Procedura 7: il prodotto vettoriale = prodotto matriciale

Scrivere una procedura che calcoli il prodotto vettoriale $v \times w$ come prodotto matriciale

$$v \times w = S(v).w$$

In particolare, vi è una corrispondenza biunivoca tra il vettore v e la matrice antisimmetrica S(v):

$$v = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \Leftrightarrow S(v) = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}$$

Ipotizzando che:

$$v = a\,e_x + b\,e_y + c\,e_z$$

$$w = \alpha\,e_x + \beta\,e_y + \gamma\,e_z$$

Il prodotto vettoriale $v \times w$ è pari a:

$$v \times w = v \times (\alpha\,e_x + \beta\,e_y + \gamma\,e_z) = \alpha\,v \times e_x + \beta\,v \times e_y + \gamma\,v \times e_z$$

Che in forma matriciale:

$$v \times w = \begin{pmatrix} \vdots & \vdots & \vdots \\ v \times e_x & v \times e_y & v \times e_z \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = S(v).w$$

In cui:

$$v \times e_x = c\,e_y - b\,e_z = \begin{pmatrix} 0 \\ c \\ -b \end{pmatrix}; v \times e_y = -c\,e_x + a\,e_z = \begin{pmatrix} -c \\ 0 \\ a \end{pmatrix}; v \times e_z = b\,e_x - a\,e_y = \begin{pmatrix} b \\ -a \\ 0 \end{pmatrix};$$

Quindi:

$$S(v) = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}$$

La funzione vect(v,w) prende in input due vettori $s, t \in \mathbb{R}^3$ e restituisce in output il prodotto vettoriale $v \times w$ effettuato tramite la matrice S antisimmetrica, popolata da due cicli for.
In particolare, gli indici $s_{1,3-\mathrm{remainder}(i+j,3)}$ permetteno di scegliere ed inserire correttamente all'interno della matrice S i valori a, b, c al fine di ottenere:
$S(v) = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}$
Infine, viene effettuato il prodotto punto S.t e restituito in output tramite la variabile res.

```
(%i12) vect(v,w):=block([res],

                              S:ident(3),
                              for i:1 thru 3 do
                                (
                                for j:1 thru 3 do
                                   (
                                     if i=j
                                        then S[i][j]:0
                                     elseif j>i
                                        then (
                                       temp:(-1)^(j-i)*v[1][3-remainder(i+j,3)],
                                              S[i][j]:temp,
                                              S[j][i]:-temp

                                               )
                                     )
                                 ),


                               res:S.w
                            )
```

(%o12) $\mathrm{vect}(v, w) := \mathbf{block}\ ([\mathrm{res}], S\colon \mathrm{ident}(3), \mathbf{for}\ i\ \mathbf{thru}\ 3\ \mathbf{do}\ \mathbf{for}\ j\ \mathbf{thru}\ 3\ \mathbf{do}\ \mathbf{if}\ i = j\ \mathbf{then}\ (S_i)_j\colon 0\ \mathbf{elseif}\ j > i\ \mathbf{then}\ (\mathrm{temp}\colon (-1)^{j-i}\,(v_1)_{3-\mathrm{remainder}(i+j,3)}, (S_i)_j\colon \mathrm{temp}, (S_j)_i\colon -\mathrm{temp}), \mathrm{res}\colon S \cdot w)$

```
(%i13) v:matrix([a,b,c]);
```

(%o13) $\begin{pmatrix} a & b & c \end{pmatrix}$

```
(%i14) w:matrix([alpha,beta,gamma]);
```

(%o14) $\begin{pmatrix} \alpha & \beta & \gamma \end{pmatrix}$

```
(%i15) vectMatrix:vect(v,w);
```

(%o15) $\begin{pmatrix} b\,\gamma - \beta\,c \\ \alpha\,c - a\,\gamma \\ a\,\beta - \alpha\,b \end{pmatrix}$

Prodotto vettoriale di Maxima

```
(%i16) load(vect)
```

vect: warning: removing existing rule or rules for ".".

```
(%i16) e:[a,b,c];
```

(%o16) C:/Maxima/bin/../share/maxima/5.44.0/share/vector/vect.mac

```
(%i18) d:[alpha,beta,gamma];
```

(%o18) $[\alpha, \beta, \gamma]$

(%i19) e~d;

(%o19) $([a,b,c],[\alpha,\beta,\gamma])$

(%i20) vectMaxima:express(%)

(%o20) $[b\,\gamma-\beta\,c,\alpha\,c-a\,\gamma,a\,\beta-\alpha\,b]$

La fuznione checkVectors(x,y) verifica se due vettori sono uguali: si scorre, tramite un ciclo for, tutto il vettore x e si confrontano tutti gli elementi di x con il corrispettivo elemento di y. Nel caso in cui vemga trovato un elemento $x_i \neq y_i$ con $i \in \{1,2,3\}$ la funzione termina e restituisce in output un messaggio di errore.
Altrimenti, il ciclo for viene terminato ed i vettori risultano uguali.

```
(%i21) checkVectors(x,y):=block([res],
                                size:length(x),
                                tempVect: transpose(y),
                                for i:1 thru size do(
                                   if(x[i]!=y[i]) then ( res:"Vectors are not
       equals",
                                                         return)
                                 ),
                                res:"Vectors are equals"
                                )
```

(%o21) checkVectors(x,y) := **block** ([res], size: length(x), tempVect: transpose(y),
for i **thru** size **do if** $x_i! = y_i$ **then** (res: Vectors are not equals , return), res: Vectors are equals)

(%i22) checkVectors(vectMatrix,vectMaxima);

(%o22) Vectors are equals