x - kh)e^{-i2\pi\nu x}dx \\ &= \sum_{k\in\mathbf{Z}} u(hk)e^{-i2\pi\nu hk}. \end{aligned} \tag{37}$$

The relation between equations (36) - (37) inspire the following definition for the Fourier Transform of the discrete signal $u^h(k)$

$$\widehat{u^h}(\nu) = h\sum_{k\in\mathbf{Z}} u^h(k)e^{-i2\pi h\nu k}. \tag{38}$$

Figure 12 illustrates some of the relations introduced above between a continuous signal $u(x)$ and it discrete counterpart $u^h(k)$ both on the spatial an Fourier domain. Equation (36) shows that the spectrum of the continuous signal is spread along the frequencies as the convolution with the Dirac distribution produces a periodic replication of the original spectrum. If the support of $\hat{u}(\nu)$ is included in the interval $[-1/2h\ 1/2h]$ no aliasing is introduced and $\widehat{u^h}(\nu) = \hat{u}(\nu)$ for $\nu \in [-1/2h\ 1/2h]$ (Nyquist-Shannon sampling theorem).

We can write the basic interpolation and restriction operators as

$$\tilde{I}_h^{2h}(u^h(k)) = u^{2h}(k) = u(k(2h)) = u^h(2k), \tag{39}$$

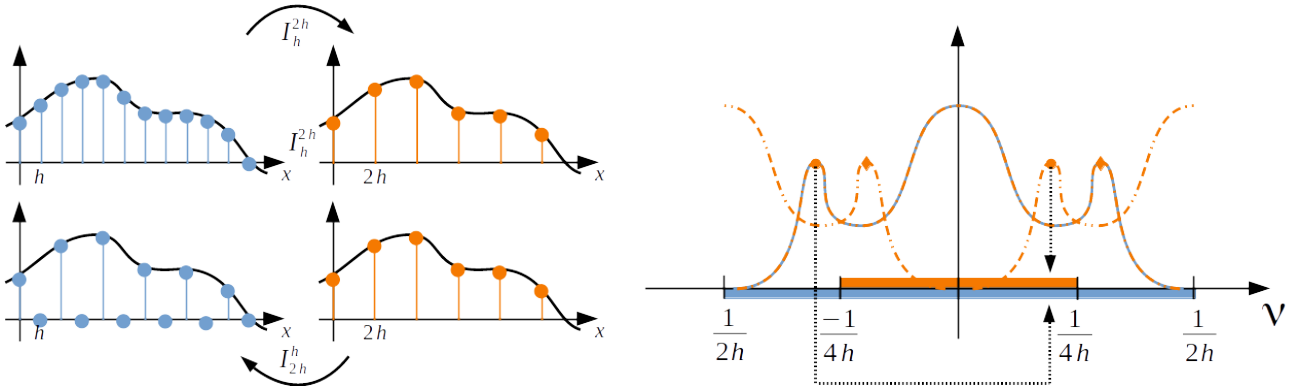

Figure 13: Illustration of basic interpolation and restriction operators.

$$\tilde{I}_{2h}^h(u^{2h}(k)) = u^h(k) = \begin{cases} u^{2h}(k/2) & \text{if } k \text{ is even} \\ 0 & \text{otherwise.} \end{cases} \tag{40}$$

It is easy to see that this basic interpolation and restriction operators wrap the frequency space to the intervals $[-1/2h\ 1/2h]$ and $[-1/2(2h)\ 1/2(2h)]$ respectively. In this process frequencies $\{\nu, \nu \pm (1/2h)\}$ in the interval $[-1/2h\ 1/2h]$ are mapped to the same frequency $\nu_0 \in [-1/2(2h)\ 1/2(2h)]$ as it is illustrated in Figure 13. In order to control aliasing effects during restriction and to avoid the artificial generation of high frequencies during the process of interpolation, more regular interpolation and restriction operators can be defined as

$$\begin{aligned} I_h^{2h}(u^h) &= \tilde{I}_h^{2h}(K^h * u^h) \\ I_{2h}^h(u^{2h}) &= c\, K^h * \tilde{I}_{2h}^h(u^{2h}). \end{aligned} \tag{41}$$

K^h represents a smoothing kernel, and c a scalar constant. The effect of the smoothing operation can be seen as a filtering step on the Fourier domain, which helps reducing the amplitude of the components of u_h (and $\tilde{I}_{2h}^h(u^{2h})$) on the spectral band $[-1/2h,\ -1/4h] \cup [1/4h,\ 1/2h]$.

The order of the kernel K_h should be at least the order of the partial equation we are willing to solve. For example, the Poisson equation is a second order partial differential equation and therefore at least a kernel of order two is required.

For the case of two dimensional signals, the full weighting (or bilinear) stencil

$$K = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix} \tag{42}$$

has order 2 (see reference [8] for a detailed discussion) and will be the kernel used in the present work.

Appendix B: Optimal Problem in a Coarser Grid

Given the discrete linear numerical problem

$$A^h u^h = b^h, \tag{43}$$

we will assume that A^h satisfies the following properties:

- $\langle A^h v_1^h, v_2^h \rangle = \langle v_1^h, A^h v_2^h \rangle$ for all $v_{1,2}^h \in \Omega^h$ (self-adjoint operator).

- $\langle A^h v^h, v^h \rangle > 0$ for any nonzero $v^h \in \Omega^h$.

Proposition 3. *Solving Equation (43) is equivalent to the minimization problem,*

$$u^h = \underset{u^h \in \Omega^h}{\arg\min} \frac{1}{2}\langle A^h u^h, u^h \rangle - \langle b^h, u^h \rangle \overset{def}{=} \underset{u^h \in \Omega^h}{\arg\min} F^h(u^h). \tag{44}$$

The previous proposition is a classic result in linear algebra, nevertheless we include its proof for completeness and because we shall reuse its formalism. Consider the value of $F^h(u^h + v^h)$ for an arbitrary vector $v^h \in \Omega^h$,

$$\begin{aligned} F^h(u^h + v^h) &= \frac{1}{2}\langle A^h(u^h + v^h), (u^h + v^h) \rangle - \langle b^h, u^h + v^h \rangle \\ &= \frac{1}{2}\left(\langle A^h u^h, u^h \rangle + \langle A^h u^h, v^h \rangle + \langle A^h v^h, u^h \rangle + \langle A^h v^h, v^h \rangle\right) - \langle b^h, u^h \rangle - \langle b^h, v^h \rangle \\ &= F^h(u^h) + \frac{1}{2}\left(2\langle A^h u^h, v^h \rangle + \langle A^h v^h, v^h \rangle\right) - \langle b^h, v^h \rangle, \end{aligned} \tag{45}$$

where the self-adjoint property of matrix A^h was used to simplify the middle term in the last step of Equation (45). Equation (45) can be expressed as

$$\begin{aligned} F^h(u^h + v^h) &= F^h(u^h) + \langle A^h u^h, v^h \rangle - \langle b^h, v^h \rangle + \frac{1}{2}\langle A^h v^h, v^h \rangle \\ &= F^h(u^h) + \langle A^h u^h - b^h, v^h \rangle + \frac{1}{2}\underbrace{\langle A^h v^h, v^h \rangle}_{>0}, \end{aligned} \tag{46}$$

where (from the definite positive condition of A^h, i.e. $\langle A^h v^h, v^h \rangle > 0$ for any nonzero $v^h \in \Omega^h$) we know that the third term of Equation (46) is always positive. Then the condition $\langle A^h u^h - b^h, v^h \rangle = 0$ for all $v^h \in \Omega$ is a sufficient condition to guarantee that u^h minimizes F^h.

We just proved that solving Equation (43) is equivalent to finding the vector $u^h \in \Omega^h$ that minimizes the discrete functional F^h defined by Equation (44) [4, Chapter 10]. This equivalence will help us understand in a very intuitive way how to define the problem in a coarse grid and how that can be used to improve a current guess in the original fine grid. To that end, let us imagine that we have a current approximation v^h of the (unknown) solution u^h of Equation (43), and that we are looking for a correction signal $w^{2h} \in \Omega^{2h}$ that improves the current guess v^h, i.e. $F^h(v^h + I_{2h}^h w^{2h}) < F^h(v^h)$. In particular, we can formally define the optimal choice for w^{2h} as

$$w^{2h} = \underset{w^{2h} \in \Omega^{2h}}{\arg\min} F^h(v^h + I_{2h}^h w^{2h}). \tag{47}$$

Proposition 2 can be proved expanding the functional $F^h(v^h + I_{2h}^h w^{2h})$

$$\begin{aligned}
F^h(v^h + I_{2h}^h w^{2h}) &= \frac{1}{2}\langle A^h(v^h + I_{2h}^h w^{2h}), v^h + I_{2h}^h w^{2h}\rangle - \langle b^h, v^h + I_{2h}^h w^{2h}\rangle \\
&= \frac{1}{2}\langle A^h v^h, v^h\rangle + \frac{1}{2}\langle A^h I_{2h}^h w^{2h}, v^h\rangle + \frac{1}{2}\langle v^h, A^h I_{2h}^h w^{2h}\rangle \\
&\quad + \frac{1}{2}\langle A^h I_{2h}^h w^{2h}, I_{2h}^h w^{2h}\rangle - \langle b^h, v^h\rangle - \langle b^h, I_{2h}^h w^{2h}\rangle \\
&= \frac{1}{2}\langle A^h v^h, v^h\rangle - \langle b^h, v^h\rangle \qquad (48) \\
&\quad + \frac{1}{2}\langle A^h I_{2h}^h w^{2h}, I_{2h}^h w^{2h}\rangle + \langle v^h, A^h I_{2h}^h w^{2h}\rangle - \langle b^h, I_{2h}^h w^{2h}\rangle \\
&= F^h(v^h) + \frac{1}{2}\langle (I_{2h}^h)^T A^h I_{2h}^h w^{2h}, w^{2h}\rangle - \langle (I_{2h}^h)^T (b^h - A^h v^h), w^{2h}\rangle \\
&= F^h(v^h) + \frac{1}{2}\langle A^{2h} w^{2h}, w^{2h}\rangle - \langle b^{2h}, w^{2h}\rangle,
\end{aligned}$$

where $A^{2h} = (I_{2h}^h)^T A^h I_{2h}^h$ and $b^{2h} = (I_{2h}^h)^T (b^h - A^h v^h)$. On the last row of Equation (48) the first term is independent of w^{2h}, which leads to the equivalence

$$\begin{aligned}
w^{2h} &= \arg\min_{w^{2h}\in\Omega^{2h}} F^h(v^h + I_{2h}^h w^{2h}) \\
&= \arg\min_{w^{2h}\in\Omega^{2h}} \frac{1}{2}\langle A^{2h} w^{2h}, w^{2h}\rangle - \langle b^{2h}, w^{2h}\rangle \qquad (49) \\
&\stackrel{def}{=} \arg\min_{w^{2h}\in\Omega^{2h}} F^{2h}(w^{2h}).
\end{aligned}$$

The deduction carried out in Equation (48) is of crucial importance to understand how the original problem should be mapped to a coarser grid. In particular, it allows to derive the following important properties:

- In the coarse grid, the data term can be defined as $b^{2h} = (I_{2h}^h)^T (b^h - A^h v^h)$, which corresponds to a coarse version of the residual $r^h = b^h - A^h v^h$. The operator $(I_{2h}^h)^T$ is mapping r^h from the fine grid Ω^h to the coarse grid Ω^{2h}, hence, it plays the role of the restriction operator $I_h^{2h} = (I_{2h}^h)^T$.
- The coarse version of the matrix A^h is $A^{2h} = (I_{2h}^h)^T A^h I_{2h}^h = I_h^{2h} A^h I_{2h}^h$.
- The definition of the problem in the finer grid should guarantee that A^h is a positive definite self-adjoint operator.

References

[1] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the solution of linear systems: building blocks for iterative methods*, Society for Industrial and Applied Mathematics (SIAM), 1994. ISBN: 978-0-89871-328-2.

[2] P. Bhat, B. Curless, M. Cohen, and C. L. Zitnick, *Fourier analysis of the 2D screened Poisson equation for gradient domain problems*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5303 LNCS (2008), pp. 114–128. https://doi.org/10.1007/978-3-540-88688-4_9.

[3] J.E. Boillat, *Load balancing and poisson equation in a graph*, Concurrency: Practice and Experience, 2 (1990), pp. 289–313.

[4] W.L. Briggs, V.E. Henson, and S.F. McCormick, *A Multigrid Tutorial (2nd Ed.)*, Society for Industrial and Applied Mathematics (SIAM), 2000. ISBN 0-89871-462-1.

[5] L.C. Evans, *Partial differential equations*, Graduate studies in mathematics, American Mathematical Society, 1998. ISBN 0-8218-0772-2.

[6] D.G. Feingold and R.S. Varga, *Block diagonally dominant matrices and generalizations of the Gershgorin circle theorem*, Pacific Journal of Mathematics, 12 (1962), pp. 1241–1250.

[7] W. H. Press, S.A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 1992. ISBN 0521431085.

[8] P.W. Hemker, *On the order of prolongations and restrictions in multigrid procedures*, Journal of Computational and Applied Mathematics, 32 (1990), pp. 423–429.

[9] A. Levin, A. Zomet, S. Peleg, and Y. Weiss, *Seamless Image Stitching in the Gradient Domain*, European Conference on Computer Vision, 4 (2004), pp. 377–389. http://dx.doi.org/10.1007/978-3-540-24673-2_31.

[10] M. Mainberger, A. Bruhn, J. Weickert, and S. Forchhammer, *Edge-based compression of cartoon-like images with homogeneous diffusion*, Pattern Recognition, 44 (2011), pp. 1859–1873. http://dx.doi.org/10.1016/j.patcog.2010.08.004.

[11] J. McCann and N. S. Pollard, *Real-time gradient-domain painting*, ACM Transactions on Graphics, 27 (2008), p. 1. http://dx.doi.org/10.1145/1360612.1360692.

[12] C.C. Paige and M.A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, 12 (1975), pp. 617–629.

[13] P. Pérez, M. Gangnet, and A. Blake, *Poisson image editing*, ACM Transactions on Graphics, 22 (2003), p. 313. http://dx.doi.org/10.1145/882262.882269.

[14] R. Raskar, A. Ilie, and J. Yu, *Image Fusion for Context Enhancement and Video Surrealism*, in International Symposium on Non-Photorealistic Animation and Rendering, 2004, pp. 85–152. http://dx.doi.org/10.1145/987657.987671.

[15] Y. Saad and M.H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[16] R.S. Strichartz, *A guide to distribution theory and Fourier transforms*, World Scientific Publishing Co Inc, 2003.

[17] J. Sun, J. Jia, C-K. Tang, and H-Y. Shum, *Poisson matting*, ACM Transactions on Graphics, 23 (2004), p. 315. http://dx.doi.org/10.1145/1015706.1015721.

[18] H.A. Van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM Journal on scientific and Statistical Computing, 13 (1992), pp. 631–644.