OPM, a collection of Optimization Problems in Matlab

Gratton, Serge; TOINT, Philippe

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

OPM, a collection of Optimization Problems in Matlab

S. Gratton* and Ph. L. Toint†

6 XII 2021

Abstract

OPM is a small collection of CUTEst unconstrained and bound-constrained nonlinear optimization problems, which can be used in Matlab for testing optimization algorithms directly (i.e. without installing additional software).

Keywords: nonlinear optimization, test problems, Matlab, CUTEst.

1 Introduction

The purpose of this short report is to introduce OPM, a small collection of unconstrained and bound-constrained nonlinear optimization problems, suitable for testing optimization algorithms written in Matlab(1). While other collections of this type are available (we obviously think of the extensive and widely used CUTEst [4] collection but also of [7, 9, 2, 1]), none of these is directly accessible in Matlab, and their interface with Matlab code is often system dependent. The collection presented here has no ambition of being as extensive or as functional as CUTEst, which we believe remains an authoritative source of test problems today, but it may help Matlab optimization developpers by providing a quick installation-free access to a small fraction of the CUTEst problems and functionalities.

2 The collection

The OPM collection consists of

1. a set of Matlab executable files (*.m), each file corresponding to a specific problem of the CUTEst collection (for instance, the `rosenbr.m` file corresponds to ROSENBR.SIF, the famous Rosenbrock banana function);
2. the `opm_eval_cpsf.m` file, which provides the mechanism for using the other files;

These files are available on `https://github.com/gratton7/OPM`. Each problem files starts with a brief description of the problem and a reference to its source (both extracted from the corresponding CUTEest SIF file).

*Université de Toulouse, INP, IRIT, Toulouse, France. Email: serge.gratton@enseeiht.fr Work partially supported by 3IA Artificial and Natural Intelligence Toulouse Institute, French "Investing for the Future - PIA3" program under the Grant agreement ANR-19-PI3A-0004"

†NAXYS, University of Namur, Namur, Belgium. Email: philippe.toint@unamur.be

(1)Matlab® is a trade mark of MathWorks, Inc.

At the time of writing, the collection contains 134 available problems in continuous variables that unconstrained or bound constrained (including problems with fixed variables), i.e.,

$$\min_{\substack{x\in\mathbb{R}^n \\ x_{\rm low}\leq x\leq x_{\rm up}}} \quad f(x).$$

It includes the problems given in [7] and [2] as well as selected problems from [9] and [1], all of which can also be found in CUTEst [4].

3 A simple access to the OPM problem files

In its simplest form the use of the OPM problems consists of two stages, which we illustrate by using the `rosenbr` problem.

1. The first stage is to specify the problem dimension (when possible), obtain the standard starting point and, if relevant, the lower and upper bounds on the problems' variables. This achieved by simply inserting the following instruction is the user's Matlab program

   ```
   x0 = rosenbr( 'setup',10 );
   ```

   In this case, the problem dimension (that is the number of variables) is set to 10. The second input argument is optional: the simpler call

   ```
   x0 = rosenbr( 'setup' );
   ```

   assigns the default dimension to the problem (2 for `rosenbr`). Setting the problem dimension is possible for the `rosenbr` problem, but not necesarily for every problem. When possible, there may be additional problem-dependent constraints on how dimension can be chosen[(2)].

2. The value of the objective function, gradient and Hessian at a given vector `x` for the Rosenbrock problem are then computed, in the course of the user's optimization code, by

   ```
   fx = rosenbr( 'objf', x );
   ```

   if only the objective-function value at `x` is requested, or

   ```
   [ fx, gx ] = rosenbr( 'objf', x );
   ```

   if the objective-function value and its gradient at `x` are requested, or finally

   ```
   [ fx, gx, Hx ] = rosenbr( 'objf', x );
   ```

   if value, gradient and Hessian at `x` are requested. If the above, `fx` is a scalar, `x` and `gx` are column vectors of size n, say, and `Hx` is an $n \times n$ square matrix.

There are variations on this simple calling sequences. In particular, bounds on the variables can also be retrieved, as we explain in the next section.

[(2)] Check the problem m-file to verify the acceptable choices.

4 An access to problems exploiting their coordinate-partially-separable structure

All OPM test problems are, just as a substantial fraction of real problems from applications, "coordinate-partially-separable" (CPS). A CPS problem is a problem whose objective function may be written in the form

$$f(x) = \sum_{i=1}^{n_e} f_i(x_{\mathcal{E}_i}), \tag{4.1}$$

where the sets $\mathcal{E}_i$ are subsets of $\{1,\ldots,n\}$ and $x_{\mathcal{E}_i}$ is the subvector of x indexed by $\mathcal{E}_i$. In other words, f is the sum of n_e *element functions* f_i, each one of them only depending on a subset of the variables. For large problems, it very often the case that $|\mathcal{E}_i|$, the size of $\mathcal{E}_i$, is independent of the problem dimension and $\max_i |\mathcal{E}_i| \ll n$, as it can be shown [5] that every sufficiently smooth function whose Hessian matrix is sparse if a CPS function. (Indeed, the $|\mathcal{E}_i|$ are the dimensions of the Hessian's dense principal submatrices.) In (4.1), the f_i are called the *element functions*, the $\mathcal{E}_i$ the *element domains* and (4.1) the *CPS decomposition* of the function f.

A quick look at the `opm_eval_cpsf.m` file reveals that this Matlab function merely builds the sum (4.1). But OPM also provides the means to access the element functions f_i (and their gradients and Hessians) individually. For example, the following sequence of calls applies to the `lminsurf` problem: one starts by setting up the problem and retrieving its structure in a Matlab `struct` called `cpsstr`[3]. This is done in the following calling sequence:

```
x0 = lminsurf( 'setup' );
cpsstr = lminsurf( 'cpsstr', length(x0) );
```

In particular, its element domains are given in the cell `cpsstr.eldom` whose i-th entry is a vector containing the indices of $\mathcal{E}_i$. Other structure parameters are given in the cell `cpsstr.param`, when relevant. Suppose now that, in the course of the user's program, the value of the `i`-th element function (for `i` $\in \{1,\ldots,n_e\}$), its gradient and Hessian are needed at the vector of variables `x`. These values are simply obtained by the call

```
[ fix, gix, Hix ] = ...
      lminsurf( 'elobjf', i, x( cpsstr.eldom{i} ), cpsstr.param{:} );
```

Note that `gix` is a column vector of length $|\mathcal{E}_i|$ =`length(cpsstr.eldom{i})` giving the first derivatives of f_i with respect to the variables whose index is in $\mathcal{E}_i$. Similarly, `Hix` is a symmetric square matrix of dimension $|\mathcal{E}_i| \times |\mathcal{E}_i|$ containing the second derivatives of f_i with respect to these variables. If the i-th element function involves p numerical parameters `par1` to `parp`, a call of the form

```
[ fix, gix, Hix ] = ...
      lminsurf( 'elobjf', i, x( cpsstr.eldom{i} ),
           cpsstr.param{:}, par1,..., parp );
```

is allowed.

The use of the CPS structure may be numerically extremely advantageous for large problems where $\max_i |\mathcal{E}_i| \ll n$, as it gives direct access to the sparsity structure of the objective

[3] For CPS structure.

function's Hessian. It is also possible to use the element gradients to construct 'elementwise quasi-Newton' methods(4). In model-based derivative-free algorithms, element-wise models can be used with considerable success [3, 8].

5 An alternative calling sequence

The reader might wonder if obtaining the problem structure by calling the problem with first argument '`cpsstr`', as explained in the previous section, might be useful even if access to individual element functions is not desired. And (you guessed it), the answer is positive. Instead to the simple calls

```
x0 = rosenbr( 'setup',10 );
...
[ fx, gx, Hx ] = rosenbr( 'objf', x );
```

may be replaced by the barely more complex

```
x0 = rosenbr( 'setup', 10 );
cpsstr = rosenbr( 'cpsstr', 10 );
...
[ fx, gx, Hx ] = rosenbr( 'objf', x, cpsstr );
```

In effect, this last calling sequence computes the problem's structure (in `cpsstr`) just once, and then passes it to every subsequent call for calculating the objective function's value (and derivatives). By contrast, the first calling sequence recomputes the problem structure each time a call for calculating the objective function's value (and derivatives) is made. Depending on problems, this may slow down evaluations.

6 A formal description of the calling sequence

The simple calling sequence discussed above is a special case of what is possible. We now give the complete list of arguments for the calls to a problems file. Suppose we consider the problem `problem` (`rosenbr` above). Then the call to `problem` always has the form

```
varargout = problem( action, varargin );
```

where

`action:` is a string whose possible values are

- `'setup'` when the call is to set up the problem,
- `'eldom'` when the lement domains of the problem are requested,
- `'objf'` when the call requests the value of the problem's objective function (and possibly those of its gradient and Hessian) at a given vector of variables,

(4) Such "partioned updating" technique were proposed in [6] and motivated the introduction of partially separable functions.

`'elobjf'` when the call requests the value of a specific element function of the problem (and possibly those of its gradient and Hessian) at a given vector of variables,

`'consf'` when the call requests values of the constraint's function (and possibly their Jacobian and Hessian) at a given vector of variables.

`varargin:` is an optional list of problems parameters.

• For a *setup call*, the complete calling sequence is given by

```
[ x0, fstar, xtype, xlower, xupper, clower, cupper, class ] ...
      = problem( 'setup', varargin )
```

where, on input,

`varargin:` is an optional list of numerical parameters for the problem. When specified, the first element of the list must be a positive integer and is interpreted as the problem's dimension.

For a problem of dimension n, the meaning of the ouput arguments is as follows.

`x0:` is a column vector of dimension n giving the standard problem's starting point ;

`fstar:` is either a numerical value giving the value of the objective function at a known minimizer, or a string giving some information on this value, when possible;

`xtype:` is a string of length n whose i-th character indicates the type of the i-th variable, and can take the values

`'c':` for a continuous variable,

`'i':` for an integer variable,

`'s':` for a categorical variable;

A empty string is equivalent to a string of all `'c'`, indicating that all variables are continuous. (Note that only `'c'` variables are used in the current problem set.)

`xlower:` is a column vector of size n, giving the lower bounds on the variable. Notice that `-Inf` is an acceptable lower bound value. An empty vector is equivalent to a vector whose every component is equal to `-Inf`, indicating that none of the variables has a finite lower bound.

`xupper:` is a column vector of size n, giving the upper bounds on the variable. Notice that `+Inf` is an acceptable upper bound value. An empty vector is equivalent to a vector whose every component is equal to `+Inf`, indicating that none of the variables has a finite upper bound.

`clower:` is a column vector of size equal to the number of constraints, giving the lower bounds on the value of the constraint functions. Notice that `-Inf` is an acceptable lower bound value. An empty vector is equivalent to a vector whose every component is equal to `-Inf`, indicating that all constraints are of the "less than" form.

`cupper:` is a column vector of size equal to the number of constraints, giving the upper bounds on the value of the constraint function. Notice that `+Inf` is an acceptable upper bound value. An empty vector is equivalent to a vector whose every component is equal to `+Inf`, indicating that all constraints are of the "greater than" form.

`class:` is a string indicating the class of the problem, according to the CUTEst classification scheme (see [4]).

Note that, as usual in Matlab, incomplete lists of output arguments are allowed.

• For a *CPS structure call*, the calling sequence reduces to

```
cpsstr = problem( 'cpsstr', n );
```

where, on input,

`n:` is the problem's dimension,

while, on output

`cpsstr:` is a Matlab struct with fields

`name:` a string giving the problem's name,

`eldom:` a cell of length equal to the number n_e of element functions in (4.1) and whose i-th entry is a vector containing the indices of $\mathcal{E}_i$.,

`param:` a cell containing additional problem parameters. For instance, this might contain a vector of measurements for a nonlinear least-squares fitting problem, or, more simply, the problem's dimension. This cell may be empty if no parameter is required for evaluating either the full objective function or its CPS element functions.

• For an *full objective value call* requiring objective-function (and possibly derivatives) values, the calling sequence is

```
[ fx, gx, Hx ] = problem( 'objf', x, varargin );
```

or

```
[ fx, gx, Hx ] = problem( 'objf', x, cpsstr, varargin );
```

where, on input,

`x:` is a column vector of dimension n, specifying the point at which the values must be computed,

`cpsstr:` is the `struct` describing the problem's CPS structure, as given on output of a (previous) CPS structure call,

`varargin:` is an optional list of numerical parameters for the problem,

and, on output,

`fx:` is the returned computed objective-function value at `x`,

`gx:` is a column vector of dimension n containing the returned computed objective-function gradient at `x`,

`Hx:` is a square symmetric matrix of dimension $n \times n$ containing the returned computed objective-function Hessian at `x`.

Note again that, as usual in Matlab, incomplete list of output arguments are allowed. Moreover, an output which is not requested is not calculated. Thus a call

```
fx = problem( 'objf', x, varargin );
```

only computes the objective's value `fx` (the gradient and Hessian are not computed), and, similarly the call

```
[ fx, gx ] = problem( 'objf', x, varargin );
```

does not computes the Hessian. Both calls however recompute the problem's structure, which is not the case for the calls

```
fx = problem( 'objf', x, cpsstr, varargin );
```

and

```
fx = problem( 'objf', x, cpsstr, varargin );
```

• For an *element objective value call* requiring the value (and possibly derivatives values) of a specific element function of the CPS decomposition (4.1), the calling sequence is

```
[ fix, gix, Hix ] = ...
      problem( 'elobjf', i, xi, cpsstr.param{:}, varargin );
```

where, on input,

`i:` is the index (in $\{1, \ldots, n_e\}$) of the considered element function;

`xi:` is a column vector of dimension $|\mathcal{E}_i|$, specifying the values of the variables occuring in the `i`-th element function (as specified in `cpsstr.eldom{i}`) for which the outputs of this element function must be calculated,

`cpsstr:` is the `struct` describing the problem's CPS structure, as given on output of a (previous) CPS structure call (see above),

`varargin:` is an optional list of numerical parameters for the problem,

and, on output,

`fix:` is the returned computed element objective-function value f_i at `xi`,

`gix:` is a column vector of dimension $|\mathcal{E}_i|$ containing the returned computed element objective-function gradient $\nabla^1_x f_i$ at `xi`, with respect to the variables specified in `cpsstr.eldom{i}`,

`Hix:` is a square symmetric matrix of dimension $|\mathcal{E}_i| \times |\mathcal{E}_i|$ containing the returned computed element objective-function Hessian $\nabla^2_x f_i$ at `xi`, with respect to the variables specified in `cpsstr.eldom{i}`.

Note once more that an incomplete list of output arguments is allowed.

7 Some provisions for future extensions

The reader has undoubtly noticed that some possibilities have not been fully described here, notably variable types other than continuous and evaluation of explicit constraint functions. There are included at this stage merely to provide room for future development, should this prove to be useful.

8 Conclusion

We have briefly outlined the OPM optimization test problem collection and provided guidance on how to use it. The authors of course welcome contributions from users in the form of additional (duly verified) test problems[(5)]. Other future developments are contingent on users' demand (and developpers' time).

References

[1] A. S. Bondarenko, D. M. Bortz, and J. J. Moré. COPS: Large-scale nonlinearly constrained optimization problems. Technical Report ANL/MCS-TM-237, Mathematics and Computer Science, Argonne National Laboratory, Argonne, Illinois, USA, 1999.

[2] A. G. Buckley. Test functions for unconstrained minimization. Technical Report CS-3, Computing Science Division, Dalhousie University, Dalhousie, Canada, 1989.

[3] B. Colson and Ph. L. Toint. Optimizing partially separable functions without derivatives. *Optimization Methods and Software*, 20(4-5):493–508, 2005.

[4] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015.

[5] A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 301–312, London, 1982. Academic Press.

[6] A. Griewank and Ph. L. Toint. Partitioned variable metric updates for large structured optimization problems. *Numerische Mathematik*, 39:119–137, 1982.

[7] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, 1981.

[8] M. Porcelli and Ph. L. Toint. Exploiting problem structure in derivative-free optimization. *ACM Transactions on Mathematical Software*, (to appear), 2021.

[9] Ph. L. Toint. Test problems for partially separable optimization and results for the routine PSPMIN. Technical Report 83/4, Department of Mathematics, FUNDP - University of Namur, Namur, Belgium, 1983.

[(5)] If you have one, please contact one of the authors.

Problem descriptions

In the following pages, we give a summary description of each problem, where the meaning of the column's headers are as follows:

`fstar:`	the value of the objective function at a minimizer (if known);
`n:`	the problem's default dimension;
`mel:`	the maximum element-domain size ($\max_i |\mathcal{E}_i|$),
`nc`	the problem's number of continuous variables;
`ni:`	the problem's number of integer variables;
`nfree:`	the problem's number of free variables;
`nlow:`	the problem's number of variables that are bounded below;
`nupp:`	the problem's number of variables that are bounded above;
`nfix:`	the problem's number of variables that are fixed;
`m:`	the problem's number of explicit constraints;
`mi:`	the problem's number of explicit inequality constraints;
`me:`	the problem's number of explicit equality constraints;
`objtype:`	some information on the problem's objective function;
`constype:`	some information on the problem's constraints;
`classif:`	the problem's CUTEst classification string.

```
************************************************************************************************************
************************************************************************************************************
********************************                                          **********************************************
********************************     OPM library content (17-Nov-2021)     **********************************************
********************************                                          **********************************************
************************************************************************************************************
************************************************************************************************************
```

Problem name	fstar	n	mel	nc	ni	nfree	nlow	nupp	nboth	nfix	m	mi	me	objtype	constype	classif
argauss	+1.12793277e-08	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-3-0
arglina	+1.00000000e+01	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
arglinb	+4.87179487e+00	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
arglinc	+6.13513514e+00	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
argtrig	+0.00000000e+00	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
arwhead	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-V-0
bard	+8.21500000e-03	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-3-0
bdarwhd	+0.00000000e+00	10	3	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
beale	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
biggs5	+0.00000000e+00	6	6	6	0	5	0	0	0	1	0	0	0	nonlin	-	SXR2-AN-6-0
biggs6	+0.00000000e+00	6	6	6	0	6	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-6-0
booth	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	QUR2-AN-2-0
box2	+0.00000000e+00	3	3	3	0	2	0	0	0	1	0	0	0	nonlin	-	OXR2-AN-2-0
box3	+0.00000000e+00	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-3-0
bratu1d	unknown	30	2	30	0	28	0	0	0	2	0	0	0	nonlin	-	SUR2-AY-V-0
brkmcc	-Inf	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-2-0
brownal	+0.00000000e+00	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
brownbs	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
brownden	+8.58222000e+04	4	4	4	0	4	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-4-0
broyden3d	+0.00000000e+00	10	3	10	0	8	0	0	0	2	0	0	0	nonlin	-	SXR2-AY-V-0
broydenbd	+0.00000000e+00	10	7	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
chandheu	+0.00000000e+00	57	57	57	0	57	0	0	0	0	0	0	0	nonlin	-	SUR2-RN-V-0
chebyqad	+0.00000000e+00	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
cliff	+1.99786610e-01	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-2-0
clplatea	unknown	16	2	16	0	12	0	0	0	4	0	0	0	nonlin	-	OXR2-MN-V-0
clplateb	unknown	16	4	16	0	12	0	0	0	4	0	0	0	nonlin	-	OXR2-MN-V-0
clustr	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
cosine	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	QUR2-AY-V-0
crglvy	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
cube	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0

OPM library content (17-Nov-2021)

Problem name	fstar	n	mel	nc	ni	nfree	nlow	nupp	nboth	nfix	m	mi	me	objtype	constype	classif
curly10	unknown	30	11	30	0	30	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-V-0
curly20	unknown	30	21	30	0	30	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-V-0
curly30	unknown	40	31	40	0	40	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-V-0
deconvu	-Inf	51	22	51	0	51	0	0	0	0	0	0	0	nonlin	-	SUR2-MN-51-0
dixmaana	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaanb	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaanc	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaand	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaane	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaanf	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaang	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaanh	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaani	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaanj	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaank	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixmaanl	+1.00000000e+00	12	4	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
dixon	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	QUR2-AY-V-0
dqrtic	+0.00000000e+00	10	1	10	0	10	0	0	0	0	0	0	0	nonlin	-	QUR2-AY-V-0
edensch	unknown	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
eg2	unknown	10	1	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
eg2s	unknown	10	3	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
eigenals	+0.00000000e+00	110	30	110	0	110	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
eigenbls	+0.00000000e+00	110	30	110	0	110	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
eigencls	+0.00000000e+00	110	30	110	0	110	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
engval1	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
engval2	+0.00000000e+00	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
expfit	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
extrosnb	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
fminsurf	+1.00000000e+00	16	16	16	0	16	0	0	0	0	0	0	0	nonlin	-	OUR2-MY-V-0

OPM library content (17-Nov-2021)

Problem name	fstar	n	mel	nc	ni	nfree	nlow	nupp	nboth	nfix	m	mi	me	objtype	constype	classif
freuroth	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
genhumps	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
gottfr	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
gulf	+0.00000000e+00	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-3-0
hairy	+2.00000000e+01	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-2-0
heart6ls	+2.00000000e+00	6	6	6	0	6	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-6-0
heart8ls	+0.00000000e+00	8	8	8	0	8	0	0	0	0	0	0	0	nonlin	-	SUR2-MN-8-0
helix	+0.00000000e+00	10	3	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
hilbert	+0.00000000e+00	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	QUR2-AN-V-0
himln3	-1.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-2-0
himm25	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
himm27	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-2-0
himm28	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
himm29	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
himm30	+0.00000000e+00	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-3-0
himm32	+3.18572000e+02	4	4	4	0	4	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-4-0
himm33	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-2-0
hydc20ls	+0.00000000e+00	99	14	99	0	99	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-99-0
hypcir	+1.23259516e-01	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
indef	unknown	10	3	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
integreq	+0.00000000e+00	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
jensmp	+1.24362000e+02	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
kowosb	+3.07505000e-03	4	4	4	0	4	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-4-0
lminsurf	+9.00000000e+00	16	4	16	0	4	0	0	0	12	0	0	0	nonlin	-	OXR2-MY-V-0
mancino	unknown	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
mexhat	-1.11715260e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
meyer3	+8.79458000e+01	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	SUR2-RN-3-0
morebv	+0.00000000e+00	12	3	12	0	10	0	0	0	2	0	0	0	nonlin	-	SXR2-AY-V-0
msqrtals	+0.00000000e+00	16	7	16	0	16	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0

```
*****************************************************************************************
*****************************************************************************************
*******************************                                     *********************
*******************************    OPM library content (17-Nov-2021)    *****************
*******************************                                     *********************
*****************************************************************************************
*****************************************************************************************
```

Problem name	fstar	n	mel	nc	ni	nfree	nlow	nupp	nboth	nfix	m	mi	me	objtype	constype	classif
msqrtbls	+0.00000000e+00	16	7	16	0	16	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
ncb20	unknown	30	30	30	0	30	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-V-0
ncb20b	unknown	21	20	21	0	21	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-V-0
ncb20c	unknown	30	30	30	0	30	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-V-0
nlminsurf	unknown	16	4	16	0	4	0	0	0	12	0	0	0	nonlin	-	OXR2-MY-V-0
nondia	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
nondquar	+0.00000000e+00	100	3	100	0	100	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-V-0
nzf1	+0.00000000e+00	13	6	13	0	13	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
osbornea	+5.46489000e-05	5	5	5	0	5	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-5-0
osborneb	+4.01377000e-02	11	11	11	0	11	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-11-0
penalty1	unknown	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
penalty2	unknown	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
penalty3	unknown	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
powellbs	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-2-0
powellsg	+0.00000000e+00	4	2	4	0	4	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
powellsq	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-2-0
powr	+0.00000000e+00	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
recipe	+0.00000000e+00	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-3-0
rosenbr	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
s308	+7.73199000e-01	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-2-0
schmvett	-3.00000000e+00	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-3-0
scosine	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	QUR2-AY-V-0
scurly10	unknown	30	11	30	0	30	0	0	0	0	0	0	0	bad scal.	-	OUR2-AN-V-0
scurly20	unknown	30	21	30	0	30	0	0	0	0	0	0	0	bad scal.	-	OUR2-AN-V-0
scurly30	unknown	30	30	30	0	30	0	0	0	0	0	0	0	bad scal.	-	OUR2-AN-V-0
sensors	unknown	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-V-0
shydc20ls	+0.00000000e+00	99	14	99	0	99	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-99-0
sisser	+0.00000000e+00	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	OUR2-AN-2-0
spmsqrt	+0.00000000e+00	10	5	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0

```
**************************************************************************************************************************
**************************************************************************************************************************
********************************                                           ***********************************************
********************************     OPM library content (17-Nov-2021)     ***********************************************
********************************                                           ***********************************************
**************************************************************************************************************************
**************************************************************************************************************************
```

Problem name	fstar	n	mel	nc	ni	nfree	nlow	nupp	nboth	nfix	m	mi	me	objtype	constype	classif
tcontact	+2.48015700e+00	100	4	100	0	48	0	0	0	52	0	0	0	nonlin	-	OXR2-MY-V-0
tlminsurf	+9.00000000e+00	16	4	16	0	4	0	0	0	12	0	0	0	nonlin	-	OXR2-MY-V-0
tlminsurfx	+9.00000000e+00	16	4	16	0	4	0	0	0	12	0	0	0	nonlin	-	OXR2-MY-V-0
tnlminsurf	unknown	16	4	16	0	4	0	0	0	12	0	0	0	nonlin	-	OXR2-MY-V-0
tnlminsurfx	unknown	16	4	16	0	4	0	0	0	12	0	0	0	nonlin	-	OXR2-MY-V-0
tquartic	+0.00000000e+00	10	1	10	0	10	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
tridia	+0.00000000e+00	10	2	10	0	10	0	0	0	0	0	0	0	nonlin	-	QUR2-AY-V-0
trigger	+0.00000000e+00	7	5	7	0	6	0	0	0	1	0	0	0	nonlin	-	SXR2-AN-7-0
vardim	+0.00000000e+00	10	10	10	0	10	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
vibrbeam	unknown	8	8	8	0	8	0	0	0	0	0	0	0	nonlin	-	SUR2-MN-8-0
watson	unknown	12	12	12	0	12	0	0	0	0	0	0	0	nonlin	-	SUR2-AN-V-0
wmsqrtals	unknown	16	7	16	0	16	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
wmsqrtbls	unknown	16	7	16	0	16	0	0	0	0	0	0	0	nonlin	-	SUR2-AY-V-0
woods	+0.00000000e+00	12	2	12	0	12	0	0	0	0	0	0	0	nonlin	-	OUR2-AY-V-0
yfitu	+0.00000000e+00	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	SUR2-MN-3-0
zangwil2	-1.82000000e+01	2	2	2	0	2	0	0	0	0	0	0	0	nonlin	-	QUR2-AN-2-0
zangwil3	+0.00000000e+00	3	3	3	0	3	0	0	0	0	0	0	0	nonlin	-	QUR2-AN-3-0

```
**************************************************************************************************************************
```