NORMALIZED CUT PROBLEMS WITH GENERALIZED LINEAR CONSTRAINTS

by

IVÁN OJEDA-RUIZ

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2017

To my parents: Antonia Ruiz and José Ojeda.

ABSTRACT

NORMALIZED CUT PROBLEMS WITH GENERALIZED LINEAR CONSTRAINTS

IVÁN OJEDA-RUIZ , Ph.D.

The University of Texas at Arlington, 2017

Supervising Professor: Ren-Cang Li

Several methods are used to process images in many fields, including clustering, image segmentation and medical imaging. The so-called graph-cut methods in graph theory are widely used for image segmentation. In these methods graphs determining the relation between several objects are divided into one or more pieces in order to solve a variety of problems. Most of these methods are unsupervised, which means there is no information known about the data objects. In some of the applications listed above some prior knowledge may be known. Using this prior knowledge can be the key to designing better methods.

A novel algorithm called the projected power method used to solve the constraint eigenvalue problem was published by Xu, Li, and Schuurmans in "Fast Normalized Cut with Linear Constraints" [*IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2866-2873, Jun. 2009]. It is a variant of the power method which is known to converge often too slow. We propose new methods that use the subspace iteration and Krylov subspaces in a similar fashion to solve the constraint eigenvalue problem more accurately and faster. The study has shown this algorithm can produce better results to a general class of optimization problems which have a larger number of applications and can impact many fields positively. We also explore generalizations to the problem.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

Background

1.1 Introduction

Image processing is a very wide field with many practical applications. Image segmentation is one area of particular interest because of the variety of applications including medical imaging, object detection and recognition tasks. In 2000 Shi and Malik [14] presented a general framework for image segmentation using a graph theory approach. The set of points are represented as a weighted undirected graph. In order to partition an image we use a graph cut on the graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ (where $\mathbf{V}$ is the set of vertices and $\mathbf{E}$ is the set of edges). A graph cut is to partition a graph $\mathbf{G}$ into two disjoint sets $\mathbf{A}$ and $\mathbf{B}$ (where $\mathbf{A} \cup \mathbf{B} = \mathbf{V}$ and $\mathbf{A} \cap \mathbf{B} = \varnothing$). The quality of such a graph cut is usually measured by

$$\text{cut}(\mathbf{A}, \mathbf{B}) = \sum_{u \in \mathbf{A}, v \in \mathbf{B}} w(u, v),$$

where $w(u, v)$ is the weight of the edge between a vertex u in $\mathbf{A}$ and a vertex v in $\mathbf{B}$. The most common graph cut used is the minimum cut. But there is a flaw in the minimum cut. It tends to isolate points. This flaw is illustrated in Figure 1.1.

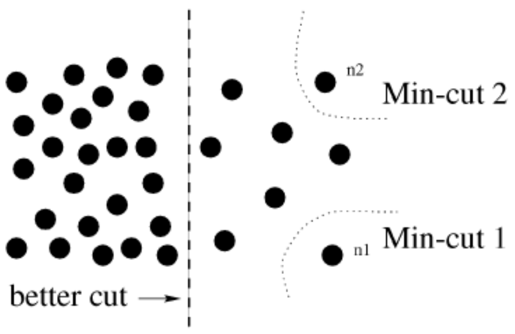

Figure 1.1. A case where the minimum cut gives a bad partition (assume the edge weights are inversely proportional to the distance between two nodes).

In order to work around this flaw, Shi and Malik proposed a new graph cut called the *normalized cut.* The normalized cut is defined as

$$\mathrm{Ncut}(\mathbf{A},\mathbf{B}) = \frac{\mathrm{cut}(\mathbf{A},\mathbf{B})}{\mathrm{assoc}(\mathbf{A},\mathbf{V})} + \frac{\mathrm{cut}(\mathbf{A},\mathbf{B})}{\mathrm{assoc}(\mathbf{B},\mathbf{V})},$$

where $\mathrm{assoc}(\mathbf{A},\mathbf{V}) = \sum_{u\in\mathbf{A},t\in\mathbf{V}} w(u,t)$ is the total connection from nodes in $\mathbf{A}$ to all nodes in the graph and $\mathrm{assoc}(\mathbf{B},\mathbf{V})$ is similarly defined. In [14], Shi and Malik found a way to compute the minimum of the normalized cut function as follows:

$$\min \mathrm{Ncut}(\mathbf{A},\mathbf{B}) = \min_{\boldsymbol{f}} \frac{\boldsymbol{f}^\top (D-W)\boldsymbol{f}}{\boldsymbol{f}^\top D\boldsymbol{f}} \qquad \text{subject to} \qquad \boldsymbol{f}^\top D\mathbf{e} = 0, \tag{1.1}$$

where D be the $n\times n$ diagonal matrix with diagonal elements $d(i) = \sum_j w(i,j)$ and W is the affinity matrix with (i,j)th $w(i,j)$. Now we let $\boldsymbol{g} := D^{\frac{1}{2}}\boldsymbol{f}$ in order to normalize $\boldsymbol{f}$. This will turn problem (1.1) into the following Rayleigh Quotient minimization problem,

$$\min_{\boldsymbol{g}} \frac{\boldsymbol{g}^\top D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}\boldsymbol{g}}{\boldsymbol{g}^\top \boldsymbol{g}} \qquad \text{subject to} \qquad \boldsymbol{g}^\top D^{\frac{1}{2}}\mathbf{e} = 0. \tag{1.2}$$

1.1.1 Normalized cut with Linear Constraints

In order to include prior information into the framework we will add extra constraints to problem (1.2). By adding a set of general linear constraints $B\boldsymbol{g} = \boldsymbol{c}$ we can consider label assignments of some sort. In this thesis we use grouping information that classifies certain pixels in a particular graph together with the normalized cut. By including the set of general linear constraints into problem (1.2) we obtain

$$\min_{\boldsymbol{g}} \boldsymbol{g}^\top D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}\boldsymbol{g} \qquad \text{subject to} \qquad \begin{array}{c} \boldsymbol{g}^\top D^{\frac{1}{2}}\mathbf{e} = 0, B\boldsymbol{g} = \mathbf{c}, \\ \|\boldsymbol{g}\| = 1, \end{array} \tag{1.3}$$

where $\|\boldsymbol{g}\|$ is the 2-norm of the vector $\boldsymbol{g}$. Notice that $D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}$ is symmetric positive semidefinite since $(D-W)$, also called the *Laplacian matrix*, is known to be positive semi-definite [12].

1.1.2 Constrained and Generalized Constraint Problems

We solve the following general problem in order to solve the Normalized Cut problem (1.3).

Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive semi-definite (s.p.s.d) matrix,

$$n > m, \quad B \in \mathbb{R}^{m\times n}, \quad \mathbf{c} \in \mathbb{R}^m.$$

$$\text{Find} \qquad \max_{\boldsymbol{v}\in\mathbb{R}^n} \boldsymbol{v}^\top A\boldsymbol{v} \qquad \text{subject to} \qquad ||\boldsymbol{v}|| = 1, \qquad B\boldsymbol{v} = \boldsymbol{c}. \tag{1.4}$$

The simplified version of this problem that does not consider the constraint $B\boldsymbol{v} = \boldsymbol{c}$ can be expressed as follows.

Let $A \in \mathbb{R}^{n\times n}$ be a s.p.s.d matrix,

$$n > m, \quad \mathbf{c} \in \mathbb{R}^m.$$

$$\text{Find} \qquad \max_{\boldsymbol{v}\in\mathbb{R}^n} \boldsymbol{v}^\top A\boldsymbol{v} \qquad \text{subject to} \qquad ||\boldsymbol{v}|| = 1. \tag{1.5}$$

Problem (1.5) is known to yield the largest eigenvalue of A. We call problem (1.5) an *eigenvalue problem* [16]. Problem (1.4) is studied extensively in [4] where it is called a *constrained eigenvalue problem*. A new numerical method to the constrained eigenvalue problem (1.4) has been studied by Xu, Li and Schuurmans in

[17]. The speed of various algorithms has been tested to determine which algorithm is faster. In chapter 3 we introduce a new method to solve this problem. We also introduce two new modifications of this problem. In general, we show how to reduce the modified problems to standard eigenvalue problems so that the standard algorithms may be used [5]. The first problem is the *Generalized Constraint Eigenvalue Problem* (GCEP),

$$\text{find} \qquad \max_{\boldsymbol{v}\in\mathbb{R}^n} \boldsymbol{v}^\top A\boldsymbol{v} \qquad \text{subject to} \qquad ||\boldsymbol{v}|| = 1, \qquad B\boldsymbol{v} \in \{\alpha\boldsymbol{c} | \alpha \in \mathbb{R}\}. \tag{1.6}$$

We will also study an even more general form called the *Modified Matrix Generalized Eigenvalue Problem* (MMEP) which can be expressed as follows.

$$\text{Find} \qquad \max_{W\in\mathbb{R}^{n\times k}} \mathbf{tr}(W^\top AW) \qquad \text{subject to} \qquad W^\top W = I_k, \quad \mathcal{R}(BW) \subseteq \mathcal{R}(C), \tag{1.7}$$

where $\mathcal{R}(X)$ be the subspace spanned by the columns of the matrix X.

In this thesis we go in detail about all the progress we have made as to understand these problem in hopes to find a solution and at the same time find fast algorithms.

CHAPTER 2

Review of Numerical Methods

In this chapter we will review some of the most fundamental methods for finding eigenvalues and finding roots. These methods have been commonly used for years in a wide range of fields.

2.1 Eigenvalue Finding Methods

In this section we will outline the basics of two commonly used methods for finding eigenvalues. Given a matrix A, if there is a vector $\boldsymbol{v}$ such that $A\boldsymbol{v} = \lambda\boldsymbol{v}$ for some scalar λ, then λ is called the eigenvalue of A with corresponding eigenvector $\boldsymbol{v}$. The power method being the most simple method to find eigenvalues is used to find the largest eigenvalue λ in magnitude. According to [6] Herman Müntz was attributed the credit of the first use of the power method by Householder in 1913 [10, 11]. It was also published by Richard Edler Von Mises as the Von Mises Iteration in 1929 [15]. The Lanczos method is a little more sophisticated and it is useful to find a few largest eigenvalues. We will see that the Lanczos process is also useful to approximate a function at a certain interval which in turn can be used to approximate zeros.

2.1.1 Power Method

Let $A \in \mathbb{C}^{n\times n}$ be a diagonalizable matrix. Futhermore, let $\lambda_1, ..., \lambda_n$ be the eigenvalues of A and $\boldsymbol{v}_1, ..., \boldsymbol{v}_n$ the corresponding eigenvectors. Assume $|\lambda_1| > |\lambda_2| \geq$

$\ldots \geq |\lambda_n|$. The basic idea behind the power method is to pick an initial vector $\boldsymbol{q}$ and form the sequence

$$\boldsymbol{q}, A\boldsymbol{q}, A^2\boldsymbol{q}, A^3\boldsymbol{q}, \ldots$$

Notice that the vectors $\boldsymbol{v}_k$ form a basis of $\mathbb{C}^n$. So we can write $\boldsymbol{q}$ as follows

$$\boldsymbol{q} = c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \ldots + c_n\boldsymbol{v}_n,$$

where $c_1, \ldots, c_n$ are scalars. Now we form the first term of the sequence

$$\begin{aligned} A\boldsymbol{q} &= c_1A\boldsymbol{v}_1 + c_2A\boldsymbol{v}_2 + \ldots + c_nA\boldsymbol{v}_n \\ &= c_1\lambda_1\boldsymbol{v}_1 + c_2\lambda_2\boldsymbol{v}_2 + \ldots + c_n\lambda_n\boldsymbol{v}_n. \end{aligned} \tag{2.1}$$

Similarly

$$A^2\boldsymbol{q} = c_1\lambda_1^2\boldsymbol{v}_1 + c_2\lambda_2^2\boldsymbol{v}_2 + \ldots + c_n\lambda_n^2\boldsymbol{v}_n.$$

If we continue this process up to k iterations, we obtain

$$A^k\boldsymbol{q} = c_1\lambda_1^k\boldsymbol{v}_1 + c_2\lambda_2^k\boldsymbol{v}_2 + \ldots + c_n\lambda_n^k\boldsymbol{v}_n.$$

If we factor out λ_1^k, we obtain

$$A^k\boldsymbol{q} = \lambda_1^k\left[c_1\boldsymbol{v}_1 + c_2\left(\frac{\lambda_2^k}{\lambda_1^k}\right)\boldsymbol{v}_2 + \ldots + c_n\left(\frac{\lambda_n^k}{\lambda_1^k}\right)\boldsymbol{v}_n\right].$$

Notice that the magnitude of λ_1 is larger than the magnitude of the rest of the eigenvalues. Now, consider the normalized sequence

$$\frac{\boldsymbol{q}}{\|\boldsymbol{q}\|}, \frac{A\boldsymbol{q}}{\|A\boldsymbol{q}\|}, \frac{A^2\boldsymbol{q}}{\|A^2\boldsymbol{q}\|}, \frac{A^3\boldsymbol{q}}{\|A^3\boldsymbol{q}\|}, \ldots$$

The k^{th} term in this sequence can be written as follows

$$\frac{A^k\boldsymbol{q}}{\|A^k\boldsymbol{q}\|} = \frac{\lambda_1^k\left[c_1\boldsymbol{v}_1 + c_2\left(\frac{\lambda_2^k}{\lambda_1^k}\right)\boldsymbol{v}_2 + \ldots + c_n\left(\frac{\lambda_n^k}{\lambda_1^k}\right)\boldsymbol{v}_n\right]}{|\lambda_1^k|\left\|\left[c_1\boldsymbol{v}_1 + c_2\left(\frac{\lambda_2^k}{\lambda_1^k}\right)\boldsymbol{v}_2 + \ldots + c_n\left(\frac{\lambda_n^k}{\lambda_1^k}\right)\boldsymbol{v}_n\right]\right\|}.$$

If we assume $\lambda_1 > 0$ and let $k \to \infty$ then

$$\frac{A^k\boldsymbol{q}}{\|A^k\boldsymbol{q}\|} \to \frac{c_1\boldsymbol{v}_1}{\|c_1\boldsymbol{v}_1\|}$$

Notice that $\boldsymbol{u} := \dfrac{c_1\boldsymbol{v}_1}{\|c_1\boldsymbol{v}_1\|}$ is a multiple of the eigenvector $\boldsymbol{v}_1$. Therefore $\boldsymbol{u}$ is also an eigenvector corresponding to the largest eigenvalue of A. We can now easily find the largest eigenvalue of A by computing the value of $\boldsymbol{u}^\top A\boldsymbol{u}$. In the case when $\lambda_1 < 0$ then the same logic applies to the vector $-\boldsymbol{u}$. Algorithm 2.1 shows the pseudo-code for the power method. We use the convergence criterion proposed by Saad in [13].

Algorithm 2.1 Power Method.

Given an initial vector $\boldsymbol{q}$, this procedure computes the largest eigenvalue λ_1 in magnitude of the matrix A.

```
1: for $k = 1$ to $k_{\mathbf{max}}$ do
2:     $\boldsymbol{y} = A\boldsymbol{q}$
3:     $\boldsymbol{q} = \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|}$
4:     $\lambda_1 = \boldsymbol{q}^\top A\boldsymbol{q}$
5:     if $\|A\boldsymbol{q} - \lambda_1\boldsymbol{q}\| < \epsilon$ then stop
6: end for
7: return $\lambda_1$
```

2.1.2 Lanczos Process

The Lanczos process is based on the similarity transformation

$$T = Q^\top A Q \tag{2.2}$$

where T is a tridiagonal matrix and the columns of Q are the orthonormal vectors $\boldsymbol{q}_1, ..., \boldsymbol{q}_n$. In order to be clear we define $\alpha_1, ..., \alpha_n$ to be the elements in the diagonal of T and $\beta_2, ..., \beta_n$ to be the elements of the subdiagonal as follows

$$T = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & \\ & & \cdots & \cdots & & \\ & & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & & \beta_n & \alpha_n \end{bmatrix}. \tag{2.3}$$

Now, multiply both sides of equation (2.2) by Q to obtain

$$QT = AQ. \tag{2.4}$$

Notice that $\alpha_1 = \boldsymbol{q}_1^\top A \boldsymbol{q}_1$ and $\beta_2 \boldsymbol{q}_2 + \boldsymbol{q}_1 \alpha_1 = A\boldsymbol{q}_1$. Let $2 \leq k \leq n$. Take the k^{th} column in each side and set them equal to each other to obtain

$$\beta_{k+1}\boldsymbol{q}_{k+1} + \boldsymbol{q}_k \alpha_k + \boldsymbol{q}_{k-1}\beta_k = A\boldsymbol{q}_k. \tag{2.5}$$

From equation (2.5) we can infer that $\alpha_k = \boldsymbol{q}_k^\top A \boldsymbol{q}_k$ and $\beta_k = \boldsymbol{q}_{k-1}^\top A \boldsymbol{q}_k$. Now we can define a recurrence relation by rearranging equation (2.5),

$$\boldsymbol{r}_k := \beta_{k+1}\boldsymbol{q}_{k+1} = A\boldsymbol{q}_k - \boldsymbol{q}_k\alpha_k - \boldsymbol{q}_{k-1}\beta_k. \tag{2.6}$$

Notice that $\beta_{k+1} = \|\boldsymbol{r}_k\|$. Then $\boldsymbol{q}_{k+1} = \dfrac{\boldsymbol{r}_k}{\beta_{k+1}}$. Let T_k be any submatrix of T of the form $T(1:k, 1:k)$ and let $Q_k := [\boldsymbol{q}_1, ..., \boldsymbol{q}_k]$. With this information and by letting

$\boldsymbol{e}_k$ be the kth column of the $k \times k$ identity matrix, at the kth step of the Lanczos algorithm we obtain

$$AQ_k = Q_k T_k + \boldsymbol{r}_k \boldsymbol{e}_k^\top. \tag{2.7}$$

Given $\boldsymbol{q}_1$, the recurrence relation (2.7) generates a set of Lanczos vectors $\boldsymbol{q}_2, ..., \boldsymbol{q}_k$ which belong to the Krylov subspace $\mathcal{K}_k(A, \boldsymbol{q}_1)$[3]. This fact is later used in Chapter 3 to reduce our original problem. Algorithm 2.2 depicts the pseudo-code for the Lanczos process [7, 1].

Algorithm 2.2 Lanczos Process.

Given a matrix A, an initial vector $\boldsymbol{q}_1$, and an integer k such that $2 \leq k \leq n$, this procedure generates Q_k and T_k.

```
1: $\boldsymbol{r}_0 = \boldsymbol{q}_1$
2: $\beta_1 = \|\boldsymbol{r}_0\|$
3: $\boldsymbol{q}_0 = 0$
4: for $j = 1, 2, ..., k$ do
5:     $\boldsymbol{q}_j = \boldsymbol{r}_{j-1}/\beta_j$
6:     $\alpha_j = \boldsymbol{q}_j^\top A\boldsymbol{q}_j$
7:     $\boldsymbol{r}_j = A\boldsymbol{q}_j - \alpha_j\boldsymbol{q}_j - \beta_j\boldsymbol{q}_{j-1}$
8:     $\beta_{j+1} = \|\boldsymbol{r}_j\|$
9: end for
10: return $Q_k, T_k$
```

2.2 Root Finding Methods

In this section we describe methods commonly used to find the roots of a function (i.e. find the solution to $f(x) = 0$). The bisection method is the most primitive one whose main idea is to find a midpoint in an interval and determining if the root is above or below this midpoint. With each step the interval is halved and we have a better approximation of where the root is. Another method is the Newton method, which relies on the fact that the slope of the tangent line of the curve can be used to approximate the root faster.

2.2.1 Bisection Method

Suppose that f is a continuous function on the interval $[a, b]$ and $f(a)f(b) < 0$. By the intermediate value theorem, f has at least one zero in the interval $[a, b]$. Now we calculate the midpoint between a and b, namely $c = \dfrac{a+b}{2}$ and then find $f(c)$. If $f(a)f(c) < 0$ then there is a root in $[a, c]$. If $f(b)f(c) < 0$ then there is a root in $[c, b]$. If $f(c) = 0$ then c is a root and we are done. This method is used as a boundary correction step in other root finding methods. Some other root finding methods could potentially approach an unwanted solution or diverge and this can be corrected in the algorithm if we know the boundaries for the sought root. This is an optional step used in Chapter 3. Algorithm 2.3 shows a pseudo-code of the bisection method [8].

Algorithm 2.3 Bisection Method.

Given the initial interval [a,b] such that $f(a)f(b) < 0$, this procedure finds a zero of a function $f(x)$ in the interval.

```
1: $u = f(a)$
2: $v = f(b)$
3: $e = b - a$
for $k = 1$ to $k_{\mathbf{max}}$ do
5:      $e = e/2$
6:      $c = a + e$
7:      $w = f(c)$
8:      if $|e| < \delta$ or $|w| < \epsilon$ then stop
9:      if $sign(w) \neq sign(u)$ then
10:         $b = c$
11:         $v = w$
12:     else
13:         $a = c$
14:         $u = w$
15:     end if
16: end for
17: return $c$
```

2.2.2 Newton Method

For the Newton method we use an initial guess x_0 then we want to take a correction step such that $f(x^*) = 0$ where $x^* = x_0+h$. This implies that $f(x_0+h) = 0$.

Now consider the 1st order Taylor expansion centered at x_0 and its corresponding remainder

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(\eta) = 0, \qquad \eta \in (x_0, x_0+h). \tag{2.8}$$

If we neglect the quadratic term in equation (2.8), then we can approximate the step

$$h \approx -\frac{f(x_0)}{f'(x_0)}.$$

This gives the next approximation in the Newton method

$$x_{n+1} = x_n - \frac{f(x_0)}{f'(x_0)}.$$

The Newton method requires the initial guess to be close to the root sought. In order to correct this problem one can use the bisection method to redirect the approximation to a value closer to the root. A pseudo-code for the Newton method is shown in algorithm 2.4 [8, 2].

Algorithm 2.4 Newton Method.

Given an initial value x, this procedure generates a root of a function $f(x)$.

```
1: for $k = 1$ to $k_{\max}$ do
2:     $d = \frac{f(x)}{f'(x)}$
3:     $x = x - d$
4:     if $|d| < \epsilon$ then stop
5: end for
6: return $x$
```

2.3 Constrained Eigenvalue Methods

In this section we review two of the methods previously used to solve problem (1.4). The first solution was outlined in [4] by Gander, Golub and von Matt in 1989. It involves finding the so-called secular equation and then finding the roots of this equation. The projected power method given in [17] utilizes a similar idea to that of the power method but changing the direction of the updated vector so that it satisfies the constraint $B\boldsymbol{v} = \boldsymbol{c}$. The numerical results of the root finding method and the projected power method are compared in [17]. Both methods can achieve the optimal solution of the problem. In terms of efficiency, the root finding method is faster on smaller size problems whereas the projected power method is faster for larger matrices.

2.3.1 Root Finding Method using the Secular Equation

Use the QR decomposition on $B^\top$. Then

$$\begin{aligned} B^\top &= QR, \\ B &= R^\top Q^\top, \end{aligned} \tag{2.9}$$

where $R^\top \in \mathbb{R}^{m\times n}$ is a lower triangular matrix and $Q \in \mathbb{R}^{n\times n}$ is a orthogonal matrix. Partition $R^\top = \left[\begin{array}{c|c} R_1^\top & 0 \end{array}\right]$, where $R_1^\top \in \mathbb{R}^{m\times m}$ is lower triangular. Once we make the substitution we obtain

$$R^\top Q^\top \boldsymbol{v} = \boldsymbol{c}.$$

Let $\boldsymbol{x} = Q^\top \boldsymbol{v}$. Then $\boldsymbol{x} \in \mathbb{R}^n$ and $\|\boldsymbol{x}\| = 1$. Using this change of variable we obtain

$$R^\top \boldsymbol{x} = \boldsymbol{c}.$$

Let $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix}$, where $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{z} \in \mathbb{R}^{n-m}$. Then

$$R_1^\top \boldsymbol{y} = \boldsymbol{c},$$

$$\boldsymbol{y} = R_1^{-\top}\boldsymbol{c}.$$

Now we enforce the constraint on equation (1.4) using $\boldsymbol{x} = Q^\top \boldsymbol{v}$. We also define $M = Q^\top A Q$. Note that M is an $n \times n$ symmetric matrix, and

$$\begin{aligned} \max_{\boldsymbol{v}\in\mathbb{R}^n} \boldsymbol{v}^\top A \boldsymbol{v} &= \max_{\boldsymbol{x}} \boldsymbol{x}^\top Q^\top A Q \boldsymbol{x} \\ &= \max_{\boldsymbol{x}} \boldsymbol{x}^\top M \boldsymbol{x}. \end{aligned} \tag{2.10}$$

Partition $M = \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^\top & M_{22} \end{bmatrix}$ to obtain

$$\begin{aligned} \boldsymbol{x}^\top M \boldsymbol{x} &= \begin{bmatrix} \boldsymbol{y} & \boldsymbol{z} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^\top & M_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix} \\ &= \boldsymbol{y}^\top M_{11} \boldsymbol{y} + \boldsymbol{z}^\top M_{12}^\top \boldsymbol{y} + \boldsymbol{y}^\top M_{12} \boldsymbol{z} + \boldsymbol{z}^\top M_{22} \boldsymbol{z} \\ &= \boldsymbol{y}^\top M_{11} \boldsymbol{y} + 2\boldsymbol{y}^\top M_{12} \boldsymbol{z} + \boldsymbol{z}^\top M_{22} \boldsymbol{z}. \end{aligned} \tag{2.11}$$

Our problem now consists of solving the following optimization problem

$$\begin{aligned} &\max_{\boldsymbol{z}} \left(\boldsymbol{y}^\top M_{11} \boldsymbol{y} + 2\boldsymbol{y}^\top M_{12} \boldsymbol{z} + \boldsymbol{z}^\top M_{22} \boldsymbol{z}\right) \\ &\text{subject to} \quad \|\boldsymbol{y}\|^2 + \|\boldsymbol{z}\|^2 = 1. \end{aligned} \tag{2.12}$$

Define $\gamma^2 := 1 - \|\boldsymbol{y}\|^2$ and $\boldsymbol{b} := M_{12}^\top \boldsymbol{y}$. Notice that the first term is constant therefore our problem is equivalent to

$$\begin{aligned} &\max_{\boldsymbol{z}} \left(2\boldsymbol{b}^\top \boldsymbol{z} + \boldsymbol{z}^\top M_{22} \boldsymbol{z}\right), \\ &\text{subject to} \quad \|\boldsymbol{z}\|^2 = \gamma^2. \end{aligned} \tag{2.13}$$

The Lagrangian of the optimization problem (2.13) is

$$\mathcal{L}(\lambda, \boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^\top M_{22} \boldsymbol{z} + \boldsymbol{z}^\top \boldsymbol{b} - \frac{\lambda}{2}(\|\boldsymbol{z}\|^2 - \gamma^2).$$

Now we take the partial derivatives of $\mathcal{L}$ with respect to $\boldsymbol{z}$ and λ to obtain the following equations.

$$\begin{aligned} (M_{22} - \lambda I)\boldsymbol{z} =& - \boldsymbol{b}, \\ \|\boldsymbol{z}\| =& \gamma^2. \end{aligned} \tag{2.14}$$

From the first equation in (2.14),

$$\boldsymbol{z} = -(M_{22} - \lambda I)^{-1}\boldsymbol{b}.$$

Then we substitute it into the second equation in (2.14) to obtain

$$f(\lambda) := \boldsymbol{b}^\top (M_{22} - \lambda I)^{-2}\boldsymbol{b} - \gamma^2 = 0.$$

We call $f(\lambda)$ the *secular function.* Now we use the eigendecomposition $M_{22} = W\Lambda W^\top$ where $\Lambda \in \mathbb{R}^{(n-m)\times(n-m)}$ is a diagonal matrix with diagonal entries δ_i. We define $\boldsymbol{d} := W^\top \boldsymbol{b}$ where $\boldsymbol{d} = [d_1, d_2, ..., d_{n-m}]^\top$. Then

$$\begin{aligned} f(\lambda) =& \boldsymbol{b}^\top (M_{22} - \lambda I)^{-2}\boldsymbol{b} - \gamma^2 \\ =& \boldsymbol{b}^\top (W\Lambda W^\top - \lambda I)^{-2}\boldsymbol{b} - \gamma^2 \\ =& \boldsymbol{b}^\top W(\Lambda - \lambda I)^{-2} W^\top \boldsymbol{b} - \gamma^2 \\ =& \boldsymbol{d}^\top (\Lambda - \lambda I)^{-2}\boldsymbol{d} - \gamma^2 \\ =& \sum_{j=1}^{n-m} \left(\frac{d_i}{\delta_i - \lambda}\right)^2 - \gamma^2. \end{aligned} \tag{2.15}$$

Now we want to calculate the largest zero of the *explicit secular function* (2.15). This is a rational equation and some methods to find the largest root have been studied extensively in [4].

2.3.2 Projected Power Method

This method was first presented in [17] by Xu, Li and Schuurmans. It also solves problem (1.4). The main idea of this method is to combine the power method with a projection step that ensures the resulting vector satisfies the constraint. We begin by defining the projection matrix $P := I - B^\top(BB^\top)^{-1}B$ and the vector $\boldsymbol{n}_0 := B^\top(BB^\top)^{-1}\boldsymbol{c}$. Notice that, for $B\boldsymbol{v} = \boldsymbol{c}$,

$$\begin{aligned} P\boldsymbol{v} &= \big[I - B^\top(BB^\top)^{-1}B)\boldsymbol{v}\big] \\ &= \boldsymbol{v} - B^\top(BB^\top)^{-1}B\boldsymbol{v} \\ &= \boldsymbol{v} - B^\top(BB^\top)^{-1}\boldsymbol{c} \\ &= \boldsymbol{v} - \boldsymbol{n}_0. \end{aligned} \tag{2.16}$$

Now we use these definitions in order to transform the optimization problem. Notice that

$$\begin{aligned} \boldsymbol{n}_0 &= \boldsymbol{v} - P\boldsymbol{v}, \\ \|\boldsymbol{n}_0\|^2 &= \|\boldsymbol{v}\|^2 - 2\boldsymbol{v}^\top P\boldsymbol{v} + \boldsymbol{v}^\top P\boldsymbol{v}, \\ \|\boldsymbol{n}_0\|^2 &= 1 - \boldsymbol{v}^\top P\boldsymbol{v}, \\ \boldsymbol{v}^\top PP\boldsymbol{v} &= 1 - \|\boldsymbol{n}_0\|^2, \\ \|P\boldsymbol{v}\|^2 &= 1 - \|\boldsymbol{n}_0\|^2. \end{aligned} \tag{2.17}$$

We will now define $\gamma := \sqrt{1 - \|\boldsymbol{n}_0\|^2}$ and $\boldsymbol{b}_0 := PA\boldsymbol{n}_0$. Then

$$\begin{aligned} \max_{\boldsymbol{v}} \boldsymbol{v}^\top A\boldsymbol{v} &= \max_{\boldsymbol{v}}[(\boldsymbol{n}_0 + P\boldsymbol{v})^\top A(\boldsymbol{n}_0 + P\boldsymbol{v})] \\ &= \max_{\boldsymbol{v}}[\boldsymbol{n}_0{}^\top A\boldsymbol{n}_0 + 2\boldsymbol{v}^\top PA\boldsymbol{n}_0 + \boldsymbol{v}^\top PAP\boldsymbol{v}] \\ &= \max_{\boldsymbol{v}}[\boldsymbol{n}_0{}^\top A\boldsymbol{n}_0 + \boldsymbol{v}^\top PAP\boldsymbol{v} + 2\boldsymbol{v}^\top P\boldsymbol{b}_0]. \end{aligned} \tag{2.18}$$

Notice that the value of the first term is constant therefore we only need to find the vector $\boldsymbol{v}$ that solves

$$\max_{\boldsymbol{v}}[\boldsymbol{v}^\top PAP\boldsymbol{v} + 2\boldsymbol{v}^\top P\boldsymbol{b}_0] \quad \text{subject to} \quad \|P\boldsymbol{v}\|^2 = \gamma^2. \tag{2.19}$$

The Lagrangian of the optimization problem (2.19) is

$$\mathcal{L}(\lambda, \boldsymbol{v}) = \frac{1}{2}\boldsymbol{v}^\top PAP\boldsymbol{v} + \boldsymbol{v}^\top P\boldsymbol{b}_0 - \frac{\lambda}{2}(\|P\boldsymbol{v}\|^2 - \gamma^2).$$

Now we take the partial derivatives of $\mathcal{L}$ with respect to $\boldsymbol{v}$ and λ to obtain the following equations.

$$\begin{aligned}(PA - \lambda I)P\boldsymbol{v} =& - \boldsymbol{b}_0,\\ \|P\boldsymbol{v}\|^2 =& \gamma^2.\end{aligned} \tag{2.20}$$

The first equation in (2.20) needs to be satisfied in order to achieve the critical point of the optimization problem (2.19). From the first equation in (2.20), the definition of $\boldsymbol{b}_0$, and assuming $\lambda > 0$ we obtain

$$\begin{aligned}(PA - \lambda I)P\boldsymbol{v} &= -\boldsymbol{v}_0,\\ PAP\boldsymbol{v} - \lambda P\boldsymbol{v} &= -PA\boldsymbol{n}_0,\\ PAP\boldsymbol{v} + PA\boldsymbol{n}_0 &= \lambda P\boldsymbol{v},\\ PA(P\boldsymbol{v} + n_0) &= \lambda P\boldsymbol{v},\\ PA\boldsymbol{v} &= \lambda P\boldsymbol{v},\\ \|PA\boldsymbol{v}\| &= \lambda \|P\boldsymbol{v}\|,\\ \frac{\|PA\boldsymbol{v}\|}{\|P\boldsymbol{v}\|} &= \lambda.\end{aligned} \tag{2.21}$$

Therefore $\lambda = \dfrac{\|PA\boldsymbol{v}\|}{\gamma}$. This fact is used together with the first equation in (2.20) to check the convergence of the vector $\boldsymbol{v}_{k+1}$ in Algorithm 2.5.

The projected power method can be understood intuitively in each step where $\boldsymbol{v}_k$ is multiplied by the matrix A. Then we multiply it by the projection matrix P. This will project the said vector onto the hyperplane $B\boldsymbol{v} = \boldsymbol{c}$. This vector is then re-scaled to the length γ and added to $\boldsymbol{n}_0$. This ensures that the resulting vector satisfies both constraints. The steps of this algorithm are illustrated in Figure 2.1. This algorithm is guaranteed to converge to the globally optimal solution of problem (1.4). This fact has been proven in [17].

Algorithm 2.5 Projected Power Method.

Solves problem (1.4).

```
1: $P = I - B^\top(BB^\top)^{-1}B$
2: $\boldsymbol{n}_0 = B^\top(BB^\top)^{-1}\boldsymbol{c}$
3: $\gamma = \sqrt{1 - \|\boldsymbol{n}_0\|^2}$
4: $\boldsymbol{v}_0 = \gamma\frac{PA\boldsymbol{n}_0}{\|PA\boldsymbol{n}_0\|} + \boldsymbol{n}_0$
5: for $k = 1$ to $k_{\mathbf{max}}$ do
6:     $\boldsymbol{u}_{k+1} = \gamma\frac{PA\boldsymbol{v}_k}{\|PA\boldsymbol{v}_k\|}$
7:     $\boldsymbol{v}_{k+1} = \boldsymbol{u}_{k+1} + \boldsymbol{n}_0$
8:     if $\boldsymbol{v}_{k+1}$ converges then stop
9: end for
10: return $\boldsymbol{v}_{k+1}$
```

The process in Algorithm 2.5 is illustrated in a 3-dimensional plot in Figure 2.1.

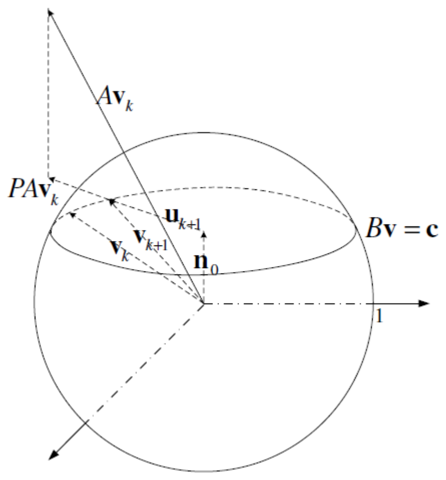

Figure 2.1. A geometric view of the projected power method [17].

CHAPTER 3

Krylov Subspace Method

3.1 Preliminaries

In this section we study a new solution to problem (1.4) in hopes to find a faster algorithm. We develop a new method of solving this problem by using the Krylov subspace. The problem is restated here for convenience.

Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive semi-definite (s.p.s.d) matrix,

$$n > m, \quad B \in \mathbb{R}^{m\times n}, \quad \mathbf{c} \in \mathbb{R}^m.$$

$$\text{Find} \qquad \max_{\boldsymbol{v}\in\mathbb{R}^n} \boldsymbol{v}^\top A\boldsymbol{v} \qquad \text{subject to} \qquad ||\boldsymbol{v}|| = 1, \qquad B\boldsymbol{v} = \boldsymbol{c}.$$

3.2 Transform the Constrained Eigenvalue problem into a zero-finding problem

In order to transform our problem we again compute the QR decomposition of $B^\top$,

$$\begin{aligned} B^\top &= QR, \\ B &= R^\top Q^\top \\ &= \left[\begin{array}{c|c} R_1^\top & 0 \end{array}\right]\left[\begin{array}{c} Q_1^\top \\ \hline Q_2^\top \end{array}\right] \\ &= R_1^\top Q_1^\top. \end{aligned}$$

Introduce a projection matrix P,

$$P = I - Q_1 Q_1^\top.$$

This projection matrix is used to define the vectors $\boldsymbol{n}_0 := Q_1 R_1^{-\top}\boldsymbol{c}$ and $\boldsymbol{b}_0 := PA\boldsymbol{n}_0$. Notice that $P\boldsymbol{v} = \boldsymbol{v} - \boldsymbol{n}_0$. Now we use these definitions in order to transform the optimization problem. Notice that $\|P\boldsymbol{v}\|^2 = 1 - \|\boldsymbol{n}_0\|^2$. We will now define $\gamma := \sqrt{1 - \|\boldsymbol{n}_0\|^2}$. We can now transform problem (1.4) equivalently to

$$\max_{\boldsymbol{v}}[\boldsymbol{v}^\top PAP\boldsymbol{v} + 2\boldsymbol{v}^\top P\boldsymbol{b}_0] \quad \text{subject to} \quad \|P\boldsymbol{v}^2\| = \gamma^2. \tag{3.1}$$

The Lagrangian of the optimization problem (3.1) is

$$\mathcal{L}(\lambda, \boldsymbol{v}) = \frac{1}{2}\boldsymbol{v}^\top PAP\boldsymbol{v} + \boldsymbol{v}^\top P\boldsymbol{b}_0 - \frac{\lambda}{2}(\|P\boldsymbol{v}\|^2 - \gamma^2).$$

Now we take the partial derivatives of $\mathcal{L}$ with respect to $\boldsymbol{v}$ and λ to obtain the following equations.

$$\begin{aligned}(PA - \lambda I)P\boldsymbol{v} =& -\boldsymbol{b}_0,\\ \|P\boldsymbol{v}\|^2 =& \gamma^2.\end{aligned} \tag{3.2}$$

Now we manipulate the first equation in (3.2) by using the properties of P.

$$\begin{aligned}(PA - \lambda I)P\boldsymbol{v} =& PAP\boldsymbol{v} - \lambda P\boldsymbol{v}\\ =& PAP^2\boldsymbol{v} - \lambda P\boldsymbol{v}\\ =& (PAP - \lambda I)P\boldsymbol{v}.\end{aligned} \tag{3.3}$$

Now we combine the two equations in (3.2) in order to obtain a zero-finding problem of a scalar function. Notice that,

$$P\boldsymbol{v} = -(PAP - \lambda I)^{-1}\boldsymbol{b}_0.$$

Then we substitute in the second equation to obtain

$$f(\lambda) := \boldsymbol{b}_0{}^\top(PAP - \lambda I)^{-2}\boldsymbol{b}_0 - \gamma^2 = 0. \tag{3.4}$$

3.3 Using Lanczos Method to Obtain a Reduced Secular Function

First we begin by defining $C := PAP$ in order to reduce our function. Let λ_i be a given approximation of the largest zero of the secular function. The reduction process on $C_{\lambda_i} = C - \lambda_i I$ begins by using the Lanczos process to generate the Krylov subspace $\mathcal{K}_k(C_{\lambda_i}^{-1}, C_{\lambda_i}^{-1}\boldsymbol{b}_0)$. We run the Lanczos process up to k steps and we have

$$C_{\lambda_i}^{-1} Y_k = Y_k H_k + H_{(k+1,k)} \boldsymbol{y}_{k+1} \mathbf{e}_k{}^\top \text{ and } Y_k{}^\top C_{\lambda_i}^{-1} Y_k = H_k,$$

where $Y_k = [\boldsymbol{y}_1, ..., \boldsymbol{y}_k]$ and $H_k \in \mathbb{R}^{k\times k}$ is a tridiagonal matrix. Now we compute the matrix $T_k = Y_k{}^\top C Y_k$ to write our reduced function $f_k(\lambda)$ in the same fashion similarly to [18].

$$f_k(\lambda) := \boldsymbol{b}_0{}^\top Y_k (T_k - \lambda I)^{-2} Y_k{}^\top \boldsymbol{b}_0 - \gamma^2. \tag{3.5}$$

The *reduced function* $f_k(\lambda)$ is an approximation of the secular function $f(\lambda)$ in (3.4). Now we use the eigendecomposition $T_k = W_k \Lambda W_k{}^\top$ where $\Lambda \in \mathbb{R}^{k\times k}$ is a diagonal matrix with diagonal entries $\delta_1 \leq \delta_2 \leq ... \leq \delta_k$ and $W_k \in \mathbb{R}^{k\times k}$ is orthogonal. We define $\boldsymbol{d} = W_k{}^\top Y_k{}^\top \boldsymbol{b}_0$ where $\boldsymbol{d} = [d_1, d_2, ..., d_k]^\top$. Then

$$\begin{aligned} f_k(\lambda) =& \boldsymbol{b}_0{}^\top Y_k (T_k - \lambda I)^{-2} Y_k{}^\top \boldsymbol{b}_0 - \gamma^2 \\ =& \boldsymbol{b}_0{}^\top Y_k (W_k \Lambda W_k{}^\top - \lambda I)^{-2} Y_k{}^\top \boldsymbol{b}_0 - \gamma^2 \\ =& \boldsymbol{b}_0{}^\top Y_k W_k (\Lambda - \lambda I)^{-2} W_k{}^\top Y_k{}^\top \boldsymbol{b}_0 - \gamma^2 \\ =& \boldsymbol{d}^\top (\Lambda - \lambda I)^{-2} \boldsymbol{d} - \gamma^2 \\ =& \sum_{j=1}^{k} \left(\frac{d_j}{\delta_j - \lambda} \right)^2 - \gamma^2. \end{aligned} \tag{3.6}$$

Later on we will describe a zero finding method to find the zeros of the *explicit secular function* (3.6). It has been shown in [9] that this approximation has a moment matching property:

$$f(\lambda) - f_k(\lambda) = O(|\lambda - \lambda_i|^k).$$

The reader is referred to [9] for details on how to prove this fact. Now we want to calculate the largest zero of the explicit reduced function (3.6). Figure 3.1 shows the behavior of the reduced function. We use a rational replacement function in order to do this.

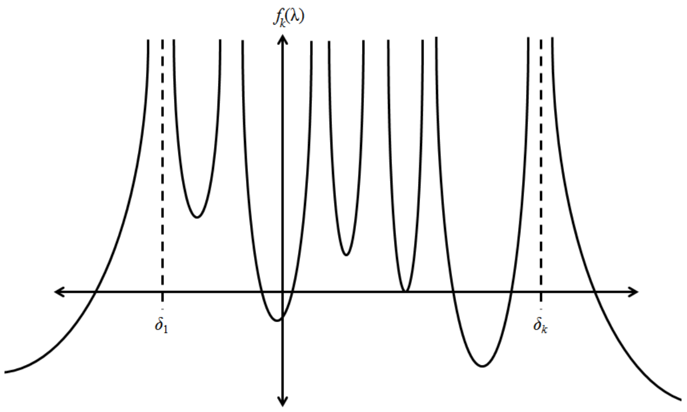

Figure 3.1. Graph of the secular function.

3.4 Finding the Zero by using a Rational Replacement Function

We begin by finding the derivative of (3.6),

$$f_k'(\lambda) = \sum_{j=1}^{k} \frac{2{d_j}^2}{(\delta_j - \lambda)^3}.$$

We want our replacement function to be similar after the largest eigenvalue of T_k so we will use the function

$$g(\lambda) = \frac{a}{(\delta_k - \lambda)^2} - c,$$

then we match the replacement function with the reduced function at a given approximation λ_i. That is

$$g(\lambda_i) = f_k(\lambda_i)$$

and

$$g'(\lambda_i) = f_k'(\lambda_i).$$

The zero of $g(\lambda)$ will determine the next approximation λ_{i+1}. By doing some algebra we obtain

$$a = \frac{1}{2}(\delta_k - \lambda_i)^3 f_k'(\lambda_i),$$
$$c = \frac{1}{2}(\delta_k - \lambda_i) f_k'(\lambda_i) - f_k(\lambda_i).$$

By solving $g(\lambda) = 0$ we obtain $\lambda_{i+1} = \delta_k + \sqrt{\frac{a}{c}}$. Notice that we use the solution with addition because we want to find the solution at the right of the largest eigenvalue of T_k. This expression is equivalent to

$$\lambda_{i+1} = \delta_k + \sqrt{\frac{\frac{1}{2}(\delta_k - \lambda_i)^3 f_k'(\lambda_i)}{\frac{1}{2}(\delta_k - \lambda_i) f_k'(\lambda_i) - f_k(\lambda_i)}}. \tag{3.7}$$

Now we give the algorithm that finds the zero of the reduced function (3.6). Notice that the iterations in this algorithm will be used later in the algorithm that generates the explicit reduced function (3.6).

Algorithm 3.1 Zero Finder

Given an initial value λ_0 close enough to the largest root of a rational function of the form $f_k(\lambda) = \sum_{j=1}^{k} \left(\frac{d_j}{\delta_j - \lambda} \right)^2 - \gamma^2$, this procedure approximates the largest root.

```
1: for $i = 1$ to $i_{\max}$ do
2:     $b = \delta_k - \lambda_i$
3:     $df = f_k'(\lambda_i)$
4:     $\lambda_{i+1} = \delta_k + \sqrt{\frac{\frac{1}{2} b^3 (df)}{\frac{1}{2} b(df) - f_k(\lambda_i)}}$
5:     if $f(\lambda_{i+1}) < \epsilon$ then stop
6: end for
7: return $\lambda_{i+1}$
```

In the previous algorithm we use the index i to iterate so that it is not confused with the outer loop of the next algorithm that we are going to present (Algorithm 3.2).

Notice that the largest zero of the reduced function (3.5) is larger than the largest eigenvalue of the matrix C. We then use the largest eigenvalue of the matrix C as the lower bound of our algorithm. Also notice that the zero is the solution to problem (3.1). This value is bounded above by the largest eigenvalue of A. Now we have an upper bound and a lower bound we can use to find an initial guess for our algorithm. We pick the midpoint between the upper bound and the lower bound as our initial guess.

Algorithm 3.2 Krylov Subspace Method

Solves problem (1.4).

```
1: $QR = \text{qr}(B^\top, 0)$
2: $\boldsymbol{n}_0 = Q(R^{-\top}\boldsymbol{c})$
3: $\gamma = \sqrt{1 - \|n_0\|^2}$
4: $\boldsymbol{t} = A\boldsymbol{n}_0$
5: $\boldsymbol{b}_0 = \boldsymbol{t} - Q(Q^\top \boldsymbol{t})$
6: $C = A - Q(Q^\top A) - (AQ)Q^\top + Q(Q^\top AQ)Q^\top$
7: $a = \max(\text{eigs}(C))$
8: $b = \max(\text{eigs}(A))$
9: $\lambda_0 = (a+b)/2$
10: for $i = 0$ to $i_{\mathbf{max}}$ do
11:     $C_{\lambda_i} = C - \lambda_i I$
12:     $[Y_k, H_k] = \text{lanczos}(C_{\lambda_i}^{-1}, C_{\lambda_i}^{-1}\boldsymbol{b}_0)$ (use Algorithm 2.2)
13:     $T_k = {Y_k}^\top C Y_k$
14:     $\lambda_{i+1} = \text{zerofinder}(\lambda_i, f_k(\lambda))$ (use Algorithm 3.1)
15:     if $f(\lambda_{i+1}) < \epsilon$ then stop
16: end for
17: return $\lambda_{i+1}$
```

In Algorithm 3.2 we use the index i to iterate the outer loop. Notice that whenever we call Algorithm 3.1 (at line 14) the variable λ_i is considered to be the initial λ_0. The inner loop runs and Algorithm 3.1 outputs a new approximation λ_{i+1} which is then checked for convergence at line 15.

CHAPTER 4

Generalized Constraint Eigenvalue Problem

We have studied several algorithms to solve (1.4). In this chapter we study an extension of the original problem by adding a degree of freedom to the linear constraint. This problem is stated in Chapter 1 in (1.6). Here we restate the problem for convenience.

$$\text{Find} \qquad \max_{\boldsymbol{v}\in\mathbb{R}^n} \boldsymbol{v}^\top A\boldsymbol{v} \qquad \text{subject to} \qquad ||\boldsymbol{v}|| = 1, \qquad B\boldsymbol{v} \in \{\alpha\boldsymbol{c} | \alpha \in \mathbb{R}\}.$$

We will show the progress we have made as to understand this problem in hopes to find a solution and at the same time find an effective algorithm.

4.1 Singular Value Decomposition

We start out by expressing our constraint

$$B\boldsymbol{v} \in \{\alpha\boldsymbol{c} | \alpha \in \mathbb{R}\}$$

as the following:

$$B\boldsymbol{v} = \alpha\boldsymbol{c}, \text{ where } \alpha \text{ varies in } \mathbb{R}.$$

We use Singular Value Decomposition (SVD) on B in order to understand how much information the constraint provides. At the same time we will use our results from SVD to simplify our eigenvalue problem. Let

$$B = U\Sigma V^\top, \tag{4.1}$$

where $U \in \mathbb{R}^{m\times m}$ is orthogonal, Σ is a diagonal matrix with diagonal entries $\sigma_i \geq 0$ for $i = 1, 2..., m$ and $V \in \mathbb{R}^{n\times n}$ is also orthogonal. Now we make the substitution and obtain

$$U\Sigma V^\top \boldsymbol{v} = \alpha \boldsymbol{c}.$$

Let $\boldsymbol{x} = V^\top \boldsymbol{v}$. Then $\boldsymbol{x} \in \mathbb{R}^n$ and $\|\boldsymbol{x}\| = 1$. Using this change of variable, we obtain

$$U\Sigma\boldsymbol{x} = \alpha\boldsymbol{c},$$

$$\Sigma\boldsymbol{x} = \alpha U^\top \boldsymbol{c}.$$

Let $\Sigma = \left[\begin{array}{c|c} \Sigma_1 & 0 \end{array}\right]$ where $\Sigma_1 \in \mathbb{R}^{m\times m}$ is a diagonal matrix with diagonal entries σ_i. Then

$$\Sigma_1\boldsymbol{y} = \alpha U^\top \boldsymbol{c},$$

$$\boldsymbol{y} = \alpha\Sigma_1^{-1}U^\top \boldsymbol{c}.$$

Let $\boldsymbol{w} = \Sigma_1^{-1}U^\top\boldsymbol{c}$ (notice that this is a known vector with m entries). Then

$$\boldsymbol{y} = \alpha\boldsymbol{w}. \tag{4.2}$$

4.2 Constraint Enforcement

Now we enforce the constraint in (1.6) using $\boldsymbol{x} = V^\top\boldsymbol{v}$ and (4.2). We also define $M = V^\top AV$. Note that M is an $n \times n$ symmetric matrix.

$$\begin{aligned} \max_{\boldsymbol{v}\in\mathbb{R}^n} \boldsymbol{v}^\top A\boldsymbol{v} &= \max_{\boldsymbol{x}} \boldsymbol{x}^\top V^\top AV\boldsymbol{x} \\ &= \max_{\boldsymbol{x}} \boldsymbol{x}^\top M\boldsymbol{x}. \end{aligned} \tag{4.3}$$

Partition $M = \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^\top & M_{22} \end{bmatrix}$ to obtain

$$
\begin{aligned}
\boldsymbol{x}^\top M \boldsymbol{x} &= \begin{bmatrix} \boldsymbol{y}^\top & \boldsymbol{z}^\top \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^\top & M_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix} \\
&= \boldsymbol{y}^\top M_{11} \boldsymbol{y} + \boldsymbol{z}^\top M_{12}^\top \boldsymbol{y} + \boldsymbol{y}^\top M_{12} \boldsymbol{z} + \boldsymbol{z}^\top M_{22} \boldsymbol{z} \qquad (4.4) \\
&= \boldsymbol{y}^\top M_{11} \boldsymbol{y} + 2\boldsymbol{y}^\top M_{12} \boldsymbol{z} + \boldsymbol{z}^\top M_{22} \boldsymbol{z} \\
&= \alpha^2 \boldsymbol{w}^\top M_{11} \boldsymbol{w} + 2\alpha \boldsymbol{w}^\top M_{12} \boldsymbol{z} + \boldsymbol{z}^\top M_{22} \boldsymbol{z}.
\end{aligned}
$$

Our problem now consists of solving the following optimization problem

$$
\begin{aligned}
&\max_{\boldsymbol{z},\alpha} \left(\alpha^2 \boldsymbol{w}^\top M_{11} \boldsymbol{w} + 2\alpha \boldsymbol{w}^\top M_{12} \boldsymbol{z} + \boldsymbol{z}^\top M_{22} \boldsymbol{z} \right), \\
&\text{subject to} \quad \alpha^2 \|\boldsymbol{w}\|^2 + \|\boldsymbol{z}\|^2 = 1.
\end{aligned} \qquad (4.5)
$$

4.3 Reduction to Eigenvalue problem

We now use (4.4) to simplify the problem into an eigenvalue problem with an $(m+1)$-vector.

$$
\begin{aligned}
\boldsymbol{x}^\top M \boldsymbol{x} &= \alpha^2 \boldsymbol{w}^\top M_{11} \boldsymbol{w} + 2\alpha \boldsymbol{w}^\top M_{12} \boldsymbol{z} + \boldsymbol{z}^\top M_{22} \boldsymbol{z} \\
&= \begin{bmatrix} \alpha\|\boldsymbol{w}\| \\ \boldsymbol{z} \end{bmatrix}^\top \begin{bmatrix} \dfrac{\boldsymbol{w}^\top M_{11} \boldsymbol{w}}{\|\boldsymbol{w}\|^2} & \dfrac{\boldsymbol{w}^\top M_{12}}{\|\boldsymbol{w}\|} \\ \dfrac{M_{12}^\top \boldsymbol{w}}{\|\boldsymbol{w}\|} & M_{22} \end{bmatrix} \begin{bmatrix} \alpha\|\boldsymbol{w}\| \\ \boldsymbol{z} \end{bmatrix}.
\end{aligned} \qquad (4.6)
$$

Let $C = \begin{bmatrix} \dfrac{\boldsymbol{w}^\top M_{11} \boldsymbol{w}}{\|\boldsymbol{w}\|^2} & \dfrac{\boldsymbol{w}^\top M_{12}}{\|\boldsymbol{w}\|} \\ \dfrac{M_{12}^\top \boldsymbol{w}}{\|\boldsymbol{w}\|} & M_{22} \end{bmatrix}$. Then our problem can now be rewritten as follows:

$$
\max \begin{bmatrix} \alpha\|\boldsymbol{w}\| \\ \boldsymbol{z} \end{bmatrix}^\top C \begin{bmatrix} \alpha\|\boldsymbol{w}\| \\ \boldsymbol{z} \end{bmatrix} \text{ subject to } \left\| \begin{bmatrix} \alpha\|\boldsymbol{w}\| \\ \boldsymbol{z} \end{bmatrix} \right\| = 1,
$$

where $\begin{bmatrix} \alpha\|\boldsymbol{w}\| \\ \boldsymbol{z} \end{bmatrix}$ varies in $\mathbb{R}^{n-m+1}$. Since the vector $\begin{bmatrix} \alpha\|\boldsymbol{w}\| \\ \boldsymbol{z} \end{bmatrix}$ is a vector of length 1 by (4.5) we can now maximize $\boldsymbol{x}^\top M\boldsymbol{x}$ by finding the maximum eigenvalue of C which, at the same time, is the solution to (1.6).

Algorithm 4.1 SVD Method.

Given an initial vector $\boldsymbol{t}_1 = [t_1^{(1)}, t_2^{(1)}, ..., t_{n-m+1}^{(1)}]$, this procedure solves (1.6)

```
1: $[U, \Sigma, V] = \text{svd}(B)$
2: $M = V^\top AV$
3: $\boldsymbol{w} = \Sigma_1^{-1}(U^\top \boldsymbol{c})$
4: $C = \begin{bmatrix} \dfrac{\boldsymbol{w}^\top M_{11}\boldsymbol{w}}{\|\boldsymbol{w}\|^2} & \dfrac{\boldsymbol{w}^\top M_{12}}{\|\boldsymbol{w}\|} \\ \dfrac{M_{12}^\top \boldsymbol{w}}{\|\boldsymbol{w}\|} & M_{22} \end{bmatrix}$
5: for $k = 1$ to $k_{\max}$ do
6:     $\boldsymbol{t}_{k+1} = \dfrac{C\boldsymbol{t}_k}{\|C\boldsymbol{t}_k\|}$
7:     if $\boldsymbol{t}_{k+1}$ converges then stop
8: end for
9: $m = \boldsymbol{t}_{k+1}^\top C\boldsymbol{t}_{k+1}$
10: $\boldsymbol{y} = \dfrac{t_1^{(k+1)}}{\|\boldsymbol{w}\|}\boldsymbol{w}$
11: $\boldsymbol{z} = \left[t_2^{(k+1)}, t_3^{(k+1)}, ..., t_{n-m+1}^{(k+1)}\right]^\top$
12: $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix}$
13: $\boldsymbol{v} = V\boldsymbol{x}$
14: return $m$ and $\boldsymbol{v}$
```

4.4 Using Economy SVD

One of the shortcomings in the SVD method is the high computational cost of line 1 and line 2 in Algorithm 4.1. In order to avoid this problem we propose a partition of the matrix V. Partition $V = \left[\begin{array}{c|c} V_1 & V_2 \end{array}\right]$ where V_1 is an $n \times m$ matrix and V_2 is an $n \times (n-m)$ to obtain

$$\begin{aligned} B &= U\Sigma V^\top \\ &= U\left[\begin{array}{c|c} \Sigma_1 & 0 \end{array}\right]\left[\begin{array}{c} V_1^\top \\ \hline V_2^\top \end{array}\right] \\ &= U\left[\Sigma_1 V_1^\top + 0\right] \\ &= U\Sigma_1 V_1^\top. \end{aligned}$$

These matrices $(U, \Sigma_1$ and $V_1)$ will represent our new decomposition called the *economy SVD*. Notice that only m columns of V are computed which saves time but V_2 **will be unavailable**. We will use an orthogonal projection in order to get around this obstacle.

Using the partition of V we obtain

$$\boldsymbol{v} = V\boldsymbol{x} = \left[\begin{array}{c|c} V_1 & V_2 \end{array}\right]\left[\begin{array}{c} \boldsymbol{y} \\ \boldsymbol{z} \end{array}\right] = V_1\boldsymbol{y} + V_2\boldsymbol{z} = \alpha V_1\boldsymbol{w} + V_2\boldsymbol{z}. \tag{4.7}$$

Now we define

$$\hat{\boldsymbol{w}} = V_1\boldsymbol{w} \text{ and } \hat{\boldsymbol{z}} = V_2\boldsymbol{z}. \tag{4.8}$$

Notice that $\hat{\boldsymbol{z}}$ is in the subspace generated by V_2 (not available). Then

$$\boldsymbol{v} = \alpha V_1\boldsymbol{w} + V_2\boldsymbol{z} = \alpha\hat{\boldsymbol{w}} + \hat{\boldsymbol{z}}. \tag{4.9}$$

Since V_2 is not available when using the economy SVD we will use the following orthogonal projection onto the column space of V_2

$$P_{V_2} = V_2 V_2^\top = I - V_1 V_1^\top. \tag{4.10}$$

Then we have $\hat{\boldsymbol{z}} \in \{P_{V_2}\tilde{\boldsymbol{z}} | \tilde{\boldsymbol{z}} \in \mathbb{R}^n\}$.

Use (4.9), make the substitution and use the fact that $P = P^\top = P^2$ in the original problem to obtain the following:

$$\begin{aligned}
\boldsymbol{v}^\top A \boldsymbol{v} &= (\alpha\hat{\boldsymbol{w}} + \hat{\boldsymbol{z}})^\top A (\alpha\hat{\boldsymbol{w}} + \hat{\boldsymbol{z}}) \\
&= \alpha^2 \hat{\boldsymbol{w}}^\top A \hat{\boldsymbol{w}} + 2\alpha \hat{\boldsymbol{w}}^\top A \hat{\boldsymbol{z}} + \hat{\boldsymbol{z}} A \hat{\boldsymbol{z}} \\
&= \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ \hat{\boldsymbol{z}} \end{bmatrix}^\top \begin{bmatrix} \frac{\hat{\boldsymbol{w}}^\top A \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \frac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \frac{A^\top \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix} \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ \hat{\boldsymbol{z}} \end{bmatrix} \\
&= \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ P_{V_2}\tilde{\boldsymbol{z}} \end{bmatrix}^\top \begin{bmatrix} \frac{\hat{\boldsymbol{w}}^\top A \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \frac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \frac{A^\top \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix} \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ P_{V_2}\tilde{\boldsymbol{z}} \end{bmatrix} \\
&= \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}} \end{bmatrix}^\top \begin{bmatrix} 1 & 0 \\ 0 & P_{V_2} \end{bmatrix}^\top \begin{bmatrix} \frac{\hat{\boldsymbol{w}}^\top A \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \frac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \frac{A^\top \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P_{V_2} \end{bmatrix} \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}} \end{bmatrix}.
\end{aligned} \tag{4.11}$$

Let

$$C_V = \begin{bmatrix} 1 & 0 \\ 0 & P_{V_2} \end{bmatrix}^\top \begin{bmatrix} \frac{\hat{\boldsymbol{w}}^\top A \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \frac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \frac{A^\top \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P_{V_2} \end{bmatrix}.$$

Notice that the vector $\begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}} \end{bmatrix}$ may not be of length 1. We claim that the resulting maximum eigenvalue λ^* of the matrix C_V is in fact the solution to the original problem.

To show this we let $C = \begin{bmatrix} \frac{\hat{\boldsymbol{w}}^\top A\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \frac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \frac{A^\top\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix}$ and $\beta = \alpha\|\hat{\boldsymbol{w}}\|$ then we may write the original problem as follows:

$$\max_{\tilde{\boldsymbol{z}}} \begin{bmatrix} \beta \\ \hat{\boldsymbol{z}} \end{bmatrix}^\top C \begin{bmatrix} \beta \\ \hat{\boldsymbol{z}} \end{bmatrix} \qquad \text{subject to} \qquad \beta^2 + \|\hat{\boldsymbol{z}}\|^2 = 1, \qquad \hat{\boldsymbol{z}} \in \{P_{V_2}\tilde{\boldsymbol{z}} | \tilde{\boldsymbol{z}} \in \mathbb{R}^n\}.$$

Theorem 1 *Let $C \in \mathbb{R}^{(n+1)\times(n+1)}$ be a symmetric matrix and $P \in \mathbb{R}^{n\times n}$ be a projection matrix. If λ^* is the maximum eigenvalue of $C_P := \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix} C \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix}$ then,*

$$\lambda^* = \max_{\tilde{\boldsymbol{z}}} \begin{bmatrix} \beta \\ \hat{\boldsymbol{z}} \end{bmatrix}^\top C \begin{bmatrix} \beta \\ \hat{\boldsymbol{z}} \end{bmatrix} \qquad \textit{subject to} \qquad \beta^2 + \|\hat{\boldsymbol{z}}\|^2 = 1, \qquad \hat{\boldsymbol{z}} \in \{P\tilde{\boldsymbol{z}} | \tilde{\boldsymbol{z}} \in \mathbb{R}^n\}.$$

Proof: We have

$$\begin{aligned} &\max_{\tilde{\boldsymbol{z}}} \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix}^\top C \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix} \\ =&\max_{\tilde{\boldsymbol{z}}} \begin{bmatrix} \beta \\ P^2\tilde{\boldsymbol{z}} \end{bmatrix}^\top C \begin{bmatrix} \beta \\ P^2\tilde{\boldsymbol{z}} \end{bmatrix} \\ =&\max_{\tilde{\boldsymbol{z}}} \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix}^\top \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix} C \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix} \\ \leq&\lambda^* \max_{\tilde{\boldsymbol{z}}} \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix}^\top \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix} = \lambda^*. \end{aligned} \tag{4.12}$$

Now, let $\begin{bmatrix} \gamma \\ \boldsymbol{g} \end{bmatrix}$ be the normalized eigenvector corresponding to the maximum eigenvalue λ^* of C_P. Then $\gamma^2 + \|\boldsymbol{g}\|^2 = 1$, and we have

$$\begin{aligned}
\lambda^* &= \begin{bmatrix} \gamma \\ \boldsymbol{g} \end{bmatrix}^\top \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix}^\top C \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} \gamma \\ \boldsymbol{g} \end{bmatrix} \\
&= \begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}^\top C \begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix} \\
&= (\gamma^2 + \|P\boldsymbol{g}\|^2) \frac{\begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}^\top}{\left\|\begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}\right\|} C \frac{\begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}}{\left\|\begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}\right\|} \\
&\leq \frac{\begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}^\top}{\left\|\begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}\right\|} C \frac{\begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}}{\left\|\begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}\right\|} \\
&= \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix}^\top C \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix} \qquad \text{by letting} \qquad \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix} := \frac{\begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}}{\left\|\begin{bmatrix} \gamma \\ P\boldsymbol{g} \end{bmatrix}\right\|}.
\end{aligned} \tag{4.13}$$

Therefore by (4.12) and (4.13) $\lambda^* = \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix}^\top C \begin{bmatrix} \beta \\ P\tilde{\boldsymbol{z}} \end{bmatrix}$. □

For the following algorithm we define $P = \begin{bmatrix} 1 & 0 \\ 0 & I - V_1^\top V_1 \end{bmatrix}$.

We will use the following procedure in the iterative step of our algorithm in order to do less operations.

$$C_V \begin{bmatrix} \beta \\ \tilde{\boldsymbol{z}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & I - V_1^\top V_1 \end{bmatrix}^\top \begin{bmatrix} \frac{\hat{\boldsymbol{w}}^\top A \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \frac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \frac{A^\top \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & I - V_1^\top V_1 \end{bmatrix} \begin{bmatrix} \beta \\ \tilde{\boldsymbol{z}} \end{bmatrix}$$
$$= \begin{bmatrix} \beta \frac{\hat{\boldsymbol{w}}^\top A \hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} + \frac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \tilde{\boldsymbol{z}} - \frac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} V_1 V_1^\top \tilde{\boldsymbol{z}} \\ \beta \frac{A\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} + A\tilde{\boldsymbol{z}} - AV_1V_1^\top \tilde{\boldsymbol{z}} - \beta V_1 V_1^\top \frac{A\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} - V_1V_1^\top A\tilde{\boldsymbol{z}} + V_1V_1^\top A V_1 V_1^\top \tilde{\boldsymbol{z}} \end{bmatrix}$$
$$= \begin{bmatrix} \frac{\hat{\boldsymbol{w}}^\top}{\|\hat{\boldsymbol{w}}\|} A \left(\beta \frac{\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} + \tilde{\boldsymbol{z}} - V_1 V_1^\top \tilde{\boldsymbol{z}} \right) \\ A \left(\beta \frac{\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} + \tilde{\boldsymbol{z}} - V_1 V_1^\top \tilde{\boldsymbol{z}} \right) - V_1 V_1^\top A \left(\beta \frac{\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} + \tilde{\boldsymbol{z}} - V_1 V_1^\top \tilde{\boldsymbol{z}} \right) \end{bmatrix}.$$

Notice that the vector $\boldsymbol{u} := A\left(\beta \frac{\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} + \tilde{\boldsymbol{z}} - V_1 V_1^\top \tilde{\boldsymbol{z}} \right)$ is multiplied in several expressions so we only need to compute it once and then find the rest of the entries.

Algorithm 4.2 Economy SVD.

Given an initial value α_0 and an initial vector $\tilde{\boldsymbol{z}}_0$, this procedure solves (1.6).

```
1: $[U, \Sigma, V_1] = \mathrm{svd}(B, 0)$
2: $\hat{\boldsymbol{w}} = V_1(\Sigma_1^{-1}(U^\top \boldsymbol{c}))$
3: $\boldsymbol{a} = A\hat{\boldsymbol{w}}$
4: $\boldsymbol{b} = \frac{\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|}$
5: for $k = 0$ to $k_{\max}$ do
6:     $\boldsymbol{u}_k = \alpha_k \boldsymbol{a} + A(\tilde{\boldsymbol{z}}_k - V_1(V_1^\top \tilde{\boldsymbol{z}}_k)))$
7:     $\alpha_{k+1} = \boldsymbol{b}^\top \boldsymbol{u}_k$
8:     $\tilde{\boldsymbol{z}}_{k+1} = \boldsymbol{u}_k - V_1(V_1^\top \boldsymbol{u}_k)$
9:     if $\begin{bmatrix} \alpha_{k+1}\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}}_{k+1} \end{bmatrix}$ converges then stop
10: end for
11: $m = \begin{bmatrix} \alpha_k\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}}_k \end{bmatrix}^\top \begin{bmatrix} \alpha_{k+1}\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}}_{k+1} \end{bmatrix}$
12: $\hat{\boldsymbol{z}} = \tilde{\boldsymbol{z}} - V_1(V_1^\top \tilde{\boldsymbol{z}})$
13: $\boldsymbol{v} = \alpha_{k+1}\hat{\boldsymbol{w}} + \hat{\boldsymbol{z}}$
14: return $m$ and $\boldsymbol{v}$
```

Notice that we do not compute the matrix P or the matrix C. This avoids the high computational cost of Algorithm 4.1.

4.5 QR Decomposition

In section 4.2 we used SVD on B in order to understand how much information the constraint provided. We use QR decomposition in this case to do the same thing and finally simplify the problem into an eigenvalue problem. Recall the constraint from our problem can be written as follows:

$$B\boldsymbol{v} = \alpha \boldsymbol{c} \qquad \text{where } \alpha \text{ varies in } \mathbb{R}.$$

Now we use QR decomposition on $B^\top$. Then

$$B^\top = QR,$$

$$B = R^\top Q^\top,$$

where $R^\top \in \mathbb{R}^{m\times n}$ is a lower triangular matrix and $Q^\top \in \mathbb{R}^{n\times n}$ is a orthogonal matrix. Partition $R^\top = \left[\begin{array}{c|c} R_1^\top & 0 \end{array}\right]$, where $R_1^\top \in \mathbb{R}^{m\times m}$ is lower triangular. Once we make the substitution we obtain

$$R^\top Q^\top \boldsymbol{v} = \alpha \boldsymbol{c}.$$

Let $\boldsymbol{x} = Q^\top \boldsymbol{v}$. Then $\boldsymbol{x} \in \mathbb{R}^n$ and $\|\boldsymbol{x}\| = 1$. Using this change of variable we obtain

$$R^\top \boldsymbol{x} = \alpha \boldsymbol{c}.$$

Let $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix}$ where $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{z} \in \mathbb{R}^{n-m}$. Then

$$R_1^\top \boldsymbol{y} = \alpha \boldsymbol{c},$$

$$\boldsymbol{y} = \alpha R_1^{-\top} \boldsymbol{c}.$$

Let $\boldsymbol{w} = R_1^{-\top} \boldsymbol{c}$. Notice that this is a known vector with m entries. Then

$$\boldsymbol{y} = \alpha \boldsymbol{w}. \tag{4.14}$$

Following the procedure in section 4.2 and section 4.3 we can now solve the initial problem using QR decomposition instead of SVD decomposition.

Algorithm 4.3 QR Method.

Given an initial vector $\boldsymbol{t}_1 = [t_1^{(1)}, t_2^{(1)}, ..., t_{n-m+1}^{(1)}]$, this procedure solves (1.6).

```
1: $[Q, R] = \mathrm{qr}(B^\top)$
2: $M = Q^\top A Q$
3: $\boldsymbol{w} = Q_1(R_1^{-\top}\boldsymbol{c})$
4: $C = \begin{bmatrix} \frac{\boldsymbol{w}^\top M_{11}\boldsymbol{w}}{\|\boldsymbol{w}\|^2} & \frac{\boldsymbol{w}^\top M_{12}}{\|\boldsymbol{w}\|} \\ \frac{M_{12}^\top\boldsymbol{w}}{\|\boldsymbol{w}\|} & M_{22} \end{bmatrix}$
5: for $k = 1$ to $k_{\mathbf{max}}$ do
6:     $\boldsymbol{t}_{k+1} = \frac{C\boldsymbol{t}_k}{\|C\boldsymbol{t}_k\|}$
7:     if $\boldsymbol{t}_{k+1}$ converges then stop
8: end for
9: $m = \boldsymbol{t}_{k+1}^\top C \boldsymbol{t}_{k+1}$
10: $\boldsymbol{y} = \frac{t_1^{(k+1)}}{\|\boldsymbol{w}\|}\boldsymbol{w}$
11: $\boldsymbol{z} = \left[t_2^{(k+1)}, t_3^{(k+1)}, ..., t_{n-m+1}^{(k+1)}\right]^\top$
12: $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix}$
13: $\boldsymbol{v} = V\boldsymbol{x}$
14: return $m$ and $\boldsymbol{v}_{k+1}$
```

4.6 Using Economy QR

Again, the shortcoming with the QR method is the high computational cost of line 2 in Algorithm 4.3. In order to avoid this problem we propose a partition of the matrix Q. Partition $Q = \left[\begin{array}{c|c} Q_1 & Q_2 \end{array}\right]$ where Q_1 is an $n \times m$ matrix and Q_2 is an $n \times (n-m)$ to obtain

$$\begin{aligned} B &= R^\top Q^\top \\ &= \left[\begin{array}{c|c} R_1^\top & 0 \end{array}\right] \left[\begin{array}{c} Q_1^\top \\ \hline Q_2^\top \end{array}\right] \\ &= R_1^\top Q_1^\top . \end{aligned}$$

These matrices (R_1 and Q_1) will represent our new decomposition called *economy QR*. Notice that only m columns of Q are computed which saves time but Q_2 **will be unavailable**. We will use an orthogonal projection in order to get around this obstacle.

Using the partition of Q we obtain

$$\boldsymbol{v} = Q\boldsymbol{x} = \left[\begin{array}{c|c} Q_1 & Q_2 \end{array}\right] \left[\begin{array}{c} \boldsymbol{y} \\ \boldsymbol{z} \end{array}\right] = Q_1\boldsymbol{y} + Q_2\boldsymbol{z} = \alpha Q_1\boldsymbol{w} + Q_2\boldsymbol{z}. \tag{4.15}$$

Now we define

$$\hat{\boldsymbol{w}} = Q_1\boldsymbol{w} \text{ and } \hat{\boldsymbol{z}} = Q_2\boldsymbol{z}. \tag{4.16}$$

Notice that $\hat{\boldsymbol{z}}$ is in the subspace generated by Q_2 (not available). Then

$$\boldsymbol{v} = \alpha Q_1\boldsymbol{w} + Q_2\boldsymbol{z} = \alpha\hat{\boldsymbol{w}} + \hat{\boldsymbol{z}}. \tag{4.17}$$

Since Q_2 is not available when using economy QR, we will use the following orthogonal projection onto the column space of Q_2

$$P_{Q_2} = Q_2Q_2^\top = I - Q_1Q_1^\top . \tag{4.18}$$

Then we have $\hat{\boldsymbol{z}} \in \{P_{Q_2}\tilde{\boldsymbol{z}} | \tilde{\boldsymbol{z}} \in \mathbb{R}^n\}$.

Use (4.17), make the substitution and use the fact that $P = P^\top = P^2$ in the original problem to obtain the following:

$$\begin{aligned}
\boldsymbol{v}^\top A\boldsymbol{v} &= (\alpha\hat{\boldsymbol{w}} + \hat{\boldsymbol{z}})^\top A(\alpha\hat{\boldsymbol{w}} + \hat{\boldsymbol{z}}) \\
&= \alpha^2\hat{\boldsymbol{w}}^\top A\hat{\boldsymbol{w}} + 2\alpha\hat{\boldsymbol{w}}^\top A\hat{\boldsymbol{z}} + \hat{\boldsymbol{z}}A\hat{\boldsymbol{z}} \\
&= \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ \hat{\boldsymbol{z}} \end{bmatrix}^\top \begin{bmatrix} \dfrac{\hat{\boldsymbol{w}}^\top A\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \dfrac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \dfrac{A^\top\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix} \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ \hat{\boldsymbol{z}} \end{bmatrix} \\
&= \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ P_{Q_2}\tilde{\boldsymbol{z}} \end{bmatrix}^\top \begin{bmatrix} \dfrac{\hat{\boldsymbol{w}}^\top A\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \dfrac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \dfrac{A^\top\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix} \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ P_{Q_2}\tilde{\boldsymbol{z}} \end{bmatrix} \\
&= \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}} \end{bmatrix}^\top \begin{bmatrix} 1 & 0 \\ 0 & P_{Q_2} \end{bmatrix}^\top \begin{bmatrix} \dfrac{\hat{\boldsymbol{w}}^\top A\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \dfrac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \dfrac{A^\top\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P_{Q_2} \end{bmatrix} \begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}} \end{bmatrix}.
\end{aligned} \tag{4.19}$$

Let

$$C_Q = \begin{bmatrix} 1 & 0 \\ 0 & P_{Q_2} \end{bmatrix}^\top \begin{bmatrix} \dfrac{\hat{\boldsymbol{w}}^\top A\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \dfrac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \dfrac{A^\top\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P_{Q_2} \end{bmatrix}.$$

Notice that the vector $\begin{bmatrix} \alpha\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}} \end{bmatrix}$ is not of length 1. We claim that the resulting maximum eigenvalue λ^* from the matrix C_Q is in fact the solution to the original problem.

To show this, we let $C = \begin{bmatrix} \dfrac{\hat{\boldsymbol{w}}^\top A\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|^2} & \dfrac{\hat{\boldsymbol{w}}^\top A}{\|\hat{\boldsymbol{w}}\|} \\ \dfrac{A^\top\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|} & A \end{bmatrix}$ and $\beta = \alpha\|\hat{\boldsymbol{w}}\|$ then we may write the original problem as follows:

$$\max \begin{bmatrix} \beta \\ \hat{\boldsymbol{z}} \end{bmatrix}^\top C \begin{bmatrix} \beta \\ \hat{\boldsymbol{z}} \end{bmatrix} \qquad \text{subject to} \qquad \beta^2 + \|\hat{\boldsymbol{z}}\|^2 = 1, \qquad \hat{\boldsymbol{z}} \in \{P_{Q_2}\tilde{\boldsymbol{z}} | \tilde{\boldsymbol{z}} \in \mathbb{R}^n\}.$$

Theorem 2 *Let λ^* be the maximum eigenvalue of C_Q then,*

$$\lambda^* = \max \begin{bmatrix} \beta \\ \hat{\boldsymbol{z}} \end{bmatrix}^\top C \begin{bmatrix} \beta \\ \hat{\boldsymbol{z}} \end{bmatrix}.$$

This can be proved in a similar way to that of Theorem 1.

Algorithm 4.4 Economy QR.

Given an initial value α_0 and an initial vector $\tilde{\boldsymbol{z}}_0$, this procedure solves (1.6).

```
1: $[Q, R] = qr(B^\top)$
2: $\hat{\boldsymbol{w}} = Q_1(R_1^{-\top}\boldsymbol{c})$
3: $\boldsymbol{a} = A\hat{\boldsymbol{w}}$
4: $\boldsymbol{b} = \frac{\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|}$
5: for $k = 0$ to $k_{\max}$ do
6:     $\boldsymbol{u}_k = \alpha_k\boldsymbol{a} + A(\tilde{\boldsymbol{z}}_k - Q_1(Q_1^\top\tilde{\boldsymbol{z}}_k)))$
7:     $\alpha_{k+1} = \boldsymbol{b}^\top\boldsymbol{u}_k$
8:     $\tilde{\boldsymbol{z}}_{k+1} = \boldsymbol{u}_k - Q_1(Q_1^\top\boldsymbol{u}_k)$
9:     if $\begin{bmatrix} \alpha_{k+1}\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}}_{k+1} \end{bmatrix}$ converges then stop
10: end for
11: $m = \begin{bmatrix} \alpha_k\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}}_k \end{bmatrix}^\top \begin{bmatrix} \alpha_{k+1}\|\hat{\boldsymbol{w}}\| \\ \tilde{\boldsymbol{z}}_{k+1} \end{bmatrix}$
12: $\hat{\boldsymbol{z}} = \tilde{\boldsymbol{z}} - V_1(V_1^\top\tilde{\boldsymbol{z}})$
13: $\boldsymbol{v} = \alpha_{k+1}\hat{\boldsymbol{w}} + \hat{\boldsymbol{z}}$
14: return $m$ and $\boldsymbol{v}$
```

Notice that we do not compute the matrix P or the matrix C.

CHAPTER 5

Modified Generalized Matrix Eigenvalue Problem

We now turn our attention to a more general form of the Normalized Cut problem by maximizing over a matrix W. This problem is stated in chapter 1, in (1.7). We restate the problem for convenience.

$$\text{Find} \quad \max_{W \in \mathbb{R}^{n \times k}} \mathbf{tr}(W^\top A W) \quad \text{subject to} \quad W^\top W = I_k, \quad \mathcal{R}(BW) \subseteq \mathcal{R}(C), \tag{5.1}$$

where $\mathcal{R}(X)$ is the subspace spanned by the columns of X. In the following sections we show the progress we have made as to understand this problem in hopes to find a solution.

5.1 Singular Value Decomposition

We start off by expressing the constraint in the following way: there exists $F \in \mathbb{R}^{k \times k}$ such that $BW = CF$. We use Singular Value Decomposition(SVD) on B in order to understand how much information the constraint provides. At the same time we will use our results from the SVD to simplify our modified eigenvalue problem. Let

$$B = U \Sigma V^\top, \tag{5.2}$$

where $U \in \mathbb{R}^{m \times m}$ is orthogonal, $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix with diagonal entries $\sigma_i \geq 0$ for $i = 1, 2, ..., m$ and $V \in \mathbb{R}^{n \times n}$ is orthogonal. Once we make the substitution we obtain

$$U \Sigma V^\top W = CF.$$

Let $X = V^\top W$. Then $X \in \mathbb{R}^{n\times k}$ and $X^\top X = I_k$. Using this change of variable we obtain

$$U\Sigma X = CF,$$

$$\Sigma X = U^\top CF.$$

Let $X = \left[\begin{array}{c} Y \\ Z \end{array}\right]$ where $Y \in \mathbb{R}^{m\times k}$ and $Z \in \mathbb{R}^{(n-m)\times k}$.

Also, partition $\Sigma = \left[\begin{array}{c|c} \Sigma_1 & 0 \end{array}\right]$ where $\Sigma_1 \in \mathbb{R}^{m\times m}$ is a diagonal matrix with diagonal entries σ_i. Then

$$\Sigma_1 Y = U^\top CF,$$

$$Y = \Sigma_1^{-1} U^\top CF.$$

Let $S = \Sigma_1^{-1} U^\top C$. Notice that this is a known matrix of size $m \times k$. Then

$$Y = SF. \tag{5.3}$$

5.2 Using SVD with partitioning

We now do a partitioning of V in order to solve the original problem. Let $V = \left[\begin{array}{c|c} V_1 & V_2 \end{array}\right]$. Then

$$W = VX = \left[\begin{array}{c|c} V_1 & V_2 \end{array}\right] \left[\begin{array}{c} Y \\ Z \end{array}\right] = V_1 Y + V_2 Z = V_1 SF + V_2 Z. \tag{5.4}$$

Now we define

$$\hat{S} := V_1 S \text{ and } \hat{Z} := V_2 Z. \tag{5.5}$$

We will use the following projection onto the space generated by V_2,

$$P_{V_2} = I - V_1 V_1^\top. \tag{5.6}$$

Then for some $\tilde{Z} \in \mathbb{R}^{n\times k}$ we have $\tilde{Z} = P_{V_2}\hat{Z}$.

We now make the substitution of (5.4) into the objective function.

$$\begin{aligned}
W^\top AW &= (V_1SF + V_2Z)^\top A(V_1SF + V_2Z) \\
&= (\hat{S}F + \hat{Z})^\top A(\hat{S}F + \hat{Z}) \\
&= F^\top \hat{S}^\top A\hat{S}F + F^\top \hat{S}^\top A\hat{Z} + \hat{Z}^\top A\hat{S}F + \hat{Z}^\top A\hat{Z} \\
&= \begin{bmatrix} F \\ \hat{Z} \end{bmatrix}^\top \begin{bmatrix} \hat{S}^\top A\hat{S} & \hat{S}^\top A \\ A\hat{S} & A \end{bmatrix} \begin{bmatrix} F \\ \hat{Z} \end{bmatrix} \qquad (5.7) \\
&= \begin{bmatrix} F \\ \tilde{Z} \end{bmatrix}^\top \begin{bmatrix} I_k & 0 \\ 0 & P_{V_2} \end{bmatrix} \begin{bmatrix} \hat{S}^\top A\hat{S} & \hat{S}^\top A \\ A\hat{S} & A \end{bmatrix} \begin{bmatrix} I_k & 0 \\ 0 & P_{V_2} \end{bmatrix} \begin{bmatrix} F \\ \tilde{Z} \end{bmatrix}.
\end{aligned}$$

We now rewrite the constraint in terms of F, $\hat{S}$ and $\hat{Z}$.

$$\begin{aligned}
W^\top W &= (V_1Y + V_2Z)^\top (V_1Y + V_2Z) \\
&= Y^\top V_1^\top V_1Y + Y^\top V_1^\top V_2Z + Z^\top V_2^\top V_1Y + Z^\top V_2^\top V_2Z \\
&= Y^\top V_1^\top V_1Y + Z^\top V_2^\top V_2Z \\
&= F^\top \hat{S}^\top \hat{S}F + \hat{Z}^\top \hat{Z} \\
&= \begin{bmatrix} F \\ \hat{Z} \end{bmatrix}^\top \begin{bmatrix} \hat{S}^\top \hat{S} & 0 \\ 0 & I_k \end{bmatrix} \begin{bmatrix} F \\ \tilde{Z} \end{bmatrix} \qquad (5.8) \\
&= \begin{bmatrix} F \\ \hat{Z} \end{bmatrix}^\top \begin{bmatrix} I_k & 0 \\ 0 & P_{V_2} \end{bmatrix} \begin{bmatrix} \hat{S}^\top \hat{S} & 0 \\ 0 & I_k \end{bmatrix} \begin{bmatrix} I_k & 0 \\ 0 & P_{V_2} \end{bmatrix} \begin{bmatrix} F \\ \tilde{Z} \end{bmatrix}.
\end{aligned}$$

Now we define the matrices

$$T := \begin{bmatrix} F \\ \tilde{Z} \end{bmatrix},$$

$$G := \begin{bmatrix} I_k & 0 \\ 0 & P_{V_2} \end{bmatrix} \begin{bmatrix} \hat{S}^\top A \hat{S} & \hat{S}^\top A \\ A\hat{S} & A \end{bmatrix} \begin{bmatrix} I_k & 0 \\ 0 & P_{V_2} \end{bmatrix},$$

$$H := \begin{bmatrix} I_k & 0 \\ 0 & P_{V_2} \end{bmatrix} \begin{bmatrix} \hat{S}^\top \hat{S} & 0 \\ 0 & I_k \end{bmatrix} \begin{bmatrix} I_k & 0 \\ 0 & P_{V_2} \end{bmatrix}.$$

The dimensions of these new matrices are $T \in \mathbb{R}^{(n+k)\times k}$, $G \in \mathbb{R}^{(n+k)\times(n+k)}$ and $H \in \mathbb{R}^{(n+k)\times(n+k)}$. Then our problem can be rewritten as:

$$\max_T \mathbf{tr}(T^\top G T) \qquad \text{subject to} \qquad T^\top H T = I_k \tag{5.9}$$

which can now be solved by using the method of Lagrange Multipliers. The Lagrange function will be the following

$$L(T,\Gamma) = \mathbf{tr}\, T^\top G T - \mathbf{tr}\, \Gamma^\top (T^\top H T - I_k).$$

Then we take the derivative with respect to T by adding a perturbation.

$$\begin{aligned} L(T+\Delta T,\Gamma) &= \mathbf{tr}[(T+\Delta T)G(T+\Delta T)] - \mathbf{tr}\,\Gamma^\top[(T+\Delta T)^\top H(T+\Delta T) - I_k] \\ &= \mathbf{tr}(T^\top G T + 2T^\top G\Delta T + \Delta T^\top G \Delta T) \\ &\qquad - \mathbf{tr}\,\Gamma^\top[(T^\top H T + 2T^\top H \Delta T + \Delta T^\top H \Delta T - I_k)]. \end{aligned} \tag{5.10}$$

In order to find the derivative we only consider the terms that are linear with respect to ΔT. The derivative will be given by

$$D_T L(T,\Gamma)(\Delta T) = \mathbf{tr}(2T^\top G \Delta T) - \mathbf{tr}\,\Gamma^\top(2T^\top H \Delta T).$$

We now set this equal to zero and group terms to obtain

$$2\,\mathbf{tr}[(T^\top G - \Gamma^\top T^\top H)\Delta T] = 0.$$

This holds for any ΔT. Therefore

$$T^\top G - \Gamma^\top T^\top H = 0$$

which implies that

$$GT = HT\Gamma. \tag{5.11}$$

Finding the values of Γ now only requires us to find the k largest eigenvalues of the following generalized eigenvalue problem,

$$G\boldsymbol{t} = \lambda H\boldsymbol{t}. \tag{5.12}$$

Several numerical solutions to (5.12) can be found in [1].

CHAPTER 6

Numerical Results

In this chapter we show the numerical results by the Krylov Subspace Method shown in Chapter 3 and the four algorithms in Chapter 4.

6.1 Krylov Subspace Method

For this section we used two examples in order to test the various algorithms in chapter 3. We selected two pictures in JPEG format. Our examples are given in Fig. 6.1 and Fig. 6.5. Our objective is to use image segmentation in order to obtain boundaries that split the image into two different parts. The first image is a bird and we want to extract the whole bird out of the background. The second image is a camel and we would like to extract the camel, and discard the sky and the sand. Our methods will return a matrix with labels 0 or 1 that correspond to each pixel in the picture. These labels are used to determine the graph that the pixel belongs to. We use the grouping algorithm in [14] but we maximize the constrained eigenvalue problem (1.4). We also show the points selected for the constrained examples (Fig. 6.2 and Fig. 6.6) and how the segmentation differs from regular normalized cut (Fig. 6.3 and Fig. 6.7) and constrained normalized cut (Fig. 6.4 and Fig. 6.8). The normalized cut segmentation code and some other examples can be found in the web site http://www.cis.upenn.edu/∼jshi/software/.

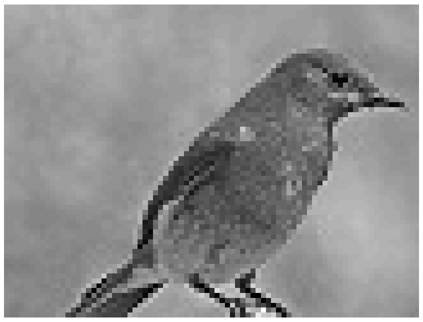
Figure 6.1. A picture of a bird.

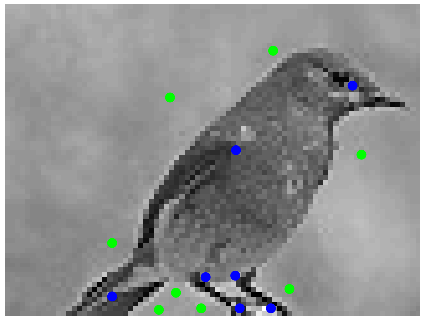
Figure 6.2. Points selected for the constraint of the bird.

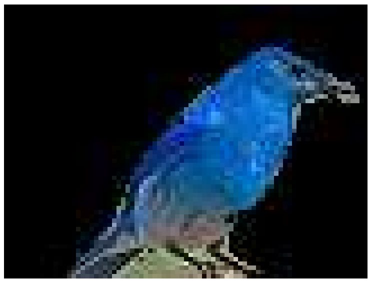
Figure 6.3. Normalized cut segmentation of the bird.

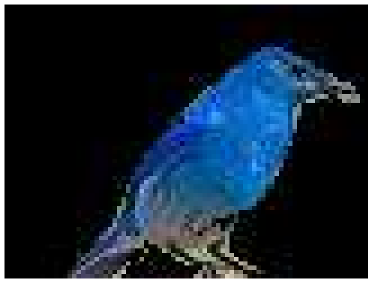
Figure 6.4. Constrained segmentation of the bird.

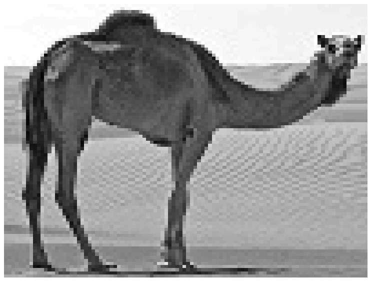
Figure 6.5. A picture of a camel.

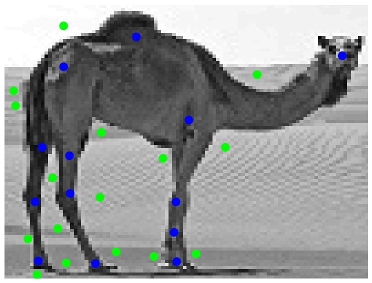
Figure 6.6. Points selected for the constraint of the camel.

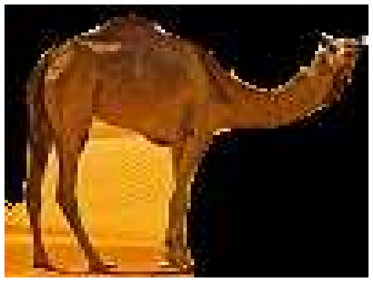
Figure 6.7. Normalized cut segmentation of the camel.

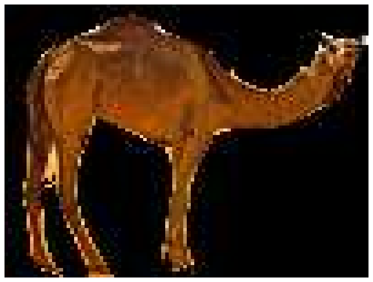
Figure 6.8. Constrained segmentation of the camel.

An affinity matrix W is generated from these pictures. The entries of the matrix W are labeled w_{ij} and generated with the following formula [14],

$$w_{ij} = \begin{cases} \left(e^{\frac{-\|\mathbf{F}_{(i)} - \mathbf{F}_{(j)}\|_2^2}{\sigma_I^2}} \right) \left(e^{\frac{-\|\mathbf{X}_{(i)} - \mathbf{X}_{(j)}\|_2^2}{\sigma_X^2}} \right) & \text{if } \|\mathbf{X}_{(i)} - \mathbf{X}_{(j)}\|_2 < r \\ 0 & \text{otherwise} \end{cases}, \quad (6.1)$$

where $\mathbf{F}$ is the brightness value of each pixel, $\mathbf{X}$ is the spacial location of the pixel, σ_I is a parameter that controls the difference in brightness and σ_X is a parameter that controls the difference in spacial location. Equation (6.1) should reflect the likelihood that two pixels (nodes i and j) belong to the same point. If the pixels are too far away (farther than a given radius r) then the value of the edge weight w_{ij} is set to 0. In the following figures we use the Lanczos process to obtain a matrix T_k of size k. Then we iterate using the zero finding algorithm as in Algorithm 3.2. We compare the error at each step for increasing values of k.

To study the performance of our algorithm first we calculated the value of the exact solution λ_{exact} by using the Newton method for the secular equation in each example. Then we use the Krylov subspace method and compare our solution λ^* to the exact solution,

$$\text{error} = \frac{|\lambda^* - \lambda_{\text{exact}}|}{\lambda_{\text{exact}}} \tag{6.2}$$

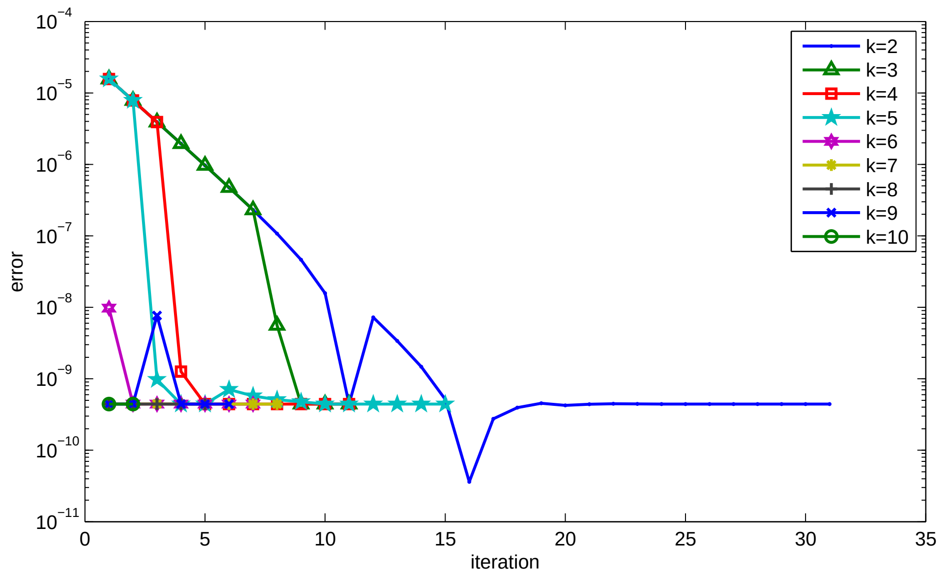

Figure 6.9. Iteration vs. Error for the bird.

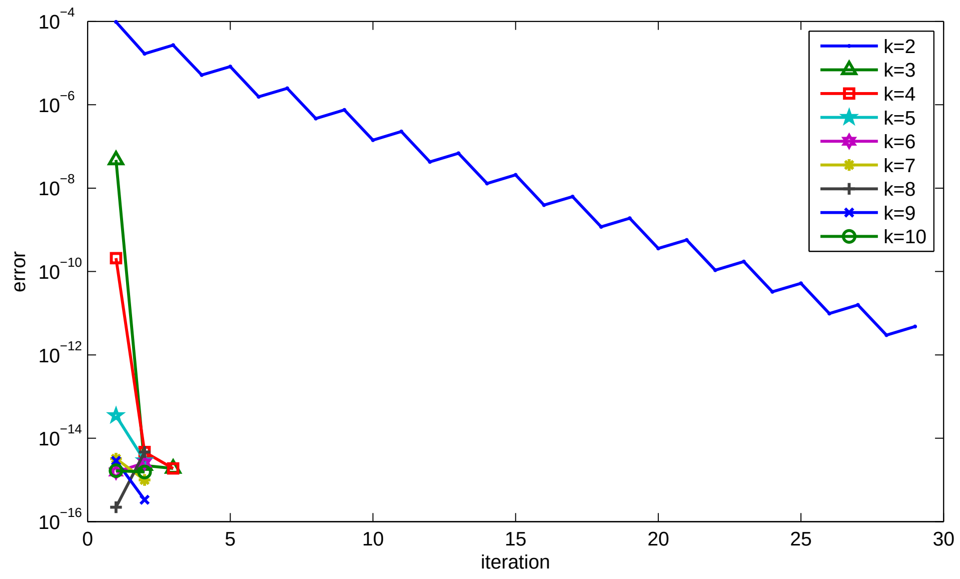

Figure 6.10. Iteration vs. Error for the camel.

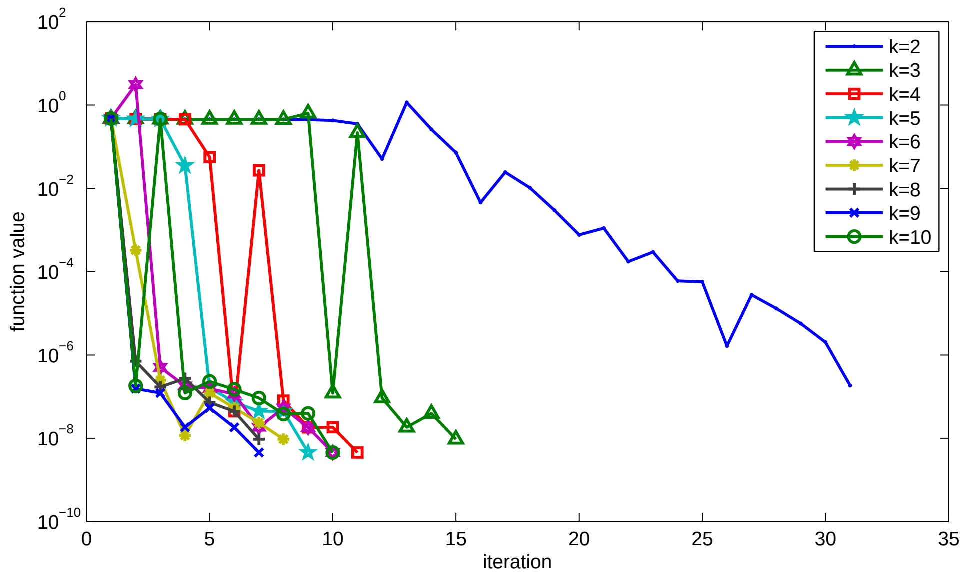

Figure 6.11. Iteration vs. Function Value for the bird.

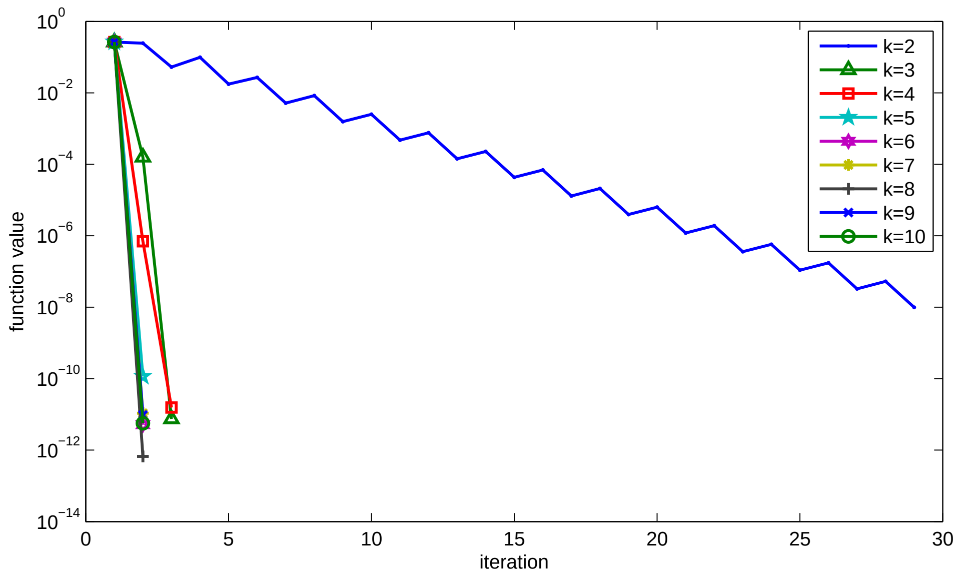

Figure 6.12. Iteration vs. Function Value for the camel.

6.2 SVD Method

In this section we compare the performance of the four algorithms presented in chapter 4. The experiment is performed on a random positive definite matrix $A \in \mathbb{R}^{n\times n}$ of the form $A = S^\top S$ for some $S \in \mathbb{R}^{n\times n}$. We also generate a random matrix $B \in \mathbb{R}^{m\times n}$ and a random $\boldsymbol{c} \in \mathbb{R}^m$. Our convergence criterion in every experiment is the condition $\|\boldsymbol{v} - \boldsymbol{v}_{prev}\| < 10^{-8}$ in each power iteration.

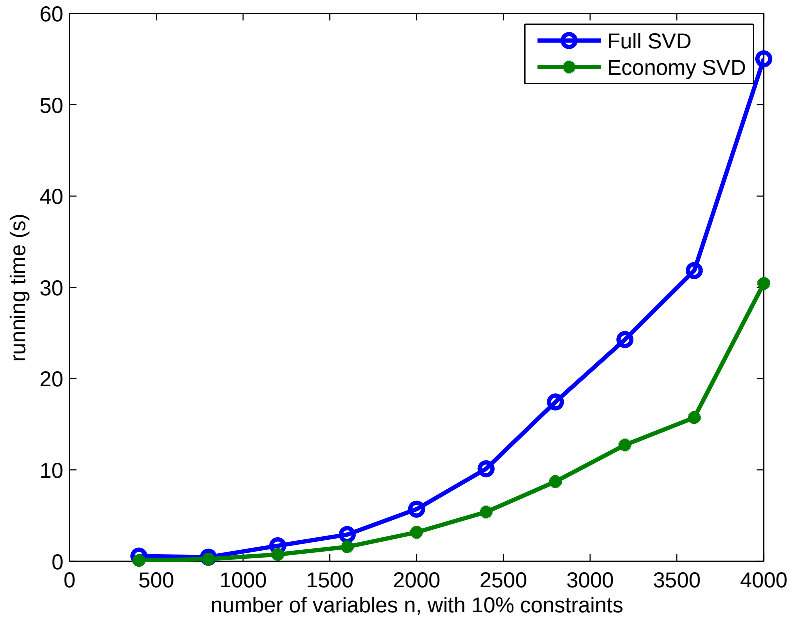

Figure 6.13. SVD Methods - Running time vs. Number of Variables.

6.3 QR Method

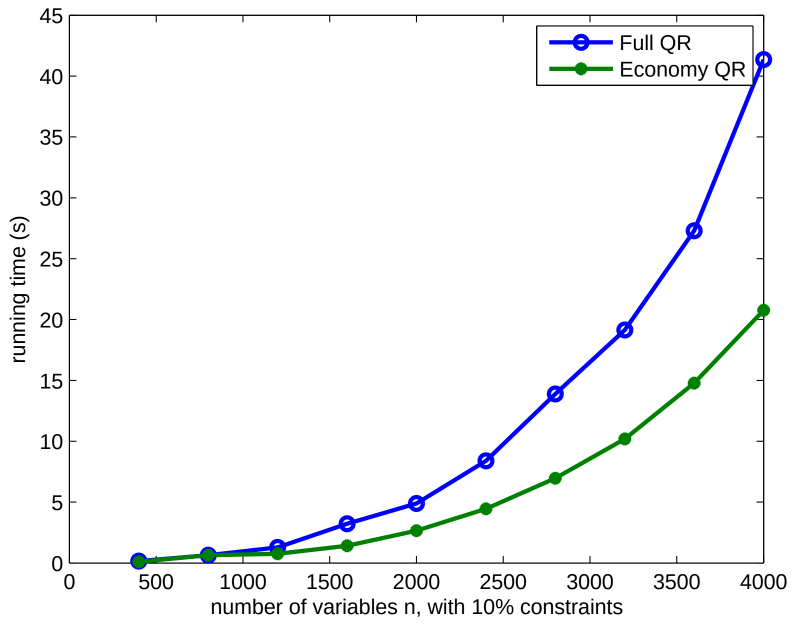

Figure 6.14. QR Methods - Running time vs. Number of Variables.

REFERENCES

[1] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide.* Society for Industrial and Applied Mathematics, 3600 Market Street, 6th Floor, Philadelphia, PA 19104-2688 USA, 2000.

[2] W. Cheney and D. Kincaid. *Numerical Mathematics and Computing, Sixth Edition.* Thomson Higher Education, 10 Davis Drive, Belmont, CA, 94002-3098 USA, 2004.

[3] J. W. Demmel. *Applied Numerical Linear Algebra.* Society for Industrial and Applied Mathematics, 3600 Market Street, 6th Floor, Philadelphia, PA 19104-2688 USA, 1997.

[4] W. Gander, G. H. Golub, and U. Matt. A constrained eigenvalue problem. *Linear Algebra and its Applications*, pages 815–839, 1989.

[5] G. H. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15(2):318–334, 1973.

[6] G. H. Golub and H. A. van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1-2):35–65, 2000.

[7] M. T. Jones and M. L. Patrick. The use of Lanczos method to solve the large generalized symmetric definite eigenvalue problem. *NASA Contractor Report, ICASE Report*, 89-69:8–10, 1989.

[8] D. Kincaid and W. Cheney. *Numerical Analysis: Mathematics of Scientific Computing, Third Edition.* Brooks/Cole, 511 Forest Lodge Road, Pacific Grove, CA 93950 USA, 2002.

[9] R.-C. Li and Q.Ye. Simultaneous similarity reductions for a pair of matrices to condensed forms. *Communications in Mathematics and Statistics*, 2:139–153, 2014.

[10] H. Müntz. Solution directe de l'équation séculaire et de quelques problèmes analogues transcendants. *Comptes Rendus de l'Académie des Sciences*, 156:43–46, 1913.

[11] H. Müntz. Sur la solution des équations séculaires et des équations intégrales. *Comptes Rendus de l'Académie des Sciences*, 156:860–862, 1913.

[12] A. Pothen, H.D. Simon, and K. P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Analytical Applications*, 11:430–452, 1990.

[13] Y. Saad. *Numerical Methods for Large Eigenvalue Problems Second Edition.* Society for Industrial and Applied Mathematics, 3600 Market Street, 6th Floor, Philadelphia, PA 19104-2688 USA, 2011.

[14] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[15] R. von Mises and H. Pollaczek-Geiringer. Praktische verfahren der gleichungsauflosung. *ZAMM - Zeitschrift fur Angewandte Mathematik und Mechanik*, 9:152–164, 1929.

[16] D. S. Watkins. *Fundamentals of Matrix Computations, Second Edition.* John Wiley and Sons, Inc., New York, 2002.

[17] L. Xu, W. Li, and D. Schuurmans. Fast normalized cut with linear constraints. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2866–2873, 2009.

[18] L.-H. Zhang, W. H. Yang, C. Shen, and R.-C. Li. A Krylov subspace method for large scale second order cone linear complementarity problem. *SIAM Journal on Scientific Computing*, 37(4):A2046–A2075, 2015.

BIOGRAPHICAL STATEMENT

Iván Ojeda-Ruiz was born in Bayamón, Puerto Rico, in 1987. He received his B.S. degree from the University of Puerto Rico, Rio Piedras Campus, in 2010, his Ph.D. degree from The University of Texas at Arlington in 2017, in Numerical Analysis.