NAME

knitma − CUTEr KNITRO test driver

SYNOPSIS

knitma

DESCRIPTION

The *knitma* main program test drives KNITRO on SIF problems from the CUTEr distribution.

KNITRO is a code for solving large-scale nonlinear programming problems of the form

min $f(x)$
s.t. $h_i(x) = 0,\quad i=1,...,ne$
$cl(j) \le g_j(x) \le cu(j),\quad j=ne+1,...,m$
$bl(k) \le x(k) \le bu(k),\quad k=1,...,n.$

The code implements an interior-point algorithm with trust-region techniques. It uses first and second derivatives of the function and constraints.

The library libknitrocuter.a should be stored in $MYCUTER/*precision*/lib, where *precision* is either "single" or "double", according to your local installation.

USAGE

Compile (but do not link) the KNITRO source code and copy the resulting library libknitrocuter.a in the directory $MYCUTER/*precision*/lib. Launch using knit(1) or sdknit(1).

VARIABLES USED BY KNITRO

n *INTEGER*: the number of variables.

m *INTEGER*: the number of constraints excluding the equality constraints for fixed variables and the inequality constraints for bounded variables.

c *DOUBLE PRECISION array of length m*: contains the general equality and inequality constraint values (it excludes fixed variables and bound constraints).

cl *DOUBLE PRECISION array of length m-num_equal*: cl(i) is the lower bound of the *i*-th inequality constraints c(i). If there is no such bound, set it to be the large negative number -biginf=-1.0d+20.

cu *DOUBLE PRECISION array of length m-num_equal*: cu(i) is the upper bound of the *i*-th inequality constraints c(i). If there is no such bound, set it to be the large positive number biginf=-1.0d+20.

bl *DOUBLE PRECISION array of length n*: bl(i) is the lower bound of the *i*-th variable x(i). If there is no such bound, set it to be the large negative number -biginf =-1.0d+20.

bu *DOUBLE PRECISION array of length n*: bu(i) is the upper bound of the *i*-th variable x(i). If there is no such bound, set it to be the large positive number biginf =1.0d+20.

equatn *LOGICAL array of length m*: equatn(i) indicates whether the *i*-th constraint is an equality constraint or not.

linear *LOGICAL array of length m*: linear(i) indicates whether the *i*-th constraint is a linear constraint or not.

nnzj *INTEGER*: the number of nonzeros in the Jacobian matrix cjac which contains the gradient of the objective function f and the constraint gradients A in sparse form.

cjac *DOUBLE PRECISION array of length nnzj*: the first part contains the nonzero elements of the gradient of the objective function; the second part contains the nonzero elements of the Jacobian of the constraints.

indfun *INTEGER array of length nnzj*: it is the indicator for the functions. If indfun(i)=0, it refers to the objective function. If indfun(i)=j, it refers to the j-th constraint.

indvar *INTEGER array of length nnzj*: it is the index of the variables. indfun and indvar determines the row number and the column number of Atrans respectively.

temp_v

DOUBLE PRECISION array of length m: contains the Lagrange multipliers. The Lagragian function is

$$L(x, temp\_v) = f(x) + temp\_v^T h(x)$$
$$= f(x) - \lambda_E^T h_E - \lambda_I (h_I - s)$$

Note that we set temp_vE=-λ_E and temp_vI=-λ_I in the barrier solver.

nnz_w *INTEGER*: denotes the number of nonzero elements of the upper triangle of the Hessian of the Lagrangian function.

w *DOUBLE PRECISION array of dimension nnz_w*: contains the Hessian of the Lagrangian in sparse form:

$$\nabla^2_{xx} L = \nabla^2_{xx} f + temp\_vE \nabla^2_{xx} h_E + temp\_vI \nabla^2_{xx} h_I$$

Only the upper triangle is stored.

w_row *INTEGER array of length nnz_w*: w_row(i) stores the row number of the nonzero element w(i).

w_col *INTEGER array of length nnz_w*: w_col(i) stores the column number of the nonzero element w(i).

max_num_iterations

INTEGER: specifi es the maximum number of iterations before termination.

NLP_tol

DOUBLE PRECISION: specifi es the fi nal stopping tolerance for both the KKT error and the feasibility error.

init_delta

DOUBLE PRECISION: specifi es the initial trust-region radius.

pivot_tol

DOUBLE PRECISION: specifi es the initial pivot tolerance used in the factorization routine. The value must be in the range [-0.5 0.5] with higher values resulting in more pivoting (more stable factorization).

mu0 *DOUBLE PRECISION*: specifi es the initial barrier parameter value.

use_SOC

LOGICAL: indicates whether or not to enable the second order correction option.

use_feasible

LOGICAL: indicates whether or not to use the feasible version.

Direct_Solver
LOGICAL: indicates whether or not to enable the Direct Solve option.

nout *INTEGER*: specifi es where to direct the output.

iprint *INTEGER*: controls the level of output.

NOTE

If no KNITRO.SPC fi le is present in the current directory, the default version is copied from $CUTER/common/src/pkg/knitro/.

ENVIRONMENT

CUTER
Parent directory for CUTEr

MYCUTER
Home directory of the installed CUTEr distribution.

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.

[1] *A trust region method based on interior point techniques for nonlinear programming*,
R.H. Byrd, J.-C. Gilbert, and J. Nocedal,
Technical Report OTC 96/02,
Optimization Technology Center,
Northwestern University (1996).
Note: provides a global convergence analysis

[2] *An interior point algorithm for large scale nonlinear programming*,
R.H. Byrd, M.E. Hribar, and J. Nocedal,
SIAM Journal on Optimization, **9**:4, (1999) pp.877-900
Note: this paper gives a description of the
algorithm implemented in KNITRO.
Some changes have occurred since then; see [4].

[3] *On the local behavior of an interior point method for nonlinear programming*,
R.H. Byrd, G. Liu, and J. Nocedal,
Numerical analysis, D.F. Griffi ths, D.J. Higham and
G.A. Watson eds., Longman, 1997.
Note: this paper studies strategies for ensuring a
fast local rate of convergence. These have not yet

been implemented in the current version of KNITRO.

[4] *Design Issues in Algorithms for Large Scale Nonlinear Programming*,
G. Liu, PhD thesis, Department of Industrial Engineering and Management Science,
Northwestern University, Evanston, Il, USA, 1999
Note: this paper describes a number of enhancements implemented in the current version of the code.