

**NAME**

CUTEST\_csgreh – CUTEst tool to evaluate the constraint gradients, the Lagrangian Hessian in finite element format and the gradient of either the objective/Lagrangian in sparse format.

**SYNOPSIS**

CALL CUTEST\_csgreh( status, n, m, X, Y, grlagf, nnzj, lj, J\_val, J\_var, J\_fun, ne, lhe\_ptr, HE\_row\_ptr, HE\_val\_ptr, lhe\_row, HE\_row, lhe\_val, HE\_val, byrows )

**DESCRIPTION**

The CUTEST\_csgreh subroutine evaluates both the gradients of the general constraint functions and the Hessian matrix of the Lagrangian function  $l(x, y) = f(x) + y^T c(x)$  for the problem decoded into OUTSDIF.d at the point  $(x, y) = (X, Y)$ . This Hessian matrix is stored as a sparse matrix in finite element format

$$H = \sum_{e=1}^{ne} H_e,$$

where each square symmetric element  $H_e$  involves a small subset of the rows of the Hessian matrix. The subroutine also obtains the gradient of either the objective function or the Lagrangian function, stored in a sparse format.

The problem under consideration consists in minimizing (or maximizing) an objective function  $f(x)$  over all  $x \in R^n$  subject to general equations  $c_i(x) = 0$ , ( $i \in 1, \dots, m_E$ ), general inequalities  $c_i^l(x) \leq c_i(x) \leq c_i^u(x)$ , ( $i \in m_E + 1, \dots, m$ ), and simple bounds  $x^l \leq x \leq x^u$ . The objective function is group-partially separable and all constraint functions are partially separable.

**ARGUMENTS**

The arguments of CUTEST\_csgreh are as follows

**status** [out] - integer

the output status: 0 for a successful call, 1 for an array allocation/deallocation error, 2 for an array bound error, 3 for an evaluation error,

**n** [in] - integer

the number of variables for the problem,

**m** [in] - integer

the total number of general constraints,

**X** [in] - real/double precision

an array which gives the current estimate of the solution of the problem,

**Y** [in] - real/double precision

an array which gives the Lagrange multipliers,

**grlagf** [in] - logical

a logical variable which should be set .TRUE. if the gradient of the Lagrangian function is required and .FALSE. if the gradient of the objective function is sought,

**nnzj** [out] - integer

the number of nonzeros in J\_val,

**HE\_row** [out] - integer

an array which holds a list of the row indices involved with each element. Those for element  $i$  directly precede those for element  $i+1$ ,  $i = 1, \dots, ne-1$ . Since the elements are symmetric, HE\_row is also the list of column indices involved with each element.

**lj** [in] - integer

the actual declared dimensions of J\_val, J\_var and J\_fun,

**J\_val** [out] - real/double precision

an array which gives the values of the nonzeros of the gradients of the objective, or Lagrangian, and general constraint functions evaluated at X and Y. The  $i$ -th entry of J\_val gives the value of the derivative with respect to variable J\_var( $i$ ) of function J\_fun( $i$ ),

**J\_var** [out] - integer

an array whose i-th component is the index of the variable with respect to which J\_val(i) is the derivative,

**J\_fun** [out] - integer

an array whose i-th component is the index of the problem function whose value J\_val(i) is the derivative. J\_fun(i) = 0 indicates the objective function whenever grlagf is .FALSE. or the Lagrangian function when grlagf is .TRUE., while J\_fun(i) = j > 0 indicates the j-th general constraint function.

**ne** [out] - integer

the number, ne, of "finite-elements" used,

**lhe\_ptr** [in] - integer

the actual declared dimensions of HE\_row\_ptr and HE\_val\_ptr,

**HE\_row\_ptr** [out] - integer

HE\_row\_ptr(i) points to the position in HE\_row of the first row index involved with element number e: the row indices of element number e are stored in HE\_row between the indices HE\_row\_ptr(e) and HE\_row\_ptr(e+1)-1. HE\_row\_ptr(ne+1) points to the first empty location in HE\_row,

**HE\_val\_ptr** [out] - integer

HE\_val\_ptr(i) points to the position in HE\_val of the first nonzero involved with element number i: the values involved in element number e are stored in HE\_val between the indices HE\_val\_ptr(e) and HE\_val\_ptr(e+1)-1. HE\_val\_ptr(ne+1) points to the first empty location in HE\_val,

**lhe\_row** [in] - integer

the actual declared dimension of HE\_row,

**HE\_row** [out] - integer

an array which holds a list of the row indices involved with each element. Those for element e directly precede those for element e+1, e = 1, ..., ne-1. Since the elements are symmetric, HE\_row is also the list of column indices involved with each element.

**lhe\_val** [in] - integer

the actual declared dimension of HE\_val,

**HE\_val** [out] - real/double precision

an array of the nonzeros in the upper triangle of H\_e, evaluated at X and stored by rows, or by columns. Those for element e directly precede those for element, e+1, i = 1, ..., ne-1. Element number e contains the values stored between

HE\_val( HE\_val\_ptr(e) ) and HE\_val( HE\_val\_ptr(e+1)-1 )

and involves the rows/columns stored between

HE\_row( HE\_row\_ptr(e) ) and HE\_row( HE\_row\_ptr(e+1)-1 ).

**byrows** [in] - logical

must be set to .TRUE. if the upper triangle of each H\_i is to be stored by rows, and to .FALSE. if it is to be stored by columns.

## AUTHORS

I. Bongartz, A.R. Conn, N.I.M. Gould, D. Orban and Ph.L. Toint

## SEE ALSO

*CUTEst: a Constrained and Unconstrained Testing Environment with safe threads*,  
N.I.M. Gould, D. Orban and Ph.L. Toint,  
Technical Report, Rutherford Appleton Laboratory, 2013.

*CUTEr (and SifDec): A Constrained and Unconstrained Testing Environment, revisited*,  
N.I.M. Gould, D. Orban and Ph.L. Toint,  
ACM TOMS, **29**:4, pp.373-394, 2003.

*CUTE: Constrained and Unconstrained Testing Environment*, I. Bongartz, A.R. Conn, N.I.M. Gould and Ph.L. Toint, ACM TOMS, **21**:1, pp.123-160, 1995.

cutest\_ugreh(3M), sifdecoder(1).