Generalized-norm conjugate direction solver

Mohammad Maysami and Nader Moussa

ABSTRACT

In optimization problems, the L_1 norm outperforms the L_2 norm in presence of noise and when a blocky or sparse solution is appropriate. These applications call for a solver that can redefine the optimum criteria for a particular problem. We have implemented a generalized norm solver that is useful for a wide range of problems. Our solver modularizes the norm function so that it can easily be interchanged to experiment with different schemes on any particular geophysical problem. We implement L_1, L_2, and two additional norms: Huber and Hybrid L_1/L_2. These are useful for problems that seek the benefits of both the L_1 and L_2 norms.

INTRODUCTION

Strict L_1 norm optimization has numerous applications in geophysical inversion problems. Examples include solving for sparse functions, such as those that describe blocky, layered geology. However, we find that approximations to L_1, or modified L_1/L_2 systems, are more computationally feasible than pure L_1 solutions. These non-strict L_1/L_2 norms – the Huber (Huber, 1973; Guitton, 2000) and Hybrid norms – still satisfy our desire to find sparse-functions, and are suitable for most of our geophysical needs.

For many years, SEP has relied on a single incarnation of the conjugate-direction descent solver. This code is based on work by Claerbout (2008). For some time in the 1990s, SEP also used a competing least squares approach, *LSQR*, developed by Paige and Saunders (1982). Because numerical optimization is such an embedded part of many geophysical algorithms, it is extremely desirable to have a backward-compatible program framework that can continue to work within these decades-old codes. For this reason, we chose to implement our solver with the same interface as the conjugate-direction L_2 solver (Claerbout, 2008). Our new solver implements a generalized definition of the norm used for minimization.

It should be noted that the term "norm" is used for convenience; whereas in fact, some of the numeric measures under consideration do not strictly satisfy the necessary mathematical criteria to be a proper norm. Notably, the Huber and Hybrid norms exhibit non-linearity with regard to a scalar multiplication (Bube and Langan, 1997). We will use this terminology for simplicity, but the cautious reader should note that these functions do not satisfy all necessary properties of a norm.

We considered many solver options in our original quest for a numeric solution to the strict L_1 optimization problem. We investigated a weighted-median algorithm to descend to the L_1-sense minimum. Early development showed that, despite the theoretical promise of the weighted-median methodology, it did not suitably achieve the desired solution on non-trivial test problems. This work led to development of a new solver, discussed below, based on theory developed in Claerbout (2009). This solver is based on a generalized plane-search using Taylor series expansion of the norm.

In this paper, we will describe our implementation of this solver framework. It is based on SEP's conjugate-direction L_2 solver, but it also allows us to substitute different norms, including L_1 and our non-strict L_1/L_2 norms. First we will review the available norms which can be used as the optimization criteria. Then we will discuss the mechanics of our new solver framework and describe a technique for iterative plane-search, potentially enabling faster convergence for certain classes of problems. Finally, we will reference applications which are described in other SEP reports (Li and Maysami, 2009; Wong et al., 2009).

NORM OPTIONS

Our solver framework allows easy interchangeability between several norms. Although the code allows easy switching of the optimization measure, we recommend a thorough understanding of the theoretical and numerical caveats that result from the application of each solver criterion.

Norm	Description
L2	Conventional L_2 norm, utilizing the new solver framework
L1	L_1 norm with discontinuous 1^{st} and 2^{nd} order derivatives
Huber	Huber L_1/L_2 norm with with 1^{st} order derivative continuity
Hybrid	L_1/L_2 hybrid norm with 1^{st} and 2^{nd} order derivative continuity

The equations below summarize the analytical formulation for the listed norms above. $C,\ C', and\ C''$ represent the norm function, its first-order derivative and its second-order derivative, respectively. Figure 1 shows these norm functions along with their derivatives. Note discontinuous derivatives and zero-valued curvatures in some cases. These analytical forms are used in the Taylor series expansion for the adapted conjugate-direction plane-search.

L2 (Least Squares):

$$\begin{aligned} C(r) &= r^2/2 \\ C'(r) &= r \\ C''(r) &= 1 \end{aligned} \tag{1}$$

L1:

$$\begin{aligned} C(r) &= |r| \\ C'(r) &= \text{sgn}(r) \\ C''(r) &= 0 \quad \text{or} \quad \infty \end{aligned} \tag{2}$$

Huber:

$$\begin{aligned} C(r) &= \begin{cases} |r| - |r_t|/2 & |r/r_t| \geq 1 \\ r^2/2r_t & |r/r_t| < 1 \end{cases} \\ C'(r) &= \begin{cases} \text{sgn}(r/r_t) & |r/r_t| \geq 1 \\ r/r_t & |r/r_t| < 1 \end{cases} \\ C''(r) &= \begin{cases} 0 & |r/r_t| \geq 1 \\ 1/r_t & |r/r_t| < 1 \end{cases} \end{aligned} \tag{3}$$

Hybrid:

$$\begin{aligned} C(r) &= r_t^2 \left(\sqrt{1 + r^2/r_t^2} - 1 \right) \\ C'(r) &= \frac{r}{\sqrt{1 + r^2/r_t^2}} \\ C''(r) &= \frac{1}{(1 + r^2/r_t^2)^{\frac{3}{2}}} \end{aligned} \tag{4}$$

The choice of norm is specified as an input argument to our solver. A further benefit of this implementation is that other norms can be added with minimal modification to the overall solver framework. To add a new norm, all that is necessary is adding the appropriate definition of the norm and its derivatives in the code.

SOLVER INTERNAL MECHANISMS

In any optimization scheme, we always attempt to minimize some measure of a data or model residual, $\mathbf{r}$. This measure, $C(\mathbf{r})$, is usually a convex function (commonly the L_2 norm, for least-squares fitting). For our solver, $C(\mathbf{r})$ can be any of the norms listed in previous section. As proposed by Claerbout (2009), the numerical value of a norm at an updated residual value, $C(\mathbf{r}_2)$ can be estimated based on a second-order Taylor series decomposition at point $\mathbf{r}_1$:

$$C(\mathbf{r}_2) \approx \; C(\mathbf{r}_1) + \frac{(\mathbf{r}_2 - \mathbf{r}_1)}{1!} \, C'(\mathbf{r}_1) + \frac{(\mathbf{r}_2 - \mathbf{r}_1)^2}{2!} \, C''(\mathbf{r}_1) \tag{5}$$

where r_2 is the point in a close neighborhood of r_1. With this generalization, we can conduct an iterative plane-search at any point r_2, without re-evaluating the forward operator. For operators with a high-op count per sample this is a less costly by finding a more optimal update to the solution.

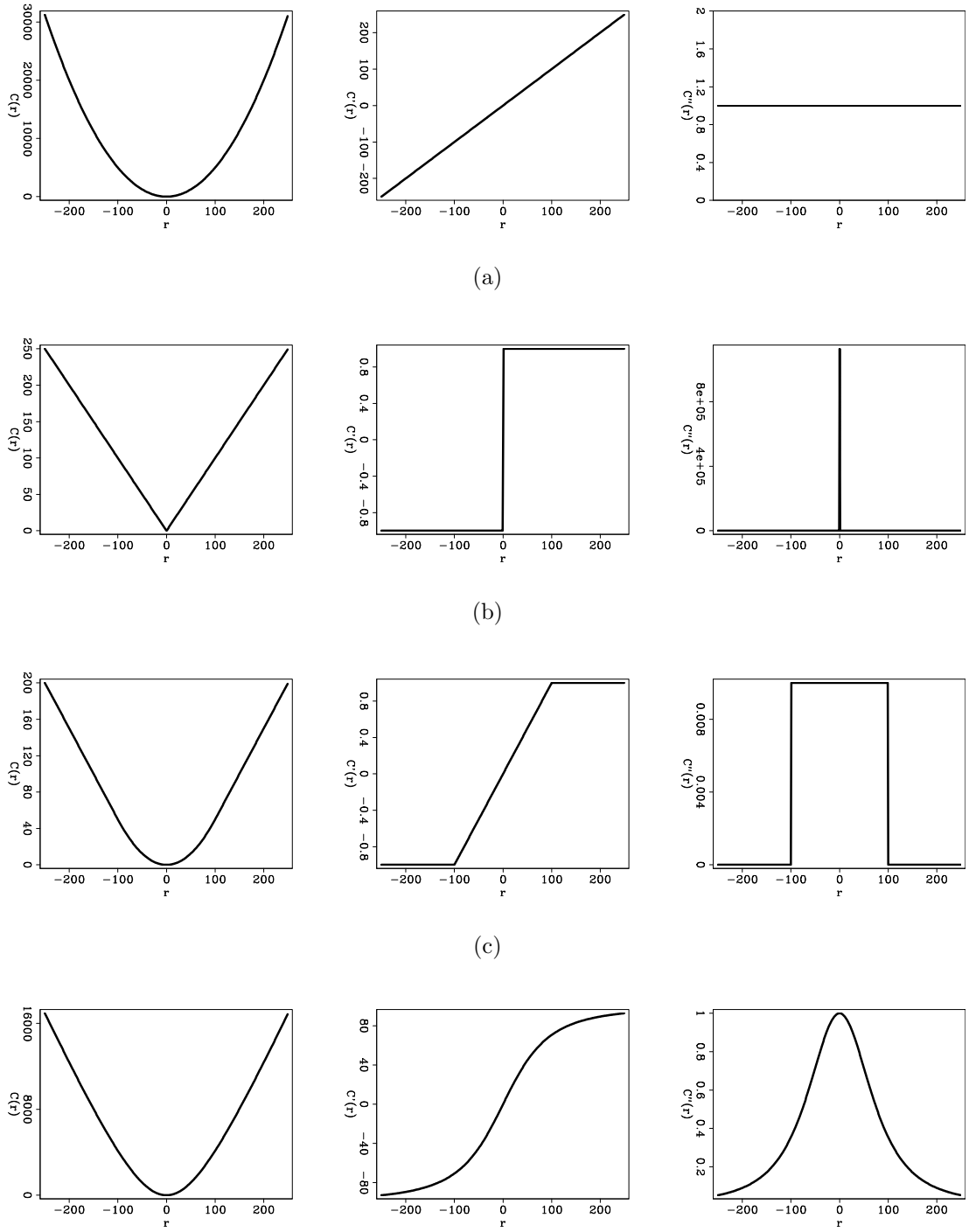

Figure 1: Norm functions and their first and second derivatives plotted for $r = -250, 250$ interval, with $r_t = 100$ where applicable. Rows from top to bottom are representing **(a)**L_2, **(b)** L_1, **(c)** Huber, and **(d)** Hybrid. Columns from left to right are representing norm function, first-order derivative, second-order derivative. [**ER**]

Iterated Plane-Search

In the conjugate-direction descent (Claerbout, 2008), the update in the model and residual is defined by a linear combination of the current gradient of the objective function and the direction of the previous update step. However, this update can be iterated inside an inner loop, assuming a linearization of the gradient around the current residual. Our algorithm can perform an iterated search in the plane defined by the current gradient and the previous update step. This search can locate the optimal update without a full re-evaluation of the gradient operator. In practice, this is equivalent to finding the solution of a 2×2 system of equations, where the unknowns are the step lengths in the direction of the gradient $\mathbf{g}^{(m)}$ and the previous step $\mathbf{s}^{(m)}$.

Taylor series expansion of the objective function around the initial residual value, assumes a small local neighborhood for the plane-search. We re-calculate this approximation at every inner iteration; this is an essential element of the plane-search, to ensure a reasonable degree of accuracy without a full re-computation of the forward operator in the main body of solver.

Algorithm Pseudo-code

The steps of actual solver have a structure very similar steps as the L_2 solver. The chief difference lies in the generalization to allow for different norms in the gradient calculation. The generalized form of the gradient for a given norm $C(\cdot)$ is given by

$$\mathbf{g}^{(m)} \;=\; \Delta\mathbf{m} \;=\; \mathbf{F}^T \; C'(\mathbf{r}) \;, \tag{6}$$

where F is the forward operator, $\mathbf{m}$ is the model value and $\mathbf{g}^{(m)}$ is the gradient. In addition, our code allows for an iterative plane-search to update $\mathbf{r}$, the intermediate estimated residual. The pseudo-code in Algorithm 1 summarizes the implementation for a simple solver (see Claerbout (2009) for more details). The benefit of our framework is that it can be easily modified to allow regularization and preconditioning without extensively changing the main solver algorithm. Such modifications primarily affect the stepper code. Fortran implementations of plane-search stepper function for both conjugate-direction and our generalized norm stepper are given in the appendices. Note that we approximate the data-space value of the gradient throughout the entire plane-search by a constant, to avoid re-evaluating the forward operator every iteration.

$$\mathbf{F}\mathbf{g}_i^{(m)} \approx \mathbf{F}\mathbf{g}_0^{(m)} \tag{7}$$

This assumption forces our plane-search to remain in a local neighborhood around $\mathbf{r}_1$, which is the desired outcome for this local approximation anyway.

Algorithm 1 Generalized Norm Solver Algorithm

$\mathbf{m} = \mathbf{m}_0$
$\mathbf{s}^{(m)}, \mathbf{s}^{(d)} = \mathbf{0}$

$\mathbf{\Delta m} = \mathbf{0}$
$\mathbf{\Delta r} = \mathbf{0}$
$\alpha = 0,\ \beta = 0$

!! Main iteration loop
for $iter = 1, niter$ **do**
  $\mathbf{r} = \mathbf{F}\mathbf{m} - \mathbf{d}$

  $\mathbf{g}^{(m)} = \mathbf{\Delta m} = \mathbf{F}'\ C'(\mathbf{r})$!! Gradient in model space
  $\mathbf{g}^{(d)} = \mathbf{\Delta r} = \mathbf{F}\ \mathbf{g}^{(m)}$!! Gradient in data space

  !! Plane-search loop
  for $i = 1, psiter$ **do**

    !! Solve for $\alpha,\ \beta$

$$\sum C_i''(\mathbf{r}) \left[\begin{pmatrix} g_i^{(d)} \\ s_i^{(d)} \end{pmatrix} \begin{pmatrix} g_i^{(d)} & s_i^{(d)} \end{pmatrix} \right] \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = -\sum C_i'(\mathbf{r}) \begin{pmatrix} g_i^{(d)} \\ s_i^{(d)} \end{pmatrix}$$

    $\mathbf{s}^{(m)} = \mathbf{\Delta m} = \alpha\ \mathbf{g}^{(m)} + \beta\ \mathbf{s}^{(m)}$!! update solution step
    $\mathbf{s}^{(d)} = \mathbf{\Delta r} = \alpha\ \mathbf{g}^{(d)} + \beta\ \mathbf{s}^{(d)}$!! update residual step

    $\mathbf{m} = \mathbf{m} + \mathbf{s}^{(m)}$!! update solution
    $\mathbf{r} = \mathbf{r} + \mathbf{s}^{(d)}$!! update residual

    if $\sum \|s_i^{(m)}/m_i\| < 10^{-6}$ **then**

      Quit plane-search iteration loop
    end if
  end for

end for

NEW EXTERNAL PARAMETERS

Setting threshold with percentile

The benefit of a mixed L_1/L_2 norm is that small residuals can be optimized in an L_2 sense while large residuals are treated by L_1. This requires a definition of "small residual"; how small is "small"?

Our implementation addresses this issue with a numerical parameter, r_t. This is the threshold for transition between L_1 and L_2 in Huber and Hybrid norms. According to the analytic definition of each norm, r_t adjusts the crossover point. Needless to say, pure L_1 and L_2 have no such transition. To reduce the problem-specific dependency of r_t, we have configured our solver to compute this threshold based on a user-defined percentile. By switching to percentile, we retain a physical meaning for this user-specified parameter. It is possible to use separate thresholds, and even different norms, for the model- and data-fitting goals of a general regularized or preconditioned optimization problem.

Plane-search iteration count

Our iterative plane-search adds additional flexibility if the user so desires. The conventional conjugate-direction descent stepper (Claerbout, 2008) lacks this feature since it only searches the plane of the gradient and the previous step once. To utilize our plane-search the user may specify another value for this parameter, allowing the solver to scan the local neighborhood around the internally-computed residual values. When the plane-search iteration is greater than one, the solver will estimate that many model updates between each full function evaluation. This is particularly useful if the forward-operator is computationally expensive.

SUMMARY

We have provided this implementation of the generalized norm solver to SEP for benchmarking and testing on a variety of sample problems. These experiments have been documented in separate SEP reports which focuses on the geophysical ramifications of the L_1 and modified L_1 optimization criteria. In conclusion, our solver has been designed to be interchangeable with existing codes, requiring minimal code modification. We hope this will encourage other researchers inside and outside SEP to experiment with these available optimization objectives.

APPENDIX A: ANALYTICAL DERIVATION OF PLANE-SEARCH STEP SIZES

This appendix shows the details on generalization of plane-search algorithm for a general norm (or measure) $C(\cdot)$. As discussed previously and by Claerbout (2009), we use Taylor series expansion to find analytical forms for the step sizes in the plane-search algorithm. We form the updates in the residual value $\mathbf{r}$ by a linear combination of the gradient $\mathbf{g}^{(d)}$ and the previous step update of the residual $\mathbf{s}^{(d)}$, i.e. $\mathbf{r} = \mathbf{r} + \alpha\mathbf{g}^{(d)} + \beta\mathbf{s}^{(d)}$. Then the misfit objective function $E(\mathbf{r})$ is given by

$$\begin{aligned} E(\mathbf{r}) &= \sum_i C(r_i + \alpha g_i^{(d)} + \beta s_i^{(d)}) \\ &= \sum_i C(r_i) + \frac{\alpha\ g_i^{(d)} + \beta\ s_i^{(d)}}{1!} C'(r_i) + \frac{\left(\alpha\ g_i^{(d)} + \beta\ s_i^{(d)}\right)^2}{2!} C''(r_i) \end{aligned} \tag{8}$$

where r_i is the residual from the current iteration. The Taylor series expansion in Equation 8 lets us find analytical derivatives of the misfit function $E(\mathbf{r})$ with respect to both α and β as follows:

$$\frac{\partial E}{\partial \alpha} = \sum_i g_i^{(d)} C'(r_i) + (\alpha\ g_i^{(d)} + \beta\ s_i^{(d)}) g_i^{(d)} C''(r_i) = 0, \tag{9}$$

$$\frac{\partial E}{\partial \beta} = \sum_i s_i^{(d)} C'(r_i) + (\alpha\ g_i^{(d)} + \beta\ s_i^{(d)}) s_i^{(d)} C''(r_i) = 0. \tag{10}$$

By setting these derivatives to zero and solving the 2×2 system of equations we find an optimal step size in both directions $\mathbf{g}^{(d)}$ and $\mathbf{s}^{(d)}$. The equation below shows the solutions α and β for this system of equations in a simplified notation.

$$\sum C_i''(r) \left[\begin{pmatrix} g_i^{(d)} \\ s_i^{(d)} \end{pmatrix} \begin{pmatrix} g_i^{(d)} & s_i^{(d)} \end{pmatrix} \right] \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = -\sum C_i'(r) \begin{pmatrix} g_i^{(d)} \\ s_i^{(d)} \end{pmatrix} \tag{11}$$

APPENDIX B: FORTRAN CODES FOR CGSTEP AND GENERALIZED CGNORM PLANE-SEARCH STEPPER

This appendix includes the fortran codes for `cgstep.f90` and `cgnorm.f90`. Note that type of norm and threshold are set inside an initialization subroutine which is called prior to stepper call.

cgstep.f90

```
integer function cgstep(forget, x, g, rr, gg)
  real, dimension (:)   :: x, g, rr, gg
```

```
    logical                 :: forget
    double precision        :: alfa, beta, determ
    double precision        :: sds, gdg, gds, gdr, sdr
    if ( .not. allocated (s)) then
      forget = .true.
      allocate ( s (size ( x)))
      allocate (ss (size (rr)))
    end if
    if ( forget) then
      s = 0.
      ss = 0.
      beta = 0.d0     ! steepest descent
      if ( dot_product(gg, gg) .eq. 0 ) then
        call erexit('cgstep: grad vanishes identically')
      end if
      alfa = - sum( dprod( gg, rr)) / sum( dprod(gg, gg))
    else
      gdg = sum( dprod( gg, gg))
      ! search plane by solving 2-by-2
      sds = sum( dprod( ss, ss))
      !  G . (R - G*alfa - S*beta) = 0
      gds = sum( dprod( gg, ss))
      !  S . (R - G*alfa - S*beta) = 0
      if ( gdg.eq.0. .or. sds.eq.0.) then
        cgstep = 1
        return
      end if
      determ = gdg*sds * max(1.d0-(gds/gdg)*(gds/sds),1.d-12)
      gdr    = - sum( dprod( gg, rr))
      sdr    = - sum( dprod( ss, rr))
      alfa   = ( sds * gdr - gds * sdr ) / determ
      beta   = (-gds * gdr + gdg * sdr ) / determ
    end if
    s  = alfa *  g + beta *  s        ! update solution step
    ss = alfa * gg + beta * ss        ! update residual step
    x  =  x +  s                      ! update solution
    rr = rr + ss                      ! update residual
    forget = .false.
    cgstep = 0
  end function
```

cgnorm.f90

```
  subroutine cgnorm_init(rthr,norm,psiter)
     ! Initialize CGNORM Stepper with r_thr, norm type and psiter_in
     !
     ! Arguments IN:
     !    rthr      : RBAR or R_THR for norm
     !    norm_in   : Norm ype to be used (Hybrid or Huber)
     !    psiter_in : Number of Plane-search iterations on Alpha,Beta
     !
```

```
    real              :: rthr
    integer           :: psiter
    character(10)     :: norm
    optional          :: psiter,norm

   ps_norm="huber"
    if (present(norm)) then
       ps_norm=norm
    end if

    ps_iter=1
    if (present(psiter)) then
       ps_iter=psiter
    end if

   rt = abs(rthr)      ! Rt is positive
    if (ps_norm(1:5)=="huber" .or. ps_norm(1:6)=="hybrid") then
       if (rt==0.0)  call erexit('r_thr = 0. is not a valid choice.')
    endif

  end subroutine

  ! ------------------------------------------
  !              CGNORM function
  ! ------------------------------------------
  integer function cgnorm( forget, x, g, rr, gg)
    ! Iterative Plane-Search Stepper function
    !    for different norms (Hybrid, Huber, etc.)

    ! Arguments:
    ! x, g   : x, dx
    ! rr,gg  : R, dR
    ! forget : set steepest-decsend if true and CG-direction if false

    ! Output Flag (cgnorm):
    !   2 : Exit
    !   1 : Exit with vanishing gradient status
    !   0 : Done without any exit or error
    !
    double precision, dimension(:), allocatable     :: sds, gdg, gds
    double precision, dimension(:), allocatable     :: c0,c1,c2
    real, dimension(:), allocatable     :: check
    real, dimension (:)  :: x, g, rr, gg
    logical              :: forget
    double precision     :: alfa, beta, determ
    double precision     :: c2dgg,c2dss,c2dgs,c1dg,c1ds
    integer              :: i,ps,stat

    cgnorm = 2
    allocate (c0(size(rr)),c1(size(rr)),c2(size(rr)))
    allocate (gdg(size(rr)),sds(size(rr)),gds(size(rr)))
    allocate ( check(size( x)))

    if( .not. allocated(s)) then
```

```
      forget = .true.
      allocate ( s(size( x)))
      allocate (ss(size(rr)))
   end if

   ! Loop over alpha , Beta
   do ps=1,ps_iter

      ! === Compute Norm and its derivatives
      if      (ps_norm(1:2)=="l2")      then
         stat = l2      (rr, c0,c1,c2)
      else if (ps_norm(1:2)=="l1")      then
         stat = l1      (rr, c0,c1,c2)
      else if (ps_norm(1:6)=="hybrid")  then
         stat = hybrid  (rr, c0,c1,c2)
      else
         stat = huber   (rr, c0,c1,c2)
      end if

     ! === Pick Line Search or Plane Search
      if( forget) then
         ! Steepest Descent
         s    = 0.
         ss   = 0.
         beta = 0.d0

         gdg   = dprod(gg,gg)
         c2dgg = sum(dprod(c2,gdg))

         !!! Discard C" if C"= 0  (OPTIONAL)
         ! if (sum(abs(c2))==0.0) c2dgg=sum(gdg)

         if( c2dgg == 0.0 )  then
            cgnorm = 1
            call erexit('NORM_CG_STEP: Gradient vanishes: C"*gg*gg=0.')
         end if
         alfa = - sum( dprod(c1,gg)) / c2dgg

      else
         ! Plane Search for alpha & beta
         c1dg = - sum(dprod(c1,gg))  ! c1dg = -C'*G
         c1ds = - sum(dprod(c1,ss))  ! c1ds = -C'*S

         gdg = dprod(gg,gg)       ! search plane by solving 2-by-2
         sds = dprod(ss,ss)       ! C"*G. (G*alfa + S*beta) = -C'*G
         gds = dprod(gg,ss)       ! C"*S. (G*alfa + S*beta) = -C'*S


         c2dgg = sum(dprod(c2,gdg))  ! c2dgg = C"*G*G
         c2dss = sum(dprod(c2,sds))  ! c2dss = C"*S*S
         c2dgs = sum(dprod(c2,gds))  ! c2dgs = C"*G*S

         !if (sum(abs(c2))==0.0) then
         ! !!! Discard C" if C"= 0 (OPTIONAL)
```

```
            ! c2dgg=sum(gdg)
            ! c2dss=sum(sds)
            !end if

            if ((c2dgg==0.d0) .OR. (c2dss==0.d0)) then
               cgnorm = 1;
               write (0,*) 'Plane Search : Gradient vanishes:
               C"*gg*gg=0 OR C"*ss*ss=0.'
               return
            end if
            determ = c2dgg * c2dss * max(1.d0 - (c2dgs/c2dgg)*(c2dgs/
                c2dss),1.d-12)
            alfa = ( c2dss * c1dg - c2dgs * c1ds ) / determ
            beta = (-c2dgs * c1dg + c2dgg * c1ds ) / determ
         end if

         !  Updates on model and residual
         s  = alfa *  g + beta *  s          ! update solution step
         ss = alfa * gg + beta * ss          ! update residual step

         check=0.
         do i=1,size(x)
             check=check+(s(i)/x(i))**2
         end do
         if  (check< 1.e-6) exit

         x  =  x +  s                        ! update solution
         rr = rr + ss                        ! update residual
         forget = .false.;

         !write (0,*) "Alpha, Beta::" ,alfa,beta
         !write (0,*) "Determ::" ,determ
      end do


      deallocate (c0,c1,c2)
      deallocate (gdg,sds,gds)
      cgnorm = 0
   end function
```

REFERENCES

Bube, K. P. and R. T. Langan, 1997, Hybrid l^1/l^2 minimization with applications to tomography: Geophysics, **62**, 1183–1195.

Claerbout, J. F., 2008, Image estimation by example.

———, 2009, Blocky models via the $l1/l2$ hybrid norm: SEP-Report, **139**, 1–10.

Guitton, A., 2000, Implementation of a nonlinear solver for minimizing the huber norm: SEP-Report, **103**, 281–289.

Huber, P. J., 1973, Robust regression: Asymptotics, conjectures, and monte carlo: Annals of Statistics, **1**, 799–821.

Li, Y. E. and M. Maysami, 2009, Dix inversion constrained by l1-norm optimization: SEP-Report, **139**, 23–36.

Paige, C. C. and M. A. Saunders, 1982, Algorithm 583 lsqr: spare linear equations and sparse least squares problems: ACM Transaction on Mathematical Software, **8**, 195–209.

Wong, M., N. W. Moussa, and M. Maysami, 2009, Applications of generalized norm solver: SEP-Report, **139**, 37–48.