NAME

CUTEst − a Constrained and Unconstrained Testing Environment for optimization software using Safe Threads.

SYNOPSIS

CUTEst is an evolution of the earlier Fortran 77 package CUTEer that avoids the latter's use of thread-unsafe common. CUTEst is written in Fortran 2003, but the Fortran 77-style subroutine interfaces mean that it may be easily called from other languages such as C and packages such as Matlab; hooks to Matlab are available.

Interfaces to a number of popular optimization packages are provided.

DESCRIPTION

CUTEST provides a user-callable interface to any optimization problem written in Standard Input Format (SIF) and subsequently decoded from its SIF file by the script *sifdecode*. The problem under consideration is to minimize or maximize an objective function $f(x)$ o ver all $x \in R^n$ subject perhaps to general equations $c_i(x) = 0$, $(i \in 1, \ldots, m_E)$, general inequalities $c_i^l(x) \le c_i(x) \le c_i^u(x)$, $(i \in m_E + 1, \ldots, m)$, and simple bounds $x^l \le x \le x^{u.}$ The objective function in SIF is group-partially separable and all constraint functions are partially separable, but the user need not be aware of this. The user will be able to compute function, gradient and Hessian values at a specfied point for a variety of relevant functions including the Lagrangian function $l(x, y) = f(x) + y^T c(x)$. Matrices may be rquested in dense, sparse and finite-element formats, and matrix-vector products between these matrices and user-provided vectors may be obtained. Evaluations may be performed in parallel on shared-memory machines if required.

TOOLS AVAILABLE

Separate evaluation tools are provided for unconstrained and constrained problems. Both unthreaded and threaded versions are available when this is relevant.

Unconstrained problems:

cutest_udimen (both threaded and unthreaded)
determine the number of variables.

cutest_usetup (unthreaded) and **cutest_usetup_threaded** (threaded)
setup internal data structures and determine variable bounds.

cutest_unames (both threaded and unthreaded)
determine the names of the problem and the variables.

cutest_uvartype (both threaded and unthreaded)
determine whether the variables are continuous or discrete.

cutest_udimsh (both threaded and unthreaded)
determine the number of nonzeros in the sparse Hessian.

cutest_udimse (both threaded and unthreaded)
determine the number of nonzeros in the finite-element Hessian.

cutest_ufn (unthreaded) and **cutest_ufn_threaded** (threaded)
evaluate the objective function value.

cutest_ugr (unthreaded) and **cutest_ugr_threaded** (threaded)
evaluate the gradient of the objective function.

cutest_uofg (unthreaded) and **cutest_uofg_threaded** (threaded)
evaluate both the values and gradients of the objective function.

cutest_udh (unthreaded) and **cutest_udh_threaded** (threaded)
evaluate the Hessian of the objective function as a dense matrix.

cutest_ugrdh (unthreaded) and **cutest_ugrdh_threaded** (threaded)
evaluate the objective gradient and dense Hessian.

cutest_ushp (both unthreaded and threaded)
evaluate the sparsity pattern of the Hessian of the objective function.

cutest_ush (unthreaded) and **cutest_ush_threaded** (threaded)
evaluate the Hessian of the objective function as a sparse matrix.

cutest_ugrsh (unthreaded) and **cutest_ugrsh_threaded** (threaded)
evaluate the objective gradient and sparse Hessian.

cutest_ueh (unthreaded) and **cutest_ueh_threaded** (threaded)
evaluate the Hessian of the objective function as a finite-element matrix.

cutest_ugreh (unthreaded) and **cutest_ugreh_threaded** (threaded)
evaluate the objective gradient and finite-element Hessian.

cutest_uhprod (unthreaded) and **cutest_uhprod_threaded** (threaded)
evaluate the product of the Hessian of the objective function with a vector.

cutest_ushprod (unthreaded) and **cutest_ushprod_threaded** (threaded)
evaluate the product of the Hessian of the objective function with a sparse vector.

cutest_ubandh (unthreaded) and **cutest_ubandh_threaded** (threaded)
obtain the part of the Hessian of the objective that lies within a specified band.

cutest_ureport (unthreaded) and **cutest_ureport_threaded** (threaded)
discover how many evaluations have occured and how long this has taken.

cutest_uterminate (both unthreaded and threaded)
remove internal data structures when they are no longer needed.

Constrained problems:

cutest_cdimen (both threaded and unthreaded)
determine the number of variables and constraints.

cutest_csetup (unthreaded) and **cutest_csetup_threaded** (threaded)
setup internal data structures and determine variable and constraint bounds.

cutest_cnames (both threaded and unthreaded)
determine the names of the problem, the variables and the constraints.

cutest_connames (both threaded and unthreaded)
determine the names of the constraints.

cutest_cvartype (both threaded and unthreaded)
determine whether the variables are continuous or discrete.

cutest_cdimsj (both threaded and unthreaded)
determine the number of nonzeros in sparse constraint Jacobian.

cutest_cdimsh (both threaded and unthreaded)
determine the number of nonzeros in the sparse Hessian.

cutest_cdimse (both threaded and unthreaded)
determine the number of nonzeros in the finite-element Hessian.

cutest_cfn (unthreaded) and **cutest_cfn_threaded** (threaded)
evaluate the objective function and constraint values.

cutest_cgr (unthreaded) and **cutest_cgr_threaded** (threaded)
evaluate the gradients of the objective function and constraints.

cutest_cofg (unthreaded) and **cutest_cofg_threaded** (threaded)
evaluate both the value and gradient of the objective function.

cutest_cofsg (unthreaded) and **cutest_cofsg_threaded** (threaded)
evaluate both the value and sparse gradient of the objective function.

cutest_csgr (unthreaded) and **cutest_csgr_threaded** (threaded)
evaluate the sparse gradients of the objective function and constraints.

cutest_ccfg (unthreaded) and **cutest_ccfg_threaded** (threaded)
evaluate the values and gradients of the constraints.

cutest_ccfsg (unthreaded) and **cutest_ccfsg_threaded** (threaded)
evaluate the values and sparse gradients of the constraints.

cutest_clfg (unthreaded) and **cutest_clfg_threaded** (threaded)
evaluate both the value and gradient of the Lagrangian function.

cutest_ccifg (unthreaded) and **cutest_ccifg_threaded** (threaded)
evaluate the value and gradient of an individual constraint.

cutest_ccifsg (unthreaded) and **cutest_ccifsg_threaded** (threaded)
evaluate the value and sparse gradient of an individual constraint.

cutest_cdh (unthreaded) and **cutest_cdh_threaded** (threaded)
evaluate the Hessian of the Lagrangian function as a dense matrix.

cutest_cdhc (unthreaded) and **cutest_cdhc_threaded** (threaded)
evaluate the Hessian of the Lagrangian function not including the objective as a dense matrix.

cutest_cidh (unthreaded) and **cutest_cidh_threaded** (threaded)
evaluate the Hessian of the objective function or an individual constraint as a dense matrix.

cutest_cgrdh (unthreaded) and **cutest_cgrdh_threaded** (threaded)
evaluate the constraint Jacobian and Hessian of the Lagrangian function as dense matrices.

cutest_cshp (both unthreaded and threaded)
evaluate the sparsity pattern of the Hessian of the Lagrangian function.

cutest_csh (unthreaded) and **cutest_csh_threaded** (threaded)
evaluate the Hessian of the Lagrangian function as a sparse matrix.

cutest_cshc (unthreaded) and **cutest_cshc_threaded** (threaded)
evaluate the Hessian of the Lagrangian function not including the objective as a sparse matrix.

cutest_cish (unthreaded) and **cutest_cish_threaded** (threaded)
evaluate the Hessian of the objective function or an individual constraint as a sparse matrix.

cutest_csgrsh (unthreaded) and **cutest_csgrsh_threaded** (threaded)
evaluate the constraint Jacobian and Hessian of the Lagrangian function as sparse matrices.

cutest_ceh (unthreaded) and **cutest_ceh_threaded** (threaded)
evaluate the Hessian of the Lagrangian function as a finite-element matrix.

cutest_csgreh (unthreaded) and **cutest_csgreh_threaded** (threaded)
evaluate the constraint Jacobian as a sparse matrix and the Hessian of the Lagrangian function as a finite-element matrix.

cutest_chprod (unthreaded) and **cutest_chprod_threaded** (threaded)
evaluate the product of the Hessian of the Lagrangian function with a vector.

cutest_cshprod (unthreaded) and **cutest_cshprod_threaded** (threaded)
evaluate the product of the Hessian of the Lagrangian function with a sparse vector.

cutest_chcprod (unthreaded) and **cutest_chcprod_threaded** (threaded)
evaluate the product of the Hessian of the Lagrangian function not including the objective with a vector.

cutest_cshcprod (unthreaded) and **cutest_cshcprod_threaded** (threaded)
evaluate the product of the Hessian of the Lagrangian function not including the objective with a sparse vector.

cutest_cjprod (unthreaded) and **cutest_cjprod_threaded** (threaded)
evaluate the product of the constraint Jacobian or its transpose with a vector.

cutest_csjprod (unthreaded) and **cutest_csjprod_threaded** (threaded)
evaluate the product of the constraint Jacobian or its transpose with a sparse vector.

cutest_creport (unthreaded) and **cutest_creport_threaded** (threaded)
discover how many evaluations have occured and how long this has taken.

cutest_cterminate (both unthreaded and threaded)
remove internal data structures when they are no longer needed.

Both unconstrained problems and constrained problems:

cutest_pname (both threaded and unthreaded)
determine the name of the problem before initialization calls to cutest_u/csetup[_threaded]

cutest_probname (both threaded and unthreaded)
determine the name of the problem.

cutest_varnames (both threaded and unthreaded)
determine the names of the variables.

APPLICATION USAGE

A call to cutest_u/csetup[_threaded] must precede calls to any other evaluation tool with the exception of cutest_pname and cutest_u/cdimen. Once cutest_u/cterminate[_threaded]. has been called, no further calls should be made without first recalling cutest_u/csetup[_threaded].

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 for mathematical optimization,
N.I.M. Gould, D. Orban and Ph.L. Toint,
Computational Optimization and Applications **60**:3, pp.545-557, 2014.

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.

sifdecode(1), cutest_udimen(3M), cutest_usetup(3M), cutest_usetup_threaded(3M), cutest_unames(3M), cutest_uvartype(3M), cutest_udimsh(3M), cutest_udimse(3M), cutest_ufn(3M), cutest_ufn_threaded(3M), cutest_ugr(3M), cutest_ugr_threaded(3M), cutest_uofg(3M), cutest_uofg_threaded(3M), cutest_udh(3M), cutest_udh_threaded(3M), cutest_ugrdh(3M), cutest_ugrdh_threaded(3M), cutest_ush(3M), cutest_ush_threaded(3M), cutest_ugrsh(3M), cutest_ugrsh_threaded(3M), cutest_ueh(3M), cutest_ueh_threaded(3M), cutest_ugreh(3M), cutest_ugreh_threaded(3M), cutest_uhprod(3M), cutest_uhprod_threaded(3M), cutest_ushprod(3M), cutest_ushprod_threaded(3M), cutest_ubandh(3M), cutest_ubandh_threaded(3M), cutest_ureport(3M), cutest_ureport_threaded(3M), cutest_uterminate(3M), cutest_cdimen(3M), cutest_csetup(3M), cutest_csetup_threaded(3M), cutest_cnames(3M), cutest_connames(3M), cutest_cvartype(3M), cutest_cdimsj(3M), cutest_cdimsh(3M), cutest_cdimse(3M), cutest_cfn(3M), cutest_cfn_threaded(3M), cutest_cgr(3M), cutest_cgr_threaded(3M), cutest_cofg(3M), cutest_cofg_threaded(3M), cutest_cofsg(3M), cutest_cofsg_threaded(3M), cutest_csgr(3M), cutest_csgr_threaded(3M), cutest_ccfg(3M), cutest_ccfg_threaded(3M), cutest_ccfsg(3M), cutest_ccfsg_threaded(3M), cutest_ccifg(3M), cutest_ccifg_threaded(3M), cutest_ccifsg(3M), cutest_ccifsg_threaded(3M), cutest_cdh(3M), cutest_cdh_threaded(3M), cutest_cdhc(3M), cutest_cdhc_threaded(3M), cutest_cidh(3M), cutest_cidh_threaded(3M), cutest_cgrdh(3M), cutest_cgrdh_threaded(3M), cutest_csh(3M), cutest_csh_threaded(3M), cutest_cshc(3M), cutest_cshc_threaded(3M), cutest_cish(3M), cutest_cish_threaded(3M), cutest_csgrsh(3M), cutest_csgrsh_threaded(3M), cutest_ceh(3M), cutest_ceh_threaded(3M), cutest_csgreh(3M), cutest_csgreh_threaded(3M), cutest_chprod(3M), cutest_chprod_threaded(3M), cutest_cshprod(3M), cutest_cshprod_threaded(3M), cutest_chcprod(3M), cutest_chcprod_threaded(3M), cutest_cshcprod(3M), cutest_cshcprod_threaded(3M), cutest_cjprod(3M), cutest_cjprod_threaded(3M), cutest_csjprod(3M), cutest_csjprod_threaded(3M), cutest_creport(3M), cutest_creport_threaded(3M), cutest_cterminate(3M), cutest_probname(3M), cutest_varnames(3M).