

线性代数矩阵操作函数

小圆滚滚

1 基坐标旋转

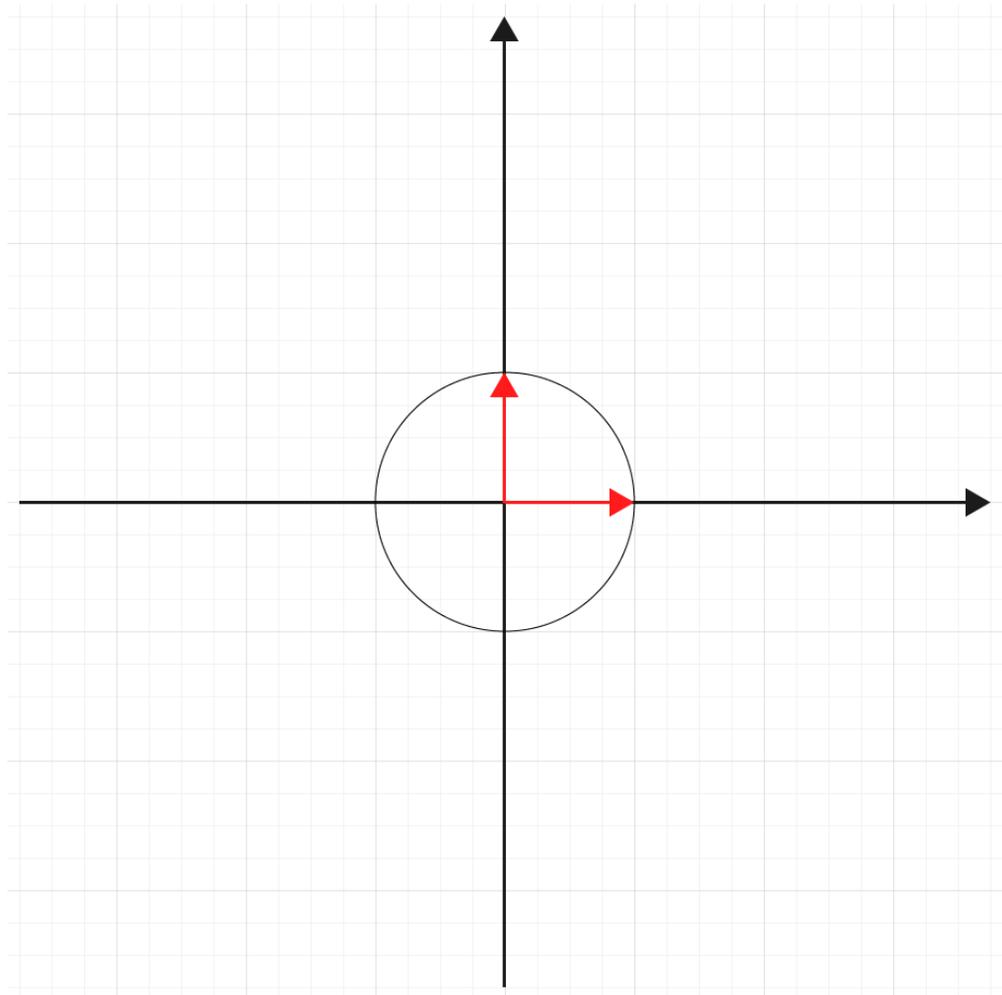


图 1: view1

在图1视角中，基可以由两个向量 $\hat{i} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\hat{j} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ 表示。这里称为我们的视角。

在图1视角的基础上将基旋转成另一个线性空间 V_2 的基，在我们的视角，看 V_2 的基，就有我们的坐

标: $\hat{i} = \begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix}$, $\hat{j} = \begin{bmatrix} -\sin\varphi \\ \cos\varphi \end{bmatrix}$

标，转换到我们视角下的坐标。需要他人给出的他们的分量顺序下分别的投影长度，转换到我们视角下分量顺序下分别的投影长度。

将这些分量顺序下的投影相加，就可以得到最终他们描述的量呈现在我们坐标系下的分量，按顺序列出，得到一个向量的坐标。

经过上面的推导，我们可以将结论转换为技巧，就是，如果别人在他的视角下给出了一个向量（也就是坐标，也就是一个数组），那么我们在我们的地图上标出，首先要将对方的 \hat{i}, \hat{j} 排一个矩阵，这个矩阵由列向量们从左到右组成，列向量的顺序按照 $\hat{i}, \hat{j}, \hat{k}$ （三维）的顺序。 $\hat{i}, \hat{j}, \hat{k}$ 就是他们的单位向量，对

应存在于我们的单位向量，x轴 $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ ，y轴 $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ ，z轴 $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ 上的投影。

为什么垂直正交？因为降维之后，要保证所有的子群都平等。

左乘再左乘是一个顺序改变的过程，

单次左乘可以看作是左边的方阵对右边的单列向量起到坐标投影映射的过程。

左边的方阵就是在我们的视角下看到的别人的基在我们的坐标下相对于我们的基的读数们。

基变换定义为：

$$\begin{cases} \beta_1 = p_{11}\alpha_1 + p_{12}\alpha_2 + \dots + p_{n1}\alpha_n \\ y_2 = p_{12}\alpha_1 + p_{22}\alpha_2 + \dots + p_{n2}\alpha_n \\ \dots \\ y_n = p_{1n}\alpha_1 + p_{2n}\alpha_2 + \dots + p_{nn}\alpha_n \end{cases}$$

注意行列标与矩阵的线性方程组表示是转置关系。把 $\alpha_1, \alpha_2, \dots, \alpha_n$

这n个有序向量记作 $(\alpha_1, \alpha_2, \dots, \alpha_n)$ ，

记n阶矩阵 $P = (p_{ij})$ ，利用向量和矩阵的形式，上式可表示为：

$$(\beta_1, \beta_2, \dots, \beta_n) = (\alpha_1, \alpha_2, \dots, \alpha_n)P$$

简单说，一个方阵（基），等于另一个方阵（基）右乘一个方阵，那么坐标变换公式如下：

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = P \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix}, \text{ 反之 } \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix} = P^{-1} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

基变换是右乘，且右乘的这个矩阵在线性方程组的右下角标行序号和列序号是转置的。原因就是满足矩阵左乘的基础来源。 β_1 是一个向量，并不是一个数值，这个向量是构成基的多个维度中的一个方向上的向量，它由过度矩阵 P 。

基变换的底层逻辑就是在一个n维线性空间中，如果基，变成另一个基，那么过度矩阵必定是一个n维的方阵。这就是线性的真正意义。他不会导致升维和降维。

我之前在知乎看到一个博主将线性代数的基变换说成是旋转，然后继续左乘变换矩阵，我说左乘和右乘一定要注意，但是他仅仅是在过程当中，因为使用了旋转就认为简单的变换坐标而忽视了坐标变换的推导法则。没错，对于具体的问题，因为巧合而把过度矩阵直接由坐标写出，但是那是因为你用了我们视角下的默认基坐标（默认视角下的坐标，那组基） $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ 去左乘了 P ，当然没有任何变化。新的基让你忽略了右乘的普遍关系式，你也就学不到普适的精髓。也就压根儿没有探索触及到右乘的奥秘。

行列式的秩可以描述矩阵张成的空间是不是能够达到矩阵的维度，面积和体积的值可以描述被拉扯的大小。

说人话就是，如果你找到了一个方阵，s可以将你现在视角下的基转换为别人视角下的基（用你的

基去右乘这个过度矩阵)。那么对方给的坐标，你用过度矩阵左乘就可以转换成你的视角下的读数。或者，你的坐标左乘过度矩阵的逆，就可以给他去读出你要表达的意思。

右乘， $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ 表示将左边的方阵第一列减去第二列，像是操作步骤。第三步就是 $\begin{pmatrix} 1 \\ -1 \\ \text{第三步} \end{pmatrix}$

第三步就是去用上面两列的结果进行第三列的组合，也就是用第三个列向量（生成向量）来补足前两列的不足。

可以说左边的方阵改变了右边的坐标，这个改变，操作，就是一个函数，将坐标映射到原有的线性空间。

也可以说右边的输入是左边列向量组合体的系数，举例来说，右边输入的第一个数值，就是左边列向量组合体的第一列向量的所有元素的公共系数，告诉他们如何应对接下来的第二个数值调教出来的第二列向量的所有元素的组合步骤。

解基底的过程是一个寻找零空间向量的过程。这个输入可以让

故事的开始是这样设计的，不管你这辈子怎么折腾，你的结果都是已知的。你不可能消失，也不可能毫无来由。你存在的这个群，经过一番运算，最后还是要落到这个群里。

那么光是这样也只是保证了一个空间。还需要线性。

线性就是一一对应，在这个空间里你总能找出一套规则，这个规则由简单的三条法则完成，也可以用在这个空间里的个体来认证。

这三条法则就是1.你可以复制任意多个你（自己）抱在一起，2.你可以和很多不同的类似你的人抱在一起。你们的力气都会叠加，3.但是单位力气（就是以你自己作为标准）是你距离一个验收点的距离，这个验收点到每个人的距离都不同。

具体问题具体分析。具体问题都是线性空间的问题。所以惯性系、所以等效原理对每个人都很公平。

一个认证方案就是，不管哪个人，他的工作，都可以由公司内几个不同的人分摊完成。这个公司的工种多少，决定了这个空间的复杂维度。技能相同的人将会变成冗余的人，并不能带给公司质的提升。工种，比如：信号收发、信息处理、GPS增强、系统监控、时间同步、电源保障。你每天会在这六个岗位上各工作一个小时，有你擅长的领域，你的产出就多一些，虽然找不到和你一样的人一天干出同样的事却是有可能的。只需要这六个队里面出一个人，告诉他你干了多少，他们就可以复制出你来，天天干出和你今天一样的工作。你就相当于被别人取代了。

线性空间有这样的特点，你可以找出一个向量，它距离原点的距离可以投影在一个线性无关的几个维度上。这几个维度撑起了这个线性空间。维度之间线性无关。并不是这几个维度唯一确定了这个线性空间。而是有很多可能性都可以撑出这个线性空间。而只要是能撑出这个线性空间的维度，他们都可以仅仅通过线性变换就可以完成基底的对应转换。

等比数列是维度升降的bug奥义。可以由他来演示数学归纳法，可以由他来开方降维加合低纬度面积得出泰勒公式。

eigenvalue(自己的、本征)值

网格可以捕到鱼，把钩子绑在线上，只能有一串鱼

巧合就是一种预示，可能你会发现万能钥匙，也可能是一种警示

平行四边形面积是行列式的根基，也是矩阵，线性空间变换的根基之一，另一个根基是原点不变
克莱默法则可以求出系数

线性代数基本定理：总维数=矩阵的秩（列空间维数）+零空间维数

代数基本定理：n次多项式在复数域里一定有n个根。

几何是人类的思维方式，迭代循环是人的思维方式，递归循环是神的思维方式，代数是唯一能够证明推演的思维方式。

平行四边形，底乘以高，侧边被拉长，那么侧边的高就变短，才能割补

Q:如何求一个矩阵的逆矩阵

A:求矩阵的逆矩阵是一个数学问题，通常使用线性代数中的矩阵运算来解决。以下是求矩阵逆矩阵的基本步骤：

1. **确定矩阵是否可逆**：一个矩阵可逆的前提是它是一个方阵，即行数和列数相等，并且它的行列式（determinant）不为零。

2. **计算行列式**：对于一个2x2矩阵

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

，其行列式

$$\det(A) = ad - bc$$

。对于更大的矩阵，计算行列式可能更复杂。

3. **如果行列式为零**：如果

$$\det(A) = 0$$

，则矩阵

$$A$$

不可逆，即不存在逆矩阵。

4. **计算伴随矩阵**：伴随矩阵（adjugate matrix）是将矩阵的每个元素替换为其代数余子式（cofactor），然后转置得到的矩阵。对于2x2矩阵，伴随矩阵

$$\text{adj}(A) = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

。

5. **求逆矩阵**：如果矩阵可逆，其逆矩阵可以通过伴随矩阵除以行列式来得到，即

$$A^{-1} = \frac{1}{\det(A)} \cdot \text{adj}(A)$$

。

6. **对于更大的矩阵**：对于3x3或更大的矩阵，计算逆矩阵的过程更为复杂，通常使用高斯消元法或者伴随矩阵的方法。

7. **使用计算工具**：在实际应用中，通常使用计算工具（如MATLAB、NumPy等）来求矩阵的逆，因为手动计算可能非常繁琐。

请注意，这里描述的是理论步骤，实际应用中可能需要根据具体情况进行调整。如果你需要具体的计算示例或者使用特定工具的方法，请提供更多的信息。

Q:初等变换法求逆矩阵举例

A:初等变换法，也称为高斯-若尔当消元法，是一种求矩阵逆矩阵的方法，特别适用于求解线性方程组。这种方法利用初等行变换将矩阵转换为单位矩阵，然后记录这些变换，最后将这些变换应用到单位矩阵上，得到原矩阵的逆矩阵。以下是使用初等变换法求逆矩阵的一个例子：

假设我们有一个2x2矩阵

$$A$$

，我们希望求其逆矩阵

$$A^{-1}$$

。矩阵

$$A$$

如下所示：

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

步骤如下：

1. **构造增广矩阵**：将矩阵

$$A$$

和单位矩阵

$$I$$

放在一起，形成一个增广矩阵：

$$\left[\begin{array}{cc|cc} a & b & 1 & 0 \\ c & d & 0 & 1 \end{array} \right]$$

2. **进行初等行变换**：通过行交换、行乘以非零常数、行相加或相减，将左边的矩阵部分转换成单位矩阵。对于2x2矩阵，这通常涉及以下步骤：- 确保左上角的元素是非零的（如果需要，可以交换行）。- 将第一行乘以

$$1/a$$

，使得左上角元素变为1。- 通过行减法，将第二行的

$$c$$

变为0。

3. **转换为单位矩阵**：在上述步骤中，左边的矩阵部分将变为单位矩阵

$$I$$

:

$$\left[\begin{array}{cc|cc} 1 & * & 0 & 0 \\ 0 & 1 & * & * \end{array} \right]$$

4. **得到逆矩阵**：此时，增广矩阵的右边部分就是原矩阵

$$A$$

的逆矩阵

$$A^{-1}$$

:

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

5. **计算逆矩阵的元素**：根据步骤3中得到的矩阵，计算逆矩阵的元素。对于2x2矩阵，逆矩阵的元素可以通过以下公式计算：

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

如果

$$ad - bc \neq 0$$

，那么矩阵

$$A$$

可逆，逆矩阵的元素就是上述公式中的

*

。

6. **特殊情况**：如果行列式

$$ad - bc = 0$$

，则矩阵

$$A$$

不可逆，没有逆矩阵。

这个例子展示了如何使用初等变换法求2x2矩阵的逆矩阵。对于更大的矩阵，过程类似，但需要更多的行变换步骤。

以下是使用初等变换法求逆矩阵的一个例子：

假设我们要求矩阵 $A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$ 的逆矩阵。

首先，构造一个增广矩阵 $[A|I]$ ，其中 I 是二阶单位矩阵 $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ ，即：

$$\begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

然后进行初等行变换：

第一步，将第一行乘以-1加到第二行，得到：

$$\begin{bmatrix} 2 & 1 & 1 & 0 \\ -1 & 0 & -1 & 1 \end{bmatrix}$$

第二步，将第一行除以2，得到：

$$\begin{bmatrix} 1 & 1/2 & 1/2 & 0 \\ -1 & 0 & -1 & 1 \end{bmatrix}$$

第三步，将第一行乘以1加到第二行，得到：

$$\begin{bmatrix} 1 & 1/2 & 1/2 & 0 \\ 0 & 1/2 & -1/2 & 1 \end{bmatrix}$$

第四步，将第二行乘以2，得到：

$$\begin{bmatrix} 1 & 1/2 & 1/2 & 0 \\ 0 & 1 & -1 & 2 \end{bmatrix}$$

第五步，将第二行乘以-1/2加到第一行，得到：

$$\begin{bmatrix} 1 & 0 & 1 & -1 \\ 0 & 1 & -1 & 2 \end{bmatrix}$$

此时，左边变成了单位矩阵，右边就是原矩阵的逆矩阵，所以矩阵A的逆矩阵 $A^{-1} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$

求逆矩阵用伴随矩阵快还是用增广矩阵快

求逆矩阵的方法中，使用伴随矩阵和使用增广矩阵（初等行变换法）各有优缺点，速度的快慢取决于多种因素，包括矩阵的大小、类型以及你使用的计算工具。下面是两种方法的一些比较：

1. **伴随矩阵法**： - 优点：公式简单，容易理解和记忆。 - 缺点：对于大矩阵，计算伴随矩阵和行列式可能非常繁琐，特别是行列式计算容易出错。 - 速度：对于小矩阵（如2x2或3x3），这种方法通常较快。但对于更大的矩阵，计算效率可能会降低。

2. **增广矩阵法（初等行变换法）**： - 优点：适用于求解线性方程组，可以同时得到矩阵的逆矩阵和线性方程组的解。 - 缺点：需要进行较多的行变换操作，对于初学者来说可能比较复杂。 - 速度：对于大矩阵，这种方法通常更快，因为现代计算机和软件库（如MATLAB、NumPy等）对行变换进行了优化。

3. **计算工具**： - 在实际应用中，大多数情况下会使用计算工具来求逆矩阵，因为它们可以自动处理复杂的计算和优化算法。

4. **特殊情况**： - 如果矩阵具有特殊结构（如对称矩阵、对角矩阵等），某些方法可能会更快。

5. **数学软件和库**： - 专业的数学软件和库通常使用高效的算法来求逆矩阵，这些算法可能结合了多种方法的优点。

总结来说，没有绝对的答案说哪种方法更快，这取决于具体情况。在实际应用中，推荐使用专业的数学软件或编程库来求逆矩阵，因为它们提供了准确、快速的解决方案。如果你需要手动计算，可以根据矩阵的大小和特点选择适合的方法。对于小矩阵，伴随矩阵法可能更简单；对于大矩阵或需要求解线性方程组的情况，增广矩阵法可能更合适。

消元法就是矩阵的行等价。

$$\text{行列式}|A| = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = \sum (-1)^t a_{1p_1} a_{2p_2} \cdots a_{np_n} \text{ 的各个元素的代数余子式}$$

$$A_{ij} = (-1)^{i+j} M_{ij} \text{ 所构成的矩阵 } A^* = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{pmatrix}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

$$AA^* = |A|E$$

$$A^{-1} = \frac{1}{|A|} A^*$$

在线性代数中，一个方形矩阵的伴随矩阵（英语：adjugate matrix）是一个类似于逆矩阵的概念。如果矩阵可逆，那么它的逆矩阵和它的伴随矩阵之间只差一个系数。然而，伴随矩阵对不可逆的矩阵也有定义，并且不需要用到除法。A的伴随矩阵记作adj(A)，或A*。

以下来自kimi:

Q:为什么线性空间中基变换只需要一个方阵?

A:在数学中,线性空间(也称为向量空间)是一个集合,其中的元素称为向量,可以进行加法和标量乘法运算。基(Basis)是线性空间中的一组向量,任何该空间中的向量都可以唯一地表示为这组基向量的线性组合。

基变换是从一个基转换到另一个基的过程。基变换可以通过矩阵乘法来实现。这里涉及到的方阵,我们称之为变换矩阵或基变换矩阵。

为什么只需要一个方阵来实现基变换,原因如下:

1. **线性组合**:在向量空间中,任何向量都可以表示为基向量的线性组合。基变换实际上是改变了这种线性组合的系数。

2. **矩阵乘法**:矩阵乘法是一种线性运算,可以用来表示线性变换。当我们有一个基变换矩阵 P 时,任何向量 v 在新基下的表示 v' 可以通过 $v' = P \cdot v$ 来得到。

3. **坐标变换**:基变换矩阵实际上是将旧基下的坐标转换为新基下的坐标。如果我们有一个向量 v 在旧基下的坐标表示为 $[v]_{\text{old}}$,那么在新基下的坐标 $[v]_{\text{new}}$ 可以通过 $[v]_{\text{new}} = P \cdot [v]_{\text{old}}$ 来计算。

4. **方阵的性质**:方阵是一个 $n \times n$ 的矩阵,其行数和列数与基的维数相同。方阵的逆矩阵 P^{-1} 可以用来将变换从新基逆回到旧基,即 $[v]_{\text{old}} = P^{-1} \cdot [v]_{\text{new}}$ 。

5. **唯一性**:基变换矩阵 P 是唯一的,只要它是可逆的(即行列式不为零)。这意味着对于任何两个基,都存在唯一的变换矩阵来描述它们之间的关系。

6. **线性空间的结构保持不变**:基变换不改变线性空间的结构,只是改变了描述向量的方式。方阵 P 确保了线性空间中的线性关系在变换后仍然保持。

总结来说,基变换只需要一个方阵,因为它通过矩阵乘法提供了一种系统的方法来转换向量在不同基下的坐标表示,同时保持了线性空间的结构不变。

Q:为什么基变换矩阵 P 是唯一的?

A:在数学中,线性空间(也称为向量空间)是一个集合,其中的元素称为向量,可以进行加法和标量乘法运算。基(Basis)是线性空间中的一组向量,任何该空间中的向量都可以唯一地表示为这组基向量的线性组合。

基变换是从一个基转换到另一个基的过程。基变换可以通过矩阵乘法来实现。这里涉及到的方阵,我们称之为变换矩阵或基变换矩阵。

为什么只需要一个方阵来实现基变换,原因如下:

1. **线性组合**:在向量空间中,任何向量都可以表示为基向量的线性组合。基变换实际上是改变了这种线性组合的系数。

2. **矩阵乘法**:矩阵乘法是一种线性运算,可以用来表示线性变换。当我们有一个基变换矩阵 P 时,任何向量 v 在新基下的表示 v' 可以通过 $v' = P \cdot v$ 来得到。

3. **坐标变换**:基变换矩阵实际上是将旧基下的坐标转换为新基下的坐标。如果我们有一个向量 v 在旧基下的坐标表示为 $[v]_{\text{old}}$,那么在新基下的坐标 $[v]_{\text{new}}$ 可以通过 $[v]_{\text{new}} = P \cdot [v]_{\text{old}}$ 来计算。

4. **方阵的性质**:方阵是一个 $n \times n$ 的矩阵,其行数和列数与基的维数相同。方阵的逆矩阵 P^{-1} 可以用来将变换从新基逆回到旧基,即 $[v]_{\text{old}} = P^{-1} \cdot [v]_{\text{new}}$ 。

5. **唯一性**:基变换矩阵 P 是唯一的,只要它是可逆的(即行列式不为零)。这意味着对于任何两个基,都存在唯一的变换矩阵来描述它们之间的关系。

6. **线性空间的结构保持不变**:基变换不改变线性空间的结构,只是改变了描述向量的方式。方阵 P 确保了线性空间中的线性关系在变换后仍然保持。

总结来说,基变换只需要一个方阵,因为它通过矩阵乘法提供了一种系统的方法来转换向量在不同基下的坐标表示,同时保持了线性空间的结构不变。

21: 注意看上面第三条, 首先, 确定线性空间基变换的过度矩阵有且仅有一个, 然后才有了P这个矩阵的具体含义, 就是“将旧基下的坐标转换为新基下的坐标”。如果 $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 则旋转之后为 $\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$, 如果是其他, 那么P自然可以求出。

平行四边形法则, 在面积相等的时候, 剪贴, 剪一个角贴到对边, 拉伸了周长, 但是缩短了高。用圆球来算面积的思考方式就不对了, 微积分平行四边形, 一定要用小正方形, 然后割补法就能想明白了。