

**NAME**

CUTEST\_ccfg – CUTEst tool to evaluate constraint functions values and possibly gradients.

**SYNOPSIS**

CALL CUTEST\_ccfg( status, n, m, X, C, jtrans, lj1, lj2, J\_val, grad )

**DESCRIPTION**

The CUTEST\_ccfg subroutine evaluates the values of the constraint functions of the problem decoded from a SIF file by the script *sifdecode* at the point  $X$ , and possibly their gradients. The problem under consideration is to minimize or maximize 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\_ccfg 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,

**C** [out] - real/double precision

an array which gives the values of the general constraint functions evaluated at  $X$ . The  $i$ -th component of  $C$  will contain the value of  $c_i(x)$ ,

**jtrans** [in] - logical

a logical variable which should be set `.TRUE.` if the transpose of the constraint Jacobian is required and `.FALSE.` if the Jacobian itself is wanted. The Jacobian matrix is the matrix whose  $i$ -th row is the gradient of the  $i$ -th constraint function,

**lj1** [in] - integer

the actual declared size of the leading dimension of  $J\_val$  (with  $lj1$  no smaller than  $n$  if  $jtrans$  is `.TRUE.` or  $m$  if  $jtrans$  is `.FALSE.`),

**lj2** [in] - integer

the actual declared size of the second dimension of  $J\_val$  (with  $lj2$  no smaller than  $m$  if  $jtrans$  is `.TRUE.` or  $n$  if  $jtrans$  is `.FALSE.`),

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

a two-dimensional array of dimension  $(lj1, lj2)$  which gives the value of the Jacobian matrix of the constraint functions, or its transpose, evaluated at  $X$ . If  $jtrans$  is `.TRUE.`, the  $i,j$ -th component of the array will contain the  $i$ -th derivative of the  $j$ -th constraint function. Otherwise, if  $jtrans$  is `.FALSE.`, the  $i,j$ -th component of the array will contain the  $j$ -th derivative of the  $i$ -th constraint function,

**grad** [in] - logical

a logical variable which should be set `.TRUE.` if the gradient of the constraint functions are required and `.FALSE.` otherwise.

**AUTHORS**

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

**SEE ALSO**

*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, TOMS, **21**:1, pp.123-160, 1995.

sifdecode(1)

**NAME**

CUTEST\_ccfg – CUTEst tool to evaluate constraint functions values and possibly gradients.

**SYNOPSIS**

CALL CUTEST\_ccfg( status, n, m, X, C, jtrans, lj1, lj2, J\_val, grad )

**DESCRIPTION**

The CUTEST\_ccfg subroutine evaluates the values of the constraint functions of the problem decoded from a SIF file by the script *sifdecode* at the point  $X$ , and possibly their gradients. The problem under consideration is to minimize or maximize 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\_ccfg 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,

**C** [out] - real/double precision

an array which gives the values of the general constraint functions evaluated at  $X$ . The  $i$ -th component of  $C$  will contain the value of  $c_i(x)$ ,

**jtrans** [in] - logical

a logical variable which should be set `.TRUE.` if the transpose of the constraint Jacobian is required and `.FALSE.` if the Jacobian itself is wanted. The Jacobian matrix is the matrix whose  $i$ -th row is the gradient of the  $i$ -th constraint function,

**lj1** [in] - integer

the actual declared size of the leading dimension of  $J\_val$  (with  $lj1$  no smaller than  $n$  if  $jtrans$  is `.TRUE.` or  $m$  if  $jtrans$  is `.FALSE.`),

**lj2** [in] - integer

the actual declared size of the second dimension of  $J\_val$  (with  $lj2$  no smaller than  $m$  if  $jtrans$  is `.TRUE.` or  $n$  if  $jtrans$  is `.FALSE.`),

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

a two-dimensional array of dimension ( $lj1, lj2$ ) which gives the value of the Jacobian matrix of the constraint functions, or its transpose, evaluated at  $X$ . If  $jtrans$  is `.TRUE.`, the  $i,j$ -th component of the array will contain the  $i$ -th derivative of the  $j$ -th constraint function. Otherwise, if  $jtrans$  is `.FALSE.`, the  $i,j$ -th component of the array will contain the  $j$ -th derivative of the  $i$ -th constraint function,

**grad** [in] - logical

a logical variable which should be set `.TRUE.` if the gradient of the constraint functions are required and `.FALSE.` otherwise.

**AUTHORS**

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

**SEE ALSO**

*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, TOMS, **21**:1, pp.123-160, 1995.

sifdecode(1)