Nile Valley University

College of Post Graduate

THE QR –FACTORIZATION FOR COMPUTING THE LAGRANGIAN MULTIPLIERS

AThesis Submitted in partial Fulfillment For The

Degree of M.SC in Mathematics

Prepared by : Almoiz Alsheikh Hamid Ibrahim

Supervised : Dr. Ahmed Abelfadel Mohamed

August 2011

Dedication

To My Family , Wife and Children

Acknowledgement

Iwould like to express my deep of gratitude and thanks to my teacher and supervisor Dr. Ahmed Abelfadel for useful comments suggestions and constructive my gratitude is also extended particularly to Nile Valley University for the great help and assistance

Abstract

In this research we discussed the Direct Methods for the Solution of Linear systems and also we discussed the definition of QR algorithm and the methods for computing the QR factorization by MATLAB , and then we use the QR factor to solve optimality condition for linear equality constraints to find the lagrangian multipliers, finally we use the lagrangian multipliers to solve nonlinear optimization problems where all constraints are linear equalities and then we comparing between the QR- method and traditional methods by MATLAB , and we find the QR Method is better to solve all nonlinear optimization problem where all constraints are linear equalities.

ملخص البحث

في هذا البحث ناقشنا الطرق المباشرة لحل النظام الخطي و أيضاً ناقشنا تعريف خوارزمية QR factor وطرق استعمال حاسبات QR factor من قبل Matlab وبعد ذلك استعملنا QR factor لحل مسائل الأمثلية لقيود المساواة الخطية لإيجاد مضاريب لاجرانج و أخيراً استعملنا مضاريب لاجرانج لحل مسائل تحقيق الأمثلية اللاخطية بشروط أمثلية بقيود خطية ، و بعد ذلك قارنا بين طريقة QR و طرق تقليدية من قبل Matlab وجدنا طريقة QR أفضل لحل كل مسائل تحقيق الأمثلية اللاخطية حيث أن كل قيود المساواة خطية .

Contents

subject	Page
Dedication	I
Acknowledgement	II
Abstract	III
Abstract (Arabic)	IV
contents	V
Introduction	VII
Back Ground History	VIII
Research problem	IX
Chapter One **MATHEMATICL BACK GROUND**	
1-1 Matrices	1
1-2 Operations with Matrices	3
1-2-1 Inverse of Matrix	4
1-2-2 Matrices and Linear Mapping	6
1-2-3 Operations with Block- Partitioned Matrices	6
1-3 Trace and Determinant of a Matrix	7
1-4 Eigenvalues and Eigenvector	9
1-5 Similarity Transformation	11
1-6 The singular value decomposition	14
1-7 Matrix Norms	15
1.7.1 Relation between Norms and the Spectral Radius of a matrix	20
1.7.2 Sequences and Series of Matrices	21
Chapter Two **Direct Method for the Solution of Linear system**	
2-0 Introduction	23
2-1 Stability Analysis of Linear Systems	24
2-1-1 The Condition Number of a Matrix	25
2-1-2 Forward a priori Analysis	26
2-1-3 Backward a priori Analysis	30
2-2 The Gaussian Elimination Methods(GEM) and LU factorization	31
2.2.1 GEM as Factorization Method	35
2-3 The LDM^T factorization	40
2-4 The Cholsky factorization	41
2-5 Rectangular Matrices : QR factorization	44
2-6 Pivoting	49
2-7 Computing the Inverse of a Matrix	54
2-8 undetermined systems	55
Chapter Three **Approximation of Eigenvalues and Eigenvectors**	
3-1 Introduction	59
3-2 Geometrical Location of the Eigenvalues	59
3-3 Power Method	62

3-3-1 Approximation of the Eigenvalue of Largest Module	**62**
3.3.2 Inverse Iteration	**66**
3.3.3 Implementation Issues	**68**
3-4 The QR Iteration	**71**
3-5 The Basic QR Iteration	**73**
3-6 The QR Method for matrices in Hessenberg Form	**75**
3-6-1 Householder and Given Transformation Matrices	**76**
3-6-2 Reducing a Matrix in Hessenberg Form	**80**
3-6-3 QR factorization of a Matrix in Hessenberg	**83**
3-6-4TheBasic QR Iteration Starting from Upper Hessenberg Form	**84**
3-6-5 Implementation of transformation Matrices	**87**
3-7 The QR Iteration with Shifting Techniques	**91**
3-7-1 The QR Iteration Method with single shift	**91**
3-7-2 The QR Iteration Method with Double shift	**95**
3-8 Computing The Eigenvector from the Schur Form of a Matrix	**98**
3-8-1 The Hessenberg Inverse Iteration	**98**
3-8-2 Computing the Eigenvectors from the Schur Form of a matrix	**99**
3-8-3 Approximate Computation of the SVD of a matrix	**100**
3-9 The Generalized Eigenvalue Problem	**102**
3-9-1 Computing the Generalized Real Schur Form	**103**
3-9-2 Generalized Real Schur Form of Symmetric –Definite Pencils	**105**
Chapter Four Applications	
4-0 Basic Concepts	**107**
4-0-1 linear constraints	**107**
4-0-2 Null and Range Spaces	**115**
4-0-3 Generating Null-Space Matrices	**118**
4.0.3.1 Variable Reduction Method	**118**
4.0.3.2 Orthogonal Projection Matrix	**123**
4.0.3.3 Other Projections	**125**
4.0.3.4 The *QR* Factorization	**126**
4-0-4 The Chain Rule	**127**
4-1 Introduction	**130**
4.2 Optimality Conditions for Linear Equality Constraints	**131**
4-3 The Lagrange Multipliers and the Lagrange Function	**140**
4.4 Computing the Lagrange Multipliers	**143**
4-5 Comparison and conclusion	**153**
References	**156**

Introduction

Since the early 1960s the standard algorithms for calculating the eigenvalues and (optionally) eigenvectors of "small" matrices have been the QR algorithm [1] and its variants. This is still the case in the year 2000 and is likely to remain so for many years to come. For us a small matrix is one that can be stored in the conventional way in a computer's main memory and whose complete eigenstructure can be calculated in a matter of minutes without exploiting whatever sparsity the matrix may have had. If a matrix is small, we may operate on its entries. In particular, we are willing to perform similarity transformations, which will normally obliterate any sparseness the matrix had to begin with.

If a matrix is not small, we call it large. The boundary between small and large matrices is admittedly vague, but there is no question that it has been moving steadily upward since the dawn of the computer era. In the year 2000 the boundary is around n = 1000, or perhaps a bit higher.

Eigenvalue problems come in numerous guises. Whatever the form of the problem, the QR algorithm is likely to be useful. For example, for generalized eigenvalue problems $Ax = \lambda Bx$, the method of choice is a variant of the QR algorithm called QZ. Another variant of QR is used to calculate singular value decompositions (SVD) of matrices. The QR algorithm is also important for solving large eigenvalue problems. Most algorithms for computing eigenvalues of large matrices repeatedly generate small auxiliary matrices whose eigensystems need to be computed as a subtask. The most popular algorithms for this subtask are the QR algorithm and its variants.

Back Ground History

In this research we discuss the QR algorithm. The subject was born in the early 1950s with Rutishauser's quotient-difference algorithm [2] which he formulated as a method for calculating the poles of a meromorphic function. He then reformulated it in terms of matrix operations and generalized it to the LR algorithm [3] . The QR algorithm was published by Kublanovskaya [4] and Francis [1] in 1961. The Francis paper is particularly noteworthy for the refinements it includes. The double-shift implicit QR algorithm laid out there is only a few details removed from codes that are in widespread use today.

And what codes are in use today? By far the most popular tool for matrix computations is Matlab. If we use Matlab to compute your eigenvalues, we will use one of its four QR-based computational kernels.

Each of these is just a few refinements removed from codes in the public-domain software packages EISPACK [5] and LINPACK [6]. In particular, the algorithm for computing eigenvalues of real, non symmetric matrices is just the Francis double-shift QR algorithm with some modifications in the shift strategy.

A newer public-domain collection is LAPACK [7], which was designed to perform well on vector computers, high-performance work stations, and shared-memory parallel computers. It also has a double-shift implicit QR code, which is used on matrices (or portions of matrices) under 50×50. For larger matrices a multishift QR code is used.

For many years the QR algorithm resisted efforts to parallelize it. The prospects for a massively parallel QR algorithm for distributed memory parallel computers were considered dim.

The pessimism was partly dispelled by van de Geijn and Hudson [8], who demonstrated the first successful highly parallel QR code.

However, their code relies on an unorthodox distribution of the matrix over the processors, which makes it hard to use in conjunction with other codes. Subsequently, Henry [9] wrote a successful parallel QR code that uses a standard data distribution. This is an implicit double-shift code that performs the iterations in pipeline fashion. This code is available in ScaLAPACK [10], a collection of matrix computation programs for distributed-memory parallel computers .

On the theoretical side, the first proof of convergence of the LR algorithm (without pivoting or shifts of origin) was provided by Rutishauser [2]. His proof was heavily laden with determinants, in the style of the time. Wilkinson [11] proved convergence of the unshifted QR algorithm using matrices, not determinants. Wilkinson [12] also proved global convergence of a shifted QR algorithm on symmetric, tridiagonal matrices.

Research problem :

A class of common optimization problems subject to equality constraints may be nicely handled by the Lagrange multiplier method. Consider an optimization problem with M equality constraints.

$$Min\, f(x)$$

$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_M(x) \end{bmatrix}$$

According to the Lagrange multiplier method, this problem can be converted to the following unconstrained optimization problem:

$$Min\, l(x,\lambda) = f(x) + \lambda^T h(x) = f(x) + \sum_{m=1}^{M} \lambda_m h_m(x)$$

The solution of this problem, if it exists, can be obtained by setting the derivatives of this new objective function l(x, λ) with respect to x and λ

$$\frac{\partial}{\partial \mathrm{x}}\, l(x,\lambda) = \frac{\partial}{\partial \mathrm{x}} f(x) + \lambda^T \frac{\partial}{\partial \mathrm{x}} h(x) = \nabla f(x) + \sum_{m=1}^{M} \lambda_m \nabla h_m\,(x) = 0$$

$$l(x,\lambda) = h(x) = 0$$

Note that the solutions for this system of equations are the extreme of the objective function. We may know if they are minima/maxima, from the positive/negative- definiteness of the second derivative (Hessian matrix) of $l(x, \lambda)$ with respect to x.

Imposing the QR method as the solving technique is the research problem.

Chapter One

MATHEMATICL BACK GROUND

1-1 Matrices:

Let m and n be two positive integers. We call a matrix having m rows and n columns, or a matrix $m \times n$, or a matrix $(m \times n)$, with element K , a set of mn scalars $a_{ij} \in K$, with $i = 1, \dots, m$ and $j = 1, \dots, n$, represented in the following rectangular array

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \qquad (1.1)$$

When $K = \mathbb{R}$ or $K = \mathbb{C}$ we shall respectively write $A \in \mathbb{R}^{m\times n}$ or $A \in \mathbb{C}^{m\times n}$, to explicitly outline the numerical fields which the elements of A belong to. Capital letters will be used to denote the matrices, while the lower case letters corresponding to those upper case letters will denote the matrix entries.

We shall abbreviate (1.1) as $A = (a_{ij})$ which $i = 1, \dots, m$ and $j = 1, \dots, n$. The index i is called row index, while j is column index. The set $(a_{i1}, a_{i2}, \dots., a_{in})$ is called the $i - th$ row of A , likewise, $(a_{1j}, a_{2j}, \dots., a_{mj})$ is the $j - th$ column of A.

If $n = m$ the matrix is called squared or having order n and the set of the entries $(a_{11}, a_{22}, \dots., a_{nn})$ is called its main diagonal.

A matrix have one row or one column is called row vector or column vector respectively. Unless otherwise specified, we shall always assume that a vector is a column vector. In the case $n = m = 1$, the matrix will simply denote a scalar of K.

Sometimes it turns out to be useful to distinguish within a matrix the set made up by specified rows and column. This prompts us to introduce the following definition.

Definition 1.1 Let A be a matrix $m \times n$. Let $1 \leq i_1 < i_2 < \cdots < i_k \leq m$ and $1 \leq j_1 < j_2 < \cdots < j_k \leq n$ two sets of contiguous indexes. The matrix $S(k \times l)$ of entries $s_{pq} = a_{i_p j_q}$ with $p = 1, \dots, k$ and $q = 1, \dots, l$ is called a submatrix of A . If $k = l$ and $i_r = j_r$ for $r = 1, \dots, k$, s is called a principal submatrix of A.

Definition 1.2 A matrix $A(m \times n)$ is called block partitioned or said to be partitioned into submatrices if

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1l} \\ A_{21} & A_{22} & \cdots & A_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k1} & A_{k2} & \cdots & A_{kl} \end{bmatrix},$$

where A_{ij} are submatrices of A.

Among the possible partitions of A, we recall in particular the partition by columns

$A = (a_1, a_2, \dots, a_n)$,

a_i being the $i - th$ column vector of A . In a similar way the partition by rows of A can be defined. To fix the notations , if A is a matrix $m \times n$, we shall denote by

$$A(i_1{:}i_2, j_1{:}j_2) = \left(a_{ij}\right) \quad i_1 \leq i \leq i_2, \qquad j_1 \leq j \leq j_2$$

The submatrix of A of size $(i_2 - i_1 + 1) \times (j_2 - j_1 + 1)$ that lies between the rows i_1 and i_2 and the columns j_1 and j_2. Likewise, if v is a vector of size n, we shall denote by $v(i_1{:}i_2)$ the vector of size $i_2 - i_1 + 1$ made up by the $i - th$ notations are convenient in view of programming the algorithms that will be presented throughout the volume in the MATLAB language.

1-2 Operations with Matrices:

Let $A = (a_{ij})$ and $B = (b_{ij})$ be two matrices $m \times n$ over K. We say that A is equal to B if $a_{ij} = b_{ij}$ for $i = 1, \dots, m,\ j = 1, \dots, n$. Moreover, we define the following operations:

- matrix sum: the matrix sum is matrix $A + B = \left(a_{ij} + b_{ij}\right)$. The neutral element in matrix sum is the null matrix, still denote by 0 and made up only null entries;
- matrix multiplication by a scalar : the multiplication of A by $\lambda \in K$, is a matrix $\lambda A = (\lambda a_{ij})$;
- matrix product: the product of two matrices A and B of size (m,p) and (p,n) respectively, is a matrix $C(m,n)$ whose entries are $c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj},\ for\ i = 1, \dots, m\ , j = 1, \dots, n$.

The matrix product is associative and distributive with respect to the matrix sum, but is not in general commutative. The square matrices for which the property A$B = BA$ holds, will be called commutative .

In the case of square matrices , the neutral element in the matrix product is a square matrix of order n called the unit matrix of order n or, more frequently, the identity matrix given by $I_n = \left(\delta_{ij}\right)$. The identity matrix is, by definition, the only matrix $n \times n$ such that $AI_n = I_n A = A$ for all square matrices A. In the following we shall omit the subscript n unless it is strictly necessary. The identity matrix is special instance of diagonal matrix of order n , that is a square matrix of the type $D = \left(d_{ii}\delta_{ij}\right)$. We will use in the following the notation $D = diag(d_{11}, d_{22}, \dots, d_{nn})$.

Finally, if A is square matrix of order n and p is an integer, we define A^p as the product of A with itself iterated p times. We let $A^0 = I$.

Let us now address the so-called elementary row operations that can be performed on a matrix. They consist of :

- multiplying the $i-th$ row of a matrix by a scalar α; this operation is equivalent to pre-multiplying A by the matrix $D = diag(1, \dots, 1, \alpha, 1, \dots, 1)$, where α occupies the $i-th$ position ;
- exchanging the $i-th$ and $j-th$ rows of a matrix , this can be denote by premultiplying A by the matrix $P^{(i,j)}$ of elements

$$p_{rs}^{(i,j)} = \begin{cases} 1 & if \ \ r = s = 1, \dots, i-1, i+1, \dots, j-1, j+1, \dots, n \\ 1 & if \ r = j, s = i \ or \ r = i, \ \ s = j \\ 0 & otherwise \end{cases} \qquad (1.2)$$

where I_r denotes the identity matrix of order $r = j - i - 1 \ \ if \ j > i$ (henceforth, matrices with size equal to zero will correspond to the empty set). Matrices like (1.2) are called elementary permutation matrices. The product of elementary permutation matrices is called a permutation matrix, and it performs the row exchanges associated with each elementary permutation matrix. In practice, a permutation matrix is a reordering by rows of the identity matrix;

- adding α times the $j-th$ row of a matrix to its $i-th$ row. This operation can also be performed by pre-multiplying A by the matrix $I + N_\alpha^{(i,j)}$, where $N_\alpha^{(i,j)}$ is a matrix having null entries except the one in position i, j whose value is α.

1-2-1 Inverse of Matrix:

Definition 1.3 A square matrix A of order n is called invertible(or regular or nonsingular)if there exists a square matrix B of order n such that $AB = BA = I$. B is called the inverse matrix of A and is denoted by A^{-1}. A matrix which is not invertible is called singular.

If A is invertible its inverse is also invertible, with $(A^{-1})^{-1} = A$. Moreover, if A and B are two invertible matrices of order n, their product AB is also invertible, with $(AB)^{-1} = A^{-1}B^{-1}$. The following property holds

Property 1.1 A square matrix is invertible iff its column vector are linearly independent.

Definition 1.4 We call the transpose of a matrix $A \in \mathbb{R}^{m\times n}$ the matrix $n \times m$, denoted by A^T , that is obtained by exchanging the rows of A with the column of A .

Clearly, $(A^T)^T = A,\ (A+B)^T = A^T + B^T,\ (AB)^T = B^T A^T$ and $(\alpha A)^T = \alpha A^T\ \forall \alpha \in \mathbb{R}$. If A is invertible, then also $(A^T)^{-1} = (A^{-1})^T = A^{-T}$.

Definition 1.5 Let $A \in \mathbb{C}^{m\times n}$; the matrix $B = A^H \in \mathbb{C}^{n\times m}$ is called the conjugate transpose (or adjoint) of A IF $b_{ij} = \bar{a}_{ji}$, where $\bar{a}_{ji}$ is the complex conjugate of a_{ij} .

In analogy with the case of the real matrices, it turns out that

$(A+B)^H = A^H + B^H,\ (AB)^H = B^H A^H$ and $(\alpha A)^H = \bar{\alpha} A^H\ \forall \alpha \in \mathbb{C}$.

Definition 1.6 A matrix $A \in \mathbb{R}^{m\times n}$ is called symmetric if $A = A^T$, while it is antisymmetric if $A = -A^T$. Finally, it is called orthogonal if $A^T A = AA^T = I$, that is $A^{-1} = A^T$.

Permutation matrices are orthogonal and the same is true for their products.

Definition 1.7 A matrix $A \in \mathbb{C}^{m\times n}$ is called hermition or self- adjoint if $A^T = \bar{A}$, that is, if $A^H = A$, while it is called unitary if $AA^H = A^H A$, A is called normal .

As a consequence, a unitary matrix is one such that $A^{-1} = A^H$.

Of course, a unitary matrix is also normal, but it is not in general hermitian. For instance, the matrix of the Example 1.1 is unitary, although not symmetric (if $s \neq 0$). We finally notic that the diagonal entries of an hermitian matrix must necessarily be real.

1-2-2 Matrices and Linear Mapping:

Definition 1.8 A linear map from $\mathbb{C}^n$ into $\mathbb{C}^m$ is a function $f\colon \mathbb{C}^n \to \mathbb{C}^m$ such that $f(\alpha x + \beta y) = \alpha f(x) + \beta f(y),\ \forall \alpha, \beta \in K$ and $\forall x, y \in \mathbb{C}^n$.

The following result links matrices and linear maps.

Property 1.2 Let $f\colon \mathbb{C}^n \to \mathbb{C}^m$ be a linear map. Then, there exists a unique matrix $A_f \in \mathbb{C}^{m\times n}$ such that

$$f(x) = A_f x \qquad \forall x \in \mathbb{C}^n \qquad (1.3)$$

Conversely, if $A_f \in \mathbb{C}^{m\times n}$ then the function defined in (1.3) is a linear map from $\mathbb{C}^n$ into $\mathbb{C}^m$.

Example 1.1 An important example of a linear map is the counterclockwise rotation by an angle ϑ in the plane (x_1, x_2). The matrix associated with such a map is given by

$$G(\vartheta) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}, \quad c = \cos(\vartheta)\ , s = \sin(\vartheta)$$

and it is called a rotation matrix.

1-2-3 Operations with Block- Partitioned Matrices:

All the operations that have been previously introduced can be extended to the case of a block- partitioned matrix A , provided that the size of each single block is such that any single matrix operation is well- defined.

Indeed, the following result can be shown([15]).

Property 1.3 Let A and B be the block matrices

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1l} \\ \vdots & \ddots & \vdots \\ A_{k1} & \cdots & A_{kl} \end{bmatrix} \qquad B = \begin{bmatrix} B_{11} & \cdots & B_{1n} \\ \vdots & \ddots & \vdots \\ B_{m1} & \cdots & B_{mn} \end{bmatrix}$$

where A_{ij} and B_{ij} are matrices $(k_i \times l_j)$ and $\left(m_i \times n_j\right)$. Then we have

1. $\lambda A = \begin{bmatrix} \lambda A_{11} & \dots & \lambda A_{1l} \\ \vdots & \ddots & \vdots \\ \lambda A_{k1} & \dots & \lambda A_{kl} \end{bmatrix}$, $\lambda \in \mathbb{C}$, $A^T = \begin{bmatrix} {A_{11}}^T & \dots & {A_{1l}}^T \\ \vdots & \ddots & \vdots \\ {A_{k1}}^T & \dots & {A_{kl}}^T \end{bmatrix}$,
2. if $k = m\,, l = n\,, m_i = k_i$ and $n_j = l_j$, then

$$A + B = \begin{bmatrix} A_{11} + B_{11} & \dots & A_{1l} + B_{1l} \\ \vdots & \ddots & \vdots \\ A_{k1} + B_{k1} & \dots & A_{kl} + B_{kl} \end{bmatrix},$$

3. if $l = m, l_i = m_i$ and $k_i = n_i$ then, letting $C_{ij} = \sum_{s=1}^{m} A_{is}B_{sj}$,

$$AB = \begin{bmatrix} C_{11} & \dots & C_{1l} \\ \vdots & \ddots & \vdots \\ C_{k1} & \dots & C_{kl} \end{bmatrix}.$$

1-3 Trace and Determinant of a Matrix:

Let us consider a square matrix A of order n . The trace of a matrix is the sum of the diagonal entries of A , that is $tr(A) = \sum_{i=1}^{n} a_{ii}$.

We call the determinant of A the scalar defined through the following formula

$$\det(A) = \sum_{\pi \in P} sign(\pi) a_{1\pi_1} a_{2\pi_2} \dots a_{v\pi_n},$$

where $P = \{\pi = (\pi_1, \dots \pi_n)^T\}$ is the set of the $n!$ vectors that are obtained by permuting the index vector $i = (1, \dots, n)^T$ and sign(π) equal to1(respectively,-1)if an even (respectively, odd) number of exchanges is needed to obtain π from i .

The following properties hold

$\det(A) = \det(A^T)\,,\ \det(AB) = \det(A)\det(B)\,,\ \det(A^{-1}) = 1/\det(A)\,,$

$$\det(A^H) = \overline{\det(A)}\,, \quad \det(\alpha A) = \alpha^n \det(A)\,, \forall \alpha \in K.$$

Moreover, if two rows or columns of a matrix coincide, the determinant vanishes, while exchanging two rows (or columns) produces a change of sign in the determinant. Of course, the determinant of a diagonal matrix is the product of the diagonal entries.

Denoting by A_{ij} the matrix of order $n-1$ obtained from A by eliminating the $i-th$ row and the $j-th$ column, we call the complementary minor associated with the entry a_{ij} the determinant of the matrix A_{ij} . We call the $k-th$ principal (dominating) minor of A , d_k, the determinant of the principal submatrix of order k , $A_k = A(1{:}k, 1{:}k)$. If we denote by $\Delta_{ij} = (-1)^{i+j} det(A_{ij})$ the cofactor of the entry a_{ij} , the actual computation of the determinant of A can be performed using the following recursive relation

$$\det(A) = \begin{cases} a_{11} & if\ n=1 \\ \sum_{j=1}^{n} \Delta_{ij} a_{ij} & for\ n > 1 \end{cases} \qquad (1.4)$$

which is known as the Laplace rule. If A is a square invertible matrix of order n , then

$$A^{-1} = \frac{1}{\det(A)} c$$

where c is matrix having entries $\Delta_{ij}, \quad i, j = 1, \dots, n$.

As a consequence, a square matrix is invertible if its determinant is non vanishing. In the case of nonsingular diagonal matrices the inverse is still a diagonal matrix having entries given by the reciprocals of the diagonal entries of the matrix.

Every orthogonal matrix is invertible, its inverse is given by A^T , moreover $\det(A) = \mp 1$.

1-4 Eigenvalues and Eigenvectors:

Let A be a square matrix of order n with real complex entries , the number $\lambda \in \mathbb{C}$ is called an eigenvalue of A if there exists a nonull vector $x \in \mathbb{C}^n$ such that $Ax = \lambda x$ .The vector x is the eigenvector associated with the eigenvalue λ and the set of the eigenvalue of A is called the spectrum of A , denoted by $\sigma(A)$. We say that x and y are respectively a right eigenvector and a left eigenvector of A , associated with the eigenvalue λ , if

$$Ax = \lambda x , \quad y^H A = \lambda y^H$$

The eigenvalue λ corresponding to the eigenvector x can be determined by computing the Rayleigh quotient $\lambda = x^H Ax \therefore (x^H x)$. The number λ is the solution of the characteristic equation

$$\rho_A(\lambda) = \det(A - \lambda I) = 0,$$

Where $\rho_A(\lambda)$ is the characteristic polynomial. since this latter is a polynomial of degree n with respect to λ,

There certainly exist n eigenvalue of A not necessarily distinct. The following properties can be proved

$$\det(A) = \prod_{i=1}^{n} \lambda_i , \qquad tr(A) = \sum_{i=1}^{n} \lambda_i \qquad (1.5)$$

And since $\det(A^T - \lambda I) = \det(A - \lambda I)^T = \det(A - \lambda I)$ one concludes that $\sigma(A) = \sigma(A^T)$ and,in an analogous way, that $\sigma(A^H) = \sigma(\bar{A})$.

From the first relation in (1.5) it can be concluded that a matrix is singular if it has at least one null eigenvalue, since $\rho_A(0) = \det(A) = \prod_{i=1}^{n} \lambda_i$.

Secondly, if A has real entries, $p_A(\lambda)$ turns out to be a real – coefficient polynomial so that complex eigenvalues of A shall necessarily occur in complex conjugate pairs.

Finally, due to the Cayley-Hamilton Theorem if $\rho_A(\lambda)$ is the characteristic polynomial of A, then $\rho_A(\mathrm{A}) = 0$, where $\rho_A(A)$ denotes a matrix polynomial.[16].

The maximum module of the eigenvalues of A is called the spectral radius of A and is denoted by

$$\rho(A) = \max_{\lambda\in\sigma(A)}|\lambda| \qquad (1.6)$$

Characterizing the eigenvalue of a matrix as the roots of a polynomial implies in particular that ⲗ is an eigenvalue of $A \in \mathbb{C}^{n\times n}$ if $\bar{\lambda}$ is an eigenvalue of A^H. An immediate consequence is that $\rho(A) = \rho(A^H)$. M0reover, $\forall A \in \mathbb{C}^{n\times n}$, $\forall\alpha \in C, \rho(\alpha A)$, and $\rho(A^K) = [\rho(A)]^K \ \forall k \in \mathbb{N}$

Finally, assume that A is block triangular matrix

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1K} \\ 0 & A_{11} & \cdots & A_{2K} \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & A_{KK} \end{bmatrix}$$

As $p_A(\lambda) = p_{A_{11}}(\lambda)p_{A_{22}}(\lambda) \ldots), p_{A_{kk}}(\lambda)$, the spectrum of A is given by the union of the spectra of each single diagonal block. As a consequence if A is triangular, the eigenvalue of A are its diagonal entries.

For each eigenvalue λ of a matrix A the set of eigenvectors associated with λ , together with the null vector, identifies a subspace of $\mathbb{C}^n$ which is called the eigenspace associated with λ and corresponds by definition to ker(A- λI) .The dimension of the eigenspace is

$$\dim[\ker(A - \lambda I)] = n - rank(a - \lambda I),$$

and is called geometric multiplicity of the eigenvalue λ . It can never be greater than the algebraic multiplicity of λ , wich is the multiplicity of λ as a root of the characteristic polynomial. Eigenvalue having geometric multiplicity strictly less than algebraic one are called defective . A matrix having at least one defective eigenvalue is called defective.

The eigenspace associated with an eigenvalue of a matrix A is invariant with respect to A in the sense of the following definition.

Definition 1.9 A subspace S in $\mathbb{C}^{n}$is called invariant with respect to a square matrix A if $AS \subset S,$ wher AS is the transformed of S through A.

1-5 Similarity Transformations:

Definition 1.10 Let C be a square nonsingular matrix having the same order as the matrix A.

We say that the matrices A and $C^{-1}AC$ are similar and the trans formation from A to $C^{-1}AC$ is called a similarity transformation. Moreover, we say that the two matrices are unitarily similar if C is unitary.

Tow similar matrices share the same spectrum and the same characteristic polynomial .Indeed, it is easy to check that if (λ , x) is an eigenvalue – eigenvector pair of A , $(\lambda , c^{-1}x)$ is the same for the matrix $C^{-1}AC$ since

$$(C^{-1}AC) C^{-1}x = C^{-1}Ax = \lambda C^{-1}x$$

We notice in particular that the product matrices AB and BA , with $A \in \mathbb{C}^{n\times m}$ and $B \in \mathbb{C}^{m\times n}$, are not similar but satisfy the following property([17])

$$\sigma(AB)/\{0\} = \sigma(BA)/\{0\}$$

That is AB and BA share the same spectrum apart from null eigenvalues so that $\quad \sigma(AB) = \sigma(BA)$.

The use of similarity transformation aims at reducing the complexity of the problem of evaluating the eigenvalues of a matrix. Indeed, if a given matrix could be transformed into a similar matrix in diagonal or triangular form, the computation of the eigenvalues would be immediate. The main result in this direction is the following theorem([18]).

Property 1.4 (schur decomposition) Given $A \in \mathbb{C}^{n\times n}$, there exists u unitary such that

$$U^{-1}AU = U^{H}AU = \begin{bmatrix} \lambda_1 & b_{12} & \cdots & b_{1n} \\ 0 & \lambda_2 & \cdots & b_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & \lambda_n \end{bmatrix} = T,$$

Where λ_i are the eigenvalues of A.

It thus turns out that every matrix A is unitary similar to an upper triangular matrix . The matrices T and U are not necessarily unique .

The Schur decomposition theorem gives rise to several important result, among them, we recall:

1- Every hermitian matrix is unitarily similar to a diagonal real matrix m that is, when A is hermitian every Schur decomposition of A is diagonal. In such an event, since

$$U^{-1}AU = A = diag(\lambda_1, \ldots, \lambda_n),$$

it turns out that $AU = UA$, that is $Au_{i=}\lambda_{i\,u_i}$ for $i = 1, \ldots, n$ so that the column vectors of U are the eigenvectors of A. Moreover , since the eigenvectors are orthogonal two by two, it turns out that anhermitian matrix has a system of orthogonal eigenvectors that generates the whole space $\mathbb{C}^n$. Finally , it can be shown that a matrix A of order n is similar to a diagonal matrix D iff the eigenvectors of A form a basis for $\mathbb{C}^n$. [19]

2- A matrix $A \in \mathbb{C}^{n\times n}$ is normal iff it is unitarily similar to diagonal matrix .As a consequence , a normal matrix $A \in \mathbb{C}^{n\times n}$ admits the following spectral[20] decomposition $A = UAU^H = \sum_{i=1}^{n} \lambda_i \; u_i u_i^H$ being U unitary and A diagonal.

3- Let A and B be two normal and commutative matrix, then, the generic eigenvalue μ_i of $A + B$ is given by the sum $\lambda_i + \xi_i$, where λ_i and ξ_i are the eigenvalues of A and B associated with the same eigenvector .

There are, of course, non symmetric matrices that are similar to diagonal matrices, but these are not unitarily similar .

The Schur decomposition can be improved as follows ([21]) .

Property 1.5 (Canonical Jordan Form):

Let A be any square matrix. Then, there exists a nonsingular matrix X which transforms A into a block diagonal matrix J such that

$$X^{-1}AX = J = diag\big(J_{K1}(\lambda_1), J_{K2}(\lambda_2), \dots, J_{Ki}(\lambda_i)\big),$$

which is called canonical Jordan form, λ_i being the eigenvalues of A and

$$J_K(\lambda) = \lambda$$

if $k = 1$ and

$$J_K(\lambda) = \begin{bmatrix} \lambda & 1 & 0 & \dots & 0 \\ 0 & \lambda & 1 & \cdots & \vdots \\ \vdots & \ddots & \ddots & 1 & 0 \\ \vdots & & \ddots & \lambda & 1 \\ 0 & \cdots & \cdots & 0 & \lambda \end{bmatrix}, \text{ for } k > 1 .$$

If an eigenvalue is defective, the size of the corresponding Jordan block is greater than one. Therefore , the canonical Jordan form tells us that a matrix can be diagonalized by a similarity transformation iff it is non defective. For this reason, the non defective matrices are called diagonalizable. In particular, normal matrices are diagonalizable.

Partitioning X by columns,$X = (x_1, \dots, x_n)$, it can be seen that the k_i vectors associated with the Jordan block $J_{ki}(\lambda_i)$ satisfy the following recursive relation

$$Ax_i = \lambda_i x_i\,, \qquad l = \textstyle\sum_{j=1}^{i-1} m_j + 1\,, \tag{1.7}$$

$$Ax_i = \lambda_i x_j + x_{j-1}\,, \quad j = l+1, \dots, l-1+k_i\,, \quad \text{if } k_i \neq 1\,.$$

The vector x_i are called principal vector or generalized eigenvectors of A .

1-6 The Singular Value Decomposition (SVD):

Any matrix can be reduced in diagonal form by a suitable pre and post multiplication by unitary matrices. Precisely, the following result holds.

Property 1.6

Let $A \in \mathbb{C}^{m\times n}$. There exist two unitary matrices $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ such that

$$U^H AV = \sum = diag(\sigma_{1\,,}, \dots, \sigma_p\,) \in \mathbb{C}^{m\times n} \quad \text{with } p = \min(m,n) \quad (1.8)$$

And $\sigma_1 \geq \cdots \geq \sigma_p \geq 0$. Formula (1.8) is called Singular Value Decomposition or (SVD) of A and the numbers σ_i $(or\ \sigma_i(A))$ are called singular value of A .

If A is a real–valued matrix , U and V will also be real –value and in (1.8) U^T must be written instead of U^H. The following characterization of the singular values holds

$$\sigma_i(A) = \sqrt{\lambda_i}(A^H A), \quad i = 1\,, \dots, n\,. \qquad (1.9)$$

Indeed , from (1.8) it follows that $A = U \sum V^H$, $A^H = V \sum U^H$ so that , U and V being unitary , $A^H A = V\ \sum^2\ V^H$, that is , $\lambda_i(A^H A) = \lambda_i\big(\sum^2\ \big) = (\sigma_i(A))^2$. Since AA^H and $A^H A$ are hermitian matrices , the columns of U , called the left singular vectors of A , turn out to be the eigenvectors of AA^H and ,therefore ,they are not uniquely defined. The same holds for the columns of V , which are the right singular vectors of A .

Relation (1.9) implies that if $A \in \mathbb{C}^{n\times n}$ is hermitian with eigenvalues given by $\lambda_1, \lambda_2, \dots \lambda_n$,then the singular values of A concide with the modules of the eigenvalues of A .Indeed because $AA^H = A^2$, $\sigma_i = \sqrt{\lambda_i{}^2} = |\lambda_i|$ for $i = 1\,, \dots, n$. As far as rank is concerned , if

$$\sigma_1 \geq \cdots \geq \sigma_r \geq \sigma_{r+1} = \cdots = \sigma_p = 0\,,$$

Then the rank of A is r , the kernel of A is the span of the column vectors of $V, \{v_{r+1}, \dots, v_n\}$, and the range of A is the span of the column vectors of $U, \{u_{r+1}, \dots, u_n\}$.

Definition 1.11

Suppose that $A \in \mathbb{C}^{n\times n}$ has rank equal to r and that it admits a AVD of the $U^H AV = \sum$ Then matrix $A^\dagger = V \sum^\dagger U^H$ is called the Mooer-Penrose pseudo – inverse matrix , being

$$\sum \dagger = diag\left(\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_r}, 0, \dots, 0\right) \qquad (1.10)$$

The matrix $A^\dagger$ is also called the generalized inverse of A . Indeed , if $rank(A) = n < m$, then $A^\dagger = (A^T A)^{-1} A^T$, while if $(n = m = rank(A),$, $A^\dagger = A^{-1}$.

1-7 Matrix Norms:

Definition 1.12 A matrix norm is mapping $||.||: \mathbb{R}^{m\times n} \to \mathbb{R}$ such that :

1. $||A|| \geq 0 \quad \forall A \in \mathbb{R}^{m\times n}$ and $||A|| = 0$ if and only if $A = 0$;
2. $||\alpha A|| = |\alpha| ||A|| \quad \forall \alpha \in \mathbb{R}$ and $\forall A \in \mathbb{R}^{m\times n}$ (homogeneity);
3. $||A + B|| \leq ||A|| + ||B|| \ \forall A, B \in \mathbb{R}^{m\times n}$ (triangular inequality).

Unless otherwise specified we shall employ the symbol $||.||$,to denote matrix norms .

We can better characterize the matrix norms by introducing the concepts of compatible norm and induced by a vector norm.

Definition 1.13 We say that a matrix norm $\|.\|$ is compatible or consistent with a vector norm $\|.\|$ if

$$\|Ax\| \leq \|A\|\|x\|, \quad \forall x \in \mathbb{R}^n \qquad (1.11)$$

More generally , given three norms, all denoted by $\|.\|$, albeit defined on $\mathbb{R}^m$, $\mathbb{R}^n$ and $\mathbb{R}^{m\times n}$, respectively, we say that they are consistent if $\forall x \in \mathbb{R}^n$, $Ax = y \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$ we have that $\|y\| \leq \|A\|\|x\|$.

In order to single out matrix norms of practical interest , following property is in general required .

Definition 1.14 We say that a matrix norm $\|.\|$ is sub- multiplicative if $\forall A \in \mathbb{R}^{n\times m}$ $\forall B \in \mathbb{R}^{m\times q}$

$$\|AB\| \leq \|A\|\|B\| \qquad (1.12)$$

This property is not satisfied by any matrix norm. For example ([22]), the norm $\|A\|_\Delta = max|a_{ij}| \;\; for\; i = 1, \dots, n,\; j = 1, \dots, m$ .does not satisfy (1.12) if applied to the matrices

$$A = B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

since $2 = \|AB\|_\Delta > \|A\|_\Delta\|B\|_\Delta = 1$.

Notice that , given a certain sub-multiplicative matrix norm $\|.\|_\alpha$, there always exists a consistent vector norm. For instance , given any fixed vector $y \neq 0$ in $\mathbb{C}^n$, it suffices to define the consistent vector norm as

$$\|x\| = \|xy^H\|_\alpha \qquad x \in \mathbb{C}^n$$

As a consequence , in the case of sub-multiplicative matrix norms it is no longer necessary to explicitly specify the vector norm with respect to the matrix norm is consistent .

Theorem 1.1 Let $\|.\|$ be a vector norm. The function

$$\|A\| = sup\frac{\|Ax\|}{\|x\|} \quad x \neq 0 \qquad (1.13)$$

is a matrix norm called induced matrix norm or natural matrix norm.

Proof. We start by noticing that (1.13) is equivalent to

$$\|A\| = \sup\|Ax\|. \quad \|x\| = 1 \quad (1.14)$$

Indeed, one can define for any $x \neq 0$ the unit vector $u = x/\|x\|$, so that (1.13) becomes

$$\|A\| = sup\|Au\| = \|Aw\| \;\; with\|w\| = 1, \;\; \|u\| = 1$$

This being taken as given , let us check that (1.13)(or equivalently, (1.14) is actually a norm, making direct use of Definition 1.9.

1. If $\|Ax\| \geq 0$, then it follows that $\|A\| = sup\|Ax\| \geq 0$, $\|x\| = 1$. Moreover

$$\|A\| = \sup\frac{\|Ax\|}{\|x\|} = 0 \iff \|Ax\| = 0 \forall x \neq 0$$

and $Ax = 0 \forall x \neq 0$ if and only if $A = 0$, therefore $\|A\| = 0 \iff A = 0$.

2. Given a scalar α ,

$$\|\alpha A\| = \sup\|\alpha Ax\| = |\alpha| sup\|Ax\| = |\alpha|\|A\|.$$

3. Finally , triangular inequality holds. Indeed , by definition of supremum , if $x \neq 0$ then

$$\frac{\|Ax\|}{\|x\|} \leq \|A\| \Rightarrow \|Ax\| \leq \|A\|\|x\|,$$

so that, taking x with unit norm, one gets

$\|(A+B)x\| \leq \|Ax\| + \|Bx\| \leq \|A\| + \|B\|$,

from which it follows that $\|(A+B)\| = sup\|(A+B)x\| \leq \|A\| + \|B\|$, $\|x\| = 1$.

Relevant instances of induced matrix norms are so-called p- norm defined as

$$\|A\|_p = sup\frac{\|Ax\|_p}{\|x\|_p} \quad ,x \neq 0$$

The $1 - norm$ and the infinity norm are easily computable since

$$\|A\|_1 = \max_{j=1,\ldots,n}\sum_{i=1}^{m}|a_{ij}| \ , \|A\|_\infty = \max_{i=1,\ldots,n}\sum_{i=1}^{n}|a_{ij}|$$

and they called the column sun norm and the row sum norm, respectively .

Moreover, we have $\|A\|_1 = \|A^T\|_\infty$ and , if A is self- adjoint or real symmetric , $\|A\|_1 = \|A\|_\infty$.

A special discussion is deserved by the $2 - norm$ or spectral norm for which the following theorem holds.

Theorem 1.2 Let $\sigma_1(A)$ be the largest singular value of A .Then

$$\|A\|_2 = \sqrt{\sigma(A^HA)} = \sqrt{\sigma(AA^H)} = \sigma_1(A). \qquad (1.15)$$

In particular , if A is hermitian (or real and symmetric), then

$\|A\|_2 = \sigma_1(A),$ (1.16)

while, if A is unitary , $\|A\|_2 = 1$.

Proof. Since $\sigma(A^HA)$ is hermitian , there exists a unitary matrix U such that

$$U^HA^HAU = diag(\mu_1, \ldots, \mu_n),$$

where μ_i are the (positive) eigenvalues of A^HA. Let $y = U^Hx,$ then

$$\|A\|_2 = sup\sqrt{\frac{(A^HAx,x)}{(x,x)}} = sup\sqrt{\frac{(U^HA^HAUy,y)}{(y,y)}} \quad ,x \neq 0, y \neq 0$$

$$= sup\sqrt{\sum_{i=1}^{n}\mu_i|y_i|^2/\sum_{i=1}^{n}|y_i|^2} = \sqrt{\max_{i=1,\ldots,n}\mu_i}\ , \ y \neq 0$$

from which (1.15) follows, thanks to (1.9).

If A is hermitian , the same considerations as above apply directly to A .

Finally , if A is unitary

$$\|Ax\|_2^2 = (Ax, Ax) = (x, A^H Ax) = \|x\|_2^2$$

so that $\|A\|_2 = 1$.

As a consequence, the computation of $\|A\|_2$ is much more expensive than that of $\|A\|_\infty$or $\|A\|_1$.However , if only an estimate of $\|A\|_2$ is required, the following relations can be profitably employed in the case of square matrices.

$$\max_{i.j}|a_{ij}| \leq \|A\|_2 \leq n \max_{i,j}|a_{i,j}|,$$

$$\frac{1}{\sqrt{n}}\|A\|_\infty \leq \|A\|_2 \leq \sqrt{n}\|A\|_\infty,$$

$$\frac{1}{\sqrt{n}}\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\|A\|_1,$$

$$\|A\|_2 \leq \sqrt{\|A\|_1\|A\|_\infty}.$$

Moreover, if A is normal then $\|A\|_2 \leq \|A\|_p$ for any n and all $p \geq 2$.

Theorem1.3 Let $|\|.\||$ be a matrix norm induced by a vector norm $\|.\|$. then

1. $\|Ax\| \leq |\|A\||\|x\|$, that is , $|\|.\||$ is norm compatible with $\|.\|$,
2. $|\|I\|| = 1$,
3. $|\|AB\|| \leq |\|A\|||\|B\||$, that is , $|\|.\||$ is sub-multiplicative.

Proof. Part 1 of the theorem is already contained in the proof of the theorem 1.1, while part 2 follows from the fact that $|\|I\|| = \sup \|Ix\|/\|x\| = 1$. Part 3 is simple to check.

Notice that the $p - norm$ are sub-multiplicative. Moreover, we remark that the sub- multiplicativety property by itself would only allow us to conclude that $|\|I\|| = |\|I.I\|| \leq |\|I\||^2$.

1.7.1 Relation between Norms and the Spectral Radius of a matrix:

Theorem 1.4 Let $\|.\|$ be a consistent matrix norm, then

$$\rho(A) \leq \|A\| \quad \forall A \in \mathbb{C}^{n\times n}.$$

Proof. Let λ be an eigenvalue of A and $v \neq 0$ an associated eigenvector . As a consequence , since $\|.\|$ is consistent, we have

$$|\lambda|\|v\| = \|\lambda v\| = \|Av\| \leq \|A\|\|v\|$$

$$\text{so that } |\lambda| \leq \|A\|.$$

More precisely, the following property holds ([23]).

Property 1.7 Let $A \in \mathbb{C}^{n\times n}$ and $\varepsilon > 0$. Then , there exists a consistent matrix norm $\|.\|_{A,\varepsilon}$ (depending on ε) such that

$$\|A\|_{A,\varepsilon} \leq \rho(A) + \varepsilon.$$

As a result, having fixed an arbitrarily small tolerance, there always exists a matrix norm which is arbitrarily close to the spectral radius of A, namely

$$\rho(A) = \inf_{\|.\|} \|A\| , \qquad (1.17)$$

the infimum being taken on the set of all the consistent norms.

For the sake of clarity, we notice that the spectral radius is a sub-multiplicative seminorm, since it is not true that $\rho(A) = 0$ iff $A = 0$.

As an example, any triangular matrix with null diagonal entries clearly has spectral radius equal to zero. Moreover , we have the following result.

Property 1.7 Let A be a square matrix and let $\|.\|$ be a consistent norm. Then

$$\lim_{m\to\infty} \|A^m\|^{1/m} = \rho(A).$$

1.7.2 Sequences and Series of Matrices:

A sequence of matrices $\{A^{(k)}\} \in \mathbb{R}^{n\times n}$ is said to converge to a matrix $A \in \mathbb{R}^{n\times n}$ if

$\lim_{k\to\infty} \|A^{(k)} - A\| = 0.$

The choice of the norm dose not influence the result since in $\mathbb{R}^{n\times n}$ all norms are equivalent.

In particular, when studying the convergence of iterative methods for solving linear system , one is interested in the so-called convergent matrices for which

$$\lim_{k\to\infty} A^{(K)} = 0,$$

0 being the null matrix .The following theorem holds.

Theorem 1.5 Let A be a square matrix , then

$$\lim_{k\to\infty} A^{(K)} = 0 \Leftrightarrow \rho(A) < 1. \qquad (1.18)$$

Moreover, the geometric series $\sum_{k=0}^{\infty} A^{(K)}$ is convergent iff $\rho(A) < 1$. In such a case

$$\sum_{k=0}^{\infty} A^{(K)} = (I - A)^{-1} \qquad (1.19)$$

As a result, if $\rho(A) < 1$ the matrix $(I - A)$ is invertible and the following inequalities hold

$$\frac{1}{1+\|A\|} \le \|(I - A)^{-1}\| \le \frac{1}{1-\|A\|} \qquad (1.20)$$

where $\|.\|$ is an induced matrix norm such that$\|A\| < 1$.

Proof. Let us prove (1.18) . Let $\rho(A) < 1$, then $\exists\varepsilon > 0$ such that $\rho(A) < 1 - \varepsilon$ and thus thanks to property 1.6, there exists a consistent matrix norm $\|.\|$ such that $\|A\| \le \rho(A) + \varepsilon < 1$. From the fact that $\|A^k\| \le \|A\|^k < 1$ and from the definition of convergence it turns out as $k \to \infty$ the sequence $\{A^{(k)}\}$ tends to zero. Conversely, assume that $\lim_{k\to\infty} A^k = 0$ and let λ denote an eigenvalue of A . Then , $A^k x = \lambda^k x$, being $x(\neq 0)$ an eigenvector associated with λ, so that $\lim_{k\to\infty} \lambda^k = 0$. As consequence, $|\lambda| < 1$and because this is true for a generic eigenvalue one gets $\rho(A) < 1$ as desired. Relation (1.19) can be obtained noting first that the eigenvalues of $(I - A)$ are given by $1 - \lambda(A), \lambda(A)$ being the generic eigenvalue of A. On the other hand, since $\rho(A) < 1$, we deduce that $I - A$ is nonsingular .Then , from the identity

$$(I - A)(I + A + \cdots + A^n) = (I - A^{n+1})$$

and taking the limit for n tending to infinity the thesis follows since

$$(I - A)\sum_{k=0}^{\infty} A^k = 1.$$

Finally , thanks to Theorem 1.3, the equality $||I|| = 1$ holds, so that

$$1 = ||I|| \leq ||I - A|| ||(I - A)^{-1}|| \leq (1 + ||A||)||(I - A)^{-1}||,$$

giving the first inequality in (1.20). As for the second part, noting that $I = I - A + A$ and multiplying both sides on the right by $(I - A)^{-1}$, one gets $(I - A)^{-1} = I + A(I - A)^{-1}$. Passing to the norms, we obtain

$$||(I - A)^{-1}|| \leq 1 + ||A|| ||(I - A)^{-1}||,$$

and thus the second inequality , since $||A|| < 1$.

Remark 1.1 The assumption that there exists an induced matrix norm such that $||A|| < 1$ is justified by Property 1.6, recalling that A is convergent and, therefore , $\rho(A) < 1$.

Notice that (1.19) suggests an algorithm to approximate the inverse of a matrix by a truncated series expansion .

Chapter Two

Direct Method for the Solution of Linear system

2-0 Introduction:

A system of m linear equations in n unknowns consists of algebraic relations of the form

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, \dots, m \qquad (2.1)$$

where x_j are the unknowns, a_{ij} are coefficients of the system and b_i are the components of the right hand side. System (2.1) can be more conveniently written in matrix form as

$$Ax = b, \qquad (2.2)$$

where we have denoted by $A = (a_{ij}) \in \mathbb{C}^{m\times n}$ the coefficient matrix , by $b = (b_i) \in \mathbb{C}^m$ the right side vector and by $x = (x_i) \in \mathbb{C}^n$ the unknown vector , respectively. We call a solution of (2.2) any n-tuple of values x_i which satisfies (2.1).

We shall be mainly dealing with real- valued square systems of order n that is, systems of the form (2.2) with $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. In such cases existence and uniqueness of the solution of (2.2) are ensured if one of the following (equivalent) hypotheses holds:

1. A is invertible;
2. $rank(A) = n$;
3. the homogeneous system $Ax = 0$ admits only the null solution.

The solution of system (2.2) is formally provided by Cramer′s rule

$$x_j = \frac{\Delta_j}{\det(A)}, \quad j = 1, \dots, n, \qquad (2.3)$$

where Δ_j is the determinant of the matrix obtained by substituting the $j-th$ column of A with the right hand side b . This formula is, however, of little practical use . Indeed, if the determinants are evaluated by the recursive relation (1.4)the computational effort of Cramer′s rule is of the order of $(n+1)$! flops and therefore turns out to be unacceptable even for small dimensions of A (for instance, a computer able to perform 10^9 flops per second would take $9.6 \,. 10^{47}$ years to solve a linear system of 50 equations).

For this reason, numerical methods that are alternatives to Cramer′s rule have been developed. They are called direct methods if they yield the solution of the system in a finite number of steps, iterative if they require (theoretically) an infinite number of steps. Iterative method will be addressed in the next chapter . We notice from now on that the choice between a direct and an iterative method does not depend only on the theoretical efficiency of the scheme, but also on the particular type of matrix, on memory storage requirements and, finally, on the architecture of the computer.

2-1 Stability Analysis of Linear Systems:

Solving a linear system by a numerical method invariably leads to the introduction of rounding errors. Only using stable numerical methods can keep a way the propagation of such errors from polluting the accuracy of the solution. In this section tow aspects of stability analysis will be addressed.

Firstly, we will analyze the sensitivity of the solution of (2.2) to changes in the data A and b (forward a priori analysis). Secondly, assuming that an approximate solution $\hat{x}$ of (2.2) is available, we shall quantify the perturbations on the data A and b in order for $\hat{x}$ to be the exact solution of a perturbed system (backward a priori analysis). The size of these perturbations will in turn allow us to measure the accuracy of the computed solution $\hat{x}$ by the use of posteriori analysis.

2-1-1 The Condition Number of a Matrix

The condition number of a matrix $A \in \mathbb{C}^{n\times n}$ is defined as

$$K(A) = \|A\|\|A^{-1}\|, \qquad (2.4)$$

where $\|.\|$ is an induced matrix norm. In general $K(A)$ depends on the choice of the norm; this will be made clear by introducing a subscript into the notation, for instance, $K_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty$. More generally, $K_p(A)$ will denote the condition number of A in the $p-norm$. Remarkable instances are $p = 1$, $p = 2$ and $p = \infty$.

Let us start by noticing that $K(A) \geq 1$ since

$$1 = \|AA^{-1}\| \leq \|A\|\|A^{-1}\| = K(A).$$

Moreover, $K(A^{-1}) = K(A)$ and $\forall\alpha \in \mathbb{C}$ with $\alpha \neq 0$, $K(\alpha A) = K(A)$. Finally, if A is orthogonal, $K_2(A) = 1$ since $\|A\|_2 = \sqrt{\rho A^T A} = \sqrt{\rho(I)} = 1$ and $A^{-1} = A^T$. The condition number of a singular matrix is set equal to infinity.

For $p = 2$, $K_2(A)$ can be characterized as follows. Starting from (1.15) it can be proved that

$$K_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sigma_1(A)}{\sigma_n(A)}$$

where $\sigma_1(A)$ and $\sigma_n(A)$ are the maximum and minimum singular values of A (see property 1.5). As a consequence, in the case of symmetric positive definite matrices we have

$$K_2(A) = \frac{\lambda_{max}}{\lambda_{min}} = \rho(A)\rho(A^{-1}) \qquad (2.5)$$

where λ_{max} and λ_{min} are the maximum and minimum eigenvalues of A. To check (2.5), notice that

$$\|A\|_2 = \sqrt{\rho A^T A} = \sqrt{\rho(A^2)} = \sqrt{\lambda_{max}^2} = \lambda_{max}.$$

Moreover, since $\lambda(A^{-1}) = 1/\lambda(A)$, one gets $||A^{-1}||_2 = 1/\lambda_{min}$ from which (2.5) follows. For that reason, $K_2(A)$ is called spectral condition number.

Remark 2.1 Define the relative distance of $A \in \mathbb{C}^{n\times n}$ from the set of singular matrices with respect to the $p - norm$ by

$$dist_p(A) = min\left\{\frac{||\delta A||_p}{||A||_p} : \ A + \delta A \ \ is \ \ singular\right\}.$$

It can then be shown that ([36])

$$dist_p(A) = \frac{1}{K_p(A)} \ . \qquad (2.6)$$

Equation (2.6) suggests that a matrix A with a high condition number can behave like a singular matrix of the form $A + \delta A$. In other words, null perturbation in the right hand side do not necessarily yield non vanishing changes in the solution since, if $A + \delta A$ is singular, the homogeneous system $(A + \delta A)z = 0$ does no longer admit only the null solution. From (2.6) it also follows that if $A + \delta A$ is nonsingular then

$$||\delta A||_p ||A||_p < 1. \qquad (2.7)$$

Relation (2.6) seems to suggest that a natural candidate for measuring the ill-conditioning of a matrix is its determinant, since from (2.3) one is prompted to conclude that small determinants mean nearly-singular matrices.

2-1-2 Forward a priori Analysis

In this section we introduce a measure of the sensitivity of the system to changes in the data.

Due to rounding errors, a numerical method for solving (2.2) does not provide the exact solution but only an approximate one, which satisfies a perturbed system. In other words, a numerical method yields an (exact) solution $x + \delta x$ of the perturbed system.

$$(A + \delta A)(x + \delta x) = b + \delta b. \qquad (2.8)$$

The next result provides an estimate of δx in terms of δA and δb.

Theorem 2.1 Let $A \in \mathbb{R}^{n\times n}$ be a nonsingular matrix and $\delta A \in \mathbb{R}^{n\times n}$ be such that (2.7) is satisfied for a matrix norm $||.||$. Then, if $x \in \mathbb{R}^n$ is the solution of $Ax = b$ with $b \in \mathbb{R}^n$ $(b \neq 0)$ and $\delta x \in \mathbb{R}^n$ satisfies (2.8) for $\delta b \in \mathbb{R}^n$,

$$\frac{||\delta x||}{||x||} \leq \frac{K(A)}{1-K(A)||\delta A||/||A||}\left(\frac{||\delta b||}{||b||} + \frac{||\delta A||}{||A||}\right). \qquad (2.9)$$

Proof. From (2.7) it follows that the matrix $A^{-1}\delta A$ has norm less than 1. Then, due to Theorem 1.5, $1 + A^{-1}\delta A$ is invertible and from (1.20) it follows that

$$||I + A^{-1}\delta A|| \leq \frac{1}{1-||A^{-1}\delta A||} \leq \frac{1}{1-||A^{-1}||||\delta A||}. \qquad (2.10)$$

On the other hands, solving for δx in (2.8) and recalling that $Ax = b$, one gets

$$\delta x = (\, I + A^{-1}\delta A)^{-1}A^{-1}(\delta b - \delta A x),$$

from which, passing to the norms and using (2.10), it follows that

$$||\delta x|| \leq \frac{||A^{-1}||}{1 - ||A^{-1}||||\delta A||}(||\delta b|| + ||\delta A||||x||).$$

Finally, dividing both sides by $||x||$ (which is nonzero since $b \neq 0$ and A is nonsingular) and noticing that $||x|| \geq ||b||/||A||$, the result follows.

Theorem 2.2 Assume that the conditions of the Theorem 3.1hold and let $\delta A = 0$. Then

$$\frac{1}{K(A)}\frac{||\delta b||}{||b||} \leq \frac{||\delta x||}{||x||} \leq K(A)\frac{||\delta b||}{||b||} \qquad (2.11)$$

Proof. We will prove only the first inequality since the second one directly follows from (2.9). Relation $\delta x = A^{-1}\delta b$ yields $||\delta b|| \leq ||A||||\delta x||$. Multiplying both sides by $||x||$ and recalling that $||x|| \leq ||A^{-1}||||b||$ it follows that $||x||||\delta b|| \leq K(A)||b||||\delta x||$, which is the desired inequality.

In order to employ the inequalities (2.10) and (2.11) in the analysis of propagation of rounding errors in the case of direct methods, $||\delta A||$ and $||\delta b||$ should be bounded in terms of the dimension of the system and of the characteristics of the floating – point arithmetic that is being used.

It is indeed reasonable to expect that the perturbations induced by a method for solving a linear system are such that $||\delta A|| \leq \gamma||A||$ and $||\delta b|| \leq \gamma||b||$, γ being a positive number that depends on the round off unit u (for example, we shall assume henceforth that $\gamma = \beta^{1-t}$ where β is the base and t is the number of digits of the mantissa of the floating – point system $\mathbb{F}$). In such a case (2.9) can be completed by the following theorem.

Theorem 2.3 Assume that $||\delta A|| \leq \gamma||A||$, $||\delta b|| \leq \gamma||b||$ with $\gamma \in \mathbb{R}^+$ and $\delta A \in \mathbb{R}^{n\times n}$, $\delta b \in \mathbb{R}^n$. Then, if $\gamma K(A) < 1$ the following inequalities hold

$$\frac{||x+\delta x||}{||x||} \leq \frac{1+\gamma K(A)}{1-\gamma K(A)}, \qquad (2.12)$$

$$\frac{||\delta x||}{||x||} \leq \frac{2\gamma}{1-\gamma K(A)} K(A). \qquad (2.13)$$

Proof. From (2.8) it follows that $(1 + A^{-1}\delta A)(x + \delta x) = x + A^{-1}\delta b$. Moreover, since $\gamma K(A) < 1$ and $||\delta A|| \leq \gamma||A||$ it turns out that $1 + A^{-1}\delta A$ is nonsingular.

Taking the inverse of such a matrix and passing to the norms we get $||x + \delta x|| \leq ||\ (1 + A^{-1}\delta A)||(||x|| + \gamma||A^{-1}||||b||)$. From Theorem 1.5 it then follows that

$$||x + \delta x|| \leq \frac{1}{1-||A^{-1}\delta A||}(||x|| + \gamma||A^{-1}||||b||),$$

which implies (2.12), since $||A^{-1}\delta A|| \leq \gamma K(A)$ and $||b|| \leq ||A||||x||$.

Let us prove (2.13). Subtracting (2.2) from (2.8) it follows that

$$A\delta x = -\delta A(x + \delta x) + \delta b.$$

Inverting A and passing to the norms, the following inequality is obtained

$$\|\delta x\| \leq \|A^{-1}\delta A\|\|x+\delta x\| + \|A^{-1}\|\|\delta b\|$$

$$\leq \gamma K(A)\|x+\delta x\| + \gamma\|A^{-1}\|\|b\|. \qquad (2.14)$$

Dividing both sides by $\|x\|$ and using the triangular inequality $\|x+\delta x\| \leq \|\delta x\| + \|x\|$, we finally get (2.13).

Remarkable instances of perturbations δA and δb are those for which $|\delta A| \leq |A|$ and $|\delta b| \leq \gamma|b|$ with $\gamma \geq 0$. Hereafter, the absolute value notation $B = |A|$ denotes the matrix $n \times n$ having entries $b_{ij} = |a_{ij}|$ with $i,j = 1,\ldots,n$ and the inequality $C \leq D$, with $C, D \in \mathbb{R}^{m\times n}$ has the following meaning

$$c_{ij} \leq d_{ij} \text{ for } i = 1,\ldots,m,\ j = 1,\ldots,n.$$

If $\|.\|_\infty$ is considered , from (3.14) it follows that

$$\frac{\|\delta x\|_\infty}{\|x\|_\infty} \leq \gamma \frac{\||A^{-1}||A||x| + |A^{-1}||b|\|_\infty}{1-\gamma\||A^{-1}||A|\|_\infty\|x\|_\infty}$$

$$\leq \frac{2\gamma}{1-\gamma\||A^{-1}||A|\|_\infty}\||A^{-1}||A|\|_\infty. \qquad (2.15)$$

Estimate (2.15) is generally too pessimistic; however, the following component wise error estimates of δx can be derived from (2.15)

$$|\delta x_i| \leq \gamma|r_{(i)}^T||A||x+\delta x|, \quad i = 1,\ldots,n \quad if \ \ \delta b = 0\ ,$$

$$\frac{|\delta x_i|}{|x_i|} \leq \gamma\frac{|r_{(i)}^T||b|}{|r_{(i)}^T b|},\ i = 1,\ldots,n \quad if \ \ \delta A = 0 \qquad (2.16)$$

being $r_{(i)}^T$ the row vector $r_{(i)}^T A^{-1}$. Estimates (2.16) are more stringent than (2.15). The first inequality in (2.16) can be used when the perturbed solution $x+\delta x$ is known, being henceforth $x+\delta x$ the solution computed by a numerical method.

In the case where $|A^{-1}||b| = |x|$, the parameter γ in (2.15) is equal perturbations to the right side. A slightly worse situation occurs when A is a triangular M-matrix and b has positive entries. In such a case γ is bounded by $2n-1$, since

$$\left|r_{(i)}^T\right||A||x| \leq (2n-1)|x_i|.$$

Example 2.1 Consider the linear system $Ax=b$ with

$$A = \begin{bmatrix} \alpha & \frac{1}{\alpha} \\ 0 & \frac{1}{\alpha} \end{bmatrix}, \qquad b = \begin{bmatrix} \alpha^2 + \frac{1}{\alpha} \\ \frac{1}{\alpha} \end{bmatrix}$$

which has solution $x^T = (\alpha, 1)$, when $0 < \alpha < 1$. Let us compare the results obtained using (2.15) and (2.16). From

$$|A^{-1}||A||x| = |A^{-1}||b| = (\alpha + \frac{2}{\alpha^2}, 1)^T \qquad (2.17)$$

it follows that the supremum of (2.17) is unbounded as $\alpha \to 0$, exactly as happens in case of $||A||_\infty$. On the other hand, the amplification factor of the error in (2.16)is bounded. Indeed, the component of the maximum absolute value, x_2, of the solution, satisfies $\left|r_{(i)}^T\right||A||x|/x_2 = 1$.

2-1-3 Backward a priori Analysis

The numerical methods that we have considered thus far do not require the explicit computation of the inverse of A to solve $Ax=b$. However, we can always assume that they yield an approximate solution of the form $\hat{x} = Cb$, where the matrix C, due to rounding errors, is an approximation of A^{-1}.

In practice, C is very seldom constructed; in case this should happen, the following result yields an estimate of the error that is made substituting C for A^{-1} ([23]) .

Property 2.1 Let $R = AC - I$; if $\|R\| < 1$, then A and C are nonsingular and

$$\|A^{-1}\| \leq \frac{\|C\|}{1-\|R\|}, \ \frac{\|R\|}{\|A\|} \leq \|C - A^{-1}\| \leq \frac{\|C\|\|R\|}{1-\|R\|} . \qquad (2.18)$$

In the frame of backward a priori analysis we can interpret C as being the inverse of $A + \delta A$ (for a suitable unknown δA). We are thus assuming that $C(A + \delta A) = I$. This yields

$$\delta A = C^{-1} - A = (AC - I)C^{-1} = -RC^{-1}$$

and, as a consequence if $\|R\| < 1$ it turn out that

$$\|\delta A\| \leq \frac{\|R\|\|A\|}{1-\|R\|}, \qquad (2.19)$$

having used the first inequality in (2.18), where A is assumed to be an approximation of the inverse of C (notice that the roles of C and A can be interchanged).

2-2 The Gaussian Elimination Methods(GEM) and LU factorization

The Gaussian elimination method aims at reducing the system $Ax = b$ to be an equivalent system (that is, having the same solution) of the form $Ux = \hat{b}$, where U is an upper triangular matrix and $\hat{b}$ is an updated right side vector . This latter system can then be solved by the backward substitution method. Let us denote original system by $A^{(1)}x = b^{(1)}$. During the reduction procedure we basically employ the property which states that replacing one of the equations by the difference between this equation and another one multiplied by a non null constant yields an equivalent system (i.e., one with the same solution).

Thus, consider a nonsingular matrix $A \in \mathbb{R}^{n\times n}$, and suppose that the diagonal entry a_{11} is non vanishing. Introducing the multipliers

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}} , \quad i = 2,3,\dots,n,$$

Where $a_{ij}^{(1)}$ denote the elements of $A^{(1)}$, it is possible to eliminate the unknown x_1 form the rows other than the first one by simply subtracting form row i , with $i = 2,3,\dots,n,$ the first row multiplied by m_{i1} and doing the same on the right side. If we now define

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)} , \qquad i,j = 2,\dots,n,$$

$$b_i^{(2)} = b_i^{(1)} - m_{i1}b_i^{(1)} , \quad i = 2,\dots,n,$$

where $b_i^{(1)}$ denote the components op b^1 , we get a new system of the form .

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix},$$

which we denote by $A^{(2)}x = b^{(2)}$, that is equivalent to starting one. Similarly , we can transform the system in such a way that the unknown x_2 is eliminated from rows $3,\dots,n$. In general , we end up with the finite sequence of systems

$$A^{(k)}x = b^{(k)}, \quad 1 \le k \le n , \qquad (2.20)$$

where , for $k \le 2$, matrix $A^{(k)}$ takes the following form

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & \dots & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & & a_{2n}^{(2)} \\ \vdots & & \ddots & & & \vdots \\ 0 & \dots & 0 & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & & \vdots & & & \vdots \\ 0 & \dots & 0 & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix}$$

having assumed that $a_{ii}^{(i)} \neq 0$ for $i = 1,2,\dots,k$. It is clear that for $k = n$ we obtain the upper triangular system $A^{(n)}x = b^{(n)}$

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & a_{2n}^{(2)} \\ \vdots & & \ddots & & \vdots \\ 0 & & & \ddots & \vdots \\ 0 & & & & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ \vdots \\ b_n^{(2)} \end{bmatrix},$$

Consistently with the notations that have been previously introduced, we denote by U the upper triangular matrix $A^{(n)}$. The entries $a_{kk}^{(k)}$ are called pivots and must obviously be non null for $k = 1,2,\dots,n-1$ we assume that $a_{kk}^{(k)} \neq 0$ and define the multiplier

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad i = k+1,\dots,n. \qquad (2.21)$$

Then we let

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)} \quad ,i,j = k+1,\dots,n.$$

$$b_i^{(k+1)} = b_i^{(k)} - m_{ik}b_k^{(k)}, \ \ i = k+1,\dots,n. \qquad (2.22)$$

Example 2.2 Let us use GEM to solve the following system

$$A^{(1)}x = b^{(1)} \begin{cases} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 = \frac{11}{16} \\ \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 = \frac{13}{12} \\ \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 = \frac{47}{60} \end{cases},$$

Which admits the solution $x = (1\ ,1\ ,1\ )^T$. At the first step we compute the multipliers $m_{21} = \frac{1}{2}$ and $m_{31} = \frac{1}{3}$, and subtract from the second and third equation of the system the first row multiplied by m_{21} and m_{31} respectively . we obtain the equivalent system.

$$A^{(2)}x = b^{(2)} \begin{cases} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 = \frac{11}{16} \\ 0 + \frac{1}{12}x_2 + \frac{1}{12}x_3 = \frac{1}{6} \\ 0 + \frac{1}{12}x_2 + \frac{4}{45}x_3 = \frac{31}{180} \end{cases} ,$$

If we subtract the second row multiplied by $m_{32} = 1$ form the third one , we end up with the upper triangular system

$$A^{(3)}x = b^{(3)} \begin{cases} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 = \frac{11}{16} \\ 0 + \frac{1}{12}x_2 + \frac{1}{12}x_3 = \frac{1}{6} \\ 0 + 0 + \frac{1}{180}x_3 = \frac{1}{180} \end{cases} ,$$

from which we immediately compute $x_3 = 1$ and then , by back substitution , the remaining unknowns $x_1 = x_2 = 1$.

Remark 2.2 The matrix in Example 2.1 is called the Hilbert matrix of order 3. In the general $n \times n$ case , its entries are

$h_{ij} = 1/(i + j - 1), \quad i, j = 1, \dots, n$. (2.23)

To complete Gaussian elimination $2(n-1)n\,(n+1)/3 + n(n-1)$ flops are required , plus n^2 flops to backsolve the triangular system $Ux = b^{(n)}$. Therefore, about $(2n^3/3 + 2n^2)$

Flops are needed to solve the liner system using GEM. Neglecting the lower order terms , we can state the Gaussian elimination process has a cost of $2n^3/3$ flops .

As previously noticed, GEM terminates safely iff the pivotal elements $a_{kk}^{(k)}$, for $k = 1, \dots, n-1$, are non vanishing . Unfortunately, having non null diagonal entries in A is not enough to prevent zero pivots to arise during the elimination process. For example , matrix A in (2.5) is nonsingular and has nonzero diagonal entries

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix}, A^{(2)} = \begin{bmatrix} 1 & 2 & 4 \\ 0 & 0 & -1 \\ 0 & -6 & -12 \end{bmatrix}. \qquad (2.24)$$

Nevertheless, when GEM is applied, it is interrupted at the second step since $a_{22}^{(2)} = 0$.

More restrictive conditions on A are needed to ensure the applicability of the method . leading dominating minors d_i of A are nonzero for $i = 1, \dots, n-1$, then the corresponding pivotal entries $a_{ii}^{(i)}$ must necessarily be non vanishing. We recall that d_i is the determinant of A_i , the $i-th$ principal submatrix made by the first i rows and columns of A . The matrix in the previous example does not satisfy this condition, having $d_1 = 1$ *and* $d_2 = 0$.

Classes of matrices exist such that GEM can be always safely employed in its basic form (2.22) .Among them, we recall the following ones:

1. Matrices diagonally dominant by rows.
2. Matrices diagonally dominant by columns. In such a case one can even show that the multipliers are in module less than or equal to 1(see Property 2.2).
3. Matrices symmetric and positive definite (see Theorem 2.6).

For a rigorous derivation of these results, we refer to the forthcoming sections.

2.2.1 GEM as Factorization Method

We show how GEM is equivalent to performing a factorization of

the matrix A into the product of two matrices , $A = LU$, with $U = A^{(n)}$. Since L and U depend only on A and not on the right hand side, the same factorization can be reused when solving several linear systems having the same matrix A but different right hand side b , with a considerable reduction of the operation count (indeed, the main computational effort, about $2n^3/3$ flops, is spent in the elimination procedure).

Let us go back to Example 3.2concerning the Hilbert matrix H_3. In practice, to pass from $A^{(1)} = H_3$ to the matrix $A^{(2)}$ at the second step, we have multiplied the system by the matrix

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -\frac{1}{3} & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -m_{21} & 1 & 0 \\ -m_{31} & 0 & 1 \end{bmatrix}.$$

Indeed,

$$M_1 A = M_1 A^{(1)} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & \frac{1}{12} & \frac{1}{12} \\ 0 & \frac{1}{12} & \frac{4}{45} \end{bmatrix} = A^{(2)} .$$

Similarly, to perform the second (and last) step of GEM, we multiply $A^{(2)}$ by the matrix

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -m_{32} & 1 \end{bmatrix}.$$

where $A^{(3)} = M_2 A^{(2)}$. Therefore

$$M_2 M_1 A = A^{(3)} = U . \qquad (2.25\ )$$

On the other hand, matrices M_1 and M_2 are triangular, their product is still lower triangular, as is their inverse , thus from (2.25) one gets

$$A = (M_2 M_1)^{-1} U = LU,$$

which is the desired factorization of A .

This identity can be generalized as follows. Setting

$$m_k = (0, \dots, 0, m_{k+1,k}, \dots, m_{n,k})^T \in \mathbb{R}^n$$

and defining

$$M_k = \begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & & \vdots & \vdots & \vdots \\ 0 & & & 1 & 0 & 0 \\ 0 & & -m_{k+1,k} & 1 & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & m_{nk} & 0 & \dots & 1 \end{bmatrix} = I_n - m_k e_k^T$$

as the $k-th$ Gaussian transformation matrix, one out that

$$(M_k)_{ip} = \delta_{ip} - (m_k e_k^T)_{ip} = \delta_{ip} - m_{ik}\delta_{kp}, \quad i,p = 1,\dots,n.$$

On the other hand, from (3.3) we have that

$$a_{ij}^{(k+1)} = a_{ij}^{k} - m_{ik}\delta_{kk}a_{kj}^{(k)} = \sum_{p=1}^{n}(\delta_{ip} - m_{ik}\delta_{kp})a_{pj}^{(k)}, \qquad i,j = k+1,\dots,n\,.$$

or , equivalently ,

$$A^{k+1} = M_k A^{(k)}. \qquad (2.26)$$

As a consequence, at the end of the elimination process the matrices M_k with

$$k = 1, \dots n-1$$

and the matrix U have been generated such that

$$M_{n-1}M_{n-2}\dots M_1 A = U.$$

The matrices M_k are unit lower triangular with inverse given by

$$M_k^{-1} = 2I_n - M_k = I_n + m_k e_k^T\,, \qquad (2.27)$$

where $(m_i e_i^T\,)(m_J e_j^T)$ are equal to the null matrix if $i \neq j$. As a consequence

$$A = M_1^{-1}M_2^{-1}\dots M_{n-1}^{-1}U$$

$$=(I_n + m_2 e_1^T)(I_n + m_2 e_2^T)\dots(I_n + m_{n-1}e_{n-1}^T)U$$

$$= (I_n + \textstyle\sum_{i=1}^{n-1} m_i e_i^T\,)U$$

$$=\begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ m_{21} & 1 & & & \vdots \\ \vdots & m_{32} & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ m_{n1} & m_{n2} & \dots & m_{n,n-1} & 1 \end{bmatrix} U. \qquad (2.28)$$

Defining $L = (M_{n-1}M_{n-2} \dots M_1) = M_1^{-1} \dots M_{n-1}^{-1}$, it follows that

$A = LU.$

We notice that, due to (2.28) the subdiagonal entries of L are the multipliers m_{ik} produced by GEM, while the diagonal entries are equal to one .

Once the matrices L and U have been computed , solving the linear system consists only of solving successively the two triangular systems

$$Ly = b$$

$Ux = y.$

The computational cost of the factorization process is obviously the same as that required by GEM.

The following result establishes a link between the leading dominant minors of a matrix and its LU factorization induced by GEM.

Theorem 2.4 Let $A \in \mathbb{R}^{n \times n}$. The LU factorization of A with $l_{ii} = 1$ for $i = 1, \dots, n$ exists and is unique iff the principal submatrices A_i of A of order $i = 1, \dots, n-1$ are nonsingular .

Proof. The existence of the LU factorization can be proved following the steps of the GEM. Here we prefer to pursue an alternative approach , which allows for proving at the same time both existence and uniqueness and that will be used again in later sections.

Let us assume that the leading minors A_i of A are nonsingular for $i = 1, \dots, n-1$ and prove , by induction on i , that under this hypothesis the LU factorization of $A(= A_n)$ with $l_{ii} = 1$ for $i = 1, \dots, n$ exists and is unique .

The property is obviously true if $i = 1$. Assume therefore that there exists an unique LU factorization of A_{i-1} of the form $A_{i-1} = L^{(i-1)}U^{(i-1)}$ with $l_{kk}^{(i-1)} = 1$ for $k = 1, \dots, i-1$, and show that there exists an unique factorization also for A_i. We partition A_i by block matrices as

$$A_i = \begin{bmatrix} A_{i-1} & c \\ d^T & a_{ii} \end{bmatrix}$$

and look for a factorization of A_i of the form

$$A_i = L^{(i)}U^{(i)} = \begin{bmatrix} L^{(i-1)} & o \\ I^T & 1 \end{bmatrix} \begin{bmatrix} U^{(i-1)} & u \\ o^T & u_{ii} \end{bmatrix}, \qquad (2.29)$$

having also partitioned by blocks factors $L^{(i)}$ and $U^{(i)}$. Computing the product of these two factors and equating by blocks the elements of A_i , it turns out that the vectors I *and* U are the solutions to the linear system $L^{(i-1)}u = c$, $I^T U^{(i-1)} = d^T$.

On the other hand, since $0 \neq \det(A_{i-1}) = \det(L^{(i-1)}) \det(U^{(i-1)})$, the matrices $L^{(i-1)}$ and $U^{(i-1)}$ are nonsingular and, as a result, U and I exist and are unique .

Thus, there exists a unique factorization of A_i, where u_{ii} is the unique solution of the equation $u_{ii} = a_{ii} - I^T u$. This completes the induction step of the proof.

It now remains to prove that, if the factorization at hand exists and is unique, then the first $n-1$ leading minors of the A must be nonsingular. we shall distinguish the case where A is singular and when it is nonsingular .

Let us start from the second one and assume that the LU factorization of A with $l_{ii=1}$ for $i = 1, \dots, n$ exists and is unique. Then, due to (2.10), we have $A_i = L^{(i)}U^{(i)}$ for $i = 1, \dots, n$. Thus

$$\det(A_i) = \det(L^{(i)}) \det(\mathrm{U}^{(i)}) = \det(\mathrm{U}^{(i)}) = \mathrm{u}_{11}\mathrm{u}_{22} \dots u_{ii}, \qquad (2.30)$$

form which, taking $i = n$ and A nonsingular, we obtain $u_{11}u_{22} \dots u_{nn} \neq 0$ and thus, necessarily , $\det(A_i) = u_{11}u_{22} \dots u_{ii} \neq 0$ for $i = 1, \dots, n-1$.

Now let A be a singular matrix and assume that (at least) one diagonal entry of U is equal to zero . Denote by u_{kk} the null entry of U with minimum index k. Thanks to (2.29), the factorization can be computed without troubles until the $k+1-th$ step. From that step on, since the matrix $U^{(k)}$ is singular, existence and uniqueness of vector I^T are certainly lost, and, thus, the same holds for the uniqueness of the factorization. In order for this not to occur before the process has factorized the whole matrix A , the u_{kk} entries must all be nonzero up to the index $k = n-1$ included, and thus due to (2.30) all the leading minors A_k must be nonsingular for $k = 1, \dots, n-1$.

From the above theorem we conclude that, if A_i , with $i = 1, \dots, n-1$, is singular , then the factorization may either not exist or not be unique.

2-3 The LDM^T factorization.

It is possible to devise other types of factorization of A removing the hypothesis that the elements of L are equal to one . Specifically, we will address some variant where the factorization of A is of the form

$A = LDM^T$.

where L , M^T and D are lower triangular, upper triangular and diagonal matrices, respectively

After the construction of this factorization, the resolution of the system can be carried out solving first the lower triangular system $Ly = b$, then the diagonal one $Dz = y$, and finally the upper triangular system $M^T x = z$, with a cost of $n^2 + n$ flops . In the symmetric case, we obtain $M = L$ and the LDL^T factorization enjoys a property analogous to the one in Theorem 2.4 for the LU factorization . In particular, the following result holds .

Theorem 2.5 If all the principal minors of a matrix $A \in \mathbb{R}^{n\times n}$ are nonzero then there exist a unique diagonal matrix D , a unique unit lower triangular matrix L and a unique unit upper triangular matrix M^T , such that $A = LDM^T$.

Proof. By Theorem 2.1 we already know that there exists a unique LU factorization of A with $l_{ii} = 1\, for\;\; i = 1, \dots, n$. If we set the diagonal entries of D equal to u_{ii} (nonzero because U is nonsingular), then $A = LU = LD(D^T U)$. Upon defining $M^T = D^{-1}U,$ the existence of the LDM^T factorization follows , where $D^{-1}U$ is a unit upper of the uniqueness of the LDM^T factorization is a consequence of the uniqueness of the LU factorization .

The above proof shows that, since the diagonal entries of D coincide with those of U , we could compute L , M^T and D starting from the LU factorization of A . It suffices to compute M^T as $D^{-1}U$. Nevertheless ,this algorithm has the same cost as the standard LU factorization.

Likewise, it is also possible to compute the three matrices of the factorization by enforcing the identity $A = LDM^T$ entry by entry

2-4 Symmetric and Positive Definite Matrices: The Cholesky factorization

As already pointed out, the factorization LDM^T simplifies considerably when A is symmetric because in such a case $M = L,$ yielding the so- called LDM^T factorization. The computational cost halves, with respect to the LU factorization , to about $(n^3/3)$ flops.

As an example , the Hilbert matrix of order 3 admits the following LDL^T factorization

$$H_3 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{12} & 0 \\ 0 & 0 & \frac{1}{180} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

In the case that A is also positive definite, the diagonal entries of D in the LDL^T factorization are positive . Moreover, we have the following result .

Theorem 2.6 Let $A \in \mathbb{R}^{n\times n}$ be a symmetric and positive definite matrix. Then, there exists a unique upper triangular matrix H with positive diagonal entries such that

$$A = H^T H. \qquad (2.31)$$

This factorization is called cholesky factorization and entries of H^T can be computed as follows :$h_{11} = \sqrt{a_{11}}$ and, for $i = 2, \dots, n,$

$$h_{ij} = (a_{ij} - \sum_{k=1}^{j-1} h_{ik}h_{jk}) \Big/ h_{jj}\,, \qquad j = 1, \dots, i-1$$

$$h_{ij} = (a_{ij} - \textstyle\sum_{k=1}^{j-1} h_{ik}^2)^{1/2}. \qquad (2.32)$$

Proof. Let us prove the theorem proceeding by induction on the size i of the matrix (as done in Theorem 2.4), recalling that if $A_i \in \mathbb{R}^{i\times i}$ is symmetric positive definite, then all its principal submatrices enjoy the same property.

For $i = 1$ the result is obviously true. Thus, suppose that it holds for $i - 1$ and prove that it also holds for i. There exists an upper triangular matrix H_{i-1} such that $A_{i-1} = H_{i-1}^T H_{i-1}$. Let us partition A_i as

$$A_i = \begin{bmatrix} A_{i-1} & V \\ V^T & \alpha \end{bmatrix},$$

with $\alpha \in \mathbb{R}^{+}, V^T \in \mathbb{R}^{i-1}$ and look for a factorization of A_i of the form

$$A_i = H_i^T H_i = \begin{bmatrix} H_i^T & 0 \\ h^T & \beta \end{bmatrix} \begin{bmatrix} H_{i-1} & h \\ 0^T & \beta \end{bmatrix}.$$

Enforcing the equality with the entries of A_i yields the equation $H_{i-1}^T h = V$ and $h^T h + \beta^2 = \alpha$. The vector h is thus uniquely determined, since H_{i-1}^T is nonsingular. As for $\beta,$ due to properties of determinants

$$0 < \det(A_i) = \det(H_i^T) \det(H_i) = \beta^2 (\det(H_{i-1}))^2,$$

we can conclude that it must be a real number . As result , $\beta = \sqrt{\alpha - h^T h}$ is the desired diagonal entry and this concludes the inductive argument .

Let us now prove formulae (2.32). The fact that $h_{11} = \sqrt{a_{11}}$ is an immediate consequence of the induction argument for $i = 1$. In the case of generic i, relations (2.16) are the forward substitution formulae for the solution of the linear system $H_{i-1}^T h = V = (a_{1i}, a_{2i}, \dots, a_{i-1,i})^T$,

while formulae (2.16) state that $\beta = \sqrt{\alpha - h^T h}$, where $\alpha = a_{ii}$.

The algorithm which implements (2.32) requires about $(n^3/3)$ flops and it turns out to be stable with respect to the propagation of rounding errors . It can indeed be shown that the upper triangular matrix $\widetilde{H}$ is such that $\widetilde{H}^T \widetilde{H} = A + \delta A,$ where δA is perturbation matrix such that $\|\delta A\|_2 \leq 8n(n+1)u \leq 1 - (n+1)u$ (see [24]).

Also, for the Cholesky factorization it is possible to overwrite the matrix H^T in the lower triangular portion of A , without any further memory storage . By doing so, A and factorization are preserved, noting that A is stored in the upper triangular section since it is symmetric and its diagonal entries can be computed as $a_{11} = h_{11}^2$, $a_{ii} = h_{ii}^2 + \sum_{k=1}^{i-1} h_{ik}^2$, $i = 2, \dots, n$.

An example of implementation of the Cholesky factorization is coded in the Program

Program 1 -chol2: the Cholesky factorization

```
$functin\ [A] = chol2$
$[n, n] = size(A);$
$for\ k = 1{:}n - 1$
    $A(k, k) = sqrt(A(k, k);\ \ A(k + 1{:}n, k) = A(k + 1{:}n, k)/A(k, k);$
    $for\ j = k + 1{:}n, \qquad A(j{:}n, j) = A(j{:}n, j) - A(j{:}n, k) * A(j, k);\ end$
end
$A(n, n) = sqrt\big(A(n, n)\big);$
```

2-5 Rectangular Matrices : QR factorization:

Definition(2.1) A matrix $A \in \mathbb{R}^{m\times n}$, with $m \geq n$, admits a QR factorization if there exist an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and an upper trapezoidal matrix $R \in \mathbb{R}^{m\times n}$ with null rows from the n+1-th one on, such that

$$A = QR \qquad (2.33)$$

This factorization can be constructed either using suitable transformation matrices (Givens or Householder matrices , see section(3.4.1) or using the Gram-Schmidt orthogonalization algorithm discussed below.

It is also possible to generate a reduced version of the QR factorization (2.33) , as stated in the following result .

Property 2.3 Let $A \in \mathbb{R}^{m\times n}$ be a matrix of rank n for which a QR factorization is known . Then there exists a unique factorization of A of the form

$$A = \tilde{Q}\tilde{R} \qquad (2.34)$$

where $\tilde{Q}$ and $\tilde{R}$ are submatrices of Q and R given respectively by

$$\tilde{Q} = Q(1{:}m, 1{:}n), \qquad \tilde{R} = R(1{:}n, 1{:}n) \qquad (2.35)$$

Moreover , $\tilde{Q}$ has orthonormal vector columns and $\tilde{R}$ is upper triangular and coincides with the Cholesky factor H of the symmetric positive definite matrix $A^T A = \tilde{R}^T \tilde{R}$.

If A has rank n (i.e., full rank),then the column vector of $\tilde{Q}$ form an orthonormal basis for the vector space range (A) $(range(A) = \{y \in \mathbb{R}^n : y = Ax \ for \ x \in \mathbb{R}^n)$. As a consequence , constructing the QR factorization can also be interpreted as a procedure for generating an orthonormal basis for a given set of vectors.

If A has rank $r < n$, the QR factorization does not necessarily yield an orthonormal basis for rang (A) . However , one can obtain a factorization of the form

$$Q^T AP = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix},$$

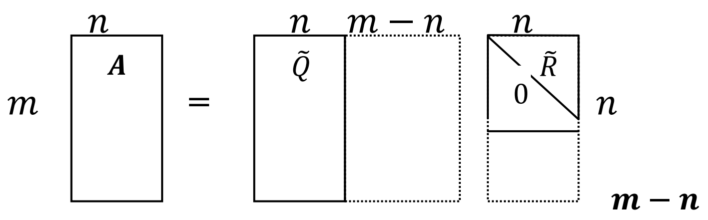

FIGURE 2.1 The reduced factorization. The matrices of the QR factorization are drawn in dashed lines

where Q is orthogonal , P is a permutation matrix and R_{11} is a nonsingular upper triangular matrix of order r .

In general, when using the QR factorization , we shall always refer to its reduced form (2.34) as it finds a remarkable application in the solution of overdetermined system.

The matrix factor $\tilde{Q}$ and $\tilde{R}$ in (2.34) can be computed using the Gramschmidt orthogonalization . Starting from a set of linearly independent vectors , $x_1, \dots, x_n$, this algorithm generates a new set of mutually orthogonal vector , $q_1, \dots, q_n$, given by

$q_1 = x_1$,

$$q_{k+1} = x_{k+1} - \sum_{i=1}^{k} \frac{(q_i, x_{k+1})}{(q_i, q_i)} q_i , \quad k = 1, \dots, n-1 . \qquad (2.36)$$

Denoting by $a_1, \dots, a_n$ the column vectors of A , we set $\tilde{q}_1 = a_1 / \|a_1\|_2$ and , for

$k = 1, \dots, n-1$, compute the column vectors of $\tilde{Q}$ as

$$\tilde{q}_{k+1} = q_{k+1} / \|q_{k+1}\|_2 ,$$

where

$$q_{k+1} = a_{k+1} - \sum_{j=1}^{k} (\tilde{q}_j, a_{k+1}) \tilde{q}_j .$$

Next, imposing that $A = \tilde{Q}\tilde{R}$ and exploiting the fact that $\tilde{Q}$ is orthogonal (that is $\tilde{Q}^{-1} = \tilde{Q}^T$), the entries of $\tilde{R}$ can easily be computed . The overall computational cost of the algorithm is of the order of mn^2 flops.

It is also worth noting that if A has full rank , the matrix A^TA is symmetric and positive definite (see Section 1.6) and thus it admits a unique Chollesky factorization of the form H^TH . On the other hand , since the orthogonality of $\tilde{Q}$ implies

$$H^TH = A^TA = \tilde{\mathrm{R}}^{\mathrm{T}}\tilde{\mathrm{Q}}^{\mathrm{T}}\tilde{\mathrm{Q}}\tilde{\mathrm{R}} = \tilde{R}^{\mathrm{T}}\tilde{\mathrm{R}} ,$$

we conclude that $\tilde{R}$ is actually the Cholesky factor A of A^TA . Thus , the diagonal entrie

of $\tilde{R}$ are all nonzero only if A has full rank .

The Gram- Schmidt method is of little practical use since the generated vectors lose their linear independence due to rounding errors . Indeed , in floating-point arithmetic the algorithm produces very small values of $\|q_{k+1}\|_2$

and $\tilde{r}_{kk}$ with a consequent numerical instability and loss of orthogonlity for matrix $\tilde{Q}$ (see Example 2.3) .

These drawbacks suggest employing a more stable version, known as modified Gram-Schmidt method . At the beginning of the $k+1$-th step, the projection of the vectors a_{k+1} along the vectors $\tilde{q}_1, \dots, \tilde{q}_k$ are progressively subtracted form a_{k+1} .On the resulting vector, the orthogonalization step is then carried out. In practice , after computing ($\tilde{q}_1, \dots, \tilde{q}_{k+1})\tilde{q}_1$

at the $k+1$-th step, this vector is immediately subtracted from a_{k+1} . As an example , one lest

$$a_{k+1}^{(1)} = a_{k+1} - (\tilde{q}_1, a_{k+1})\tilde{q}_1 \ .$$

This new vector $a_{k+1}^{(1)}$ is projected along the direction of $\tilde{q}_2$ and the obtained projection is subtracted from $a_{k+1}^{(1)}$ yielding

$$a_{k+1}^{(2)} = a_{k+1}^{(1)} - (\tilde{q}_2, a_{k+1}^{(1)})\tilde{q}_2$$

and so on , until $a_{k+1}^{(k)}$ is computed .

It can be checked that $a_{k+1}^{(k)}$ coincides with the corresponding vector q_{k+1} in the standard Gram-Schmidt process , since due to the orthogonality of vectors $\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_k$,

$$a_{k+1}^{(k)} = a_{k+1} - (\tilde{q}_1, a_{k+1})\tilde{q}_1 - (\tilde{q}_2, a_{k+1} - (\tilde{q}_1, a_{k+1})\tilde{q}_1)\tilde{q}_2 + \cdots$$

$$= a_{k+1} - \sum_{j=1}^{k} (\tilde{q}_j, a_{k+1})\tilde{q}_j.$$

Program 2 implements the modified Gram-Schmidt method .Notice that it is not possible to overwrite the computed QR factorization on the matrix A. In general, the matrix $\tilde{R}$ is overwrite on A , whilst $\tilde{Q}$ is stored separately. The computational cost of the modified Gram-Schmidt has the order of $2mn^2$ flops.

program 2-mod_grams: Modified Gram-Schmidt method:

```
 $function\ [Q,R] = mod\_grams(A)$
$[m,n] = size(A)$ ;
$Q = zeros(m,n)$;   Q(1: m, 1) = A(1: m, 1);  R = zeros(n);  R(1,1) = 1;
$for\ k = 1{:}n$
R(k,k)=norm(A(1:m,k));      Q(1:m,k)=A(1:m,k)/R(k,k);
for j=k+1:n
R(k,j)=Q(1:m,j)*A(1:m,j);
A(1:m,j)=A(1:m,j)-Q(1:m,k) *R(k, j) ;
end
end
```

Example 2.3 Let us consider the Hilbert matrix H_4 of order 4 (see (2.23)).The matrix $\tilde{Q}$, generated by the standard Gram-Schmidt algorithm , is orthogonal up to the order of 10^{-10} ,being

$$I - \tilde{Q}^T\tilde{Q} = 10^{-10}\begin{bmatrix} 0.0000 & -0.0000 & 0.0001 & -0.0041 \\ -0.0000 & 0 & 0.0004 & -0.0099 \\ 0.0001 & -0.0099 & -0.4785 & 0 \end{bmatrix}$$

and $\left\|I - \tilde{Q}^T\tilde{Q}\right\|_\infty = 4.9247.10^{-11}$. Using the modified Gram-Schmidt method , we would obtain

$$I - \tilde{Q}^T\tilde{Q} = 10^{-12}\begin{bmatrix} 0.0001 & -0.0005 & 0.0069 & -0.2853 \\ -0.0005 & 0 & 0.0023 & -0.0213 \\ -0.2853 & 0.0213 & -0.0103 & 0 \end{bmatrix}$$

and this time $\left\|I - \tilde{Q}^T\tilde{Q}\right\|_\infty$ =301686. 10^{-13} .

An improved result can be obtained using, instead of program 2, the intrinsic function QR of MATLAB. This function can be properly employed to generate both the factorization (2.33) as well as its reduced version (2.34).

2-6 Pivoting:

As previously out , the GEM process breaks down as soon as zero pivotal entry is computed. In such an event, one needs to resort to the so called pivoting technique which amounts to exchanging rows (or columns) of the system in such a way that non vanishing pivots are obtained.

Example 2.4 Let us go back to matrix (2.24)for which GEM furnishes at the second step a zero pivotal element. By simply exchanging the second row with the third one, we can execute one step further of the elimination method , finding a nonzero pivot . The generated system is equivalent to the original one and it can be noticed that it is already in upper triangular form. Indeed

$$A^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -6 & -12 \\ 0 & 0 & -1 \end{bmatrix} = U$$

while the transformation matrices are given by

$$M^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -7 & 0 & 1 \end{bmatrix}, \quad M^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

From algebraic standpoint , a permutation of the rows of A has been performed. In fact, it now no longer holds that $A \overset{\square}{=} M_1^{-1} M_2^{-1} U$, but rather $A = M_1^{-1}\, P\, M_2^{-1} U$, P being the permutation matrix

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad (2.37)$$

The pivoting strategy adopted in Example 3.4 can be generalization by looking , at each step k of the elimination procedure, for a nonzero pivotal entry by searching within the entries of subcolumn $A^{(k)}(k:n,k)$. For that reason, it is called partial pivoting (by rows) .

From (2.21)it can be seen that a large value of m_{ik} (generated for example by a small value of the pivot $a_{kk}^{(k)}$ might amplify the rounding errors affecting the entries $a_{kj}^{(k)}$. Therefore, in order to ensure a better stability , the pivotal element is chosen as the largest entry (in module) of the column $A^{(K)}(k:n,k)$ and partial pivoting is generally performed at every step of the elimination procedure, even if not strictly necessary (that is , even if nonzero pivotal entries are found).

Alternatively, the searching process could have been extended to the whole submatrix $A^{(K)}(k:n,k:n)$, ending up with a complete pivoting (see Figure 2.2). Notice , however , that while partial pivoting requires an additional cost of about n^2 searches , complete pivoting need about $2n^3/3$, with a considerable increase of the computational cost of GEM.

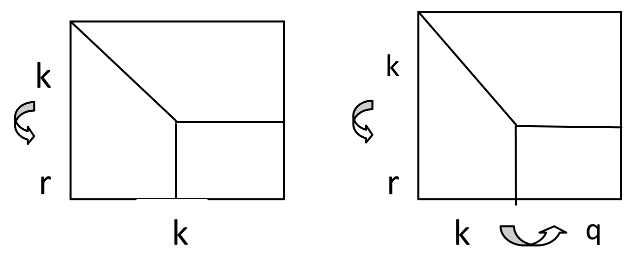

FIGURE 2.2. Partial pivoting by row (left) or complete pivoting (right). Shaded areas of the matrix are those involved in the searching for the pivotal entry

Example 2.5 Let us consider the linear system $Ax = b$ with

$$A = \begin{bmatrix} 10^{-13} & 1 \\ 1 & 1 \end{bmatrix}$$

and where b is chosen in such a way that $x = (1,1)^T$ is the exact solution. Suppose we use base 2 and 16 significant digits. GEM without pivoting would give $x_{MEG} = (0.999200722162464,1)^T$, while GEM plus partial pivoting furnishes the exact solution up to the 16^{th} digit.

Let us analyze how partial pivoting affects the LU factorization induced by GEM. At the first step of GEM with partial pivoting , after finding out the entry a_{r1} of maximum module in the first column , the elementary permutation matrix P_1 which exchanges the first row with r-th row is constructed(if r =1, P_1 is identity matrix). Next , the firs Gaussian transformation matrix M_1 is generated and we set $A^{(2)} = M_1 P_1 A^{(1)}. A$ similar approach is now taken on $A^{(2)}$, searching for a new permutation matrix P_2 and a new matrix M_2 such that

$$A^{(3)} = M_2 P_2 A^{(2)} = M_2 P_2 M_1 P_1 A^{(1)} .$$

Executing all the elimination steps, the resulting upper triangular matrix U is now given by

$$U = A^{(N)} = M_{n-1} P_{n-1} \dots M_1 P_1 A^{(1)}. \qquad (2.38)$$

Letting $M = M_{n-1} P_{n-1} \dots M_1 P_1$ and $P = P_{n-1} \dots p_1$, we obtain that $U = MA$ and , thus, $U = (MP^{-1})PA$. It can easily be checked that the matrix $L = PM^{-1}$ is unit lower triangular , so that the LU factorization reads

$$PA = LU. \qquad (2.39)$$

$$M^{-1} = P_1^{-1} M_1^{-1} \dots P_{n-1}^{-1} M_{n-1}^{-1} \text{ and } P_i^{-1} = P_i^T \text{ while } M_i^{-1} = 2I_n - M_i.$$

Once L, U and P are available solving the initial linear system amounts to solving the triangular systems $Ly = Pb \ \ and \quad Ux = y$. Notice that the entries of the matrix L coincide with the multipliers computed by LU factorization , without pivoting, when applied to the matrix PA.

If complete pivoting is performed , at the first step of the process , once the element a_{qr} of largest module in submatrix $A(1{:}n, 1{:}n)$ has been found , we must exchange the first row and column with the $q-th$ row and the permutation matrices by rows and by columns, respectively.

As a consequence, the action of matrix M_1 is now such that $a^{(2)} = M_1 P_1 A^{(1)} Q_1$. Repeating the process, at the last step, instead of (2.22) we obtain

$$U = A^{(n)} = M_{n-1} P_{n-1} \dots M_1 P_1 A^{(1)} Q_1 \dots Q_{N-1}.$$

In the case of complete pivoting the LU factorization becomes

$$PAQ = LU,$$

where $Q = Q_1 \dots Q_{n-1}$ is permutation matrix accounting for all permutations that have been operated . By construction, matrix L is still lower triangular, with module entries less than or equal to 1 .As happens in partial pivoting, the entries of L are the multipliers produced by the LU factorization process without pivoting , when applied to the matrix PAQ.

Program 3 is an implementation of the LU factorization with complete pivoting . For an efficient computer implementation of the LU factorization with partial pivoting, we refer to the MALAB intrinsic function LU.

Program 3 –LU pivtot : LU factorization with complete pivotin:

```
function [L,U,P,Q]=LU pivtot(A,n)
P=eye(n) ; Q=P;Minv=P;
for k=1:n-1
[PK,QK]=pivot(A,K,n) ;    A=PK*A*QK;
[MK,MKinv]=MGauss(A,K,n) ;
A=MK*A;    P=PK*K;       Q=Q*QK;
Minv*PK*MKinv;
end
U=triu(A) ;      L=P*Minv;
function[MK,MKinv]=MGauss(A,K,n)
```

```
MK=eye(n) ;
fori=k+1:n,    MK(i,K)=-A(I,k)/A(K,K) ; end
Mkinv=2*eye(n)-Mk;
function [PK,QK]=pivot(A,k,n)
[y,i]=max(abs(A(k:n,k:n))) ;   [piv,jpiv]=max(y) ;
ipiv=i(jpiv) ;    jpiv=jpiv+k-1;    ipiv=ipiv+k-1;
pk=eye(n) ;  pk(ipiv,ipiv)=0;   pk(k,k)=0;   pk(k,ipiv)=1;    pk(ipiv,k)=1;
                    jpiv.jpiv)=0;  Qk(k,k)=0; Qk(k,jipv)=1;  Qk(jpiv,k)=1;(Qk=eye(n)
;   Qk
```

Remark 2.3 The presence of large pivotal entries is not in itself sufficient to guarantee accurate solution, as demonstrated by the following example (taken from [25] .For the linear system $Ax = b$

$$\begin{bmatrix} -4000 & 2000 & 2000 \\ 2000 & 0.78125 & 0 \\ 2000 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 400 \\ 1.3816 \\ 1.9273 \end{bmatrix}$$

at the first step the pivotal entry coincides with the diagonal entry -400 itself . However , executing GEM on such a matrix yields the solution

$$\hat{x} = [0.00096365, -0.698496, 0.90042329]^T$$

whose first component drastically differs from that of exact solution

$x = [1.9273, -0.698496, 0.9004233]^T$. The cause of this behaviour should be ascribed to the wide variation among the system coefficients. Such cases can be remedied by a suitable scaling of the matrix.

Remark 2.4 (**pivoting for symmetric matrices)** As already noticed, pivoting is not strictly necessary if A is symmetric and positive definite. A separate comment is deserved when A is symmetric but not positive definite, since pivoting could destroy the symmetry of the matrix . This can be avoided by employing a complete pivoting of the form PAP^T , even though this pivoting can only turn out into a reordering of the diagonal entries of A . As a consequence, the presence on the diagonal of A of small entries might inhibit the advantages of the pivoting .To del with matrices of this kind, special algorithms are needed (like the Parlett-Reid method [26] or the Aasen method [27] for whose description we refer to [28], and to [29] for the case of sparse matrices.

2-7 Computing the Inverse of a Matrix:

The explicit computation of the inverse of matrix can be carried out using the UL factorization as follows. Denoting by X the inverse of a nonsingular matrix $A \in \mathbb{R}^{n\times n}$, the column vectors of X are the solutions to linear systems $Ax_i = e_i$, for $i = 1,..,n$.

Supposing that $PA = LU$, where P is partial pivoting permutation matrix, we must solve $2n$ triangular systems of the form

$$Ly_i = Pe_i, \quad Ux_i = y_i, \quad i = 1,\dots,n,$$

i.e., a succession of linear systems having the same coefficient matrix but different right hand sides. The computation of the inverse of matrix is a costly procedure which can sometimes be even less stable than MEG.

An alternative approach for computing the inverse of A is provided by the faddev or leverrier formula, which, letting $B_0 = I$, recursively computes

$$\alpha_k = \frac{1}{k} tr(AB_{k-1}), \quad \mathrm{B_k} = -AB_{k-1} + \alpha_k I, \quad k = 1,2,\dots,n.$$

Since $B_n = 0$, if $\alpha_n \neq 0$ we get

$$A^{-1} = \frac{1}{\alpha_n} B_{n-1,}$$

and the computational cost of the method for a full matrix is equal to $(n-1)n^3$ flops (for further details [30],[31]).

2-8 undetermined systems:

We have seen that the solution of the linear system $Ax = b$ exists and is unique if $n = m$ and A is nonsingular. In this section we give a meaning to the solution of a linear system both in the overdetermined case, where $m > n$, and in the underdetermined case, corresponding to $m < n$. We notice that an underdetermined system generally has no solution unless the right side b is an element of range (A)

For a detailed presentation, we refer to [32], [22]and [33].

Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n, b \in \mathbb{R}^m$, we say that $x^* \in \mathbb{R}^n$ is a solution of the linear system $Ax = b$ in the least-square sense if

$$\Phi(x) = \|Ax^* - B\|_2^2 \leq \min_{x\in\mathbb{R}^n}\|Ax - b\|_2^2 = \min_{x\in\mathbb{R}^n} \Phi(x). \qquad (2.40)$$

The problem thus consists of minimizing the Euclidean norm of the residual. The solution of (2.40) can be found by imposing the condition that the gradient of the function Φ in (2.40) must be equal to zero at x^*. From

$$\Phi(x) = (Ax - b)^T(Ax - b) = \mathrm{x}^\mathrm{T}\mathrm{A}^\mathrm{T}Ax - 2\mathrm{x}^\mathrm{T}\mathrm{A}^\mathrm{T}\mathrm{b} + \mathrm{b}^\mathrm{T}\mathrm{b},$$

we find that

$$\nabla\Phi(x^*) = 2\mathrm{A}^\mathrm{T}Ax^* - 2\mathrm{A}^\mathrm{T}\mathrm{b} = 0,$$

from which it follows that x^* must be the solution of the square system

$$\mathrm{A}^\mathrm{T}Ax^* = \mathrm{A}^\mathrm{T}\mathrm{b} \qquad (2.41)$$

known as the system of normal equation. The system is nonsingular if A has full rank and in such a case the least- squares solution exists and is unique. We notice that $\mathrm{B} = \mathrm{A}^\mathrm{T}A$ is a symmetric and positive definite matrix. Thus, in order to solve the normal equations, one could first compute the cholesky factorization $\mathrm{B} = \mathrm{H}^\mathrm{T}H$ and then solve the two systems $\mathrm{H}^\mathrm{T}y = \mathrm{A}^\mathrm{T}b$ and $Hx^* = y$. However, due to roundoff errors, the computation of $\mathrm{A}^\mathrm{T}A$ may be affected by a loss of significant digits, with a consequent loss of positive definiteness or nonsingularity of the matrix, as happens in the following example(implemented in MATLAB) where for a matrix A with full rank, the corresponding matrix $fl(A^T A)$ turns to be singular

$$A = \begin{bmatrix} 1 & 1 \\ 2^{-27} & 0 \\ 0 & 2^{-27} \end{bmatrix}, \qquad fl(A^T A) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Therefore, in the case of ill-conditioned matrices it is more convenient to utilize the QR factorization introduced in Section 2.5. Indeed, the following result holds.

Theorem 2.7 let $A \in \mathbb{R}^{m\times n}$, with $m \geq n$, be a full rank matrix. Then the unique solution of (3.40) is given by

$$x^* = \tilde{R}^{-1}\tilde{Q}^T b \qquad (2.42)$$

where $\tilde{R} \in \mathbb{R}^{n\times n}$ and $\tilde{Q} \in \mathbb{R}^{m\times n}$ are the matrices defined in (2.35) starting from the QR factorization of A . Moreover, the minimum of Φ is given by

$$\Phi(x^*) = \sum_{i=n+1}^{m} [(Q^T b)_i]^2.$$

Proof. The QR factorization of A exists and is unique since A has full rank. Thus, there exists two matrices, $Q \in \mathbb{R}^{m\times m}$ and $R \in \mathbb{R}^{m\times n}$ such that $A = QR,$ where Q is orthogonal. Since orthogonal matrices preserve the Euclidean scalar product it follows that

$$||Ax - b||_2^2 = ||Rx - Q^T b||_2^2 .$$

Recalling that R is upper trapezoidal, we have

$$||Rx - Q^T b||_2^2 = \left\|\tilde{R}x - \tilde{Q}^T b\right\|_2^2 + \sum_{i=n+1}^{m} [(Q^T b)_i]^2.$$

so that the minimum is achieved when $x = x^*$.

is x^* If A does not have full rank, the solution techniques above fail, since in this case if a solution to (2.40), the vector $x^* + z$, with $z \in \ker(A)$, is a solution too. We must therefore introduce a further constraint to enforce the uniqueness of solution. Typically , one requires that x^* has minimal Euclidean norm, so that the least-squares problem can be formulated as find $x^* \in \mathbb{R}^n$ with minimal Euclidean norm such that

$$||Ax^* - b||_2^2 \leq \min_{x \in \mathbb{R}^n} ||Ax - b||_2^2. \qquad (2.43)$$

This problem is consistent with (2.40) if A has full rank, since in this case (2.40) has a unique solution which necessarily must have minimal Euclidean norm.

The tool for solving (2.43) is singular value decomposition (or SVD, see Section 1.6), for which the following theorem holds.

Theorem (2.8) Let $A \in \mathbb{R}^{m \times n}$ with SVD given by $A = U \sum V^T$. Then the unique solution to (2.43) is

$$x^* = A^{\dagger} b \qquad (2.44)$$

where $A^{\dagger}$ is the pseudo- inverse of A introduced in Definition (1.11).

As for the stability of problem (2.43), we point out that if the matrix A does not have full rank, the solution x^* is not necessarily a continuous function of the data, so that small changes on these latter might produce large variation in x^*.

In the case of underdetermined systems, for which $m < n$, if A has full rank the QR factorization can be used. In particular, when applied to the transpose matrix A^T , the method yields the solution of minimal Euclidean norm. If, instead, the matrix has not full rank, one must resort to SVD.

Remark 2.5 If $m = n$ (square system), both SVD and QRfactorization can be used to solve the linear system $Ax = b$, as alternatives to GEM. Even though these algorithms require a number of flops far superior to GEM (SVD, for instance, requires $12n^3$ flops), they turn out to be more accurate when the system is ill-conditioned and nearly singular.

Chapter Three

Approximation of Eigenvalues and Eigenvectors

3.1 Introduction:

We deal with approximation of the eigenvalues and eigenvectors of a matrix $A \in \mathbb{C}^{n\times n}$ .Two main classes of numerical methods exist to this purpose, partial methods, which compute the extremal eigenvalues of A (that is, those having maximum and minimum module), or global methods, which approximate the whole spectrum of A .

It is worth noting that methods which are introduced to solve the matrix eigenvalue problem are not necessarily suitable for calculating the matrix eigenvectors . For example , the power method provides an approximation to particular eigenvalue/ eigenvector pair .

The QR method (a global method) instead computes the real Schur from of A , a canonical form that displays all the eigenvalues of A but not its eigenvectors . These eigenvectors can be computed, starting from the real Schur form of A , with an extra amount of work.

Finally , some ad hoc methods for dealing effectively with special case where A is a symmetric $(n \times n)$ matrix .

3-2 `Geometrical Location of the Eigenvalues:

Since the eigenvalues of A are the roots of the characteristic polynomial $p_A(\lambda)$(see Section 1-4) , iterative methods must be used for their approximation when $n \geq 5$. Knowledge of eigenvalue location in complex plane can thus be helpful in accelerating the convergence of the process .

A first estimate is provided by Theorem 1.4

$$|\lambda| \leq \|A\|, \qquad \forall \lambda \in \sigma(A) \qquad (3.1)$$

for any consistent matrix norm $||.||$. Inequality (4.1), which is often quite rough, states that all eigenvalues of A are contained in a circle of radius $R_{||A||} = ||A||$ centered at the origin of the Gauss plane.

Theorem 3.1 If $A \in \mathbb{C}^{n\times n}$, let

$$H = (A + A^H)/2 \text{ and } iS = (A - A^H)/2$$

be the hermition and skew-hermition parts of A , respectively, i being the imaginary unit. For any $\lambda \in \sigma(A)$.

$$\lambda_{min}(H) \leq Re(\lambda) \leq \lambda_{max}(H), \quad \lambda_{min}(S) \leq Im(\lambda) \leq \lambda_{max}(S). \qquad (3.2)$$

Proof. From the definition of H and S it follows that $A = H + iS$. Let $u \in \mathbb{C}^n$, $||u||_2 = 1$, be the eigenvector associated with the eigenvalue λ; the Rayleigh quotient (introduced in Section 1.4)

$$\lambda = u^H A u = u^H H u + i u^H S u, \qquad (3.3)$$

Notice that both H and S are hermitian matrices, whiles iS is skew-hermitian.

Matrices H and S are thus unitarily similar to real diagonal matrix(see Section 2.4) and therefore their eigenvalues are real . In such case, (3.3) yields

$$Rel\,(\lambda) = u^H H u, \qquad Im(\lambda) = i u^H S u$$

form which (3.2) follows.

An a priori bound for the eigenvalues of A is given by the following result.

Theorem 3.2 (of the Gershgorin circles) Let $A \in \mathbb{C}^{n\times n}$. Then

$$\sigma(A) \subseteq S_R = \cup_{i=1}^{n} R_i, \qquad R_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| \}. \qquad (3.4)$$

The sets R_i are called Gershgorin circles.

Proof. Let us decompose A as $A = D + E$, where D is diagonal part of A , whilst $e_{ii} = 0$ for $i = 1, \ldots, n$. For $\lambda \in \sigma(A)$ (with $\lambda \neq a_{ii}$, $i = 1, \ldots, n$), let us introduce the matrix $B_\lambda = A - \lambda I = (D - \lambda I) + E$. Since B_λ is singular , there exists a non –null vector $x \in \mathbb{C}^n$ such that $B_\lambda x = 0$. This means that $((D - \lambda I) + E)x = 0$, that is passing to the $\|.\|_\infty$ norm,

$$x = -(D - \lambda I)^{-1}Ex, \quad \|x\|_\infty \leq \|(D - \lambda I)^{-1}\|_\infty \|x\|_\infty,$$

and thus

$$1 \leq \|(D - \lambda I)^{-1}\|_\infty = \sum_{j=1}^{n} \frac{|e_{kj}|}{|a_{kk} - \lambda|} = \sum_{\substack{j=1 \\ j \neq k}}^{n} \frac{|e_{kj}|}{|a_{kk} - \lambda|} \qquad (3.5)$$

for a certain $k,\ 1 \leq k \leq n$. Inequality (4.5) implies $\lambda \in R_k$ and thus (3.4).

The bounds (3.4)ensure that any eigenvalue of A lies within the union of the circles R_i. Moreover, since A and A^T share the same spectrum, Theorem 3.2 also holds in the form.

$$\sigma(A) \subseteq S_C = \bigcup_{j=1}^{n} C_j\ ,\ C_j = \{z \in \mathbb{C}: |z - a_{jj}| \leq \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}|\ \}. \quad (3.6)$$

The circles R_i in the complex plane are called row circles, and C_i column circles . The immediate consequence of (3.4) and (3.6) is the following.

property 3.1(First gershgorin theorem) For a given matrix $A \in \mathbb{C}^{n \times n}$,

$$\forall \lambda \in \sigma(A), \quad \lambda \in S_R \cap S_C\ . \qquad (3.7)$$

The following two location theorems can also be proved ([30], [31).

property 3.2(Second gershgorin theorem) Let

$$S_1 = \bigcup_{i=1}^{m} R_i \quad , S_2 = \bigcup_{i=1}^{n} R_i\ .$$

If $S_1 \cap S_2 = \emptyset$, then S_1 contains exactly m eigenvalues of A , each one being accounted for with its algebraic multiplicity , while the remaining eigenvalues are contained in S_2 .

Remark 3.1 Properties 3.1 and 3.2 do not exclude the possibility that there exist circles containing no eigenvalues .

Definition 3.1 A matrix $A \in \mathbb{C}^{n\times n}$ is called reducible if there exists a permutation matrix P such that

$$PAP^T = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix},$$

where B_{11} and B_{22} are square matrices , A is irreducible if it is not reducible.

To check if a matrix is reducible , the oriented graph of the matrix can be conveniently employed, that the oriented graph of a real matrix A is obtained by joining n points (called vertices of the graph) $P_1, \dots, P_n$ through a line oriented from P_i to P_j if the corresponding matrix entry $a_{ij} \neq 0$. An oriented graph is strong connected if for any pair of distinct vertices P_i and P_j there exists an oriented path from P_i to P_j . The following result holds ([32]).

Property 3.3 A matrix $A \in \mathbb{C}^{n\times n}$ is irreducible iff its oriented graph is strongly connected.

Property 3.4 (Third Gershgorin theorem) Let $A \in \mathbb{C}^{n\times n}$ be an irreducible matrix . An eigenvalue $\lambda \in \sigma(A)$ cannot lie on the boundary of S_R unless it belongs to the boundary of every circle R_i , for $i = 1, \dots, n$.

3-3 Power Method:

The power method is very good at approximating the extremal eigenvalues of the matrix, that is eigenvalues having largest and smallest module, denoted by λ_1 and λ_n respectively, as well their associated eigenvectors.

3-3-1 Approximation of the Eigenvalue of Largest Module:

Let $A \in \mathbb{C}^{n\times n}$ be a diagonalizable matrix and let $X \in \mathbb{C}^{n\times n}$ be the matrix of its eigenvector x_i for $i = 1, \dots, n$. Let us also suppose that eigenvalues of A are ordered as

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \dots \geq |\lambda_n|, \qquad (3.8)$$

where λ_1 has algebraic multiplicity equal to 1. Under these assumptions, λ_1 is called the dominant eigenvalue of matrix A .

Given an arbitrary initial vector $q^{(0)} \in \mathbb{C}^n$ of unit Euclidean norm, consider for $k = 1,2,\dots$ the following iteration based on the computation of powers of matrices, commonly known as the power method :

$$z^{(k)} = Aq^{(k-1)}$$

$$q^{(k)} = z^{(k)} / \left\| z^{(k)} \right\|_2 \qquad (3.9)$$

$$v^{(k)} = (q^{(k)})^H A q^{(k)}$$

Let us analyze the convergence properties of method (3.9). By induction on k one check that

$$q^{(k)} = \frac{A^{(k)} q^{(0)}}{\left\| A^{(k)} q^{(0)} \right\|_2}, \qquad k > 1. \qquad (3.10)$$

This relation explains the role played the power of A in the method. Because A is diagonalizable , its eigenvectors x_i form a basis of $\mathbb{C}^n$; it is thus possible to represent $q^{(0)}$ as

$$q^{(0)} = \sum_{i=1}^{n} \alpha_i x_i \quad , \qquad \alpha_i \in \mathbb{C}^n, \quad i = 1, \dots, n \qquad (3.11)$$

Moreover , since $Ax_i = \lambda_i x_i$, we have

$$A^{(k)} q^{(0)} = \alpha_1 \lambda_1^k \left(x_1 + \sum_{i=2}^{n} \frac{\alpha}{\alpha} \left(\frac{\lambda_i}{\lambda_1} \right)^k x_i \right), \; k = 1,2,\dots \quad (3.12)$$

Since $|\lambda_i / \lambda_1| < 1 \;\; for \;\; i = 2, \dots, n,$ as k increases the vector $A^{(k)} q^{(0)}$ (and thus also $q^{(k)}$, due to (3.10)) , tends to assume an increasingly singnificant component in the direction of the eigenvector x_j decrease. Using (3.10) and (3.12) , we get

$$q^{(k)} = \frac{\alpha_1 \lambda_1^k (x_1 + y^{(k)})}{\left\| \alpha_1 \lambda_1^k (x_1 + y^{(k)}) \right\|_2} = \mu_k \frac{x_1 + y^{(k)}}{\| x_1 + y^{(k)} \|_2},$$

where μ_k is the sign of $\alpha_1\lambda_1^k$ and $y^{(k)}$ denotes a vector that vanishes as $k \to \infty$.

As $k \to \infty$, the vector $q^{(k)}$ thus aligns it self along the direction of eigenvector x_1 , and the following error estimate holds at step k .

Theorem 3.4 Let $A \in \mathbb{C}^{n\times n}$ be a diagonalizable matrix whose eigenvalues satisfy (3.8). Assuming that $\alpha_1 \neq 0$, there exists a constant $C > 0$ such that

$$\left\|\tilde{q}^{(k)} - x_1\right\|_2 \leq C\left|\frac{\lambda_2}{\lambda_1}\right|^k, \quad k \geq 1, \qquad (3.13)$$

where

$$\tilde{q}^{(k)} = \frac{q^{(k)}\|A^{(k)}q^{(0)}\|_2}{\alpha_1\lambda_1^k} = x_1 + \sum_{i=2}^{n}\frac{\alpha_i}{\alpha_1}(\frac{\lambda_i}{\lambda_1})^k x_i\,, \qquad k = 1,2,\dots \qquad (3.14)$$

Proof. Since A is diagonalizable , without losing generality, we can pick up the nonsingular matrix X in such a way that is columns have unit Euclidean length, that is $\|x_i\|_2 = 1$ for $i = 1,\dots,n$. From (3.12) it thus follows that

$$\left\|x_1 + \sum_{i=2}^{n}\left[\frac{\alpha_i}{\alpha_1}(\frac{\lambda_i}{\lambda_1})^k x_i\right] - x_i\right\|_2 = \left\|\sum_{i=2}^{n}\frac{\alpha_i}{\alpha_1}(\frac{\lambda_i}{\lambda_1})^k x_i\right\|_2$$

$$\leq \left(\sum_{i=2}^{n}\left[\frac{\alpha_i}{\alpha_1}\right]^2(\frac{\lambda_i}{\lambda_1})^{2k}\right)^{1/2} \leq \left|\frac{\lambda_2}{\lambda_1}\right|^k\left(\sum_{i=2}^{n}\left[\frac{\alpha_i}{\alpha_1}\right]^2\right)^{1/2}$$

that is (3.13) with $C = (\sum_{i=2}^{n}(\alpha_i/\alpha_1)^2)^{1/2}$.

Estimate (3.13) expresses the convergence of the sequence $\tilde{q}^{(k)}$ towards x_1 .

Therefore the sequence of the Rayleigh quotients

$$((\tilde{q}^{(k)})^H A\tilde{q}^{(k)})/\|\tilde{q}^{(k)}\|_2^2 = (q^{(k)})^H Aq^{(k)} = v^k$$

will converge to λ_1 .As a consequence, $\lim_{k\to\infty} v^k = \lambda_1$ and the convergence will be faster when the ratio $|\lambda_2/\lambda_1|$ is smaller.

If the matrix A is real and symmetric it can be proved, always assuming that $\alpha_1 \neq 0$, that ([33])

$$\left|\lambda_1 - v^k\right| \leq |\lambda_1 - \lambda_n| tan^2(\theta_0) \left|\frac{\lambda_2}{\lambda_1}\right|^{2k}, \qquad (3.15)$$

where $\cos(\theta_0) = \left|x_1^T q^{(0)}\right| \neq 0$. Inequality (3.15) out lines that the convergence of the sequence v^k to λ_1 is quadratic with respect to the ratio $|\lambda_2/\lambda_1|$.

We conclude the section by proving a stopping criterion for the iteration (3.9). For this purpose, let us introduce the residual at step k

$$r^{(k)} = Aq^{(k)} - v^{(k)}q^{(k)} \quad , k \geq 1,$$

and, for $\varepsilon > 0$, the matrix $\varepsilon E^{(k)} = -r^{(k)}[q^{(k)}]^H \in \mathbb{C}^{n\times n}$ with $\left\|E^{(k)}\right\|_2 = 1$. since

$$\varepsilon E^{(k)} q^{(k)} = -r^{(k)}, \qquad k \geq 1, \qquad (3.16)$$

an eigenvalue of the perturbed matrix $A + \varepsilon E^{(k)}$.

From (3.16) it follows that $\varepsilon = \left\|r^{(k)}\right\|_2$ for $k = 1,2,\ldots$ .Plugging this identity back into $\left|\frac{\partial\lambda}{\partial\epsilon}(0)\right| \leq \frac{1}{|y^H x|}$ and approximating the partial derivative in it by the incremental ratio $\left|\lambda_1 - v^k\right|/\varepsilon$,we get

$$\left|\lambda_1 - v^k\right| \simeq \frac{\left\|r^{(k)}\right\|_2}{|\cos(\theta_\lambda)|}, \qquad k \geq 1 \qquad (3.17)$$

where θ_λ is the angle between the right and the left eigenvectors, x_1 and y_1 associated with λ_1. Notice that, if A is an hermitian matrix, then $\cos(\theta_\lambda) = 1$, so that (3.17)yields an estimate which is analogue to($\min_{\lambda_i\in\sigma(A)}\left|\hat{\lambda} - \lambda_i\right| \leq \frac{\|\hat{r}\|_2}{\|\hat{x}\|_2}$).

In practice, in order to employ the estimate (3.17) it is necessary at each step k to replace $|\cos(\theta_\lambda)|$ with the module of the scalar product between two approximations $q^{(k)}$ and $w^{(k)}$ of x_1 and y_1 , computed by the power method. The following a posteriori estimate is thus obtained

$$\left|\lambda_1 - v^k\right| \simeq \frac{\left\|r^{(k)}\right\|_2}{\left|(w^{(k)})^H q^{(k)}\right|}, \qquad k \geq 1 \qquad (3.18)$$

3.3.2 Inverse Iteration:

We look for an approximation of the eigenvalue of matrix $A \in \mathbb{C}^{n\times n}$ which is closest to a given number $\mu \in \mathbb{C}$, where $\mu \notin \sigma(A)$. For this, the power iteration (3.17) can be applied to the matrix $(M_\mu)^{-1} = (A-\mu I)^{-1}$, yielding the so-called inverse iteration or inverse power method. The number μ is called a shift.

The eigenvalues of $(M_\mu)^{-1}$ are $\xi_i = (\lambda_i - \mu)^{-1}$; let us assume that there exists an integer m such that

$$|\lambda_m - \mu| < |\lambda_i - \mu|, \quad \forall i = 1, \dots, n \text{ and } i \neq m. \qquad (3.19)$$

This amounts to requiring that the eigenvalue λ_m which is closest to μ has multiplication equal to 1. Moreover, (3.19) shown that ξ_m is the eigenvalue of $(M_\mu)^{-1}$with largest module; in particular, if $\mu = 0$, λ_m turns out to be the eigenvalue of A with smallest module.

Given an arbitrary initial vector $q^{(0)} \in \mathbb{C}^n$ of unit Euclidean norm, for $k = 1,2, \dots$ the following sequence is constructed :

$$(A-\mu I)z^{(k)} = q^{(k-1)}$$

$$q^{(k)} = z^{(k)}/\|z^{(k)}\|_2 \qquad (3.20)$$

$$\sigma^{(K)} = (q^{(k)})^H A q^{(k)}.$$

Notice that the eigenvectors of M_μ are the same as those of A since $M_\mu = X(\Lambda - \mu I_n)X^{-1}$, where $\Lambda = diag(\lambda_1, \dots, \lambda_n)$. For this reason, the Rayleigh quotient in(3,28) is computed directly on the matrix A (and not on $(M_\mu)^{-1}$).The main difference with respect to (3.9) is that at each step k a linear system with coefficient matrix $M_\mu = A - \mu I$ must be solved. For numerical convenience, the LU factorization of M_μ is computed once for all at $k = 1$, so that at each step only two triangular systems are to be solved, with a cost of the order of n^2 flops.

Although being more computationally expensive than the power method (3.9), the inverse iteration has the advantage that it can converge to any desired eigenvalue of A (namely, the one closest to the shift μ). Inverse iteration is thus ideally suited for refining an initial estimate μ of an eigenvalue of A, which can be obtained, for instance, by applying the localization techniques introduced in Section 3.1. Inverse iteration can be also effectively employed to compute the eigenvector associated with a given (approximate) eigenvalue, as described in Section 3.8.1.

In view of the convergence analysis of the iteration (3.20) we assume that A is diagonalizable, so that $q^{(0)}$ can be represented in the form (3.11). Proceeding in the same way as in the power method, we let

$$\tilde{q}^{(k)} = x_m + \sum_{i=1,i\neq m}^{n} \frac{\alpha_i}{\alpha_m} (\frac{\xi_i}{\xi_m})^k x_i,$$

where x_i are the eigenvector of $(M_\mu)^{-1}$(and thus also of A), while α_i are as in (3.11). As a consequence, recalling the definition of ξ_i and using (3.19), we get

$$\lim_{k\to\infty} \tilde{q}^{(k)} = x_m, \qquad \lim_{k\to\infty} \sigma^{(k)} = \lambda_m.$$

Convergence will be faster when μ is closer to λ_m. Under the same assumptions made for proving (3.18), the following a posteriori estimate can be obtained for the approximation error on λ_m

$$\left|\lambda_m - \sigma^{(k)}\right| \simeq \frac{\|\hat{r}^{(k)}\|_2}{\left|(\widehat{w}^{(k)})^H q^{(k)}\right|}, \qquad k \geq 1 \quad (3.20)$$

where $\hat{r}^{(k)} = Aq^{(k)} - \sigma^{(k)} q^{(k)}$ and $\widehat{w}^{(k)}$ is the $k-th$ iteration of the inverse power method to approximate the left eigenvector associated with λ_m.

3.3.3 Implementation Issues:

The convergence analysis shows that the effectiveness of the power method strongly depends on the dominant eigenvalues being well-separated (that is $|\lambda_1|/|\lambda_2| \ll 1$). Let us now analyze the behavior of iteration (3.9) when two dominant eigenvalues of equal module exist (that is, $|\lambda_2| = |\lambda_1|$). Three cases must be distinguished :

1. $\lambda_2 = \lambda_1$: the two dominant eigenvalues are coincident. The method is still convergent, since for k sufficiently large(3.12)yields

$$A^k q^{(0)} \simeq \lambda_1^k(\alpha_1 x_1 + \alpha_2 x_2)$$

which is an eigenvector of A .For $k \to \infty$, the sequence $\tilde{q}^{(k)}$ (after a suitable

redefinition) converges to a vector lying in the subspace spanned by the eigenvectors

x_1 and x_2 , while the sequence $v^{(k)}$ still converges to λ_1.

2. $\lambda_2 = -\lambda_1$: the two dominant eigenvalues are opposite. In this case the eigenvalue of largest module can be approximated by applying the power method to the matrix A^2. Indeed, for $i = 1, \dots, n, \lambda_i(A^2) = [\lambda_i(A)]^2$, so that $\lambda_1^2 = \lambda_2^2$ and the analysis falls into the previous case, where the matrix is now A^2.
3. $\lambda_2 = \bar{\lambda}_1$: the two dominant eigenvalues are complex conjugate. Here undamped oscillations arise in the sequence of vectors $q^{(k)}$ and the power method is not convergent ([24]).

As for the computer implementation of (3.9), it is worth noting that normalizing the vector $q^{(k)}$ to 1 keeps away from overflow (when $|\lambda_1| > 1$) or underflow(when $|\lambda_1| < 1$) in (3.12). We also point out that the requirement $\alpha_1 \neq 0$ (which is a priori impossible to fulfil when no information about the eigenvector x_1 is available) is not essential for the actual convergence of the algorithm .

Indeed, although it can be proved that, working in exact arithmetic, the sequence (3.9) converges to the pair $(\lambda_2, x_{2)}$ if $\alpha_1 = 0$, the arising of (unavoidable) rounding errors ensures that in practice the vector $q^{(k)}$ contains a non-null component also in the direction of x_1 .

This allows for the eigenvalue $\lambda_1$to "show-up" and the power method to quickly converge to it.

An implementation of the power method is given in Program 5. Here and in the following algorithm, the convergence check is based on the a posteriori estimate(3.18).

Here and in the remainder of the chapter, the input z0, tool and nmax are the initial vector, the tolerance for the stopping test and the maximum admissible number of iterations, respectively. In output, the vectors nu1 and err contain the sequences $\{v^{(k)}\}$ and $\left\{\|r^{(k)}\|_2/|\cos(\theta_\lambda)|\right\}$ (see(3.18)), whilst x_1 and niter are the approximation of the eigenvector x_1 and the number of iterations taken by the algorithm to converge, respectively.

<u>**Program 5 –powerm :power method:**</u>

```
function [nu1,x1,niter,err]=powerm(A,z0,toll,nmax)

q=z0/norm(z0);q2=q;err=[];nul=[];res=toll+1;nier=0;z=A*q;

while(res>=toll& niter<=nmax)

q=z/norm(z) ;z=A*q; lam=q*z;x1=q;

z2=q2`*A;q2=z2/norm(z2) ;q2=q2;

y1=q2;costheta=abs(y1`*x1);

if( costheta >=5e-2),
niter =niter+1;res=norm(z-lam*q)/costheta ;
err=[err,res];nul=[nul;lam];
else
disp (` Multiple eigenvalue `);break;
  end
end
```

A coding of the inverse power method is provided in Program 6. The input parameter mu is the initial approximation of the eigenvalue. In output, the vectors sigma and err contain the sequences $\{\sigma^{(k)}\}$ and $\left\{\|\hat{r}^{(k)}\|_2 / |(\hat{w}^{(k)})^H q^{(k)}|\right\}$ (see (3.21)). The LU factorization (with partial pivoting) of the matrix M_μ is carried out using the MATLAB intrinsic function 1u.

Program 6 – invpower :Inverse power method:

```
function [sigma,x,niter,err]=invpower(A,z0,mu,toll,nmax)
n=max(size(A)); M=A-mu*eye(n); [L,U,P]=lu(M);
q=z0/norm(z0); q2=q`; err=[]; sigma=[]; res=tool+1; niter=0;
while(res>=tool&niter<=nmax)
niter=niter+1; b=P*q; y=L/b; z=U/y;
q=z/norm(z); z=A*q; lam=q`*z;
b=q2; y=U`/b; w=L`/y;
q2=(P`*w)`; q2=q2/norm(q2); costheta=abs(q2*q);
if (costheta >=5e-2),
   res=norm(z-lam*q)/costheta ; err=[err; res]; sigma=[sigma; lam];
else
```

```
    disp(' Multiple eigenvalue '); break;
end
x=q;
end
```

3-4 The QR Iteration:

We present some iterative techniques for simultaneously approximating all the eigenvalues of a given matrix A .The basic idea consists of reducing A, by means of suitable similarity transformations, into a form for which the calculation of the eigenvalues is easier than on the starting matrix .

The problem would be satisfactorily solved if the unitary matrix U of the Schur decomposition theorem 1.4 , such that $T = U^H AU$. T being upper triangular and with $t_{ii} = \lambda_i(A)$ for $i = 1, \dots, n$ could be determined in direct way, that is , with a finite number of operations. Unfortunately , it is a consequence of Abel`s theorem that for $n \geq 5$ the matrix U cannot be computed in an elementary way. Thus, our problem can be solved only resorting to iterative techniques.

Let $A \in \mathbb{R}^{n\times n}$, given an orthogonal matrix $Q^{(0)} \in \mathbb{R}^{n\times n}$ and letting

$T^{(0)} = (Q^{(0)})^T AQ^{(0)}$ for $k = 1,2, \dots,$ until convergence , the QR iteration consists of :

determine $Q^{(k)}, R^{(k)}$ such that

$Q^{(k)}R^{(k)} = T^{(k-1)}$ (QR factorization), (3.22)

then , let

$$T^{(k)} = R^{(k)}Q^{(k)}$$

At each step $k \geq 1$, the first phase of the iteration is the factorization of the matrix $T^{(k-1)}$ into product of an orthogonal matrix $Q^{(k)}$ with an upper triangular matrix $R^{(k)}$ (see Section3.6.3) . The second phase is a simple matrix product. Notice that

$$T^{(k)} = R^{(k)}Q^{(k)} = (Q^{(k)})^T\big(Q^{(k)}R^{(k)}\big)Q^{(k)} = (Q^{(k)})^T T^{(k-1)}Q^{(k)}$$

$$= \big(Q^{(0)}Q^{(1)} \dots Q^{(k)}\big)^T A(\big(Q^{(0)}Q^{(1)} \dots Q^{(k)}\big), \quad k \geq 0 \qquad (3.23)$$

i.e., every matrix $T^{(k)}$ is orthogonally similar to A . This is particularly relevant for the stability of the method .

A basic implementation of the QR iteration (3.22), assuming $Q^{(0)} = I_n$, while a more computationally efficient version, starting from $T^{(0)}$ in upper Hessenberg form, is described in detail in Section 3.6.

If A has real eigenvalues, distinct in module, it will be seen in Section 3.5 that the limit of $T^{(k)}$ is an upper triangular matrix (with the eigenvalues of A on the main diagonal). However, if A has complex eigenvalues the limit of $T^{(k)}$ cannot be an upper triangular matrix T . Indeed if it were T would necessarily have real eigenvalues, although it is similar to A.

Failure to converge to triangular matrix may also happen in more general situations, as addressed in Example 3.1.

For this, it is necessary to introduce variants of the QR iteration (3.22), based on deflation and shift techniques (see Section 3.7 and, for a more detailed discussion of the subject, [18], ,[35]).

These techniques allow for $T^{(k)}$ to converge to an upper quasi-triangular matrix , known as the real Schur decomposition of A , for which the following result holds (for the proof we refer to [22]).

Property 3.5 Given a matrix $A \in \mathbb{R}^{n\times n}$,there exists an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ such that

$$Q^T A Q = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ 0 & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R_{mm} \end{bmatrix} \qquad (3.24)$$

where each block R_{ii} is either a real number or matrix of order 2 having complex conjugate eigenvalues, and

$$Q = \lim_{k\to\infty}[Q^{(0)}Q^{(1)} \dots Q^{(K)}] \qquad (3.25)$$

$Q^{(K)}$being the orthogonal matrix generated by the $k - th$ factorization step of the QR iteration (3.22).

The QR iteration can be also employed to compute all the eigenvectors of a given matrix. For this purpose, we describe in Section 3.8 two possible approaches, one based on the coupling between (3.22) and the inverse iteration (3.20), the other working on the real Schur form (3.24).

3-5 The Basic QR Iteration:

In the basic version of the QR method , one sets $Q^{(0)} = I_n$ in such a way that $T^{(0)} = A$. At each step $k \geq 1$ the QR factorization of the matrix $T^{(k-1)}$ can be carried out using the modified Gram-Schmidt procedure introduced 2.4with a cost of the order of $2n^3$ flops (for a full matrix A). The following convergence result holds ([22]).

property 3.6 (convergence of QR method) Let $A \in \mathbb{R}^{n\times n} A \in \mathbb{R}^{n\times n}$ be a matrix such that

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| \,.$$

Then

$$\lim_{k\to+\infty} T^{(k)} = \begin{bmatrix} \lambda_1 & t_{12} & \dots & t_{1n} \\ 0 & \lambda_2 & t_{23} & \dots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \qquad (3.26\ )$$

As for the convergence rate , we have

$$\left|t_{i,i-1}^{(k)}\right| = \varphi(\left|(\frac{\lambda_i}{\lambda_{i-1}})^k\right|) \qquad , i = 2,..,n, \quad \text{for } k \to +\infty \quad (3.27\ )$$

aim of accelerating it, one can resort to the so-called shift technique, which will be addressed in in Section 3.7.

Remark 3.2 It is always possible to reduce the matrix A into a triangular form by means of an iterative algorithm employing nonorthogonal similarity transformations .In such a case, the so-called LR iteration (known also as Rutishauser method, [2]) can be used, form which the QR method has actually been derived (see also[24]). The LR iteration is based on the factorization of the matrix A into the product of two matrices L and R , respectively unit lower triangular and upper triangular , and on the (nonorthogonal) similarity transformation

$L^{-1}AL = L^{-1}(LR)L = RL.$

The rare use of the LR method in practical computations is due to the loss of accuracy that can arise in the LR factorization because of the increase in module of the upper diagonal entries of R . This aspect, together with the details of the implementation of the algorithm and some comparisons with the QR method, is examined in [24].

Example 3.1 we apply the QR method to the symmetric matrix $A \in \mathbb{R}^{4\times4}$ such that $a_{ii} = 4,$ for $i = 1, \dots, 4,$ and $a_{ii} = 4 + i - j$ for $i < j \leq 4,$ whose eigenvalues are (to three significant figures) $\lambda_1 = 11.09$, $\lambda_2 = 3.41,$ $\lambda_3 = 0.90$ and $\lambda_4 = 0.59$. After 20 iterations, we get

$$T^{(20)} = \begin{bmatrix} 11.09 & 6.44\cdot 10^{-10} & -3.62\cdot 10^{-15} & 9.49\cdot 10^{-15} \\ 6.47\cdot 10^{-10} & 3.41 & 1.43\cdot 10^{-11} & 4.60\cdot 10^{-16} \\ 1.74\cdot 10^{-21} & 1.43\cdot 10^{-11} & 0.90 & 1.16\cdot 10^{-4} \\ 2.32\cdot 10^{-25} & 2.68\cdot 10^{-15} & 1.16\cdot 10^{-4} & 0.58 \end{bmatrix}$$

Notice the almost-diagonal structure of the matrix $T^{(20)}$ and, at the same time, the effect of rounding errors which slightly alter its expected symmetry. Good agreement can also be found between the under-diagonal entries and the estimate (3.27).

A computer implementation of the basic QR iteration is given in program 7. The QR factorization is executed using the modified Gram-Schmidt method (program2).The input parameter niter denotes the maximum admissible number of iterations, while the output parameters T, Q and R are the matrices T, Q and R in (3.22) after niter iteration of the QR procedure.

program 7 –basicqr :Basic QR iteration:

```
function [T,Q,R]=basicqr(A,niter)

T=A

fori=1:niter,

[Q,R]=mod –grams(T);

T=R*Q;

end
```

3-6The QR Method for matrices in Hessenberg Form:

The naïve implementation of the QR method discussed in the previous section requires (for full matrix) a computational effort of the order of n^3 flops per iteration. In this section we illustrate a variant for the QR iteration, known as Hessenberg –QR iteration, with a greatly reduced computational cost. The idea consists of starting the iteration from a matrix $T^{(0)}$ in upper Hessenberg form, that is $t_{ij}^{(0)} = 0$ for $i > j + 1$. Indeed ,it can be checked that with this choice the computation of $T^{(k)}$ in (3.22) requires only an order of n^2 flops per iteration.

To achieve maximum efficiency and stability of the algorithm, suitable transformation matrices are employed. Precisely , the preliminary reduction of a matrix A into upper Hessenberg form is realized with Householder matrices, whilst the QR factorization of $T^{(k)}$ is carried out using Givens matrices, instead of the modified Gram-Schmidt procedure introduced in Section 3.4.3.

We briefly describe Householder and Givens matrices in the next section, referring to Section 3.6.5 for their implementation. The algorithm and examples of computations of the real Schur form of A starting from its upper Hessenberg form are then discussed in Section 3.6.4.

3-6-1 Householder and Given Transformation Matrices:

For any vector $v \in \mathbb{R}^n$, let us introduce the orthogonal and symmetric matrix

$$P = I - 2vv^T/\|v\|_2^2 \qquad (3.28)$$

Given a vector $x \in \mathbb{R}^n$, the vector $y = Px$ is the reflection of x with respect to the hyperplane $\pi = span\{v\}^\perp$ formed by the set of the vectors that are orthogonal to v (see Figure 3.1,left). Matrix P and the vector v are called the Householder reflection matrix and the householder vector, respectively.

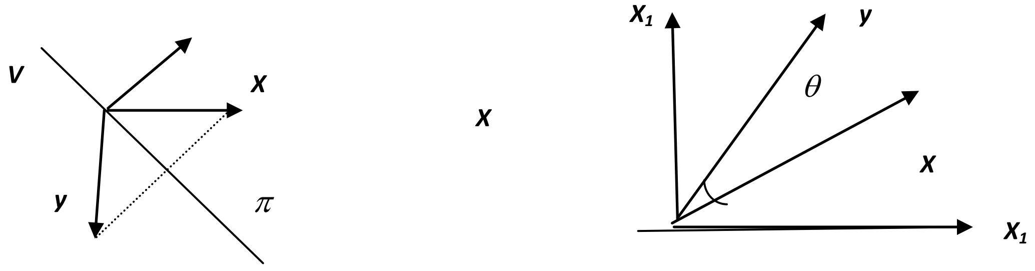

FIGURE 3.1. Reflection across the hyperplane orthogonal to v (left), rotation by an angle θ in the plane (x_i, x_k) (right)

Householder matrices can be used to set to zero a block of components of a given vector $x \in \mathbb{R}^n$. If, in particular, one would like to set to zero all the components of x , except the $m-th$ one , Householder vector ought to be chosen as

$$v = x \pm \|x\|_2 e_m, \qquad (3.29)$$

e_m being the $m-th$ unit vector of $\mathbb{R}^n$.The matrix P computed by (3.28) depends on the vector x itself, and it can be checked that

$$Px = [0,0,\dots,\underbrace{\pm\|x\|_2}_{m},0,\dots,0]^T \qquad (3.30)$$

Example 3.2 Let $x = [1,1,1,1]^T$ and $m = 3$, then

$$v = \begin{bmatrix} 1 \\ 1 \\ 3 \\ 1 \end{bmatrix} \quad , P = \frac{1}{6}\begin{bmatrix} 5 & -1 & -3 & -1 \\ -1 & 5 & -3 & -1 \\ -3 & -3 & -3 & -3 \\ -1 & -1 & -3 & 5 \end{bmatrix}, \quad Px = \begin{bmatrix} 0 \\ 0 \\ -2 \\ 0 \end{bmatrix}$$

If , for some $k \geq 1$, the first k components of x must remain unaltered , while the components from $k+2$ on are to be set to zero, the Householder matrix $P = P_{(k)}$ takes the following form

$$P_{(k)} = \begin{bmatrix} I_k & 0 \\ 0 & R_{n-k} \end{bmatrix}, \quad R_{n-k} = I_{n-k} - 2\frac{w^{(k)}(w^{(k)})^T}{\|w^{(k)}\|_2^2}. \qquad (3.31)$$

As usual, I_k is the identity matrix of order k , while R_{n-k} is the elementary Householder matrix of order $n-k$ associated with the reflection across the hyperplane orthogonal to the vector $w^{(k)} \in \mathbb{R}^{n-k}$. According to (4.29), the Householder vector is given by

$$w^{(k)} = x^{(n-k)} \pm \left\|x^{(n-k)}\right\|_2 e_1^{(n-k)}, \qquad (3.32)$$

where $x^{(n-k)} \in \mathbb{R}^{n-k}$ is the vector formed by the last $n-k$ components of x and $e_1^{(n-k)}$ is the first unit vector of the canonical basis of $\mathbb{R}^{n-k}$. We notice that $P_{(k)}$ is a function of x through $w^{(k)}$ will be discussed in Section 3.6.5.

The components of the transformed vector $y = P_{(k)}x$ read

$$\begin{cases} y_i = x_i & j = 1, \dots, k, \\ y_i = 0 & j = k+2, \dots, n, \\ y_{k+1} = \pm \left\| x^{(n-k)} \right\|_{2.} & \end{cases}$$

The Householder matrices will be employed in Section 3.6.2 to carry out the reduction of a given matrix A to a matrix $H^{(0)}$ in upper Hesssnberg form.

This is the first step for an efficient implementation of the QR iteration (3.22)with $T^{(0)} = H^{(0)}$ (see Section 3.6).

Example 3.3 Let $x = [1,2,3,4,5]^T$ and $k = 1$ (this means that we want to set to zero the components x_j, with $j = 3,4,5$). The matrix $P_{(1)}$ and the transformed vector $y = P_{(1)}x$ are given by

$$P_{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.2722 & 0.4082 & 0.5443 & 0.6804 \\ 0 & 0.4082 & 0.7710 & -0.3053 & -0.3816 \\ 0 & 0.5443 & -0.3053 & 0.5929 & -0.5089 \\ 0 & 0.6804 & -0.3816 & -0.5089 & 0.3639 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 7.34850 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The Givens elementary matrices are orthogonal rotation matrices that allow for setting to zero in a selective way the entries of a vector or matrix.

For a given pair of indices i and k, and a given angle θ , these matrices are defined as

$$G(i,k,\theta) = I_n - Y \qquad (3.33)$$

where $Y \in \mathbb{R}^{n\times n}$ is a null matrix expect for the following entries : $y_{ii} = y_{kk} = 1 - \cos(\theta), y_{ik} = -\sin(\theta) = -y_{ik}$. A Givens matrix is of the form

$$G(i,k,\theta) = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{c} \begin{array}{ccccccccc} & & & i & & k & & & \end{array} \\ \begin{bmatrix} 1 & & & & & & & & 0 \\ & 1 & & & & & & & \\ & & \ddots & & & & & & \\ & & & \cos(\theta) & & \sin(\theta) & & & \\ & & & & \ddots & & & & \\ & & & -\sin(\theta) & & \cos(\theta) & & & \\ & & & & & & \ddots & & \\ & & & & & & & 1 & \\ 0 & & & & & & & & 1 \end{bmatrix} \end{array} \begin{array}{c} \\ \\ \\ \\ i \\ \\ k \\ \\ \\ \\ \end{array}$$

For a given vector $x \in \mathbb{R}^n$, the product $y = (G(i,k,\theta))^T x$ is equivalent to rotating x counterclockwise by an angle θ in the coordinate plane (x_i, x_k) (see Figure 3.4, right). After letting $c = cos\theta, s = sin\theta,$ it follows that

$$y_j = \begin{cases} x_j & j \neq i,k, \\ cx_i - sx_k & j = i, \\ sx_i - cx_k & j = k \end{cases} \qquad (3.34)$$

Let $\alpha_{ik} = \sqrt{x_i^2 + x_k^2}$ and notice that if c and s satisfy $c = x_i/\alpha_{ik},\ s = -x_k/\alpha_{ik},$ (in each a case, $\theta = \arctan(-x_k/x_i,))$, we get $y_k = 0$, $y_i = \alpha_{ik}$ and $y_i = x_j$ for $j \neq i,k$. Similarly , if $c = x_k/\alpha_{ik},\ s = x_i/\alpha_{ik},$ (that is , $\theta = \arctan(x_i/x_k,))$, then $y_i = 0, y_k = \alpha_{ik}$ and $y_j = x_j$ for $j \neq i,k$.

Remark 3.3 (Householder deflation for power iterations)

The elementary Householder transformations can be conveniently employed to compute the first (largest or smallest) eigenvalues of given matrix $A \in \mathbb{R}^{n\times n}$.

Assume that the eigenvalues of A are ordered as in (3.8) and suppose that the eigenvalue/ eigenvector pair (λ_1, x_1) has been computed using the power method. Then the matrix A can be transformed into the following block form ([34])

$$A_1 = HAH = \begin{pmatrix} \lambda_1 & b^T \\ 0 & A_2 \end{pmatrix}$$

where $b \in \mathbb{R}^{n-1}$, H is the Householder matrix such that $Hx_1 = \alpha x_1$ for some $\alpha \in \mathbb{R}$, the matrix $A_2 \in \mathbb{R}^{(n-1)\times(n-1)}$ and the eigenvalues of A_2 are the same as those of A except for λ_1 . The matrix H can be computed using (3.28) with $v = x_1 \pm \|x_1\|_2 e_1$.

The deflation procedure consists of computing the second dominant (sub dominant) eigenvalue of A by applying the power method to A_2 provided that $\|\lambda_2\| \neq \|\lambda_3\|$. Once$\lambda_2$ is available , the corresponding eigenvector x_2 can be computed by applying the inverse power iteration to the matrix A taking $\mu = \lambda_2$ (see Section 3.3.2) and proceeding in the same manner with the remaining eigenvalue/eigenvector pair.

3-6-2 Reducing a Matrix in Hessenberg Form:

A given matrix $A \in \mathbb{R}^{n\times n}$ can be transformed by similarity transformations into upper Hessenberg form with a cost of the order of n^3 flops. The algorithm takes $n-2$ steps and similarity transformation Q can be computed as the product of Householder matrices $P_{(1)} \dots P_{(n-2)}$. For this, the reduction procedure is commonly known as the Householder method.

Precisely , the $k-th$ step consists of similarity transformation of A through the Householder matrix $P_{(k)}$ which aims at setting to zero the elements in positions $k+2, \dots, n$ of the $k-th$ column of A , for $k = 1, .., (n-2)$ (see Section 3.6.1). For example , in the case $n = 4$ the reduction process yields

$$\begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{bmatrix} \vec{P}_{(1)} \begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{bmatrix} \vec{P}_{(2)} \begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{bmatrix},$$

having denoted by $\blacksquare$ the entries of the matrices that are a priori non zero. Given $A^{(0)} = A,$ the method generates a sequence of matrices $A^{(k)}$ that are orthogonally similar to A

$$A^{(k)} = P_{(k)}^T A^{(k-1)} P_{(k)} = \left(P_{(k)} \dots P_1\right)^T A\left(P_{(k)} \dots P_1\right)$$

$$= Q_{(k)}^T A Q_{(k),} \qquad\qquad k \geq 1. \qquad\qquad (3.35)$$

For any $k \geq 1$ the matrix $P_{(k)}$ is given by (3.31), where x is substituted by the $k-th$ column vector in matrix $A^{(k-1)}$ .From the definition (3.31) it is easy to check that the operation $P_{(k)}^T A^{(k-1)} P_{(k)} = A^{(k)}$ does the same on the first k columns. After $n-2$ steps of Householder reduction, we obtain a matrix $H = A^{(n-2)}$ in upper Hessenberg form.

Remark 3.4 (The symmetric case) If is symmetric , the transformation (3.35) maintains such a property. Indeed

$$(A^{(k)})^T = \left(Q_{(k)}^T A Q_{(k)}\right)^T = A^{(k)}, \quad \forall k \geq 1,$$

so that H must be tridiagonal. Its eigenvalues can be efficiently computed using the method of Sturm sequences with a cost of the order of n flops.

A coding of the Householder reduction method is provided in Program 8. To compute the Householder vector , Program 11 is employed . In output , the tow matrices H and Q respectively in Hessenberg form and orthogonal, are such that $H = Q^T A Q$.

Program 8 – Houshess :Hessenberg- Householder method:

```
function [H,Q]=houshess(A)
n=max(size(A)); Q=eye(n); H=A;
for k=1:(n-2),
   [v,beta]=vhouse(H(k+1:n,k)); I=eye(k); N=zeros(k,n-k);
   m=length(v); R=eye(m)-beta*v*v`; H(k+1:n,k:n)=R*H(k+1:n,k:n);
   H(1:n,k+1:n)=H(1:n,k+1:n)*R; P=[1,N;N`;R];  Q=Q*P;
end
```

The algorithm coded in Program 8 requires a cost of $10n^3/3$ flops and is well- conditioned with respect to rounding errors . Indeed, the following estimate holds ([34])

$$\widehat{H} = Q^T(A+E)Q, \qquad \|E\|_F \le cn^2u\|A\|_F \qquad (3.36)$$

where $\widehat{H}$ is the Hessenberg matrix computed by Program 8, Q is an orthogonal matrix , c is a constant, u is the roundoff unit and $\|.\|_F$ is the Frobenius norm .

Example 3.3 Consider the reduction in upper Hessenberg form of the Hilbert matrix $H_4 \in \mathbb{R}^{4\times4}$. Since H_4 is symmetric, its Hessenberg form should be a triadigonal symmetric matrix .Program 8 yields the following results

$$Q = \begin{bmatrix} 1.00 & 0 & 0 & 0 \\ 0 & 0.77 & -0.61 & 0.20 \\ 0 & 0.51 & 0.40 & -0.76 \\ 0 & 0.38 & 0.69 & 0.61 \end{bmatrix} \quad H = \begin{bmatrix} 1.00 & 0.65 & 0 & 0 \\ 0.65 & 0.65 & 0.06 & 0 \\ 0 & 0.06 & 0.02 & 0.001 \\ 0 & 0 & 0.001 & 0.0003 \end{bmatrix}$$

The accuracy of the transformation procedure (3.35) can be measured by computing the $\|.\|_F$ norm of the difference between H and Q^TH_4Q . This yields $\|H - Q^TH_4Q\|_F = 3.38.10^{-17}$, which confirms the stability estimate (3.36).

3-6-3 QR factorization of a Matrix in Hessenberg Form:

We explain how to efficiently implement the generic step of the QR iteration, starting form a matrix $T^{(0)} = H^{(0)}$ in upper Hessenberg form.

For any $k \geq 1$, the first phase consists of computing the QR factorization of $H^{(k-1)}$ by means of $n-1$ Givens rotation

$$(Q^{(k)})^T H^{(k-1)} = (G_{n-1}^{(k)})^T \dots (G_1^{(k)})^T H^{(k-1)} = R^{(k)} \qquad (3.37)$$

where, for any $j = 1, \dots, n-1$, $G_j^{(k)} = G(j, j+1, \theta_j)^{(k)}$ is , for any $k \geq 1$, the $j-th$ Gives rotation matrix (3.43) in which θ_j is chosen according to (3.34) in such a way that the entry of indices (j+1,j) of the matrix $(G_j^{(k)})^T, \dots (G_1^{(k)})^T H^{(k-1)}$ is set equal to zero. The product (3.37) requires a computational cost of the order of $3n^2$ flops.

The next step consists of completing the orthogonal similarity transformation

$$H^{(k)} = R^{(k)} Q^{(k)} = R^{(k)} \left(G_1^{(k)}, \dots, G_{n-1}^{(k)} \right). \qquad (3.38)$$

The orthogonal matrix $Q^{(k)} = \left(G_1^{(k)}, \dots, G_{n-1}^{(k)} \right)$ is in upper Hessenberg form .Indeed , taking for instance $n = 3$, and recalling Section (3.6.1), we get

$$Q^{(k)} = G_1^{(k)} G_2^{(k)} = \begin{bmatrix} \blacksquare & \blacksquare & 0 \\ \blacksquare & \blacksquare & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare \end{bmatrix} = \begin{bmatrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare \end{bmatrix}.$$

Also (3.38) requires a cost of the order of $3n^2$ operations, for an overall effort of the order of $6n^2$ flops. In conclusion, performing the QR factorization with elementary Givens rotations on a starting matrix in upper Hessenberg form yields a reduction of operation count of one order of magnitude with respect to the corresponding factorization with the modified Gram Schmidt procedure of Section 3.4.

3-6-4 The Basic QR Iteration Starting from Upper:

Hessenberg Form.

A basic implementation of the QR iteration to generate the real Schur decomposition A is given in Program 9.

This program uses Program 8 to reduce A in upper Hessenberg form , then each QR factorization step in (4.22) is carried out with Program 10 which utilizes Givens rotations. The overall efficiency of the algorithm is ensured by pre-and post-multiplying with Givens matrices as explained in Section 3.6.5, and by constructing the matrix $Q^{(k)} = G_1^{(k)}, \dots, G_{n-1}^{(k)}$ in the function prodgiv , with a cost of $n^2 - 2$ flops and without explicitly forming the Givens matrices $G_j^{(k)}$, for $j = 1, \dots, n-1$.

As for the stability of the QR iteration with respect to rounding error propagation, it can be shown that the computed real Schur form $\hat{T}$ is orthogonally similar to a matrix "close" to A , i.e.

$$\hat{T} = Q^T(A+E)Q$$

where Q is orthogonal and $\|E\|_2 \simeq u\|A\|_2$, u being the machine roundoff unit.

Program 9 returns in output, after niter iterations of the QR procedure , the matrices T,Q and R in (3.22).

<u>Program 9- hessqu :Hessenberg – QR method:</u>

```
function [T,Q,R]=hessqr(A,niter)
n=max(size(A));
[T,Qhess]=houshess(A);
for j=1:niter
   [Q,R,c,s]=qrgivens(T);
T=R;
```

```
for k=1:n-1
    T=gacol(T,c(k),s(k),1,k+1,k,k+1);
     end
end
```

Program 10 - givensqr :QR factorization with Givens rotations:

```
function [Q,R,c,s]=qrgivens(H)
[m,n]=size(H);
for k=1:n-1
   [c(k),s(k)]=givcos(H(k,k),H(k+1,k));
   H=garow(H,c(k),s(k),k,k+1,k,n);
end
R=H; Q=prodgiv(c,s,n);
function Q=prodgiv(c,s,n)
n1=n-1;n2=n-2;
Q=eye(n);Q(n1,n1)=c)n1)=-s(n1);
for k=n2:-1:1,
k1=k+1;Q(k,k)=c(k);Q(k1,k)=-s(k);
q=Q(k1,k1:n); Q(k,k1:n)=s(k)*q;
Q(k1,k1:n)=c(k)*q;
end
```

Example 3.5 consider the matrix A (already in Hessenberg form)

$$A = \begin{bmatrix} 3 & 17 & -37 & 18 & -40 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

To compute its eigenvalues, given by $-4, \pm i$ and 5 , we apply the QR method and we compute the matrix $T^{(40)}$ after 40 iteration of Program 9. Notice that the algorithm converges to the real Schur decomposition of A (3.24), with three blocksR_{ii} of order 1 $i = (1,2,3)$ and with the block $R_{44} = T^{(40)}(4{:}\,5,4{:}\,5)$ having eigenvalues equal to $\pm i$

$$T^{(40)} = \begin{bmatrix} 4.9997 & 18.9739 & -34.2570 & 32.8760 & -28.4604 \\ 0 & -3.9997 & 6.7693 & -6.4968 & 5.6216 \\ 0 & 0 & 2 & -1.4557 & 1.1562 \\ 0 & 0 & 0 & 0.3129 & -0.8709 \\ 0 & 0 & 0 & 1.2607 & -0.3129 \end{bmatrix}$$

Example 3.6 Let us now employ the QR method to generate the Schur real decomposition of the matrix A below, after reducing it to upper Hessenberg form

$$A = \begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix}$$

The eigenvalues of A are real and given (to four significant figures) by $\lambda_1 = 65$,and $\lambda_{2,3} = \pm 21.28$ and $\lambda_{4,5} = \pm 13.13$. After 40 iterations of Program 9, the computed matrix reads

$$T^{(40)} = \begin{bmatrix} 65 & 0 & 0 & 0 & 0 \\ 0 & 14.6701 & 14.2435 & 4.4848 & 5.6216 \\ 0 & 16.6735 & -14.6701 & -1.2159 & 2.0416 \\ 0 & 0 & 0 & -13.0293 & -0.7643 \\ 0 & 0 & 0 & -3.3173 & 13.0293 \end{bmatrix}$$

It is not upper triangular, but block upper triangular, with a diagonal block $R_{11} = 65$ and the two blocks

$$R_{22} = \begin{bmatrix} 14.6701 & 14.2435 \\ 16.6735 & -14.6701 \end{bmatrix}, R_{33} = \begin{bmatrix} -13.0293 & -0.7643 \\ -3.3173 & 13.0293 \end{bmatrix}$$

having spectrums given by $\sigma(R_{22}) = \lambda_{2,3}$ and $\sigma(R_{33}) = \lambda_{4,5}$ respectively.

It is important to recognize that matrix $T^{(40)}$ is not the real Schur decomposition of A , but only a "cheating" version of it . In fact, in order for the QR method to converge to the Schur decomposition of A , it is mandatory to resort to the shift techniques introduced in Section 3.7.

3-6-5 Implementation of transformation Matrices:

In the definition (3.32) it is convenient to choose the minus sign, obtaining $W^{(k)} = x^{(n-k)} - \|x^{(n-k)}\|_2 e_1^{(n-k)}$, in such a way that the vector $R_{n-k}x^{(n-K)}$ is a positive multiple of $e_1^{(n-k)}$. If x_{k+1} is positive, in order to avoid numerical cancellations, the computation can be rationalized as follows

$$w_1^{(k)} = \frac{x_{k+1}^2 - \|x^{(n-k)}\|_2^2}{x_{k+1} + \|x^{(n-k)}\|_2} = \frac{-\sum_{j=k+2}^{n} x_j^2}{x_{k+1} + \|x^{(n-k)}\|_2}.$$

The construction of the Householder vector is performed by Program 11, which takes as input a vector $p \in \mathbb{R}^{n-k}$ (formerly, the vector$x^{(n-k)}$) and returns a vector $q \in \mathbb{R}^{n-k}$ (the Householder vector $W^{(k)}$), with a cost of the order of n flops .

If $M \in \mathbb{R}^{m\times m}$is the generic matrix to which the Householder matrix P (3.28) is applied (where I is the identity matrix of order m and $v \in \mathbb{R}^m$), letting $w = M^T v$, then

$$PM = M - \beta v w^T, \qquad \beta = 2/\|v\|_2^2. \qquad (3.39)$$

Therefore, performing the product PM amounts to a matrix- vector product ($w = M^T v$) plus an external product vector- vector (vw^T). The overall computational cost of the product PM is thus equal to $2(m^2 + m)$ flops. Similar consideration hold in the case where the product MP is to be computed ; defining $w = Mv,$ we get

$$MP = M - \beta wv^T. \qquad (3.40)$$

Notice that (3.39) and (3.40) do not require the explicit construction of the matrix P. This reduces the computational cost to an order of m^2 flops , whilst executing the product PM without taking advantage of the special structure of P would increase the operation count to an order of m^3 flops.

Program 11 - vhouse :Construction of the Householder vector:

```
function [v,beta]=vhouse(x)

n=length(x); x=x/norm(x); s=x(2:n)`*x(2:n); v=[1;x(2:n)];

if (s==0), beta=0;

else

    mu=sqrt(x(1)^2+s);

if (x(1)<=0), v(1)=x(1)=x(1)-mu;

else,       v(1)=-s/(x(1)+mu);   end

beta=2*v(1)^2/(s+v(1)^2);  v=v/v(1);

end
```

Concerning the Givens rotation matrices, the computation of c and s is carried out as follows. Let i and k be two fixed indices and assume that the $k-th$ component of a given vector $x \in \mathbb{R}^n$ must be set to zero. Letting $r = \sqrt{x_i^2 + x_k^2}$, relation (3.34)yields

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x_i \\ x_k \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \qquad (3.41)$$

hence there is no need of explicitly computing θ, nor evaluating any trigonometric function.

Executing Program 12 to solve system (3.41), requires 5 flops, plus the evaluation of a square root. As already noticed in the case of Householder matrices, even for Givens rotations we don't have to explicitly compute the matrix $G(i,k,\theta)$ to perform its product with a given matrix $M \in \mathbb{R}^{m\times m}$. For that purpose Program 13 and 14 are used, both at the cost of $6m$ flops. Looking at the structure (3.33) of matrix $G(i,k,\theta)$, it is clear that the first algorithm only modifies rows i and k of M.

We conclude by noticing that the computation of Householder vector v and of the Givens sine and cosine (c,s), are well-conditioned operations with respect to rounding errors ([33]). parameters are the vector components x_i and x_k , whilst the output data are the Givens cosine and sine c and s.

<u>Program 12 –givcos :Computation of Givens cosine and sine:</u>

```
function [c,s]=givcos{xi,xk)
if (xk==0), c=1; s=0;else,
    if abs(xk)> abs(xi)
  t=-xi/xk; s=1/sqrt(1+t^2); c=s*t;
else
t=-xk/xi; c=1/sqrt(1+t^2); s=c*t;
    end
end
```

Program 13 and 14 compute $G(i,k,\theta)^T M$ and $MG(i,k,\theta)$ respectively. The input parameters c and s are the Givens cosine and sine .In Program 13, the indices i and k identify the rows of the matrix M, while $j1$and $j2$ are indices of the columns involved in the computation. Similarly, in Program 14 i and k identify the columns effected by the update $M \leftarrow MG(i,k,\theta)$, while $j1$and $j2$ are the indices of the rows involved in the computation .

Program 13 –garow : Product $\mathbf{G(i,k,\theta)^T M}$

```
function [M]=garow(M,c,s,i,k,j1,j2)
for j=j1:j2
   t1=M(i,j);
   t2=M(k,j);
   M(i,j)=c*t1-s*t2;
   M(k,j)=s*t1+c*t2;
end
```

Program 14 –gacol :Product $\mathbf{MG(i,k,\theta)}$

```
function [M]=gacol(M,c,s,j1,j2,i,k)
for j=j1:j2
    t1=M(j,i);
   t2=M(j,k);
   M(j,i)=c*t1-s*t2;
  M(j,k)=s*t1+c*t2;
End
```

3-7 The QR Iteration with Shifting Techniques:

Example 3.9 reveals that the QR iteration does not always converge to the real Schur form of a given matrix A . To make this happen, an effective approach consists of incorporating in the QR iteration (3.22) a shifting technique similar to that introduced for inverse iteration in Section 3.3.2.

This leads to the QR method with single shift described in Section 3.7.1, which is used to accelerate the convergence of the QR iteration when A has eigenvalues with moduli very close to each other.

In Section 3.7.2, a more sophisticated shifting technique is considered, which guarantees the convergence of the QR iteration to the (approximate) Schur form of matrix A (see Property3.5). The resulting method (known as QR iteration with double shift) is the most popular version of the QR iteration (3.22) for solving the matrix eigenvalue problem, and is implemented in the MATLAB intrinsic function eig.

3-7-1 The QR Method with single shift:

Given $\mu \in \mathbb{R}$, the shifted QR iteration is defined as follows. For $k = 1,2, \dots$, until convergence :

determine $Q^{(k)}R^{(k)} = T^{(k-1)} - \mu I$ (QR factorization);

then , let (3.42)

$$T^{(k)} = R^{(k)}Q^{(k)} + \mu I.$$

where $T^{(0)} = (Q^{(0)})^T A Q^{(0)}$ is in upper Hessenberg form. Since the QR factorization in (3.52) is performed on the shifted matrix $T^{(k-1)} - \mu I$, the scalar μ is called shift. The sequence of matrices $T^{(k)}$ generated by (3.42) is still similar to the initial matrix A, since for any $k \geq 1$

$$R^{(k)}Q^{(k)} + \mu I = \left(Q^{(k)}\right)^T (Q^{(k)}R^{(k)}Q^{(k)} + \mu Q^{(k)})$$

$$= \left(Q^{(k)}\right)^T \left(Q^{(k)}R^{(k)} + \mu I\right)Q^{(k)} = \left(Q^{(k)}\right)^T T^{(k-1)}Q^{(k)}$$

$$= \left(Q^{(0)}Q^{(1)} \dots Q^{(k)}\right)^T A\left(Q^{(0)}Q^{(1)} \dots Q^{(k)}\right), \quad k \geq 0$$

Assume μ is fixed and that the eigenvalues of A are ordered in such a way that

$$|\lambda_1 - \mu| \geq |\lambda_2 - \mu| \geq \cdots |\lambda_n - \mu|.$$

Then it can be shown that, for $1 < j \leq n$, the subdiagonal entry $t_{j,j-1}^{(k)}$ tends to zero with a rate that is proportional to the ratio

$$\left|(\lambda_j - \mu)/(\lambda_{j-1} - \mu)\right|^k.$$

This extends the convergence result (3.27) to the shifted QR method ([35]).

The result above suggests that if μ is chosen in such a way that

$$|\lambda_n - \mu| < |\lambda_i - \mu| \qquad i = 1, \dots, n-1,$$

Then the matrix entry $t_{n,n-1}^{(k)}$ in iteration (3.42) tends rapidly to zero as k increases. (In the limit, if μ were equal to an eigenvalue of $T^{(k)}$, that is of A, then $t_{n,n-1}^{(k)} = 0$ and $t_{n,n}^{(k)} = \mu$. In practice one takes

$$\mu = t_{n,n}^{(k)}, \qquad (3.43)$$

yielding the so called QR iteration with single shift. Correspondingly, the convergence to zero of the sequence $\left\{t_{n,n-1}^{(k)}\right\}$ is quadratic in the sense that if $\left|t_{n,n-1}^{(k)}\right|/\|T^{(0)}\|_2 = \eta_k < 1$, for some $k \geq 0$, then $\left|t_{n,n-1}^{(k)}\right|/\|T^{(0)}\|_2 = \mathcal{O}(\eta_k^2)$, ([18], [22]).

This can be profitably taken into account when programming the QR iteration with single shift by monitoring the size of the subdiagonal entry $\left|t_{n,n-1}^{(k)}\right|$. In practice, $t_{n,n-1}^{(k)}$ is set equal to zero if

$$\left|t_{n,n-1}^{(k)}\right| \leq \varepsilon \left|t_{n-1,n-1}^{(k)}\right| + \left|t_{n,n}^{(k)}\right|, \qquad k \geq 0, \qquad (3.44)$$

for a prescribed ε, in general of the order of the roundoff unit. If A is an Hessenberg matrix, when for a certain k $a_{n,n-1}^{(k)}$ is set to zero, $t_{n,n}^{(k)}$ provides the desired approximation of λ_n. Then the QR iteration with shift can continue on the matrix $T^{(k)}(1:n-1,1:n-1)$, and so on. This is a deflation algorithm.

Example 3.7 We consider again the matrix A as in Example 3.6. Program 15, with toll equal to the roundoff unit, converges in 14 iterations to the following approximate real Schur form of A, which displays the correct eigenvalues of matrix A on its diagonal (to six significant figures)

$$T^{(40)} = \begin{bmatrix} 65 & 0 & 0 & 0 & 0 \\ 0 & -21.2768 & 2.5888 & -0.0445 & -4.2959 \\ 0 & 0 & -13.1263 & -4.0294 & -13.079 \\ 0 & 0 & 0 & 21.2768 & -2.6197 \\ 0 & 0 & 0 & 0 & 13.1263 \end{bmatrix}.$$

We also report in Table 3.1 the convergence rate $p^{(k)}$ of the sequence $\left\{t_{n,n-1}^{(k)}\right\}$ $(n = 5)$ computed as

$$p^{(k)} = 1 + \frac{1}{\log \eta_k} \log \frac{t_{n,n-1}^{(k)}}{t_{n,n-1}^{(k-1)}}, \; k \geq 1.$$

The results show good agreement with the expected quadratic rate.

k	$\left\|t_{n,n-1}^{(k)}\right\| / \left\|T^{(0)}\right\|_2$	$p^{(k)}$
0	0.13865	
1	$1.5401.10^{-2}$	2.1122
2	$1.2213.\ 10^{-4}$	2.1591
3	$1.8268.\ 10^{-8}$	1.9775
4	$8.9036.\ 10^{-16}$	1.9449

TABLE 3.1. Convergence rate of the sequence $\left\{t_{n,n-1}^{(k)}\right\}$ in the QR iteration with single shift

The coding of the QR iteration with single shift (3.42) is given in Program 15. The code utilizes Program 8 to reduce the matrix A in upper Hessenberg from and Program 10 to perform the QR factorization step. The input parameters toll and itmax are the tolerance ε in (3.44) and the maximum admissible number of iterations, respectively . In output, the program returns the (approximated) real Schur form of A and the number of iterations needed for its computation.

Program 15 – qrshift :QR iteration with single shift

```
function[T,itr]=qrshift(A,toll,itmax)
n=max(size(A)); iter=0; [T,Q]=houshess(A);
for k=n:-1:n
I=eye(k);
while abs(T(k,k-1))> tool*(abs(T(k,k)+abs(T(k,k-1)))
      iter=iter+1;
      if(iter>itmax),
        return
      end
      mu=T(k,k); [Q,R,c,s]=qrgivens(T(1:k,1:k)-mu*I);
     T(1:k,1:k)=R*Q+mu*I;
 end
T(k,k-1)=0;
end
```

3-7-2 The QR Iteration Method with Double shift:

The single-shift QR iteration (3.42) with the choice (3.43) for μ is effective if the eigenvalues of A are real, but not necessarily when complex conjugate eigenvalues are present, as happens in the following example.

Example 3.8 The matrix $A \in \mathbb{R}^{4\times 4}$ (reported below to five significant figures)

$$A = \begin{bmatrix} 1.5726 & -0.6392 & 3.7696 & -1.3143 \\ 0.2166 & -0.0420 & 0.4006 & -1.2054 \\ 0.0226 & 0.3592 & 0.2045 & -0.1411 \\ -0.1814 & 1.1146 & -3.2330 & 1.2648 \end{bmatrix}$$

has eigenvalues $\{\pm i, 1,2\}, i$ being imaginary unit. Running Program 15 with toll equal to the roundoff unit yields after 100 iterations

$$T^{(101)} = \begin{bmatrix} 2 & 1.1999 & 0.5148 & 4.9004 \\ 0 & -0.0001 & -0.8575 & 0.7182 \\ 0 & 1.1662 & 0.0001 & -0.8186 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The obtained matrix is the real Schur form of A, where the 2×2 block $T^{(101)}$ (2:3,2:3) has complex conjugate eigenvalues $\pm i$. These eigenvalues cannot be computed by the algorithm (3.42)-(3.43) since μ is real .

The problem with this example is that working with real matrices necessarily yields a real shift, whereas a complex one would be needed. The QR iteration with double shift is set up to account for complex eigenvalues and allows for removing the 2×2 diagonal blocks of the real Schur form of A .

Precisely, suppose that the QR iteration with single shift (3.42) detects at some step k a 2×2 diagonal block $R_{kk}^{(k)}$ that cannot be reduced into upper triangular form. Since the iteration is converging to the real Schur form of the matrix A the two eigenvalues of $R_{kk}^{(k)}$ are complex conjugate and will be denoted by $\lambda^{(k)}$ and $\bar{\lambda}^{(k)}$. The double shift strategy consists of the following steps:

determine $Q^{(k)}, R^{(k)}$ such that

$Q^{(k)}R^{(k)} = T^{(k-1)} - \lambda^{(k)}I$ (first QR factorization);

then, let

$T^{(k)} = R^{(k)}Q^{(k)} - \lambda^{(k)}I$;

determine $Q^{(k+1)}, R^{(k+1)}$ such that [3.45]

$Q^{(k+1)}R^{(k+1)} = T^{(k)} - \bar{\lambda}^{(k)}I$ (second QR factorization)

then, let

$T^{(k+1)} = Q^{(k+1)}R^{(k+1)} - \bar{\lambda}^{(k)}I.$

Once the double shift has been carried out the QR iteration with single shift is continued until a situation analogous to the one above is encountered.

The QR iteration incorporating the double shift strategy is the most effective algorithm for computing eigenvalues and yields the approximate Schur form of a given matrix A. Its actual implementation is far more sophisticated than the outline above and is called QR iteration with Francis shift ([18] ,[22]). As for the case of the QR iteration with single shift, quadratic convergence can also be proven for the QR method with Francis shift. However, special matrices have recently been found for which the method fails to converge. We refer for some analysis and remedies to [39],[40], although the finding of a shift strategy that guarantees convergence of the QR iteration for all matrices is still an open problem.

Example 3.9 Let us apply the QR iteration with double shift to the matrix A in Example 3.8. After 97 iterations of Program 16, with toll equal to the roundoff unit, we get the following (approximate) Schur form of A, which displays on its diagonal the four eigenvalues of A

$$T^{(97)} = \begin{bmatrix} 2 & 1+2i & -2.33+0.86i & 4.90 \\ 0 & 5.02.10^{-14}+i & -2.02+6.91.10^{-14}i & 0.72 \\ t_{31}^{(97)} & 0 & -1.78.10^{-14}-i & -0.82 \\ t_{41}^{(97)} & i_{42}^{(97)} & 0 & 1 \end{bmatrix}$$

where $t_{31}^{(97)} = 2.06.10^{-17} + 7.15.10^{-49}i$, $t_{41}^{(97)} = -5.59.10^{-17}$ and $i_{42}^{(97)} = -4.26.10^{-18}$, respectively .

A basic implementation of the QR iteration with double shift is provided in Program 16. The input/output parameters are the same as those of Program 15 . The output matrix T is the approximate Schur form of matrix A .

Program 16 -qr2shift :QR iteration with double shift

```
  function [T,iter]=qr2shift(A,toll,itmax)
 n=max(size(A)); iter=0; [T,Q]=houshess(A);
for k=n:-1:2
I=eye(k);
while abs(T(k,k-1))>toll*(abs(T(k,k))+abs(T(k-1,k-1)))
iter=iter+1;  if (iter > itmax), return, end
mu=T(k,k); [Q,R,c,s]=qrgivens(T(1:k,1:k)-mu*I);
T(1:k,1:k)=R*Q+mu*I;
if (k>2),
T diag2=abs(T(k-1,k-1))+abs(T(k-2,k-2));
 if   abs(T(k-1,k-2)) i=toll*T diag2;
 [Iambda]=eig(T(k-1:k,k-1:k));
 [Q,R,c,s]=qrgivens(T(1:k,1:k)-Iambda(1)*I);
T(1:k,1:k)=R*Q+Iambda(1)*I;
```

```
   [Q,R,c,s]=qrgivens(T(1:k,1:k)-Iambda(2)*I);
   T(1:k,1:k)R*Q+Iambda(2)*I;
end
end
end, T(k,k-1)=0;
end
I=eye(2);
while (abs(T(2,1))>toll*(abs(T(2,2))+abs(T(1,1))))&(iter<=itmax)
  iter=iter+1; mu=T(2,2);
  [Q,R,c,s]=qrgivens(T(1:2,1:2)-mu*I);  T(1:2,1:2)=R*Q+mu*I;
end
```

3-8 Computing The Eigenvector and the SVD of a Matrix:

The power and inverse iteration described in Section 3.3.2 can be used to compute a selected number of eigenvalue/eigenvector pairs. If all the eigenvalues and eigenvectors of matrix are needed, the QR iteration can be profitably employed to compute the eigenvectors as shown in Section 3.8.1and 3.8.2. In Section 3.8.3 we deal with the computation of the singular value decomposition (SVD) of a given matrix.

3-8-1 The Hessenberg Inverse Iteration:

For any approximate eigenvalue λ computed by the QR iteration as described in Section 3.7.2, the inverse iteration (3.20)cab be applied to the matrix $H = Q^T AQ$ in Hessenberg form, yielding an approximate eigenvector q. Then, the eigenvector x associated with λ is computed as $x = Qq$. Clearly, one can take advantage of the structure of the Hessenberg matrix for an efficient solution of the linear system at each step of (3.20). Typically, only one iteration is required to produce an adequate approximation of the desired eigenvector x ([22]).

3-8-2 Computing the Eigenvectors from the Schur Form of a matrix:

Suppose that the (approximate) Schur form $Q^H AQ = T$ of a given matrix $A \in \mathbb{R}^{n\times n}$ has been computed by the QR iteration with double shift, $being$ a unitary matrix and T being upper triangular.

Then if $Ax = \lambda x$, we have $Q^H AQQ^H x = Q^H \lambda x,$ i.e letting $y = Q^H x,$ $Ty = \lambda y$holds. Therefore y is eigenvector of T, so that to compute the eigenvectors of A we can work directly on Schur form T.

Assume for simplicity that $\lambda = t_{kk} \in \mathbb{C}$ is a simple eigenvalue of A. Then the upper triangular matrix T can be decomposed as

$$T = \begin{bmatrix} T_{11} & V & T_{13} \\ 0 & \lambda & W^T \\ 0 & 0 & T_{33} \end{bmatrix},$$

where $T_{11} \in \mathbb{C}^{(k-1)\times(k-1)}$ and $T_{33} \in \mathbb{C}^{(n-k)\times(n-k)}$ are upper triangular matrices, $V \in \mathbb{C}^{k-1}$, and $\lambda \notin \sigma(T_{11}) \cup \sigma(T_{33})$.

Thus, letting $y = (y_{k-1}^T, y, y_{n-k}^T)$, with $y_{k-1} \in \mathbb{C}^{k-1}$, $y \in \mathbb{C}$ and $y_{n-k} \in \mathbb{C}^{n-k}$, the matrix eigenvector problem $(T - \lambda I)Y = 0$ can be written as

$$\begin{cases} (T_{11} - \lambda I_{k-1})y_{k-1} + V_y + & T_{13} y_{n-k} & = 0 \\ & W^T y_{n-k} & = 0 \\ & (T_{33} - \lambda I_{n-k})y_{n-k} & = 0 \end{cases} \qquad (3.46)$$

Since λ is simple, both matrices $T_{11} - \lambda I_{k-1}$ and $T_{33} - \lambda I_{n-k}$ are nonsingular, so that the third equation in (3.46) yield $y_{n-k} = 0$ and the first equation becomes

$$(T_{11} - \lambda I_{k-1})y_{k-1} = -V_y \,.$$

Setting arbitrarily $y = 1$ and solving the triangular system above for y_{k-1} yields (formally)

$$y = \begin{pmatrix} -(T_{11} - \lambda I_{k-1})^{-1} V \\ 1 \\ 0 \end{pmatrix}.$$

The desired eigenvector x can then be computed as $x = Qy$.

Invoking this function with the format $[V, D] = eig(A)$ yields the matrix V whose columns are the right eigenvectors of A and the diagonal matrix D contains its eigenvalues. Further details can be found in strvec subroutine in the LAPACK library, while for the computation of eigenvectors in the case where A is symmetric, we refer to [22], [37].

3-8-3 Approximate Computation of the SVD of a matrix:

We describe the Golub-Kahan-Reinsh algorithm for computation of the SVD of a matrix $A \in \mathbb{R}^{m\times n}$ with $m > n$ ([22]). The method consists of two phases, a direct one and an iterative one.

In the first phase A is transformed into an upper trapezoidal matrix of the form

$$\mathcal{U}^T A \mathcal{V} = \begin{pmatrix} B \\ 0 \end{pmatrix}, \qquad (3.47)$$

where $\mathcal{U}$ and $\mathcal{V}$ are two orthogonal matrices and $B \in \mathbb{R}^{n\times n}$ is upper bidiagonal. The matrices $\mathcal{U}$ and $\mathcal{V}$ are generated using $n + m - 3$ Householder matrices $\mathcal{U}_1, \dots \mathcal{U}_{m-1}, \mathcal{V}_1, \dots, \mathcal{V}_{n-2}$ as follows.

The algorithm initially generates $\mathcal{U}_1$ in such a way that the matrix $A^{(1)} = \mathcal{U}_1 A$ has $a_{i1}^{(1)} = 0$ if $i > 1$. Then, $\mathcal{V}_1$ is determined so that $A^{(2)} = A^{(1)}\mathcal{V}_1$ has $a_{1j}^{(2)} = 0$ for $j > 2$, preserving at the same time the null entries of the previous step. The procedure is repeated starting from $A^{(2)}$, and taking $\mathcal{U}_2$ such that $A^{(3)} = \mathcal{U}_2 A^{(2)}$ has $a_{i2}^{(3)} = 0$ for $i > 2$ and $\mathcal{V}_2$ in such a way that $A^{(4)} = A^{(3)}\mathcal{V}_2$

has $a_{2j}^{(4)} = 0$ for $j > 3$, yet preserving the null entries already generated. For example, in the case $m = 5, n = 4$ the first two steps of the reduction process yield

$$A^{(1)} = \mathcal{U}_1 A = \begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare \end{bmatrix} \to A^{(2)} = A^{(1)} \mathcal{V}_1 = \begin{bmatrix} \blacksquare & \blacksquare & 0 & 0 \\ 0 & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare \end{bmatrix},$$

having denoted by ■the entries of the matrices that in principle are different than zero. After at most $m - 1$ steps, we find (3.47) with

$$\mathcal{U} = \mathcal{U}_1 \mathcal{U}_2 \ldots \mathcal{U}_{m-1}, \quad \mathcal{V} = \mathcal{V}_1 \mathcal{V}_2 \ldots \mathcal{V}_{n-2}.$$

In the second phase, the obtained matrix B is reduced into a diagonal matrix $\sum$ using the QR iteration. Precisely, a sequence of upper bidiagonal matrices $B^{(k)}$ are constructed such that, as $k \to \infty$, their off-diagonal entries tend to zero quadratically and the diagonal entries tend to the singular values σ_i of A. In the limit, the process generates two orthogonal matrices W and Z such that

$W^T BZ = \sum = diag(\sigma_1, \ldots, \sigma_N).$

The SVD of A is then given by

$U^T AV = \binom{\Sigma}{0},$

with $U = \mathcal{U} diag(W, I_{m-n)}$ and $V = \mathcal{V}Z.$

The computational cost of this procedure is $2m^2n + 4mn^2 + \frac{9}{2}n^3$ flops, which reduces to $2mn^2 - \frac{2}{3}n^3$ flops if only the singular values are computed. In this case, recalling what was stated in Section 3.8 about $A^T A$, the method described in the present section is preferable to computing directly the eiagenvalues of $A^T A$ and then taking their square roots.

As for the stability of this procedure, it can be shown that the computed σ_i turn out to be the singular values of the matrix $A + \delta A$ with

$\|\delta A\|_2 \leq C_{mn} u \|A\|_2,$

C_{mn} being a constant dependent on n, m and the roundoff unit u. For other approaches to the computation of the SVD of a matrix, ([22] , [34]).

3-9 The Generalized Eigenvalue Problem:

Let $A, B \in \mathbb{C}^{n\times n}$ be two given matrices, for any $z \in \mathbb{C}$, we call $A - zB$ a matrix pencil and denote it by (A, B). The set $\sigma(A, B)$of the eigenvalues of (A, B) is defined as

$\sigma(A, B) = \{\mu \in \mathbb{C} : \det(A - \mu B) = 0\}.$

The generalized matrix eigenvalue problem can be formulated as : find $\lambda \in \sigma(A, B)$ and a nonnull vector $x \in \mathbb{C}^n$ such that

$Ax = \lambda Bx.$ (3.48)

The pair (λ, x) satisfying (3.48) is an eigenvalue/eigenvector pair of the pencil (A, B). Note that by setting $B = I_n$ in (3.48) we recover the standard matrix eigenvalue problem considered thus far.

Problems like (3.48) arise frequently in engineering applications, e.g., in the study of vibrations of structures (buildings, aircrafts and bridges) or in the mode analysis for waveguides ([37] , [38]). Another example is the computation of the extermal eigenvalues of a preconditioned matrix $P^{-1}A$ (in which case $B = P$ in (3.48) when solving a linear system with an iterative method (see Remark 3.2).

Let us introduce some definitions. We say that the pencil (A, B) is regular if $\det(A - zB)$ is not identically zero, otherwise the pencil is singular. When (A, B) is regular, $p(z) = \det(A - zb)$ is the characteristic polynomial of the pencil; denoting by k the degree of p, the eigenvalues of (A, B) are defined as :

1. the roots of $p(z) = 0\,, if\ k = n$;
2. $\infty\ if\ k < n$ (with multiplicity equal to $n - k$).

Example 3.10

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad p(z) = z^2 + 1 \Rightarrow \sigma(A,B) = \pm i$$

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad p(z) = z \qquad \Rightarrow \sigma(A,B) = \{0, \infty\}$$

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad p(z) = 0 \qquad \Rightarrow \sigma(A,B) = \mathbb{C}.$$

The first pair of matrices shows that symmetric pencils, unlike symmetric matrices, may exhibit complex conjugate eigenvalues. The second pair is regular pencil displaying an eigenvalue equal to infinity, while the third pair is an example of singular pencil.

3-9-1 Computing the Generalized Real Schur Form:

The definitions and examples above imply that the pencil (A, B)has n finite eigenvalues iff B is nonsingular.

In such a case, a possible approach to the solution of problem (3.48) is to transform it into the equivalent eigenvalue problem $Cx = \lambda x$, where the matrix C is the solution of the system $BC = A$,then apply the QR iteration to C. For actually computing the matrix C, one can use Gauss elimination with pivoting or the techniques shown in Section 2.7. This procedure can yield inaccurate result if B is ill-conditioned, since computing C is sffected by rounding errors of the order of $u\|A\|_2\|B^{-1}\|_2$ ([22]).

A more attractive approach is based on the following result, which generalizes the Schur decomposition theorem 1.5 to the case of regular pencils to ([34]).

Property 3.9 (Generalized Schur decomposition) Let (A, B) be a regular pencil. Then, there exist two unitary matrices $U$and Z such that $U^H AZ = T$, $U^H BZ = S$, where T and $S$are upper triangular. For $i = 1, \dots, n$ the eigenvalues of (A, B) are given by

$$\lambda_i = {}^{t_{ii}}/_{s_{ii}} \ \ if \ s_{ii} \neq 0,$$

$$\lambda_i = \infty \ \ if \ t_{ii} \neq 0, s_{ii} \neq 0.$$

Exactly as in the matrix eigenvalue problem, the generalized Schur form cannot be explicitly computed, so the counterpart of the real Schur form (3.24) has to be computed. Assuming that the A and B are real, it can be shown that there exist two orthogonal matrices $\tilde{U}$ and $\tilde{Z}$ such that $\tilde{T} = \tilde{U}^T A\tilde{Z}$ is upper quasi-triangular $\tilde{S} = \tilde{U}^T B\tilde{Z}$ is upper triangular. This decomposition is known as the generalized real Schur decomposition of a pair (A, B) and can be computed by a suitably modified version of the QR algorithm, known as QZ iteration, which consists of the following steps ([22], ,[34]):

1. reduce A and B into upper Hessenberg form and upper triangular form, respectively, i.e., find two orthogonal matrices Q and Z such that $\mathcal{A} = Q^T AZ$ is upper Hessenberg and $\mathcal{B} = Q^T BZ$ is upper triangular ;
2. the QR iteration is applied to the matrix $\mathcal{A}\mathcal{B}^{-1}$ to reduce it to real Schur form.

To save computational resources, the QZ algorithm overwrites the matrices A and B on their upper Hessenberg and triangular forms and requires $30n^3$ flops; an additional cost of $36n^3$ operations is required if Q and $Z$are also needed.

3-9-2 Generalized Real Schur Form of Symmetric –Definite Pencils:

A remarkable situation occurs when both A and B are symmetric, and one of them, say B, is also positive definite. In such a case, the pair (A, B) forms a symmetric-definite pencil for which the following result holds.

Theorem 3.5 The symmetric-definite pencil (A, B) has real eigenvalues and linearly independent eigenvectors. Moreover, the matrix A and B can be simultaneously diagonalized. Precisely, there exists a nonsingular matrix $X \in \mathbb{R}^{n\times n}$ such that

$X^T A X = \Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_n),\ X^T B X = I_n,$

where for $i = 1, \dots, n,\ \lambda_i$ are the eigenvalue of the pencil (A, B).

Proof. Since B is symmetric positive definite, it admits a unique Cholesky factorization $B = H^T H$, where H is upper triangular (see Section 2.4). From (3.48) we deduce that $C_z = \lambda_z$ with $C = H^{-T} A H^{-1}$, $z = Hx$, where $(\lambda, x)$is an eigenvalue/eigenvector pair (A, B).

The matrix C is stmmetric; therefore, its eigenvalues are real and a set of orthonormal eigenvectors $(y_1, \dots., y_n) = Y$ exists. As a consequence, letting $X = H^{-1} Y$ allows for simultaneously diagonalizing both A and B since

$$X^T A X = Y^T H^{-T} A H^{-1} Y = Y^T C Y = \Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_n),$$

$$X^T B X = Y^T H^{-T} B H^{-1} Y = Y^T Y = I_n.$$

The following QR –Cholesky algorithm computes the eigenvalues λ_i and corresponding eigenvectors x_i of a symmetric-definite pencil (A, B), for $i = 1, \dots, n$ ([22], [34]):

1. compute the Cholesky factorization $B = H^{-T} H$;
2. compute $C = H^{-T} A H^{-1}$;
3. for $i = 1, \dots, n$, compute the eigenvalues λ_i and eigenvectors z_i of the symmetric matrix C using the QR iteration. Then construct from the set

$\{z_i\}$ an orthonormal set of eigenvectors $\{y_i\}$ (using, for instance, the modified Gram-Schmidt procedure of Section 3.4);

4. for $i = 1, \dots, n$, compute the eigenvectors x_i of pencil (A, B) by solving the systems $Hx_i = y_i$.

This algorithm requires an order of $14n^3$ flops and it can be shown ([22]) that if $\hat{\lambda}$ is a computed eigenvalue, then

$$\hat{\lambda} = \sigma(H^{-T}AH^{-1} + E), \qquad \text{with } \|E\|_2 \simeq u\|A\|_2\|B^{-1}\|_2.$$

Thus, the generalized eigenvalue problem in the symmetric-definite case may become unstable with respect to rounding errors propagation if B is ill-conditioned. For a stabilized version of the QR-Cholesky method, [22] and the references cited therein.

Chapter Four

4.0 Basic Concepts

4.0.1 Linear Constraints

In this chapter we examine ways of representing linear constraints. The goal is to write.

The constraints in a form that makes it easy to move from one feasible point to another.

The constraints specify interrelationships among the variables so that, for example, if we increase the first variable, retaining feasibility might require making a complicated sequence of changes to all the other variables. It is much easier if we express the constraints using acoordinate system that is “natural” for the constraints. Then the interrelationships among the variables are taken care of by the coordinate system, and moves between feasible pointsare almost as simple as for a problem without constraints.

In the general case these constraints may be either equalities or inequalities. Since any inequality of the “less than or equal” type may be transformed to an equivalent constraint of the “greater or equal” type, any problem with linear constraints may be written as follows:

minimize $f\ (x)$

subject to $a_i^T x = b_i\ ,\quad i \in \varepsilon$

$a_i^T x \geq b_i\ ,\quad i \in \tau$

Each a_i here is a vector of length n and each b_i is a scalar. ε is an index set for the equality constraints and τ is an index set for the inequality constraints. We denote by A the matrix whose rows are the vectors a_i^T and denote by b the vector of right-hand side coefficients b_i .

Let S be the set of feasible points. A set of this form, defined by a finite number of linear constraints, is sometimes called a *polyhedron* or a *polyhedral set*. In this chapter we are notconcerned with the properties of the objective function f .

Example 4.1 (Problem with Linear Constraints)**.** Consider the problem

minimize $f\,(x) = x_1^2 + x_2^3 x_3^4$

subject to $x_1 + 2x_2 + 3x_3 = 6$

$$x_1, x_2\,, x_3 \geq 0.$$

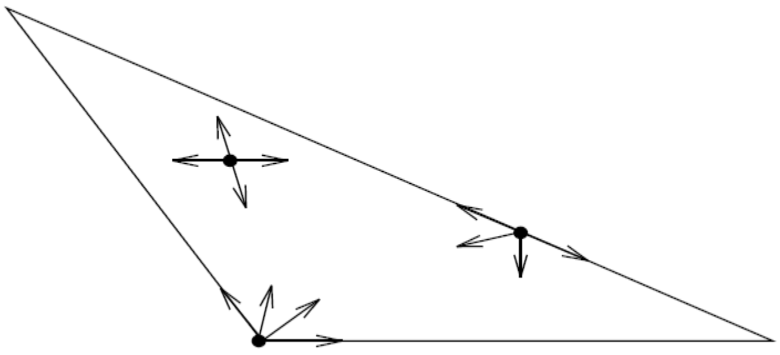

Figure 4.1. *Feasible directions.*

For this example $\varepsilon = \{1\}\, and\, \tau = \{2, 3, 4\}$. The vectors $\{a_i\}$ that determine the constraints

are

$$a_1 = (1\;2\;3)^T\,, a_2 = (1\;0\;0)^T$$

$$a\,3 = (0\;1\;0)^T, a\,4 = (0\;0\;1)^T$$

and the right-hand sides are

$$b_1 = 6, \quad b_2 = 0, \qquad b_3 = 0, and \qquad b_4 = 0.$$

We start by taking a closer look at the relation between a feasible point and its neighboring feasible points. We shall be interested in determining how the function value changes as we move from a feasible point $\bar{x}$ to nearby feasible points.

First let us look at the direction of movement. We define p to be a *feasible direction* at the point $\bar{x}$ if a small step taken along p leads to a feasible point in the set. Mathematically, p is a feasible direction if there exists some $\epsilon > 0$ such that $\bar{x} + \alpha p \in S$ for all $0 \leq \alpha \leq \epsilon$.

Thus, a small movement from $\bar{x}$ along a feasible direction maintains feasibility. In addition, since the feasible set is convex, any feasible point in the set can be reached from $\bar{x}$ by moving along some feasible direction. Examples of feasible directions are shown in Figure 4.1.

In many applications, it is useful to maintain feasibility at every iteration. For example, the objective function may only be defined at feasible points. Or if the algorithm is terminated before an optimal solution has been found, only a feasible point may have practical value. These considerations motivate a class of methods called *feasible-point methods*. These methods have the following form.

Algorithm 4.1.

Feasible-Point Method

1. Specify some initial feasible guess of the solution x_0

2. For $k = 0, 1, \ldots$

(i) Determine a feasible direction of descent p_k at the point x_k. If none exists, stop.

(ii) Determine a new feasible estimate of the solution: $x_{K+1} = x_K + a_K p_K$, where

$$f\ (x_{K+1}) \ < \ f(x_K).$$

In this chapter we are mainly concerned with representing feasible directions with respect to S in terms of the constraint vectors a_i . We begin by characterizing feasible directions with respect to a single constraint. Specifically, we determine conditions that ensure that small movements away from a feasible point $\bar{x}$ will keep the constraint satisfied.

Consider first an equality constraint $a_i^T x = b_i$. Let us examine the effect of taking asmall positive step α in the direction p . Since $a_i^T \bar{x} = b_i$, then $a_i^T\ (\bar{x} + \alpha p) = b_i$ will hold if

and only if $a_i^T p = 0$.

Example 4.2 (An Equality Constraint)**.** Suppose that we wished to solve

$$minimize\ \ f\ (x_1, x_2)$$

$$subject\ to\ \ x_1 + x_2 = 1.$$

For this constraint $a_1 = (1,1)^T$ and $b_1 = 1$. Let $\bar{x} = (0,1)^T$ so that $\bar{x}$ satisfies the constraint.

Then $\bar{x} + \alpha p$ will satisfy the constraint if and only if $a_1^T p = 0$, that is,

$$p_1 + p_2 = 0.$$

For this example

$$a_1^T(\bar{x} + \alpha p) = (\bar{x}_1 + \bar{x}_2) + \alpha(p_1 + p_2) = (1) + \alpha(0) = 1,$$

as expected.

The original problem is equivalent to

$$minimize_{\alpha}\ f\ (\bar{x} + \alpha p)\ \ ,$$

where $\bar{x} = (0,1)^T$, as before, and where $p = (1,-1)^T$ is a vector satisfying $a_1^T p = 0$.

Expressing feasible points in the form $\bar{x} + \alpha p$ will be a way for us to transform constrained problems to equivalent problems without constraints.

Continuing to inequality constraints, consider first some constraint $a_i^T x \geq bi$ which

is inactive at $\bar{x}$. Since $a_i^T \bar{x} > bi$, then $a_i^T (\bar{x} + \alpha p) > bi$ for all α sufficiently small. Thus, we can move a small distance in any direction p without violating the constraint.

If the inequality constraint is active at $\bar{x}$, we have $a_i^T \bar{x} = b_i$. Then to guarantee that

$a_i^T (\bar{x} + \alpha p) \geq b_i$ for small positive step lengths α, the direction p must satisfy $a_i^T p \geq 0$.

Example 4.3 (An Inequality Constraint)**.** Suppose that we wished to solve

$$minimize \;\; f\,(x_1, x_2)$$

$$subject\ to \;\; x_1 + x_2 \geq 1.$$

For this constraint $a_1 = (1,1)^T$ and $b_1 = 1$. If $\bar{x} = (0,2)^T$, then the constraint is inactive and any nearby point is feasible.

If $\bar{x} = (0,1)^T$, then the constraint is active and nearby points can be expressed in the form $\bar{x} + \alpha p$ with $a_1^T p \geq 0$. For this example this corresponds to the condition $p_1 + p_2 \geq 0$, or $p_1 \geq -p_2$

In summary, we conclude that the feasible directions at a point $\bar{x}$ are determined by the equality constraints and the active inequalities at that point. Let $\hat{\tau}$ denote the set of active inequality constraints at $\bar{x}$. Then p is a feasible direction with respect to the feasible set at $\bar{x}$ if and only if

$$a_i^T p = 0, i \in \varepsilon, \qquad a_i^T p \geq 0, i \in \hat{\tau}\,.$$

In the following, it will be convenient to consider separately problems that have only equality constraints, or only inequality constraints.

The general form of the *equality-constrained problem* is

$$minimize \quad f\ (x)$$

$$subject\ to \quad Ax\ =\ b.$$

It is evident from our discussion above that a vector p is a feasible direction for the linear equality constraints if and only if

$$Ap\ =\ 0.$$

We call the set of all vectors p such that $Ap\ =\ 0$ the *null space* of A. A direction p is a feasible direction for the linear equality constraints if and only if it lies in the null space of A.

The general form of the *inequality-constrained problem* is

$$minimize\ f\ (x)$$

$$subject\ to\ Ax\ \geq\ b.$$

Let $\bar{x}$ be a feasible point for this problem. We have observed already that the inactive constraints at $\bar{x}$ do not influence the feasible directions at this point. Let $\hat{A}$ be the submatrix of A corresponding to the rows of the active constraints at $\bar{x}$. Then a direction p is a feasible direction for S at $\bar{x}$ if and only if

$\hat{A}p\ \geq\ 0.$

Since the inactive constraints at a point have no impact on its feasible directions, such constraints can be ignored when testing whether the point is locally optimal. In particular, if we had prior knowledge of which constraints are active at the optimum, we could cast aside the inactive constraints and treat the active constraints as equalities. A solution of the inequality-constrained problem is a solution of the equality-constrained problem defined by the active constraints.

The theory for inequality-constrained problems draws on the theory for equality-constrained problems. For this reason, it is important to study problems with only equality constraints. In particular, it will be useful to study ways to represent all the vectors in the null space of a matrix.

Once a feasible direction p is determined, the new estimate of the solution is of the form $\bar{x} + \alpha p$ where $\alpha \geq 0$. Since the new point must be feasible, in general there is an upper limit on how large α can be.

For an equality constraint we have $a_i^T p = 0$, and so

$$a_i^T (\bar{x} + \alpha p) = a_i^T \bar{x} = bi$$

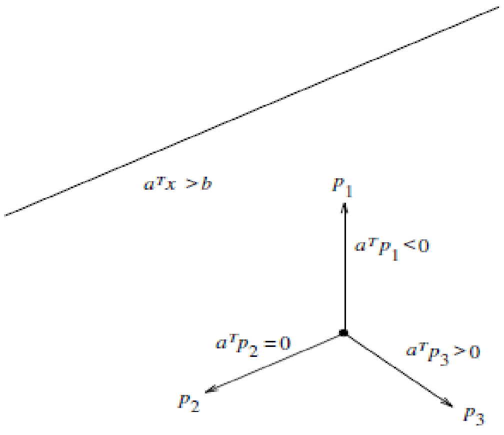

Figure 4.2. *Movement to and away from the boundary.*

for *all* values of α . For an active inequality constraint we have $a_i^T p \geq 0$, and so

$$a_i^T (\bar{x} + \alpha p) \geq a_1^T \bar{x} \geq b_i$$

for all values of $\alpha \geq 0$. Thus only the inactive constraints are relevant when determining an upper bound on α .

Because $\bar{x}$ is feasible, $a_i^T\,\bar{x} > bi$ for all inactive constraints. Thus, if $a_i^T\,p \geq 0$, the

constraint remains satisfied for all $\alpha \geq 0$. As α increases, the movement is *away* from the boundary of the constraint. On the other hand, if $a_i^T p < 0$, the inequality will remain valid only if $\alpha \leq (a_i^T\,\bar{x} - bi)/(-a_i^T\,p)$. Apositive step along p is a move towards the boundary, and any step larger than this bound will violate the constraint. (See Figure 4.2.) The maximum step length $\bar{\alpha}$ that maintains feasibility is obtained from a *ratio test*:

$$\bar{\alpha} = min\,\{(a_i^T\,\bar{x} - bi)/(-a_i^T\,p) : a_i^T p < 0\},$$

where the minimum is taken over all inactive constraints. If $a_i^T\,p \geq 0$ for all inactive constraints, then an arbitrarily large step can be taken without violating feasibility.

Example 4.4 (Ratio Test)**.** *Let* $\bar{x} = (1, 1)^T$ and $p = (4, -2)^T$. Suppose that there are three inactive constraints with

$$a_1^T = (1\ 4) \quad and \quad b_1 = 3$$

$$a_2^T = (0\ 3) \quad and \quad b_2 = 2$$

$$a_3^T = (5\ 1) \quad and \quad b_3 = 4.$$

Then

$$a_1^T p = -4 < 0, \quad a_2^T p = -6 < 0, \quad and \quad a_3^T p = 18 > 0,$$

so only the first two constraints are used in the ratio test:

$$\bar{\alpha} = min\,\{(a_i^T\,\bar{x} - b_i)/(-a_i^T\,p) : a_i^T p < 0\}$$

$$= min\,\{\,(5 - 3)/4, (3 - 2)/6\,\} = 1/6.$$

Notice that the point $\bar{x} + \bar{\alpha}p = (\frac{5}{3}, \frac{2}{3})^T$ is on the boundary of the second constrain

4.0.2 Null and Range Spaces:

Let A be an $m \times n$ matrix with $m \leq n$. We denote the *null space* of A by

$$N(A) = \{p \in \Re^n : Ap = 0\}.$$

The null space of a matrix is the set of vectors orthogonal to the rows of the matrix. Recall that the null space represents the set of feasible directions for the constraints $Ax = b$. It is easy to see that any linear combination of two vectors in $N(A)$ is also in $N(A)$, and thus the null space is a subspace of $\Re^n$. It can be shown that the dimension of this subspace is $n - rank(A)$. When A has full row rank (i.e., its rows are linearly independent), this is just

$n - m.$

Another term that will be important to our discussions is the *range space* of a matrix.

This is the set of vectors spanned by the columns of the matrix (that is, the set of all linear combinations of these columns). In particular, we are interested in the range space of A^T,

defined by

$$R(A^T) = \{q \in \Re^n : q = A^T\lambda \, for\ some\ \lambda \in \Re^m\}.$$

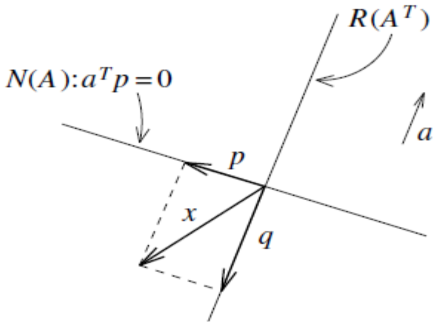

Figure 4.3. *Null space and range space of* $A = (a^T)$.

Throughout this text, if we mention a range space without specifying a matrix, it refers to the range space of A^T. The dimension of the range space is the same as the rank of A^T, orequivalently the rank of A.

There is an important relationship between $N(A)$ and $R(A^T)$: they are *orthogonal subspaces*. This means that any vector in one subspace is orthogonal to any vector in theother. To verify this statement, we note that any vector $q \in RA^T)$ can be expressed as $q = A^T\lambda$ for some $\lambda \in \Re^m$, and therefore, for any vector $p \in N(A)$ we have

$$q^T p = \lambda^T A p = 0.$$

There is more. Because the null and range spaces are orthogonal subspaces whose dimensions sum to n, any $n - dimensional$ vector x can be written uniquely as the sum of a null-space and a range-space component:

$$x = p + q,$$

where $p \in N(A)$ and $q \in R(A^T)$. Figure 3.5 illustrates the null and range spaces for $A = (a^T)$, where a is a two-dimensional nonzero vector. Notice that the vector a is orthogonal to the null space and that any range-space vector is a scalar multiple of a. The decomposition of a vector *x* into null-space and range-space components is also shown in

Figure 4.3.

How can we represent vectors in the null space of A? For this purpose, we define a matrix Z to be a *null-space matrix* for A if any vector in $N(A)$ can be expressed as a linear combination of the columns of Z. The representation of a null-space matrix is not unique.

If A has full row $rank\, m$, any matrix Z of dimension $n \times r$ and rank $n - m$ that satisfies

$AZ = 0$ is a null-space matrix. The column dimension r must be at least $(n - m)$. In the special case where r is equal to $n - m$, the columns of Z are linearly independent, and Z is then called a *basis* matrix for the null space of A. If Z is an $n \times r$ null-space matrix, the null space can be represented as

$$N(A) = \{ p : p = Zv \quad for\, some\, v \in \Re^r \},$$

thus $N(A) = R(Z)$. This representation of the null space gives us a practical way to generate feasible points. If $\bar{x}$ is any point satisfying $Ax = b$, then all other feasible points can be written as

$$x = \bar{x} + Zv$$

for some vector v.

As an example consider the rank-two matrix

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

The null space of A is the set of all vectors p such that

$$Ap = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix} = \begin{pmatrix} p_1 - p_2 \\ p_3 + p_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

that is, the vector must satisfy $p_1 = p_2$ and $p_3 = -p_4$. Thus any null-space vector must

have the form

$$p = \begin{pmatrix} v_1 \\ v_1 \\ v_2 \\ -v_2 \end{pmatrix}$$

for some scalars v_1 and v_2. A possible basis matrix for the null space of A is

$$Z = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix}$$

and the null space can be expressed as

$$N(A) = \{p : p = Zv \, for \, some \, v \in \Re^2\} .$$

The matrix

$$\bar{Z} = \begin{pmatrix} 1 & 0 & 2 \\ 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

is also a null-space matrix for A, but it is not a basis matrix since its third column is a linear combination of the first two columns. The null space of A can be expressed in terms of $\bar{z}$

as

$$N(A) = \{p : p = \bar{Z}\bar{v} \quad for \, some \, \bar{v} \in \Re^3 \}.$$

4.0.3 Generating Null-Space Matrices:

We present here four commonly used methods for deriving a null-space matrix for A. Thediscussion assumes that A is an $m \times n$ matrix of full row rank (and hence $m \leq n$). Two of the approaches, the variable reduction method and the QR factorization, yield an $n \times (n - m)$ basis matrix for $N(A)$. The other two methods yield an $n \times n$ null-space matrix.

4.0.3.1 Variable Reduction Method:

This method is the approach used by the simplex algorithm for linear programming. It is also used in nonlinear optimization. We start with an example.

Consider the linear system of equations:

$$p_1 + p_2 - p_3 = 0$$

$$-2p_2 + p_3 = 0.$$

This system has the form $Ap = 0$. We wish to generate all solutions to this system.

We can solve for any two variables whose associated columns in A are linearly independent in terms of the third variable. For example, we can solve for p_1 and p_3 in terms of

p_2 as follows:

$$p_1 = p_2$$

$$p_3 = 2p_2.$$

The set of all solutions to the system can be written as

$$p = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} p_2,$$

where p_2 is chosen arbitrarily. Thus $Z = (1, 1, 2)^T$ is a basis for the null space of A.

Since the values of p_1 and p_3 depend on p_2, they are called *dependent variables*.

They are also sometimes called *basic* variables. The variable p_2 which can take on any value is called an *independent variable*, or a *nonbasic variable*.

To generalize this, consider the $m \times n$ system $Ap = 0$. Select any set of m variables whose corresponding columns are linearly independent—these will be the basic variables.

Denote by B the $m \times m$ matrix defined by these columns. The remaining variables will be the nonbasic variables; we denote the $m \times (n - m)$ matrix of their respective columns by N. The general solution to the system $Ap = 0$ is obtained by expressing the basic variables in terms of the nonbasic variables, where the nonbasic variables can take on any arbitrary value.

For ease of notation we assume here that the first m variables are the basic variables.

Thus

$$Ap = (B \quad N)\begin{pmatrix} p_B \\ p_N \end{pmatrix} = Bp_B + Np_N = 0.$$

Premultiplying the last equation by B^{-1}we get

$p_B = -B^{-1}Np_N$.

Thus the set of solutions to the system $Ap = 0$ is

$$p = \begin{pmatrix} p_B \\ p_N \end{pmatrix} = \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix} p_N$$

and the $n \times (n - m)$ matrix

$$Z = \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix}$$

is a basis for the null space of A.

Consider now the system $Ax = b$. One feasible solution is

$$\bar{x} = \begin{pmatrix} -B^{-1}b \\ 0 \end{pmatrix}$$

If x is *any* point that satisfies $Ax = b$, then x can be written in the form

$$x = \bar{x} + p = \bar{x} + Zp_N = \begin{pmatrix} -B^{-1}b \\ 0 \end{pmatrix} + \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix} p_N .$$

If the basis matrix B is chosen differently, then the representation of the feasible points changes, but the set of feasible points does not.

In this derivation we assumed that the first m variables were the basic variables. If this is not true, the rows in Z must be reordered to correspond to the ordering of the basic and nonbasic variables. This technique is illustrated in the following example.

Example 4.5 (Variable Reduction)**.** Consider the system of constraints $Ax = b$ with

$$A = \begin{pmatrix} 1 & -2 & 1 & 3 \\ 0 & 1 & 1 & 4 \end{pmatrix} \text{ and } b = \begin{pmatrix} 5 \\ 6 \end{pmatrix}.$$

Let B consist of the first two columns of A, and let N consist of the last two columns:

$$B = \begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix} \quad \text{and } N = \begin{pmatrix} 1 & 3 \\ 1 & 4 \end{pmatrix}.$$

Then

$$\bar{x} = \begin{pmatrix} -B^{-1}b \\ 0 \end{pmatrix} = \begin{pmatrix} 17 \\ 6 \\ 0 \\ 0 \end{pmatrix}$$

and

$$Z = \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix} = \begin{pmatrix} -3 & -11 \\ -1 & -4 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

It is easy to verify that $A\bar{x} = b$ and $AZ = 0$. Every point satisfying $Ap = 0$ is of the form

$$Zp_N = \begin{pmatrix} -3 & -11 \\ -1 & -4 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p_3 \\ p_4 \end{pmatrix} = \begin{pmatrix} -3p_3 - 11p_4 \\ -p_3 - 4p_4 \\ p_3 \\ p_4 \end{pmatrix}.$$

If instead B is chosen as columns 4 and 3 of A (in that order), and N as columns 2

and 1, then

$$B = \begin{pmatrix} 3 & 1 \\ 4 & 1 \end{pmatrix} \; and \; N = \begin{pmatrix} -2 & 1 \\ 1 & 0 \end{pmatrix} .$$

Care must be taken in defining $\bar{x}$ and Z to ensure that their components are positioned correctly. In this case

$$B^{-1}b = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ and } \bar{x} = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix}.$$

Notice that the components of $B^{-1}b$ are at positions 4 and 3 in $\bar{x}$, corresponding to the columns of A that were used to define B. Similarly

$$-B^{-1}N = \begin{pmatrix} -3 & 1 \\ 11 & 4 \end{pmatrix} \text{ and } Z = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 11 & -4 \\ -3 & 1 \end{pmatrix}$$

The rows of $-B^{-1}N$ are placed in rows 4 and 3 of Z, and the rows of I are placed in rows

2 and 1. As before, $A\bar{x} = b$ and $AZ = 0$. Every point satisfying $Ap = 0$ is of the form

$$Zp_N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 11 & -4 \\ -3 & 1 \end{pmatrix} \begin{pmatrix} p_2 \\ p_1 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \\ 11p_2 - 4p_1 \\ -3p_2 + p_1 \end{pmatrix}$$

In practice the matrix Z itself is rarely formed explicitly, since the inverse of B should not be computed. This is not a limitation; Z is only needed to provide matrix-vector products of the form $p = Z_v$, or the form $Z^T g$. These computations do not require Z explicitly.

For example, the vector $p = Z_v$ may be computed as follows. First we compute $t = Nv$.

Next we compute $u = -B^{-1}t$, by solving the system $Bu = -t$. (This should be done via a numerically stable method such as the LU factorization.) The vector $p = Zv$ is now given

by $p = (u^T, v^T)^T$.

The variable reduction approach for representing the null space is the method used in the simplex algorithm for linear programming. This approach has been enhanced so that ever larger problems can be solved. These enhancements exploit the sparsity that is oftenpresent in large problems, in order to reduce computational effort and increase accuracy.

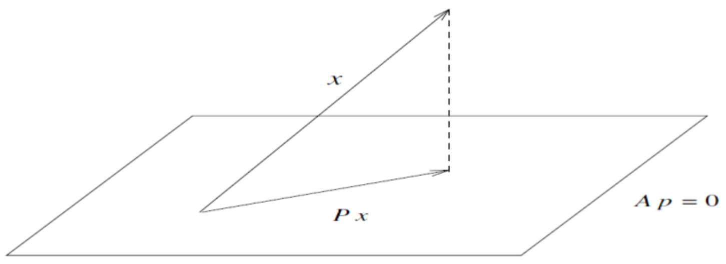

Figure 4.4. *Orthogonal projection.*

4.0.3.2 Orthogonal Projection Matrix:

Let x be an n −dimensional vector, and let A be an $m \times n$ matrix of full row rank. Then x can be expressed as a sum of two components, one in $N(A)$ and the other in $R(A^T)$:

$$x = p + q,$$

where $Ap = 0$, and $q = A^T\lambda$ for some m −dimensional vector λ. Multiplying this equation on the left by A gives $Ax = AA^T\lambda$, from which we obtain $\lambda = (AA^T)^{-1}Ax$. Substituting for q gives the null-space component of x:

$$p = x - A^T(AA^T)^{-1}Ax = (I - A^T(AA^T)^{-1}A)x.$$

The $n \times n$ matrix

$$P = I - A^T(AA^T)^{-1}A$$

is called an *orthogonal projection matrix* into $N(A)$. The null-space component of the vector x can be found by premultiplying x by P; the resulting vector Px is also termed the orthogonal projection of x onto $N(A)$ (see Figure 4.4).

The orthogonal projection matrix is the unique matrix with the following properties:

• It is a null-space matrix for A;

• $P^2 = P$, which means repeated application of the orthogonal projection has no further effect;

• $P^T = P\ (P\ is\ symmetric)$.

The name "orthogonal projection" may be misleading—unless P is the identity matrix it is *not* orthogonal.

There are a number of ways to compute the projection matrix. Selection of the method depends in general on the application, the size of $m\ and\ n$, as well as the sparsity of A. We point out that by "computing the matrix" we mean representing the matrix so that a matrixvector product of the form Px can be formed for any vector x. The projection matrix itself is rarely formed explicitly.

To demonstrate this point, suppose that A consists of a single row: $A = a^T$, where a is an $n - vector$. Then

$$P = I - \frac{1}{a^T a} aa^T.$$

Forming P explicitly would require approximately $n^2/2$ multiplications and $n^2/2$ storage locations. Forming the product Px for some vector x would require n^2 additional multiplications. These costs can be reduced dramatically if only the vector a and the scalar $a^T a$ are stored. "Forming" P this way only requires n multiplications in the calculation of *(aa^T)*. The matrix-vector product is computed as $Px = x - a(a^T x)/(a^T a)$. This requires only $2n$ multiplications.

In the example above the matrix AA^T is the scalar aTa, which is easy to invert. In the more general case where A has several rows, the task of "inverting" AA^T becomes expensive, and care must be taken to perform this in a numerically stable manner. Often, this is done by the Cholesky factorization. However, if A is dense it is not advisable to form the matrix AA^T explicitly, since it can be shown that its condition number is the square of that of A. A more stable approach is to use a *QR* factorization of A^T.

For the case when A is large and sparse, the *QR* factorization may be too expensive, since it tends to produce dense factors. Special techniques that attempt to exploit the sparsity structure of A have been developed for this situation.

4.0.3.3 Other Projections:

As before, let A be an $m \times n$ matrix of full row rank. Let D be a positive-definite $n \times n$ matrix, and consider the $n \times n$ matrix

$$P_D = I - DA^T(ADA^T)^{-1}A.$$

It is easy to show that P_D is a null-space matrix for A. Also, $P_D P_D = P_D$. An $n \times n$ matrix with these two properties is called a *projection matrix*. An orthogonal projection is therefore a symmetric projection matrix.

Many of the new interior point algorithms for optimization use projections of this form. In the case of linear programming, the matrix D is generally a diagonal matrix with positive diagonal terms. This matrix D changes from iteration to iteration, while A remains unchanged. Special techniques for computing and updating these projections have been developed.

4.0.3.4 The *QR* Factorization:

Again let A be an $m \times n$ matrix with full row rank. We perform an orthogonal factorization of A^T :

$$A^T = QR.$$

Let $Q = (Q_1, Q_2)$, where Q_1 consists of the first m columns of Q, and Q_2 consists of the last $n - m$ columns. Also denote the top $m \times m$ triangular submatrix of R by R_1. The rest of R is an $(n - m) \times m$ zero matrix. Since Q is an orthogonal matrix, it follows that

$$AQ = R^T, \text{ or}$$

$$AQ_1 = R_1^T \text{ and } AQ_2 = 0.$$

Thus

$$Z = Q_2$$

is a basis for the null space of A. This basis is also known as an *orthogonal basis*, since

$$Z^T Z = I .$$

Example 4.6 (Generating a Basis Matrix Using the *QR* Factorization)**.** Consider the matrix

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

An orthogonal factorization of A^T yields

$$Q = \begin{pmatrix} \frac{-\sqrt{2}}{2} & 0 & \frac{-1}{2} & \frac{-1}{2} \\ \frac{\sqrt{2}}{2} & 0 & \frac{-1}{2} & \frac{-1}{2} \\ 0 & \frac{-\sqrt{2}}{2} & \frac{1}{2} & \frac{-1}{2} \\ 0 & \frac{-\sqrt{2}}{2} & \frac{-1}{2} & \frac{1}{2} \end{pmatrix}$$

is a basis for the null space of A.

The *QR* factorization method has the important advantage that the basis Z can be formed in a numerically stable manner. Moreover, computations performed with respect to the resulting basis Z are numerically stable. However, this numerical stability comes at a price, since computing the *QR* factorization is relatively expensive. If m is small relative to n, some savings may be gained by not forming Q explicitly. An additional drawback of the *QR* method is that the basis Z can be dense even when A is sparse. As a result it may be unsuitable for large sparse problems

4.0.3 The Chain Rule:

The rule for obtaining the derivative of a function of a function is called the *chain rule*.

Consider a function $g(x) = g(x_1, \ldots, x_n)$, and suppose that each x_i is in turn a function

of the variables $t_1, \ldots, t_m$; that is, $x_i = x_i\,(t_1, \ldots, t_m)$ for $i = 1, \ldots, n$. We examine the composite function

$$h(t) = g(x(t\,)).$$

The chain rule states that if g is continuously differentiable in $\Re^n$, and $x_1, \ldots, x_m$ are continuously differentiable in $\Re^m$, then h is continuously differentiable in $\Re^m$ and

$$\nabla h(t) = \nabla x(t) \nabla g(x(t)),$$

where

$$\nabla x(t) = (\nabla x_1(t) \cdot \cdot \cdot \nabla x_n(t)).$$

The chain rule can be generalized to the case where g is a k-dimensional vector of functions g_i. In this case h will also be a k-dimensional vector of functions. If ∇h denotes the $n \times k$ matrix whose $j\,th$ column is ∇h_i , and ∇g denotes the $n \times k$ matrix whose $j\,th$ column is ∇g_i , then the above formula remains valid.

Example 4.7 (Chain Rule)**.** Suppose that

$$g_1(x) = x_1^2 - x_1 x_2$$

$$g_2(x) = -x_1^4 + 2x_2^2 ,$$

where

$$x_1 = x_1(t_1, t_2, t_3) = t_1 + 2t_2 - 3t_3$$

$$x_2 = x_2(t_1, t_2, t_3) = t_1^2 + t_2$$

and let $h(t) = g(x(t))$. Then

$$\nabla x(t) = \begin{pmatrix} 1 & 2t_1 \\ 2 & 1 \\ -3 & 0 \end{pmatrix}$$

and

$$\nabla g(x(t)) = \begin{pmatrix} 2x_1(t) - x_2(t) & -4x_1^3(t) \\ -x_1(t) & 4x_2(t) \end{pmatrix}$$

$$= \begin{pmatrix} 2(t_1 + 2t_2 - 3t_3) - (t_1^2 + t_2) & -4(t_1 + 2t_2 - 3t_3)^3 \\ -(t_1 + 2t_2 - 3t_3) & 4(t_1^2 + t_2) \end{pmatrix}$$

hence

$$\nabla h(t) = \begin{pmatrix} 1 & 2t_1 \\ 2 & 1 \\ -3 & 0 \end{pmatrix} \begin{pmatrix} 2(t_1 + 2t_2 - 3t_3) - (t_1^2 + t_2) & -4(t_1 + 2t_2 - 3t_3)^3 \\ -(t_1 + 2t_2 - 3t_3) & 4(t_1^2 + t_2) \end{pmatrix}$$

A particular application of the chain rule is if

$$x = \begin{pmatrix} y(t) \\ t \end{pmatrix}$$

If $h(t) = g(x(t))$, then

$$\nabla h(t) = \nabla x(t) \nabla g(x(t)) = (\nabla y(t) \quad I) \begin{pmatrix} \nabla_y g(x(t)) \\ \nabla_t g(x(t)) \end{pmatrix}$$

$$= \nabla y(t) \nabla_y g(y(t), t) + \nabla_t g(y(t), t).$$

Note that ∇_t refers to the gradient of a function with respect to the vector of variables t .

The chain rule can also be used to obtain second derivatives. We will assume here that g is a scalar function. If g and x_i are twice continuously differentiable, then h is twice continuously differentiable in $\Re^m$ and

$$\nabla^2 h(t) = \nabla^2 x(t) \nabla g(x(t)) + \nabla x(t) \nabla^2 g(x(t)) \nabla x(t)^T,$$

where a product of the form $(\nabla^2 x)v$ is interpreted as

$$(\nabla^2 x)v = \sum_{i=1}^{n} (\nabla^2 x_i\ (t)) v_i$$

4-1 Introduction:

In this part we study techniques for solving nonlinear optimization problems. We concentrate on problems that can be written in the general form

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g_i(x) = 0\ ,\ i \in \varepsilon \end{aligned}$$

$$g_i(x) \geq 0 \,, i \in \mathcal{T}.$$

Here ε is an index set for the equality constraints and $\mathcal{T}$ is an index set for the inequality constraints.

We assume that the objective function f and the constraint functions g_i are twice continuously differentiable.

Hear we study the conditions satisfied by solutions to the constrained optimization problem. We shall focus only on local solutions, for the same reasons as in the unconstrained case. In the case of convex problems, that is, when the feasible region is convex and f is a convex function, any local solution is also a global solution.

In the unconstrained case the optimality conditions were derived by using a Taylor series approximation to examine the behavior of the objective function f about a local minimizer x_*. In particular, at points "near" x_* the value of f does not decrease.

A similar approach is used in the constrained case. Taylor series approximations are used to analyze the behavior of the objective f and the constraints g_i about a local constrained minimizer x_*. In this case, at *feasible* points "near" x_* the value of f does not decrease.

The optimality conditions will be derived in stages, first for problems with linear constraints, and then for problems with nonlinear constraints. The intuition in both cases is similar, but is easier to comprehend when the constraints are linear. In the nonlinear case the details are more complicated and can disguise the basic ideas involved.

If all the constraints are linear, feasible movements are completely characterized by feasible directions. (See Section 4.1.) At a local minimizer there can be no feasible directions of descent for f , hence

$$p^T \nabla f(x_*) \geq 0 \text{ for all feasible directions } p \text{ at } x_*. \quad (4.1)$$

The first-order optimality condition is a direct result of this statement.

If the problem has nonlinear constraints, it may no longer be possible to move to nearby points along feasible directions. Instead, movements will be made along feasible *curves*. Analyzing movement along curves is more complicated than along directions, and more complicated situations can arise. Even so, the basic idea is that the objective value will not decrease at feasible points near x_*.

Some new concepts arise in the constrained case, in particular, the Lagrange multipliers and the Lagrangian function. The Lagrange multipliers are analogous to the dual variables in linear optimization. The Lagrangian is a single function that combines the objective and constraint functions; it plays a central role in the theory and algorithms of constrained optimization.

4.2 Optimality Conditions for Linear Equality Constraints:

In this section we discuss the optimality conditions for nonlinear problems where all constraints are linear equalities:

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad Ax = b \,, \end{aligned}$$

where A is an $m \times n$ matrix. We assume that f is twice continuously differentiable overthe feasible region. We also assume that the rows of A are linearly independent, that is, Ahas full row rank. This is not an unduly restrictive assumption since in theory, if a problemis consistent, we can discard any redundant constraints.

The main idea is to transform this constrained problem into an equivalent unconstrained problem. The theory and methods for unconstrained optimization can then beapplied to the new problem.

To demonstrate the approach consider the problem

$$\text{minimize} \quad f(x) = x_1^2 - 2x_1 - x_2^2 + 4x_3$$

$$\text{subject to} \quad x_1 - x_2 + 2x_3 = 2\ .$$

At any feasible point, the variable x_1 can be expressed in terms of x_2 and x_3 using $x_1 = 2 + x_2 - 2x_3$. Substituting this into the formula for $f(x)$, we obtain the equivalent unconstrained problem

$$\text{minimize} \quad 2x_2^2 + 3x_3^3 - 4x_2x_3 + 2x_2.$$

(The number of variables has been reduced from three to two.) It is easy to verify that a strict local minimizer to the unconstrained problem is $x_2 = -1.5, x_3 = -1$. The solution to the original problem is $x_* = (2.5\,, -1.5\,, -1)^T$with an optimal objective value of $f(x_*) = -1.5$.

Any problem with linear equality constraints $Ax = b$ can be recast as an equivalent unconstrained problem. Suppose we have a feasible point $\bar{x}$, that is, $A\bar{x} = b$. Then any otherfeasible point can be expressed as $x = \bar{x} + p$, where p is a feasible direction. Any feasible direction must lie in the null space of A , the set of vectors p satisfying $Ap = 0$. Denoting this null space by $N(A)$, the feasible region can be described by $\{x\colon x = \bar{x} + p, p \in N(A)\}$.

Let Z be an $n \times r$ null-space matrix for A (with $r \geq n - m$). Then the feasible region is given by $\{\, x : x = \bar{x} + Zv, \text{where } v \in \Re^r \,\}$. Consequently, our constrained problem in x is equivalent to the unconstrained problem

$$\underset{v \in \Re^r}{\text{minimize}} \ \phi(v) = f(\bar{x} + Zv)\ .$$

The function φ is the restriction of f onto the feasible region; we shall refer to it as the reduced function.

If Z is a basis matrix for the null space of A, then ϕ will be a function of $n - m$ variables. Not only has the constrained problem been transformed into an unconstrained problem, but also the number of variables has been reduced as well.

Example 4.8 (Reduced Function)**.** Consider again the problem

$$\text{minimize} \quad f(x) = x_1^2 - 2x_1 + x_2^2 - x_3^2 + 4x_3$$

$$\text{subject to} \quad x_1 - x_2 + 2x_3 = 2.$$

Select

$$Z = \begin{pmatrix} 1 & -2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

as a null-space matrix for the constraint matrix $A = (1, -1, 2)$. Using the (arbitrary) feasible point $\bar{x} = (2, 0, 0)^T$, any feasible point can be written as

$$x = \bar{x} + Zv = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & -2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} v$$

for some $v = (v_1, v_2)^T$. Substituting into f , we obtain the reduced function $\phi(v) = 2v_1^2 + 3v_2^2 - 4v_1v_2 + 2v_1$.This is the same reduced function as before, except that now the variables are called v_1 and v_2 rather than x_2 and x_3.

The optimality conditions involve the derivatives of the reduced function. If $x = \bar{x} + Zv$, then by the chain rule ,

$$\nabla\phi(v) = z^T\nabla f(\bar{x} + Zv) = z^T\nabla f(x)$$

and

$$\nabla^2\phi(v) = z^T\nabla^2 f(\bar{x} + Zv)Z = z^T\nabla^2 f(x)Z.$$

The vector $\nabla\phi(v) = z^T\nabla f(x)$ is called the reduced gradient of fat x. If Z is an orthogonal projection matrix, it is sometimes called the projected gradient.

Similarly the matrix $\nabla^2\phi(v) = z^T\nabla^2 f(x)Z$ is called the reduced Hessian matrix, or sometimes the projected Hessian matrix. The reduced gradient and Hessian matrix are the gradient and Hessian of the restriction of f onto the feasible region, evaluated at x.

If x_* is a local solution of the constrained problem, then $x_* = \bar{x} + Zv_*$ for some v_*, and v_* is a local minimizer of ϕ. Hence $\nabla\phi(v_*) = 0$ and $\nabla^2\phi(v_*)$ is positive semidefinite.

Using the formulas for the reduced gradient and Hessian matrix, we obtain the first- and second-order necessary conditions for a local minimizer. They are summarized in the following lemma.

Lemma 4.1 (Necessary Conditions, Linear Equality Constraints)**.** If x_* is a local minimizer

of f over $\{ x : Ax = b \}$ and Z is a null-space matrix for A, then

- $z^T\nabla f(x_*) = 0$, *and*
- $z^T\nabla^2 f(x_*)Z$ *is positive semidefinite;*

that is, the reduced gradient is zero and the reduced Hessian matrix is positive semidefinite.

A point at which the reduced gradient is zero is a *stationary point*. Such a point may be a local minimizer of f , or a local maximizer, or neither, in which case it is a *saddle point*.

Second derivative information is used to distinguish local minimizers from other stationary points.

The second-order condition is equivalent to the condition

$$v^T z^T\nabla^2 f(x_*)Zv \geq 0 \quad \text{for all } v.$$

Observing that $p = Zv$ is a null-space vector, this can be rewritten as

$$p^T z^T \nabla^2 f(x_*)p \geq 0 \quad \text{for all } p \in N(A);$$

that is, the Hessian matrix at x_* must be positive semidefinite on the null space of A.

This condition does not require that the Hessian matrix itself be positive semidefinite. It is a less stringent requirement. If the Hessian matrix at x_* is positive semidefinite, however, then of course the second-order condition will be satisfied.

The second-order sufficiency conditions are also analogous to the unconstrained case.

We will assume that Z is a basis matrix for the null space of A, so that the columns of Z are linearly independent. The corresponding second-order sufficiency conditions are given in the lemma below.

Lemma 4.2 (Sufficient Conditions, Linear Equality Constraints)**.** *If* x_* *satisfies*

- $Ax_* = b,$
- $z^T \nabla f(x_*) = 0$*, and*
- $z^T \nabla^2 f(x_*) Z$ *is positive definite,*

where Z *is a basis matrix for the null space of* A*, then* x_* *is a strict local minimizer of* f *over* $\{ x : Ax = b \}$.

The following example illustrates the optimality conditions.

Example 4.9 (Necessary Conditions for Optimality)**.** We examine again the problem

$$\begin{array}{ll} \text{minimize} & f(x) = x_1^2 - 2x_1 + x_2^2 + 4x_3 \\ \text{subject to} & x_1 - x_2 + 2x_3 = 2 \end{array}$$

Since $\nabla f\,(x) = (2x_1 - 2, 2x_2, -2x_3 + 4)^T$ then at the feasible point $x_* = (2.5, -1.5, -1)^T$

the gradient of *f* is *(*3,–3, 6*)T*. Selecting

$$Z = \begin{pmatrix} 1 & -2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

as the null-space matrix of $A = (1, -1, 2)$, it is easily verified that $\mathrm{z}^{\mathrm{T}}\nabla \mathrm{f}(x_*) = (0, 0)^T$.

Thus, the reduced gradient vanishes at x_*, and the first-order necessary condition for a local

minimum is satisfied at this point. Checking the reduced Hessian matrix, we find that

$$\mathrm{z}^{\mathrm{T}}\nabla^2 \mathrm{f}(x_*)\mathrm{Z} = \begin{pmatrix} 1 & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix}\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -2 \end{pmatrix}\begin{pmatrix} 1 & -2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 4 & -4 \\ -4 & 6 \end{pmatrix}$$

The reduced Hessian matrix is positive definite at x_*. Hence the second-order sufficiency conditions are satisfied, and x_* is a strict local minimizer of f . Notice that $\nabla^2 \mathrm{f}(x_*)$ itself is not positive definite.

Let us choose some other feasible point, say $x = (2, 0, 0)^T$. The reduced gradient at this point is

$$\mathrm{z}^{\mathrm{T}}\nabla \mathrm{f}(\mathrm{x}) = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix};$$

hence this point is not a local minimizer. To move to a better point we should use a descent direction. Any vector $v = (v_1, v_2)^T$ such that $\mathrm{v}^{\mathrm{T}}(\mathrm{z}^{\mathrm{T}}\nabla \mathrm{f}(\mathrm{x})) = 2v_1 < 0$ will be a descent direction for the reduced function at this point. The corresponding direction $p = Zv$ will be a feasible descent direction for f .

Let us take another look at the first-order necessary condition. Let x_* be a local minimum, and let Z be any $n \times r$ null-space matrix for A . Breaking $\nabla f\ (x_*)$ into its null-space and range-space components gives

$$\nabla f\ (x_*) = Zv_* + A^T\lambda_*,$$

where v_* is in $\Re^r$ and λ_* is in $\Re^m$. Premultiplying by Z^T and recalling that the reduced gradient vanishes at x_*, we find that$Z^T Z v_* = 0$. This can occur only if $Z v_* = 0$, that is, if the null-space component of the gradient is zero. Therefore, if x_* is a local minimizer,

$$\nabla f\ (x_*)\ =\ A^T \lambda_* \tag{4.2}$$

for some *m*-vector λ_*. Thus, at a local minimum the gradient of the objective is a linear combination of the gradients of the constraints. The vector λ_* gives the coefficients of this linear combination. It is known as the vector of *Lagrange multipliers*. Its ith component is the Lagrange multiplier for the ith constraint.

The optimality conditions are demonstrated in Figure (4.5). This problem involves a single linear constraint $a^T x\ =\ b$. At the minimizer x_* the gradient is parallel to the vector a. Therefore there exists some number λ_* such that $\nabla f\ (x_*)\ =\ a\lambda_*$. On the other hand, at the point $\bar{x}$ the gradient is not parallel to the vector a; thus there is no λ that satisfies

$\nabla f\ (\bar{x})\ =\ a\lambda$, and the point is not optimal.

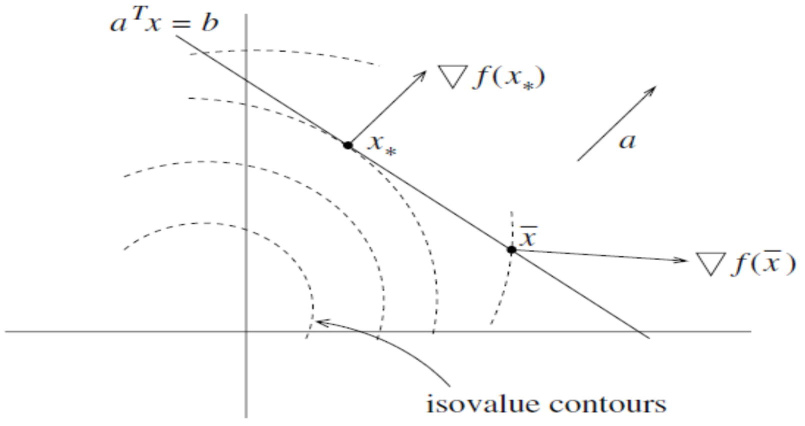

Figure 4.5. *Existence of Lagrange multipliers.*

Example 4.10 (Necessary Conditions for Optimality—Lagrange Multipliers)**.** Consider again the problem in Example 4.8. The first-order necessary condition is

$$\begin{pmatrix} 2x_1 - 2 \\ 2x_2 \\ -2x_3 + 4 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix} \lambda.$$

This implies that $x_1 = 1 + \lambda/2, x_2 = -\lambda/2,$ and $x_3 = 2 - \lambda$. Since the solution must be feasible, we substitute these values into the constraint $x_1 - x_2 + 2x_3 = 2$ to obtain $\lambda_* = 3$ asthe only solution. This indicates that $x_* = (2.5, -1.5, -1)^T$ is the unique stationary point.

Since we have seen that the second-order sufficiency conditions are satisfied at x_*, this isthe unique local solution.

In Example 4.10 we used condition (4.2) to obtain a local solution. In most cases,however, these equations will not have a closed-form solution. This is demonstrated in Example 4.11.

Example 4.11 (Intractability of the Optimality Conditions)**.** Consider the problem

$$\text{Minimize} \quad f(x) = x_1^4 x_2^2 + x_1^2 x_3^4 + \frac{1}{2} x_1^2 + x_1 x_2 + x_3$$

$$\text{subject to} x_1 + x_2 + x_3 = 1.$$

The first-order necessary condition implies that, at a local minimum,

$$\begin{pmatrix} 4x_1^3 x_2^2 + 2x_1 x_3^4 + x_1 + x_2 \\ 2x_1^4 x_2 + x_1 \\ 4x_1^2 x_3^3 + 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \lambda$$

for some number λ. These three equations together with the constraint $x_1 + x_2 + x_3 = 1$ give four equations in the four unknowns x_1, x_2, x_3, and λ. These equations are not easy to solve, however. If we try to solve them numerically, there is no guarantee that the solution will be a local minimizer; it may be a saddle point or even a local maximizer.

We have shown that if the reduced gradient is zero, then there exists a vector of Lagrange multipliers λ_* that satisfies the optimality condition (4.2).

The reverse is also true; that is, (4.2) implies that the reduced gradient vanishes. Thus, the two versions of the first-order optimality condition are equivalent. From a practical point of view there is a difference, however. If the reduced gradient at a given point is nonzero, it can be used to find a descent direction for the reduced function, and in turn for f. In contrast, the fact that Lagrange multipliers do not exist at a point does not assist in finding a better estimate of a solution.

Then why do we care about Lagrange multipliers? The Lagrange multipliers provide important information in sensitivity analysis. Furthermore, for problems with inequality constraints, estimates of the multipliers can indicate how to improve an estimate of the solution. Consequently, the two equivalent optimality conditions are used together in optimization software. A common procedure is to find a point x_* for which the reduced gradient is zero; at x_* condition (4.2) is consistent and the corresponding Lagrange multipliers can be computed.

Our derivation assumes that the matrix A has full row rank, that is, its rows are linearly independent. This assumption is called a *regularity assumption*. The results in this section can be extended to the case where the rows of A are linearly dependent, but then the vector of Lagrange multipliers will not generally be unique. For problems with nonlinear constraints, some assumption, such as a regularity assumption on the gradients of the constraints at the local minimum, is needed to state the optimality condition

4-3The Lagrange Multipliers and the Lagrange Function:

The Lagrange multipliers express the gradient at the optimum as a linear combination of the rows of the constraint matrix A. These multipliers have a significance which goes beyond this purely mathematical interpretation. In this section we shall see that they indicate the sensitivity of the optimal objective value to changes in the data. We also present the Lagrangian function and show how it can be used to express the optimality conditions in a concise way.

In most applications, only approximate data are available. Measurement errors, fluctuations in data, and unavailability of information are some of the factors that contribute to imprecision in the optimization model. In the absence of precise data, there may be no choice but to solve the problem using the best available estimates. Once a solution is obtained, the next step is to assess the quality of the resulting solution. A key question is, how sensitive is the solution to variations in the data?

Here we address this question for the particular case where small variations are made in the right-hand side of the constraints and investigate their effect on the optimal objective value. Our presentation will be informal. Amore formal proof is somewhat more complex.

We start with the problem

$$\begin{array}{ll} \text{minimize} & f\,(x) \\ \text{subject to} & Ax = b. \end{array}$$

We assume that f is twice continuously differentiable, and that A is an $m \times n$ matrix of full row rank. We also assume that a local minimizer x_* has been found, with correspondingoptimal objective value $f\,(x_*)$. Suppose now that the right-hand side b is perturbed to $b + \delta,$ where δ is a vector of “small” perturbations. We shall investigate how the optimal objective value changes as a result of these perturbations. If the perturbations are sufficiently small, it is reasonable to assume that the new problem has an optimum that is close to x_*. In fact this can be shown to be true, provided that the second-order sufficiency conditions are satisfied at x_*. For $\bar{x}$ close to x_* with $A\,\bar{x} = b + \delta$, we can use a Taylor series approximation to obtain

$$f(\bar{x}) \approx f(x_*) + (\bar{x} - x_*)^T \nabla f(x_*)$$

$$= f(x_*) + \left(\bar{x} - x_*\right)^T A^T \lambda_*$$

$$= f(x_*) + \delta^T \lambda_*$$

$$= f(x_*) + \sum_{i=1}^{m} \delta_i \lambda_{*i}.$$

In particular, this is valid if $\bar{x}$ is the minimizer of the perturbed problem. If the right-hand side of the ith constraint changes by δ_i, then the optimal objective value changes by approximately $\delta_i \lambda_{*i}$. Hence λ_{*i} represents the change in the optimal objective per unit change in the ith right-hand side. For this reason, the Lagrange multipliers are also called *shadow prices* or *dual variables*.

Example 4.12 (Solution of a Perturbed Problem)**.** Consider again the problem

minimize $\quad f(x) = x_1^2 - 2x_1 + x_2^2 - x_3^2 + 4x_3$

subject to $\quad x_1 - x_2 + 2x_3 = 2.$

In Example 4.10 we determined that $x_* = (2.5, -1.5, -1)^T$, with $f(x_*) = -1.5$ and $\lambda_* = 3$.

Consider now the perturbed problem

$$\text{minimize} \quad f(x) = x_1^2 - 2x_1 + x_2^2 - x_3^2 + 4x_3$$

$$\text{subject to} \quad x_1 - x_2 + 2x_3 = 2 + \delta.$$

and denote its minimum value by $f_*(\delta)$. The interpretation of the Lagrange multipliers as shadow prices indicates that a first-order estimate of this minimum value is

$$f_*(\delta) \approx -1.5 + 3\delta.$$

For example, $if\ \delta = 0.5$, the approximate optimal objective value is zero.

The precise solution to the perturbed problem is $x_1 = 2.5 - \delta/2$, $x_2 = -1.5 + \delta/2$,

and $x_3 = -1 + \delta$, with an objective value of

$$f_*(\delta) = -1.5 + 3\delta - 0.5\delta 2.$$

If $\delta = 0.5$, the true value of the optimal objective is -0.125.

Let us now take another look at the optimality conditions (5.2). Since any solution must be feasible, a local optimum is the solution to the system of $n + m$ equations in the $n + m$ unknowns x and λ:

$$\nabla f(x) - A^T\lambda = 0$$

$$Ax = b.$$

This is another representation of the first-order optimality conditions.

These conditions were used by Lagrange, although his work was done in a moregeneral setting ([35]) . Following Lagrange's approach we can construct a

function of x and λ:

$$L(x,\lambda) = f(x) - \sum_{i=1}^{m} \lambda_i\left(a_i^T x - b_i\right) = f(x) - \lambda^T(Ax - b).$$

where a_i^T denotes the ith row of A. This function is called the *Lagrangian function*. The gradient of the Lagrangian with respect to x is $\nabla_x L(x,\lambda) = \nabla f(x) - A^T\lambda$, and the gradient with respect to λ is $\nabla_\lambda L(x,\lambda) = b - Ax$. Hence, the first-order optimality conditions can simply be stated as

$$\nabla L(x_*,\lambda_*) = 0.$$

Thus a local minimizer is a stationary point of the Lagrangian function.

4.4 Computing the Lagrange Multipliers:

Consider the linear equality-constrained problem

$$\begin{aligned} &\text{minimize} \quad f\,(x) \\ &\text{subject to} \quad Ax \;=\; b. \end{aligned}$$

Assume that the regularity condition holds, that is, that the rows of A are linearly independent. Consider the optimality condition

$$A^T\lambda_* \;=\; \nabla f\,(x_*).$$

This is a system of n equations in $m \;\le\; n$ unknowns, and so it cannot normally be expected to have a solution. At most feasible points x_*, this overdetermined system will be inconsistent, but if x_* is a local solution of the optimization problem, then the system will have a solution. How can such a solution λ_*be computed?

A useful tool is a matrix known as the *right inverse*. We define an $n \,\times\, m$ matrix A_r to be a *right inverse* for the $m \,\times\, n$ matrix A, if $AA_r \;=\; I_m$. It is easy to see that a matrix A has a right inverse only if it has full row rank. In this case, and if $m \;=\; n$, then the right inverse is unique, and $A_r \;=\; A^{-1}$. If $m \;<\; n$, the right inverse is generally not unique. For example, the matrices

$$\begin{pmatrix} \frac{3}{4} & 0 \\ \frac{-1}{4} & 0 \\ 0 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

are both right inverses for the matrix

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

To see how right inverses are of use in solving the system $A^T\lambda_* = \nabla f\,(x_*)$, suppose that a solution to this system exists. If both sides of this equation are multiplied by A_r^T , then we obtain

$$\lambda_* = A_r^T \nabla f\,(x_*)$$

(Here A_r^T refers to $(A_r)^T$ and not $(A^T)_r$.) If the system $A^T\lambda_* = \nabla f\,(x_*)$ is consistent, its solution $\lambda_* = A_r^T\nabla f\,(x_*)$ is unique, even though the right inverse may not be unique. To verify this, note that $A^T\lambda_* = \nabla f\,(x_*)$ implies that $AA^T\lambda_* = A\nabla f\,(x_*)$, and so the unique solution is

$$\lambda_* = (AA^T)^{-1}A\nabla f\,(x_*).$$

(If A has full row rank, the matrix AA^T is positive definite, and hence its inverse exists.) The linear system $A^T\lambda_* = \nabla f\,(x_*)$ is consistent if and only if $\nabla f\,(x_*)$ is a linear combination of the rows of A. Hence a vector $\lambda*$ computed via $\lambda_* = A_r^T\nabla f\,(x_*)$ will be a solution to the system if and only if

$$(I - A^TA_r^T)\nabla f\,(x_*) = 0$$

In practice we will almost never find a point x_* that satisfies the optimality conditions precisely. Rather, we will (if successful) find some point x_k that satisfies the optimality conditions to within some specified tolerance. The point x_k will be an estimate of the optimal solution. Correspondingly, the vector $\lambda = A_r^T\nabla f\,(x_k)$ will be only an estimate of the vector of Lagrange multipliers at the solution. It is sometimes termed a *first-order* estimate, because, for sufficiently small ϵ , if $\|x_k - x_*\| = O(\epsilon), then \|\lambda - \lambda_*\| = O(\epsilon)$ also .

In the rest of this section, we discuss methods for computing a right-inverse matrix.

To avoid unnecessary work, the computation of a right-inverse matrix for a matrix A should be performed in conjunction with the computation of the null-space matrix for A. We will show that each of the methods for computing a null-space matrix for A .(see Section 4.0.3) provides a right-inverse matrix at little or no additional cost. The discussion assumes that A is an $m \times n$ matrix of full row rank.

• *The variable reduction method*.(see Section 4.0.3.1) In this method the variables are partitioned into m basic and $n - m$ nonbasic variables. The matrix A is partitioned into basic and nonbasic columns correspondingly. Assuming that the first m columns are basic, we have $A = (B, N)$ where B is an $m \times m$ nonsingular matrix, and the $n \times (n - m)$ matrix

$$Z = \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix}$$

is a basis matrix for the null space of A. The matrix

$$A_r = \begin{pmatrix} B^{-1} \\ 0 \end{pmatrix}$$

is a right-inverse matrix for A that is available with no additional computation.

• *Orthogonal projection matrix* . Let the $n \times n$ matrix

$$P = I - A^T(AA^T)^{-1}A$$

be the orthogonal projection matrix(see Section 4.0.3.2) into the null space of A. A right inverse for Aassociated with the orthogonal projection is the matrix

$$A_r = A^T(AA^T)^{-1}.$$

This matrix, which we will denote by A^+, is a special right inverse. It satisfies the following four conditions:

$$AA^+A = A, \qquad (AA^+)^T = AA^+,$$

$$A^+AA^+ = A^+, \qquad (A^+A)^T = A^+A.$$

It can be shown that, for *any* $m \times n$ matrix A, there is a unique $n \times m$ matrix A^+ that satisfies these conditions. A^+ is called the *Penrose–Moore generalized inverse* of A.

If A has full row rank, then $A^+ = A^T(AA^T)^{-1}$, and if A has full column rank, then $A^+ = (A^TA)^{-1}A^T$. Formulas for A^+ can also be developed when A does not have full row or column rank; [41].

Given a point x_k , the vector of Lagrange multiplier estimates $(A^+)^T \nabla f\ (x_k)$ obtained from the Penrose–Moore generalized inverse has the appealing property that it solves the problem

$$\underset{\lambda \in \Re^m}{minimize} \|A^T\lambda - \nabla f(x_k)\|_2$$

For this reason it is termed the *least-squares Lagrange multiplier estimate* at x_k Because the condition number of AA^T is the square of the condition number of A, the computation of $(AA^T)^{-1}$ is potentially unstable. The *QR* factorization provides a stable approach to computing this matrix that is practical for smaller problems

• *A nonorthogonal projection* (see Section 4.0.3.3) Let *D* be a positive-definite $n \times n$ matrix. Then the $n \times n$ projection matrix

$$P_D = I - DA^T(ADA^T)^{-1}A$$

is a null-space matrix for A. A right inverse for A associated with this projection is

$$A_r = DA^T(ADA^T)^{-1}.$$

• *The QR factorization* . (see Section 5.0.3.4)The *QR* factorization represents A^T as a product of an orthogonal matrix Q and an upper triangular matrix R. Denoting the first m columns of Q by Q_1 and the last $n - m$ columns by Q_2, we have

$$A^T = QR = (Q_1\, Q_2)\begin{pmatrix} R_1 \\ 0 \end{pmatrix}$$

where R_1 is an $m \times m$ triangular matrix. The $n \times (n - m)$ matrix

$$Z = Q_2$$

is an orthogonal basis for the null space of A. The matrix

$$A_r = Q_1 R_1^{-T}$$

is a right inverse for A available from the QR factorization at little additional cost. In fact, this matrix need not be formed explicitly: a computation of the form $\lambda = A_r^T \nabla f(x_k)$

may be done by first computing $y_1 = Q_1^T \nabla f(x_k)$ and then solving the triangular system $R_1 \lambda = y_1$. It is easy to show that $A_r = A^T(AA^T)^{-1}$, and hence this right inverse is in fact the Penrose–Moore generalized inverse of A.

Just as with the "regular" matrix inverse, a right inverse is a useful notational tool, but it should rarely be formed explicitly. Instead, computations with respect to the right inverse should use the specific matrix factorizations that were employed to obtain the null-space matrix.

Example 4.13 (Right Inverses)**.** We will construct several right inverses for

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

If variable reduction is used, with columns 2 and 3 of A being selected as the basic columns, then

$$B = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \; and \;\; N = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

From these we determine that

$$Z = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad A_r = \begin{pmatrix} 0 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

(For all the right inverses that we compute in this example, it is straightforward to verify that $AZ = 0$ and $AA_r = I$.)

If the orthogonal projection matrix is used, then

$$P = I - A^T(AA^T)^{-1}A = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{-1}{2} \\ 0 & 0 & \frac{-1}{2} & \frac{1}{2} \end{pmatrix}$$

The corresponding right inverse is

$$A_r = A^T(AA^T)^{-1} = \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{-1}{2} & 0 \\ 0 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix}.$$

A nonorthogonal projection can also be used. If

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$

then

$$P_D = I - DA^T(ADA^T)^{-1}A = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} & 0 & 0 \\ \frac{2}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{4}{7} & \frac{-3}{7} \\ 0 & 0 & \frac{-4}{7} & \frac{3}{7} \end{pmatrix}$$

The corresponding right inverse is

$$A_r = DA^T(ADA^T)^{-1} = \begin{pmatrix} \frac{1}{3} & 0 \\ \frac{-2}{3} & 0 \\ 0 & \frac{3}{7} \\ 0 & \frac{4}{7} \end{pmatrix}$$

If a QR factorization of A^T is used, then

$$Q = \begin{pmatrix} \frac{-1}{\sqrt{2}} & 0 & \frac{-1}{2} & \frac{-1}{2} \\ \frac{1}{\sqrt{2}} & 0 & \frac{-1}{2} & \frac{-1}{2} \\ 0 & \frac{-1}{\sqrt{2}} & \frac{4}{7} & \frac{-1}{2} \\ 0 & \frac{-1}{\sqrt{2}} & \frac{-1}{2} & \frac{1}{2} \end{pmatrix} \quad \text{and} \quad R = \begin{pmatrix} -\sqrt{2} & 0 \\ 0 & -\sqrt{2} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Z consists of the last two columns of Q :

$$Z = \begin{pmatrix} \frac{-1}{2} & \frac{-1}{2} \\ \frac{-1}{2} & \frac{-1}{2} \\ \frac{1}{2} & \frac{-1}{2} \\ \frac{-1}{2} & \frac{1}{2} \end{pmatrix}$$

The right inverse is obtained from the formula

$$A_r = Q_1 R_1^{-T}$$

where Q_1 consists of the first two columns of Q,

$$Q_1 = \begin{pmatrix} \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{-1}{\sqrt{2}} \\ 0 & \frac{-1}{\sqrt{2}} \end{pmatrix}$$

and R_1 consists of the first two rows of R,

$$R_1 = \begin{pmatrix} -\sqrt{2} & 0 \\ 0 & -\sqrt{2} \end{pmatrix}$$

Hence

$$A_r = \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{-1}{2} & 0 \\ 0 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix}$$

If

$\nabla f\ (x)\ =\ (\,7\ -7\ -2\ -2\,)^T$,

then for all of the above right inverses,

$$\lambda = A_r^T \nabla f(x) = \begin{pmatrix} 7 \\ -2 \end{pmatrix}$$

No matter which right inverse is used, the same values of the Lagrange multipliers are obtained.

Program 17 Lagrange _ Multiplier _ Method:

```
 clear
A=[1 -1 0 0;0 0 1 1]
Deltaf=[7 -7 -2 -2]'
%A=input('Please, Enter A matrix  ')
%Deltaf=input('Please, Enter ?f (x) matrix ')
 [Q,R]=qr(A')
[m,n]=size(R)
Q1=Q(1:m,1:n)
R1=R(1:n,1:n)
y1=Q1'*Deltaf
c=length(y1)
lambda(c)=y1(c)/R1(c,c)
for i=c-1:-1:1
    s=0;
    for k=i+1:c
        s=s+R1(i,k)*lambda(k);
    end
    lambda(i)=(y1(i)-s)/R1(i,i);
end
lambda=lambda'
```

Program 18 Lagrange _ Multiplier _ Method1:

```
% clear
% clc
   ] A=[4 1 1 1;1 4 1 1;1 1 4 1;1 1 1 4
  ] Deltaf=[1 1 1 1
A=input('Please, Enter A matrix ')%
Deltaf=input('Please, Enter ?f (x) matrix ')%
Deltaf=Deltaf'
 [Q,R]=qr(A ')
)m,n]=size(R [
Q1=Q(1:m,1:n )
R1=R(1:n,1:n )
y1=Q1'*Deltaf
c=length(y1 )
lambda(c)=y1(c)/R1(c,c)
for i=c-1:-1:1
   ; s=0
    for k=i+1:c
        s=s+R1(i,k)*lambda(k);
  end
  ); lambda(i)=(y1(i)-s)/R1(i,i
end
lambda=lambda'
```

Program19 traditional _ methods :

```
clc
] A=[4 1 1 1;1 4 1 1;1 1 4 1;1 1 1 4
  Deltaf=[1 1 1 1[
lambda=inv(A)* Deltaf'
lambda=lambda'
```

5-5 Comparison and conclusion:

We use the QR-methods to solve the previous problem , and as we know it was an under determinant problem which lead us to the following observations:

- In under determinant problems in optimization (the problem in linear mathematics which the number of linear equation is less than the number of unknown is called the under determinate problem and has infinite solutions) the QR-methods can solve easily as the example shows. The traditional methods fails to get a solutions.
- In well determinant problems in optimization (the problem in linear mathematics which the number of linear equation is equal the number of unknown is called the well determinate problem and has just unique solution) the QR-methods can solve easily as the following example shows.
  - Run the m-files Lagrange_Multiplier_Method1.m and traditional_methods.m

        - Use the MATLAB Command TIC-TOC to get the time of execution for each m-files
        - We get the following result
            - QR-method = Elapsed time is 0.001140 seconds.
            - The traditional Method = Elapsed time is 0.000470 seconds.

Although , the time of traditional methods is a little bit faster than the QR Method we have to keep in mind it depend on the Matrix A itself. We can conduct another experiment and get result that shows the QR-method is faster. Another advantage of QR-method is that what if A is Singular or semi singular that will lead to fail of the traditional methods .

- In over determinant problems in optimization (the problem in linear mathematics which the number of linear equation is greater than the number of unknown is called the over determinate problem and has no solutions) the QR-methods can solve it. The over determinant problem are un likely to happen in optimization ,but it happen when the model is built by normal people (not professionals). So, we can say that the QR-method can help the normal user to try to improve their work by using optimizations techniques and not afraid of complicates of the solution of the model.
- The computations methods is much more faster than traditional methods and make the computer is much useful tool by applying the numerical techniques .

References :

1. J.G.F. Francis, The QR transformation, Parts I and II, Computer J. 4 1961 265{272, 332{345.
2. H. Rutishauser, Der Quotienten-Di_erenzen-Algorithmus, Mitt. Inst. Angew. Math. ETH, Vol. 7, Birkhauser, Basel, 1957.
3. H. Rutishauser, Solution of eigenvalue problems with the LR-transformation, Nat. Bur. Standards Appl. Math. Series 49 (1958) 47-81.
4. V.N. Kublanovskaya, On some algorithms for the solution of the complete eigenvalue problem, USSR Comput. Math. Math. Phys. 3 (1961) 637-657.
5. B.T. Smith et al., Matrix Eigensystem Routines - EISPACK Guide, Springer, Berlin, 2nd Edition, 1976.
6. J.J. Dongarra, J.R. Bunch, C.B. Moler, G.W. Stewart, LINPACK Users' Guide, SIAM, Philadelphia, 1979.
7. E. Anderson et al., LAPACK Users' Guide, SIAM, Philadelphia, 2nd Edition, 1995. http://www.netlib.org/lapack/lug/lapack lug.html.
8. R. van de Geijn, D.G. Hudson, An efficient parallel implementation of the nonsymmetric QR algorithm, in:Proceedings of the Fourth Conference on Hypercube Concurrent Computers and Applications, 1989.
9. G. Henry, D. Watkins, J. Dongarra, A parallel implementation of the nonsymmetric QR algorithm for distributedmemory architectures, Technical Report LAPACK Working Note 121, University of Tennessee, 1997.
10. L.S. Blackford et al., ScaLAPACK Users' Guide, SIAM, Philadelphia, 1997. http://www.netlib.org/scalapack/slug/scalapack slug.html.
11. J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford University, 1965.

12. B.N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, New Jersey, 1980. Reprinted by SIAM, 1997.
13. J. Della-Dora, Numerical linear algorithms and group theory, Linear Algebra Appl. 10 (1975) 267-283.
14. D.S. Watkins, L. Elsner, Convergence of algorithms of decomposition type for the eigenvalue problem, Linear Algebra Appl. 143 (1991) 19-47.
15. Stewar G. (1973) Introduction to Matrix Computations. Academic Press, New York.
16. Axelsson O. (1994) Iterative Solution Methods . Cambridge University , New York.
17. Hackbush W. (1994) Iterative Solution of Large Sparse Systems of Equations. Springer-Verlag, New York.
18. Demmel J. (1997) Applied Numerical Linear Algebra. SIAM, Philadelphia.
19. Axelsson O. (1994) Iterative Solution Methods. Cambridge University Press, New York.
20. Stewart G. and Sun J. (1990) Matrix Perturbation Theory. Academic Press, New York.
21. Godeman R. (1966) Algebra. Kershaw, London.
22. Golub G. and Loan C.V. (1989) Matrix computations. The John Hopkins Univ. Press, Baltimore and London.
23. Isaacson E. and Keller H. (1966) Analysis of Numerical Methods. Wiley, New York.
24. Wilkinson J. (1968) A priori Error Analysis of Algebraic Processes. In Intern. Congress Math., volume 19, pages 629-639. Izdat. Mir, Moscow.
25. Jennings A. and Mckeown J. (1992) Matrix Computation. Wiley , chichester.
26. Parlett B. and Reid J.(1970) on the solution of the System of the Linear Equation Whose Matrix is Symmetric but not Definite BIT10:386-397

27. Aasen J(1971) on the Reduction of a Symmetric Matrix to Tridiagonal Form BIT 11:233-242.
28. Golub G. and Liu J. (1981) Computer Solution of Larg Sparse Positive Definite Systems. Prentice-Hall, Englewood Cliffs, New York.
29. Jennings A. and Mckeown J (1992) Matrix Computation Wiley Chichester.
30. Faddeev D. K. and Faddeeva V. N. (1963) Computational Methods of Linear Algebra. Freeman, San Francisco and London.
31. Barnett S. (1989) Leverrier's Algorithm : A New Proof and Extensions. Numer. Math. 7:338-352.
32. Lawson C. and Hanson R. (1974) Solving Least Squares Problems. Prentice-Hall, Englewood Cliffs, New York.
33. Bjorck A. (1988) Least Square Methods: Handbook of Numerical Analysis Vol. 1 Solution of Equations in $\mathbb{R}^N$. Elsevier North Holland.
34. Datta B.(1995) Numerical Linear Algebra and Application BrooksL\ cole publishing , pacific Grore , CA.
35. Igor G. and G. Nash and A. Sofer (2008) Linear and Non Linear Optimization . George Mason University Fairfax , Virginia.
36. Gastinel N.(1983) Linear Numerical Analysis. Kershaw Pub – Lishing London.
37. Inman D (1993) Engineering Vibration Prenticevue Hall , Engle wood cliffs , NT.
38. Bossavit A. (1993) Electromgnetisme, emvue de la modelistation. Springer – Verlag , paris.
39. Batterson S. (1990) Convergence of Shifet QR Algorithm on 3 by 3 Normal Matrices. Numer. Math -58:341-352.

40. Day D. (1996) HOW the QR algorithm Fails to converge and How to Fix It . Technical Report 96-0913J, Sandia National Laboratory , Albuquerque .

41. G.H.Golub and C.Van Matrix Computations (third edition), The Johns Hopkins University press, Baltimor , 1996.