2.2.4 Functions of a matrix

In the same way that a function of a variable $f(x)$ can be constructed through its Taylor series, functions $f(\mathbf{M})$ of a matrix $\mathbf{M}$ can be defined through the corresponding Taylor series, e.g.

$$\exp(M) = \mathbf{1} + \mathbf{M} + \frac{\mathbf{M}^2}{2} + \cdots = \sum_{n=0}^{\infty} \frac{\mathbf{M}^n}{n!} . \tag{2.2.13}$$

Individual components of the matrix are obtained using standard rules of multiplication of matrices, e.g.

$$\exp(\mathbf{M})_{ij} = \delta_{ij} + M_{ij} + \frac{M_{ik}M_{kj}}{2} + \cdots . \tag{2.2.14}$$

Upon acting on an eigenvector,

$$\mathbf{M}^2\vec{e}^{\alpha} = \mathbf{M}\mathbf{M}\vec{e}^{\alpha} = \lambda_\alpha \mathbf{M}\vec{e}^{\alpha} = \lambda_\alpha^2 \vec{e}^{\alpha} , \text{ and similarly } \mathbf{M}^n\vec{e}^{\alpha} = \lambda_\alpha^n \vec{e}^{\alpha} . \tag{2.2.15}$$

Thus $\vec{e}^{\alpha}$ are eigenvectors of any function $f(\mathbf{M})$ of the matrix $\mathbf{M}$ with corresponding eigenvalues being $f(\lambda_\alpha)$. The action of the matrix function $f(\mathbf{M})$ on any vector $\vec{v}$ can then be calculated by the same procedure as used in calculating $x_i(t)$ in the previous section:

- Compute the coordinates of the vector $\vec{v}$ in the basis formed by the eigenvectors, as $a_\alpha = v_i e_i^\alpha$.

- Under the action of $f(\mathbf{M})$, each component in the eigenvector basis is multiplied by $f(\lambda_\alpha)$, i.e. $f(\mathbf{M})a_\alpha \vec{e}^{\alpha} = f(\lambda_\alpha)a_\alpha \vec{e}^{\alpha}$.

- From the components in the eigenvector basis we can reconstruct the coordinates in the original basis as

$$[f(\mathbf{M})v]_i = a_\alpha f(\lambda_\alpha) e_i^\alpha = v_j e_j^\alpha f(\lambda_\alpha) e_i^\alpha \equiv f(\mathbf{M})_{ij} v_j . \tag{2.2.16}$$

- Thus quite generally the elements of a matrix (in any basis) can be computed in terms of a sum over its eigenvectors and eigenvalues as

$$f(\mathbf{M})_{ij} = \sum_\alpha e_i^\alpha f(\lambda_\alpha) e_j^\alpha . \tag{2.2.17}$$

- Note that the trace of $f(\mathbf{M})$ is obtained as

$$f(\mathbf{M})_{ii} = \sum_\alpha f(\lambda_\alpha) e_i^\alpha e_i^\alpha = \sum_\alpha f(\lambda_\alpha) . \tag{2.2.18}$$

  since $e_i^\alpha e_i^\alpha = \vec{e}^{\alpha} \cdot \vec{e}^{\alpha} = 1$.

We can now see that the time evolution operation in Eq. (2.2.12) is carried out by the matrix $\mathbf{U}(t) = \exp(t\mathbf{M})$. Indeed this amount to solving the linear set of ODEs as

$$\frac{d\vec{x}}{dt} = \mathbf{M}\vec{x} \quad \Longrightarrow \quad \vec{x}(t) = \exp\left(t\mathbf{M}\right)\vec{x}(0)\,, \tag{2.2.19}$$

treating the vector of ODEs similar to one for a scalar x. However, treating matrices in functions and in equations as in the case of scalars has to be done very carefully, and fails in dealing with *non-commuting* matrices. The commuting property of two scalar quantities $XY = YX$ does not expend to matrices, and generically $\mathbf{X}\cdot\mathbf{Y} \neq \mathbf{Y}\cdot\mathbf{X}$. The Taylor series of a function of two variables must then be ordered appropriately as, for example $2\mathbf{X}\cdot\mathbf{Y} \neq \mathbf{X}\cdot\mathbf{Y} + \mathbf{Y}\cdot\mathbf{X}$.

Suppose we want to solve the ODE in Eq. (2.2.19) for a scalar $x(t)$, but with M that changes from M_1 after a time t_1 to M_2. After a subsequent time interval of t_2, we find

$$x(t_1+t_2) = \exp(t_2M_2)x(t_1) = \exp(t_2M_2)\exp(t_1M_1)x(0) = \exp(t_1M_1+t_2M_2)x(0)\,. \tag{2.2.20}$$

For the matrix version, the last step cannot be performed for non-commuting matrices, as

$$\vec{x}(t_1+t_2) = \exp(t_2\mathbf{M}_2)x(t_1) = \exp(t_2\mathbf{M}_2)\exp(t_1\mathbf{M}_1)x(0) \neq \exp(t_1\mathbf{M}_1+t_2\mathbf{M}_2)x(0)\,. \tag{2.2.21}$$

For a time-varying $\mathbf{M}_n$, the time evolution operator must strictly follow the ordering of matrices acting on the initial vector, i.e.

$$\mathbf{U}(t_1+t_2+\cdots t_N) = \exp(t_N\mathbf{M}_N)\exp(t_{N-1}\mathbf{M}_{N-1})\cdots\exp(t_2\mathbf{M}_2)\exp(t_1\mathbf{M}_1)\,. \tag{2.2.22}$$

In field theory, this is referred to as *path ordering* or *time ordering* of operators.