NLPAPI: An API to Nonlinear Programming Problems. User's Guide

Michael E. Henderson
IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598
mhender@watson.ibm.com

June 20, 2003

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

$$O(\mathrm{v}) = \sum_{i=1}^{n_g} \frac{1}{s_i} g_i\Big(\sum_{j=1}^{n_{e_i}} w_{ij} f_{ij}(R\mathrm{ev}_{ij}) + < a_i, \mathrm{v} > - b_i \mathrm{The}\ g_i : \mathbb{R} \rightarrow \mathbb{R}\ \text{are caled groups fu}$$

2.1.1 The Objective

linear element, a_i

The idea here is that the user creates either a sparse or dense vector, and passes it to the routine which computes the gradient, which fills in the appropriate values, or a matrix in one of several formats, and passes it to the routine which evaluates the Hessian. (See the sections below on vectors and matrices.)

```
"[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);
```

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

and second derivativesP051. It can be created by passing routines, or by way of a string.

```
NLGroupFunction g;
NLProblem P;
double P050*GP051P050double,void*P051;
double P050*dGP051P050double,void*P051;
double P050*ddGP051P050double,void*P051;
void *data;
void P050*freedataP051P050void*P051;
```

```
NLElementFunction ef;
int *vars;

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

the vars array gives a list of the problem variables (by number, starting with 0!) which become the element variables.

A nonlinear element can be added to a group using the appropriate routine:

```
int c;
int g;
double w;
NLNonlinearElement N;

NLPAddNonlinearElementToObjectiveGroup(P,g,w,N):
```

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.

```
int NLGetNErrors();
```

returns the total number of errors that have occured, and

```
void NLClearErrors();
```

4.1 Using the SetObjective/AddConstraint routines

```
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,
```

The main program and declarations –

```
int main(int argc,char *argv[])
 {
  NLProblem P;
  NLGroupFunction g;
  NLElementFunction f;
  NLNonlinearElement ne;
  int group;
  NLVector a;
  double x0[3];
  NLLancelot Lan;
  double x[3];

  intleme514(a;)]TJ0-14.445Td[(in)1(t)v[11(gv)-.6];
```

```
rc=NLVSetC(a,1,-1.);
rc=NLVSetC(a,2,0.);
```

```
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);
```

```
  {
   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
```