```
      SUBROUTINE DGEMV(TRANS,M,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY)
*     .. Scalar Arguments ..
      DOUBLE PRECISION ALPHA,BETA
      INTEGER INCX,INCY,LDA,M,N
      CHARACTER TRANS
*     ..
*     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),X(*),Y(*)
*     ..
*
*  Purpose
*  =======
*
*  DGEMV  performs one of the matrix-vector operations
*
*     y := alpha*A*x + beta*y,   or   y := alpha*A'*x + beta*y,
*
*  where alpha and beta are scalars, x and y are vectors and A is an
*  m by n matrix.
*
*  Arguments
*  ==========
*
*  TRANS  - CHARACTER*1.
*           On entry, TRANS specifies the operation to be performed as
*           follows:
*
*              TRANS = 'N' or 'n'   y := alpha*A*x + beta*y.
*
*              TRANS = 'T' or 't'   y := alpha*A'*x + beta*y.
*
*              TRANS = 'C' or 'c'   y := alpha*A'*x + beta*y.
*
*           Unchanged on exit.
*
*  M      - INTEGER.
*           On entry, M specifies the number of rows of the matrix A.
*           M must be at least zero.
*           Unchanged on exit.
*
*  N      - INTEGER.
*           On entry, N specifies the number of columns of the matrix A.
*           N must be at least zero.
*           Unchanged on exit.
*
*  ALPHA  - DOUBLE PRECISION.
*           On entry, ALPHA specifies the scalar alpha.
*           Unchanged on exit.
*
*  A      - DOUBLE PRECISION array of DIMENSION ( LDA, n ).
*           Before entry, the leading m by n part of the array A must
*           contain the matrix of coefficients.
*           Unchanged on exit.
*
*  LDA    - INTEGER.
*           On entry, LDA specifies the first dimension of A as declared
*           in the calling (sub) program. LDA must be at least
*           max( 1, m ).
*           Unchanged on exit.
*
```

```
*  X      - DOUBLE PRECISION array of DIMENSION at least
*           ( 1 + ( n - 1 )*abs( INCX ) ) when TRANS = 'N' or 'n'
*           and at least
*           ( 1 + ( m - 1 )*abs( INCX ) ) otherwise.
*           Before entry, the incremented array X must contain the
*           vector x.
*           Unchanged on exit.
*
*  INCX   - INTEGER.
*           On entry, INCX specifies the increment for the elements of
*           X. INCX must not be zero.
*           Unchanged on exit.
*
*  BETA   - DOUBLE PRECISION.
*           On entry, BETA specifies the scalar beta. When BETA is
*           supplied as zero then Y need not be set on input.
*           Unchanged on exit.
*
*  Y      - DOUBLE PRECISION array of DIMENSION at least
*           ( 1 + ( m - 1 )*abs( INCY ) ) when TRANS = 'N' or 'n'
*           and at least
*           ( 1 + ( n - 1 )*abs( INCY ) ) otherwise.
*           Before entry with BETA non-zero, the incremented array Y
*           must contain the vector y. On exit, Y is overwritten by the
*           updated vector y.
*
*  INCY   - INTEGER.
*           On entry, INCY specifies the increment for the elements of
*           Y. INCY must not be zero.
*           Unchanged on exit.
*
*
*  Level 2 Blas routine.
*
*  -- Written on 22-October-1986.
*     Jack Dongarra, Argonne National Lab.
*     Jeremy Du Croz, Nag Central Office.
*     Sven Hammarling, Nag Central Office.
*     Richard Hanson, Sandia National Labs.
*
*
*     .. Parameters ..
      DOUBLE PRECISION ONE,ZERO
      PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
*     ..
*     .. Local Scalars ..
      DOUBLE PRECISION TEMP
      INTEGER I,INFO,IX,IY,J,JX,JY,KX,KY,LENX,LENY
*     ..
*     .. External Functions ..
      LOGICAL LSAME
      EXTERNAL LSAME
*     ..
*     .. External Subroutines ..
      EXTERNAL XERBLA
*     ..
*     .. Intrinsic Functions ..
      INTRINSIC MAX
*     ..
*
*     Test the input parameters.
```

```
*
      INFO = 0
      IF (.NOT.LSAME(TRANS,'N') .AND. .NOT.LSAME(TRANS,'T') .AND.
     +    .NOT.LSAME(TRANS,'C')) THEN
          INFO = 1
      ELSE IF (M.LT.0) THEN
          INFO = 2
      ELSE IF (N.LT.0) THEN
          INFO = 3
      ELSE IF (LDA.LT.MAX(1,M)) THEN
          INFO = 6
      ELSE IF (INCX.EQ.0) THEN
          INFO = 8
      ELSE IF (INCY.EQ.0) THEN
          INFO = 11
      END IF
      IF (INFO.NE.0) THEN
          CALL XERBLA('DGEMV ',INFO)
          RETURN
      END IF
*
*     Quick return if possible.
*
      IF ((M.EQ.0) .OR. (N.EQ.0) .OR.
     +    ((ALPHA.EQ.ZERO).AND. (BETA.EQ.ONE))) RETURN
*
*     Set  LENX  and  LENY, the lengths of the vectors x and y, and set
*     up the start points in  X  and  Y.
*
      IF (LSAME(TRANS,'N')) THEN
          LENX = N
          LENY = M
      ELSE
          LENX = M
          LENY = N
      END IF
      IF (INCX.GT.0) THEN
          KX = 1
      ELSE
          KX = 1 - (LENX-1)*INCX
      END IF
      IF (INCY.GT.0) THEN
          KY = 1
      ELSE
          KY = 1 - (LENY-1)*INCY
      END IF
*
*     Start the operations. In this version the elements of A are
*     accessed sequentially with one pass through A.
*
*     First form  y := beta*y.
*
      IF (BETA.NE.ONE) THEN
          IF (INCY.EQ.1) THEN
              IF (BETA.EQ.ZERO) THEN
                  DO 10 I = 1,LENY
                      Y(I) = ZERO
   10             CONTINUE
              ELSE
                  DO 20 I = 1,LENY
                      Y(I) = BETA*Y(I)
```

```
   20             CONTINUE
              END IF
          ELSE
              IY = KY
              IF (BETA.EQ.ZERO) THEN
                  DO 30 I = 1,LENY
                      Y(IY) = ZERO
                      IY = IY + INCY
   30             CONTINUE
              ELSE
                  DO 40 I = 1,LENY
                      Y(IY) = BETA*Y(IY)
                      IY = IY + INCY
   40             CONTINUE
              END IF
          END IF
       END IF
       IF (ALPHA.EQ.ZERO) RETURN
       IF (LSAME(TRANS,'N')) THEN
*
*        Form  y := alpha*A*x + y.
*
          JX = KX
          IF (INCY.EQ.1) THEN
              DO 60 J = 1,N
                  IF (X(JX).NE.ZERO) THEN
                      TEMP = ALPHA*X(JX)
                      DO 50 I = 1,M
                          Y(I) = Y(I) + TEMP*A(I,J)
   50                 CONTINUE
                  END IF
                  JX = JX + INCX
   60         CONTINUE
          ELSE
              DO 80 J = 1,N
                  IF (X(JX).NE.ZERO) THEN
                      TEMP = ALPHA*X(JX)
                      IY = KY
                      DO 70 I = 1,M
                          Y(IY) = Y(IY) + TEMP*A(I,J)
                          IY = IY + INCY
   70                 CONTINUE
                  END IF
                  JX = JX + INCX
   80         CONTINUE
          END IF
       ELSE
*
*        Form  y := alpha*A'*x + y.
*
          JY = KY
          IF (INCX.EQ.1) THEN
              DO 100 J = 1,N
                  TEMP = ZERO
                  DO 90 I = 1,M
                      TEMP = TEMP + A(I,J)*X(I)
   90             CONTINUE
                  Y(JY) = Y(JY) + ALPHA*TEMP
                  JY = JY + INCY
  100         CONTINUE
          ELSE
```

```
              DO 120 J = 1,N
                  TEMP = ZERO
                  IX = KX
                  DO 110 I = 1,M
                      TEMP = TEMP + A(I,J)*X(IX)
                      IX = IX + INCX
  110             CONTINUE
                  Y(JY) = Y(JY) + ALPHA*TEMP
                  JY = JY + INCY
  120         CONTINUE
          END IF
      END IF
*
      RETURN
*
*     End of DGEMV .
*
      END
```