# miqp.m:
# A Matlab function for solving Mixed Integer Quadratic Programs
# Version 1.06
# User Guide

Alberto Bemporad, Domenico Mignone
Institut für Automatik, ETH - Swiss Federal Institute of Technology
ETHZ - ETL, CH 8092 Zürich, Switzerland
tel.+41-1-632 6679 fax +41-1-632 1211
{bemporad,mignone}@aut.ee.ethz.ch
http://control.ethz.ch

May 9, 2001

**Abstract**

This manual describes `miqp.m`, a Matlab function for solving mixed integer quadratic programs and mixed integer linear programs. The solver is implemented using a branch and bound technique and allows the user to specify various options, like tree exploring strategies, branching variable selection rules, and many more.

## 1 Purpose

The matlab function `miqp.m` solves the following mixed integer quadratic program (MIQP):

$$\begin{aligned} \min_x \quad & 0.5x^T Hx + f^T x \\ \text{subject to} \quad & Ax \le b \\ & A_{eq}x = b_{eq} \\ & v_{lb} \le x \le v_{ub} \\ & x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_d} \\ & x(i_{vartype}) \in \{0,1\}^{n_d} \end{aligned}$$

The length of the optimization vector $x$ is $n = n_c + n_d$. The variables indexed by $i_{vartype}$, which is a subset of $\{1, \ldots, n_c + n_d\}$, are constrained to be binary. The matrix $H \in \mathbb{R}^{n \times n}$ is positive semidefinite. The special case where $H = 0$ is called mixed integer linear program (MILP) and it can also be handled by `miqp.m`. The matrix $A \in \mathbb{R}^{m \times n}$ and the vector $b \in \mathbb{R}^m$ define linear inequality constraints on the optimization variables. Linear equality constraints are given by $A_{eq} \in \mathbb{R}^{m' \times n}$ and $b_{eq} \in \mathbb{R}^{m'}$ Bounds on $x$ can be specified by the vectors $v_{lb} \in \mathbb{R}^n, v_{ub} \in \mathbb{R}^n$.

## 2 What's new?

Compared to version 1.02, this distribution features the following improvements:

- Solver option `qp_dantz.m` added

- Default initialization of cost function has been set to $\infty$
- Options with whom linprog and quadprog are called can be set with the field optimset
- Routine qphess.m is not required anymore and is generated on the fly, if necessary.
- Rounding of integer variables implemented

# 3 Contents of the Package

The package is available from the Automatic Control Laboratory of ETH Zürich, Switzerland, or directly from the URL:
`http://control.ethz.ch/~hybrid/miqp/`

The following files are distributed:

`miqp.pdf:` This user guide

`miqp.m:` The matlab function

`test.m:` A simple test problem reported in Section 7

# 4 Requirements

## 4.1 Matlab Version

The function `miqp.m` runs with Matlab Version 5.2 or higher

## 4.2 The QP and LP Solvers

The function `miqp.m` requires the availability of solvers for linear programs (LP) and quadratic programs (QP). The solvers listed in Table 1 are currently sopported by `miqp.m`.

| QP solvers | | input parameter |
|---|---|---|
| quadprog.m | QP solver from Matlab's optimization toolbox | 'quadprog' |
| qp.m | Old version of quadprog.m | 'qp' |
| e04naf.m | QP solver from NAG foundation toolbox | 'qpnag' |
| qp_dantz.m | QP solver from MPC toolbox | 'qp_dantz' |
| LP solvers | | input parameter |
| linprog.m | LP solver from matlab's optimization toolbox | 'linprog' |
| lp.m | Old version of linprog.m | 'lp' |
| e04mbf.m | LP solver from NAG foundation toolbox | 'lpnag' |

Table 1: QP and LP solvers supported by `miqp.m`. The column "input parameter" denotes the code to be specified for choosing the corresponding solver, see the usage of `Options` in Section 5.3

If one or more solvers for LP and QP are not available on the user's platform, it is recommended to force the use of the available solvers by setting the parameter `Options` appropriately, as explained in Section 5.3.

# 5 Input Parameters

The header of `miqp.m` is:

```
[xmin,fmin,flag,Extendedflag]=miqp(H,f,A,b,Aeq,beq,vartype,lb,ub,x0,Options)
```

### 5.1 Mandatory Arguments

The input arguments listed in Table 2 are mandatory when calling `miqp.m`:

| | |
|---|---|
| `H, f` | parameters of the cost function |
| `A, b` | parameters defining the inequality constraints |

Table 2: Mandatory input arguments of `miqp.m`

### 5.2 Optional Arguments

The input arguments listed in Table 3 are optional when calling `miqp.m`. If the input arguments are not specified, have values lying out of the allowed range, or they are passed as empty arguments ([ ]), the default values are assumed.

| argument | meaning | default |
|---|---|---|
| `Aeq` | Equality constraints | [ ] |
| `beq` | Equality constraints | [ ] |
| `vartype` | Vector defining binary variables | [ ] |
| `lb` | Lower bounds on x | $-\infty$ |
| `ub` | Upper bounds on x | $+\infty$ |
| `x0` | Initial condition | 0 |
| `Options` | further options | see Section 5.3 |

Table 3: Optional input arguments of `miqp.m`

### 5.3 The Optional Argument `Options`

The input argument `Options` is used to define several features of the branch and bound procedure. `Options` is a structure in Matlab format. To specify an option, the corresponding field must be defined and its value must be set to an admissible value listed in Tables 4 and 5.

**Example:** The field `solver` is used to specify the QP/LP solver, to be used in `miqp.m`. The list of supported solvers is given in Table 4. To set the solver to `quadprog.m`, the syntax is:

```
options        = [];
options.solver = 'quadprog';

[xmin,fmin,flag,Eflag] = miqp(H,f,A,b,Aeq,beq,vartype,lb,ub,x0,Options)
```

Any fields other than those listed in Tables 4 and 5 are ignored. If `options.field` is a valid field, but its value lies outside the domain specified for each field, the default value is taken, and a warning message is produced. Default values are also assumed, if `Options` is not defined, or if `Options` does not have the corresponding fields.

Some fields of the structure `Options` are explained next:

**`method`:** Specifies the tree exploring strategy. The values `depth` and `breadth` specify standard depth first and breadth first strategies. The value `best` specifies the best first strategy: the binary tree is explored, such that those problems are

| field in `Options` | meaning of field | possible values | meaning of values |
|---|---|---|---|
| `solver` | solver to be used | 'lp'<br>'linprog'<br>'lpnag'<br>'qp'<br>'quadprog' (def.)<br>'qpnag' | see Table 1 |
| `method` | tree exploring strategy | 'depth' (def.)<br>'breadth'<br>'best'<br>'bestdepth' | depth first search<br>breadth first search<br>best first search<br>normalized best first search |
| `branchrule` | branching rule, node selection | 'first'(def.)<br>'max'<br>'min' | first free variable<br>maximum fractional part<br>minimum fractional part |
| `order` | prioritized problem | 0 (def.)<br>1 | last binary var. set to 0<br>last binary var. set to 1 |
| `verbose` | amount of messages | 0 (def.)<br>1<br>2 | quiet<br>medium number of messages<br>high number of messages |

Table 4: Optional input arguments of `miqp.m`, part 1, "def." denotes the default value. Note that all values in quotation marks '.' must be entered as strings in Matlab format.

| field in `Options` | meaning of field | possible values |
|---|---|---|
| `maxqp` | Maximum number of LPs or QPs allowed to be solved. | positive integer (def. $= \infty$) |
| `inftol` | Large number to be considered as infinity in constraints. This is only used with the solvers from the NAG toolbox. | positive real (def. $= 10^8$) |
| `matrixtol` | Tolerance for recognizing that the MIQP is an MILP, by testing, whether max(svd(H)) $\leq$ matrixtol. | nonnegative real (def. $= 10^{-6}$) |
| `postol` | Tolerance for recognizing H $> 0$, by testing on the relaxed QP, whether cond(H) $\leq$ postol. | nonnegative real (def. $= \emptyset$) |
| `integtol` | Tolerance to recognize integers | nonnegative real (def. $= 10^{-4}$ |
| `maxQPiter` | Maximum number of iterations within each QP or LP. | positive integer (def. $= 1000$) |

Table 5: Optional input arguments of `miqp.m`, part 2, "def." denotes the default value

solved first, that have the lowest cost in the father problem. With the option `bestdepth` the cost of the father problem is normalized with the depth of the node in the tree.

`branchrule`: Specifies the node section strategy. If `branchrule`= `first`, the first free variable among the relaxed binary variables is chosen. If `branchrule`= `max` or `min`, the relaxed variable is chosen, where the fractional part is nearer or further away from the next binary variable.

`order`: During the branch and bound procedure the subproblems are separated by setting the branching variable to zero for one relaxed problem and to one for the other. The problems are then pushed onto the stack. The parameter `order` specifies which problem is put onto the stack as *second*, i.e. is solved first. If `order` = 0, the problem where the branching variable has been set to zero is solved first.

`maxqp`: The computations are stopped, as soon as the number of relaxed QPs or LPs has reached `maxqp`. The currently best feasible solution is returned as the optimum and the optimizer. If no feasible solution has been found up to the instant, when maxQPiter programs are solved, the problem is reported to be infeasible. Reaching this limit is also reported in the output parameter `flag` mentioned in Section 6. This option can be used to get suboptimal solutions within a short computation time.

`inftol`: Using a solver from the NAG foundation toolbox, it is possible to specify a bound `inftol`. All numbers exceeding this value in magnitude are then considered as infinity.

`matrixtol`: Tolerance for recognizing that the MIQP is actually an MILP. This option is only used, if `options.solver` is undefined. In this case a check on the matrix $H$ is made and the solver is automatically chosen as 'linprog' if the maximum singular value of $H$ is less than `matrixtol`. Otherwise the default QP solver is chosen. If `options.solver` is defined, `matrixtol` is ignored.

`postol`: Tolerance for recognizing, whether $H$ is positive definite or only positive semidefinite. If any relaxed QP has

$$cond(H) \leq \texttt{options.postol}$$

and `options.verbose` $\geq 1$ then a warning message is produced. To avoid the computation of the condition number for each relaxed QP, leave this field undefined. As a default, this parameter is left undefined, i.e. no check is performed. If the parameter is defined, but its value is not a nonnegative real, the value is set to $10^{-6}$.

`integtol`: A variable is recognized as 0 or 1, if it lies closer than this threshold to either 0 or 1. This parameter has a significant influence on the branch and bound procedure. It should not be chosen smaller, than the expected absolute precision of the QP or LP solver.

`maxQPiter`: The QP and LP solvers supported by `miqp.m` allow to specify the maximum number of steps that are allowed to be done within the QP and LP solver. This value can be be specified with `options.maxQPiter`.

The parameters options.optimset, options.deletefile and options.round are explained in the header of miqp.m

# 6 Output Parameters

The output parameters have the following interpretations:

`xmin`: Minimizer of the MIQP

`fmin`: Minimum value of the cost function

`flag`: characterization of solution according to the following list

- **1:** optimum found
- **5:** feasible, but not integer feasible
- **7:** infeasible, since relaxed problem is infeasible
- **11:** integer feasible, however the limit `maxqp` of QPs has been reached, i.e. the solution might be suboptimal
- **15:** feasible, but not integer feasible, however the limit maxqp of QPs has been reached, i.e. the search might not have lasted long enough
- **-1:** the solution is unbounded

`Extendedflag` Structure in with following fields

- `Qpiter`: total number of QPs (LPs) solved
- `time`: time elapsed for running the function
- `optQP`: number of QP (LP) at which the optimum was found

# 7 Function Call

A typical example of usage of `miqp.m` is given next.

```
Q = zeros(4,4);     % MILP
b = [ 2, -3, -2, -3]';
C = [-1  -1  -1  -1;
    10 5   3 4;
    -1 0   0 0];
d = [-2  10   0]';
vlb  = [-1e10 0 0 0];
vub  = [ 1e10 1 1 1];
ivar = [ 2 3 4];
x0 = zeros(size(Q,1),1);

options = [];
options.integtol = 1e-6;
options.solver   = 'qpnag';

[xsol,fsol,flag,Eflag]=miqp(Q,b,C,d,[],[],ivar,vlb,vub,x0,options)
```

This produces the solution

```
xsol = [0 1 0 1];
fsol = -6;
```

# 8 Conditions of Use



# 9 Notes

- The bounds on the optimization variable `lb` and `ub` are set automatically to 0 and 1 for the binary variables, i.e. the specification

$$v_{lb}(i_{vartype}) = [0, \ldots, 0]$$
$$v_{ub}(i_{vartype}) = [1, \ldots, 1]$$

  are added by `miqp.m`, if they sould be missing.
- If, for some reason, a problem has to be solved, where one or more binary variables are set to 0 or 1 a priori, this can be specified by setting the corresponding entries of `lb` and `ub` both to 0 or to 1 respectively.
- For some solvers a number of messages about infeasibility of the problem are reported on the screen. These messages are due to the infeasibility of the relaxed problems during branch and bound and are normally not referred to the infeasibility of the overall problem. For the feasibility of the problem, please check the output parameter `flag`.
- In case of infeasibilities when using 'linprog' or 'quadprog', it is recommended to set

  ```
  options.optimset = [];
  ```

  This resets the input parameters of the solvers to their default values.

# 10 Contacts

We are glad to hear any kind of feedback, suggestions and correction about `miqp.m`. Please contact us at the address mentioned on the first page of this document.

# 11 Acknowledgements

We would like to thank Fabio Danilo Torrisi for the extremely useful comments and suggestions during several stages of the development of the solver. We would also thank Giancarlo Ferrari-Trecate, Francesco Borrelli, Dario Castagnoli and Eric Kerrigan for having tested the solver.