ALGEBRA LINEARE

```
> with(linalg):
```

Procedura di standardizzazione rispetto la colonna p-ma.

La seguente procedura standardizza la matrice A rispetto la p-ma proiezione (=colonna p-ma)

```
> stand_p:=proc(A,p)
  local L,C,j,B,i,AA,BB,CC,h,hh,kk,kkk,CCC,LL,m,M;
  L:=convert(col(A,p),list);
  LL:=[seq(0^m,m=1..rowdim(A))];
  if L<>LL then
  C:=NULL;
  for j to nops(L) do
     if L[j]<>0 then
      C:=C,j;
     fi;
  od;
  B:=[C];
  i:=B[1];
  AA:=mulrow(A,i,1/L[i]);
  BB:=swaprow(AA,1,i);
  CC:=NULL;
  for h from 2 to rowdim(BB) do
  hh:=convert(row(BB,h),list);
  kk:=matadd(row(BB,h),row(BB,1),1,-hh[p]);
  kkk:=convert(kk,list);
  CC:=CC,kkk;
  od;
  CCC:=[convert(row(BB,1),list),CC];
  M:=matrix(CCC);
  else
  M:=evalm(A);
  fi;
  end;
```

stand_p := **proc**(*A*, *p*)

local *L*, *C*, *j*, *B*, *i*, *AA*, *BB*, *CC*, *h*, *hh*, *kk*, *kkk*, *CCC*, *LL*, *m*, *M*;

L := convert(col(A, p), *list*);

LL := [seq(0, $m = 1$ .. rowdim(A))];

if $L \neq LL$ **then**

C := $NULL$;

for j **to** nops(L) **do if** $L[j] \neq 0$ **then** $C := C, j$ **fi od**;

$B := [C]$;

$i := B[1]$;

$AA :=$ mulrow($A, i, 1 / L[i]$);

$BB :=$ swaprow($AA, 1, i$);

$CC := NULL$;

for h **from** 2 **to** rowdim(BB) **do**

$hh :=$ convert(row(BB, h), $list$);

$kk :=$ matadd(row(BB, h), row($BB, 1$), 1, $-hh[p]$);

$kkk :=$ convert($kk, list$);

$CC := CC, kkk$

od;

$CCC :=$ [convert(row(BB, 1), $list$), CC];

$M :=$ matrix(CCC)

else $M :=$ evalm(A)

fi

end

Esempio:

```
> A:=matrix([[4,3,2,4,1],[2,2,7,4,3],[1,0,1,7,2],[3,8,6,4,1],[0,0,5,
  4,9]]);
```

$$A := \begin{bmatrix} 4 & 3 & 2 & 4 & 1 \\ 2 & 2 & 7 & 4 & 3 \\ 1 & 0 & 1 & 7 & 2 \\ 3 & 8 & 6 & 4 & 1 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

```
> stand_p(A,3);
```

$$\begin{bmatrix} 2 & \frac{3}{2} & 1 & 2 & \frac{1}{2} \\ -12 & \frac{-17}{2} & 0 & -10 & \frac{-1}{2} \\ -1 & \frac{-3}{2} & 0 & 5 & \frac{3}{2} \\ -9 & -1 & 0 & -8 & -2 \\ -10 & \frac{-15}{2} & 0 & -6 & \frac{13}{2} \end{bmatrix}$$

Procedura di gradinizzazione per righe di una matrice.

```
> gradiniz:=proc(A)
  local R,j,B,L,LL,m,r,RR,mm,S,SS,n,C;
  R:=NULL;
  r:=rowdim(A);
  C:=A;
  for j to coldim(C) do
    B:=stand_p(C,j);
    L:=convert(col(B,j),list);
    LL:=[seq(0^m,m=1..rowdim(B))];
       if L<>LL then
         C:=submatrix(B, 2..rowdim(B), 1..coldim(B));
         R:=R,convert(row(B,1),list);
       fi;
   od;
   if nops([R])<r then
   RR:=[seq(0^mm,mm=1..coldim(A))];
   S:=seq(n*RR,n=1..(r-nops([R])));
   SS:=R,S;
     else
   SS:=R;
   fi;
   matrix([SS]);
  end;
```

gradiniz := **proc**(*A*)

local *R*, *j*, *B*, *L*, *LL*, *m*, *r*, *RR*, *mm*, *S*, *SS*, *n*, *C*;

R := *NULL*;

r := rowdim(*A*);

C := *A*;

for *j* **to** coldim(*C*) **do**

B := stand_p(*C*, *j*);

L := convert(col(*B*, *j*), *list*);

LL := [seq(0, $m = 1$ .. rowdim(*B*))];

if $L \neq LL$ **then**

C := submatrix(*B*, 2 .. rowdim(*B*), 1 .. coldim(*B*));

R := *R*, convert(row(*B*, 1), *list*)

fi

od;

if nops([R]) < r **then**

RR := [seq(0, mm = 1 .. coldim(A))];

S := seq(n*RR, n = 1 .. r − nops([R]));

SS := R, S

else SS := R

fi;

matrix([SS])

end

```
> gradiniz(A);
```

$$\begin{bmatrix} 1 & \frac{3}{4} & \frac{1}{2} & 1 & \frac{1}{4} \\ 0 & 1 & 12 & 4 & 5 \\ 0 & 0 & 1 & \frac{18}{19} & \frac{11}{19} \\ 0 & 0 & 0 & 1 & \frac{168}{743} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Altro esempio:

```
> B:=matrix([[1,3,2,4,1],[0,2,7,4,3],[1,5,9,8,4],[2,8,11,12,5],[0,0,
  5,4,9]]);
```

$$B := \begin{bmatrix} 1 & 3 & 2 & 4 & 1 \\ 0 & 2 & 7 & 4 & 3 \\ 1 & 5 & 9 & 8 & 4 \\ 2 & 8 & 11 & 12 & 5 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

```
> gradiniz(A);
```

$$\begin{bmatrix} 1 & 3 & 2 & 4 & 1 \\ 0 & 1 & \frac{7}{2} & 2 & \frac{3}{2} \\ 0 & 0 & 1 & \frac{4}{5} & \frac{9}{5} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Gradinizzazione interattiva.

Possiamo usare tre comandi che eseguono le tre operazioni elementari.
-Permutazione di due righe:
swaprow(A,i,j): permuta la riga i con la j.
-Prodotto di una riga per uno scalare non nullo:
mulrow(A,i,k): moltiplica la riga i per lo scalare k.
-Somma di una riga per un’altra moltiplicata per uno scalare.
addrow(A, i, j, m): sostituisce in A la riga j con m*i + j.
Esempio:
eseguiamo il processo interagendo con la macchina.

```
> evalm(A);
```

$$\begin{bmatrix} 4 & 3 & 2 & 4 & 1 \\ 2 & 2 & 7 & 4 & 3 \\ 1 & 0 & 1 & 7 & 2 \\ 3 & 8 & 6 & 4 & 1 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

Portiamo al primo posto la terza riga.

```
> A1:=swaprow(A,1,3);
```

$$A1 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 2 & 2 & 7 & 4 & 3 \\ 4 & 3 & 2 & 4 & 1 \\ 3 & 8 & 6 & 4 & 1 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

Sostituiamo alla seconda riga di A1 la seconda meno la prima riga moltiplicata per 2.

```
> A2:=addrow(A1,1,2,-2);
```

$$A2 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 2 & 5 & -10 & -1 \\ 4 & 3 & 2 & 4 & 1 \\ 3 & 8 & 6 & 4 & 1 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

Proseguiamo analogamente per le altre.

```
> A3:=addrow(A2,1,3,-4);
```

$$A3 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 2 & 5 & -10 & -1 \\ 0 & 3 & -2 & -24 & -7 \\ 3 & 8 & 6 & 4 & 1 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

```
> A4:=addrow(A3,1,4,-3);
```

$$A4 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 2 & 5 & -10 & -1 \\ 0 & 3 & -2 & -24 & -7 \\ 0 & 8 & 3 & -17 & -5 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

```
> A5:=mulrow(A4,2,1/2);
```

$$A5 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 1 & \frac{5}{2} & -5 & \frac{-1}{2} \\ 0 & 3 & -2 & -24 & -7 \\ 0 & 8 & 3 & -17 & -5 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

Iteriamo il procedimento di standardizzazione rispetto le altre proiezioni.

```
> A6:=addrow(A5,2,3,-3);
```

$$A6 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 1 & \frac{5}{2} & -5 & \frac{-1}{2} \\ 0 & 0 & \frac{-19}{2} & -9 & \frac{-11}{2} \\ 0 & 8 & 3 & -17 & -5 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

```
> A7:=addrow(A6,2,4,-8);
```

$$A7 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 1 & \frac{5}{2} & -5 & \frac{-1}{2} \\ 0 & 0 & \frac{-19}{2} & -9 & \frac{-11}{2} \\ 0 & 0 & -17 & 23 & -1 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

```
> A8:=mulrow(A7,3,-2/19);
```

$$A8 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 1 & \frac{5}{2} & -5 & \frac{-1}{2} \\ 0 & 0 & 1 & \frac{18}{19} & \frac{11}{19} \\ 0 & 0 & -17 & 23 & -1 \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

```
> A9:=addrow(A8,3,4,17);
```

$$A9 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 1 & \frac{5}{2} & -5 & \frac{-1}{2} \\ 0 & 0 & 1 & \frac{18}{19} & \frac{11}{19} \\ 0 & 0 & 0 & \frac{743}{19} & \frac{168}{19} \\ 0 & 0 & 5 & 4 & 9 \end{bmatrix}$$

```
> A10:=addrow(A9,3,5,-5);
```

$$A10 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 1 & \frac{5}{2} & -5 & \frac{-1}{2} \\ 0 & 0 & 1 & \frac{18}{19} & \frac{11}{19} \\ 0 & 0 & 0 & \frac{743}{19} & \frac{168}{19} \\ 0 & 0 & 0 & \frac{-14}{19} & \frac{116}{19} \end{bmatrix}$$

```
> A11:=mulrow(A10,4,19/743);
```

$$A11 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 1 & \frac{5}{2} & -5 & \frac{-1}{2} \\ 0 & 0 & 1 & \frac{18}{19} & \frac{11}{19} \\ 0 & 0 & 0 & 1 & \frac{168}{743} \\ 0 & 0 & 0 & \frac{-14}{19} & \frac{116}{19} \end{bmatrix}$$

```
> A12:=addrow(A11,4,5,14/19);
```

$$A12 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 1 & \frac{5}{2} & -5 & \frac{-1}{2} \\ 0 & 0 & 1 & \frac{18}{19} & \frac{11}{19} \\ 0 & 0 & 0 & 1 & \frac{168}{743} \\ 0 & 0 & 0 & 0 & \frac{4660}{743} \end{bmatrix}$$

```
> A13:=mulrow(A12,5,743/4660);
```

$$A13 := \begin{bmatrix} 1 & 0 & 1 & 7 & 2 \\ 0 & 1 & \frac{5}{2} & -5 & \frac{-1}{2} \\ 0 & 0 & 1 & \frac{18}{19} & \frac{11}{19} \\ 0 & 0 & 0 & 1 & \frac{168}{743} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Estrazione di una base da un sistema di generatori di un sottospazio.

La procedura calcola una base del sottospazio di R^n generato dalla lista L, cancellando i vettori linearmente dipendenti.

```
> estrae_base:=proc(L)
  local A,B;
  A:=matrix(L);
  B:=basis(A,'rowspace');
  end;
```

estrae_base := **proc**(*L*) **local** *A*, *B*; *A* := matrix(*L*); *B* := basis(*A*, '*rowspace*') **end**

Esempio: cerchiamo una base estratta dalla seguente lista di generatori di W:= < v_1, v_2, v_3 >.

```
> L:=[[1,2,3],[4,4,3],[5,6,6]];
```

$$L := [[1, 2, 3], [4, 4, 3], [5, 6, 6]]$$

```
> estrae_base(L);
```

$$[[1, 2, 3], [4, 4, 3]]$$

Calcolare una base di un sottospazio.

La procedura permette di calcolare una base del sottospazio di R^n generato dalla lista L (la base non e', necessariamente estratta da L).

```
> calcola_base:=proc(L)
  local A,B,N,R,Rj,j,i;
  A:=matrix(L);
  B:=gausselim(A);
  N:=[seq(0^i,i=1..coldim(B))];
  R:=NULL;
  for j to rowdim(B) do
    Rj:=convert(row(B,j),list);
      if Rj<>N then
        R:=R,Rj;
      fi;
  od;
  [R];
  end;
```

calcola_base := **proc**(L)
local A, B, N, R, Rj, j, i;
A := matrix(L);
B := gausselim(A);
N := [seq(0, i = 1 .. coldim(B))];
R := $NULL$;
for j **to** rowdim(B) **do** Rj := convert(row(B, j), *list*); **if** $Rj \neq N$ **then** R := R, Rj **fi od**;
[R]
end

Esempio.

```
> L:=[[4,2,2,4],[2,1,5,3],[6,3,7,7],[0,0,0,0]];
```

$$L := [[4, 2, 2, 4], [2, 1, 5, 3], [6, 3, 7, 7], [0, 0, 0, 0]]$$

```
> calcola_base(L);
```

$$[[4, 2, 2, 4], [0, 0, 4, 1]]$$

Completamento in una base.

La procedura permette di controllare se i vettori di una lista L (lista di liste) sono una base di R^n; in caso contrario la procedura controlla se sono linearmente indipendeti e, in tale caso, completa la lista L in una base di R^n scegliendo opportuni vettori di una base B (lista di liste) fissata. Un segnale di errore appare nel caso che B non sia una base.

```
> completa_base:=proc(L,B)
  local A,C,i,Di,Ci,j,H;
  H:=matrix(L);
  if rank(matrix(B))=nops(B[1])then
  if rank(H)<nops(B) then
  if rank(H)=nops(L) then
  A:=matrix(B);
  C:=seq(L[j],j=1..nops(L));
  for i to nops(B) do
     Ci:=matrix([C]);
     Di:=stackmatrix(Ci,row(A,i));
        if rank(Ci)<rank(Di) then
        C:=C,B[i];
        fi;
  od;
  [C];
  else
  print("errore! i vettori",L,"sono linearmente dipendenti.");
  fi;
  else
  print("i vettori",L,"sono gia’ una base.");
  fi;
  else
  print("errore!",B,"non e’ une base.");
  fi;
  end;
```

completa_base := **proc**(L, B)

local A, C, i, Di, Ci, j, H;

H := matrix(L);

if rank(matrix(B)) = nops(B[1]) **then**

if rank(H) < nops(B) **then**

if rank(H) = nops(L) **then**

A := matrix(B);

C := seq(L[j], j = 1 .. nops(L));

for i **to** nops(B) **do**

Ci := matrix([C]);

Di := stackmatrix(Ci, row(A, i));

```
                if rank(Ci) < rank(Di) then C := C, B[i] fi
            od;
            [C]
        else print("errore! i vettori", L, "sono linearmente dipendenti.")
        fi
    else print("i vettori", L, "sono gia’ una base.")
    fi
  else print("errore!", B, "non e’ une base.")
  fi
end
```

Esempi.

```
> L:=[[1,0,0],[0,2,3],[0,0,3]];
```

$$L := [[1, 0, 0], [0, 2, 3], [0, 0, 3]]$$

```
> B:=[[1,1,1],[0,1,1],[0,0,1]];
```

$$B := [[1, 1, 1], [0, 1, 1], [0, 0, 1]]$$

```
> completa_base(L,B);
```

"i vettori", [[1, 0, 0], [0, 2, 3], [0, 0, 3]], "sono gia’ una base."

```
> LL:=[[1,1,1],[0,3,3],[1,4,4]];
```

$$LL := [[1, 1, 1], [0, 3, 3], [1, 4, 4]]$$

```
> completa_base(LL,B);
```

"errore! i vettori", [[1, 1, 1], [0, 3, 3], [1, 4, 4]], "sono linearmente dipendenti."

```
> BB:=[[1,1,1],[0,1,1],[1,2,2]];
```

$$BB := [[1, 1, 1], [0, 1, 1], [1, 2, 2]]$$

```
> completa_base(LL,BB);
```

"errore!", [[1, 1, 1], [0, 1, 1], [1, 2, 2]], "non e’ une base."

```
> LLL:=[[1,0,0],[0,2,3]];
```

$$LLL := [[1, 0, 0], [0, 2, 3]]$$

```
> completa_base(LLL,B);
```

$$[[1, 0, 0], [0, 2, 3], [1, 1, 1]]$$

Matrice di un’applicazione lineare definita su una base.

Siano B1 eB2 basi di R^n ed R^m rispettivamente scritte come liste di liste ; sia F una lista di n vettori di R^m che definiscono (teorema di determinazione !) l’applicazione lineare f.
La procedura calcola la matrice, rispetto B1 e B2 rispettivamente, dell’applicazione lineare di dominio R^n e codominio R^m definita da F. Fornisce la dimensione del sottospazio immagine e quella del ker(f). Vari segnali appaiono nel caso di scelta errata di qualche elemento.

```
> matrice_f_b1_b2:=proc(B1,F,B2)
  local M,i,B22,A;
  if nops(F)=nops(B1) then
  if rank(matrix(B1))=nops(B1[1])then
     if rank(matrix(B2))=nops(B2[1])then
     B22:=transpose(matrix(B2));
     M:=NULL;
     for i to nops(F) do
      M:=M,convert(linsolve(B22,vector(F[i])),list);
     od;
     A:=transpose(matrix([M]));
     print("la dimensione dell’immagine e’:",rank(A));
     print("la dimensione del nucleo e’:",nops(B1)-rank(A));
     A:=transpose(matrix([M]));else
     print("errore!",B2,"non e’ une base.");
    fi;
  else
  print("errore!",B1,"non e’ une base.");
  fi;
  else
  print("errore! il numero dei vettori di",F,"deve essere quello di
  B1.");
  fi;
  end;
```

matrice_f_b1_b2 := **proc**(*B1*, *F*, *B2*)

local *M*, *i*, *B22*, *A*;

if nops(F) = nops($B1$) **then**

if rank(matrix($B1$)) = nops($B1[1]$) **then**

if rank(matrix($B2$)) = nops($B2[1]$) **then**

$B22$:= transpose(matrix($B2$));

M := $NULL$;

for i **to** nops(F) **do** M := M, convert(linsolve($B22$, vector($F[i]$)), *list*) **od**;

A := transpose(matrix([M]));

print("la dimensione dell’immagine e’:", rank(A));

print("la dimensione del nucleo e’:", nops($B1$) – rank(A));

A := transpose(matrix([M]))

else print("errore!", *B2*, "non e’ une base.")

fi

```
        else print("errore!", B1, "non e' une base.")
        fi
    else print("errore! il numero dei vettori di", F, "deve essere quello di B1.")
    fi
end
```

Esempi:

```
> B1:=[[1,1,1,1],[0,1,1,1],[0,0,1,1],[0,0,0,1]];
```

$$B1 := [[1, 1, 1, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 1]]$$

```
> B2:=[[1,1],[0,1]];
```

$$B2 := [[1, 1], [0, 1]]$$

```
> F:=[[1,2],[2,1],[4,3],[7,8]];
```

$$F := [[1, 2], [2, 1], [4, 3], [7, 8]]$$

```
> H:=matrice_f_b1_b2(B1,F,B2);
```

"la dimensione dell'immagine e':", 2

"la dimensione del nucleo e':", 2

$$H := \begin{bmatrix} 1 & 2 & 4 & 7 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

```
> B1:=[[1,1,1,1],[0,1,1,1],[0,0,1,1],[0,0,0,0]];
```

$$B1 := [[1, 1, 1, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 0]]$$

```
> B2:=[[1,1],[0,1]];
```

$$B2 := [[1, 1], [0, 1]]$$

```
> F:=[[1,2],[2,1],[4,3],[7,8]];
```

$$F := [[1, 2], [2, 1], [4, 3], [7, 8]]$$

```
> matrice_f_b1_b2(B1,F,B2);
```

"errore!", [[1, 1, 1, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 0]], "non e' une base."

```
> B1:=[[1,1,1,1],[0,1,1,1],[0,0,1,1],[0,0,0,1]];
```

$$B1 := [[1, 1, 1, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 1]]$$

```
> F:=[[1,2],[2,1],[4,3]];
```

$$F := [[1, 2], [2, 1], [4, 3]]$$

```
> matrice_f_b1_b2(B1,F,B2);
```

"errore! il numero dei vettori di", [[1, 2], [2, 1], [4, 3]], "deve essere quello di B1."

Matrice di un'applicazione lineare.

Siano B1 eB2 basi di R^n ed R^m rispettivamente scritte come liste di liste. La procedura calcola la matrice, rispetto B1 e B2 rispettivamente, dell'applicazione lineare H:=f(x_1, ... ,x_n) di dominio R^n e codominio R^m. Fornisce, inoltre, la dimensione del sottospazio immagine e quella del ker(f). Vari segnali appaiono nel caso di scelta errata di qualche elemento.

```
> matrice_f2_b1_b2:=proc(B1,H,B2,Ind)
  local F,i,j,eqi,Fi;
  if nops(Ind)=nops(B1) then
  if nops(H)=nops(B2) then
  F:=NULL;
     for i to nops(B1) do
      eqi:=NULL;
      for j to nops(Ind) do
         eqi:=eqi,Ind[j]=B1[i][j];
      od;
      Fi:=eval(H,[eqi]);
      F:=F,Fi;
     od;
  matrice_f_b1_b2(B1,[F],B2);
  else
  print("errore! la cardinalita’ di",H,"deve essere:",nops(B2));
  fi;
  else
  print("errore! il numero delle indeterminate deve
  essere:",nops(B1));
  fi;
  end;
```

matrice_f2_b1_b2 := **proc**(*B1*, *H*, *B2*, *Ind*)

local *F*, *i*, *j*, *eqi*, *Fi*;

if nops(*Ind*) = nops(*B1*) **then**

if nops(*H*) = nops(*B2*) **then**

F := *NULL*;

for *i* **to** nops(*B1*) **do**

eqi := *NULL*;

for *j* **to** nops(*Ind*) **do** *eqi* := *eqi*, *Ind*[*j*] = *B1*[*i*][*j*] **od**;

Fi := eval(*H*, [*eqi*]);

F := *F*, *Fi*

od;

matrice_f_b1_b2(*B1*, [*F*], *B2*)

else print("errore! la cardinalita’ di", *H*, "deve essere:", nops(*B2*))

fi

else print("errore! il numero delle indeterminate deve essere:", nops(*B1*))

fi

end

Esempi:

```
> B1;
```

$$[[1, 1, 1, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 1]]$$

```
> B2:=[[1,0,0],[0,1,0],[0,0,1]];
```

$$B2 := [[1, 0, 0], [0, 1, 0], [0, 0, 1]]$$

```
> H:=[x-y,x+2*t,x+3*y-t];
```

$$H := [x - y, x + 2\,t, x + 3\,y - t]$$

```
> matrice_f2_b1_b2(B1,H,B2,[x,y,z,t]);
```

"la dimensione dell’immagine e’:", 3

"la dimensione del nucleo e’:", 1

$$\begin{bmatrix} 0 & -1 & 0 & 0 \\ 3 & 2 & 2 & 2 \\ 3 & 2 & -1 & -1 \end{bmatrix}$$

```
> H:=[x-y,x+2*t];
```

$$H := [x - y, x + 2\,t]$$

```
> matrice_f2_b1_b2(B1,H,B2,[x,y,z,t]);
```

"errore! la cardinalita’ di", $[x - y, x + 2\,t]$, "deve essere:", 3

```
> H:=[x-y,x+2*t,x+3*y-t];
```

$$H := [x - y, x + 2\,t, x + 3\,y - t]$$

```
> Ind:=[x,y,z];
```

$$Ind := [x, y, z]$$

```
> matrice_f2_b1_b2(B1,H,B2,Ind);
```

"errore! il numero delle indeterminate deve essere:", 4

Applicazione lineare date le basi e la matrice.

Siano date le due basi B1 e B2 di R^n ed R^m rispettivamente ed una matrice M di tipo mxn.
La procedura calcola la lista dei vettori di R^m immagine dei vettori di B1 nell’applicazione lineare definita da M. Vengono determinate una base per il ker ed una per l’immagine. Vari segnali appaiono nel caso di scelta errata di qualche elemento.

```
> appl_lin_m_b1_b2:=proc(M,B1,B2)
  local A,B,C,H,i,L,K,KK,BB,CC,HH,j;
  if (rowdim(M)=nops(B2) and coldim(M)=nops(B1)) then
  if rank(matrix(B1))=nops(B1[1])then
     if rank(matrix(B2))=nops(B2[1])then
```

```
        A:=transpose(M);
        B:=matrix(B2);
        C:=multiply(A,B);
        H:=NULL;
           for i to rowdim(C) do
           H:=H,convert(row(C,i),list);
           od;
        L:=B1,[H];
        if kernel(M)<>{} then
        K:=convert(kernel(M),list);
        KK:=matrix(K);
        BB:=matrix(B1);
        CC:=multiply(KK,BB);
        HH:=NULL;
           for j to rowdim(CC) do
           HH:=HH,convert(row(CC,j),list);
           od;
        print("una base per il nucleo e’",[HH]);
        else
        print("il nucleo e’ nullo");
        fi;
        print("L’applicazione e’ definita da",L);
        print("una base per l’immagine e’",basis(C,’rowspace’));
      else
        print("errore!",B2,"non e’ une base.");
      fi;
   else
   print("errore!",B1,"non e’ une base.");
   fi;
   else
   print("errore! la matrice",M,"non e’ del tipo corretto.");
   fi;
   end;
```

appl_lin_m_b1_b2 := **proc**(*M*, *B1*, *B2*)

local *A*, *B*, *C*, *H*, *i*, *L*, *K*, *KK*, *BB*, *CC*, *HH*, *j*;

if rowdim(*M*) = nops(*B2*) **and** coldim(*M*) = nops(*B1*) **then**

if rank(matrix(*B1*)) = nops(*B1*[1]) **then**

if rank(matrix(*B2*)) = nops(*B2*[1]) **then**

A := transpose(*M*);

B := matrix(*B2*);

C := multiply(*A*, *B*);

H := *NULL*;

```
            for i to rowdim(C) do H := H, convert(row(C, i), list) od;
            L := B1, [H];
            if kernel(M) ≠ {  } then
                K := convert(kernel(M), list);
                KK := matrix(K);
                BB := matrix(B1);
                CC := multiply(KK, BB);
                HH := NULL;
                for j to rowdim(CC) do HH := HH, convert(row(CC, j), list) od;
                print("una base per il nucleo e’", [HH])
            else print("il nucleo e’ nullo")
            fi;
            print("L’applicazione e’ definita da", L);
            print("una base per l’immagine e’", basis(C, ’rowspace’))
        else print("errore!", B2, "non e’ une base.")
        fi
    else print("errore!", B1, "non e’ une base.")
    fi
  else print("errore! la matrice", M, "non e’ del tipo corretto.")
  fi
end
```

Esempi:

```
> B1;
```

[[1, 1, 1, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 1]]

```
> B2;
```

[[1, 0, 0], [0, 1, 0], [0, 0, 1]]

```
> M:=matrix([[0, -1, 0, 0], [3, 2, 2, 2], [3, 2, -1, -1]]);
```

$$M := \begin{bmatrix} 0 & -1 & 0 & 0 \\ 3 & 2 & 2 & 2 \\ 3 & 2 & -1 & -1 \end{bmatrix}$$

```
> appl_lin_m_b1_b2(M,B1,B2);
```

"una base per il nucleo e’", [[0, 0, -1, 0]]

"L’applicazione e’ definita da", [[1, 1, 1, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 1]], [[0, 3, 3], [-1, 2, 2], [0, 2, -1], [0, 2, -1]]

"una base per l'immagine e'", [[0, 3, 3], [-1, 2, 2], [0, 2, -1]]

```
> M:=matrix([[0, -1, 0, 0], [3, 2, 2, 2]]);
```

$$M := \begin{bmatrix} 0 & -1 & 0 & 0 \\ 3 & 2 & 2 & 2 \end{bmatrix}$$

```
> appl_lin_m_b1_b2(M,B1,B2);
```

"errore! la matrice", $\begin{bmatrix} 0 & -1 & 0 & 0 \\ 3 & 2 & 2 & 2 \end{bmatrix}$, "non e' del tipo corretto."

Immagine di un vettore in un'applicazione lineare definita da una matrice.

Siano date le due basi B1 e B2 di R^n ed R^m rispettivamente, una matrice M di tipo mxn ed un vettore v (anche generico) di R^n.
La procedura calcola l'immagine di v nell'applicazione lineare definita da M. Vari segnali appaiono nel caso di scelta errata di qualche elemento.

```
> immagine_f:=proc(v,M,B1,B2)
  local B11,V,VV,C;
  if nops(v)=nops(B1) then
  if (rowdim(M)=nops(B2) and coldim(M)=nops(B1)) then
  if rank(matrix(B1))=nops(B1[1])then
     if rank(matrix(B2))=nops(B2[1])then
        B11:=transpose(matrix(B1));
        V:=convert(linsolve(B11,vector(v)),matrix);
        VV:=multiply(M,V);
        C:=multiply(transpose(VV),matrix(B2));
        convert(row(C,1),list);
     else
       print("errore!",B2,"non e' une base.");
     fi;
  else
  print("errore!",B1,"non e' una base.");
  fi;
  else
  print("errore!la matrice",M,"non e' del tipo corretto.");
  fi;
  else
  print("errore!",v,"non e' nel dominio.");
  fi;
  end;
```

```
immagine_f := proc(v, M, B1, B2)
local B11, V, VV, C;
    if nops(v) = nops(B1) then
        if rowdim(M) = nops(B2) and coldim(M) = nops(B1) then
            if rank(matrix(B1)) = nops(B1[1]) then
                if rank(matrix(B2)) = nops(B2[1]) then
                    B11 := transpose(matrix(B1));
                    V := convert(linsolve(B11, vector(v)), matrix);
                    VV := multiply(M, V);
                    C := multiply(transpose(VV), matrix(B2));
                    convert(row(C, 1), list)
                else print("errore!", B2, "non e' une base.")
                fi
            else print("errore!", B1, "non e' una base.")
            fi
        else print("errore!la matrice", M, "non e' del tipo corretto.")
        fi
    else print("errore!", v, "non e' nel dominio.")
    fi
end
```

Esempi:

```
> B1;
```

$$[[1, 1, 1, 1], [0, 1, 1, 1], [0, 0, 1, 1], [0, 0, 0, 1]]$$

```
> B2;
```

$$[[1, 0, 0], [0, 1, 0], [0, 0, 1]]$$

```
> evalm(M);
```

$$\begin{bmatrix} 0 & -1 & 0 & 0 \\ 3 & 2 & 2 & 2 \\ 3 & 2 & -1 & -1 \end{bmatrix}$$

```
> immagine_f([2,1,1,1],M,B1,B2);
```

$$[1, 4, 4]$$

```
> immagine_f([x,y,z,t],M,B1,B2);
```

$$[x - y, x + 2\,t, x + 3\,y - t]$$

```
> immagine_f([2,1,1],M,B1,B2);
```

"errore!", [2, 1, 1], "non e' nel dominio."