**Author(s) :** H. Lipps **Library:** KERNLIB
**Submitter:** **Submitted:** 18.12.1979
**Language :** Fortran or Assembler or COMPASS **Revised:** 15.11.1995

## Elementary Matrix Processing

These subprograms perform elementary matrix operations.

**Structure:**

`SUBROUTINE` and `FUNCTION` subprograms
User Entry Names: `RMADD`, `RMBIL`, `RMCPY`, `RMDMP`, `RMMNA`, `RMMNS`, `RMMPA`, `RMMPS`, `RMMPY`, `RMRAN`, `RMSCL`, `RMSET`, `RMSUB`, `RMUTL`, `RUMNA`, `RUMNS`, `RUMPA`, `RUMPS`, `RUMPY`,
`DMADD`, `DMBIL`, `DMCPY`, `DMDMP`, `DMMNA`, `DMMNS`, `DMMPA`, `DMMPS`, `DMMPY`, `DMRAN`, `DMSCL`, `DMSET`, `DMSUB`, `DMUTL`, `DUMNA`, `DUMNS`, `DUMPA`, `DUMPS`, `DUMPY`,
`CMADD`, `CMBIL`, `CMCPY`, `CMDMP`, `CMMNA`, `CMMNS`, `CMMPA`, `CMMPS`, `CMMPY`, `CMRAN`, `CMSCL`, `CMSET`, `CMSUB`, `CMUTL`, `CUMNA`, `CUMNS`, `CUMPA`, `CUMPS`, `CUMPY`, `CMMPYC`, `CCMMPY`, `CUMPYC`, `CCUMPY`
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 tMSET (M,N,S,Z11,Z12,Z21)` | $z_{ij} = s$ |
| `CALL tMRAN (M,N,A,B,Z11,Z12,Z21)` | $z_{ij} =$ random (see **Note** 2) |
| `CALL tMCPY (M,N,X11,X12,X21,Z11,Z12,Z21)` | $z_{ij} = x_{ij}$ |
| `CALL tMUTL (N,X11,X12,X21)` | $x_{jk} = x_{kj}$ $(j > k)$ (see **Note** 3) |
| `CALL tMSCL (M,N,S,X11,X12,X21,Z11,Z12,Z21)` | $z_{ij} = sx_{ij}$ |
| `CALL tMDMP (M,N,D1,D2,X11,X12,X21,Z11,Z12,Z21)` | $z_{ij} = d_i x_{ij}$ |
| `CALL tMADD (M,N,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)` | $z_{ij} = x_{ij} + y_{ij}$ |
| `CALL tMSUB (M,N,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)` | $z_{ij} = x_{ij} - y_{ij}$ |
| `CALL tMMPY (M,N,X11,X12,X21,Y1,Y2,Z1,Z2)` | $z_i = x_{i1}y_1 + \cdots + x_{in}y_n$ |
| `CALL tMMPA (M,N,X11,X12,X21,Y1,Y2,Z1,Z2)` | $z_i = x_{i1}y_1 + \cdots + x_{in}y_n + z_i$ |
| `CALL tMMPS (M,N,X11,X12,X21,Y1,Y2,Z1,Z2)` | $z_i = x_{i1}y_1 + \cdots + x_{in}y_n - z_i$ |
| `CALL tMMNA (M,N,X11,X12,X21,Y1,Y2,Z1,Z2)` | $z_i = -x_{i1}y_1 - \cdots - x_{in}y_n + z_i$ |
| `CALL tMMNS (M,N,X11,X12,X21,Y1,Y2,Z1,Z2)` | $z_i = -x_{i1}y_1 - \cdots - x_{in}y_n - z_i$ |
| `CALL tUMPY (N,U11,U12,U22,Y1,Y2,Z1,Z2)` | $z_j = u_{jj}y_j + \cdots + u_{jn}y_n$ |
| `CALL tUMPA (N,U11,U12,U22,Y1,Y2,Z1,Z2)` | $z_j = u_{jj}y_j + \cdots + u_{jn}y_n + z_j$ |
| `CALL tUMPS (N,U11,U12,U22,Y1,Y2,Z1,Z2)` | $z_j = u_{jj}y_j + \cdots + u_{jn}y_n - z_j$ |
| `CALL tUMNA (N,U11,U12,U22,Y1,Y2,Z1,Z2)` | $z_j = -u_{jj}y_j - \cdots - u_{jn}y_n + z_j$ |
| `CALL tUMNS (N,U11,U12,U22,Y1,Y2,Z1,Z2)` | $z_j = -u_{jj}y_j - \cdots - u_{jn}y_n - z_j$ |
| `F = tMBIL (N,V1,V2,X11,X12,X21,Y1,Y2)` | $f = \sum_{k,j=1}^{n} v_k x_{kj} y_j$ |
| `CALL CMMPYC(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)` | $z_i = x_{i1}\bar{y}_i + \cdots + x_{in}\bar{y}_n$ |
| `CALL CCMMPY(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)` | $z_i = \bar{x}_{i1}y_i + \cdots + \bar{x}_{in}y_n$ |
| `CALL CUMPYC(N,U11,U12,U22,Y1,Y2,Z1,Z2)` | $z_j = u_{jj}\bar{y}_j + \cdots + u_{jn}\bar{y}_n$ |
| `CALL CCUMPY(N,U11,U12,U22,Y1,Y2,Z1,Z2)` | $z_j = \bar{u}_{jj}y_j + \cdots + \bar{u}_{jn}y_n$ |

where $\bar{x}_{ij}, \bar{u}_{jk}, \bar{y}_j$ are the complex conjugates of $x_{ij}, u_{jk}, y_j$, respectively.

| | |
|---|---|
| `M,N` | (`INTEGER`) The mathematical dimensions of the matrices and vectors ($i = 1, 2, \ldots$ ,`M`; $j, k = 1, 2, \ldots$ ,`N`). |
| `S,A,B` | (Type according to `t`) The scalar values $s$, $a$, and $b$, respectively. |
| `X11,X12,X21` | (Type according to `t`) Array elements. They must contain the elements $x_{11}, x_{12}, x_{21}$ of the matrix $(x_{ij})$. |
| `Y11,Y12,Y21` | (Type according to `t`) Array elements. They must contain the elements $y_{11}, y_{12}, y_{21}$ of the matrix $(y_{ij})$. |
| `Y1,Y2` | (Type according to `t`) Array elements. They must contain the elements $y_1, y_2$ of the vector $(y_j)$. |
| `D1,D2` | (Type according to `t`) Array elements. They must contain the elements $d_1, d_2$ of the vector $(d_i)$. |
| `V1,V2` | (Type according to `t`) Array elements. They must contain the elements $v_1, v_2$ of the vector $(v_k)$. |
| `U11,U12,U22` | (Type according to `t`) Array elements. They must contain the elements $u_{11}, u_{12}, u_{22}$ of the upper-triangular matrix $(u_{jk})$. |
| `Z11,Z12,Z21` | (Type according to `t`) Array elements. On exit, they will contain the elements $z_{11}, z_{12}, z_{21}$ of the result matrix $(z_{ij})$. |
| `Z1,Z2` | (Type according to `t`) Array elements. On exit, they will contain the elements $z_1, z_2$ of the result vector $(z_j)$. |

For `M` $< 1$ or `N` $< 1$ all subroutines return control without action and all functions assume the value zero.

**Accuracy:**

On computers with IBM 370 architecture, all routines that accumulate the inner product of type `REAL` or `COMPLEX` use double-precision arithmetic internally; the final result is then rounded to single precision.

**Notes:**

1. The vectors $(y_j)$ etc. need not be packed: any equidistant spacing of their elements is permitted. The subprograms determine the location of the vector element $y_j$ from the actual arguments `Y1` and `Y2`. Similarly, the matrices $(x_{ij})$ etc. need not be stored according to the Fortran convention; any equidistant spacing of their rows and columns is permitted. In particular, matrices may be stored row-wise. The subprograms determine the location of the matrix element $x_{ij}$ from the actual arguments `X11`, `X12`, and `X21`.
2. `tMRAN` sets $z_{ij}$ to a random value of type `t` that is uniformly distributed in the interval `(A,B)`. For `CMRAN`, the real and imaginary parts of $z_{ij}$ are distributed uniformly and independently in `(REAL(A),REAL(B))` and in `(AIMAG(A),AIMAG(B))`.
3. `tMUTL` copies the upper triangle of the square matrix $(x_{jk})$ of order `N` to the lower triangle of this matrix, thus creating a symmetric matrix.
4. The use of in-line `DO` loops will be more efficient than calling the equivalent matrix processing subprogram when the matrix dimensions are sufficiently small, due to the overhead of the subprogram call.

•