Combinatorial Identity Rule for the Sum of Matrix Determinants

Author:
- Sai Ganesh Reddy

Contact Information:
- Email: ganesh.sai432@gmail.com
- Git-Hub: Schrodingercat-tech
-YouTube: __sAi__
-Instagram: self.sai

We know that adding the determinants of matrices $\det(A)$ and $\det(B)$ doesn't equal the determinant of their sum $\det(A+B)$.

$$\det(A+B) \neq \det A + \det B$$

However, there's a captivating pattern: the determinant of the sum $\det(A+B)$ seems to have a connection with the sum of the determinants of $\det(A)$ and $\det(B)$. This intriguing relationship introduces an additional factor denoted as $c(A,B)$ into the equation.

$$\det(A+B) \ \equiv \ \det A + \det B + c(A,B)$$

Interestingly $c(A+B)$ is nothing but combinational arrangement's of these two matrices either by column or by row, lets see what we mean by that with an example

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}_{2\times2} \; ; \; B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}_{2\times2}$$

$$\det(A+B) = \det \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \end{pmatrix}_{2\times2}$$

Column Combinations

$$\det \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \end{pmatrix} \equiv \det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} + \det \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} + \det \begin{pmatrix} a_{11} & b_{12} \\ a_{21} & b_{22} \end{pmatrix} + \det \begin{pmatrix} b_{11} & a_{12} \\ b_{21} & a_{22} \end{pmatrix}$$

Row Combinations

$$\det \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \end{pmatrix} \equiv \det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} + \det \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} + \det \begin{pmatrix} a_{11} & a_{12} \\ b_{21} & b_{22} \end{pmatrix} + \det \begin{pmatrix} b_{11} & b_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$$LHS = (a_{11}+b_{11})(a_{22}+b_{22}) - (a_{12}+b_{12})(a_{21}+b_{21})$$

$$RHS_{row\ comb} = (a_{11}a_{22} - a_{12}a_{21}) + (b_{11}b_{22} - b_{12}b_{21}) + (a_{11}b_{22} - a_{12}b_{21}) + (b_{11}a_{22} - b_{12}a_{21})$$

Re-arranging terms

$$= (a_{11}a_{22} + a_{11}b_{22} + b_{11}a_{22} + b_{11}b_{22}) - (a_{12}a_{21} + a_{12}b_{21} + b_{12}a_{21} + b_{12}b_{21})$$

$$= (a_{11}(a_{22}+b_{22}) + b_{11}(a_{22}+b_{22})) - (a_{12}(a_{21}+b_{21}) + b_{12}(a_{21}+b_{21}))$$

$$= (a_{11}+b_{11})(a_{22}+b_{22}) - (a_{12}+b_{12})(a_{21}+b_{21})$$

$$LHS \equiv RHS_{row\ comb} \equiv RHS_{column\ comb}$$

From above we can clearly see both LHS and RHS are equal, we can extend this method to nxn grid note that the index positions of the elements remain the same, the elements themselves are altered based on their positions in either the column or row arrangements. This extension allows for the computation of determinants for square matrices of any size.

With increase in dimension of matrices the number of elements will also increases. The total number of terms will be equal to

$$no\,of\,terms(T) \;=\; \sum_{r=o}^{r=n} {}^{n}C_r \;\; where \;\; {}^{n}C_r \;=\; \frac{n!}{(n-r)!r!}$$

if we notice closely this similar to Binomial Function

$$(a+b)^n \;= \sum_{r=o}^{r=n} {}^{n}C_r a^{n-r} b^r$$

$$a, b = 1$$

$$2^n = \sum_{r=o}^{r=n} {}^{n}C_r$$

also if we you notice for 2 term determinant we have 2^n terms similarly we can have t terms
and their corresponding terms will be t^n

$$(1+a)^n = \sum_{r=o}^{r=n} {}^{n}C_r a^r$$

$$a := a+1$$

$$(1+(a+1))^n = \sum_{r_1=o}^{r=n} \sum_{r_2=o}^{r_2=r_1} \binom{n}{r_1}\binom{r_1}{r_2} a^r \;\; ; \;\; \binom{n}{r} = {}^{n}C_r$$

$$3^n \;=\; \sum_{r_1=o}^{r=n} \sum_{r_2=o}^{r_2=r_1} \binom{n}{r_1}\binom{r_1}{r_2} ; \; at\; a = 1$$

$$t^n = \sum_{r_1=o}^{r=n} \sum_{r_2=o}^{r_2=r_1} \cdots \sum_{r_{t-1}=o}^{r_{t-2}=r_{t-1}} \binom{n}{r_1}\binom{r_1}{r_2} \cdots\cdot \binom{r_{t-2}}{r_{t-1}}$$

In General for a given set of square matrices, we transform them into a rank-3 tensor $a_{ij}^{(\lambda)}$ where λ controls the terms of the matrices. To account for the interchangeability of rows or columns, we create a subset of λ values, considering all possible combinations. The determinant is then calculated for each arrangement, and the sum of these determinants yields the same result as determinant of the sum of matrices.

$$A^{(\lambda)} = \begin{pmatrix} a_{11}^{(\lambda)} & a_{12}^{(\lambda)} & \dots & a_{1n}^{(\lambda)} \\ a_{21}^{(\lambda)} & a_{22}^{(\lambda)} & \dots & a_{2n}^{(\lambda)} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(\lambda)} & a_{n2}^{(\lambda)} & \dots & a_{nn}^{(\lambda)} \end{pmatrix}_{(n\times n)}$$

$$\det\left(A^{(0)} + A^{(1)} + \dots + A^{(t-1)}\right) \equiv \sum_{(\lambda_1,\lambda_2 \dots \lambda_n) \subset \, all \ comb}^{terms=t^n} \det \begin{pmatrix} a_{11}^{(\lambda_1)} & a_{12}^{(\lambda_1)} & \dots & a_{1n}^{(\lambda_1)} \\ a_{21}^{(\lambda_2)} & a_{22}^{(\lambda_2)} & \dots & a_{2n}^{(\lambda_2)} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(\lambda_n)} & a_{n2}^{(\lambda_n)} & \dots & a_{nn}^{(\lambda_n)} \end{pmatrix}_{(n\times n)}$$

$$where\ (\lambda_1, \lambda_2 \dots \lambda_n)_{1\times n} \subset all\ combinations\ of\ terms_{(t^n, n)}$$

for example in 2 and 3 terms of 2x2 and 3x3 matrices we get 4 and 27 terms

$$t \to terms\ ,\ n \to\ no\ of\ rows\ or\ columns$$

$$terms\ =\ t^n$$

$$(\lambda_1, \lambda_2) \subset [(0,\ 0),\ (0,\ 1),\ (1,\ 0),\ (1,\ 1)] \to 2^2 = 4\ terms$$

$$\begin{gathered}(\lambda_1, \lambda_2, \lambda_3) \subset [(0,\ 0,\ 0),\ (0,\ 0,\ 1),\ (0,\ 0,\ 2),\ (0,\ 1,\ 0),\ (0,\ 1,\ 1),\ (0,\ 1,\ 2), \\ (0,\ 2,\ 0),\ (0,\ 2,\ 1),\ (0,\ 2,\ 2),\ (1,\ 0,\ 0),\ (1,\ 0,\ 1),\ (1,\ 0,\ 2),\ (1,\ 1,\ 0), \\ (1,\ 1,\ 1),\ (1,\ 1,\ 2),\ (1,\ 2,\ 0),\ (1,\ 2,\ 1),\ (1,\ 2,\ 2),\ (2,\ 0,\ 0),\ (2,\ 0,\ 1) \\ ,\ (2,\ 0,\ 2),\ (2,\ 1,\ 0),\ (2,\ 1,\ 1),\ (2,\ 1,\ 2),\ (2,\ 2,\ 0),\ (2,\ 2,\ 1),\ (2,\ 2,\ 2)] \to 3^3 = 27\ terms\end{gathered}$$

To check rule correctness I have implemented code below and for the results you can check my git-hub :

```python
import numpy as np
import itertools
from scipy.special import comb

class DetRule:

    def __init__(self,*Matrices: np.ndarray) -> None:
        self.mats = np.array(Matrices) # (i,j,k) i-th matrix j-th row k-th col
        self.shape = self.mats.shape
        assert self.shape[1]==self.shape[2],\
                'Check if all matrices are squared and of same dimension'
        self.terms = self.mats.shape[0]**self.mats.shape[1] # (t,n,n) ->t^n
        self.addMats = sum([self.mats[i]
                            for i in range(self.shape[0])])
        self.addedMatsDet = np.linalg.det(self.addMats)
        self.matsDet = [np.linalg.det(self.mats[i])
                          for i in range(self.shape[0])]

    def ComMats(self):
        terms = np.arange(self.shape[0])
        for combination in itertools.product(terms,repeat=self.shape[1]):
            yield np.array([self.mats[term][row]
                            for row,term in enumerate(combination)])

    def ComMatsDet(self):
        for mat in self.ComMats():
            yield np.linalg.det(mat)

    def ComMatDetResult(self):
        Sum = 0
        for result in self.ComMatsDet():
            Sum+=result
        return Sum

```