THE UNIVERSITY *of* EDINBURGH

This thesis has been submitted in fulfilment of the requirements for a postgraduate degree (e. g. PhD, MPhil, DClinPsychol) at the University of Edinburgh. Please note the following terms and conditions of use:

- This work is protected by copyright and other intellectual property rights, which are retained by the thesis author, unless otherwise stated.
- A copy can be downloaded for personal non-commercial research or study, without prior permission or charge.
- This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author.
- The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.
- When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.

Efficient Interior Point Algorithms for Large Scale Convex Optimization Problems

Filippo Zanetti

Doctor of Philosophy
University of Edinburgh
2023

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text. This work has not been submitted for any other degree or professional qualification.

(*Filippo Zanetti*)

a mia mamma

Acknowledgements

My deepest gratitude goes to my supervisor Professor Jacek Gondzio, who taught me about optimization and how to do proper research, and whose constant encouragement has been fundamental in reaching the end of my PhD. I am grateful to my second supervisor Dr. John Pearson, for many discussions about research and for the help in various projects. I am deeply thankful to Dr. Julian Hall, who financed the last months of my PhD and who welcomed me with open arms into the HiGHS team. A special thank you to Professor Luca Bergamaschi, for introducing me to numerical linear algebra and for helping me to obtain the PhD position in Edinburgh.

I would like to thank the School of Mathematics and Professor Gondzio for funding my PhD scholarship, and the Laura Wisewell Fund and SIAM Student Travel Fund for providing funding for various conferences.

I would like to thank Professor Kenneth McKinnon and Professor Jordi Castro, for being the examiners of my viva and for providing helpful comments and suggestions to improve this thesis.

I would like to thank Professors Samuli Siltanen and Matti Lassas for a fruitful collaboration and for teaching me about inverse problems; a special thank you goes to Salla, for the constant interactions and exchanges that made the collaboration possible.

I would like to thank Stefano, Santolo and Spyros for many discussions, chats and beers spent talking about research and about many other things. Thank you also to the many colleagues that I had the pleasure to meet during my time in Edinburgh: Barbara, Monse, Claire, Nagisa, Malte, Tom, Andres, Cat, Siemen, Albert.

A special thank you goes to my Italian friends: Enrico, Andrea and Michael, for all the time spent together laughing and drinking; Nicole and Federica, for so many happy moments and chats; Sara, for many long chats and laughs that kept me from going crazy during the long months of lockdown; Davide, because despite the distance we share so many happy memories; Andrea, for the intriguing bike tours; Davide and Federico for many discussions (more or less serious) about PhD life and mathematics.

I would like to thank my father Pier Paolo, my brother Enrico, Daniela and Gianni for supporting me during all these years. Gianni, I hope to inspire you to study, because the modern world seems so full of ignorance.

Lastly, I would like to thank Shunee, with whom I shared almost the entirety of the PhD and to whom I owe all the happy moments of the last three years (and thanks to Covid for getting us together).

Abstract

Interior point methods (IPMs) are among the most widely used algorithms for convex optimization problems. They are applicable to a wide range of problems, including linear, quadratic, nonlinear, conic and semidefinite programming problems, requiring a polynomial number of iterations to find an accurate approximation of the primal-dual solution. The formidable convergence properties of IPMs come with a fundamental drawback: the numerical linear algebra involved becomes progressively more and more challenging as the IPM converges towards optimality. In particular, solving the linear systems to find the Newton directions requires most of the computational effort of an IPM. Proposed remedies to alleviate this phenomenon include regularization techniques, predictor-corrector schemes, purposely developed preconditioners, low-rank update strategies, to mention a few.

For problems of very large scale, this unpleasant characteristic of IPMs becomes a more and more problematic feature, since any technique used must be efficient and scalable in order to maintain acceptable computational requirements. In this Thesis, we deal with convex linear and quadratic problems of large "dimension": we use this term in a broader sense than just a synonym for "size" of the problem. The instances considered can be either problems with a large number of variables and/or constraints but with a sparse structure, or problems with a moderate number of variables and/or constraints but with a dense structure. Both these type of problems require very efficient strategies to be used during the algorithm, even though the corresponding difficulties arise for different reasons.

The first application that we consider deals with a moderate size quadratic problem where the quadratic term is 100% dense; this problem arises from X-ray tomographic imaging reconstruction, in particular with the goal of separating the distribution of two materials present in the observed sample. A novel non-convex regularizer is introduced for this purpose; convexity of the overall problem is maintained by careful choice of the parameters. We derive a specialized interior point method for this problem and an appropriate preconditioner for the normal equations linear system, to be used without ever forming the fully dense matrices involved.

The next major contribution is related to the issue of efficiently computing the Newton direction during IPMs. When an iterative method is applied to solve the linear equation system in IPMs, the attention is usually placed on accelerating their convergence by designing appropriate preconditioners, but the linear solver is applied as a black box with a standard termination criterion which asks for a sufficient reduction of the residual in the linear system. Such an

approach often leads to an unnecessary "over-solving" of linear equations. We propose new indicators for the early termination of the inner iterations and test them on a set of large scale quadratic optimization problems. Evidence gathered from these computational experiments shows that the new technique delivers significant improvements in terms of inner (linear) iterations and those translate into significant savings of the IPM solution time.

The last application considered is discrete optimal transport (OT) problems; these kind of problems give rise to very large linear programs with highly structured matrices. Solutions of such problems are expected to be sparse, that is only a small subset of entries in the optimal solution is expected to be nonzero. We derive an IPM for the standard OT formulation, which exploits a column-generation-like technique to force all intermediate iterates to be as sparse as possible. We prove theoretical results about the sparsity pattern of the optimal solution and we propose to mix iterative and direct linear solvers in an efficient way, to keep computational time and memory requirement as low as possible. We compare the proposed method with two state-of-the-art solvers and show that it can compete with the best network optimization tools in terms of computational time and memory usage. We perform experiments with problems reaching more than four billion variables and demonstrate the robustness of the proposed method.

We consider also the optimal transport problem on sparse graphs and present a primal-dual regularized IPM to solve it. We prove that the introduction of the regularization allows us to use sparsified versions of the normal equations system to inexpensively generate inexact IPM directions. The proposed method is shown to have polynomial complexity and to outperform a very efficient network simplex implementation, for problems with up to 50 million variables.

Lay Summary

Optimization problems are ubiquitous in the modern technological world: finding the *optimal* directions to reach a destination given the current traffic congestion, deciding how to *optimally* schedule air flights with a given fleet of airplanes, or simply finding the *optimal* shape for a tin of beverage are all examples of optimization problems that affect our life. These problems require a very large number of decisions to be made: for example there could be thousands of possible roads and intersections among which to choose how to reach our destination.

With more and more powerful computers becoming available, we are now able to solve optimization problems with billions of decision variables, in a matter of minutes. Interior point methods are one of the reasons why this is possible: these kind of methods are extremely reliable and versatile and can find very accurate solutions much faster than other alternative methods. In particular, these methods find a sequence of approximate solutions that get progressively closer to the required optimum. To move from one approximation to the next, a direction must be computed: this step is responsible for most of the computing time of the algorithm and is where researchers put most of the effort to make these methods more efficient.

One of the main contributions of this Thesis is a strategy to compute these directions in a cheap way, losing as little time as possible but still guaranteeing a good progress of the method; this strategy can bring substantial benefit in particular when the problems are large and challenging.

The second main contribution of this work is the derivation of specific interior point algorithms for two common practical problems. The first one involves medical imaging: when an X-ray image is taken, a *direct* reconstruction is not possible, due to imperfections in the imaging process; an optimization problem is used instead to find the best image that fits the measurements and satisfies some additional properties (for example, we may look for an image with very few edges, or with a lot of empty pixels). The second problem is called *optimal transport* and is strictly related to logistics: this problem requires to find the best possible way of moving objects from one configuration into another. It could be the case of ants moving their nest away from a source of danger grain by grain consuming the least amount of energy, or a harbour crane moving containers onto a ship at the lowest possible cost. To be useful in real-life applications, both these problems require solutions to be computed quickly. In this Thesis we exploit the structure of these problems to obtain efficient algorithms when dealing with instances with a very large number of decision variables.

Contents

Notation and Abbreviations

Below we summarize the notation used in the Thesis. Individual Chapters may define and use a slightly different notation.

I_k	Identity matrix of size k
$\mathbf{e}_k$	Vector of all ones of size k
$\hat{\mathbf{e}}_k$	$k-$th column of the identity matrix
$\mathbb{R}^k_+$	Set of $k-$vectors with non-negative real components
$\mathbf{x}^k_j$	(In an iterative method) Component j of vector $\mathbf{x}$ at iteration k
$\|\cdot\|$	Euclidean norm
$\otimes$	Kronecker product
$\mathrm{vec}(\cdot)$	Vectorization operator that takes a $m \times n$ matrix X and stacks its columns together, to produce an mn vector $\mathbf{x}$
$\lvert\mathcal{S}\rvert$	Cardinality of the set $\mathcal{S}$
$A_{\mathcal{S}}$	(For a matrix A and a set of indices $\mathcal{S}$) Submatrix that contains only the columns with index in $\mathcal{S}$
$\mathrm{diag}(A)$	(For a matrix A) Diagonal matrix with the same diagonal as A
$\mathrm{diag}(\mathbf{a})$	(For a vector $\mathbf{a}$) Diagonal matrix with $\mathbf{a}$ on the diagonal
$\mathrm{nnz}(A)$	(For a matrix A) Number of nonzero entries of A
$\deg(v)$	(In a graph) Degree of node v, i.e. number of edges incident to v
`axpy`	Generalized scaled vector addition operation $\alpha\mathbf{x} + \mathbf{y}$
CG	Conjugate Gradient method
IP	Inner Product regularizer
IPCG	Interior Point Conjugate Gradient method
IPM	Interior Point Method
IPMINRES	Interior Point MINRES method
JTV	Joint Total Variation
KKT	Karush–Kuhn–Tucker conditions
LP	Linear Program

MINRES	MINimum RESidual method
OT	Optimal Transport
PCG	Preconditioned Conjugate Gradient method
PDE	Partial Differential Equation
QP	Quadratic Program
spd	Symmetric Positive Definite matrix

Introduction

Interior Point Methods (IPMs) [63, 131] are among the most successful algorithms for convex optimization: the main reasons for this are the provable polynomial worst-case complexity and the formidable performance of practical implementations.

However, when it comes to large scale problems, with millions or billions of variables, IPMs suffer greatly with respect to first-order methods. While the number of iterations that an IPM needs to perform is usually much smaller than in the case of simplex or first-order methods, each of these iterations can be computationally very expensive. The Newton direction to be taken at each step requires the solution of a large and extremely ill-conditioned linear system, for which it is difficult to find good and efficient preconditioners. Most of the research on IPMs in recent years has focused on alleviating this difficulty, by finding appropriate preconditioners (e.g. [18, 19, 23, 42, 64, 114]), by introducing regularization (e.g. [6, 31, 57, 105, 106]), by employing low-rank strategies (e.g. [45, 70, 71]) or by deriving dedicated algorithms for specific problems (e.g. [27, 28, 29, 32, 40, 52, 68, 101, 133]).

In this Thesis, we deal with optimization problems of *large scale*: with this term, we mean problems which contain a large amount of information to be used, either in the form of a large number of variables, or in the form of dense matrices. In the first case, the matrices involved are of large size, but highly sparse and potentially structured; any factorization-based approach would produce a large fill-in (i.e. the resulting Cholesky or LDL^T factors would be much denser than the original matrix) and is therefore not applicable in practice. A typical example of this kind of problems comes from network applications, where the constraint matrix is often given by a very sparse incidence matrix of the underlying graph. The second class of large scale problems that we consider instead may have a moderate number of variables, but very dense constraint or Hessian matrices. Usually these matrices are not even stored explicitly and can only be accessed as matrix-free operators; any factorization method is therefore inapplicable and iterative linear solvers must be used. Typical examples come from applications that use discretized integral operators, e.g. Fourier or Radon transforms.

In this Thesis, we derive techniques to be used for large scale convex optimization problems and we put most of the emphasis on evaluating their performance in terms of computational time and memory footprint. A lot of attention is also put on how these methods scale with the dimension of the problem; in particular, we show multiple times that an efficient interior point method, for sufficiently large problems, can outperform first-order methods, which often display an exponential

growth of the computational time.

The Thesis is structured as follows:

- In Chapter 1 we introduce some preliminary notions related to graph theory, numerical linear algebra and optimization that are useful for the rest of the Thesis, and we prove some minor original results about the sparsity structure of a particular class of matrices.

- In Chapter 2, we derive a standard primal-dual interior point method and give an overview of its convergence properties. We discuss the linear algebra difficulties that arise from IPMs and prove a result about clustering of eigenvalues in IPM-like matrices; this result highlights the already well known issues of degeneracy of linear programs and how this feature affects IPMs. Finally, we introduce the most common predictor-corrector techniques used in practice.

- In Chapter 3 we present the first main contribution of the Thesis: we propose a novel regularizer for tomographic imaging reconstruction and derive a specialized interior point method for the quadratic program that arises. This problem involves a fully dense Hessian matrix that can only be accessed through matrix-free operators given by the Radon and inverse Radon transforms. The regularizer proposed is non-convex and needs to be tuned properly in order to obtain an optimization problem that is convex overall. The problem includes non-negativity constraints, but no linear constraints, allowing for a simplified IPM formulation; the quadratic term has a Kronecker product structure, where the repeated block, which is fully dense, has a Toeplitz-like structure that arises from the application of the Radon and inverse Radon operators. All these properties are exploited to obtain a simple yet effective preconditioner for the Newton system. We prove that the eigenvalues of the preconditioned normal equations matrix are bounded independently of the IPM iteration (a very desirable feature that is usually difficult to achieve in IPMs) and we also observe this phenomenon experimentally. Numerical tests show that the novel regularizer achieves the proposed goal and that the specialized IPM solves the problem efficiently with up to $500,000$ variables.

- In Chapter 4 we present the second novel contribution of this work. Here we deal with a fundamental issue of IPMs, related to how precisely the Newton directions have to be computed. In literature, whenever an iterative linear solver is used within an IPM, the quality of the *inexact* Newton direction is measured with the relative residual of the linear system; if this quantity is small, the approximate direction is considered safe to use from the linear algebra perspective. However, this approach is not necessarily the best one, since the direction is only used to progress the optimization method: when very far from optimality, even a very roughly computed direction can bring a substantial improvement from the IPM point of view (i.e. it can bring a substantial reduction to the IPM convergence indicators), even though such a direction represents a very poor approximation of the exact one from the

purely linear algebra point of view. In this Chapter we modify the standard conjugate gradient and MINRES algorithms to compute new indicators that allow to stop the linear iterations based on the IPM convergence indicators rather than just on the residual of the linear system. We provide a rationale as to why the proposed approach behaves similarly to the frequently studied inexact variants of IPMs. Moreover, we observe in practice that the proposed stopping criterion brings a substantial benefit to the quadratic optimization problems considered, with up to 70% reduction in computational time, and outperforms other classical termination strategies for linear solvers.

- In Chapter 5 we present an interior point method for discrete Optimal Transport (OT). This problem is becoming very popular in many applications, in particular for the computation of the *Wasserstein* distance; some of these applications have seen the use of interior point methods, but many other dedicated methods have been derived. In the simplest formulation, OT problems require the solution of a linear program with a large number of variables and a highly structured and sparse constraint matrix. The optimal solution is expected to be very sparse as well, that is only a small subset of entries in the optimal solution is expected to be nonzero; to take full advantage of this feature, we embed a column generation strategy into the IPM algorithm, to keep all intermediate vectors and matrices sparse. A standard IPM would need to work with fully dense vectors (as all components are required to be strictly positive), which produces dense blocks in the normal equations matrix; this in turn produces a high peak memory requirement and long computational times. The proposed hybrid of IPM and column generation overcomes these difficulties and produces an efficient and scalable method. Extensive numerical experiments show that the proposed method can solve problems with up to 4 billion variables and outperforms some state-of-the-art network simplex implementations in terms of computational time and memory requirement.

- In Chapter 6 we extend the work on optimal transport from the previous Chapter to problems of transportation over sparse graphs. These problems share many features with the standard OT formulation, in particular in relation to the sparsity of the optimal solution. We apply a proximally stabilized interior point method to this problem and propose to use an inexact sparsified normal equations matrix to compute Newton directions. We prove that the inexactness introduced in this way preserves the polynomial complexity of the method; moreover, we show that the normal equations matrix takes the form of the Laplacian of a re-weighted graph, which allows for a fast solution of the linear systems. We show results on graphs with up to 50 million edges: the proposed approach outperforms a very efficient first-order method for large problems, for both randomly generated and real world graphs.

- Finally, we present conclusions and future perspectives in Chapter 7.

The work presented in this Thesis resulted in the following papers:

- [68]: Gondzio, Lassas, Latva-Aijo, Siltanen and Zanetti, *Material-separating regularizer for multi-energy x-ray tomography*, Inverse Problems, 38 (2022).
- [134]: Zanetti and Gondzio, *A new stopping criterion for Krylov solvers applied in interior point methods*, SIAM J Sci Comput, 45 (2023).
- [133]: Zanetti and Gondzio, *An interior-point-inspired algorithm for linear programs arising in discrete optimal transport*, INFORMS J Comput, 35 (2023).
- [32]: Cipolla, Gondzio and Zanetti, *A regularized interior point method for sparse optimal transport on graphs*, arXiv:2307.05186v1[math.OC] (2023).

Chapter 3 is based on [68]; Chapter 4 is based on [134]; Chapter 5 is based on [133] and Chapter 6 on [32].

Permission to use the material for this Thesis was granted by all co-authors.

Chapter 1

Preliminary notions

In this Chapter, we introduce some basic results about graph theory, linear algebra and optimization that are useful for the rest of the Thesis. A minor original result is presented in Section 1.1.3.

1.1 Useful results from graph theory

1.1.1 Graphs

A graph is a pair $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} \subset \mathbb{N}$ is the set of nodes of the graph and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of edges. $\mathcal{E}$ includes all pairs of nodes belonging to $\mathcal{N}$ that are connected in $\mathcal{G}$:

$$\mathcal{E} = \{(x, y) \,|\, x, y \in \mathcal{N},\ x \text{ is connected to } y\}.$$

A graph is *undirected* if the edges do not have a direction and *directed* otherwise; i.e. in an undirected graph, both edges (u, v) and (v, u) must exist, while in a directed graph only one of them may be present.

Define the *adjacency matrix* of graph $\mathcal{G}$ as the matrix $A \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ such that

$$A_{ij} = \begin{cases} 1, & \text{if there is an edge from node } i \text{ to node } j \\ 0, & \text{otherwise} \end{cases}.$$

Notice that the adjacency matrix of an undirected graph is symmetric. A known property of the powers of the adjacency matrix is the following: $(A^n)_{ij}$ gives the number of paths of length n from node i to node j.

Define the *incidence matrix* of an undirected graph $\mathcal{G}$ as the matrix $E \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{E}|}$ such that

$$E_{ij} = \begin{cases} 1, & \text{if node } i \text{ is the head or tail of edge } j \\ 0, & \text{otherwise} \end{cases}.$$

For a directed graph instead, E is defined as

$$E_{ij} = \begin{cases} -1, & \text{if node } i \text{ is the tail of edge } j \\ 1, & \text{if node } i \text{ is the head of edge } j \\ 0, & \text{otherwise} \end{cases}.$$

1.1.2 Bipartite graphs

A graph is called *bipartite* if the nodes set $\mathcal{N}$ can be divided into two disjoint sets $\mathcal{N}_0$ and $\mathcal{N}_1$, such that every edge connects a node in $\mathcal{N}_0$ and a node in $\mathcal{N}_1$. An interesting properties of these graphs is that they cannot contain loops of odd length.

The adjacency matrix of an undirected bipartite graph takes the following form

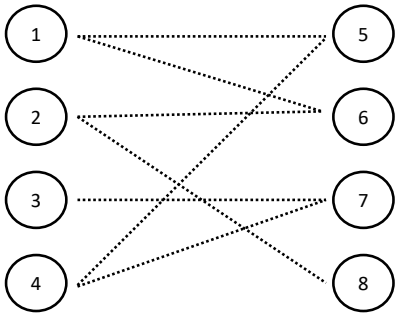
$$A = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix}, \tag{1.1}$$

where B is called the *biadjacency* matrix of the bipartite graph. Given any matrix B, there exists a bipartite graph such that B is its biadjacency matrix.

Figure 1.1 shows an example of bipartite graph. The corresponding biadjacency and incidence matrices are

$$B = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \qquad E = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 1.1: An example of bipartite graph



Consider a complete undirected bipartite graph (i.e. an undirected bipartite graph where it is not possible to add any edge) with $|\mathcal{N}_0|$ and $|\mathcal{N}_1|$ nodes in the two groups. Then, the incidence matrix of such a graph displays the following structure

$$E = \begin{bmatrix} I_{|\mathcal{N}_0|} & I_{|\mathcal{N}_0|} & I_{|\mathcal{N}_0|} & \dots \\ \mathbf{e}_{|\mathcal{N}_0|}^T & & & \\ & \mathbf{e}_{|\mathcal{N}_0|}^T & & \\ & & \mathbf{e}_{|\mathcal{N}_0|}^T & \\ & & & \ddots \end{bmatrix} = \begin{bmatrix} \mathbf{e}_{|\mathcal{N}_1|}^T \otimes I_{|\mathcal{N}_0|} \\ I_{|\mathcal{N}_1|} \otimes \mathbf{e}_{|\mathcal{N}_0|}^T \end{bmatrix}, \tag{1.2}$$

where $\otimes$ represents the Kronecker product (see Section 1.2.1).

1.1.3 Chordal graphs

Given a loop in a graph, we call a *chord* an edge that connects two non-adjacent nodes in the loop. An undirected graph is called *chordal*, or *triangulated*, if any loop with 4 or more nodes contains at least a chord. A matrix is said to be chordal if it can be interpreted as the adjacency matrix of a chordal graph (see Section 1.2.2).

Given a generic matrix $M \in \mathbb{R}^{m \times n}$, we can build a bipartite graph for which M is the biadjacency matrix. We call this graph the *primary graph* $\mathcal{PG}(M)$. Let us call $\mathcal{N}_0$ and $\mathcal{N}_1$ the two groups of nodes in the bipartite graph. The adjacency matrix A of $\mathcal{PG}(M)$ has the structure shown in (1.1) and thus A^2 has the structure

$$A^2 = \begin{bmatrix} MM^T & 0 \\ 0 & M^TM \end{bmatrix}.$$

Let us define a *secondary graph* $\mathcal{SG}(M)$ as follows: $\mathcal{SG}(M)$ has nodes given by $\mathcal{N}_0$; two nodes are connected by an edge if there is a path of length two between them in $\mathcal{PG}(M)$. Since the nonzero entries of A^2 correspond to the nodes that are connected by paths of length two in the original bipartite graph, notice that the adjacency matrix of $\mathcal{SG}(M)$ has the same sparsity pattern as MM^T.

Figure 1.2a shows an example of a primary graph and the corresponding secondary graph (without self-loops) corresponding to the matrix

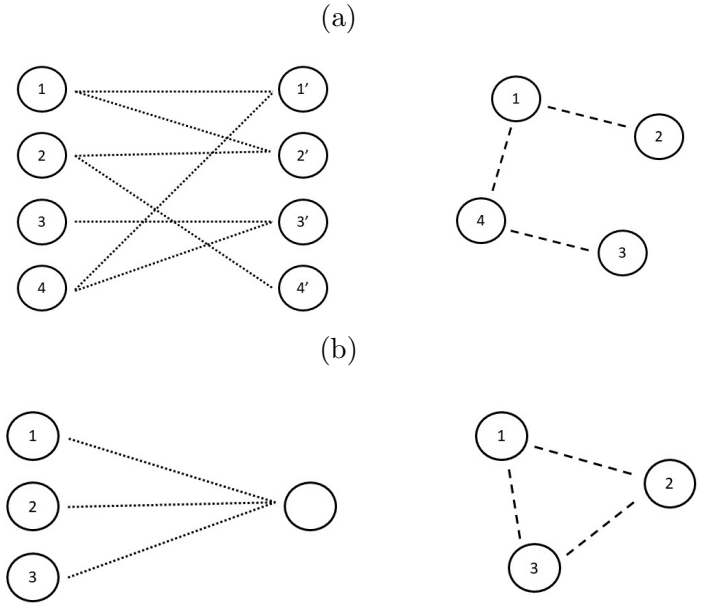
$$M = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

The following result follows trivially from the definitions of $\mathcal{PG}(M)$ and $\mathcal{SG}(M)$:

Lemma 1.1. *If there exists a subset $\mathcal{S} \subseteq \mathcal{N}_0$ and a node $\hat{q} \in \mathcal{N}_1$ such that the graph $\mathcal{PG}(M)$ contains edges $(s, \hat{q})$ $\forall s \in \mathcal{S}$, then the nodes in $\mathcal{S}$ form a complete subgraph of $\mathcal{SG}(M)$.*

Figure 1.2b shows an example of a subset of nodes of $\mathcal{N}_0$ which are all connected to the same node $q \in \mathcal{N}_1$; the graph $\mathcal{SG}(M)$ includes a complete subgraph.

Figure 1.2: **(a)** Primary graph and associated secondary graph. **(b)** Example of Lemma 1.1.



Theorem 1.1. *Given a matrix $M \in \mathbb{R}^{m\times n}$, if every cycle of length larger than or equal to 8 in $\mathcal{PG}(M)$ has at least a chord, then MM^T has chordal sparsity pattern.*

Proof. It is equivalent to show that if every cycle of length at least 8 in $\mathcal{PG}(M)$ has a chord, then $\mathcal{SG}(M)$ is chordal.

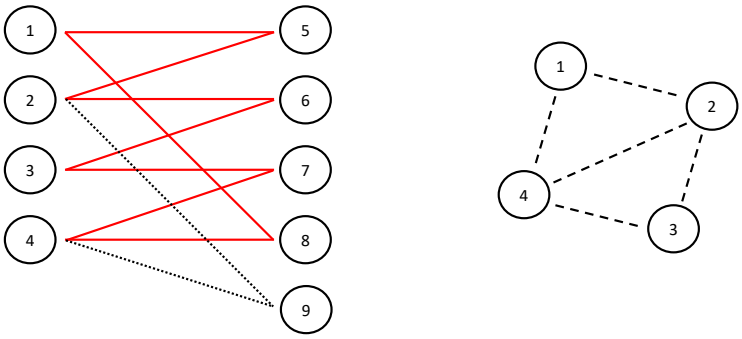
Let us argue by contradiction: let us try to construct a chordless cycle in $\mathcal{SG}(M)$ of length at least 4 . Let us suppose that the cycle is formed by a subset of nodes $\mathcal{Q} \subseteq \mathcal{N}_0$. Due to Lemma 1.1, if there are three or more nodes in $\mathcal{Q}$ connected to the same node in $\mathcal{N}_1$ in $\mathcal{PG}(M)$, then there exists a chord in the proposed cycle; thus, let us assume that this does not happen. This means that we need an *auxiliary* node in $\mathcal{N}_1$ to connect in the primary graph each couple of subsequent nodes in the cycle. Suppose that the chordless cycle in the secondary graph is $q_1 \to q_2 \to \cdots \to q_k \to q_1$, with $k \geq 4$. To form the edge between q_1 and q_2 in $\mathcal{SG}(M)$, we need a path of length two in the primary graph which involves a node $q_1' \in \mathcal{N}_1$. To form the edge between q_2 and q_3 in $\mathcal{SG}(M)$, we need a path of length two in the primary graph which involves a node $q_2' \in \mathcal{N}_1$, since we cannot use q_1' again or we would produce a chord. Continuing this reasoning, to have a chordless cycle we need to use a different auxiliary node $q_j' \in \mathcal{N}_1$ for each pair of subsequent nodes in the cycle. This implies that any edge in $\mathcal{SG}(M)$ corresponds to two edges in $\mathcal{PG}(M)$. To construct the chordless cycle, we would need k auxiliary nodes in $\mathcal{N}_1$ and $2k$ edges in the primary graph. Since $k \geq 4$, this implies that there must be a cycle of length at least 8 in $\mathcal{PG}(M)$; this cycle must be chordless since the only way to have a chord would be if three nodes in $\mathcal{N}_0$ were all connected to the same node in $\mathcal{N}_1$.

We reached a contradiction. □

Notice that, if we apply the Theorem to matrix M^T, it follows that also $M^T M$ has a chordal sparsity pattern. Notice also that the Theorem applies, as a particular case, to matrices M that correspond to acyclic bipartite graphs (i.e. bipartite graph where there are no loops).

Notice also that the converse of Theorem 1.1 does not hold in general. Figure 1.3 shows a counterexample, where the bipartite graph has a loop of size 8 without any chord (shown with solid red lines), but the corresponding secondary graph is chordal.

Figure 1.3: Counterexample to the converse of Theorem 1.1



1.2 Useful results from linear algebra

1.2.1 Kronecker product

Given two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, the *Kronecker product* $A \otimes B \in \mathbb{R}^{mp\times nq}$ is defined as

$$A \otimes B = \begin{bmatrix} A_{11}B & \dots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \dots & A_{mn}B \end{bmatrix}. \tag{1.3}$$

Let us recall some properties of the Kronecker product.

Property 1.1. *The transpose of a Kronecker product can be computed as*

$$(A \otimes B)^T = A^T \otimes B^T.$$

Property 1.2. *The application of $A \otimes B$ to a vector $\mathbf{x} \in \mathbb{R}^{nq}$ can be done as*

$$(A \otimes B)\mathbf{x} = vec(BXA^T)$$

where $vec(\cdot)$ is the vectorization operator and $vec(X) = \mathbf{x}$.

Property 1.3. *If $m = n$ and $p = q$, then the eigenvalues of $A \otimes B$ are given by*

$$\lambda_i \mu_j, \quad i = 1, \dots, m, \quad j = 1, \dots, p$$

where λ_i and μ_j are generic eigenvalues of matrices A and B, respectively.

These are standard results in linear algebra; a proof of these facts can be found, for instance, in [76].

1.2.2 Chordal matrices

Given a sparse matrix A, we call *sparsity pattern* S the matrix build as

$$S_{ij} = \begin{cases} 0, & \text{if } A_{ij} = 0 \\ 1, & \text{if } A_{ij} \neq 0 \end{cases}.$$

Any sparsity pattern can be seen as the adjacency matrix of a directed graph, or undirected if S is symmetric. If the sparsity pattern of a matrix can be interpreted as the adjacency matrix of a chordal graph, we say that the matrix is chordal. Chordal matrices satisfy the following Theorem:

Theorem 1.2. *A sparsity pattern S is chordal if and only if for every positive definite matrix with sparsity pattern S there exists a symmetric permutation of its rows and columns that produces a Cholesky factor with zero fill-in.*

Proof. See e.g. [109, 122]. □

The symmetric permutation that produces zero fill-in in the Cholesky factor is called the *perfect elimination ordering* and it can be found using the *maximum cardinality search* [119] algorithm. The time taken for this operation is $\mathcal{O}(|\mathcal{N}|+|\mathcal{E}|)$, where $|\mathcal{N}|$ and $|\mathcal{E}|$ are respectively the number of nodes and edges of the graph corresponding to the sparsity pattern S.

However, if a matrix is not chordal, a perfect elimination ordering does not exist; it is possible to find an ordering that minimizes the fill-in in the Cholesky factor, but this problem is NP-complete [132]. In practice, some heuristics like *minimum degree* or *nested dissection* are usually applied (see [110, Chapter 3]).

1.2.3 Nonzero entries of the normal equations

Given a sparse matrix $A \in \mathbb{R}^{m \times n}$, the number of nonzero entries of the *normal equations* matrix AA^T can be characterized by the following Lemma.

Lemma 1.2. *Consider a matrix $A \in \mathbb{R}^{m \times n}$; define as w_j the number of nonzero entries in column j and $w = \max_j w_j$. Then, the number of nonzero entries of AA^T satisfies*

$$w^2 \leq nnz(AA^T) \leq \sum_{j=1}^{n} w_j^2.$$

Proof. Notice that the normal equations matrix can be written as

$$AA^T = \sum_{j=1}^{n} \mathbf{a}_j \mathbf{a}_j^T$$

where $\mathbf{a}_j$ is the $j-$th column of matrix A. The number of nonzeros in $\mathbf{a}_j \mathbf{a}_j^T$ is exactly w_j^2. Therefore, the number of nonzeros of AA^T can be upper bounded by summing all the values w_j^2. If the sparsity patterns of all columns are mutually disjoint, then this value represents the exact number of total nonzeros of AA^T.

Given a certain column $\mathbf{a}_j$ and ignoring numerical cancellations, the sparsity pattern of $\mathbf{a}_j \mathbf{a}_j^T$ is contained in the sparsity pattern of AA^T; this holds in particular for the column with the most nonzeros, explaining the lower bound w^2.

□

This Lemma shows that a sparse matrix A can produce a dense normal equations matrix AA^T if there are dense columns in A. However, dense normal equations matrix can arise also without dense columns in A.

Notice that

$$\sum_{j=1}^{n} w_j^2 \leq w \cdot \text{nnz}(A)$$

and define the density of a matrix M as

$$\text{den}(M) = \frac{\text{number of nonzero entries of } M}{\text{number of total entries of } M}.$$

Then Lemma 1.2 implies that

$$\text{den}(AA^T) \leq \frac{w \cdot \text{nnz}(A)}{m^2} = \text{den}(A)\frac{n}{m}w. \tag{1.4}$$

Potentially the density of AA^T may be much larger than the density of A, either because there are dense columns (i.e. w is large) or because the aspect ratio of the matrix (i.e. n/m) is large.

Since this is only an upper bound, the density of AA^T can be much smaller than what is predicted by (1.4). However, the bound can be tight, for example, when A is the incidence matrix of a very sparse graph with many connected components. If A is the incidence matrix of a fully connected bipartite graph, as in (1.2), then the bound is within a factor 2 of the actual density.

1.3 Useful results about Krylov methods

In this Section we give a brief introduction of some Krylov subspace methods, recall some of their properties and discuss some common preconditioning techniques.

Given a linear system $A\mathbf{x} = \mathbf{b}$, where $A \in \mathbb{R}^{n \times n}$, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$, a Krylov subspace method finds a sequence of approximations $\mathbf{x}_1$, $\mathbf{x}_2$, ... that converge to the true solution $\hat{\mathbf{x}}$. In practice, the linear system is transformed to the equivalent one

$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}$, where P is a *preconditioner*, i.e. an approximation of matrix A that is easy to compute, invert and apply to a vector.

Depending on the properties of the matrix and on the choice of Krylov subspaces, various methods can arise: in this work, we focus on the *Conjugate Gradient* (CG) method [73] and the *Minimum Residual* (MINRES) method [100]. Many other methods exist, see e.g. [78] or [110].

1.3.1 Conjugate gradient method

The conjugate gradient method can be used for symmetric positive definite (spd) matrices. Algorithm PCG shows the standard implementation of the preconditioned CG using an spd preconditioner P.

Notice that, at each CG iteration, the most expensive operations are the matrix-vector product $\mathbf{u} = A\mathbf{p}$ and the preconditioner application $\mathbf{z} = P^{-1}\mathbf{r}$. All the other operations involve vectors or scalars. Notice also that the stopping criterion is based on the relative reduction of the residual of the preconditioned linear system.

Algorithm PCG Preconditioned conjugate gradient method

Input: right hand side $\mathbf{b}$, initial approximation $\mathbf{x}_0$, matrix A, preconditioner P, tolerance τ, max iterations `itmax`

$\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$
$\mathbf{r} = \mathbf{r}_0$
$\mathbf{z} = P^{-1}\mathbf{r}$
$\mathbf{p} = \mathbf{z}$
$\rho = \mathbf{r}^T\mathbf{z}$
`iter` $= 0$
while $\|\mathbf{r}\| > \tau\|\mathbf{r}_0\|$ and `iter` $<$ `itmax` **do**
    `iter` $=$ `iter` $+ 1$
    $\mathbf{u} = A\mathbf{p}$
    $\alpha = \rho/\mathbf{u}^T\mathbf{p}$
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
    $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{u}$
    $\mathbf{z} = P^{-1}\mathbf{r}$
    $\rho^N = \mathbf{r}^T\mathbf{z}$
    $\beta = \rho^N/\rho$
    $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
    $\rho = \rho^N$
end while

Let us recall some important properties of this method.

Property 1.4 (Finite termination property)**.** *The CG algorithm finds the exact solution within n iterations (in exact arithmetic).*

Notice that, in the presence of numerical inaccuracies this property may not hold. Notice also that in practice much fewer iterations are needed to find a sufficiently precise approximation.

Property 1.5. *The distance of the $k-$th approximation $\mathbf{x}^k$ from the true solution $\hat{\mathbf{x}}$ can be bounded as*

$$\|\mathbf{x}^k - \hat{\mathbf{x}}\|_A \leq \|\mathbf{x}^0 - \hat{\mathbf{x}}\|_A \min_{p \in \mathcal{P}_0^k} \max_{\lambda \in \sigma(A)} |p(\lambda)|, \tag{1.5}$$

where $\|\cdot\|_A$ represents the norm induced by matrix A, $\mathcal{P}_0^k$ represents the set of polynomials p of degree k such that $p(0) = 1$, $\sigma(A)$ represents the spectrum of eigenvalues of matrix A.

If a preconditioner P is used, then Property 1.5 holds with $P^{-1/2}AP^{-1/2}$ in place of A. Notice the meaning of this property: if the preconditioned matrix has only k distinct eigenvalues, then there exists a polynomial of degree k such that $p(\lambda_j) = 0$, $\forall j = 1, \ldots, k$ and $p(0) = 1$. Therefore, the right hand side of (1.5) vanishes and the approximation $\mathbf{x}^k$ is exact. This fact is summarized in Property 1.6.

Property 1.6. *If the preconditioned matrix $P^{-1}A$ has k distinct eigenvalues, then the CG algorithm terminates within k iterations.*

Property 1.6 is the fundamental tool to build good preconditioners for the CG method. Indeed, rather than reducing the condition number of A, a preconditioner should aim at producing narrow clusters of eigenvalues for matrix $P^{-1/2}AP^{-1/2}$.

1.3.2 Minimum residual method

The MINRES method can be applied to indefinite symmetric linear systems. Algorithm MINRES shows a simplified implementation; some steps (like the computation of the residual) have been omitted for simplicity.

Notice that, as in the case of Algorithm PCG, the most expensive operations at each iteration are the application of the matrix and preconditioner to a vector. Notice also that, despite being applied to indefinite matrices, the preconditioner for MINRES has to be positive definite.

The MINRES method possesses similar properties to the CG method and crucially the same result about clustering of eigenvalues for preconditioning holds.

1.3.3 Preconditioners

A good preconditioner for the CG or MINRES method should be an easy to compute operator, whose inverse can be quickly applied to a vector, and that can produce a distribution of eigenvalues of the preconditioned matrix with as few clusters of eigenvalues as possible. Two of the most common and simple choices for the preconditioner P are summarized below.

The *Jacobi* preconditioner captures only the diagonal of A, i.e. P is a diagonal matrix such that $\text{diag}(P) = \text{diag}(A)$. This technique works well when the mass of the matrix is concentrated mostly on the diagonal; in particular, consider a symmetric diagonally dominant matrix, i.e. an spd matrix for which

$$|A_{jj}| \geq \sum_{i \neq j} |A_{ji}|, \quad \forall j = 1, \ldots, n. \tag{1.6}$$

Algorithm MINRES Preconditioned Minimum Residual method

Input: right hand side $\mathbf{b}$, matrix A, preconditioner P, tolerance τ, max iterations `itmax`

$\mathbf{p} = P^{-1}\mathbf{b}$
$\mathbf{r}_1 = \mathbf{b}$, $\mathbf{r}_2 = \mathbf{r}_1$
$\beta = \sqrt{\mathbf{b}^T\mathbf{p}}$
$\mathbf{w} = 0$, $\mathbf{w}_2 = 0$
$c_s = -1$, $s_n = 0$, $\bar{\varphi} = \beta$, $\epsilon = 0$
$\mathbf{x} = 0$
`iter` $= 0$
while $\|$`residual`$\| > \tau\|$`residual`$_0\|$ and `iter` $<$ `itmax` **do**
    `iter` $=$ `iter` $+ 1$
    $\mathbf{v} = \mathbf{p}/\beta$
    $\mathbf{p} = A\mathbf{v}$
    if `iter` ≥ 2 **then** $\mathbf{p} \leftarrow \mathbf{p} - (\beta/\beta_0)\mathbf{r}_1$ **end if**
    $\alpha = \mathbf{v}^T\mathbf{p}$
    $\mathbf{p} \leftarrow \mathbf{p} - (\alpha/\beta)\mathbf{r}_2$
    $\mathbf{r}_1 = \mathbf{r}_2$, $\mathbf{r}_2 = \mathbf{p}$
    $\mathbf{p} = P^{-1}\mathbf{r}_2$
    $\beta_0 = \beta$, $\beta = \sqrt{\mathbf{r}_2^T\mathbf{p}}$
    $\epsilon_0 = \epsilon$, $\delta = c_s\bar{\delta} + s_n\alpha$, $\bar{g} = s_n\bar{\delta} - c_s\alpha$, $\epsilon = s_n\beta$, $\bar{\delta} = -c_s\beta$
    $\gamma = \max(\sqrt{\bar{g}^2 + \beta^2}, \epsilon)$
    $c_s = \bar{g}/\gamma$, $s_n = \beta/\gamma$, $\varphi = c_s\bar{\varphi}$, $\bar{\varphi} = s_n\bar{\varphi}$
    $\mathbf{w}_1 = \mathbf{w}_2$, $\mathbf{w}_2 = \mathbf{w}$
    $\mathbf{w} = (\mathbf{v} - \epsilon_0\mathbf{w}_1 - \delta\mathbf{w}_2)/\gamma$
    $\mathbf{x} \leftarrow \mathbf{x} + \varphi\mathbf{w}$
end while

Define the level of dominance of each row as δ_j:

$$\delta_j = 1 - \frac{1}{|A_{jj}|}\sum_{i \neq j} |A_{ji}|.$$

The diagonally preconditioned matrix $P^{-1}A$ has as rows the rows of A divided by the corresponding diagonal entry. Therefore, $P^{-1}A$ is still diagonally dominant and has ones on the diagonal. Using Gershgorin's circle Theorem, it is easy to see that the eigenvalues of $P^{-1}A$ belong to the interval $[\delta, 2 - \delta]$, where $\delta = \min \delta_j$. Therefore, all the eigenvalues of the preconditioned matrix belong to a narrow cluster around $\lambda = 1$, and the more dominant the matrix is, i.e. the closest δ is to 1, the more efficient the preconditioner becomes.

Another common preconditioner is the *incomplete Cholesky factorization*: as described in Section 1.2.2, the Cholesky factorization of a generic non-chordal matrix, even when using the best ordering possible, produces fill-in, i.e. produces a triangular factor that is denser that the original matrix. For large matrices this means that a full factorization is intractable. An incomplete factorization produces an approximate Cholesky factor with a specific sparsity pattern: for example, the factor could be forced to have the same sparsity pattern as the original matrix. A popular technique to compute the incomplete factorization is *threshold dropping*: given the parameter `droptol`, elements that are smaller

than a local drop tolerance (that depends on `droptol` and on the norm of the column considered) are set to zero in the incomplete factorization. This type of preconditioner works well for matrices that are not too ill-conditioned and is widely used in PDE applications.

The incomplete factorization process could break down if it encounters a non-positive pivot and understanding exactly for which matrices an incomplete factorization exists is extremely complicated (see [60, Chapter 11]). For diagonally dominant matrices, it is always possible to compute the incomplete Cholesky factor, in exact arithmetic [86]; a common strategy to overcome the break down of the process is to lift the diagonal of the matrix by a sufficient amount (potentially until the matrix is diagonally dominant), however doing so may produce a preconditioner of poor quality.

1.4 Useful results from optimization

In this Section, we recall some important properties of linear and quadratic programming problems for convex optimization. The results presented here can be found, for instance, in [21] or [95].

1.4.1 Linear programming

The primal formulation of a standard Linear Program (LP) takes the form

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0, \end{aligned} \tag{1.7}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$ has full row rank.

The corresponding dual problem is

$$\begin{aligned} \max_{\substack{\mathbf{y}\in\mathbb{R}^m \\ \mathbf{s}\in\mathbb{R}^n}} \quad & \mathbf{b}^T\mathbf{y} \\ \text{s.t.} \quad & A^T\mathbf{y} + \mathbf{s} = \mathbf{c} \\ & \mathbf{s} \geq 0, \end{aligned} \tag{1.8}$$

where $\mathbf{y}$ is the Lagrange multiplier associated with the constraint $A\mathbf{x} = \mathbf{b}$ and $\mathbf{s}$ is the dual slack variable.

The *first-order optimality conditions* or *Karush-Kuhn-Tucker conditions* (KKT) for problem (1.7) state that $(\mathbf{x}, \mathbf{y}, \mathbf{s})$ is a solution of (1.7)-(1.8) if and only if

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ A^T\mathbf{y} + \mathbf{s} &= \mathbf{c} \\ \mathbf{x}_i\mathbf{s}_i &= 0, \quad \forall i = 1, \ldots, n \\ \mathbf{x}, \mathbf{s} &\geq 0 \end{aligned} \tag{1.9}$$

Notice that the third block of equations, known as *complementarity conditions*, imposes that at least one of $\mathbf{x}_i$ and $\mathbf{s}_i$ has to be zero, for each $i = 1, \dots, n$. Therefore, given an optimal solution $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$, we can consider the following sets

$$\mathcal{B} = \{j \,|\, \mathbf{x}_j^* \neq 0\} \qquad \mathcal{N} = \{j \,|\, \mathbf{s}_j^* \neq 0\}, \tag{1.10}$$

that we denote as *basic* and *nonbasic* index sets. Their intersection is clearly empty, in order to not violate the complementarity conditions. Their union instead satisfies the following result:

Theorem 1.3 (Goldman-Tucker)**.** *There exists at least one primal-dual solution* $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$ *of* (1.7)-(1.8) *such that* $\mathbf{x}^* + \mathbf{s}^* > 0$.

A solution that satisfies Theorem 1.3 is called *strictly complementary*; the corresponding basic and nonbasic sets satisfy $\mathcal{B} \cap \mathcal{N} = \emptyset, \quad \mathcal{B} \cup \mathcal{N} = \{1, \dots, n\}$, i.e. they partition the set $\{1, \dots, n\}$.

1.4.2 Quadratic programming

The primal formulation of a standard convex Quadratic Program (QP) is as follows

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^n} \quad & \mathbf{c}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T Q \mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0, \end{aligned} \tag{1.11}$$

where $Q \in \mathbb{R}^{n \times n}$ is a positive semi-definite matrix.

The corresponding dual problem is

$$\begin{aligned} \max_{\substack{\mathbf{y} \in \mathbb{R}^m \\ \mathbf{s} \in \mathbb{R}^n}} \quad & \mathbf{b}^T \mathbf{y} - \frac{1}{2} \mathbf{x}^T Q \mathbf{x} \\ \text{s.t.} \quad & A^T \mathbf{y} + \mathbf{s} - Q\mathbf{x} = \mathbf{c} \\ & \mathbf{s} \geq 0, \end{aligned} \tag{1.12}$$

The first-order optimality conditions in this case take the form

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ A^T \mathbf{y} + \mathbf{s} - Q\mathbf{x} &= \mathbf{c} \\ \mathbf{x}_i \mathbf{s}_i &= 0, \quad \forall i = 1, \dots, n \\ \mathbf{x}, \mathbf{s} &\geq 0 \end{aligned} \tag{1.13}$$

Notice that the complementarity conditions are still present, however strict complementarity does not hold in general for QPs, i.e. there may not exist an optimal solution $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$ such that $\mathbf{x}^* + \mathbf{s}^* > 0$.

1.4.3 Central path

The concept of *central path* is fundamental in the understanding of interior point methods. Given the optimality conditions (1.9), one can solve the perturbed system

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ A^T\mathbf{y} + \mathbf{s} &= \mathbf{c} \\ \mathbf{x}_i\mathbf{s}_i &= \mu, \quad \forall i = 1, \dots, n \\ \mathbf{x}, \mathbf{s} &\geq 0 \end{aligned} \tag{1.14}$$

for some positive parameter μ. The set of solutions $(\mathbf{x}_\mu, \mathbf{y}_\mu, \mathbf{s}_\mu)$, for all values of $\mu > 0$, defines the central path. For LPs, it can be shown that the solution to (1.14) is unique for each value of μ and thus the central path is well defined, provided that the relative interior of the feasible region is non-empty (see e.g. [131]).

In the limit $\mu \to \infty$, the central path converges to the *analytic center* of the feasible region; in the limit $\mu \to 0$, the perturbed system (1.14) approximates (1.9) better and better and, if there exists a primal-dual optimal solution of (1.7)-(1.8), the central path converges to it. Therefore, the central path provides a way to converge to the optimal solution using intermediate approximations that are "well centered", i.e. that satisfy $\mathbf{x}_i\mathbf{s}_i \simeq \mu, \forall i$. In practical algorithms that exploit the central path, it is not possible to move perfectly along it, but rather it is preferred to produce approximations that are close to it, in the sense that they belong to a certain neighbourhood of the central path.

1.4.4 Column generation

Problem (1.7) can be rewritten as

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \sum_{j=1}^{n} \mathbf{c}_j\mathbf{x}_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} \mathbf{a}_j\mathbf{x}_j = \mathbf{b} \\ & \mathbf{x}_j \geq 0, \quad j = 1, \dots, n, \end{aligned}$$

where $\mathbf{a}_j$ is the $j-$th column of matrix A.

If there are many more variables than constraints, i.e. if matrix A has many more columns than rows, we can consider a restricted problem of the form

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \sum_{j\in\mathcal{I}} \mathbf{c}_j\mathbf{x}_j \\ \text{s.t.} \quad & \sum_{j\in\mathcal{I}} \mathbf{a}_j\mathbf{x}_j = \mathbf{b} \\ & \mathbf{x}_j \geq 0, \quad j \in \mathcal{I}, \end{aligned}$$

where $\mathcal{I}$ is a subset of $\{1, \ldots, n\}$. Any feasible solution of the restricted problem $\hat{\mathbf{x}}_R$ produces a feasible solution of the original master problem $\hat{\mathbf{x}}$, simply by setting $\hat{\mathbf{x}}_j = (\hat{\mathbf{x}}_R)_j$, for $j \in \mathcal{I}$ and $\hat{\mathbf{x}}_j = 0$ for $j \notin \mathcal{I}$. Therefore, the optimal objective of the restricted problem is an upper bound on the optimal objective of the original problem.

A column generation algorithm chooses a sequence of sets $\mathcal{I}_k$ and solves the sequence of corresponding restricted problems, looking for the solution of the original master problem. The set $\mathcal{I}_k$ is usually obtained from $\mathcal{I}_{k-1}$ by adding more indices to the set. A common strategy to do this is to consider the reduced costs $\mathbf{s}^{k-1} = \mathbf{c} - A^T\mathbf{y}^{k-1}$: for the current approximation to be dual feasible, the components of the reduced costs have to be nonnegative; any negative component of $\mathbf{s}^{k-1}$ should be added to the set $\mathcal{I}_{k-1}$ to obtain $\mathcal{I}_k$. When all the components of the reduced costs are nonnegative and the primal and dual restricted objectives coincide $\mathbf{b}^T\mathbf{y} = \mathbf{c}^T\mathbf{x}$, then an optimal solution of the master problem has been found.

See [85, 123, 124] for more details on column generation and [66, 67] for the use of interior point methods as solver for the restricted problems.

Chapter 2

Interior point methods

In this Chapter, we introduce a primal-dual interior point method for linear and quadratic programming: we derive the formulation of the method, we summarize the convergence results, we illustrate the difficulties arising from the linear algebra point of view and present some remedies. At the end, we present several strategies to improve the behaviour of the algorithm that are used in practice. Some minor original results are presented in Sections 2.3.1 and 2.3.2.

2.1 Fundamentals of interior point methods

Consider the standard linear program (1.7); the inequality constraints on variable $\mathbf{x}$ can be enforced using a logarithmic barrier function. In particular, using a positive parameter μ, we can consider the following problem

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \mathbf{c}^T\mathbf{x} - \mu\sum_{i=1}^{n}\log \mathbf{x}_i \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b}. \end{aligned} \tag{2.1}$$

The logarithmic barrier ensures that each component $\mathbf{x}_i$ is kept in the interior of the feasible region; when the parameter μ is driven to zero, we can recover a solution of the original LP.

The Lagrangian function associated with (2.1) has the following form

$$\mathcal{L}(\mathbf{x},\mathbf{y}) = \mathbf{c}^T\mathbf{x} + \mathbf{y}^T(\mathbf{b} - A\mathbf{x}) - \mu\sum_{i=1}^{n}\log \mathbf{x}_i$$

and its gradients are

$$\nabla_x\mathcal{L}(\mathbf{x},\mathbf{y}) = \mathbf{c} - A^T\mathbf{y} - \mu X^{-1}\mathbf{e}_n,$$

$$\nabla_y\mathcal{L}(\mathbf{x},\mathbf{y}) = \mathbf{b} - A\mathbf{x},$$

where X is a diagonal matrix with $\mathbf{x}$ on the diagonal. We can introduce a new

variable $\mathbf{s} = \mu X^{-1}\mathbf{e}_n$, so that the first order optimality conditions become

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ A^T\mathbf{y} + \mathbf{s} &= \mathbf{c} \\ XS\mathbf{e}_n &= \mu\mathbf{e}_n \\ \mathbf{x}, \mathbf{s} &\geq 0. \end{aligned} \tag{2.2}$$

Notice that this system is identical to the perturbed system used to define the central path (1.14).

Consider a primal-dual feasible point; the duality gap $\mathbf{c}^T\mathbf{x} - \mathbf{b}^T\mathbf{y}$ can be rewritten, using the dual constraint $\mathbf{c} = A^T\mathbf{y} + \mathbf{s}$, as

$$\mathbf{c}^T\mathbf{x} - \mathbf{b}^T\mathbf{y} = \mathbf{s}^T\mathbf{x} + \mathbf{y}^T A\mathbf{x} - \mathbf{b}^T\mathbf{y}.$$

Now, using the primal constraint $A\mathbf{x} = \mathbf{b}$, we obtain

$$\mathbf{c}^T\mathbf{x} - \mathbf{b}^T\mathbf{y} = \mathbf{x}^T\mathbf{s},$$

i.e. the duality gap is equal to the complementarity gap. It is easy to see that the third block of equations in (2.2) imposes that $\mathbf{x}^T\mathbf{s} = n\mu$. Therefore, assuming primal and dual feasibility, the parameter μ is directly related to the duality gap and thus controls the distance to optimality.

System (2.2) is mildly nonlinear, due to the equations $XS\mathbf{e}_n = \mu\mathbf{e}_n$. Applying the Newton method to this system yields the following system of linear equations

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I_n \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}\mathbf{x} \\ \boldsymbol{\Delta}\mathbf{y} \\ \boldsymbol{\Delta}\mathbf{s} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_P \\ \mathbf{r}_D \\ \mathbf{r}_\mu \end{bmatrix} = \begin{bmatrix} \mathbf{b} - A\mathbf{x} \\ \mathbf{c} - A^T\mathbf{y} - \mathbf{s} \\ \sigma\mu\mathbf{e}_n - XS\mathbf{e}_n \end{bmatrix}, \tag{2.3}$$

where $(\boldsymbol{\Delta}\mathbf{x}, \boldsymbol{\Delta}\mathbf{y}, \boldsymbol{\Delta}\mathbf{s})$ is the Newton step to be taken at the current iteration. It is often not possible to take the full Newton step without violating the constraints.

In the third block of equations, the parameter μ refers to the complementarity measure of the current approximation, while $\sigma\mu$ is the complementarity value of the target point that we are trying to reach. σ is called the centering parameter and it controls where the Newton step is pointing: a value $\sigma = 0$ indicates that we are solving the original optimality conditions (1.9); this direction is called the *affine scaling direction* and usually produces approximations that are poorly centered (i.e. they are far from the central path), since the term involving μ has been completely ignored. Instead, choosing a value of $\sigma = 1$ indicates that we are not trying to get closer to optimality (the duality gap of the target point is the same as that of the current approximation), but we just want to improve the centrality of the current approximation, i.e. get closer to the central path.

Practical IPM implementations either consider an intermediate value of σ, to obtain both an improvement towards optimality and a reasonably well centered approximation, or use a combination of multiple directions, obtained with different values of σ. The latter approach is usually referred to as *predictor-corrector*; many

of these strategies exist [33, 61, 87], some of which are introduced in Section 2.4.

After the Newton direction is computed, the step can be taken:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \boldsymbol{\Delta}\mathbf{x},$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha \boldsymbol{\Delta}\mathbf{y},$$

$$\mathbf{s}^{k+1} = \mathbf{s}^k + \alpha \boldsymbol{\Delta}\mathbf{s}.$$

An appropriate stepsize α has to be computed, to guarantee that the next point belongs to the neighbourhood of the central path considered and that the point satisfies the inequality constraints strictly ($\mathbf{x}_i, \mathbf{s}_i > 0$, $\forall i = 1, \ldots, n$).

The iterations are stopped when a sufficiently accurate solution is found, i.e. when, given a prescribed tolerance `tol`, the following holds

$$\frac{\|\mathbf{r}_P\|}{1 + \|\mathbf{b}\|} < \texttt{tol}, \qquad \frac{\|\mathbf{r}_D\|}{1 + \|\mathbf{c}\|} < \texttt{tol}, \qquad \mu = \frac{\mathbf{x}^T\mathbf{s}}{n} < \texttt{tol}. \tag{2.4}$$

Algorithm IPM shows a simplistic interior point method with the features just described. Notice that the parameter α_0 is responsible for keeping the components of the iterates strictly positive.

Algorithm IPM Interior Point Method

Given $\sigma \in (0, 1)$, $\alpha_0 = 0.999$, $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0)$
Compute $\mu^0 = (\mathbf{x}^0)^T\mathbf{s}^0$ and $\mathbf{r}_P^0$, $\mathbf{r}_D^0$, $\mathbf{r}_\mu^0$
while (2.4) is not satisfied **do**
  Solve (2.3) to find $(\boldsymbol{\Delta}\mathbf{x}^k, \boldsymbol{\Delta}\mathbf{y}^k, \boldsymbol{\Delta}\mathbf{s}^k)$
  Find α^k as the largest α such that $\mathbf{x}^k + \alpha\boldsymbol{\Delta}\mathbf{x}^k \geq 0$ and $\mathbf{s}^k + \alpha\boldsymbol{\Delta}\mathbf{s}^k \geq 0$
  Make the step:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_0\alpha^k\boldsymbol{\Delta}\mathbf{x}^k$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha_0\alpha^k\boldsymbol{\Delta}\mathbf{y}^k$$

$$\mathbf{s}^{k+1} = \mathbf{s}^k + \alpha_0\alpha^k\boldsymbol{\Delta}\mathbf{s}^k$$

  Compute $\mu^{k+1} = (\mathbf{x}^{k+1})^T\mathbf{s}^{k+1}/n$, $\mathbf{r}_P^{k+1}$, $\mathbf{r}_D^{k+1}$, $\mathbf{r}_\mu^{k+1}$
end while

Similarly to (2.3), an IPM applied to a standard quadratic program (1.11) produces a linear system of the kind

$$\begin{bmatrix} A & 0 & 0 \\ -Q & A^T & I_n \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}\mathbf{x} \\ \boldsymbol{\Delta}\mathbf{y} \\ \boldsymbol{\Delta}\mathbf{s} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_P \\ \mathbf{r}_D \\ \mathbf{r}_\mu \end{bmatrix} = \begin{bmatrix} \mathbf{b} - A\mathbf{x} \\ \mathbf{c} - A^T\mathbf{y} - \mathbf{s} + Q\mathbf{x} \\ \sigma\mu\mathbf{e}_n - XS\mathbf{e}_n \end{bmatrix}. \tag{2.5}$$

2.2 Convergence of IPMs

The speed of convergence of an IPM depends on how the parameter μ is reduced during the iterations and on which neighbourhood of the central path is used.

Define the primal-dual feasible and strictly feasible sets as

$$\mathcal{F} = \{(\mathbf{x}, \mathbf{y}, \mathbf{s}) | A\mathbf{x} = \mathbf{b}, A^T\mathbf{y} + \mathbf{s} = \mathbf{c}, (\mathbf{x}, \mathbf{s}) \geq 0\}$$

$$\mathcal{F}^0 = \{(\mathbf{x}, \mathbf{y}, \mathbf{s}) | A\mathbf{x} = \mathbf{b}, A^T\mathbf{y} + \mathbf{s} = \mathbf{c}, (\mathbf{x}, \mathbf{s}) > 0\}$$

In the case of a feasible method (i.e. a method that keeps all intermediate iterates inside the feasible region), two common choices for the neighbourhood are the $2-$norm neighbourhood

$$\mathcal{N}_2(\theta) = \{(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{F}^0 \,|\, \|XS\mathbf{e}_n - \mu\mathbf{e}_n\| \leq \theta\mu\} \tag{2.6}$$

and the one-sided $\infty-$norm neighbourhood

$$\mathcal{N}_{-\infty}(\gamma) = \{(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{F}^0 \,|\, \mathbf{x}_i\mathbf{s}_i \geq \gamma\mu, \, \forall i = 1, \ldots, n\}, \tag{2.7}$$

where $\theta, \gamma \in (0, 1)$.

Feasible IPM algorithms that employ these two neighbourhoods for solving a linear program can be shown [63, 95, 131] to converge to an ε-accurate solution in a number of iterations proportional to $\mathcal{O}(\sqrt{n}\log 1/\varepsilon)$ for $\mathcal{N}_2$ and $\mathcal{O}(n\log 1/\varepsilon)$ for $\mathcal{N}_{-\infty}$. These bounds show that IPMs can solve LPs in a polynomial number of iterations; moreover, the dependence on the accuracy ε is very weak, so that it is possible to obtain very accurate solutions with little extra effort.

In the case of an infeasible method (i.e. an IPM where the intermediate iterations do not need to satisfy the linear constraints), a common neighbourhood used is the infeasible ∞-norm neighbourhood

$$\mathcal{N}_{-\infty}(\gamma, \beta) = \{(\mathbf{x}, \mathbf{y}, \mathbf{s}) \,|\, \|\mathbf{r}_P\| \leq \|\mathbf{r}_P^0\|\beta\mu/\mu^0, \, \|\mathbf{r}_D\| \leq \|\mathbf{r}_D^0\|\beta\mu/\mu^0, \\ \mathbf{x}_i\mathbf{s}_i \geq \gamma\mu, \, \forall i = 1, \ldots, n\}, \tag{2.8}$$

where $\beta \geq 1$. This neighbourhood guarantees that the infeasibilities $\mathbf{r}_P$ and $\mathbf{r}_D$ are reduced at the same rate as μ throughout the iterations.

Infeasible IPMs for linear programming using neighbourhood (2.8) have been shown to converge (e.g. [80]) and to retain polynomial complexity [131], with a number of iterations proportional to $\mathcal{O}(n^2\log 1/\varepsilon)$.

Similar results about polynomial complexity in the context of quadratic programming exist, see e.g. [130].

Much attention has also been put into the study of inexact IPMs, e.g. [9, 12, 14, 25, 26, 35, 44, 56, 65, 81, 84, 89, 90, 135]: in most practical applications, the Newton linear system cannot be solved exactly, but only an approximation of its solution can be found (e.g. due to the use of an iterative Krylov solver). These inaccuracies have an effect on the convergence of the IPM, but with suitable assumptions polynomial complexity can still be achieved. A typical assumption, coming from the theory of inexact Newton methods [41, 78], is that the linear

systems (2.3) or (2.5) are solved with a perturbed right-hand side

$$\begin{bmatrix} \mathbf{r}_P \\ \mathbf{r}_D \\ \mathbf{r}_\mu \end{bmatrix} + \mathbf{r}_{\text{error}}, \quad \text{with } \|\mathbf{r}_{\text{error}}\| \leq \tau_{\text{inexact}} \left\| \begin{bmatrix} \mathbf{r}_P \\ \mathbf{r}_D \\ \mathbf{r}_\mu \end{bmatrix} \right\|$$

with $0 \leq \tau_{\text{inexact}} < 1$ appropriately chosen. Due to the neighbourhood used, this often implies that the error in the right hand side satisfies $\|\mathbf{r}_{\text{error}}\| = \mathcal{O}(\mu)$. In practice, this is accomplished by setting a variable tolerance on the residual of the linear system when using a Krylov method; the tolerance should decrease at the same rate as μ, which means computing more inexact directions in the early stage of IPM and finding more accurate Newton directions when close to the optimal solution. Various strategies have been derived to choose a sequence of tolerances to apply to the CG or MINRES method, in order to reduce the overall number of Krylov iterations; these techniques however all rely on the residual of the linear system. In Chapter 4 we show that this is not necessarily a good choice and there are alternative strategies that can improve substantially the performance of the linear solver.

2.3 Linear algebra of IPMs

System (2.3) is a non-symmetric and highly sparse linear system. To solve it, it is usually reduced to the *augmented system*

$$\begin{bmatrix} -\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{\Delta x} \\ \mathbf{\Delta y} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_D - X^{-1}\mathbf{r}_\mu \\ \mathbf{r}_P \end{bmatrix} \tag{2.9}$$

which is symmetric and indefinite and solved using an indefinite factorization [24, 125] or the MINRES method presented in 1.3.2. Here, $\Theta = XS^{-1}$ is a diagonal matrix with entries $\Theta_{ii} = \mathbf{x}_i/\mathbf{s}_i$. The system can be reduced further to the *normal equations*

$$A\Theta A^T \mathbf{\Delta y} = \mathbf{r}_P + A\Theta(\mathbf{r}_D - X^{-1}\mathbf{r}_\mu). \tag{2.10}$$

which is symmetric and positive definite, and solved using a Cholesky factorization [60] or the CG method presented in Section 1.3.1.

Recalling the complementarity conditions from (1.9) and the definition of basic and nonbasic index sets from (1.10), one can see that the entries Θ_{ii} corresponding to the index set $\mathcal{B}$ tend to ∞ as $\mu \to 0$, while the entries corresponding to the index set $\mathcal{N}$ tend to zero. Therefore, the condition number of Θ get progressively larger during the IPM iterations. In [64], under reasonable assumptions, it is shown that the condition number may be as large as $\mathcal{O}(\mu^{-2})$; if a very accurate solution is required, with μ of the order of 10^{-8} or smaller, this means that the condition number of $A\Theta A^T$ can be as large as 10^{16}. For very large problems, for which a direct Cholesky factorization of $A\Theta A^T$ is not possible, the conjugate gradient method has to be used; however, such a large condition number of the

system matrix is likely to produce an excessively large number of linear iterations, or even the stagnation of the Krylov method without the possibility of actually finding an accurate Newton direction.

In the case of QPs, the normal equations approach is not usually viable, due to the presence of matrix Q; some exceptions include the case of separable QPs, where matrix Q is diagonal and thus easy to invert, and problems with only non-negativity constraints and no linear equality constraints, in which case the normal equations matrix simply takes the form $Q + \Theta^{-1}$.

A lot of attention has been put in recent years into the study of the spectral properties of the matrices in (2.3), (2.5), (2.9), (2.10), see e.g. [38, 91, 92]. Recently, there has been interest in developing IPMs that employ quasi-Newton methods and low rank updates (see [16] for a survey) for the solution of the Newton linear system [45, 70, 71].

We now present two remedies to the ill-conditioning of IPM matrices. The first one is regularization: by slightly perturbing the original optimization problem, it is possible to find a similar problem, with a similar solution, that is much easier to compute. In particular, suppose we add primal and dual diagonal regularization matrices R_p and R_d to the augmented system, to obtain the regularized augmented system matrix

$$\begin{bmatrix} -\Theta^{-1} - R_p & A^T \\ A & R_d \end{bmatrix}$$

and the regularized normal equations matrix

$$A(\Theta^{-1} + R_p)^{-1}A^T + R_d.$$

It can be shown (see [64]) that the condition number of the regularized matrices is bounded independently of μ; this means that the condition number of the IPM matrices is not going to infinity when $\mu \to 0$. However, to achieve accurate solutions the regularization parameters need to be particularly small, in order to minimize the perturbation to the original problem; in turn this means that the bound on the condition number, despite being independent of μ, can still be very large. Much more work has been done in the field of regularization for IPMs, see e.g. [6, 31, 57, 105, 106], where various techniques are introduced and analyzed.

The second remedy to the ill conditioning of IPM matrices is preconditioning. Unfortunately, standard techniques like diagonal scaling or incomplete factorizations are not sufficient in most cases to deal with IPM matrices. Much attention has been put into the development of specific preconditioners tailored to the IPM needs. Some of the techniques commonly used are

- Block preconditioners for the saddle-point matrix (2.9), e.g. inexact constraint or triangular preconditioners [17, 18, 19, 42]

- Splitting preconditioners: they try to predict the splitting of the entries of matrix Θ into basic and nonbasic and exploit this structure to build efficient preconditioners [23, 30, 59, 69, 97, 98, 114, 126].

- Matrix free preconditioner based on partial Cholesky factorization, that captures the information about the largest eigenvalues of (2.10) [64].
- Generic preconditioners for saddle point systems [15, 42, 43].
- Preconditioners developed for a specific class of problems, e.g. compressed sensing [52], PDE constraint optimization [101], image reconstruction inverse problems [68], network optimization [27], and many more.

However, a general purpose strategy applicable to large scale problems (ideally matrix-free), sufficiently robust to be used as a black-box preconditioner, is still elusive.

2.3.1 Clustering of eigenvalues in IPM matrices

In this Section, we derive a result about the eigenvalues of a specific class of matrices, which resemble the structure of the normal equations matrix (2.10) for an LP. This result shows in a different way the well known fact that degeneracy in a linear program leads to an extremely ill conditioned normal equations system.

Consider a matrix $A \in \mathbb{R}^{m\times n}$, with $n > m$ and of full row rank, and a diagonal matrix $\Theta \in \mathbb{R}^{n\times n}$, with strictly positive entries on the diagonal. We suppose the matrices are partitioned in the following way

$$A = \begin{bmatrix} A_B & A_N \end{bmatrix}, \qquad \Theta = \begin{bmatrix} \Theta_B & 0 \\ 0 & \Theta_N \end{bmatrix}$$

and we suppose that the following holds

$$(\Theta_B)_{jj} = \mathcal{O}(\mu^{-1}) \quad (\Theta_N)_{jj} = \mathcal{O}(\mu)$$

for some parameter μ and $\max(\|A_B\|, \|A_N\|) \leq K \in \mathbb{R}$. We assume also that matrix A_B may be rank deficient and define as $\mathcal{N}$ the null space of A_B^T. Notice that $\mathcal{N}^\perp = \mathrm{Im}(A_B)$.

Proposition 2.1. $\mathcal{N} \equiv null(A_B\Theta_B A_B^T)$.

Proof. Trivially, if $A_B^T\mathbf{x} = 0$, then $\mathbf{x} \in \mathrm{null}(A_B\Theta_B A_B^T)$.

If instead $\mathbf{x} \in \mathrm{null}(A_B\Theta_B A_B^T)$ then

$$\mathbf{x}^T A_B\Theta_B A_B^T\mathbf{x} = 0 \quad \Rightarrow \quad A_B^T\mathbf{x} = 0$$

because Θ_B is positive definite. □

We are interested in the properties of the eigenvalues of matrix

$$M = A\Theta A^T = A_B\Theta_B A_B^T + A_N\Theta_N A_N^T$$

in the limit $\mu \to 0$. Notice that M is positive definite and that $A_B\Theta_B A_B^T$ is positive definite over the subspace $\mathcal{N}^\perp$.

Consider an eigenpair $(\lambda, \mathbf{v})$ satisfying

$$(A_B \Theta_B A_B^T + A_N \Theta_N A_N^T)\mathbf{v} = \lambda \mathbf{v},$$

with $\|\mathbf{v}\| = 1$ and $\lambda > 0$.

We can define the orthogonal projectors onto $\mathcal{N}$ and onto its orthogonal complement $\mathcal{N}^\perp$ as $\mathcal{P}_\mathcal{N}$ and $\mathcal{P}_{\mathcal{N}^\perp}$ respectively; we can then split the eigenvector as $\mathbf{v} = \mathbf{v}_\parallel + \mathbf{v}_\perp$, where $\mathbf{v}_\parallel = \mathcal{P}_\mathcal{N}\mathbf{v}$ and $\mathbf{v}_\perp = \mathcal{P}_{\mathcal{N}^\perp}\mathbf{v}$. The eigenpair then has to satisfy

$$A_B \Theta_B A_B^T \mathbf{v}_\perp + A_N \Theta_N A_N^T \mathbf{v} = \lambda \mathbf{v} \tag{2.11}$$

and

$$\lambda = \mathbf{v}_\perp^T A_B \Theta_B A_B^T \mathbf{v}_\perp + \mathbf{v}^T A_N \Theta_N A_N^T \mathbf{v}. \tag{2.12}$$

From (2.12), we obtain

$$\lambda \geq \mathbf{v}_\perp^T A_B \Theta_B A_B^T \mathbf{v}_\perp \geq \min(\Theta_B)_{jj} (\mathbf{v}_\perp^T A_B A_B^T \mathbf{v}_\perp) \geq \min(\Theta_B)_{jj} \sigma_{+,\min}^2(A_B) \|\mathbf{v}_\perp\|^2 \tag{2.13}$$

where $\sigma_{+,\min}(A_B)$ is the smallest strictly positive singular value of matrix A_B. Notice that, if $\|\mathbf{v}_\perp\| \neq 0$, then λ is bounded away from zero.

Projecting (2.11) onto $\mathcal{N}$ we obtain

$$\lambda \mathbf{v}_\parallel = \underbrace{\mathcal{P}_\mathcal{N}(A_B \Theta_B A_B^T \mathbf{v}_\perp)}_{=\mathbf{0}} + \mathcal{P}_\mathcal{N}(A_N \Theta_N A_N^T \mathbf{v})$$

because $A_B \Theta_B A_B^T \mathbf{v}_\perp \in \mathcal{N}^\perp$. Considering that the norm of the orthogonal projection is smaller than or equal to the norm of the original vector, we obtain

$$\lambda \|\mathbf{v}_\parallel\| \leq K^2 \|\Theta_N\|. \tag{2.14}$$

Projecting (2.11) onto $\mathcal{N}^\perp$ instead, we obtain

$$\lambda \mathbf{v}_\perp = \underbrace{\mathcal{P}_{\mathcal{N}^\perp}(A_B \Theta_B A_B^T \mathbf{v}_\perp)}_{A_B \Theta_B A_B^T \mathbf{v}_\perp} + \mathcal{P}_{\mathcal{N}^\perp}(A_N \Theta_N A_N^T \mathbf{v}).$$

Similar considerations as before yield

$$\|A_B \Theta_B A_B^T \mathbf{v}_\perp\| \leq \lambda \|\mathbf{v}_\perp\| + K^2 \|\Theta_N\|. \tag{2.15}$$

Define now $\hat{\lambda}$ and $\hat{\mathbf{v}}$ such that $\lambda \to \hat{\lambda}$ and $\mathbf{v} \to \hat{\mathbf{v}}$, when $\mu \to 0$. Taking the limit of (2.14)-(2.15), we obtain

$$\hat{\lambda} \|\hat{\mathbf{v}}_\parallel\| \leq 0, \quad \|A_B \Theta_B A_B^T \hat{\mathbf{v}}_\perp\| \leq \hat{\lambda} \|\hat{\mathbf{v}}_\perp\|.$$

Therefore, at least one of $\hat{\lambda} = 0$ or $\|\hat{\mathbf{v}}_\parallel\| = 0$ must hold. If $\hat{\lambda} = 0$, then

$$\|A_B \Theta_B A_B^T \hat{\mathbf{v}}_\perp\| \leq 0 \quad \Rightarrow \quad \|\hat{\mathbf{v}}_\perp\| = 0 \quad \Rightarrow \quad \|\hat{\mathbf{v}}_\parallel\| = 1.$$

The first implication follows from Proposition 2.1 and from the fact that $\hat{\mathbf{v}}_\perp$ is orthogonal to the null space of A_B^T. As a consequence, only one of $\hat{\lambda} = 0$ or

$\|\hat{\mathbf{v}}_\|\| = 0$ can hold at the same time. Finally observe that if $\|\hat{\mathbf{v}}_\|\| = 0$, then $\|\hat{\mathbf{v}}_\perp\| = 1$ and from (2.13) it follows that $\hat{\lambda}$ is infinite.

Given that $\mathcal{N}$ has dimension $m-\text{rank}(A_B)$, there exist $m-\text{rank}(A_B)$ orthogonal eigenvectors $\hat{\mathbf{v}}$ such that $\|\hat{\mathbf{v}}_\|\| \neq 0$. We summarize the results of this section in the following Lemma.

Lemma 2.1. *In the limit when $\mu \to 0$, matrix M has $m - rank(A_B)$ eigenpairs that satisfy $\|\mathbf{v}_\|\| \to 1$, $\|\mathbf{v}_\perp\| \to 0$, $\lambda \to 0$ and $rank(A_B)$ eigenpairs that satisfy $\|\mathbf{v}_\|\| \to 0$, $\|\mathbf{v}_\perp\| \to 1$, $\lambda \to \infty$.*

Therefore, the condition number of M tends to ∞ as $\mu \to 0$; this fact however is not caused solely by the clustering of the diagonal entries of Θ, but rather by the rank deficiency of A_B. If $\mathcal{N} = \emptyset$, then there would not be any eigenvalue converging to zero.

Making a parallel with the IPM matrices, we showed that the separation of the entries of Θ and the degeneracy of the problem, produce two separate clusters of eigenvalues and eigenvectors. However, we also showed that this separation is not only caused by the behaviour of matrix Θ, but also by the degeneracy of the basic submatrix A_B; moreover, the larger the degree of degeneracy, the more eigenvalues converging to zero appear, making the situation worse. This fact suggests that, looking from the numerical linear algebra point of view, many of the issues that IPM face are actually due to poor modelling of the problem and could be mitigated by a reformulation of the problem (indeed, many IPM packages employ a pre-processing phase to try and reduce the degeneracy of the problem).

2.3.2 Normal equations and deflation

We now show that if the spectrum of a matrix exhibits two separated clusters of eigenvalues, then the previous solution vectors can be re-used in a deflation scheme effectively. The assumptions of this section do not necessarily reflect the properties of the normal equations of an IPM, but we argue that close to optimality these assumptions are close to be satisfied.

Consider a symmetric positive definite matrix M that satisfies the following

Assumption 2.1. *M has two clusters of eigenvalues: $N_1 > 0$ (large) eigenvalues in the interval $I_1 = [\lambda_{1L}, \lambda_{1U}]$, with $\lambda_{1U} = \lambda_{1L}/d$; $N_2 > 0$ (small) eigenvalues in the interval $I_2 = [\lambda_{2L}, \lambda_{2U}]$, with $\lambda_{2U} = \lambda_{2L}/t$; $0 < d, t < 1$ and $N_1 + N_2 = N$. Denote as r the relative separation of the clusters, such that $\lambda_{1L} = \lambda_{2U}/r$, $0 < r < 1$.*

We can denote the set of orthonormal eigenvectors of M as

$$M\mathbf{u}_i = \lambda_{1i}\mathbf{u}_i, \quad i = 1, \ldots, N_1, \quad \lambda_{1i} \in I_1$$

$$M\mathbf{v}_i = \lambda_{2i}\mathbf{v}_i, \quad i = 1, \ldots, N_2, \quad \lambda_{2i} \in I_2$$

with $N = N_1 + N_2$. Then the respective eigenspaces are

$$\mathcal{U} = \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{N_1}\} = \mathcal{V}^\perp,$$

$$\mathcal{V} = \text{span}\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{N_2}\} = \mathcal{U}^\perp.$$

Denote by $\xi_\mathcal{U}$ and $\xi_\mathcal{V}$ the components of a generic vector ξ along the respective subspaces.

We suppose to drive the separation parameter r to zero in a way that changes only the eigenvalues but not the eigenvectors $\mathbf{u}_i$ and $\mathbf{v}_i$ of matrix M. This can be achieved, for example, in the following way: consider a sequence of real numbers $(r^k)_{k\in\mathbb{N}}$, such that when $k \to \infty$, $r^k \to 0$, and a matrix of eigenvectors $W = [\mathbf{u}_1 \dots \mathbf{u}_{N_1} \mathbf{v}_1 \dots \mathbf{v}_{N_2}]$ (which is independent of k); we build a sequence of diagonal matrices of eigenvalues $(\Lambda^k)_{k\in\mathbb{N}}$ and a sequence of matrices $(M^k)_{k\in\mathbb{N}}$, with $M^k = W\Lambda^k W^T$, such that M^k satisfies Assumption 2.1 with a relative separation r^k.

Proposition 2.2. *Consider the sequence of spd matrices $M^k \in \mathbb{R}^{N\times N}$ described above, satisfying Assumption 2.1. Given a vector $\boldsymbol{\psi} \in \mathbb{R}^N$, the sequence of solutions of the linear systems $M^k\boldsymbol{\xi}^k = \boldsymbol{\psi}$, when $k \to \infty$, satisfy*

$$\frac{\|\boldsymbol{\xi}^k_\mathcal{U}\|}{\|\boldsymbol{\xi}^k_\mathcal{V}\|} \to 0$$

for all right hand sides $\boldsymbol{\psi}$ that have non-zero component along the subspace $\mathcal{V}$.

Proof. Since the orthonormal eigenvectors span $\mathbb{R}^N$, the vector $\boldsymbol{\psi}$ can be decomposed uniquely as

$$\boldsymbol{\psi} = \sum_{i=1}^{N_1} \alpha_i \mathbf{u}_i + \sum_{i=1}^{N_2} \beta_i \mathbf{v}_i.$$

This decomposition does not depend on k, since the eigenvectors do not depend on k. The solution $\boldsymbol{\xi}^k$ can be expressed as

$$\boldsymbol{\xi}^k = \boldsymbol{\xi}^k_\mathcal{U} + \boldsymbol{\xi}^k_\mathcal{V} = \sum_{i=1}^{N_1} \frac{\alpha_i}{\lambda^k_{1i}} \mathbf{u}_i + \sum_{i=1}^{N_2} \frac{\beta_i}{\lambda^k_{2i}} \mathbf{v}_i.$$

Let us define the constant $C = \frac{\sum_{i=1}^{N_1} \alpha_i^2}{\sum_{i=1}^{N_2} \beta_i^2}$. We can say that

$$\frac{\|\boldsymbol{\xi}^k_\mathcal{U}\|^2}{\|\boldsymbol{\xi}^k_\mathcal{V}\|^2} = \frac{\sum_{i=1}^{N_1} \alpha_i^2/(\lambda^k_{1i})^2}{\sum_{i=1}^{N_2} \beta_i^2/(\lambda^k_{2i})^2} \leq \frac{\sum_{i=1}^{N_1} \alpha_i^2/(\lambda^k_{1L})^2}{\sum_{i=1}^{N_2} \beta_i^2/(\lambda^k_{2U})^2} = C(r^k)^2.$$

If the right hand side $\boldsymbol{\psi}$ is chosen such that $\sum_{i=1}^{N_2} \beta_i^2 \neq 0$, then C is finite; the vectors $\boldsymbol{\psi}$ such that $\sum_{i=1}^{N_2} \beta_i^2 = 0$ are the vectors that belong completely to the subspace $\mathcal{U}$. Therefore, if $\boldsymbol{\psi}$ has component along the subspace $\mathcal{V}$, $C(r^k)^2$ goes to zero, as $k \to \infty$.

□

Notice that the vectors that belong completely to the subspace $\mathcal{U}$ are a set of Lebesgue measure zero in $\mathbb{R}^N$, due to the assumption that $N_2 > 0$. Therefore, apart from pathological right hand sides, Proposition 2.2 shows that, when the two clusters of eigenvalues of M tend to be infinitely separated, the solutions

to the linear system $M\boldsymbol{\xi} = \boldsymbol{\psi}$ tend to have component only in the eigenspace corresponding to small eigenvalues. This implies that a set of solutions $\boldsymbol{\xi}_j$, computed with various right hand sides $\boldsymbol{\psi}_j$, may be used effectively in a deflation conjugate gradient method [16, 111], to reduce the negative effect of the cluster of small eigenvalues.

The normal equations matrix in IPMs, close to optimality, satisfies Lemma 2.1 and thus also Assumption 2.1. However, the normal equations matrix may change significantly from one iterations to the next during an IPM, due to significant changes in the entries of matrix Θ. Provided that the basic subset of variables has stabilized close to optimality, we can expect the subspaces $\mathcal{U}$ and $\mathcal{V}$ to also be quite stable and thus we can expect that previous directions can indeed be recycled in a deflation scheme.

2.4 Predictor correctors techniques in IPMs

The most common technique used to improve the convergence of practical IPM algorithms is using a predictor-corrector strategy. A first direction, called *predictor* or *affine-scaling direction* is computed, by setting $\sigma = 0$ in the right hand side of (2.3); this direction is poorly centered and cannot be used in an IPM (as explained in Section 2.1). A second direction, called *corrector* is computed by solving (2.3) with the right hand side

$$\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \sigma\mu\mathbf{e}_n \end{bmatrix},$$

i.e. what was neglected in the predictor direction, and potentially other terms that take into account the error made with the predictor. In this way, the choice of σ can be made after the information coming from the predictor is available and, by carefully selecting a right hand side for the corrector, it is possible to produce directions that are much more useful than the plain Newton direction used in the theoretical setting.

A very common technique is due to Mehrotra [87]: after computing the affine-scaling direction $\boldsymbol{\Delta}_{\text{aff}} = (\boldsymbol{\Delta}_{\text{aff}}\mathbf{x}, \boldsymbol{\Delta}_{\text{aff}}\mathbf{y}, \boldsymbol{\Delta}_{\text{aff}}\mathbf{s})$ and the corresponding stepsize α, the parameter σ is chosen with the following heuristic

$$\sigma = \left(\frac{(\mathbf{x} + \alpha\boldsymbol{\Delta}_{\text{aff}}\mathbf{x})^T(\mathbf{s} + \alpha\boldsymbol{\Delta}_{\text{aff}}\mathbf{s})}{\mathbf{x}^T\mathbf{s}} \right)^3$$

and the right hand side for the corrector is set to

$$\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \sigma\mu\mathbf{e}_n - \Delta_{\text{aff}}X\Delta_{\text{aff}}S\mathbf{e}_n \end{bmatrix}.$$

The corrector is then added to the affine-scaling direction to obtain the final

direction for the current IPM iteration.

Another common corrector choice, that improves upon Mehrotra's strategy, is the *multiple centrality correctors* technique [33, 61]: after computing the same predictor $\mathbf{\Delta}_{\text{aff}}$ and stepsize α as before, a trial point is constructed with an increased stepsize $\tilde{\alpha} > \alpha$

$$\tilde{\mathbf{x}} = \mathbf{x} + \tilde{\alpha}\mathbf{\Delta}_{\text{aff}}\mathbf{x}, \quad \tilde{\mathbf{s}} = \mathbf{s} + \tilde{\alpha}\mathbf{\Delta}_{\text{aff}}\mathbf{s}$$

This trial point is then projected onto a symmetric neighbourhood

$$\mathcal{N}_s(\gamma) = \{(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{F}^0 \mid \frac{\mu}{\gamma} \geq \mathbf{x}_i\mathbf{s}_i \geq \gamma\mu, \, \forall i = 1, \ldots, n\}, \tag{2.16}$$

by first computing the vector $\mathbf{t}$

$$\mathbf{t}_j = \begin{cases} \gamma\mu - \tilde{\mathbf{x}}_j\tilde{\mathbf{s}}_j & \text{if } \tilde{\mathbf{x}}_j\tilde{\mathbf{s}}_j \leq \gamma\mu \\ \mu/\gamma - \tilde{\mathbf{x}}_j\tilde{\mathbf{s}}_j & \text{if } \tilde{\mathbf{x}}_j\tilde{\mathbf{s}}_j \geq \mu/\gamma \\ 0 & \text{otherwise.} \end{cases}$$

and then setting the right hand side for the corrector direction to $[\mathbf{0}, \mathbf{0}, \mathbf{t}]^T$. This technique can be applied multiple times, i.e. more than one corrector can be computed, iterating the procedure just described on the new direction. The number of directions to compute depends on how efficiently one can solve multiple linear systems with the same matrix: if a direct method is used, then the same factorization can be used to compute all directions, with a small increase in computational time; if instead an iterative method is used, then the same preconditioner can be applied, but the Krylov method needs to be run multiple times, which could produce a substantial increase in the computational time.

A safeguard that is usually used is to check whether the new corrected direction actually improves with respect to the previous one, in term of the maximum stepsize that can be taken. If the new direction does not allow to take a large enough stepsize (based on some predetermined criterion), then the last corrector direction is rejected and the previously found direction is used for the current IPM iteration.

Chapter 3

An interior point method for tomographic imaging

This Chapter presents the first main contribution of the Thesis: a specialized interior point method for tomographic imaging applications. The work presented here is based on [68]; permission to use the material for this Thesis was granted by the co-authors.

The problem considered in this Chapter produces a large and dense normal equations matrix (approximately 500,000 rows and columns with 100% nonzero entries), which can only be used in a matrix-free way. The main features of the method proposed are the simple and efficient preconditioning strategy and the positive effect of the newly proposed regularizer on the final image quality.

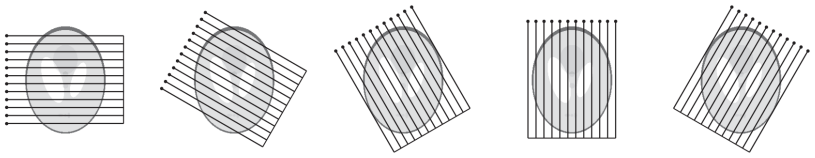
3.1 Introduction

The problem presented in this Chapter involves the reconstruction of an image obtained with a dual-energy x-ray tomography. This is a classical inverse problem which appears in many practical fields: examples include diagnostic medicine, non-destructive testing of historical or archaeological artefacts, live monitoring of industrial processes that happen inside pipes or chambers and many others.

The noise in the measurements and the requirement of using as few angles of measurement as possible (e.g. to minimize the radiation dose to a patient), make this kind of problem challenging. The goal is to understand the spatial distribution of two different materials, for example bone and soft tissue; to do so, the domain of interest Ω is discretized into $N \times N$ square-shaped pixels. There are two unknowns: non-negative $N \times N$ matrices $G^{(1)}$ and $G^{(2)}$ modelling the distributions of material 1 and material 2, respectively. In the physical object being analysed, in each pixel either material 1 or material 2 is present, i.e. there cannot be both materials at the same time and there is no third material present. The number $G^{(\ell)}_{i,j} \geq 0$ represents the concentration of material ℓ in pixel (i, j) of the reconstruction, where i is row index and j is column index. Notice that in the numerical reconstruction, both materials may be present in the same pixel, because it is impossible to enforce a perfect separation.

In numerical computations we represent the elements of the pair of material

Figure 3.1: Image taken from [93] that illustrates the parallel beam geometry used to obtain the measurements. In this case, the number of angles is $P = 5$ and the number of parallel X-ray beams is $r_0 = 12$.



matrices $(G^{(1)}, G^{(2)}) \in (\mathbb{R}^{N\times N})^2$, as a vertical vector

$$\mathbf{g} = \begin{bmatrix} \mathbf{g}^{(1)} \\ \mathbf{g}^{(2)} \end{bmatrix} \in \mathbb{R}^{2N^2}.$$

We consider recording X-ray transmission data with two different energies, low and high, resulting in two M-dimensional data vectors called $\mathbf{m}^L$ and $\mathbf{m}^H$. The low-energy measurement is given by

$$\mathbf{m}^L = c_{11}A^L\mathbf{g}^{(1)} + c_{12}A^L\mathbf{g}^{(2)},$$

as both materials attenuate the low-energy X-rays with individual strengths described by the constants $c_{11} > 0$ and $c_{12} > 0$. Note that empirical values of c_{11} and c_{12} can be found by measuring pure samples of each of the two known materials. The $M\times N^2$ matrix A^L encodes the geometry of the tomographic measurement in a standard way [93, Section 2.3.4]; it contains the path lengths of X-rays travelling inside the pixels in Ω. P projection directions are used to image the object and, for each direction, a set of r_0 parallel X-ray beams is sent through the materials, as shown in Figure 3.1 . The number of rows of matrix A^L is then $M = r_0P$.

Analogously we get for the high-energy measurement

$$\mathbf{m}^H = c_{21}A^H\mathbf{g}^{(1)} + c_{22}A^H\mathbf{g}^{(2)},$$

where the geometric system matrix A^H is possibly different from A^L. Again, $c_{21} > 0$ and $c_{22} > 0$ can be determined empirically. Notice that in the numerical experiments presented in this chapter, the geometry of the measurements for high and low energy are always the same, so that $A^L = A^H$.

Now we can combine both measurements in a unified linear system:

$$\mathbf{m} = \begin{bmatrix} \mathbf{m}^L \\ \mathbf{m}^H \end{bmatrix} = \begin{bmatrix} c_{11}A^L & c_{12}A^L \\ c_{21}A^H & c_{22}A^H \end{bmatrix} \begin{bmatrix} \mathbf{g}^{(1)} \\ \mathbf{g}^{(2)} \end{bmatrix} = \mathcal{A}\mathbf{g}.$$

The core idea in dual-energy X-ray tomography for material decomposition is to choose the two energies so that the two materials respond to them differently. For example, one material might be quite indifferent to the energy change while the

other could attenuate very differently according to energy.

In [68], the Authors propose a new regularization technique which replaces the standard Joint Total Variation (JTV) approach and exploits the inner product $\langle \mathbf{g}^{(1)}, \mathbf{g}^{(2)} \rangle$ to enforce the separation of the two materials. The JTV regularizer favors reconstructed images with piecewise constant areas, with the shortest possible separation between these areas. It has been used extensively in imaging problems and also in dual-energy applications similar to the one presented in this chapter.

The optimization problem that arises in [68] is the following

$$\arg\min_{\mathbf{g}^{(j)} \geq 0} \left\{ \|\mathbf{m} - \mathcal{A}\mathbf{g}\|_2^2 + \alpha \mathcal{R}(\mathbf{g}) + \beta \mathcal{S}(\mathbf{g}) \right\}, \tag{3.1}$$

where $\alpha, \beta > 0$ are regularization parameters, $\mathbf{g}^{(j)} \geq 0$ means that the elements of the vector are non-negative numbers and the regularizer $\mathcal{R}$ can be any of the standard choices such as the Tikhonov penalty

$$\mathcal{R}(\mathbf{g}) = \|\mathbf{g}\|_2^2, \tag{3.2}$$

while $\mathcal{S}$ is the newly introduced regularizer

$$\mathcal{S}(\mathbf{g}) = \mathcal{S}\left(\begin{bmatrix} \mathbf{g}^{(1)} \\ \mathbf{g}^{(2)} \end{bmatrix} \right) := 2\langle \mathbf{g}^{(1)}, \mathbf{g}^{(2)} \rangle = 2 \sum_{i=1}^{N^2} \mathbf{g}_i^{(1)} \mathbf{g}_i^{(2)}. \tag{3.3}$$

Together with the non-negativity constraint, $\mathcal{S}$ promotes the point-wise separation between the two materials: at each pixel, at least one of the images, $G^{(1)}$ or $G^{(2)}$, needs to have a zero value to make $\mathcal{S}$ minimal. This regularizer is referred to as IP in the following, due to the presence of the inner product term.

Notice that the use of a more standard L_1 regularizer to induce sparsity is not directly applicable, because it would not guarantee the correct sparsity pattern of the solution vector (i.e. at least one of the two materials must have a zero value at each pixel). However, an L_1 regularizer could be used in place of the Tikhonov regularizer, while still employing the inner product term. This however would complicate the optimization problem due to non-differentiability.

The Tikhonov regularizer is used to balance the effect of the inner product term, which otherwise would risk to make the problem non-convex. By keeping α larger then β, we can make sure that the problem is always convex.

The quadratic program resulting from the application of the novel variational regularization is solved using an interior point method; we develop an efficient preconditioner for the normal equations which guarantees the spectrum of the preconditioned matrix to remain independent of the IPM iteration. The numerical experience indicates that this approach allows us to solve the largest problem (N=512) in a matter of minutes on a standard laptop.

The Chapter is organized as follows: Section 3.2 shows the details of the IPM applied to problem (3.1); Section 3.3 describes the structure of the normal equations matrix; Section 3.4 presents the preconditioner used and derives eigenvalue bounds; Section 3.5 shows the numerical results in terms of quality of the

reconstruction with the new regularizer.

3.2 Specialized interior point method

By combining the use of the Tikhonov regularizer (3.2) and the Inner Product regularizer (3.3), which promotes the point-wise separation of two materials, we arrive at the constrained quadratic programming task

$$\underset{\mathbf{g}^{(j)}\geq 0}{\arg\min} \left\{ \|\mathbf{m} - \mathcal{A}\mathbf{g}\|_2^2 + \alpha\|\mathbf{g}\|_2^2 + \beta\, \mathbf{g}^T L \mathbf{g} \right\}, \tag{3.4}$$

where

$$L = \begin{bmatrix} 0 & I_{N^2} \\ I_{N^2} & 0 \end{bmatrix}.$$

The problem may be written as an explicit quadratic program with inequality (non-negativity) constraints

$$\underset{\mathbf{g}^{(j)}\geq 0}{\arg\min} -\mathbf{m}^T\mathcal{A}\mathbf{g} + \frac{1}{2}\mathbf{g}^T(Q_1 + Q_2)\mathbf{g} \tag{3.5}$$

where

$$Q_1 = \begin{bmatrix} c_{11}^2(A^L)^TA^L + c_{21}^2(A^H)^TA^H & c_{11}c_{12}(A^L)^TA^L + c_{21}c_{22}(A^H)^TA^H \\ c_{11}c_{12}(A^L)^TA^L + c_{21}c_{22}(A^H)^TA^H & c_{12}^2(A^L)^TA^L + c_{22}^2(A^H)^TA^H \end{bmatrix}, \tag{3.6}$$

$$Q_2 = \begin{bmatrix} \alpha I_{N^2} & \beta I_{N^2} \\ \beta I_{N^2} & \alpha I_{N^2} \end{bmatrix}. \tag{3.7}$$

Notice that $Q = Q_1 + Q_2$ can be written as

$$Q = \begin{bmatrix} c_{11}^2 & c_{11}c_{12} \\ c_{11}c_{12} & c_{12}^2 \end{bmatrix} \otimes (A^L)^TA^L + \begin{bmatrix} c_{21}^2 & c_{21}c_{22} \\ c_{21}c_{22} & c_{22}^2 \end{bmatrix} \otimes (A^H)^TA^H + \begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix} \otimes I_{N^2}, \tag{3.8}$$

where $\otimes$ represents the Kronecker product (1.3).

Lemma 3.1. *If $\alpha \geq \beta$, problem (3.5) is convex.*

Proof. We just need to show that matrix Q in (3.8) is positive semi-definite. We know that matrix

$$\begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix}$$

is positive semi-definite if $\alpha \geq \beta$; the other matrices in the right hand side of (3.8) are always positive semi-definite. Therefore, using Property 1.3, Q is the sum of semi-definite matrices and is then positive semi-definite. □

Therefore, in the following we always assume that $\alpha \geq \beta$.

Notice that another possible strategy to tackle problem (3.4) would be to use a separable formulation by defining $\mathbf{y} = \mathbf{m} - \mathcal{A}\mathbf{g}$ and including the term $\mathbf{y}^T\mathbf{y}$ in the objective function. In this way, the quadratic term would be simplified, with the regularization terms separated from the least squares term. However, the complication coming from operator $\mathcal{A}$ would still arise in the form of linear constraints. Moreover, as it will be clear later, keeping the quadratic term $\mathcal{A}^T\mathcal{A}$ together with the regularization terms helps in producing a powerful preconditioner, because the effect of neglecting elements in $\mathcal{A}^T\mathcal{A}$ is reduced.

3.2.1 Interior point method formulation

We decided to solve problem (3.5) using an interior point method: these methods are among the most efficient solvers for quadratic programs of large dimensions and can often outperform the more common first order methods in terms of speed of convergence and accuracy. For this problem, we aim at reaching large dimensions, and the FISTA method [13], already for moderate problem sizes ($N = 128$), was not able to match the results of the interior point solver; we thus decided to consider only the latter in this work.

Since the problem (3.4) does not involve any linear equality constraints, we can obtain a simpler IPM formulation, as mentioned in Section 2.3. To apply an interior point method to (3.4), we proceed in the usual way and start from adding a logarithmic barrier to form the Lagrangian:

$$L(\mathbf{g}, \mu) = \frac{1}{2}\mathbf{g}^T Q\mathbf{g} - \mathbf{m}^T\mathcal{A}\mathbf{g} - \mu \sum_{i=1}^{2N^2} \log g_i.$$

The optimality conditions are

$$\begin{cases} Q\mathbf{g} - \mathbf{s} = \mathcal{A}^T\mathbf{m} \\ GS\mathbf{e}_{N^2} = \mu\mathbf{e}_{N^2} \\ \mathbf{g}, \mathbf{s} > 0. \end{cases}$$

and the Newton step $(\Delta\mathbf{g}, \Delta\mathbf{s})$ can be found solving

$$\begin{bmatrix} Q & -I_{N^2} \\ S & G \end{bmatrix} \begin{bmatrix} \Delta\mathbf{g} \\ \Delta\mathbf{s} \end{bmatrix} = \begin{bmatrix} \mathbf{r_1} \\ \mathbf{r_2} \end{bmatrix},$$

where $\mathbf{r_1} = \mathcal{A}^T\mathbf{m} - Q\mathbf{g} + \mathbf{s}$ and $\mathbf{r_2} = \sigma\mu\mathbf{e}_{N^2} - GS\mathbf{e}_{N^2}$. If we form the normal equations, we obtain the final linear system that we need to solve:

$$(Q + G^{-1}S)\Delta\mathbf{g} = \mathbf{r_1} + G^{-1}\mathbf{r_2}. \tag{3.9}$$

We can then retrieve $\Delta\mathbf{s}$ as

$$\Delta\mathbf{s} = G^{-1}(\mathbf{r_2} - S\Delta\mathbf{g}). \tag{3.10}$$

To stop the IPM iterations, we check the normalized dual residual and the complementarity measure:

$$\frac{\|\mathcal{A}^T\mathbf{m} - Q\mathbf{g} + \mathbf{s}\|}{1 + \|\mathcal{A}^T\mathbf{m}\|} < \texttt{tol}, \quad \mu < \texttt{tol}, \tag{3.11}$$

where `tol` is the IPM tolerance.

The matrix Q in (3.9) cannot be stored or accessed directly, but it is accessible only via matrix-vector products performed using the Radon transform. Hence, to solve the linear system we need to use a matrix free approach; this is done employing conjugate gradient with an appropriate preconditioner.

3.3 Structure of the normal equations matrix

The normal equations matrix is $Q_1 + Q_2 + G^{-1}S$, with Q_1 given in (3.6) and Q_2 given in (3.7). $G^{-1}S$ is diagonal, Q_2 has a 2×2 block structure with diagonal blocks, while the structure of Q_1 depends on matrices $(A^L)^T A^L$ and $(A^H)^T A^H$.

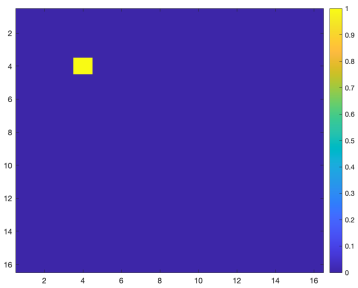
Let us analyze an instance where $A^L = A^H = A$. Matrix A applies the Radon transform to a vector, while A^T applies the inverse Radon transform. To understand the structure of matrix $A^T A$, let us analyse the effect of applying it to the vectors of the canonical basis of $\mathbb{R}^{N^2}$ $\hat{\mathbf{e}}_i$, i.e. to the vectors with all zeros apart from component i, which is equal to 1. $A^T A\hat{\mathbf{e}}_i$ corresponds to the $i-$th column of matrix $A^T A$, so analysing the structure of these vectors allows us to understand the structure of the whole matrix.

The vector $\hat{\mathbf{e}}_i$ of size N^2 can be reshaped into an $N \times N$ matrix, which looks as in Figure 3.2a. Applying the Radon transform to this image produces its *sinogram*, shown in Figure 3.2b. Applying the inverse Radon transform to the sinogram attempts to reconstruct the original image, but the reconstruction is blurred and the original mass, concentrated in a single point, gets spread to the adjacent pixels, as can be seen in Figure 3.2c.

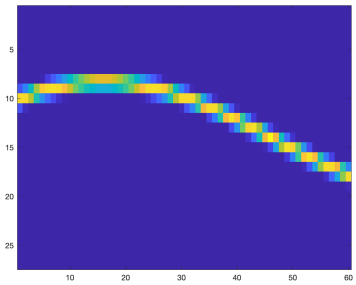
If the image shown in Figure 3.2c is reshaped as a vector, it represents the $i-$th column of matrix $A^T A$; due to the blurring in the reconstruction, the extremely sparse initial vector $\hat{\mathbf{e}}_i$ is turned into a fully dense vector. The magnitude of the pixels of Figure 3.2c has a pattern given by the distance from the central pixel considered. This translates into a pattern for the $i-$th column of $A^T A$: for example, the four pixels which are the closest to the central pixel, all have approximately the same magnitude and represent four entries of the $i-$th column. Two of these entries will be immediately above and below the central pixel (i.e. they appear in the first upper diagonal and in the first lower diagonal of $A^T A$), while the other two have a shift of N (i.e. they appear in the $N-$th upper and lower diagonal), due to the fact that we are stacking the columns of Figure 3.2c.

In Figure 3.3 we show this relation between how close the pixels are to the central pixel and the magnitude of the elements on the diagonals. The plots on the right show the average magnitude of the elements along any diagonal of $A^T A$, where 0 is the main diagonal and positive (negative) numbers indicate upper (lower) diagonals. The highlighted pixels on the left, for the $i-$th column

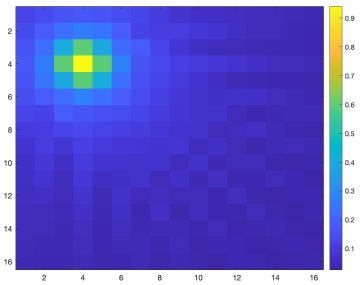
Figure 3.2: **(a)** Unit vector $\hat{\mathbf{e}}_i$ reshaped as a matrix, in the case $N = 16$. **(b)** Sinogram of (a), obtained with the Matlab runction `radon`, equivalent to $A\hat{\mathbf{e}}_i$. The angles of imaging are on the x-axis, while the shifts are on the y-axis. **(c)** Reconstructed image $A^T A\hat{\mathbf{e}}_i$.



(a)



(b)



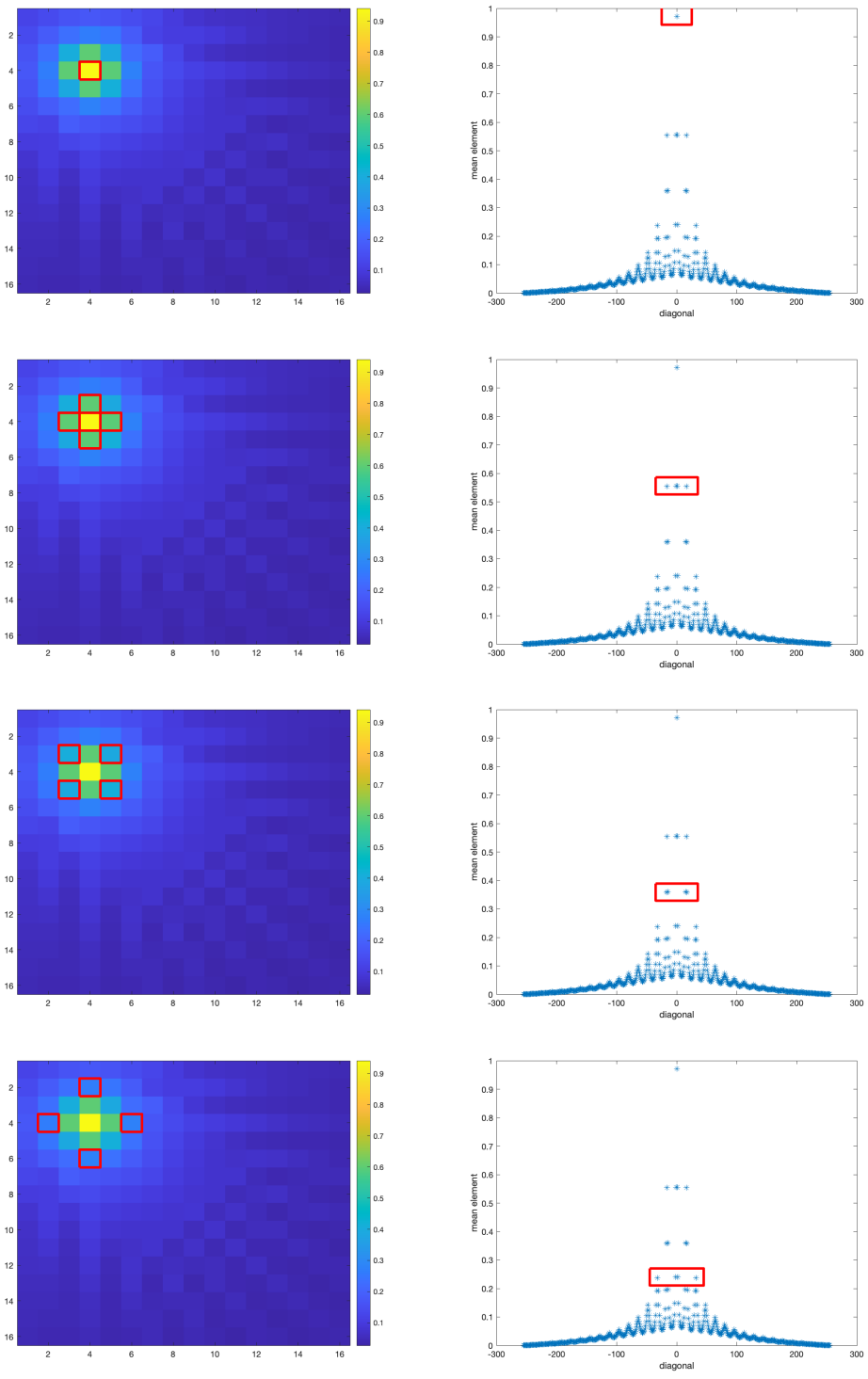
(c)

considered, are the ones that produce the entries corresponding to the highlighted diagonals on the right. The first row of Figure 3.3 shows the central pixel, which corresponds to the entry along the main diagonal. The second row shows the four pixels mentioned above, which are closest to the central pixel: of the four highlighted diagonals on the right, two of them are next to the main diagonal and two of them are N units from the main diagonal. The next rows show the subsequent blocks of pixels with roughly the same magnitude, due to their distance from the central pixel.

Due to the blurring in Figure 3.2c, notice that all the entries in any column are different from zero. Matrix $A^T A$ thus is completely dense. The intensity of the elements of the whole matrix $A^T A$ can be seen in Figure 3.4.

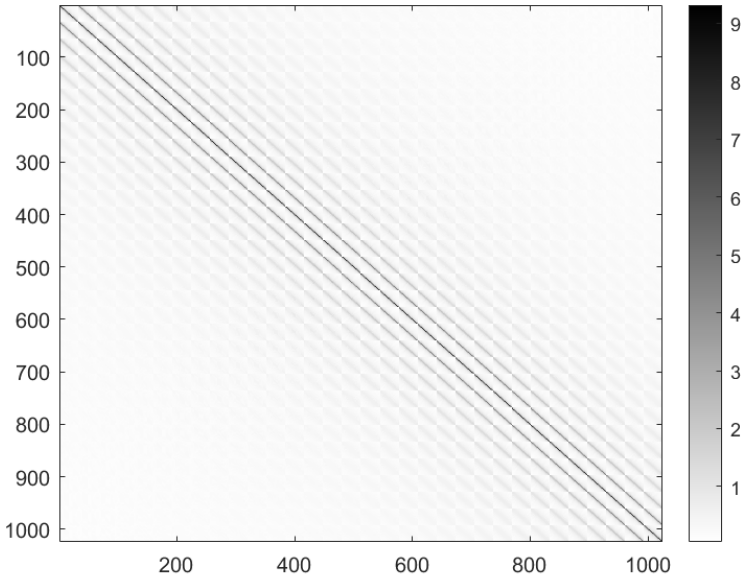
The structure just described suggests that to approximate the matrix $A^T A$ in order to build a preconditioner, we should consider some of these clusters of diagonals. Considering too many of them is undesirable, since applying the preconditioner to a vector could become challenging, and because there is no guarantee that the resulting approximation would be positive definite. The simplest option is to consider only the main diagonal in the approximation. One could also use a penta-diagonal approximation of $A^T A$ (i.e. consider two upper and two lower diagonals), or a nona-diagonal approximation, (i.e. consider four upper and four lower diagonals) ....

Figure 3.3: Connection between the pixels in $A^T A\hat{\mathbf{e}}_i$ with the same distance from the central pixel and the diagonals of $A^T A$ with similar mean value.



Notice that the magnitude of the elements along any diagonal of $A^T A$ depends only on the distance of the corresponding pixel from the central pixel in Figure 3.2c; therefore, the values along any diagonal are almost constant, apart from noise and numerical inaccuracies.

Figure 3.4: Magnitude of the elements of matrix $A^T A$.



3.4 Preconditioner for the normal equations

We assume we can approximate matrix $A^T A$ with a scaled identity that well approximates its diagonal. This is similar to what is done in compressed sensing [52]. Thus, matrix Q_1 (3.6) can be approximated using a 2×2 block matrix with diagonal blocks; adding matrix Q_2 (3.7) and $G^{-1}S$ we get the preconditioner:

$$P = \begin{bmatrix} (c_{11}^2 + c_{21}^2)\rho I_{N^2} + \alpha I_{N^2} + (G^{-1}S)_1 & (c_{11}c_{12} + c_{21}c_{22})\rho I_{N^2} + \beta I_{N^2} \\ (c_{11}c_{12} + c_{21}c_{22})\rho I_{N^2} + \beta I_{N^2} & (c_{12}^2 + c_{22}^2)\rho I_{N^2} + \alpha I_{N^2} + (G^{-1}S)_2 \end{bmatrix}, \tag{3.12}$$

where we have split the entries of $G^{-1}S$ into two blocks; the scalar ρ is an approximation of the diagonal elements of $A^T A$, obtained through random sampling of this matrix. We denote the diagonal blocks as D_{11}, D_{12} and D_{22} according to their position. This preconditioner is easy to invert: when we need to apply it, we have to solve

$$\begin{bmatrix} D_{11} & D_{12} \\ D_{12} & D_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} = \begin{bmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{bmatrix}.$$

This system can be solved forming the Schur complement, which is diagonal:

$$(D_{22} - D_{12}^2 D_{11}^{-1})\mathbf{x_2} = \mathbf{y_2} - D_{12}D_{11}^{-1}\mathbf{y_1}$$

and retrieving $\mathbf{x_1}$ from $\mathbf{x_1} = D_{11}^{-1}(\mathbf{y_1} - D_{12}\mathbf{x_2})$.

Notice that most of the terms involved in the preconditioner are constant, while some vary throughout the IPM iterations, but are immediately available from vectors $\mathbf{g}$ and $\mathbf{s}$. This preconditioner is thus very cheap both to compute

and apply.

Remark 3.1. *Notice that, if $A^L \neq A^H$, the same preconditioner can be used with a small modification: we just need to approximate both the diagonal of $(A^L)^T A^L$ and $(A^H)^T A^H$ with two different coefficients ρ^L and ρ^H.*

In order to use PCG with the proposed preconditioner, we need to show that matrices $Q_1 + Q_2 + G^{-1}S$ and P are positive definite.

Lemma 3.2. *If $\alpha \geq \beta$, $M = Q_1 + Q_2 + G^{-1}S$ and P are symmetric positive definite.*

Proof. From Lemma 3.1 we know that if $\alpha \geq \beta$, matrix Q is positive semi-definite. Matrix $G^{-1}S$ is trivially strictly positive definite, hence M is positive definite.

For P, write it as

$$P = \begin{bmatrix} c_{11}^2 & c_{11}c_{12} \\ c_{11}c_{12} & c_{12}^2 \end{bmatrix} \otimes \rho^L I_{N^2} + \begin{bmatrix} c_{21}^2 & c_{21}c_{22} \\ c_{21}c_{22} & c_{22}^2 \end{bmatrix} \otimes \rho^H I_{N^2} + \begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix} \otimes I_{N^2} + G^{-1}S$$

and proceed in the same way. □

Let us define the matrices

$$F = \begin{bmatrix} f_1 & f_2 \\ f_2 & f_3 \end{bmatrix} \quad K = \begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix} \tag{3.13}$$

where $f_1 = c_{11}^2 + c_{21}^2$, $f_2 = c_{11}c_{12} + c_{21}c_{22}$, $f_3 = c_{12}^2 + c_{22}^2$. We can now analyze the spectrum of the preconditioned matrix:

Lemma 3.3. *The eigenvalues of the preconditioned matrix $P^{-1}M$, where P is defined in (3.12) and $M = Q_1 + Q_2 + G^{-1}S$, when $A^L = A^H = A$ satisfy*

$$\lambda \in \left[\frac{\alpha - \beta}{\rho\Lambda_F + \alpha + \beta}, \frac{\sigma_{max}^2(A)\Lambda_F + \alpha + \beta}{\rho\lambda_F + \alpha - \beta} \right], \tag{3.14}$$

where $\Lambda_F \geq \lambda_F$ are the two eigenvalues of matrix F.

Proof. We want to study the generalized eigenvalue problem $M\mathbf{v} = \lambda P\mathbf{v}$, where

$$M = F \otimes A^T A + K \otimes I_{N^2} + G^{-1}S,$$

$$P = F \otimes \rho I_{N^2} + K \otimes I_{N^2} + G^{-1}S.$$

Let us fix $\|\mathbf{v}\| = 1$. The eigenvalues can be expressed as

$$\lambda = \frac{\mathbf{v}^T M\mathbf{v}}{\mathbf{v}^T P\mathbf{v}} = \frac{\mathbf{v}^T(F \otimes A^T A)\mathbf{v} + \mathbf{v}^T(K \otimes I_{N^2})\mathbf{v} + \mathbf{v}^T(G^{-1}S)\mathbf{v}}{\mathbf{v}^T(F \otimes \rho I_{N^2})\mathbf{v} + \mathbf{v}^T(K \otimes I_{N^2})\mathbf{v} + \mathbf{v}^T(G^{-1}S)\mathbf{v}}.$$

Let us call the eigenvalues of matrix F as $\Lambda_F > \lambda_F \geq 0$, where the last inequality follows from noticing that

$$F = \mathcal{C}^T\mathcal{C}, \quad \text{where} \quad \mathcal{C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

The eigenvalues of K are $\alpha \pm \beta$ and under the assumption $\alpha \geq \beta$, we are sure that this matrix is positive semidefinite.

Using Property 1.3, we can say that:

$$\begin{aligned} \mathbf{v}^T(K \otimes I_{N^2})\mathbf{v} &\in [\alpha - \beta, \alpha + \beta], \\ \mathbf{v}^T(F \otimes \rho I_{N^2})\mathbf{v} &\in [\rho\lambda_F, \rho\Lambda_F], \\ \mathbf{v}^T(F \otimes A^T A)\mathbf{v} &\in [0, \Lambda_F \sigma^2_{\max}(A)]. \end{aligned}$$

Therefore

$$\lambda \leq \frac{\sigma^2_{\max}(A)\Lambda_F + \alpha + \beta + \mathbf{v}^T(G^{-1}S)\mathbf{v}}{\rho\lambda_F + \alpha - \beta + \mathbf{v}^T(G^{-1}S)\mathbf{v}},$$

$$\lambda \geq \frac{\alpha - \beta + \mathbf{v}^T(G^{-1}S)\mathbf{v}}{\rho\Lambda_F + \alpha + \beta + \mathbf{v}^T(G^{-1}S)\mathbf{v}}.$$

Recall the following result: if $A, B, C > 0$ then

$$\frac{A + C}{B + C} \geq \frac{A}{B} \Leftrightarrow B \geq A.$$

It is clear that $\rho\Lambda_F + \alpha + \beta \geq \alpha - \beta$ and that $\sigma^2_{\max}(A)\Lambda_F + \alpha + \beta \geq \rho\lambda_F + \alpha - \beta$, since ρ is an approximation of the mean eigenvalue of $A^T A$ while $\sigma^2_{\max}(A)$ is the maximum one. Thus

$$\lambda \in \left[\frac{\alpha - \beta}{\rho\Lambda_F + \alpha + \beta}, \frac{\sigma^2_{\max}(A)\Lambda_F + \alpha + \beta}{\rho\lambda_F + \alpha - \beta}\right].$$

□

Remark 3.2. *Both these bounds do not depend on the IPM iteration. The lower bound depends only on α, β, the coefficients c_{ij} and ρ, which do not depend on N; hence the lower bound does not depend on N. The upper bound, instead, grows as N increases, since the term $\sigma^2_{max}(A)$ depends on N. Thus, the spectral properties of the preconditioned matrix and the performance of the PCG may deteriorate as N grows.*

A similar result holds in the case $A^L \neq A^H$:

Lemma 3.4. *The eigenvalues of the preconditioned matrix $P^{-1}M$, with $A^L \neq A^H$, satisfy*

$$\lambda \in \left[\frac{\alpha - \beta}{\Lambda_\rho + \alpha + \beta}, \frac{\sigma^2_{max}(A^L)\Lambda_{F_L} + \sigma^2_{max}(A^H)\Lambda_{F_H} + \alpha + \beta}{\lambda_\rho + \alpha - \beta}\right],$$

where λ_ρ, Λ_ρ, Λ_{F_L} and Λ_{F_H} are defined below.

Proof. In this case, the eigenvalue satisfies

$$\lambda = \frac{\mathbf{v}^T(F_L \otimes (A^L)^T A^L)\mathbf{v} + \mathbf{v}^T(F_H \otimes (A^H)^T A^H)\mathbf{v} + \mathbf{v}^T(K \otimes I_{N^2})\mathbf{v} + \mathbf{v}^T(G^{-1}S)\mathbf{v}}{\mathbf{v}^T((\rho_L F_L + \rho_H F_H) \otimes I_{N^2})\mathbf{v} + \mathbf{v}^T(K \otimes I_{N^2})\mathbf{v} + \mathbf{v}^T(G^{-1}S)\mathbf{v}},$$

where

$$F_L = \begin{bmatrix} c_{11}^2 & c_{11}c_{12} \\ c_{11}c_{12} & c_{12}^2 \end{bmatrix}, \quad F_H = \begin{bmatrix} c_{21}^2 & c_{21}c_{22} \\ c_{21}c_{22} & c_{22}^2 \end{bmatrix}.$$

As before, fix $\|\mathbf{v}\| = 1$; we can say that

$$\begin{aligned} \mathbf{v}^T(F_L \otimes (A^L)^T A^L)\mathbf{v} &\in [0, \Lambda_{F_L}\sigma_{\max}^2(A^L)], \\ \mathbf{v}^T(F_H \otimes (A^H)^T A^H)\mathbf{v} &\in [0, \Lambda_{F_H}\sigma_{\max}^2(A^H)], \\ \mathbf{v}^T((\rho_L F_L + \rho_H F_H) \otimes I_{N^2})\mathbf{v} &\in [\lambda_\rho, \Lambda_\rho], \end{aligned}$$

where we have defined

$$\lambda_\rho = \lambda_{\min}(\rho_L F_L + \rho_H F_H), \quad \Lambda_\rho = \lambda_{\max}(\rho_L F_L + \rho_H F_H).$$

Therefore

$$\lambda \leq \frac{\sigma_{\max}^2(A^L)\Lambda_{F_L} + \sigma_{\max}^2(A^H)\Lambda_{F_H} + \alpha + \beta + \mathbf{v}^T(G^{-1}S)\mathbf{v}}{\lambda_\rho + \alpha - \beta + \mathbf{v}^T(G^{-1}S)\mathbf{v}},$$

$$\lambda \geq \frac{\alpha - \beta + \mathbf{v}^T(G^{-1}S)\mathbf{v}}{\Lambda_\rho + \alpha + \beta + \mathbf{v}^T(G^{-1}S)\mathbf{v}}.$$

In the same way as before, the final bound becomes

$$\lambda \in \left[\frac{\alpha - \beta}{\Lambda_\rho + \alpha + \beta}, \frac{\sigma_{\max}^2(A^L)\Lambda_{F_L} + \sigma_{\max}^2(A^H)\Lambda_{F_H} + \alpha + \beta}{\lambda_\rho + \alpha - \beta}\right].$$

□

Notice that the bound presented in (3.14) suggests that, in order to keep the spectral interval narrow, we should choose the regularization parameters α and β such that their difference is large, but their sum is kept small. This is clear also from the bound on the condition number

$$\kappa \leq \frac{\sigma_{\max}^2(A)\Lambda_F + \alpha + \beta}{\rho\lambda_F + \alpha - \beta} \cdot \frac{\rho\Lambda_F + \alpha + \beta}{\alpha - \beta}.$$

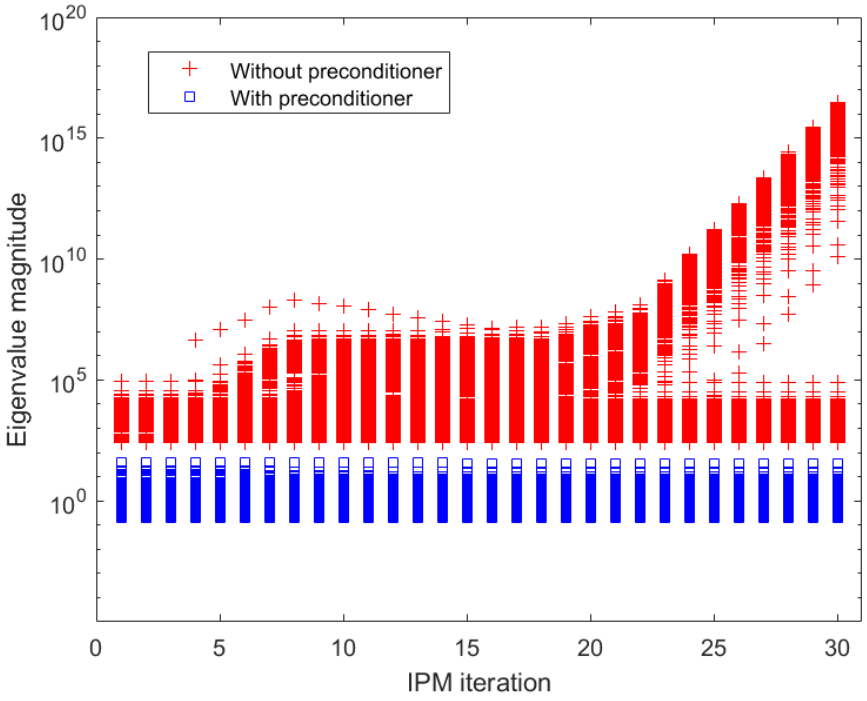
Looking at the definition of the preconditioner (3.12), one can see that the larger the regularization parameters are, the less important it becomes to approximate well matrix $A^T A$. In particular, if the regularization parameters are sufficiently larger than $\rho \cdot \max_{j=1,2,3} f_j$, where f_j are defined in (3.13), then the off-diagonal entries that are dropped from $A^T A$ to form the preconditioner are negligible compared to the diagonal terms that are added.

Therefore, α and β should be chosen large enough to dominate the diagonal of

$A^T A$ and such that $\alpha - \beta$ is as large as possible. Obviously, this analysis is only from the linear algebra point of view: in practice, there needs to be a trade off between the needs of the preconditioner and the requirement to obtain a good quality of the reconstructed images.

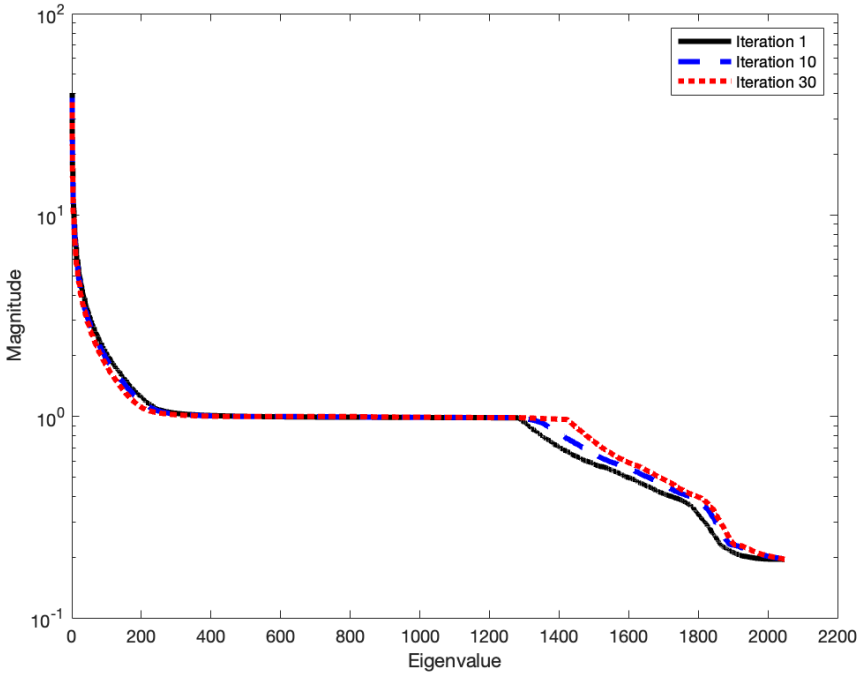
The independence of the spectral interval from the IPM iteration is observed in practice: in Figure 3.5 we plot the eigenvalues of the unpreconditioned (in red) and preconditioned (in blue) normal equations matrix, for one instance with $N = 32$. The eigenvalues of the original matrix are bounded below, due to the regularization, but are unbounded above, and become extremely large when the IPM approaches optimality. The eigenvalues of the preconditioned matrix instead are always lying in almost exactly the same interval.

Figure 3.5: Eigenvalues of the normal equations with and without preconditioner for $N = 32$, $\alpha = 500$, $\beta = 250$



On top of being bounded independently from the IPM iteration, from Figure 3.6 we can appreciate how the cluster of eigenvalues around the value $\lambda = 1$ tends to grow in size when the IPM iterations progress. This is an extremely desirable feature, due to Property 1.6. This phenomenon can be explained noticing that, when the IPM is close to optimality, some of the entries of $G^{-1}S$, that are added to the diagonal of the normal system, become extremely large; this means that the corresponding non-diagonal terms of $A^T A$ (that are ignored when building the preconditioner) become even more negligible and P becomes a better approximation of the matrix.

Figure 3.6: Magnitude of the eigenvalues of the preconditioned normal equations at the $1-$st, $10-$th and $30-$th (final) IPM iteration, for $N = 32$, $\alpha = 500$, $\beta = 250$.



3.5 Results

In this Section, we show the results of applying the proposed regularizer to some test images, with the goal of trying to separate the distribution of two materials. The new regularizer is compared to Joint Total Variation [22] and it shows better results in terms of recognition of the separation of the materials. Results in terms of computational time of the interior point method are postponed after the description of the specialized termination criterion for Krylov solvers inside IPMs; they can be found in Section 4.5.1 of the next Chapter.

For the purpose of numerical testing, all the images have resolution 128×128 pixels, in order to keep the computational time reasonable. When using simulated data, it is important to avoid the so-called *inverse crime* (see [93]), i.e. the use of the same model to generate the simulated data and to reconstruct the solution of the inverse problem. This can happen for instance by using the same angles to produce the synthetic measurement and for the reconstruction, or by ignoring the noise that would be introduced by a real tomographic imaging process. To avoid these issues, the simulated data was generated with a slightly different set of angles than the ones used in the reconstruction process, and random Gaussian noise was added to the simulated measurement. These perturbations coincide with the disturbances that a real tomographic process would introduce. The simulated tomography is produced using 65 angles, equally distributed in the interval 0 to 180 degrees. From each angle, a parallel set of X-ray beams is sent to the image, as shown in Figure 3.1. The number of lines r_0 is automatically chosen by the

Matlab function `radon`, based on the angles and size of the image.

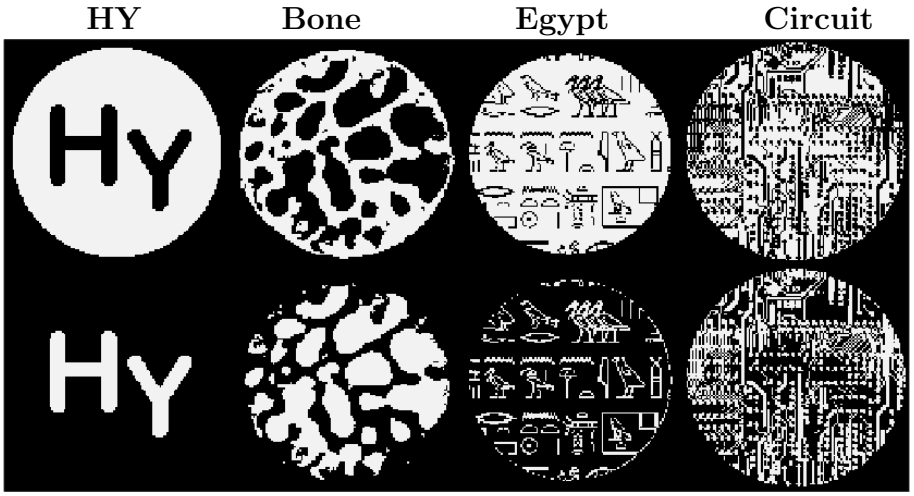
The materials considered are PVC (polyvinyl chloride) and iodine, because they have different behaviours at different energy levels. The attenuation coefficients c_{ij} can be found in Table 3.1

Table 3.1: Attenuation coefficients of PVC and iodine with low and high energies.

	Attenuation coefficient	Material	Tube voltage
c_{11}	1.491	PVC	30 kV
c_{12}	8.561	Iodine	30 kV
c_{21}	0.456	PVC	50 kV
c_{22}	12.32	Iodine	50 kV

The images used for the experiments are shown in Figure 3.7; the two rows present the distribution of the first and second material. These four phantoms contain an increasing level of details; the *Egypt* and *Circuit* images represent an extreme challenge for any reconstruction method, because the two materials are deeply intertwined and are difficult to separate. Notice that in each pixel either the first or the second material is present; therefore, the two rows in Figure 3.7 are perfectly complementary.

Figure 3.7: Images used in the numerical experiments. The first row shows the distribution of PVC and the second row of iodine.



Concerning the comparison with JTV, the competitor method is stopped after 400 iterations and is tuned empirically; an off-the-shelf JTV regularizer is used rather than a specialized one, in order to compare the new method to one of the standard choices for these kind of problems. The parameters for the new regularizer α and β are also chosen empirically, in a way that produces satisfactory reconstructed images (in terms of the quality indicators descrived below), while

guaranteeing a good behaviour of the preconditioner; termination of the IPM happens with a tolerance of 10^{-8}.

In Table 3.2 we report the comparison between the reconstruction obtained with JTV and with the new regularizer, in terms of L_2 error norm, SSIM (structural similarity index measure), HPSI (Haar wavelet-based perceptual similarity index) and fraction of misclassified pixels (i.e. the number of pixels that are classified as the wrong material, out of all the pixels in the image). Notice that for L_2 and misclassification, a smaller number is better, while for SSIM and HPSI a larger number means better quality. The Table shows a slight advantage of the new regularizer in terms of standard quality indices (L_2, SSIM, HPSI) and a clear advantage in terms of correctly identified pixels, in particular regarding the first of the two materials. Given that the goal of the new regularizer is to better separate the materials present in the image, these results show that this purpose has been achieved.

Table 3.2: Results in terms of L_2 error norm, SSIM, HPSI and fraction of misclassified pixels for each phantom and for each material. Where there is a clear winner, it is indicated in bold.

Phantom	Material	Method	L_2	SSIM	HPSI	misclassification
HY	1	JTV	0.30	0.23	0.21	0.05
HY	1	IP	**0.27**	**0.29**	**0.28**	**0.02**
HY	2	JTV	**0.27**	**0.75**	**0.56**	0.01
HY	2	IP	0.28	0.60	0.53	0.01
Bone	1	JTV	0.55	0.24	0.15	0.14
Bone	1	IP	**0.44**	**0.41**	**0.36**	**0.06**
Bone	2	JTV	0.32	0.66	0.50	0.04
Bone	2	IP	**0.29**	**0.71**	0.50	**0.03**
Egypt	1	JTV	0.40	0.25	**0.30**	0.13
Egypt	1	IP	**0.38**	**0.33**	0.29	**0.08**
Egypt	2	JTV	0.62	0.69	0.56	0.06
Egypt	2	IP	**0.61**	0.69	0.56	0.06
Circuit	1	JTV	0.62	0.17	**0.30**	0.28
Circuit	1	IP	**0.56**	**0.32**	0.28	**0.18**
Circuit	2	JTV	0.59	0.59	0.50	0.16
Circuit	2	IP	0.59	**0.62**	0.50	0.16

Figures 3.8a and 3.8b show the reconstruction of the distribution of the two materials for the last two phantoms, obtained with JTV (left column) and IP (middle column) regularizer; the right column instead shows the original phantoms. It is clear how the IP regularizer separates better the materials, especially the

first one. It is worth pointing out the bad behaviour of JTV, which reconstructs the second material much better than the first one: a possible explanation comes from the fact that the method is not tuned to the specific problem considered; this could be alleviated by some special tuning of the parameters of the algorithm; however, for a fair comparison, we preferred to use the plain version of both methods, without any additional tuning. Notice also that the IP method produces a much better reconstruction without needing the tuning that JTV would require.

3.5.1 Effect of the regularization

We also show some results that underline the effect of the newly added penalty term (3.3). We expect this regularizer to create a separation in the vectors $\mathbf{g}^{(1)}$ and $\mathbf{g}^{(2)}$, i.e. we expect the scalar product $\mathbf{g}^{(1),T}\mathbf{g}^{(2)}$ to be pushed close to zero. We performed some tests with different values of β and a fixed value $\alpha = 500$, in the case $N = 64$.

Table 3.3 shows the number of elements of the component-wise products of $\mathbf{g}^{(1)}$ and $\mathbf{g}^{(2)}$ that are smaller than 10^{-6}, and the average value of the same product, i.e. $(\mathbf{g}^{(1),T}\mathbf{g}^{(2)})/N^2$. We can see that as β is increased, the number of small elements grows and the average product decreases, confirming the effect that we expected.

Table 3.3: Number of small elements and average product of $\mathbf{g}^{(1)}$ and $\mathbf{g}^{(2)}$ for different values of β; $\alpha = 500$, $N = 64$.

β	small elements	$(\mathbf{g}^{(1),T}\mathbf{g}^{(2)})/N^2$
50	1056	4.86E3
100	1091	4.07E3
150	1123	3.17E3
200	1161	2.36E3
250	1607	1.54E3
300	2075	1.23E3
350	2210	1.07E3
400	2412	0.93E3
450	2581	0.83E3

3.6 Conclusion

This Chapter presented a specialized interior point method for a tomographic reconstruction problem; the main challenges were given by the size and density of the matrices involved and by the requirement of a matrix-free algorithm. The specific structure of the problem was exploited to obtain a simplified IPM formulation; the structure of the linear operators involved allowed for a simple yet efficient matrix-free preconditioning strategy.

The results suggest that the IPM provides accurate solutions to the optimization problem and the newly proposed regularizer improves the separation of the materials and maintains the other quality indicators at a similar level as JTV, without requiring a dedicated tuning process.

Future research should focus on the following points:

- Extending the methodology to three dimensional tomography: the structure of the problem should not change significantly, but the size of the matrices involved becomes much larger and the IPM and linear solver need to be extremely efficient. Notice that the problems that may be solved with a 3D technique are potentially different: whether to use a 2D slice tomography or a 3D imaging process is highly dependant on the application (e.g. based on the amount of radiation that a patient can receive).

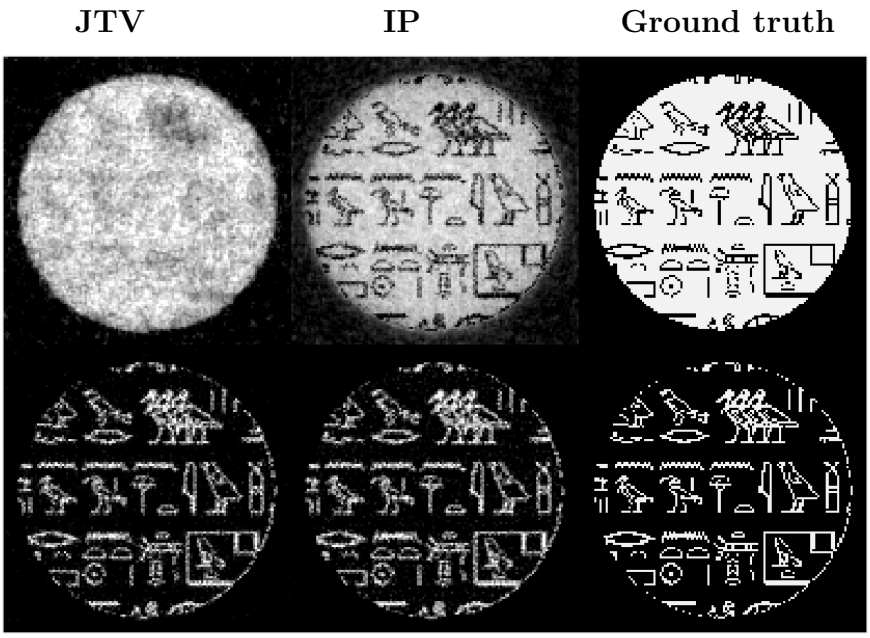
- Extending the regularizer to more than two materials: the preconditioner (3.12) should easily generalize to more than two materials, provided that the regularizer is defined properly.

- Testing the current formulation of real world tomographic images, rather than simulated data, since real applications involve extremely noisy and corrupted data, with imperfections that are difficult to simulate properly.

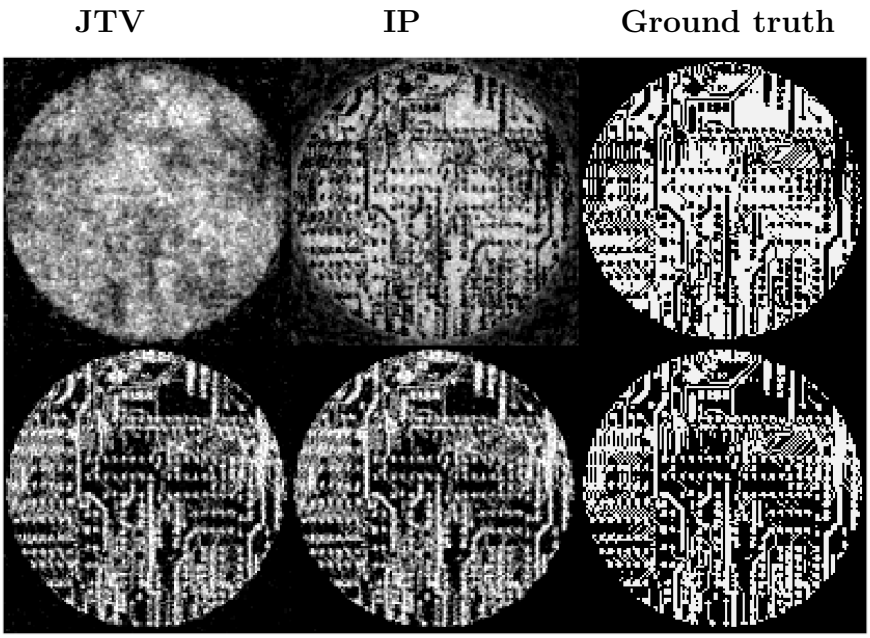
- Improving the preconditioning strategy by using a better approximation of matrix $A^T A$; the difficulty comes from the necessity to include more than one diagonal in the approximation of $A^T A$ and still be able to invert the resulting matrix efficiently.

Figure 3.8: Reconstruction results for the Egypt phantom **(a)** and Circuit phantom **(b)**. The first row represents material 1 and the second row represents material 2.

(a)



(b)



Chapter 4

Early stopping of the linear solver in interior point methods

This Chapter introduces the second main contribution of the Thesis. The work presented in this Chapter is based on [134].

When solving large optimization problems, a key issue of IPMs is the iterative solution of the Newton system (2.9) or (2.10): being able to find an approximate Newton direction in reasonable time is fundamental for the overall efficiency of the algorithm. In this Chapter, we show that the level of inexactness in the Newton direction that can be tolerated by an IPM, without excessively compromising its convergence properties, is much larger than what the most commonly used techniques suggest. Impressive gains in computational time can be obtained by realizing that the Newton direction is only used within the IPM setting and should therefore be computed accordingly.

4.1 Introduction

Interior point methods represent the state-of-the-art for the solution of convex optimization problems. Being second-order methods, they usually converge in merely a few iterations and if the cost of a single iteration is kept small they are able to outperform the first-order methods, especially when it comes to problems of very large dimensions. In these instances, the linear system that arises at each iteration is usually solved with an iterative Krylov subspace method, as presented in Section 1.3. The common approach is to employ a stopping criterion based on the reduction of the residual, i.e. the internal solver is stopped as soon as the initial residual is reduced by a certain predetermined factor. Different strategies have been developed in order to choose a stopping tolerance that allows the outer IPM iterations to converge, without requiring too many inner (linear solver) iterations (e.g. [25, 89, 90]). However, these techniques always rely on a tolerance imposed on the residual of the linear system, although with a varying relative reduction requested.

Such approach does not necessarily represent the best choice, since the overall goal is not obtaining an accurate solution to the sequence of linear systems, but finding a suitable (though inexact) search direction for the optimization problem;

in particular, it may be possible to obtain an inexact Newton direction that would be considered too rough from a purely linear algebra perspective (i.e. its residual still would be too high and any standard stopping criterion would reject it) but that could be good enough to perform the next iteration of IPM successfully (i.e. the direction guarantees sufficient reductions of infeasibilities and the duality gap). Deriving a stop criterion that accepts a direction not based on its *residual*, but based on a *potential improvement* it can bring to the outer IPM iterations could reduce the number of inner iterations required at each outer step, with little or no disadvantage to the overall convergence properties of the IPM.

Specialized early stopping strategies have been used in other fields: in [75, 96, 118] a CG stop criterion is applied to the Jacobi-Davidson eigensolver, when finding eigenvalues of large matrices; in [55] a termination criterion is applied to inverse iterations for solving generalized eigenvalue problems; early stopping is also used in inverse problems and machine learning as a regularizer, to avoid the phenomenon known as semiconvergence (see e.g. [58]); in [7, 11, 47, 116] other stopping criteria are derived for various applications.

In order to obtain an early stopping criterion, specifically designed to be applied in IPMs, that does not rely entirely on the residual of the linear system, the convergence indicators of IPM (i.e. primal and dual infeasibility and complementarity) need to be estimated while performing the inner iterations with CG or MINRES. The main problem is that, to compute the complete primal-dual Newton direction and to compute the infeasibilities, additional matrix-vector products would be required at each inner iteration. Since in general only one matrix-vector product and one preconditioner application per iteration are performed, adding extra matrix applications would considerably slow down the linear solver. Fortunately, with some judicious implementation and exploiting the matrix operations that are already executed, the IPM convergence indicators can be estimated using only vector operations, resulting in a minimal increase in the cost of a single linear iteration.

From the theoretical point of view, this Chapter introduces an ideal stopping criterion that does not rely on the reduction of the residual of the linear system This is novel with respect to the standard literature on inexact IPMs, that mostly focuses on choosing the appropriate sequence of tolerances for each IPM iteration. A sketch of complexity analysis is given together with a rationale as to why the algorithm is expected to perform similarly to the exact version. The main assumptions used are boundedness of the iterates and the ability of the chosen Krylov solver to produce a direction that satisfies the stopping criterion; a rigorous proof of this fact is difficult, due to the complicated interaction between IPM and linear solvers, and is left as an item for further research.

The Chapter also introduces new indicators to estimate the optimal stopping point for the inner linear iterations and analyzes their behaviour in comparison to the residual of the linear system for the problems considered. The empirical evidence suggests a new technique to terminate early the inner iterations, which is mainly based on these new indicators rather than on a sequence of residual tolerances. Although the theoretical and practical stopping criteria are different and the complexity analysis does not directly apply to the method used in the empirical section, the proposed practical termination indicators are strongly

influenced by the theoretical results.

The resulting algorithms for the solution of the linear systems are called *Interior Point Conjugate Gradient* (IPCG) or *Interior Point MINRES* (IPMINRES), depending on the approach chosen: they are specialized for the specific task which needs to be solved and show significant improvements with respect to the standard CG or MINRES on the problems that were considered, which include quadratic programs derived from image processing, compressed sensing and Partial Differential Equation (PDE) constrained optimization. In particular, the new strategy is able to avoid unnecessary inner iterations in the early stage of the IPM, while retaining the good behaviour of the method in its late iterations.

The rest of the Chapter is organized as follows: in Section 4.2 the interior point method is described; in Section 4.3 the new IPCG and IPMINRES iterations, that allow the estimation of the convergence of IPM, are introduced; Section 4.4 introduces a theoretical stopping criterion, for which the complexity analysis is performed, and the new indicators used in practice; in Section 4.5 the test problems and numerical results are presented.

4.2 Interior Point Method

In this Chapter, we consider an IPM applied to a standard pair of primal-dual quadratic programming problems (1.11)-(1.12). As shown in Section 2.1, the Newton direction is found solving the linear system

$$\begin{bmatrix} A & 0 & 0 \\ -Q & A^T & I_n \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \mathbf{\Delta x} \\ \mathbf{\Delta y} \\ \mathbf{\Delta s} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_P \\ \mathbf{r}_D \\ \mathbf{r}_\mu \end{bmatrix} = \begin{bmatrix} \mathbf{b} - A\mathbf{x} \\ \mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mathbf{s} \\ \sigma\mu\mathbf{e}_n - XS\mathbf{e}_n \end{bmatrix}, \tag{4.1}$$

where σ is the parameter responsible for the reduction in the complementarity measure $\mu = (\mathbf{x}^T\mathbf{s})/n$.

System (4.1) is usually reduced to the augmented system

$$\begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{\Delta x} \\ \mathbf{\Delta y} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_D - X^{-1}\mathbf{r}_\mu \\ \mathbf{r}_P \end{bmatrix} \tag{4.2}$$

where $\Theta = XS^{-1}$, and solved using an indefinite factorization or the MINRES method, or it is further reduced to the normal equations

$$A(Q + \Theta^{-1})^{-1}A^T\mathbf{\Delta y} = \mathbf{r}_P + A(Q + \Theta^{-1})^{-1}(\mathbf{r}_D - X^{-1}\mathbf{r}_\mu) \tag{4.3}$$

and solved with a Cholesky factorization or using the Conjugate Gradient method. The direction is then used to compute the stepsize α and to find the next point $(\mathbf{x} + \alpha\mathbf{\Delta x}, \mathbf{y} + \alpha\mathbf{\Delta y}, \mathbf{s} + \alpha\mathbf{\Delta s})$. The outer iterations are stopped as soon as the

approximation satisfies the following IPM stopping criterion, similar to (2.4)

$$\frac{\|\mathbf{b} - A\mathbf{x}\|}{1 + \|\mathbf{b}\|} \leq \tau_P, \qquad \frac{\|\mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mathbf{s}\|}{1 + \|\mathbf{c}\|} \leq \tau_D, \qquad \mu \leq \tau_\mu, \tag{4.4}$$

where τ_P, τ_D and τ_μ are predetermined tolerances.

In this Chapter, we consider the infeasible version of neighbourhood (2.16): at iteration k, the point $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k)$ is in the neighbourhood $\mathcal{N}_\infty(\gamma, \beta)$ if it satisfies

$$(\mathbf{x}^k, \mathbf{s}^k) > 0, \tag{4.5a}$$

$$\gamma\mu^k \leq \mathbf{x}_j^k \mathbf{s}_j^k \leq \mu^k/\gamma, \quad \forall j, \tag{4.5b}$$

$$\|\mathbf{r}_P^k\| \leq \|\mathbf{r}_P^0\|\beta\mu^k/\mu^0, \quad \|\mathbf{r}_D^k\| \leq \|\mathbf{r}_D^0\|\beta\mu^k/\mu^0, \tag{4.5c}$$

where $0 < \gamma < 1$ and $\beta \geq 1$ are two constants chosen at the beginning of the IPM algorithm. This Chapter focuses both on the augmented system approach (4.2), when dealing with generic QPs, and on the normal equations approach (4.3), when dealing with LPs or special cases of QPs.

4.3 Estimating the convergence of the outer iterations

This section provides a description of how to estimate the IPM convergence indicators throughout the CG or MINRES iterations; these quantities are used to terminate the linear iterations prematurely, without relying on the residual of the linear system. The main indicators that are commonly used, as shown in (4.4), are the primal and dual infeasibilities and the complementarity gap. Algorithms and stopping criteria are derived both for CG and MINRES, to be used for LPs and QPs, respectively.

In the case of the normal equations for an LP, the CG is applied to system (4.3) with $Q = 0$; this means that at every inner iteration the approximation for $\mathbf{\Delta y}$ is considered. In order to estimate the IPM indicators, $\mathbf{\Delta x}$ and $\mathbf{\Delta s}$ are also needed, which are computed as follows

$$\mathbf{\Delta x} = (S^{-1}\mathbf{r}_\mu - \Theta\mathbf{r}_D) + \Theta A^T\mathbf{\Delta y},$$

$$\mathbf{\Delta s} = X^{-1}\mathbf{r}_\mu - \Theta^{-1}\mathbf{\Delta x}. \tag{4.6}$$

When using the augmented system instead, $\mathbf{\Delta x}$ and $\mathbf{\Delta y}$ are readily available and just $\mathbf{\Delta s}$ needs to be computed.

These two formulas contain a first term, which is constant during the inner iterations, and a second term which varies as the Krylov method progresses. Once the full direction is known, the step to the boundary can be computed as

$$\alpha_x^{\max} = \min_{j:\, \mathbf{\Delta x}_j < 0} -\frac{\mathbf{x}_j}{\mathbf{\Delta x}_j}, \quad \alpha_s^{\max} = \min_{j:\, \mathbf{\Delta s}_j < 0} -\frac{\mathbf{s}_j}{\mathbf{\Delta s}_j}. \tag{4.7}$$

To determine primal and dual stepsizes, at each iteration a practical IPM algorithm

uses a fraction of the maximum step to the boundary. It is computed as in (4.7), scaled by a certain factor (e.g. 0.995) to guarantee that each point is in the interior of the feasible region. In this way the stepsizes are

$$\alpha_x = 0.995\,\alpha_x^{\max}, \quad \alpha_s = 0.995\,\alpha_s^{\max}. \tag{4.8}$$

4.3.1 IPCG for LP

Consider now the normal equations approach for an LP (i.e. $Q = 0$). Suppose the algorithm stops the CG at a certain iteration for which the full direction $(\boldsymbol{\Delta x}, \boldsymbol{\Delta y}, \boldsymbol{\Delta s})$ and the stepsizes α_x and α_s have been computed. Let us indicate the new point by $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{s}})$, then the infeasibilities can be written as

$$A\bar{\mathbf{x}} - \mathbf{b} = (A\mathbf{x} - \mathbf{b}) + \alpha_x\Big((A\Theta A^T\boldsymbol{\Delta y}) + (AS^{-1}\mathbf{r}_\mu - A\Theta\mathbf{r}_D)\Big),$$

$$A^T\bar{\mathbf{y}} + \bar{\mathbf{s}} - \mathbf{c} = (A^T\mathbf{y} + \mathbf{s} - \mathbf{c}) + \alpha_s(A^T\boldsymbol{\Delta y} + \boldsymbol{\Delta s}).$$

The problematic terms in these formulas are given by $A^T\boldsymbol{\Delta y}$ and $A\Theta A^T\boldsymbol{\Delta y}$: computing these quantities at each linear iteration would require extra matrix operations to be performed. Define the vectors $\mathbf{v}_1 = X^{-1}\mathbf{r}_\mu$, $\mathbf{v}_2 = S^{-1}\mathbf{r}_\mu - \Theta\mathbf{r}_D$, $\mathbf{v}_3 = AS^{-1}\mathbf{r}_\mu - A\Theta\mathbf{r}_D$, $\boldsymbol{\xi}_1 = A^T\boldsymbol{\Delta y}$, $\boldsymbol{\xi}_2 = A\Theta A^T\boldsymbol{\Delta y}$; then, the previous expressions become

$$\boldsymbol{\Delta x} = \mathbf{v}_2 + \Theta\boldsymbol{\xi}_1, \qquad \boldsymbol{\Delta s} = \mathbf{v}_1 - \Theta^{-1}\boldsymbol{\Delta x}, \tag{4.9}$$

$$A\bar{\mathbf{x}} - \mathbf{b} = (A\mathbf{x} - \mathbf{b}) + \alpha_x(\boldsymbol{\xi}_2 + \mathbf{v}_3), \tag{4.10}$$

$$A^T\bar{\mathbf{y}} + \bar{\mathbf{s}} - \mathbf{c} = (A^T\mathbf{y} + \mathbf{s} - \mathbf{c}) + \alpha_s(\boldsymbol{\xi}_1 + \boldsymbol{\Delta s}). \tag{4.11}$$

Vectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ remain constant during the CG iterations and can be computed once at the beginning of the algorithm. Recall from Section 1.3.1 that, during the CG process, the approximation $\boldsymbol{\Delta y}$ is updated as

$$\boldsymbol{\Delta y} \leftarrow \boldsymbol{\Delta y} + \alpha^{CG}\mathbf{u}$$

where α^{CG} is the CG stepsize and $\mathbf{u}$ is the CG direction. Therefore, the quantities $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ can be updated in a similar way:

$$\boldsymbol{\xi}_1 \leftarrow \boldsymbol{\xi}_1 + \alpha^{CG}A^T\mathbf{u}, \qquad \boldsymbol{\xi}_2 \leftarrow \boldsymbol{\xi}_2 + \alpha^{CG}A\Theta A^T\mathbf{u}.$$

The quantity $A\Theta A^T\mathbf{u}$ is already computed during the CG algorithm, because it is needed to find the stepsize α^{CG} and to update the residual. While computing it, the quantity $A^T\mathbf{u}$ can be obtained as a byproduct:

$$\mathbf{w}_1 = A^T\mathbf{u}, \qquad \mathbf{w}_2 = A\Theta\mathbf{w}_1.$$

In this way, it is possible to update the quantities $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ at each inner iteration inexpensively, which in turn allows to compute the IPM convergence indicators at each CG iteration using only vector operations. Notice that the products with matrix Θ needed to compute the directions in (4.9), in practice are performed as vector operations, since Θ is diagonal. Notice also that $\mathbf{w}_1$ and $\mathbf{w}_2$ need to

be computed at the beginning of the CG process, to initialize the residual; thus, initializing $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ does not add operations. However, one single matrix-vector product with matrix A is added at the beginning of the algorithm, to compute the constant vector $\mathbf{v}_3$. Algorithm IPCG summarizes the process just described: the main differences with the standard Algorithm PCG are in lines 2, 17, 18, 26, 27, 28. The IPM convergence indicators are estimated only after a number `itstart` of iterations. The algorithm does not contain any stopping criterion for now, it simply computes the primal and dual infeasibilities and the duality gap at each CG iteration, if the CG process was stopped at that iteration. Other indicators can also be computed if needed, as will be clear in the next sections. The choice of the stopping criterion based on these indicators will be discussed in the following sections.

At each iteration, the standard CG algorithm performs one matrix-vector product, one preconditioner application, two scalar products and three `axpy` operations; what is added in the IPCG algorithm requires, at each iteration, the equivalent of three scalar products (to compute μ, $\Theta\boldsymbol{\xi}_1$, $\Theta^{-1}\boldsymbol{\Delta}\mathbf{x}$), approximately ten `axpy` operations and the computation of the stepsizes (which are computed as in (4.7)-(4.8) and thus involve only comparison of vector components and component-wise divisions). Therefore, we expect the computational cost of the IPCG iteration to be only slightly larger than that of the standard CG step, especially if the applications of the matrix or the preconditioner are particularly expensive.

Remark 4.1. *Notice that, when using a predictor-corrector strategy, the algorithm just proposed works only when computing the predictor direction. For the corrector, equations (4.10)-(4.11) need to be modified. In particular, calling* $(\boldsymbol{\Delta}\mathbf{x}^P, \boldsymbol{\Delta}\mathbf{y}^P, \boldsymbol{\Delta}\mathbf{s}^P)$ *the predictor computed previously, the terms to add are* $A\boldsymbol{\Delta}\mathbf{x}^P$ *to the expression for the primal residual (4.10) and* $A^T\boldsymbol{\Delta}\mathbf{y}^P + \boldsymbol{\Delta}\mathbf{s}^P$ *to the expression for the dual residual (4.11); they can be computed at the beginning since they are constant, but they add matrix operations to be performed at every call of the algorithm. Alternatively, these can be avoided by saving the final values of the vectors* $\boldsymbol{\xi}_1$ *and* $\boldsymbol{\xi}_2$ *from the previous IPCG call that computed the predictor direction.*

Notice also that sometimes the linear system to be solved at each IPM iteration has a different structure, for example because the normal equations are further reduced by an additional elimination step. In such cases, the proposed technique to estimate IPM-related quantities throughout the CG iterations may still be applicable in a similar way. However, in certain situations it may not be possible to do so without requiring additional matrix-vector products to be computed.

4.3.2 IPMINRES for QP

Similarly, in the case of the augmented system for a QP, the infeasibilities can be written as

$$A\bar{\mathbf{x}} - \mathbf{b} = (A\mathbf{x} - \mathbf{b}) + \alpha_x(A\boldsymbol{\Delta}\mathbf{x})$$
$$A^T\bar{\mathbf{y}} + \bar{\mathbf{s}} - Q\bar{\mathbf{x}} - \mathbf{c} = (A^T\mathbf{y} + \mathbf{s} - Q\mathbf{x} - \mathbf{c}) + \alpha_s(A^T\boldsymbol{\Delta}\mathbf{y} + \boldsymbol{\Delta}\mathbf{s}) - \alpha_x(Q\boldsymbol{\Delta}\mathbf{x}).$$

Algorithm IPCG Interior Point Conjugate Gradient method

Input: rhs $\mathbf{f}$, tolerance τ_{inner}, max iterations `itmax`, matrices A, Θ, preconditioner P, initial approximation $\boldsymbol{\Delta}\mathbf{y}$, minimum iterations `itstart`
Input from IPM: current point $(\mathbf{x}, \mathbf{y}, \mathbf{s})$, vectors $\mathbf{r}_P, \mathbf{r}_D, \mathbf{r}_\mu$

1: **Initialize:**
2: $\mathbf{v}_1 = X^{-1}\mathbf{r}_\mu$, $\mathbf{v}_2 = \Theta(\mathbf{v}_1 - \mathbf{r}_D)$, $\mathbf{v}_3 = A\mathbf{v}_2$
3: $\boldsymbol{\xi}_1 = A^T\boldsymbol{\Delta}\mathbf{y}$
4: $\boldsymbol{\xi}_2 = A\Theta\boldsymbol{\xi}_1$
5: $\mathbf{r}_0 = \mathbf{f} - \boldsymbol{\xi}_2$
6: $\mathbf{r} = \mathbf{r}_0$
7: $\mathbf{z} = P^{-1}\mathbf{r}$
8: $\mathbf{u} = \mathbf{z}$
9: $\rho = \mathbf{r}^T\mathbf{z}$
10: `iter` $= 0$
11: **while** $\|\mathbf{r}\| > \tau_{\text{inner}}\|\mathbf{r}_0\|$ and `iter` $<$ `itmax` **do**
12: `iter` $=$ `iter` $+ 1$
13: $\mathbf{w}_1 = A^T\mathbf{u}$
14: $\mathbf{w}_2 = A\Theta\mathbf{w}_1$
15: $\alpha^{CG} = \rho/\mathbf{w}_2^T\mathbf{u}$
16: $\boldsymbol{\Delta}\mathbf{y} = \boldsymbol{\Delta}\mathbf{y} + \alpha^{CG}\mathbf{u}$
17: $\boldsymbol{\xi}_1 = \boldsymbol{\xi}_1 + \alpha^{CG}\mathbf{w}_1$
18: $\boldsymbol{\xi}_2 = \boldsymbol{\xi}_2 + \alpha^{CG}\mathbf{w}_2$
19: $\mathbf{r} = \mathbf{r} - \alpha^{CG}\mathbf{w}_2$
20: $\mathbf{z} = P^{-1}\mathbf{r}$
21: $\rho^N = \mathbf{r}^T\mathbf{z}$
22: $\beta = \rho^N/\rho$
23: $\mathbf{u} = \mathbf{z} + \beta\mathbf{u}$
24: $\rho = \rho^N$
25: **if** (`iter` $\geq$ `itstart`) **then**
26: Compute Newton directions: $\boldsymbol{\Delta}\mathbf{x} = \mathbf{v}_2 + \Theta\boldsymbol{\xi}_1, \quad \boldsymbol{\Delta}\mathbf{s} = \mathbf{v}_1 - \Theta^{-1}\boldsymbol{\Delta}\mathbf{x}$
27: Compute stepsizes α_x, α_s using $\mathbf{x}$, $\boldsymbol{\Delta}\mathbf{x}$, $\mathbf{s}$, $\boldsymbol{\Delta}\mathbf{s}$
28: Compute convergence indicators:

$$\mathbf{p}_{\text{inf}} = -\mathbf{r}_P + \alpha_x(\boldsymbol{\xi}_2 + \mathbf{v}_3)$$

$$\mathbf{d}_{\text{inf}} = -\mathbf{r}_D + \alpha_s(\boldsymbol{\xi}_1 + \boldsymbol{\Delta}\mathbf{s})$$

$$\mu = (\mathbf{x} + \alpha_x\boldsymbol{\Delta}\mathbf{x})^T(\mathbf{s} + \alpha_s\boldsymbol{\Delta}\mathbf{s})$$

29: **end if**
30: **end while**

Thus, at each inner iteration, the quantities to update are $\boldsymbol{\xi}_x = A\boldsymbol{\Delta}\mathbf{x}$, $\boldsymbol{\xi}_y = A^T\boldsymbol{\Delta}\mathbf{y}$, $\boldsymbol{\xi}_Q = Q\boldsymbol{\Delta}\mathbf{x}$; this can be done at little extra cost by exploiting the matrix-vector products already present in the MINRES algorithm, similarly to what was done earlier for the CG. The implementation is slightly more complicated, since the MINRES updates the approximation using the two previous iterations; Algorithm IPMINRES shows the standard MINRES method, according to the implementation in [4], with the additional operations required: the main differences with the standard Algorithm MINRES are in lines 3, 7, 17, 19, 21, 23, 24, 25. The estimation of the residual is more complicated than in the CG case and it is not shown to avoid further over-complicating of the displayed algorithm and because it is not affected by the new approach. As before, the additional cost is

Algorithm IPMINRES Interior Point Minimum Residual method

Input: rhs $\mathbf{f}$, tolerance τ_{inner}, max iterations `itmax`, matrices A, Θ,Q, preconditioner P, minimum iterations `itstart`

Input from IPM: current point $(\mathbf{x}, \mathbf{y}, \mathbf{s})$, vectors $\mathbf{r}_P, \mathbf{r}_D, \mathbf{r}_\mu$

Initialize:

$\boldsymbol{\psi} = P^{-1}f$, $\mathbf{r}_1 = \mathbf{f}$, $\mathbf{r}_2 = \mathbf{r}_1$, $\beta = \sqrt{\mathbf{f}^T\boldsymbol{\psi}}$, $\mathbf{w} = 0$, $\mathbf{w}_2 = 0$, $c_s = -1$, $s_n = 0$, $\bar{\varphi} = \beta$, $\epsilon = 0$, $\boldsymbol{\Delta} = 0$, `iter` $= 0$

$\mathbf{w}^v = 0$, $\mathbf{w}_2^v = 0$, $\boldsymbol{\xi} = 0$, $\boldsymbol{\zeta} = X^{-1}\mathbf{r}_\mu$

while `residual` $> \tau_{\text{inner}}\|$`residual`$_0\|$ and `iter` $<$ `itmax` **do**

`iter` $=$ `iter` $+ 1$

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} = \frac{1}{\beta}\boldsymbol{\psi}$$

$$\boldsymbol{\psi} = \begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}, \text{ with byproduct } \mathbf{z}^v = \begin{bmatrix} Q\mathbf{v}_1 \\ A\mathbf{v}_1 \\ A^T\mathbf{v}_2 \end{bmatrix}$$

if `iter` ≥ 2 **then** $\boldsymbol{\psi} = \boldsymbol{\psi} - (\beta/\beta_0)\mathbf{r}_1$ **end if**

$\alpha = \mathbf{v}^T\boldsymbol{\psi}$

$\boldsymbol{\psi} = \boldsymbol{\psi} - (\alpha/\beta)\mathbf{r}_2$

$\mathbf{r}_1 = \mathbf{r}_2$, $\mathbf{r}_2 = \boldsymbol{\psi}$

$\boldsymbol{\psi} = P^{-1}\mathbf{r}_2$

$\beta_0 = \beta$, $\beta = \sqrt{\mathbf{r}_2^T\boldsymbol{\psi}}$

$\epsilon_0 = \epsilon$, $\delta = c_s\bar{\delta} + s_n\alpha$, $\bar{g} = s_n\bar{\delta} - c_s\alpha$, $\epsilon = s_n\beta$, $\bar{\delta} = -c_s\beta$, $r = \sqrt{\bar{g}^2 + \bar{\delta}^2}$

$\gamma = \max(\sqrt{\bar{g}^2 + \beta^2}, \epsilon)$

$c_s = \bar{g}/\gamma$, $s_n = \beta/\gamma$, $\varphi = c_s\bar{\varphi}$, $\bar{\varphi} = s_n\bar{\varphi}$

$\mathbf{w}_1 = \mathbf{w}_2$, $\mathbf{w}_2 = \mathbf{w}$, $\mathbf{w}_1^v = \mathbf{w}_2^v$, $\mathbf{w}_2^v = \mathbf{w}^v$

$\mathbf{w} = (\mathbf{v} - \epsilon_0\mathbf{w}_1 - \delta\mathbf{w}_2)/\gamma$

$\mathbf{w}^v = (\mathbf{z}^v - \epsilon_0\mathbf{w}_1^v - \delta\mathbf{w}_2^v)/\gamma$

$$\Delta = \begin{bmatrix} \boldsymbol{\Delta}\mathbf{x} \\ \boldsymbol{\Delta}\mathbf{y} \end{bmatrix} = \Delta + \varphi\mathbf{w}$$

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\xi}_Q \\ \boldsymbol{\xi}_x \\ \boldsymbol{\xi}_y \end{bmatrix} = \boldsymbol{\xi} + \varphi\mathbf{w}^v$$

if (`iter` $\geq$ `itstart`) **then**

Compute Newton direction: $\boldsymbol{\Delta}\mathbf{s} = \boldsymbol{\zeta} - \Theta^{-1}\boldsymbol{\Delta}\mathbf{x}$

Compute stepsizes α_x, α_s using $\mathbf{x}$, $\boldsymbol{\Delta}\mathbf{x}$, $\mathbf{s}$, $\boldsymbol{\Delta}\mathbf{s}$

Compute convergence indicators:

$$\mathbf{p}_{\text{inf}} = -\mathbf{r}_P + \alpha_x\boldsymbol{\xi}_x$$

$$\mathbf{d}_{\text{inf}} = -\mathbf{r}_D + \alpha_s(\boldsymbol{\xi}_y + \boldsymbol{\Delta}\mathbf{s}) - \alpha_x\boldsymbol{\xi}_Q$$

$$\mu = (\mathbf{x} + \alpha_x\boldsymbol{\Delta}\mathbf{x})^T(\mathbf{s} + \alpha_s\boldsymbol{\Delta}\mathbf{s})$$

end if

end while

given only by vector operations (scalar products, `axpy` operations and stepsizes

computation).

Remark 4.2. *The letter α has been used to indicate multiple stepsizes related to IPM and CG. Since these are the standard notations in both fields, we did not change them but added subscripts and superscripts to identify them (α_x and α_s for the IPM stepsizes, while α^{CG} for the CG stepsize).*

4.4 Stopping criterion

The next section provides some arguments for the complexity analysis of the proposed method. Due to the complex interaction between the IPM and the linear solvers, a full polynomial complexity proof is not presented; instead, a rationale is provided which suggests that the proposed inexact algorithm should have only slightly weaker complexity than the exact method.

4.4.1 Complexity analysis

This section will follow [131, chapter 6], with the difference that here the problem is a quadratic program. We make some standard assumptions: the relative interior of problems (1.11)-(1.12) is non-empty; the neighbourhood is defined by (4.5); the parameter σ_k is chosen in the interval $[\sigma_{\min}, \sigma_{\max}]$, $\sigma_{\max} \leq 1$; a single stepsize α_k is considered instead of two different ones for the primal and dual direction; the stepsize is chosen such that the next point is inside the central path neighbourhood and it satisfies the Armijo condition

$$\mu_{k+1} \leq (1 - 0.01\alpha_k)\mu_k. \tag{4.12}$$

It is already known (see e.g. [131]) that, when dealing with an LP and using an exact method to find the direction, there is a minimum stepsize that can be taken, $\alpha_k \geq \bar{\alpha}$; this fact, together with the Armijo condition above, guarantees convergence of the algorithm. It is also known that both the primal and dual infeasibilities are reduced by a factor $(1 - \alpha_k)$. Moreover, the third equation in (4.1) yields

$$\frac{\mathbf{\Delta x}_j^k}{\mathbf{x}_j^k} + \frac{\mathbf{\Delta s}_j^k}{\mathbf{s}_j^k} = -1 + \frac{\sigma_k \mu_k}{\mathbf{x}_j^k \mathbf{s}_j^k}, \quad \forall j. \tag{4.13}$$

Notice that the right hand side in the last equation is $\mathcal{O}(1)$, due to the symmetric neighbourhood used (4.5b). Therefore, given these facts, the stopping criterion is chosen as follows: the direction produced by the inner solver is accepted as soon as

$$\max_j \left| \frac{\mathbf{\Delta x}_j^k}{\mathbf{x}_j^k} \right| \leq M, \qquad \max_j \left| \frac{\mathbf{\Delta s}_j^k}{\mathbf{s}_j^k} \right| \leq M \tag{4.14}$$

for some fixed constant M. Moreover, it should also be required that

$$\|\mathbf{r}_P^{k+1}\| \leq \eta_{k+1} \|\mathbf{r}_P^k\|, \quad \|\mathbf{r}_D^{k+1}\| \leq \eta_{k+1} \|\mathbf{r}_D^k\|, \tag{4.15}$$

where $1 > \eta_{k+1} \geq 1 - \alpha_k$, since an inexact direction cannot reasonably perform as well as the exact one. Thus, suppose that $\eta_{k+1} = 1 - \omega_{k+1}\alpha_k$, for some $\omega_{k+1} \leq 1$; the choice of ω_{k+1} will be clarified in the next Lemma. Notice also that the equation

$$S^k \mathbf{\Delta x}^k + X^k \mathbf{\Delta s}^k = \sigma_k \mu_k \mathbf{e}_n - X^k S^k \mathbf{e}_n \tag{4.16}$$

continues to hold even if the direction is inexact; this is because $\mathbf{\Delta s}$ is calculated from $\mathbf{\Delta x}$ when employing the normal equations or the augmented system, as shown in (4.6). Algorithm IPM-I summarizes the choices made here and shows also some other features that will be clear during the proof of Lemma 4.1.

Remark 4.3. *In the algorithm, it may seem like the stepsize α_k is used at step 6 but is computed only at step 7. This happens because the stepsize is estimated at every inner iteration, as shown in Algorithms IPCG and IPMINRES. To avoid confusion, the estimated stepsize in step 6 is denoted as $\hat{\alpha}_k$.*

Algorithm IPM-I Interior Point Method with early stopping of the linear solver

Input: $\gamma \in [0, 1]$, $\beta \geq 1$, $0 < \delta < \sigma_{\min} < \sigma_{\max} \leq 1$, $M > 0$

1: Choose $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0)$ with $(\mathbf{x}^0, \mathbf{s}^0) > 0$
2: **while** (4.4) is not satisfied **do**
3: Choose $\sigma_k \in [\sigma_{\min}, \sigma_{\max}]$
4: Choose $\omega_k \in [1 - \sigma_k + \delta, 1]$
5: Compute $\mu_k = (\mathbf{x}^k)^T \mathbf{s}^k / n$
6: Find a direction $(\mathbf{\Delta x}^k, \mathbf{\Delta y}^k, \mathbf{\Delta s}^k)$ such that

$$\max_j \left| \frac{\mathbf{\Delta x}_j^k}{\mathbf{x}_j^k} \right| \leq M, \qquad \max_j \left| \frac{\mathbf{\Delta s}_j^k}{\mathbf{s}_j^k} \right| \leq M,$$

$$\|\mathbf{r}_P^k\| \leq (1 - \omega_k \hat{\alpha}_k) \|\mathbf{r}_P^{k-1}\|, \quad \|\mathbf{r}_D^k\| \leq (1 - \omega_k \hat{\alpha}_k) \|\mathbf{r}_D^{k-1}\|,$$

$$S^k \mathbf{\Delta x}^k + X^k \mathbf{\Delta s}^k = \sigma_k \mu_k \mathbf{e}_n - X^k S^k \mathbf{e}_n.$$

7: Choose α_k as the largest $\alpha \in [0, 1]$ such that

$$(\mathbf{x}^k + \alpha \mathbf{\Delta x}^k, \mathbf{y}^k + \alpha \mathbf{\Delta y}^k, \mathbf{s}^k + \alpha \mathbf{\Delta s}^k) \in \mathcal{N}_\infty(\gamma, \beta),$$

$$(\mathbf{x}^k + \alpha \mathbf{\Delta x}^k)^T (\mathbf{s}^k + \alpha \mathbf{\Delta s}^k) \leq (1 - 0.01\alpha)(\mathbf{x}^k)^T (\mathbf{s}^k).$$

8: Set

$$(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}) = (\mathbf{x}^k + \alpha_k \mathbf{\Delta x}^k, \mathbf{y}^k + \alpha_k \mathbf{\Delta y}^k, \mathbf{s}^k + \alpha_k \mathbf{\Delta s}^k)$$

9: **end while**

The Lemma below asserts that, if the direction is chosen using the stopping criterion defined by (4.14)-(4.15), then there exists a minimum stepsize $\tilde{\alpha}$ such that a new iterate after a step in the (inexact) Newton direction belongs to the $\mathcal{N}_\infty(\gamma, \beta)$ neighbourhood and delivers a guaranteed reduction of the complementarity product. In the following, the iteration index k will be omitted, for sake of clarity.

Let us show some results that are useful to prove the Lemma 4.1. Start by noticing that (4.14) implies that the positivity constraints $\mathbf{x} + \alpha \mathbf{\Delta x} > 0$ and $\mathbf{s} + \alpha \mathbf{\Delta s} > 0$ are automatically satisfied for any $\alpha \in [0, \frac{1}{M}[$. Moreover, the following

bounds hold

$$|\Delta \mathbf{x}_j \Delta \mathbf{s}_j| = \left|\frac{\Delta \mathbf{x}_j}{\mathbf{x}_j}\right|\left|\frac{\Delta \mathbf{s}_j}{\mathbf{s}_j}\right|\mathbf{x}_j\mathbf{s}_j \leq M^2\frac{\mu}{\gamma} \tag{4.17}$$

$$|\Delta \mathbf{x}^T \Delta \mathbf{s}| = \left|\sum_{j=1}^{n} \Delta \mathbf{x}_j \Delta \mathbf{s}_j\right| \leq \sum_{j=1}^{n}\left|\frac{\Delta \mathbf{x}_j}{\mathbf{x}_j}\right|\left|\frac{\Delta \mathbf{s}_j}{\mathbf{s}_j}\right|\mathbf{x}_j\mathbf{s}_j \leq M^2 n\mu \tag{4.18}$$

Lemma 4.1. *Consider an IPM algorithm where each direction satisfies conditions (4.14)-(4.15)-(4.16). Suppose that ω is chosen such that $\omega \geq 1 - \sigma + \delta$, where $\delta < \sigma_{min}$ is a constant.*

Then, there exists a value $\tilde{\alpha} \in (0, 1)$ such that the following conditions are satisfied for all $\alpha \in [0, \tilde{\alpha}]$ at each IPM iteration and for all components j:

$$(\mathbf{x}_j + \alpha\Delta\mathbf{x}_j)(\mathbf{s}_j + \alpha\Delta\mathbf{s}_j) \geq \gamma(\mathbf{x} + \alpha\Delta\mathbf{x})^T(\mathbf{s} + \alpha\Delta\mathbf{s})/n, \tag{4.19a}$$

$$(\mathbf{x}_j + \alpha\Delta\mathbf{x}_j)(\mathbf{s}_j + \alpha\Delta\mathbf{s}_j) \leq (1/\gamma)(\mathbf{x} + \alpha\Delta\mathbf{x})^T(\mathbf{s} + \alpha\Delta\mathbf{s})/n, \tag{4.19b}$$

$$(\mathbf{x} + \alpha\Delta\mathbf{x})^T(\mathbf{s} + \alpha\Delta\mathbf{s})/n \leq (1 - 0.01\alpha)\mu, \tag{4.19c}$$

$$(\mathbf{x} + \alpha\Delta\mathbf{x})^T(\mathbf{s} + \alpha\Delta\mathbf{s}) \geq \eta\mathbf{x}^T\mathbf{s}. \tag{4.19d}$$

In particular

$$\tilde{\alpha} = \min\left(\frac{\sigma_{min}\gamma(1-\gamma)}{M^2(1+\gamma^2)}, \frac{\sigma_{min}(1-\gamma)}{2M^2}, \frac{0.99 - \sigma_{max}}{M^2}, \frac{\delta}{M^2}, 1\right). \tag{4.20}$$

Proof. Using (4.16), one can show that

$$(\mathbf{x}_j + \alpha\Delta\mathbf{x}_j)(\mathbf{s}_j + \alpha\Delta\mathbf{s}_j) = \mathbf{x}_j\mathbf{s}_j(1-\alpha) + \alpha\sigma\mu + \alpha^2\Delta\mathbf{x}_j\Delta\mathbf{s}_j$$

and

$$(\mathbf{x} + \alpha\Delta\mathbf{x})^T(\mathbf{s} + \alpha\Delta\mathbf{s})/n = \frac{1}{n}\Big[\mathbf{x}^T\mathbf{s}(1-\alpha) + \alpha\sigma\mu n + \alpha^2\Delta\mathbf{x}^T\Delta\mathbf{s}\Big].$$

Using (4.17), (4.18) and (4.5b), the previous equalities become

$$(\mathbf{x}_j + \alpha\Delta\mathbf{x}_j)(\mathbf{s}_j + \alpha\Delta\mathbf{s}_j) \geq (1-\alpha)\gamma\mu + \alpha\sigma\mu - \alpha^2M^2\mu/\gamma, \tag{4.21a}$$

$$(\mathbf{x}_j + \alpha\Delta\mathbf{x}_j)(\mathbf{s}_j + \alpha\Delta\mathbf{s}_j) \leq (1-\alpha)\mu/\gamma + \alpha\sigma\mu + \alpha^2M^2\mu/\gamma, \tag{4.21b}$$

$$(\mathbf{x} + \alpha\Delta\mathbf{x})^T(\mathbf{s} + \alpha\Delta\mathbf{s})/n \geq (1-\alpha)\mu + \alpha\sigma\mu - \alpha^2M^2\mu, \tag{4.21c}$$

$$(\mathbf{x} + \alpha\Delta\mathbf{x})^T(\mathbf{s} + \alpha\Delta\mathbf{s})/n \leq (1-\alpha)\mu + \alpha\sigma\mu + \alpha^2M^2\mu. \tag{4.21d}$$

Using (4.21a) and (4.21d), it follows that

$$\begin{aligned}
&(\mathbf{x}_j + \alpha\Delta\mathbf{x}_j)(\mathbf{s}_j + \alpha\Delta\mathbf{s}_j) - \gamma(\mathbf{x} + \alpha\Delta\mathbf{x})^T(\mathbf{s} + \alpha\Delta\mathbf{s})/n \geq \\
&\geq (1-\alpha)\gamma\mu + \alpha\sigma\mu - \alpha^2M^2\frac{\mu}{\gamma} - \gamma((1-\alpha)\mu + \alpha\sigma\mu + \alpha^2M^2\mu) \geq \\
&\geq \alpha\sigma_{\min}\mu(1-\gamma) - \alpha^2M^2\mu(\gamma + 1/\gamma)
\end{aligned}$$

and thus (4.19a) is satisfied if the final expression is non-negative, i.e.

$$\alpha \leq \frac{\sigma_{\min}\gamma(1-\gamma)}{M^2(1+\gamma^2)}.$$

Using (4.21b) and (4.21c), we obtain

$$\begin{aligned}
&(1/\gamma)(x+\alpha\Delta x)^T(s+\alpha\Delta s)/n - (x_j+\alpha\Delta x_j)(s_j+\alpha\Delta s_j) \geq \\
&\geq (1/\gamma)((1-\alpha)\mu + \alpha\sigma\mu - \alpha^2M^2\mu) - (1-\alpha)\frac{\mu}{\gamma} - \alpha\sigma\mu - \alpha^2M^2\frac{\mu}{\gamma} \geq \\
&\geq \alpha\sigma_{\min}\mu(1/\gamma - 1) - 2\alpha^2M^2\mu/\gamma
\end{aligned}$$

and thus (4.19b) is satisfied if the final expression is non-negative, i.e.

$$\alpha \leq \frac{\sigma_{\min}(1-\gamma)}{2M^2}.$$

Using (4.21d), we obtain

$$\begin{aligned}
&(1-0.01\alpha)\mu - (x+\alpha\Delta x)^T(s+\alpha\Delta s)/n \geq \\
&\geq (1-0.01\alpha)\mu - (1-\alpha)\mu - \alpha\sigma\mu - \alpha^2M^2\mu \geq \\
&\geq 0.99\alpha\mu - \alpha\sigma_{\max}\mu - \alpha^2M^2\mu
\end{aligned}$$

and thus (4.19c) is satisfied if the final expression is non-negative, i.e.

$$\alpha \leq \frac{0.99 - \sigma_{\max}}{M^2}.$$

Using (4.21c) and setting $\eta = 1 - \omega\alpha$, it follows that

$$\begin{aligned}
&(\mathbf{x}+\alpha\boldsymbol{\Delta}\mathbf{x})^T(\mathbf{s}+\alpha\boldsymbol{\Delta}\mathbf{s}) - (1-\omega\alpha)\mathbf{x}^T\mathbf{s} \geq \\
&\geq (1-\alpha)\mathbf{x}^T\mathbf{s} + \alpha\sigma\mathbf{x}^T\mathbf{s} - \alpha^2M^2\mathbf{x}^T\mathbf{s} - (1-\omega\alpha)\mathbf{x}^T\mathbf{s} \geq \\
&\geq \alpha(\sigma+\omega-1)\mathbf{x}^T\mathbf{s} - \alpha^2M^2\mathbf{x}^T\mathbf{s}
\end{aligned}$$

and thus (4.19d) is satisfied if the final expression is non-negative, i.e.

$$\alpha \leq \frac{\sigma+\omega-1}{M^2}.$$

This condition makes sense only if $\omega > 1-\sigma$; therefore, at each IPM iteration, after choosing σ, ω should be chosen from the interval $]1-\sigma, 1]$. Notice what this means: in the early IPM iterations, σ is closer to 1 and thus ω can be closer to 0, which makes the stop criterion easier to satisfy. In the later iterations, σ might get closer to 0 and thus ω is closer to 1, which makes the stop criterion harder to satisfy. If ω is chosen such that $\omega \geq 1-\sigma+\delta$, with δ a fixed constant, $\delta < \sigma_{\min}$, then $\omega+\sigma-1 \geq \delta$ and it follows that

$$\alpha \leq \frac{\delta}{M^2} \quad \Rightarrow \quad \alpha \leq \frac{\sigma+\omega-1}{M^2}.$$

This explains the choice of ω made in the statement of the Lemma.

Therefore, the minimum stepsize that can be taken at each IPM iteration is given by

$$\tilde{\alpha} = \min\Big(\frac{\sigma_{\min}\gamma(1-\gamma)}{M^2(1+\gamma^2)}, \frac{\sigma_{\min}(1-\gamma)}{2M^2}, \frac{0.99-\sigma_{\max}}{M^2}, \frac{\delta}{M^2}, 1\Big).$$

□

Notice that inequalities (4.19a)-(4.19b) imply that the next IPM iteration satisfies condition (4.5b); the inequality (4.19c) represents the Armijo condition, while inequality (4.19d) implies that

$$\frac{\|\mathbf{r}_P^k\|}{\mu^k} \leq \frac{\eta_k\|\mathbf{r}_P^{k-1}\|}{\mu^k} \leq \frac{\|\mathbf{r}_P^{k-1}\|}{\mu^{k-1}} \leq \frac{\beta\|\mathbf{r}_P^0\|}{\mu^0}$$

and similarly for the dual residual, which is equivalent to condition (4.5c). Therefore, the value $\tilde{\alpha}$ represents the minimum stepsize that can be taken at each IPM iteration; a given iteration may be allowed to use a larger stepsize, but it is always possible to make a step of length at least $\tilde{\alpha}$.

To obtain a polynomial complexity result, the value of M should be specified as a function of n. Here, the complexity analysis becomes problematic, since it is difficult to determine exactly the properties of the IPM directions at intermediate Krylov iterations. This is the subject of further research, but for now a rationale is given, based on the properties of the exact directions. To start, recall the results in [131, Chapter 6] about convergence of LPs (similar results for QPs can be found in [130]): a minimum stepsize, proportional to n^{-2} can be found at each iteration, provided that the starting point is chosen appropriately. In the following, this result is generalized to a generic starting point, under some mild assumptions.

Lemma 4.2. *Suppose that the optimal solution* $(\mathbf{x}^*, \mathbf{s}^*)$ *satisfies* $0 \leq \mathbf{x}_i^*, \mathbf{s}_i^* \leq \xi$, *for some large constant* ξ. *Given any starting point* $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0) \in \mathcal{N}_\infty(\gamma, \beta)$ *such that* $0 < \mathbf{x}_i^0, \mathbf{s}_i^0 \leq \xi \ \forall i$ *and* $\mu^0 > \varepsilon^* > 0$, *the minimum stepsize for an exact IPM applied to an LP is* $\bar{\alpha} \geq C_3/n^3$, *for some positive constant* C_3 *independent of* n.

Proof. Recall the following constants from [131, Lemma 6.3 and 6.5], used to find the minimum stepsize for an LP:

$$C_1 = \Big(\beta n + n + \beta\frac{\max^0}{\mu^0}\|(x^*, s^*)\|_1\Big)\frac{1}{\min^0},$$

$$C_2 = 2\frac{C_1}{\gamma^{1/2}}\max\big(\|x^0 - x^*\|, \|s^0 - s^*\|\big) + \frac{n}{\gamma^{1/2}},$$

where $(\mathbf{x}^*, \mathbf{s}^*)$ is the optimal solution, $\min^0$ and $\max^0$ are the minimum and maximum components, respectively, of the vector $(\mathbf{x}^0, \mathbf{s}^0)$. Here, $(\mathbf{x}, \mathbf{s})$ indicates the vector obtained stacking vertically the vectors x and s.

From the definition of the neighbourhood (4.5) and the hypothesis used, it

follows that, for each component i, $\mathbf{x}_i^0\mathbf{s}_i^0 \geq \gamma\mu^0 > \gamma\varepsilon^*$ and thus

$$\mathbf{x}_i^0 > \frac{\gamma\varepsilon^*}{\mathbf{s}_i^0} \geq \frac{\gamma\varepsilon^*}{\xi}.$$

Therefore $\mathbf{x}_i^0, \mathbf{s}_i^0 \in [\gamma\varepsilon^*/\xi, \xi]$, $\forall i$. Hence $\min^0 \geq \gamma\varepsilon^*/\xi$ and $\max^0 \leq \xi$. Notice also that $\|(\mathbf{x}^*, \mathbf{s}^*)\|_1 \leq 2n\xi$. Therefore

$$C_1 \leq \Big(\beta n + n + 2\beta\frac{\xi^2}{\varepsilon^*}n\Big)\frac{\xi}{\gamma\varepsilon^*}.$$

Given that $\mathbf{x}_i^0, \mathbf{s}_i^0, \mathbf{x}_i^*, \mathbf{s}_i^*$ all belong to the interval $[0, \boldsymbol{\xi}]$, for each component i, it follows that

$$\|\mathbf{x}^0 - \mathbf{x}^*\|^2 = \sum_{i=1}^{n}(\mathbf{x}_i^0 - \mathbf{x}_i^*)^2 \leq \sum_{i=1}^{n}\xi^2 = \xi^2 n,$$

and the same holds for $\|\mathbf{s}^0 - \mathbf{s}^*\|$. Therefore

$$C_2 \leq n^{3/2}\Big(\frac{2\xi^2}{\gamma^{3/2}\varepsilon^*}\Big)\Big(\beta + 1 + 2\frac{\beta\xi^2}{\varepsilon^*}\Big) + \frac{n}{\gamma^{1/2}}$$

which implies $C_2 \leq \mathcal{O}(n^{3/2})$. The minimum stepsize $\bar{\alpha}$ that can be taken at each iteration in the exact IPM is proportional to C_2^{-2} as shown in [131, Lemma 6.7], thus $\bar{\alpha} \geq C_3 n^{-3}$. □

Therefore, if the starting point is not the optimal one indicated in [131], the minimum stepsize is proportional to n^{-3}, instead of n^{-2}. Consider the IPM termination criterion (4.4) and property (4.5c); then, the algorithm converges if

$$\mu \leq \min\Big(\tau_\mu, \tau_P\|\mathbf{b}\|\frac{\mu^0}{\beta\|\mathbf{r}_P^0\|}, \tau_D\|\mathbf{c}\|\frac{\mu^0}{\beta\|\mathbf{r}_D^0\|}\Big) =: \varepsilon^*.$$

Therefore, assuming that the iterates of the *inexact* algorithm are bounded by ξ, at each iteration two things can happen: if $\mu \leq \varepsilon^*$, then the algorithm converged; if $\mu > \varepsilon^*$, then the current point can be seen as a starting point of an exact IPM where the minimum stepsize is $\bar{\alpha} \geq C_3 n^{-3}$.

Given that the stepsize must be strictly smaller than the step to the boundary (4.7), it is immediate to see that, when using an *exact* direction, the following holds

$$\frac{-\boldsymbol{\Delta}\mathbf{x}_j}{\mathbf{x}_j} \leq \frac{n^3}{C_3} \quad \forall j \text{ s.t. } \boldsymbol{\Delta}\mathbf{x}_j < 0, \qquad \frac{-\boldsymbol{\Delta}\mathbf{s}_j}{\mathbf{s}_j} \leq \frac{n^3}{C_3} \quad \forall j \text{ s.t. } \boldsymbol{\Delta}\mathbf{s}_j < 0.$$

Additionally, using (4.13), one can see that

$$\Big|\frac{\boldsymbol{\Delta}\mathbf{x}_j}{\mathbf{x}_j}\Big| \leq \mathcal{O}(n^3), \qquad \Big|\frac{\boldsymbol{\Delta}\mathbf{s}_j}{\mathbf{s}_j}\Big| \leq \mathcal{O}(n^3), \qquad \forall j,$$

since the terms $\frac{\boldsymbol{\Delta}\mathbf{x}_j}{\mathbf{x}_j}$ and $\frac{\boldsymbol{\Delta}\mathbf{s}_j}{\mathbf{s}_j}$ must balance in order to give a sum that is $\mathcal{O}(1)$.

Therefore, when using an *exact* IPM for LPs, with a generic starting point,

the computed direction satisfies criterion (4.14) with $M = \mathcal{O}(n^3)$. This provides a rationale to expect that *inexact* steps applied should satisfy conditions like (4.14) with a constant M of comparable magnitude. Therefore, we infer that it is possible to use a constant $M = \mathcal{O}(n^q)$, with $q \geq 3$. This is of course only a rationale: a proper proof would require to understand whether the chosen Krylov method is able to deliver such a direction; potentially, an exponent q specific to the linear solver used may be found, but this has been found to be complicated and is the subject of further research. Notice also that the rationale argument is given for an LP, but the criterion is used for QPs (similar arguments can be found for QPs, based on the results in [130]).

Given $M = \mathcal{O}(n^q)$, [131, Theorem 3.2] and Lemma 4.1 imply that the number of iterations to achieve a $\nu-$accurate solution would be $\mathcal{O}(n^{2q}|\log \nu|)$. In the best case where $q = 3$, this would mean a number of iterations proportional to n^6; this is higher than the $\mathcal{O}(n^2)$ iterations required by the exact algorithm, as it is to be expected from the very inexact stopping criterion considered.

Remark 4.4. *The analysis presented in this section has used the results from [131, Chapter 6]; it is worth pointing out that the results presented there are obtained using a neighbourhood without the upper bound in* (4.5b)*. However, with some simple calculations, it is possible to see that the final results do not change after adding the upper bound. A similar conclusion was obtained in [33], where the upper bound was added in the case of a feasible algorithm.*

4.4.2 Indicators for early stopping

In this section, new indicators are derived that can be used to terminate the inner linear iterations early, before the relative residual has become small enough to be accepted by a standard residual test. The behaviour is shown for one test problem, but the same pattern can be observed also for the other test problems. This specific problem is the QP arising from tomographic imaging described in Chapter 3.

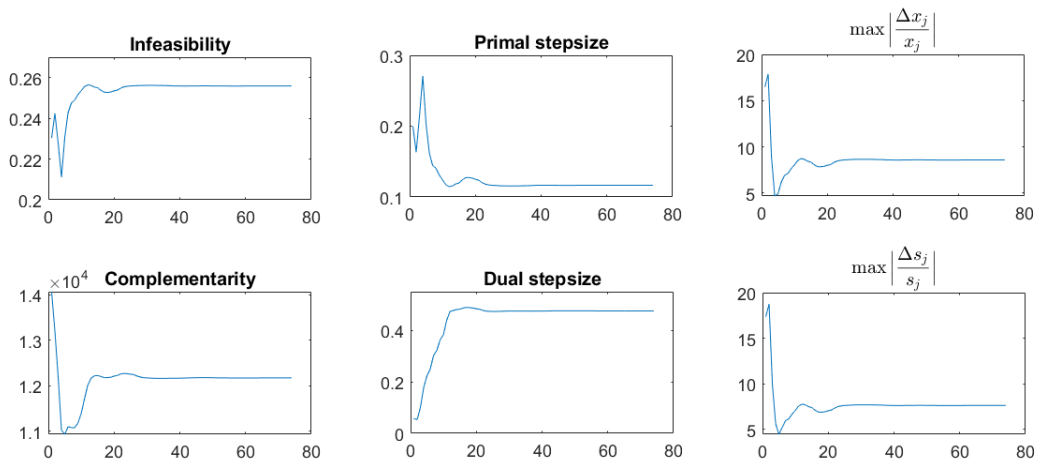
The indicators that are introduced are based on the complexity argument given in the previous section; they are related to the following quantities:

- $M_x = \max_i |\frac{\mathbf{\Delta x}_i}{\mathbf{x}_i}|$ and $M_s = \max_i |\frac{\mathbf{\Delta s}_i}{\mathbf{s}_i}|$
- infeasibilities: $\mathbf{p}_{\text{inf}} = \mathbf{b} - A\mathbf{x}$ and $\mathbf{d}_{\text{inf}} = \mathbf{c} - A^T\mathbf{y} - \mathbf{s}$
- complementarity measure: $\mu = (\mathbf{x}^T\mathbf{s})/n$.

These are computed at each inner Krylov iteration using the IPCG or IPMINRES algorithms; an index j is used to indicate the value obtained at the inner iteration j and the IPM iteration index is omitted instead. Therefore, M_x^j means the value of the quantity M_x that would be obtained by stopping at the $j-$th inner iteration, for a given IPM iteration.

Figure 4.1 displays the behaviour of the dual infeasibility, complementarity, primal and dual stepsizes and the quantities M_x and M_s at an intermediate IPM iteration; they are computed at every inner CG iteration, using Algorithm IPCG.

Figure 4.1: Infeasibility, complementarity, stepsizes and quantities M_x and M_s computed at every CG iteration, for an intermediate IPM iteration.



It can be seen that all the quantities represented reach a point where their variation becomes extremely small, almost impossible to notice from the picture; this "stagnation" point may arrive very early in the CG iterations, meaning that a large portion of the inner iterations are used to adjust the IPM direction in a way that has a small effect on the quality of the new IPM point.

This fact suggests the following early termination indicators:

$$\texttt{var}_P^j = \frac{1}{5}\sum_{i=0}^{4}\left|\frac{\|\mathbf{p}_{\text{inf}}^{j-i}\| - \|\mathbf{p}_{\text{inf}}^{j-i-1}\|}{\|\mathbf{p}_{\text{inf}}^{j-i-1}\|}\right|, \quad \texttt{var}_D^j = \frac{1}{5}\sum_{i=0}^{4}\left|\frac{\|\mathbf{d}_{\text{inf}}^{j-i}\| - \|\mathbf{d}_{\text{inf}}^{j-i-1}\|}{\|\mathbf{d}_{\text{inf}}^{j-i-1}\|}\right|$$

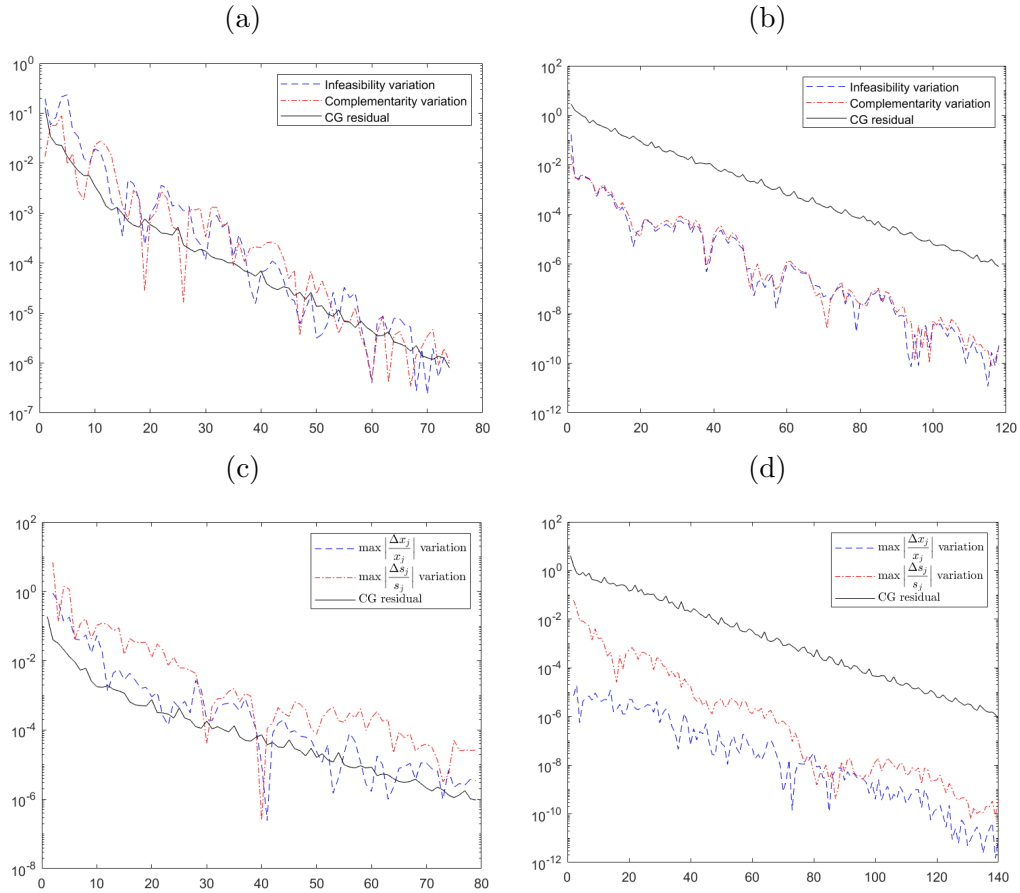
$$\texttt{var}_{Mx}^j = \frac{1}{5}\sum_{i=0}^{4}\left|\frac{|M_x^{j-i}| - |M_x^{j-i-1}|}{|M_x^{j-i-1}|}\right|, \quad \texttt{var}_{Ms}^j = \frac{1}{5}\sum_{i=0}^{4}\left|\frac{|M_s^{j-i}| - |M_s^{j-i-1}|}{|M_s^{j-i-1}|}\right|$$

These are the average relative variations, in the last five inner iterations, of the quantities $\mathbf{p}_{\text{inf}}$, $\mathbf{d}_{\text{inf}}$, M_x and M_s. From the previous Figure, one expects these quantities to decrease during the CG iterations and, since they are related to the IPM convergence, to be better indicators than the simple relative residual of the linear system. For some problems it can be useful to consider also the indicator $\texttt{var}_\mu$ defined in the same way as before but considering the complementarity measure μ.

Figure 4.2 shows the proposed indicators compared to the relative residual, at every CG iteration, during the computation of various IPM directions. Notice that these are only some of the behaviours that were observed; the purpose of these images is to show that the indicators can sometimes decrease similarly to the residual, while on occasions they may display an erratic behaviour, which is difficult to capture looking only at the residual.

To summarize, the following termination criterion is proposed: the inner

Figure 4.2: Various behaviours of the proposed indicators compared to the relative residual



iterations are stopped if

$$\Big((\mathtt{var}_P^j < \varepsilon) \wedge (\mathtt{var}_D^j < \varepsilon) \wedge (\mathtt{var}_{Mx}^j < \varepsilon) \wedge (\mathtt{var}_{Ms}^j < \varepsilon)\Big) \vee \big(\|\mathbf{r}\|/\|\mathbf{r}_0\| < \tau_{\text{inner}}\big).$$

The first four conditions check if the new indicators are all smaller than a tolerance ε; however, if it happens that the residual gets sufficiently small before the new indicators do, then the stopping criterion is triggered anyway, as with a standard residual test.

The practical criterion presented here is clearly a different technique than the theoretical one shown in Algorithm IPM-I; however, it is strongly inspired by the arguments of the previous Section. The indicators considered involve the same quantities used in the criterion (4.14)-(4.15) and the condition of small relative variations ensures that the inexact direction found is likely to produce a point that gets close to satisfying the theoretical criterion as well. Some safeguards are required in order to keep the behaviour of the practical criterion close to the ideal one: in particular, the practical criterion may be triggered by chance in the very first linear iterations, when the theoretical stopping criterion is not yet satisfied. For this reason, the parameter `itstart` is important since it prevents this phenomenon from happening.

Such a difference between theoretical and practical methods is not unusual in the IPM literature, where often the theoretical properties are proven for the methods, but to achieve the best performance the practical algorithms slightly deviate from the rigorous theoretical settings.

Notice also that the quantities M_x and M_s used in the practical stopping criterion are the same quantities that are bounded by the constant M in the theoretical setting. However, given the large magnitude of M, which is chosen such that $M = \mathcal{O}(n^q)$, with $q \geq 3$, the criterion (4.14) is satisfied almost immediately during the Krylov iterations. This means that the extremely inexact directions computed using the theoretical criterion require very few Krylov iterations (if any) and make only a very small progress in terms of convergence of the IPM. This is reflected in the increased iteration complexity, from n^2 to n^6. In practice, however, this small progress made at each IPM iteration would likely be cancelled by numerical inaccuracies, yielding an algorithm that is not able to converge to the optimal solution. Therefore, to obtain a viable practical stopping criterion, we need to rely on a more complicated strategy like the one proposed above, that guarantees a substantial progress at each IPM iteration, while requiring a small number of Krylov iterations.

4.5 Numerical results

In this section, the test problems are introduced and the results obtained with the standard CG or MINRES and with the novel IPCG or IPMINRES are presented. This section shows overall results in terms of IPM iterations, inner iterations and computational time, and then provides also an insight into the individual IPM iterations to demonstrate where the gains resulting from the new method are the most significant.

The numerical experiments were performed using MATLAB R2018a and were run on the University of Edinburgh School of Mathematics computing server, which is equipped with four 3.3GHz octa-core Intel Gold 6234 processors and 500GB of RAM; the experiments never used more than 4 cores and 20GB of memory.

The new technique is compared with two options that are usually employed when dealing with a Krylov method inside an IPM: a fixed tolerance on the relative residual of the linear system and a variable tolerance proportional to the complementarity measure μ (see e.g. [25, 65, 90]). In particular, the tolerance for the second option is chosen at each IPM iteration k as

$$\tau_{\text{inner}}^k = \max\left(\texttt{tol}^{\min}, \frac{\mu^k}{\mu^0}\,\texttt{tol}^0\right)$$

where $\texttt{tol}^0$ is the initial tolerance, $\texttt{tol}^{\min}$ is the minimum tolerance considered, μ^k is the value of the complementarity measure at the current iteration and μ^0 is the initial one. In this way, the tolerance decreases at the same rate as μ until it reaches the value $\texttt{tol}^{\min}$. In the following, these two options are denoted as `fixtol` and `vartol` respectively.

Despite having multiple options available to choose a variable tolerance, we compared the results with this one, since it is very simple and widespread. There may be other tolerance sequences, tailored specifically to the problem considered, that produce better results; however, the new criterion that is introduced does not need to be redesigned for a specific problem and hence a variable tolerance was selected in the same simple way for all problems. It is worth pointing out that other specialized stopping criteria, developed for different problems (e.g. [7, 11, 55, 58, 96, 116, 118]) cannot be easily generalized and used inside an IPM, since the quantities used for these criteria may not even have a meaningful interpretation in this context.

The values of $\texttt{tol}^0$ and $\texttt{tol}^{\min}$ were chosen after a quick tuning process in order to obtain the best results with the variable tolerance method; the same holds for the parameters ε and `itstart` of the new stopping criterion. The specific values are given below for each problem class.

4.5.1 Tomographic reconstruction

The first test problem has been thoroughly described in Chapter 3. Let us recall that this problem is a quadratic program without linear equality constraints and only with non-negativity constraints. An efficient preconditioner can be found for the normal equations matrix and the Newton direction can thus be found using the conjugate gradient method. The application of the matrix of the system is particularly expensive, since it involves the call of the Radon and inverse Radon transforms; thus, a single CG iteration is relatively expensive and we expect the IPCG to bring a substantial benefit. Notice that, since there are no linear equality constraints, only the dual infeasibility can be computed.

An IPM with centrality correctors was applied to this problem: the IPM tolerance was set to 10^{-8}, the CG tolerance for the `fixtol` approach was 10^{-6} and the parameters for the `vartol` approach were $\texttt{tol}^{\min} = 10^{-6}$, $\texttt{tol}^0 = 10^{-3}$. These parameters were selected because they allow fewer linear iterations, without compromising too much the quality of the inexact direction and the IPM convergence speed. The new IPCG approach was applied with parameters $\varepsilon = 0.01$ and 0.001, $\texttt{itstart} = 5$ and $\tau_{\text{inner}} = 10^{-6}$. Since the problems contain noise that is randomly initialized at every run, the results shown are the average over 10 runs, for each discretization level. We observed than the mean results over 10 runs are a reliable average: for example, for $\texttt{level} = 32$, the standard deviation of computational times over 10 runs was in the range $0.15 - 0.35$, for all the cases shown below (`fixtol`, `vartol` and IPCG).

Table 4.1 reports the results using the `fixtol` and `vartol` approaches. The parameter `level` indicates how fine the discretization of the problem is; the size of the matrix is equal to $2 \cdot \texttt{level}^2$, so that the largest instance has $524,288$ variables.

Tables 4.2 and 4.3 instead show the results obtained with IPCG with $\varepsilon = 0.01$ and $\varepsilon = 0.001$ respectively; the last columns show the reduction in linear iterations and computational time when compared with the previous approaches: this is computed as the difference between the result with standard termination criterion and with IPCG, divided by the result with standard criterion. For instance, the reduction of number of iterations of IPCG compared to `fixtol` is given by

Table 4.1: Results with CG: `fixtol` and `vartol`

	CG `fixtol`			CG `vartol`		
`level`	IPM It	Inner It	Time	IPM It	Inner It	Time
32	16.7	3,892.1	9.02	17.1	2,001.5	4.71
64	21.0	6,436.9	27.37	21.4	3,350.0	14.64
128	22.5	9,514.8	104.96	26.0	4,952.6	55.48
256	24.8	14,183.5	511.87	33.0	8,059.3	295.45
512	29.5	21,897.7	3,035.63	44.0	14,413.0	1,954.11

Table 4.2: Results with IPCG ($\varepsilon = 0.01$)

	IPCG			Reduction `fixtol`		Reduction `vartol`	
`level`	IPM It	Inner It	Time	Inner It %	Time %	Inner It %	Time %
32	16.7	842.2	2.11	78.3	76.6	57.9	55.2
64	20.5	1,329.1	6.67	79.4	75.6	60.3	54.4
128	22.7	1,688.1	22.73	82.2	78.3	65.9	59.0
256	26.8	2,090.2	93.46	85.3	81.7	74.1	68.4
512	34.0	2,849.1	509.11	87.0	83.2	80.2	73.9

$(It_{\texttt{fixtol}} - It_{IPCG})/It_{\texttt{fixtol}}$.

It is worth observing that when using IPM with the new stopping criterion, the number of outer (IPM) iterations is very close to the one obtained with the original IPM using `fixtol`; this confirms that the inexact direction is sufficiently precise so as not to destroy the convergence properties of IPM. In particular, it can be noticed that using a lower tolerance ε guarantees an IPM iteration count almost identical to the original one; the `vartol` approach instead produces a substantial increase in the IPM iterations, particularly for larger problems.

Observe also that the IPCG with $\varepsilon = 0.01$ produces a similar number of IPM

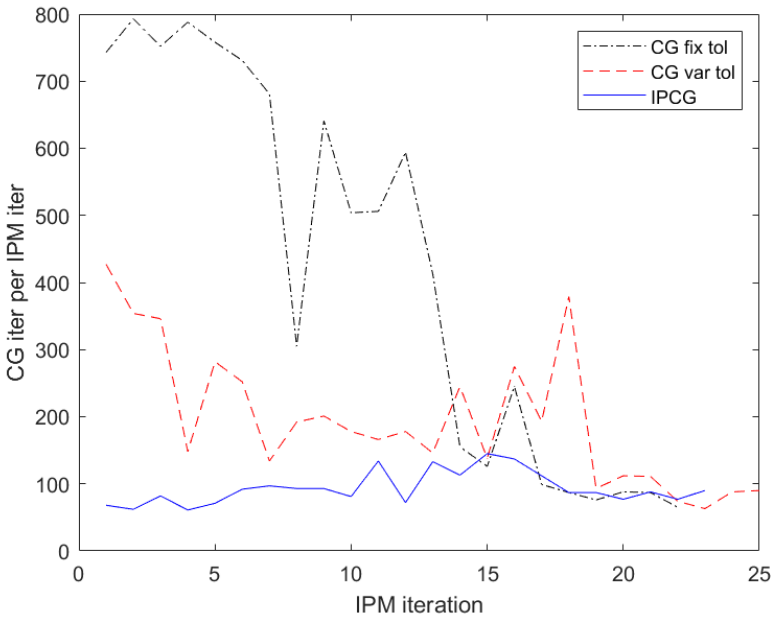
Table 4.3: Results with IPCG ($\varepsilon = 0.001$)

	IPCG			Reduction `fixtol`		Reduction `vartol`	
`level`	IPM It	Inner It	Time	Inner It %	Time %	Inner It %	Time %
32	16.5	1,219.6	3.07	68.7	66.0	39.0	34.8
64	20.5	1,870.8	8.77	70.9	68.0	44.1	40.1
128	22.0	2,538.2	33.82	73.3	67.8	48.8	39.0
256	25.5	3,475.7	152.67	75.5	70.2	56.9	48.3
512	30.3	4,950.0	859.22	77.4	71.7	65.7	56.0

iterations as the `vartol` approach, but uses far fewer inner iterations. This means that the new technique is better at choosing when to stop the linear iterations and does not compromise the overall IPM convergence more than a standard inexact IPM would.

The reduction in terms of linear iterations is very high and reaches values of more than 70% for both choices of ε for the largest instance considered. This translates into a significant computational time reduction, which confirms that the operations added inside the IPCG algorithm are inexpensive. Indeed, the time per CG iteration when `level` $= 512$ goes roughly from 140ms in the case of standard CG to 175ms in the case of IPCG; a small increase which is offset by a large reduction in the number of inner iterations.

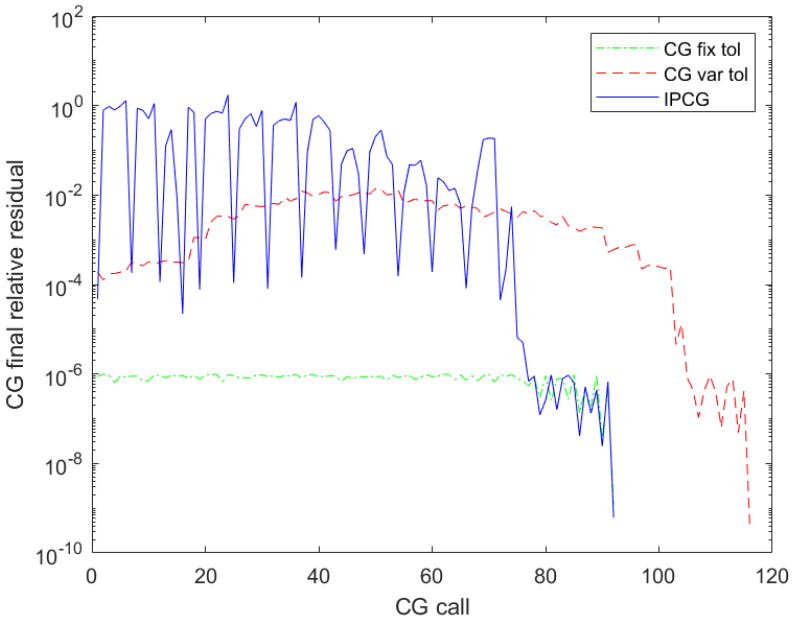
Figure 4.3: Number of inner CG iterations performed at each outer IPM iteration, for `level` $= 128$. Similar behaviours can be observed for other values of `level`.



Next, it is useful to understand how the gain of IPCG is distributed during the IPM iterations. To do this, we recorded the number of CG iterations at each IPM iteration (summing together the inner iterations for predictor and correctors) in three different situations: using CG with `fixtol`; using CG with `vartol`; using IPCG with $\varepsilon = 10^{-3}$. Figure 4.3 shows the comparison of the iterations for the problem with `level` $= 128$.

Notice that, when using standard CG with fixed tolerance, the number of iterations decreases at the end, since less correctors are computed; when using IPCG, this decrease is not observed, since the smaller number of correctors is balanced by the increased accuracy needed. Indeed, in the late IPM phase, the new stop criterion is not triggered and IPCG stops with the standard reduction test; the reader may observe that the graphs overlap in the last iterations. However, in the initial phase, a significant advantage of IPCG over the standard CG can be noticed, both for the `fixtol` and `vartol` approaches. It is striking how the IPCG

Figure 4.4: Final CG relative residual for the three approaches considered



requires almost every time fewer inner iterations than the CG with `vartol`, but still manages to converge in a smaller number of IPM iterations. This is because the number of inner iterations used for the predictor and for the correctors is distributed differently: the standard CG applies the same tolerance to all the directions during a certain IPM iteration, while IPCG chooses when to stop the inner iterations based on the improvement that the direction can bring to the IPM convergence. In this way, some correctors are computed very roughly, without spoiling the overall IPM convergence speed.

This is clear from the next analysis that was performed: the final relative residual at each CG call (for predictors and correctors) was recorded for all the three approaches; the results are shown in Figure 4.4. Notice that the IPCG computes directions both more accurately and less accurately than the `vartol` approach, depending on how much a certain direction is able to improve the quality of the IPM point. The extreme oscillations observed for IPCG can be understood considering that some of the CG calls are made for predictor directions and some for correctors; while predictors need to be more accurate, since they are the first direction computed in a given iteration, correctors (especially subsequent ones) have limited effect on the overall direction, because they are added to the predictor. The IPCG method can detect the effect of the direction being computed on the overall IPM algorithm and can stop the computation of correctors much sooner than for predictors. The surprising variability of the final residual suggests that there is much to be gained by an approach that does not involve only the residual tolerance, because otherwise it would not be possible to capture this behaviour. This graph highlights also that no stopping criterion based only on a residual tolerance (potentially different from the `vartol` approach considered here) could match the performance of the proposed solver, given the variability

observed.

The graphs displayed in these two figures undeniably confirm that a high accuracy in the first IPM iterations is not needed at all, and that the best method to decide when a direction is sufficiently precise to perform the next IPM iteration successfully should be based on the IPM indicators and not on the residual of the linear system.

4.5.2 Compressed sensing

The second test problem arises from compressed sensing [52]: a sparse solution to an undetermined linear system $A\mathbf{x} = \mathbf{b}$ is sought, where sparsity is enforced by means of a 1-norm regularization. After linearizing the 1-norm by adding extra variables, the optimization problem that arises is the following

$$\min_{\mathbf{z}\geq 0} \tau \mathbf{e}^T\mathbf{z} + \frac{1}{2}\|F^T\mathbf{z} - \mathbf{b}\|^2,$$

where $\tau > 0$, $\mathbf{z} = \begin{bmatrix}\mathbf{u} \; ; & \mathbf{v}\end{bmatrix}$, $\mathbf{u}$ and $\mathbf{v}$ being the positive and negative parts of vector $\mathbf{x}$, and $F^T = \begin{bmatrix}A & -A\end{bmatrix}$. Rewriting it as a standard quadratic program and formulating the IPM normal equations, the matrix of the linear system to be solved becomes

$$H = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes A^TA + \Theta^{-1}.$$

Due to the structure of matrix A, which satisfies the *restricted isometry property* (see [52] for more details), matrix H can be efficiently preconditioned by the block diagonal matrix

$$P = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes \eta I + \Theta^{-1}$$

for an appropriate constant η. The difference with respect to the previous test problem is that now the IPM direction is computed using a very low accuracy for the CG: the residual tolerance is 10^{-1} or 10^{-2}, depending on the problem, throughout all the IPM iterations. This low accuracy of CG is motivated by the properties of problem being solved; in particular, for problems with a higher level of noise, a lower accuracy is used, while for noiseless problems a better accuracy is needed. Due to this very rough tolerance, the `vartol` approach was not able to bring any substantial improvement. For this class of problems only the `fixtol` approach was used.

The test problems are taken from the Sparco collection [121]; of the 18 problems considered in [52], 5 did not show any improvement when using IPCG instead of CG (in part because they were easy enough and the CG was already performing a low number of iterations). In Table 4.4 the results for the remaining 13 that did show an improvement are presented. The IPM tolerance varies between 10^{-6} and 10^{-10} according to the problem being solved and no corrector direction is used. The default values for IPCG are $\varepsilon = 0.01$ and `itstart` $= 5$; for some problems we

Table 4.4: Results for Compressed sensing IPM.

		CG `fixtol`			IPCG			Inner It	Time
ID	Size	IPM	Inner It	Time	IPM	Inner It	Time	red %	red %
6	4,096	22	2,128	40.22	23	193	4.11	90.9	89.8
9	256	11	382	0.23	11	147	0.13	61.5	43.5
10	2,048	12	2,210	0.57	16	874	0.34	60.5	40.4
11 †*	2,048	19	663	1.41	21	536	1.14	19.2	19.1
401	114,688	14	160	15.72	12	55	7.03	65.6	55.3
402 †	172,032	14	238	28.14	12	59	9.76	75.2	65.3
403	393,216	19	2,282	201.44	20	205	36.94	91.0	86.6
601 †	8,192	20	2,146	104.20	21	652	28.41	69.6	72.7
602	8,192	22	2,280	124.39	20	453	19.12	80.1	84.6
603	8,192	16	1,085	16.53	13	86	2.13	92.1	87.1
701 †	131,072	12	1,028	38.64	12	236	13.73	77.0	64.5
702	32,768	8	926	15.00	8	181	4.42	80.5	70.5
903 †	2,048	13	1,794	2.52	16	687	0.93	61.7	63.1

†: $\varepsilon = 0.001$ instead of 0.01, *: `itstart` $= 20$ instead of 5

used different parameters (indicated at the bottom of the table) to obtain the best possible results. In particular, for problem 11, setting `itstart` to 20 is required to guarantee convergence in a reasonable number of IPM iterations.

All these problems display an impressive reduction in the number of CG iterations and CPU time, even if the original CG tolerance is very rough. The added cost of IPCG varies throughout the problems, but on average is roughly $35 - 40\%$ of the original iteration cost. Sometimes a reduction in IPM iterations is also observed; this may be because the inexact method proposed is finding by chance a direction that is better than the exact one.

4.5.3 PDE constrained optimization

As a last test example, we consider PDE constrained optimization problems (see e.g. [101]) and use the augmented system approach, in order to test Algorithm IPMINRES. In this section, $\hat{v}$ and $\mathbf{v}$ will denote respectively the continuous and discretized version of a variable v. The kind of problems considered involve PDE as constraints and they take the standard form

$$\begin{aligned} \min_{y,u} \quad & \frac{1}{2}\|\hat{y} - \hat{y}_0\|^2_{L^2} + \frac{\beta}{2}\|\hat{u}\|^2_{L^2} \\ \text{s.t.} \quad & -\nabla^2 \hat{y} = \hat{u} + \hat{f}, \quad \hat{y} \in \Omega \\ & \hat{y} = \hat{g}, \quad \hat{y} \in \partial\Omega \\ & \hat{u}_a \leq \hat{u} \leq \hat{u}_b \end{aligned}$$

where Ω is the domain of evolution of the problem, $\hat{y}, \hat{u}$ are the state and control variables, $\hat{y}_0$ is the desired state function, $\hat{f}, \hat{g}, \hat{u}_a, \hat{u}_b$ are given functions and $\beta > 0$ is the regularization parameter. The objective of this formulation is to keep the state $\hat{y}$ close to the fixed desired state $\hat{y}_0$ and minimize the control $\hat{u}$, while satisfying the PDE and bound constraints. Problems of this kind arise, for example, in optimal control theory: practical applications include optimal design of semiconductors, shape optimization, optimal gas cooling and many others (the interested reader can find more details in [74]).

A standard IPM is applied to this problem, using the *discretize-then-optimize* approach, as described in [101], to obtain the discretized quantities $\mathbf{y}, \mathbf{u}, \mathbf{y}_0, \mathbf{u}_a, \mathbf{u}_b$; we then introduce the variables $\mathbf{z}_a$ and $\mathbf{z}_b$ defined as $(\mathbf{z}_a)_j = \mu/(\mathbf{u} - \mathbf{u}_a)_j$ and $(\mathbf{z}_b)_j = \mu/(\mathbf{u}_b - \mathbf{u})_j$. After using a standard Q1 finite elements discretization, the augmented system has the form

$$\begin{bmatrix} M & 0 & K \\ 0 & \beta M + \Theta & -J \\ K & -J & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}\mathbf{y} \\ \boldsymbol{\Delta}\mathbf{u} \\ \boldsymbol{\Delta}\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_u \\ \mathbf{r}_\lambda \end{bmatrix}, \tag{4.22}$$

where $M \in \mathbb{R}^{n \times n}$ is the finite elements mass matrix, $J \in \mathbb{R}^{n \times n}$ is the same matrix but with boundary conditions applied, $K \in \mathbb{R}^{n \times n}$ is the stiffness matrix, $\boldsymbol{\lambda} \in \mathbb{R}^n$ is the vector of Lagrange multipliers; due to the presence of upper and lower bounds on $\mathbf{u}$, matrix Θ has the following form $\Theta = Z_a(U - U_a)^{-1} + Z_b(U_b - U)^{-1}$. The dimension of the matrices n is determined by the grid level n_c, as $n = (2^{n_c} + 1)^2$ (i.e. each grid has roughly twice the number of subdivisions in each dimension with respect to the previous grid); the whole augmented system has size $3n$.

This linear system can be solved using MINRES, provided that the preconditioner is positive definite; exploiting the ideas in [101] and [102], the following preconditioner is employed

$$P = \begin{bmatrix} \tilde{M} & 0 & 0 \\ 0 & \beta\tilde{M} + \Theta & 0 \\ 0 & 0 & \tilde{S} \end{bmatrix},$$

where $\tilde{M}$ contains only the diagonal of M and $\tilde{S}$ is an approximation of the Schur complement of (4.22) given by

$$\tilde{S} = \left(K + \frac{1}{\sqrt{\beta}}J\right) M^{-1} \left(K + \frac{1}{\sqrt{\beta}}J\right).$$

The Schur complement preconditioner is constant throughout the IPM iterations and to apply it, it suffices to compute the Cholesky factorization of $(K + J/\sqrt{\beta})$ once at the beginning of the algorithm. The finite element matrices were computed using the IFISS package [2, 48, 49].

An IPM with centrality correctors was applied to this problem. The parameters used are: IPM tolerance 10^{-8}; for the `fixtol` approach, MINRES tolerance 10^{-8};

Table 4.5: Results with `fixtol` and `vartol` approaches

		MINRES `fixtol`			MINRES `vartol`		
β	n_c	IPM	Inner It	Time	IPM	Inner It	Time
10^{-4}	5	10	747	0.32	10	455	0.19
	6	11	812	2.08	11	515	1.34
	7	13	919	29.76	13	621	19.93
	8	14	930	327.33	14	655	236.58
	9	14	839	5,094.79	14	672	3,722.21
10^{-5}	5	11	1,424	0.48	11	782	0.27
	6	13	1,711	4.27	13	978	2.48
	7	14	1,861	60.08	14	1,036	32.51
	8	16	2,037	706.28	16	1,231	437.32
	9	16	1,950	11,783.63	16	1,163	6,996.73
10^{-6}	5	14	3,511	1.09	13	1,798	0.57
	6	15	3,902	9.66	15	2,217	5.55
	7	16	4,216	125.13	16	2,316	72.18
	8	17	4,450	1,530.93	17	2,346	814.10
	9	20	4,959	29,979.89	19	2,679	16,260.61

for the `vartol` approach, $\texttt{tol}^{\min} = 10^{-8}$, $\texttt{tol}^0 = 10^{-2}$; for the IPCG, $\varepsilon = 10^{-3}$ and $\texttt{itstart} = 15$. Values of n_c from 5 to 9 were considered, so that the largest problem had dimension $789,507$; for β, the values used were 10^{-4}, 10^{-5} and 10^{-6}.

Table 4.5 shows the results using the `fixtol` and `vartol` approaches. Table 4.6 shows the results using the IPMINRES method and the last columns report the reductions in the number of inner iterations and computational time compared with the previous approaches. A negative reduction means that the new approach produces a larger number of iterations or larger computational time: this happens in some small problems either because the inexact direction produces a large number of IPM iterations, or because the reduction in inner iterations is not enough to balance the more expensive CG iteration. This phenomenon however disappears for larger problems.

The reader can observe that when considering larger problems and smaller values of β, there is a significant reduction in inner iterations and computational time, while the IPM iteration count is almost constant in all three approaches. The improvement that the new method brings is more significant when the linear system becomes more ill conditioned (larger size and smaller β); this is not surprising, since it is known that the residual of the linear system can be a misleading indicator for ill conditioned problems. The proposed new approach does not suffer from this issue, as these results suggest, because it is related to the IPM properties rather than to the algebraic properties of the matrix, thus making it a more suitable termination criterion for ill conditioned matrices.

Table 4.6: Results using IPMINRES

		IPMINRES			Reduction `fixtol`:		Reduction `vartol`:	
β	n_c	IPM	Inner It	Time	Inner It %	Time %	Inner It %	Time %
10^{-4}	5	13	535	0.32	28.4	0.0	-17.6	-68.4
	6	16	653	1.85	19.6	11.2	-26.8	-38.9
	7	15	594	19.07	35.4	35.9	4.3	4.3
	8	16	640	228.36	31.2	30.2	2.3	3.5
	9	16	630	3,534.22	24.9	20.9	6.3	5.1
10^{-5}	5	12	734	0.31	48.5	35.4	6.1	-14.8
	6	14	821	2.30	52.0	46.1	16.1	7.3
	7	16	862	27.36	53.7	54.5	16.8	15.8
	8	16	852	299.81	58.2	57.6	30.8	31.4
	9	18	834	5,072.63	50.8	49.0	28.3	27.5
10^{-6}	5	24	2,041	0.62	41.9	24.8	-13.5	-8.8
	6	22	1,861	5.19	52.3	46.3	16.1	6.5
	7	18	1,533	48.04	63.6	61.6	33.8	33.4
	8	19	1,583	550.35	64.4	64.1	32.5	32.4
	9	18	1,318	8,058.47	73.4	73.1	50.8	50.4

These last results show that also the IPMINRES method works as expected and can potentially bring a significant improvement. Moreover, they also show that the new early stopping technique can be applied to different classes of problems, with similar results.

4.6 Conclusion

This Chapter has shown that it is possible to stop the inner Krylov iterations during an interior point method earlier than it was previously thought, provided that the stopping criterion used is based on the IPM convergence indicators and not only on the reduction of the residual of the linear system. We have given a rationale to explain the expected effect of the termination criterion and have proposed two practical algorithms for the normal equations and augmented system approaches. They exploit new indicators, related to the convergence of the outer iterations, and are only marginally more computationally expensive then the original algorithms. A proof of polynomial complexity of such inexact IPM is still elusive and is the subject of further research, as well as a characterization of the constant M involved in the criterion, potentially depending on the linear solver chosen. This could provide a theoretical result on the minimum threshold of accuracy needed for the convergence of IPMs.

This Chapter provided computational evidence for a wide range of problems,

from image processing, compressed sensing and PDE-constrained applications; they all display a significant reduction in the number of inner Krylov iterations and computational time. In particular, the largest gain appears in the early IPM phase, where it is already known that a lower accuracy of Newton directions is sufficient; however, we have also shown that it is extremely difficult to mimic the behaviour of the proposed stopping criterion using only a residual test, since the residual of the optimal stopping point may vary drastically during the IPM iterations. Indeed, the new technique outperforms also the termination criterion that uses a variable residual tolerance. Moreover, the new IPCG seems to keep the IPM iteration count closer to the original one than with a variable tolerance. This fact strongly supports the initial claim that a good stopping criterion for CG or MINRES should be based on the IPM convergence indicators.

The analysis of the numerical results suggests that for ill conditioned problems the performance gain of the new stopping criterion is larger. However, there are some problems that are so badly conditioned and/or require so much precision in the IPM direction that the new stopping criterion is not able to perform well; more research is needed to find a suitable more advanced termination strategy for these challenging problems.

We strongly believe that many other practical optimization algorithms in which a Krylov subspace method is used to solve the linear equation systems are likely to benefit from a specialized stopping criterion developed with an understanding of the specific needs of the method.

Chapter 5

Interior point method for discrete optimal transport

In this Chapter, based on the work [133], we introduce a specialized hybrid of interior point method and column generation for linear programs arising from optimal transport applications. The proposed method is shown to solve efficiently problems with up to 4 billion variables, displaying a linear growth of the computational time.

5.1 Introduction

As shown in the previous Sections, IPMs possess many interesting features: they are well suited for the solution of problems with some underlying structure that allows for a simplified formulation and efficient preconditioning (e.g. [27, 28, 52, 68]); they have been used in combination with column-generation approaches (see e.g. [66, 67]) to solve problems with many more variables than constraints; they can be formulated in a matrix-free way for solving very large problems (e.g. [52, 64]) and they have been used to solve a variety of sparse approximation problems (e.g. [40]).

Linear programs arising from discrete Optimal Transport (OT) possess many of these attractive characteristics (see e.g. [103]): they can have an extremely large number of variables, but relatively few constraints and as a consequence of this, their optimal solution is extremely sparse; they have a very particular constraint matrix with a Kronecker structure which leads to a simplified formulation of the normal equations; they need to be dealt with in a matrix-free way, to avoid forming the huge, very sparse and highly structured constraint matrix.

Linear programs with many more variables than constraints are well suited to be solved using a column generation approach, as shown in Section 1.4.4. In this Chapter, we propose a hybrid method for general purpose discrete optimal transport problems: the proposed algorithm can be interpreted both as a very aggressive column generation approach (with just one Newton step applied to each restricted master problem) and as a very relaxed IPM (which works only with a subset of variables and ignores the other ones). The method is highly specialized and exploits the structure of the underlying problem and the known

properties of the optimal solution to efficiently mix iterative and direct solvers for the Newton linear system. It is implemented without ever forming the constraint matrix, but only accessing it via matrix-vector products, exploiting its Kronecker structure; the only matrix that is formed is the much smaller Schur complement of the normal equations. The method uses a specialized sparse linear algebra in order to tackle problems of very large dimension.

In particular, the normal equations within the IPM are further reduced to the Schur complement and then solved either with the conjugate gradient method or with a general sparsity-exploiting Cholesky factorization. This is possible because the fill-in of the Cholesky factor of the Schur complement gets smaller and smaller when the method approaches optimality; this is rigorously proven using the graph interpretation of the OT problem and confirmed by extensive computational evidence. Moreover, the proposed theoretical result allows some non-trivial chordal sparsity patterns to be characterized in a new way.

Many methods have been designed for solving OT problems, see e.g. [10, 36, 51, 72, 83, 88, 113] and the comprehensive summary [115]. A similar idea to the one proposed here, where a sparse version of IPM is used to solve optimal transport problems, was recently proposed in [129], for a particular subset of OT problems, and in [94] an IPM was applied to solve optimal transport problems involving finite volumes discretization.

The strengths of the algorithm proposed in this Chapter, compared to the previously mentioned approaches, are:

- Very general formulation, able to deal with many types of problems; indeed, the proposed method is tested on a large and varied collection of problems.
- Adaptability to multiple cost functions, while other methods are often specialized only to one particular metric.
- A highly specialized strategy to solve linear systems, which allows for lower and scalable requirements, both for time and memory.

Among other features, the proposed method heavily exploits the network structure of the constraint matrix in the problem. Several interior point algorithms have been developed for network optimization problems: in [29], the Authors derive a specialized IPM for the minimum cost flow problem on bipartite networks; the structure of the constraint matrix and normal equations is very similar to the one presented here. In [107], another specialized IPM method is applied to the same type of networks. In [53], preconditioners for IPMs applied to minimum cost flow problems are discussed. In the current work however, the IPM is mixed with a column generation technique and the linear algebra is specialized even further, in order to exploit the extreme sparsity of the solution of the optimal transport problem.

The proposed method is compared with the IBM ILOG Cplex [1] network simplex solver [20, 99], which is a highly optimized commercial software that has been shown to be very fast and reliable when dealing with OT problems, and with the LEMON (Library for Efficient Modelling and Optimization in Networks) network simplex solver [82]. The computational experiments are performed on

the DOTmark collection of images [115], considering cost functions given by the $1-$norm, $2-$norm and $\infty-$norm; they show that the proposed method outperforms the Cplex solver and matches the performance of LEMON for many of the problems considered, while requiring less memory than both of them. We performed tests with extremely large problems, with up to 4.3 billion variables and show that the proposed method is scalable both in terms of computational time and memory requirements.

The rest of the Chapter is organized as follows: in Section 5.2 we introduce the discrete optimal transport problem formulation; in Section 5.3 we present the hybrid interior point-column generation method; in Section 5.4 we analyze the structure of the normal equations and introduce the mixed iterative-direct approach for their solution; in Section 5.5 we present the test problems and show the computational results.

5.2 From optimal transport to optimization

Below we recall the Kantorovich formulation [77] of the discrete Optimal Transport problem: given a starting vector $\mathbf{a} \in \mathbb{R}^m_+$ and a final vector $\mathbf{b} \in \mathbb{R}^n_+$, such that $\sum \mathbf{a}_j = \sum \mathbf{b}_j$, find a coupling matrix $\mathcal{P}$ inside the set

$$U(\mathbf{a}, \mathbf{b}) = \left\{ \mathcal{P} \in \mathbb{R}^{m \times n}_+,\ \mathcal{P}\mathbf{e}_n = \mathbf{a},\ \mathcal{P}^T \mathbf{e}_m = \mathbf{b} \right\} \tag{5.1}$$

that is optimal with respect to a certain cost matrix $\mathcal{C} \in \mathbb{R}^{m \times n}_+$; i.e. find the solution of the following optimization problem

$$\min_{\mathcal{P} \in U(\mathbf{a}, \mathbf{b})} \sum_{i,j} \mathcal{C}_{ij} \mathcal{P}_{ij}. \tag{5.2}$$

We can interpret this OT problem as minimizing the cost of moving some mass in the configuration $\mathbf{a}$ into the configuration $\mathbf{b}$: $\mathcal{C}_{ij}$ gives the cost of moving a unit of mass from $\mathbf{a}_i$ to $\mathbf{b}_j$ and the optimal solution $\hat{\mathcal{P}}_{ij}$ tell us how much we should move from $\mathbf{a}_i$ to $\mathbf{b}_j$. The constraints given by the set $U(\mathbf{a}, \mathbf{b})$ impose three conditions: we only move positive quantities of mass; we ensure that from each bin i of configuration $\mathbf{a}$, we move out exactly a quantity $\mathbf{a}_i$ overall; we ensure that for each bin j of configuration $\mathbf{b}$, we move in exactly a quantity $\mathbf{b}_j$ overall.

In practice, the "mass" of $\mathbf{a}$ and $\mathbf{b}$ could be anything, from a probability distribution to actual physical quantities that need to be moved. If the cost matrix $\mathcal{C}$ is given by a distance raised to the power q (e.g. if $\mathcal{C}_{ij} = \mathcal{D}^q_{ij}, \forall i, j$, where $\mathcal{D}$ is a distance over the set of integers $1, \ldots, n$, when $n = m$, see [103, Proposition 2.2]), then the optimal solution of (5.2) defines a distance between $\mathbf{a}$ and $\mathbf{b}$, called the *q-Wasserstein* distance:

$$W_q(\mathbf{a}, \mathbf{b}) = \Big(\sum_{i,j} \mathcal{C}_{ij} \hat{\mathcal{P}}_{ij} \Big)^{1/q}. \tag{5.3}$$

5.2.1 Kantorovich linear program

We can rewrite the optimization problem (5.2) as a standard linear program (1.7):

$$\begin{aligned} \min_{\mathbf{p}\in\mathbb{R}^{mn}} \quad & \mathbf{c}^T\mathbf{p} \\ \text{s.t.} \quad & \begin{bmatrix} \mathbf{e}_n^T \otimes I_m \\ I_n \otimes \mathbf{e}_m^T \end{bmatrix} \mathbf{p} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} =: \mathbf{f} \\ & \mathbf{p} \geq 0, \end{aligned} \tag{5.4}$$

where $\otimes$ denotes the Kronecker product and $\mathbf{c} \in \mathbb{R}^{mn}$ and $\mathbf{p} \in \mathbb{R}^{mn}$ are the vectorized versions of $\mathcal{C}$ and $\mathcal{P}$ respectively, $\mathbf{c} = \text{vec}(\mathcal{C})$ and $\mathbf{p} = \text{vec}(\mathcal{P})$.

The matrix of constraints in (5.4) is the incidence matrix of a complete bipartite graph, as shown in (1.2); therefore, its structure is

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_n^T \otimes I_m \\ I_n \otimes \mathbf{e}_m^T \end{bmatrix}, \tag{5.5}$$

where A_1 is an operator that computes the sum of the entries of the rows of $\mathcal{P}$, while A_2 computes the sum of the entries of the columns. Notice that in this chapter the incidence matrix is denoted by A, rather than E as in Section 1.1.1, since it is the constraint matrix of the optimization problem. Notice also that matrix A is rank deficient by 1. These operators can be applied in a matrix-free way. Using Property 1.1 and Property 1.2, we can apply matrices A_1, A_2, A_1^T and A_2^T to a vector as follows:

$$A_1\mathbf{x} = (\mathbf{e}_n^T \otimes I_m)\mathbf{x} = \text{vec}(I_m X \mathbf{e}_n) = X\mathbf{e}_n,$$

$$A_2\mathbf{x} = (I_n \otimes \mathbf{e}_m^T)\mathbf{x} = \text{vec}(\mathbf{e}_m^T X I_n) = X^T\mathbf{e}_m,$$
$$A_1^T\mathbf{u} = (\mathbf{e}_n \otimes I_m)\mathbf{u} = \text{vec}(I_m \mathbf{u}\mathbf{e}_n^T) = \text{vec}(\mathbf{u}\mathbf{e}_n^T),$$
$$A_2^T\mathbf{w} = (I_n \otimes \mathbf{e}_m)\mathbf{w} = \text{vec}(\mathbf{e}_m \mathbf{w}^T I_n) = \text{vec}(\mathbf{e}_m \mathbf{w}^T),$$

where $\mathbf{x} \in \mathbb{R}^{mn}$, $X \in \mathbb{R}^{m\times n}$, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{w} \in \mathbb{R}^n$. Notice that the matrices involved X, $\mathbf{u}\mathbf{e}_n^T$ and $\mathbf{e}_m\mathbf{w}^T$ are all of dimension $m \times n$, and thus much smaller than matrix A.

5.2.2 Graph formulation

We can reformulate the optimal transport problem as a minimum cost flow problem over a bipartite graph: we have m source nodes, each one with an output $\mathbf{a}_i$, connected to n sink nodes, each requiring an input $\mathbf{b}_j$.

A known result about the optimal solution of the optimal transport problem is the following:

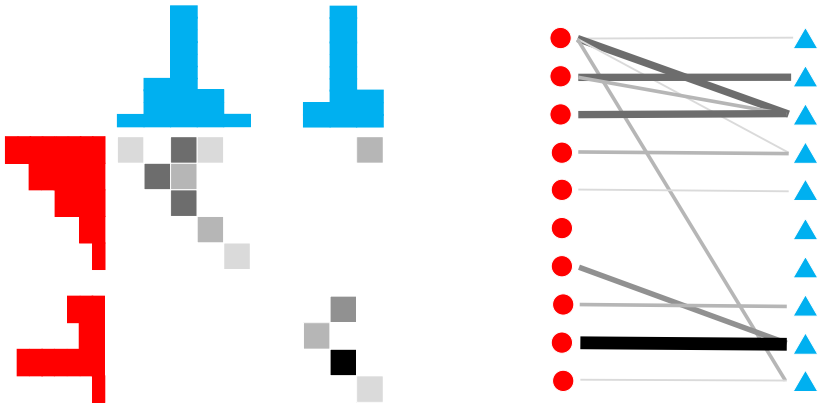
Proposition 5.1. *Given a vertex V of the polytope $U(\mathbf{a}, \mathbf{b})$ in (5.1), the bipartite graph corresponding to matrix V, where edges corresponding to zero flow have been removed, is acyclic.*

Proof. See [5] or [79] or [103]. □

In particular, this implies that there exists an optimal solution $\hat{\mathcal{P}}$ with at most $m + n - 1$ nonzero entries.

Figure 5.1 shows a small example of discrete optimal transport and the corresponding bipartite graph formulation: the problem is to move the red configuration (on the left) onto the blue configuration (on the top), where the cost is given by the physical distance between the bins of the histogram.

Figure 5.1: A small example of discrete optimal transport and the corresponding bipartite graph formulation; here the intensity of the colour is proportional to the quantity of mass to be moved.



Remark 5.1. *Notice that the optimal transport linear program can have multiple solutions, even if the cost matrix represents physical distances. For example, consider a problem with $m = n = 4$, where the supply vector is $\mathbf{a} = [2\,3\,3\,4]^T$ and the demand vector is $\mathbf{b} = [1\,3\,3\,5]^T$. Suppose that the cost matrix is given by the 1-dimensional distance between the positions, i.e.*

$$\mathcal{C} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}.$$

Then, the following are all optimal solutions with the same total cost of 3:

$$\mathcal{P} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}, \quad \mathcal{P} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 4 \end{bmatrix},$$

$$\mathcal{P} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 4 \end{bmatrix}, \quad \mathcal{P} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}.$$

5.3 Interior-point-inspired algorithm for optimal transport

Problem (5.4) is a standard linear program, that can be solved using an interior point method. However, the constraint matrix can be extremely large for large values of m and n: indeed, the number of variables is $m \cdot n$ and the number of constraints is $m + n$. A linear program with such a structure of the constraint matrix is well suited to be solved by a column generation approach. We propose a method that mixes these two techniques: an interior-point-inspired method that keeps the iterates sparse at each iteration and updates the list of variables that are allowed to be non-zero in a similar way to the column generation method. We use the term *support* for this list of presumed "basic" variables, to avoid confusion with the notion of *basis* in the simplex and IPM communities.

In the next sections, we first introduce the standard IPM formulation and then the sparsified version, highlighting the relation with standard IPM and column generation algorithms, and the pros and cons of using such an approach.

5.3.1 Interior point method

Recall from Section 2.1 that at each IPM iteration, one step of the Newton method is performed and the linear system which needs to be solved is

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I_{mn} \\ S & 0 & P \end{bmatrix} \begin{bmatrix} \mathbf{\Delta p} \\ \mathbf{\Delta y} \\ \mathbf{\Delta s} \end{bmatrix} = \begin{bmatrix} \mathbf{r_1} \\ \mathbf{r_2} \\ \mathbf{r_3} \end{bmatrix} = \begin{bmatrix} \mathbf{f} - A\mathbf{p} \\ \mathbf{c} - A^T\mathbf{y} - \mathbf{s} \\ \sigma\mu\mathbf{e}_{mn} - PS\mathbf{e}_{mn} \end{bmatrix}. \tag{5.6}$$

When reduced to the normal equations approach, it takes the form

$$A\Theta A^T \mathbf{\Delta y} = \mathbf{r_1} + A\Theta(\mathbf{r_2} - P^{-1}\mathbf{r_3}) \tag{5.7}$$

where $\Theta = \text{diag}(\boldsymbol{\theta})$ is a diagonal matrix; its entries are $\boldsymbol{\theta}_j = \mathbf{p}_j/\mathbf{s}_j$, $\mathbf{s}$ being the dual slack variable associated with the LP (5.4), $P = \text{diag}(\mathbf{p})$ and $S = \text{diag}(\mathbf{s})$.

Notice that the linear system (5.7) is of dimension $(m + n)$ and thus much smaller than (5.6), that is of dimension $(m + n + 2mn)$; however, (5.7) contains blocks that are fully dense. Moreover, the IPM needs to work with many vectors of size $m \cdot n$, which requires excessive storage for large values of m and n.

5.3.2 Sparse approach

It has been observed (see e.g. [64]) that the entries of matrix Θ can be divided in two groups: for "basic" indices $j \in \mathcal{B}$, for which $\mathbf{p}_j \to \hat{\mathbf{p}}_j > 0$ and $\mathbf{s}_j \to \hat{\mathbf{s}}_j = 0$, $\boldsymbol{\theta}_j$ becomes very large as the IPM approaches convergence; for the "nonbasic" indices $j \in \mathcal{N}$, such that $\mathbf{p}_j \to \hat{\mathbf{p}}_j = 0$ and $\mathbf{s}_j \to \hat{\mathbf{s}}_j > 0$, $\boldsymbol{\theta}_j$ becomes very small. Concerning the structure of the normal equations matrix (5.7), this separation into "basic" and "nonbasic" indices implies that the following holds close to optimality

$$A\Theta A^T = \sum_{j=1}^{mn} \boldsymbol{\theta}_j A_j A_j^T \approx \sum_{j \in \mathcal{B}} \boldsymbol{\theta}_j A_j A_j^T \tag{5.8}$$

where A_j represents the $j-$th column of A.

In the non-degenerate case, the optimal solution $\hat{\mathbf{p}}$ is expected to have only $m + n - 1$ non-zeros, out of $m \cdot n$ entries: in the common scenario where $m = n$, this means that the density of the optimal solution decreases as $1/m$ and that at optimality most entries of $\boldsymbol{\theta}$ are close to zero, and only a small fraction of them is large. We can exploit these properties to derive a sparsified method, where each iteration is comprised by two phases: an interior point phase on the reduced problem, and a column-generation-style update of the support.

Notice that, while the primal variable is expected to be sparse at optimality, the dual slack $\mathbf{s}$ instead is expected to be dense, due to strict complementarity (see Theorem 1.3); in the proposed approach, only the sparsity pattern of the primal variable $\mathbf{p}$ is considered. Indeed, the entries for which $\mathbf{s}_j \to \hat{\mathbf{s}}_j > 0$ generate corresponding components $\boldsymbol{\theta}_j = \mathbf{p}_j / \mathbf{s}_j$ that converge to zero and are therefore ignored, in order to exploit (5.8).

In the *IPM phase*, we apply a standard IPM to a reduced form of the problem. We start from considering the subset of $\{1, 2, \ldots, mn\}$ that corresponds to the indices of the primal variables that are allowed to attain nonzero values; this subset, that we call `index`, contains the indices of the variables belonging to the current support. We would like to select the support such that if $j \in$ `index`, then $\hat{\mathbf{p}}_j > 0$, $\hat{\mathbf{s}}_j = 0$, and if $j \notin$ `index`, then $\hat{\mathbf{p}}_j = 0$, $\hat{\mathbf{s}}_j > 0$.

Entries $\mathbf{p}_j$ and $\mathbf{s}_j$ for $j \notin$ `index` are forced to be zero and the same applies to the components of the Newton direction, since we are not updating them; we expect $\mathbf{s}_j > 0$ for all $j \notin$ `index`, but we set them to zero because they are ignored by the IPM phase. We define as $\mathbf{p}_{\text{red}}$ and $\mathbf{s}_{\text{red}}$ the reduced $\mathbf{p}$ and $\mathbf{s}$ vectors that contain only the components corresponding to the `index` subset. The logarithmic barrier is applied only to the set of variables in the support and the complementarity measure μ is thus computed as $\mu = (\mathbf{p}_{\text{red}}^T \mathbf{s}_{\text{red}})/\psi$, where $\psi = |$`index`$|$. If we call A_{red} the submatrix of A obtained considering only the columns in `index`, then the linear system to solve becomes

$$\begin{bmatrix} A_{\text{red}} & 0 & 0 \\ 0 & A_{\text{red}}^T & I_\psi \\ S_{\text{red}} & 0 & P_{\text{red}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}\mathbf{p}_{\text{red}} \\ \boldsymbol{\Delta}\mathbf{y} \\ \boldsymbol{\Delta}\mathbf{s}_{\text{red}} \end{bmatrix} = \begin{bmatrix} \mathbf{r_1} \\ \mathbf{r_2}_{\text{red}} \\ \mathbf{r_3}_{\text{red}} \end{bmatrix} = \begin{bmatrix} \mathbf{f} - A_{\text{red}}\mathbf{p}_{\text{red}} \\ \mathbf{c}_{\text{red}} - A_{\text{red}}^T\mathbf{y} - \mathbf{s}_{\text{red}} \\ \sigma\mu\mathbf{e}_\psi - P_{\text{red}}S_{\text{red}}\mathbf{e}_\psi \end{bmatrix}. \tag{5.9}$$

With this approach, the diagonal of Θ is sparsified and thus the normal equations matrix is much more sparse than $A\tilde{\Theta}A^T$ for any possible diagonal $\tilde{\Theta}$ with strictly positive diagonal entries (as it would be for a standard IPM).

In the *update phase*, we update the subset `index` and expand or reduce the support; this can happen in two ways. The support is enlarged according to the reduced cost $\mathbf{c} - A^T\mathbf{y}$; if any component is negative, the indices corresponding to the most negative reduced costs are added to `index`. We expect these newly added variables to satisfy $\mathbf{p}_j \to \hat{\mathbf{p}}_j > 0$, $\mathbf{s}_j \to \hat{\mathbf{s}}_j = 0$ and thus $\boldsymbol{\theta}_j \gg 1$. The new entries are initialized with $(\mathbf{p}_{\text{red}})_j = (\mathbf{s}_{\text{red}})_j = \sqrt{\mu}$, in order to maintain the overall complementarity measure unaltered, at the expense of adding a small infeasibility. This approach has been used in warm starting IPMs in [62] and since then has been used with success in numerous applications of IPMs which requires warm starting with newly introduced variables.

Variables are removed from the support based on the value of $\mathbf{p}_{\text{red}}$; if $(\mathbf{p}_{\text{red}})_j$ is smaller than a certain threshold when the IPM is near convergence, we assume that this component should satisfy $\mathbf{p}_j \to \hat{\mathbf{p}}_j = 0$, $\mathbf{s}_j \to \hat{\mathbf{s}}_j > 0$ and thus $\boldsymbol{\theta}_j \ll 1$. Other techniques could be used to detect which variables should be removed, for example if $\boldsymbol{\theta}_j$ is very small, if $\mathbf{s}_j$ is very large, or using some specifically developed indicators such as in [46].

In order to not perturb too much the IPM algorithm, only m variables are allowed to enter or leave the support at every iteration. In practice, in the first iterations of IPM many variables enter the support, while no entry of $\mathbf{p}$ is small enough to leave it; in the late phase of IPM instead many variables leave the support, while almost no index is added to it.

The stopping criterion involves primal infeasibility, dual infeasibility of the current restricted problem and the complementarity measure; the method is stopped when

$$\max\left(\frac{\|\mathbf{r_1}\|}{1+\|\mathbf{f}\|}, \frac{\|\mathbf{r_{2\text{red}}}\|}{1+\|\mathbf{c}_{\text{red}}\|}, \mu\right) < \texttt{tol},$$

where `tol` is a predetermined tolerance. The method could also check if there are variables with negative reduced cost that should still be added to the support; however, in practice this was never the case, since the support would stabilize (and actually start to drop variables) before the IPM indicators would reach convergence.

The initial `index` subset can be chosen in many ways according to heuristics that try to capture the sparsity pattern of the optimal solution. A simple approach is to include in the initial support the variables that correspond to a small cost $\mathbf{c}_j$, according to a predetermined threshold, since the optimal solution will try to allocate as much mass as possible in these low-cost variables. We know that the optimal support should include only $n + m - 1$ entries, but at the beginning we choose the subset `index` with a larger number of entries, usually between three and ten times more than the expected number of indices in the `index` subset corresponding to the optimal solution.

5.3.3 Comments on the method

In a column generation method, it is possible not to solve the restricted master problems to optimality and still converge to a solution of the master problem (see e.g. [66, 67]); the approach presented here can be seen as an extreme situation, where only one IPM iteration is applied to each restricted master problem before updating the support. In this way, the method does not "waste" too much time optimizing the early restricted master problems, which have a very inexact support, while it spends most of the time looking for the correct support and optimizing to full accuracy the late restricted master problems (when the support has stabilized).

The method is clearly not an interior point method, since the iterates are sparsified and do not belong to the interior of the feasible region. Such a method benefits from the sparsity of the vectors and matrices involved, at the expense of losing some dual information. Indeed, while vectors $\mathbf{p}$ and $\boldsymbol{\theta}$ are expected to be extremely sparse close to optimality, the dual variable $\mathbf{s}$ is not, due to the strict complementarity of linear programs. The stopping criterion indeed relies only on the dual infeasibility of the restricted problem. However, the use of a primal-dual method allows to obtain fast convergence once the support of the optimal solution has been established. An interior point method that accurately captures all the dual information would inevitably need to use fully dense vectors and normal equations matrices, which reduces substantially its applicability to large problems. The method presented here instead can be applied to huge problems, but does not reconstruct accurately the dual information during the iterations; however, the optimal dual variable $\hat{\mathbf{s}}$ can be computed after the algorithm terminates with one extra reduced costs computation $\hat{\mathbf{s}} = \mathbf{c} - A^T\hat{\mathbf{y}}$, since the Lagrange multiplier $\hat{\mathbf{y}}$ is reconstructed accurately.

We highlight that the IPM phase uses multiple centrality correctors [33] to improve the performance; a standard symmetric neighbourhood of the central path (see e.g. [131]) is used to find the correctors. Each iteration of the method however relies on a neighbourhood built with a different support. The stepsize of the method is found simply by computing the step to the boundary and scaling it down by a constant close to 1 (e.g. 0.995).

Problems for which the support changes heavily between subsequent iterations (for example because the initial and final configuration $\mathbf{a}$ and $\mathbf{b}$ have mass concentrated on a narrow region and their nonzero patterns have small or no intersection) can create difficulties for the method, because the next IPM iterations would solve a restricted problem with potentially a very different support than the current one. These difficulties will appear for some specific problems in the numerical results section, where a large number of iterations is required before the support stabilizes.

5.3.4 Pricing method.

In order to update the support, we need to compute the full reduced cost vector $\mathbf{c} - A^T\mathbf{y}$; if m and n are large, this can be very expensive. To reduce the cost of

this operation, we propose a heuristic pricing approach. Notice the following

$$(\mathbf{c} - A^T\mathbf{y})_j = \mathbf{c}_j - \mathbf{y}^T\mathbf{a}_j = \mathbf{c}_j - \mathbf{y}_{k_{1j}} - \mathbf{y}_{k_{2j}},$$

where $\mathbf{a}_j$ is the $j-$th column of A and k_{1j} and k_{2j} are the source and destination nodes of edge j, defined as

$$k_{1j} = \begin{cases} (j \mod m) & \text{if } (j \mod m) \neq 0 \\ m & \text{if } (j \mod m) = 0 \end{cases}, \qquad k_{2j} = \left\lceil \frac{j}{m} \right\rceil.$$

We make the assumption that a large portion of the smallest (most negative) reduced costs corresponds to small values of $\mathbf{c}_j$; the accuracy of this statement depends on the values attained by the Lagrange multiplier $\mathbf{y}$, but we observed in practice that it is true in most of the iterations. Therefore, before starting the algorithm, we can compute a subset of indices J, such that if $j \in J$, $\mathbf{c}_j < C_{\max}$, for some fixed value $C_{\max}$, and the corresponding indices k_{1j} and k_{2j}. Then, during the update phase, we can quickly compute the subset of the entries of the reduced cost vector corresponding to the indices in J and use only these to update the support. Of course, this method misses some of the variables to be added; thus, after a certain number of IPM iterations where we use this method, we should compute the full vector of reduced costs. In this way, we keep low the cost of a single iteration, but we likely increase the number of iterations required. The number of consecutive iterations in which we use the heuristic needs to be chosen carefully: we would like it to be large, to reduce the computational cost as much as possible, but if it is too large we risk performing useless iterations, since the next update with the full reduced costs might change drastically the support. Numerical evidence suggests that the reduced costs should be refreshed every 3 or 4 iterations.

5.3.5 Structure of the normal equations matrix

The normal equations matrix in (5.7) takes the form

$$A\Theta A^T = \begin{bmatrix} M & V \\ V^T & N \end{bmatrix} \tag{5.10}$$

where $V \in \mathbb{R}^{m\times n}$, $\text{vec}(V) = \boldsymbol{\theta}$, $M \in \mathbb{R}^{m\times m}$ and $N \in \mathbb{R}^{n\times n}$ are diagonal matrices with

$$M_{ii} = \sum_{t=0}^{n-1} \boldsymbol{\theta}_{i+tm},\ i = 1, \ldots, m, \qquad N_{jj} = \sum_{t=1}^{m} \boldsymbol{\theta}_{(j-1)m+t},\ j = 1, \ldots, n.$$

To see this, notice that

$$A\Theta A^T = \begin{bmatrix} A_1\Theta A_1^T & A_1\Theta A_2^T \\ A_2\Theta A_1^T & A_2\Theta A_2^T \end{bmatrix}.$$

Let us compute the $(1,1)$ term.

$$(A_1 \Theta A_1^T)_{ij} = \sum_{k=1}^{mn} (A_1)_{ik}(A_1)_{jk}\boldsymbol{\theta}_k.$$

Given the structure of A_1 in (5.5), the only nonzero terms in the summation are obtained if $i = j = (k \mod m)$; in this case $(A_1)_{ik} = (A_1)_{jk} = 1$ and we obtain

$$(A_1 \Theta A_1^T)_{ij} = \delta_{ij} \sum_{t=0}^{n-1} \boldsymbol{\theta}_{j+tm},$$

where δ_{ij} is the Kronecker delta. Similarly, we can compute the $(2,2)$ term

$$(A_2 \Theta A_2^T)_{ij} = \sum_{k=1}^{mn} (A_2)_{ik}(A_2)_{jk}\boldsymbol{\theta}_k.$$

In this case, the nonzero terms are given by $i = j = \lceil k/m \rceil$. Therefore

$$(A_2 \Theta A_2^T)_{ij} = \delta_{ij} \sum_{t=1}^{m} \boldsymbol{\theta}_{(j-1)m+t}.$$

Let us now compute the terms in the off-diagonal blocks

$$(A_1 \Theta A_2^T)_{ij} = \sum_{k=1}^{mn} (A_1)_{ik}(A_2)_{jk}\boldsymbol{\theta}_k.$$

For any combination of i and j, there is only one value of k that produces a nonzero coefficient in the summation, so

$$(A_1 \Theta A_2^T)_{ij} = \boldsymbol{\theta}_{k(i,j)},$$

where $k(i,j)$ is identified by $i = (k \mod m)$ and $j = \lceil k/m \rceil$. With this in mind, it should be clear that

$$\mathrm{vec}(A_1 \Theta A_2^T) = \boldsymbol{\theta}.$$

As an example, if $m = 2$ and $n = 3$, the normal equations matrix would take the following form

$$\begin{bmatrix} \boldsymbol{\theta}_1 + \boldsymbol{\theta}_3 + \boldsymbol{\theta}_5 & 0 & \boldsymbol{\theta}_1 & \boldsymbol{\theta}_3 & \boldsymbol{\theta}_5 \\ 0 & \boldsymbol{\theta}_2 + \boldsymbol{\theta}_4 + \boldsymbol{\theta}_6 & \boldsymbol{\theta}_2 & \boldsymbol{\theta}_4 & \boldsymbol{\theta}_6 \\ \boldsymbol{\theta}_1 & \boldsymbol{\theta}_2 & \boldsymbol{\theta}_1 + \boldsymbol{\theta}_2 & 0 & 0 \\ \boldsymbol{\theta}_3 & \boldsymbol{\theta}_4 & 0 & \boldsymbol{\theta}_3 + \boldsymbol{\theta}_4 & 0 \\ \boldsymbol{\theta}_5 & \boldsymbol{\theta}_6 & 0 & 0 & \boldsymbol{\theta}_5 + \boldsymbol{\theta}_6 \end{bmatrix}.$$

Notice the meaning of matrices M and N:

$$V\mathbf{e}_n = M\mathbf{e}_m, \qquad V^T\mathbf{e}_m = N\mathbf{e}_n. \tag{5.11}$$

i.e. M_{kk} is the sum of the entries of row k of V, while N_{kk} is the sum of the entries of column k of V.

In a standard IPM, matrix V would be completely dense, since the entries $\boldsymbol{\theta}_j$ are all strictly positive. However, in the proposed hybrid method most entries of $\boldsymbol{\theta}$ are exactly zero, making matrix V extremely sparse.

5.4 Solution of the normal equations

To solve a linear system involving matrix (5.10), we can use the Schur complement approach: the solution of

$$\begin{bmatrix} M & V \\ V^T & N \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha_1} \\ \boldsymbol{\alpha_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\beta_1} \\ \boldsymbol{\beta_2} \end{bmatrix}$$

is given by either

$$\boldsymbol{\alpha_1} = S_N^{-1}(\boldsymbol{\beta_1} - VM^T\boldsymbol{\beta_2}), \quad \boldsymbol{\alpha_2} = N^{-1}(\boldsymbol{\beta_2} - V^T\boldsymbol{\alpha_1}),$$

or

$$\boldsymbol{\alpha_2} = S_M^{-1}(\boldsymbol{\beta_2} - V^TM^{-1}\boldsymbol{\beta_1}), \quad \boldsymbol{\alpha_1} = M^{-1}(\boldsymbol{\beta_1} - V\boldsymbol{\alpha_2}),$$

where $S_N = M - VN^{-1}V^T$ and $S_M = N - V^TM^{-1}V$ are the two Schur complements. Such an approach is convenient in this case since matrices M and N are diagonal and thus very easy to invert. Notice that both Schur complements are rank deficient by 1, but this is not a problem in practice: both iterative methods and direct solvers can deal with singular matrices by solving the corresponding least squares problem. A common strategy to avoid this issue is to remove one of the nodes of the graph; however, doing so would perturb the structure of the constraint matrix (5.5) and make the matrix-vector operations with A_1 and A_2 slightly more complicated to implement. Since the rank deficiency is not producing any particular issue, we preferred not to remove any nodes in this application.

Let us analyze S_M: if we sum the rows of matrix $V^TM^{-1}V$ we obtain, exploiting (5.11):

$$V^TM^{-1}V\mathbf{e}_n = V^TM^{-1}M\mathbf{e}_m = V^T\mathbf{e}_m = N\mathbf{e}_n.$$

The Schur complement S_M is equal to matrix N minus the previous matrix.

Therefore

$$\begin{aligned}
|(S_M)_{kk}| - \sum_{i\neq k} |(S_M)_{ki}| &= |N_{kk} - (V^T M^{-1} V)_{kk}| - \sum_{i\neq k} |(V^T M^{-1} V)_{ki}| \\
&= N_{kk} - (V^T M^{-1} V)_{kk} - \sum_{i\neq k} (V^T M^{-1} V)_{ki} \\
&= N_{kk} - \sum_{i=1}^{n} (V^T M^{-1} V)_{ki} \\
&= N_{kk} - (V^T M^{-1} V \mathbf{e}_n)_k = 0. \qquad (5.12)
\end{aligned}$$

This means that the Schur complement S_M is weakly diagonally dominant. If we consider the other Schur complement $S_N = M - VN^{-1}V^T$, then

$$VN^{-1}V^T\mathbf{e}_m = VN^{-1}N\mathbf{e}_n = V\mathbf{e}_n = M\mathbf{e}_m$$

and we conclude similarly that

$$|(S_N)_{kk}| - \sum_{i\neq k} |(S_N)_{ki}| = 0. \qquad (5.13)$$

5.4.1 Sparsity pattern of the Schur complement

We use Theorem 1.1 to analyze the sparsity pattern of the Schur complements obtained using the optimal solution $(\hat{\mathbf{p}}, \hat{\mathbf{y}}, \hat{\mathbf{s}})$; however, when the algorithm approaches optimality and the complementarity products get close to zero, the entries of Θ tend to zero or infinity. For the sake of formalism, define the matrix $\mathcal{V}$ as the sparsity pattern of V; then during the IPM iterations, $\mathcal{V} \to \hat{\mathcal{V}}$, where

$$\begin{cases} \hat{\mathcal{V}}_{ij} = 1 & \text{if} \quad \hat{\mathcal{P}}_{ij} > 0 \\ \hat{\mathcal{V}}_{ij} = 0 & \text{if} \quad \hat{\mathcal{P}}_{ij} = 0 \end{cases},$$

where $\text{vec}(\hat{\mathcal{P}}) = \hat{\mathbf{p}}$. At any IPM iteration, the Schur complements have the same sparsity pattern as VV^T or V^TV, because they involve only V, V^T and diagonal matrices; therefore, getting close to optimality, the sparsity patterns of the Schur complements get close to the sparsity pattern $\hat{\mathcal{V}}\hat{\mathcal{V}}^T$ or $\hat{\mathcal{V}}^T\hat{\mathcal{V}}$. The next result shows that these are chordal matrices.

Corollary 5.1. *Matrices $\hat{\mathcal{V}}\hat{\mathcal{V}}^T$ and $\hat{\mathcal{V}}^T\hat{\mathcal{V}}$ corresponding to an optimal solution $\hat{\mathbf{p}}$ that is a vertex of the transportation polytope (5.1) have chordal sparsity patterns.*

Proof. Recall the graph formulation presented in Section 5.2.2: $\hat{\mathcal{P}}$ is the biadjacency matrix of the bipartite graph corresponding to the optimal solution, which is known to be acyclic. By construction, $\hat{\mathcal{V}}$ and $\hat{\mathcal{P}}$ have the same sparsity pattern. Therefore, matrix $\hat{\mathcal{V}}$ satisfies the assumption of Theorem 1.1. Notice that if there is more than one optimal solution, at least one of them has to correspond to an acyclic bipartite graph; therefore, the corollary holds at least for this specific solution $\hat{\mathbf{p}}$. □

Therefore, the two Schur complements S_M and S_N tend to become chordal, which is a very desirable property. In order to choose which of the two Schur complements to consider, we can use Lemma 1.2: the number of nonzeros of S_N is controlled by the maximum number of nonzeros in any column of matrix V; the number of nonzeros of S_M instead is determined by the maximum number of nonzeros in any row of V. We could choose dynamically which Schur complement to compute and factorize based solely on the number of nonzero entries in the rows and columns of matrix V.

5.4.2 Mixing direct and iterative solvers

We consider two options for solving system (5.7) with matrix (5.10): direct approach using Cholesky factorization and iterative one using preconditioned conjugate gradient.

We can identify two stages of the IPM. In the early iterations, the support is far from the optimal one, so we expect that if we compute the Cholesky factorization of the Schur complement, there would be a lot of fill-in; in this phase though, Θ is well conditioned and we can expect a simple preconditioner like incomplete Cholesky to work well. In the late iterations instead, we know that the normal equations matrix becomes extremely ill-conditioned due to the behaviour of matrix Θ, which makes it difficult to find a good preconditioner for the conjugate gradient method; however, computing a full Cholesky factorization of the Schur complement would be extremely cheap, due to Theorem 1.2 and Corollary 5.1, which shows that both S_M and S_N get closer and closer to a sparse chordal matrix.

The proposed approach first applies the conjugate gradient with incomplete Cholesky as preconditioner, with a value of the drop tolerance that is lowered every time too many linear iterations are performed; then, based on a switching criterion, when we decide that the Schur complement is "close enough" to the optimal one, we switch to a direct solution using an exact factorization with approximate minimum degree ordering.

The Schur complements are weakly diagonally dominant, as shown in (5.12)-(5.13) and this guarantees that the incomplete factorization never breaks down in exact arithmetic [86]; however, since the matrices are only weakly diagonally dominant, we may need to lift their diagonal with a small perturbation, in order to prevent from numerical inaccuracies making the matrices lose diagonal dominance property.

When using a full factorization, we employ the LDL^T algorithm; since both Schur complements are singular, we expect one entry of matrix D to be zero. We deal with this problem adding a small shift to the diagonal of D, since the entry that should be zero could become negative due to numerical errors, and using the Matlab backslash operator to apply matrices L and D. Notice that this issue could also be dealt with using a modified Cholesky factorization that substitutes small pivots with an infinitely large value in the computation of the factors.

Remark 5.2. *Notice that in this work the plain Cholesky factorization from Matlab was employed; however, to achieve maximum efficiency, it is possible to update a symbolic factorization, that keeps track of the sparsity pattern of the*

factors, every time the support is updated.

5.4.3 Switching strategy

In order to switch between the iterative solver and the direct factorization, we need to be sure that the level of fill-in in the Cholesky factor is not going to be too large, or otherwise the method may lose efficiency. Matrix V which is used to build the Schur complement is the biadjacency matrix of a bipartite graph that is not perfectly acyclic (this happens only at optimality), but gets closer and closer to being so; we can expect the number of cycles found in the graph to decrease and correspondingly the fill-in level of the Cholesky factor of the Schur complements to be reduced when the IPM gets close to optimality. One strategy to switch between iterative and direct solver can be based on the number of cycles that are found in the bipartite graph associated with matrix V at any IPM iteration; however, counting the number of cycles in the graph can be quite expensive. A simpler approach is to count the number of edges in the bipartite graph to decide when to switch: intuitively, the more edges there are, the more likely it is to find cycles; moreover, if the number of edges is decreasing fast, it means that the IPM is getting close to optimality and thus we can expect the Cholesky factor to be sufficiently sparse. Notice that each edge corresponds to a variable that is allowed to be nonzero; the number of edges is thus readily available, since it is the number of variables in the support. Therefore, we can switch from iterative to direct solver as soon as we detect that the number of variables in the support is decreasing fast enough: to do so, we compute how many variables were removed in the last 5 iterations as a fraction of the total number of variables, and switch method as soon as this number is sufficiently large.

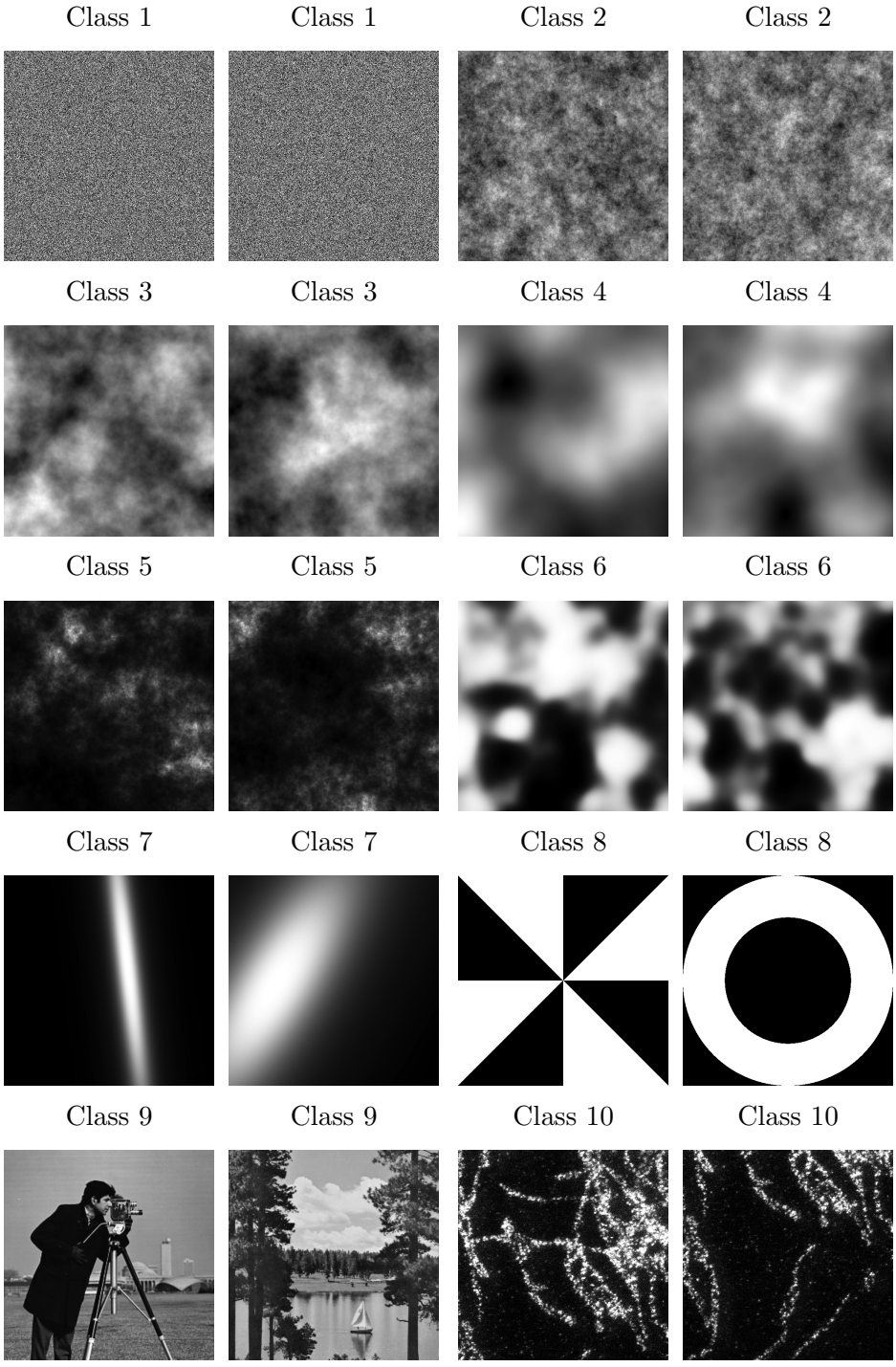
Remark 5.3. *Notice that it is possible to employ other switching strategies, for example based on the value of the IPM parameter μ; however, there is little difference in the final results and the current strategy, based on the speed of edge removal, is particularly simple to tune. Notice also that, if the number of variables in the support happens to increase after the method has switched to the full factorization, it may be necessary to switch back to an iterative solution of the linear system. This problem never occurred in the experiments presented here, but such a strategy may be required for solving other problems.*

5.5 Numerical results

5.5.1 Test problems

We tested the proposed algorithm on the *DOTmark* (Discrete Optimal Transport benchmark) collection of problems [115]. It includes 10 classes of images, each containing 10 images; the images come from simulations based on various probability distributions (class 1-7), geometric shapes (class 8), classic test images (class 9) and scientific observations using microscopy (class 10). Figure 5.2 shows two examples from each class.

Figure 5.2: Example of images from each class of the DOTmark collection.



Within a certain class, we can solve the OT problem between any couple of images, giving 45 problems per each class and a total of 450 instances overall. The images are available at different resolutions: we consider mainly the case 32×32, 64×64 and 128×128 pixels and show partial results also with 256×256. The problems that arise for a certain resolution `res` have a number of constraints equal to $2\,\texttt{res}^2$ and a number of variables $\texttt{res}^4$; Table 5.1 shows the dimension of the problems considered.

Table 5.1: Size of the problems considered

`res`	Constraints	Variables ($\times 10^6$)
32	2,048	1.0
64	8,192	16.8
128	32,768	268.4
256	131,072	4,295.0

As cost function, we consider the $1-$distance, $2-$distance and $\infty-$distance: the vectors $\mathbf{a}$ and $\mathbf{b}$ represent the vectorized forms of two images $\mathcal{A}$ and $\mathcal{B}$; $\mathcal{C}_{ij}$ is the cost of moving mass from position i to position j; in the images, position i corresponds to the entry $\mathcal{A}_{\alpha_i,\beta_i}$, while position j corresponds to $\mathcal{B}_{\alpha_j,\beta_j}$. Then, the distances used are

$$\mathcal{C}^1_{ij} = |\alpha_i - \alpha_j| + |\beta_i - \beta_j|,$$
$$\mathcal{C}^2_{ij} = \sqrt{(\alpha_i - \alpha_j)^2 + (\beta_i - \beta_j)^2},$$
$$\mathcal{C}^\infty_{ij} = \max(|\alpha_i - \alpha_j|, |\beta_i - \beta_j|).$$

Notice the following: if we consider an image with k pixels per side, the maximum possible distance between two pixels is approximately $2k$, $\sqrt{2}k$ or k, respectively for the 1, 2 and ∞ distance; all of these values grow linearly with k. Therefore, the parameter $C_{\max}$ in Section 5.3.4 should also scale linearly with the size of the image considered: in this way, the pricing heuristics considers always the same fraction of the total number of variables for each resolution.

5.5.2 Solvers

There are many methods that have been proposed for the solution of the Kantorovich linear program (5.4) (see e.g. [10, 36, 51, 72, 83, 88, 113]); a comparison of them can be found in [115]. We compared the proposed method to two very efficient implementations of the network simplex algorithm [20, 99], coming from IBM Cplex [1] and the LEMON (Library for Efficient Modelling and Optimization in Networks) library [3, 82].

Cplex is a highly optimized commercial software that showed excellent reliability and consistency of computational times throughout all the 10 classes of the DOTmark collection in [115]. To use it directly from its callable library, the optimal transport problems are generated in `C` and the network structure, in terms of nodes and edges, is passed to a Cplex CPXNET model. Only the network

optimizer of Cplex is considered in this work, since the standard dual simplex and the barrier solver were approximately 10 to 20 times slower, for resolutions 32 and 64 (this fact was also noticed in [115]).

LEMON [82] is a library for network optimization written in `C++` that has shown to outperform Cplex in some applications [29]. Out of the four algorithms available in LEMON for the minimum cost flow problem (network simplex, cost scaling, capacity scaling and cycle cancelling), the network simplex was the method which gave the best results for the problems considered in this Chapter. To use LEMON, the OT problem is generated directly in `C++` in terms of nodes and edges lists, that are then passed to a LEMON network simplex model. However, since LEMON only accepts integer input data for a network simplex model, we had to convert the OT problem data to integer form (multiplying them by a large constant) before performing the computation. This approach may lead to a loss of accuracy and potentially to integer overflow if larger problems are considered. The model was run with the pivot rule parameter set to `block-search`.

The interior point method instead is implemented in Matlab, in a way that exploits as much as possible the built-in functions, to reduce the computational time required. The parameters for the IPM are: feasibility and optimality tolerance 10^{-6}; conjugate gradient tolerance for predictors 10^{-6} and correctors 10^{-3} (we used a different tolerance for predictors and correctors, as was shown in Section 4.5.1); maximum number of IPM iterations 200; maximum number of conjugate gradient iterations (for each call) 1000; maximum number of correctors 3. All the experiments were performed on the University of Edinburgh School of Mathematics computing server, which is equipped with four 3.3GHz octa-core Intel Gold 6234 processors and 500GB of RAM. The specific versions of the software used were as follows: Matlab R2018a, Cplex 20.1, Lemon 1.3.1 and GCC 4.8.5 as compiler.

Remark 5.4. *The Lemon library could not be installed on the computing server, but we could only use it by compiling the source code directly. The performance of Lemon relative to the other solvers, when installed on a local machine, may be slightly better than the one shown here.*

5.5.3 Results

We report the results for the whole DOTmark collection, for all three solvers, for resolutions 32, 64 and 128 pixels.

Remark 5.5. *The IPM is an inexact method, while the network simplex finds the exact solution. The accuracy of the approximate solution can be measured with the relative Wasserstein error (RWE):*

$$RWE(\mathbf{a},\mathbf{b}) = \left|\frac{W_2^{IPM}(\mathbf{a},\mathbf{b}) - W_2^{exact}(\mathbf{a},\mathbf{b})}{W_2^{exact}(\mathbf{a},\mathbf{b})}\right|$$

where the 2-Wasserstein distance is defined in (5.3). *Notice that*

$$\begin{aligned} RWE(\mathbf{a},\mathbf{b}) &= \left| \frac{\sqrt{\mathbf{c}^T\hat{\mathbf{p}}^{IPM}} - \sqrt{\mathbf{c}^T\hat{\mathbf{p}}^{exact}}}{\sqrt{\mathbf{c}^T\hat{\mathbf{p}}^{exact}}} \right| \\ &= \left| \frac{\mathbf{c}^T\hat{\mathbf{p}}^{IPM} - \mathbf{c}^T\hat{\mathbf{p}}^{exact}}{\mathbf{c}^T\hat{\mathbf{p}}^{exact} + \sqrt{\mathbf{c}^T\hat{\mathbf{p}}^{IPM}}\sqrt{\mathbf{c}^T\hat{\mathbf{p}}^{exact}}} \right| \\ &\leq \left| \frac{\mathbf{c}^T\hat{\mathbf{p}}^{IPM} - \mathbf{c}^T\hat{\mathbf{p}}^{exact}}{\mathbf{c}^T\hat{\mathbf{p}}^{exact}} \right|. \end{aligned}$$

We expect the last quantity to be of the order of magnitude of the IPM tolerance and thus the RWE to be even smaller.

In all the experiments performed, the RWE was of the order of $10^{-6} - 10^{-8}$, *which indicates that the IPM solution was very accurate.*

Table 5.2 reports the average results for each class, for resolutions 32, 64 and 128 and for each cost function considered, in terms of iterations (`iter`) and time (`time`) of the proposed method, Cplex time (`Cplex`) and Lemon time (`Lemon`). Table 5.3 instead reports the average over all the three cost functions for each class of the number of CG iterations performed (`CG`), the maximum fill-in percentage level of the factorization of the Schur complement (`fill`) and the number of iterative (`iter`) and direct (`dir`) iterations performed. Figure 5.3 shows the performance profiles of the computational time, while Figure 5.4 compares the time taken by the three solvers for each problem at resolution 128.

Table 5.2: Average results with the proposed IPM for each class and cost function

Dist	Class	32×32				64×64				128×128			
		`iter`	`time`	`Cplex`	`Lemon`	`iter`	`time`	`Cplex`	`Lemon`	`iter`	`time`	`Cplex`	`Lemon`
1	1	10.9	0.30	0.41	0.18	13.6	2.12	12.02	7.83	19.1	31.43	1,220.27	219.01
	2	11.5	0.35	0.36	0.18	18.0	3.84	11.17	7.25	35.4	108.06	1,140.72	216.28
	3	16.0	0.58	0.36	0.16	26.9	7.79	10.75	6.63	44.4	161.26	1,100.77	159.85
	4	20.3	0.84	0.34	0.17	38.9	15.69	10.77	6.42	68.0	279.05	1,097.77	145.91
	5	25.1	1.11	0.35	0.16	40.1	18.93	12.11	7.04	139.3	1,107.91	1,234.57	143.99
	6	18.7	0.60	0.40	0.17	36.0	16.88	13.25	7.54	57.4	227.49	1,355.14	217.01
	7	30.9	1.42	0.29	0.20	68.2	40.14	12.86	7.30	83.3	1,074.91	1,306.70	154.39
	8	17.2	0.58	0.33	0.16	50.2	51.89	11.66	6.55	74.8	729.58	1,189.68	155.40
	9	14.8	0.45	0.33	0.15	24.4	7.20	11.67	7.17	49.8	155.11	1,196.78	172.44
	10	22.2	0.79	0.37	0.18	41.2	18.30	10.80	6.41	60.1	246.45	1,107.03	156.48
2	1	18.5	0.69	0.51	0.18	26.2	5.16	13.48	7.57	30.4	45.01	1,139.28	190.94
	2	29.7	1.18	0.50	0.19	72.1	16.90	16.15	9.00	114.0	481.26	1,369.31	446.40
	3	49.8	2.22	0.53	0.18	90.6	30.47	18.66	9.41	150.9	814.64	1,583.77	600.02
	4	60.0	2.93	0.54	0.20	96.4	40.41	20.17	9.66	166.8	720.83	1,715.39	615.64
	5	56.2	2.50	0.56	0.20	108.6	43.24	21.77	9.69	181.7	2,628.16	1,846.61	502.28
	6	46.6	2.14	0.56	0.19	86.5	32.06	21.54	10.23	159.3	4,613.47	1,829.04	705.84
	7	63.6	3.55	0.56	0.23	106.3	56.12	21.38	9.40	183.1	902.35	1,819.55	332.04
	8	34.4	1.48	0.48	0.17	70.8	29.91	18.51	8.70	127.8	424.25	1,573.27	488.31
	9	43.8	1.94	0.53	0.18	99.7	30.85	18.57	10.33	145.5	501.34	1,577.44	644.97
	10	50.4	2.17	0.55	0.21	87.1	28.17	16.49	7.00	154.2	1,425.74	1,394.50	273.86
∞	1	11.7	0.48	0.46	0.18	15.0	3.57	12.67	7.06	19.1	36.52	945.47	199.09
	2	12.2	0.50	0.42	0.17	17.6	6.09	12.28	6.74	34.2	119.78	923.97	156.30
	3	15.6	0.75	0.36	0.15	25.2	10.33	12.35	6.40	48.4	210.68	928.24	133.26
	4	19.1	0.98	0.37	0.16	35.6	16.12	12.63	6.21	63.8	292.34	948.83	132.83
	5	23.7	1.31	0.38	0.16	36.4	16.56	12.62	6.52	84.4	467.41	949.53	127.95
	6	18.7	0.91	0.43	0.16	30.2	13.85	14.47	6.86	61.4	312.69	1,087.24	163.88
	7	28.3	1.69	0.40	0.18	50.5	25.33	13.00	6.41	98.7	1,663.26	966.95	128.30
	8	17.0	0.96	0.38	0.15	33.1	36.00	11.08	6.31	62.1	664.18	829.04	151.12
	9	15.0	0.67	0.42	0.15	23.9	9.46	12.11	6.60	49.2	212.13	907.62	139.94
	10	20.7	1.07	0.42	0.17	35.6	19.29	11.60	6.34	59.4	277.50	872.06	129.68

Table 5.3: Details of the proposed IPM for each class

	32×32				64×64				128×128			
Class	CG	fill	iter	dir	CG	fill	iter	dir	CG	fill	iter	dir
1	1,354.7	9.6	13.2	0.4	2,697.6	3.9	17.0	1.3	3,916.1	1.5	21.1	1.7
2	1,342.2	11.7	14.9	2.9	3,460.2	4.3	27.9	8.0	10,925.3	1.6	74.3	2.1
3	1,840.6	12.2	23.8	3.3	4,564.0	2.8	43.5	4.1	15,844.7	1.9	104.0	2.6
4	2,144.5	11.7	29.9	3.2	5,130.7	3.0	53.7	3.2	16,666.0	1.5	107.2	5.1
5	1,670.1	12.9	28.9	6.1	4,765.0	3.0	51.1	10.6	20,031.6	2.6	168.8	4.5
6	1,976.5	11.4	25.8	2.3	5,046.1	3.2	48.6	2.3	14,258.0	2.9	95.0	2.9
7	2,498.0	11.7	38.7	2.3	6,230.1	3.3	72.3	2.7	19,147.9	2.4	161.8	6.3
8	1,521.1	10.9	21.7	1.2	6,544.9	3.6	49.4	2.0	13,875.7	1.6	90.2	3.9
9	1,798.9	11.5	22.2	2.4	4,647.1	3.0	43.5	5.9	15,194.9	1.9	94.9	2.6
10	1,728.7	13.3	26.2	4.9	5,211.4	4.7	44.8	9.9	14,045.0	1.8	97.9	17.1

The reported computational time accounts only for the optimization phase. However, there is some pre-processing time that Cplex and Lemon need to take in order to prepare the network model. In particular, after generating the problem (in terms of the vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$), some arrays are created which contain the nodes of the graph and the respective supply, and the edges and the respective cost. Then, the network model for the respective solver needs to be created and the graph information needs to be passed to the model. These two phases, which are not required by the hybrid IP solver, in the larger instance of resolution 128 require overall approximately 5 to 10 seconds. This extra time needed by both Cplex and LEMON is not included in the table and in the performance profiles. Moreover, the creation of the full network requires a substantial amount of memory, that is avoided by the proposed method.

Remark 5.6. *The reader should keep in mind that the number of iterations presented refers to the overall process of adding/removing variables and optimizing; it is thus not a surprise that there are instances where the iteration count is higher than what would be expected from a standard IPM. The cost of a single iteration however is considerably lower, given the high degree of sparsity of the vectors and matrices involved.*

From these results, it is clear that for small problems (32×32) Lemon outperforms both Cplex and the proposed method; however, the IPM scales better and becomes competitive for larger instances. Cplex instead scales badly and is considerably slower than the other solvers at resolution 128. Given that both Cplex and Lemon are highly optimized libraries written in a compiled language, while the method proposed here is implemented in Matlab, we consider these results very promising.

From Figure 5.4, we can see that the proposed method behaves particularly well for problems in classes 1, 2, 3, 9 and 10, since the time taken is close to the one taken by Lemon, and overall the spread of the times for all the problems in these classes remains narrow. Classes 5, 7 and 8 instead are the ones with the weaker results: the computational time is considerably larger than the one taken by Lemon, with a large variability among the instances of these classes. Observing

the images in Figure 5.2, it seems that the method performs better when the mass of the image is distributed evenly everywhere, while it struggles when it is concentrated in a narrow region. It is also evident that all three solvers struggle more when using the 2−distance as cost function.

From Table 5.3 we can see that the maximum fill-in level is independent of the class of the images and gets smaller when the problem becomes larger, since the density of the optimal solution decreases as $1/\texttt{res}^2$. We can also see that on average only the very last IPM iterations employ a direct factorization, which is what we wanted to achieve with the switching criterion proposed.

5.5.4 Results for large instances

When considering problem with resolution 256×256, both Cplex and Lemon crash: they require more than 400GB of memory, in large part needed to store the huge network structure. The IPM however does not need to store the network and thus scales better in terms of memory, requiring at most approximately 120GB of memory; indeed, the only large data that the IPM stores are the cost vector $\mathbf{c}$, the reduced costs and the factorization of the Schur complement. Given the huge size of the problem, we show partial results for classes 1, 2 and 3 and distances 1 and ∞ (due to the long computational time required by the 2−norm cost function at resolution 256).

Figure 5.5 shows the growth of the average computational time, in a logarithmic plot, for the three solvers, from resolution 32 to 128 (for Cplex and Lemon) or 256 (for the IPM), for the three classes considered; the dotted line shows a linear growth for comparison.

This Figure highlights that the computational time of the proposed method grows linearly with the number of variables, while the other two solvers scale super-linearly. The growth of the computational time of the IPM appears to be sub-linear in some cases, for instance for the problems in class 1. A possible explanation is the following: while the number of variables in the optimization problem grows by a factor 16 (proportional to $m \cdot n$) when the resolution doubles, the expected number of nonzeros in the solution grows only by a factor 4 (proportional to $m + n$); therefore, the number of variables kept in the support only needs to grow by a factor of 4 to capture the nonzeros of the solution. As a consequence, the cost of vector operations grows proportionally to the square root of the number of variables and similarly for the number of nonzeros in the Schur complement and Cholesky factor (indeed, the Cholesky decomposition becomes relatively sparser, as shown in Table 5.3, under the columns `fill`). However, some operations require the use of the large vector $\mathbf{c}$, like the computation of the full reduced costs; it is not surprising then that, if the number of IPM and conjugate gradient iterations does not grow excessively when changing resolution (as in the case of class 1), the computational time growth is sub-linear.

The numerical results have shown that the method proposed scales better both in terms of computational time and memory requirements, despite being still a prototype code written in Matlab. The scalable results presented for problems with up to 4 billion variables prove the efficacy of the proposed column-generation-inspired IPM for optimal transport problems and potentially for other optimization

problems with a large network structure.

5.6 Conclusion

In this Chapter, a hybrid method that mixes interior point and column generation algorithms was introduced for solving linear programs arising from discrete optimal transport. A careful analysis of the matrices involved allows for an efficient way of solving the Newton linear systems that arise within the interior point phase. Experimental results show that the proposed approach is able to outperform the network simplex solver of Cplex and compete with the highly efficient library LEMON, in particular in terms of memory required. Results with huge scale problems, with up to four billion variables, confirm the robustness of the method.

Further research can potentially improve the performance even more, in particular in relation to:

- The choice of the starting subset of nonzero variables, which could be obtained by some more sophisticated heuristics, based on the initial and final distribution of mass; a better starting point could allow for a smaller size of the support, which would lead to faster computations.
- The switching strategy from iterative method to direct factorization, which could reduce even further the maximum fill-in level of the Cholesky factor and/or the number of overall conjugate gradient iterations.
- The pricing algorithm.

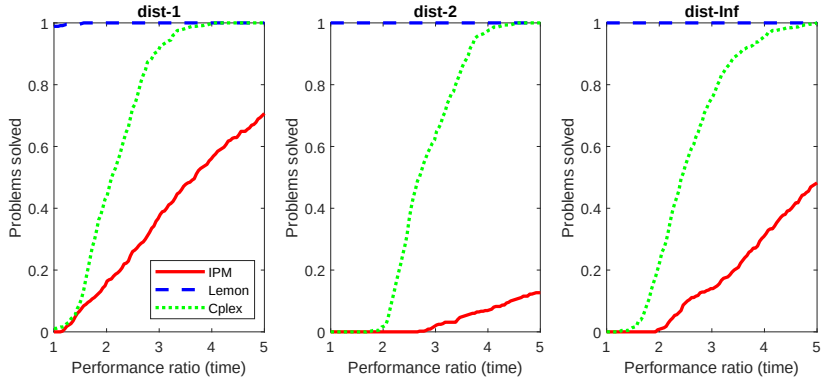
Moreover, the entropic regularized OT problem can be trivially dealt with using an interior point method, just by adding regularization to matrix Θ; however, the structure of the optimal solution changes and the considerations about the linear solver made in this Chapter need to be reconsidered for that specific case. A regularized IPM for optimal transport on graphs is presented in the next Chapter.

Further research is also needed to deal with the convergence properties of the proposed method. Indeed, the combination with column generation prevents the method from keeping the known polynomial complexity of IPMs and the inexact solution of the restricted master problems, which are terminated after merely one IPM iteration, makes it difficult to apply standard results about convergence of column generation techniques. Theoretical results on convergence may also give an explanation to the relatively poorer behaviour of the method when the mass of the initial and final distribution is highly concentrated.

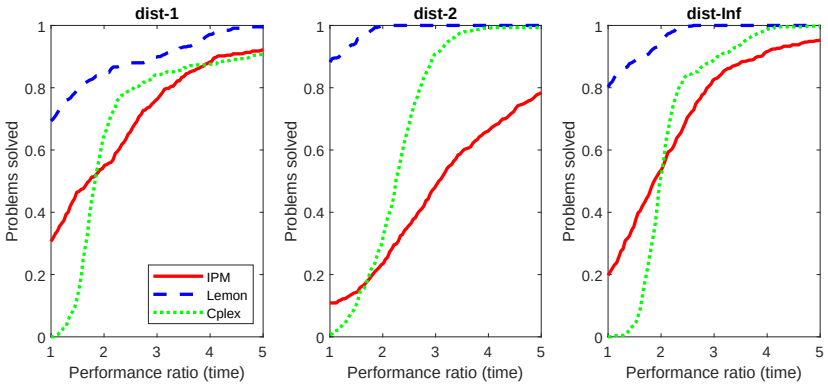
It is also worth mentioning the fact that the proposed IPM has been tested in the case $m = n$, i.e. with the same number of nodes in both groups of the bipartite graph. This is typical in optimal transport applications, but it is not in generic minimum cost flow problems. In particular, when $m = n$, for a given number of variables, the number of constraints is minimized. Other solvers, like the one developed in [29], are able to solve problems with an unbalanced number of nodes (e.g. $n \gg m$), but struggle in the case $m = n$. Future efforts should aim at evaluating the performance of the proposed solver in the case in which the initial and final images have a different number of pixels.

Figure 5.3: Performance profiles of the computational time for the whole DOTmark collection, for the three cost functions, at different resolution.

Resolution 32×32



Resolution 64×64



Resolution 128×128

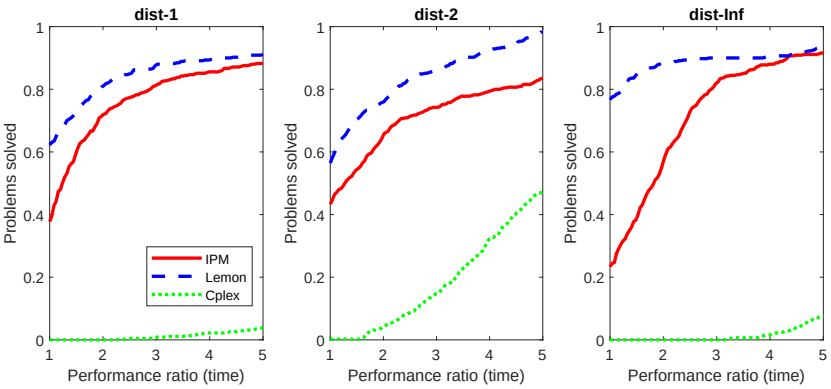
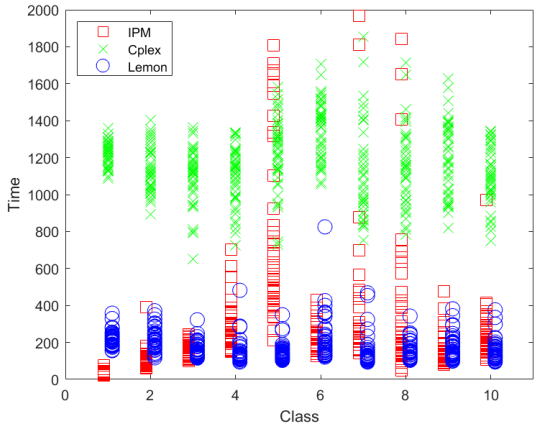
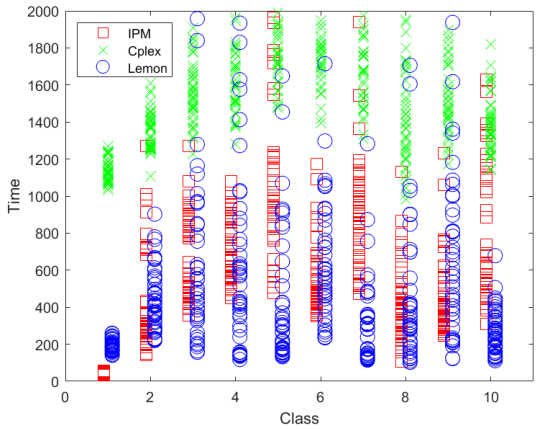


Figure 5.4: Computational time taken by the three solvers for each problem in the DOTmark collection, grouped by class and distance, for resolution 128.

Dist 1



Dist 2



Dist ∞

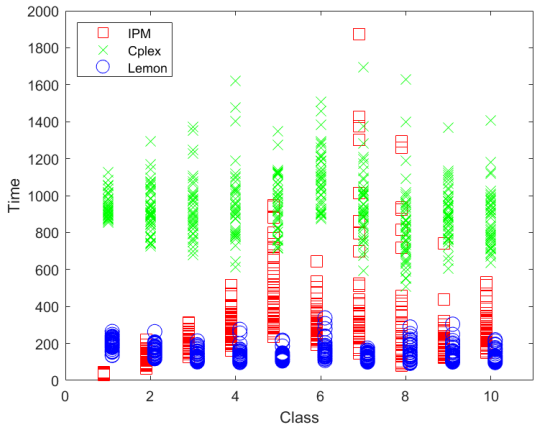
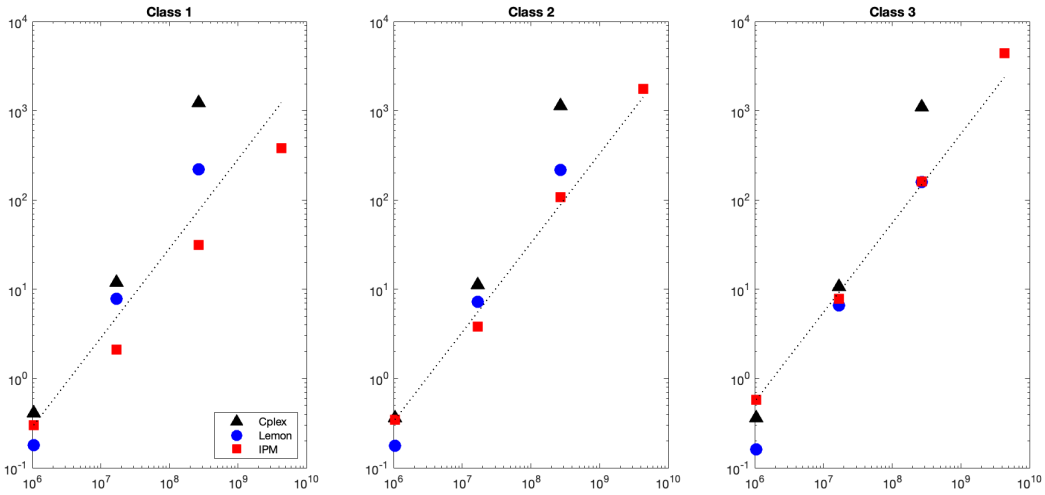
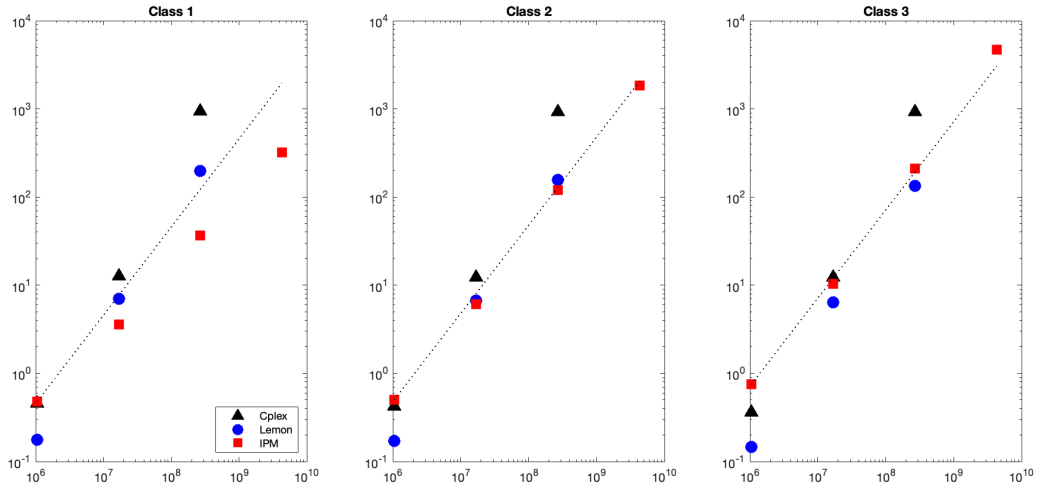


Figure 5.5: Logarithmic plot of the average computational time against the number of variables for classes 1, 2 and 3 and cost functions 1 and ∞.

(a) $1-$distance



(b) $\infty-$distance



Chapter 6

Regularized interior point method for optimal transport on graphs

This Chapter presents the last main contribution of the Thesis and is based on the work [32]. Similarly to the previous Chapter, an interior point method is derived for optimal transport problems; in this case however, the transportation plan is sought over a sparse graph and regularization is introduced to ease the IPM difficulties. The numerical tests, performed on large graphs with up to 50 million edges, show that the proposed IPM outperforms an efficient implementation of the network simplex algorithm.

6.1 Introduction

An interesting formulation of the OT problem presented in the previous Chapter is the optimal transport over *sparse* graphs: in this case, the transport of mass is only possible among a specific subset of connections, which is noticeably smaller than the full list of edges of a fully connected bipartite graph, as it would happen in the standard discrete OT formulation (5.4). The specific formulation of the problem is the following: suppose that $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a connected graph with directed edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ and weights (which correspond to the costs of the OT problem) $\mathbf{c} \in \mathbb{R}_+^{|\mathcal{E}|}$. Define the signed incidence matrix $A \in \{-1, 0, 1\}^{|\mathcal{V}| \times |E|}$ as

$$A_{ve} = \begin{cases} -1, & \text{if } \ e = (v, w) \text{ for some } w \in \mathcal{V} \\ 1, & \text{if } \ e = (w, v) \text{ for some } w \in \mathcal{V} \\ 0, & \text{otherwise.} \end{cases}$$

Notice that in this Chapter the incidence matrix is denoted by A, rather than E as in Section 1.1.1, since it is the constraint matrix of the optimization problem.

We consider the optimal transport problem in the *Beckmann* form [50]

$$\mathcal{W}_1(\boldsymbol{\rho_0}, \boldsymbol{\rho_1}) = \begin{cases} \min_{\mathbf{x} \in \mathbb{R}^{|\mathcal{E}|}} & \sum_{e \in \mathcal{E}} \mathbf{c}_e \mathbf{x}_e \\ s.t. & A\mathbf{x} = \boldsymbol{\rho_1} - \boldsymbol{\rho_0} \ , \\ & \mathbf{x} \geq 0 \end{cases} \tag{6.1}$$

where $\boldsymbol{\rho_0}, \boldsymbol{\rho_1} \in \{\boldsymbol{\rho} \in \mathbb{R}^{|\mathcal{V}|} : \mathbf{e}_{|\mathcal{V}|}^T \boldsymbol{\rho} = 1 \text{ and } \boldsymbol{\rho} \geq 0\}$. In the following we define $|\mathcal{E}| = n$ and $|\mathcal{V}| = m$. OT on graphs has been recently studied in [50, 51] and, in this formulation, it is similar to the more general minimum cost flow problem on networks (or single-commodity network flow) [5], which has also seen extensive use of IPMs, e.g. [29, 53, 104, 107]. Notice however that in these more general problems, there may be a maximum amount of mass that can be moved along any given edge, called the capacity.

Sparse graphs have on average very few edges per node, which can lead to nearly disconnected regions and seriously limit the possible paths where mass can be moved. As a result, finding a solution to the optimal transport problem on a sparse graph requires more sophisticated algorithms and may be more computationally challenging compared to solving the same problem on a denser graph. In particular, first order methods like the network simplex may struggle and move slowly towards optimality, due to the limited number of edges available, while an interior point method manages to identify quickly the subset of basic variables (i.e. the subset of edges with non-zero flow) and converges faster.

In this work we address the efficient solution of the optimal transport problem (6.1) considering the Proximal-Stabilized Interior Point framework (PS-IPM for short), recently introduced and analysed in [31].

As originally observed in [104], when IPMs are used to solve the minimum cost flow problem on networks, the normal form of the related Newton systems is structured as a Laplacian matrix of the graph and the iterates of IPM determine the associate weights of this matrix. In [37], this observation was exploited to solve such Laplacian linear systems (which are, in turn, particular instances of symmetric M-matrices) through the fast specialized solution of $\mathcal{O}(\log m)$ linear systems involving symmetric diagonally dominant matrices [117]. We refer the interested reader to [128] for a survey on fast Laplacian solvers and to [54] for information concerning the distribution of Laplacian's singular values.

6.1.1 Contribution and organization

This Chapter focuses on the efficient solution of large scale OT problems on sparse graphs using a bespoke IPM algorithm able to suitably exploit primal-dual regularization in order to enforce scalability. The organization of the work and its main contributions can be summarized as follows:

- In Section 6.2 we briefly recall the proximal stabilized framework responsible for the primal-dual regularization of the IPMs here considered.

- In Section 6.3, we prove that the normal form of the related Newton system is naturally structured as a shifted Laplacian matrix, characterized by a strict diagonal dominance. Such feature consistently simplifies the factorization of the normal equations and allows the use of standard libraries for the solution of the corresponding linear systems. On the other hand, such factorizations could incur a significant fill-in even when the original graph is sparse, hence limiting the applicability of the proposed approach for the solution of large scale problems.

- In Section 6.4, aiming at overcoming potential scalability issues related to the fill-in mentioned above, we propose to generate IPM search directions using *sparsified* versions of the IPM normal equations which have a natural interpretation in terms of detected optimal transportation plan. Indeed, in the approach here presented, the IPM Newton directions are generated using a *perturbed* normal matrix obtained by taking into account only the variables recognized as "basic" by the IPM procedure. The resulting sparsified linear systems are solved either using a Cholesky factorization (if that displays only negligible fill-in) or using the conjugate gradient method and employing a simple and inexpensive incomplete Cholesky preconditioner. In both these cases either the *complete* or the *incomplete* Cholesky factorization remains very sparse, and this translates into a remarkable efficiency of the proposed method. Moreover, always leveraging on the primal-dual regularization here considered, we are able to interpret the Newton directions generated using sparsified Newton matrices as *inexact* Newton directions. And indeed, relying on the convergence theory developed in [32], we are able to prove that under suitable choice of the sparsification parameters, the above described approach gives rise to polynomially convergent inexact, infeasible, primal-dual regularized interior point method.

- In Section 6.5 we present experimental results which demonstrate the efficiency and robustness of the proposed approach. Extensive numerical experiments, involving very large and sparse graphs coming from public domain random generators as well as from real world applications, show that, for sufficiently large problems, the approach presented in this work consistently outperforms, in terms of computational time, the Lemon network simplex implementation [82], one of the state-of-the-art solvers available for network problems.

This Chapter uses a slightly different notation than the rest of the Thesis: we indicated with $\mathbf{x}_k^j$ the vector $\mathbf{x}$ at iteration k of the (outer) proximal scheme and iteration j of the (inner) interior point algorithm. Individual components of a vector $\mathbf{x}$ are indicated with non-bold characters x_j.

6.2 Computational Framework

6.2.1 Proximal-Stabilized Interior Point Methods

Let us consider the standard primal-dual formulation of an LP:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \ & \mathbf{c}^T\mathbf{x} \\ \text{s.t. } & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned} \qquad\qquad \begin{aligned} \max_{\mathbf{s}\in\mathbb{R}^n,\, \mathbf{y}\in\mathbb{R}^m} \ & \mathbf{b}^T\mathbf{y} \\ \text{s.t. } & \mathbf{c} - A^T\mathbf{y} - \mathbf{s} = 0 \\ & \mathbf{s} \geq 0 \end{aligned} \tag{6.2}$$

where $A \in \mathbb{R}^{m\times n}$ with $m \leq n$ is not required to have full rank. Notice that problem (6.1) is indeed formulated in this way.

We solve this problem using PS-IPM [31], which is a *proximal-stabilized* version of classic interior point method. Broadly speaking, PS-IPM resorts to the Proximal Point Method (PPM) [108] to produce primal-dual regularized forms of problem (6.2). Indeed, given an approximation $(\mathbf{x}_k, \mathbf{y}_k)$ of the solution of such problem, PS-IPM uses interior point methods to produce the next PPM step $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1})$, which, in turn, represents a better approximation of the solution of problem (6.2).

The problem that needs to be solved at every PPM step takes the form

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n, \mathbf{y}\in\mathbb{R}^m} \ & \mathbf{c}^T\mathbf{x} + \frac{\rho}{2}\|\mathbf{x}-\mathbf{x}_k\|^2 + \frac{\delta}{2}\|\mathbf{y}\|^2 \\ \text{s.t. } & A\mathbf{x} + \delta(\mathbf{y}-\mathbf{y}_k) = \mathbf{b} \\ & \mathbf{x} \geq 0, \end{aligned} \qquad \begin{aligned} \max_{\mathbf{x},\, \mathbf{s}\in\mathbb{R}^n,\, \mathbf{y}\in\mathbb{R}^m} \ & \mathbf{y}^T\mathbf{b} - \frac{\rho}{2}\|\mathbf{x}\|^2 - \frac{\delta}{2}\|\mathbf{y}-\mathbf{y}_k\|^2 \\ \text{s.t. } & (\mathbf{c}-\rho\mathbf{x}_k) - A^T\mathbf{y} - \mathbf{s} = 0 \\ & \mathbf{s} \geq 0. \end{aligned} \tag{6.3}$$

It may look as if the solution of problem (6.3) does not converge to the original one (6.2), unless ρ and δ converge to zero. However, unlike an IPM, which generates a sequence of problems that converge to the original one, a PPM works with a sequence of problems that do not have to converge to the original problem, but the corresponding sequence of solutions generated by the algorithm does converge to the optimal solution of the original problem.

Definition 6.1. *Solution of problem* (6.3)
Using standard duality theory, we say that $(\mathbf{x}_k^*, \mathbf{y}_k^*, \mathbf{s}_k^*)$ *is a solution of problem* (6.3) *if the following identities hold*

$$\begin{aligned} A\mathbf{x}_k^* + \delta(\mathbf{y}_k^* - \mathbf{y}_k) - \mathbf{b} &= 0 \\ \rho(\mathbf{x}_k^* - \mathbf{x}_k) - A^T\mathbf{y}_k^* - \mathbf{s}_k^* + \mathbf{c} &= 0 \\ (\mathbf{x}_k^*)^T\mathbf{s}_k^* = 0 \ \textit{and} \ (\mathbf{x}_k^*, \mathbf{s}_k^*) &\geq 0 \end{aligned} \tag{6.4}$$

More in particular, the PS-IPM here considered uses two nested cycles to solve problem (6.2). The outer loop uses an inexact proximal point method, as shown in Algorithm PPM: the current approximate solution $(\mathbf{x}_k, \mathbf{y}_k)$ is used to regularize the LP problem, which is then solved approximately using an IPM to find the next approximate solution $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}) \approx (\mathbf{x}_k^*, \mathbf{y}_k^*)$. And indeed, at the inner loop level, an inexact infeasible interior point method is used to solve the PPM sub-problems, see Algorithm RI-IPM.

Regularization in interior point methods was originally introduced in [112] and extensively used in [6], as a tool to stabilize and improve the linear algebra routines needed for their efficient implementation. In this work and in [31], the regularization is achieved by the application of the PPM at the outer cycle level. To summarize, in the following we use three acronyms: PPM refers to the outer cycle; IPM refers to the inner cycle; PS-IPM refers to the overall procedure, combining PPM and IPM.

Concerning the stopping criteria, we highlight that Algorithm PPM is stopped based on the criterion (6.22). Algorithm RI-IPM instead, is stopped according to the accuracy that is required for the solution of current sub-problem, which is measured using the following *natural residual* of problem (6.3)

Definition 6.2 (Natural Residual)**.**

$$\texttt{res}_k(\mathbf{x},\mathbf{y}) = \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} - \Pi_{\mathbb{R}^n_{\geq 0}\times\mathbb{R}^m}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} - \begin{bmatrix}\rho(\mathbf{x}-\mathbf{x}_k)+\mathbf{c}-A^T\mathbf{y}\\ A\mathbf{x}-\mathbf{b}+\delta(\mathbf{y}-\mathbf{y}_k)\end{bmatrix}\right),$$

where Π *is the projection operator onto the given set.*

Algorithm PPM Outer loop of PS-IPM

Input: $tol > 0$, $\sigma_r \in (0,1)$, $\tau_1 > 0$.

Initialization: Iteration counter $k = 0$; initial point $(\mathbf{x}_0, \mathbf{y}_0)$

1: **while** Stopping Criterion (6.22) False **do**

2: Use Algorithm RI-IPM with starting point $(\mathbf{x}_k^0, \mathbf{y}_k^0) = (\mathbf{x}_k, \mathbf{y}_k)$ to find $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1})$ s.t.

$$\|\texttt{res}_k(\mathbf{x}_{k+1},\mathbf{y}_{k+1})\| < \frac{\sigma_r^k}{\tau_1}\min\{1, \|(\mathbf{x}_{k+1},\mathbf{y}_{k+1}) - (\mathbf{x}_k,\mathbf{y}_k)\|\} \tag{6.5}$$

3:

4: Update the iteration counter: $k = k+1$.

5: **end while**

6.2.2 Interior point method

We now focus on the inner cycle and give a brief description of the IPM used to solve problem (6.3). To this aim, we introduce a particular Lagrangian function which uses a logarithmic barrier to take into account the inequality constraints

$$\begin{aligned}\mathcal{L}_k(\mathbf{x},\mathbf{y}) =& \frac{1}{2}[\mathbf{x}^T,\mathbf{y}^T]\begin{bmatrix}\rho I_n & 0\\ 0 & \delta I_m\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} + [\mathbf{c}^T - \rho\mathbf{x}_k^T, 0]\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\\ &- \mathbf{y}^T(A\mathbf{x} + \delta(\mathbf{y}-\mathbf{y}_k) - \mathbf{b}) - \mu\sum_{i=1}^n \log(x_i).\end{aligned}$$

We write the corresponding KKT conditions

$$\nabla_{\mathbf{x}}\mathcal{L}_k(\mathbf{x},\mathbf{y}) = \rho\mathbf{x} - A^T\mathbf{y} + \mathbf{c} - \rho\mathbf{x}_k - \begin{bmatrix}\frac{\mu}{x_1}\\ \vdots\\ \frac{\mu}{x_n}\end{bmatrix} = 0;$$

$$-\nabla_{\mathbf{y}}\mathcal{L}_k(\mathbf{x},\mathbf{y}) = (A\mathbf{x} + \delta(\mathbf{y}-\mathbf{y}_k) - \mathbf{b}) = 0.$$

Setting $s_i = \frac{\mu}{x_i}$ for $i \in \{1,\dots,n\}$, we consider the following function

Algorithm RI-IPM Regularized Inexact IPM, inner loop of PS-IPM

Input: $0 < \sigma, \bar{\sigma} < 1$ barrier reduction parameters s.t. $\sigma < \bar{\sigma}$;
$C_{\text{inexact}} \in (0,1)$ inexactness parameter s.t. $\gamma_p C_{\text{inexact}} < \sigma$;
Initialization: Iteration counter $j = 0$;
Primal–dual point $(\mathbf{x}_k^0, \mathbf{y}_k^0, \mathbf{s}_k^0) \in \mathcal{N}_k(\bar{\gamma}, \underline{\gamma}, \gamma_p, \gamma_d)$
Compute $\mu_k^0 = {\mathbf{x}_k^0}^T \mathbf{s}_k^0 / n$, $\boldsymbol{\xi}_{d,k}^0$, and $\boldsymbol{\xi}_{p,k}^0$.

1: **while** Stopping criterion (6.5) false **do**
2: Solve the KKT system (6.16) using $[\boldsymbol{\xi}_{p,k}^j, \boldsymbol{\xi}_{d,k}^j, \boldsymbol{\xi}_{\mu_k^j,\sigma}^j]^T$ with $\|\mathbf{r}_k^j\| \leq C_{\text{inexact}} (\mathbf{x}_k^j)^T \mathbf{s}_k$ to find $[\boldsymbol{\Delta}\mathbf{x}_k^j,\ \boldsymbol{\Delta}\mathbf{y}_k^j,\ \boldsymbol{\Delta}\mathbf{s}_k^j]^T$
3: Find α_k^j as the maximum for $\alpha \in [0,1]$ s.t.

$$(\mathbf{x}_k^j(\alpha), \mathbf{y}_k^j(\alpha), \mathbf{s}_k^j(\alpha)) \in \mathcal{N}_k(\bar{\gamma}, \underline{\gamma}, \gamma_p, \gamma_d) \text{ and}$$
$$\mathbf{x}_k^j(\alpha)^T \mathbf{s}_k^j(\alpha) \leq (1 - (1-\bar{\sigma})\alpha) {\mathbf{x}_k^j}^T \mathbf{s}_k^j$$

4: Set $\begin{bmatrix} \mathbf{x}_k^{j+1} \\ \mathbf{y}_k^{j+1} \\ \mathbf{s}_k^{j+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k^j \\ \mathbf{y}_k^j \\ \mathbf{s}_k^j \end{bmatrix} + \begin{bmatrix} \alpha_k^j \boldsymbol{\Delta}\mathbf{x}_k^j \\ \alpha_k^j \boldsymbol{\Delta}\mathbf{y}_k^j \\ \alpha_k^j \boldsymbol{\Delta}\mathbf{s}_k^j \end{bmatrix}$
5: Compute the infeasibilities $\boldsymbol{\xi}_{d,k}^{j+1}$, $\boldsymbol{\xi}_{p,k}^{j+1}$ and barrier parameter $\mu_k^{j+1} = (\mathbf{x}_k^{j+1})^T \mathbf{s}_k^{j+1} / n$
6: Update the iteration counter: $j = j + 1$.
7: **end while**

$$F_k^{\mu,\sigma}(\mathbf{x}, \mathbf{y}, \mathbf{s}) = \begin{bmatrix} \rho(\mathbf{x} - \mathbf{x}_k) - A^T \mathbf{y} - \mathbf{s} + \mathbf{c} \\ A\mathbf{x} + \delta(\mathbf{y} - \mathbf{y}_k) - \mathbf{b} \\ SX\mathbf{e}_n - \sigma\mu\mathbf{e}_n \end{bmatrix}, \tag{6.6}$$

where $\sigma \in (0,1)$ is the barrier reduction parameter, $S = \text{diag}(\mathbf{s})$ and $X = \text{diag}(\mathbf{x})$. A primal–dual interior-point method applied to problem (6.3) relies on the use of Newton iterations to solve a nonlinear problem of the form

$$F_k^{\mu,\sigma}(\mathbf{x}, \mathbf{y}, \mathbf{s}) = 0, \quad \mathbf{x},\ \mathbf{s} > 0.$$

A Newton step for (6.6) from the current iterate $(\mathbf{x}, \mathbf{y}, \mathbf{s})$ is obtained by solving the system

$$\begin{bmatrix} \rho I_n & -A^T & -I_n \\ A & \delta I_m & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}\mathbf{x} \\ \boldsymbol{\Delta}\mathbf{y} \\ \boldsymbol{\Delta}\mathbf{s} \end{bmatrix} = -F_k^{\mu,\sigma}(\mathbf{x}, \mathbf{y}, \mathbf{s}) = \begin{bmatrix} \boldsymbol{\xi}_d \\ \boldsymbol{\xi}_p \\ \boldsymbol{\xi}_{\mu,\sigma} \end{bmatrix}, \tag{6.7}$$

i.e. the following relations hold

$$\rho\boldsymbol{\Delta x} - A^T\boldsymbol{\Delta y} - \boldsymbol{\Delta s} = \boldsymbol{\xi}_d \tag{6.8}$$
$$A\boldsymbol{\Delta x} + \delta\boldsymbol{\Delta y} = \boldsymbol{\xi}_p \tag{6.9}$$
$$S\boldsymbol{\Delta x} + X\boldsymbol{\Delta s} = \boldsymbol{\xi}_{\mu,\sigma}. \tag{6.10}$$

The solution of (6.7) is delivered by the following computational procedure

$$\big(A(\Theta^{-1}+\rho I_n)^{-1}A^T + \delta I_m\big)\boldsymbol{\Delta y} = \boldsymbol{\xi}_p - A(\Theta^{-1}+\rho I_n)^{-1}(X^{-1}\boldsymbol{\xi}_{\mu,\sigma} + \boldsymbol{\xi}_d) \tag{6.11}$$
$$(\Theta^{-1}+\rho I_n)\boldsymbol{\Delta x} = A^T\boldsymbol{\Delta y} + \boldsymbol{\xi}_d + X^{-1}\boldsymbol{\xi}_{\mu,\sigma} \tag{6.12}$$
$$X\boldsymbol{\Delta s} = (\boldsymbol{\xi}_{\mu,\sigma} - S\boldsymbol{\Delta x}), \tag{6.13}$$

where $\Theta = XS^{-1}$. Before continuing let us give basic definitions used in the remainder of this work:

Definition 6.3. *Normal System:*

$$M_{\rho,\delta} = (A(\Theta^{-1}+\rho I_n)^{-1}A^T + \delta I_m). \tag{6.14}$$

Neighbourhood of the infeasible central path:

$$\begin{aligned}\mathcal{N}_k(\bar\gamma,\underline{\gamma},\gamma_p,\gamma_d) =& \{(\mathbf{x},\mathbf{y},\boldsymbol{s}) \in \mathbb{R}^n_{>0}\times\mathbb{R}^m\times\mathbb{R}^n_{>0} \ : \\ & \bar\gamma\mathbf{x}^T\mathbf{s}/n \geq x_i s_i \geq \underline{\gamma}\mathbf{x}^T\mathbf{s}/n \ \ for \ \ i=1,\dots,n; \\ & \mathbf{x}^T\mathbf{s} \geq \gamma_p\|A\mathbf{x}+\delta(\mathbf{y}-\mathbf{y}_k)-\mathbf{b}\|; \\ & \mathbf{x}^T\mathbf{s} \geq \gamma_d\|\rho(\mathbf{x}-\mathbf{x}_k) - A^T\mathbf{y} - \mathbf{s} + \mathbf{c}\|\},\end{aligned}$$

where $\bar\gamma > 1 > \underline{\gamma} > 0$ *and* $(\gamma_p,\gamma_d) > 0$.

Moreover, we consider an *inexact* solution of the linear system (6.11):

Assumption 6.1.

$$M_{\rho,\delta}\boldsymbol{\Delta y} = \bar{\boldsymbol{\xi}}_p + \mathbf{r} \ \ with \ \|\mathbf{r}\| \leq C_{inexact}\,\mathbf{x}^T\mathbf{s}, \tag{6.15}$$

where $C_{inexact} \in (0,1)$ *and we defined*

$$\bar{\boldsymbol{\xi}}_p = \boldsymbol{\xi}_p - A(\Theta^{-1}+\rho I_n)^{-1}(X^{-1}\boldsymbol{\xi}_{\mu,\sigma} + \boldsymbol{\xi}_d).$$

It is important to note that the above Assumption 6.1 is a non-standard requirement in the inexact Newton method [41, 78]. Its particular form is motivated by the use of IPM and the needs of the complexity analysis performed in [32]. It is chosen in agreement with the definition of the infeasible neighbourhood of the central path. Notice also that $\mathbf{r}$ corresponds to the residual of the linear system for a given approximation of the solution $\boldsymbol{\Delta y}$. This means that condition (6.15) can be enforced, for instance, by choosing an appropriate residual tolerance for the Krylov method used.

Using (6.12) and (6.15) in (6.9), we have

$$\|A\mathbf{\Delta x} + \delta\mathbf{\Delta y} - \boldsymbol{\xi}_p\| = \|M_{\rho,\delta}\mathbf{\Delta y} - \bar{\boldsymbol{\xi}}_p\| = \|\mathbf{r}\|,$$

whereas equations (6.8) and (6.10) are satisfied exactly. Therefore the inexact Newton direction which originates from (6.15) results in a direction $[\Delta\mathbf{x},\ \Delta\mathbf{y},\ \Delta\mathbf{s}]^T$ satisfying a Newton equation of the form:

$$\begin{bmatrix} \rho I_n & -A^T & -I_n \\ A & \delta I_m & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \mathbf{\Delta x} \\ \mathbf{\Delta y} \\ \mathbf{\Delta s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\xi}_d \\ \boldsymbol{\xi}_p \\ \boldsymbol{\xi}_{\mu,\sigma} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{r} \\ \mathbf{0} \end{bmatrix}. \tag{6.16}$$

Define

$$\begin{bmatrix} \mathbf{x}_k^j(\alpha) \\ \mathbf{y}_k^j(\alpha) \\ \mathbf{s}_k^j(\alpha) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k^j \\ \mathbf{y}_k^j \\ \mathbf{s}_k^j \end{bmatrix} + \begin{bmatrix} \alpha\mathbf{\Delta x}_k^j \\ \alpha\mathbf{\Delta y}_k^j \\ \alpha\mathbf{\Delta s}_k^j \end{bmatrix}. \tag{6.17}$$

We report in Algorithm RI-IPM a prototype IPM scheme for the solution of problem (6.3). The fundamental steps involved in the algorithm are: computing the Newton direction by solving (6.16) with a level of inexactness that satisfies (6.15); finding the largest stepsize that guarantees to remain inside the neighbourhood and to sufficiently reduce the complementarity products; preparing the quantities to be used in the next iteration.

In [32], inspired by the proofs in [14, 34, 57, 80, 81] and exploiting the results in [8], the following result about polynomial complexity of the proposed regularized inexact IPM algorithm is proven

Theorem 6.1. *Under Assumption 6.1, Algorithm RI-IPM has polynomial complexity, i.e. given $\varepsilon > 0$ there exists $K \in \mathcal{O}\big(n^2 \log(\frac{1}{\varepsilon})\big)$ s.t. $\mu^j \leq \varepsilon$ for all $j \geq K$.*

Proof. See [32] □

6.3 Properties of the regularized normal equations system

In this section we show some properties of matrix $M_{\rho,\delta}$, that are useful for the analysis performed in the next section.

For the original graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we define the *adjacency matrix* $\mathcal{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ by setting $\mathcal{A}_{ij} = 1$ if there exists an edge between nodes i and j and $\mathcal{A}_{ij} = 0$ otherwise. Notice that for an undirected graph $\mathcal{A}$ is symmetric; we assume that there are no self-loops, so that the diagonal of $\mathcal{A}$ is made of zeros. Let us define the *degree matrix* of a graph $\mathcal{D} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ such that $\mathcal{D}$ is diagonal and $\mathcal{D}_{jj}$ is the degree of node j. Notice that $\mathcal{D}_{jj} = (\mathcal{A}\mathbf{e})_j$.

Let us define also the *Laplacian matrix* of a graph as $\mathcal{L} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ by $\mathcal{L} = \mathcal{D} - \mathcal{A}$. An important relation between the Laplacian $\mathcal{L}$ and the node-arc incidence matrix A is that $\mathcal{L} = AA^T$.

Given a diagonal matrix Θ and a parameter ρ, we define the re-weighted graph G_Θ as the graph with the same connectivity of G, in which the weight of every edge j is scaled by a factor $\sqrt{\frac{\Theta_{jj}}{1+\rho\Theta_{jj}}}$. The adjacency matrix of the new graph $\mathcal{A}_\Theta$ has the same sparsity pattern of $\mathcal{A}$, but takes into account the new weight of the edges. The same happens for the degree matrix $\mathcal{D}_\Theta$. The incidence matrix therefore becomes $A_\Theta = A(\Theta^{-1} + \rho I)^{-1/2}$.

The new Laplacian matrix thus reads $\mathcal{L}_\Theta = \mathcal{D}_\Theta - \mathcal{A}_\Theta$ and can be written as

$$\mathcal{L}_\Theta = A_\Theta A_\Theta^T = A(\Theta^{-1} + \rho I)^{-1} A^T.$$

We have just shown that the normal equations matrix can be interpreted as the Laplacian matrix of the original graph, where the edges are weighted according to the diagonal entries of matrix Θ. This result is important because solving linear systems that involve Laplacian matrices is much easier than solving general linear systems. We summarize this result in the following Lemma.

Lemma 6.1. *The matrix $A(\Theta^{-1} + \rho I)^{-1} A^T$ is the Laplacian of a weighted undirected graph and hence, for every $\Theta \in \mathbb{R}^{n\times n}_+$*

$$A(\Theta^{-1} + \rho I)^{-1} A^T = \mathcal{D}_\Theta - \mathcal{A}_\Theta, \tag{6.18}$$

where $\mathcal{D}_\Theta$ and $\mathcal{A}_\Theta$ are the degree and adjacency matrices of the weighted graph.

The next result shows that the normal matrix is strictly diagonally dominant, due to the presence of dual regularization. This property is significant because it assures that the incomplete Cholesky factorization of the normal equations matrix $M_{\rho,\delta}$ can always be computed without the algorithm breaking down (in exact arithmetic), see e.g. [86].

Lemma 6.2. *If $\delta > 0$ the matrix $M_{\rho,\delta}$ is strictly diagonally dominant.*

Proof. From Lemma 6.1, we have that $A(\Theta^{-1} + \rho I_n)^{-1} A^T = \mathcal{D}_\Theta - \mathcal{A}_\Theta$ and hence

$$\sum_{j\neq i} |(M_{\rho,\delta})_{ij}| = \sum_{j\neq i} |-(\mathcal{A}_\Theta)_{ij}| = \sum_{j\neq i} (\mathcal{A}_\Theta)_{ij} < (\mathcal{D}_\Theta)_{ii} + \delta = (M_{\rho,\delta})_{ii}.$$

□

The next two technical results are related to the distribution of eigenvalues of the normal matrix. They are used later to show that the inexactness introduced when sparsifying the normal matrix remains bounded.

Lemma 6.3. $\lambda_{max}(AA^T) \leq 2 \max_{v\in V} deg(v)$.

Proof. The matrix AA^T is the Laplacian matrix of the original graph, hence we can write it as

$$AA^T = \mathcal{D} - \mathcal{A},$$

where $\mathcal{A}$ is the adjacency matrix and $\mathcal{D}$ is the diagonal degree matrix.

From Gershgorin's circle theorem, we know that the spectrum of eigenvalues of AA^T satisfies

$$\lambda(AA^T) \in \bigcup_{j=1}^{m} \mathcal{B}\Big((AA^T)_{jj}, \sum_{i \neq j} |AA^T|_{ji}\Big),$$

where $\mathcal{B}(c, r)$ is the ball centered at c with radius r. In particular, this means that a generic eigenvalue λ satisfies

$$\begin{aligned}
\lambda &\leq \max_{j=1,\ldots,m} \Big((AA^T)_{jj} + \sum_{i \neq j} |AA^T|_{ji}\Big) \\
&\leq \max_{j=1,\ldots,m} \Big(\mathcal{D}_{jj} + \sum_{i \neq j} \mathcal{A}_{ji}\Big) \\
&\leq \max_{v \in V} \; 2\text{deg}(v).
\end{aligned}$$

□

Lemma 6.4. *The eigenvalues λ of matrix $M_{\rho,\delta}$ satisfy*

$$\delta \leq \lambda < \delta + \frac{2}{\rho} \max_{v \in V} deg(v).$$

Proof. Using the Rayleigh quotient for some vector $\mathbf{u}$ and $\mathbf{v} = A^T\mathbf{u}$, the eigenvalues can be written as

$$\lambda = \frac{\mathbf{v}^T(\Theta^{-1} + \rho I_n)^{-1}\mathbf{v}}{\mathbf{v}^T\mathbf{v}} \frac{\mathbf{u}^T AA^T\mathbf{u}}{\mathbf{u}^T\mathbf{u}} + \delta.$$

The lower bound $\lambda \geq \delta$ is trivial; the upper bound follows from Lemma 6.3 and observing that

$$(\rho I_n + \Theta^{-1})^{-1}_{ii} = \frac{\Theta_{ii}}{\rho\Theta_{ii} + 1} = \frac{\rho\Theta_{ii}}{\rho\Theta_{ii} + 1} \frac{1}{\rho} < \frac{1}{\rho}.$$

□

6.4 Sparsification of the reduced matrix

We now propose a technique to reduce the number of nonzeros in the normal equations $M_{\rho,\delta}$, based on the weights of the edges in the re-weighted graph (according to Lemma 6.1). We then show that this sparsification strategy is sound and produces a polynomially convergent interior point algorithm. The results of this section strongly depend on the fact that the intermediate iterates of the proposed IPM belong to the neighbourhood of the central path given in Definition 6.3.

In this section we omit the IPM iteration counter j and we consider all the IPM-related quantities as a function of $\mu \to 0$. As the IPM progresses towards optimality, we expect the following partition of the diagonal matrix Θ contributed

by the barrier term (similarly to (1.10))

$$\mathcal{B} = \{i = 1, \dots, n \text{ s.t. } x_i \to x_i^* > 0,\ s_i \to s_i^* = 0\}$$
$$\mathcal{N} = \{i = 1, \dots, n \text{ s.t. } x_i \to x_i^* = 0,\ s_i \to s_i^* > 0\}.$$

where the optimal solution (x^*, y^*, s^*) was defined in (6.4).

Given the neighbourhood of the central path in Definition 6.3, notice that the following asymptotic estimates hold (as shown in [64]):

$$\begin{aligned} &s_i \in \mathcal{O}(\mu) \text{ and } x_i \in \mathcal{O}(1) \text{ for } i \in \mathcal{B} \\ &x_i \in \mathcal{O}(\mu) \text{ and } s_i \in \mathcal{O}(1) \text{ for } i \in \mathcal{N}. \end{aligned} \tag{6.19}$$

Then, since $x_i^{-1} s_i \approx \mu x_i^{-2}$ when an IPM iterate is sufficiently close to the central path, using (6.19) we obtain

$$\Theta_{ii}^{-1} = x_i^{-1} s_i = \mathcal{O}(\mu) \text{ for } i \in \mathcal{B} \text{ and } \quad \Theta_{ii}^{-1} = x_i^{-1} s_i = \mathcal{O}(\frac{1}{\mu}) \text{ for } i \in \mathcal{N}.$$

As a consequence, we consider the following asymptotic estimates of $(\Theta^{-1} + \rho I_n)^{-1}$

$$(\Theta^{-1} + \rho I_n)_{ii}^{-1} = \begin{cases} \mathcal{O}(\frac{1}{\rho+\mu}) & \text{if } i \in \mathcal{B} \\ \mathcal{O}(\frac{\mu}{1+\rho\mu}) & \text{if } i \in \mathcal{N}. \end{cases}$$

The diagonal entries of Θ give a specific weight to each column of matrix A (or equivalently, give a weight to each edge of the original sparse graph, as shown in Lemma 6.1). The columns for which the corresponding Θ_{ii} is $\mathcal{O}(\mu)$ have a very small impact on the normal matrix, but still contribute to its sparsity pattern. In order to save time and memory when forming (complete or incomplete) Cholesky factorizations, we propose the following sparsification strategy: we introduce a suitable threshold $C_t \in \mathbb{R}_+$ and define

$$\widehat{\Theta}_{ii}^{\dagger} = \begin{cases} (\Theta^{-1} + \rho I_n)_{ii}^{-1} & \text{if } (\Theta^{-1} + \rho I_n)_{ii}^{-1} \geq \frac{C_t \mu}{1+\rho\mu} \\ 0 & \text{if } (\Theta^{-1} + \rho I_n)_{ii}^{-1} < \frac{C_t \mu}{1+\rho\mu}. \end{cases} \tag{6.20}$$

We define the following μ-sparsified version $\widehat{M}_{\rho,\delta}$ of $M_{\rho,\delta}$ as

$$\widehat{M}_{\rho,\delta} = A \widehat{\Theta}^{\dagger} A^T + \delta I_m. \tag{6.21}$$

Notice that this matrix completely ignores some of the columns of A (and some of the edges of the graph). The dual regularization δI guarantees that the resulting matrix is non-singular, irrespective of the level of sparsification chosen. In this Chapter, we consider using inexact Newton directions produced by solving linear systems with matrix $\widehat{M}_{\rho,\delta}$, rather than $M_{\rho,\delta}$.

Remark 6.1. *It is important to note that, in general, the sparsity pattern of the matrix $\widehat{M}_{\rho,\delta}$ depends on the choice of the parameter C_t and on the partitioning $(\mathcal{B}, \mathcal{N})$. Indeed, when μ is sufficiently small, we expect that*

$$\left|\left\{i \in \{1,\dots,n\} \ s.t. \ (\Theta^{-1}+\rho I_n)^{-1}_{ii} \geq \frac{C_t\mu}{1+\rho\mu}\right\}\right| = |\mathcal{B}|.$$

We highlight the fact that a similar idea was proposed in [107], where the Authors build a maximum-weight spanning tree of the original graph, with weights given by the values in Θ, and use it as a preconditioner for CG in the final iterations of an IPM. The preconditioner can be applied very efficiently, exploiting its tree structure. In [53], this idea was extended by considering a subgraph larger than a spanning tree, while still guaranteeing a small fill-in in the corresponding Cholesky factor.

However, notice that these approaches rely on the ability to identify accurately the basic components of the solution; being able to do so is notoriously difficult, especially in the early stage of an IPM, and may spoil the performance of the overall approach. The method proposed in (6.20)-(6.21) considers potentially many more than m variables, especially when far from optimality, and does not depend on the identification of a good basis for the current iteration. Notice also that the method proposed in this chapter employs an inexact direction, while [53]-[107] use their idea to produce a preconditioner.

Let us now show how the algorithm is affected by the use of the proposed sparsified normal matrix. Notice that the results presented below depend strongly on two facts: the optimization problem evolves on a graph and thus the normal matrix is a Laplacian, with very desirable properties; the interior point method employs primal-dual regularization. We start by showing how much the normal matrix deviates from its sparsified counterpart.

Theorem 6.2. *The sparsification strategy in* (6.20) *produces a matrix that satisfies*

$$\|M_{\rho,\delta} - \widehat{M}_{\rho,\delta}\| = \mathcal{O}\Big(\frac{C_t\mu}{1+\rho\mu}\Big).$$

Proof. We have that $M_{\rho,\delta} - \widehat{M}_{\rho,\delta} = AE^\mu A^T$ where E^μ is the diagonal matrix defined as

$$E^\mu_{ii} = (\Theta^{-1}+\rho I_n)^{-1}_{ii} - \widehat{\Theta}^\dagger_{ii} = \begin{cases} 0 & \text{if } (\Theta^{-1}+\rho I_n)^{-1}_{ii} \geq \frac{C_t\mu}{1+\rho\mu} \\ (\Theta^{-1}+\rho I_n)^{-1}_{ii} & \text{if } (\Theta^{-1}+\rho I_n)^{-1}_{ii} < \frac{C_t\mu}{1+\rho\mu} \end{cases}.$$

Hence we have $\lambda_{max}(E^\mu) \leq \frac{C_t\mu}{1+\rho\mu}$. the thesis follows using Lemma 6.3 and observing that

$$\frac{\mathbf{v}^T(M_{\rho,\delta} - \widehat{M}_{\rho,\delta})\mathbf{v}}{\mathbf{v}^T\mathbf{v}} = \frac{\mathbf{v}^T(AE^\mu A^T)\mathbf{v}}{\mathbf{v}^T\mathbf{v}} \leq \lambda_{max}(E^\mu)\lambda_{max}(AA^T).$$

□

We now show that the solution of the sparsified linear system is "close" to the solution of the original one, and the bound depends on μ. This result depends on the spectral distribution that was shown in the previous section.

Theorem 6.3. *For all* $\mathbf{v} \in \mathbb{R}^m$ *we have that*

$$\widehat{M}_{\rho,\delta}^{-1}\mathbf{v} = M_{\rho,\delta}^{-1}\mathbf{v} + \boldsymbol{\psi}$$

where $\|\boldsymbol{\psi}\| \in \mathcal{O}(\frac{C_t\mu}{\delta^2(1+\rho\mu)}\|\mathbf{v}\|)$.

Proof. We can compute explicitly

$$(M_{\rho,\delta}^{-1} - \widehat{M}_{\rho,\delta}^{-1})\mathbf{v} = M_{\rho,\delta}^{-1}\,\widehat{M}_{\rho,\delta}^{-1}(\widehat{M}_{\rho,\delta}\mathbf{v} - M_{\rho,\delta}\mathbf{v})$$

and therefore

$$\|\boldsymbol{\psi}\| \leq \|M_{\rho,\delta}^{-1}\|\|\widehat{M}_{\rho,\delta}^{-1}\|\|\widehat{M}_{\rho,\delta} - M_{\rho,\delta}\|\|\mathbf{v}\|.$$

Using Lemma 6.4 and Theorem 6.2, we have that

$$\|\boldsymbol{\psi}\| \leq \frac{1}{\delta}\frac{1}{\delta}\,\mathcal{O}\Big(\frac{C_t\mu}{1+\rho\mu}\Big)\|\mathbf{v}\| = \mathcal{O}\Big(\frac{C_t\mu}{\delta^2(1+\rho\mu)}\|\mathbf{v}\|\Big).$$

□

The next technical result is useful for the proof of Corollary 6.1.

Lemma 6.5. $\|\bar{\boldsymbol{\xi}}_p^j\|$ *is uniformly bounded, i.e. there exists a constant* $C_p > 0$ *such that for all* $j \in \mathbb{N}$

$$\|\bar{\boldsymbol{\xi}}_p^j\| \leq C_p$$

Proof. For the sake of clarity of notation, we do not include the index j in the proof.

To bound $\|\bar{\boldsymbol{\xi}}_p\|$, consider the following estimate

$$\|\bar{\boldsymbol{\xi}}_p\| \leq \|\boldsymbol{\xi}_p\| + \|A\|\Big(\|(\Theta^{-1}+\rho I)^{-1}X^{-1}\boldsymbol{\xi}_{\mu,\sigma}\| + \|(\Theta^{-1}+\rho I)^{-1}\|\|\boldsymbol{\xi}_d\|\Big).$$

We already know the following estimates

$$\|\boldsymbol{\xi}_p\| \leq \frac{\mu n}{\gamma_p} \leq \frac{\mu^0 n}{\gamma_p}, \qquad \|\boldsymbol{\xi}_d\| \leq \frac{\mu n}{\gamma_d} \leq \frac{\mu^0 n}{\gamma_d}, \qquad \|(\Theta^{-1}+\rho I)^{-1}\| \leq \frac{1}{\rho}.$$

To estimate $\|(\Theta^{-1}+\rho I)^{-1}X^{-1}\boldsymbol{\xi}_{\mu,\sigma}\|$, we proceed as follows

$$\|(\Theta^{-1}+\rho I)^{-1}X^{-1}\boldsymbol{\xi}_{\mu,\sigma}\| = \|(\Theta^{-1}+\rho I)^{-1}(S\mathbf{e} - \sigma\mu X^{-1}\mathbf{e})\| \leq$$
$$\leq \|(\Theta^{-1}+\rho I)^{-1/2}\|\|(\Theta^{-1}+\rho I)^{-1/2}X^{-1/2}S^{1/2}\|\big(\|X^{1/2}S^{1/2}\mathbf{e}\| + \sigma\mu\|X^{-1/2}S^{-1/2}\mathbf{e}\|\big).$$

It is straightforward to prove that

$$\|(\Theta^{-1}+\rho I)^{-1/2}\| \leq \frac{1}{\rho^{1/2}}, \qquad \|(\Theta^{-1}+\rho I)^{-1/2}X^{-1/2}S^{1/2}\| \leq 1.$$

The remaining terms can be bounded using the properties of the neighbourhood

$$\|X^{1/2}S^{1/2}\mathbf{e}\| \leq \sqrt{\mu\bar{\gamma}n}, \quad \sigma\mu\|X^{-1/2}S^{-1/2}\mathbf{e}\| \leq \sigma\sqrt{\frac{\mu n}{\underline{\gamma}}}.$$

Since $\mu \leq \mu^0$, we deduce that $\|\bar{\boldsymbol{\xi}}_p\| \leq C_p$, for some positive constant C_p. □

We have shown in Theorem 6.3 that sparsifying $M_{\rho,\delta}$ can be interpreted as having an inexact solution of the normal equations system. The following result shows that this inexactness satisfies Assumption 6.1, if the constant C_t is chosen appropriately. Therefore, Theorem 6.1 applies.

Corollary 6.1. *If in Algorithm RI-IPM we generate the search directions using $\widehat{M}_{\rho,\delta}^{-1}$ with C_t sufficiently small, i.e. if we compute the search directions using (6.11), (6.12) and (6.13) where $\widehat{M}_{\rho,\delta}$ substitutes $M_{\rho,\delta}$, then Algorithm RI-IPM is convergent and has polynomial complexity.*

Proof. Using Theorem 6.3, we have

$$\widehat{M}_{\rho,\delta}^{-1}\bar{\boldsymbol{\xi}}_p = M_{\rho,\delta}^{-1}\bar{\boldsymbol{\xi}}_p + \boldsymbol{\psi}$$

where $\|\boldsymbol{\psi}\| \leq C_e \frac{C_t\mu}{\delta^2(1+\rho\mu)}\|\bar{\boldsymbol{\xi}}_p\|$ for some constant $C_e > 0$. Hence, recalling (6.15), we have

$$M_{\rho,\delta}\underbrace{\widehat{M}_{\rho,\delta}^{-1}\bar{\boldsymbol{\xi}}_p}_{=\boldsymbol{\Delta}\mathbf{y}} = \bar{\boldsymbol{\xi}}_p + \underbrace{M_{\rho,\delta}\boldsymbol{\psi}}_{=\mathbf{r}}.$$

Observe that, using the previous results, we obtain

$$\|\mathbf{r}\| = \|M_{\rho,\delta}\boldsymbol{\psi}\| \leq \|M_{\rho,\delta}\|C_e\frac{C_t\mu}{\delta^2(1+\rho\mu)}\|\bar{\boldsymbol{\xi}}_p\| \leq \|M_{\rho,\delta}\|C_eC_p\frac{C_t\mu}{\delta^2(1+\rho\mu)}.$$

Using Lemma 6.4, we can say that

$$\|\mathbf{r}\| \leq \Big(\delta + \frac{2}{\rho}\max_{v\in V}\deg(v)\Big)C_eC_p\frac{C_t\mu}{\delta^2(1+\rho\mu)}.$$

In order to satisfy $\|\mathbf{r}\| \leq C_{\text{inexact}}\mathbf{x}^T\mathbf{s}$, we impose that

$$C_t \leq \frac{\delta^2(1+\rho\mu)nC_{\text{inexact}}}{C_eC_p(\delta + 2\max_{v\in V}\deg(v)/\rho)}.$$

With such a value of C_t, Assumption 6.1 is satisfied and thus Theorem 6.1 applies. □

6.5 Numerical Results

The proposed method is compared with Lemon [82], which has been already described in Section 5.5.2.

All the computational tests discussed in this section are performed using a Dell PowerEdge R740 running Scientific Linux 7 with 4× Intel Gold 6234 3.3G, 8C/16T, 10.4GT/s, 24.75M Cache, Turbo, HT (130W) DDR4-2933, with 500GB of memory. The PS-IPM implementation closely follows the one from [31] and is written in Matlab. The software versions used for the numerical experiments are as follows: Matlab R2022a, Lemon 1.3.1 and GCC 4.8.5 as the `C++` compiler.

We stop Algorithm PPM, when

$$\|\mathbf{g} - A^T\mathbf{y} - \mathbf{s}\|_\infty \leq \ R \cdot tol \wedge \|\mathbf{b} - A\mathbf{x}\|_1 \leq \ R \cdot tol \wedge C_{\mathbf{x},\mathbf{s}} \leq \ tol, \tag{6.22}$$

where

$$tol = 10^{-10}, \quad R = \max\{\|A\|_\infty, \|\mathbf{b}\|_1, \|\mathbf{c}\|_1\},$$

and

$$C_{\mathbf{x},\mathbf{s}} = \max_i\{\min\{|x_i s_i|, |x_i|, |s_i|\}\}.$$

Concerning the choice of the parameters in Algorithm PPM, we set $\sigma_r = 0.7$. Moreover, to prevent wasting time on finding excessively accurate solutions in the early PPM sub-problems, we set $\tau_1 = 10^{-4}$, i.e. we use as inexactness criterion for the PPM method

$$\|\mathtt{res}_k(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}))\| < 10^4 \sigma_r^k \min\{1, \|(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}) - (\mathbf{x}_k, \mathbf{y}_k)\|.$$

Indeed, in our computational experience, we have found that driving the IPM solver to a high accuracy in the initial PPM iterations is unnecessary and, usually, leads to a significant deterioration of the overall performance.

Concerning Algorithm RI-IPM, we set as regularization parameters $\rho = 10^{-4}$ and $\delta = 10^{-6}$. Moreover, in order to find the search direction, we employ a widely used predictor-corrector method.

Finally, concerning the test problems, in all the following experiments we generate the load vector $\boldsymbol{\rho_1} - \boldsymbol{\rho_0}$ in (6.1) randomly and such that the sum of its entries is zero (to guarantee feasibility of the optimization problem), with only 10% of them being nonzeros. Moreover, we fix the weight of each edge at 1.

6.5.1 Analysis of the sparsification strategy

In this section, we compare three possible solution strategies inside the PS-IPM: Cholesky factorization (using Matlab's `chol` function) applied to the full normal equations matrix (6.14); Cholesky factorization (always using Matlab's `chol` function) applied to the sparsified matrix (6.21); PCG (using Matlab's `pcg` function) applied to the sparsified matrix (6.21) with incomplete Cholesky preconditioner (computed using Matlab's `ichol` function). More in particular, as sparsification parameter in (6.20) we use $C_t = 0.4$, `'droptol'` $= 10^{-3}$ in `ichol` and `'tol'` $= 10^{-1}\mu$ in `pcg`.

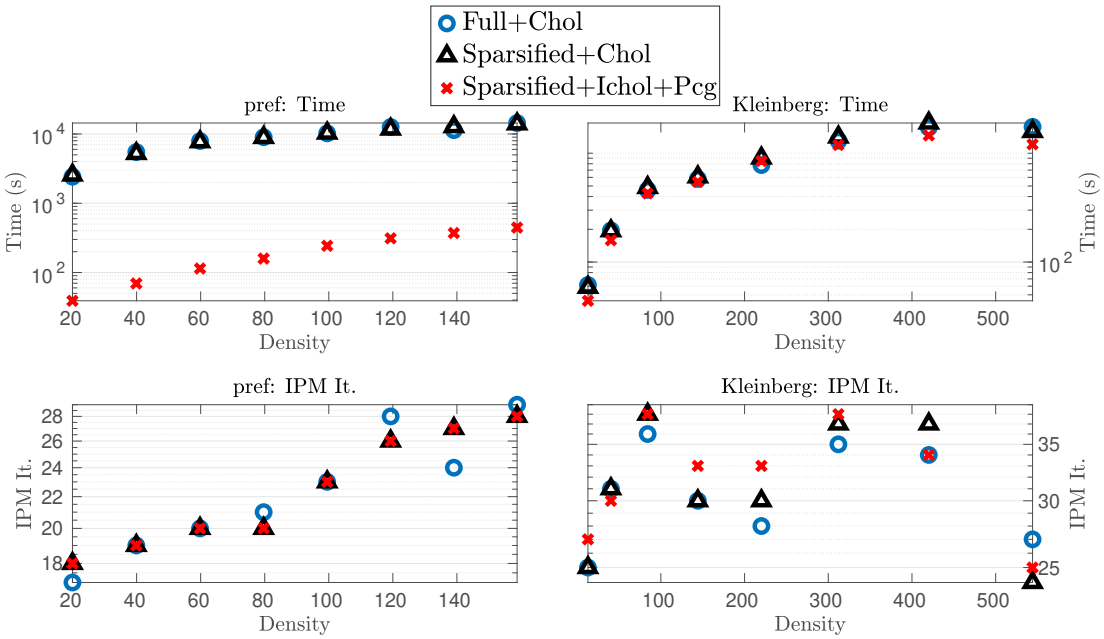
We test the above mentioned solution strategies on various instances generated with the `CONTEST` generator [120]; in particular, we considered the graphs `pref`, `kleinberg`, `smallw` and `erdrey`, with a fixed number of $100,000$ nodes and different densities (i.e. average number of edges per node). Therefore, for these instances, $m = |\mathcal{V}| = 100,000$ and $n = |\mathcal{E}| = m \cdot \text{density}$.

In the upper panels of Figures 6.1 and 6.2 we report the computational time of the three approaches for various values of densities (chosen in relation to the properties of the graph), whereas in the lower panels we report the total number of IPM iterations. From the presented numerical results, it is clear that the sparsification strategy, in conjunction with the iterative solution of the linear

systems, provides a clear advantage over the use of a direct factorization and, as can be expected, the iterative method and the sparsification strategy become more advantageous when the size of the problem (number of edges) increases. On the other hand, it is important to note that the use of the sparsified Newton equations in conjunction with the full Cholesky factorization presents only limited advantages in terms of computational time when compared to the Cholesky factorization of the full Newton normal equation. This is the case because the resulting inexact IPM requires, generally, more iterations to converge (see lower panels of Figures 6.1 and 6.2). Advantages of the proposed approach become clearer when the graphs are denser.

Notice that the computational time for the `kleinberg` graphs depends much less on the approach used than the other three problems. This fact is still not properly understood, but we believe that it is related to the density of the adjacency matrix of the graph and of the corresponding Cholesky factor. The average connectivity of the nodes (i.e. how well a given node is connected to any other node) may also play an important role in determining which approach is the most competitive.

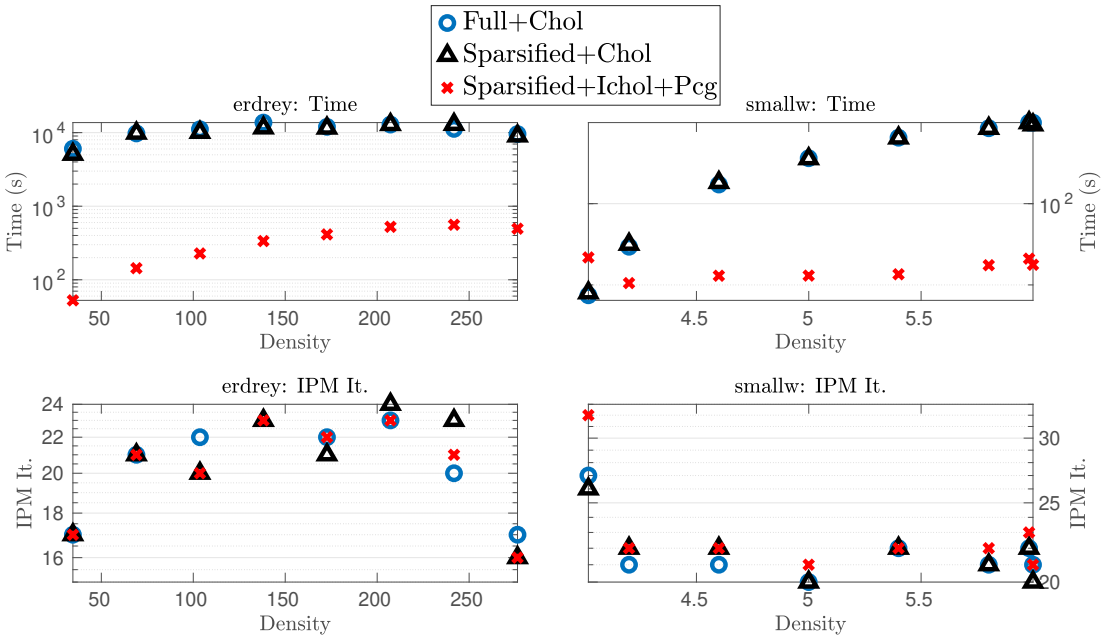
Figure 6.1: Comparison of sparsified and full normal equations approach, using full Cholesky factorization or incomplete Cholesky as preconditioner for PCG, in terms of IPM iterations and computational time for problems `pref` and `Kleinberg`.



6.5.2 Results on randomly generated graphs

In this section, we compare the PS-IPM algorithm, using the sparsified normal equations matrix and the PCG with the network simplex solver of Lemon. For PS-IPM we use the same parameters as proposed in Section 6.5.1. The graphs used in this section come from the generator developed in [127] and already used for OT on graphs in [50]. This generator produces random connected graphs with

Figure 6.2: Comparison of sparsified and full normal equations approach, using full Cholesky factorization or incomplete Cholesky as preconditioner for PCG, in terms of IPM iterations and computational time for problems `erdrey` and `smallw`.



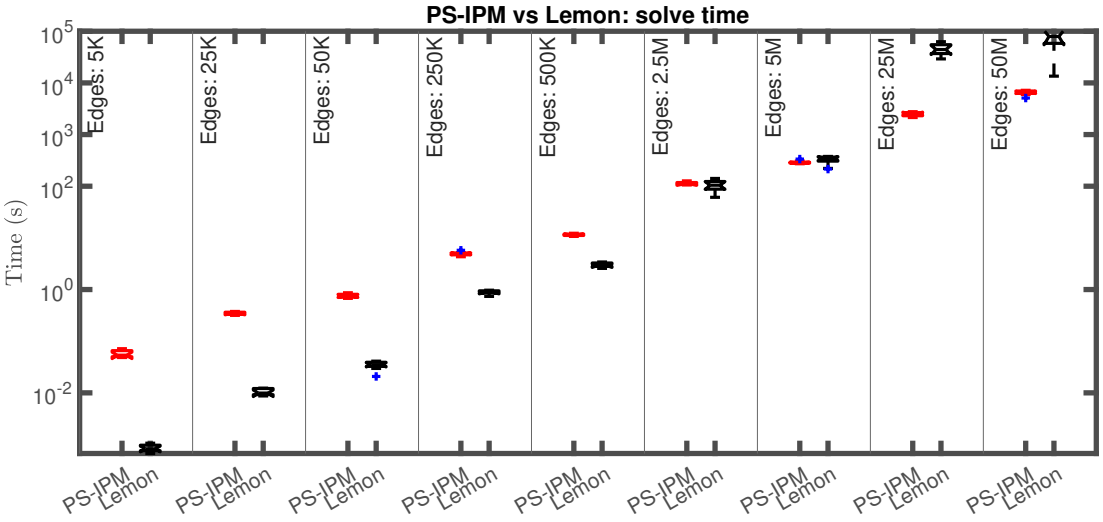
a number of nodes varying from $5,000$ to $10,000,000$ and degrees of each node in the range $[1, 10]$, with an average of 5 edges per node. For each size, 10 graphs and load vectors are generated and tested. These parameters closely resemble the ones used in [50].

Figure 6.3 shows the comparison of the computational time between PS-IPM and Lemon: for each size of the problem (indicated by the total number of edges), we report the summary statistics of the execution times using Matlab's `boxplot`.

For small size problems, Lemon is the clear winner, by two orders of magnitude; however, as the size increases, the performance difference between the two methods reduces and for the largest instance considered, Lemon becomes one order of magnitude slower than PS-IPM. It is also worth observing that the spread of the measured execution times for each size is smaller in the case of PS-IPM than for Lemon, indicating that the method is more robust and less dependent on the specific problem being solved. This is a very desirable property.

Figure 6.4 shows the average computational time against the number of edges (from $5,000$ to $50M$) in a logarithmic scale, the corresponding regression lines and their slopes. From the computational results presented, we can estimate the practical time complexity of both methods. Recall that, in a log-log plot, polynomials of the type x^m appear as straight lines with slope m. Using linear regression, we can estimate that the time taken by Lemon grows with exponent approximately 2.06, while the time taken by PS-IPM grows with exponent approximately 1.28, providing a considerable advantage for large sizes.

Figure 6.3: Box Plots of the computational times of PS-IPM and Lemon, for randomly generated graphs. The red and black intervals show the spread of the measured computational times (red, on the left, is PS-IPM; black, on the right, is Lemon).



6.5.3 Results on SuiteSparse graphs

Results on randomly generated problems do not necessarily represent the ability of an optimization method to tackle problems coming from real world applications. Therefore, in this section, we show the results of applying PS-IPM and Lemon to some sparse graphs from the SuiteSparse matrix collection [39]. The characteristics of the considered graphs are shown in Table 6.1 in terms of number of nodes, edges and average number of edges per node. All the graphs are undirected and connected. Due to the fact that the considered graphs are particularly sparse, in the numerical results presented in this section, we solve the sparsified normal equations using the full Cholesky factorization.

Table 6.1: Details of the graphs from the SuiteSparse matrix collection, ordered by increasing number of edges.

Name	Nodes	Edges	Density
`nc2010`	288,987	1,416,620	4.9
`NACA0015`	1,039,183	6,229,636	6.0
`great-britain-osm`	7,733,822	16,313,034	2.1
`hugetric-00010`	6,592,765	19,771,708	3.0
`hugetric-00020`	7,122,792	21,361,554	3.0
`hugetrace-00010`	12,057,441	36,164,358	3.0
`hugetrace-00020`	16,002,413	47,997,626	3.0
`delaunay-n23`	8,388,608	50,331,568	6.0

Figure 6.4: Logarithmic plot of the computational time for randomly generated graphs. The time taken by Lemon (red circles) grows as (number of edges)$^{2.06}$; the time taken by PS-IPM (blue triangles) grows as (number of edges)$^{1.28}$.

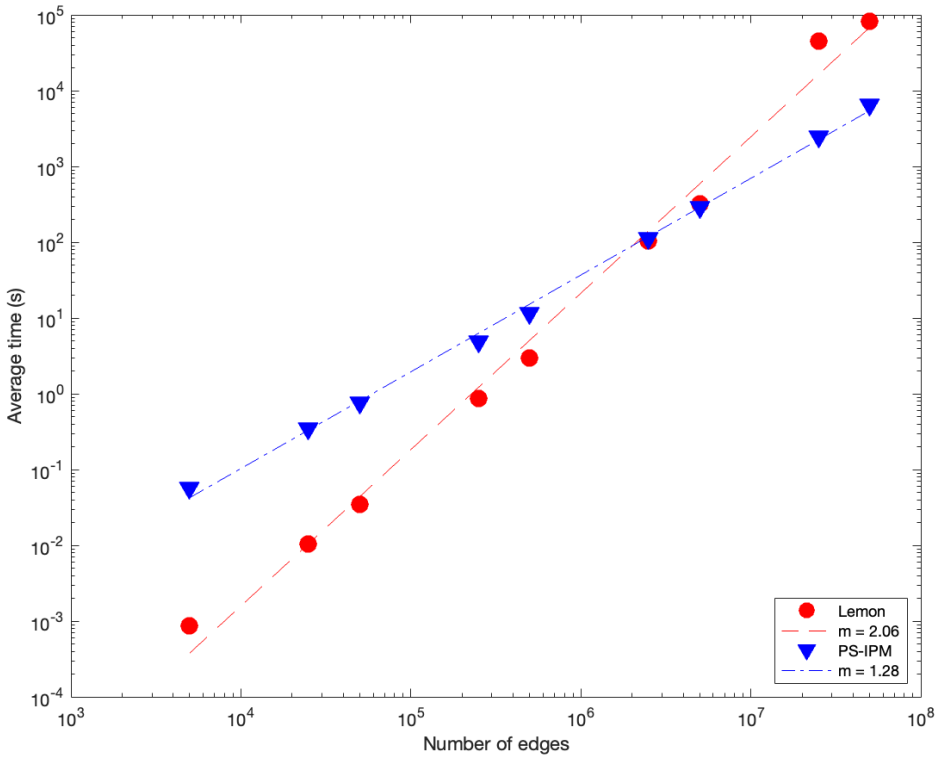
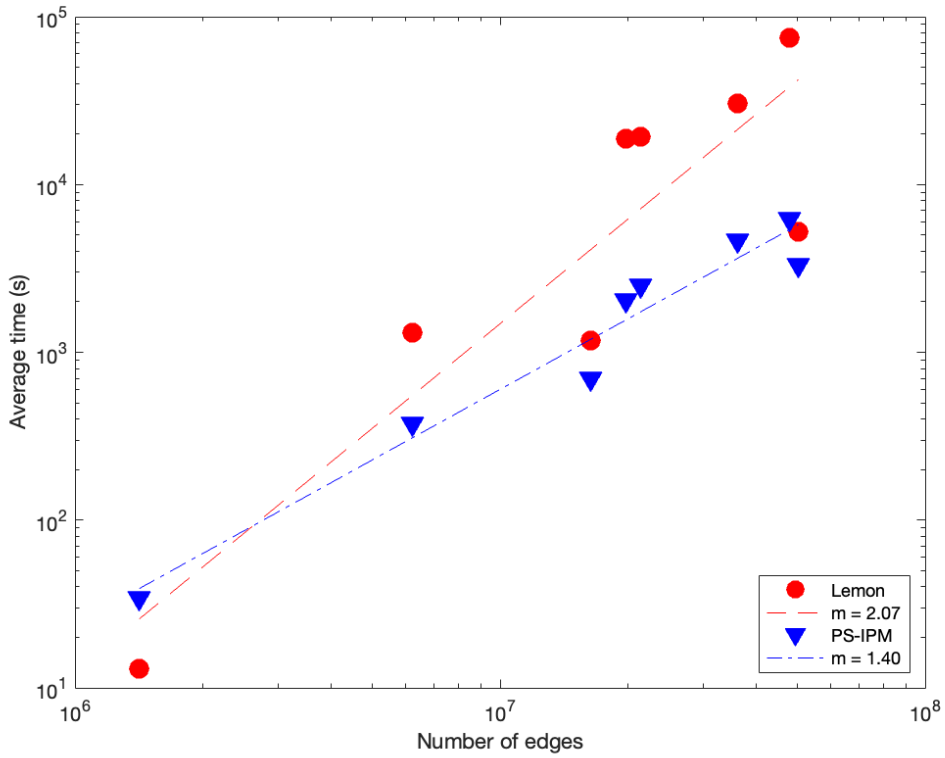


Figure 6.5 shows the computational times for the eight problems considered, using PS-IPM and Lemon. Apart from the problem `nc2010`, which represents a relatively small instance in our dataset, on all the other problems PS-IPM consistently outperforms Lemon in terms of required computational time. In particular, for the problems `hugetric-00010`, `hugetric-00020`, `hugetrace-00010` and `hugetrace-00020`, which reach up to 16 million nodes and 48 million edges, PS-IPM is one order of magnitude faster than Lemon.

Looking at the regression lines and their slopes, we notice that the time taken by Lemon grows with exponent approximately 2.07 while the time taken by PS-IPM grows with exponent approximately 1.40. These values are very close to the ones found previously for randomly generated graphs. The data of Figure 6.5 however has a more erratic behaviour than the times shown in Figure 6.3, because the properties of each graph considered are different and because we are not averaging over 10 different instances of each problem.

Let us highlight also that the time taken by Lemon seems to be more problem dependent, while PS-IPM looks more consistent and robust.

Figure 6.5: Logarithmic plot of the computational time for the SuiteSparse problems. The time taken by Lemon (red circles) grows as (number of edges)$^{2.07}$; the time taken by PS-IPM (blue triangles) grows as (number of edges)$^{1.40}$.



6.6 Conclusion

An efficient computational framework for the solution of Optimal Transport problems on graphs has been presented in this Chapter. Such framework relies on Proximal-Stabilized Interior Point Method and clever sparsification of the normal equations matrix to compute the inexact search directions. The proposed technique is sound and polynomial convergence guarantee has been established for the inner inexact IPM. Extensive numerical experiments show that for large scale problems, a simple prototype Matlab implementation is able to outperform consistently a highly specialized and very efficient `C++` implementation of the network simplex method.

Chapter 7

Conclusions

In this Thesis, we developed efficient interior point solvers for convex linear and quadratic programming problems of large scale. We showed that for sufficiently large problems, second-order methods outperform first-order methods, in terms of computational time and memory requirements, provided that enough attention is put into the numerical linear algebra involved in the solution of the Newton linear systems.

We showed that a strong technique to improve the efficiency of IPMs is the early stopping of the inner Krylov linear solver: this can be done in a standard way, by imposing a variable tolerance on the residual of the linear system, or using some carefully designed indicators that are linked to the convergence indicators of the outer IPM iterations. In the early stage of the IPM, and in particular for the computations of corrector directions, the proposed indicators bring a substantial benefit, reducing the number of Krylov iterations needed and the computational time by up to 70%, with only a small increase in the number of IPM iterations.

We showed that another way of significantly improving the efficiency of IPMs is designing appropriate preconditioners that can keep the spectral interval bounded. To do this, a deep understanding of the operators involved and a careful analysis of the matrices that arise is needed, to take full advantage of the properties of the problem considered. In particular, matrices with Kronecker structure, which appear in many practical applications, can simplify the task of designing proper preconditioners by analysing the repeated smaller blocks, rather than the complete matrix.

As a last major idea to devise efficient solution techniques, we proposed to embed sparsification strategies into the interior point solvers, either by sparsifying the vectors considered (in a column-generation-like style), or by dropping entries in the Newton system (producing an inexact search direction). These techniques are particularly attractive for network optimization problems, where the solution is expected to be sparse but the operators involved can potentially produce dense matrices and/or factorizations.

All these techniques improve the performance of interior point methods significantly. We notice that performance may not always be the most important feature for these algorithms, since many implementations favour robustness, reliability and easiness of use. However, the problems considered in this work often involve time constraints and therefore require the development of efficient specialized

solvers: in tomographic imaging, performance is fundamental to deliver a diagnosis earlier or to free the imaging device quickly; in optimal transport, performance is important because often this problem is solved within another optimization problem (for example because the Wasserstein distance is used as a metric to solve other problems) and therefore there may be many OT instances to be solved in succession.

Future developments

We briefly summarize some possible ideas to improve and extend each of the techniques proposed in this Thesis.

- The proposed early stopping of Krylov solvers is not sufficiently robust to tackle any possible linear or quadratic program; more sophisticated indicators may be needed, as well as some safeguards to avoid the use of directions that are too inexact. This technique can be extended to other optimization problems, including conic, nonlinear and semidefinite programming problems. Given the high computational cost of the linear algebra involved in these type of problems, a dedicated stopping criterion may bring substantial benefit. Moreover, many other optimization algorithms where Krylov solvers are applied could benefit from a similar technique. Theoretical bounds on the maximum inexactness that an IPM can tolerate without losing its favourable convergence properties are also needed to properly justify the use of such techniques.

- Concerning tomographic imaging, extending the proposed regularizer to more than two spatial dimensions and to more than two materials are the obvious next steps. It is unclear whether the current preconditioner can easily generalize to these situations and a more sophisticated preconditioning strategy might be required. Despite there being many ways to approximate the action of the Radon and inverse Radon transforms, it is not clear how they can be used in combination with the diagonal IPM contribution, that changes at each iteration, to produce efficient preconditioners. An accurate analysis of results coming from real world imaging is also needed, to test whether the proposed IPM and preconditioner produce satisfactory results also in the presence of noisy and corrupted measurements.

- Concerning the IPM for discrete optimal transport, for some classes of problems the performance of the proposed approach is not yet satisfactory; in particular, this is the case when the initial and/or final distributions of mass are concentrated in a narrow region. Further research is needed to analyse the impact of such configurations on the linear algebra side and on the proposed column generation technique. Safeguard on the way in which variables are added or removed may be needed, to guarantee good properties of the resulting matrices.

- The proposed sparsified approach for optimal transport on sparse graphs should be compared with the tree preconditioner of [107]. Moreover, rather

than using the sparsified matrix directly to generate inexact directions, the possibility of using it as a preconditioner for the full normal equations matrix should be explored.

- Both approaches developed for OT problems should be extended and tested for single and multi commodity network problems. Given the similar structure of the matrices involved, we expect that good results can be achieved; naturally, a proper technique to deal with the linking constraints in the case of multi commodities has to be employed.

  More in general, any linear program with many more variables than constraints (either in the primal or dual formulation), could potentially benefit significantly from the use of column generation embedded into the IPM. Some preliminary results confirm this fact, but more research is needed to obtain a general purpose approach able to tackle multiple type of problems.

- Finally, when solving a family of related linear programs, many existing algorithms (e.g. active set methods) can exploit the solutions of the previous instances to gain an advantage. It is interesting to understand how much the proposed column generation approach can benefit from reusing the set of columns from a previously solved LP to speed up the initial phase. Potentially, if the set of basic variables does not change substantially, such an approach could significantly improve the performance for subsequent problems.

Bibliography

[1] *CPLEX*. https://www.ibm.com/analytics/cplex-optimizer.

[2] *IFISS*. https://personalpages.manchester.ac.uk/staff/david.silvester/ifiss/.

[3] *LEMON*. https://lemon.cs.elte.hu/trac/lemon.

[4] *MINRES*. https://web.stanford.edu/group/SOL/software/minres/.

[5] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[6] A. Altman and J. Gondzio, *Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization*, Optim. Methods Softw., 11-12 (1999), pp. 275–302.

[7] M. Arioli, *A stopping criterion for the conjugate gradient algorithm in a finite element method framework*, Numer. Math., 97 (2004), pp. 1–24.

[8] P. Armand and J. Benoist, *Uniform boundedness of the inverse of a Jacobian matrix arising in regularized interior-point methods*, Math. Program., 137 (2013), pp. 587–592.

[9] P. Armand, J. Benoist, and J.-P. Dussault, *Local path-following property of inexact interior methods in nonlinear programming*, Comput. Optim. Appl., 52 (2012), pp. 209–238.

[10] F. Aurenhammer, F. Hoffmann, and B. Aronov, *Minkowski-type theorems and least-squares clustering*, Algorithmica, 20 (1998), pp. 61–76.

[11] O. Axelsson and I. Kaporin, *Error norm estimation and stopping criteria in preconditioned conjugate gradient iterations*, Numer. Linear Algebra Appl., 8 (2001), pp. 265–286.

[12] V. Baryamureeba and T. Steihaug, *On the convergence of an inexact primal-dual interior point method for linear programming*, in Large-Scale Scientific Computing, 2006.

[13] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.

[14] S. Bellavia, *Inexact interior-point method*, J. Optim. Theory Appl., 96 (1998), pp. 109–121.

[15] M. Benzi, G. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., (2005), pp. 1–137.

[16] L. Bergamaschi, *A survey of low-rank updates of preconditioners for sequences of symmetric linear systems*, Algorithms, 13 (2020).

[17] L. Bergamaschi, J. Gondzio, A. Martínez, J. W. Pearson, and S. Pougkakiotis, *A new preconditioning approach for an interior point-proximal method of multipliers for linear and convex quadratic programming*, Numer. Linear Algebra Appl., (2021), p. e2361.

[18] L. Bergamaschi, J. Gondzio, M. Venturin, and G. Zilli, *Inexact constraint preconditioners for linear systems arising in interior point methods*, Comput. Optim. Appl., 36 (2007), pp. 137–147.

[19] L. Bergamaschi, J. Gondzio, and G. Zilli, *Preconditioning indefinite systems in interior point methods for optimization*, Comput. Optim. Appl., 28 (2004), pp. 149–171.

[20] D. Bertsekas, *Network optimization: continuous and discrete models*, Athena Scientific, Belmont, USA, 1998.

[21] D. Bertsimas and J. Tsitsiklis, *Introduction to linear optimization*, Athena Scientific, New Hampshire, USA, 1997.

[22] P. Blomgren and T. Chan, *Color TV: total variation methods for restoration of vector-valued images*, IEEE Trans. Image Process., 7 (1998), pp. 304–309.

[23] S. Bocanegra, F. Campos, and A. Oliveira, *Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods*, Comput. Optim. Appl., 36 (2007), pp. 149–164.

[24] J. R. Bunch and B. N. Parlett, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal., 8 (1971), pp. 639–655.

[25] S. Cafieri, M. D'Apuzzo, V. De Simone, and D. di Serafino, *Stopping criteria for inner iterations in inexact potential reduction methods: a computational study*, Comput. Optim. Appl., 36 (2007), pp. 165–193.

[26] S. Cafieri, M. D'Apuzzo, V. De Simone, D. di Serafino, and G. Toraldo, *Convergence analysis of an inexact potential reduction method for convex quadratic programming*, J. Optim. Theory Appl., 135 (2007), pp. 355–366.

[27] J. Castro, *A specialized interior-point algorithm for multicommodity network flows*, SIAM J. Optim., 10 (2000), pp. 852–877.

[28] J. Castro and J. Cuesta, *Quadratic regularizations in an interior point method for primal block-angular problems*, Math. Program., 130 (2011), pp. 415–445.

[29] J. Castro and S. Nasini, *A specialized interior-point algorithm for huge minimum convex cost flows in bipartite networks*, European J. Oper. Res., 290 (2021), pp. 857–869.

[30] J.-S. Chai and K.-C. Toh, *Preconditioning and iterative solution of symmetric indefinite linear systems arising from interior point methods for linear programming*, Comput. Optim. Appl., 36 (2007), pp. 221–247.

[31] S. Cipolla and J. Gondzio, *Proximal stabilized interior point methods and low-frequency-update preconditioning techniques*, J. Optim. Theory Appl., 197 (2023), pp. 1061–1103.

[32] S. Cipolla, J. Gondzio, and F. Zanetti, *A regularized interior point method for sparse optimal transport on graphs*, 2023. arXiv:2307.05186v1[math.OC].

[33] M. Colombo and J. Gondzio, *Further development of multiple centrality correctors for interior point methods*, Comput. Optim. Appl., 41 (2008), pp. 277–305.

[34] J. Cornelis and W. Vanroose, *Convergence analysis of a regularized inexact interior-point method for linear programming problems*, 2021. arxiv.2105.01333[math.OC].

[35] Y. Cui, K. Morikuni, T. Tsuchiya, and K. Hayami, *Implementation of interior-point methods for LP based on Krylov subspace iterative solvers with inner-iteration preconditioning*, Comput. Optim. Appl., 74 (2019), pp. 143–176.

[36] M. Cuturi, *Sinkhorn distances: Lightspeed computation of optimal transport*, in Advances in Neural Information Processing Systems, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds., vol. 26, Curran Associates, Inc., 2013.

[37] S. I. Daitch and D. A. Spielman, *Faster approximate lossy generalized flow via interior point algorithms*, in Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 2008, Association for Computing Machinery, p. 451–460.

[38] M. D'Apuzzo, V. De Simone, and D. di Serafino, *On mutual impact of numerical linear algebra and large-scale optimization with focus on interior point methods*, Comput. Optim. Appl., 45 (2010), pp. 283–310.

[39] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), pp. 1–25.

[40] V. De Simone, D. Di Serafino, J. Gondzio, S. Pougkakiotis, and M. Viola, *Sparse approximations with interior point methods*, SIAM Rev., 64 (2022), pp. 954–988.

[41] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[42] D. di Serafino and D. Orban, *Constraint-preconditioned Krylov solvers for regularized saddle-point systems*, SIAM J. Sci. Comput., 43 (2021), pp. A1001–A1026.

[43] H. S. Dollar, N. Gould, W. Schilders, and A. Wathen, *Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 170–189.

[44] D. Ek and A. Forsgren, *Approximate solution of system of equations arising in interior-point methods for bound-constrained optimization*, Comput. Optim. Appl., 79 (2021), pp. 155–191.

[45] ——, *A structured modified Newton approach for solving systems of nonlinear equations arising in interior-point methods for quadratic programming*, Comput. Optim. Appl., (2023).

[46] A. S. El-Bakry, R. A. Tapia, and Y. Zhang, *A study of indicators for identifying zero variables in interior-point methods*, SIAM Rev., 36 (1994), pp. 45–72.

[47] L. Elden and V. Simoncini, *A numerical solution of a Cauchy problem for an elliptic equation by Krylov subspaces*, Inverse Problems, 25 (2009), p. 065002.

[48] H. C. Elman, A. Ramage, and D. J. Silvester, *IFISS: a Matlab toolbox for modelling incompressible flow*, ACM Trans. Math. Software, 33 (2007), pp. 14–es.

[49] ——, *IFISS: a computational laboratory for investigating incompressible flow problems*, SIAM Rev., 56 (2014), pp. 261–273.

[50] M. Essid and J. Solomon, *Quadratically regularized optimal transport on graphs*, SIAM J. Sci. Comput., 40 (2018), pp. A1961–A1986.

[51] E. Facca and M. Benzi, *Fast iterative solution of the optimal transport problem on graphs*, SIAM J. Sci. Comput., 43 (2021), pp. A2295–A2319.

[52] K. Fountoulakis, J. Gondzio, and P. Zhlobich, *Matrix-free interior point method for compressed sensing problems*, Math. Program. Comput., 6 (2014), pp. 1–31.

[53] A. Frangioni and C. Gentile, *New preconditioners for KKT systems of network flow problems*, SIAM J. Optim., 14 (2004), pp. 894–913.

[54] A. Frangioni and S. Serra Capizzano, *Spectral analysis of (sequences of) graph matrices*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 339–348.

[55] M. A. Freitag and A. Spence, *Convergence of inexact inverse iteration with application to preconditioned iterative solves*, BIT Numerical Mathematics, 47 (2007), pp. 27–44.

[56] R. Freund, F. Jarre, and S. Mizuno, *Convergence of a class of inexact interior point algorithms for linear programs*, Math. Oper. Res., 24 (1999), pp. 50–71.

[57] M. P. Friedlander and D. Orban, *A primal-dual regularized interior-point method for convex quadratic programs*, Math. Program. Comput., 4 (2012), pp. 71–107.

[58] S. Gazzola and M. Sabaté Landman, *Krylov methods for inverse problems: surveying classical, and introducing new, algorithmic approaches*, GAMM-Mitt., 43, p. e202000017.

[59] C. T. L. S. Ghidini, A. R. L. Oliveira, J. Silva, and M. I. Velazco, *Combining a hybrid preconditioner and a optimal adjustment algorithm to accelerate the convergence of interior point methods*, Linear Algebra Appl., 436 (2012), pp. 1267–1284.

[60] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, USA, 1996.

[61] J. Gondzio, *Multiple centrality corrections in a primal-dual method for linear programming*, Comput. Optim. Appl., 6 (1996), pp. 137–156.

[62] ——, *Warm start of the primal-dual method applied in the cutting plane scheme*, Math. Program., 83 (1998), pp. 125–143.

[63] ——, *Interior point methods 25 years later*, European J. Oper. Res., 218 (2012), pp. 587–601.

[64] ——, *Matrix-free interior point method*, Comput. Optim. Appl., 51 (2012), pp. 457–480.

[65] ——, *Convergence analysis of an inexact feasile interior point method for convex quadratic programming*, SIAM J. Optim., 23 (2013), pp. 1510–1527.

[66] J. Gondzio, P. Gonzalez-Brevis, and P. Munari, *New developments in the primal-dual column generation technique*, European J. Oper. Res., 224 (2013), pp. 41–51.

[67] ——, *Large-scale optimization with the primal-dual column generation method*, Math. Program. Comput., 8 (2016), pp. 47–82.

[68] J. Gondzio, M. Lassas, S.-M. Latva-Aijo, S. Siltanen, and F. Zanetti, *Material-separating regularizer for multi-energy x-ray tomography*, Inverse Problems, 38 (2022), p. 025013.

[69] J. Gondzio, S. Pougkakiotis, and J. W. Pearson, *General-purpose preconditioning for regularized interior point methods*, Comput. Optim. Appl., 83 (2022), pp. 727–757.

[70] J. Gondzio and F. N. C. Sobral, *Quasi-Newton approaches to interior point methods for quadratic problems*, Comput. Optim. Appl., 74 (2019), pp. 93–120.

[71] ——, *Polynomial worst-case iteration complexity of quasi-Newton primal-dual interior point algorithms for linear programming*, 2022. arXiv:2208.08771[math.OC].

[72] C. Gottschlich and D. Schuhmacher, *The shortlist method for fast computation of the Earth mover's distance and finding optimal solutions to transportation problems*, PLoS ONE, 9 (2014), p. e110214.

[73] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of NIST, 49 (1952), pp. 409–436.

[74] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE constraints*, in Mathematical modelling: theory and applications, Springer, 2009.

[75] M. E. Hochstanbach and Y. Notay, *Controlling inner iterations in the Jacobi-Davidson method*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 460–477.

[76] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge University Press, New York, USA, 1990.

[77] L. Kantorovich, *On the translocation of masses*, Manag. Sci., 5 (1958), pp. 1–4.

[78] C. T. Kelly, *Iterative methods for linear and nonlinear equations*, SIAM, Philadelphia, USA, 1995.

[79] J. L. Kennington and R. V. Helgason, *Algorithms for network programming*, John Wiley and Sons, New York, 1980.

[80] M. Kojima, N. Megiddo, and S. Mizuno, *A primal—dual infeasible-interior-point algorithm for linear programming*, Math. Program., 61 (1993), pp. 263–280.

[81] J. Korzak, *Convergence analysis of inexact infeasible-interior-point algorithms for solving linear programming problems*, SIAM J. Optim., 11 (2000), pp. 133–148.

[82] P. Kovacs, *Minimum-cost flow algorithms: an experimental evaluation*, Optim. Methods Softw., 30 (2015), pp. 94–127.

[83] H. Ling and K. Okada, *An efficient Earth mover's distance algorithm for robust histogram comparison*, IEEE Trans. Pattern Anal. Mach. Intell., 29 (2007), pp. 840–853.

[84] Z. Lu, R. Monteiro, and J. W. O'Neal, *An iterative solver-based infeasible primal-dual path-following algorithm for convex quadratic programming*, SIAM J. Optim., 17 (2006), pp. 287–310.

[85] M. E. Lubbecke and J. Desrosiers, *Selected topics in column generation*, Oper. Res., 53 (2005), pp. 1007–1023.

[86] T. A. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473–497.

[87] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM J. Optim., 2 (1992), pp. 575–601.

[88] Q. Merigot, *A multiscale approach to optimal transport*, Computer Graphics Forum, 30 (2011), pp. 1583–1592.

[89] S. Mizuno and F. Jarre, *Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation*, Math. Program., 84 (1999), pp. 105–122.

[90] B. Morini and V. Simoncini, *Stability and accuracy of inexact interior point methods for convex quadratic programming*, J. Optim. Theory Appl., 175 (2017), pp. 450–477.

[91] B. Morini, V. Simoncini, and M. Tani, *Spectral estimates for unreduced symmetric KKT systems arising from interior point methods*, Numer. Linear Algebra Appl., 23 (2016), pp. 776–800.

[92] ——, *A comparison of reduced and unreduced KKT systems arising from interior point method*, Comput. Optim. Appl., 68 (2017), pp. 1–27.

[93] J. L. Mueller and S. Siltanen, *Linear and nonlinear inverse problems with practical applications*, SIAM, 2012.

[94] A. Natale and G. Todeschi, *Computation of optimal transport with finite volumes*, ESAIM: M2AN, 55 (2021), pp. 1847–1871.

[95] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.

[96] Y. Notay, *Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl., 9 (2002), pp. 21–44.

[97] A. R. L. Oliveira, *A new class of preconditioners for large-scale linear systems from interior point methods for linear programming*, PhD thesis, Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005, 1997.

[98] A. R. L. Oliveira and D. C. Sorensen, *A new class of preconditioners for large-scale linear systems from interior point methods for linear programming*, Linear Algebra Appl., 394 (2005), pp. 1–24.

[99] J. B. Orlin, *A polynomial time primal network simplex algorithm for minimum cost flows*, Math. Program., 78 (1997), pp. 109–129.

[100] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[101] J. W. Pearson and J. Gondzio, *Fast interior point solution of quadratic programming problems arising from PDE-constrained optimization*, Numer. Math., 137 (2017), pp. 959–999.

[102] J. W. Pearson and A. J. Wathen, *A new approximation of the Schur complement in preconditioners for PDE-constrained optimization*, Numer. Linear Algebra Appl., 19 (2011), pp. 816–829.

[103] G. Peyre and M. Cuturi, *Computational optimal transport: with applications to data science*, Found.Trends in Machine Learning, 11 (2019), pp. 355–607.

[104] L. F. Portugal, M. G. C. Resende, G. Veiga, and J. J. Júdice, *A truncated primal-infeasible dual-feasible network interior point method*, Networks, 35 (2000), pp. 91–108.

[105] S. Pougkakiotis and J. Gondzio, *Dynamic non-diagonal regularization in interior point methods for linear and convex quadratic programming*, J. Optim. Theory Appl., 181 (2019), pp. 905–945.

[106] ——, *An interior point-proximal method of multipliers for convex quadratic programming*, Comput. Optim. Appl., 78 (2021), pp. 307–351.

[107] M. G. Resende and G. Veiga, *An implementation of the dual affine scaling algorithm for minimum-cost flow on bipartite uncapacitated networks*, SIAM J. Optim., 3 (1993), pp. 516–537.

[108] R. T. Rockafellar, *Monotone operators associated with saddle-functions and minimax problems*, in Nonlinear Functional Analysis (Proc. Sympos. Pure Math., Vol. XVIII, Part 1, Chicago, Ill., 1968), Amer. Math. Soc., Providence, R.I., 1970, pp. 241–250.

[109] D. J. Rose, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, Academic Press, 1972, pp. 183–217.

[110] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, Philadelphia, USA, 2003.

[111] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'H, *A deflated version of the conjugate gradient algorithm*, SIAM J. Sci. Comput., 21 (2000), pp. 1909–1926.

[112] M. Saunders and J. A. Tomlin, *Solving regularized linear programs using barrier methods and KKT systems*, Technical Report SOL 96-4, Systems Optimization Laboratory, Dept. of Operations Research, Stanford University, (1996).

[113] B. Schmitzer, *A sparse multiscale algorithm for dense optimal transport*, J. Math. Imaging Vision, 56 (2016), pp. 238–259.

[114] L. Schork and J. Gondzio, *Implementation of an interior point method with basis preconditioning*, Math. Prog. Comp., 12 (2020), pp. 603–635.

[115] J. Schrieber, D. Schuhmacher, and C. Gottschlich, *DOTmark - a benchmark for discrete optimal transport*, IEEE Access, 5 (2017), pp. 271–282.

[116] D. Silvester and V. Simoncini, *An optimal iterative solver for symmetric indefinite systems stemming from mixed approximation*, ACM Trans. Math. Software, 37 (2011), pp. 1–22.

[117] D. A. Spielman and S.-H. Teng, *Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 835–885.

[118] A. Stathopoulos, *Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. Part I: seeking one eigenvalue*, SIAM J. Sci. Comput., 29 (2007), pp. 481–514.

[119] R. E. Tarjan and M. Yannakakis, *Simple linear-time algorithms to test chordality of graphs, test cyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM J. Comput., 13 (1984).

[120] A. Taylor and D. J. Higham, *CONTEST: A controllable test matrix toolbox for Matlab*, ACM Trans. Math. Softw., 35 (2009).

[121] E. Van Den Berg, M. P. Friedlander, G. Hennenfent, F. J. Herrman, R. Saab, and O. Yilmaz, *Sparco: a testing framework for sparse reconstruction*, ACM Trans. Math. Softw., 35 (2009), pp. 1–16.

[122] L. Vandenberghe and M. Andersen, *Chordal graphs and semidefinite optimization*, Found.Trends Optimization, 1 (2014), pp. 241–433.

[123] F. Vanderbeck, *Implementing mixed integer column generation*, in Column Generation, J. Desaulniers, J. Desrosiers, and M. M. Solomon, eds., Springer, US, 2005, pp. 331–358.

[124] F. Vanderbeck and L. A. Wolsey, *Reformulation and decomposition of integer programs*, in 50 Years of Integer Programming 1958-2008, M. Junger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, eds., Springer, Berlin Heidelberg, 2010, pp. 431–502.

[125] R. J. Vanderbei, *Symmetric quasidefinite matrices*, SIAM J. Optim., 5 (1995), pp. 100–113.

[126] M. I. Velazco, A. R. L. Oliveira, and F. F. Campos, *A note on hybrid preconditioners for large-scale normal equations arising from interior-point methods*, Optim. Methods Softw., 25 (2010), pp. 321–332.

[127] F. Viger and M. Latapy, *Efficient and simple generation of random simple connected graphs with prescribed degree sequence*, in International Computing and Combinatorics Conference, Springer, 2005, pp. 440–449.

[128] N. K. Vishnoi, *$Lx = b$ Laplacian solvers and their algorithmic applications*, Found. Trends Theor. Comput. Sci., 8 (2012).

[129] J. Wijesinghe and P. Chen, *Matrix-free interior point methods for point set matching problems*, 2022. arXiv:2202.09763[math.OC].

[130] S. J. Wright, *A path-following interior-point algorithm for linear and quadratic problems*, Ann. Oper. Res., 62 (1996), pp. 103–130.

[131] ——, *Primal-dual interior-point methods*, SIAM, Philadelphia, USA, 1997.

[132] M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM Journal on Algebraic Discrete Methods, 2 (1981).

[133] F. Zanetti and J. Gondzio, *An interior-point-inspired algorithm for linear programs arising in discrete optimal transport*, INFORMS J. Comput., 35 (2023), pp. 1061–1078.

[134] ——, *A new stopping criterion for Krylov solvers applied in interior point methods*, SIAM J. Sci. Comput., 45 (2023), pp. A703–A728.

[135] G. Zhou and K.-C. Toh, *Polynomiality of an inexact infeasible interior point algorithm for semidefinite programming*, Math. Program., 99 (2004), pp. 261–282.