1.0.3 Inequality Constraints

NLPAddLinearI1(oL)1(in)1(ea)1(rI1(oL)1(in)1(ea).40Td[(74)]T/F2111.95Tf-224

1.2 Vectors

NLm209.1(Fr)1(eeV)1(ec)1(to)1(r)]T/F111.95Tf02

1.3 Matrices

1.4 GrouR Functions

1.5 Element Functions

1.5.1 Nonlinear Elements

1.5.3 The nonlinear elements (of each group)

1.6.3 Setting LANCELOT Parameters

NLCreateProblem

Purpose

Allocates and initializes an NLProblem data structure.

Library

`libNLPAPI.a`

C Syntax

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

P

NLRefProblem

Purpose

Releases storage associated with an NLProblem data structure.

Library

libNLPAPI.a

C Syntax

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

NLProblem *P* The problem.

Description

The routine NLRefProblem adds a reference to a problem. NLFreeProblem (page 15) removes one reference and releases the storage associated with the problem if the reference count is zero. This allows the p.ro.Q(lhucture./F18his)-354578.hishe

NLFreeProblem

NLPrintProblem

Purpose

Prints a NLProblem data structure.

Library

`libNLPAPI.a`

C Syntax

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

NLPrintProblemShort

Purpose

Prints a NLProblem data structure.

Library

libNLPAPI.a

NLPGetNumberOfVariables

Purpose

Returns the number of variables for a problem.

Library

`libNLPAPI.a`

C Syntax

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

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

NLPSetVariableScale

Purpose

Sets the scale factor of a variable.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetVariableScale(P,i,s);
```

i7Tfm *rc* The return code.

NLPGetVariableScale

Purpose

NLPSetVariableName

Purpose

Assig7s the name of a variable.

Library

`libNLPAPI.a`

C Sytax

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

`i7t`	*rc*	The return code.
`NLProblem`	*P*	The problem.
`i7t`	*i*	The number of the variable.
`char`	**name*	The problem name.

Description

This routine sets the name of a variable. This may be queried with the `NLPSetVariableName` subroutine (page 22). 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

A copy of the string is made. The copy is freed when the problem is freed.

NLPSetUpperSimpleBound

NLPGetUpperSimpleBPund

Purpose

Gets the upper bound Pn a variable.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
u=NLPGetUpperSimpleBound(P,var);
```

`dPuble`	*u*	The upper bound.
`NLProblem`	*P*	The problem.
`int`	*var*	Which variable.

Description

NLPIsUpperSimpleBoundSet

Purpose

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

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
ans=NLPIsUpperSimpleBoun50.t(P,var);
```

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

Description

This routines queries whether a upperupe58ound hasupe5827(een)-31(ns)7(set)-3o(een)-3aset vari

NLConvertToEqualityAndBoundsOnly

Purpose

Eliminates the inequality constraints from a Problem by introducing slack variables.

Library

C Syntax

libNLPAPI.aLibrary

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

NLProblem

is replaced by an equality constraint and simple bounds on the slack –

$$f(v) - s = l$$

NLCopyProblem

Purpose

Creates a copy of an NLProblem data structure.

LObrary

`l0bNLPAPI.a`

C Syntax

```
#0nclude <NLPAPI.h>
```

NLCreateAugmentedLagrangian

Purpose

Replaces the equality constraints in a Problem with a quadratic penalty and Lagrangian terms in the objective.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
NLCreateAugmentedLagrangian(P,mu,l, g,b,s);
```

`NLProblem`	*P*	The problem.
`double`	*mu*	The penalty parameter μ.
`double*`	*l*	The Lagrange multipliers λ_i

Description

The routine `NLCreateAugmentedLagrangian` takes a problem and replaces the equality constraints with a quadratic penalty function and lagrangian in the objective. That is, a problem

$$\text{minimize } O(\mathrm{v})$$

$$f_i(\mathrm{v}) = 0$$

is replaced by a problem with no equality constraints and objective –

$$\text{minimize } O(\mathrm{v}) + \frac{1}{2\mu}$$

NLSetLambdaAndMuInAugmentedLagrangian

NLEliminateFixedVariables

Purpose

For each variable whoie upper and lower simple bounds are identical, introduces a linear equality constraint and removes the bounds.

NLPSetObjective

of the problem variables. The subset is defined by way of the *nv*, and *v* arguments. When the objective is evaluated the routine *f* will be called.

double *f*(`int` *nv*,`double *`*x*,`void *`*data*);

The first argument to *f* will be *nv*. The second argument is an array *x*

NLPSetObjectiveByString

Purpose

Sets the objective to be a function defined by a string.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
NLPSetObjectiveByString(P
```

```
v[0]=1;v[1]=45;v[2]=0;
```

NLPAddGroupToObjective

Purpose

Adds a group th the objective function.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
g=NLPAddGroupToObjective(P,name,type);

    int        g
```

NLPAddNonlinearElementToObjectiveGroup

Purpose

Adds an empty nonlinear element to a group.

Library

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
```

NLPSetObjectiveGroupFunction

Purpose

Sets the group function of a group.

Library

libNLPAPI.a

NLPEvaluateObjective

Purpose

Evaluates the objective function.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
o=NLPEvaluateObjective(P,x);
```

double *o*	The value of the objective function.
NLP7oblem *P*	The p7oblem.
NLV(cti)1x*r*	The point (problem variables) at which to evaluate the objective.

Desctiiption

The routine NLPEvaluateObjective evaluatesc-483(the)-483(o)1(b)-55(jectiv)28(e)-483(at)-483(x
app7opriate values.

Errors

Messade.Error5Tf299.7780Td[(x)]TJ/F1811.955T664F1811

C Syntax

```
#include <NLPAPI.h>
rc=NLPEvaluateGradie-tOfObjective(P,x,g);
```

i-t *rc*

NLPEvaluateHessianOfObjective

NLPAddEqualityConstraint

Purpose

routines *f*, *df*, and *ddf* define the constraint function. These are scalar valued functions of a subset of the problem variables. The subset is defined by way of the *nv*, and *v* arguments. When the constraint is evaluated the routine *f*

NLPAddEqualityConstraintByString

Purpose

Adds an equality constraint defined by an expression in a string to a problem.

Library

`libNLPAPI.a`

```
int v[3];
int c;
```

NLPAddLinearEqualityConstraint

Purpose

Adds a linear equality constraint.

Library

NLPAddNonlinearElementToEqualityConstraint

Purpose

Adds an empty nonlinear element to an equality constraint.

Library

`libNLPAPI.a`

C Syntax

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,const0(c)aint,a);
```

`int`	*rc*	The return code.
`NLProblem`	*P*	The problem.
`int`	*const0(c)aint*	The index of the constraint.
`NLVector`	*a*	The linear element.

Description

This routine sets the linear part of the lineear element of an equality const0aint.

Errors

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

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

NLPEvaluateEqualityConstraint

Purpose

Evaluates an equality constraint.

Library

`libNLPAPI.a`

NLPEvaluateHessianOfEqualityConstraint

Purpose

`int` *c* — The number assigned to the new constraint.

`NLProblem` *P*

constraNLProblemT-196.6f-236.663-10.112Td[(NL)1(Pr)1(obl)1(em)]TJ/F2111.95

NLPAddInequalityConstraintByString

Purpose

Adds an inequality constraint defined by an expression in a string to a problem.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
c=NLPAddInequalityConstraintByString(P,name,l,u,nv,v,varlist,expr);
```

```
int c
```

45, and c the value of problem variable 0. The main restriction on the expression is that constants may *not* be specified using exponential notation

NLPAddNonlinearInequalityConstraint

Purpose

Adds a nonlinear inequality constraint.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
g=NLPAddNonlinearInequalityConstraint(P,name);
    int
```

NLPAddLinearInequalityConstraint

Purpose

Adds a linear inequality constraint.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
g=NLPAddLinearInequalityCo3straint(P,name,a,b);
```

int	*g*	The index of the new group.
NLProblem	*P*	The problem.
chai	**name*	The name of the new group.
double	**a*	The linear part of the linear element.
double	*b*	The constant part of the linear element.

Descriptio3

This routine adds a linear inequality constraint. The *name* of the group must be unique.

A trivial group is added, with no nonlinear element, and the given linear element. The constraint may be changed with the NLPSetGroupFunctio3 (page 187), NLPSetGroupScale (page 196), NLPSetGroupA (page 190), and NLPSetGroupB (page 193) routinei11.9) routinerou75ae9) routic-312(t)1nutii11(li)1(ishoul(li)1(iwit

NLPSetInequalityConstraintA

Purpose

NLPGetInequalityConstraintLowerBound

Purpose

Gets the lower bound for an inequality constraint.

Library

`libNLPAPI.a`

C Syntax

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

`double`	*l*	The lower bound.
`NLProblem`	*P*	The problem.
`int`	*c*	Which constraint.

Description

This routine returns the lower bound for the inequality constraint.

Initially the bound is $-\infty$. (A value of $-1.e20$ is considered by Lancelot to be infinity.)

Errors

Errors return DBL_

NLPSetInequalityConstraintLowerBound

Purpose

Sets the lower bound on an inequality constraint.

NLPGetInequalityConstraintUpperBound

Purpose

Gets the upper bound for an inequality constraint.

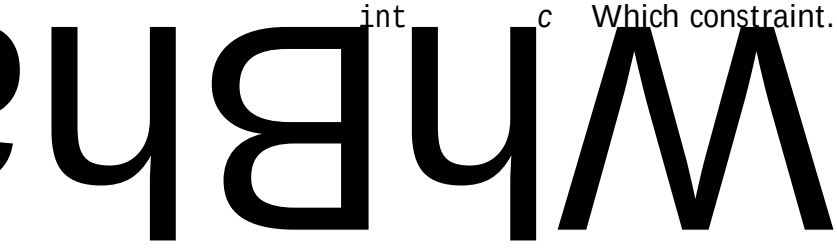
Library

`libNLPAPI.a`

C Syntax

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

`double`	*u*	The upper bound.
`NLProblem`	*P*	The problem.
`int`	*c*	Which constraint.



NLPSetInequalityConstraintUpperBound

Purpose

Sets the upper bound on an inequality constraint.

Library

`libNLPAPI.a`

C Syntax

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

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

Description

This routine sets the upper bound on the inequality constraint. This can]TJ/F1e queried with the NLPPGetInequalityConstraintUpperBound (page 81) routine. The bound can also be set at the same time using the NLPPSet- Inequality (page 83) routine.

Initially the bound is ∞ NLP. (A value of 1

NLPGetInequalityConstraintGroupNumber

Purpose

Returns the index of the group representing an inequality constraint.

Library

`libNLPAPI.a`

C Syntax

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

NLPEvaluateInequalityConstraint

NLPEvaluateHessianOfInequalityConstraint

Purpose

Evaluates the Hessian of an inequality constraint.

NLError

NLGetNErrors

Purpose

Returns the number of errors that have been flagged.

Library

`libNLPAPI.a`

C Syntax

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

`int` *n* The number of errors.

This routine returns the number of errors that have been set.

NLGetErrorSev

Purpose

Returns the severity of an error.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
sev=NLGetErrorSev(i);
```

int	*sev*	The severity.
int	*i*	i(ns)-32rol1.9.1(ns)-327iThe 14.4-2110330TdD1.955s542c(br)-(i)pt(i)onse

NLGetErrorLine

Purpose

NLGetErrorFile

Purpose

Returns the file containing the source code from which an error was issued.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
file=NLGetErrorFile(i);
```

`char`	*`file`	The file.
`int`	`i`	Which error.

NLClearErrors

Purpose

Clears all errors.

Library

libNLPAPI.a

C Syntax

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

Description

This routine clears the error stack.

NLCreateVector

Purpose

Allocate7 and initialize7 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.

Decription

The routine `NLCreateVector` allocates an NLVector data structure and initialize7 it to a vector of given length with no non-zero coordinates. The coordinates may be changed with the `NLVSetC` routine (page 108). Vectors with supplied coordinate values can be created with the `NLCreateVectorWithSpar7eData` and `NLCreateVectorWithFullData` subroutine7 (page7 98 and 100).

The NLVector data structure use7 reference counting. The data structure should be deleted using the

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
```

12 errors return (NLVector)NULL.

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

NLCreateVectorWithFullData

Purpose

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

Message	Severity
"Length of Vector %d (argument 1) is Illegal. Must be positive."	12
"The pointer to the array of coordinates (argument 2) is NULL"	4
"Out of memory, trying to allocate %d bytes"	12

NLVector NLCreateDenseWrappedVector(int n,double *data)

NLFreeVector

Purpose

Frees the storage associated with an NLVector data structure.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
void NLFreeVector(
```

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.

Descrim..4ene

The NLVectoristucturr referinegTisr .74eu(ince)-927shoucd
tectro dataucture.-529[(The)]TJ0-14.460Td[vectorwillrtelrsamceuctura[1eoth45etedoth45e(or)f27

NLPrintVector

Purpose

Prints an NLVector.

Library

libNLPAPI.a

C Syntax

NLVGetNC

Purpose

NLVSetC

Purpose

Sets the specified coordinate of a vector.

Library

libNLPAPI.a

C Syntax

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

int	*rc*	The return code.
NLVector	*v*	The vector.
int	*i*	Which coordinate to return.
double	*c*	The value of the coordinate.

Description

This routine changes the value of a coordinate of a vector. The index *i* must be nonnegative and less than the number of coordinates (NLVGetNC). If the

NLCopyVector

Purpose

Returns a copy of a vector.

NLVIncrementC

[illegible]

r(uuutinee)218(a)-1ddseaat(u)2127acebinauutee(uufe)218(a)r

Purpose

Increments a couuebinatme auue.e

Syna

co 1e [illegible]

NLVInnerProd

Purpose

Returns the inner product (Euclidean) of two vectors.

Library

libNLPAPI.a

C Syntax

#include m21.T.S(h>31(ta)35x)]T/F2111.94]T50-21p31(ta)1(x)]T/F2115.9 0.95Tf=d[(li[(N

NLVPlusV

Purpose

Returns the sum of two vectors (actually the daxpy).

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
p=NLVPlusV(u,v,a);
```

`NLVector`	*u*	The first vector.
`NLVector`	*v*	The second vector.
`double`	*a*	The multiplication factor.

Description

This routine sets $u = u + a\ v$. Errors

Message	Severity

NLNegateVector

Purpose

Sets vector to it's pro with

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
NLNegateVector(u);
```

NLVector *u* Theector.

Description

This sets aector negatTve. Errors

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

NLVnonZeri

Purpose

For a sparse vector returns a pointer to the array containing the list of nonzeros.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
c=NLVnonZero(u);
```

`int*`	*c*	The list of which coordinates are nonzero.
`NLVector`	*u*	The vector.

Description

This routine returns a pointer to the array of which coordinates of a vector are nonzero if the vector is a sparse vector (otherwise returns (int*)NULL). Note that this array may be reallocated when a coordinate becomes nonzero. In that case the pointer is no longer valid and should not be used (get a new pointer!). Errirs

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

NLVGetNumberOfNonZeros

Purpose

Returns the number of nonzeros (if sparse) or the number of coordinates (if dense).

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
n=NLVGetNumberOfNonZeros(u);
```

`int`	*n*	The answer.
`NLVector`	*u*	The vector.

Description

This routine returns the number of nonzeros if the vector is a sparse vector (otherwise returns the total umber of coordinates). Errors

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

NLVGetNonZero

NLVWrapped

Purpose

Queries if a vector is wrapped.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
flag=NLVWrapped(u);
```

`int`	*flag*	The answer. 1 indicates wrapped, 0 denF2111.9y2i-91.35780(1.912955Tf0(Ve)6(v)28(e)-d[(in)1(t)]TJ/F3511.955

NLVData

Purpose

Returns a pointer to the data array of a vector.

NLCreateMatrixWithData

Purpose

Message	Jeverity
”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
”Out of memory, trying to allocate %d bytes”	12
”Out of memory, trying to allocate %dx%d matrix (%d bytes)”	1214.5

NLCrea6eSparseMatrix

Purpose

Alloca6es 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*	

NLRefMatrix

NLFreeMatrix

Purpose

Frees the storage associated with an NLMatrix data structure.

Library

libNLPAPI.a

C Syntax

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

NLMatrix *A* The matrix.

Description

The NLMatrix data structure uo9(Rx)-34reference counting. This routine should

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

This routine returns the number of rows in the matrix. This is set when the matrix is created.

Errors

Errors return -1.

Message	Severity

NLMGetNumberOfCols

Purpose

Returns the number of columns in an NLMatrix.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
m=NLMGetNumberOfCols(A);
```

`int`	*m*	The number of columns.
`NLMatrix`	*A*	The matrix.

Description

This 546utines Returns the number of columnf in the matrix.-434[(This)-324his s thetrie ceatedx.

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 `NLMSetElement` routine (page 135).

Errors

Errors return DBL_QNAN.

Message	Severity
"Matrix (argument 1) is NULL"	12
"Row index %d (argument 2) is negative."	12
"Row index %d (argument 2) is too large. Must be less than %d"	12
"Column index %d (argument 3) is negative".	12
"Column index %d (argument 3) is too large. Must be less than %d"	12

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.

Message	Severity
"Matrix (argument 1) is NULL"	12
"Row index %d (argument 2) is negative."	12

NLMatrixDoubleProduct

Purpose

Computes the product $u^T A v$.

Library

`libNLPAPI.a`

[illegible]

NLMVMult

Purpose

Computes the product $b = Ax$.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
NLMVMult(A,x,b);
```

`NLMatrix`	*A*	The catrix.
`double*`	*x*	An array containing the coordinates of the vector x.
`double*`	*b*	

NLMVMultT

Purpose

Computes the product $b = A^T x$.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
NLMVMultT(A,x,b);
```

NLMSetToZero

Purpose

Sets all elements of an NLMatrix to zero.

Library

`libNLPAPI.a`

C Syntax

NLMatrixClone

Purpose

Creates a matrix of the same type and size of another, with the same element values.

Library

`libNLPAPI.a`

C Syntax

NLMatrixOneNorm

Purpose

Computes the 1-norm of a matrix (with an optional diagonal scaling).

Library

`libNLPAPI.a`

C Syntax

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

L_1

NLMSumRankOneInto

Purpose

Adds a rank one matrix into a matrix

NLMMMMProd

Purpose

Computes the matrix-matrix-matrix product $B = M^T A M$.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
NLMMMMProd(A,M,B);
```

`NLMatrix`	A	An $n \times n$ matrix.
`double*`	M	

NLMSparse

Purpose

Queries if a matrix is sparse.

Library

`libNLPAPI.a`

C Syntax

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

NLMDetermineHessianSparsityStructure

Purpose

Updates the sparsity structure of a matrix to accomodate the nonzeros in the Hessian of the objective or a constraint of a problem.

Library

`libNLPAPI.a`

C Syntax

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

NLM Data

Purpose

NLMnE

Purpose

Returns the number of “nonzero” entries in a matrix.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
nE=NLMnE(A);
```

`double*`	*nE*	The number of nonzeros.
`NLMatrix`	*A*	The matrix.

Description

Thiis routine returns the number of “nonzero” entries in a matrix. Note that this is the number of possible nonzeros, not the number of actual nonzeros. So for an $n \times m$ matrix stored as a dense matrix the result is always $n \ m$. For matrices stored in one of the sparse formats it is the number of allocated nonzeros. Errors

Message	Severity
”Matrix (argument 1) is NULL”	4

NLMRow

Purpose

Returns a pointer to the “row” array of the matrix.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
row=NLMRow(A);
```

`double*`	*row*	The row array.
`NLMatrix`	*A*	The matrix.

Description

NLMCol

Purpose

Returns a pointer to the “col” array of the matrix.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
col=NLMCol(A);
    double*   col
```

tsanNLMatre1;ix.

NLCreateGroupFunctionByString

Purpose

Allocates and initializes an NLGroupFunction data structure by way of an expression.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
G=NLCreateGroupFunctionByString(P,type,var,expr);
```

`NLGroupFunction` *G*	The group function.
`NLProblem` *P*	The problem to which the group function belongs.
`char *`*type*	A name associated to the group function.
`char *`*var*	The identifer used in the expression for the argument of the group function.
`char *`*expr*	An expression for the group function.

Description

The routine `NLCreateGroupFunctionByString` allocates and initializes an NLGroupFunction data structure. The *var* strings "s" or "[s]") identifies the identifier used in the expression string (e.g. "sin(s)") for the argument of the group function.

Message	Severity
”Problem (argument 1) is NULL”	12
”type (argument 2) is NULL”	12
”var (argument 3) is NULL”	12
”expr (argument 4) is NULL”	12
”Out of memory, trying to allocate %d bytem”	12

NLCreateGroupFunction

Purpose

Allocates and initializes an NLGroupFunction data structure.

Library

`libNLPAPI.a`

C Syntax

`urpose`

goes to zero. References may be added using the `NLRefGroupFunction` subroutine (page 158).

Errors

Errors return (NLGroupFunction)NULL.

Message	Severity
"Out of memory, trying to allocate %d bytes"	12

NLRefGroupFunction

Purpose

Registers a reference to an NLGroupFunction data structure.

Library

`libNLPAPI.a`

C Syntax

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

`NLGroupFunction` *G* The group function.

Description

NLFreeGroupFunction

Purpose

Frees the storage associated with an NLGroupFunction data structure.

NLGEvalDer

Purpose

Evaluates the derivative of an NLGroupFunction.

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.

Description

This routin.palue of the secondative of aroup
d*g*(*x*)/dx.

Errors

Errors 5”turn DBL_QNAN.

Message	Severity
”Group Function (argument 1) is NULL”	12
”Group Function Second Derivative function is NULL”	12

NLCreateElementFunctionByString

Purpose

if the count g417(g417(g417Flhe)-217)nt. Referen1(un)Flhey417(g4b4178(c)-1(o)addedhe)-21usingc

The routine NLCreateElementFunction allocates and initializes an NLElementFunction data structure.

NLRefElementFunction

Purpose

Registers a reference to an NLElementFunction data structure.

Library

`libNLPAPI.a`

C Syntax

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

`NLElementFunction` *F* The element function.

Description

The NLElementFunction 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

NLEGetDimension

Purpose

NLEEval

Purpose

Evaluates an NLElementFunction.

Library

`libNLPAPI.a`

C Syntax

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

`double`	*f*	The value of the element function.
`NLElementFunction`	*F*	The element function.
`int`	*n*	The number of coordinates.
`double`	**x*	The point.

DescLPiption

This routine returns the value of a element function $f(x)$.

ELPrors

Errors return DBL_QNAN.

Message	Severity

NLEEvalDer

Purpose

Evaluates the derivative of an NLElementFunction.

Library

`libNLPAPI.a`

C Syntax

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

NLCreateNonlinearElement

Purpose

Allocates and initializes an NLElementFunction data structure.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
NE=NLCreateNonlinearElement(P,type,fn,vars);
```

`NLNonlinearElement` *NE*	The new nonlinear element.
`NLProblem` *P*	The problem.
`char *`*type*	The type given to the new nonlinear element.
`NLElementFuntion` *fn*	The element function for the new nonlinear element.
`int *`*vars*	A list of the element variables for the new nonlinear element.

Description

The routine `NLCreateNonlinearElement` allocates and initializes an NLNonlinearElement data structure.

The NLNonlinearElement data structure uses reference counting. The data structure should be deleted using the `NLFreeNonlinearElement` subroutine (page .eist. This will decrement the reference count and free the storage if the structure is [illegible].

NLRefNonlinearElement

Purpose

Registers a reference to an NLNonlinearElement data structure.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
void NLRefNonlinearElement(P,F);
```

`NLProblem`	*P*	The problem.
`NLNonlinearElement`	*F*	The element function.

Description

The NLNonlinearElement data structure uses reference counting. This rou-

NLFreeNonlinearElemedt

Purpose

NLNEGetName

Purpose

Returns the name of a nonlinear element.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
name=NLNEGetName(
```

NLNEGetIndex

Purpose

Returns the index of an element variable of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

NLPGetNumberOfNonlinearElements

Purpose

Returns the number of nonlinear elements.

Library

`libNLPAPI.a`

C Syntax

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

n

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 (n)

Td[426(g70253f0-14600Td[(PA)etur

Purpo6me

NLPGetGroupTypeName

Purpose

NLPGetGroupName

Purpose

Returns the name of a group.

Library

`libNLPAPI.a`

C Syntax

NLPSetGroupFunction

Purpose

NLPGetGroupFunction

Purpose

NLPIsGroupFunctionSet

Purpose

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

Library

libNLPAPI.a

C Syntax

NLPGetGroupA

Purpose

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

Library

`libNLPAPI.a`

C Syntax

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

`NLVector`	*a*	The linear element.
`NLProblem`	*P*	The problem.
`int`	*group*	The index of s.75-326(g)1(roup.)]TJ/F-130.76

NLPIsGroupASet

Purpose

Queries whether the linear part of the linear element of a group has been set.

Library

`libNLPAPI.a`

C Syntax

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

int group

int

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);
```

NLPIsGroupBSet

Purpose

NLPSetGroupScale

Purpose

Sets the scale factor of a group.

Library

libNLPAPI.a

C Syntax

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

iat NLProblem	*rc*	The return code.

NLPGetGroupScale

Purpose

Gets the scale factor of a group.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
s=NLPGetGroupScale(P,group);
```

double	*s*	The scalf0-21d009-l6.p NAGrbalme	The t	*group*	The of thego(oup)1p.

NLPAddNonlinearElementToGroup

Purpose

Adds an empty nonlinear element to a group.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
e=NLPAddNonlinearElementToGroup(P,group,type,weight,f,variables,xfrm);
```

int	*e*	The index of the new nonlinear ele-

NLPGetElementWeight

Purpose

NLPSetElementWeight

Purpose

Changes the weight of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

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

NLPIsElementWeightSet

Purpose

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

Library

`libNLPAPI.a`

C Syntax

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

`int`	*ans*	The answer, 1==Set, 0=Not Set.
`NLProblem`	*P*	The problem.
`int`	*group*	The index of the group.
`int`	*element*	The number of the element.

NLPGetElementFunctionOfGroup

Purpose

Returns the nonlinear element function of a nonlinear element.

NLPGetGroupNonlinearElement

Purpose

Returns a nonlinear element of a group.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
ne=NLPGetGroupNonlinearElement(P,group,i);
```

ne The nonlinear element.

`int` *group*

NLPSetElementFunction

Purpose

Changes the nonlinear element function of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetElementFunction(P,group,element,f,variables);
```

`int`	*rc*	The return code.
`NLProblem`	*P*	The problem.
`int`	*group*	The index of the group.
`int`	*element*	The number of the nonlinear element.
`NLElementFunction`	*f*	The element function.
`int`	it variables	A list of the internal variables.

Description

This routine changes the nonlinear element function of an element of a group. There must be as many entries in the list of internal variables as the element function has unknowns.

Errors

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

Message	Severity

NLPSetElementFunctionWithRange

NLPGetElementRangeTransformationOfGroup

Purpose

Returns the range transformation of a nonlinear element.

Library

libNLPAPI.a

C Syntax

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

NLMatrix	*f*	The range transformation.
NLProblem	*P*	
		NLProbl[(,)0()93(r)-J/F3511.955Tf67.3220Td[(P)]TJ/F1indexrange tF

NLPGetElementRangeTransformation

Purpose

Returns the range transformation of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

NLPGetNumberOfInternalVariablesInElement

Purpose

Returns the number of internal variables of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

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

`int`	*n*	The number of internal variables.
`NLProblem`	*P*	The problem.
`int`	*group*	The index of the group.
`int`	*element*	The number of the nonlinear element.

Description

This routine returns the number of internal variables of a nonlinear element

NLPGetElementIndexIntoWhole

Purpose

Returns the number of internal variables of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
var=NLPGetElementIndexIntoWhole(P,group,element,int i);
```

`int`	*var*	The index of the internal variable.
`NLProblem`	*P*	The problem.
`int`	*group*	The index of the group.
`int`	*element*	The number of the nonlinear element.
`int`	*i*	Which internal variable.

Description

This routine returns the index of an internal variable of a nonlinear element

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);
```

`int`	*n*	The number of unknowns.
`NLProblem`	*P*	The problem.
`int`	*group*	The index of the group.
`int`	*element*	The number of the nonlinear element.

Description

in a up.
group.

Note: this is not the number of internal va of an element, since the

ated.

Errors

Errors return -1.

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

NLPGetNumberOfElementsInGroup

Purpose

Returns the total number of nonlinear elements in a group.

Library

`libNLPAPI.a`

C Syntax

NLPGetNumberOfElementsO

Purpose: Returns the number of nonlinear elements in the Objective.

```
n=NLPGetNumberOfElementsiO(P);
```

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

Description: This routine returns the total number of nonlinear elements in the Objective function.

Errors return -1.

Message

NLPGetNumberOfElementsE

Purpose

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

Library

`libNLPAPI.a`

C Syntax

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

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

Description

TPhis5r2(cr2PThispF185TC3)]TJd[of noet2(returns)-326(the)-327(equan)31(t)Tf-8Tf0tyaisaints.

NLPGetNumberOfElementsI

Purpose

NLPGetElementTypeName

Purpose

Returns the type name of a nonlinear element.

Library

libNLPAPI.a

C Syntax

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

char %25131.955.955T0-14.445[(ch)1(ar)]TJ45.2511.955TThe the tense.20Td[(elemen

NLPGetTypeOfElement

Purpose

Returns the type name of a nonlinear element.

Library

libNLPAPI.a

C Syntax

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

int	*type*	Th1(men)2type of the element.
NLProblem	*P*	Th1(men)2problem.
int	*group*	Th1(men)2index of th1(men)2group.
int	*element*	Th1(men)2number of the element.

Description

This routine returns the type of a nonlinear element. Element types are assigned with the NLCreateNonlinearElement (page 173) subroutine. A new(men)2type name is assigned a number, and the name is stored.

Errors

Errors return -1.

Message	Severity

NLPGetNumberOfElementTypes

Purpose

Returns the number of distinct types of nonlinear elements.

Library

`libNLPAPI.a`

C Syntax

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

`int`	*n*	The number of element types.
`NLProblem`	*P*	The problem.

Description

This routine returns the number of distinct element types. Element types are assigned with the `NLCreateNonlinearElement` (page 173) subroutine. A new type name is assigned a number, and the name is stored.

Errors

NLPGetE6ementType

Purpose

Returns the index of a type of nonlinear element.

Library

libNLPAPI.a

C Syntax

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

NLPGetNumberOfGroupTypes

Purpose

NLPGetGroupType

Purpose

Returns the index of a type of group.

Library

`libNLPAPI.a`

C Syntax

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

`int`	*type*	The type.
`NLProblem`	*P*	The problem.
`int`	*i*	The index of the type.

Description

NLCreateLancelot

Purpose

Allocates and initializes an NLLancelot data structure.

Library

`libNLPAPI.a`

C Syntax

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

`NLLancelot` *Lan* The solver.

Description

The routine `NLCreateLancelot` allocates and initializes an NLLancelot data structure. The solver returned has default parameters values, which can be set with various subroutines. Multiple instances are legal.

The storage used by the solver can be returned to the system using the `NLFreeLancelot` subroutine (page 226).

Errors

Errors return (NLLancelot)NULL.

Message	Severity

NLFreeLancelot

Purpose

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
```

LNMaximize and LNMaximizeDLL

Purpose

Allocates and initializes an NLLancelot data structure.

Library

`libNLPAPI.a`

C Syntax

LNGetCheckDerivatives

Purpose

Gets the parameter controlling how Lancelot test derivatives.

Library

`libNLPAPI.a`

C Syntax

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

flag

LNSetConstraintAccuracy

Purpose

Sets the parameter controlling how accurately constraints are solved.

Library

`libNLPAPI.a`

C Syntax

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

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

Description

The routine `LNSetConstraintAccuracy` sets the parameter controlling how accurately the constraints are solved. The default value is 0.00001. The `SPEC.SPC` file entry this corresponds to is `CONSTRAINT-ACCURACY-REQUIRED`.

Errors

LNGetConstraintAccuracy

Purpose

Gets the parameter controlling how accurately Lancelot solves constraints.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
limit=LNGetConstraintAccuracy(Lan);
```

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

Description

The routine `LNGetConstraintAccuracy` gets the parameter controlling how accurately Lancelot solves the constraints. The default value is 0.00001. The `SPEC.SPC` fice entry this gets is `CONSTRAINT-ACCURACY-REQUIRED`.

Errors

Errors return DBL_QNAN.

Message	Severity

LNSetFirstConstraintAccuracy

Purpose

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

Library

`libNLPAPI.a`

C Syntax

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

LNGetFirstConstraintAccuracy

LNSetFirstGradientAccuracy

Purpose

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.

LNSetGradientAccuracy

Purpose

Sets the parameter controlling the accuracy for the gradients.

Library

rc *Lan*, *limit*);

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

Description

The routine sets the parameter controlling the ac- file entry this corresponds to is .

Errors

Message	Severity

LNGetGradientAccuracy

Purpose

Gets the parameter controlling the accuracy for the gradients.

Library

`libNLPAPI.a`

C Syntax

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

LNSetInitialPenalty

Purpose

Sets the parameter controlling the initial penalty.

Library

libNLPAPI.a

C Syntax

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

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

Description

The routine LNSetInitialPenalty sets the parameter controlling the initial penalty. The default value is 0.1. The SPEC.SPC file entry this sets is INITIAL-PENALTY-PARAMETER.

Errors

Errors return 0, normal execution returns 1.

Message	Severity

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 initial penalty. The default value is 0. The SPEC.SPC file entry this sets is INITIAL-PENALTY-PARAMETER.

Errors

Errors return DBL_QNAN.

Message	Severity

LNGetMaximumNumberOfIterations

Purpose

Gets the parameter controlling how long Lancelot runs.

Library

`libNLPAPI.a`

C Syntax

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

`int`	*iter*	Maximum number of iterations.
`NLLancelot`	*Lan*	The solver.

Description

The routine `LNGetMaximumNumberOfIterations` sets the parameter controlling how long Lancelot runs. The default value is 100.

Errors

Errors return -1.

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

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);
```

doud5T[(p)07Tf0-Rl)alLimeter con

Purpose

Sets the parameter controlling how often Lancelot prints.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=LNSetPrintEvery(Lan,iter);
```

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

Description

The routine LNSetPrintEvery sets the parameter controlling how often L51(cTx;)]TJ-56.6629-14.44

LNGetPrintEvery

Purpose

Gets the parameter controlling how often Lancelot prints.

Library

`libNLPAPI.a`

C Syntax

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

`int`	*iter*	
`NLLancelot`	*Lan*	The solver.

Description

The routine `LNGetPrintEvery` sets the parameter controlling how often Lancelot prints. The default value is 1.

Errors

Errors return -1.

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

LNSetPriatLevel

Purpose

Sets the parameter coatrolling how much output Lancelot produces.

Library

libNLPAPI.a

LNGetPrintLevel

Purpose

LNSetPrintStart

Purpose

t

herou(tier)]TJ/F2111.955Tf65.04560Td[(Lt)1(ec)1tPryS(t)1ope

LNSetPrintStop

Lnecon31(t)]T/35111.955Tf73.47400Td[()50(anx)]T/F111.955Tf314.2400Td[T(he)-327solver

Purpose

Sets the parameter controlling when Lancelot stops printing.

Library

libNLPAPI.a

C Syntax

#inclua(C).n(-4(<N(L)1LPA)1(PI)1(h>x)]T/35111.955Tf0140.457Td[r)50(cx)]T/F2111.955T

LNGetPrintStop

LNSetRequireExactCauchyPoint

Purpose

LNSetSaveDataEvery

Purpose

Sets the parameter controlling how often Lancelot saves data.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc=LNSetSaveDataEvery(Lan,iter);
```

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

Gets the parameter controlling how Lancelot uses scalings.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
choice=LNGetScalings(Lan);
```

`char`	`*choice`	How to use scalings.
`NLLancelot`	`Lan`	The solver.

Description

TheiTutin

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

LNGetLinearSolverMethod

Purpose

Gets the parameter determining what linear solver is used.

Library

`libNLPAPI.a`

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

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

LNGetLinearSolverBandwidth

bandwidth(Bo)1(lv)1(rBv)1and(e

Purpose

Gets the parameter determining what bandwidth the linear 36(bSo)1(lv327e(r)-326usne)-1s.e

Syia

cou(d)1(e)514(<NLi)1PAi

LNSetStopOnBadDerivatives

Purpose

Sets the parameter controlling how Lancelot deals with bad derivatives.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc=LNSetStopOnBadDerivatives(Lan,flag);
```

`int`	*rc*	The return code.
`NLLancelot`	*Lan*	The solver.
`int`	*flag*	What to do.

Description

The routine `LNSetStopOnBadDerivatives` Sets the parameter controlling how Lancelot deals with bad derivatives. Legal va7ues for the flag and their meaning –

0 stop on warning
1 stop on element derivative warning
2 stop on group derivative warning

The default va7ue is 0.

Errors

Errors return 0, normal execution returns 1.

Message	Severity
”Solver (argument 1) is NULL”	12

LNGetStopOnBadDerivatives

Purpose

Gets the parameter controlling how Lancelot deals with bad derivatives.

Library

`libNLPAPI.a`

LNSetTrustRegionRadius

Purpose

Sets the parameter controlling the radius of the trust region.

T646durary

```
l4duN4dLPA4dPI4d.a
```

C Sy46dnta46dx

```
#4dnc4dlude <NL4dPA4dPI4d.h>
rc=L4dNSetTrustRegio4dnRa4dius(lam,radius);
```

int	*rc*	The return code.
NL4dLa4dnce4dlot	*lam*	The solver.
doub4dle	*radius*	The ra46ddius.

Description

The routine LNSe4dtTrustRegionRadius sets the para46dmeter co46dntro46dlling the ra46ddius o46df the trust regio46dn. The default value is 0.46d4 and the entry this sets is TRUST-REGIO4dN-RADIUS

Errors

Erro46drs retur46dn 046d, no46drmal execution returns 1.

LNGetTrustRegionRadius

Purpose

Gets the parameter controlling the radius of the trust region.

Library

LNGetTrustRegionType

Purpose

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

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
choice=LNGetTrustRegionType(Lan
```

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=LNSetUseExactSecondDerivatives(Lan,flag);
```

int	*rc*	The return code.
NLLancelot	*Lan*	The solver.
char	**flag*	What to do - one of "Exact", "BFGS","DFP","PSB", or "SR1"

Description

The routine LNSetUseExactSecondDerivatives sets the parameter control-
ling how [(

LNGetUseExactSecondDerivatives

Purpose

Gets the parameter c31xmc31xllingeiv54(ra)1t(iv327es.e)]TJ/F7811.955Tf0-21.646Td[L(i)1ber
i
Sy31xn
<NLi