

# A Collection of 100 Test Problems for Nonlinear Mixed-Integer Programming - User's Guide -

*Address:* Klaus Schittkowski  
Department of Computer Science  
University of Bayreuth  
D - 95440 Bayreuth

*E-mail:* klaus.schittkowski@uni-bayreuth.de

*Web:* <http://www.klaus-schittkowski.de>

*Date:* December 12, 2010

## Abstract

The availability of mixed-integer nonlinear programming test problems is extremely important to test optimization codes or to develop new algorithms. We describe the usage of Fortran subroutines for a set of 100 test problems, where most of them are taken from existing collections, especially from the GAMS collection MINLPLib. For each test example, relevant problem data like number of variables or constraints and code for function evaluation are summarized in a single file. Program organization and numerical test results are presented, moreover some auxiliary routines to facilitate integration under own test environments. A frame to evaluate all test problems in a loop, is included together with the individual source codes of the collection. The implementation is thread-safe basic Fortran and the codes are easily transferred to C by f2c.

## 1 Introduction

We consider the nonlinear mixed-integer optimization problem to minimize an objective function  $f$  under nonlinear equality and inequality constraints, i.e.,

$$\begin{aligned} & \min f(x, y) \\ & g_j(x, y) = 0, \quad j = 1, \dots, m_e, \\ x \in \mathbb{R}^{n_c}, y \in \mathbb{N}^{n_i} : & g_j(x, y) \geq 0, \quad j = m_e + 1, \dots, m, \\ & x_l \leq x \leq x_u, \\ & y_l \leq y \leq y_u \end{aligned} \tag{1}$$

where  $x$  and  $y$  denote the continuous and the integer variables, respectively. It is assumed that the problem functions  $f(x, y)$  and  $g_j(x, y)$ ,  $j = 1, \dots, m$ , are continuously differentiable subject to  $x \in \mathbb{R}^{n_c}$ . Integer variables include binary variables.

Most of the test problems are taken from the GAMS Model Library MINLPLib, cf. Bussieck, Drud, and Meeraus [2]. The collection is widely used to test and compare algorithms, see e.g. Still and Westerlund [16] or Maniezzo, Stützle, and Voß [15]. Many of the underlying optimization problems are non-convex and we are not sure whether our own solutions are always global ones. All test problems are relaxable, i.e., function values can be computed not only for integer values  $y \in \mathbb{N}^{n_i}$ , but also for any real values in between. In other words, the integrality condition  $y \in \mathbb{N}^{n_i}$  in (1) can be replaced by  $y \in \mathbb{R}^{n_i}$ .

The Fortran source codes of all test problems are available through the link

<http://www.ai7.uni-bayreuth.de/home.htm>

A brief summary of all examples together with relevant data for  $n_c$ ,  $n_i$ ,  $m$ ,  $m_e$  and in particular the best known solution values is presented in Section 2. The usage of the subroutines is documented in Section 3 together with an example. A driver program is listed that shows how a nonlinear mixed-integer code, in this case the code MISQP of Exler, Lehmann, and Schittkowski [7], can be implemented. Section 4 contains a list of all individual results obtained by MISQP including objective function values, constraint violations, number of function calls, number of iterations, and especially errors in objective function subject to the best solution we know. Our optimal solutions were found by a series of test runs by the codes MISQP, a mixed-integer sequential quadratic programming method, two outer approximation codes MISQPOA and MISQPN, respectively, see Lehmann and Schittkowski [12, 14], and the branch-and-bound code MINLPB4 of Lehmann and Schittkowski [13].

The test examples are implemented in thread-safe Fortran 90 without global data (COMMON) or any special Fortran tricks (EQUIVALENCE, ENTRY). They are easily transferred to C by f2c.

## 2 The Test Problems

Most of the test problems are taken from the GAMS model library MINLPLib, see Bussieck, Drud, and Meeraus [2], and can be downloaded from

<http://www.gamsworld.org/minlp/minlplib.htm>

They are implemented in the GAMS modeling language and are easily transformed into other modeling languages like AMPL, BARON, GAMS, LINGO, or MINOPT, for example.

Our own motivation for collecting test problems is to develop new nonlinear mixed-integer optimization software for solving practical engineering optimization problems with expensive function evaluations. A typical application is the solution of mechanical structural optimal design problems based on a time-consuming FE analysis. Thus, the main

goal is to derive codes which require as few function evaluations as possible. To adjust the test problems to our needs, we implemented them in Fortran.

A list of characteristic problem data is presented in Table 1, where the following data are shown:

- no* - test problem number,
- name* - name of the test problem as used in our collection and in the literature,
- ref* - reference, if available,
- n<sub>c</sub>* - number of continuous variables,
- n<sub>i</sub>* - number of integer variables without binary ones,
- n<sub>b</sub>* - number of binary variables,
- m<sub>e</sub>* - number of equality constraints,
- m* - number of all constraints,
- $f(x^*, y^*)$  - best known objective function value.

Note that at least all test problems with nonlinear equality constraints are not convex.

Table 2: Mixed-Integer Test Problems

<i>no</i>	<i>name</i>	<i>ref</i>	<i>n<sub>c</sub></i>	<i>n<sub>i</sub></i>	<i>n<sub>b</sub></i>	<i>m<sub>e</sub></i>	<i>m</i>	$f(x^*, y^*)$
1	MITP1		2	3	0	0	1	-0.10010E+05
2	QIP1		0	4	0	0	4	-0.20000E+02
3	MITP2		2	0	3	0	7	0.35000E+01
4	ASAADI11	[1]	1	3	0	0	3	-0.40957E+02
5	ASAADI12	[1]	0	4	0	0	3	-0.38000E+02
6	ASAADI21	[1]	3	4	0	0	4	0.69490E+03
7	ASAADI22	[1]	0	7	0	0	4	0.70000E+03
8	ASAADI31	[1]	4	6	0	0	8	0.37220E+02
9	ASAADI32	[1]	0	10	0	0	8	0.43000E+02
10	DIRTY		12	13	0	0	10	-0.30472E+01
11	BRAAK1	[4]	4	3	0	0	2	0.10000E+01
12	BRAAK2	[4]	4	3	0	0	4	-0.27183E+01
13	BRAAK3	[4]	4	3	0	0	4	-0.19656E+00
14	DEX2	[5]	0	2	0	0	2	-0.56938E+02
15	TP83	[11]	3	2	0	0	6	-0.30666E+05
16	WP02	[18]	1	1	0	0	2	-0.24444E+01
17	NVS01	[2]	1	2	0	1	3	0.12470E+02
18	NVS02	[2]	3	5	0	3	3	0.59642E+01
19	NVS03	[2]	0	2	0	0	2	0.16000E+02
20	NVS04	[2]	0	2	0	0	0	0.72000E+00
21	NVS05	[2]	6	2	0	4	9	0.54709E+01
22	NVS06	[2]	0	2	0	0	0	0.17703E+01

(continued)

<i>no</i>	<i>name</i>	<i>ref</i>	$n_c$	$n_i$	$n_b$	$m_e$	$m$	$f(x^*, y^*)$
23	NVS07	[2]	0	3	0	0	2	0.40000E+01
24	NVS08	[2]	1	2	0	0	3	0.23450E+02
25	NVS09	[2]	0	10	0	0	0	-0.43134E+02
26	NVS10	[2]	0	2	0	0	2	-0.31080E+03
27	NVS11	[2]	0	3	0	0	2	-0.43100E+03
28	NVS12	[2]	0	4	0	0	4	-0.48120E+03
29	NVS13	[2]	0	5	0	0	5	-0.58520E+03
30	NVS14	[2]	3	5	0	3	3	-0.40358E+05
31	NVS15	[2]	0	3	0	0	1	0.10000E+01
32	NVS16	[2]	0	2	0	0	0	0.70312E+00
33	NVS17	[2]	0	7	0	0	7	-0.11004E+04
34	NVS18	[2]	0	6	0	0	6	-0.77840E+03
35	NVS19	[2]	0	8	0	0	8	-0.10984E+04
36	NVS20	[2]	11	5	0	0	8	0.23092E+03
37	NVS21	[2]	1	2	0	0	2	-0.56848E+01
38	NVS22	[2]	4	4	0	4	9	0.60582E+01
39	NVS23	[2]	0	9	0	0	9	-0.11252E+04
40	NVS24	[2]	0	10	0	0	10	-0.10332E+04
41	GEAR	[2]	0	4	0	0	0	0.10000E+01
42	GEAR2	[2]	4	24	0	4	4	0.10000E+01
43	GEAR3	[2]	4	4	0	4	4	0.10000E+01
44	GEAR4	[2]	2	4	0	1	1	0.16434E+01
45	WINDFAC	[2]	11	3	0	13	13	0.25449E+00
46	DG1	[6]	3	0	3	0	6	0.60097E+01
47	DG2	[6]	6	0	5	0	14	0.17214E+02
48	DG3	[6]	9	0	8	2	23	0.68010E+02
49	FLOUDAS1	[9]	2	0	3	2	5	0.76672E+01
50	FLOUDAS2	[9]	2	0	1	0	3	0.10765E+01
51	FLOUDAS3	[9]	3	0	4	0	9	0.45796E+01
52	FLOUDAS4	[9]	3	0	8	3	7	-0.94347E+00
53	FLOUDAS5	[9]	0	2	0	0	4	0.31000E+02
54	FLOUDAS6	[9]	1	1	0	0	3	-0.17000E+02
55	OAER	[2]	6	0	3	3	7	-0.19231E+01
56	SPRING	[2]	5	1	11	5	8	0.84625E+00
57	DAKOTA	[3]	2	2	0	0	2	0.13634E+01
58	PROB02	[2]	0	6	0	0	8	0.11224E+06
59	PROB03	[2]	0	2	0	0	1	0.10000E+02
60	PROB10	[2]	1	1	0	0	2	0.34455E+01
61	BATCH	[2]	23	0	24	12	73	0.28551E+00
62	BATCHDES	[2]	10	0	9	6	19	0.16743E+06

(continued)

<i>no</i>	<i>name</i>	<i>ref</i>	$n_c$	$n_i$	$n_b$	$m_e$	$m$	$f(x^*, y^*)$
63	DU_OPT5	[2]	7	13	0	0	9	0.80737E+01
64	DU_OPT	[2]	7	13	0	0	9	0.35563E+01
65	ST_E13	[2]	1	0	1	0	2	0.20000E+01
66	ST_E32	[2]	16	19	0	17	18	-0.14304E+01
67	ST_E36	[2]	1	1	0	1	2	-0.24600E+03
68	ST_E38	[2]	2	2	0	0	3	0.71977E+04
69	ST_MIQP1	[2]	0	0	5	0	1	0.28100E+03
70	ST_MIQP2	[2]	0	4	0	0	3	0.20000E+01
71	ST_MIQP3	[2]	0	2	0	0	1	-0.60000E+01
72	ST_MIQP4	[2]	3	0	3	0	4	-0.45740E+04
73	ST_MIQP5	[2]	5	2	0	0	13	-0.33389E+03
74	ST_TEST1	[2]	0	5	0	0	1	0.10000E+01
75	ST_TEST2	[2]	0	6	0	0	2	-0.92500E+01
76	ST_TEST3	[2]	0	13	0	0	10	-0.70000E+01
77	ST_TEST4	[2]	0	6	0	0	5	-0.70000E+01
78	ST_TEST5	[2]	0	10	0	0	11	-0.11000E+03
79	ST_TEST6	[2]	0	0	10	0	5	0.47100E+03
80	ST_TEST8	[2]	0	24	0	0	20	-0.29605E+05
81	ST_TESTGR1	[2]	0	10	0	0	5	-0.12812E+02
82	ST_TESTGR3	[2]	0	20	0	0	20	-0.20590E+02
83	ST_TESTPH4	[2]	0	3	0	0	10	-0.80500E+02
84	TLN2	[2]	0	6	2	0	12	0.53000E+01
85	TLN4	[2]	0	20	4	0	24	0.83000E+01
86	TLN5	[2]	0	30	5	0	30	0.10300E+02
87	TLN6	[2]	0	42	6	0	36	0.15300E+02
88	PROCSEL	[2]	7	0	3	4	7	-0.19231E+01
89	TLOSS	[2]	0	42	6	0	53	0.16300E+02
90	TLTR	[2]	0	36	12	0	54	0.48067E+02
91	ALAN	[2]	4	0	4	2	7	0.28990E+01
92	MEANVARX	[2]	21	0	14	8	44	0.14369E+02
93	HMITTELMANN	[2]	0	0	16	0	7	0.13000E+02
94	MIP_EX	[10]	2	0	3	0	7	0.35000E+01
95	MGRID_CYCLES1	[19]	0	5	0	0	1	0.80000E+01
96	MGRID_CYCLES2	[19]	0	10	0	0	1	0.30000E+03
97	CROP5	[17]	0	5	0	0	3	0.10041E+00
98	CROP20	[17]	0	20	0	0	3	0.13178E+00
99	CROP50	[17]	0	50	0	0	3	0.37459E+00
100	CROP100	[17]	0	100	0	0	3	0.15684E+01

### 3 The Fortran Subroutines

This section describes the organization of the Fortran subroutines and shows how to execute a test problem. Since it is assumed that at least a subset of the problems is used within a series of test runs for different optimization programs, the problems are coded

in a very flexible manner.

All 100 nonlinear mixed-integer test problems of our collection are available together with a test frame in form of Fortran source codes, see

<http://www.ai7.uni-bayreuth.de/home.htm>

To allow computation of relative errors in function values, a given optimal function value 0 (FEX) is replaced by one. The modified examples are GEAR, GEAR3, GEAR4, and NVS15. Examples DIRTY and BRAAK3 are very difficult to solve and required additional scaling of objective function. In two cases, GEAR4 and MEANVARX, we got better solutions than those found in the GAMS MINLPLib.

A test problem is identified by its name as used, e.g., in the GAMS MINLPLib, say <TP>. For each test problem, we provide a file with name <TP>.FOR by which problem data and objective and constraint function values are retrieved. To call a subset or all of them within a loop and to identify them by a problem number, a subroutine is included with name GET\_MINLP\_PROB, which has the same calling sequence plus one additional integer variable called IPROB for a test problem number between 1 and 100.

#### Usage:

```
CALL GET_MINLP_PROB (  MODE,  IPROB,   M,   ME,  MMAX,
/                      NCONT,  NBIN,  NINT,  NMAX,   X,
/                      XL,    XU,    F,    G,   PNAM,
/                      PREF,   FEX                      )
```

#### Parameter Definition:

- MODE : Status for returning data,  
0 : Returns M, ME, NCONT, NBIN, NINT, starting values in X, lower and upper bounds in XL and XU, the best known optimal objective function value in FEX, and documentation strings in PNAM and PREF.  
1 : Given M, ME, NCONT, NBIN, NINT and X, objective and constraint function values are computed subject to the variable values found in X, and returned in F and G(1), ..., G(M).
- IPROB : Input of an available problem number between 1 and 100, by which a specific test problem with this serial number is evaluated.
- M : Number of all constraints, without bounds.
- ME : Number of all equality constraints.
- MMAX : Dimension of G. MMAX has to be at least one and at least M for the largest problem to be executed.

NCONT :       Number of all continuous optimization variables.  
 NBIN :        Number of all binary variables.  
 NINT :        Number of all integer variables.  
 NMAX :        Dimension of X, XL, and XU. NMAX has to be at least two and at least  
               NCONT+NBIN+NINT for the largest problem to be executed.  
 X(NMAX) :     When called with MODE=0, X returns starting values. In the driving  
               program, the dimension of X must be equal to NMAX. X contains first  
               NCONT continuous, then NBIN boolean variables followed by NINT inte-  
               ger variables.  
 XL(NMAX),    On return, the one-dimensional arrays XL and XU contain the lower and  
 XU(NMAX):    upper bounds of the variables, first for the continuous, then for the binary  
               and subsequently for the integer variables.  
 F :           When called with MODE=1, the double precision parameter F returns the  
               objective function value computed at X.  
 G(MMAX) :     When called with MODE=1, the double precision array G contains the  
               constraint function values G(1),...,G(M) computed at X.  
 PNAME :       On return with MODE=0, PNAME contains the test problem name identical  
               to the subroutine and file name. The string length is 30.  
 PREF :        On return with MODE=0, PREF contains a Latex reference to biblio-  
               graphic data, as used for this documentation. The string length is 30.  
 FEX :         On return with MODE=0, FEX contains best known optimal objective  
               function value.

It is important that the values of M, ME, N, NBIN, NINT, XL, and XU must not be changed after the first call of GET\_MINLP\_PROB with MODE=0 for the same IPROB value.

Any of the individual test problems has the same calling sequence without the parameter IPROB, i.e.,

```

      CALL <TP> (  MODE ,      M,      ME,  MMAX,  NCONT,
/                NBIN,  NINT,  NMAX,      X,      XL,
/                XU,    F,      G,   PNAME,  PREF,
/                FEX
                )
  
```

To give an example, we consider test problem with number 56 called SPRING in the

GAMS test problem library MINLPLib [2],

$$\begin{aligned}
& \min (1.570796327 + 0.7853981635y_1) x_1 x_2^2 \\
& -\frac{x_1}{x_2} + x_4 = 0, \\
& -\frac{4x_4 - 1}{4x_4 - 4} + \frac{0.615}{x_4} + x_5 = 0, \\
& -6.95652173913044 \frac{y_1 x_4^3}{x_2} + x_3 = 0, \\
& x_2 - 0.207y_2 - 0.225y_3 - 0.244y_4 - 0.263y_5 - 0.283y_6 - 0.307y_7 \\
& \quad - 0.331y_8 - 0.362y_9 - 0.394y_{10} - 0.4375y_{11} - 0.5y_{12} = 0, \\
x \in \mathbb{R}^5, y \in \mathbb{N}^{12} : & \quad y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_{10} + y_{11} + y_{12} - 1 = 0, \\
& -2546.47908913782 \frac{x_5 x_4}{x_2^2} + 189000 \geq 0, \\
& -(2.1 + 1.05y_1)x_2 - 1000x_3 + 14 \geq 0, \\
& -x_1 - x_2 + 3 \geq 0, \\
& 0.414 \leq x_1 \leq 10, \quad 0.207 \leq x_2 \leq 10, \quad 0.0018 \leq x_3 \leq 0.02, \\
& 1.1 \leq x_4 \leq 10, \quad 0.1 \leq x_5 \leq 9.5, \quad 5 \leq x_5 \leq 10, \\
& 0 \leq y_1 \leq 10, \quad 0 \leq y_i \leq 1, i = 2, \dots, 11
\end{aligned}$$

We have five continuous, eleven binary, and one integer variable, moreover five non-linear equality and three inequality constraints. The code for test example SPRING is listed subsequently and is found in the file SPRING.FOR. Note that we follow the implementation of the GAMS MINLPLib as much as possible.

```

subroutine spring( mode,      m,      me,  mmax, ncont,
/                   nbin,  nint,   nmax,   x,    xl,
/                   xu,    f,      g,   pnam,  pref,
/                   fex )

implicit none
integer m, me,ncont, nint, nbin, n, nmax, mmax, mode, i
double precision x(nmax), y(1000), xl(nmax), xu(nmax), f, fex,
/      g(mmax), x1, x2, x3, i4, x5, x6, b7, b8, b9, b10, b11,
/      b12, b13, b14, b15, b16, b17, sqr
character*30 pnam, pref

if (mode.eq.0) then
    pnam = 'SPRING'

```

```

pref = '\cite{MINLPLib}'
fex = 0.8462252d0
ncont = 5
nint = 1
nbin = 11
n = ncont + nbin + nint
m = 8
me = 5
do i=1,ncont
  xl(i) = 0.0d0
  x(i) = 1.0d0
  xu(i) = 100.0d0
enddo
do i=ncont+1,ncont+nbin
  xl(i) = 0.0d0
  x(i) = 0.0d0
  xu(i) = 1.0d0
enddo
do i=ncont+nbin+1,n
  xl(i) = 0.0d0
  x(i) = 1.0d0
  xu(i) = 100.0d0
enddo
xl(1) = 0.414d0
xl(2) = 0.207d0
xl(3) = 0.00178571428571429d0
x(3) = 0.002d0
xu(3) = 0.02d0
xl(4) = 1.1d0
x(4) = 2.0d0
xl(5) = 1.0d0
x(5) = 2.0d0
xu(5) = 9.5d0
xl(n) = 1.0d0
x(n) = 7.0d0
xu(n) = 100.0d0
goto 999
endif

do i=1,nbin
  y(nint+i) = x(ncont+i)
enddo
do i=1,nint
  y(i) = x(ncont+nbin+i)
enddo

```

```

x1 = x(1)
x2 = x(2)
x3 = x(3)
x5 = x(4)
i4 = y(1)
x6 = x(5)
b7 = y(2)
b8 = y(3)
b9 = y(4)
b10 = y(5)
b11 = y(6)
b12 = y(7)
b13 = y(8)
b14 = y(9)
b15 = y(10)
b16 = y(11)
b17 = y(12)

f = (1.570796327d0 + 0.7853981635d0*i4)*x1*sqr(x2)

g(1) = - x1/x2 + x5
g(2) = - ((4.0d0*x5 - 1.0d0)/(4.0d0*x5 - 4.0d0) + 0.615d0/x5) + x6
g(3) = (-6.95652173913044d-7*i4*x5**3/x2 + x3)
g(4) = x2 - 0.207d0*b7 - 0.225d0*b8 - 0.244d0*b9 - 0.263d0*b10
/      - 0.283d0*b11 - 0.307d0*b12 - 0.331d0*b13 - 0.362d0*b14
/      - 0.394d0*b15 - 0.4375d0*b16 - 0.5d0*b17
g(5) = b7 + b8 + b9 + b10 + b11 + b12 + b13 + b14 + b15
/      + b16 + b17 - 1.0d0
g(6) = (-2546.47908913782d0*x6*x5/sqr(x2) + 189000.0d0)
g(7) = -(2.1d0 + 1.05d0*i4)*x2 - 1000.0d0*x3 + 14.0d0
g(8) = -x1 - x2 + 3.0d0

999 continue
return
end

```

The function SQR serves to compute squares of real number as in GAMS. The subsequent code shows how test example SPRING is executed from the framework given by subroutine SET\_MINLP\_PROB. It's serial number is 56. Alternatively, we can replace the statements by which SET\_MINLP\_PROB is called, by equivalent ones calling SPRING directly. The main program for executing MISQP by reverse communication can be implemented as follows,

```

implicit      none
integer       nmax, mmax, mmax0, maxnde, maxcut, lerw, leiw, lelw
parameter    (nmax = 1000,

```

```

/          mmax   = 3000,
/          maxcut = 500,
/          mmax0  = 2*mmax + maxcut + 20,
/          maxnde = 1000)
parameter (lerw = 7*nmax*nmax/2 + mmax0*nmax + 102*nmax
/          + 34*mmax0 + 3*maxnde + 3*mmax*mmax/2
/          + 4*mmax*nmax + 400,
/          leiw  = 14*nmax + 5*mmax0 + 6*maxnde + 105,
/          lelw  = 4*nmax + mmax0 + 100)
double precision x(nmax), g(mmax), df(nmax), dg(mmax,nmax),
/          xl(nmax), xu(nmax), geps(mmax), rw(lerw)
logical        lw(lelw),  ideriv(nmax), lopt(60)
character*30 pnam, pref
double precision f, feps, fex, acc, eps, xbck, ropt(60)
integer        m, me, n, ncont, nint, nbin, ifail, maxit, iprint,
/          iout, iprob, i, j, iw(leiw), iopt(60)

```

c Set test problem number and prepare initial data

```

iprob = 56
call get_minlp_prob(    0, iprob,    m,    me,    mmax,
/          ncont,    nbin,    nint,    nmax,    x,
/          xl,    xu,    f,    g,    pnam,
/          pref,    fex )

```

c or call SPRING directly

```

c    call spring(    0,    m,    me,    mmax,    ncont,
c /          nbin,    nint,    nmax,    x,    xl,
c /          xu,    f,    g,    pnam,    pref,
c /          fex )

```

c Set constants and tolerances for calling MISQP

```

do i = 1,60
  ropt(i) = -1.d0
  iopt(i) = -1
  lopt(i) = .true.
enddo
iout   = 6      ! output channel
iprint = 2      ! print flag
ifail  = 0      ! initialize flag
maxit  = 1000   ! maximum number of iterations
eps    = 1.0d-6 ! tolerance for forward differences
acc    = 1.0d-4 ! final termination tolerance
n = ncont + nbin + nint

```

```

do i=ncont+1,n
  nderiv(i) = .false.
enddo
write(iout,*)
write(iout,*) ' *** solving now ',pnam(1:10), ', fex =',fex

c   Begin of optimization block

c   -----
c   Call MISQP with reverse communication, integer variables treated as
c   non-relaxable

      ifail = 0
      1 continue

c   Evaluation of function values

      if ((ifail.eq.0).or.(ifail.eq.-1)) then
        call get_minlp_prob( 1, iprob, m, me, mmax,
/                               ncont, nbin, nint, nmax, x,
/                               xl, xu, f, g, pnam,
/                               pref, fex )

c   or call SPRING directly

c      call spring( 1, m, me, mmax, ncont,
c /                nbin, nint, nmax, x, xl,
c /                xu, f, g, pnam, pref,
c /                fex )
      endif

c   Approximation of partial derivatives subject to continuous
c   variables by forward differences

      if ((ifail.eq.0).or.(ifail.eq.-2)) then
        do i = 1, ncont
          xbck = x(i)
          x(i) = x(i) + eps
          call get_minlp_prob( 1, iprob, m, me, mmax,
/                               ncont, nbin, nint, nmax, x,
/                               xl, xu, feps, geps, pnam,
/                               pref, fex )

c   or call SPRING directly

c      call spring( 1, m, me, mmax, ncont,

```

```

c /          nbin,  nint,  nmax,    x,    xl,
c /          xu,   feps,  geps,   pnam,  pref,
c /          fex )
      df(i) = (feps - f)/eps
      do j = 1, m
          dg(j,i) = (geps(j) - g(j))/eps
      enddo
      x(i) = xbck
    enddo
  endif

c Call driving routine

      call MISQP(      m,      me,  mmax,      n,  nbin,
/          nint,      x,      f,      g,      df,
/          dg,      xl,      xu,      acc,  maxit,
/          maxcut, maxnde, iprint,  iout,  ifail,
/          ideriv,  ropt,  iopt,  lopt,  rw,
/          lerw,  iw,  leiw,  lw,  lelw )
      if (ifail.lt.0) goto 1

c End of optimization block
c -----

      stop
      end

```

The subsequent output is generated by MISQP, which needs 48 iterations including one restart. Termination accuracy is  $10^{-8}$ .

```

*** solving now SPRING      , fex = 0.846245700000000

```

```

-----
START OF THE MIXED-INTEGERSQP CODE MISQP
-----

```

Parameters:

Number of all variables:	17	N
Number of continuous variables:	5	NCONT
Number of binary variables:	11	NBIN
Number of integer variables:	1	NINT
Total number of constraints:	8	M

```

Number of equality constraints:          5  ME
Termination accuracy:                  0.100D-03  ACC
Maximum number of iterations:          1000  MAXIT
Maximum number of QP cuts:             500  MAXCUT
Maximum number of nodes:               1000  MAXNDE
Output level:                           2  IPRINT

```

Output in the following order:

```

IT      - iteration number
F       - objective function value
MCV    - maximum constraint violation
SIGMA  - penalty parameter
IL     - number inner loops
DMAXC  - maximum norm of continuous step D_C
D1B    - 1-norm of binary step DELTA_B
DMAXI  - maximum norm of integer step D_I

```

IT	F	MCV	SIGMA	IL	DMAXC	D1B	DMAXI
0	0.70686D+01	0.10D+01	0.10D+04	0	0.00D+00	0.00D+00	0.00D+00
1	0.15558D+01	0.10D+01	0.20D+06	3	0.40D-01	0.55D-11	0.50D-12
2	0.48059D+00	0.10D+01	0.40D+08	2	0.20D-01	0.00D+00	0.00D+00
3	0.46317D+00	0.89D+00	0.40D+08	1	0.79D-01	0.10D+01	0.10D+01
4	0.39586D+00	0.77D+00	0.40D+08	2	0.65D-01	0.00D+00	0.00D+00
5	0.30690D+00	0.19D+00	0.40D+08	1	0.32D-01	0.00D+00	0.00D+00
6	0.33670D+00	0.12D-01	0.40D+08	1	0.65D-01	0.20D+01	0.10D+01
7	0.44316D+00	0.15D-02	0.40D+08	1	0.53D-01	0.00D+00	0.40D+01
8	0.75466D+00	0.11D-02	0.40D+08	1	0.34D-02	0.20D+01	0.16D+02
9	0.11335D+01	0.56D-03	0.40D+08	1	0.17D-01	0.00D+00	0.16D+02
10	0.11406D+01	0.61D-03	0.40D+09	1	0.12D-01	0.00D+00	0.00D+00
11	0.13074D+01	0.40D-01	0.40D+09	1	0.25D+00	0.00D+00	0.00D+00
12	0.13035D+01	0.18D-04	0.40D+09	1	0.52D-01	0.00D+00	0.00D+00
13	0.10916D+01	0.65D-02	0.40D+09	1	0.86D-01	0.00D+00	0.10D+02
14	0.11474D+01	0.11D-04	0.40D+09	1	0.13D-01	0.00D+00	0.20D+01
15	0.11466D+01	0.66D-06	0.40D+09	1	0.24D-02	0.00D+00	0.00D+00
16	0.11429D+01	0.16D-04	0.40D+09	1	0.98D-02	0.00D+00	0.00D+00
17	0.11425D+01	0.15D-06	0.40D+09	1	0.89D-03	0.00D+00	0.00D+00
18	0.11132D+01	0.48D-04	0.40D+09	2	0.00D+00	0.00D+00	0.10D+01
19	0.11425D+01	0.15D-10	0.40D+09	1	0.30D-06	0.00D+00	0.10D+01
20	0.10839D+01	0.97D-04	0.40D+09	2	0.30D-06	0.00D+00	0.20D+01
21	0.11425D+01	0.30D-10	0.40D+09	1	0.30D-06	0.00D+00	0.20D+01
22	0.79477D+00	0.20D+01	0.40D+09	2	0.21D+00	0.40D+01	0.18D+02
23	0.22373D+00	0.14D+00	0.40D+09	1	0.35D+00	0.40D+01	0.18D+02
24	0.84947D+00	0.46D-05	0.40D+09	1	0.15D+00	0.00D+00	0.80D+01
25	0.84626D+00	0.75D-05	0.40D+09	1	0.87D-02	0.00D+00	0.00D+00

26	0.60444D+00	0.85D-01	0.40D+09	1	0.13D+00	0.20D+01	0.60D+01
27	0.42995D+00	0.33D+00	0.40D+09	1	0.21D+00	0.20D+01	0.50D+01
28	0.70019D+00	0.36D+00	0.40D+09	1	0.37D+00	0.20D+01	0.40D+01
29	0.41927D+00	0.39D+00	0.40D+09	1	0.40D+00	0.20D+01	0.60D+01
30	0.98325D+00	0.28D+00	0.40D+09	1	0.46D+00	0.20D+01	0.00D+00
31	0.94526D+00	0.11D-01	0.40D+09	1	0.40D+00	0.00D+00	0.10D+01
32	0.88998D+00	0.22D-03	0.40D+09	1	0.83D-02	0.00D+00	0.10D+01
33	0.48221D+00	0.97D-01	0.40D+09	1	0.16D+00	0.20D+01	0.80D+01
34	0.61501D+00	0.11D+00	0.40D+09	1	0.15D+00	0.20D+01	0.50D+01
35	0.77897D+00	0.30D-01	0.40D+09	1	0.12D+00	0.20D+01	0.70D+01
36	0.68060D+00	0.64D-01	0.40D+09	1	0.17D+00	0.20D+01	0.70D+01
37	0.84679D+00	0.43D-05	0.40D+09	1	0.44D-01	0.00D+00	0.20D+01
38	0.84625D+00	0.22D-06	0.40D+09	1	0.33D-03	0.00D+00	0.00D+00
39	0.60444D+00	0.85D-01	0.40D+09	1	0.13D+00	0.20D+01	0.60D+01
40	0.85900D+00	0.17D-05	0.40D+09	2	0.30D-01	0.00D+00	0.20D+01
41	0.85928D+00	0.37D-07	0.40D+09	1	0.17D-03	0.00D+00	0.00D+00
42	0.76266D+00	0.20D+01	0.40D+09	2	0.84D-01	0.40D+01	0.20D+01
43	0.83539D+00	0.43D-02	0.40D+09	1	0.44D-01	0.40D+01	0.20D+01
44	0.84637D+00	0.77D-04	0.40D+09	1	0.52D-02	0.00D+00	0.00D+00
45	0.84625D+00	0.11D-07	0.40D+09	1	0.16D-03	0.00D+00	0.00D+00
46	0.60444D+00	0.85D-01	0.40D+09	1	0.13D+00	0.20D+01	0.60D+01
47	0.42995D+00	0.33D+00	0.40D+09	1	0.21D+00	0.20D+01	0.50D+01
48	0.70019D+00	0.36D+00	0.40D+09	1	0.37D+00	0.20D+01	0.40D+01
49	0.41928D+00	0.39D+00	0.40D+09	1	0.40D+00	0.20D+01	0.60D+01
50	0.98325D+00	0.28D+00	0.40D+09	1	0.46D+00	0.20D+01	0.00D+00
51	0.71922D+00	0.57D-01	0.40D+09	1	0.28D+00	0.20D+01	0.60D+01
52	0.48292D+00	0.32D+00	0.40D+09	1	0.20D+00	0.20D+01	0.50D+01
53	0.67367D+00	0.31D+00	0.40D+09	1	0.30D+00	0.20D+01	0.50D+01
54	0.86602D+00	0.49D-03	0.40D+09	1	0.15D+00	0.00D+00	0.10D+01
55	0.85933D+00	0.22D-04	0.40D+09	2	0.43D-02	0.00D+00	0.00D+00
56	0.85928D+00	0.14D-08	0.40D+09	1	0.49D-04	0.00D+00	0.00D+00

--- FINAL CONVERGENCE ANALYSIS ---

Objective function value:      F(X) = 0.84624568D+00  
Approximation of solution:      X =  
0.12230410D+01 0.28300000D+00 0.17857143D-02 0.43216998D+01  
0.13680931D+01 0.00000000D+00 0.00000000D+00 0.00000000D+00  
0.00000000D+00 0.10000000D+01 0.00000000D+00 0.00000000D+00  
0.00000000D+00 0.00000000D+00 0.00000000D+00 0.00000000D+00  
0.90000000D+01  
Constraint function values:      G(X) =  
-0.18918200D-11 -0.10600015D-07 -0.10831486D-09 -0.54783955D-12  
0.00000000D+00 0.10088103D+04 0.89456357D+01 0.14939590D+01  
Distances from lower bounds:      XL-X =  
-0.80904103D+00 -0.76000000D-01 0.00000000D+00 -0.32216998D+01

```

-0.36809312D+00  0.00000000D+00  0.00000000D+00  0.00000000D+00
 0.00000000D+00 -0.10000000D+01  0.00000000D+00  0.00000000D+00
 0.00000000D+00  0.00000000D+00  0.00000000D+00  0.00000000D+00
-0.80000000D+01
Distances from upper bounds:      XU-X =
 0.98776959D+02  0.99717000D+02  0.18214286D-01  0.95678300D+02
 0.81319069D+01  0.10000000D+01  0.10000000D+01  0.10000000D+01
 0.10000000D+01  0.00000000D+00  0.10000000D+01  0.10000000D+01
 0.10000000D+01  0.10000000D+01  0.10000000D+01  0.10000000D+01
 0.91000000D+02
Number of function calls:          NFUNC =      809
- within TR method:              NF_TR =      69
- integer derivatives:            NF_2D =     740
Number of gradient calls:         NGRAD =      57
Number of calls of QP solver:     NQL =      87
- 2nd order corrections:         NQL2 =      7
Number of B&B nodes:              NODES =    1659
Termination reason:              IFAIL =      0

```

## 4 Numerical Results

A summary of numerical results of MISQP of Exler, Lehmann, and Schittkowski [8] for the 100 test problems under consideration are summarized below. With the default tolerances given, all problems can be solved successfully, i.e., terminated with IFAIL=0. The results have been obtained on Intel(R)Core(TM) i7 CPU 860 with 2.8 GHz, Windows 7, and the Intel(R) Visual Fortran Compiler, Version 10.1.021, 64 bit. The following data are displayed:

- no* - serial number
- name* - test problem name (PNAM)
- n<sub>grad</sub>* - number of gradient evaluations subject to the continuous variables, if exist,
- n<sub>func</sub>* - number of equivalent function calls, i.e., all function calls including those needed for used for approximating partial derivatives,
- f(x\*, y\*)* - final objective function value,
- e(x\*, y\*)* - relative error of objective function value subject to the best known one,
- r(x\*, y\*)* - constraint violation at final solution,
- time* - average execution times in seconds.

Table 3: Individual Test Results for Mixed-Integer Problems

<i>no</i>	<i>name</i>	$n_{grad}$	$n_{func}$	$f(x^*, y^*)$	$e(x^*, y^*)$	$r(x^*, y^*)$	$time$		
1	MITP1	0	32	281	-0.100097E+05	0.00E+00	0.00E+00	0.02	
2	QIP1	0	0	22	-0.200000E+02	0.00E+00	0.00E+00	0.00	
3	MITP2	0	8	48	0.350000E+01	-0.95E-10	0.12E-09	0.00	
4	ASAADI11	0	17	115	-0.409574E+02	-0.20E-04	0.47E-10	0.00	
5	ASAADI12	0	0	420	-0.380000E+02	0.00E+00	0.00E+00	0.00	
6	ASAADI21	0	28	338	0.694903E+03	0.48E-08	0.00E+00	0.00	
7	ASAADI22	0	0	440	0.700000E+03	0.00E+00	0.00E+00	0.02	
8	ASAADI31	0	20	332	0.372195E+02	-0.18E-05	0.63E-12	0.02	
9	ASAADI32	0	0	202	0.430000E+02	0.00E+00	0.00E+00	0.02	
10	DIRTY	0	157	5454	-0.304704E+09	0.64E-04	0.00E+00	0.27	
11	BRAAK1	0	60	707	0.100677E+01	0.68E-02	0.00E+00	0.02	
12	BRAAK2	0	223	2446	-0.271828E+01	-0.26E-06	0.36E-10	0.03	
13	BRAAK3	0	50	518	-0.196559E+07	0.66E-05	0.00E+00	0.02	
14	DEX2	0	0	33	-0.569375E+02	0.00E+00	0.00E+00	0.00	
15	TP83	0	18	126	-0.306655E+05	0.58E-07	0.25E-07	0.00	
16	WP02	0	9	35	-0.244444E+01	-0.15E-05	0.00E+00	0.00	
17	NVS01	0	18	108	0.124697E+02	-0.95E-07	0.59E-14	0.00	
18	NVS02	0	27	363	0.596418E+01	-0.80E-07	0.27E-12	0.02	
19	NVS03	0	0	37	0.160000E+02	0.00E+00	0.00E+00	0.00	
20	NVS04	0	0	31	0.720000E+00	0.83E-14	0.00E+00	0.00	
21	NVS05	0	60	638	0.547093E+01	0.19E-07	0.13E-08	0.02	
22	NVS06	0	0	195	0.177031E+01	0.28E-06	0.00E+00	0.00	
23	NVS07	0	0	22	0.400000E+01	0.00E+00	0.00E+00	0.00	
24	NVS08	0	37	235	0.234497E+02	-0.13E-06	0.12E-06	0.00	
25	NVS09	0	0	32	-0.431343E+02	0.71E-07	0.00E+00	0.00	
26	NVS10	0	0	130	-0.310800E+03	-0.18E-15	0.00E+00	0.00	
27	NVS11	0	0	296	-0.431000E+03	0.00E+00	0.00E+00	0.02	
28	NVS12	0	0	99	-0.481200E+03	-0.12E-15	0.00E+00	0.00	
29	NVS13	0	0	806	-0.585200E+03	-0.19E-15	0.00E+00	0.02	
30	NVS14	0	21	280	-0.403582E+05	-0.12E-06	0.43E-13	0.00	
31	NVS15	0	0	33	0.100000E+01	0.00E+00	0.00E+00	0.00	
32	NVS16	0	0	28	0.703125E+00	0.00E+00	0.00E+00	0.00	
33	NVS17	0	0	917	-0.110040E+04	0.21E-15	0.00E+00	0.03	
34	NVS18	0	0	601	-0.778400E+03	0.15E-15	0.00E+00	0.02	
35	NVS19	0	0	1123	-0.109840E+04	0.00E+00	0.00E+00	0.05	
36	NVS20	0	65	1197	0.230929E+03	0.29E-04	0.14E-10	0.03	
37	NVS21	0	27	164	-0.568478E+01	0.88E-07	0.77E-10	0.00	
38	NVS22	0	18	209	0.605822E+01	0.00E+00	0.38E-12	0.02	
39	NVS23	0	0	1235	-0.112520E+04	-0.20E-15	0.00E+00	0.06	

(continued)

$i_p$	$i_{fail}$	$n_{grad}$	$n_{equ}$	$f(x^*, y^*)$	$e(x^*, y^*)$	$r(x^*, y^*)$	$time$	
40	NVS24	0	0	873	-0.102380E+04	0.91E-02	0.00E+00	0.03
41	GEAR	0	0	448	0.100000E+01	0.45E-07	0.00E+00	0.02
42	GEAR2	0	39	1132	0.100000E+01	0.41E-06	0.26E-11	1.65
43	GEAR3	0	78	959	0.100000E+01	0.14E-08	0.81E-10	0.05
44	GEAR4	4	186	2188	0.100000E+01	-0.39E+00	0.44E-05	1.19
45	WINDFAC	0	41	744	0.254487E+00	-0.36E-06	0.19E-08	0.03
46	DG1	0	21	147	0.600876E+01	-0.16E-03	0.87E-06	0.00
47	DG2	0	52	631	0.172142E+02	0.27E-05	0.50E-11	0.03
48	DG3	0	52	938	0.680097E+02	-0.38E-05	0.50E-11	0.06
49	FLOUDAS1	0	4	24	0.766718E+01	-0.33E-08	0.16E-08	0.00
50	FLOUDAS2	0	7	28	0.107654E+01	0.29E-05	0.55E-12	0.00
51	FLOUDAS3	0	34	278	0.457958E+01	-0.98E-07	0.17E-06	0.02
52	FLOUDAS4	0	59	713	-0.943471E+00	-0.39E-10	0.46E-08	0.05
53	FLOUDAS5	0	0	17	0.310000E+02	0.00E+00	0.00E+00	0.00
54	FLOUDAS6	0	8	32	-0.833333E+01	0.51E+00	0.00E+00	0.00
55	OAER	0	17	170	-0.192310E+01	0.25E-06	0.68E-09	0.00
56	SPRING	0	66	1257	0.846246E+00	-0.40E-07	0.13E-09	0.11
57	DAKOTA	0	16	109	0.136340E+01	-0.28E-12	0.31E-13	0.00
58	PROB02	0	0	130	0.112235E+06	0.00E+00	0.00E+00	0.00
59	PROB03	0	0	15	0.100000E+02	0.00E+00	0.00E+00	0.00
60	PROB10	0	4	16	0.344550E+01	0.13E-10	0.00E+00	0.00
61	BATCH	0	179	8598	0.285506E+06	0.16E-05	0.36E-07	6.63
62	BATCHDES	0	30	600	0.239960E+06	0.43E+00	0.10E-10	0.05
63	DU_OPT5	0	197	4703	0.258751E+02	0.22E+01	0.00E+00	0.58
64	DU_OPT	0	360	9679	0.604305E+01	0.70E+00	0.00E+00	0.34
65	ST_E13	0	4	12	0.223607E+01	0.12E+00	0.00E+00	0.00
66	ST_E32	0	16	631	-0.143041E+01	0.12E-07	0.35E-11	0.19
67	ST_E36	0	62	266	-0.168310E+03	0.32E+00	0.52E-06	0.00
68	ST_E38	0	57	389	0.719773E+04	0.61E-12	0.20E-12	0.00
69	ST_MIQP1	0	0	36	0.281000E+03	0.00E+00	0.00E+00	0.00
70	ST_MIQP2	0	0	65	0.200000E+01	0.00E+00	0.00E+00	0.00
71	ST_MIQP3	0	0	13	-0.600000E+01	0.00E+00	0.00E+00	0.00
72	ST_MIQP4	0	4	28	-0.457400E+04	-0.16E-09	0.30E-08	0.00
73	ST_MIQP5	0	4	42	-0.333889E+03	0.33E-07	0.17E-09	0.00
74	ST_TEST1	0	0	6	0.100000E+01	0.00E+00	0.00E+00	0.00
75	ST_TEST2	0	0	42	-0.925000E+01	0.00E+00	0.00E+00	0.00
76	ST_TEST3	0	0	59	-0.700000E+01	0.00E+00	0.00E+00	0.02
77	ST_TEST4	0	0	45	-0.700000E+01	0.00E+00	0.00E+00	0.00
78	ST_TEST5	0	0	44	-0.110000E+03	0.00E+00	0.00E+00	0.00
79	ST_TEST6	0	0	88	0.471000E+03	0.00E+00	0.00E+00	0.00

(continued)

$i_p$	$i_{fail}$	$n_{grad}$	$n_{equ}$	$f(x^*, y^*)$	$e(x^*, y^*)$	$r(x^*, y^*)$	$time$	
80	ST_TEST8	0	0	494	-0.296050E+05	0.00E+00	0.00E+00	0.09
81	ST_TESTGR1	0	0	134	-0.127976E+02	0.11E-02	0.00E+00	0.02
82	ST_TESTGR3	0	0	259	-0.205900E+02	0.00E+00	0.00E+00	0.14
83	ST_TESTPH4	0	0	24	-0.805000E+02	0.00E+00	0.00E+00	0.00
84	TLN2	0	0	106	0.530000E+01	0.00E+00	0.00E+00	0.02
85	TLN4	0	0	1994	0.830000E+01	0.00E+00	0.00E+00	3.43
86	TLN5	0	0	7158	0.112000E+02	0.87E-01	0.00E+00	21.08
87	TLN6	0	0	1615	0.163000E+02	0.65E-01	0.00E+00	3.84
88	PROCSEL	0	32	353	-0.192310E+01	0.14E-06	0.77E-11	0.00
89	TLOSS	0	0	1470	0.163000E+02	0.00E+00	0.00E+00	4.96
90	TLTR	0	0	309	0.480667E+02	-0.74E-15	0.00E+00	0.34
91	ALAN	0	48	450	0.292500E+01	0.90E-02	0.10E-07	0.02
92	MEANVARX	0	43	1554	0.141897E+02	-0.12E-01	0.27E-11	0.41
93	HMITTELMANN	0	0	52	0.160000E+02	0.23E+00	0.00E+00	0.03
94	MIP_EX	0	6	36	0.350000E+01	0.00E+00	0.00E+00	0.00
95	MGRID_CYCLES1	0	0	77	0.800000E+01	0.00E+00	0.00E+00	0.00
96	MGRID_CYCLES2	0	0	1216	0.300000E+03	0.00E+00	0.00E+00	0.06
97	CROP5	0	0	227	0.100409E+00	-0.51E-05	0.00E+00	0.00
98	CROP20	0	0	3112	0.148424E+00	0.13E+00	0.00E+00	2.48
99	CROP50	0	0	7640	0.374595E+00	0.13E-04	0.00E+00	8.39
100	CROP100	0	0	6874	0.170283E+01	0.86E-01	0.00E+00	13.17

## References

- [1] Asaadi J. (1973): *A computational comparison of some non-linear programs*, Mathematical Programming, Vol. 4, 144–154
- [2] Bussieck M.R., Drud A.S., Meeraus A. (2007): *MINLPLib - A collection of test models for mixed-integer nonlinear programming*, GAMS Development Corp., Washington D.C., USA
- [3] Eldred M.S., Giunta A.A., van Bloemen Waanders B.G., Wojtkiewicz S.F., Hart W.E., Alleva M.D. (2002): *DAKOTA: A multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis, Version 3.1 Users Manual*, Report SAND20001-3796, Sandia National Laboratories, PO Box 5800, Albuquerque, NM 87185-0847
- [4] van de Braak G. (2001): *Das Verfahren MISQP zur gemischt ganzzahligen nicht-linearen Programmierung für den Entwurf elektronischer Bauteile*, Diploma Thesis, Department of Numerical and Instrumental Mathematics, University of Münster, Germany
- [5] Cha J.Z., Mayne R.W. (1989): *Optimization with discrete variables via recursive quadratic programming: Part 2 - algorithms and results*, Transactions of the ASME,

- Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 111, 130–136
- [6] Duran M., Grossmann I.E. (1986): *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Mathematical Programming, Vol. 36, 307–339
  - [7] Exler O., Lehmann T., Schittkowski K. (2009): *MISQP: A Fortran implementation of a trust region SQP algorithm for mixed-integer nonlinear programming - User's guide*, Report, Department of Computer Science, University of Bayreuth, Germany
  - [8] Exler O., Lehmann T., Schittkowski K. (2009): *A comparative study of SQP-type algorithms for nonlinear and nonconvex mixed-integer optimization*, submitted for publication
  - [9] Floudas C.A., Pardalos P.M., Adjiman C.S., Esposito W.R., Gumus Z.H., Harding S.T., Klepeis J.L., Meyer C.A., Schweiger C.A. (1999): *Handbook of Test Problems in Local and Global Optimization*, Kluwer Academic Publishers
  - [10] Grossmann I.E., Kravanja Z. (1997): *Mixed-integer nonlinear programming: A survey of algorithms and applications*, in: Conn A.R., Biegler L.T., Coleman T.F., Santosa F.N. (eds.): *Large-Scale Optimization with Applications, Part II: Optimal Design and Control*, Springer, New York, Berlin
  - [11] Hock W., Schittkowski K. (1981): *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer
  - [12] Lehmann T., Schittkowski K. (2009): *MISQPOA: A Fortran subroutine for mixed-integer nonlinear optimization by outer approximations - user's guide*, Report, Department of Computer Science, University of Bayreuth
  - [13] Lehmann T., Schittkowski K. (2009): *MINLP4: A Fortran subroutine for mixed-integer nonlinear optimization by branch-and-bound - user's guide*, Report, Department of Computer Science, University of Bayreuth
  - [14] Lehmann T., Schittkowski K. (2009): *MISQPN: A Fortran subroutine for nonlinear mixed-integer optimization by outer approximations and mixed-integer search steps - user's guide*, Report, Department of Computer Science, University of Bayreuth
  - [15] Maniezzo V., Stützle T., Voß S. (2009): *A good recipe for solving MINLPs*, in: Metaheuristics, Ser. Annals of Information Systems, Vol. 10, 231-244, Springer
  - [16] Still C., Westerlund T. (2006): *Solving convex MINLP optimization problems using a sequential cutting plane algorithm*, Computational Optimization and Applications, Vol. 34, 63-83
  - [17] Sun X., Ruan N., Li D. (2006): *An efficient algorithm for nonlinear integer programming problems arising in series-parallel reliability systems*, Optimization Methods and Software, Vol. 21 617-634

- [18] Westerlund T., Pörn R. (2002): *Solving pseudo-convex mixed integer optimization problems by cutting plane techniques*, Optimization and Engineering, Vol. 3, 253-280
- [19] Thekale A., Gradl T., Klamroth K., Rüdè U. (2009): *Optimizing the number of multigrid cycles in the full multigrid algorithm*, Lehrstuhlbericht 09-5, Friedrich-Alexander-Universität Erlangen-Nürnberg, Institut für Informatik