

https://dspace.org/

Minimum L norm solutions to finite dimensional algebraic underdetermined linear systems

Earle, Adam Christopher

2015-02-04
http://hdl.handle.net/10539/16858

Downloaded from a DSpace repository. https://dspace.org/

Minimum ℓ_∞ Norm Solutions To Finite Dimensional Algebraic Underdetermined Linear Systems

by

Adam Christopher Earle

Submitted to the Department of Computational and Applied Mathematics
in partial fulfillment of the requirements for the degree of

Master of Science by Dissertation

at the

University of the Witwatersrand

September 2014

© Adam Christopher Earle, MMXIV. All rights reserved.

The author hereby grants to WITS permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created.

Author .....
Department of Computational and Applied Mathematics
September 10, 2014

Certified by .....
Montaz Ali
Research Professor
Thesis Supervisor

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Minimum ℓ_∞ Norm Solutions To Finite Dimensional Algebraic Underdetermined Linear Systems

by

Adam Christopher Earle

Submitted to the Department of Computational and Applied Mathematics
on September 10, 2014, in partial fulfillment of the
requirements for the degree of
Master of Science by Dissertation

Abstract

A new method of solution to the problem of finding the minimum ℓ_∞ norm solution to an algebraic underdetermined linear system is developed. The new method is a geometrically clear, primal method. Like some existing methods, the new method can be logically divided into two parts. In the first part of the solution process a solution to the linear system is generated that has at least $(n - m + 1)$ components equal in absolute value and equal to the ℓ_∞ norm of that particular solution. A number of new techniques are suggested in this part of the algorithm, including an iterative ascent procedure.

In the second part of the solution process, the particular solution obtained in the first part is iteratively improved. We have developed a number of new techniques here corresponding to both single and multi-element exchange procedures. Central to the new method is the development of descent criteria for a direction vector, and the stopping condition.

A thorough numerical investigation of the new techniques proposed, in both the first and second part of the solution process, is carried out using 5000 problem instances with systems up to $(m, n) = (500, 505)$, with some strong conclusions being drawn. The performance of our algorithm is also compared with two well-known methods from the literature. Our method is shown to be much superior to these well known-methods with respect to both the number of iterations and the wall-clock time required. The iterative computational complexity of the new method also compares favourably with most well-known methods.

A geometric heuristic is developed for initial active constraint set selection and a number of theoretical results are given. The heuristic stands to be much more valuable if the results presented herein can be generalised.

Thesis Supervisor: Montaz Ali
Title: Research Professor

Acknowledgments

I would like to express my sincerest gratitude to my supervisor Prof. Montaz Ali who has given graciously and abundantly of himself. His guidance has left its indelible mark upon this thesis. His unfaltering attention to detail, immaculate professionalism and personal sacrifice have taught me lessons beyond mathematics.

Special acknowledgement must go to Mr Dario Fanucchi who has been a boundless source of creative inspiration. It is through his example that I strive to see more.

Finally I wish to thank my parents. This thesis is dedicated to you. I am thankful every day for your profound guidance and wisdom.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

1 Introduction **17**

1.1 Mathematical Primer 20

1.2 Problem Statement 22

1.2.1 Underdetermined 22

1.2.2 Overdetermined 23

2 Literature Review **25**

2.1 Underdetermined 25

2.1.1 The Shim-Yoon Method 26

2.1.2 Cadzow's Method 29

2.1.3 Linear Programming Formulation 35

2.2 Overdetermined 38

2.2.1 Polya's Method 40

2.2.2 Stiefel's Method 42

2.2.3 Linear Programming Formulation 44

2.2.4 Modern Path-Following Methods 47

3 Primal Path-Following (PPF) Method For Underdetermined Systems 53

3.1 Part 1 of the PPF Method 57

3.1.1 Initial Solution 57

3.1.2 Calculating the Direction Vector 58

3.1.2.1 The Standard Method 58

3.1.2.2 The Null Space Method 59

3.1.3 Calculating the Step Length Parameter 60

3.2 Part 2 of the PPF Method 62

3.2.1 Expedient Computation of Indicators 65

3.3 Ascent Heuristics 67

3.3.1 Moore-Penrose Pseudo-Inverse 67

3.3.2 Iterative Ascent 68

3.4 Exchange Heuristics 70

3.4.1 Single Element Exchange 70

3.4.2 Multiple Element Exchange 71

3.5 The PPF Algorithm 75

3.6 Computational Complexity of the PPF Method 78

3.6.1 Part 1 79

3.6.1.1 Iterative Ascent (IA) 79

3.6.1.2 Pseudo-Inverse (PI) 81

3.6.2 Part 2 82

3.6.2.1 Single Element Exchange 83

3.6.2.2 Multiple Element Exchange 84

3.7 Geometric Heuristic for Initial Active Constraint Selection 88

3.7.1 Hyperplane bounding 89

4 Numerical Results **105**

4.1 Implementation Details and Test Problems for the PPF Method . . . 105

4.1.1 Ascent Heuristics With Single Element Exchange 108

4.1.2 Exchange Heuristics With Multiple Element Exchange 114

4.2 Numerical Comparison of Current Best Methods 122

4.3 Discussion 127

5 Conclusion **131**

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

3-1 Contour plot of 2-dimensional solution space 55

3-2 Elementwise propagation along $\boldsymbol{d}^k$ [1] 61

3-3 Regions in $\mathbf{R}^2$ 90

3-4 Regions in the solution space in $\mathbf{R}^2$ 94

4-1 Natural logarithm of the number of iterations in Part 2 - Comp(IA-small, Big, NA) 109

4-2 Natural logarithm of the number of iterations in Part 2 - Cap(IA-small, Big, NA) 109

4-3 Natural logarithm of the number of iterations in Part 2 - Comp(IA-big, Big, NA) 110

4-4 Natural logarithm of the number of iterations in Part 2 - Cap(IA-big, Big, NA) 110

4-5 Natural logarithm of the number of iterations in Part 2 - Comp(PI, Big, NA) 110

4-6 Natural logarithm of the number of iterations in Part 2 - Cap(PI, Big, NA) 110

4-7 Natural logarithm of the number of iterations in Part 2 - Comp(IA-small, Small, NA) . . . 112

4-8 Natural logarithm of the number of iterations in Part 2 - Cap(IA-small, Small, NA) . . . 112

4-9 Natural logarithm of the number of iterations in Part 2 - Comp(IA-big, Small, NA) . . . 112

4-10 Natural logarithm of the number of iterations in Part 2 - Cap(IA-big, Small, NA) . . . 112

4-11 Natural logarithm of the number of iterations in Part 2 - Comp(PI, Small, NA) . . . 113

4-12 Natural logarithm of the number of iterations in Part 2 - Cap(PI, Small, NA) . . . 113

4-13 Natural logarithm of the total number of iterations - Comp(IA-small, IntOnes, PI) . . . 115

4-14 Natural logarithm of the total number of iterations - Cap(IA-small, IntOnes, PI) . . . 115

4-15 Natural logarithm of the total number of iterations - Comp(IA-small, IntOnes, IA-small) . . . 116

4-16 Natural logarithm of the total number of iterations - Cap(IA-small, IntOnes, IA-small) . . . 116

4-17 Natural logarithm of the total number of iterations - Comp(IA-small, IntOnes, IA-old) . . . 116

4-18 Natural logarithm of the total number of iterations - Cap(IA-small, IntOnes, IA-old) . . . 116

4-19 Natural logarithm of the total number of iterations - Comp(IA-small, IntOnes, IA-big) . . . 117

4-20 Natural logarithm of the total number of iterations - Cap(IA-small, IntOnes, IA-big) . . . 117

4-21 Natural logarithm of the total number of iterations - Comp(IA-small, IntEqual, PI) . . . 118

4-22 Natural logarithm of the total number of iterations - Cap(IA-small, IntEqual, PI) . . . 118

4-23 Natural logarithm of the total number of iterations - Comp(IA-small, IntEqual, IA-small) . . . 118

4-24 Natural logarithm of the total number of iterations - Cap(IA-small, IntEqual, IA-small) . . . 118

4-25 Natural logarithm of the total number of iterations - Comp(IA-small, IntEqual, IA-old) . . . 119

4-26 Natural logarithm of the total number of iterations - Cap(IA-small, IntEqual, IA-old) . . . 119

4-27 Natural logarithm of the total number of iterations in - Comp(IA-small, IntEqual, IA-big) . . . 119

4-28 Natural logarithm of the total number of iterations in - Cap(IA-small, IntEqual, IA-big) . . . 119

4-29 Natural logarithm of the total number of iterations - Cap(IA-small, IntOnes, PI) . . . 121

4-30 Natural logarithm of the total number of iterations - Cap(IA-small, IntEqual, PI) . . . 121

4-31 Natural logarithm of the total number of iterations - Cap(IA-small, Small, NA) 121

4-32 Natural logarithm of the total number of iterations - Comp - Cadzow's Method 123

4-33 Natural logarithm of the total number of iterations - Cap - Cadzow's Method 123

4-34 Natural logarithm of the total number of iterations - Comp - LP Formulation 123

4-35 Natural logarithm of the total number of iterations - Cap - LP Formulation 123

4-36 Natural logarithm of the total number of iterations - Comp(PI, Small, NA) 124

4-37 Natural logarithm of the total number of iterations - Cap(PI, Small, NA) 124

List of Tables

3.1 Summary of complexity analysis of the PPF method 87

4.1 Systems used in numerical testing 107

4.2 Comparative results for a representative subset of the problem ensemble (Complete - run to convergence) 125

4.3 Comparative results for a representative subset of the problem ensemble (Capped - run with iteration limit) 126

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

It goes without saying that the solutions to systems of linear equations are of paramount importance in almost every field of applied mathematics, and indeed in many other fields as well. Elementary linear algebra gives the conditions under which a given linear system has a solution or not and, if the system does permit a solution, whether the solution is unique.

A linear system of equations is called consistent if it has a solution and inconsistent otherwise. In terms of the well known Rouchè-Capelli theorem a system of linear equations, $A\boldsymbol{x} = \boldsymbol{b}$, has a solution if and only if the rank of A is equal to the rank of the augmented system $[A|\boldsymbol{b}]$. In the event that the solution is unique, there exist many well established methods of solution with which we shall not concern ourselves here.

In practice however it is common to encounter systems that do not admit a unique solution, rather the system is consistent and an infinite number of solutions exist, or the system is inconsistent and no solution exists. In the former case the system is called *underdetermined* and in the latter the system is called *overdetermined*.

When the system is underdetermined practitioners then seek to pick the solution with a specific set of properties that is best suited to their needs. Minimum norm solutions

are often chosen for various practical reasons - these being related to the choice of norm. When the system is overdetermined practitioners then seeks to determine an approximate solution that is in some sense best. The 'best' solution is often chosen to be that which minimizes the error in some norm. The choice of norm is again application dependent.

In this thesis we shall restrict our attention to ℓ_p norms. These are by far the most widely utilized norms in this context; little in the way of generalization is thus lost by this restriction. The choice of p is again application dependent. By far the most common choices for p in finding minimum ℓ_p norm solutions to both overdetermined and underdetermined linear systems are $p = \{1, 2, \infty\}$.

In the overdetermined setting we present applications for various choices of p in the context of linear approximation theory. Therein the choice $p = 2$ corresponds to the well known least squares solution, a closed form expression for which is given by the Moore-Penrose pseudo-inverse. A proprietary dispute between Gauss and Legendre over the invention of the method of least squares places its discovery around the early 19^{th} century. In the underdetermined setting the choice $p = 2$ picks out the so called minimum energy solution. Again the existence of a closed form expression for its determination has made it the popular choice - if only for its ease of computation.

The choice $p = 1$, in the overdetermined setting, provides a linear fit to a discrete set of data that is insensitive to outliers, and may therefore be preferable to the least squares fit for certain data types. In practice, statisticians may choose $1 < p < 2$ as a way of fine-tuning a fit in terms of its sensitivity to outliers. In the underdetermined setting, interest in the minimum ℓ_1 norm solution has increased dramatically in the last decade following foundational results by Emmanuel Candes and Terence Tao [8] in 2004 which gave the conditions under which the ℓ_1 norm would, with high probability, produce the sparsest solution. There is widespread applicability of this result to the field of signal processing.

We pay special attention to the selection $p = \infty$.

The problem resulting from the choice $p = \infty$ in the overdetermined setting, is also known as that of obtaining the linear Chebyshev approximation to a discrete set of data. It is a highly practical problem and is thus suitably steeped in a rich and illustrious history. The problem seems to date back to the mid 18^{th} century and in fact predates the introduction of the well known least squares problem [18]. The best linear fit to a discrete set of data in terms of the Chebyshev norm is the fit that will minimize the maximum error. It is noted in [7] that such a choice is obviously best when the cost of making any single large error is of primary importance.

A key observation upon which many existing solution methods rely is that, at optimum, the residual vector has a well defined number of components equal in absolute value and maximal. If this critical set of elements can be found, the problem may be considered all but solved. A 'brute force' method is explicitly outlined in [7] in which every possible subset of elements is checked. A method due to Polya [9] seeks to establish this critical set by iteratively solving the minimization problem for increasing values of p. An exchange algorithm was presented in 1959 by Stiefel [17] in which elements are exchanged between an initial subset in a systematic fashion such that the value of the ℓ_∞ norm increases on each iteration. A linear programming (LP) formulation of the primal problem is intractable owing to the number of slack variables required. A competitive LP formulation of the dual problem was presented in 1975 by Abdelmalek [1]. Finally, a primal path-following exchange procedure was put forward by Cadzow in 2002 [7] in which the ℓ_∞ norm decreases on each iteration in contrast with the method due to Stiefel.

In the underdetermined case the choice $p = \infty$ finds multitudinous applications. Minimization of the ℓ_∞ norm is equivalent to minimizing the maximum absolute value of any component of the solution vector to the linear system. As such the ℓ_∞ norm finds application whenever one seeks a solution such that the load on each node of a system, corresponding to each component of the solution vector, does not exceed its limitation.

A similar observation, as in the overdetermined case, is key to many of the existing

solution methods: namely that the solution vector, at optimum, contains a well defined number of components equal in absolute value and maximal. In 1998 Shim and Yoon [16] presented a primal method based on geometric consideration of the problem. A foundational method was presented by Cadzow in 1974 [6] in which an augmented dual problem is solved. Again the primal LP formulation is inaccessible owing to the number of slack variables required. Abdelmalek in 1977 [2] presented a competitive LP formulation of the dual problem following his work in the overdetermined setting.

As of yet no primal path-following algorithm exists in the underdetermined setting. It will be a primary goal of this thesis to develop such a method. Moreover, many of the ideas developed herein shall be immediately transferable to the overdetermined setting; as many ideas from the overdetermined setting have inspired developments in the new method.

1.1 Mathematical Primer

Throughout this thesis we will consider a system of linear equations over the field of real numbers, explicitly then we consider systems of the form $A\boldsymbol{x} = \boldsymbol{b}$ where $A \in \mathbf{R}^{m\times n}, \boldsymbol{x} \in \mathbf{R}^{n\times 1}$ and $\boldsymbol{b} \in \mathbf{R}^{m\times 1}$.

By virtue of the Rouchè-Capelli theorem, every linear system of equations may be uniquely classified as underdetermined, uniquely determined or overdetermined. The Rouchè-Capelli theorem gives the number of degree of freedom of the solution space in terms of the rank of A: $\text{DOF} = n - \text{Rank}\,(A)$. Assuming that the matrix A is of full rank, we have that the system is inconsistent and permits no solutions if $m > n$; is consistent and permits a unique solution if $m = n$; and is consistent and permits an infinite number of solutions if $m < n$.

We now introduce some salient properties of the finite dimensional ℓ_p norm. The

finite dimensional ℓ_p norm is defined as follows:

$$||\boldsymbol{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}.$$

While the function is well defined for $0 < p \leq \infty$ it is only a norm for $1 \leq p \leq \infty$ since the function is not subadditive in the range $0 < p < 1$.

Importantly, the ℓ_p norm is a monotonically decreasing function of p, i.e. for $q > p$ we have $||\boldsymbol{x}||_p \geq ||\boldsymbol{x}||_q$. This is intuitively obvious from consideration of the level curves of the ℓ_p norm for various values of p. Moreover, reverse bounds may be establish for arbitrary ℓ_p norms as follows: suppose $p > q$ then we have $||\boldsymbol{x}||_p \leq ||\boldsymbol{x}||_q \leq n^{\frac{1}{q}-\frac{1}{p}} ||\boldsymbol{x}||_p$, where $\boldsymbol{x} \in \mathbf{R}^n$. In terms of practical considerations, these bounds show that norms are more weakly bounded as the dimension of the vector space increases.

The ℓ_p norm is in fact also a strictly convex function for $1 < p < \infty$, and weakly convex for $p = \{1, \infty\}$. This important, albeit well known, fact will greatly simplify the optimization as, of course, any local minimum will also be a global minimum.

The ℓ_p norm is also a smooth function for $1 < p < \infty$. Alternatively put, it is a non-smooth function only for $p = \{1, \infty\}$, where it is piecewise differentiable. This fact is again intuitive upon consideration of the level curves. This is of great practical importance as the ℓ_1 and ℓ_∞ norms are not amenable to solution by methods requiring derivative information.

With an eye to assisting in the conceptual development of the algorithms to come, we take a moment to consider the geometry of the level curves of the ℓ_p norm for three specific values of p. For $p = 1$ the level curves are the n-dimensional simplexes. A vertex of such a level curve lies along a coordinate axis and corresponds to a location at which $n - 1$ elements are zero. An edge joining two neighbouring vertices corresponds to a line along which $n - 2$ elements are zero etc. For $p = 2$ the ℓ_p norm is a smooth function and the level curves are the n-balls, these are the level curves of the standard Euclidean norm.

For $p = \infty$ the level curves are the n-dimensional hypercubes. To see this more readily, notice that $||\boldsymbol{x}||_\infty = \max_i \{|x_i|\}$. A vertex of the level curve corresponds to a location at which all n components are equal in absolute value. An edge joining two neighbouring vertices corresponds to an edge along which $n-1$ components are equal in absolute value etc. This geometry shall be exploited to great effect in the sections to come.

1.2 Problem Statement

Throughout this thesis we make the assumption that the matrix $A \in \mathbf{R}^{m\times n}$ is of full rank. This is done primarily to simplify the exposition of key ideas; indeed in many of the methods that will be presented only minor alternations are required to deal with the degeneracy imposed by relaxing the full rank assumption.

Explicitly then, invoking the Rouché-Capelli theorem, we shall assume that in the underdetermined case $m < n$ and the system is consistent, and in the overdetermined case $m > n$ and the system is inconsistent.

1.2.1 Underdetermined

In this setting we consider in the general case the following problem:

$$\min \ ||\boldsymbol{x}||_p \ \text{ subject to } \ A\boldsymbol{x} = \boldsymbol{b}. \tag{1.1}$$

Specifically we shall direct our primary focus to the ℓ_∞ norm problem,

$$\min \ ||\boldsymbol{x}||_\infty \ \text{ subject to } \ A\boldsymbol{x} = \boldsymbol{b}. \tag{1.2}$$

Geometrically, the above problem can be seen as finding the first point at which the inflated n-dimensional hypercube centred at the origin first touches the solution

space.

It is noted that while it is well known that problem (1.2) as stated above has at least one solution, its uniqueness is certainly not guaranteed. This can be seen quite clearly in a geometric light. Whenever the solution space should happen to lie *on* a face of the hypercube (of arbitrary dimension) then clearly any point in the intersection space will produce the same infinity norm solution.

Also of interest is the determination of the minimum infinity norm solution to an underdetermined linear system subject to inequality constraints:

$$\min \ ||\boldsymbol{x}||_{\infty} \ \text{ subject to } \ ||A\boldsymbol{x} - \boldsymbol{b}||_2 \leq \epsilon. \tag{1.3}$$

Such a situation may correspond to any real world application that would require the equality constraint formulation but is subjected to non-negligible measurement error.

1.2.2 Overdetermined

In this case the system under consideration is inconsistent and no solution exists. An approximation is thus sought which minimizes the residual error vector $\boldsymbol{r}(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b}$ in some norm. We formulate the general problem as follows:

$$\min \ ||\boldsymbol{r}(\boldsymbol{x})||_p \rightarrow \min \ ||A\boldsymbol{x} - \boldsymbol{b}||_p \,. \tag{1.4}$$

The above problem is an unconstrained optimization of a differentiable function for $p \neq \{1, \infty\}$. Problem (1.4) may of course be asked subject to linear constraints of the form $B_1\boldsymbol{x} = \boldsymbol{d}_1$ and $B_2\boldsymbol{x} \geq \boldsymbol{d}_2$, where $B_1, B_2 \in \mathbf{R}^{r \times n}$ and $d_1, d_2 \in \mathbf{R}^r$. Such generalizations may be suitably handled by some of the methods presented in the section to follow, although it shall not be our primary focus.

Specifically, we shall consider the ℓ_∞ norm problems:

$$\min \ ||\boldsymbol{r}(\boldsymbol{x})||_\infty \rightarrow \min \ ||A\boldsymbol{x} - \boldsymbol{b}||_\infty \, . \tag{1.5}$$

Since the systems we will consider are assumed to be inconsistent, it is clear that the range space of A will not intersect $\boldsymbol{b}$. The above problem may be interpreted geometrically as the first intersection of the inflated n-dimensional hypercube, centred at $\boldsymbol{b}$, with the range space of A. The region of intersection is the set of all points in the range space that are closest to $\boldsymbol{b}$ in terms of the Chebyshev distance.

Chapter 2

Literature Review

In this chapter some established methods of solution are expounded. We consider these in two parts, firstly in the underdetermined setting and then in the overdetermined setting.

2.1 Underdetermined

In this section some established methods of solution for problem (1.2) are considered. We consider primarily the methods put forward by Cadzow [6], Shim and Yoon [16], and the LP formulation to which the problem is amenable. The LP formulation due to Abdelmalek [2] being the one presented.

Cadzow's method is computationally efficient and performs the optimization by solving an augmented overdetermined problem in the dual ℓ_1 norm. The solution to the primal problem is then obtained by so called 'alignment' criteria. This method has been criticised by some authors (e.g. Ha & Lee [13]) for its lack of geometric clarity. This issue has been partially alleviated in the paper by Ha and Lee [13] in which the similarity of this method to the geometrically clear method of Shim and Yoon [16] is expounded and its geometric nature partially clarified in its own right.

The algorithm proposed by Shim and Yoon [16] (henceforth referred to as the SY method) is certainly more clear geometrically. The problem is solved in the primal by means of geometric argument. This method, while computationally inferior to that of Cadzow, has the benefits of being both conceptually clear and of not requiring that the matrix A satisfy the Haar condition[1] - a condition strongly required by Cadzow's algorithm [6].

Since problem (1.2) is clearly amenable to a linear programming formulation, such a formulation is included in its standard form in the primal, primarily for the purpose of tangible comparison. The standard primal formulation proves to be distinctly computationally inferior to the alternative methods presented. An adapted formulation of the dual linear programme, with special consideration of the structure and symmetry of the infinity norm, is shown to be computationally comparable with alternative methods and thus possibly preferable for practitioners not wishing to investigate new techniques.

2.1.1 The Shim-Yoon Method

The SY method is inspired by the geometric observation that the optimum solution is the point at which the inflated hypercube first touches the solution space. The vertices (read corners) of the n-dimensional hypercube are mapped by the matrix A into a convex polytope in the solution space; this follows since a convex polytope under linear transformation is mapped into another convex polytope. By the above observation it is thus concluded that the solution to problem (1.2) must lie on a boundary plane of the convex polytope induced by A. Otherwise stated, the vector $\boldsymbol{b}$ must touch a boundary plane of the mapped polytope.

The SY algorithm proceeds by iteratively constructing hyperplanes in the range space of A and checking that they are indeed boundary planes, and that the intersection of

[1] A set of vectors in $\mathbf{R}^n$ is said to satisfy the Haar condition if every set of n vectors is linearly independent

the vector $\mathbf{b}$ with the boundary hyperplane is valid; where the validity corresponds to the intersection being *on* the polytope and not at some other point on the boundary plane.

In the formulation to follow, let $\boldsymbol{Q}_i$ represent a vertex of the hypercube in $\mathbb{R}^n$ and $\boldsymbol{P}_i$ the corresponding mapped vertex of the polytope such that $\boldsymbol{P}_i = A\boldsymbol{Q}_i$. Notice that $\boldsymbol{Q}_i$ is of the form $\gamma\,[\pm 1, \cdots, \pm 1]$, where $\gamma \in \mathbf{R}$ and γ is taken to be 1 for convenience.

The set of vectors $\{(\boldsymbol{P}_1 - \boldsymbol{P}_0), \cdots, (\boldsymbol{P}_{m-1} - \boldsymbol{P}_0)\}$ forms a spanning set for a hyperplane in the range space of A. The condition that the vector $\mathbf{b}$ touches the hyperplane in the range space can therefore be expressed as $\alpha\boldsymbol{b} = \boldsymbol{P}(\boldsymbol{a}),\ (\alpha > 0)$ where $\boldsymbol{P}(\boldsymbol{a}) = \boldsymbol{P}_0 + a_1(\boldsymbol{P}_1 - \boldsymbol{P}_0) + \cdots + a_{m-1}(\boldsymbol{P}_{m-1} - \boldsymbol{P}_0), a_i \in \mathbf{R}$. We may then form the following system:

$$[\,(\boldsymbol{P}_1 - \boldsymbol{P}_0), \cdots, (\boldsymbol{P}_{m-1} - \boldsymbol{P}_0), -\boldsymbol{b}\,] \begin{bmatrix} \\ \boldsymbol{a} \\ \\ \\ \alpha \end{bmatrix} = -\boldsymbol{P}_0.$$

The system above is square, of size $(m \times m)$, we may thus solve for $\left[\boldsymbol{a}^T, \alpha\right]^T$ which gives the optimum solution along the specified hyperplane in $\mathbb{R}^m$. This solution can then be mapped back to the hypercube in $\mathbb{R}^n$ by:

$$\boldsymbol{x} = \frac{1}{\alpha}\,(\,\boldsymbol{Q}_0 + a_1(\boldsymbol{Q}_1 - \boldsymbol{Q}_0) + \cdots + a_{m-1}(\boldsymbol{Q}_{m-1} - \boldsymbol{Q}_0)\,)\,.$$

The algorithm is systematically implemented as follows:

1. Select a vertex of the n-dimensional hypercube at random. We have $A\boldsymbol{Q}_0^{(1)} = \boldsymbol{P}_0^{(1)}$ where $\boldsymbol{Q}_0^{(1)}$ is such a vertex.

2. Determine the n nearest neighbours of the selected vertex. The nearest neighbours are the vertices of the hypercube which differ from $\boldsymbol{Q}_0^{(1)}$ by a single sign

change of one of the components. By way of example note that for a given neighbour, $\boldsymbol{P}_i^{(1)} = A\boldsymbol{Q}_i^{(1)} \Rightarrow \boldsymbol{Q}_i^{(1)} = E_i\boldsymbol{Q}_0^{(1)}$ where the matrix E_i is given as, $E_i = diag\,[1, \cdots, 1, -1, 1, \cdots, 1]$ with the negative at the i^{th} location. There are clearly $n-1$ such neighbours.

3. Of those $n-1$ nearest neighbours, select $m-1$ at random and then form the matrix whose columns are the set of spanning vectors for a hyperplane in $\mathbf{R}^m$ as $B = [\boldsymbol{P}_1 - \boldsymbol{P}_0, \cdots, \boldsymbol{P}_{m-1} - \boldsymbol{P}_0]$. We now check that the hyperplane is indeed a bounding hyperplane to the mapped polytope by checking the sign of the dot product of the normal to the hyperplane with those nearest neighbours not used in the formation of the hyperplane.

   A normal to the hyperplane $\boldsymbol{u}$, is obtained from the system $B^T\boldsymbol{u} = 0$. For each of the $n-(m-1)$ remaining nearest neighbours check $w_i = \boldsymbol{u}^T \cdot (\boldsymbol{P}_i - \boldsymbol{P}_0),\ i = (m, m+1, \ldots, n)$. If $\text{sgn}(w_i) = \text{sgn}(w_j),\ \forall i, j$ then all the nearest neighbours are on the same 'side' and the constructed hyperplane is indeed a boundary hyperplane. Otherwise select a different set of $m-1$ nearest neighbours.

4. Solve $[B, -\boldsymbol{b}] \begin{bmatrix} \\ \boldsymbol{a} \\ \\ \alpha \end{bmatrix} = -\boldsymbol{P}_0$, for $\boldsymbol{a}$ and α. If we have that $0 \leq a_i \leq 1,\ \forall\, i$ then an optimal solution has been obtained. Otherwise the solution obtained on the hyperplane is not in fact *on* the polytope and the solution is invalid. We then perform an update as follows:

$$\boldsymbol{Q}^{k+1} = \boldsymbol{Q}^k + \sum_{i=1}^{m-1} \hat{a}_i (E_i - I)\boldsymbol{Q}^k,$$

$$\boldsymbol{P}_0^{k+1} = \boldsymbol{P}_0^k + \sum_{i=1}^{m-1} \hat{a}_i (\boldsymbol{P}_i - \boldsymbol{P}_0^k),$$

where

$$\hat{a}_i = \begin{cases} 0 & a_i \leq 1, \\ \\ 1 & a_i > 1. \end{cases}$$

The updates above amount to selecting a new vertex in $\mathbb{R}^n$ by changing the signs of the current vertex at the offending locations.

The SY algorithm, while conceptually clear, is computationally unsound. If the vertex initially selected is an interior vertex in the mapped polytope then there will exist no boundary hyperplane in $\mathbb{R}^m$ containing the vertex. In such an instance the algorithm will continue to select different sets of $m-1$ nearest neighbours until every possibility is exhausted. Only then will a new vertex be selected. The algorithm will therefore go through $\binom{n}{m-1}$ iterations before a new vertex is selected.

At each iteration a normal to the hyperplane corresponding to that particular selection of $m-1$ nearest neighbours must be computed at step 3, as per the algorithm outlined above; this requires determining the null space of an $(m-1) \times m$ system. Thereafter a further $n-(m-1)$ dot products may need to be carried out to determine that this particular selection of nearest neighbours does not constitute a boundary plane. Moreover there is no guarantee that the next vertex selected will not also be an interior vertex.

2.1.2 Cadzow's Method

Cadzow's method works by solving an augmented overdetermined ℓ_1 norm minimization problem and then recovering the ℓ_∞ solution by means of so called 'alignment' criteria. We shall consider firstly the way in which the problem is reformulated and then the means by which the reformulated problem is solved. The conceptual process followed in solving the augmented problem will show strong similarity with our proposed solution method presented in the next chapter.

Application of the duality principle yields:

$$\begin{aligned} \min \ ||\boldsymbol{x}||_\infty \ &= \ \max \boldsymbol{b}^T\boldsymbol{u}, \\ A\boldsymbol{x} = \boldsymbol{b} \qquad &\qquad \left|\left|A^T\boldsymbol{u}\right|\right|_1 \leq 1. \end{aligned}$$

Furthermore a solution to the primal problem is said to be aligned with a solution to the dual problem as follows:

$$(x^*)_i = \begin{cases} (\boldsymbol{b}^T\boldsymbol{u}^*)\mathrm{sgn}(A^T\boldsymbol{u}^*)_i & \text{if } (A^T\boldsymbol{u}^*)_i \neq 0 \\ \alpha_i & \text{if } (A^T\boldsymbol{u}^*)_i = 0 \end{cases}$$

where the superscript * is used to denote the solution at optimum and $\alpha_i \leq \boldsymbol{b}^T\boldsymbol{u}^*$ is a scalar.

Consider the useful decomposition of the matrix A as $A = B + C$, where $B = [\xi_1\boldsymbol{b}, \cdots, \xi_n\boldsymbol{b}]$, with $\xi_i \in \mathbf{R}$ and $C = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_n]$ with the columns of C orthogonal to the vector $\boldsymbol{b}$ i.e. $\boldsymbol{b}^T\boldsymbol{c}_i = 0$. Such a decomposition is unique and is always possible. Note that any vector $\boldsymbol{u}$ may be written as $\boldsymbol{u} = \alpha\boldsymbol{b} + \boldsymbol{p}$ where $\boldsymbol{b}^T\boldsymbol{p} = 0$ and $\alpha \in \mathbf{R}$. Substitution of this expression for $\boldsymbol{u}$ into the dual formulation results in,

$$\begin{aligned} \max \boldsymbol{b}^T\boldsymbol{u} \quad &= \quad \boldsymbol{b}^T\boldsymbol{b} \max \alpha \\ \left|\left|A^T\boldsymbol{u}\right|\right|_1 \leq 1 \qquad &\qquad \left|\left|\alpha B^T\boldsymbol{b} + C^T\boldsymbol{u}\right|\right|_1 \leq 1. \end{aligned}$$

The constraint on the right may be written as:

$$|\alpha| \leq \frac{1}{\left|\left|B^T\boldsymbol{b} + C^T\frac{\boldsymbol{u}}{\alpha}\right|\right|_1}. \tag{2.1}$$

It is noted in [6] that since we wish to maximize α over all possible values of $\boldsymbol{u} \in \mathbf{R}^m$ satisfying the above equation we may equivalently solve the following problem,

$$\min \left|\left|B^T\boldsymbol{b} + C^T\boldsymbol{u}\right|\right|_1,$$

this is the overdetermined ℓ_1 norm minimization referred to previously.

The statement of the initial problem may therefore be reformulated in terms of the augmented dual as,

$$||\boldsymbol{x}^*||_\infty \quad = \quad \max \boldsymbol{b}^T\boldsymbol{u} \quad = \quad \boldsymbol{b}^T\boldsymbol{b}\, \tfrac{1}{||B^T\boldsymbol{b}+C^T\boldsymbol{u}^*||_1},$$

$$\left|\left|A^T\boldsymbol{u}\right|\right|_1 \leq 1.$$

The augmented problem is now to minimize $\left|\left|B^T\boldsymbol{b} + C^T\boldsymbol{u}\right|\right|_1$ for $\boldsymbol{u} \in \mathbf{R}^m$, subject to equation 2.1. Note explicitly that since $C^T \in \mathbf{R}^{n\times m}$ the augmented problem is an unconstrained, *overdetermined* ℓ_1 norm minimization problem.

At each iteration the current feasible solution is updated by a line search like procedure as follows $\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \epsilon\boldsymbol{\eta}$. Consider the function $f(\boldsymbol{u}) = B^T\boldsymbol{b} + C^T\boldsymbol{u}$. For a given iteration of the algorithm we have:

$$\left|\left|f\left(\boldsymbol{u}^{k+1}\right)\right|\right|_1 = \sum_{i=1}^{n} \left(B^T\boldsymbol{b} + C^T\left(\boldsymbol{u}^k + \epsilon\boldsymbol{\eta}\right)\right) \operatorname{sgn}\left(B^T\boldsymbol{b} + C^T\left(\boldsymbol{u}^k + \epsilon\boldsymbol{\eta}\right)\right).$$

Let us define the so-called index set Π as follows:

$$\Pi = \{i : \left(B^T\boldsymbol{b} + C^T\boldsymbol{u}\right)_i = 0\}.$$

Then for sufficiently small values of ϵ we have:

$$\left|\left|f\left(\boldsymbol{u}^{k+1}\right)\right|\right|_1 = \sum_{i\notin\Pi}\left(B^T\boldsymbol{b} + C^T\left(\boldsymbol{u}^k + \epsilon\boldsymbol{\eta}\right)\right)\operatorname{sgn}\left(B^T\boldsymbol{b}^k + C^T\boldsymbol{u}^k\right) + \sum_{i\in\Pi}|C^T\boldsymbol{\eta}|_i.$$

The expression above may be usefully rewritten as:

$$\left|\left|f\left(\boldsymbol{u}^{k+1}\right)\right|\right|_1 = \left|\left|f\left(\boldsymbol{u}^{k}\right)\right|\right|_1 + \epsilon g\left(\boldsymbol{\eta}\right),$$

where the functional $g(\boldsymbol{\eta})$ is given by:

$$\begin{aligned} g\left(\boldsymbol{\eta}\right) &= \boldsymbol{d}^T\boldsymbol{\eta} + \textstyle\sum_{i\in\Pi}|\left(C^T\boldsymbol{\eta}\right)_i|, \\ &= C\operatorname{sgn}\left[B^T\boldsymbol{b} + C^T\boldsymbol{u}^k\right]\boldsymbol{\eta} + \textstyle\sum_{i\in\Pi}|\left(C^T\boldsymbol{\eta}\right)_i|. \end{aligned}$$

Every feasible solution $\boldsymbol{u}^k$ produces a corresponding index set Π and vector $\boldsymbol{d}$. The algorithm works by obtaining a vector $\boldsymbol{\eta}$ such that $g\left(\boldsymbol{\eta}\right) < 0$, and then finding an ϵ such that $\left|\left|f\left(\boldsymbol{u}^{k+1}\right)\right|\right|_1 < \left|\left|f\left(\boldsymbol{u}^{k}\right)\right|\right|_1$ i.e. a sufficiently small ϵ such that $\operatorname{sgn}\left(f\left(\boldsymbol{u}^{k+1}\right)\right)_i = \operatorname{sgn}\left(f\left(\boldsymbol{u}^{k}\right)\right)_i, \forall i \notin \Pi$.

Now let us form the following $m \times m$ matrix:

$$D = \left[C_\Pi, \boldsymbol{e}_{i_1}, \cdots, \boldsymbol{e}_{i_{m-k}}\right],$$

where C_Π are the columns of C corresponding to the index set as defined above, and the remaining $m - |\Pi|$ columns are the unit coordinate vectors $\boldsymbol{e}_i$ which contain all zero components except for the i^{th} component which is a one. It is noted in [6] that the coordinate vectors may always be chosen in such a way as to ensure that D is of full rank.

We now consider the vector $\boldsymbol{\beta}$ computed from the following equation:

$$\boldsymbol{d} = D\boldsymbol{\beta} = D \begin{bmatrix} \boldsymbol{\beta}_1 \\ \\ \boldsymbol{\beta}_2 \end{bmatrix} \Rightarrow \begin{bmatrix} \boldsymbol{\beta}_1 \\ \\ \boldsymbol{\beta}_2 \end{bmatrix} = D^{-1}\boldsymbol{d},$$

where $\boldsymbol{\beta}_1$ is of size $|\Pi|\times 1$, and $\boldsymbol{\beta}_2$ is of size $(m-|\Pi|)\times 1$. The direction vector $\boldsymbol{\eta}$ may now be determined from the vector $\boldsymbol{\beta}$. We consider two cases.

In case one, $\boldsymbol{\beta}_2 \neq \mathbf{0}$ and the direction vector may be taken to be $\boldsymbol{\eta}^T = -\left[\mathbf{0}^T, \operatorname{sgn}\left(\boldsymbol{\beta}_2\right)^T\right] D^{-1}$. It is shown in [6] that such a selection for $\boldsymbol{\eta}$ results is $g\left(\boldsymbol{\eta}\right) < 0$. Moreover, $\boldsymbol{\eta}$ is orthogonal to each column of C_Π. Therefore as we increase the value of ϵ a previously nonzero element of $f\left(\boldsymbol{u}^k\right)$ is driven to zero.

In the second case, $\boldsymbol{\beta}_2 = \mathbf{0}$ and it is clear that $\boldsymbol{d}$ is a linear combination of C_Π with coefficients $\boldsymbol{\beta_1}$. It is shown in [6] that if $|\left(\beta_1\right)_i| \leq 1$ for all $i = 1, \cdots, |\Pi|$ then there exists no $\boldsymbol{\eta}$ such that $g\left(\boldsymbol{\eta}\right) < 0$. This is the stopping condition used by Cadzow's method. Alternatively, if there exists at least one $|\left(\beta_1\right)_p| > 1$, then the direction vector $\boldsymbol{\eta}$ may be taken to be $\boldsymbol{\eta}^T = -\left[\boldsymbol{e}_p^T, \mathbf{0}\right] D^{-1}\operatorname{sgn}\left(\left(\beta_1\right)_p\right)$. This choice of direction vector results in $g\left(\boldsymbol{\eta}\right) < 0$. No suggestion is given in [6] as to what should be done in the instance that there exists more than one component of $\boldsymbol{\beta}_1$ with absolute value greater than 1. This question corresponds to the single element exchange heuristic discussed in the corresponding section of the PPF method, section 3.4.1.

Furthermore, this selection for $\boldsymbol{\eta}$ is orthogonal to each column of $C_{\Pi\setminus p}$ and thus, as we increase the value of ϵ, $m-1$ components of the function $f\left(\boldsymbol{u}\right)$ are maintained at zero while one previously nonzero component is driven to zero at each iteration.

We now consider the procedure by which ϵ is to be selected. Once a direction vector $\boldsymbol{\eta}$ has been selected, we calculate the set of all positive ϵ such that a previously nonzero element of the function $f\left(\boldsymbol{u}\right)$ is driven to zero. The positivity constraint is clearly necessary to ensure $\left|\left|f\left(\boldsymbol{u}^{k+1}\right)\right|\right|_1 < \left|\left|f\left(\boldsymbol{u}^{k}\right)\right|\right|_1$. It is further required that ϵ be sufficiently small, so that $\operatorname{sgn}\left(f\left(\boldsymbol{u}^{k+1}\right)_i\right) = \operatorname{sgn}\left(f\left(\boldsymbol{u}^{k}\right)_i\right)\ \forall\ i \notin\ \Pi$. It is shown in

[6] that the smallest such ϵ will always satisfy this sign relation. It is further noted that for larger values of ϵ, the sign relation may still hold, and the infinity norm may be further decreased. Cadzow suggests that the largest valid ϵ be selected at each iteration.

In summary then, the process by which the function $||f(\boldsymbol{u})||_1 = \left|\left|B^T\boldsymbol{b} + C^T\boldsymbol{u}\right|\right|_1$ is minimized, is premised on the observation that, at minimum, at least $m-1$ components of the function $f(\boldsymbol{u}) = B^T\boldsymbol{b} + C^T\boldsymbol{u}$, are zero [5]. The algorithm may therefore be logically divided into two parts. Firstly a feasible solution with $m-1$ zeros is obtained and secondly the locations of those zero components are changed in such a way as to reduce the value of the function $||f(\boldsymbol{u})||_1$ at each iteration. The way this is done in [6] is akin to a line search procedure in which a descent direction $\boldsymbol{\eta}$ and a step length ϵ is found at each iteration. Note that the existence of an optimal solution to the augmented dual problem with $m-1$ zero elements directly implies, by way of the alignment criteria, that at least $n-(m-1)$ elements of the solution vector to the primal problem are equal in absolute value and are equal to the ℓ_∞ norm. This is then a necessary condition for optimality.

The solution is iteratively updated as $\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \epsilon\boldsymbol{\eta}^k$ such that $\left|\left|f(\boldsymbol{u}^{k+1})\right|\right|_1 \leq \left|\left|f(\boldsymbol{u}^k)\right|\right|_1$ and a previously nonzero element is driven to zero at each step. It is acknowledged in [6] that many procedures for the selection of $\boldsymbol{\eta}$ may exist and that only one such method is therein presented. Once a solution with $m-1$ zeros has been obtained, one previously nonzero element is made to be zero while one element that is currently zero is allowed to become nonzero. This so called column-swop procedure is the key to the efficiency of Cadzow's algorithm. It is conceptually equivalent to the process in the SY algorithm in which a new vertex is selected by changing the sign of the offending elements.

The expediency of this procedure is critical to the success of this method as there exists an upper bound of $\binom{n}{m-1}$ possible column-swops to which the algorithm is theoretically exposed - this is the number of possible ways to place $m-1$ zero elements in the n vector resulting from the function $f(\boldsymbol{u})$. The caveat of Cadzow's method,

as pointed out by numerous authors including Cadzow himself [6, 13, 16], is that the algorithm requires that the columns of the matrix C satisfy a Haar condition. This is the condition that requires that for a given set of vectors in $\mathbb{R}^n$, any set of n vectors is linearly independent. If we explicitly define, $\{\boldsymbol{c}_i\} \in \mathbf{R}^m$ to be the columns of the matrix C, then the Haar condition requires any m columns of C to be linearly independent. This assumption is required in the determination of a valid direction vector $\boldsymbol{\eta}$ at each iteration.

2.1.3 Linear Programming Formulation

Problem (1.2) may be easily couched in linear programming formalism. To these ends we present the formulation carried out in [6] below.

$$\begin{aligned} \min ||\boldsymbol{x}||_\infty &= \min s, \\ A\boldsymbol{x} = \boldsymbol{b}, \qquad & A\boldsymbol{x} = \boldsymbol{b}, \\ & |x_i| \leq s. \end{aligned}$$

The positivity constraint on x_i and the transformation of the inequality constraints to equality constraints may be carried out respectively as follow:

$$\begin{aligned} \hat{\boldsymbol{x}} = \boldsymbol{x} + s\boldsymbol{e} \;\rightarrow\; 0 \leq \hat{x}_i \leq 2s, \quad s \in \mathbb{R}, \\ \hat{\boldsymbol{x}} + \boldsymbol{q} = 2s\boldsymbol{e}, \qquad\qquad q_i \geq 0. \end{aligned}$$

where $\boldsymbol{e}$ is a vector of ones and s is a positive scalar. An initial feasible solution is obtained in the standard manner by way of the addition of m slack variables, $[a_1, \cdots, a_m]$. Finally the objective function may be transformed to a maximization

as $Z = -s - \boldsymbol{a}M$. The final programme obtained may be presented as,

$$\max Z$$

$$\text{subject to} \quad \begin{cases} A(\hat{\boldsymbol{x}} - s\boldsymbol{e}) + I\boldsymbol{a} = \boldsymbol{b}, \\ \\ \hat{\boldsymbol{x}} + \boldsymbol{q} - 2s\boldsymbol{e} = \boldsymbol{0}, \end{cases}$$

where $\hat{x}_i, q_i, b_i, a_i, s \geq 0$, I is the identity matrix and $M \in \mathbb{R}$ with $M >> 1$ so as to drive out the artificial variables. The programme consists of $m + n$ equations in $2n+m+1$ unknowns. The size of the programme is the main reason for its inefficiency and the limitation of the simplex algorithm in this formulation. The reduction of the size of the programme was the primary contribution of Abdelmalek [2] and indeed the reason for the resurgence of linear programming as a plausible alternative to the methods presented. We consider the formulation of the linear programme that leads to the reduction:

$$\min s, \qquad \min z = \boldsymbol{e}_{n+1}^T \begin{pmatrix} \boldsymbol{x} \\ \\ s \end{pmatrix},$$

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad A\boldsymbol{x} \geq \boldsymbol{b},$$

$$\boldsymbol{x} + s\boldsymbol{e} \geq \boldsymbol{0},$$

$$|x_i| \leq s, \qquad -A\boldsymbol{x} \geq -\boldsymbol{b},$$

$$-\boldsymbol{x} + s\boldsymbol{e} \geq \boldsymbol{0}.$$

where $\boldsymbol{e}_{n+1}^T = [0, \cdots, 0, 1]$, $\boldsymbol{e} = [1, \cdots, 1]$ and $s \geq 0$. The above programme is then transformed to its dual:

$$\max z = \left[\boldsymbol{b}^T, \mathbf{0}^T, -\boldsymbol{b}^T, -\mathbf{0}\right] \boldsymbol{u},$$

subject to the following equality constraints

$$\begin{pmatrix} A^T & I & -A^T & -I & \mathbf{0} \\ \\ \mathbf{0}^T & \boldsymbol{e}^T & \mathbf{0}^T & \boldsymbol{e}^T & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \\ u_s \end{pmatrix} = \boldsymbol{e}_{n+1},$$

where $I \in \mathbf{R}^{n \times n}$ is the identity matrix. In the above, u_s represents the slack variables added to make the inequality constraints equality, also $u_i \geq 0$. It is shown in [2] that we must have $u_s = 0$ and the above system may thus be slightly reduced, i.e.

$$\begin{pmatrix} A^T & I & -A^T & -I \\ \\ \mathbf{0}^T & \boldsymbol{e}^T & \mathbf{0}^T & \boldsymbol{e}^T \end{pmatrix} \boldsymbol{u} = \boldsymbol{e}_{n+1} \;\rightarrow\; D\boldsymbol{u} = \boldsymbol{e}_{n+1}.$$

This is a system of $n + 1$ constraints in $2(m + n)$ variables. It is the symmetry of the matrix D that is exploited in the formulation by Abdelmalek [2] which allows the tableu specified by D to be halved. The computations in the algorithm are carried out on the $(n+1) \times (m+n)$ reduced tableu. Furthermore the existence of the $(n \times n)$ identity sub matrix in the reduced tableu is exploited by Abdelmalek [2] to obtain an initial feasible solution without the need to invoke any artificial variables.

These reductions to the size of the systems considered in the linear programming formulation are the key to making these methods competitive. A slightly modified simplex algorithm is employed on the reduced tableu after an initial feasible solution has been found.

2.2 Overdetermined

In this section some existing solution methods to problem (1.5) are considered.

A key observation upon which many of the available algorithms in the literature rely heavily is that, at optimum, the residual vector $\boldsymbol{r}(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b}$ has at least $n+1$ components equal in absolute value and maximal. This fundamental result is due to de la Vallée Poussin [11, 18].

We will often make reference to the index set which may be defined as follows:

$$\Lambda(\boldsymbol{x}) = \{i : |r_i(\boldsymbol{x})| = ||\boldsymbol{r}(\boldsymbol{x})||_\infty\},$$

where $\boldsymbol{r}(\boldsymbol{x})$ is the residual vector as previously defined. A useful characterisation of an optimal solution may then be given as follows: $\boldsymbol{x}$ is an optimal solution to problem (1.5) if and only if there exists a subset $\Lambda^* \subseteq \Lambda$ with at most $n+1$ elements, and a nonzero vector $\boldsymbol{v} \in \mathbf{R}^m$, such that:

$$v_i = 0 \qquad \forall i \notin \Lambda^*,$$

$$A^T\boldsymbol{v} = \mathbf{0},$$

$$v_i \text{sgn}(r_i) \geq 0 \quad \forall i \in \Lambda^*.$$

It is noted in [18] that this constitutes a 'zero in the convex hull' type characterisation. For full rank systems, the sort we will consider, there always exists a solution with exactly $n+1$ elements of the residual vector equal in absolute value and maximal [18].

For the special case of an overdetermined system of size $(n+1) \times n$ there exist relatively simple solution methods, the most appealing perhaps being one attributed to de la Vallée Poussin [11], which we discuss in the section on Polya's method. A

closed form solution to the special case was given in [7] in terms of the optimal ℓ_2 solution. Problem (1.5) therefore may be considered all but solved if the critical set Λ can be determined. Indeed most of the methods which we consider will seek to determine this critical set in various ways.

The obvious 'brute force' method is explicitly mentioned in [7], in which every possible $n+1$ component subset is checked. This is clearly computationally infeasible and is noted only for completeness. Polya, in the late 1950's proposed a method to determine the critical set Λ by iteratively solving problem (1.4) for increasing values of p [9]. The critical set is then determined by numerical extrapolation. An improved extrapolation procedure was presented by Fletcher et al. in [12] to accelerate the convergence of Polya's method [18]. This class of methods can nevertheless be slow to converge and is not considered competitive.

An exchange algorithm was proposed by Stiefel in 1959 [9, 17, 18], in which different $n+1$ row sub-matrices are considered in a systematic fashion. The algorithm is also known as the ascent algorithm [9] and represents the first in a new class of algorithms. It was later shown to be equivalent to the linear programming formulation of the dual problem [15].

The large number of slack variables that are required and the fact that the variables need not satisfy a positivity constraint made the linear programming formulation of the primal problem intractable. These issue are not present in the dual formulation in which only a single slack variable is required and the variables all satisfy the requisite positivity constraints. Moreover the basis matrices are potentially of a substantially reduced size [18]. It is thus the dual problem which received the most attention. In the following section we consider the LP formulation due to Abdelmalek [1] which shares many fundamental traits with the foundational algorithm of Barrodale and Phillips [3] which made use of the special symmetric structure of the dual problem to notably reduce the computational expense of the algorithm. We prefer the exposition of Abdelmalek primarily for consistency and comparison with his algorithm in the underdetermined setting.

Primal formulations of problem (1.5) do exist and correspond to 'descent methods'. These methods were argued by Bartels et al. [4] to indeed be competitive with the class of 'ascent' methods after improved starting points were outlined in [4]. While we will not directly consider the method therein outlined, we take critical note of the suggestion that the popular belief that dual methods are innately superior for problem (1.5) is possibly fallacious and has been deleterious to the development of effective primal methods.

We note finally that all of the methods outlined above (aside from Poly's method) make use of the polyhedral nature of the objective function and that they all constitute vertex-to-vertex methods. In more recent times greater attention has been focused on the class of interior point methods. It is noted in [18] that unlike in the ℓ_1 norm case, a thorough review and benchmarking of algorithms for solution of large scale problems (1.5) seemed to be unavailable at the time of publishing. To the best of our knowledge such a comparison remains void.

2.2.1 Polya's Method

Polya's method is premised on the fact that at optimum the solution to problem (1.5) is equal to the solution of some relevant $n+1$ row submatrix of A [9, 11]. One may therefore, having established this critical subset, disregard the remaining equations. The solution to problem (1.5) for this special case of $n+1$ equations in n variables permits a variety of solution methods - the method due to de la Vallèe Poussin being presented here. It turns out that the solution to a system of this size, $A \in \mathbf{R}^{(n+1)\times n}$, and therefore the solution to the full system, has $n+1$ components of the residual vector equal in absolute value and maximal. The chosen method of solution to this

special case is outlined below:

$$A\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{\sigma} z,$$

$$[A, -\boldsymbol{\sigma}] \begin{bmatrix} \boldsymbol{x} \\ \\ z \end{bmatrix} = \boldsymbol{b} \;\; \rightarrow \;\; D \begin{bmatrix} \boldsymbol{x} \\ \\ z \end{bmatrix} = \boldsymbol{b},$$

where $D \in \mathbf{R}^{(n+1)\times(n+1)}$, $\boldsymbol{\sigma} \in \mathbf{R}^{n+1}$ and $z \in \mathbf{R}$. The constant z may now be computed using Cramer's rule as,

$$z = \frac{\mathrm{Det}(D_{n+1})}{\mathrm{Det}(D)},$$

where D_{n+1} denotes the matrix D with the $(n+1)^{st}$ column replaced with the vector $\boldsymbol{b}$. In order to minimize the value of the infinity norm of $\boldsymbol{r} = z\boldsymbol{\sigma}$ we are required to maximize the value of $\mathrm{Det}(D)$. If we consider the computation of the determinant by cofactor expansion it becomes evident that $\mathrm{Det}(D)$ is a linear function of σ_i. The solution may therefore be found simply by taking σ_i to be equal to the sign of the corresponding cofactor.

Polya's method provides a way to determine this crucial subset Λ, by iteratively solving problem (1.4) for increasing values of p. To see that this is a valid approach we note that for a given vector $\boldsymbol{x} \in \mathbf{R}^n$ we have $||\boldsymbol{x}||_p \rightarrow ||\boldsymbol{x}||_\infty = \max_i\{|x_i|\}$ as $p \rightarrow \infty$. It is further shown in [9] that this convergence is in a monotonically decreasing fashion. Thus we have,

$$||\boldsymbol{r}^\infty||_\infty \leq ||\boldsymbol{r}^p||_\infty \leq ||\boldsymbol{r}^p||_p \leq ||\boldsymbol{r}^\infty||_p \,,$$

where $\boldsymbol{r}(\boldsymbol{x})$ represents the residual error vector, and the superscript denotes optimality of the solution with respect to that norm. If we now let $p \rightarrow \infty$ we have that $||\boldsymbol{r}^\infty||_p$ decreases monotonically and converges to $||\boldsymbol{r}^\infty||_\infty$. Therefore by the squeeze theorem we have $||\boldsymbol{r}^p||_p \rightarrow ||\boldsymbol{r}^\infty||_\infty$. We conclude that for large values of p, $\boldsymbol{r}^p \approx \boldsymbol{r}^\infty$. It is noted with greater detail in [9] that a convergent sequence may thus be extracted $\{\boldsymbol{r}^{p_1}, \cdots, \lim_{k\rightarrow\infty} \boldsymbol{r}^{p_k} = \boldsymbol{r}^\infty\}$. This observation conceptually validates Polya's

algorithm.

One is of course still required to actually compute the values of $\boldsymbol{r}^p$ for each value of p. This is easily accomplished due to the differentiability of $||\boldsymbol{r}(\boldsymbol{x})||_p$ for $p \neq \{1, \infty\}$. Newton's method, for example, may be used to compute these values.

Once $\boldsymbol{r}^p$ has been computed for 'sufficiently' high values of p, we may numerically extrapolate for the limit vector we seek. In practice, it proves more expedient to compute $\boldsymbol{r}^p$ only for values large enough to establish the critical set Λ, after which a solution may be computed directly by the procedure outlined above.

Note that we have spoken throughout this section of the residual vector $\boldsymbol{r}(\boldsymbol{x})$. In the case when A is of full rank the vector $\boldsymbol{x}$ may be uniquely calculated from the optimal residual vector.

2.2.2 Stiefel's Method

The method herein presented, due to Stiefel, is known interchangeably as Stiefel's method, the exchange method and the ascent method. We shall use the latter to emphasis a property of the algorithm that results in an *increased* value for the infinity norm of the residual vector at each iteration. Note that we assume throughout this section that the rows of the matrix A satisfies the Haar condition as previously described.

As was mentioned in the previous section, it is known that the solution to the full system is equal to the solution for some $n+1$ row submatrix specified by, $\Delta = \{i_1, \cdots, i_{n+1}\}$. A useful characterisation of an optimal solution to the case when $A \in \mathbf{R}^{(n+1)\times n}$ is as follows:

1. $r_i(\boldsymbol{x}) = \sigma_i z \ \ \forall\, i \in \Delta$,
2. $\mathbf{0} \in \mathbf{H}\{\sigma_1 A^{i_1}, \cdots, \sigma_{n+1} A^{i_{n+1}}\}$,

where $\sigma_i = \pm 1$, $\mathbf{H}$ represents the convex hull of the specified vectors and A^i are rows

of A. The second requirement is akin to the 'zero in the convex hull' requirement mentioned earlier in this section. If the above criteria is satisfied, then $\boldsymbol{r} = \boldsymbol{\sigma} z$ is an optimal solution of the reduced system.

The algorithm proceeds by iteratively computing the minimum infinity norm solution for a succession of subsystems of size $(n+1)\times n$. To these ends we observe the so-called exchange theorem [9, p. 45] which states that given a set of vectors $\{\boldsymbol{v}_0, \cdots, \boldsymbol{v}_{n+1}\}$ which satisfies the Haar condition, and given that $\mathbf{0} \in \mathbf{H}\{\boldsymbol{v}_0, \cdots, \boldsymbol{v}_n\}$, then there exists an index $j \leq n$ such that replacing $\boldsymbol{v}_j$ by $\boldsymbol{v}_{n+1}$ keeps $\mathbf{0}$ in the convex hull of the newly specified set.

At each iteration we have an index set Δ specifying the $n+1$ row submatrix, and a vector of signs $\boldsymbol{\sigma}$ such that $\mathbf{0}$ lies in the convex hull as specified in the characterisation above. By the method of de la Vallèe Poussin [11], we then compute the vector $\boldsymbol{x}$ and scalar z such that $\sigma_j r_{i_j}(\boldsymbol{x}) = z$ as per the characterisations above. The requirement that we be able to compute $\boldsymbol{x}$ and z requires some assumptions on the data matrix - specifically that $\left[A^{\Delta}, -\boldsymbol{\sigma}\right]$ be nonsingular. If it so happens that $z < 0$ then taking $\boldsymbol{\sigma} = -\boldsymbol{\sigma}$ ensures $z > 0$ and that $\mathbf{0}$ remains in the convex hull.

The exchange procedure is as follows: if we have $z = ||\boldsymbol{r}(\boldsymbol{x})||_{\infty}$ for the full system then we are at the optimum and the algorithm terminates, otherwise there exists an element of the residual vector that is larger than z, suppose α is such an element. Let $\mu = \text{sgn}\,(r_{\alpha}(\boldsymbol{x}))$, then one of $\{\sigma_1 A^1, \cdots, \sigma_{n+1} A^{n+1}\}$ is replaced with μA^{α} in such a way that $\mathbf{0}$ remains in the convex hull, as is guaranteed to be possible by the exchange theorem stated above, and the infinity norm of the submatrix is increased. Once again we have an index set Λ' and a vector of signs $\boldsymbol{\sigma}'$ and we may iterate the algorithm.

To prove that the algorithm offers guaranteed finite convergence it is shown in [9] that the ascent method cannot possibly 'cycle' in the sense that no index set is ever revisited. This follows from the fact that $z = f\,(\Delta)$ is a strictly increasing function of Δ - that there are a finite number of such sets proves the claim.

To see that $z = f\,(\Delta)$ is a strictly increasing function of Δ we consider a single

iteration of the algorithm from Δ^k to Δ^{k+1}. For simplicity of exposition we assume that $\Delta = \{1, \cdots, n+1\}$, $\alpha = n+2$, and $\Delta^{k+1} = \{2, \cdots, n+2\}$.

We observe that since $|r_2\left(\boldsymbol{x}^k\right)| < |r_{n+2}\left(\boldsymbol{x}^k\right)|$ and $|r_2\left(\boldsymbol{x}^{k+1}\right)| = |r_{n+2}\left(\boldsymbol{x}^{k+1}\right)|$, we have $\boldsymbol{x}^k \neq \boldsymbol{x}^{k+1}$. Furthermore since $\sigma_i r_i\left(\boldsymbol{x}^k\right) - \sigma_i r_i\left(\boldsymbol{x}^{k+1}\right) = z^k - z^{k+1} = \sigma_i A^i \boldsymbol{x}^k - \sigma_i A^i \boldsymbol{x}^{k+1}$, for $i = 2, \cdots, n+1$, and by assumption the rows of A satisfy the Haar condition, we infer that $z^k \neq z^{k+1}$.

Finally we note that since $\sigma_{n+2} r_{n+2}\left(\boldsymbol{x}^k\right) - \sigma_{n+2} r_{n+2}\left(\boldsymbol{x}^{k+1}\right) > z^k - z^{k+1}$, if $z^k - z^{k+1} > 0$ then $\sigma_i r_i\left(\boldsymbol{x}^k\right) - \sigma_i r_i\left(\boldsymbol{x}^{k+1}\right) > 0$, for $i = 2, \cdots n+2$. This contradicts the fact that $\mathbf{0} \in \mathbf{H}\{\sigma_i A^i, 2 \leq i \leq n+2\}$ [9, p. 19] - essentially the theorem therein stated gives a 'zero in the convex hull' as a sufficient condition that a system of inequalities be inconsistent. We therefore conclude that $z^k - z^{k+1} < 0$ as desired.

The algorithm may be initiated with an arbitrary selection of the $n+1$ row sub-matrix Δ. The initial vector of signs $\boldsymbol{\sigma}$ may be found by solving the following system:

$$\sum_{i=1}^{n+1} \gamma_i A^i = \mathbf{0},$$

for real constant γ_i and then setting $\sigma_i = \operatorname{sgn}(\gamma_i)$.

2.2.3 Linear Programming Formulation

In this section the formulation of the primal and dual linear programmes are considered and salient points specific to the improvements made by various authors is discussed. Consider the procedure by which problem (1.5) may be transformed to a linear programme below. Our initial problem maybe stated as:

$$\min \ ||\boldsymbol{r}(\boldsymbol{x})||_\infty \ = \ \min \ ||A\boldsymbol{x} - \boldsymbol{b}||_\infty .$$

$$\boldsymbol{x} \in \mathbf{R}^n \qquad\qquad \boldsymbol{x} \in \mathbf{R}^n$$

This is restated as a primal linear programme as:

$$\min \boldsymbol{e}_{n+1}^T \begin{bmatrix} \boldsymbol{r} \\ \\ s \end{bmatrix},$$

subject to,

$$s \geq 0,$$

$$\begin{bmatrix} A & \boldsymbol{e} \\ \\ -A & \boldsymbol{e} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \\ s \end{bmatrix} \geq \begin{bmatrix} \boldsymbol{b} \\ \\ -\boldsymbol{b} \end{bmatrix},$$

where $\boldsymbol{e}$ is a vector of ones. The constraints in the above programme follow from $|\boldsymbol{r}_i| \leq s \Rightarrow \boldsymbol{r}_i \leq s$ and $-\boldsymbol{r}_i \geq -s$. The above system is of size $2m \times (n+1)$. Since $\boldsymbol{x}$ does not satisfy any positivity constraint, some $2m$ slack variable would have to be added. It is explicitly noted in [15] that these issues are overcome by transformation to the dual problem since all of the dual variables satisfy the requisite positivity constraints and the "dual constraints corresponding to the columns of the primal associated with unconstrained variables, are equations". Explicitly the dual problem may be formulated as follows:

$$\max Z = \left[\boldsymbol{b}^T, -\boldsymbol{b}^T\right] \boldsymbol{u},$$

subject to the following constraints,

$$\begin{bmatrix} A^T & -A^T & \mathbf{0} \\ \\ \boldsymbol{e}^T & \boldsymbol{e}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \\ u_s \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \\ 1 \end{bmatrix}.$$

Here, u_s is a single slack variable added to ensure that all variables satisfy equality constraints. It is shown in [1, 15] that at optimum we must take $u_s = 0$ and the system may be suitably reduced.

Strong symmetries in the constraint matrix allow for a modified simplex method to be carried out on a greatly reduced tableu. By way of explicit example, it is shown in [1, 15] that if a given column of A^T is in the basis then its corresponding column in $-A^T$ cannot also be in the basis at optimum.

Abdelmalek, in 1977, proposed a linear programming formulation for the minimum infinity norm solution to an underdetermined system of linear equations [2] (indeed his method is outlined in the corresponding section of this work) which strongly parallels his method here. His main contributions are similar in both cases and are outlined below.

Firstly it is noted in [15] that the linear programming formulation of the dual problem is conceptually equivalent to Stiefel's exchange method and that it differs only in implementation. One direct improvement of the linear programming formulation is the relaxation of the restrictive requirement of nondegeneracy - that the matrix A satisfies the Haar condition. As can be deduced from the statement of the exchange theorem in the section on Stiefel's method, the Haar condition is strictly required therein. This is a benefit which the method of Abdelmalek shares.

As mentioned previously, symmetries in the constraint matrix are exploited to great effect by Abdelmalek in allowing calculations to be carried out on a substantially reduced tableu. This is an improvement shared with the earlier foundational paper of Barrodale and Phillips [3]. A further improvement made by Abdelmalek was to show the existence of a basic feasible solution without the need to invoke any artificial variables. Thereafter only a slightly modified simplex algorithm is implemented - the modification arriving in the form of an augmented procedure for determining which elements should enter the basis on a given iteration.

2.2.4 Modern Path-Following Methods

The class of path following algorithms are premised on the fact that at optimum the residual error vector has at least $n+1$ components equal in absolute value and maximal. This class of algorithm follows a concise logical structure: first a feasible solution is obtained with $n+1$ components of the residual vector equal in absolute value and maximal, and secondly the location of those components is changed in such a way that $\left|\left|\boldsymbol{r}(\boldsymbol{x}^{k+1})\right|\right|_\infty \leq \left|\left|\boldsymbol{r}(\boldsymbol{x}^{k})\right|\right|_\infty$. By changing the location of components of $\boldsymbol{r}$, we mean to select a different set of $n+1$ components to be equal in absolute value and maximal.

These methods are members of the class of *descent* methods which stand in stark conceptual contrast with the class of ascent methods, some members of which were previously discussed. In the class of ascent methods the infinity norm of the feasible solution is decrease on each iteration. We recall the sentiment expressed in [4] that the class of descent methods may be, contrary to popular belief, superior to ascent methods. Their purported inferiority has been historically attributed to the increased number of required variables in the linear programming formulation and the fact that the variables do not satisfy any positivity constraints. The claimed superiority of descent methods was attributed in [4] to their improved starting solutions for the exchange procedure.

In describing the methods to follow we shall refer frequently to the index set Λ defined, as before, as:

$$\Lambda\left(\boldsymbol{x}\right) = \{i : |r_i(\boldsymbol{x})| = ||\boldsymbol{r}(\boldsymbol{x})||_\infty\}.$$

This corresponds to the set of active constraints as can be seen by consideration of the reformulation of problem (1.5) as, min z such that $|\boldsymbol{r}_i| \leq z$, where $z \geq 0$ and $i = \{1, \ldots, n\}$. The complement of the index set, written as Λ^c, is the set of all indices not in Λ.

In these methods elements are exchanged between Λ and Λ^c by means of a line search procedure: $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k$. An iteration analysis is now carried out in which procedures by which $\boldsymbol{d}^k$ and α^k may be determined are considered.

Consider the useful decomposition of the residual vector as follows:

$$\boldsymbol{r}(\boldsymbol{x}) = [A\boldsymbol{x} - \boldsymbol{b}] = \begin{bmatrix} A_\Lambda \boldsymbol{x} - \boldsymbol{b}_\Lambda \\ \\ A_{\Lambda^c} \boldsymbol{x} - \boldsymbol{b}_{\Lambda^c} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_\Lambda(\boldsymbol{x}) \\ \\ \boldsymbol{r}_{\Lambda^c}(\boldsymbol{x}) \end{bmatrix},$$

where A_Λ here refers to those rows of A corresponding to those components of the residual error vector with indices in the index set Λ - similarly for A_{Λ^c}. A single iteration of the line search procedure is realized in terms of the above decomposition as:

$$\left|\left|\boldsymbol{r}(\boldsymbol{x}^{k+1})\right|\right|_\infty = \left|\left|\boldsymbol{r}(\boldsymbol{x}^k + \alpha \boldsymbol{d}^k)\right|\right|_\infty = \left|\left| \begin{array}{c} A_\Lambda \boldsymbol{x}^k - \boldsymbol{b}_\Lambda + \alpha A_\Lambda \boldsymbol{d}^k \\ \\ A_{\Lambda^c} \boldsymbol{x}^k - \boldsymbol{b}_{\Lambda^c} + \alpha A_{\Lambda^c} \boldsymbol{d}^k \end{array} \right|\right|_\infty .$$

We demand that the step length parameter is always greater than zero, $\alpha > 0$ - the procedure by which the step length parameter is to be determined is considered at the end of this section. A normalized direction vector therefore corresponds to a descent direction if and only if $A_\Lambda \boldsymbol{d}^k = -\text{sgn}\,(A_\Lambda \boldsymbol{x}^k - \boldsymbol{b}_\Lambda)$, this logic is expounded in section 3.2. The direction vector may be computed from this relation.

In the instance in which $|\Lambda| < n$ the system $A_\Lambda \boldsymbol{x} - \boldsymbol{b}_\Lambda = \boldsymbol{0}$ is underdetermined and the solution for $\boldsymbol{d}^k$ may be realized in a number of ways. It is suggested in [7] that the direction vector always be computed by using the Moore-Penrose pseudo inverse which gives: $\boldsymbol{d}^k = -A_\Lambda^T (A_\Lambda A_\Lambda^T)^{-1} \text{sgn}\,(A_\Lambda \boldsymbol{x}^k - \boldsymbol{b}_\Lambda)$. A direction vector may also in this case be computed in a fashion similar to the 'Iterative Ascent' method, see section 3.3.2, outlined in the corresponding section on the Primal Path Following (PPF) method. Such a procedure would correspond to setting $n - |\Lambda|$ components of $\boldsymbol{d}^k$ to zero and simply solving the resulting square system for the remaining components.

When the index set is 'full', i.e. $|\Lambda| = n+1$, the solution must be checked for optimality before any exchange procedure can be initiated. An optimal solution will correspond to the situation in which no descent direction exists.

To these ends we first consider a result due to Cadzow [7] which gives a closed form expression for the optimum solution to problem (1.5) for the special case, $A \in \mathbf{R}^{(n+1)\times n}$ in terms of the least squares solution as:

$$\boldsymbol{x}_\infty^* = \boldsymbol{x}_2^* - \frac{\boldsymbol{r}(\boldsymbol{x}_2^*)\boldsymbol{b}}{||\boldsymbol{r}(\boldsymbol{x}_2^*)||_1}[A^TA]^{-1}A^T\text{sgn}\,(\boldsymbol{r}(\boldsymbol{x}_2^*)). \tag{2.2}$$

Here $\boldsymbol{x}_\infty^*$ and $\boldsymbol{x}_2^*$ are the optimal solutions to min $||A\boldsymbol{x}-\boldsymbol{b}||_\infty$ and min $||A\boldsymbol{x}-\boldsymbol{b}||_2$, respectively.

We seek to determine whether a solution is optimal by asking whether or not a descent direction exists, but by the result in equation (2.2) above, we have a closed form expression for the direction vector from the reduced expression $A_\Lambda \boldsymbol{d}^k = -\text{sgn}\,(A_\Lambda \boldsymbol{x}^k - \boldsymbol{b}_\Lambda)$, since it is of size $(n+1)\times n$ as required.

Explicitly then we have:

$$\boldsymbol{d}^k = \boldsymbol{d}_2^k - \frac{\boldsymbol{r}_\Lambda(\boldsymbol{d}_2^k)\boldsymbol{r}_\Lambda(\boldsymbol{x})}{\left|\left|\boldsymbol{r}_\Lambda(\boldsymbol{d}_2^k)\right|\right|_1}[A_\Lambda^TA_\Lambda]^{-1}A_\Lambda^T\text{sgn}\,(\boldsymbol{r}_\Lambda(\boldsymbol{d}_2^k)),$$

where $\boldsymbol{d}_2^k$ is the minimum ℓ_2 norm solution of the reduced system.

A solution is optimal if and only if $\boldsymbol{d}^k = \boldsymbol{0}$. In this situation a local minimum has been found and, by the convexity of the objective function it is the global minimum also. Otherwise it is clear that the infinity norm of the residual may be reduced. The direction vector in [7] is determined in this manner.

An alternative procedure, not considered in [7] or elsewhere, which we propose for checking optimality, and one that will lead to heuristic choices for the direction vector, is described below. It parallels the reasoning in the corresponding section on the PPF method presented in section 3.2. The new heuristics suggested are shown in section

4.1.2 to be powerful in the underdetermined setting and may provide equal gains in the overdetermined setting.

Once a feasible solution has been obtained with $n+1$ components equal in absolute value and maximal, it may still be possible to reduce the infinity norm of $\boldsymbol{r}(\boldsymbol{x})$ by moving in some direction such that for some $i \in \Lambda$, x_i decreases faster than the remaining elements in Λ. This is equivalent to selecting a different $n+1$ components to be equal in absolute value and maximal, i.e. selecting a different index set. We consider a process by which only one element is removed from the index set at each iteration.

To remove a single element from the index set, we simply pretend it is not there on the current iteration - i.e. we solve for $\boldsymbol{d}^{k+1}$ by considering some n row sub-matrix of A_Λ corresponding to all but one of the elements currently in Λ. The resulting direction vector is uniquely determined and may or may not correspond to a descent direction.

Suppose that element i is to be removed from Λ - the resulting direction vector is then uniquely determined and is a descent direction if and only if,

1. $\text{sgn}\left(A_i\boldsymbol{d}^k\right) = -\text{sgn}\left(A_i\boldsymbol{x}^k - b_i\right)$

2. $|A_i\boldsymbol{d}^k| > 1$,

where A_i is the i^{th} row of A. The first condition ensures that for a positive step length the i^{th} component will decrease. The second condition is required to ensure that the i^{th} component decreases faster than the remaining components in the index set. In the absence of the second condition, we have $|\boldsymbol{x}^{k+1}_{\Lambda\setminus i}| < |x_i^{k+1}|$ for a small positive choice of the step length parameter - the maximality of the index set elements is not maintained.

We may thus check these conditions for each of the elements of Λ and determine which elements, if removed, would result in a descent direction. If a unique such element

exists, then clearly that must be the choice of element to remove from Λ and a unique direction vector results. If however more than one element, if removed, would result in a descent direction vector, then a heuristic choice for the direction vector must be made. We may take the largest valid $|A_i \boldsymbol{d}^k|$, the smallest, or perhaps some linear combination of the resulting descent directions. Each of these options is considered in detail in the corresponding section of the PPF method entitled "Exchange Heuristics", see section 3.4.

Suppose it is found that indices $\{u, v, w\}$ all correspond to descent directions when removed from Λ. Then we may take $\boldsymbol{d}^* = c_1 \boldsymbol{d}^u + c_2 \boldsymbol{d}^v + c_3 \boldsymbol{d}^w$, for appropriate choice of constants $\boldsymbol{c} = [c_1, c_2, c_3]$. It is possible to choose $\boldsymbol{c}$ in such a way as to decrease all of these components equally, see section 3.4.2, in a sense hoping to ensure that they have a minimal chance of reentering Λ by ensuring that the decrease in any one component is not too small. If no element were ever to reenter the index set, then the algorithm would be very rapidly convergent indeed. The details of such a selection can be followed in the corresponding underdetermined section. Notice that after performing a line search in this direction, the index will be diminished, $|\Lambda| < n+1$. The index set will then need to be 'rebuilt' in subsequent iterations until once again $|\Lambda| = n+1$.

Both the choice of direction vector outlined in the previous paragraph and the choice made by Cadzow in [7] result in a process that does *not* maintain $n+1$ elements in the index set Λ at each iteration. Thus, the procedure has the potential to be greatly superior to standard exchange procedures in which only a single element is exchanged between Λ and Λ^c at each iteration. These multiple element exchange algorithms are perhaps closer in spirit to interior point algorithms which are of primary interest in the current research.

In all cases the step length parameter is equivalently computed as the smallest positive value for which some index previously in Λ^c first enters Λ. Since all elements in Λ have the same absolute value by design we may consider a single representative component,

r_i for some $i \in \Lambda$ (not the element we have chosen to remove) and compute α as:

$$\alpha^* = \min_\alpha \{ \, |r_i(\boldsymbol{x}^k + \alpha \boldsymbol{d}^k)| = |\boldsymbol{r}_{\Lambda^c}(\boldsymbol{x}^k + \alpha \boldsymbol{d}^k)| \, \},$$

for each $j \in \Lambda^c$. To see why the smallest positive value for α ensures that elements in Λ remain maximal, consult section3.1.3.

Chapter 3

Primal Path-Following (PPF) Method For Underdetermined Systems

In this section a new method of solution is proposed for problem (1.2). It is a primal method like that of the Shim-Yoon method (SY) but it is decidedly more in the spirit of Cadzow's path following algorithm. It is geometrically and conceptually clear and provides important new insight into the nature of the problem.

The method is premised on the observation that at the optimum, at least $n-(m-1)$ components of the solution vector are equal in absolute value and that these components are also maximal. This assertion may be deduced from the alignment criteria between the solution to the primal and the dual problem, as previously described in Cadzow's method, and the fact that an optimal solution to the dual problem contains at least $m-1$ zero components [5]. An alternative proof is given in [13], although this result has long been known.

The algorithm is thus logically divided into two parts; in Part 1 a feasible solution with $n-(m-1)$ components equal is absolute value and maximal is obtained, and in Part 2 the location of those maximal components in the feasible solution is changed

in such a way as to reduce the infinity norm at each step. What is meant by 'changing the location of maximal components' shall be discussed below.

We shall frequently refer to the set of components that are equal in absolute value and equal to the ℓ_∞ norm of that solution; the following definition establishes some useful notation in this regard.

Definition 1 (Index Set)**.** *The index set, which we will denote as* Λ*, is the set of indices all components of a vector that are maximal in absolute value. The index set may be defined as follows:* $\Lambda(\boldsymbol{x}) = \{i : |x_i| = ||\boldsymbol{x}||_\infty\}$.

Also then we have the complement as, $\Lambda^c = \{i : i \notin \Lambda\}$. The definition of the index set here is similar, but slightly, different to that made previously in the overdetermined setting. Where there the index set gave the indices of the critical components of the residual error vector, it now refers to the critical components of the solution vector $\boldsymbol{x}$.

Here and throughout this section when we speak of 'changing the location of maximal components' of a feasible solution, we will mean to bring elements into and out of the index set by perturbing the feasible solution in some direction until a element previously in Λ^c joins Λ by virtue of the corresponding component of feasible solution vector becoming maximal in absolute value. Similarly when we speak of 'removing an element from the index set' we will mean to perturb the current feasible solution in some direction in such a manner that the corresponding component of the feasible solution is no longer maximal in absolute value.

An element previously in Λ^c is brought into the index set by means of a *line search* procedure; $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k$ where $\boldsymbol{d}^k$ denotes the direction vector at iteration k and α^k the step length. The direction vector is chosen so as to maintain the elements in Λ while reducing their absolute value, $|x_i^{k+1}| < |x_i^k|$, $\forall i \in \Lambda$.

We shall henceforth understand $\boldsymbol{x}_\Lambda$ to the set of all components of $\boldsymbol{x}$ with indices in the index set Λ, $\boldsymbol{x}_\Lambda = \{x_i : |x_i| = ||\boldsymbol{x}||_\infty\}$, similarly for $\boldsymbol{x}_{\Lambda^c}$. The process by which

elements in Λ^c are brought into Λ is fundamentally the same regardless of whether we are in Part 1 or Part 2 of the algorithm.

To graphically probe the conceptual nature of the proposed algorithm, we shall consider a contour plot of the 2-dimensional solution space to a higher dimensional instance of problem (1.2). Specifically, we consider the system $A\boldsymbol{x} = \boldsymbol{b}$ where, $A \in \mathbf{R}^{8\times 10}$ and the matrix A is of full rank. As such, the solution space has two degrees of freedom as per the Rouchè-Capelli theorem.

Each contour line represents the set of all points in the solution space that produce the same infinity norm.

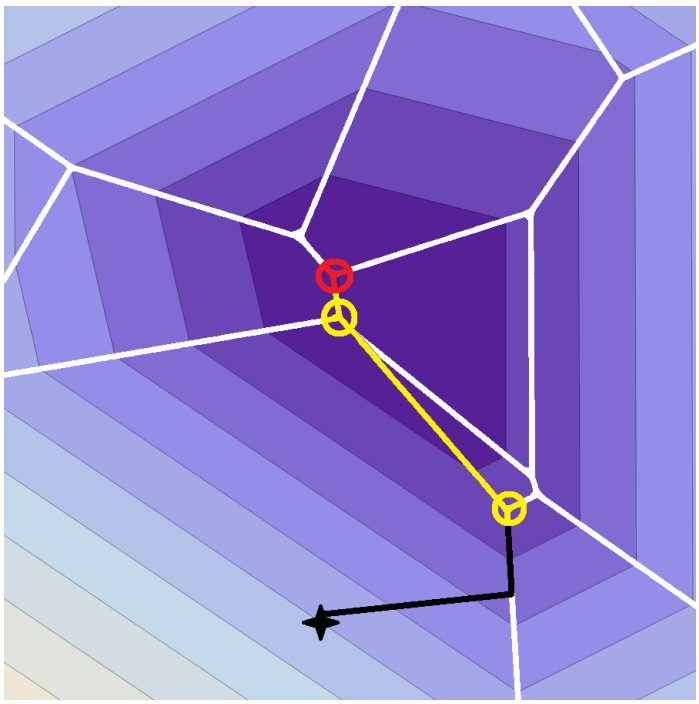


Figure 3-1: Contour plot of 2-dimensional solution space

We consider a 'region' in the above figure to be a space separated by white lines. Each region in the above figure, is the collection of points in the solution space in which a given component of the solution vector is maximal in absolute value, i.e. is equal to the infinity norm. Each white line then is the collection of points in the solution space in which two components are equal in absolute value to each other and

to the infinity norm of that solution. A vertex is a point at which three components are equal in absolute value to each other and to the infinity norm of that solution. This being a 2-dimensional solution space, each vertex corresponds to a 'full' index set. Here and throughout this section we shall understand a 'full' index set to mean $|\Lambda| = n - m + 1$.

The proposed algorithm proceeds in a fashion depicted by the coloured lines. Initially some solution to the underdetermined linear system must be found, in all likelihood this solution will not lie on an edge or vertex - such an initial solution is depicted by the black star. By virtue of a line search procedure, more elements are brought into the index set one at a time. This corresponds to the black lines on the figure: firstly two components of the feasible solution are made equal in absolute value and maximal and the black line terminates on the white edge, and finally three components are made maximal and equal in absolute value and the black line terminates on a vertex. At this point no further progress can be made unless at least one element is removed from the index set and the line search can move away from the initial vertex. Notice also that since we perform the line search in the solution space, all direction vectors reside in the null space of the system under consideration.

The yellow lines in the figure correspond to a single element exchange procedure in Part 2 of the proposed method. That is to say that at each iteration, $|\Lambda| = n - m = 2$ elements are to remain in the index set and the line search proceeds along a white edge. A descent direction for the line search may of course exist in which fewer than $n - m$ elements remain in the index set. Such a direction vector selection procedure will result in a move not only off a vertex, but also off the white edge and back into the interior of a region. A full index set would then need to be reconstructed as before. By the convexity of the objective function, if no adjacent vertex results in a decreased infinity norm, then a local, and thus the global, optimum has been found. The red circle represents such an optimal vertex. At this point the algorithm would terminate.

The proposed method is a primal path-following method and shall henceforth be

referred to as the PPF method.

3.1 Part 1 of the PPF Method

In this section we consider firstly the procedure by which the direction vector for the line search is computed, and secondly the procedure by which the step length parameter is computed.

3.1.1 Initial Solution

It is noted up front that an initial solution to the linear system is required for the line search - while this may be computed in numerous ways, we assume throughout this thesis that the initial solution is taken to be the minimum ℓ_2 norm solution, obtained by way of the Moore-Penrose pseudo-inverse: that is:

$$\boldsymbol{x}^0 = \boldsymbol{x}_2^* = A^T \left(AA^T\right)^{-1} \boldsymbol{b}.$$

Notice that AA^T is invertible if and only if the rows of A are linearly independent. This is guaranteed by the assumption made in the problem statement that A is of full rank.

Once $\boldsymbol{x}^0$ has been determined an initial index set is implicitly defined; in general $|\Lambda| = 1$, although the initial index set may comprise more than one element. If the initial solution is selected such that $n - (m-1)$ components are already equal in absolute value and maximal, then Part 1 of the PPF method is not required. Such a situation would correspond to an initial solution lying on a vertex in Figure (3-1).

3.1.2 Calculating the Direction Vector

We now consider the selection procedure for the direction vector at iteration k. Since $\boldsymbol{x}^{k+1}$ must still satisfy the constraint equations, the direction vector is chosen such that:

$$A\boldsymbol{x}^{k+1} = A(\boldsymbol{x}^k + \alpha \boldsymbol{d}^k) = A\boldsymbol{x}^k + \alpha A\boldsymbol{d}^k = \boldsymbol{b},$$

$$\alpha A\boldsymbol{d}^k = \boldsymbol{0} \Rightarrow \boldsymbol{d}^k \in \text{Null}(A).$$

In order to maintain the equality of the components of $\boldsymbol{x}_\Lambda$ we require that the corresponding components of the direction vector $\boldsymbol{d}^k_\Lambda$, be equal in magnitude to each other. Furthermore, we require that their signs be opposite to those components in $\boldsymbol{x}^k_\Lambda$. To ensure that the infinity norm is reduced at each iteration we require that the step length parameter α satisfies the positivity constraint $\alpha > 0$. We develop two general procedures for the determination of the direction vector.

3.1.2.1 The Standard Method

Consider the natural decomposition of A as $A = [A_\Lambda, A_{\Lambda^c}]$, where A_Λ refers to those columns of A corresponding to the indices in Λ, similarly for A_{Λ^c}. We may thus compute the direction vector as follows:

$$A\boldsymbol{d}^k = A_\Lambda \boldsymbol{d}^k_\Lambda + A_{\Lambda^c}\boldsymbol{d}^k_{\Lambda^c} = 0,$$

$$A_{\Lambda^c}\boldsymbol{d}^k_{\Lambda^c} = -A_\Lambda \boldsymbol{d}^k_\Lambda = A_\Lambda \text{sgn}(\boldsymbol{x}^k_\Lambda),$$

$$\boldsymbol{d}^k_{\Lambda^c} = A^\dagger_{\Lambda^c} A_\Lambda \text{sgn}(\boldsymbol{x}^k_\Lambda), \tag{3.1}$$

where we have *chosen* $\boldsymbol{d}^k_\Lambda = -\text{sgn}\left(\boldsymbol{x}^k_\Lambda\right)$ and the symbol † represents some sort of

general inversion. We then have $\boldsymbol{d}^k = [\boldsymbol{d}^k_\Lambda, \boldsymbol{d}^k_{\Lambda^c}] = [-\text{sgn}(\boldsymbol{x}^k_\Lambda), \boldsymbol{d}^k_{\Lambda^c}]$.

Notice that we have 'normalized' the components of the direction vector corresponding to elements in the index set, i.e. $\boldsymbol{d}^k_\Lambda = -\text{sgn}\,(\boldsymbol{x}^k_\Lambda)$. It follows that, during the line search procedure, elements in Λ^c corresponding to $\boldsymbol{d}^k_i < 1$ will change value more slowly than elements in Λ, and vice versa for elements in Λ^c corresponding to $\boldsymbol{d}^k_i > 1$.

The matrix A_{Λ^c} is of size $m \times (n - |\Lambda|)$ and the system that is required to be solved for $\boldsymbol{d}^k_{\Lambda^c}$ is underdetermined for Part 1 of the algorithm except for the penultimate iterate, when the system is uniquely determined. A heuristic choice is thus to made be in the instance in which $|\Lambda| < (n - m)$; this is considered in section 3.3.

3.1.2.2 The Null Space Method

The direction vector may be found in an alternative manner by invoking the trivial fact that any solution to the linear system can be written as the sum of a particular solution and a linear combination of null space vectors. Let N denote the matrix whose columns form a minimal spanning set for the null space, $N = [\boldsymbol{n}_1, \cdots, \boldsymbol{n}_{n-m}]$; thus for any solution $\boldsymbol{x}$ there exist constants $\boldsymbol{\beta} = [\beta_1, \cdots, \beta_{n-m}]^T$ such that $\boldsymbol{x} = \boldsymbol{x}_p + \sum_{i=1}^{n-m} \beta_i \boldsymbol{n}_i$ for some particular solution $\boldsymbol{x}_p$ and vectors $\boldsymbol{n}_i \in N$. Then since $\boldsymbol{d}^k$ is a member of the null space of A we have a vector $\boldsymbol{c}^T = [c_1, \cdots, c_{n-m}]$ such that:

$$\boldsymbol{d}^k = \textstyle\sum_{i=1}^{n-m} c_i \boldsymbol{n}_i = N\boldsymbol{c},$$

$$\boldsymbol{d}^k_\Lambda = -\text{sgn}(\boldsymbol{x}^k_\Lambda) = N_\Lambda \boldsymbol{c},$$

$$\boldsymbol{c} = -N^\dagger_\Lambda \text{sgn}(\boldsymbol{x}^k_\Lambda),$$

$$\boldsymbol{d}^k_{\Lambda^c} = N_{\Lambda^c}\boldsymbol{c} = -N_{\Lambda^c} N^\dagger_\Lambda \text{sgn}(\boldsymbol{x}^k_\Lambda),$$

where N_Λ are the rows of N corresponding to Λ, and we have *chosen* $\boldsymbol{d}^k_\Lambda = -\text{sgn}\left(\boldsymbol{x}^k_\Lambda\right)$.

The matrix N_Λ is of size $|\Lambda| \times (n-m)$. Again the system is underdetermined in Part 1, except for the penultimate iterate of the algorithm, and again a heuristic choice is to be made in the instance in which $|\Lambda| < (n-m)$, this is considered in section 3.3.

A natural question to ask is whether the Null Space method or the Standard method should be used. It is fair to suppose that in the absence of any other consideration, one should choose to use the method that requires inversion of the smaller matrix. It is concluded superficially that for $n \leq 2m$ the null space scheme is preferable to the standard method since the matrix to be inverted is smaller.

3.1.3 Calculating the Step Length Parameter

By virtue of the selection of the $\boldsymbol{d}_\Lambda^k$ components of the direction vector it is clear that the step length α needs to satisfy a positivity constraint if the direction vector is indeed to reduce the infinity norm. We thus select α as the minimum positive value for which an element in Λ^c joins Λ. This occurs when $|\boldsymbol{x}_\Lambda^{k+1}(\alpha)| = |\boldsymbol{x}_j^{k+1}(\alpha)|$ for some $j \in \Lambda^c$. Since the absolute values of all the components of $\boldsymbol{x}_\Lambda$ are the same by design, we arbitrarily select a single representative component x_i for any $i \in \Lambda$, and solve for α as follows:

$$|x_i + \alpha d_i| = |x_j + \alpha d_j| \Rightarrow \; x_i + \alpha d_i = \pm(x_j + \alpha d_j),$$

$$\alpha = \tfrac{x_i - x_j}{d_j - d_i} \qquad \text{or} \qquad \alpha = \tfrac{(-x_i) - x_j}{d_j - (-d_i)},$$

where $j \in \Lambda^c$. We compute a different α for each $j \in \Lambda^c$ and form the set $\boldsymbol{\alpha} = \{\alpha_j : j \in \Lambda^c\}$, where we take α_j to be the smallest positive value from the two expressions above.

From the above set of scalars we select the smallest positive value to be our step length. The positivity constraint is discussed above as being necessary to decrease the infinity norm; we choose the minimum value so as to ensure that those components of the

current feasible solution corresponding to elements in the index set remain maximal.

To see why we select the smallest positive value from $\boldsymbol{\alpha}$, we shall consider an example. Suppose at iteration k we have $\boldsymbol{x}^k = [1, 0.5, -1, 0.3]$ and $\boldsymbol{d}^k = [-1, -2, 1, 1.5]$, clearly we have $\Lambda = \{1, 3\}$ and $\Lambda^c = \{2, 4\}$. Consider now the change in the absolute value of each component of the solution vector as we move in the direction $\boldsymbol{d}^k$.

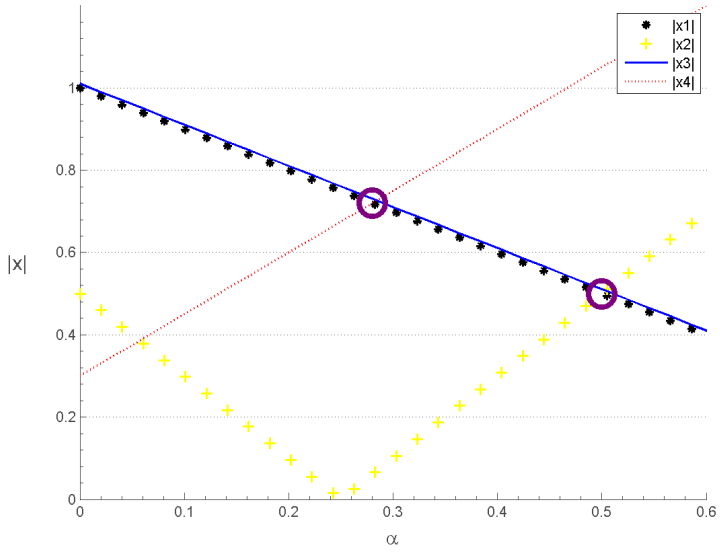


Figure 3-2: Elementwise propagation along $\boldsymbol{d}^k$ [1]

Making use of the procedure in the beginning of this section, one would compute the positive values of α to be $\boldsymbol{\alpha} = \{0.28, 0.50\}$ and indeed these are the intersection points at which components of the solution vector corresponding to elements currently in the index set are equal in absolute value to some component corresponding to an element currently in the complement - indicated in the figure above by magenta circles. Since each component of the solution vector is linearly dependent on the step length parameter it follows that, for values of $\alpha > \min(\boldsymbol{\alpha} > 0)$, those elements currently in the index set may no longer be so on the next iteration.

A natural question to ask is what value of α would result in the largest decrease in the infinity norm while still maintaining the current index set (Λ^k in Part 1 or $\Lambda^k \setminus i$ in Part 2). It is certainly possible that there exists $\alpha > \min(\boldsymbol{\alpha} > 0)$ such that

[1]The propagation of components of $\boldsymbol{x}_\Lambda$ have been intentionally slightly off-set for visual clarity.

$\Lambda^k \subset \Lambda^{k+1}$, and although such a selection for α may result in a greater decrease in the infinity norm, we do not stop to consider the details of such a procedure here.

3.2 Part 2 of the PPF Method

In this section, we consider once again the procedures by which the direction vector and step length parameter for the line search are determined. Part 2 of the PPF method refers to the situation when $|\Lambda| = n-(m-1)$. It has been noted previously that if the Standard Method, presented in section 3.1.2.1, is employed for the determination of the direction vector, then the system that must be solved for $\boldsymbol{d}^k_{\Lambda^c}$ has coefficient matrix of size $m \times (n - |\Lambda|)$. On the other hand, if the Null Space Method, presented in section 3.1.2.2, is employed the system that must be solve for $\boldsymbol{d}^k_{\Lambda^c}$ has coefficient matrix of size $|\Lambda| \times (n - m)$. In both cases the systems are underdetermined for Part 1 of the PPF method, except for the penultimate iterate in which the systems are square, and overdetermined for Part 2.

Conceptually, we proceed by removing one or more elements from the index set and solving the resulting system for the complement components of the direction vector. By removing an element from the index set, we mean to simply pretend it is not there so as to remove a constraint equation from the system to be solved for $\boldsymbol{d}^k_{\Lambda^c}$. The resulting system is, in both the Standard and Null Space methods, uniquely determined if only a single element is removed from the index set at a given iteration, and is underdetermined if multiple elements are removed. If the direction vector calculated in this fashion corresponds to a descent direction, then we perturb the current feasible solution $\boldsymbol{x}^k$ in the calculated direction.

When a single element is removed from the index set during Part 2, the direction vector is uniquely determined. However, the resulting direction vector may *not* correspond to a descent direction. The criteria for a direction vector to be descent may be expressed in terms of the components of the direction vector corresponding to

those elements removed from Λ as follows: suppose element j is to be removed from the index set, the resulting direction vector is a descent direction if and only if:

$$\left.\begin{array}{l} \text{sgn}\,(d_j^k) = -\text{sgn}\,(x_j^k), \\ \\ |d_j^k| > 1. \end{array}\right\} \tag{3.2}$$

The above conditions shall henceforth be referred to as the *Descent Conditions* and the j^{th} component of the direction vector shall be referred to as the j^{th} indicator - indicating whether or not the removal of that element results in a descent direction.

The first condition ensures that for a positive step length, $\alpha > 0$, the j^{th} component will in fact decrease, and the second condition ensures that the j^{th} component will decrease more rapidly than those components corresponding to elements in the index set; recall $\boldsymbol{d}_{\Lambda\setminus j}^k = -\text{sgn}\,(\boldsymbol{x}_{\Lambda\setminus j}^k) \Rightarrow |d_i^k| = 1\ \forall\, i \in \Lambda \setminus j$. This ensures that those components corresponding to elements in Λ remain maximal. This idea is readily generalised when multiple elements are removed from Λ, indeed each component corresponding to elements removed from Λ simply needs to satisfy the descent conditions individually.

Once a valid descent direction has been chosen the line search procedure is again carried out with the step length parameter being chosen precisely as in section 3.1.3. If no valid descent direction exists; that is to say no direction vector resulting from the removal of any element from the index set satisfies the descent conditions, then we are at the optimum and the algorithm is terminated with $\boldsymbol{x}_\infty^* = \boldsymbol{x}^k$.

A natural question to ask is which method should be used for the determination of the direction vector; the so-called Standard Method given in section 3.1.2.1 or the Null Space Method given in section 3.1.2.2. As mentioned previously it is fair to suppose that in the absence of any other consideration, one should choose to use the method that requires inversion of the smaller matrix.

Notice that in both of the above methods for the selection of the direction vector

a square matrix is required to be inverted in Part 2 of the algorithm after a single element has been removed from Λ whereafter $|\Lambda| = (n-m)$. We now show that the matrices that require inversion, namely A_{Λ^c} and N_Λ for the Standard and Null Space methods respectively, will have at each iteration, only a single row or column changed in the calculation of the indicators.

This follows since in Part 1 of the algorithm, when $|\Lambda| = n-m$, a single column will be removed from the matrix A_{Λ^c} and a single row will be added to the matrix N_Λ whereafter $|\Lambda| = n-(m+1)$. Thereafter a direction vector is to calculated in Part 2 of the algorithm, but firstly the indicators must be calculated. To compute an indicator, a single row or column will then be removed from these matrices and the resulting square system solved for $\boldsymbol{d}^k_{\Lambda^c}$.

By way of example, suppose that at iteration k we have $\Lambda^k = \{1, \cdots, n-m\}$, then we are required to compute $A^{-1}_{(\Lambda^c)^k}$ to obtain $\boldsymbol{d}_{\Lambda^c}$. Further suppose that at iteration $k+1$ we have $\Lambda^{k+1} = \{1, \cdots, n-m+1\}$. Then to obtain the indicators we are required to compute $A^{-1}_{(\Lambda^c)^{k+1}\setminus i}$ for some $i \in \Lambda \setminus n-m+1$. These two matrices that require inversion, namely $A_{(\Lambda^c)^k}$ and $A_{(\Lambda^c)^{k+1}\setminus i}$, clearly differ by only a single column (up to column permutation) - so too for any single element removed from the index set.

Furthermore, once a descent direction has been found, we line search in that direction until a new element joins the index set, where again $|\Lambda| = n-m+1$. Suppose then at iteration $k+2$ we have $\Lambda^{k+2} = \{2, \cdots, n-m+2\}$, again we are required to compute $A^{-1}_{(\Lambda^c)^{k+2}\setminus i}$ for some $i \in \Lambda \setminus n-m+2$. Thus, on any two consecutive iterations, the matrices requiring inversion differ by only a single row or column.

The inverse for these matrices may thus be expediently computed by appealing to a theorem by Cheney given below.

Theorem 1 (Matrix inverse update for a single row change)**.** *Let A be an $(n \times n)$ non-singular matrix and $\{C_1, \cdots, C_n\}$ be the columns of its inverse, $A^{-1} = [C_1, \cdots, C_n]$. Let $\tilde{A}$ be the matrix obtained by replacing the p^{th} row of A by the vector $\boldsymbol{g}$. If $C_p^T \cdot \boldsymbol{g} \neq 0$*

then $\tilde{A}$ is nonsingular and the columns of its inverse $\tilde{A}^{-1}$, are given by:

$$\tilde{C}_p = \frac{1}{C_p^T \boldsymbol{g}} C_p,$$

$$\tilde{C}_i = C_i - (C_i^T \boldsymbol{g})\tilde{C}_p \quad i \neq p.$$

While the above theorem gives an expedient formula for the inversion of a matrix with a single *row* replaced, it serves equally well if a single column is instead replaced. Notice that, $A^{-1}A = I \Rightarrow A^T(A^{-1})^T = I = A^T(A^T)^{-1} \Rightarrow (A^T)^{-1} = (A^{-1})^T$, therefore, considering the replacement of a column of A as the replacement of a row of A^T, we may all but use the theorem in its current form.

3.2.1 Expedient Computation of Indicators

As discussed above, the criteria for a direction vector to be a descent direction can be given in terms of the component of the direction vector corresponding to the element to be removed from Λ - the indicators as described by equation (3.2). These conditions must be checked for each element of the current index set.

Suppose that element j is chosen to exit the index set; the direction vector will be a descent direction if and only if sgn $(d_j^k) = -$sgn (x_j^k) and $|d_j^k| > 1$ - the Descent Conditions. The first condition demands that the resulting direction will indeed decrease x_j, the second condition demands that the element will decrease more rapidly than those in the index set. We consider the determination of the j^{th} component of the direction vector by means of the Standard Method, presented in section 3.1.2.1, as:

$$\boldsymbol{d}^k_{\Lambda^c} = A^{-1}_{\Lambda^c} A_\Lambda \text{sgn}(\boldsymbol{x}^k_\Lambda),$$

$$(\boldsymbol{d}^k_{\Lambda^c})_j = \left(A^{-1}_{\Lambda^c} A_\Lambda \text{sgn}(\boldsymbol{x}^k_\Lambda)\right)_j = \left(A^{-1}_{\Lambda^c}\right)_j A_\Lambda \text{sgn}(\boldsymbol{x}^k_\Lambda).$$

where $(A^{-1}_{\Lambda^c})_j$ is the j^{th} row of the matrix $A^{-1}_{\Lambda^c}$ corresponding to the replacement of the j^{th} column of A_{Λ^c}. Since only one column is changed on each iteration, the matrix is, as noted previously, amenable to expedient inverse by **Theorem 1**. Moreover we have an explicit formula for the j^{th} row of the inverse. Let $C = A^{-1}_{\Lambda^c}$ before the replacement of element j^* by j. The j^{th} component of the direction vector may then be computed as:

$$(\boldsymbol{d}^k_{\Lambda^c})_j = \left(\frac{1}{C^{j^*} A_j} C^{j^*}\right) A_\Lambda \text{sgn}(x^k_\Lambda),$$

where C^{j^*} is the *row* of C corresponding to j^*, j is the element selected to leave the index set and j^* is the last element to have joined the index set on the previous iteration. This follows from the fact that $(A^T)^{-1} = (A^{-1})^T$, as noted above. We may thus obtain $A^{-1}_{\Lambda^c} = ((A^{-1}_{\Lambda^c})^T)^T = ((A^T_{\Lambda^c})^{-1})^T$.

We shall refer to the heuristic choice to be made in Part 1 of the algorithm in determining $\boldsymbol{d}^k_{\Lambda^c}$ as the Ascent method (AM). We shall also refer to the heuristic choice to be made in Part 2 of the algorithm in the selection of the direction vector, as the Exchange method (EM). Finally, in the instance that a multiple element exchange procedure is selected, the index set may need to be rebuilt many times and a further heuristic choice is to be made following the line search in which a diminished index set remains. The options for this choice are the same as those for the Ascent method - this heuristic choice is therefore known as the Transitional Ascent method (TAM). The algorithmic details and discussion of the PPF method can be found in section 3.5.

3.3 Ascent Heuristics

We now focus our attention on the first part of the algorithm in which a solution is iteratively constructed until $|\Lambda| = n - (m-1)$. As previously noted, the linear system we are required to solve in order to obtain the components of the direction vector associated with Λ^c, see section 3.1, is underdetermined for Part 1 of the algorithm, except for the penultimate iterate. One option is to, at each iteration, compute the minimum ℓ_2 norm solution by means of the well known Moore-Penrose pseudo inverse. Another is to simply zero-off an appropriate number of columns, and to solve the resulting square system. We consider both possibilities in this section.

Throughout this section we shall, for illustrative purposes only, consider the determination of the direction vector by means of the so-called Standard Method; the ideas carry freely over to the Null Space Method.

3.3.1 Moore-Penrose Pseudo-Inverse

Suppose we elect to, at each iteration, compute the Moore-Penrose pseudo-inverse as follow:

$$A_{\Lambda^c}\boldsymbol{d}^k_{\Lambda^c} = A_{\Lambda}\text{sgn}(\boldsymbol{x}^k_{\Lambda}),$$

$$\boldsymbol{d}^k_{\Lambda^c} = A^T_{\Lambda^c}(A_{\Lambda^c}A^T_{\Lambda^c})^{-1}A_{\Lambda}\text{sgn}\,(\boldsymbol{x}^k_{\Lambda}).$$

It is well-known that the Moore-Penrose pseudo-inverse produces the least squares, or minimum ℓ_2 norm solution [10]. The computation is, in general, $\mathcal{O}(n^3)$. This would, for large systems, be much slower then the Iterative Ascent procedure outlined in the following section. In our particular case, however, we can make use of the fact that only a single column is removed from the matrix A_{Λ^c} at each iteration, as a new element enters the index set, to expedite the computation of the new pseudo-inverse. As will be seen shortly, this amounts to a rank-1 update procedure.

Suppose we have $A \in \mathbf{R}^{m\times n}$ and $B = (A^TA)^{-1}$. Further, let us construct $\tilde{A} = [A, \boldsymbol{v}]$, where $\boldsymbol{v} \in \mathbf{R}^{m\times 1}$: we wish to determine $\tilde{B} = (\tilde{A}^T\tilde{A})^{-1}$. What follows is taken from [14]. Consider the following decomposition and inverse of the partitioned matrix:

$$\tilde{B} = \begin{bmatrix} A^TA & A^T\boldsymbol{v} \\ \boldsymbol{v}^TA & \boldsymbol{v}^T\boldsymbol{v} \end{bmatrix}^{-1} = \begin{bmatrix} F_{11}^{-1} & -dBA^T\boldsymbol{v} \\ -d\boldsymbol{v}^TAB^T & d \end{bmatrix},$$

where,

$$d = \frac{1}{\boldsymbol{v}^T\boldsymbol{v} - \boldsymbol{v}^TABA^T\boldsymbol{v}}$$

$$F_{11}^{-1} = B + dBA^T\boldsymbol{v}\boldsymbol{v}^TAB^T.$$

The inverse of a partitioned matrix as above has been known for some time. The above procedure essentially outlines a rank-1 update when a column is added to the matrix A, a similar update is easily deduced when a column is instead removed from the matrix A by interchanging the roles of B and $\tilde{B}$. This update will prove crucial in establishing the computational viability of this heuristic choice.

Importantly, the update procedure extracted from the decomposition above is not limited to the addition or deletion of the *last* column of a matrix. If another column is added or deleted the same rank-1 update procedure can be used with only the addition of a column permutation. The details of such an implementation are given in [14].

3.3.2 Iterative Ascent

An alternative to computing the pseudo-inverse of an underdetermined system at each iteration is to instead adopt a different conceptual approach. Instead of working 'down' from the full underdetermined system $A^0_{\Lambda^c}$ to iteratively smaller, more square systems $A^k_{\Lambda^c}$, we may instead build 'up' the index set by initially selecting m columns

of A and forming the effective A_{Λ^c}, and at each iteration introducing a single new column of A to the effective A_{Λ^c} and removing a column previously in A_{Λ^c}, as one previous element of the effective Λ^c joins Λ. This is equivalent to zeroing-off the requisite number of columns and simply solving the resulting square system. Such a procedure shall be referred to as *Iterative Ascent*.

Following from equation (3.1) we have,

$$A_{\Lambda^c}\boldsymbol{d}^k_{\Lambda^c} = A_\Lambda \text{sgn}\,(\boldsymbol{x}^k_\Lambda),$$

where $A_{\Lambda^c} \in \mathbf{R}^{m\times(n-|\Lambda|)}$. To make the system square we are required, at each iteration, to set $n-m-|\Lambda|$ components of $\boldsymbol{d}_{\Lambda^c}$ to zero. Let Φ be the subset of those elements of Λ^c whose corresponding components are set to zero; the direction vector is as follows:

$$\boldsymbol{d}^k_\Lambda = -\text{sgn}\,(\boldsymbol{x}^k_\Lambda), \quad \boldsymbol{d}^k_\Phi = 0, \quad \boldsymbol{d}^k_{\Lambda^c\setminus\Phi} = A^{-1}_{\Lambda^c\setminus\Phi}\left(A_\Lambda \text{sgn}\,(\boldsymbol{x}^k_\Lambda)\right).$$

For $A_{\Lambda^c\setminus\Phi}$ to be nonsingular for all Φ, the matrix A is required to satisfy the Haar condition.

We may, of course, set any $n-m-|\Lambda|$ components of $\boldsymbol{d}^k_{\Lambda^c}$ to zero. It seems reasonable to choose those elements of Λ^c corresponding to either the smallest (IA-small), or largest (IA-big) components of the current solution vector $\boldsymbol{x}^k$ in terms of absolute value. Indeed both of these possibilities are considered in the results section 4.1.1, and are therein identified as 'IA-small' and 'IA-big' respectively. Moreover, specific components may be set to zero if need be. Notice that those components of the current solution vector corresponding to the components of the direction vector set to zero, will clearly not change value during the line search.

Once a direction vector has been computed, the step length parameter is once again found in the manner described in section 3.1.3, and a line search carried out resulting in one more element joining the index set. On the next iteration one fewer component

is to be set to zero. If we keep at zero all but one of the previously zero components, then the index of that single components no longer set to be zero joins $\Lambda^c \setminus \Phi$. This amounts to a single column change to $A_{\Lambda^c \setminus \Phi}$ and the computation of the inverse is thus amenable to expedient inversion by **Theorem 1**.

3.4 Exchange Heuristics

In Part 2 of the PPF method, one or more elements are required to be removed from the index set in order for a direction vector to be calculated. The conditions under which a direction vector would be a descent direction were given in equation (3.2), and it was shown that not every element, when removed from Λ would necessarily result in a descent direction.

In this section we examine the heuristic choices that can be made regarding the direction vector selection in Part 2 of the PPF method. To these ends, we shall first define the set of indices corresponding to valid indicators for a given direction vector as follows:

$$\Omega = \{j \in \Lambda \ : \ \boldsymbol{d}^k_{\Lambda^c \cup j} = A^{-1}_{\Lambda^c \cup j} \left(A_{\Lambda \setminus j} \text{sgn} \left(\boldsymbol{x}^k_{\Lambda \setminus j} \right) \right) \ , \ |d^k_j| > 1, \text{sgn} \left(d^k_j \right) = -\text{sgn} \left(x^k_j \right) \}.$$

The set Ω may be thought of as the set of indices which, when removed from Λ, result in a descent direction - as calculated by means of equation (3.1).

3.4.1 Single Element Exchange

If $|\Omega| = \phi$ then no valid descent direction exists and the algorithm is terminated. If $|\Omega| = 1$, then clearly that element must be chosen to exit the index set since there is only one viable descent direction. If however $|\Omega| > 1$ then multiple descent directions

exist, a heuristic choice must be made as to which element should exit the index set.

We may, for example, choose the element to exit the index set to be that associated with the valid indicator which is largest in absolute value (Big), i.e. the largest $|d_j|$ for $j \in \Omega$; the intuition being that even for a small value of the step length parameter α, the element $|x_j|$ will decrease substantially. Alternatively we may choose the element to exit the index set to be that associated with the valid indicator smallest in absolute value (Small), i.e. the smallest $|d_j|$ for $j \in \Omega$; the intuition in this case being that a larger value of α will need to be selected before d_j^k is no longer a maximal component, and the objective function may observe a greater decrease.

These heuristics are tested in the results section, see section 4.1.1, where these single element exchange heuristics are identified as 'Small' and 'Big' respectively.

3.4.2 Multiple Element Exchange

If $|\Omega| > 1$ then an alternative heuristic may be implemented. Rather than selecting which single element should be removed from the index set, we select multiple elements to be removed simultaneously; we do so in such a fashion as to guarantee that the resulting direction vector is a descent direction. Explicitly, each component of the direction vector corresponding to those elements to be removed from Λ will satisfy the descent conditions given in equation (3.2).

Of course there are many ways in which a descent direction may be computed that removes multiple elements from the index set. We shall consider only two such methods that, in different ways, remove all elements from Λ corresponding to valid indicators.

Let us suppose that $\Omega = \{u, v, w, \dots\}$. Then by the descent condition expressed above we know that there exists a set of direction vectors, $D = [\boldsymbol{d}^u, \boldsymbol{d}^v, \boldsymbol{d}^w, \cdots] \in \text{Null}\,(A)$ such that $\boldsymbol{d}^u_{\Lambda \setminus u} = -\text{sgn}\,(\boldsymbol{x}_{\Lambda \setminus u})$, $|d^u_u| > 1$ and $\text{sgn}\,(d^u_u) = -\text{sgn}\,(x_u)$, for each $\{u, v, w, \cdots\} \in \Omega$. Note that in the above, the super-index denotes the index of the vector in Ω, where as the sub-index denotes the coordinates of the component within

that vector: i.e. d_v^u is the v^{th} component of the u^{th} vector in Ω.

We wish to determine whether a nontrivial linear combination of the vectors in D may be constructed in such a way as to represent a descent direction. Explicitly we wish to determine whether constants, $[c_1, c_2, \cdots]^T$ may be determined such that $\boldsymbol{d}^* = c_1\boldsymbol{d}^u + c_2\boldsymbol{d}^v + c_3\boldsymbol{d}^w + \cdots = D\boldsymbol{c}$, where $\boldsymbol{d}^*_{\Lambda\setminus\Omega} = -\text{sgn}\,(\boldsymbol{x}_{\Lambda\setminus\Omega})$ and $\text{sgn}\,(\boldsymbol{d}^*_\Omega) = -\text{sgn}\,(\boldsymbol{x}_\Omega)$ and $|d_u^*| > 1\ \forall u \in \Omega$.

Firstly note that the obvious selection, $c_1 = c_2 = \cdots = 1$ almost works, only the normalization condition is violated for $\boldsymbol{d}^*_{\Lambda\setminus\Omega}$. To see that a simple scaling will remedy the situation observe that,

$$\boldsymbol{d}^*_{\Lambda\setminus\Omega} = (1+1+1+\cdots)(-\text{sgn}\,(\boldsymbol{x}_{\Lambda\setminus\Omega})) = -\text{sgn}\,(\boldsymbol{x}_{\Lambda\setminus\Omega})|\Omega|,$$

where clearly a scaling by $1/|\Omega|$ will renormalize the direction vector. Consider now those components of $\boldsymbol{d}^*_\Omega$,

$$\boldsymbol{d}^*_\Omega = \boldsymbol{d}^u_\Omega + \boldsymbol{d}^v_\Omega + \boldsymbol{d}^w_\Omega + \cdots = \begin{bmatrix} (|\Omega|-1)(-\text{sgn}\,(x_i)) + d_u^u \\ (|\Omega|-1)(-\text{sgn}\,(x_j)) + d_v^v \\ \vdots \end{bmatrix},$$

and since by assumption we have, $|d_u^u| > 1$ and $\text{sgn}\,(d_u^u) = \text{sgn}\,(d_u^{\Omega\setminus u})$ we therefore conclude that $|\boldsymbol{d}^*_\Omega| > |\boldsymbol{d}^*_{\Lambda\setminus\Omega}|$. Therefore the selection $c_i = \frac{1}{|\Omega|}$ for $i \in \Omega$, indeed produces a descent direction in which all the elements in Ω decrease faster than those in $\Lambda \setminus \Omega$. This method shall be referred to as *IntOnes*, the name stems from the fact that the direction vector represents an *interior*, multiple element exchange, and that the coefficients in the linear combination are selected to be *ones*. Note that such a choice of direction vector results in an index set that is not full, $|\Lambda| \leq (n-m)$, after the line search procedure has been carried out. The implications of this fact will be discussed

shortly.

We ask now whether it is possible to determine constants $\boldsymbol{c} = \{c_i\}$ such that the absolute value of those elements in $\boldsymbol{d}_\Omega$ is equal, $|\boldsymbol{d}^*_\Omega| = \beta > 1$, where $\beta \in \mathbf{R}$ and all the elements in Ω decrease equally during the line search. For an arbitrary selection of constants c_i we have,

$$\boldsymbol{d}^*_{\Lambda\setminus\Omega} = \left(\sum_{i=1}^{|\Omega|} c_i\right)(-\operatorname{sgn}\boldsymbol{x}_{\Lambda\setminus\Omega}),$$

and the normalization condition on $\boldsymbol{d}^*_{\Lambda\neq\Omega}$is satisfied for the scaling $1/(\sum_i c_i)$ provided, $(\sum_i c_i) > 0$. In order to obtain a uniform decrease amongst components of $\boldsymbol{d}^*$ corresponding to elements in Ω we are required to solve,

$$D_\Omega \boldsymbol{c} = -\beta \operatorname{sgn}(\boldsymbol{x}_\Omega),$$

where D_Ω denotes the rows of D corresponding to elements in Ω. Noticed that various choices for β result only in scaled solutions to $\boldsymbol{c}$. This system is underdetermined by a single constraint owing to the presence of the variable β. This is not a problem though, as an additional constraint is implicitly imposed by the normalization condition: namely, that the magnitude of components of the descent direction vector corresponding to those elements in $\Lambda \setminus \Omega$ are equal to 1. Consider any representative row, α of $\Lambda \setminus \Omega$, where α may be selected randomly. Suppose instead of performing a scaling transform later we demand immediately that $D_\alpha \boldsymbol{c} = -\operatorname{sgn}(x_\alpha)$. The resulting system,

$$\begin{bmatrix} D_\Omega & \operatorname{sgn}(\boldsymbol{x}_\Omega) \\ D_\alpha & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \beta \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\operatorname{sgn}(x_\alpha) \end{bmatrix},$$

is of size $(|\Omega|+1) \times (|\Omega|+1)$ and permits a unique solution. Importantly, since a solution will satisfy $D_\alpha \boldsymbol{c} = -\operatorname{sgn}(x_\alpha)$ for the representative row α, we have that $D_{\Lambda\setminus\Omega}\boldsymbol{c} = -\operatorname{sgn}(\boldsymbol{x}_{\Lambda\setminus\Omega})$. The only requirement for $\boldsymbol{d}^*$ determined in this manner to

be a descent direction is that $\beta > 1$. The value for β calculated numerically has always obeyed $\beta > 1$ although the theoretical justification for this fact is still under investigation. This method shall be referred to as *IntEqual*, the name stems from the fact that the direction vector is an *interior*, multiple element selection, and the coefficients in the linear combination are chosen so as to make the critical components of the resulting direction vector, corresponding to elements in Ω, *equal* to each other.

For the choice of direction vector as $\boldsymbol{d}^*$ a full index set will not be maintained after line search. This has two important consequences. Firstly, the direction vector at the following iteration will need to be determined as a heuristic choice, as it was for Part 1 of the PPF method. Secondly, the inverse square matrix required in Part 2 will no longer be amenable to expedient update by **Theorem 1**.

It is proposed that the direction vector at the next iteration may be computed using ideas expressed in the 'Iterative Ascent' method outlined in section 3.3.2. Explicitly, $|\Omega|-1$ components of $\boldsymbol{d}_{\Lambda^c}$ are set to zero and the resulting square system is solved - we choose those $|\Omega|-1$ components to be the components of $\boldsymbol{d}_{\Lambda^c}$ corresponding to the elements which were driven from Λ on the previous iteration. This Transitional Ascent Method (TAM) shall be referred to as (IAold) - since those components being set to zero correspond to those *old* members of Λ.

Since there were $|\Omega|$ elements removed from the index set at the previous iteration, but we only require $|\Omega|-1$ components of the $\boldsymbol{d}_{\Lambda^c}$ to be set to zeros, clearly one such element cannot correspond to a zero components in $\boldsymbol{d}_{\Lambda^c}$. If we were to set all components corresponding to elements is Ω to zero, the system to be solved for $\boldsymbol{d}_{\Lambda^c}$ would be overdetermined. The choice as to which single component, corresponding to an element in Ω, is not to be set to zero is to be made randomly.

The intuition behind such a choice for the direction vector selection is based on the idea that if no element were ever to reenter the index set the PPF method would converge very rapidly indeed. Specifically there would be an upper bound of $m-1$ iterations in Part 2 of the algorithm. This is a powerful restriction as in practice

elements may exit and reenter the index set multiple times.

The choice of direction vector above will guarantee that, on the following iteration, at least $|\Omega|-1$ of those elements previously in Λ will remain in Λ^c or they will all reenter the index set simultaneously. It is noted for completeness that the idea to remove more than one element at a time from the index set was proposed in [7] for both the ℓ_1 and ℓ_∞ norm solutions to the overdetermined problem. The reported convergence results were superior to classical exchange algorithms in which only a single element is exchanged at each iteration.

3.5 The PPF Algorithm

In this section, the implementation of the PPF method is discussed. The algorithm may be initialized in many different ways, all that is required is that an initial solution to the underdetermined system be found. Throughout this thesis we have assumed that the initial solution is the minimum ℓ_2 norm solution obtained from the Moore-Penrose pseudo-inverse, $\boldsymbol{x}^0 = \boldsymbol{x}_2^*$, although many alternatives exist.

Once an initial solution has been found, the initial index set Λ is computed that comprises the indices of the components of the initial feasible solution that are largest in absolute value. If the initial index set just so happens to contain at least $n-(m-1)$ elements, then Part 1 of the PPF method is not required.

If the number of elements in the initial index is fewer than $n-(m-1)$, then we begin Part 1 of the PPF method. Firstly a valid descent direction must be calculated, and secondly the step length in that direction must be computed. The system to be solved for the descent direction vector is underdetermined in this part of the method, except for the penultimate iterate, and a heuristic choice is to be made here. We refer to this choice as the Ascent Method (AM). Two such choices within AM, outlined in the corresponding sections above, are the Moore-Penrose pseudo-inverse and the Iterative Ascent procedure. Once a direction vector has been computed, the step

length is calculated as per section (3.1.3).

At this point the new iterate $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k$ is computed and the index set updated. If the number of elements in the index set is still less than $n - (m - 1)$, then the procedure outlined for Part 1 of the PPF is repeated. If however the index set contains at least $n - (m - 1)$ elements, we move to Part 2 of the PPF method. With the full rank assumption in place the index set must contain exactly $n-(m-1)$ elements at optimum.

In Part 2 of the PPF method a descent direction must again be found, and a step length in that direction computed. In order for a descent direction to be found, one or more elements currently in the index set must be removed. If no descent direction can be found, then we are at the optimum and the algorithm terminates.

When one element is removed from the index set, the resulting descent direction is uniquely determined. The process of determining whether or not the resulting direction is in fact a descent direction is known as the process of computing the indicators.

If no valid indicators exist then we are at the optimum. If a single valid indicator exists then clearly that element must be removed from the index set. If however multiple valid indicators exist then a choice must be made as to whether multiple elements should be simultaneously removed from the index set or whether a single element should be removed. In both cases there are numerous potential choices to be made - we call this choice the Exchange Method (EM). Some potential Exchange Methods outlined in the corresponding section above include the single element exchange heuristics 'Small' and 'Big', and the multiple element exchange heuristics, 'IntOnes' and 'IntEqual'.

Once a direction vector has been computed in Part 2 of the PPF method, the step length is computed again as per section (3.1.3), and the new iterate calculated as $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k$. The index set is then updated. If the number of elements in the index set is fewer than $n - (m - 1)$, as it would be for multiple element exchange

procedures, then we essentially move back into Part 1 of the algorithm. The same heuristic choices now present themselves. To distinguish the heuristic choice here from that which began the process we refer to this heuristic as the Transitional Ascent Method (TAM). Of course if the number of elements in the index set remains at $n-(m-1)$, as it would for single element exchange procedures, then the TAM is rendered void.

The PPF algorithm is presented, in Algorithm 1, in the form of pseudo-code.

Algorithm 1 The PPF Method

Require: $\boldsymbol{x}_0 = \boldsymbol{x}_2^*$, $k = 0$, Λ, (EM, AM, TAM)
while Not At Optimum **do**
  if $|\Lambda| < n-(m-1)$ **then**
    Compute: $\boldsymbol{d}^k$ - by AM, and α^k - as per section (3.1.3)
    Update: $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k$, Λ^{k+1}, $k = k+1$
  else
    Compute Indicators
    if No Valid Indicators **then**
      $\boldsymbol{x}_\infty^* = \boldsymbol{x}^k$
      STOP
    else
      Compute: $\boldsymbol{d}^k$, α^k by EM
      Update: $\boldsymbol{x}^{k+1} = \boldsymbol{x}^{k+1} + \alpha^k \boldsymbol{d}^k$, Λ^{k+1}, $k = k+1$
      if EM = Multiple Element Exchange **then**
        AM = TAM
      end if
    end if
  end if
end while

3.6 Computational Complexity of the PPF Method

In this section we consider the computational complexity of the PPF method with all heuristic possibilities in mind. The PPF method proceeds in two logical parts. In Part 1 a feasible solution is obtained for which $n-(m-1)$ components are equal in absolute value and maximal. In Part 2 the location of those components is exchanged by line search in such a way as to reduce the ℓ_∞ norm at each step. In Part 1 of the algorithm both a direction vector and a step length must be calculated at each iteration. In general $n-m$ iterations are needed during this part of the algorithm since the initial index set comprises, in general, just a single element, i.e. the index of the maximal element, and only a single element is added to the index set at each iteration. In Part 2 of the algorithm the additional step of determining which element to remove from the now full index set must be performed, after which a direction vector and a step length are again calculated.

The PPF method as described above presents a number of different procedures by which an initial full index set may be obtained, and by which a direction vector may be calculated. The computational complexity of each of these heuristic choices is determined in isolation.

We consider the complexity of the determination of the step length parameter, as per section 3.1.3, separately since the procedure is the same for both Part1 and Part 2 of the PPF method.

Recall that the step length α is computed as:

$$\alpha = \tfrac{x_i - x_j}{d_j - d_i} \quad \text{or} \quad \alpha = \tfrac{(-x_i) - x_j}{d_j - (-d_i)}.$$

where i is any representative element of the index set Λ and $j \in \Lambda^c$. The determination of all the possible α values requires $4|\Lambda^c|$ additions and $2|\Lambda^c|$ multiplications. We are then required to determine the minimum positive value of the resulting unsorted

array. In total, just $2|\Lambda^c|$ multiplications are required and this step of the method has complexity of $\mathcal{O}(|\Lambda^c|)$.

3.6.1 Part 1

In this section, the computational complexity of the determination of the direction vector in Part 1 of the PPF method is established. The heuristic choices for the Ascent Method are discussed in section 3.3. Furthermore, it is noted in section 3.1.2 that the direction vector may be computed using either the 'Standard' or 'Null Space' scheme regardless of the heuristic choice made. All such possible combinations are examined.

3.6.1.1 Iterative Ascent (IA)

If the so-called *Standard Method* is employed, then the direction vector is computed as:

$$\boldsymbol{d}_{\Lambda}^{k} = -\text{sgn}\left(\boldsymbol{x}_{\Lambda}^{k}\right) \quad \text{and} \quad \boldsymbol{d}_{\Lambda^c}^{k} = -\left(A_{\Lambda^c}\right)^{-1} A_{\Lambda}\boldsymbol{d}_{\Lambda}^{k} = \left(A_{\Lambda^c}\right)^{-1} A_{\Lambda}\text{sgn}\left(\boldsymbol{x}_{\Lambda}^{k}\right),$$

where we have $|\Lambda^c| = m$ for each iteration. It is precisely the fact that the size of the complement of the index set remains constant in this approach that makes it computationally efficient. At each iteration Λ obtains, in general, just a single new element. Suppose that element i were to join the index set on iteration k; we have:

$$A_{\Lambda}\boldsymbol{d}_{\Lambda}^{k+1} = A_{\Lambda\setminus i}\text{sgn}\left(\boldsymbol{x}_{\Lambda\setminus i}^{k+1}\right) + A_i\text{sgn}\left(x_i\right).$$

The update for A_{Λ}^{k+1} therefore requires only m additions per iteration.

Since only a single column of A_{Λ^c} changes at each iteration we may compute $\left(A_{\Lambda^c}\right)^{-1}$ expediently by **Theorem** 1. This requires just $2m^2$ multiplications [9]. The matrix-

vector product $(A_{\Lambda^c})^{-1}(A_\Lambda \text{sgn}\,(\boldsymbol{x}_\Lambda^k))$ requires a further m^2 multiplications.

In total then a single iteration of this procedure requires $3m^2$ multiplications and this method of computation of the direction vector in Part 1 is $\mathcal{O}(m^2)$.

If alternatively, the so-called *Null Space Method* is employed, then the direction vector is computed as follows:

$$\boldsymbol{d}_\Lambda^k = \sum_{i=1}^{n-m} c_i \boldsymbol{n}_{\Lambda_i} = N_\Lambda \boldsymbol{c} = -\text{sgn}\,(\boldsymbol{x}_\Lambda^k)$$

$$\boldsymbol{d}_{\Lambda^c}^k = \sum_{i=1}^{n-m} c_i \boldsymbol{n}_{\Lambda_i^c} = N_{\Lambda^c} \boldsymbol{c},$$

where N is the matrix whose columns are a set of minimal spanning vectors for the null space, and N_Λ denotes the rows of N corresponding to the current index set Λ. Notice that $N_\Lambda \in \mathbf{R}^{|\Lambda| \times (n-m)}$ and so the system to be solved for the coefficient vector $\boldsymbol{c}$ is underdetermined in Part 1 of the algorithm, except for the penultimate iterate, at which point the system is uniquely determined.

The Iterative Ascent procedure works by zeroing-off an appropriate number of columns of A and solving the resulting square system. To zero-off a column of A is equivalent to setting the corresponding component of the direction vector to zero. In the context of the *Null Space Method*, this results in a greater number of constraint equations on the coefficient vector $\boldsymbol{c}$ resulting ultimately in a square system. Let us define Φ, as before, to be the set of elements of the complement index set corresponding to those components of the direction vector that were set to zero, $\Phi = \{i \in \Lambda^c \ : \ \boldsymbol{d}_i^k = 0\}$. We then have:

$$N_{\Lambda \cup \Phi} \boldsymbol{c} = \begin{bmatrix} \\ \boldsymbol{d}_\Phi \\ \\ \boldsymbol{d}_\Lambda \\ \\ \end{bmatrix} \quad \rightarrow \quad \boldsymbol{c} = (N_{\Lambda \cup \Phi})^{-1} \begin{bmatrix} \\ \mathbf{0} \\ \\ -\text{sgn}\,(\boldsymbol{x}_\Lambda) \\ \\ \end{bmatrix},$$

where $N_{\Lambda\cup\Phi} \in \mathbf{R}^{(n-m)\times(n-m)}$ and only a single row is changed on each iteration, as a new element of Λ^c joins Λ from the line search. The inverse is therefore amenable to expedient computation by **Theorem 1**, and involves $2(n-m)^2$ multiplications [9]. The matrix-vector product for the determination of the coefficient vector $\boldsymbol{c}$ requires a further $(n-m)^2$ multiplications, and finally the product $\boldsymbol{d}_{\Lambda^c} = N_{\Lambda^c}\boldsymbol{c}$ requires $m(n-m)$ multiplications since $N_{\Lambda^c} \in \mathbf{R}^{m\times(n-m)}$.

In totality then the determination of the direction vector in Part 1 by the *Null Space Method* requires $3(n-m)^2+m(n-m)$ multiplications and is of order $\mathcal{O}(n^2+m^2-mn)$.

3.6.1.2 Pseudo-Inverse (PI)

If the *Standard Method* is employed for the determination of the direction vector, the computational complexity is to be determined from the following equation:

$$\boldsymbol{d}^k_{\Lambda^c} = A^T_{\Lambda^c}(A_{\Lambda^c}A^T_{\Lambda^c})^{-1}A_\Lambda \mathrm{sgn}\,(\boldsymbol{x}^k_\Lambda).$$

Since Λ obtains, in general, only a single additional element on each iteration, by the argument above, the update for $A_\Lambda \mathrm{sgn}\,(\boldsymbol{x}^k_\Lambda)$ requires only m additions. Furthermore, since only a single element is removed from Λ^c on each iteration, only a single column of A_{Λ^c} is removed, and the product $(A_{\Lambda^c}A^T_{\Lambda^c})^{-1}$ may be expediently computed by the rank-1 update given in section 3.3.1. Notice that if more than one element were to join the index set on a given iteration, the rank-1 update should simply be used in its current form repeated.

The rank-1 update procedure requires $|\Lambda^c|(|\Lambda^c|-1)$ multiplications - this is easily deduced from the explicit algorithmic form of the update given by Khan [14].

The final two matrix-vector products each require a further m^2 multiplications. In totality then, one iteration of this procedure requires $|\Lambda^c|(|\Lambda^c|-1)+2m^2$ multiplications and is, at worst, of order $\mathcal{O}(n^2+m^2)$ since $\max|\Lambda^c| = n-1$.

If the *Null Space Method* is employed then the direction vector may be computed as:

$$\boldsymbol{d}_{\Lambda}^{k} = N_{\Lambda}\boldsymbol{c} = -N_{\Lambda}\text{sgn}\left(\boldsymbol{x}_{\Lambda}^{k}\right) \quad ; \quad \boldsymbol{d}_{\Lambda^c}^{k} = N_{\Lambda^c}\boldsymbol{c},$$

and in terms of the pseudo-inverse the coefficient vector is computed as,

$$\boldsymbol{c} = -N_{\Lambda}^{T}(N_{\Lambda}N_{\Lambda}^{T})^{-1}\text{sgn}\left(\boldsymbol{x}_{\Lambda}^{k}\right).$$

We again make use of the fact that Λ changes by only a single element on each iteration and employ the rank-1 update for $(N_{\Lambda}N_{\Lambda}^{T})^{-1}$ given in section 3.3.1. Since $N_{\Lambda} \in \mathbf{R}^{|\Lambda|\times(n-m)}$ the update requires $|\Lambda|(|\Lambda|-1)$ multiplications. Furthermore the remaining two matrix-vector products require $2(|\Lambda|)^2$ multiplications.

In total then, this method of computing the direction vector in Part 1 of the PPF is, at worst, of order $\mathcal{O}(n^2 + m^2 - nm)$, since $|\Lambda|$ is at most equal to $(n - m - 1)$ before the system to be solved for $\boldsymbol{d}_{\Lambda^c}$ becomes uniquely determined.

3.6.2 Part 2

In this section the computational complexity of the determination of the direction vector in Part 2 of the PPF method is established. Both single and multiple element exchange heuristics are considered, these being developed in section 3.4.1 and section 3.4.2 respectively. Firstly however, the expedient determination of indicators is considered as this is required by all exchange heuristics considered in this thesis - indeed the termination condition is derived from consideration of the indicators.

The indicators may be computed as per section 3.2.1 as follows:

$$d_j = \left(\frac{1}{C^{j^*}A_j}C^{j^*}\right)A_{\Lambda}\text{sgn}\left(\boldsymbol{x}_{\Lambda}\right) = BA_{\Lambda}\text{sgn}\left(\boldsymbol{x}_{\Lambda}\right),$$

where j is the element under consideration to leave the index set, and j^* is the

previous element to have joined the index set. For this portion of the method we have $A_\Lambda \in \mathbf{R}^{m\times(n-m)}, C \in \mathbf{R}^{m\times m}$ and $C = A_{\Lambda^c}^{-1}$ before j^* was replaced by j.

As per previous arguments, since Λ changes by only a single element at each iteration we require only additions to compute the updated product $A_\Lambda \text{sgn}\,(\boldsymbol{x}_\Lambda)$. To see this explicitly let us assume that at a given iteration element i exits and element j enters Λ. We then have,

$$A_\Lambda^{k+1}\text{sgn}\,(\boldsymbol{x}_\Lambda^{k+1}) = A_{\Lambda\setminus i}^{k+1}\text{sgn}\,(\boldsymbol{x}_{\Lambda\setminus i}^{k+1}) + A_j\text{sgn}\,(x_j)$$

$$= \left(A_\Lambda^k\text{sgn}\,(\boldsymbol{x}_\Lambda^k) - A_i\text{sgn}\,(x_i)\right) + A_j\text{sgn}\,(x_j),$$

and it is clear that the full matrix-vector product need not be calculated. The vector-vector products $(C^{j^*}A_j)$ and $B(A_\Lambda\text{sgn}\,(\boldsymbol{x}_\Lambda^k))$ require m multiplications each.

In total then, computing all the possible values for d_j requires $2m(n-m)$ multiplications since there are $(n-m)$ elements in Λ to be check at each iterations, and the order of complexity is $\mathcal{O}(nm-m^2)$.

3.6.2.1 Single Element Exchange

Once again, the direction vector may be computed by either the *Standard Method* or the *Null Space Method*.

We consider first the *Standard Method*, for which the direction vector is determined as:

$$\boldsymbol{d}_{\Lambda^c}^k = (A_{\Lambda^c})^{-1}A_\Lambda\text{sgn}\,(\boldsymbol{x}_\Lambda^k)$$

Once again, since Λ changes by a single element on each iteration, only additions are required for the update of $A_\Lambda\text{sgn}\,(\boldsymbol{x}_\Lambda^{k+1})$. Moreover, even an initial computation of

the matrix-vector product may not be required as its value is provided from the Part 1 of the algorithm if the Iterative Ascent heuristic is used.

Inversion of A_{Λ^c} requires $2m^2$ multiplications as the inverse is computed by use of **Theorem 1**. The matrix-vector product $(A_{\Lambda^c})^{-1}(A_\Lambda \text{sgn}\,(\boldsymbol{x}_\Lambda^k))$ requires a further m^2 multiplications. In totality $3m^2$ multiplications are required for each iteration of this procedure. The procedure is thus of order $\mathcal{O}(m^2)$

The direction vector is determined using the *Null Space Method* as follows:

$$\boldsymbol{d}_\Lambda^k = \sum_{i=1}^{n-m} c_i \boldsymbol{n}_{\Lambda_i} = N_\Lambda \boldsymbol{c} = -\text{sgn}\,(\boldsymbol{x}_\Lambda^k) \;\rightarrow\; \boldsymbol{c} = -(N_\Lambda)^{-1}\text{sgn}\,(\boldsymbol{x}_\Lambda^k),$$

$$\boldsymbol{d}_{\Lambda^c}^k = \sum_{i=1}^{n-m} c_i \boldsymbol{n}_{\Lambda_i^c} = N_{\Lambda^c} \boldsymbol{c} = -N_{\Lambda^c}(N_\Lambda)^{-1}\text{sgn}\,(\boldsymbol{x}_\Lambda^k).$$

During Part 2 of the algorithm we have $N_\Lambda \in \mathbf{R}^{(n-m)\times(n-m)}$. At each iteration one *row* of N_Λ is changed and so by **Theorem 1** we may obtain expediently $(N_\Lambda^T)^{-1}$.

The inversion of N_Λ by **Theorem 1** requires $2(n-m)^2$ multiplications since $N_\Lambda \in \mathbf{R}^{(n-m)\times(n-m)}$. The matrix-vector product $N_\Lambda^{-1}\text{sgn}\,(\boldsymbol{x}_\Lambda^k)$ requires $(n-m)^2$ multiplications and the matrix-vector product $N_{\Lambda^c}(N_\Lambda^{-1}\text{sgn}\,(\boldsymbol{x}_\Lambda^k))$ requires a further $m(m-n)$ multiplications.

In total then this procedure requires $3(n-m)^2+m(n-m)$ multiplications per iteration and is of order $\mathcal{O}(n^2+m^2-nm)$.

3.6.2.2 Multiple Element Exchange

In this section we consider the complexity of the determination of the direction vector by the multiple element exchange procedures developed in section 3.4.2. While the numerical results for the two specific multiple element exchange procedures developed in this thesis are shown to be inferior to the single element exchange procedure 'Small', there is certainly further investigation to be done into multiple element exchange

procedures: specifically those in which not every element corresponding to a valid indicator is removed from the index set.

For the multiple element exchange procedure IntOnes only a single linear system of size $m \times (m-1)$ needs to be solved, and a single matrix-vector product computed, for the determination of the direction vector. Note that although the system is overdetermined a solution is guaranteed to exist as shown in the corresponding section 3.4.2. Using the notation from section 3.4.2, we have:

$$\boldsymbol{d}^k_{\Lambda\setminus\Omega} = -\text{sgn}\,(\boldsymbol{x}_{\Lambda\setminus\Omega}), \quad \boldsymbol{d}^k_{\Omega} = \begin{bmatrix} \\ (|\Lambda|-1)(-\text{sgn}\,(x_i)) + d^u_u \\ \\ (|\Lambda|-1)(-\text{sgn}\,(x_j)) + d^v_v \\ \\ \vdots \end{bmatrix}, \quad \boldsymbol{d}^k_{\Lambda^c} = (A_{\Lambda^c})^{-1} A_{\Lambda} \boldsymbol{d}^k_{\Lambda}.$$

The matrix-vector product, $A_\Lambda \boldsymbol{d}^k_\Lambda$ requires $m(n-m+1)$ multiplications since $|\Lambda| = n-m+1$ and the solution of the linear system for $\boldsymbol{d}^k_{\Lambda^c}$ requires, $m(m-1)^2$ multiplications - where the order of complexity of linear system inversions has been taken generically to be that of Gauss-Jordan elimination. In totality then this procedure is of order $\mathcal{O}(m^3)$, although this may be improved by more advanced techniques for the solution to uniquely determined linear systems.

If the multiple element exchange procedure IntEqual is employed, then the solution to two linear systems of equations is required. Since we are first required to solve:

$$\begin{bmatrix} \\ D_\Omega & \text{sgn}\,(\boldsymbol{x}_\Omega) \\ \\ D_\alpha & 0 \end{bmatrix} \begin{bmatrix} \\ \boldsymbol{c} \\ \\ \beta \end{bmatrix} = \begin{bmatrix} \\ \mathbf{0} \\ \\ -\text{sgn}\,(x_\alpha) \end{bmatrix},$$

for $\left[\boldsymbol{c}^T, \beta\right]^T$, this requires $(|\Omega|+1)^3$ multiplications for the matrix inversion by Gauss-Jordan elimination, and a further $(|\Omega|+1)$ multiplication for the matrix-vector product. This follows from the fact that all but one component is zero on the right hand side of the above equation. We then have:

$$\boldsymbol{d}^k_{\Lambda\setminus\Omega} = -\text{sgn}\left(\boldsymbol{x}^k_{\Lambda\setminus\Omega}\right), \quad \boldsymbol{d}^k_{\Omega} = -\beta\text{sgn}\left(\boldsymbol{x}^k_{\Omega}\right), \quad \boldsymbol{d}^k_{\Lambda^c} = (A_{\Lambda^c})^{-1} A_{\Lambda} \boldsymbol{d}^k_{\Lambda}$$

where the solution for $\boldsymbol{d}^k_{\Lambda^c}$ requires $m(m-1)^2 + m(n-m+1)$ multiplications. Theoretically $|\Omega|$ may be as large as $n-m$ and the total complexity of this method would be of order $\mathcal{O}(n^3 - n^2 m + nm^2)$.

For both of the multiple element exchange procedures a further consideration arises. Once the full index set has been destroyed as multiple elements exit Λ, the index set must be rebuilt by the chosen Transitional Ascent method. Regardless of the heuristic choice here, either $(A_{\Lambda^c})^{-1}$ or $(N_{\Lambda})^{-1}$ will need to be explicitly computed once before a lower order update procedure can be applied.

Owing to the increased asymptotic complexity of the multiple exchange procedures, the improved number of iterations to convergence will have to be great for these method to be competitive at a large scale. Otherwise a lower order procedure will need to be developed.

The results for the computational complexity of the various heuristics for the PPF method have been tabulated for comparison and easy reference in Table 3.1.

Depending on the choice of heuristics in the PPF method, the computational complexity compares well with current best methods. It is in fact superior for some combination of heuristic choices in Part 1 and Part 2. A lower bound for the number of multiplications per iteration for Cadzow's method is $3m^2 + 2mn + n$ [6].

	Procedure	Method	Order	Multiplications
Part 1 & 2	α		$\mathcal{O}(\lvert\Lambda^c\rvert)$	$2\lvert\Lambda^c\rvert$
Part 1	IA	Standard	$\mathcal{O}(m^2)$	$3m^2$
		Null	$\mathcal{O}(n^2+m^2-nm)$	$3(n-m)^2+m(n-m)$
	PI	Standard	$\mathcal{O}(n^2+m^2)$	$\lvert\Lambda^c\rvert(\lvert\Lambda^c\rvert-1)+2m^2$
		Null	$\mathcal{O}(n^2+m^2-nm)$	$\lvert\Lambda\rvert(\lvert\Lambda\rvert+1)+2(\lvert\Lambda\rvert)^2$
Part 2	Indicators		$\mathcal{O}(nm-m^2)$	$2m(n-m)$
	Single	Standard	$\mathcal{O}(m^2)$	$3m^2$
		Null	$\mathcal{O}(n^2+m^2-nm)$	$3(n-m)^2+m(n-m)$
	Multiple	IntOnes	$\mathcal{O}(m^3)$	$m(m-1)^2+m(n-m+1)$
		IntEqual	$\mathcal{O}(n^3-n^2m+nm^2)$	$(\lvert\Omega+1\rvert)^3+(\lvert\Omega\rvert+1)+m(m-1)^2+m(n-m+1)$

Table 3.1: Summary of complexity analysis of the PPF method

3.7 Geometric Heuristic for Initial Active Constraint Selection

A natural question to ask is why Part 1 of the algorithm cannot be avoided entirely by arbitrarily selecting $n-(m-1)$ elements to be in the index set. The answer is twofold. Firstly the feasible solution, while valid, may lie far from the optimum both in terms of the distance between the feasible solution and the optimum in some norm, and in terms of the difference between their index sets. Secondly it may simply be impossible for some set of the elements selected to be in the index set in the solution space. To illustrate this point an example is considered below,

$$\min \ ||\boldsymbol{x}||_{\infty} \text{ subject to } \begin{bmatrix} 5 & 3 & -1 & 3 \\ 0 & -4 & 4 & 5 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 2 \\ -5 \end{bmatrix},$$

where, due to the dimension of the constraint matrix $A \in \mathbf{R}^{2\times 4}$, and the relevant non-degeneracy requirements being satisfied, we know an optimal solution exists with $n-(m-1)=3$ components equal in absolute value and maximal. Suppose we guess the 'signed' index set to be $\boldsymbol{x}_{\Lambda} = \{-x_1, -x_2, +x_3\}$; the unique solution to the system may then be computed as in the equation below, where z represents the absolute value of the components with indices in the selected index set.

$$\begin{bmatrix} 5 & 3 & -1 & 3 & 0 \\ 0 & -4 & 4 & 5 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ -5 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The solution to the above system is $[\boldsymbol{x}^T, z]^T = [0.36, 0.36, -0.36, -0.42, 0.36]$ and it is clear that such a selection for the index set does *not* result in the components which are equal in absolute value being maximal. The minimum infinity norm solution to this system is actually 0.38 and corresponds to the 'signed' index set $\boldsymbol{x}_\Lambda = \{+x_2, -x_3, -x_4\}$.

Much is to be gained from the development of criteria for the selection of the initial active set of constraints. Indeed exclusion criteria would prove equally useful in providing a theoretical basis for heuristic choices regarding which elements should enter the index set in both Part 1 and Part 2 of the PPF algorithm. In the work to follow such criteria will be develop based on geometric considerations specific to problem (1.2).

3.7.1 Hyperplane bounding

We begin this section by seeking to find all points in $\mathbf{R}^2$ such that $|x_1| \geq |x_2|$. Consider the figure below:

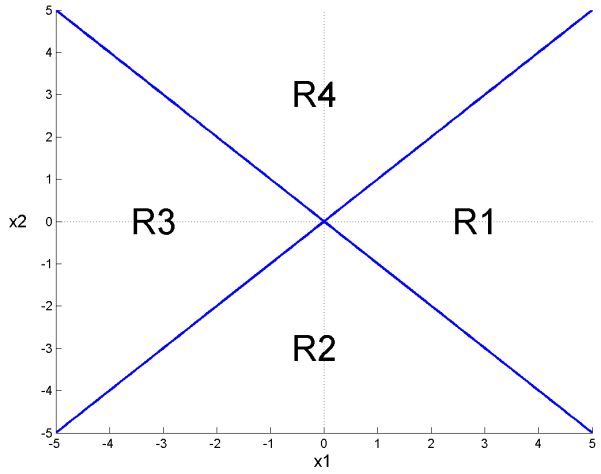


Figure 3-3: Regions in $\mathbf{R}^2$

The two lines in Figure 3-3 have gradient equal to ± 1. Clearly all points in R_1 and R_3 satisfy $|x_1| \geq |x_2|$. Similarly, all points in R_2 and R_4 satisfy $|x_2| \geq |x_1|$. To facilitate the discussion to follow we make two useful definitions. Firstly,

Definition 2 (Signed Index Set)**.** *A signed index set* $\hat{\Lambda}$ *specifies both the indices of components maximal in absolute value and their signs. Explicitly then:*

$$\hat{\Lambda} = \{\pm i \ : \ |x_i| = ||\boldsymbol{x}||_\infty , \pm x_i \geq 0\}.$$

By way of example, we have:

$$\hat{\Lambda} = \{+1, -2, +4\} \Rightarrow \{|x_1| = |x_2| = |x_4| = ||\boldsymbol{x}||_\infty , +x_1 \geq 0, -x_2 \geq 0, +x_4 \geq 0\}.$$

Secondly, we define:

Definition 3 (A region)**.** *A region* $R\left(\hat{\Lambda}\right)$ *is the set of all points in* $\mathbf{R}^n$ *in which the signed index set* $\hat{\Lambda}$ *holds.*

By way of example, we have in $\mathbf{R}^2$:

$$R(\{-2\}) = \{\boldsymbol{x} \in \mathbf{R}^2 \ : \ |x_2| = ||\boldsymbol{x}||_\infty, -x_2 \geq 0\},$$

and considering an example in $\mathbf{R}^3$,

$$R(\{+1, -3\}) = \{\boldsymbol{x} \in \mathbf{R}^3 \ : \ |x_1| = |x_3| = ||\boldsymbol{x}||_\infty, +x_1 \geq 0, -x_3 \geq 0\}.$$

With this new notation in mind we reconsider the regions in Figure 3-3 above and observe,

$$R_1 = R(\{+1\}),$$

$$R_2 = R(\{-2\}),$$

$$R_3 = R(\{-1\}),$$

$$R_4 = R(\{+2\}).$$

These regions may be explicitly characterised by their bounding hyperplanes as follows:

$$R_1 \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad R_2 \rightarrow \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$R_3 \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad R_4 \rightarrow \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The description of regions by their bounding hyperplanes, as above, corresponds to an H-space representation of an infinite convex polytope.

For geometric clarity we note in passing that the single element regions, i.e. those for which $|\hat{\Lambda}|= 1$, are those hypervolumes from the origin 'through' the faces of the ℓ_∞ norm level curves.

There is a natural generalization of the formalism presented above to higher dimensions. In 3-dimensions, for example, the region $R(\{+1\})$ is the pyramidal volume from the origin through a face of the origin-centered cubes orientated to align with the coordinate axes - i.e. a face of the ℓ_∞ norm level curves in $\mathbf{R}^3$. The H-space description also permits a natural extension to higher dimensions.

It follows both intuitively and as a direct result of the H-space description, that single element regions are infinite convex polytopes. Furthermore, we notice that $R\left(\hat{\Lambda}\right) = \cap_{i=1}^{|\hat{\Lambda}|} R\left(\hat{\Lambda}_i\right)$, and therefore all regions are convex, regardless of dimension. This follows since the intersection of convex polytopes is a convex polytope. Also, all regions as defined above are infinite.

We consider the dimension of the region $R\left(\hat{\Lambda}\right)$ to be $n + 1 - |\hat{\Lambda}|$. Notice that when $|\hat{\Lambda}|= n$ we have the dimension of $R\left(\hat{\Lambda}\right) = 1$ - this corresponds to the case in which all components are equal in absolute value. Geometrically, this is the line from the

origin through the appropriate corner of the hypercube (the ℓ_∞ norm level curve) specified by the signs of the elements in $\hat{\Lambda}$.

We now wish to consider regions in the solution space. Let H denote the solution space to the linear system under consideration, i.e. $H = \{\boldsymbol{x} \ : \ A\boldsymbol{x} = \boldsymbol{b}\}$; we wish to probe the topological structure of $H \cap R\left(\hat{\Lambda}\right)$.

We immediately notice two important properties of the intersection space $H \cap R\left(\hat{\Lambda}\right)$. Firstly, the convexity of the intersection space follows directly from the convexity of the solution space, which we prove below.

Theorem 2. *Convexity of the Solution Space*

The solution space to a linear system $A\boldsymbol{x} = \boldsymbol{b}$ is a convex set.

Proof 1 (Convexity of the Solution Space)**.** *Let $\boldsymbol{x}, \boldsymbol{y} \in H$, then $A\boldsymbol{x} = \boldsymbol{b}$ and $A\boldsymbol{y} = \boldsymbol{b}$. Clearly then, $\lambda A\boldsymbol{x} = \lambda\boldsymbol{b}$ and $(1-\lambda)A\boldsymbol{y} = (1-\lambda)\boldsymbol{b}$, $\forall\lambda \in \mathbf{R}$.*

It follows that $\lambda A\boldsymbol{x} + (1-\lambda)A\boldsymbol{y} = \lambda\boldsymbol{b} + (1-\lambda)\boldsymbol{b}$, and $A(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) = \boldsymbol{b}$, and so $\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in H$.

□

Secondly, the topological structure of $H \cap R\left(\hat{\Lambda}\right)$ is invariant under nonzero scaling to $\boldsymbol{b}$. Suppose $H \cap R\left(\hat{\Lambda}\right) \neq \phi$ and $\boldsymbol{x} \in R\left(\hat{\Lambda}\right)$, then:

For $\boldsymbol{b}' \to \lambda\boldsymbol{b}, \lambda > 0$ the transformation $\boldsymbol{x}' \to \lambda\boldsymbol{x}$ ensures $A\boldsymbol{x}' = \boldsymbol{b}'$ and $\boldsymbol{x}' \in R\left(\hat{\Lambda}\right)$.

For $\boldsymbol{b}' \to \lambda\boldsymbol{b}, \lambda < 0$ the transformation $\boldsymbol{x}' \to \lambda\boldsymbol{x}$ ensures $A\boldsymbol{x}' = \boldsymbol{b}'$ and $\boldsymbol{x}' \in R\left(-\hat{\Lambda}\right)$.

While all regions were infinite in the embedding space $\mathbf{R}^n$, they are not necessarily so in the solution space. This idea is illustrated in Figure 3-4 below.

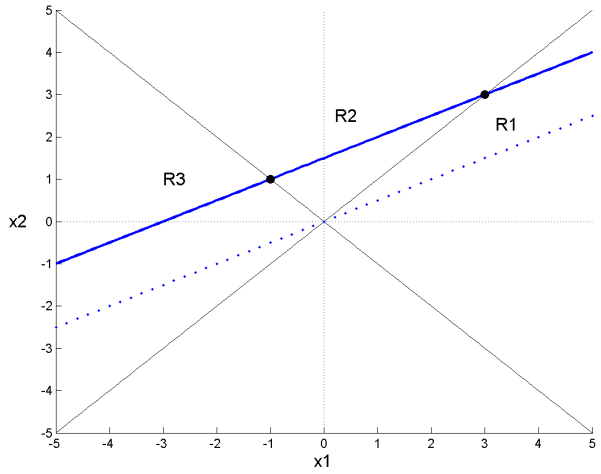


Figure 3-4: Regions in the solution space in $\mathbf{R}^2$

The blue line represents the solution space to the equation $\left[-\frac{1}{2}, 1\right] \cdot \boldsymbol{x} = \frac{3}{2}$. Clearly the regions $H \cap R\left(\{+1\}\right)$ and $H \cap R\left(\{-1\}\right)$ are infinite where as $H \cap R\left(\{+2\}\right)$ is finite, and $H \cap R\left(\{-2\}\right)$ is empty.

By the convexity of all regions we may classify a region as being infinite if and only if there exists $\boldsymbol{d} \in \mathbf{R}^n$ and $N \in \mathbf{R}^+$, such that, $\boldsymbol{x} + \alpha \boldsymbol{d} \in R\left(\hat{\Lambda}\right)$, $\forall \alpha > N$ and $\boldsymbol{x} \in H \cap R\left(\pm\hat{\Lambda}\right)$.

Notice how the dimension of the intersection space corresponds with the geometrical explanation of the PPF method in Figure 3-1. Consider a point at which $n-(m-1)$ components of the feasible solution are equal in absolute value and maximal. Such a point corresponds to a full index set $|\hat{\Lambda}| = n-(m-1)$, and $Dim\left(H \cap R\left(\hat{\Lambda}\right)\right) = (n-m)+1-(n-(m-1)) = 0$ - this is a vertex in Figure 3-1, and follows from the dimension of the null space for full rank A (the rank-nullity theorem). The geometrical picture has a natural extension.

We wish to show that a region in the solution space is infinite if and only if the same region is infinite in the null space. Thereafter we conclude that if a region is not infinite in the null space, it is finite or empty in the solution space.

Theorem 3 (Infinite Regions)**.** *A region $R\left(\hat{\Lambda}\right)$ is infinite in the solution space of a linear system $A\boldsymbol{x} = \boldsymbol{b}$, if and only if the region is infinite in the null space of A.*

Proof 2 (Theorem 3)**.** *This proof will rely on the classification of an infinite region as one in which there exists a valid direction, in which one can move arbitrarily far while still remaining in the region.*

$\Rightarrow$ *Assume $R\left(\hat{\Lambda}\right)$ is a non-empty infinite region in the null space.*

Then there exists some $\boldsymbol{d} \in Null(A)$ such that, $|d_i| = |d_j| \ \forall \ i, j \in \hat{\Lambda}, sgn(d_i) = sgn(x_i) \ \forall \ i \in \hat{\Lambda}$ and $|d_{\hat{\Lambda}}| > |d_{\hat{\Lambda}^c}|$.

$\Rightarrow R\left(\hat{\Lambda}\right)$ *is a non-empty infinite region in the solution space, since for any particular solution to the linear system $\boldsymbol{x}_p \in H \cap R\left(\pm\hat{\Lambda}\right)$, there exists some number N such that for all $\alpha > N$ we have, $\boldsymbol{x}_p + \alpha\boldsymbol{d} \in H \cap R\left(\hat{\Lambda}\right)$. This follows from the fact that $|d_{\hat{\Lambda}}| > |d_{\hat{\Lambda}^c}|$.*

$\Leftarrow$ *Assume that $R\left(\hat{\Lambda}\right)$ is a non-empty infinite region in the solution space H.*

Then similarly there exists some $\boldsymbol{d} \in Null(A)$ such that, $|d_i| = |d_j| \ \forall \ i, j \in \hat{\Lambda}, sgn(d_i) = sgn(x_i) \ \forall \ i \in \hat{\Lambda}$ and $|d_{\hat{\Lambda}}| > |d^c_{\hat{\Lambda}}|$.

$\Rightarrow R\left(\hat{\Lambda}\right)$ *is an infinite region in the null space of A, since for any $\alpha > 0$ we have $\alpha\boldsymbol{d} \in R\left(\hat{\Lambda}\right)$.*

□

It is immediately clear that all infinite regions are paired in terms of their signed index sets, i.e. if $H \cap R\left(\hat{\Lambda}\right) \neq \phi$ and is an infinite region then, $H \cap R\left(-\hat{\Lambda}\right) \neq \phi$ and is also an infinite region. Notice how this corresponds with the topological invariance of the intersection space under nontrivial scaling of $\boldsymbol{b}$.

An element is called bounded if it is not present in the index set of any infinite region. Alternatively put, an element is called bounded if the corresponding component is never maximal in an infinite region. Such a component is also referred to as being

bounded. By the theorem above, we may study boundedness in the null space. It turns out that this is a more tractable approach.

We pause to notice that while different feasible solutions may belong to different regions in the solution space, the topological structure of regions in the solutions space is completely determined by A and $\boldsymbol{b}$.

Theorem 4 (Bounding conditions on a single hyperplane)**.** *Let H be a hyperplane given by $H : \boldsymbol{\alpha}^T \cdot \boldsymbol{x} = b$. A component x_p is bounded on H if and only if $\sum_{i \neq p} \frac{|\alpha_i|}{|\alpha_p|} < 1$.*

Proof 3 (Theorem 4)**.** *We consider the element x_p of $\boldsymbol{x} \in \mathbf{R}^n$*

$$\boldsymbol{\alpha}^T \cdot \boldsymbol{x} = 0 \;\Rightarrow\; |x_p| = \frac{|-\sum_{i \neq p}^{n} \alpha_i x_i|}{|\alpha_p|}$$

$\Rightarrow$ *Assume that* $|\alpha_p| > \sum_{i \neq p}^{n} |\alpha_i|$

$$|x_p| \leq \sum_{i \neq p}^{n} \frac{|\alpha_i||x_i|}{|\alpha_p|} \qquad \textit{Triangle Inequality}$$

$$\frac{|x_p|}{|x_k|} \leq \sum_{i \neq p}^{n} \frac{|\alpha_i|}{|\alpha_p|} < 1 \quad \textit{where } |x_k| = max\{|x_i|\}, \forall i \neq p$$

$$\therefore |x_k| > |x_p|.$$

$\Leftarrow$ *Assume that x_p is bounded on H i.e. $\forall\, \boldsymbol{x} \in \mathbb{R}^n$, such that $A\boldsymbol{x} = \boldsymbol{b}$, $\exists\, j \neq p$ such that $|x_p| < |x_j|$. Note also that on a hyperplane we may make arbitrary selection of $n-1$ components*

$let\, x_i = sgn(\alpha_i), \forall i \neq p$

$|x_p| = \sum_{i \neq p}^{n} \frac{|\alpha_i|}{|\alpha_p|} < 1 \qquad$ *by assumption, since* $|x_i| = 1\ \forall i \neq p$

$\therefore \sum_{i \neq p}^{n} |\alpha_i| < |\alpha_p|.$

□

To illustrate the proof above we consider a simple example. Consider the hyperplane in two dimensions given by $H : \boldsymbol{\alpha}^T \cdot \boldsymbol{x} = 0 \ \rightarrow [3, 1] \cdot \boldsymbol{x} = 0$ - this is a straight line in two dimensional passing through the origin with gradient $m = -3$. We have that $|\alpha_1| > \sum_i |\alpha_i| = |\alpha_2|$. We therefore conclude, by the above results, that x_1 is bounded. Since we are in the null space we conclude that there exists no point on the hyperplane H for which x_1 is maximal. Clearly we have $|x_1| = \frac{|x_2|}{|3|}$ and x_1 can never be maximal.

To see the application of **Theorem 4** in practice a small example is considered in which the solution to problem (1.2) is sought, subject to,

$$\begin{bmatrix} 3 & 1 & \mathbf{-17} & 5 & -1 & 2 \\ 6 & -4 & 5 & 0 & 4 & -5 \\ -4 & -2 & -3 & \mathbf{30} & 3 & 3 \\ 4 & 0 & 5 & -4 & 5 & \mathbf{-29} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 2 \\ 3 \\ 2 \\ -1 \end{bmatrix}.$$

The ℓ_∞ solution to the above is $\boldsymbol{x}^T = [0.278, -0.278, -0.077, 0.039, 0.278, 0.102]$, we have therefore $\Lambda^* = \{1, 2, 5\}$. By inspection we observe that an element is bounded on rows $1, 3$ and 4 of A. Those elements correspond to components $\{x_3, x_4, x_6\}$ - this observation is typical. The heuristic suggested by this example and conceptually validated by numerical testing is that bounded elements do not appear in abundance

in the index set at the optimal solution.

The above proof demonstrates the conditions under which an element is bounded on a single hyperplane. It is obvious that if an element is bounded on a hyperplane then it is clearly also bounded on any particular portion of that plane. The solution space is the intersection of the hyperplanes specified by the rows of A. Thus if any row of the matrix A satisfies the condition given above then that element will certainly be bounded in the solution space. One may also note that at most one element can be bounded on a hyperplane. A simple proof by contradiction is considered below. Suppose that two components x_p and x_q are bounded on the hyperplane $H : \boldsymbol{\alpha}^T \cdot \boldsymbol{x} = 0$; this implies that,

$$|\alpha_p| > \sum_{i \neq p}^{n} |\alpha_i| = |\alpha_q| + \sum_{i \neq p,q} |\alpha_i| > |\alpha_q|$$

$$\Rightarrow \; |\alpha_p| > |\alpha_q|,$$

on the other hand,

$$|\alpha_q| > \sum_{i \neq q}^{n} |\alpha_i| = |\alpha_p| + \sum_{i \neq p,q} |\alpha_i| > |\alpha_p|$$

$$\Rightarrow \; |\alpha_q| > |\alpha_p|,$$

and we obtain the desired contradiction.

At this point we would like to motivate some further investigation into bounded elements. Firstly it should be acknowledged that given both the elements in the index set at the optimum and their signs, a solution may be immediately determined as the system to be solved is reduced to a square system. An example of such a procedure was observed at the start of this section. To see this in the general case,

note that given the signed index set at optimum we may write:

$$\begin{bmatrix} A_\Lambda & A_{\Lambda^c} & \mathbf{0} \\ I & 0 & -\text{sgn}\,(\boldsymbol{x}_\Lambda) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_\Lambda \\ \boldsymbol{x}_{\Lambda^c} \\ z \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \mathbf{0} \end{bmatrix},$$

where $z = ||\boldsymbol{x}||_\infty$. Moreover, knowing that an element cannot possibly be in the index set at the optimum or indeed that it must be in the index set at the optimum provides a powerful means for determining the elements that should enter and exit the index set in both Part 1 and Part 2 of the PPF algorithm. The benefit of such knowledge is only compounded as the dimension of the system increases.

The condition described in **Theorem 4** for when an element is bounded on a single hyperplane is quite restrictive; as such we seek to investigate more general settings than a single hyperplane.

We now consider the necessary conditions upon which an element is bounded in the intersection of two hyperplanes. A standard parameterization of the intersection space in terms of $n-2$ components shall be considered, i.e. we write two components as a function of the remaining $n-2$ which then serve as a parameterization of the intersection space.

Theorem 5 (Bounding condition for the intersection of two hyperplanes)**.** *Let H and K be hyperplanes given respectively by: $\boldsymbol{\alpha}^T \cdot \boldsymbol{x} = b_1$ and $\boldsymbol{\beta}^T \cdot \boldsymbol{x} = b_2$. Let $H \cap K$ be parameterized in the standard fashion by $\boldsymbol{y} = \{\boldsymbol{x}_{i \neq \{p,q\}}\}$ such that $x_p = f(\boldsymbol{y})$ and $x_q = g(\boldsymbol{y})$. Then a component x_p is bounded on $H \cap K$ if $\sum_{i \neq \{p,q\}}^{n} \frac{|\alpha_q \beta_i - \beta_q \alpha_i|}{|\alpha_p \beta_q - \alpha_q \beta_p|} < 1$.*

The proof follows in a similar fashion to that for a single hyperplane above. Again we exploit the observation that an element is bounded on a hyperplane if and only if it is bounded in the null space for the plane.

Proof 4 (Theorem 5)**.** *We again consider the component x_p of $\boldsymbol{x} \in \mathbb{R}^n$. We may write the system resulting from the intersection as follows:*

$$H \cap K \;\rightarrow\; \begin{bmatrix} \\ \boldsymbol{\alpha}^T \\ \\ \boldsymbol{\beta}^T \\ \\ \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \\ 0 \\ \\ 0 \\ \\ \end{bmatrix}$$

Cramer's rule then gives:

$$x_p = \sum_{i \neq \{p,q\}}^{n} \frac{(\alpha_q \beta_i - \beta_q \alpha_i) x_i}{\alpha_p \beta_q - \alpha_q \beta_p}$$

$\Rightarrow$ *Assume* $\sum_{i \neq \{p,q\}}^{n} \frac{|\alpha_q \beta_i - \beta_q \alpha_i|}{|\alpha_p \beta_q - \alpha_q \beta_p|} < 1$

$$|x_p| \leq \sum_{i \neq \{p,q\}}^{n} \frac{|\alpha_q \beta_i - \beta_q \alpha_i||x_i|}{|\alpha_p \beta_q - \alpha_q \beta_p|} \quad \textit{Triangle Inequality}$$

$$\frac{|x_p|}{|x_k|} \leq \sum_{i \neq \{p,q\}}^{n} \frac{|\alpha_q \beta_i - \beta_q \alpha_i|}{|\alpha_p \beta_q - \alpha_q \beta_p|} \quad \textit{where } |x_k| = max\,\{|x_i|\}, \forall i \neq p, q$$

$$\therefore \frac{|x_p|}{|x_k|} < 1 \quad \textit{by assumption}$$

and so x_p is bounded.

$\Leftarrow$ *Assume x_p is bounded by $\{\boldsymbol{y}\}$, that is to say $\forall\, \boldsymbol{x} \in \mathbb{R}^n$ such that $A\boldsymbol{x} = \boldsymbol{b}$, $\exists\; j \neq \{p, q\}$ such that $|x_p| < |x_j|$. Since we may arbitrarily select values for $n-2$ components on the intersection space we take:*

$$x_i = sgn(\alpha_q \beta_i - \beta_q \alpha_i), \forall i \neq p, q$$

$$x_p = \sum_{i \neq \{p,q\}}^{n} \frac{|\alpha_q \beta_i - \beta_q \alpha_i|}{|\alpha_p \beta_q - \alpha_q \beta_p|} < 1$$

The last inequality follows from the fact that x_p is bounded by $\{\boldsymbol{y}\}$. □

The above proof gives a necessary and sufficient condition for x_p to be bounded by $\{\boldsymbol{y}\}$. It is conceivable however that x_p may still be bounded on $H \cap K$ but just not by that particular $(n-2)$ components subset. If one could show that if x_p was not bounded by *any* $n-2$ parameter subset then x_p was infinite, then the above proof would constitute an *if and only if* condition. It is trivially noted that interchanging the roles of p and q in the proof we obtain a condition for the bounding of the second dependent element in terms of the chosen $n-2$ components.

Corrolary 1. *Let H and K be hyperplanes given respectively by: $\boldsymbol{\alpha}^T \cdot \boldsymbol{x} = b_1$ and $\boldsymbol{\beta}^T \cdot \boldsymbol{x} = b_2$. Then a component x_q is bounded on $H \cap K$ if $\sum_{i \neq \{p,q\}}^{n} \frac{|\alpha_p \beta_i - \beta_p \alpha_i|}{|\alpha_q \beta_p - \alpha_p \beta_q|} < 1$.*

The above proofs contain interesting theoretical results, but one may be concerned that in their current form they may not be computationally viable. With particular reference to **Theorem 5**, one should note that in order to check if an element is bounded in the intersection of two hyperplanes all $n-2$ component subsets must be checked. There are $n-1$ such subsets. This must then be done for each element. We now consider an observation that may help alleviate this computational burden.

Proposition 1. *An element is bounded on the intersection of two hyperplanes if and only if there exists a third hyperplane containing that intersection, upon which the element is bounded.*

A proof of the above proposition will allow us to probe the question of boundedness in a different way. Instead of *checking* whether or not an element is bounded, we may now ask whether or not there exists a *construction* which will bound an element. This is a conceptual adjustment which may prove fruitful.

Proof 5 (Proposition 1)**.** *Let H and K be hyperplanes given respectively by: $\boldsymbol{\alpha}^T \cdot \boldsymbol{x} = 0$ and $\boldsymbol{\beta}^T \cdot \boldsymbol{x} = 0$. Every hyperplane L containing $H \cap K$ has a normal expressible as*

$\boldsymbol{\gamma}^T \cdot \boldsymbol{x} = (c_1\boldsymbol{\alpha}^T + c_2\boldsymbol{\beta}^T) \cdot \boldsymbol{x} = 0.$

$\Rightarrow$ *Assume that* x_1 *is bounded on* L

$$\boldsymbol{x} \in (H \cap K) \Rightarrow \boldsymbol{\alpha}^T \cdot \boldsymbol{x} = 0 \text{ and } \boldsymbol{\beta}^T \cdot \boldsymbol{x} = 0 \Rightarrow (c_1\boldsymbol{\alpha}^T + c_2\boldsymbol{\beta}^T) \cdot \boldsymbol{x} = 0$$

$$\therefore (H \cap K) \subset L.$$

Which implies that x_1 *is bounded on* $H \cap K$.

$\Leftarrow$ *Assume that* x_1 *is bounded on* $H \cap K$. *We have that*

$$\sum_{i=3}^{n} \frac{|\alpha_2\beta_i - \beta_2\alpha_i|}{|\alpha_1\beta_2 - \alpha_2\beta_1|} < 1 \quad \textit{By } \textbf{Theorem 5}$$

x_1 *is bounded on* L *if and only if*

$$\sum_{i=2}^{n} \frac{|\gamma_i|}{|\gamma_1|} = \sum_{i=2}^{n} \frac{|c_1\alpha_i + c_2\beta_i|}{|c_1\alpha_1 + c_2\beta_1|} < 1 \quad \textit{By } \textbf{Theorem 4}$$

Let $c_1 = \beta_2$ *and* $c_2 = -\alpha_2$. *We then have*

$$\sum_{i=2}^{n} \frac{|\gamma_i|}{|\gamma_1|} = \frac{|\gamma_2|}{|\gamma_1|} + \sum_{i=3}^{n} \frac{|\gamma_i|}{|\gamma_1|}$$

$$\frac{|\beta_2\alpha_2 - \alpha_2\beta_2|}{|\beta_2\alpha_1 - \alpha_2\beta_1|} + \sum_{i=3}^{n} \frac{|\beta_2\alpha_i - \alpha_2\beta_i|}{|\beta_2\alpha_1 - \alpha_2\beta_1|} < 1$$

$$\therefore \sum_{i=2}^{n} \frac{|\gamma_i|}{|\gamma_1|} < 1.$$

Hence x_1 *is bounded on* L. □

To illustrate the proof above we again consider a simple example. Consider two hyperplanes in $\mathbf{R}^3$ given by $H = \boldsymbol{\alpha}^T \cdot \boldsymbol{x} = [6, 3, 4] \cdot \boldsymbol{x} = 0$ and $K = \boldsymbol{\beta}^T \cdot \boldsymbol{x} = [4, 3, -2] \cdot \boldsymbol{x} = 0$. By **Theorem 4** it is clear that no element is bounded on either hyperplane.

The intersection space may be determined as the null space of the following system:

$$A = H \cap K = \begin{bmatrix} \\ \boldsymbol{\alpha}^T \\ \\ \boldsymbol{\beta}^T \\ \\ \end{bmatrix} = \begin{bmatrix} & & \\ 6 & 3 & 4 \\ & & \\ 4 & 3 & -2 \\ & & \end{bmatrix}$$

where $\boldsymbol{x} \in H \cap K \;\Rightarrow\; \boldsymbol{x} = c\boldsymbol{n}$ where $c \in \mathbf{R}$ and $\boldsymbol{n} \in null(A)$. For this system $\boldsymbol{n} = [-0.532, 0.828, 0.177]$; clearly then both x_1 and x_3 are bounded. By **Proposition 1** there exist constants $\{c_1, \cdots, c_4\}$ such that x_1 and x_2 are bounded on the hyperplanes with normals $\boldsymbol{\gamma}_1 = c_1\boldsymbol{\alpha}+c_2\boldsymbol{\beta}$ and $\boldsymbol{\gamma}_2 = c_3\boldsymbol{\alpha}+c_4\boldsymbol{\beta}$ respectively. The constants $c_1, \cdots, c_4$ may be found by **Proposition 1**, we note simply that the selection $c_1 = c_2 = 1$ and $c_3 = -c_4 = 1$ gives normals $\boldsymbol{\gamma}_1 = [5, 3, 1]$ and $\boldsymbol{\gamma}_2 = [1, 0, 3]$ respectively. Then by **Theorem 4** x_1 is bounded on $\boldsymbol{\gamma}_1$ and x_3 is bounded on $\boldsymbol{\gamma}_2$.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Numerical Results

In this chapter we shall consider a numerical investigation into the heuristic choices to be made in the PPF method, and contrast the flagship implementation of the PPF method with alternate methods for solution to problem (1.2).

4.1 Implementation Details and Test Problems for the PPF Method

The numerical investigations in this section were carried out on a problem ensemble generated using MATLAB's *rand()* function. That is to say that the real matrix elements were drawn from a uniform random distribution: they were then arbitrarily scaled to lie between -1 and 1, i.e. $a_{ij}, b_i \in \{-1, 1\}$. The problem ensemble consisted of 100 differently sized linear system $A\boldsymbol{x} = \boldsymbol{b}$ where, $A \in \mathbf{R}^{m\times n}$, $0 < n \leq 505$ and $\frac{n-5}{10} \leq m \leq (n-5)$. The systems under consideration were evenly distributed through the space of potential problem sizes.

The purpose of this numerical investigation is to compare the heuristic options available to the PPF method so as to determine its flagship implementation, and to probe any interesting characterizations unique to the heuristics. A dual set of results is

presented, wherein the first case the various implementations are allowed to run to *completion* over a smaller number of problems (5), and in the second case the maximum number of iterations is *capped* so as to allow the various implementations to be run over a much larger number of problems (50).

When the various implementations were allowed to run to completion, for each particular system size ($m \times n$), five systems were randomly generated and the results averaged. Explicitly, once a system was generated, it was presented to each of the various implementations tested in this section. The algorithms were then allowed to run to completion, i.e. until the solution to problem (1.2) was found. Thereafter a second system of the same size was generated, and it too was presented to each of the implementations. This procedure was repeated until five systems of the same size had been generated and presented to each of the implementations, after which the entire procedure was repeated for a differently sized system ($m \times n$).

When the various implementations were allowed to run with the iteration cap in place, for each particular system size ($m \times n$), fifty systems were randomly generated and the results averaged. Once a system was generated, it was presented to each of the various implementations tested in this section. The algorithms were then allowed to run until the solution to problem (1.2) had been found, or the iteration limit had been reached - whichever came first. Thereafter a second system of the same size was generated, and it too was presented to each of the implementations. This continued until fifty systems of the same size had been presented. Thereafter the results were averaged and the entire procedure repeated for a differently sized system ($m \times n$).

The maximum number of iterations was capped at $10,000$. This value was chosen by inspection only after all of the PPF implementations had been run to completion with the smaller number of problems. This value was chosen so as to provide a fair view of the quantitative differences between the various PPF implementations (as well as the alternative methods presented in the comparison section, 4.2), while avoiding the prohibitive numerical cost of bad case for some implementations.

The systems under consideration were structured as follows:

m	$(n-5)$	.	.	.	.	.	.	.	.	.	.
	$\frac{9(n-5)}{10}$	.	.	.	.	.	.	.	.	.	.
	$\frac{8(n-5)}{10}$	.	.	.	.	.	.	.	.	.	.
	$\frac{7(n-5)}{10}$	.	.	.	.	.	.	.	.	.	.
	$\frac{6(n-5)}{10}$	.	.	.	.	.	.	.	.	.	.
	$\frac{5(n-5)}{10}$	.	.	.	.	.	.	.	.	.	.
	$\frac{4(n-5)}{10}$	.	.	.	.	.	.	.	.	.	.
	$\frac{3(n-5)}{10}$	.	.	.	.	.	.	.	.	.	.
	$\frac{2(n-5)}{10}$	.	.	.	.	.	.	.	.	.	.
	$\frac{(n-5)}{10}$	.	.	.	.	.	.	.	.	.	.
	$n \rightarrow$	55	105	155	205	255	305	355	405	455	505

Table 4.1: Systems used in numerical testing

All of the procedures tested in this section, as well as the problem ensembles, were coded using MATLAB.

For the most part the numerical results are presented as 2-dimensional colour maps in which the colour represents the measured quantity, which should be thought of as being overlayed onto the representative grid in Figure 4.1 above. Also note that all the numerical results were run on an HP ProBook 4530s with a 2.30 GHz Intel i5

processor and 3,00GB of RAM.

The results in this section are considered in a manner that directly parallels their presentation in the corresponding chapter. We begin by considering the heuristic choices to be made in Part 1 of the PPF method described in section 3.3, and then consider the heuristic choices to be made in Part 2 of the PPF method, see section 3.4.

A particular implementation of the PPF method can be identified by a 3-tuple indicating the choice of heuristics. Firstly the Ascent heuristic is specified, then the Exchange heuristic and finally the Transitional Ascent heuristic - this corresponds to the choice of Ascent heuristic following a multiple element exchange heuristic selection for which a full index set, $|\Lambda| = n - m + 1$, may need to be rebuilt many times. In the instance that a single element exchange procedure is selected, the Transitional Ascent heuristic is clearly irrelevant since a full index set is maintained at each iteration. Furthermore, the 3-tuple specifying the chosen heuristics is prefaced with either 'Comp' or 'Cap' indicating whether the algorithm was run to *completion* or with the maximum number of iterations *capped* respectively.

Throughout this section, an initial solution to the linear system was obtained as the minimum ℓ_2 solution, by way of the Moore-Penrose pseudo-inverse; see section 3.1.1.

4.1.1 Ascent Heuristics With Single Element Exchange

In this section the various Ascent heuristics are compared. These are the heuristic choices to be made when iteratively solving underdetermined linear systems in Part 1 of the method, i.e. when $|\Lambda| < n - m + 1$. The details of these heuristics can be found in section 3.3. We consider two such choices: firstly the Moore-Penrose pseudo-inverse (PI), as expounded in section 3.3.1, and secondly the Iterative Ascent procedure (IA), as per section 3.3.2.

The Iterative Ascent procedure works by zeroing-off selected columns of the under-

determined system to be solved. The resulting square system is solved instead. This naturally presents a further heuristic choice as to which columns should be zeroed-off, see section 3.3.2. We consider the instances in which the columns to be zeroed-off are selected to be those corresponding to the largest (IA-big) and the smallest (IA-small) components of the current feasible solution vector, corresponding to elements of the complement index set, i.e. min or max in $\{|x_j| : \ j \in \Lambda^c\}$.

Furthermore, each of the Ascent heuristics in Part 1 of the PPF method is tested with two different single element Exchange heuristics in Part 2 of the PPF method - this is done so as to show that the improvements in starting feasible solution for Part 2 of the PPF method offered by certain Ascent procedures is not depended on the subsequent choice of exchange procedure.

We consider now the results of the heuristics IA-big, IA-small and the pseudo-inverse as Ascent heuristics in Part 1, with the single element exchange heuristic 'Big' selected for Part 2. We consider only the number of iterations required in Part 2 of the PPF method as this provides a more direct way of gauging the improvement in starting solution for Part 2 offered by the various Ascent heuristics in Part 1. Specifically, the natural logarithm of the number of iterations in Part 2 is displayed.

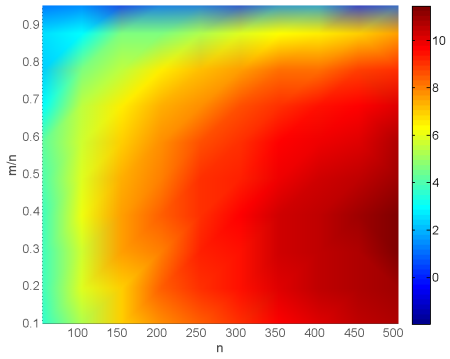


Figure 4-1: Natural logarithm of the number of iterations in Part 2 - Comp(IA-small, Big, NA)

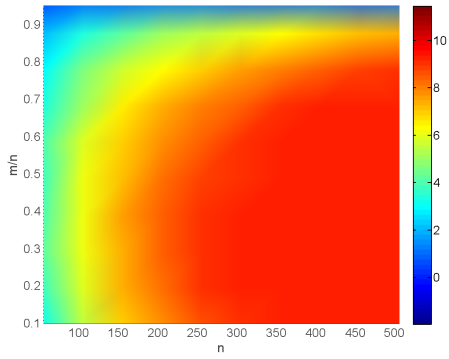


Figure 4-2: Natural logarithm of the number of iterations in Part 2 - Cap(IA-small, Big, NA)

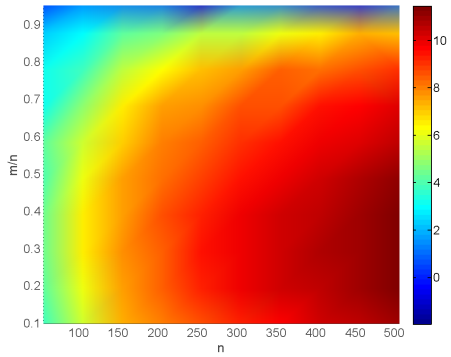


Figure 4-3: Natural logarithm of the number of iterations in Part 2 - Comp(IA-big, Big, NA)

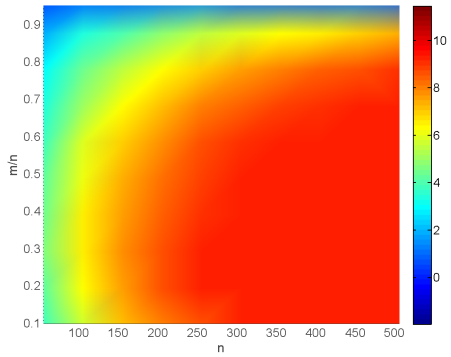


Figure 4-4: Natural logarithm of the number of iterations in Part 2 - Cap(IA-big, Big, NA)

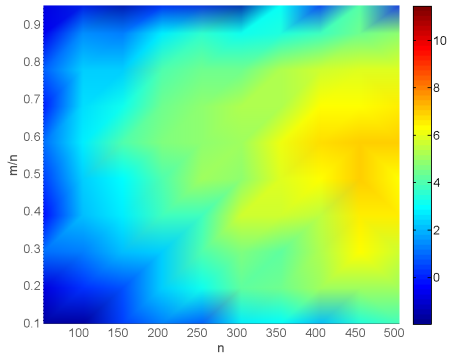


Figure 4-5: Natural logarithm of the number of iterations in Part 2 - Comp(PI, Big, NA)

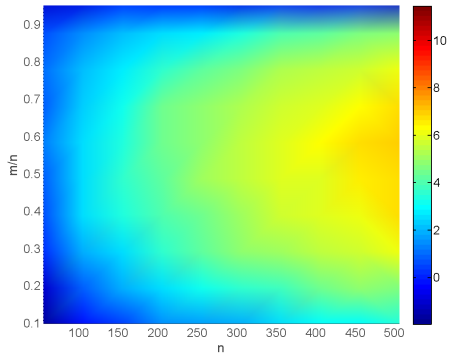


Figure 4-6: Natural logarithm of the number of iterations in Part 2 - Cap(PI, Big, NA)

Notice that each figure is captioned with a 3-tuple uniquely specifying the heuristic choices made in that implementation, and a prefix 'Comp' or 'Cap' indicating whether the algorithms were allowed to run to completion or with the iteration cap in place. Since a single element exchange procedure was selected, the Transitional Ascent Method (TAM) is irrelevant and is identified as 'NA' - Not Applicable. All of the above figures are plotted over the same dynamic range; that is to say that they all share the same colour scale depicted to the right of the figures. Red, therefore, represents the same number of iterations for each of the figures. This is done so that

the figures are directly comparable.

We notice that the 'Cap' figures exhibit slightly smoother behaviour than the 'Comp; figures, as is to be expected with the results generated over a much larger number of problems. That said, the fundamental behaviour of the implementations is essentially identical to the 'Comp' figures over the entire problem ensemble. Furthermore, the worst cases for the Iterative Ascent procedures IA-big and IA-small - when $\frac{m}{n} \approx 0$ - are seen to be slightly better for the 'Cap' figures. This suggests that the iteration cap was in fact reached for a number of runs.

It is clear then that the choice of pseudo-inverse for Ascent method seems to dramatically and uniformly out performs both of the Iterative Ascent procedures in terms of iteration count, although not in terms of iteration complexity, see section 3.6. While fifty problem instances is not a huge number for any particular system size, this conclusions is strengthened by the fact that similar results are observed for many systems over the entire problem ensemble. This point will hold true for each of the procedures tested in this section.

The Iterative Ascent method may be preferable in certain situations. It also seems as though there is little to choose between the two Iterative Ascent procedures as they both produce starting solutions for Part 2 of the algorithm that require approximately the same number of iterations, over the entire problem ensemble. Note also that the number of iterations required to obtain an initial full index set, $|\Lambda| = n - m + 1$ is, in general, the same for all ascent methods: this is why only iterations in Part 2 of the method are considered.

Observe further that for the choice of pseudo-inverse as Ascent method, Figures 4-5 and 4-6, the problems requiring the greatest number of iterations in Part 2 are approximately of size $m \times 2m$, alternatively put $\frac{m}{n} \approx \frac{1}{2}$. This is not observed in the case of either of the Iterative Ascent procedures, where those problems requiring the most iterations in Part 2 are those of size $m \ll n$, where $\frac{m}{n} \approx 0$. This suggests that the superiority of the choice of pseudo-inverse as Ascent procedure is cumulative, and

thus for less square systems which require a greater number of iterations to obtain an initial full index set, the pseudo-inverse procedure outperforms the Iterative Ascent procedures to a greater degree.

We now compare the performance, again in terms of iteration count in Part 2 of the PPF method, of the Ascent heuristics IA-small, IA-big and PI in Part 1, with the single element exchange heuristic 'Small' selected for Part 2. Again the natural logarithm of the number of iterations is the quantity presented.

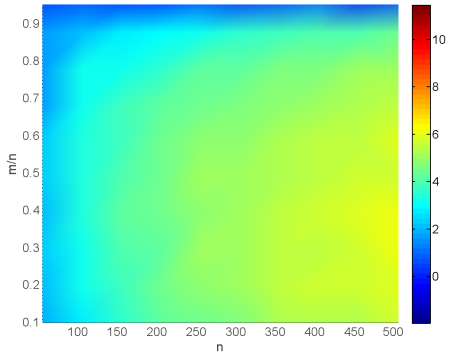


Figure 4-7: Natural logarithm of the number of iterations in Part 2
- Comp(IA-small, Small, NA)

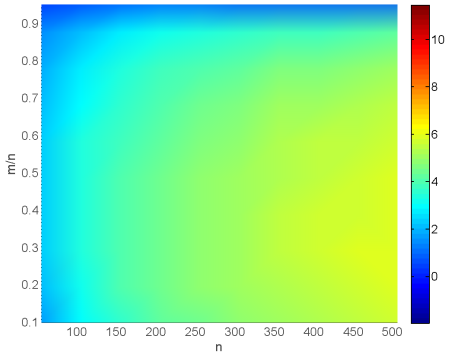


Figure 4-8: Natural logarithm of the number of iterations in Part 2
- Cap(IA-small, Small, NA)

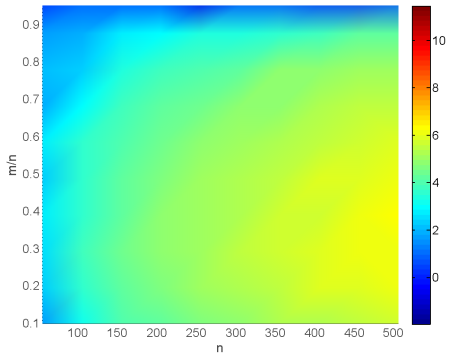


Figure 4-9: Natural logarithm of the number of iterations in Part 2
- Comp(IA-big, Small, NA)

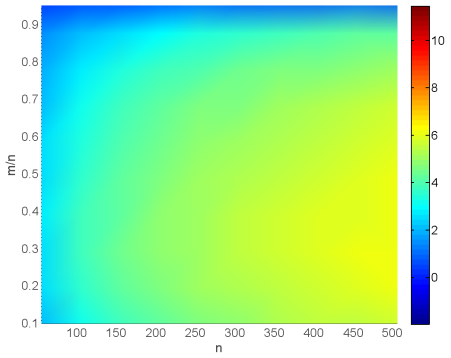


Figure 4-10: Natural logarithm of the number of iterations in Part 2
- Cap(IA-big, Small, NA)

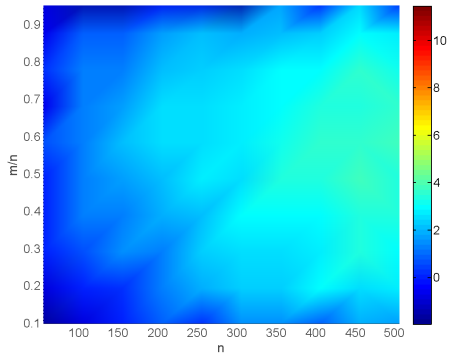


Figure 4-11: Natural logarithm of the number of iterations in Part 2 - Comp(PI, Small, NA)

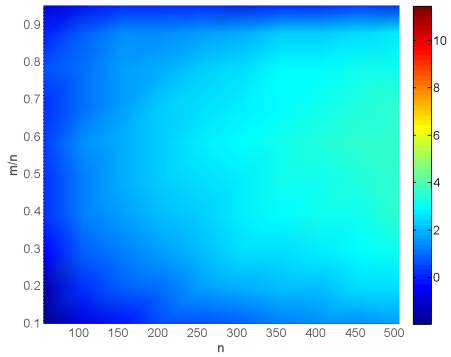


Figure 4-12: Natural logarithm of the number of iterations in Part 2 - Cap(PI, Small, NA)

The above six figures, Figures 4-7 to 4-12, are plotted over the same dynamic range as the previous six figures for the selection of the single element exchange procedure 'Big'. That is to say all twelve figures share a colour scale. This is done for ease of comparison.

We again observe that the 'Cap' figures exhibit smoother behaviour than the 'Comp' figures. In this instance the 'Cap' figures are almost identical to the 'Comp' figures, suggesting that few if any runs hit the iteration cap, and if they did, they would not have exceeded it by much had they been allowed to run to completion.

The results parallel those discussed above. It is again observed that the choice of pseudo-inverse seems greatly and uniformly superior, in terms of iteration count, to either choice of Iterative Ascent heuristic. Furthermore, for the choice of pseudo-inverse, those problems requiring the greatest number of iterations in Part 2 are approximately of size $m \times 2m$, where, $\frac{m}{n} \approx \frac{1}{2}$. For either Iterative Ascent heuristic those problems requiring the greatest number of iterations in Part 2 of the method are approximately of size $m \ll n$, where, $\frac{m}{n} \approx 0$.

It seems plausible that different exchange procedures may perform better for different starting solutions for Part 2 of the algorithm. The above figures, however, suggest that the choice of pseudo-inverse as Ascent method is preferable in terms of iteration count to either of the Iterative Ascent procedures regardless of the single element

exchange procedure used.

Furthermore, by directly comparing the same Ascent heuristics for different single element exchange heuristics, it is clear that the single element exchange procedure 'Small' is starkly and uniformly better than the single element exchange procedure 'Big', regardless of the Ascent procedure used.

4.1.2 Exchange Heuristics With Multiple Element Exchange

In this section the various heuristic choices for multiple element exchange procedures are considered. This corresponds to the situation in which a full index set has been obtained, $|\Lambda| = n - m + 1$, and one or more elements is removed from the index set in order to solve for the components of the direction vector corresponding to the complement of the index set, $\boldsymbol{d}^k_{\Lambda^c}$, see section 3.1.2.

The mathematical basis for the heuristics considered here may be found in the corresponding section of the thesis, section 3.4.2.

When a full index set is encountered, i.e. $|\Lambda| = n - m + 1$, there exist many possible descent direction vectors for the line search that can remove multiple elements from the index set. In this thesis only two such direction vectors were considered. Both procedures remove all possible elements from the index set . Firstly, the choice in which all elements corresponding to valid indicators are decreased in relation to their valid indicator was developed (IntOnes), and secondly the choice in which all elements corresponding to valid indicators are decreased equally (IntEqual) was developed.

When multiple elements are removed from a full index set, the set must once again be built up. In this instance one is required to solve an underdetermined linear system for the components of the direction vector corresponding to the complement of the index set. The heuristic choices to be made in this case are exactly the same as those for the Ascent Method, indeed we have termed this heuristic choice the Transitional Ascent Method. We consider explicitly the Moore-Penrose pseudo-inverse (PI), the

Iterative Ascent procedures (IA-big) and (IA-small) as outlined in section (3.3.2). Moreover we consider a specific instance of the Iterative Ascent procedure in which the columns to be zeroed-off are those corresponding to the components of the feasible solution vector with indices that were removed from the index set on the previous iteration (IA-old). Details of all of these heuristic choices may be found in section 3.4.2.

We proceed by first establishing the Transitional Ascent heuristic that is best for each multiple element exchange procedure, and then compare each of the exchange procedures. In order for the exchange procedures to be compared directly, they all share the same starting solution for Part 2 resulting from the ascent heuristic IA-small. Consider the exchange procedure in which the components corresponding to elements removed from the index set are decreased in relation to the absolute value of their respective indicators.

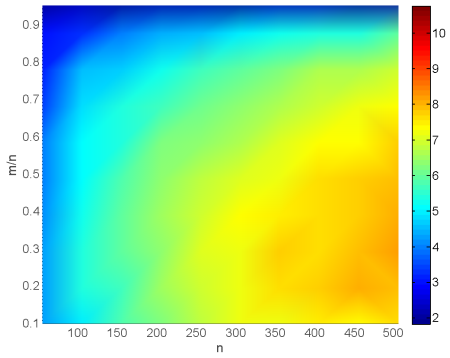


Figure 4-13: Natural logarithm of the total number of iterations
- Comp(IA-small, IntOnes, PI)

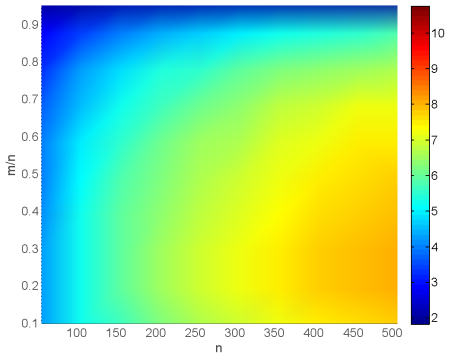


Figure 4-14: Natural logarithm of the total number of iterations
- Cap(IA-small, IntOnes, PI)

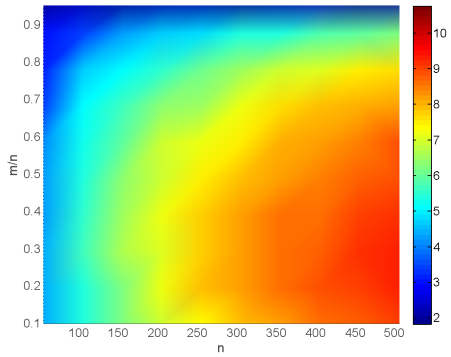


Figure 4-15: Natural logarithm of the total number of iterations
- Comp(IA-small, IntOnes, IA-small)

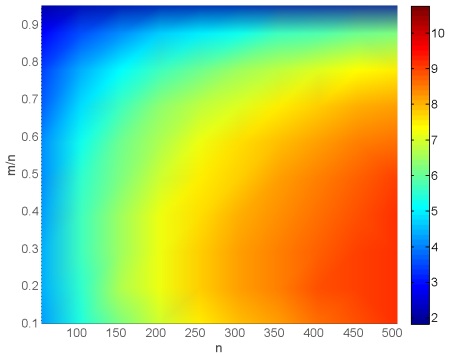


Figure 4-16: Natural logarithm of the total number of iterations
- Cap(IA-small, IntOnes, IA-small)

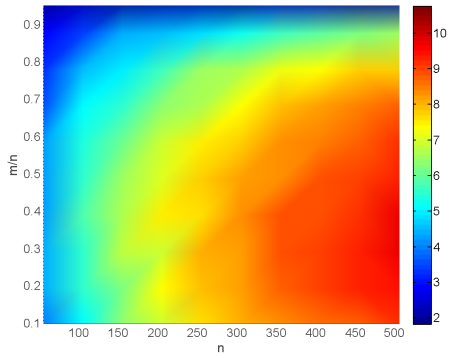


Figure 4-17: Natural logarithm of the total number of iterations
- Comp(IA-small, IntOnes, IA-old)

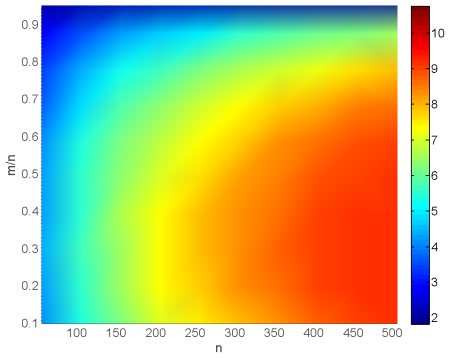


Figure 4-18: Natural logarithm of the total number of iterations
- Cap(IA-small, IntOnes, IA-old)

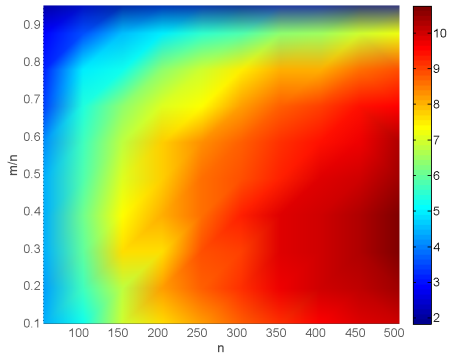


Figure 4-19: Natural logarithm of the total number of iterations
- Comp(IA-small, IntOnes, IA-big)

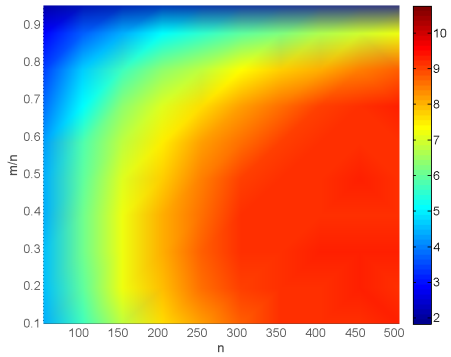


Figure 4-20: Natural logarithm of the total number of iterations
- Cap(IA-small, IntOnes, IA-big)

The eight figures above were plotted over the same dynamic range for direct comparison. The smoother behaviour of the 'Cap' figures is again observed. The worst cases for the Transitional Ascent heuristics IA-big are observed to be worse for the 'Comp' figure. This may suggest that the iteration limit was reached for a number of runs with this particular implementation.

It is again clear that the choice of pseudo-inverse as Transitional Ascent method, Figures 4-13 and 4-14, is preferable in terms of total number of iterations. Figures 4-13 to 4-20 show that the choice Iterative Ascent 'IA-big' is uniformly outperformed by the choice Iterative Ascent 'IA-small' which in turn shows little improvement over Iterative Ascent 'IA-old'. Again, the preferential status of more square systems for Iterative Ascent is observed for all of the above figures, although this is almost certainly a characteristic of the starting feasible solution for Part 2 generated by the Ascent method and *not* the choice of Transitional Ascent method.

We now turn our attention to the multiple element exchange heuristic in which the components corresponding to elements removed from the index set are decreased equally (IntEqual).

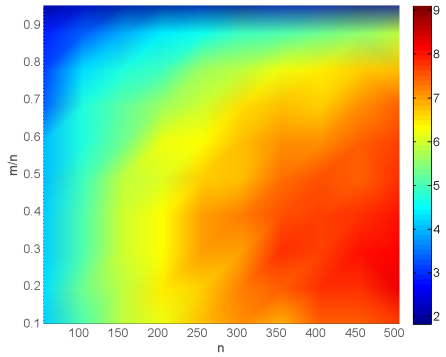


Figure 4-21: Natural logarithm of the total number of iterations
- Comp(IA-small, IntEqual, PI)

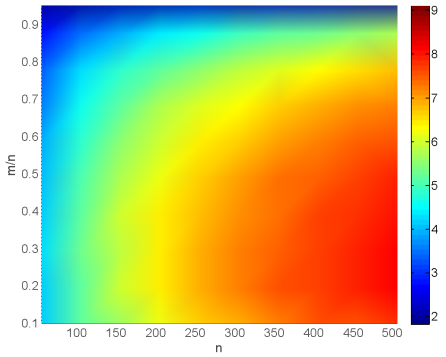


Figure 4-22: Natural logarithm of the total number of iterations
- Cap(IA-small, IntEqual, PI)

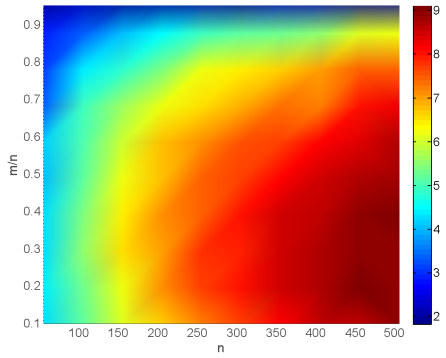


Figure 4-23: Natural logarithm of the total number of iterations
- Comp(IA-small, IntEqual, IA-small)

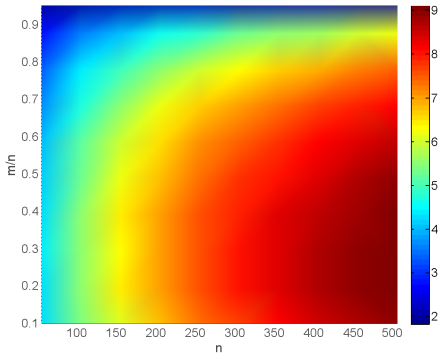


Figure 4-24: Natural logarithm of the total number of iterations
- Cap(IA-small, IntEqual, IA-small)

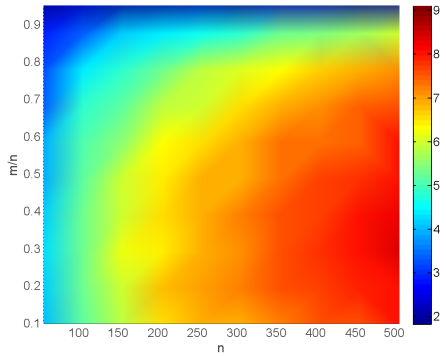


Figure 4-25: Natural logarithm of the total number of iterations
- Comp(IA-small, IntEqual, IA-old)

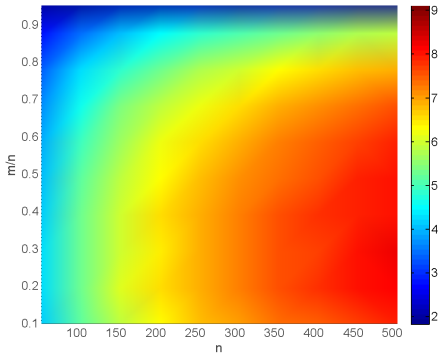


Figure 4-26: Natural logarithm of the total number of iterations
- Cap(IA-small, IntEqual, IA-old)

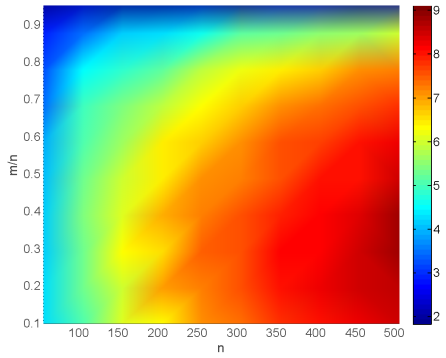


Figure 4-27: Natural logarithm of the total number of iterations in
- Comp(IA-small, IntEqual, IA-big)

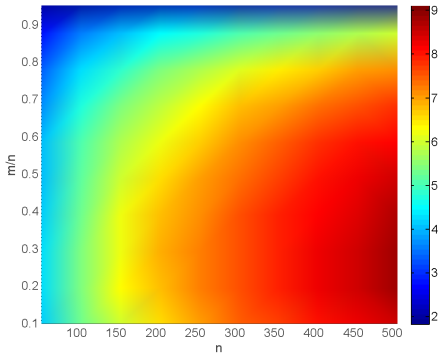


Figure 4-28: Natural logarithm of the total number of iterations in
- Cap(IA-small, IntEqual, IA-big)

The above eight figures were plotted over the same dynamic range for comparative purposes. The smoother behaviour of the 'Cap' figures is again observed. The 'Cap' and 'Comp' figures are otherwise nearly identical. This can be explained by observing the colour scale to the right of the figures. We see immediately that the maximum number of iterations was approximately $\exp^9 \approx 8,100$ which is below the iteration cap of $10,000$.

For this choice of exchange procedure, the choice of pseudo-inverse for Transitional Ascent method, Figure 4-22, does not greatly out perform, in terms of iteration count,

all of the Transitional Iterative Ascent procedures. Indeed, it is comparable to the Transitional Iterative Ascent procedure 'IA-old', Figures 4-25 and 4-26 - in which the columns to be zeroed-off are those corresponding to the elements removed from the index set.

This is likely due to the special fact that since the elements removed from the index set were decreased equally, and the fact that by the very nature of the Iterative Ascent procedures all but one of these elements do not change in value during the succeeding line search. Multiple elements may thus frequently join the index set in a single line search iteration.

For the purposes of direct visual comparison we present the two multiple element exchange heuristics (IntOnes) and (IntEqual), with the preferential choice of Transitional Ascent method as pseudo-inverse, and the single element exchange procedure 'Small', over the same dynamic range. These figures below are not identical to those produced previously since they have been plotted over the dynamic range as each other. This essentially qualifies as a comparison of the best exchange procedure for the PPF method. All exchange heuristics once again share a starting solution for Part 2.

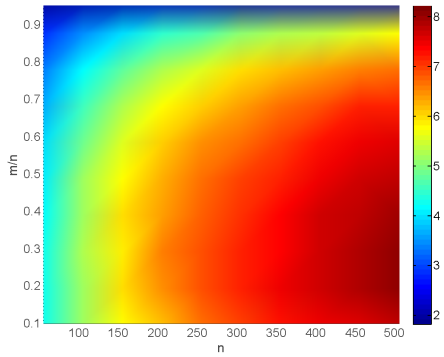


Figure 4-29: Natural logarithm of the total number of iterations
- Cap(IA-small, IntOnes, PI)

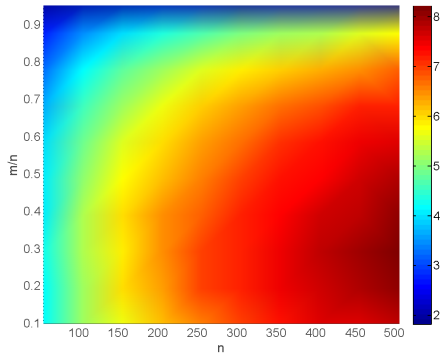


Figure 4-30: Natural logarithm of the total number of iterations
- Cap(IA-small, IntEqual, PI)

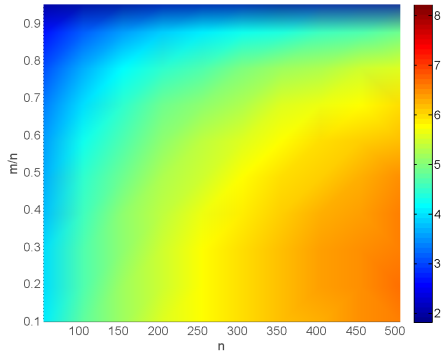


Figure 4-31: Natural logarithm of the total number of iterations
- Cap(IA-small, Small, NA)

It is immediately clear that there is relatively little to choose between the multiple element exchange procedures. The single element exchange procedure 'Small' is dramatically and uniformly superior to both of the interior direction vector selection procedures, over the full problem ensemble. The single element exchange heuristic 'Small' shall thus be taken as the flagship exchange heuristic for the PPF method, with the pseudo-inverse ascent heuristic selected.

4.2 Numerical Comparison of Current Best Methods

In this section we compare the flagship implementation of the PPF method with two of the current best algorithms for the solution of problem (1.2). Explicitly, the PPF method is compared with the method proposed by Cadzow [6] and the LP formulation due to Abdelmalek [2].

The problem ensemble considered here was generated in a similar fashion to that for the internal testing of the PPF method. The only difference being that a different random seed was used so as not to bias the comparison. The figures are again captions with the prefix 'Cap' or 'Comp' to indicate whether the algorithms were allowed to run to completion or whether the were run with the iteration cap in place. The maximum number of iterations was again capped at $10,000$.

Again, all numerical testing in this section was run on an HP ProBook 4530s with a 2.30 GHz Intel i5 processor and 3,00GB of ram. Also, the starting points for each algorithm were computed as per the initialization procedures outlined in the respective articles.

Consider now the performance of the flagship implementation of the PPF method, Cadzow's method, and the LP formulation of Abdelmalek over this problem ensemble. Note that in contrast with the preceding section, we consider the *total* number of iterations for the PPF method (both Part 1 and Part 2).

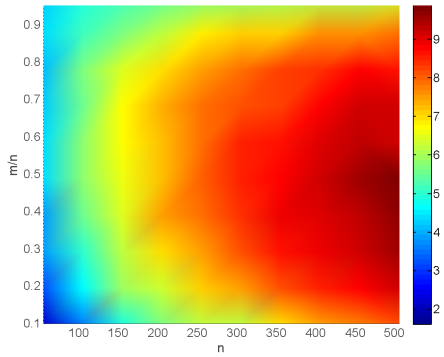


Figure 4-32: Natural logarithm of the total number of iterations
- Comp - Cadzow's Method

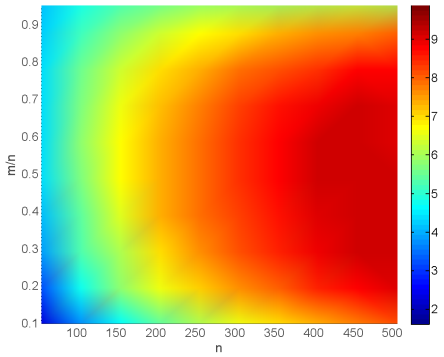


Figure 4-33: Natural logarithm of the total number of iterations
- Cap - Cadzow's Method

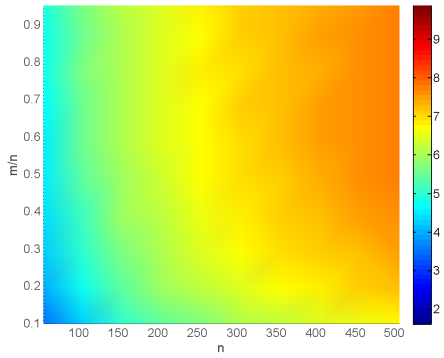


Figure 4-34: Natural logarithm of the total number of iterations
- Comp - LP Formulation

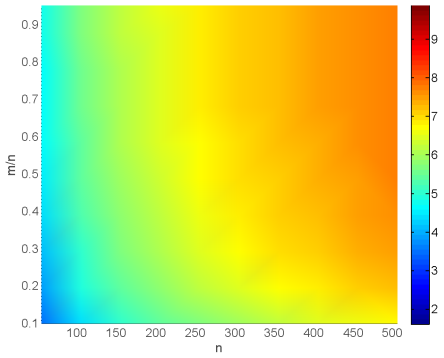


Figure 4-35: Natural logarithm of the total number of iterations
- Cap - LP Formulation

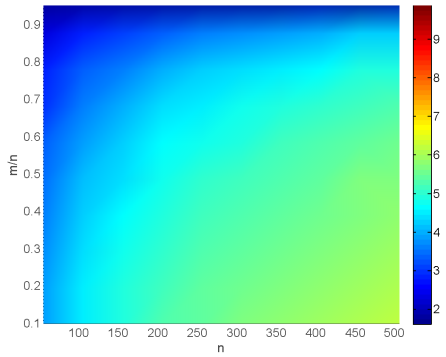


Figure 4-36: Natural logarithm of the total number of iterations
- Comp(PI, Small, NA)

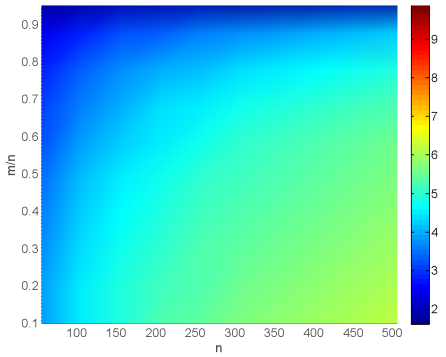


Figure 4-37: Natural logarithm of the total number of iterations
- Cap(PI, Small, NA)

The Figures 4-32 to 4-37 above were plotted over the same dynamic range for comparative purposes. The 'Cap' figures are observed to exhibit slightly smoother behaviour than the 'Comp' figures, although their fundamental behaviour over the entire problem ensemble is otherwise the same.

The results are stark: the PPF method greatly outperforms the LP formulation which in turn greatly outperforms Cadzow's method.

It is interesting that the class of problems requiring the greatest number of iterations is different for each method. For the PPF method, the characteristic preferential status of more square systems is again observed. This stems from the greater number of iterations required to attain an initial full index set for systems where $m \ll n$. The class of problems requiring the greatest number of iterations for Cadzow's method are those for which $n \approx 2m$ where $\frac{m}{n} \approx \frac{1}{2}$. This is very similar to the behaviour of the PPF method when only iterations in Part 2 are considered. This is evidence for the improved starting positions offered by primal methods at the seemingly negligible cost of attaining an initial full index set. The LP formulation seems to perform more poorly for more square systems. This is intuitively consistent as the LP formulation considered, in fact solves the dual problem.

For the purposes of attaining a sense of tangible comparison, the explicit results for a representative subset of the problem ensemble are tabulated in Tables 4.2 and 4.3

below - corresponding to the 'Comp' and 'Cap' results respectively. Note that since the results for a given problem size were averaged over multiple runs, the number of iterations in the table may not be an integer value. Also, the time as presented is in seconds.

(n,m)	PPF		Cadzow		LP	
	Iterations	Time (s)	Iterations	Time (s)	Iterations	Time (s)
(105, 20)	86.0	0.025820	100.0	0.071290	124.6	0.058537
(105, 50)	58.6	0.029735	339.4	0.269086	217.6	0.109177
(105, 80)	28.6	0.021857	272.8	0.230393	276.0	0.153930
(255, 50)	211.4	0.115914	1151.8	1.063301	475.2	0.490364
(255, 125)	145.4	0.271330	2847.6	3.631058	785.2	1.008168
(255, 200)	66.0	0.275234	1679.6	3.329964	923.0	1.479027
(405, 80)	344.8	0.375891	4755.8	5.472940	985.0	2.970690
(405, 200)	236.0	1.175420	9461.6	20.755131	1605.0	6.136187
(405, 320)	105.0	1.439382	4270.2	26.048189	1638.2	10.271416

Table 4.2: Comparative results for a representative subset of the problem ensemble (Complete - run to convergence)

(m,n)	PPF		Cadzow		LP	
	Iterations	Time (s)	Iterations	Time (s)	Iterations	Time (s)
(20, 105)	86.4	0.028021	116.1	0.080436	126.1	0.054873
(50, 105)	58.3	0.032339	292.7	0.225803	214.5	0.106290
(80, 105)	28.2	0.027411	258.4	0.211419	264.6	0.145643
(50, 255)	210.2	0.134892	1210.6	1.029653	471.7	0.451317
(125, 255)	143.7	0.303053	2708.0	3.198553	782.1	0.963923
(200, 255)	65.62	0.298590	1574.1	2.854415	902.3	1.343694
(80, 405)	333.7	0.377400	4665.7	5.019515	981.5	2.881342
(200, 405)	228.8	1.263181	9048.9	20.470478	1533.6	6.277001
(320, 405)	104.3	1.541051	4252.9	22.445823	1699.9	11.074329

Table 4.3: Comparative results for a representative subset of the problem ensemble (Capped - run with iteration limit)

The results in Tables 4.2 and 4.3 are extremely similar.

Note that while the iteration count and computational complexity of the relevant algorithms serves as a fair metric for comparison, the explicit time required for algorithms to run is, of course, subject to the inherent quality of the code. Every reasonable effort was made to optimize the code on the given platform. Moreover, the rank-1 update to the pseudo-inverse, considered in section 3.3.1, is not consid-

ered in the article on Cadzow's method [6], although this method might also stand to benefit from the implementation of such a procedure. In order to remain true to its current interpretation, the rank-1 update procedure was not implemented in Cadzow's method in the comparison above.

For a fair comparison of the methods we also did not implement the rank-1 update in the PPF method in the comparison testing or elsewhere in this subsection. While the iteration count would have remained unchanged had the rank-1 update to the pseudo-inverse been included, the explicit time required for the PPF method and Cadzow's method would have decreased. The values presented in tables 4.2 and 4.3 for the time required are therefore upper bounds.

4.3 Discussion

In this section we will present a concise yet critical discussion of the merits and drawbacks of various solution methods proposed for the solution to problem (1.2). Special attention will be paid to the prospects for future research following the development of the PPF method.

Many varied methods for the solution to problem (1.2) have been proposed over the years. There is much to be gained from the consideration of each of these methods, regardless of their relative computational efficacy. The method put forward by Ha and Lee in 2002 [13], for example, seeks to combine the computational efficiency of Cadzows' method [6] with the geometrical clarity of the SY method [16]. While the method due to Ha and Lee is computationally inferior to Cadzow's method, its mere existence speaks to the importance of the conceptual and geometric clarity of a method to practitioners.

Those methods explicitly considered in section 2.1 all utilize the polyhedral structure of the objective function in obtaining a solution. The LP method due to Abdelmalek [2], employs the well known simplex method for its iterative computation. The me-

chanics of this method are therefore not considered in this thesis. The efficiency of the method is due to the fact that these computations are carried out on a reduced tableu. The reduced tableu is obtained by appealing to the strong symmetries in the constraint matrix of the dual formulation. This method has the benefit for practitioners of being 'familiar' in its implementation. It is worth noting that the LP formulation may also stand to benefit from the use of interior-point algorithms.

Arguably the flagship method for solution to problem (1.2) is the method due to Cadzow [6]. It is an efficient path-following method which solves the dual problem, and infers a solution to the primal problem by way of the so called alignment criteria, see section 2.1.2 - what this method gains in computational efficacy, it seems to lose in conceptual clarity [13, 16].

Brandishing the torch of conceptual clarity, the SY method sought to provide a clear primal method at the expense of computational inferiority to the methods of Cadzow and the LP formulation of Abdelmalek. The SY method is, computationally unsound. If the randomly selected vertex of the hypercube just so happens to be an interior point of the mapped polytope, then there will exist no boundary hyperplane containing that point. The algorithm will take $\binom{n}{m-1}$ iterations to discover this fact.

The PPF method fills the gap in the literature as a conceptually and geometrically clear, computationally efficient primal path-following algorithm. Numerical testing in this thesis suggests that the PPF method requires many fewer iterations to convergence than does Cadzow's method, or the LP formulation due to Abdelmalek. Moreover, the PPF method is comparable, and indeed, given the appropriate combination of heuristic choices, superior to these methods in terms of iterative computational complexity. The improved iteration count may be intuitively attributed to the improved starting positions provided by a primal formulation, as well as the powerful exchange heuristics presented.

There is much future research to be done in expanding the analysis of multiple element exchange procedures in the PPF method, in particular considering multiple element

exchange procedures in which not all of the elements corresponding to valid indicators are removed from the index set.

A further novel contribution of this thesis was to develop a set of geometric initial active constraint heuristics for the PPF method. The requirements were, however, too stringent to be of great practical usage in the particular problem ensemble considered numerically. An extension of these results will certainly prove fruitful to both the PPF method and the field of minimum ℓ_∞ norm solutions in general.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Conclusion

In this thesis a representative overview of varied solution methods to the minimum ℓ_∞ norm solution to a linear system of equations was presented, in both the underdetermined and overdetermined setting. The merits and demerits of various solution methods were expounded, and the insights gained employed in the development of a new method of solution for the underdetermined case - the Primal Path-Following (PPF) method.

The PPF method provides us with a conceptually and geometrically clear, primal path-following algorithm. Many powerful heuristics were developed which sought to further the conceptual understanding of the problem, extend a number of ideas currently confined to the overdetermined setting, and improve the computational complexity of the algorithm. Indeed, with an appropriate choice of heuristics for the implementation of the PPF method, the computational complexity is in fact superior to the alternative methods presented.

A numerical investigation was carried out on heuristics internal to the PPF method. The results suggested that a single element exchange procedure, in which the element corresponding to the valid indicator smallest in absolute value is removed from the index set, is uniformly superior to both other single element exchange procedures as well as those multiple element exchange procedures considered herein. Furthermore,

a numerical comparison of the flagship implementation of the PPF method with alternative methods showed a great improvement in terms of both iteration count and wall-clock time over a randomly generated problem ensemble of 100 differently sized systems.

The ideas developed in this thesis for the PPF method, were done so in the underdetermined setting. These ideas carry freely to the overdetermined setting as well. It would be a worthwhile exercise to carry out the analysis explicitly in this setting.

A novel set of geometric considerations for initial active constraint set select has been developed. While the requirements here were too stringent to be greatly applicable to the particular problem ensemble considered numerically here, there is much to be gained from continuing with this line of enquiry.

Bibliography

[1] N. N. Abdelmalek. Chebyshev solution of overdetermined systems of linear equations. *BIT Numerical Mathematics*, 15(2):117–129, 1975.

[2] N. N. Abdelmalek. Minimum ℓ-infinity solution of underdetermined systems of linear equations. *Journal of Approximation Theory*, 20(1):57–69, 1977.

[3] I. Barrodale and C. Phillips. An improved algorithm for discrete chebyshev linear approximation. In *Proc. Fourth Manitoba Conference on Numerical Mathematics, BL Hartnell and HC Williams (Ed.)*, pages 177–190. Utilitas Mathematica Pub., 1975.

[4] R. H. Bartels, A. R. Conn, and Y. Li. Primal methods are better than dual methods for solving overdetermined linear systems in the ℓ_∞ sense. *SIAM journal on numerical analysis*, 26(3):693–726, 1989.

[5] J. A. Cadzow. A finite algorithm for the minimum l_∞ solution to a system of consistent linear equations. *SIAM Journal on Numerical Analysis*, 10(4):607–617, 1973.

[6] J. A. Cadzow. An efficient algorithmic procedure for obtaining a minimum ℓ_∞-norm solution to a system of consistent linear equations. *SIAM Journal on Numerical Analysis*, 11(6):pp. 1151–1165, 1974.

[7] J. A. Cadzow. Minimum ℓ_1, ℓ_2, and ℓ_∞ norm approximate solutions to an overdetermined system of linear equations. *Digital Signal Processing*, 12(4):524–560, 2002.

[8] E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.

[9] E. W. Cheney. *Introduction to Approximation Theory*. AMS Chelsea Publishing Series. American Mathematical Society, 1982.

[10] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*, volume 76. John Wiley & Sons, 2013.

[11] C.J. de la Vallée Poussin. Sur la methode de l'approximation minimum. *Societe Scientifique de Bruxelles*, 35:116, 1911.

[12] R. Fletcher, J. A. Grant, and M. D. Hebden. Linear minimax approximation as the limit of best ℓ_p-approximation. *SIAM Journal on Numerical Analysis*, 11(1):pp. 123–136, 1974.

[13] I. Ha and J. Lee. Analysis on a minimum infinity-norm solution for kinematically redundant manipulators. *ICASE*, 4(2):130–139, June 2002.

[14] M. Khan. Updating inverse of a matrix when a column is added/removed. Technical report, UBC, 2008. Unpublish.

[15] M. R. Osborne and G. A. Watson. On the best linear chebyshev approximation. *The Computer Journal*, 10(2):172–177, 1967.

[16] I. Shim and Y. Yoon. Stabilized minimum infinity-norm torque solution for redundant manipulators. *Robotica*, 16(2):193–205, March 1998.

[17] E. Stiefel. Über diskrete und lineare tschebyscheff-approximationen. *Numerische Mathematik*, 1(1):1–28, 1959.

[18] G. A. Watson. Approximation in normed linear spaces. *Journal of Computational and Applied Mathematics*, 121(1):1–36, 2000.