1 Contents

1.0.3 Inequality Constraints

1.2 Vectors

1.3 Matrices

1.4 Group Functions

1.5 Element Functions

1.5.2 The list of groups

1.6.3 Setting LANCELOT Garameters

LNStCheckDerivatives	192
LNGtCheckDerivatives	46
LNStConstraintAccuracy	193
LNGtConstraintAccuracy	47
LNStFirstConstraintAccuracy	194
LNGtFirstConstraintAccuracy	53
LNStFirstGradStepLength	195
LNGtFirstGradStepLength	54
LNStSecondConstraintAccuracy	196
LNGtSecondConstraintAccuracy	55
LNStMaxItCG	197
LNGtMaxItCG	56
LNStMaxItQP	198
LNGtMaxItQP	57
LNStMaxItNewton	199
LNGtMaxItNewton	58
LNStMaxSkipPGradient	200
LNGtMaxSkipPGradient	59
LNStMaxItPGradient	201
LNGtMaxItPGradient	60
LNStMaxItFace	202
LNGtMaxItFace	61
LNStStepControlRadius	203
LNGtStepControlRadius	62
LNStMaxItMultipliers	204
LNGtMaxItMultipliers	63
LNStMaxItSLP	205
LNGtMaxItSLP	64
LNStMaxItArtificials	206
LNGtMaxItArtificials	65
LNStMaxCGReduce	207
LNGtMaxCGReduce	66
LNStStartAllArtificials	208
LNGtStartAllArtificials	67
LNStSteepestDescent	209
LNGtSteepestDescent	68
LNStForceQPBasis	210
LNGtForceQPBasis	69
LNStNoMarginalStop	211
LNGtNoMarginalStop	70
LNStSmallestMultiplier	212
LNGtSmallestMultiplier	71
LNStInitialPenalty	213
LNGtInitialPenalty	72
LNStPenaltyIncrease	214
LNGtPenaltyIncrease	73
LNStMinPenaltyIncrease	215
LNGtMinPenaltyIncrease	74
LNStInitialMaxCGIts	216
LNGtInitialMaxCGIts	75
LNStMaxCGIncrease	217
LNGtMaxCGIncrease	76
LNStInitialRadius	218
LNGtInitialRadius	77
LNStMaximumRadius	219
LNGtMaximumRadius	78
LNStStepAccuracy	220
LNGtStepAccuracy	79
LNStLinearAccuracy	221
LNGtLinearAccuracy	80
LNStRadiusAccuracy	222
LNGtRadiusAccuracy	81
LNStFactorizationAccuracy	223
LNGtFactorizationAccuracy	82
LNStLargestAccuracy	224
LNGtLargestAccuracy	83
LNStMaximumStepFactor	225
LNGtMaximumStepFactor	84
LNStMaximumStepForLP	226
LNGtMaximumStepForLP	85
LNStInitialMaximumStep	227
LNGtInitialMaximumStep	86
LNStMinimumDecreaseInFeas	228
LNGtMinimumDecreaseInFeas	87
LNStLinearStepFactor	229
LNGtLinearStepFactor	88
LNStViolationLimitFactor	230
LNGtViolationLimitFactor	89
LNStReductionInPenalty	231
LNGtReductionInPenalty	90
LNStReductionInWeight	232
LNGtReductionInWeight	91
LNStInfinityValue	233
LNGtInfinityValue	92
LNStPivotTolerance	234
LNGtPivotTolerance	93
LNStPivotToleranceForDependencies	235
LNGtPivotToleranceForDependencies	94
LNStZeroPivot	236
LNGtZeroPivot	95
LNStInactiveBoundTolerance	237
LNGtInactiveBoundTolerance	96
LNStDropTolerance	238
LNGtDropTolerance	97
LNStDropTolerance1	239
LNGtDropTolerance1	98
LNStDropTolerance2	240
LNGtDropTolerance2	99
LNStDropTolerance3	241
LNGtDropTolerance3	100
LNStMaximumGrowthAllowed	242
LNGtMaximumGrowthAllowed	101

NLCreateElementFunction

Purpose

Allocates and initializes an NLElementFunction data structure.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
    NLElementFunction NLCreateElementFunction(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 routine NLCreateElementFunction allocates and initializes an NLElementFunction data structure.

The NLElementFunction data structure uses reference counting. The data structure should be deleted using the NLFreeElementFunction subroutine (page 36). This will decrement the reference count and free the storage if

NLCreateGroupFunction

goes to zero. References may be added using the `LNRefGroupFunction` subroutine (page 188).

Errors

Errors return (NLGroupFunction)NULL.

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

NLCreateMatrix

Purpose

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

Library

`libNLPAPI.a`

C Syntax

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

`NLMatrix`	*A*	The matrix.
`int`	*n*	

NLCreateSparseMatrix

NLCreateMatrixWithData

Purpose

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

Library

`libNLPAPI.a`

C Syntax

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

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

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
”Out of memory, trying to allocate %d bytes”	%d bytes”ut of memo

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 ntes nonlinear1(r)-3267element.nDscaopion

NLCreateProblem

Purpose

Allocates and initializes an NLProblem data structure.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
P=NLCreateProblem(probName,nV);
```

`NLProblem`	*P*	The problem.
`char`	*probName*	Name

NLCreateVectorWithFu6lData

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

12 errors return (NLVector)NULL.

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

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

Purpose

Queries whether an error condition exists.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc

  int  rc
```

NLFreeGroupFunction

Purpose

Frees the storage associated with an NLGroupFunction data structure.

Library

`libNLPAPI.a`

C Syntax

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

`NLGroupFunction` *G* The group function.

Description

The NLGroupFunction data structure uses reference counting. This routine should be used to indicate that a vector is no longer needed. It will decrement the reference count and free the storage if the count goes to zero. The `LNRef-GroupFunction` subroutine (page 188.773--251(aun)2y--251(9(b)-27(e)-320(used)-326oth)-326dsed

NLFreeLancelot

Purpose

Releases storage associated with an NLLancelot data structure.

Library

libNLPAPI.a

C Syntax

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

NLLancelot *Lan* The solver.

Description

The routine NLFreeLancelot returns storage associated with a solver to the system.

Errors

Errors return without changing the solver.

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

NLFreeMatrix

Purpose

Frees the storage associated with an NLMatrix data structure.

Library

NLFreeNonlinearElemedt

Purpose

Frees the storage associated with an NLNonlinearElemedt data structure.

Library

NLFreeVector

Purpose

NLGEvalDer

Purpose

Evaluates the derivative of an NLGroupFunction.

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>
limit=LNGetConstraintAccuracy(Lan);
```

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

Description

The routine

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.

NLGetErrorLine

Purpose

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)Version 12.1 (ns) 3 The 14 Apr 21 10:30TdD1.955s542c(br)-(i)pt(i)onse

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.

Description

The routine

LNGetGradientAccuracy

Purpose

Gets the parameter controlling the accuracy for the gradients.

Library

libNLPAPI.a

C Syntax

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

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

Description

The routine LNGetGradientAccuracy

LNGetJi□yTuneTolerance

Purpose

LNGetLinearSolverBandwidth

Purpose

Gets the parameter determining what bandwidth the linear solver uses.

Library

`libNLPAPI.a`

C Syntax

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

`int`	*bandwidth*	The bandwidth.
`NLLancelot`	*Lan*	

"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 -1.

LNGetMaximumNumberOfIterations

Purpose

NLGetNErrors

Purpose

Returns the number of errors that have been flagged.

Library

`libNLPAPI.a`

C Syntax

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

erii

GetLhemeterc(n)27(a)1ro27pLah8wncc(r)eoth8w(n)26(m)27(uc)27(h)-326(1(i)utput)-0L1(ia)1(nce

LNGetPrintLevel

Purpose

Gets the parameter controlling how much output Lancelot produces.

Library

`libNLPAPI.a`

C Syntax

addels `#include <NLPAPI.h>`

LNGetPrintStart

Purpose

LNGetSaveDataEvery

Purpose

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

LNGetSolveBQPAccurately

Purpose

Gets the parameter controlling the solution of the BQP.

Library

libNLPAPI.a

C Syntax

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

int	*choice*	
NLLancelot	*Lan*	The solver.

Description

The routine LNGetSolveBQPAccurately gets the parameter controlling the solution of the BQP. The default is 0.

The corresponding SPEC.SPC file entry isveBQPAcSOLVE-BQP-ACCURATELY

Errors

Errors return -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);
```

`char`	*`choice`	Which type. `NLLancelot`

NLMGetNumberOfCoMf

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

Errors

Errors return -1.

Message	Severity

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
"Matrix (argument 1) is NULL"	12

NLMSetElement

Purpose

Changes the value of an element of an NLMatrix.

Library

libNLPAPI.a

C Syntax

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

LNMaximize and LNMaximizeDLL

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
```

NLNEGetIndex

Purpose

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

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
var              P,ne,i);
```

`int`	*var*	The variable.
`NLProblem`	*P*	The problem.

NLNEGetName

Purpose

Returns the name of a nonlinear element.

Library

libNLPAPI.a

C Syntax

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

NLPAddLinearEqualityConstraint

Purpose

NLPAddLinearInequalityConstraint

Purpose

Adds a linear inequality constraint.

Library

`libNLPAPI.a`

C Syntax

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

g

NLPAddMinMaxConstraint

THIS IS AN EXTENSTION TO VANILLA LANCELOT

Purpose

Adds a nonlinear inequality constraint.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
g=NLPAddMinh-21.5f0216627.0716716(<NL)(1PA)(1PI)(1.h>)TJF5PAo5f0216627.0027.0027.55hr.216.
```

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-

NLPAddNonlinearElementToObjectiveGroup

Purpose

Adds an empty nonlinear element to a group.

Library

libNLPAPI.a

C Syntax

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

NLPAddNonlinearElementToEqualityConstraint

Purpose

Adds an empty nonlinear element to an equality constraint.

Library

`libNLPAPI.a`

C Syntax

NLPAddNonlinearElementToInequalityConstraint

NLPAddNonlinearElementToMinMaxConstraint

THIS IS AN EXTENSTION TO VANILLA LANCELOT

Purpose

Adds an empty nonlinear element to a minmax constraint. (Requires the two step version of Lancelot.)

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
e=NLPAddNonlinearElementToMinMaxConstraint(P,constraint,weight,ne,variables,xfrm);
```

int NLPAddNonlinearElementToMinMaxConstraint(NLProblem P,int constraint,double w,NLNonlinearElement E);

`int`	*e*	The index of the new nonlinear element.
`NLProblem`	*P*	The problem.
`int`	*constraint*	The index of the mlem.11.95i2-185nearTd(TB7PI)1(eTJe near57(t.)TJF111.955Tf5.380ETJe1LPA111.955Tf-185.00

NLPAddNonlinearInequalityConstraint

Purpose

Adds a nonlinear inequality constraint.

Library

`libNLPAPI.a`

C Syntax

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

`int`	*g*	The index of the new group.
`NLProblem`	*P*	The problem.
`char`	**name*	The name of the new group.

Description

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

A trivial group is added, with no nonlinear element, and a zero linear

NLPGetElementFunctionOfGroup

Purpose

Returns the nonlinear element function of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

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

`NLElementFunction`	*f*	The element function.
`NLProblem`	*P*	The problem.
`int`	*group*	The index of the group.
`int`	*element*	The number0Td(38U70111.955Tf-167.6477TJ761o26(eleme5

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

NLPGetElementRangeTransformation

Purpose

Returns the range transformation of a nonlinear element.

Library

`libNLPAPI.a`

ment.

NLPGetElementRangeTransformationOfGroup

Purpose

Returns the range transformation of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

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

NLPGetElementWeight

Purpose

NLPGetGroupA

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);
```

NLPGetGroupFunction

Purpose

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*

NLPGetGroupScale

Purpose

Gets the scale factor of a group.

Library

libNLPAPI.a

C Syntax

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

This routine returns the index pe404(a)-403(g)-2roup type. Group types(e404(a)1(re(e404(a)1(s

NLPGetGroupTypeName

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 QNAN.

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

NLPSetLowerMinMaxBound

NLPGetMinMaxConstraintGroupNudber

THIS IS AN EXTENSTION TO VANILLA LANCELOT
Purpose

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.

NLPthe

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)TJdof noet2(returns)-326(the)-327(equan)31(t)Tf-8Tf0tyaisaints.

NLPGetNumberOfElementsI

Purpose

NLPGetNumberOfElementsInGroup

Purpose

Returns the total number of nonlinear elements in a group.

Library

`libNLPAPI.a`

C Syntax

NLPGetNumberOfElementsM

Purpose

Retf9.64ns the to.64tal number o.64f no.64nlinear elements in t.64he min-max co.64nstraints.

Library

libNLPAPI.a

C Sy.64nta.64x

```
#include <NLPAPI.h>
n=NLPGetNumberO.64fElemen.64sM(P);
```

in.64t	*n*	The number o.64f elements.
NLP9.64blem	*P*	The pro.64blem.

Des.64cription

This 9.64outine retf9ns the tota.64l number of nonlinear elements in t.64he min-max constra.64ints.

Errors

Erro.64rs retf9.64n -1.

Messa.64ge	Severity
"Pro.64blem (argument 1) is NULL"	12

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

NLPGetNumberOfEqualityConstraints

Purpose

Returns the number of equality constraints in a problem.

Library

`libNLPAPI.a`

C Syntax

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

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

Description

This routine returns the current number of equality constraints in ablem.m.

Errors

NLPGetNumberOfGroupTypes

Purpose

NLPGetNumberOfInequalityConstraints

Purpose

Returns the number of inequality constraints in a problem.

Library

`libNLPAPI.a`

C Syntax

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

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

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

NLPGetNumberOfMinMaxConstraints

THIS IS AN EXTENSTION TO VANILLA LANCELOT

Purpose

Returns the number of equality constraints in a problem.

Library

libNLPAPI.a

C Syntax

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

    int        n
```

NLPGetNumberOfNonlinearElements

Purpose

Returns the number of nonlinear elements.

Library

`libNLPAPI.a`

C Syntax

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

n

NLPGetNumberOfVariables

Purpose

NLPGetObjectiveGroupNumber

NLPGetProblemName

Purpose

Returns thenameof aproblem.

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
name=NLPGetProblemName(P);
```

char	*name	The problem name.
NLProblem	P	The problem.

Description

This routine returns the name of a problem, which was passed to the NLCreate-Problem (page 24)subroutine.

Note: The usershouldnotfree thereturnedstring.

Errors

Errors return (char*)NULL.

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

NLPGetTypeOfElement

Purpose

Returns the type name of a nonlinear element.

Library

`libNLPAPI.a`

C Syntax

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

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)

Purpo6me

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`*	

NLPGetVariableScale

Purpose

Returns the scale factor of a variable.

Library

`libNLPAPI.a`

C Syntax

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

`double`	*s*	The scale factor.
`NLProblem`	*P*	The problem.
`int`	*i*	The variable.

Description

NLPIsElementFunctionSet

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.

NLPIsGroupBSet

Purpose

NLPIsGroupFunctionSet

Purpose

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

Library

libNLPAPI.a

C Syntax

NLPIsLowerSimpleBoundSet

Purpose

Queries whether a lower =ound has =een set on a variable.

NLPIsUpperSimpleBoundSet

Purpose

Queries whether a upper bohe has been set on a variahe.

Library

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,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

NLPSetEqualityConstraintB

Purpose

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

Library

`libNLPAPI.a`

C Syntax

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

rc

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*	The index of the group.
NLVector	*a*	The linear element.

Description

This routine sets the linear part of the lineear element of a group. This can be queried with the NLPGetGroupA (page 113) subroutine.

c1(gea)1(n)l

NLPSetGroupB

Purpose

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

Library

`libNLPAPI.a`

C Syntax

```
#incluary
```

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 code.

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.

NLPSetInequalityConstraintB

Purpose

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

Library

libNLPAPI.a

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetInequalityConstraintB(P,constraint,b
```

NLPSetInequalityConstraintBounds

Purpose

Sets the bounds on an inequality constraint.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
rc=NLPSetInequalitnstraintBounds(    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 L55nl7520Tedds

Ne8

8mnsnt

NLI inequalityCo train owe Boun

eeSePit12(y(o)1(so)111.955 Tf)11L(o)1wetB(8)1(u8)1d(e

Purpose

Sets the lower b8und on an inequality const28

rrye

PAt

Syin28

cauda<NLt

NLPSetInequalityConstraintUpperBound

Purpose

Sets the upper bound on an inequality constraint.

Library

`libNLPAPI.a`

C Syntax

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

rc

NLPSetMinMaxBounds

THIS IS AN EXTENSTION TO VANILLA LANCELOT

Purpose

Sets the bounds on the min-max variable.

Library

NLPSetMinMaxConstraintA

THIS IS AN EXTENSTION TO VANILLA LANCELOT

Purpose

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

Library

`libNLPAPI.a`

C Syntax

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

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

Description

This routine sets the linear part of the lineear element of an minmax constraint.

Errors

NLPSetMinMaxConstraintB

THIS IS AN EXTENSTION TO VANILLA LANCELOT

Purpose

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

Library

`libNLPAPI.a`

C Syntax

NLPSetObjectiveGroupA

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

Library

`libNLPAPI.a`

C Syntax

```
int NLPSetObjectiveGroupA(NLProblem P, int group, NLVector 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

NLPSetObjectiveGroupFunction

Purpose

Sets the group function of a group.

Library

libNLPAPI.a

C Syntax

NLPGetUpperMinMaxBound

THIS IS AN ETENSTION TO VANILLA LANCELOT

Purpose

Gets the upper bound on the min-max variable.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
u=NLPGetUpperMinMaxBound(P);
```

`double`	*u*	The upper bound.
`NLProblem`	*P*	The problem.

Description

This routine gets the upper bound on the min-max variable.

Initially the bound is ∞. (A s41T7U(P)TJF2911.95max0F2911.95m26.547Oqurpose

NLPSetVariableName

Purpose

Assig7s the name of a variable.

Library

libNLPAPI.a

C Sytax

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

NLPSetVariableScale

Purpose

NLPrintProblem

Purpose

Prints a NLProblem data structure.

Library

`libNLPAPI.a`

C Syntax

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

NLRefMatrix

Purpose

Registers a reference to an NLMatrix data structure.

Library

NLRefNonlinearElement

Purpose

NLRefVector

Purpose

Registers a reference to an NLVector data structure.

Library

`libNLPAPI.a`

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

LNSetFirstConstraintAccuracy

Purpose

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

Library

`libNLPAPI.a`

C Syntax

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

LNSetFirstGradientAccuracy

Purpose

LNSetGradientAccuracy

LNSetInitialPenalty

Purpose

Sets the parameter controllinA the initial penalty.

Library

`libNLPAPI.a`

C Syntax

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

rc

LNSetJi□yTuneTolerance

Purpose

Sets the parameter controlling the “Ji□ytune Tolerance”. NOTE: this re-

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

LNSetMaximumNumberOfIterations

Purpose

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-56.6629-14.446

LNSetPriatLevel

Purpose

Sets the parameter coatrolling how much output Lancelot produces.

Library

libNLPAPI.a

LNSetPrintStart

Purpose

t

herou(tier)TJF2111.955Tf65.04560Td(Lt)1(ec)1tPryS(t)1ope

LNSetPrintStop

Lneconl(t)TJ15f7.40d()(anx)TJF15f120dT(he)-27solver.r

Purpose

Sets the parameter controlling when Lancelot stops printing.

Library

libNLPAPI.a

C Syntax

#inclua(C).n(-4<N(L)1PA)1PI)1h>x)TJ15f04Tdr)(cx)TJF215f970d=(L)1Sa(C)tPa(Cr#i)1tSa

LNSetRequireExactCauchyPoint

Purpose

Sets the parameter determining whether an exact cauchy point is rired.

Library

`libNLPAPI.a`

C Syntax

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

`int` *rc*

choice

LNSetSaveDataEvery

Purpose

LNSetStopOnBadDerivatives

Purpose

Sets the parameter controlling how Lancelot deals with bad derivatives.

LNSetTrustRegionRadius

Purpose

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-

NLVGetNC

Purpose

NLVGetNonZeroCoord

Purpose

NLVGetNumberOfNonZeros

NLVSetC

Purpose

Sets 1he specified coordinate of a vec1or.

Library

libNLPAPI.a

C Syntax

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

NLVIncrementC

LNGetMajorIterationCount

Purpose

Returns the current major iteration count. NOTE: this requires a modified version of Lancelot. Without that modification the count will always be zero.

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
#include <fortran.h>
count=F77NAME(LNGetMajorIterationCount)();
```

F77INTEGER *count* The major iteration count.

Description

The routine `LNGetMajorIterationCount` returns the count of the current major iteration. It get this count from a common block added to Lancelot by Andy Conn on Dec. 7, 2000. Users with vanilla Lancelot will always receive the same count, which will be whatever the common block is initialized with.

NLPEvaluateObjective, NLPEvaluateGradientOfObjective, NLPEvaluateHessianOfO

Purpose

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

Library

`libNLPAPI.a`

C Syntax

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

NLPEvaluateEqualityConstraint, NLPEvaluateGradientOfEqualityConstraint, NLPE

Purpose

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

Library

`libNLPAPI.a`

C Syntax

```
#include <NLPAPI.h>
v=NLPEvaluateEqualit(t)Jtraint(P,x);
NLPEvaluateGradientOfEqualityConstraint(P,x,g);
NLPEvaluateHeJsianOfEqualit(t)straint(P,x,H);
```

`double`	*v*	The value of the objective functiT.
`NLProblem`	*P*	The problem.
`NLVector`	*x*	The point.
`NLVector`	*g*	The gradient.
`NLMatrix`	*H*	The Hessian.

DeJcripti(t)T

These routines are for evaluating the objective, its gradient and Hessian. The values of the elements and groups are cached, and currently a new point is signaled by the routine `NLPInvalidateGroupAndElement(t)aches`

Errors

Message	Severity
"Problem (argument 1) is NULL"	12
"X (argument 2) is NULL"	12
"g (argument 3) is NULL"	12
"H (argument 3) is NULL"	12

NLPInvalidateGroupAndElementCaches

Purpose

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

Library

libNLPAPI.a

C Syntax

#include <NLPAPI.h>
NLPInvalidateGroupAndElementCaches(*P*);PI-9721-214Td(NL)1Pr)1obl)1em)]TJF55f6.220d(P