Metodo di Eliminazione di Gauss e Decomposizione LU

Enrico Bertolazzi

Carica la libreriaLinear Algebraper la gestione di vettori e Matrici

```
> restart :
  with(LinearAlgebra):
```

Procedura upper_solve e lower_solve

```
> upper_solve := proc (A,b)
    local j, k, n, sol ;
    n := RowDimension(A) ;
    sol := Vector(n) ;
    for k from n by -1 to 1 do
      sol[k] := b[k] ;
      for j from k+1 to n do
        sol[k] := sol[k] - A[k,j]*sol[j] ;
      end do ;
      sol[k] := sol[k] / A[k,k] ;
    end do ;
    return sol ;
  end proc :
```

```
> lower_solve := proc (A,b)
    local j, k, n, sol ;
    n := RowDimension(A) ;
    sol := Vector(n) ;
    for k from 1 to n do
      sol[k] := b[k] ;
      for j from 1 to k-1 do
        sol[k] := sol[k] - A[k,j]*sol[j] ;
      end do ;
      sol[k] := sol[k] / A[k,k] ;
    end do ;
    return sol ;
  end proc :
```

Procedura gauss_lu

Costruisce la decomposizione LU della matrice A

Notate che la matrice A viene MODIFICATA!

```
> # Applica il metodo di Gauss in loco sulla matrice A
  # e ritorna il vettore delle parmutazioni P
  gauss_lu := proc(A)
    local P, L, U, i, j, k, imax, n, bf ;
    n := RowDimension(A) ;
    P := Vector([seq(k,k=1..n)]) ;
    for i from 1 to n-1 do
      # pivoting parziale
      imax := i ;
      for k from i+1 to n do
        if abs(A[P[k],i]) > abs(A[P[imax],i]) then
          imax := k ;
        end if ;
      end do ;

      # scambio delle righe
      bf      := P[i] ;
      P[i]    := P[imax] ;
      P[imax] := bf ;

      # eliminazione
      for k from i+1 to n do
        bf        := A[P[k],i] / A[P[i],i] ;
        A[P[k],i] := bf ;
        for j from i+1 to n do
          A[P[k],j] := A[P[k],j] - bf * A[P[i],j] ;
        end do ;
      end do ;
    end do ;

    return P ;
  end proc :
```

Procedura lu_build. Notate che la matrice A viene MODIFICATA!

```
> # Costruisce la  decomposizione LU
  # e ritorna il vettore delle parmutazioni P
  # le matrici L ed U
  lu_build := proc(A)
    local P, L, U, i, j, n ;
    n := RowDimension(A) ;
    P := gauss_lu(A) ;
    L := Matrix(n,n,0);
    U := Matrix(n,n,0) ;
    for i from 2 to n do
      for j from 1 to i-1 do
        L[i,j] := A[P[i],j] ;
```

```
      end do;
    end do ;
    for i from 1 to n do
      L[i,i] := 1 ;
      for j from i to n do
        U[i,j] := A[P[i],j] ;
      end do;
    end do ;
    return P,L,U ;
  end proc :
```

Procedura lu_solve.

```
> # Dato il vettore delle parmutazioni P
  # e le matrici L ed U risolve il problema
  # L.U.x = P.b
  lu_solve := proc(P,L,U,b)
    local n, i, sol ;
    # permute b
    n   := RowDimension(L) ;
    sol := Vector(n) ;
    for i from 1 to n do
      sol[i] := b[P[i]] ;
    end do ;
    return upper_solve(U,lower_solve(L,sol)) ;
  end proc :
```

Esempio d'uso

```
> A := Transpose(<<0,-1,1>|<1,4,1>|<-1,-1,4>>) ;
  b := A . <1,2,3> ;
```

$$A := \begin{bmatrix} 0 & -1 & 1 \\ 1 & 4 & 1 \\ -1 & -1 & 4 \end{bmatrix}$$

$$b := \begin{bmatrix} 1 \\ 12 \\ 9 \end{bmatrix}$$

```
> p,l,u := lu_build(A) ;
```

$$p, l, u := \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & \frac{-1}{3} & 1 \end{bmatrix}, \begin{bmatrix} 1 & 4 & 1 \\ 0 & 3 & 5 \\ 0 & 0 & \frac{8}{3} \end{bmatrix}$$

```
> lu_solve(p,l,u,b) ;
```

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
> print(A) ;
```

$$\begin{bmatrix} 0 & \frac{-1}{3} & \frac{8}{3} \\ 1 & 4 & 1 \\ -1 & 3 & 5 \end{bmatrix}$$

```
>
```