Indian Statistical Institute, Kolkata
Numerical Analysis (BStat–I)

Instructor: Sourav Sen Gupta
Scribe: Sayan Bhadra, Sinchan Snigdha Adhikary
Date of Lecture: 18 January 2016

LECTURE

2

Role of Linear Algebra in Numerical Analysis

2.1 Introduction : Why Linear Algebra?

One of the most important works of Numerical Analysis, turns out to be often of this form: find the best approximate solution to a system of simultaneous equations $AX = B$. Now the B is always the experimental observations, hence error is bound to creep in , often leading the whole system to be inconsistent. Due to all these difficulties, the only way left to get a satisfactory solution of the system is to try to minimise $|AX - B|$ i.e. ,find out the solution set for which the above is closest to zero.

For this, we need various techniques.Consider the following facts:

- Factorising a matrix in efficient ways can greatly help reduce computational time and complexity.
- If we are able to know somehow what effect the matrix(the coefficients of the system of equations) has on the vectors(the variables), then we can some sort of get a lead as how to minimise $|AX - B|$.
- The variables of our system of equations often have very high probabilities of being dependent on each other. In such cases, if we are able to corner out the biased ones(as well as how and upon what) then minimisation can become a lot simpler as then only minimising the independent ones would suffice, thus reducing the number of variables to deal with.

It is easy to guess now, why Linear Algebra is a part and parcel of Numerical Analysis.

2.2 A Don't

For any analytical purpose, always avoid direct computation of A^{-1}.This is because it not only involves an $O(n^3)$ computational time,but also most of the time, the purpose can be served in a shorter way without even computing it.

2.3 What does a Matrix do to a vector?

In simple words, a matrix *'transforms'* a *vector* in some sense. If A is a m x n matrix, and $v \in \mathbb{R}^n$, then Av defines a **linear transformation(T)** from $\mathbb{R}^n \rightarrow \mathbb{R}^m$, which satisfies the following properties:

- $T(v_1 + v_2) = T(v_1) + T(v_2)$
- $T(\alpha v_1) = \alpha T(v_1) \forall v_1, v_2, \alpha (\in \mathbb{R})$

We can perform a variety of operations on vectors namely *reflection* (w.r.t a subspace),*rotation* (w.r.t the origin), *scaling up or down a vector, projection on a subspace,* etc.

However there is something related to this called *Affine Transformation* which does not follow strict *"linearity"* but follows *Affine linearity* [1]. Mathematically, this can be expressed as $U(v) = \mathbf{A}v + v_0$ where $v_0 = U(0) \neq 0$. Examples include *translation, rotation w.r.t a non-origin point* etc.

Example: Let the vector space be $\mathbb{R}^2$ and the matrix be $M = \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix}$ w.r.t the canonical basis. This matrix maps the basis vectors ϵ_1, ϵ_2 to the vectors $(2, 1)$ & $(3, 2)$ respectively.

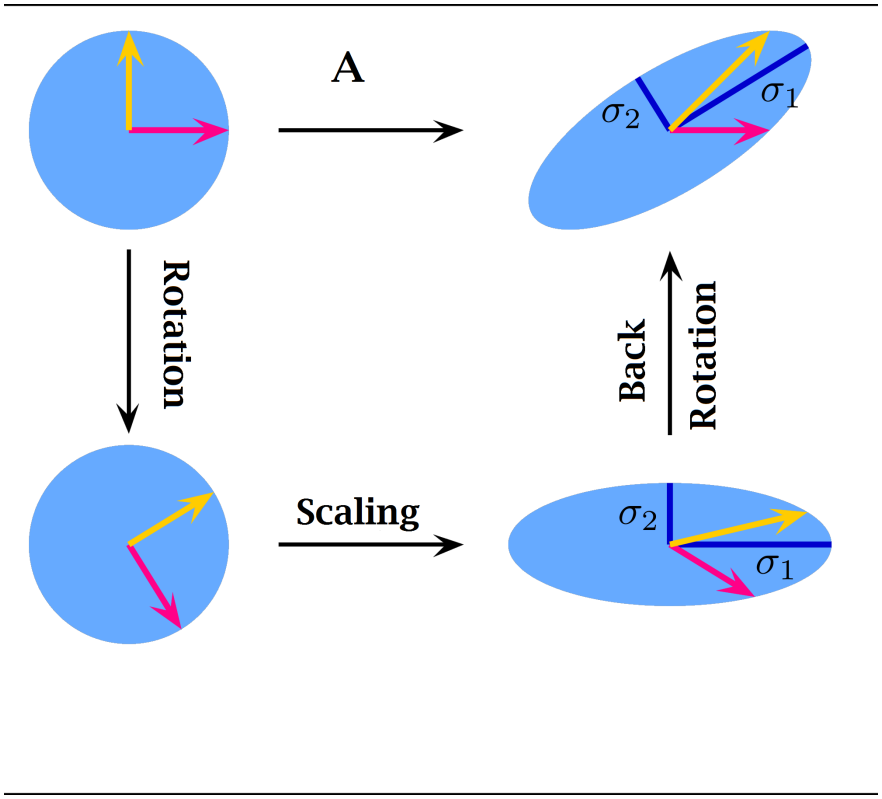

Figure 2.1: The working of a matrix A

[1] Affine combination of n vectors $\{x_1, \ldots, x_n\}$is of the form $\sum_{i=1}^{n} \alpha_i x_i$ where $\sum_{i=1}^{n} \alpha_i = 1$. Affine linearity means $U(\sum_{i=1}^{n} \alpha_i x_i) = \sum_{i=1}^{n} \alpha_i U(x_i)$; where $\sum_{i=1}^{n} \alpha_i = 1$.

2.4 Norms: Lengths, angles etc.

You can easily visualize a *circle* or a *square* in $2D$ space increase 2 to 3. Here also you can visualize an analog, namely a *sphere* or a *plane* or a *line*. But can you visualize an analog in more than 3 dimensions? In $2D$ or $3D$, you know what is meant by *distance* and *angle*. Can you visualise an analog in $4D$? Does this mean they don't exist in higher dimensions?

The answer is given by **norms**. There are a variety of norms, but they are, in general defined in the following way:

Definition 2.1. *A* norm *on a vector space* V $(F = \mathbb{R}$ *or* $\mathbb{C})$ *is a map* $\boldsymbol{x} \to ||x||$ *from* V *to* $\mathbb{R}$ *satisfying the conditions*

- $||x|| \geq 0$ *;equality iff* $x = 0$
- $||\alpha x|| = |\alpha|.||x||$
- $||x + y|| \leq ||x|| + ||y||$

A particular class of norms is the $\mathscr{L}_p$ norms $(p \geq 1)$ [2], given by the following general formula : if a particular vector $v = (\alpha_1, \ldots, \alpha_n)$,then

$$\mathscr{L}_p = \left(\sum_{i=1}^{n} |\alpha_i|^p \right)^{\frac{1}{p}}$$

e.g.

$$\mathscr{L}_1 = \sum_{i=1}^{n} |\alpha_i| \quad \mathscr{L}_2 = \left(\sum_{i=1}^{n} |\alpha_i|^2 \right)^{\frac{1}{2}} \quad \mathscr{L}_\infty = \max\left(|\alpha_i|\right)$$

However the corresponding notions of *lengths* are quite different as are evident from the unit **"circles"**

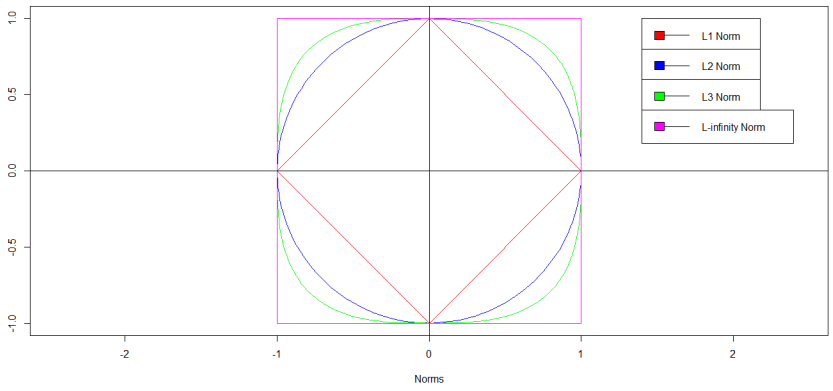

Figure 2.2: The unit *CIRCLE* in various *norms*

From Fig. 2.2, we see that the *unit circle* is a *"circle"* in case of only $\mathscr{L}_2$. The *unit circle* tends to *shrink* towards the **rhombic square** (as in case of $\mathscr{L}_1$ norm) and *expands* towards the pink square as p increases to ∞.

[2] For $p < 1$, the *triangle inequality* does not hold , hence $\mathscr{L}_p$ for such p is not a *norm* (strictly speaking)

2.5 Eigenvectors and Eigenvalues

While dealing with linear operators on vector spaces, have you come across the following question:

Are there any vectors which are mapped to a scalar multiple of itself?

Well, the answer is definitely yes if the *field* is $\mathbb{C}$, but such vectors may or may not exist if the *field* is $\mathbb{R}$.

Definition 2.2. *If a non-null vector $v \in V$ satisfies $Av = \lambda v$ for some $\lambda \in \mathbb{F}$, then v is an* eigenvector *of the matrix(transformation) A w.r.t* eigenvalue λ.

Example: *Reflection* along X-axis has *eigenvectors* $(1, 0)$ & $(0, 1)$ with respect to *eigenvalues* 1 & (-1) respectively.

Let matrix A have a *eigenvector* v w.r.t *eigenvalue* λ, then

$$\begin{aligned} Av = \lambda v &\implies A^2 v = A(\lambda v) = \lambda^2 v \\ &\implies \ldots\ldots\ldots\ldots \\ &\implies A^k v = \lambda^k v \end{aligned}$$

$\therefore$ A^k has the same *eigenvector* v w.r.t *eigenvalue* λ^k

Now the following question arises:

How to find out the eigenvalues for a particular given matrix A?

For this, we have to solve the following determinant[3] equation for λ

$$\det(A - \lambda\mathbb{I}) = 0$$

Remember this is only a necessary (not sufficient) condition for finding λ.The idea is that $Av = \lambda v$ implies $(A - \lambda\mathbb{I})v = 0$. But if $\det(A - \lambda\mathbb{I}) \neq 0$ then its *inverse* exists which would imply that v is identically 0, a contradiction.

A common problem that is often encountered in the solving determinants approach is the huge time complexity ($\mathbf{O}(n!)$) involved solely in the determinant evaluation. Another perhaps greater hurdle in evaluating λ (was proved by **Abel** namely the *Abel-Ruffini theorem*) is the fact that no polynomial with degree 5 or more has a general closed form solution. But seldom are matrices less than degree 5 in large computational practices(like those used in search engines!). In these atypical cases, numerical analysis is our only way out.

2.6 A few things about A^{-1}

The following statements are equivalent:

A^{-1} exists ... (i)

$\Leftrightarrow$ A is a square matrix with full row and column rank ... (ii)

$\Leftrightarrow \mathscr{R}(A) = \mathscr{C}(A) = \mathbb{R}^n$ ... (iii)

[3]For a better explanation of *determinants* ,follow the link `http://mathworld.wolfram.com/Determinant.html`

$\Leftrightarrow \{v_1, \ldots, v_n\}$ is a basis $\implies \{A(v_1), \ldots, A(v_n)\}$ is also a basis. ...(iv)
$\Leftrightarrow$ det(A) $\neq 0$ ...(v)

Outline of the proof goes as follows
$(i) \implies (ii) : AA^{-1} = I_n = A^{-1}A$. As I_n has rank n , so has A and A^{-1}
$(ii) \implies (iii)$: As A is $n \times n$, $\mathscr{C}(A) \subseteq \mathbb{R}^n$.But also $dim(\mathscr{C}(A)) = n$, so $\mathscr{C}(A) = \mathbb{R}^n$. Similarly, $\mathscr{R}(A) = \mathbb{R}^n$.
$(iii) \implies (i)$:

$$\begin{aligned} & \mathscr{C}(A) = \mathbb{R}^n \\ \implies & \exists \ \{u_1, \ldots, u_n\} \text{ such that } Au_i = \epsilon_i \ \forall \ i \in \{1, \ldots, n\} \\ \implies & A[u_1|u_2|\ldots|u_n] = [\epsilon_1|\epsilon_2|\ldots|\epsilon_n] = I \\ \implies & A^{-1}\text{exists} \end{aligned}$$

$(i) \implies (iv)$:

$$\begin{aligned} & \sum_{i=1}^{n} \alpha_i A v_i = 0 \\ \implies & A\left(\sum_{i=1}^{n} \alpha_i v_i\right) = 0 \\ \implies & \sum_{i=1}^{n} \alpha_i v_i = 0 \quad \left(\text{pre-multiplying with } A^{-1}\right) \\ \implies & \alpha_i = 0 \ \ \forall i \in \{1, \ldots, n\} \end{aligned}$$

$(iv) \implies (v)$: Sorry! The proof is beyond the scope , but is quite easy once the concept of determinants has been covered in class.
$(v) \implies (i)$: $A^{-1} = \frac{\mathbf{adj}(A)}{\mathbf{det}A}$

2.7 Factorisations of a Matrix

2.7.1 QR factorisation

Any square matrix A an be decomposed as

$$A = QR$$

where Q is an orthogonal matrix (its columns are orthogonal unit vectors meaning $Q^tQ = I$) and R is an upper triangular matrix. If A is invertible, then the factorization is unique if we require that the diagonal elements of R be positive.

Procedure: Assuming A to be full column rank, let us denote the i^{th} column as A_{*i} with *inner product* defined in the normal way, i.e. $< v, w >= v^*w$. Now using *Gram–Schmidt orthogonalisation*[4] we can find a set *orthonormal* basis vectors $\{e_1, \ldots, e_n\}$ from the set$\{A_{*1}, \ldots, A_{*n}\}$ such

[4]For a detailed description of the process check this video `https://www.khanacademy.org/math/linear-algebra/alternate_bases/orthonormal_basis/v/linear-algebra-the-gram-schmidt-process`

that

$$A_{*i} = \sum_{j=1}^{i} < e_j, A_{*k} > e_j$$

Now we construct our matrix Q and R as follows:

$$Q = [e_1, \ldots, e_n] \qquad R' = \begin{pmatrix} < e_1, A_{*1} > & < e_1, A_{*2} >, & \cdots & < e_1, A_{*n} > \\ 0 & < e_2, A_{*2} >, & \cdots & < e_2, A_{*n} > \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & < e_n, A_{*n} > \end{pmatrix}$$

Thus in this way we get an orthogonal matrix Q and an upper triangular matrix R'. Now, to get our R, we have to make all the diagonal entries positive; so we perform the following operations on R' & Q: If $< e_i, a_i >$ is negative, multiply the i^{th} row of R' as well as the i^{th} column of Q by (-1).Thus one particular characteristic of matrix R is that it has all of its diagonal elements positive.

Now we prove the uniqueness of QR factorisation.

Let A have two QR factorisations namely Q_1, R_1 & Q_2, R_2

$$\therefore A = Q_1 R_1 = Q_2 R_2$$

$$\implies Q_2^{-1} Q_1 = R_2 R_1^{-1} = P \text{ (say)}$$
$$\implies P \text{ is orthogonal(from LHS) as well as Upper triangular(from RHS)}$$
$$\implies P^t \text{ is Lower triangular}$$
$$\implies (P^t)^{-1} \text{ is Lower triangular}$$

Now $(P^t)^{-1} = P$ (as P is orthogonal). $\therefore P$ is both Upper and Lower Triangular. $\therefore P$ must be a diagonal matrix.

Now $I = PP^t = ((a_{ij}^2))$ where $((a_{ij}))$ are the elements of P. So we have $a_{ii}^2 = 1. \implies a_{ii} = \pm 1$. But as R_1 & R_2 both have positive diagonal entries (R has +ve diagonal entries $\implies R^{-1}$ also has positive diagonal entries) , so has $P = R_2 R_1^{-1}$.

$$\therefore P = I$$
$$\implies Q_2^{-1} Q_1 = R_2 R_1^{-1} = I$$
$$\implies Q_2 = Q_1 \text{ \& } R_2 = R_1 \qquad \text{Q.e.d}$$

2.7.2 LU factorization

In real life, we are often faced with solving a system of linear equations, in the general format $Ax = b$.The layman's technique would be to solve it using $x = A^{-1}b$ procedure , which from the very beginning has been a taboo.

So what now? A very efficient solution is provided by factorizing the given (nonsingular) matrix into a lower triangular and an upper triangular matrix in the way given below.

Let A be a square matrix. An LU factorization refers to the factorization of A, with proper row or column orderings or permutations, into two factors, a lower triangular matrix L and an upper triangular matrix U,

$$A = LU$$

The huge benefit is achieved in the following way:

Consider the system $Ax = b$ with LU factorization $A = LU$. Then we have

$$L\underbrace{Ux} = b;$$

$$=: y$$

$\therefore$ we can perform (a now familiar) 2-step solution procedure:
1. Solve the lower triangular system $Ly = b$ for y by forward substitution.
2. Solve the upper triangular system $Ux = y$ for x by back substitution.

In order to appreciate the usefulness of this approach note that the operations count for the matrix factorization is $O(m^3)$, while that for forward and back substitution is $O(m^2)$.

Another advantage of the technique is that suppose in a setup, you have the same *design matrix*(A) but different experimental values i.e. different right hand sides. In this case we need to compute the factorization $A = LU$ only once, and then

$$AX = B \Leftrightarrow LUX = B$$

and we proceed as before
1. Solve $LY = B$ by many forward substitutions (in parallel).
2. Solve $UX = Y$ by many back substitutions (in parallel).

However there can be cases when a square matrix is not factorisable into $A = LU$, even if it is non-singular. For e.g. if the leading element (or some leading minor) turns out to be zero.In such cases , we have to modify the original matrix so as to forcibly make it factorisable, i.e.

$$PA = LU$$

where P is a permutation matrix.The reason why we can multiply A with P is due to the fact that the permutation matrix changes *only* the order of rows (the equations) and nothing else and this doesn't change the solution set.

Algorithm: [5]

Initialize $U = A, L = I, P = I$
for $(k = 1 : m - 1)$
{
  find i $\geq$ k to maximize $|U(i, k)|$
  $U(k, k : m) \longleftrightarrow U(i, k : m)$
  $L(k, 1 : k - 1) \longleftrightarrow L(i, 1 : k - 1)$
  $P(k, :) \longleftrightarrow P(i, :)$
  for $(j = k + 1 : m)$
  {

[5] For a more detailed algorithm check this link : `https://www.youtube.com/watch?v=5hO3MrzPa0A`

$L(j,k) = U(j,k)/U(k,k)$
$U(j,k:m) = U(j,k:m) - L(j,k)U(k,k:m)$
}
}

Here $A \longleftrightarrow B$ means exchange A and B and $x : y$ implies from x to y.

There is another famous (and perhaps better) algorithm known as *Crout's Algorithm*[6], the discussion of which is beyond the scope of this lecture.

2.7.3 Eigenvalue factorisation / diagonalisation

For this type of factorisation,the matrix A must be a square matrix. with full rank. More specifically, , if A is a $n \times n$ matrix, and q_i ($i \in \{1 : k-1\}$) be the n distinct *eigenvectors*(a **must** for this type of factorisation), then A can be factorised into

$$A = Q\Lambda Q^{-1}$$

where Q is the square $n \times n$ matrix whose i^{th} column is the eigenvector q_i of A and Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, i.e., $\Lambda_{ii} = \lambda_i$.

Now the question is, why do we do this type of decomposition? These are the following benefits:

- It is the only form that helps us with the *intuitive* application of matrices.

  Let us look at a special property of this factorisation:

$$\begin{aligned} & A = Q\Lambda Q^{-1} \\ \implies & A^2 = Q\Lambda Q^{-1} Q\Lambda Q^{-1} \\ & = Q\Lambda (Q^{-1}Q)\Lambda Q^{-1} \\ & = Q\Lambda^2 Q^{-1} \qquad\qquad (\text{as } QQ^{-1} = I) \\ \implies & A^k = Q\Lambda^k Q^{-1} \qquad (\text{similarly using induction}) \end{aligned}$$

  What does a matrix, in general, do to a space? Scales it. It scales a vector(say v) to Av. Now what if we further scale the space,i.e. $A(Av)$? Intuitively , only the magnitude of scaling should change, not the directions right? This is what it exactly does, as is quite evident from the above property: only the scaling factor of each direction gets changed, not the directions themselves!

- It helps in reducing the number of *working dimensions* in high-dimensional vector spaces.

  This factorisation clearly tells the amount of scaling the matrix does (the λ_i's) to the different (orthogonal) directions (the eigenvectors of Q). If we sort them (the λ_i's) there may be seen a huge difference between the scaling the matrix does in different directions.

  What we mean is this: say in a 100 dimensional space, the first 20 dimensions get scaled by (say) above 1 and the rest by less than one. With such a matrix, if the space is scaled a huge number of times(as is often required), then those 80 dimension will some sort of remain unaffected by the matrix, while the initial 20 will dominate.

[6] A detailed description of Crout's algorithm can be found in Wikipedia or `http://faculty.ksu.edu.sa/Almutaz/Documents/Summer-2010/ChE-401/LU_decomposition.pdf`

So it isn't worthwhile working with all the 100 dimensions and trying to solve 100 degree polynomials. Instead we can, sort of, throw the worthless dimensions out, i.e. neglect those eigenvectors (and corresponding eigenvalues) which are very less scaled. Just think of how much easier it is to work with 20 rather than with 100!

This breaking up of a matrix into the **scaling** part and the **direction** part is only possible via eigenvalue decomposition.

- Another benefit, whose description, perhaps, is somewhat out of our scope ,is related to *matrix exponentials.*[7]

Singular Value Decomposition In a nutshell, the SVD is a more general form of factorisaion; namely given any rectangular matrix $M_{m\times n}$ it can be factorised into:

$$M_{m\times n} = U_{m\times\rho}\Sigma_{\rho\times\rho}(V_{n\times\rho})^*$$

where V^* denotes the adjoint of matrix V. As obvious, eigenvalue decomposition is a special case of this type of factorisation, where M needs to be a square matrix.Not only that, it must also be non-singular and diagonalizable[8]

References

[1] Wolfram MathWorld: `mathworld.wolfram.com`

[2] Stack Exchange : `http://math.stackexchange.com/`

[3] *Linear Algebra* by A.Ramachandra Rao & P. Bhimsankaram

[4] YouTube videos of lectures by Justin Solomon (`https://www.youtube.com/user/justinmsolomon`) and G. Strang (`https://www.youtube.com/watch?v=ZK3O402wf1c&list=PL49CF3715CB9EF31D`)

[7] To know more about matrix exponentials, see `http://mathworld.wolfram.com/MatrixExponential.html`. To know how this factorisation is useful, check(the bottom portion of) `http://mathworld.wolfram.com/EigenDecomposition.html`

[8] refer to this link for knowing what is meant by a diagonalizable matrix `https://en.wikipedia.org/wiki/Diagonalizable_matrix`