Author(s) : H. Lipps
Submitter :
Language : Fortran or Assembler or COMPASS

Library: KERNLIB
Submitted: 18.12.1979
Revised: 27.05.1987

Elementary Vector Processing

These subprograms perform elementary vector operations.

Structure:

`SUBROUTINE` and `FUNCTION` subprograms
User Entry Names: `RVADD`, `RVCPY`, `RVDIV`, `RVMPA`, `RVMPY`, `RVMUL`, `RVMULA`, `RVMUNA`, `RVRAN`, `RVSCA`, `RVSCL`, `RVSCS`, `RVSET`, `RVSUB`, `RVSUM`, `RVXCH`, `DVADD`, `DVCPY`, `DVDIV`, `DVMPA`, `DVMPY`, `DVMUL`, `DVMULA`, `DVMUNA`, `DVRAN`, `DVSCA`, `DVSCL`, `DVSCS`, `DVSET`, `DVSUB`, `DVSUM`, `DVXCH`, `CVADD`, `CVCPY`, `CVDIV`, `CVMPA`, `CVMPY`, `CVMUL`, `CVMULA`, `CVMUNA`, `CVRAN`, `CVSCA`, `CVSCL`, `CVSCS`, `CVSET`, `CVSUB`, `CVSUM`, `CVXCH`, `CVMPYC`, `CVMPAC`
External References: `LOCF` (N100), `RANF` (G900), `DRANF` (G900) (some Fortran versions only).

Usage:

For `t` = `R` (type `REAL`), `t` = `D` (type `DOUBLE PRECISION`), `t` = `C` (type `COMPLEX`):

`CALL tVSET (N,S,Z1,Z2)`	$z_j = s$
`CALL tVRAN (N,A,B,Z1,Z2)`	$z_j =$ random (see **Note** 2)
`CALL tVCPY (N,X1,X2,Z1,Z2)`	$z_j = x_j$
`CALL tVXCH (N,X1,X2,Y1,Y2)`	interchanges x_j with y_j
`CALL tVADD (N,X1,X2,Y1,Y2,Z1,Z2)`	$z_j = x_j + y_j$
`CALL tVSUB (N,X1,X2,Y1,Y2,Z1,Z2)`	$z_j = x_j - y_j$
`CALL tVMUL (N,X1,X2,Y1,Y2,Z1,Z2)`	$z_j = x_j y_j$
`CALL tVMULA(N,X1,X2,Y1,Y2,Z1,Z2)`	$z_j = x_j y_j + z_j$
`CALL tVMUNA(N,X1,X2,Y1,Y2,Z1,Z2)`	$z_j = -x_j y_j + z_j$
`CALL tVDIV (N,X1,X2,Y1,Y2,Z1,Z2,IFAIL)`	$z_j = x_j / y_j$ (see **Note** 3)
`CALL tVSCL (N,S,X1,X2,Z1,Z2)`	$z_j = s x_j$
`CALL tVSCA (N,S,X1,X2,Y1,Y2,Z1,Z2)`	$z_j = s x_j + y_j$
`CALL tVSCS (N,S,X1,X2,Y1,Y2,Z1,Z2)`	$z_j = s x_j - y_j$
`F = tVSUM (N,X1,X2)`	$f = x_1 + \cdots + x_n$
`F = tVMPY (N,X1,X2,Y1,Y2)`	$f = x_1 y_1 + \cdots + x_n y_n$
`F = tVMPA (N,X1,X2,Y1,Y2,S)`	$f = x_1 y_1 + \cdots + x_n y_n + s$
`F = CVMPYC(N,X1,X2,Y1,Y2)`	$f = x_1 \bar{y}_1 + \cdots + x_n \bar{y}_n$
`F = CVMPAC(N,X1,X2,Y1,Y2,S)`	$f = x_1 \bar{y}_1 + \cdots + x_n \bar{y}_n + s$

where $\bar{y}_j$ is the complex conjugate of y_j.

N	(INTEGER) The mathematical dimension of the vectors ($j = 1, 2, \ldots, \texttt{N}$).
S,A,B	(Type according to t) The scalar values s, a, and b, respectively.
X1,X2	(Type according to t) Array elements. They must contain the elements x_1, x_2 of the vector (x_j).
Y1,Y2	(Type according to t) Array elements. They must contain the elements y_1, y_2 of the vector (y_j).
Z1,Z2	(Type according to t) Array elements. On exit, they will contain the elements z_1, z_2 of the result vector (z_j).
IFAIL	(INTEGER) On exit, IFAIL is set to zero if all elements y_j are non-zero. Otherwise IFAIL is set to the smallest index k for which $y_k = 0$.

For N $<$ 1 all subroutines return control without action; functions tVSUM, tVMPY and CVMPYC assume the value zero, and tVMPA and CVMPAC assume the value S.

Restrictions:

If vector (z_j) overlaps with vector (x_j) or (y_j), results will be correct provided each element z_j coincides with an element x_k or y_k, where $k < j$.

Accuracy:

On computers with IBM 370 architecture, RVMPY, RVMPA, CVMPY and CVMPA accumulate the inner product using double-precision arithmetic internally; the final result is then rounded to single precision.

Notes:

1. The vectors (x_j) etc. need not be packed: any equidistant spacing of their elements is permitted. The subprograms determine the location of the vector element x_j from the actual arguments X1 and X2.
2. tVRAN sets z_j to a random value of type t that is uniformly distributed in the interval (A,B). For CVRAN, the real and imaginary parts of z_j are distributed uniformly and independently in (REAL(A),REAL(B)) and in (AIMAG(A),AIMAG(B)).
3. If $y_k = 0$ and $y_1, \ldots, y_{k-1}$ are non-zero, tVDIV computes only $z_1, \ldots, z_{k-1}$ and sets IFAIL $= k$.
4. The use of an in-line DO loop will be more efficient than calling the equivalent vector processing subprogram when the vector length is sufficiently small, due to the overhead of the subprogram call.

•