Example usage of BLAS and LAPACK

Calculation of dot product of two matrices using `dot_product` and its optimized version in BLAS

```
  function array2_dotproduct(a,b) result(res)

    implicit none
    type(array2), intent(in) :: a,b
    real(realk) :: ddot
    real(realk) :: res
    integer :: nelements
    integer :: i

    res=0.0d0

#ifdef USE_BLAS
    nelements = a%dims(1)*a%dims(2)
    res = ddot(nelements,a%val,1,b%val,1)
#else
    do i=1,a%dims(2)
      res = res + dot_product(a%val(:,i),b%val(:,i))
    end do
#endif

    return
  end function array2_dotproduct
```

Matrix matrix multiplication using `matmul` and external `dgemm` from BLAS

```
    subroutine array2_matmul(a,b,c,trans_a,trans_b,alpha,beta)

    implicit none
    type(array2), intent(inout) :: a,b,c
    character, intent(in) :: trans_a, trans_b
    real(realk), intent(in) :: alpha, beta

    ! transpose
    if(trans_a == 't' .or. trans_a == 'T') call array2_transpose(a)
    if(trans_b == 't' .or. trans_b == 'T') call array2_transpose(b)

#ifdef USE_BLAS
    call dgemm('n','n',a%dims(1),b%dims(2),a%dims(2),alpha,a%val, &
            a%dims(1),b%val,b%dims(1),beta,c%val,c%dims(1))
#else
    c%val = beta*c%val+alpha*matmul(a%val,b%val)
#endif

    ! transpose back
    if(trans_a == 't' .or. trans_a == 'T') call array2_transpose(a)
    if(trans_b == 't' .or. trans_b == 'T') call array2_transpose(b)

    return
  end subroutine array2_matmul
```

Calculation of Lowdin matrix decomposition (into inverse square root) using LAPACK

```
subroutine get_lowdin(msqr,n,inpm)

  implicit none
  integer, intent(in) :: n
  integer :: ifail, lwork, i, liwork
  integer, dimension(128*n) :: iwork
  real(realk), intent(in), dimension(n,n) :: inpm
  real(realk), intent(out), dimension(n,n) :: msqr
  real(realk), dimension(n,n) :: x, d
  real(realk), dimension(n) :: eval
  real(realk), dimension(128*n+64*n*n) :: work
  external dsyevd

  liwork=128*n
  lwork = 128*n+64*n*n
  x = inpm
  ifail = 0
  work = 0.0D0
  eval = 0.0D0

  call dsyevd('v','u',n,x,n,eval,work,lwork,iwork,liwork,ifail)
  if(ifail /= 0) then
      stop 'error :: error in dsyevd'
  end if
  d = 0.0D0

  do i = 1, n
      d(i,i) = 1.0d0/sqrt(eval(i))
  end do
  msqr = matmul(matmul(x,d),transpose(x))
  return
end subroutine get_lowdin
```