Vetores em Geometria Analítica

O que segue aqui é apenas uma amostra de utilização do Maple V no estudo de vetores em Geometria Analítica.

Vetores

Consideramos vetores dados em coordenadas , no plano 2D ou no espaço 3D , em relação a alguma base ordenada.

```
> restart; # limpando resquícios anteriores
> with(linalg) : # chamando o pacote de Álgebra Linear
```

Definindo um vetor 2D ou 3D dadas as suas coordenadas:

```
> v := vector( [3, 4] );
  w := vector( [2, 3, 4] );
```

$$v := \begin{bmatrix} 3 & 4 \end{bmatrix}$$

$$w := \begin{bmatrix} 2 & 3 & 4 \end{bmatrix} \quad (1)$$

Definindo um vetor 2D ou 3D dadas as suas coordenadas:

```
> v[1];
  w[3];
```

$$3$$

$$4 \quad (2)$$

Definindo um vetor 2D ou 3D literal, para definição posterior das coordenadas:

```
> w := vector(3); print(w);
```

$$w := \begin{bmatrix} ?_1 & ?_2 & ?_3 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \quad (3)$$

Dependência e independência linear

Para analisar dependência e independência linear, via coordenadas, devemos analisar matrizes construídas com os vetores:

Se o posto da matriz cujas linhas (ou colunas) são as coordenadas dos vetores for exatamente o número de vetores, então os vetores são l.i. (lineramente independentes) , caso contrário são l.d. (linearmente dependentes).

Construindo a matriz cujas linhas [colunas] são as coordenadas dos vetores:

> $u := vector(\ [1, 2, 3]); v := vector(\ [4, 5, 6]);$
>
> $w := vector(\ [7, 8, 9]);$

$$u := \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$
$$v := \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$$
$$w := \begin{bmatrix} 7 & 8 & 9 \end{bmatrix} \quad \textbf{(4)}$$

> $A := stackmatrix(u, v, w);$ *# juntando como linhas*

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \textbf{(5)}$$

> $M := concat(u, v, w);$ *# juntando como colunas*

$$M := \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \quad \textbf{(6)}$$

> $augment(u, v, w);$ *# o mesmo que concat*

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \quad \textbf{(7)}$$

Obs : stack , concat e augment não manipulam somente com vetores. Pode-se misturar com matrizes, desde que de tamanhos coerentes, já que tanto vetores como matrizes são arrays .

> $stackmatrix(\ A, v); concat(\ M, v, w);$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 & 7 & 4 & 7 \\ 2 & 5 & 8 & 5 & 8 \\ 3 & 6 & 9 & 6 & 9 \end{bmatrix} \qquad \textbf{(8)}$$

Calculando o posto (característica) da matriz A

> *rank*(A);

$$2 \qquad \textbf{(9)}$$

> *rank*(M); *# Observe que M é a transposta de A e portanto, deve ter o mesmo posto*

$$2 \qquad \textbf{(10)}$$

Lembramos que o posto de A é o número de linhas não nulas de uma matriz escada l-equivalente a A :

> *gausselim*(A); *gaussjord*(A);

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix} \qquad \textbf{(11)}$$

Analisando a dependência linear

Como o número de vetores para compor A era 3 (vetores u, v e w) e o posto de A é 2, os vetores em questão são l. d. O posto sendo 2, é possível extrair no máximo 2 vetores l. i. do conjunto de 3.

> $B :=$ *stackmatrix*(u, v); *rank*(B);

$$B := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$
$$2 \qquad \textbf{(12)}$$

Como os dois primeiros são l.i ., o terceiro, w, pode ser obtido como c.l. (combinação linear) dos dois primeiros.

(*) Ver tópico abaixo, sobre combinações lineares.:

```
> linsolve(transpose(B), w);
```

$$\begin{bmatrix} -1 & 2 \end{bmatrix} \tag{13}$$

Obs: É sempre bom lembrar alguns detalhes para evitar de fazer contas desnecessárias. Por exemplo:

1. Vetor nulo é combinação linear de quaisquer outros vetores. Logo, qualquer conjunto de vetores contendo o vetor nulo entre eles é sempre l.d .

2. Três ou mais vetores no plano são sempre l.d . , assim como quatro ou mais no espaço 3D. Ou seja, o número máximo de vetores l.i . num espaço é a dimensão do espaço.

Combinação linear

```
> restart; with(linalg) :
```

Uma combinação linear dos vetores $u[1]$, ..., $u[n]$ é um vetor da forma $\alpha[1]u[1]$+ ... + $\alpha[n]u[n]$, onde α[1] , ..., α[n] são escalares .

Vimos acima um exemplo de como obter um vetor como combinação linear de outros.

```
> u := vector([1, 2, 3]); v := vector([4, 5, 6]); w := vector([7, 8, 9]);
  M := concat(u, v);
```

$$u := \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$
$$v := \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$$
$$w := \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}$$
$$M := \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \tag{14}$$

```
> X := linsolve(M, w);
```

$$X := \begin{bmatrix} -1 & 2 \end{bmatrix} \tag{15}$$

A resposta acima indica que w = -u + 2v , como podemos checar:

```
> evalm(-u + 2*v);
```

$$\begin{bmatrix} 7 & 8 & 9 \end{bmatrix} \tag{16}$$

Produtos interno, vetorial e misto

O produto interno ou escalar de dois vetores é um escalar . Todas as coordenadas são segundo uma base ortonormal de vetores (2 a 2 ortogonais e unitários).

O produto vetorial de dois vetores resulta num vetor ortogonal aos dois. As coordenadas devem ser dadas segundo uma base ortonormal positiva do espaço. É um produto exclusivo do espaço 3D.

```
> restart; with(linalg) :
> u := vector( [2, 3, 4]); v := vector( [3,-4, 5]);
```

$$u := \begin{bmatrix} 2 & 3 & 4 \end{bmatrix}$$

$$v := \begin{bmatrix} 3 & -4 & 5 \end{bmatrix} \tag{17}$$

```
> dotprod(u, v); # produto interno ou escalar de u com v, u.v
```

$$14 \tag{18}$$

Observação: o comando "innerprod(u,v)" é mais utilizado em Álgebra Linear, realizando o produto interno usual de vetores e também de matrizes.

```
> w := crossprod(u, v); # produto vetorial de u com v, u x v
```

$$w := \begin{bmatrix} 31 & 2 & -17 \end{bmatrix} \tag{19}$$

Veja que o produto vetorial de u com v, w = u x v , é ortogonal aos vetores u e v. A ortogonalidade de dois vetores é verificada quando o produto escalar é nulo

```
> dotprod(w, u); dotprod(w, v);
```

$$0$$

$$0 \tag{20}$$

O produto vetorial não é simétrico :

```
> crossprod(v, u); crossprod(u, v);
```

$$\begin{bmatrix} -31 & -2 & 17 \end{bmatrix}$$

$$\begin{bmatrix} 31 & 2 & -17 \end{bmatrix} \tag{21}$$

A norma (comprimento) de um vetor relativamente a uma base ortonormal é dado pela raiz quadrada de cada elememto elevado ao quadrado

e no Maple V, pode ser expresso na forma sqrt(dotprod(u,u)) ou diretamente, usando norm(u,2) .

```
> sqrt(dotprod(u, u)); norm(u, 2);
```

$$\sqrt{29}$$

$$\sqrt{29} \tag{22}$$

Como u . v = ||u|| ||v|| cos θ , onde θ é o ângulo entre u e v, temos que

θ = arccos(u.v / (||u|| ||v||)),
exceto quando um dos vetores é nulo. O Maple V possui a função angle .

```
> theta := angle(u, v);
  beta := arccos(dotprod(u, v) / (norm(u, 2) * norm(v, 2)));
```

$$\theta := \arccos\left(\frac{7\sqrt{29}\sqrt{50}}{725}\right)$$

$$\beta := \arccos\left(\frac{7\sqrt{29}\sqrt{2}}{145}\right) \tag{23}$$

```
> evalf(theta); evalf(beta); # em radianos
```

$$1.194306936$$

$$1.194306936 \tag{24}$$

Produto misto: (u x v). w = [u, v, w] . Vamos denotar, para nosso uso, misto(u, v, w), que pode ser calculado através do determinante da matriz cujas linhas são u, v, e w. Vamos definir uma função para calcular o produto misto:

```
> misto := proc(u :: vector, v :: vector, w :: vector) local M;
  M := stackmatrix(u, v, w);
  det(M);
   end;
```

```
misto := proc(u::vector, v::vector, w::vector)                    (25)
    local M;
    M := linalg:-stackmatrix(u, v, w); linalg:-det(M)
end proc
```

```
> u := vector([1, 1, 1]); v := vector([0, 2, 1]); w := vector([0, 0, 3]);
```

$$u := \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$v := \begin{bmatrix} 0 & 2 & 1 \end{bmatrix}$$

$$w := \begin{bmatrix} 0 & 0 & 3 \end{bmatrix} \tag{26}$$

```
> misto(u, v, w);
  dotprod(crossprod(u, v), w); # compare!
```

$$6$$

$$6 \tag{27}$$

A troca de ordem dos vetores no produto misto pode alterar ou não o sinal, conforme se altera ou não a orientação (positiva ou negativa) do conjunto. Quando {u, v, w} forma uma base positiva do espaço, [u, v, w] > 0 e representa o volume do paralelepípedo de lados u, v e w.

```
> misto(u, v, w), misto(v, w, u), misto(w, u, v); # bases positivamente orientadas
```

$$6, 6, 6 \tag{28}$$

```
> misto(v, u, w), misto(w, v, u), misto(u, w, v); # bases negativamente orientadas
```

$$-6, -6, -6 \tag{29}$$