NLPAPI: An API to Nonlinear Programming Problems

(i.e. LANCELOT's form). For example

$$O(\mathbf{v}) = \sum_{i=1}^{n_g} \frac{1}{s_i} g_i\left(\sum_{j=1}^{n_{e_i}} w_{ij} f_{ij}(R\mathbf{ev}_{ij}) + < a_i, \mathbf{v} > - b\right)$$

g → IR are caled groups funct
nonlinear elements, $< a_i, \mathbf{v} >$ is called the
is called the constant element.

To invoke a solver the user creates a s

2.1.1 The Objective

linear element, $\mathbf{a}_i.\mathbf{v}$, a constant element b_i, and a nonlinear element $N_i(\mathbf{v})$. (See the sections below on "Groups").

$$O(\mathbf{v}) = \square \; 1$$

```
"[x1,x2,x3]","48-x1**2-x2**2-x3**2");
```

The array v lists the nv variables whose values are substituted for the variables "[x1,x2,x3]".

When the constraint is first created it has a single, empty group. Additional groups can be added using the NLPAddGroupToInequalityConstraint routine. Each group has a group function, a scale, a linear element, a constant element, and a set of nonlinear elements. (See the section below on "Groups").

```
NLPAddNonlinearInequalityConstraint(P,name);

NLPAddLinearInequalityConstraint(P,name,a,b);

NLPSetInequalityCtrairt
NLPSetInequalityCtrairt
```

```
NLPEvaluateHessianOfInequalityConstraint(P,c,v,H);
```

2.1.3 Equality Constraints

Equality constraints are handled exactly as the inequality constraints are handled, but without the upper and lower bounds. Equality constraints are added with the `NLPAddEqualityConstraint` or `NLPAddEqualityConstraint-ByString` routine, or can be built as the sum of a number of groups.

```
NLProblem P;
char name[]="Eq0";
double l,u;
int nv;
int *v;
double (*F)(int,double*,void*);
double (*dF)(int,int,double*,void*);
double (*ddF)(int,int,int,double*,void*);
void *data;
void (*freedata)(void*));
```

The first operation simply creates a copy of the problem:

```
NLProblem Q;

Q=NLCopyProblem(P);
```

The other transformations below change the problem, so a copy can be useful to compare results.

Next theresat1(ransfle)1a1(tha)1rescbs347(ftha)rressime(s347(7(b)-2tha)undsres)-1w(s)-(theres

In the limit of small penalty parameter μ, and minimizing w.r.t. both $\mathbf{v}$ and λ

group is of the form

$$\square \quad \frac{1}{s_i} g_i(N_i(\mathrm{v}) + \mathrm{a}_i.\mathrm{v} - b_i)$$

and second derivatives). It can be created by passing routines, or by way of a string.

```
NLGroupFunction g;
```

```
NLElementFunction ef;
int *vars;

N=NLCreateNonlinearElement(P,"type",ef,vars);
```

the vars

2.1.10 Error Handling

Most routines return a return code that indicates whether the operation was successful. If the routine creates or returns a data structure an invalid value is returned if the routine is not successful. In addition a simple error handling is also provided.

4.1 Using the SetObjective/AddConstraint routines

This shoi be fa clear. First we include the NLPAPI hea fiide14ide14o62()]TJ0-14.446Td[(declar

```
#include <NLPAPI.h>


 {
  NLPrem P;
  i4o62t v[3o62];
```

Then we create the problem, g it the name "H –

```
P=CreePrle
```

and change the names of the var (although these are the default names 14o28(yw)27(a)27(y)-1) and set the bounds on the variables.

```
NLPSleme(
NLPSpleuno62(s()-1P,5,4.5

NLPSleme(
NLPSpleuno62(s()-1P,5,4.5

NLPSleme(
NLPSpleuno62(s()-1P,,5.);
```

Next we specify thejectivo28(e)-43-1functionhs caseendsll three pro

```
v[0]=0;v[1]=1;v[2]=2;
NLPAddInequalityConstraintByString(P,"I1",0.,1.e40,3,v,
    "[x1,x2,x3]","48-x1**2-x2**2-x3**2");
```

And that' all there is to it.

If instead we had subroutines to evaluate the objective (o and do and ddo to evaluate the first and second derivatives) and constraint (c, dc and ddc) the code would change slightly:

```
v[0]=0;v[1]=1;v[2]=2;
NLPSetObjectiveByString(P,"Obj",3,v,o,do,ddo,NULL,NULL):
NLPAddInequalityConstraintByString(P,"I1",0.,1.e40,3,v,
    c,dc,ddc,NULL,NULL);
```

4.2 Using the AddGroup routines

This approach uses the same definition as the SIF file for HS65 would. The objective consists of three groups with the same group function, none has any nonlinear elements, and the first has no constant part to the linear element.

So we will define one `LNGroupFunction`

The main program and declarations –

```
int main(int argc,char *argv[])
 {
  NLProblem P;
  NLGrour(*)iunction g;
```

```
  rc=NLVSetC(a,1,-1.);
  rc=NLVSetC(a,2,0.);
  rc=NLPSetObjectiveGroupA(P,group,a);
  NLFreeVector(a);
```

(x_1

```
constraint=NLPAddNonlinearInequalityConstraint(P,"C1");
rc=NLPSetInequalityConstraintB(P,constraint,-48.);
v[0]=0;
ne=NLCreateNonlinearElement(P,"Sq1",f,v);
element=NLPAddNonlinearElementToInequalityConstraint
                                    (P,constraint,-1.,ne);
NLFreeNonlinearElement(P,ne);

v[0]=1;
ne=NLCreateNonlinearElement(P,"Sq2",f,v);
element=NLPAddNonlinearElementToInequalityConstraint
                                    (P,constraint,-1.,ne);
NLFreeNonlinearElement(P,ne);

v[0]=2;
ne=NLCreateNonlinearElement(P,"Sq3",f,v);
element=NLPAddNonlinearElementToInequalityConstraint
                                    (P,constraint,-1.,ne);
NLFreeNonlinearElement(P,ne);
```

4.3 Invoking LANCELOT to solve the problem

No matter which method was used to define the problem, the invocation of LANCELOT (Oh great and powerful LANCELOT, we pray that you find a solution ...) is the same. First we call the constructor for the NLLancelot object, then set the initial guess and ask for the minimization to be performed.

```
Lan=NLCreateLancelot();

x0[0]=-5.;
x0[1]=5.;
x0[2]=0.;
rc=LNMinimize(Lan,P,x0,(double*)NULL,x);
```

The rest of the example simply prints the solution

```
printf("Solution is (");
for(i=0;i<3;i++)
```

```
  {
   if(i>0)printf(",");
   printf("%lf",x[i]);
  }
 printf(")\n");
```

and any errors that may have occured. I've embedded a couple, just to be

```
            C1                0.00000000000000D+00
Solution is (3.650460,3.650460,4.620420)
There were 0 errors
```

5 Reference

5.1 Nonlinear Optimization Problems

5.1.1 The Objective

`NLPSetObjective`

5.4 Matrices

5.6.2 The list of groups

5.6.3 The nonlinear elements (of each group)

LNSetScalings

NLClearErrors

Purpose

Clears all errors.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
void NLClearErrors();
```

Description

This routine clears the error stack.

NLCreateElementFunction

Purpose

Allocates and initializes an NLElementFunction data structure.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
    NLElementFunctionNLCreateElementFunction(NLProblem P,char *type,int
n,NLMatrix R,double (*f)(int,double*,void*),double (*df)(int,int,double*,void*),double
(*dd f)(int,int,int,double*,void*),void *data,void (*freedata)(void*));
F=NLCreateElementFunction(P,type,n,R,f,df,ddf,datafreeData);
```

`NLElementFunction` *F*	The element function.
`NLProblem` *P*	The problem.
`char` **type*	The type given to the new element function.
`int`	

NLCreateGroupFunction

Purpose

goes to zero. References may be added using the `LNRefGroupFunction` subroutine (page 212).

Errors

Errors return (NLGroupFunction)NULL.

Message	Severity

NLCreateSparseMatrix

Purpose

Allocates and initializes an NLMatrix data structure of a given size.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
A=NLCreateSparseMatrix(n,m);
```

`NLMatrix`	*A*	The matrix.
`int`	*n*	The number of rows in the matrix.
`int`	*m*	

NLCreateMatrixWithData

Purpose

Message	Severity
”Number of rows (argument 1) is negative %d”	12
”Number of columns (argument 2) is negative %d”	12
”Pointer to data (argument 3) is NULL”	4

NLCreateProblem

Purpose

Allocates and initializes an NLProblem data staucture.

Library

`libNLPAPI.a`

C Syntax

NLCreateVector

Purpose

Allocates and initializes an NLVector data structure of a given length.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
v=NLCreateVector(n);
```

NLVector	*v*	The vector.
int	*n*	The length of the vector.

Descrip2i

Thread(at)1e42((NL)109(V)28(e)-1(cto)1(r)-42(deat)1ar)-42((structud)1(e)3431(a)1ar
(tse)-65(mav)27yeboecvwratthen

NLCreateVectorWithFullData

Purpose

Allocates and initializes an NLVector data structure of a given length.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
v=NLCreateVectorWithFullData(n,vl);
```

`NLVector`	*v*	The vector.
`int`	*n*	The length of the vector.
`double`	**vl*	The values of the coordinates.

Description

The routine This routine, `NLCreateVectorWithFullData` returns an NLVector data structure of a given length and coordinates. The vector returned has all coordinates marked as non-zero.

The coordinates may be changed using the `LNVSetC` routine (page 245). Zero vectors and sparse vectors can be created with the `NLCreateVector` (page 49) and `NLCreateVectorWithSparseData` (page 52) subroutines.

Note that the coordinates and values are copied out of the *vl* array, so subsequent changes to *vl* will not affect the vector.

Message	Severity
”Length of Vector %d (argument 1) is Illegal. Must be positive.”	12

NLCreateVectorWithSparseData

Purpose

Allocates and initializes an NLVector data structure of a given length.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
v=NLCreateVectorWithSparseData(n,nz,el,vl);
```

`NLVector`	*v*	The vector.
`int`	*n*	The length of the vector.
`int`	*nz*	The number of non-zeros in the vector.
`int`	**el*	

NLCreateDenseWrappedVector

Purpose

Allocates and initializes an NLVector data structure of a given length with data at a given storage location.

Library

`libNLPAPI.a`

C Syntax

NLEEvalDer

Purpose

Evaluates the derivative n0f an NLElementFunction.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
f=NLEEvalDer(F,
```

Purpose

Evaluates the second derivative of an NLElementFunction.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
f=NLEEvalSecDer(F,i,j,n,x);
```

double	*f*	The value of the second derivative.
NLElementFunction	*F*	The element function.
int	*i*	The first variable.
int	*j*	The second variable.
int	*n*	The number of coordinates in the point.
double	*x	The point.

Dr3.26bL#

Tiesrout(in)1hernt(the)-09(Ev)54(a)1(lhe)-09((of)-09((the)-09((see)-1io)1(de)-09((deriv)55((tiv)8(

NLEGetDimension

NLFreeGroupFunction

Purpose

Frees the storage associated with an NLGroupFunction data structure.

Library

lib6o

C Syntax

```
#include <NLo
void NLoeeGroupoiono     G);
```

NLoouponction *G* The group function.

Desiption

NLFreeLancelot

Purpose

Releases storage associated with an NLLancelot data structure.

Library

libNLPAPI.a

NLFreeNonlinearElement

Purpose

Frees the storage associated with an NLNonlinearElement data structure.

Library

`libNLPAPI.a`

C Syntax

```
#ele#eludage514(<NLPA)1LPAPIt
```

NLFreeProblem

Purpose

NLFreeVector

Purpose

Frees the storage associated with an NLVector data structure.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
```

NLGEval

NLGEvalSecDer

Purpose

Evaluates the second derivative of an NLGroupFunction.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
g=NLGEvalSecDer(G,x);
```

`double`	*g*	The value of the second derivative.
`NLGroupFunction`	*G*	The group function.
`double`	*x*	The point.

LNGetCheckDerivatives

Purpose

Gets the parameter controlling how Lancelot test derivatives.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
```

flag

LNGetConstraintAccuracy

Purpose

Gets the parameter controlling how accurately Lancelot solves constraints.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
```

NLGetErrorLine

Purpose

Returns the statement at whi4etuht ntrot

Syr

<NLt

NLGetErrorRoutine

Purpose

Returns the name of the routine that issued an error.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
routine=NLGetErrorRou7T17oF.50h>
```

NLGetErrorSev

Purpose

Returns the severity of an error.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
sev=NLGetErrorSev(i/F2124]TJ0-87.8.1-149.9Td[(#in1(]TJ/F3411.955Tf93.44110Td[(/ev)
```

LNGetFiestConsteaintAccueacy

Purpose

Gets the parameter controlling the initial accuracy Lancelot uses for the constraints.

Libeary

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
acc=LNGetFirstConstraintAccuracy(Lan
```

LNGetFirstGradientAccuracy

Purpose

Gets the parameter controlling the initial accuracy for the gradients.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
acc=LNGetFirstGradientAccuracy(Lan);
```

`double`	*acc*	The accuracy.
`NLLancelot`	*Lan*	The solver.

LNGetInitialPenalty

Purpose

Gets the parameter controlling the initial penalty.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
penalty=LNGetInitialPenalty(Lan);
```

double	*penalty*	The penalty.
NLLancelot	*Lan*	The solver

Description

The routine LNGetInitialPenalty gets the parameter controlling the penalty. The default value is 0. The SPEC.SPC file entry this sets is INITIAL-PENALTY-PARAMETER

Errors

_QE2.40712-206F34Td[Meseia Sev Errors return DBL

LNGetJi□yTuneTolerance

Purpose

Gets the parameter controlling the “Ji□ytune Tolerance”. NOTE: this re-
quie9514”(a9514”(mo)-n)2d4Jied95147(v)28(ersi(co514”(of95147(L))-1(anlote”.)]TJ/F2211.955Tf0

”Schnabel-Eskow preconditioned”
SCHNABEL-ESKOW-PRECONDITIONED-CG-SOLVER-USED

”Users preconditioned”

precbdi(ne)-1(d")]TJ/F2011.955Tf13.92215-14.194Td1(ON)1(-P)CT(VE)MO8dEVEL1(ON)T

LNGetMaximumNumberOfIterations

Obtain the value of the INTEGER parameter controlling how long Lancelot runs.
iter

LNGetPenaltyBound

Purpose

Gets the parameter controlling the bound on the penalty Lancelot uses.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
void penalty=LNGetPenaltyBound(Lan);
```

double	*penalty*	The bound.
NLLancelot	*Lan*	The solver.

Description

The routine LNGetPenaltyBound gets the ameter controlling the bound on the penalty Lancelot uses. The default value is 0.1. The SPEC.SP entry this sets is DECREASE-PENALTY-PARAMETER-UNTIL.

Errorp45an

LNGetPrintLevel

Purpose

Gets the parameter controlling how much output Lancelot produces.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
level=LNGetPrintLevel(Lan
```

(i)1((t)1((t)1r(t)1(e)]TJ/342 11.955 Tf104.5830 0 Td[()50(ans)]TJ/F02 11.955 Tf19.2890 0 Td[];t

LNGetPrintS/F22tart

Purpose

Gets th5Pur4(part)1metersn327(roi)1lling5Pur4(wth5P)1(nPur4(Lt)1nch5P)1loisS/F22tars5Pur4(

ici<NLi.h>e

LNGetPrintStop

Purpose

Gets the parameter controlling when Lancelot stops printing.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
iter=LNGetPrintStop(Lan);
```

int	*iter*	
NLLancelot	*Lan*	Nhe solver.

Description

Nhe routine LNGetPrintStop

LNGetRequireExactCauchyPoint

Purpose

LNGetSaveDataEvery

Purpose

Gets the parameter controlling how often Lancelot saves data.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
iter=LNGetSaveDataEvery(Lan);
```

int	*iter*	
NLLancelot	*Lan*	The solver.

Description

The routine LNGetSaveDataEvery

LNGetScalings

Purpose

LNGetSolveBQPAccurately

Purpose

LNGetStopOnBadDerivatives

Purpose

Gets the parameter controlling how Lancelot deads with bad derivatives.

Library

`libNLPAPI.a`

C Syntax

```
#incdude <NLPAPI.h>
```

LNGetTrustRegionRadius

Purpose

Gets the parameter controlling the radius of the trust region.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
radius=LNGetTrustRegionRadius(Lan);

    double      radius
```

LNGetUseExactFirstDerivatives

Purpose

Gets the parameter controlling how Lancelot gets derivatives.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
flag=LNGetUseExactFirstDerivatives(Lan);
```

`int`	*flag*	What to do
`NLLancelot`	*Lan*	

LNGetUseExactSecondDerivatives

Purpose

NLMGetElement

Purpose

Returns an element of an NLMatrix.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
aij=NLMGetElement(A,i,j);
```

double	*aij*	The element of the matrix.
NLMatrix	*A*	The matrix.
int	*i*	The row index of the element.
int	*j*	The column index of the element.

Description

This routine returns the specified element of the matrix. This is set when the matrix is created, or with the LNMSetElement routine (page 104).

Errors

Errors return DBL

NLMGetNumberOfCols

Purpose

NLMGetNumberOfRows

Purpose

Returns the number of rows in an NLMatrix.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
n=NLMGetNumberOfRows(A);
```

`int`	*n*	The number of rows.
`NLMatrix`	*A*	The matrix.

Description

NLMSetElement

Purpose

Changes the value of an element of an NLMatrix.

NLMIncrementElement

Purpose

Increments the value of an element of an NLMatrix.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc=NLMIncrementElement(A,i,j,aij);
```

`int`	*rc*	The return code.
`NLMatrix`	*A*	The matrix.
`int`	*i*	The row index of the element.
`int`	*j*	The column index of the element.
`double`	*aij*	The increment element of the matrix.

Description

This routine changes the specified element of the matrix, by adding the specified increment. If the matrix is sparse, and the element does not have a value, the value is set to the increment.

Errors

Errors return 0, with no changes to the matrix. Normal execution returns 1.

LNMaximize and LNMaximizeDLL

Purpose

Allocates and initializes an NLLancelot data structure.

Library

libNLPAPI.a

C Syntax

#include <NLPAPI.h>
c=LNMaximize(*Lan*,*P*,*x0*,*z0*,*l0*,*x*
c=LNMaximizeDLL(*Lan*,*P*,*x0*,*z0*,*l0*,*x*

int *rc* The r341173.434tslocatedF2211.950Tf-173.64704.7186Td[(c)]TJ/FPILa(l

LNMinimize

Purpose

Allocates and initializes an NLLancelot data structure.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
c=LNMinimize(Lan,P,x0,z0,l0,x);
```

int *rc* The return code.

NLNEGetElementDimension

Purpose

Returns the number of element variables for a nonlinear element.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
n=NL64625e46tEle46menns   P,ne);
    in                  n
```

NLEGetRangeXForm

Purpose

NLPAddLinearInequalityConstraint

NLPAddMinMaxConstraint

THIS IS AN EXTENSTION TO VANILLA LANCELOT
Purpose

NLPAddNonlinearElementToGroup

Purpose

Adds an empty nonlinear element to a group.

Library

`libNLPAPI.a`

NLPAddNonlinearElementToObjectiveGroup

NLPAddNcnlinearElementTcInequalityCcnstraint

Purpose

Adds an empty nonlinear element to an inequality ccnstraint.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
e=NLPAddNcnlinearElen38nx
```

NLPAddNonlinearElementToMinMaxConstraint

THIS IS AN EXTENSTION TO VANILLA LANCELOT
Purpose

NLPAddNonlinearInequalityConstraint

Purpose

Adds a nonlinear inequality constraint.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
g=NLPTJ/NonlinearInequalityConstraint(P,name);
```

`int`	*g*	The index of the new group.

NLPGetElementFunction

Purpose

NLPGetElementFunctionOfGroup

Purpose

NLPGetElementIndexIntoWhole

Purpose

NLPGetElementNumberOfUnknowns

Purpose

Returns the number of unknowns of a nonlinear element function.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
n=NLPGetElementNumberOfUnknowns(P,group,element
```

NLPGetElementRangeTransformation

Purpose

Returns the range transformation of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
xfrm=NLPGetElementRangeTransformation(P,element);
```

`NLMatrix`	*xfrm*	The range transformation.
`NLProblem`	*P*	The problem.
`int`	*element*	The index of the nonlinear element.

Description

NLPGetElementRangeTransformationOfGroup

Purpose

Returns the range transformation of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
f
```

NLPGetElementTypeName

NLPGetElementWeight

Purpose

Returns the weight of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
weight=NLPGetElementWeight(P,group,element);
```

`double`	*weight*	The weight.
`NLProblem`	*P*	The problem.
`int`	*group*	The index of the group.
`int`	*element*	The number of the nonlinear element.

Description

NLPGetEqualityConstraintGroupNumber

Purpose

Returns the index of the group representing an equality constraint.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
```

NLPGetGroupA

Purpose

NLPGetGroupB

Purpose

Gets the constant part of the linear element of a group.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
b=NLPGetGroupB(P,group);
```

`NLVector`	*b*	The constant.
`NLProblem`	*P*	The problem.
`int`	*group*	The index of the group.

Description

This routine returns the constant part of the lineear element of a group. This can be queried with the `NLPGetGroupB` (page 138) subroutine.

Errors

Errors return DBL_QNAN.

Message	Severity
"Problem (argument 1) is NULL"	12
"Group %d is illegal (argument 2). Must be in range 0 to %d"	12

NLPGetGroupFunction

Purpose

Gets the group function of a group.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
g=NLPGetGroupFunction(
```

NLPGetGroupScale

Purpose

Gets the scale factor of a group.

Library

libNLPAPI.ancd1(LP)1(LPA)1(Ph>1(ary)34J/F2011.955T140-51.647s1(ary)]TJ/F2011.954

NLPGetGroupType

Purpose

Returns the index of a type of g(ns)oup.

Linsary

`lins`

C Syntax

```
#ins <NLns
type=NnsGetGrpe(     P,i);
```

`int`	*type*	The type.
`NLns`	*P*	The pr
`int`	*i*	The index of the type.

Descrins

This routine rnsturns the index of a group type. Gr types are assigned with the `NLnsroupivns` (page 116) subroutine. A new type name is assigned a number and the name is stored.

Errors

E311.1.9J8(yp)(name)-326(is)-327 l.to5452.11.9(E31Me(sig)1ge)-243(17594(Sevum)2eri(yp)-28

NLPGetInequalityConstraintLowerBound

Purpose

Gets the lower bound for an inequality constraint.

Library

NLPGetMinMaxConstraintGroupNumber

THIS IS AN EXTENSTION TO VANILLA LANCELOT
Purpose

NLPGetNumberOfElementTypes

/F20 11.955 Tf-87.2090 140.460 Td[(NA)1(ur)1obldmx

Purpose

Returns the number of distinct types of nonlinear elements.

Library

`libNLPAPI.a`

C Syntax

```
#include <L23..PAPIh>x
```

NLPGetNumberOfElements

Purpose

Returns the total number of nonlinear elements for a problem.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
n=NLPGetNumberOfElements(P);
```

`int`	*n*	The number of elements.
`NLProblem`	*P*	The problem.

NLPGetNumberOfElementsE

Purpose

Returns the tota3711eNu-322(b-32erns)-32ootet(ns)-32inhe the

NLPGetNumberOfElementsI

Purpose

Returns the total number of nonlinear elements in the inequality constraints.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
n=NLPGetNumberOfElementsI(P);
```

24633(The)Dnmsary Thein8(rf)3utual4(in8(r)1(nn74(in8(Nu)1hl4(i

int	*n*	The number of elements.

NLPGetNumberOfElementsInGroup

Purpose

NLPGetNumberOfElementsM

Purpose

NLPGetNumberOfElementsO

Purpose

Returns the total number of nonlinear elements in the Objective.

Library

`libNLPAPI.a`

C Syntax

```
#inLlude <NLPAPI.h>
```

n

NLPGetNumberOfEqualityConstraints

Purpose

Returns the number of equality constraints in a problem.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
n=NLPGetNumberOfEqualityConstraints(P);

    int        n
```

NLPGetNumberOfGr4upsInObjective

Purpose

Returns the number of groups in the 4bjective of a problem.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
n=NLPGetNumberOfGroupsInObjective(P);
```

`int`	*n*	The number of groups.
`NLProblem`	*P*	The problem.

Description

This routine returns the current number of groups in the 4bjective 4f a problem. Each time a group is added to the objective this number increases. It never decreases.

Errors

Errors return -1.

Message	Severity

NLPGetNumberOfInequalityConstraints

Purpose

Returns the number of inequality constraints in a problem.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
n=NLPGetNumberOfInequalityConstraints(P);
```

`int`	*n*	The number of constraints.
`NLProblem`	*P*	The problem.

NLPGetNumberOfVariablei

Purpose

NLPGetObjectiveGroupNumber

Purpose

Returns the index of a group in the objective of a problem.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
g=NLPGetObjectiveGroupNumber(P,i);
```

`int`	*g*	The index of the group.
`NLProblem`	*P*	The problem.
`int`	*i*	Which group.

Description

NLPGetTypeOfElement

Purpose

Returns the type name of a nonlinear element.

Library

NLPGetTypeOfGroup

Purpose

Returns the type of a group.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
type=NLPGetTypeOfGroup(P,i);
```

`char`	*`name`	The type of the group.
`NLProblem`	`P`	The problem.
`int`	`i`	The number of the group type.

Description

NLPGetVariableName

Purpose

Returns the name of a variable.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
name=NLPGetVariableName(P,i);
```

`char`	*`*name`*	The problem name.
`NLProblem`	*`P`*	The problem.
`int`	*`i`*	The number of the variable.

Description

This routine returns the name of a variable. If the variable has not yet been given a name, the default is “Xxxxxxxx”, where ’x’ is a hex digit 0-9A-F. This is create with the C-format “X

Note: The user should not free the returned string.

Errors

Errors return (char*)NULL.

Message	Severity
”Problem (argument 1) is NULL”	12
”Variable number %d (argument 2) is illegal. Must be in range 0 to %d”	12

NLPGetVariableScale

NLPIsElementFunctionSet

Purpose

Queries whether the weight of a nonlinear element has been set.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
ans=NLPIsElementFunctionSet(P,group,element);
```

NLPIsGroupFunctionSet

Purpose

Queries whether the group function of a group has been set.

Library

libNLPAPI.a

C Syntax

NLPIsLowerSimpleBoundSet

Purpose

Queries whether a lower bound has been set on a variable.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
ans=NLPIsLowerSimpleBoundSet(P,var);
```

`int`	*ans*	The answer, 1==Set, 0=Not Set.
`NLProblem`	*P*	The problem.
`int`	*var*	The index of the variable.

Description

NLPIsUpperSimpleBoundSet

Purpose

Queries whether a upper bound has been set on a variable.

Library

`libNLPAPI.a`

C Syntax

NLPSetElementFunction

Purpose

Changes the nonlinear element function of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

NLPSetElementFunctionWithRange

Purpose

Changes the nonlinear element function of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

NLPSetElementWeight

Purpose

Changes the weight of a nonlinear element.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetElementWeight(P,group,element,
```

NLPSetEqualityConstraintA

Purpose

Sets the linear part of the linear element of an equality constraint.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetEqualityConstraintA(P,constraint,a);
```

`int`	*rc*	The return code.
`NLProblem`	*P*	The problem.
`int`	*constraint*	The index of the constraint.
`NLVector`		

NLPSetGroupA

Purpose

Sets the linear part of the linear element of a group.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetGroupA(P,group,a);
```

`int`	*rc*	The return code.
`NLProblem`	*P*	The problem.
`int`	*group*	

NLPSetGroupFunction

Purpose

Sets the group function of a group.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetGroupFunction(P,group,g);
```

int	*rc*	The return ccde.
NLProblem		

NLPSetGroupScale

Purpose

Sets the scale factor of a group.

Library

libNLPAPI.a

C Syntax

#include <NLPAPI.h>
rc=NLPSetGroupScale(*P*,*group*,*s*);*)(c)]F2011.955f0-239.9r(he)-327retu5L*
NAGrbalmntF3411.955f.1527.32d[Pcgroup
bcent/F3411.955Tf6.1527.32Td[sp
otde: heefini(t)1ionetheue unction(a)]TJ/F2211.955Tf-172.450-
ndehe

NLPSetInequalityConstraintA

Purpose

Sets the linear part of the linear element of an inequality constraint.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetInequalityConstraintA(
```

NLPSetInequalityConstraintB

Purpose

Sets the constant part of the linear element of an inequality constraint.

Library

libf.PAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=f.PSetInequalityConstraintB(P,
```

NLPSetInequalityConstraintBounds

Purpose

Sets the bounds on an inequality constraint.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetInequalityConstraintBounds(P,c,l,u);
```

`int`	*rc*	The return code.
`NLProblem`	*P*	The problem.
`int`	*c*	Which constraint.
`double`	*l*	The lower bound.
`double`	*u*	The upper bound.

Description

This routine sets the bounds on the inequality constraint. This can be queried with the `NLPGe2Td0nequalityConstraintUpperBoun`(page 147) and `NLPGe2-InequalityConstraintLowerBound` (page 146) subroutine. The bounds can also be se2 one a2 a time using the `NLPSe2Td0nequalityConstraintUpper-Bound` (page 195) and `NLPSe2Td0nequalityConstraintLowerBoun`(page 194) routines.

Initially the bounds are $-\infty$ to ∞. (A value of $-1.e20$ is onsidered by Lancelot to be infinity.) Obviously this is no onstraint at all unless the bounds are se2.

Errors

Errors return 0 and make no changes to the problem. Normal execution returns 1.

NLPSetInequalityConstraintLowerBound

Purpose

Sete the lower bound on an inequality constraint.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetInequalityConstraintLowerBound(P
```

NLPSetInequalityConstraintUpperB8und

Purpose

Sets the upper b8und on an inequality constraint.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
```

rc

NLPGetLowerM6nMaxBound

THIS IS AN EXTENSTION TO VANILLA LANCELOT
Purpose

NLPSetLowerSimpleBound

Purpose

Sets the lower bound on a variable.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetLowerSimpleBound(P,var,l);
```

int	*rc*	The return code.
NLProblem	*P*	The problem.
int	*var*	Which variable.

NLPSetMinMaxBounds

THIS IS AN EXTENSTION TO VANILLA LANCELOT

Purpose

Sets the bounds on the min-max variable.

NLPSetMinMaxConstraintB

THIS IS AN EDTENSTION TO VANILLA LANCELOT

Purpose

Sets the constant part of the linear element of an minmax constraint.

Library

`libNLPAPI.a`

NLPSetObjectiveGroupA

Purpose

Sets the linear part of the linear element of a group in the objective.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetObjectiveGroupA(P,group,a);
```

`int`	*rc*	The return code.
`NLProblem`	*P*	The problem.
`int`	*group*	The index of the group.
`NLVector`	*a*	The linear element.

Description

This routine sets the linear part of the lineear element of a group in the objective.

Errors

Errors return 0 and make no changes to the problem. Normal execution returns 1.

NLPSetObjectiveGroupB

Purpose

Sets the constant part of the linear element of a group in the objective.

Library

`libNLPAPI.a`

C Syntax

NLPSetO,jectiveGroupFunction

Purpose

NLPSetObjectiveGroupScale

SetObjectiveGroupScale1((e)]TJ/342 11.955 Tf166.1020 0 Td[Pe

Purpose

Sets tSa5roufroua5funcGtofa5rot

Syi

udG<NLi

NLPGetUpperMinMaxBound

THIS IS AN EXTENSTION TO VANILLA LANCELOT

Purpose

Gets the upper bound on the min-max variable.

Library

libNLPAPI.a

NLPSetVariableScale

Purpose

retur51hnwitho51hutprintingt51hheproi
pe51hr51hfo51hrmanc51he51h.-277(Thi)1(ai-276(structur51her)-1.)-418[The o51hut--att51hertomimicthe
51hPr51hobi(

NLRefElementFunction

Purpose

Registers a reference to an NLElementFunction data structure.

Library

`libNLPAPI.a`

NLRefGroupFunction

Purpose

Registers a reference to an NLGroupFunction data structure.

L7brary

NLRefMatrix

Purpose

Registers a reference to an NLMatrix data structure.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
void NLRefMatrix(ATd[d#iRetrixACrur4917.h
```

NLRefNonlinearElement

Purpose

NLRefVector

Purpose

Registers a reference to an NLVector data structure.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
void NLRefVector(v);
```

`NLVector` *v* The vector.

Description

The NLVector data structure uses reference counting. This routine should be used to indicate that a vector is needed by another data structure. The vector will not be deleted until the same data structure indicates that it no longer needed (for example, when the data structure itself is deleted). This works as long as the `NLFreeVector` subroutine (page 66) is used to delete the vector, and is only used once per added reference.

Errors

Severity 4 errors return without changing the vector.

Message	Severity
"Pointer to Vector (argument 1) is NULL"	4

LNSetCheckDerivatives

Purpose

Sets the parameter controlling how Lancelot testtests derivatives.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
```

rc

LNSetConstraintAccuracy

Purpose

Sets the parameter controlling how arcurar constrarts ar solved.

Library

libNLPAPI.a

C Syntar

```
#include <NLPAPI.h>
r =LNSetConstraintAccuracy(L  ,limit);

    int         r
```

LNSetFirstConstraintAccuracy

Purpose

Sets the parameter controlling the initial accuracy Lancelot uses for the constraints.

LNSetFrstGradentAccuracy

Purpose

Sets the parameter controlling the initial accuracy for the grad-ents.

Lbrary

```
l-bNLPAPI.a
```

C Syntax

```
#-nclude <NLPAPI.h>
```

rc

LNSetGradientAccuracy

Purpose

Sets the parameter controlling the accuracy for the gradients.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=LNSetGradientAccuracy(Lan,limit);
```

int	*rc*	The return code.
NLLancelot	*Lan*	The solver.
double	*limit*	The accuracy.

Description

The routine LNSetGradientAccuracy sets the parameter controlling the accuracy for the gradients. The default value is 0.00001. The SPEC.SPC file entry this corresponds to is GRADIENT-ACCURACY-REQUIRED.

Errors

Errors return 0, normal execution returns 1.

Message	Severity

LNSetInitialPenalty

Purpose

Sets the parameter controlling the initial penalty.

Library

libNLPAPI.a

C Syntax

#include <NLPAPI.h>
rc=LNSetInitialPenalty(*Lan*,*penalty*);

int *rc* *Lan* *penalty*Sets,

LNSetJi□yTuneTolerance

”Modified MA27 preconditioned”
MODIFIED-MA27-PRECONDITIONED-CG-SOLVER-USED

”Schnabel-Eskow preconditioned”
SCHNABEL-ESKOW-PRECONDITIONED-CG-SOLVER-USED

”Users preconditioned”
USERS-PRECONDITIONED-CG-SOLVER-USED

”Bandsolver preconditioned”
BANDSOLVER-PRECONDITIONED-CG-SOLVER-USED

”Multifront”
MULTIFRONT-SOLVER-USED

”Direct modified”
DIRECT-MODIFIED-MULTIFRONTAL-SOLVER-USED

”CG method used”
CG-METHOD-USED

Errors

Errors return 0, normal execoISED

LNSetPenaltyBound

LNSetPrintEvery

Purpose

LNSetPrintLevel

Purpose

Sets the parameter controlling how much output Lancelot produces.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=LNSetPrintLevel(Lan,level);
    int          rc
```

Purdose

Sets the parameter controlling when Lancelot starts printing.

Library

libNLPAPI.a

C Syntax

fTdf]TdludTdh<N(L)1LPAPIh>x

se

LNSetSaveDataEvery

Purpose

Sets the parameter controlling how often Lancelot saves data.

Library

LNSetScalings

Purpose

Sets the parameter controlling how Lancelot uses scalings.

LNSetSolveBQPAccurately

Purpose

Sets the parameter controlling thesolution of theBQP.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=LNSetSolveBQPAccurately(Lan,choice);
```

int	*rc*	The return code.
NLLancelot	*Lan*	The solver.
int	*choice*	

Description

The routine LNSetSolveBQPAccurately sets the parameter controlling the solution of the BQP.The defaultis0.

The corresponding SPEC.SPC file entryis SOLVE-BQP-ACCURATELY.

Errors

LNSetTrustRegionRadius

Purpose

Sets the parameter controlling the radius of the trust region.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=LNSetTrustRegionRadius(Lan,radius);
```

int	*rc*	The return code.
NLLancelot	*Lan*	The solver.
double	*radius*	The radius.

Description

The routine LNT9555Tf-116tRegiionLt

LNSetTrustRegionType

Purpose

Sets the parameter controlling the type of trust region Lancelot uses.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc=LNSetTrustReaTf0onType(Lan,choice);
```

`int`	*rc*	The return code.
`NLLancelot`	*Lan*	The solver.
`char`	**choice*	Which type.

Description

The routine `LNSetTrustRegionType` sets the parameter controlling the type of trust reaTf0on Lancelot uses. LeaTfal v f1.927(t)1(rollingust)-33he11.955Tf5.8530259.594hoice

LNSetUseExactFirstDerivatives

Purpose

Sets the parameter controlling how Lancelot gets derivatives.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
```

rc

LNSetUseExactSecondDerivatives

Purpose

Sets the parameter controlling second derivatives.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
```

rc

NLVGetNonZero

Purpose

Returns the coordinate index of a non-zero coordinate in a vector.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
```

NLVGetNonZeroCoord

Purpose

NLVGetNumberOfNonZeros

Purpose

Returns the number of non-zero coordinates in a vector.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
```

n

NLVSetC

Purpose

NLVIncrementC

Purpose

Increments the specified coordinate of coordinates) of a vector.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=NLVIncrementC(v,i,c);
```

int *rc* The return code. [illegible]

NLPEvaluateInequalityConstraint, NLPEvaluateGradientOfInequalityConstraint, NL

Purpose

Return the value of the objective and its derivatives at the given point.

Library

`libNLPAPI.a`

NLPInvalidateGroupAndElementCaches

Purpose

Marks the cahced values for the element and group functions as invalid.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
NLPInvalidateGroupAndElementCaches(P);
```

`NLProblem` *P* The problem.

Description

This routine signals that the stored values for the element and group functions need to be recomputed.

Errors

Message	Severity
"Problem (argument 1) is NULL"	12