BLAS-RMD Reference Manual

Kristjan Jonasson, Sven Sigurdsson, Hordur Freyr Yngvason,
Petur Orri Ragnarsson, Pall Melsted

August 2019

Contents

1 Introduction

The BLAS_RMD package consists of a set Fortran subroutines that compute the reverse mode algorithmic derivatives of all the real BLAS operations, and in addition there is a routine for the derivative of potrf from LAPACK. Like the non-complex BLAS themselves, these routines come in two flavors, single and double precision. Corresponding to each BLAS routine there is a derivative subroutine with a name obtained by prefixing s or d and suffixing _rmd. For BLAS that have scalar parameters (alpha, beta) there is an additional derivative routine with a name obtained by suffixing _rmds. Thus, the double precision derivatives of axpy are implemented in the subroutines daxpy_rmd and daxpy_rmds.

This reference manual contains documentation for all the 53 subroutines of the package, 35 to compute vector and matrix adjoints of 34 BLAS routines and Lapack's potrf, in the same order as the corresponding BLAS routines appear in the Netlib Quick Reference Guide [1], and an additional 18 to compute scalar adjoints of those BLAS routines that include scalar parameters. Only single precision versions of the subroutines are covered, as the corresponding double precision versions are direct translations.

The parameter list of an RMD-subroutine begins with the parameters of the corresponding BLAS, in the original order, leaving out parameters that are not needed. This part of the list includes control character options (`side`, `uplo`, `diag` and `trans_`), vector and matrix sizes, leading dimensions, bandwidths and vector increments. If the parameters were changed by the BLAS, usually they should have the value that they had on exit from the BLAS, but in a few cases it is the entry value that is needed, and then the parameter name is suffixed with 0. Note that the reference BLAS documentation is a little inaccurate in the description of symmetric matrix parameters: A is used both to denote the symmetrix matrix itself and the corresponding triangular matrix parameter. The RMD-documentation uses sym(A) or sym(A') for the matrix and A for the parameter.

This list of original BLAS arguments is followed by a list of adjoints, both those needed as input and those computed/updated by the RMD-routine. These are listed in the same order as the corresponding BLAS parameters. In all cases it is assumed that the adjoints are stored in vectors/matrices with the same size, shape, leading dimension, bandwidth and/or increment as the corresponding original quantities. Thus there are no new character control or integer parameters in the list.

Next in the list for a few of the routines is a work space parameter. For the sake of efficiency no memory allocation takes place in the RMD-routines. Finally, for many routines the parameter list ends with a character parameter called SEL that is used to select which adjoints should be computed. Normally a computation involves some constant matrices and vectors (such as measurements entering regression analysis) and adjoints of these are normally not required. The SEL parameter takes the form '110', indicating that adjoints of the first two BLAS vectors/matrices will be computed, but not of the third. Note that parameters, which do not appear in formulae for adjoints that are computed according to the SEL-value, will not be referenced. In such cases a dummy argument may be passed.

One issue that needs attention is whether the RMD-routines should update (i.e. add to or subtract from) or assign to the adjoint parameters. In Table 3 in the accompanying article [2] all the formulae are specified as updates, with either $+=$ or $-=$. However many BLAS routines overwrite input parameters with new values. Inspection of the examples in the demo folder (see also Section 6 in the accompanying article [2]) demonstrates that it is natural to let the RMD-routines assign to the adjoints of these parameters. The value of such a parameter on input to the BLAS routine will not be used again during the forward traversing of the computation tree, and thus if the RM computation is done in reverse order, this will be the first occurence of the corresponding adjoint in the reverse traversing of the computation tree. The documentation below provides information on which parameters are updated and which ones are assigned to. In the latter case, we state that the parameter is computed.

The Fortran language and the BLAS specification allows for the possibility of repeated input arguments, i.e. the same variable being passed to multiple parameters. There are three BLAS operations, dot, ger and gemm, where such use may be sensible (e.g. the differentiation of $B = A^2$). For these three operations the reference manual contains notes explaining how the corresponding adjoints could be computed.

For an example, assume that BLAS was called with:

```
call stbsv(uplo, trans, diag, n, k, A, n, x, 1),
```

and that the adjoint of `A` should be updated, but not that of `x`. Then the RMD-call could be:

```
call stbsv_rmd(uplo, trans, diag, n, k, A, n, x, 1, Ai, 0.0, wrk, '10'),
```

where all parameters before `Ai` should be as on the BLAS call, `Ai` should be a triangular matrix with leading dimension `n`, bandwith `k`, and storage properties the same as `A` (according to `uplo`, `trans`, and `diag`), and `wrk` should be a single precision workspace vector of dimension at least `n`. Since `xi` is not accessed it can be specified as `0.0`.

For another example, assume the BLAS call

```
call dtpmv('U', 'T', 'N', n, AP, x, 1).
```

The corresponding RMD-call could be

```
call dtpmv_rmd('U', 'T', 'N', n, AP, x0, 1, 0d0, xi, 1, '01')
```

where the assignment `x0 = x` should be placed before the call to `dtpmv`.

2 Derivatives of level 1 BLAS

SROTG

```
SUBROUTINE SROTG_RMD(c, s, d, aa, ba, ca, sa, da)

PURPOSE
   Calculates the reverse mode derivative of SROTG from BLAS.

ARGUMENTS
   If SROTG was called with the arguments

      a, b, c, s

   then the corresponding call to SROTG_RMD should begin with the arguments

      c, s

   with the same values as they had on exit from SROTG. Both of these
   arguments will remain unchanged on exit. Note that a and b are omitted.
   In addition the following arguments should be provided:

      d   (input, real scalar)
          the d computed by SROTG and returned in the a-parameter

      aa  (output, real scalar)
          aa := the adjoint of the a supplied to SROTG

      ba  (output, real scalar)
          ba := the adjoint of the b supplied to SROTG

      ca  (input, real scalar)
          the adjoint of the c produced by SROTG

      sa  (input, real scalar)
          the adjoint of the s produced by SROTG

      da  (input, real scalar)
          the adjoint of the d returned by SROTG in the a-parameter

NOTES
   a) A sel parameter is not offered. The adjoints of a and b are always
      computed together; they are considered to form a pair.
   b) When d = 0 the adjoints of a and b are undefined and returned as 0
   c) Adjoints via z which SROTG computes and returns in the b-parameter
      (mostly due to historical reasons) are not supported.

OPERATIONS
   BLAS: d := sigma*sqrt(a^2 + b^2)
         c := a/d unless d=0, then c := 1
         s := b/d unless d=0, then s := 0
         where:
            sigma = sign(a) if |a| > |b|
            sigma = sign(b) if |a| <= |b|
   RMD:  aa := c1 + c*d1
         ba := s1 + s*d1
         where:
            c1 = ca/d
            s1 = sa/d
            d1 = da - s*s1 - c*c1
```

SROTMG

```
SUBROUTINE SROTMG_RMD(d1, d2, x1, param, d1a, d2a, x1a, y1a, parama)

PURPOSE
   Calculates the reverse mode derivative of SROTMG from BLAS.

ARGUMENTS
   If SROTMG was called with the arguments

      d1, d2, x1, y1, param

   then the corresponding call to SROTMG_RMD should begin with arguments

      d1, d2, x1, param

   having the same values as they had on exit from SROTMG. These arguments
   will remain unchanged on exit from SROTMG_RMD. Note that y1 is omitted.
   In addition the following arguments should be provided:

      d1a, d2a
         (input, output, real scalars)
         On entry: The adjoints of the d1 and d2 produced by SROTMG
         On exit: The adjoints of the d1 and d2 supplied to SROTMG

      x1a
         (input, output, real scalar)
         On entry: The adjoint of the x1 produced by SROTMG
         On exit: The adjoint of the x1 supplied to SROTMG

      y1a
         (output, real scalar)
         The adjoint of the y1 supplied to SROTMG

      parama
         (input, output, real vector of dimension 5 or 8)
         On entry:
              The entries corresponding to elements in param with elements
              of the matrix H should contain the adjoints of these H
              elements. If parama(1) = 2 then parama should have dimension 8
         On exit with parama(1) = 2:
              Information about the computations, cf. [1]:
              parama(6): The value of Flag before scaling
              parama(7): Gamma for d1
              parama(8): Gamma for d2

NOTES
   A sel parameter is not offered. The adjoints of d1, d2, x1 and y1 are
   always computed together.

OPERATIONS
   BLAS: Provided by the formulae in [1]
   RMD:  Obtained by differentiating the formulae in [1]

   See also comments in srotm_rmd.f90

[1] CL Lawson et. al., Basic linear algebra subprograms for Fortran usage,
    ACM TOMS 5, 1979, 308-323.
```

SROT

```
SUBROUTINE SROT_RMD(n, x, incx, y, incy, c, s, xa, ya, ca, sa, sel)

PURPOSE
   Calculates the reverse mode derivative of SROT from BLAS.

ARGUMENTS
   If SROT was called with the arguments

      n, x, incx, y, incy, c, s

   then the corresponding call to SROTG_RMD should begin with the arguments

      n, x, incx, y, incy, c, s

   with the same values as they had on exit from the SROT-call. All these
   arguments will remain unchanged on exit. In addition the following
   arguments should be provided:

      xa
         (input, output, real vector of the same dimension and increment as x)
         On entry: The adjoint of the x produced by SROT
         On exit: The adjoint of the x supplied to SROT

      ya
         (input, output, real vector of the same dimension and increment as y)
         On entry: The adjoint of the y produced by SROT
         On exit: The adjoint of the y supplied to SROT

      ca
         (input, output, real scalar)
         ca += the adjoint of c due to the SROT-call

      sa
         (input, output, real scalar)
         ca += the adjoint of c due to the SROT-call

      sel
         (input, character*3)
         Used to select which adjoints to update/compute:
           sel(1:1) = '1' if (xa,ya) should be computed, else sel(1:1) = '0'
           sel(2:2) = '1' if (ca,sa) should be updated, else sel(2:2) = '0'
         For example, to update only (xa,ya), set sel = '10'.

OPERATIONS
   BLAS: [x'; y'] := G*[x'; y']
   RMD:  [xa'; ya'] := G'*[xa' ya']
         [ca; sa] += G'*[c1; s1]
         where:
            c1 = dot(xa,x) + dot(ya,y)  (xa, ya are values on entry)
            s1 = dot(xa,y) - dot(ya,x)  (xa, ya are values on entry)
            and G = [c s; -s c]
```

SROTM

```
SUBROUTINE SROTM_RMD(n, x, incx, y, incy, param, xa, ya, parama, sel)

PURPOSE
   Calculates the reverse mode derivative of SROTM from BLAS.

ARGUMENTS
   If SROTM was called with the arguments

      n, x, incx, y, incy, param

   then the corresponding call to SROTM_RMD should begin with the arguments

      n, x, incx, y, incy, param

   with the same values as they had on exit from the SROTM-call. All these
   arguments will remain unchanged on exit. In addition the following
   arguments should be provided:

   xa     (input, output, real vector of the same dimension and increment as x)
          On entry: The adjoint of the x produced by SROTM
          On exit: The adjoint of the x supplied to SROTM

   ya     (input, output, real vector of the same dimension and increment as y)
          On entry: The adjoint of the y produced by SROTM
          On exit: The adjoint of the y supplied to SROTM

   parama (input, output, real scalar)
          parama += the adjoint of param due to the SROTM-call. Only entries
          corresponding to non-fixed param entries are updated

   sel    (input, character*3)
          Used to select which adjoints to update/compute:
            sel(1:1) = '1' if (xa,ya) should be computed, else sel(1:1) = '0'
            sel(2:2) = '1' if parama should be updated, else sel(2:2) = '0'
          For example, to update only (xa,ya), set sel = '10'.

OPERATIONS
   BLAS: [x'; y'] := H*[x'; y']
         where:
            H = [ 1  0;  0  1] if flag = -2
            H = [p2 p4; p3 p5] if flag = -1
            H = [ 1 p4; p3  1] if flag = 0
            H = [p2 -1;  1 p5] if flag = 1
            flag = param(1), pi = param(i)
   RMD:  [ya'; xa'] := K*[ya'; xa']
         Ha' += inv(H)*A
         where:  K = [h22 h12; h21 h22] = matrix with param = [flag p5 p3 p4 p2]
                 a11 = dot(x,xa)  a12 = dot(x,ya)  (xa, ya are values on entry)
                 a21 = dot(y,xa)  a22 = dot(y,ya)  (xa, ya are values on entry)
         Elements of Ha corresponding to fixed elements of H remain unchanged

NOTES
   1. The elements and structure of H is passed in param = [flag, p2, p3, p4, p5])
   2. H-elements that are -1, 0 or 1 are referred to as *fixed*
   3. For further details, see (a) Table 5 in the accompanying article [1],
      (b) Remark in srotm-rmd.f90, (c) The Netlib documentation of drotm and
      (d) The online NAG documentation of F06EQF (DROTM)

[1] K Jonasson et al. RMAD of BLAS Operations, ACM TOMS 2019.
```

SSWAP

```
SUBROUTINE SSWAP_RMD(n, incx, incy, xa, ya)

PURPOSE
   Calculate the reverse mode derivative of SSWAP from BLAS.


ARGUMENTS
   If SSWAP was called with the arguments

      n, x, incx, y, incy

   then the corresponding call to SSWAP_RMD should begin with the arguments

      n, incx, incy

   with the same values. These arguments will remain unchanged on exit. In
   addition the following arguments should be provided:

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa := the value of ya on entry

   ya
      (input, output, real vector of the same dimension and increment as y)
      ya := the value of xa on entry

OPERATIONS
   BLAS: x <--> y
   RMD:  xa <--> ya
```

SSCAL

```
SUBROUTINE SSCAL_RMD(n, alpha, incx, xa)

PURPOSE
   Calculate the reverse mode derivative of SSCAL from BLAS.


ARGUMENTS
   If SSCAL was called with the arguments

      n, alpha, x, incx

   then SSCAL_RMD should be called with the arguments

      n, alpha, incx

   with the same values. These arguments will remain unchanged on exit. Note
   that x is omitted. In addition the following parameter should be provided:

   xa
      (output, real vector of the same dimension and increment as x)
      xa := adjoint of x

OPERATIONS
   BLAS: x := alpha*x
   RMD:  xa := alpha*xa
```

SCOPY

```
SUBROUTINE SCOPY_RMD(n, incx, incy, xa, ya)

PURPOSE
   Calculate the reverse mode derivative of SCOPY from BLAS.

ARGUMENTS
   If SCOPY was called with the arguments

      n, x, incx, y, incy,

   then the call to SCOPY_RMD should begin with the arguments

      n, incx, incy

   with the same values. These arguments will remain unchanged on exit. In
   addition the following arguments should be provided:

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += adjoint of x

   ya
      (input, real vector of the same dimension and increment as y)
      ya += adjoint of y

NOTE
   As there is only one output there is no need for a sel parameter

OPERATIONS
   BLAS: y = x
   RMD:  xa += ya
         ya unchanged
```

SAXPY

```
SUBROUTINE SAXPY_RMD(n, alpha, incx, incy, xa, ya)

PURPOSE
   Calculates the reverse mode derivative of SAXPY from BLAS.

ARGUMENTS
   If SAXPY was called with the arguments

      n, alpha, x, incx, y, incy

   then the corresponding call to SAXPY_RMD should begin with the arguments

      n, alpha, incx, incy

   with the same values. These arguments will remain unchanged on exit. In
   addition the following arguments should be provided:

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SAXPY call.

   ya
      (input, real vector of the same dimension and increment as y)
      The adjoint of y.

NOTE
   SAXPY computes y := alpha*x + y, so that the adjoint of y is unchanged,
   and needs no update. Therefore a sel parameter is not needed (it is
   implicitly assumed to be '1X', to update xa, X may be 0 or 1 because ya
   is unchanged)

OPERATIONS
   BLAS: y := alpha*x + y
   RMD:  xa += alpha*ya
         ya unchanged
```

SDOT

```
SUBROUTINE SDOT_RMD(n, x, incx, y, incy, dota, xa, ya, sel)

PURPOSE
   Calculate the reverse mode derivative of SDOT from BLAS.

ARGUMENTS
   If SDOT was called with the statement

      dot = SDOT(n, x, incx, y, incy)

   then SDOT_RMD should be called with the same arguments:

      n, x, incx, y, incy

   with the same values. These arguments will remain unchanged on exit. In
   addition the following arguments should be provided:

   dota
      (input, real scalar)
      The adjoint of dot.

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SDOT call.

   ya
      (input, output, real vector of the same dimension and increment as y)
      ya += the adjoint of y due to the SDOT call.

   sel
      (input, character*2)
      Used to select which adjoints to update:
          sel(1:1) = '1' if xa should be updated, else sel(1:1) = '0'
          sel(2:2) = '1' if ya should be updated, else sel(2:2) = '0'
      For example, to update only xa, set sel = '10'.

NOTE
   To compute the adjoint of square norm, s = sdot(n, x, 1, x, 1) one may
   use the following calls:

      call sdot_rmd(n, x, 1, x, 1, sa, xa, dummy, '10')
      call sscal(n, 2.0, xa, 1)

OPERATIONS
   BLAS: dot = x'*y
   RMD:  dota unchanged
         xa += y*dota
         ya += x*dota
```

SDSDOT

```
SUBROUTINE SDSDOT_RMD(n, x, incx, y, incy, dota, ba, xa, ya, sel)

PURPOSE
   Calculate the reverse mode derivative of SDSDOT from BLAS.

ARGUMENTS
   If SDSDOT was called with the statement

      dot = SDSDOT(n, b, x, incx, y, incy)

   then SDSDOT_RMD should be called with the same arguments:

      n, x, incx, y, incy

   with the same values as they had on the SDSDOT call. These arguments will
   remain unchanged on exit. Note that b is omitted. In addition the
   following arguments should be provided:

   dota
      (input, real scalar)
      The adjoint of dot.

   ba
      (input, output, real scalar)
      ba += the adjoint of b due to the SDSDOT call.

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SDSDOT call.

   ya
      (input, output, real vector of the same dimension and increment as y)
      ya += the adjoint of y due to the SDSDOT call.

   sel
      (input, character*2)
      Used to select which adjoints to update:
          sel(1:1) = '1' if ba should be updated, else sel(1:1) = '0'
          sel(2:2) = '1' if xa should be updated, else sel(1:1) = '0'
          sel(3:3) = '1' if ya should be updated, else sel(2:2) = '0'
      For example, to update only xa, set sel = '010'.

NOTE
   SDSDOT has no double precision version and neither does SDSDOT_RMD

OPERATIONS
   BLAS: dot = b + x'*y
   RMD:  dota unchanged
         ba += dota
         xa += y*dota
         ya += x*dota
```

SNRM2

```
SUBROUTINE SNRM2_RMD(n, x, incx, b, xa, ba)

PURPOSE
   Calculate the reverse mode derivative of SNRM2 from BLAS.

ARGUMENTS
   If SNRM2 was called with the statement

      b = SNRM2(n, x, incx)

   then SNRM2_RMD should be called with the same arguments:

      n, x, incx

   with the same values. These arguments will remain unchanged on exit. In
   addition the following arguments should be provided:

   b
      (input, real scalar)
      The result of the SNRM2 call.

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SDOT call.

   ba
      (input, real scalar)
      The adjoint of b

NOTE
   As there is only one vector input a sel parameter is not needed.

OPERATIONS
   BLAS:  b := norm(x)  (2-norm)
   RMD:   xa += ba*x/b
          ba unchanged
```

SASUM

```
SUBROUTINE SASUM_RMD(n, x, incx, xa, ba)

PURPOSE
   Calculate the reverse mode derivative of SASUM from BLAS.

ARGUMENTS
   If SASUM was called with the statement

      b = SASUM(n, x, incx)

   then SASUM_RMD should be called with the arguments:

      n, x, incx

   with the values which they had on the SASUM call. These arguments will
   remain unchanged on exit. Note that b is omitted. In addition the
   following arguments should be provided:

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SASUM call.

   ba
      (input, real scalar)
      The adjoint of b

NOTES
   SASUM is not differentiable for x-elements which are 0. The adjoints
   of such elements is returned as 0.

OPERATIONS
   BLAS:  b := sum |x(i)|  (1-norm)
   RMD:   xa += |ba|*sign(x)
          ba unchanged
          where sign(x) = 1 where x > 0, -1 where x < 0 and 0 where x = 0
```

3 Derivatives of level 2 BLAS

SGEMV

```
SUBROUTINE SGEMV_RMD(trans, m, n, alpha, A, lda, x, incx, beta, incy, Aa, xa, ya, sel)

PURPOSE
   Calculate the reverse mode derivative of SGEMV from BLAS.

ARGUMENTS
   If SGEMV was called with the arguments

      trans, m, n, alpha, A, lda, x, incx, beta, y, incy

   then the corresponding call to SGEMV_RMD should begin with the arguments

      trans, m, n, alpha, A, lda, x, incx, beta, incy

   with the same values. Note that y is omitted. All these arguments will
   remain unchanged on exit. In addition the following arguments should
   be provided:

   Aa
      (input, output, real matrix of the same dimensions as A and stored in
      the same way)
      Aa += the adjoint of A due to the SGEMV call.

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SGEMV call.

   ya
      (input, output, real vector of the same dimension and increment as y)
      On entry: the adjoint of the y produced by SGEMV
      On exit: the adjoint of the y supplied to SGEMV

   sel
      (input, character*3)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if xa should be updated, else sel(2:2) = '0'
         sel(3:3) = '1' if ya should be computed, else sel(3:3) = '0'
      For example, to update only Aa, set sel = '100'.

OPERATIONS
   (when trans = 'N')
   BLAS: y = alpha*A*x + beta*y  for general matrix A
   RMD:  Aa += alpha*ya*x'       where ya is value on entry
         xa += alpha*A'*ya       do.
         ya := beta*ya
   (when trans = 'T')
   BLAS: y = alpha*A'*x + beta*y for general matrix A
   RMD:  Aa += alpha*x*ya'       where ya is value on entry
         xa += alpha*A*ya        do.
         ya := beta*ya
```

SGBMV

```
SUBROUTINE SGBMV_RMD(trans, m, n, kl, ku, alpha, A, lda, x, incx, beta, incy,&

PURPOSE
   Calculate the reverse mode derivative of the BLAS routine SGBMV


ARGUMENTS
   If SGBMV was called with the arguments

      trans, m, n, kl, ku, alpha, A, lda, x, incx, beta, y, incy

   then SGBMV_RMD should be called with the arguments:

      trans, m, n, kl, ku, alpha, A, lda, x, incx, beta, incy

   with the same values. Note that y is omitted. All these arguments will
   remain unchanged on exit. In addition the following arguments should
   be provided:

   Aa
      (input, output, real matrix of the same dimensions and stored in the
      same band form as A)
      Aa += the adjoint of A due to the SGBMV call.

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SGBMV call.

   ya
      (input, output, real vector of the same dimension and increment as x)
      ya := the adjoint of y due to the SGBMV call.

   sel
      (input, character*3)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if xa should be updated, else sel(2:2) = '0'
         sel(3:3) = '1' if ya should be computed, else sel(3:3) = '0'
      For example, to update only xa, set sel = '010'.

OPERATIONS
   Same as for SGEMV except that A and Aa use banded storage.
```

SSYMV

```
SUBROUTINE SSYMV_RMD(uplo, n, alpha, A, lda, x, incx, beta, incy, Aa, xa, ya, sel)

PURPOSE
   Calculate the reverse mode derivative of SSYMV from BLAS.


ARGUMENTS
   If SSYMV was called with the arguments

      uplo, n, alpha, A, lda, x, incx, beta, y, incy

   then the corresponding call to SSYMV_RMD should begin with the same
   arguments

      uplo, n, alpha, A, lda, x, incx, beta, incy

   with the same values. All these arguments will remain unchanged on exit.
   Note that y is omitted. In addition the following arguments should be
   provided:

   Aa
      (input, output, real triangular matrix of the same dimensions as A,
      and stored in the same half according to uplo)
      Aa += the adjoint of A due to the SSYMV call

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SSYMV call

   ya
      (input, output, real vector of the same dimension and increment as y)
      On entry: the adjoint of the y produced by SSYMV
      On exit: the adjoint of the y supplied to SSYMV

   sel
      (input, character*3)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if xa should be updated, else sel(2:2) = '0'
         sel(3:3) = '1' if ya should be computed, else sel(3:3) = '0'
      For example, to update only Aa, set sel = '100'.

OPERATIONS
   (with uplo = 'L')
   BLAS: y = alpha*sym(A)*x + beta*y                  A lower triangular
   RMD:  Aa += alpha*(tril(x*ya'+ya*x') - diag(x*ya'))  where ya is value on entry
         xa += alpha*sym(A)*ya                        do.
         ya := beta*ya
   (with uplo = 'U')
   BLAS: y = alpha*sym(A')*x + beta*y                 A upper triangular
   RMD:  Aa += alpha*(triu(x*ya'+ya*x') - diag(x*ya'))  where ya is value on entry
         xa += alpha*sym(A')*ya                       do.
         ya := beta*ya
```

SSBMV

```
SUBROUTINE SSBMV_RMD(uplo, n, k, alpha, A, lda, x, incx, beta, incy, Aa, xa, ya, sel)

PURPOSE
   Calculate the reverse mode derivative of SSBMV from BLAS.


ARGUMENTS
   If SSBMV was called with the arguments

      uplo, n, k, alpha, A, lda, x, incx, beta, y, incy

   then the corresponding SSBMV_RMD call should begin with the arguments

      uplo, n, k, alpha, A, lda, x, incx, beta, incy

   with the same values. All these arguments will remain unchanged on exit.
   Note that y is omitted. In addition the following arguments should be
   provided:

   Aa
      (input, output, real triangular band matrix of the same dimensions as
      A, stored in the same band form, and stored in the same half according
      to uplo)
      Aa += adjoint of A due to the SSBMV call

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += adjoint of x due to the SSBMV call

   ya
      (input, output, real vector of the same dimension and increment as y)
      On entry: the adjoint of the y produced by SSBMV
      On exit: the adjoint of the y supplied to SSBMV

   sel
      (input, character*3)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if xa should be updated, else sel(2:2) = '0'
         sel(3:3) = '1' if ya should be computed, else sel(3:3) = '0'
      For example, to update only Aa, set sel = '100'.

OPERATIONS
   Same as for SSYMV_RMD except that A and Aa use banded storage
```

SSPMV

```
SUBROUTINE SSPMV_RMD(uplo, n, alpha, AP, x, incx, beta, incy, APa, xa, ya, sel)

PURPOSE
   Calculate the reverse mode derivative of SSPMV from BLAS.


ARGUMENTS
   If SSPMV was called with the arguments

      uplo, n, alpha, AP, x, incx, beta, y, incy

   then the corresponding SSPMV_RMD call should begin with the arguments

      uplo, n, alpha, AP, x, incx, beta, incy

   with the same values. All these arguments will remain unchanged on exit.
   Note that y is omitted. In addition the following arguments should be
   provided:

   APa
      (input, output, real triangular packed matrix stored in a vector with
      (n*(n+1))/2 elements in the same way as AP)
      APa += adjoint of AP due to the SSPMV call

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += adjoint of x due to the SSPMV call

   ya
      (input, output, real vector of the same dimension and increment as y)
      On entry: the adjoint of the y produced by SSPMV
      On exit: the adjoint of the y supplied to SSPMV

   sel
      (input, character*3)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if APa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if xa should be updated, else sel(2:2) = '0'
         sel(3:3) = '1' if ya should be computed, else sel(3:3) = '0'
      For example, to update only APa, set sel = '100'.

OPERATIONS
   Same as for SSYMV_RMD except that A and Aa use packed storage
```

STRMV

```
SUBROUTINE STRMV_RMD(uplo, trans, diag, n, A, lda, x0, incx, Aa, xa, sel)

PURPOSE
   Calculate the reverse mode derivative of STRMV from BLAS.


ARGUMENTS
   If STRMV was called with the arguments

      uplo, trans, diag, n, A, lda, x, incx

   then the corresponding call to STRMV_RMD should begin with the arguments

      uplo, trans, diag, n, A, lda, x0, incx

   which all except x0 should have the same values as they had on the STRMV
   call, and x0 should have the value that x had on entry to the STRMV-call
   (STRMV only changes the x-argument). All these arguments will remain
   unchanged on exit. In addition the following arguments should be
   provided:

   Aa
      (input, output, real triangular matrix of the same dimensions as A,
      and stored in the same half according to uplo)
      Aa += the adjoint of A due to the STRMV call

   xa
      (input, output, real vector of the same dimension and increment as x)
      On entry: the adjoint of the x produced by STRMV
      On exit: the adjoint of the x supplied to STRMV

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if xa should be computed, else sel(2:2) = '0'
      For example, if sel = '01' only xa will be computed

OPERATIONS
   (with uplo = 'L', trans = 'N' and diag = 'N' or 'U')
   BLAS: x := A*x           (*), A is lower triangular
   RMD:  xa := A'*xa        (**)
         Aa += tril(xa*x')  where x is input to (*), xa is input to (**)

   (with uplo = 'U', trans = 'N' and diag = 'N' or 'U')
   BLAS: x := A*x           (*), A is upper triangular
   RMD:  xa := A'*xa        (**)
         Aa += triu(xa*x')  where x is input to (*), xa is input to (**)

   (with uplo = 'L', trans = 'T' and diag = 'N' or 'U')
   BLAS: x := A'*x          (*), A is lower triangular
   RMD:  xa := A*xa         (**)
         Aa += tril(x*xa')  where x is input to (*), xa is input to (**)

   (with uplo = 'U', trans = 'T' and diag = 'N' or 'U')
   BLAS: x := A'*x          (*), A is upper triangular
   RMD:  xa := A*xa         (**)
         Aa += triu(x*xa')  where x is input to (*), xa is input to (**)
```

STBMV

```
SUBROUTINE STBMV_RMD(uplo, trans, diag, n, k, A, lda, x0, incx, Aa, xa, sel)

PURPOSE
   Calculate the reverse mode derivative of the STBMV from BLAS.


ARGUMENTS
   If STBMV was called with the following arguments:

      uplo, trans, diag, n, k, A, lda, x, incx

   then the corresponding call to STBMV_RMD should begin with the arguments

      uplo, trans, diag, n, k, A, lda, x0, incx

   which all except x0 should have the same values as they had on the STBMV
   call, and x0 should have the value that x had on entry to the STBMV-call
   (STBMV only changes the x-argument). All these arguments will remain
   unchanged on exit. In addition the following arguments should be
   provided:

   Aa
      (input, output, real matrix of the same dimensions as A, stored in the same
      band form as A, and stored in the same half according to uplo)
      Aa += the adjoint of A due to the STBMV call

   xa
      (input, output, real vector of the same dimension and increment as x)
      On entry: the adjoint of the x produced by STBMV
      On exit: the adjoint of the x supplied to STBMV

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if xa should be computed, else sel(2:2) = '0'
      For example, to compute only xa, set sel = '01'

OPERATIONS
   Same as for STRMV_RMD except that A and Aa use banded storage
```

STPMV

```
SUBROUTINE STPMV_RMD(uplo, trans, diag, n, AP, x0, incx, APa, xa, sel)

PURPOSE
   Calculate the reverse mode derivative of STPMV from BLAS.


ARGUMENTS
   If STPMV was called with the following arguments:

      uplo, trans, diag, n, AP, x, incx

   then the corresponding call to STPMV_RMD should begin with the arguments

      uplo, trans, diag, n, AP, x0, incx

   which all except x0 should have the same values as they had on the STRMV
   call, and x0 should have the value that x had on entry to the STPMV-call
   (STPMV only changes the x-argument). All these arguments will remain
   unchanged on exit. In addition the following arguments should be
   provided:

   APa
      (input, output, real triangular matrix of the same dimensions as AP,
      stored in a vector with (n*(n+1))/2 elements in the same way as AP)
      APa += the adjoint of AP due to the STPMV call

   xa
      (input, output, real vector of the same dimension and increment as x)
      On entry: the adjoint of the x produced by STPMV
      On exit: the adjoint of the x supplied to STPMV

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if APa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if xa should be computed, else sel(2:2) = '0'
      For example, if sel = '01', then only xa will be computed.

OPERATIONS
   Same as for for STRMV_RMD except that packed storage is used
```

STRSV

```
SUBROUTINE STRSV_RMD(uplo, trans, diag, n, A, lda, x, incx, Aa, xa, wrk, sel)

PURPOSE
   Calculate the reverse mode derivative of STRSV from BLAS.

ARGUMENTS
   If STRSV was called with the arguments

      uplo, trans, diag, n, A, lda, x, incx

   then the corresponding call to STRSV_RMD should begin with the same arguments

      uplo, trans, diag, n, A, lda, x, incx

   with the same values. All these arguments will remain unchanged on exit.
   In addition the following arguments should be provided:

   Aa   (input, output, real triangular matrix of the same dimensions as A,
        and stored in the same half according to uplo)
        Aa += adjoint of A due to the STRSV call

   xa   (input, output, real vector of the same dimension and increment as x)
        On entry: The adjoint of the x produced by STRSV
        On exit: The adjoint of the x supplied to STRSV

   wrk  (output, real vector of dimension at least n)
        When sel(2:2) = '0' so that a new xa should not be computed it is
        necessary to supply STRSV_RMD with a workspace vector. When sel(2:2) =
        '1', wrk is not referenced, because xa serves its purpose. In this
        case a dummy value may be given instead

   sel  (input, character*2)
        Used to select which adjoints to update/compute:
           sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
           sel(2:2) = '1' if xa should be computed, else sel(2:2) = '0'
        For example, if sel = '01', then only xa will be computed

OPERATIONS
   (with uplo = 'L', trans = 'N' and diag = 'N' or 'U';
   BLAS: x := inv(A)*x      (*), A is lower triangular
   RMD:  xa := inv(A)'*xa   (**)
         Aa -= tril(xa*x')  x is output from (*), xa is output from (**)

   (with uplo = 'U', trans = 'N' and diag = 'N' or 'U')
   BLAS: x := inv(A)*x      (*), A is upper triangular
   RMD:  xa := inv(A)'*xa   (**)
         Aa -= triu(xa*x')  x is output from (*), xa is output from (**)

   (with uplo = 'L', trans = 'T' and diag = 'N' or 'U')
   BLAS: x := inv(A)'*x     (*), A is lower triangular
   RMD:  xa := inv(A)*xa    (**)
         Aa -= tril(x*xa')  x is output from (*), xa is output from (**)

   (with uplo = 'U', trans = 'T' and diag = 'N' or 'U')
   BLAS: x := inv(A)'*x     (*), A is upper triangular
   RMD:  xa := inv(A)*xa    (**)
         Aa -= triu(x*xa')  x is output from (*), xa is output from (**)
```

STBSV

```
SUBROUTINE STBSV_RMD(uplo, trans, diag, n, k, A, lda, x, incx, Aa, xa, wrk, sel)

PURPOSE
   Calculate the reverse mode derivative of the STBSV from BLAS.

ARGUMENTS
   If STBSV was called with the following arguments:

      uplo, trans, diag, n, k, A, lda, x, incx

   then the corresponding call to STBSV_RMD should begin with the same arguments

      uplo, trans, diag, n, k, A, lda, x, incx

   with the same values. All these arguments will remain unchanged on exit.
   In addition the following arguments should be provided:

   Aa
      (input, output, real matrix of the same dimensions as A, stored in the same
      band form as A, and stored in the same half according to uplo)
      Aa += adjoint of A due to the STBSV call

   xa
      (input, output, real vector of the same dimension and increment as x)
      On entry: The adjoint of the x produced by STBSV
      On exit: The adjoint of the x supplied to STBSV

   wrk
      (output, real vector of dimension at least n)
      When sel(2:2) = '0' so that a new xa should not be computed it is
      necessary to supply STBSV_RMD with a workspace vector. When sel(2:2) =
      '1', wrk is not referenced, because xa serves its purpose. In this case
      a dummy value may be given instead

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if xa should be computed, else sel(2:2) = '0'
      For example, if sel = '01', then only xa will be computed

OPERATIONS
   Same as for STRSV_RMD except that A and Aa use banded storage
```

STPSV

```
SUBROUTINE STPSV_RMD(uplo, trans, diag, n, AP, x, incx, APa, xa, wrk, sel)

PURPOSE
   Calculate the reverse mode derivative of STPSV from BLAS.

ARGUMENTS
   If STPSV was called with the following arguments:

      uplo, trans, diag, n, AP, x, incx

   then the corresponding call to STPSV_RMD should begin with the same arguments

      uplo, trans, diag, n, AP, x, incx

   with the same values. All these arguments will remain unchanged on exit.
   In addition the following arguments should be provided:

   APa
      (input, output, real triangular matrix of the same dimensions as AP,
      stored in a vector with (n*(n+1))/2 elements in the same way as AP)
      APa += adjoint of AP due to the STPSV call

   xa
      (input, output, real vector of the same dimension and increment as x)
      On entry: The adjoint of the x produced by STPSV
      On exit: The adjoint of the x supplied to STPSV

   wrk
      (output, real vector of dimension at least n)
      When sel(2:2) = '0' so that a new xa should not be computed it is
      necessary to supply STPSV_RMD with a workspace vector. When sel(2:2) =
      '1', wrk is not referenced, because xa serves its purpose. In this case
      a dummy value may be given instead

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if APa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if xa should be computed, else sel(2:2) = '0'
      For example, if sel = '01', then only xa will be computed

OPERATIONS
   Same as for STRSV_RMD except that packed storage is used
```

SGER

```
SUBROUTINE SGER_RMD(m, n, alpha, x, incx, y, incy, lda, xa, ya, Aa, sel)

PURPOSE
   Calculate the reverse mode derivative of SGER from BLAS.


ARGUMENTS
   If SGER was called with the arguments

      m, n, alpha, x, incx, y, incy, A, lda

   then the corresponding call to SGER_RMD should begin with the arguments:

      m, n, alpha, x, incx, y, incy, lda

   with the same values. All these arguments will remain unchanged on exit.
   Note that A is omitted. In addition the following arguments should be
   provided:

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SGER call.

   ya
      (input, output, real vector of the same dimension and increment as y)
      ya += the adjoint of y due to the SGER call.

   Aa
      (input, real matrix of the same dimensions as A)
      The adjoint of A.

   sel
      (input, character*2)
      Used to select which adjoints to update:
         sel(1:1) = '1' if xa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if ya should be updated, else sel(2:2) = '0'
      For example, to update only xa, set sel = '10'.

NOTE
   To compute A += alpha*x*x' one may call sger with a repeated argument,
   e.g. call sger(n, n, alpha, x, 1, x, 1, A, n). The adjoint of x can
   then be computed with:
      call sger_rmd(n, n, alpha, x, 1, x, 1, n, xa, dummy, Aa, '10')
      call sscal(n, 2.0, xa, 1).

OPERATIONS
   BLAS: A += alpha*x*y'
   RMD:  xa += alpha*Aa*y
         ya += alpha*Aa'*x
         Aa unchanged
```

SSYR

```
SUBROUTINE SSYR_RMD(uplo, n, alpha, x, incx, lda, xa, Aa)

PURPOSE
   Calculate the reverse mode derivative of SSYR from BLAS.


ARGUMENTS
   If SSYR was called with the arguments

      uplo, n, alpha, x, incx, A, lda

   then the corresponding call to SSYR_RMD should begin with the arguments

      uplo, n, alpha, x, incx, lda

   with the same values. All these arguments will remain unchanged on exit.
   Note that A is omitted. In addition the following arguments should be
   provided:

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SSYR call.

   Aa
      (input, real triangular matrix of the same dimensions as A, and stored
      in the same half according to uplo)
      The adjoint of A.

NOTE
   Aa is always unchanged so that a sel parameter is not needed.

OPERATIONS
   BLAS: A += alpha*tril(x*x') i.e. sym(A) := alpha*x*x' + sym(A)
   RMD:  xa += alpha*(Aa + Aa')*x, equiv.to: xa += alpha*(diag(Aa)+sym(Aa))*x
         Aa unchanged
```

SSPR

```
SUBROUTINE SSPR_RMD(uplo, n, alpha, x, incx, xa, APa)

PURPOSE
   Calculate the reverse mode derivative of SSPR from BLAS.


ARGUMENTS
   If SSPR was called with the arguments

      uplo, n, alpha, x, incx, AP

   then the corresponding call to SSPR_RMD should begin with the arguments

      uplo, n, alpha, x, incx

   with the same values. All these arguments will remain unchanged on exit.
   Note that AP is omitted. In addition the following arguments should be
   provided:

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SSPR operation

   APa
      (input, real packed triangular matrix stored in a vector with
      n*(n+1)/2 elements in the same way as AP)
      The adjoint of AP

NOTE
   A sel parameter is not needed because APa is unchanged.

OPERATIONS
   The same as for SSYR_RMD except that packed storage is used.
```

SSYR2

```
SUBROUTINE SSYR2_RMD(uplo, n, alpha, x, incx, y, incy, lda, xa, ya, Aa, sel)

PURPOSE
   SSYR2_RMD calculates the reverse mode derivative of the SSYR2 routine from BLAS.


ARGUMENTS
   If SSYR2 was called with the arguments:

      uplo, n, alpha, x, incx, y, incy, A, lda

   then the corresponding call to SSYR2_RMD should begin with the arguments

      uplo, n, alpha, x, incx, y, incy, lda

   with the same values. All these arguments will remain unchanged on exit.
   Note that A is omitted. In addition the following arguments should be
   provided:

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SSYR2 call.

   ya
      (input, output, real vector of the same dimension and increment as y)
      ya += the adjoint of y due to the SSYR2 call.

   Aa
      (input, real triangular matrix of the same dimensions as A, and stored
      in the same half according to uplo)
      The adjoint of A.

   sel
      (input, character*2)
      Used to select which adjoints to update:
         sel(1:1) = '1' if xa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if ya should be updated, else sel(2:2) = '0'
      For example, to update only xa, set sel = '10'.

OPERATIONS
   BLAS: (with uplo = 'L')
         A += tril(alpha*x*y' + alpha*y*x'), where A is lower triangluar
         i.e. sym(A) := alpha*(x*y' + y*x') + sym(A)
         (with uplo = 'U')
         A += triu(alpha*x*y' + alpha*y*x'), where A is upper triangluar
         i.e. sym(A') := alpha*(x*y' + y*x') + sym(A')
   RMD:  (with uplo = 'L' or 'U':)
         xa += alpha*(Aa + Aa')*y (eqiv.to: xa += alpha*(diag(Aa)+sym(Aa))*y)
         ya += alpha*(Aa + Aa')*x (eqiv.to: ya += alpha*(diag(Aa)+sym(Aa))*x)
         Aa unchanged
```

SSPR2

```
SUBROUTINE SSPR2_RMD(uplo, n, alpha, x, incx, y, incy, xa, ya, APa, sel)

PURPOSE
   Calculate the reverse mode derivative of SSPR2 from BLAS.


ARGUMENTS
   If SSPR2 was called with the arguments

      uplo, n, alpha, x, incx, y, incy, AP

   then the corresponding call to SSPR2_RMD should begin with the arguments

      uplo, n, alpha, x, incx, y, incy

   with the same values. All these arguments will remain unchanged on exit.
   Note that AP is omitted. In addition the following arguments should be
   provided:

   xa
      (input, output, real vector of the same dimension and increment as x)
      xa += the adjoint of x due to the SSPR2 call.

   ya
      (input, output, real vector of the same dimension and increment as y)
      ya += the adjoint of y due to the SSPR2 call.

   APa
      (input, real packed triangular matrix of the same dimensions as AP,
      and stored in a vector with n*(n+1)/2 elements in the same way as AP)
      The adjoint of AP.

   sel
      (input, character*2)
      Used to select which adjoints to update:
         sel(1:1) = '1' if xa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if ya should be updated, else sel(2:2) = '0'
      For example, to update only xa, set sel = '10'.

OPERATIONS
   The same as for SSYR2_RMD except that packed storage is used
```

4 Derivatives of level 3 BLAS

SGEMM

```
SUBROUTINE SGEMM_RMD(transa, transb, m, n, k, alpha, A, lda, B, ldb, beta, ldc, Aa, Ba, Ca, sel)

PURPOSE
   Calculate the reverse mode derivative of SGEMM from BLAS.

ARGUMENTS
   If SGEMM was called with the arguments

      transa, transb, m, n, k, alpha, A, lda, B, ldb, beta, C, ldc

   then the corresponding call to SGEMM_RMD should begin with the arguments

      transa, transb, m, n, k, alpha, A, lda, B, ldb, beta, ldc

   with the same values. Note that C is omitted. All these arguments will
   remain unchanged on exit. In addition the following arguments should
   be provided:

   Aa   (input, output, real matrix of the same dimensions as A)
        Aa += the adjoint of A due to the SGEMM call.

   Ba   (input, output, real matrix of the same dimensions as B)
        Ba += the adjoint of B due to the SGEMM call.

   Ca   (input, output, real matrix of the same dimensions as C)
        On entry: the adjoint of the C produced by SGEMM
        On exit: the adjoint of the C supplied to SGEMM

   sel  (input, character*3)
        Used to select which adjoints to update/compute:
           sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
           sel(2:2) = '1' if Ba should be updated, else sel(2:2) = '0'
           sel(3:3) = '1' if Ca should be computed, else sel(3:3) = '0'
        For example, to update only Aa, set sel = '100'.

NOTE
   To compute C := alpha*A*A + beta*C one may call sgemm with a repeated
   argument, e.g.
      call sgemm('N', 'N', n, n, n, 1.0, A, n, A, n, 1.0, C, n).
   The correct adjoint of A, Aa = Ca*A' + A'*Ca, can then be computed
   with:
      call sgemm_rmd('N','N',n,n,n,1.0,A,n,A,n,1.0,n,Aa,dummy,dummy,'100')
      call sgemm_rmd('N','N',n,n,n,1.0,A,n,A,n,1.0,n,dummy,Aa,dummy,'010')

OPERATIONS
         SGEMM('N', 'N'...)        SGEMM('T', 'N'...)
   BLAS: C = alpha*A*B + beta*C    C = alpha*A'*B + beta*C
   RMD:  Aa += alpha*Ca*B'         Aa += alpha*B*Ca'
         Ba += alpha*A'*Ca         Ba += alpha*A*Ca
         Ca := beta*Ca             Ca := beta*Ca

         SGEMM('N', 'T'...)        SGEMM('T', 'T'...)
   BLAS: C = alpha*A*B' + beta*C   C = alpha*A'*B' + beta*C
   RMD:  Aa += alpha*Ca*B          Aa += alpha*B'*Ca'
         Ba += alpha*Ca'*A         Ba += alpha*Ca'*A'
         Ca := beta*Ca             Ca := beta*Ca
   The Ca on the right hand sides of the equals signs is its value on entry
```

SSYMM

```
SUBROUTINE SSYMM_RMD(side, uplo, m, n, alpha, A, lda, B, ldb, beta, ldc, Aa, Ba, Ca, sel)

PURPOSE
   Calculate the reverse mode derivative of SSYMM from BLAS.


ARGUMENTS
   If SSYMM was called with the arguments

      side, uplo, m, n, alpha, A, lda, B, ldb, beta, C, ldc.

   then the corresponding call to SSYMM_RMD should begin with the same
   arguments

      side, uplo, m, n, alpha, A, lda, B, ldb, beta, ldc

   with the same values. All these arguments will remain unchanged on exit.
   Note that C is omitted. In addition the following arguments should be
   provided:

   Aa
      (input, output, real triangular matrix of the same dimensions as A, and
      stored in the same half according to uplo)
      Aa += the adjoint of A due to the SSYMM call.

   Ba
      (input, output, real matrix of the same dimensions as B)
      Ba += the adjoint of B due to the SSYMM call.

   Ca
      (input, output, real matrix of the same dimensions as C)
      On entry: the adjoint of the C produced by SSYMM
      On exit: the adjoint of the C supplied to SSYMM

   sel
      (input, character*3)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if Ba should be updated, else sel(2:2) = '0'
         sel(3:3) = '1' if Ca should be computed, else sel(3:3) = '0'
      For example, to update only Aa, set sel = '100'.

OPERATIONS
   (for SSYMM('L', 'L...'))
   BLAS: C = alpha*sym(A)*B + beta*C, where A is a lower triangular matrix
   RMD:  Aa += alpha*(tril(B*Ca'+Ca*B') - diag(B*Ca'))
         Ba += alpha*sym(A)*Ca
         Ca := beta*Ca

   (for SSYMM('R', 'L...'))
   BLAS: C = alpha*B*sym(A) + beta*C, where A is a lower triangular matrix
   RMD:  Aa += alpha*(tril(B'*Ca+Ca'*B) - diag(B'*Ca))
         Ba += alpha*sym(A)*Ca
         Ca := beta*Ca
```

SSYRK

```
SUBROUTINE SSYRK_RMD(uplo, trans, n, k, alpha, A, lda, beta, ldc, Aa, Ca, sel)

PURPOSE
   Calculate the reverse mode derivative of SSYRK from BLAS.

ARGUMENTS
   If SSYRK was called with the arguments

      uplo, trans, n, k, alpha, A, lda, beta, C, ldc

   then the corresponding call to SSYRK_RMD should begin with the arguments

      uplo, trans, n, k, alpha, A, lda, beta, ldc

   with the same values. All these arguments will remain unchanged on exit.
   Note that C is omitted. In addition the following arguments should be
   provided:

   Aa
      (input, output, real matrix of the same dimensions as A)
      Aa += the adjoint of A due to the SSYRK call

   Ca
      (input, output, real triangular matrix of the same dimensions as C,
      and stored in the same half according to uplo)
      On entry: the adjoint of the C produced by SSYRK
      On exit: the adjoint of the C supplied to SSYRK

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if Ca should be computed, else sel(2:2) = '0'
      For example, to update only Aa, set sel = '10'.

OPERATIONS
   (for SSYRK('L', 'N'...); C lower triangular)
   BLAS: C := alpha*tril(A*A') + beta*C, i.e. sym(C) := alpha*A*A' + beta*sym(C)
   RMD:  Aa += alpha*(Ca + Ca')*A, where Ca is value on entry
         Ca := beta*Ca
```

SSYR2K

```
SUBROUTINE SSYR2K_RMD(uplo, trans, n, k, alpha, A, lda, B, ldb, beta, ldc, Aa, Ba, Ca, sel)

PURPOSE
   Calculate the reverse mode derivative of SSYR2K from BLAS.

ARGUMENTS
   If SSYR2K was called with the arguments

      uplo, trans, n, k, alpha, A, lda, B, ldb, beta, C, ldc

   then the corresponding call to SSYR2K_RMD should begin with the arguments

      uplo, trans, n, k, alpha, A, lda, B, ldb, beta, ldc

   with the same values. All these arguments will remain unchanged on exit.
   Note that C is omitted. In addition the following arguments should be
   provided:

   Aa
      (input, output, real matrix of the same dimensions as A)
      Aa += the adjoint of A due to the SSYR2K call

   Ba
      (input, output, real matrix of the same dimensions as A)
      Ba += the adjoint of B due to the SSYR2K call

   Ca
      (input, output, real triangular matrix of the same dimensions as C, and
      stored in the same half according to uplo)
      On entry: the adjoint of the C produced by SSYR2K
      On exit: the adjoint of the C supplied to SSYR2K

   sel
      (input, character*3)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if Ba should be updated, else sel(2:2) = '0'
         sel(3:3) = '1' if Ca should be computed, else sel(3:3) = '0'
      For example, to update only Aa, set sel = '100'.

OPERATIONS
   (for SSYR2K('L', 'N'...); C lower triangular)
   BLAS: C := alpha*tril(A*B' + B*A') + beta*C
   RMD:  Aa += alpha*(Ca + Ca')*B (equiv.to: Aa += alpha*(sym(Ca) + diag(Ca))*B)
         Ba += alpha*(Ca + Ca')*A (equiv.to: Ba += alpha*(sym(Ca) + diag(Ca))*A)
         Ca := beta*Ca
```

STRMM

```
SUBROUTINE STRMM_RMD(side, uplo, transa, diag, m, n, alpha, A, lda, B0, ldb, Aa, Ba, sel)

PURPOSE
   Calculate the reverse mode derivative of STRMM from BLAS.


ARGUMENTS
   If STRMM was called with the arguments

      side, uplo, transa, diag, m, n, alpha, A, lda, B, ldb

   then the corresponding call to STRMM_RMD should begin with the arguments

      side, uplo, transa, diag, m, n, alpha, A, lda, B0, ldb

   which all except B0 should have the same values as they had on the STRMM
   call, and B0 should have the value that B had on entry to the STRMM-call
   (STRMM only changes the B-argument). All these arguments will remain
   unchanged on exit. In addition the following arguments should be
   provided:

   Aa
      (input, output, real matrix of the same dimensions as A) The
      Aa := the adjoint of A due to the STRMM call

   Ba
      (input, output, real vector of the same dimension and increment as x)
      On entry: the adjoint of the B produced by STRMM
      On exit: the adjoint of the B supplied to STRMM

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if Ba should be computed, else sel(2:2) = '0'
      For example, to update only Aa, set sel = '10'.

OPERATIONS
   (for STRMM('L', 'L', 'N'...); A lower triangular)
   BLAS: B := alpha*A*B           (*)
   RMD:  Ba := alpha*A'*Ba        (**)
         Aa += alpha*tril(Ba*B')  where B and Ba are inputs to (*) and (**)
```

STRSM

```
SUBROUTINE STRSM_RMD(side, uplo, transa, diag, m, n, alpha, A, lda, B, ldb, Aa, Ba, wrk, sel)

PURPOSE
   Calculate the reverse mode derivative of STRSM from BLAS.


ARGUMENTS
   If STRSM was called with the arguments

      side, uplo, transa, diag, m, n, alpha, A, lda, B, ldb

   then the corresponding call to STRSM_RMD should begin with the same
   arguments, containing the values which they had on exit from STRSM. These
   arguments will remain unchanged on exit from STRSM_RMD. In addition the
   following arguments should be provided:

   Aa
      (input, output, real matrix of the same dimensions as A)
      Aa += adjoint of A due to the STRSM call

   Ba
      (input, output, real matrix of the same dimensions as B)
      On entry: The adjoint of the B produced by STRSM
      On exit: The adjoint of the B supplied to STRSM

   wrk
      (output, real vector of dimension at least max(m,n))
      When sel(2:2) = '0' so that a new Ba should not be computed it is
      necessary to supply STRSV_RMD with a workspace vector. When sel(2:2) =
      '1', wrk is not referenced, because Ba serves its purpose. In this case
      a dummy value may be given instead

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if Aa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if Ba should be computed, else sel(2:2) = '0'
      For example, to update only Aa, set sel = '10'.

OPERATIONS
   (for STRMM('L', 'L', 'N'...); A lower triangular)
   BLAS: B := inv(A)*B      (*)
   RMD:  Ba := inv(A)'*Ba   (**)
         Aa -= tril(Ba*B')  where B and Ba are outputs from (*) and (**)
```

5 Adjoints of scalars

SSCAL-SCALARS

```
SUBROUTINE SSCAL_RMDS(n, x0, incx, alphaa, xa)

PURPOSE
   Calculate the adjoint of alpha for SSCAL from BLAS.


ARGUMENTS
   If SSCAL was called with the arguments

      n, alpha, x, incx

   then SSCAL_RMDS should be called with the arguments

      n, x0, incx

   which all except x0 should have the same values as they had on the SSCAL-
   call, and x0 should have the value that x had on entry to the SSCAL-call
   (SSCAL only changes x). All these arguments will remain unchanged on
   exit. Note that alpha is omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += adjoint of alpha due to the SSCAL-call

   xa
      (input, real vector if the same dimension and increment as x)
      The adjoint of the x produced by SSCAL

OPERATIONS
   BLAS: x := alpha*x0
   RMD:  alphaa += xa'*x0
```

SAXPY-SCALARS

```
SUBROUTINE SAXPY_RMDS(n, x, incx, incy, alphaa, ya)

PURPOSE
   Calculates the adjoint of alpha for SAXPY from BLAS.

ARGUMENTS
   If SAXPY was called with the arguments

      n, alpha, x, incx, y, incy

   then the corresponding call to SAXPY_RMDS should begin with the arguments

      n, x, incx, incy

   with the same values. These arguments will remain unchanged on exit. Note
   that alpha and y are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SAXPY call.

   ya
      (input, real vector of the same dimension and increment as y)
      The adjoint of the y produced by SAXPY.

OPERATIONS
   BLAS: y := alpha*x + y
   RMD:  alphaa += ya'*x
```

SGEMV-SCALARS

```
SUBROUTINE SGEMV_RMDS(trans, m, n, A, lda, x, incx, y0, incy, alphaa, betaa, ya, sel)

PURPOSE
   Calculate the adjoint of alpha and/or beta for SGEMV from BLAS.

ARGUMENTS
   If SGEMV was called with the arguments

      trans, m, n, alpha, A, lda, x, incx, beta, y, incy

   then the corresponding call to SGEMV_RMDS should begin with the arguments

      trans, m, n, A, lda, x, incx, y0, incy

   which all except y0 should have the same values as they had on the SGEMV-
   call, and y0 should have the value that y had on entry to the SGEMV-call
   (SGEMV only changes the y-argument). All these arguments except y0 will
   remain unchanged on exit, but y0 is used as workspace by SGEMV_RMDS. Note
   that alpha and beta are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SGEMV-call.

   betaa
      (input, output, real scalar)
      betaa += the adjoint of beta due to the SGEMV-call.

   ya
      (input, real vector of the same dimension and increment as y)
      The adjoint of the y produced by SGEMV

   sel
      (input, character*2)
      Used to select which adjoints to update:
         sel(1:1) = '1' if alphaa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if betaa should be updated, else sel(2:2) = '0'
      For example, to update only alphaa, set sel = '10'.

NOTE
   ya must not have been updated when SGEMV_RMDS is called and therefore a
   potential call to SGEMV_RMD must come after a corresponding call to
   SGEMV_RMDS.

OPERATIONS
   (when trans = 'N')
   BLAS: y = alpha*A*x + beta*y0
   RMD:  alphaa += ya'*A*x
         betaa += ya'*y0
   (when trans = 'T')
   BLAS: y = alpha*A'*x + beta*y0
   RMD:  alphaa += ya'*A'*x
         betaa += ya'*y0
```

SGBMV-SCALARS

```
SUBROUTINE SGBMV_RMDS(trans, m, n, kl, ku, A, lda, x, incx, y0, incy, alphaa, betaa, ya, sel)

PURPOSE
   Calculate the adjoint of alpha and/or beta for SGBMV from BLAS.


ARGUMENTS
   If SGBMV was called with the arguments

      trans, m, n, kl, ku, alpha, A, lda, x, incx, beta, y, incy

   then SGBMV_RMD should be called with the arguments:

      trans, m, n, kl, ku, A, lda, x, incx, y0, incy

   which all except y0 should have the same values as they had on the SGBMV-
   call, and y0 should have the value that y had on entry to the SGBMV-call
   (SGBMV only changes the y-argument). All these arguments except y0 will
   remain unchanged on exit, but y0 is used as workspace by SGBMV_RMDS. Note
   that alpha and beta are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SGBMV call.

   betaa
      (input, output, real scalar)
      betaa += the adjoint of beta due to the SGBMV call.

   ya
      (input, real vector of the same dimension and increment as y)
      The adjoint of the y produced by SGBMV

   sel
      (input, character*2)
      Used to select which adjoints to update:
         sel(1:1) = '1' if alphaa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if betaa should be updated, else sel(2:2) = '0'
      For example, to update only alphaa, set sel = '10'.

NOTE
   ya must not have been updated when SGBMV_RMDS is called and therefore a
   potential call to SGBMV_RMD must come after a corresponding call to
   SGBMV_RMDS.

OPERATIONS
   (when trans = 'N')
   BLAS: y = alpha*A*x + beta*y0
   RMD:  alphaa += ya'*A*x
         betaa += ya'*y
   (when trans = 'T')
   BLAS: y = alpha*A'*x + beta*y
   RMD:  alphaa += ya'*A'*x
         betaa += ya'*y0
```

SSYMV-SCALARS

```
SUBROUTINE SSYMV_RMDS(uplo, n, A, lda, x, incx, y0, incy, alphaa, betaa, ya, sel)

PURPOSE
   Calculate the adjoint of alpha and/or beta for SSYMV from BLAS.


ARGUMENTS
   If SSYMV was called with the arguments

      uplo, n, alpha, A, lda, x, incx, beta, y, incy

   then the corresponding call to SSYMV_RMDS should begin with the same
   arguments

      uplo, n, A, lda, x, incx, y0, incy

   which all except y0 should have the same values as they had on the SSYMV-
   call, and y0 should have the value that C had on entry to the SSYMV-call
   (SSYMV only changes the C-argument). All these arguments except C0 will
   remain unchanged on exit, but y0 is used as workspace by SSYMV_RMDS. Note
   that alpha and beta are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SSYMV call.

   betaa
      (input, output, real scalar)
      betaa += the adjoint of beta due to the SSYMV call.

   ya
      (input, real vector of the same dimension and increment as y)
      The adjoint of the y produced by SSYMV

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if alphaa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if betaa should be updated, else sel(2:2) = '0'
      For example, to update only betaa, set sel = '01'.

NOTE
   ya must not have been updated when SSYMV_RMDS is called and therefore a
   potential call to SSYMV_RMD must come after a corresponding call to
   SSYMV_RMDS.

OPERATIONS
   (with uplo = 'L')
   BLAS: y = alpha*sym(A)*x + beta*y0    A lower triangular
   RMD:  alphaa += ya'*sym(A)*x
         betaa += ya'*y0
   (with uplo = 'U')
   BLAS: y = alpha*sym(A')*x + beta*y0   A upper triangular
   RMD:  alphaa += ya'*sym(A')*x
         betaa += ya'*y0
```

SSBMV-SCALARS

```
SUBROUTINE SSBMV_RMDS(uplo, n, k, A, lda, x, incx, y0, incy, alphaa, betaa, ya, sel)

PURPOSE
   Calculate the adjoint of alpha and/or beta for SSBMV from BLAS.


ARGUMENTS
   If SSBMV was called with the arguments

      uplo, n, k, alpha, A, lda, x, incx, beta, y, incy

   then the corresponding call to SSBMV_RMDS should begin with the same
   arguments

      uplo, n, k, A, lda, x, incx, y, incy

   which all except y0 should have the same values as they had on the SSBMV-
   call, and y0 should have the value that C had on entry to the SSBMV-call
   (SSBMV only changes the C-argument). All these arguments except C0 will
   remain unchanged on exit, but y0 is used as workspace by SSBMV_RMDS. Note
   that alpha and beta are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SGEMV call.

   betaa
      (input, output, real scalar)
      betaa += the adjoint of beta due to the SGEMV call.

   ya
      (input, real vector of the same dimension and increment as y)
      The adjoint of the y produced by SSBMV

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if alphaa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if betaa should be updated, else sel(2:2) = '0'
      For example, to update only betaa, set sel = '01'.

NOTE
   ya must not have been updated when SSBMV_RMDS is called and therefore a
   potential call to SSBMV_RMD must come after a corresponding call to
   SSBMV_RMDS.

OPERATIONS
   (with uplo = 'L')
   BLAS: y = alpha*sym(A)*x + beta*y0    A lower triangular band
   RMD:  alphaa += ya'*sym(A)*x
         betaa += ya'*y0
   (with uplo = 'U')
   BLAS: y = alpha*sym(A')*x + beta*y0   A upper triangular band
   RMD:  alphaa += ya'*sym(A')*x
         betaa += ya'*y0
```

SSPMV-SCALARS

```
SUBROUTINE SSPMV_RMDS(uplo, n, AP, x, incx, y0, incy, alphaa, betaa, ya, sel)

PURPOSE
   Calculate the adjoint of alpha and/or beta for SSPMV from BLAS.


ARGUMENTS
   If SSPMV was called with the arguments

      uplo, n, alpha, AP, x, incx, beta, y, incy

   then the corresponding call to SSPMV_RMDS should begin with the same
   arguments

      uplo, n, AP, x, incx, y, incy

   which all except y0 should have the same values as they had on the SSPMV-
   call, and y0 should have the value that C had on entry to the SSPMV-call
   (SSPMV only changes the C-argument). All these arguments except C0 will
   remain unchanged on exit, but y0 is used as workspace by SSPMV_RMDS. Note
   that alpha and beta are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SGEMV call.

   betaa
      (input, output, real scalar)
      betaa += the adjoint of beta due to the SGEMV call.

   ya
      (input, real vector of the same dimension and increment as y)
      The adjoint of the y produced by SSPMV

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if alphaa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if betaa should be updated, else sel(2:2) = '0'
      For example, to update only betaa, set sel = '01'.

NOTE
   ya must not have been updated when SSPMV_RMDS is called and therefore a
   potential call to SSPMV_RMD must come after a corresponding call to
   SSPMV_RMDS.

OPERATIONS
   (with uplo = 'L')
   BLAS: y = alpha*sym(AP)*x + beta*y0    AP packed lower triangular
   RMD:  alphaa += ya'*sym(AP)*x
         betaa += ya'*y0
   (with uplo = 'U')
   BLAS: y = alpha*sym(AP')*x + beta*y0   AP packed upper triangular
   RMD:  alphaa += ya'*sym(AP')*x
         betaa += ya'*y0
```

SGER-SCALARS

```
SUBROUTINE SGER_RMDS(m, n, x, incx, y, incy, lda, alphaa, Aa)

PURPOSE
   Calculate the adjoint of alpha for SGER from BLAS.


ARGUMENTS
   If SGER was called with the arguments

      m, n, alpha, x, incx, y, incy, A, lda

   then the corresponding call to SGER_RMD should begin with the arguments:

      m, n, x, incx, y, incy, lda

   with the same values. All these arguments will remain unchanged on exit.
   Note that alpha and A are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += adjoint of alpha due to the SGER-call

   Aa
      (input, real matrix of the same dimensions as A)
      The adjoint of A.

OPERATIONS
   BLAS: A += alpha*x*y'
   RMD:  alphaa += x'*Aa*y
```

SSYR-SCALARS

```
SUBROUTINE SSYR_RMDS(uplo, n, x, incx, lda, alphaa, Aa)

PURPOSE
   Calculate the adjoint of alpha for SSYR from BLAS


ARGUMENTS
   If SSYR was called with the arguments

      uplo, n, alpha, x, incx, A, lda

   then the corresponding call to SSYR_RMD should begin with the arguments

      uplo, n, x, incx, lda

   with the same values. All these arguments will remain unchanged on exit.
   Note that alpha and A are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SSYR call.

   Aa
      (input, real triangular matrix of the same dimensions as A, and stored
      in the same half according to uplo)
      The adjoint of A.

OPERATIONS
   (with uplo = 'L')
   BLAS: A += alpha*tril(x*x') i.e. sym(A) := alpha*x*x' + sym(A)
   RMD:  alphaa += x'*Aa*x
   (with uplo = 'U')
   BLAS: A += alpha*triu(x*x') i.e. sym(A') := alpha*x*x' + sym(A')
   RMD:  alphaa += x'*Aa*x
```

SSPR-SCALARS

```
SUBROUTINE SSPR_RMDS(uplo, n, x, incx, alphaa, APa)

PURPOSE
   Calculate the adjoint of alpha for SSPR from BLAS


ARGUMENTS
   If SSPR was called with the arguments

      uplo, n, alpha, x, incx, AP

   then the corresponding call to SSPR_RMD should begin with the arguments

      uplo, n, x, incx

   with the same values. All these arguments will remain unchanged on exit.
   Note that alpha and AP are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SSPR call.

   APa
      (input, real packed triangular matrix stored in a vector with
      n*(n+1)/2 elements in the same way as AP)
      The adjoint of AP

OPERATIONS
   (with uplo = 'L')
   BLAS: AP += alpha*tril(x*x') i.e. sym(AP) := alpha*x*x' + sym(AP)
   RMD:  alphaa += x'*APa*x
   (with uplo = 'U')
   BLAS: AP += alpha*triu(x*x') i.e. sym(AP') := alpha*x*x' + sym(AP')
   RMD:  alphaa += x'*APa*x
```

SSYR2-SCALARS

```
SUBROUTINE SSYR2_RMDS(uplo, n, x, incx, y, incy, lda, alphaa, Aa)

PURPOSE
   Calculate the adjoint of alpha for SSYR2 from BLAS


ARGUMENTS
   If SSYR2 was called with the arguments:

      uplo, n, alpha, x, incx, y, incy, A, lda

   then the corresponding call to SSYR2_RMD should begin with the arguments

      uplo, n, x, incx, y, incy, lda

   with the same values. All these arguments will remain unchanged on exit.
   Note that alpha and A are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SSYR2 call.

   Aa
      (input, real triangular matrix of the same dimensions as A, and stored
      in the same half according to uplo)
      The adjoint of A.

OPERATIONS
   BLAS: (with uplo = 'L')
         A += alpha*tril(x*y' + y*x'), where A is lower triangluar
         i.e. sym(A) := alpha*(x*y' + y*x') + sym(A)
         (with uplo = 'U')
         A += alpha*triu(x*y' + y*x'), where A is upper triangluar
         i.e. sym(A') := alpha*(x*y' + y*x') + sym(A')
   RMD:  (with uplo = 'L' or 'U':)
         alphaa += x'*Aa*y + y'*Aa*x
```

SSPR2-SCALARS

```
SUBROUTINE SSPR2_RMDS(uplo, n, x, incx, y, incy, alphaa, APa)

PURPOSE
   Calculate the adjoint of alpha for SSPR2 from BLAS


ARGUMENTS
   If SSPR2 was called with the arguments:

      uplo, n, alpha, x, incx, y, incy, AP

   then the corresponding call to SSPR2_RMD should begin with the arguments

      uplo, n, x, incx, y, incy

   with the same values. All these arguments will remain unchanged on exit.
   Note that alpha and AP are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SSPR2 call.

   APa
      (input, real packed triangular matrix of the same dimensions as AP,
      and stored in the same half according to uplo)
      The adjoint of AP.

OPERATIONS
   BLAS: (with uplo = 'L')
         AP += tril(alpha*x*y' + alpha*y*x'), where AP is lower triangluar packed
         i.e. sym(AP) := alpha*(x*y' + y*x') + sym(AP)
         (with uplo = 'U')
         AP += triu(alpha*x*y' + alpha*y*x'), where AP is upper triangluar packed
         i.e. sym(AP') := alpha*(x*y' + y*x') + sym(AP')
   RMD:  (with uplo = 'L' or 'U':)
         alphaa += x'*APa*y + y'*APa*x
```

SGEMM-SCALARS

```
SUBROUTINE SGEMM_RMDS(transa, transb, m, n, k, A, lda, B, ldb, C0, ldc, alphaa, betaa, Ca, sel)

PURPOSE
   Calculate the adjoint of alpha and/or beta for SGEMM from BLAS.

ARGUMENTS
   If SGEMM was called with the arguments

      transa, transb, m, n, k, alpha, A, lda, B, ldb, beta, C, ldc

   then the corresponding call to SGEMM_RMDS should begin with the arguments

      trans, transb, m, n, k, A, lda, B, ldb, C0, ldc

   which all except C0 should have the same values as they had on the SGEMM-
   call, and C0 should have the value that C had on entry to the SGEMM-call
   (SGEMM only changes the C-argument). All these arguments except C0 will
   remain unchanged on exit, but C0 is used as workspace by SGEMM_RMDS. Note
   that alpha and beta are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SGEMM call.

   betaa
      (input, output, real scalar)
      betaa += the adjoint of beta due to the SGEMM call.

   Ca   (input, real matrix of the same dimensions as C)
        The adjoint of the C produced by SGEMM

   sel
      (input, character*2)
      Used to select which adjoints to update:
         sel(1:1) = '1' if alphaa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if betaa should be updated, else sel(2:2) = '0'
      For example, to update only alphaa, set sel = '10'.

NOTE
   Ca must not have been updated when SGEMM_RMDS is called and therefore a
   potential call to SGEMM_RMD must come after a corresponding call to
   SGEMM_RMDS.

OPERATIONS
         SGEMM('N', 'N'...)             SGEMM('T', 'N'...)
   BLAS: C = alpha*A*B + beta*C         C = alpha*A'*B + beta*C
   RMD:  alphaa += vec(Ca)'*vec(A*B)    alphaa += vec(Ca')*vec(A'*B)
         betaa += vec(Ca)'*vec(C0)      betaa += vec(Ca)'*vec(C0)

         SGEMM('N', 'T'...)             SGEMM('T', 'T'...)
   BLAS: C = alpha*A*B' + beta*C        C = alpha*A'*B' + beta*C
   RMD:  alphaa += vec(Ca)'*vec(A*B')   alphaa += vec(Ca')*vec(A'*B')
         betaa += vec(Ca)'*vec(C0)      betaa += vec(Ca)'*vec(C0)
```

SSYMM-SCALARS

```
SUBROUTINE SSYMM_RMDS(side, uplo, m, n, A, lda, B, ldb, C0, ldc, alphaa, betaa, Ca, sel)

PURPOSE
   Calculate the adjoint of alpha and/or beta for SSYMM from BLAS.


ARGUMENTS
   If SSYMM was called with the arguments

      side, uplo, m, n, alpha, A, lda, B, ldb, beta, C, ldc.

   then the corresponding call to SSYMM_RMD should begin with the same
   arguments

      side, uplo, m, n, A, lda, B, ldb, C0, ldc

   which all except C0 should have the same values as they had on the SSYMM-
   call, and C0 should have the value that C had on entry to the SSYMM-call
   (SSYMM only changes the C-argument). All these arguments except C0 will
   remain unchanged on exit, but C0 is used as workspace by SSYMM_RMDS. Note
   that alpha and beta are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SGEMM call.

   betaa
      (input, output, real scalar)
      betaa += the adjoint of beta due to the SGEMM call.

   Ca   (input, real matrix of the same dimensions as C)
        The adjoint of the C produced by SSYMM

   sel
      (input, character*2)
      Used to select which adjoints to update:
         sel(1:1) = '1' if alphaa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if betaa should be updated, else sel(2:2) = '0'
      For example, to update only alphaa, set sel = '10'.

OPERATIONS
   (for SSYMM('L', 'L...'))
   BLAS: C = alpha*sym(A)*B + beta*C, where A is a lower triangular matrix
   RMD:  alphaa += vech(Ca)'*vech(sym(A)*B)
         betaa += vech(Ca)'*vech(C0)

   (for SSYMM('R', 'L...'))
   BLAS: C = alpha*B*sym(A) + beta*C, where A is a lower triangular matrix
   RMD:  alphaa += vech(Ca)'*vech(B*sym(A))
         betaa += vech(Ca)'*vech(C0)
```

SSYRK-SCALARS

```
SUBROUTINE SSYRK_RMDS(uplo, trans, n, k, A, lda, C0, ldc, alphaa, betaa, Ca, sel)

PURPOSE
   Calculate the adjoint of alpha and/or beta for SSYRK from BLAS

ARGUMENTS
   If SSYRK was called with the arguments

      uplo, trans, n, k, alpha, A, lda, beta, C, ldc

   then the corresponding call to SSYRK_RMD should begin with the arguments

      uplo, trans, n, k, A, lda, C0, ldc

   which all except C0 should have the same values as they had on the SSYRK-
   call, and C0 should have the value that C had on entry to the SSYRK-call
   (SSYRK only changes the C-argument). All these arguments except C0 will
   remain unchanged on exit, but C0 is used as workspace by SSYRK_RMDS. Note
   that alpha and beta are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SSYRK call.

   betaa
      (input, output, real scalar)
      betaa += the adjoint of beta due to the SSYRK call.

   Ca
      (input, real triangular matrix of the same dimensions as C, and stored
      in the same half according to uplo)
      The adjoint of the C produced by SSYRK

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if alphaa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if betaa should be updated, else sel(2:2) = '0'
      For example, to update only betaa, set sel = '01'.

OPERATIONS
   (for SSYRK('L', 'N'...); C n by n lower triangular)
   BLAS: C := alpha*tril(A*A') + beta*C
   RMD:  alphaa += vech(Ca)'*vech(A*A')
         betaa += vech(Ca)'*vech(C)
```

SSYR2K-SCALARS

```
SUBROUTINE SSYR2K_RMDS(uplo, trans, n, k, A, lda, B, ldb, C0, ldc, alphaa, betaa, Ca, sel)

PURPOSE
   Calculate the adjoint of alpha and/or beta for SSYR2K from BLAS

ARGUMENTS
   If SSYR2K was called with the arguments

      uplo, trans, n, k, alpha, A, lda, beta, C, ldc

   then the corresponding call to SSYR2K_RMD should begin with the arguments

      uplo, trans, n, k, A, lda, C0, ldc

   which all except C0 should have the same values as they had on the SSYR2K-
   call, and C0 should have the value that C had on entry to the SSYR2K-call
   (SSYR2K only changes the C-argument). All these arguments except C0 will
   remain unchanged on exit, but C0 is used as workspace by SSYR2K_RMDS. Note
   that alpha and beta are omitted. In addition the following arguments
   should be provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SSYR2K call.

   betaa
      (input, output, real scalar)
      betaa += the adjoint of beta due to the SSYR2K call.

   Ca
      (input, real triangular matrix of the same dimensions as C, and stored
      in the same half according to uplo)
      The adjoint of the C produced by SSYR2K

   sel
      (input, character*2)
      Used to select which adjoints to update/compute:
         sel(1:1) = '1' if alphaa should be updated, else sel(1:1) = '0'
         sel(2:2) = '1' if betaa should be updated, else sel(2:2) = '0'
      For example, to update only betaa, set sel = '01'.

OPERATIONS
   (for SSYR2K('L', 'N'...); C n by n lower triangular)
   BLAS: C := alpha*tril(A*B' + B*A') + beta*C
   RMD:  alphaa += vech(Ca)'*vech(A*B' + B*A')
         betaa += vech(Ca)'*vech(C)
```

STRMM-SCALARS

```
SUBROUTINE STRMM_RMDS(side, uplo, transa, diag, m, n, A, lda, B0, ldb, alphaa, Ba, wrk)

PURPOSE
   Calculate the adjoint of alpha for STRMM from BLAS.


ARGUMENTS
   If STRMM was called with the arguments

      side, uplo, transa, diag, m, n, alpha, A, lda, B, ldb

   then the corresponding call to STRMM_RMDS should begin with the arguments

      side, uplo, transa, diag, m, n, A, lda, B0, ldb

   which all except B0 should have the same values as they had on the STRMM
   call, and B0 should have the value that B had on entry to the STRMM-call
   (STRMM only changes the B-argument). All these arguments except B0 will
   remain unchanged on exit, but B0 is used as workspace by STRMM_RMDS. Note
   that alpha is omitted. In addition the following arguments should be
   provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SGEMM call.

   Ba
      (input, real matrix of the same dimensions as B)
      The adjoint of the B produced by SGEMM

   wrk
      (output, real vector of dimension max(m,n))
      Workspace

OPERATIONS
   (for STRMM('L', 'L', 'N'...); A lower triangular)
   BLAS: B := alpha*A*B0
   RMD:  alphaa += vec(Ba)'*vec(A*B0)
```

STRSM-SCALARS

```
SUBROUTINE STRSM_RMDS(side, uplo, transa, diag, m, n, A, lda, B0, ldb, alphaa, Ba)

PURPOSE
   Calculate the adjoint of alpha for STRSM from BLAS.


ARGUMENTS
   If STRSM was called with the arguments

      side, uplo, transa, diag, m, n, alpha, A, lda, B, ldb

   then the corresponding call to STRSM_RMDS should begin with the arguments

      side, uplo, transa, diag, m, n, A, lda, B0, ldb

   which all except B0 should have the same values as they had on the STRSM
   call, and B0 should have the value that B had on entry to the STRSM-call
   (STRSM only changes the B-argument). All these arguments except B0 will
   remain unchanged on exit, but B0 is used as workspace by STRSM_RMDS. Note
   that alpha is omitted. In addition the following arguments should be
   provided:

   alphaa
      (input, output, real scalar)
      alphaa += the adjoint of alpha due to the SGEMM call.

   Ba   (input, real matrix of the same dimensions as B)
        The adjoint of the B produced by SGEMM

OPERATIONS
   (for STRSM('L', 'L', 'N'...); A lower triangular)
   BLAS: B := alpha*inv(A)*B0
   RMD:  alphaa += vec(Ba)'*vec(inv(A)*B0)
```

6 Derivatives of other subroutine(s)

SPOTRF

```
SUBROUTINE SPOTRF_RMD(uplo, n, A, lda, Aa)

PURPOSE
   Calculate the reverse mode derivative of the Lapack Cholesky factorization
   subroutine SPOTRF.

ARGUMENTS
   If SPOTRF was called with the arguments

      uplo, n, A, lda, info

   and finished successfully, then the corresponding call to SPOTRF_RMD
   should begin with the arguments:

      uplo, n, A, lda

   with the values which they had on exit from SPOTRF. In particular A
   should contain the Cholesky factor of the original matrix. All these
   arguments will remain unchanged on exit from SPOTRF_RMD. In addition the
   following argument should be provided:

   Aa
      (input, output, real triangular matrix of the same dimensions as A, and
      stored in the same half according to uplo)
      On entry: The adjoint of the Cholesky factor L
      On exit: The adjoint of the original matrix A due to the SPOTRF call.

OPERATIONS
   BLAS: A := solution to L*L' = sym(A) (i.e. A := Cholesky factor of sym(A))
   RMD:  Aa := adjoint of A due to the BLAS operation

NOTES
   1) The call to SPOTRF must have returned with info = 0
   2) Observe that Aa is assigned to and not added to
   3) On entry to SPTORF A is in the upper or the lower triangle of the
      parameter A and on exit the Cholesky factor L is in the same triangle.
      On entry to SPOTRF_RMD the parameters are:
         A:  Cholesky factor, L
         Aa: the adjoint of L
      and on exit:
         A:  unchanged
         Aa: the adjoint of A

ALGORITHM
   The algorithm below is obtained by finding, line by line, the adjoint
   of the "recursive" version of Cholesky factorization, which can be
   derived as follows. Consider the block matrix equalities:
     LL' =  | d   0  | * | d  l1' | = | d^2   d*l1'          | = | a11  a' |
            | l1  L1 |   | 0  L1' |   | l1*d  L1*L1'+l1*l1' |   | a    A1 |

   From these one obtains the following formulae for d, l1 and L1:
     d = sqrt(a11)
     l1 = a/d
     B1 = A1 - l1*l1'
     L1 = chol(B1)

   the last one of which can be applied recursively until B1 is empty.
```

7 References

[1] Oak Ridge National Laboratory, Numerical Algorithms Group Ltd., *Basic Linear Algebra Subprograms – A Quick Reference Guide*, 1997, available at http://www.netlib.org/blas/blasqr.pdf

[2] Kristjan Jonasson, Sven Sigurdsson, Hordur Freyr Yngvason, Petur Orri Ragnarsson, Pall Melsted, *Algorithm xxx: Fortran subroutines for reverse mode algorithmic differentiation of BLAS matrix operations*, ACM Transactions on Mathematical Software (TOMS), xx, xx, 2020.