AMATH 584A: Applied Linear Algebra

Marlin Figgins

Oct. 1, 2020

Introduction

Table of Contents

1. Overview: The problem Ax=b

This course will be almost entirely about the problem of $\mathbf{A}\mathbf{x} = \mathbf{b}$. That is, we're concerning with linear systems. In fact, many problems are of this form. In the age of data science, these matrix $\mathbf{A}$ and vector $\mathbf{x}$ can get huge quickly.

1.1 Matrix Decompositions

In what follows, let's assume we are given a complex matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ and a vector $\mathbf{b}$. Suppose that we're given the problem

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{1.1}$$

In your typical linear algebra classes, you learn to solve this with Gaussian elimination, but the reality is that this is one of the slowest ways you can solve this problem. To solve this problem with Gaussian Elimination, the cost would be on the order of $O(n^3)$. This is fine for small matrices, but immagine you're dealing with large matrices and this begins to blow up in computation time rather quickly. Matrix decompositions allow us to solve the problem $\mathbf{A}\mathbf{x} = \mathbf{b}$ much faster. We'll start with a simple overview of several matrix decompositions such as the **LU**, **QR**, eigenvalue, and singular value decompositions.

1.1.1 LU decomposiiton

The **LU** decomposition allows us to represent our matrix $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{L}\mathbf{U} \tag{1.2}$$

where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{U}$ is upper triangular. Our problem becomes

$$Ax = b \tag{1.3}$$
$$LUx = b \tag{1.4}$$
$$Ux = y \tag{1.5}$$
$$Ly = b \tag{1.6}$$

This allows us to use forward and back substituion individually which are of order $O(n^2)$ to solve this probelm. This **LU** decompoisition already gives a saving of order of n. This is all well and good, but what does it take to get an **LU** decomposition?

1.1.2 QR decomposition

We want to express our matrix $\mathbf{A}$ in the form

$$\mathbf{A} = \mathbf{Q}\mathbf{R} \tag{1.7}$$

where $\mathbf{Q}$ is a unitary matrix and $\mathbf{R}$ is upper triangular. Solving $vecAx = b$ with this decomposiiton gives us,

$$
\begin{aligned}
QRx &= b && (1.8)\\
Rx &= y && (1.9)\\
Qy &= b && (1.10)\\
Q^T[Qy &= b] && (1.11)\\
y &= Q^T b && (1.12)
\end{aligned}
$$

1.1.3 Eigenvalue Decomposition

We can write the eigenvalue decomposition as

$$\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1} \tag{1.13}$$

Using this to solve $\mathbf{Ax} = \mathbf{b}$, we get that

$$
\begin{aligned}
V^{-1}[V\Lambda V^{-1}x &= b] && (1.14)\\
\Lambda y &= V^{-1}b && (1.15)
\end{aligned}
$$

Since $\boldsymbol{\Lambda}$ is diagonal, the answer is very clear here.

1.1.4 Singular Value Decomposition

The singular value decomposition is one of the most important decomposition algorithms. We decompose $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{*}. \tag{1.16}$$

Solving $\mathbf{A}x = b$,

$$
\begin{aligned}
U\Sigma V^* x &= b && (1.17)\\
\Sigma V^* x &= U^* b && (1.18)\\
\Sigma \hat{x} &= \hat{b} && (1.19)
\end{aligned}
$$

1.2 Under and over determined systems

In reality, we're often dealing with systems and matrices which are not perfectly square. Many problems are not perfectly square. In reality, very few are. The rest of these problems fall into two general categories of systems which we call underdetermined and overdetermined systems. When it comes to solving $\mathbf{A}x = b$ for these problems, the question is ill-posed. Though these systems may have no solutions or infinitely many solutions, most software will still be able to solve the problem $\mathbf{A}x = b$, how is this done?

Underdetermined systems ($m < n$). These systems fundamentally have infinitely many solutions, so $\mathbf{A}x = b$ is instead posed as an optimization problem

$$\min_x ||x||_2 \text{ such that } Ax = b. \tag{1.20}$$

In this case, the minimization of the L^2 norm acts as a regularizer for our desired solution x.

Overdetermined systsems ($m > n$). Due to the abundance of constraints, satisfying $\mathbf{A}x = b$ is technically impossible. In this case, we attempt to find the closest possible solution i.e.

$$\min_x ||Ax + b|| + \lambda ||x||_2 . \tag{1.21}$$

Here the L^2 norm acts as a regularizer for our solution x. We use the parameter λ as a hyperparameter which determines the relative importance of the regularizer λ. Indeed, there are several different ways to do this regularization such as using the L^1 norm.

$$\min_x ||Ax + b|| + \lambda_1 ||x||_1 + \lambda_2 ||x||_2 . \tag{1.22}$$

2. Linear Operators

Linear operators. Linear operators are communtative and associative under addition.

(i) *Commutative* (+). $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$

(ii) *Associative* (+). $\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$

(iii) *Distributive.* $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{BC}$

(iv) *Associative* ($\cdot$). $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$

Though we have all these algebraic properties, it is important to know that multiplication is not commutative for matrices in general i.e. given any two linear operators, it is not necessary true that

$$BA \neq AB. \tag{2.1}$$

2.1 Matrix Fundamentals

Matrices and vectors. For the majority of these notes, the linear operators we will work with will be complex matrices $\mathbf{A} \in \mathbb{C}^{n\times m}$ which operate on complex (column) vectors $\mathbf{x} \in \mathbb{C}^m$. We can illustrate these matrices and vectors with the following notation:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & & a_{2m} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \text{ and } \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}. \tag{2.2}$$

where the numbers $a_{ij} \in \mathbb{C}$ and $x_i \in \mathbb{C}$ are called the entries of $\mathbf{A}$ and the elements of $\mathbf{x}$ respectively. Alternatively, we can represent this same matrix by its columns $\mathbf{a}_1, \ldots, \mathbf{a}_n$:

$$\mathbf{A} = \begin{pmatrix} \vdots & \vdots & & \vdots \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ \vdots & \vdots & & \vdots \end{pmatrix}. \tag{2.3}$$

Notice that the matrix $\mathbf{A}$ has n rows and m columns and that $\mathbf{x}$ is written so that it has m rows.

Addition on matrices and vectors. Addition between vectors happens element-wise. Similarly, addition between matrices occurs entry-wise. This means that addition is only well defined between matrices and vectors of the same size.

Scalar multiplication of vectors and matrices.

Multiplying vectors by matrices. Given a matrix $\mathbf{A} \in \mathbb{C}^{n\times m}$ and a vector $\mathbf{x} \in \mathbb{C}^m$, we can make $\mathbf{A}$ act as a linear operator on $\mathbf{x}$. We can write the resulting vector n-vector $\mathbf{Ax}$ element-wise as:

$$(\mathbf{Ax})_i = \sum_{j=1}^{n} a_{ij} x_j \tag{2.4}$$

We can also write this as a linear combination of the columns of $\mathbf{A}$.

$$\mathbf{Ax} = \begin{pmatrix} \vdots \\ x_1\mathbf{a}_1 \\ \vdots \end{pmatrix} + \begin{pmatrix} \vdots \\ x_2\mathbf{a}_2 \\ \vdots \end{pmatrix} + \cdots + \begin{pmatrix} \vdots \\ x_n\mathbf{a}_n \\ \vdots \end{pmatrix}. \tag{2.5}$$

Matrix multiplication.

Matrix inverses.

2.2 Norms and inner products

Defining the norm In short, a norm is just a way of quantifying distance. In particular, the two most interesting norms that we'll cover are the L^2 and L^1 norms.

L^2 norm We can define the L^2 norm of a vector $\mathbf{x}$ as

$$||\mathbf{x}||_2 = \sqrt{|x_1|^2 + |x_2|^2} \tag{2.6}$$

Notice, this is the distance that we're used to in most geometric contexts.

L^1 norm There are also various other norms such as the L^1 norm which we denote as $||\cdot||_1$

$$||\mathbf{x}||_1 = |x_1| + |x_2| \tag{2.7}$$

In applications, the L^1 norm tends to promote sparsity in solutions.

L^p norms In general, we can compute the L^p norm of a vector x as

$$||\mathbf{x}||_p = (|x_1|^p + |x_2|^p)^{1/p}. \tag{2.8}$$

There are additionally two *special* L norms which are the L^∞ and L^0 norms.

L^∞ norm

L^0 norm There are also the L^∞ and L^0 norms.

Inner products The inner product of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{C}^m$ is given by

$$\mathbf{x}^*\mathbf{y} = \sum_{i=1}^{m} \bar{x}_i y_i, \tag{2.9}$$

where $\bar{z}$ denotes the *complex conjugate of* z. Notice that the inner product $\mathbf{x}^*\mathbf{y}$ is a scalar. The inner product is bilinear in the following sense. Suppose we have vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{y} \in \mathbb{C}^m$ and two scalars $\alpha, \beta \in \mathbb{C}$.

$$(\mathbf{x}_1 + \mathbf{x}_2)^*\mathbf{y} = \mathbf{x}_1^*\mathbf{y} + \mathbf{x}_2^*\mathbf{y} \tag{2.10}$$

$$\mathbf{y}^*(\mathbf{x}_1 + \mathbf{x}_2) = \mathbf{y}^*\mathbf{x}_1 + \mathbf{y}^*\mathbf{x}_2 \tag{2.11}$$

$$(\alpha\mathbf{x})^*(\beta\mathbf{y}) = \bar{\alpha}\beta\mathbf{x}^*\mathbf{y}. \tag{2.12}$$

2.3 Adjoint and Unitary Operators

Adjoint We define the *adjoint* of a matrix $\mathbf{A}$ to be the complex conjugate of its transpose $\mathbf{A}^*$. That is,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \implies \mathbf{A}^* = \begin{pmatrix} \bar{a}_{11} & \cdots & \bar{a}_{n1} \\ \bar{a}_{12} & & \bar{a}_{n2} \\ \vdots & \ddots & \vdots \\ \bar{a}_{1m} & \cdots & \bar{a}_{nm} \end{pmatrix}. \tag{2.13}$$

Similarly, we note that the adjoint of a vector $\mathbf{x}$ is also the complex conjugate of its transpose. Therefore, our definition of the inner product simplifies to be multiplication of a vector and its adjoint. This why we use the same symbol * to denote both the inner product and adjoint.

Example 2.1. As a concrete example, consider the following matrix:

$$\mathbf{A} = \begin{pmatrix} 2+i & 7 & i \\ 7i & 4 & 12-i \end{pmatrix} \tag{2.14}$$

Under our definition, its adjoint is given by

$$\mathbf{A}^* = \begin{pmatrix} 2-i & -7i \\ 7 & 4 \\ -i & 12+i \end{pmatrix}. \tag{2.15}$$

Hermitian Matrices In the case, that a matrix is its own adjoint i.e. $\mathbf{A} = \mathbf{A}^*$. We say that it is *self-adjoint* or *Hermitian*. All Hermitian matrices are square matrices.

Unitary Matrices Another class of matrices related to the adjoint are the unitary matrices. A matrix $\mathbf{U}$ is said to be *unitary* if $\mathbf{U}^*\mathbf{U} = \mathbf{I}$.

2.4 Nullspaces and zero eigenvalues

Consider the problem $\mathbf{A}^*y = 0$ where $\mathbf{A}^*$ is the adjoint of the matrix $\mathbf{A}$. When does $\mathbf{A}x = b$ have a solution?

$$Ax \cdot y = b \cdot y \tag{2.16}$$
$$x \cdot A^*y = b \cdot y \tag{2.17}$$
$$b \cdot y = 0 \tag{2.18}$$

The Fredholm alternative is the statement that b is not orthogonal to y, then $\mathbf{A}x = b$ has no solutions.

Suppose we have the problem $\mathbf{A}x = b$ and a vector x_0 with 0 eigenvalue $\mathbf{A}x_0 = 0$. Then we can generate solutions as any vector of the form

$$x = \xi + \alpha x_0, \tag{2.19}$$

where ξ is a solution. The regularization process is an attempt to avoid this by minimizing across the vectors x_0 in the nullspace.

3. Singular Value Decomposition

Here, we return to the singular value decomposition which is one of the most important matrix decompositions in the modern world. We'll begin with an example in 2-dimensions and then scale up the problem.

Example 3.1. Consider the following matrix and vector:

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ -1 & 1 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \tag{3.1}$$

Using our standard linear algebra muscles, we can easily compute the product $\mathbf{Ax}$ as

$$\mathbf{Ax} = \begin{pmatrix} 2 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \end{pmatrix} = \mathbf{y}. \tag{3.2}$$

Thinking about this geometrically, we can say that the matrix $\mathbf{A}$ rotated the vector $\mathbf{A}$ and then stretched it. In order to decompose this operation, we might ask: "How can we decompose the matrix $\mathbf{A}$ as a rotation and a stretching?". The former aspect is done with the standard rotation matrix R_θ which rotates a vector by an angle θ

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{3.3}$$

Similarly, the stretching can be accomplishing by multiplication by the matrix

$$\alpha \mathbf{I} = \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix}. \tag{3.4}$$

Mathematically, we can write this as:

$$\mathbf{A}\mathbf{v}_i = \sigma_1 \mathbf{u}_i. \tag{3.5}$$

We can also stack these vectors $\mathbf{v}_i$ and $\mathbf{u}_i$ into a matrices of size $n \times n$, so that

$$\mathbf{A} \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \end{pmatrix} \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{pmatrix}. \tag{3.6}$$

More compactly, we can write this as

$$\mathbf{AV} = \mathbf{U\Sigma}. \tag{3.7}$$

Since our vectors $\mathbf{v}_i$ map to another orthonormal basis $\mathbf{u_i}$, we can simply think of them as a kind of rotation. In fact the matrices, we construct $\mathbf{V}$ and $\mathbf{U}$ are both unitary matrices. Lastly, as we seen the matrix Σ is a diagonal matrix. We can re-write this by taking advatange of the fact that $\mathbf{V}$ is unitary. Simply, right multiplying by the inverse of $\mathbf{V}$ shows that

$$A = \mathbf{U\Sigma V}^*. \tag{3.8}$$

This is called the reduced singular value decomposition.

Usually, the columns of the singular value decomposition are ordered, so that $\sigma_1 \geq \sigma_2 \ldots$.

This decomposition is extremely robust in fact it is guaranteed to exist for any matrix $\mathbf{A}$.

Theorem 3.2. *Every matriz $\mathbf{A} \in \mathbb{C}^{m \times n}$ has a singular value decomposition. Moreover,*

(i) The singular values are uniquely determined and if the matrix $\mathbf{A}$ is square, then they are also distinct.

(ii) The vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are also unique up to a complex sign.

Suppose we've taken the SVD of $\mathbf{A}$, we can compute the following relationship between the eigenvalues and the SVD.

$$\mathbf{A}^*\mathbf{A} = \mathbf{V\Sigma}^2\mathbf{V}^*. \tag{3.9}$$

Right multiplying by $\mathbf{V}$, we see that

$$\mathbf{A}^*\mathbf{AV} = \mathbf{V\Sigma}^2 \implies \lambda_j = \sigma_j^2. \tag{3.10}$$

With a similar computation, we can show that

$$\mathbf{AA}^*\mathbf{U} = \mathbf{U\Sigma}^2, \tag{3.11}$$

which allows us to recover our matrix $\mathbf{U}$.

4. Principal Component Analysis

5. QR Decomposition

The idea of the QR decomposition is to write it as a product of matrices $\mathbf{QR}$, where the matrix $\mathbf{Q}$ is unitary and $\mathbf{R}$ is upper triangular. In the case of $\mathbf{Ax} = \mathbf{b}$. We can decompose $\mathbf{A}$ using the $\mathbf{QR}$ decomposition, so that

$$\mathbf{y} = \mathbf{R}\mathbf{x} = \mathbf{Q}^*\mathbf{b} \tag{5.1}$$
$$\mathbf{Q}\mathbf{y} = \mathbf{b}. \tag{5.2}$$

This reduces solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ to a single matrix multiplication and back substitution.

Consider a matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ $(m > n)$ with column vectors $\mathbf{a}_i$ which are linearly independent. These columns then form an n-dimensional basis

$$< \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n > . \tag{5.3}$$

Therefore, given any vector of interest $\mathbf{x} \in \mathbb{C}^n$, we can write it as a linear combination of these basis vectors, so that

$$x = \sum_{j=1}^{n} \alpha_j \mathbf{a}_j. \tag{5.4}$$

We then have a linear system of equations

In order to simplify this, we'll attempt to work with an orthonormal basis instead. This would greatly simpliftly the math since if we have an orthonormal basis $\{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$, then $\alpha_i = \mathbf{q}_i^*\mathbf{x}$. Therefore,

$$\mathbf{x} = \sum \alpha_j \mathbf{q}_j = \sum (\mathbf{q}_j^*\mathbf{x})\mathbf{q}_j. \tag{5.5}$$

One such approach to finding these orthonormal vectors is to construct them from our original basis $< \mathbf{a}_1, \ldots, \mathbf{a}_n >$. This is the GS orthonormalization procedure.

We have an iteration scheme $C\mathbf{q}_2 = \mathbf{a}_2 - (\mathbf{q}_1^*\mathbf{a}_2)\mathbf{q}_1$ where C is a normalization constant so that $\mathbf{q}_2$ has unit magnitude. This is the core idea behind Gram-Schmidt. By moving from the coordinate system of column space to an orthonormal system, we can simply the number of computations necessary for computations. We have that

$$\mathbf{q}_1 = \frac{\mathbf{a}_1}{r_{11}} \tag{5.6}$$
$$\mathbf{q}_2 = \frac{1}{r_{22}}\left(\mathbf{a}_2 - r_{12}\mathbf{q}_1\right) \tag{5.7}$$
$$\mathbf{q}_n = \frac{1}{r_{nn}}\left(\mathbf{a}_n - \sum_{j=1}^{n-1} r_{jn}\mathbf{q}_j\right) \tag{5.8}$$
$$\tag{5.9}$$

We can also work out the entries of $\mathbf{R}$ as

$$r_{ij} = \mathbf{q}_i^*\mathbf{a}_j \tag{5.10}$$
$$|r_{jj}| = \left\|\mathbf{a}_j - \sum_{i=1}^{j-1} r_{ij}\mathbf{q}_i\right\|_2 . \tag{5.11}$$

Projectors With the matrix $\mathbf{Q}$ at hand, we can write the vectors of $\mathbf{x} \in \mathbb{C}^m$ as

$$\mathbf{x} = \mathbf{r} + \sum_{j=1}^{n} (\mathbf{q}_j^* \mathbf{x}) \mathbf{q}_j \tag{5.12}$$

where $\mathbf{r}$ is orthogonal to the span of $\mathbf{Q}$ and the relationship

$$\mathbf{x} \mapsto \sum_{j=1}^{n} (\mathbf{q}_j^* \mathbf{x}) \mathbf{q}_j \tag{5.13}$$

is the projector from $\mathbf{x}$ to range($\mathbf{Q}$) which is n-dimensional. We could instead write this projector as

$$\mathbf{x} \mapsto \mathbf{Q}\mathbf{Q}^* \mathbf{x} \tag{5.14}$$

taking advantage of the fact that the $\mathbf{q}_j$ form the columns of $\mathbf{Q}$. This gives us the opportunity to define projectors in general.

Definition 5.1. A projector is a linear operator which satistifes that $\mathbf{P}^2 = \mathbf{P}$

Proof.

$$\mathbf{P}[\mathbf{y} = \mathbf{P}\mathbf{x}] \tag{5.15}$$

□

We define the complementary projectors as $\mathbf{I} - \mathbf{P}$. This is indeed a projecor itself

$$(\mathbf{I} - \mathbf{P})^2 = \mathbf{I} - 2\mathbf{P} + \mathbf{P}^2 \tag{5.16}$$
$$= \mathbf{I} - \mathbf{P}. \tag{5.17}$$

We also see that range($\mathbf{I} - \mathbf{P}$) = null($\mathbf{P}$). Sometimes, we'll just denote this as

$$\mathbf{P}_\perp = \mathbf{I} - \mathbf{P} \tag{5.18}$$

since the complementary projector is orthogonal to the original projector. There are plenty examples of projectors. For example, take a column vector $\mathbf{q}$, we can define the projector by $\mathbf{q}$ as

$$\mathbf{P}_\mathbf{q} = \mathbf{q}\mathbf{q}^*. \tag{5.19}$$

GS by projectors We can use this idea of projectors to implement our GS process. In short, the algorithm proceeds as follows for projectors $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n$ and input vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$

$$\mathbf{q}_1 = \frac{\mathbf{P}_1 \mathbf{a}_1}{||\mathbf{P}_1 \mathbf{a}_1||} \tag{5.20}$$
$$\mathbf{q}_2 = \frac{\mathbf{P}_2 \mathbf{a}_2}{||\mathbf{P}_2 \mathbf{a}_2||} \tag{5.21}$$
$$\vdots \tag{5.22}$$
$$\mathbf{q}_n = \frac{\mathbf{P}_n \mathbf{a}_n}{||\mathbf{P}_n \mathbf{a}_n||} \tag{5.23}$$

Here, we define the projectors according to the following

$$\mathbf{P}_j = \mathbf{I} - \mathbf{Q}_{j-1}\mathbf{Q}_{j-1}^*, \tag{5.24}$$

where $\mathbf{Q}_{j-1}$ is just the matrix with columns $\mathbf{q}_i$ for $i = 1, \ldots, j-1$. Another way of writing this as

$$\mathbf{P}_j = \mathbf{P}_{\perp \mathbf{q}_{j-1}} \mathbf{P}_{\perp \mathbf{q}_{j-2}} \cdots \mathbf{P}_{\perp \mathbf{q}_1}, \tag{5.25}$$

where $\mathbf{P}_1 = \mathbf{I}$. As discussed before, the multiplication $\mathbf{P}_{\perp \mathbf{q}_i}$ ensures that the next vector $\mathbf{q}_j$ is orthogonal to $\mathbf{q}_i$. This iteration scheme is stable comparted to the one discussed initially.