Quadratic Programming with MATLAB and quadprog

This guide assumes that you have already installed the Optimization Toolbox for your version of MATLAB. You can check if it is installed, and which version you have, by issuing the `ver` command in the MATLAB command window. In particular, please note the release version of the Optimization Toolbox you have (given in parentheses, e.g., 'R2011a').[1]

We will use the `quadprog` function provided by the optimization toolbox. Let us first review the standard form of a QP (following `quadprog` notation):

$$\begin{aligned} \min_x \quad & \tfrac{1}{2} x^\top H x + f^\top x \\ \text{subject to} \quad & A x \preceq b \\ & A_{\text{eq}} x = b_{\text{eq}} \\ & l \preceq x \preceq u \end{aligned}$$

Note that $x^\top$ denotes the transpose of x, and $Ax \preceq b$ means that the inequality is taken element-wise over the vectors Ax and b. The above objective function is convex if and only if H is positive-semidefinite, which is the realm we are concerned with.[2]

The `quadprog` function expects a problem of the above form, defined by the parameters $\{H, f, A, b, A_{\text{eq}}, b_{\text{eq}}, l, u\}$; H and f are required, the others are optional (empty matrix `[]`). Alternate QP formulations must be manipulated to conform to the above form; for example, if the inequality constraint was expressed as $Ax \succeq b$, then it can be rewritten $-Ax \preceq -b$. Note that x itself is not provided to the solver, since it is an internal variable being optimized over. In particular, this means that the solver has no explicit knowledge of x itself; everything is implicity defined by the supplied parameters. It is essential that the same variable order is maintained for the relevant parameters (e.g., f_i, b_i, l_i should correspond to variable x_i).

[1] There will be a slight difference later depending on whether your version is 2011 or not. We recommend that you acquire the latest version of MATLAB; in particular, we recommend at least a 2010 version.

[2] Non-convexity implies the existence of local optima, making it difficult to find global optima. `quadprog` requires that H be positive definite, for which the problem is strictly convex. To solve for non-convex QPs, see the `fmincon` function: `http://www.mathworks.com/help/toolbox/optim/ug/fmincon.html`

Let us consider a simple example:

$$\begin{aligned} \min_{x,y} \quad & \tfrac{1}{2}x^2 + 3x + 4y \\ \text{subject to} \quad & x, y \geq 0 \\ & x + 3y \geq 15 \\ & 2x + 5y \leq 100 \\ & 3x + 4y \leq 80 \end{aligned}$$

First, we rewrite the above in the given standard form:

$$\begin{aligned} \min_{x,y} \quad & \tfrac{1}{2} \begin{bmatrix} x \\ y \end{bmatrix}^\top \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \end{bmatrix}^\top \begin{bmatrix} x \\ y \end{bmatrix} \\ & \begin{bmatrix} -1 & -3 \\ 2 & 5 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \preceq \begin{bmatrix} -15 \\ 100 \\ 80 \end{bmatrix} \\ & \begin{bmatrix} 0 \\ 0 \end{bmatrix} \preceq x \end{aligned}$$

By inspection, the variable is defined by $\begin{bmatrix} x \\ y \end{bmatrix}$, and the parameters H, f, A, b, l are given. Note how we reversed some inequalities to achieve the standard form. Since there are no equality constraints and upper bounds, we do not need to provide the empty $A_{\text{eq}}, b_{\text{eq}}, u$. Note that even though y^2 did not appear in the original objective, we had to include it with zero coefficients in H because the solver parameters must be defined using the full set of variables. Even if certain variables only appear in constraints, they will still need to be expressed with zero coefficients in the objective parameters, and *vice versa*.

Let us first define the above parameters in MATLAB:

```
H = diag([1; 0]);
f = [3; 4];
A = [-1 -3; 2 5; 3 4];
b = [-15; 100; 80];
l = zeros(2,1);
```

We also need to set what MATLAB solver to use with the `Algorithm` field in the optimization options. A generally recommend choice is to use interior point methods, which is usually superior to the default choice. If your MATLAB version is R2011a or later, do:

```
options = optimset('Algorithm','interior-point-convex');
```

Otherwise, for pre-2011 versions of MATLAB, do:

```
options = optimset('Algorithm','interior-point','LargeScale','off','MaxIter',1000);
```

The hard work is mostly over now! As you will often find, formulating the problem is usually the hard step. Invoking a solver is straightforward:

```
[x,fval] = quadprog(H,f,A,b,[],[],l,[],[],options);
```

That's it! The full list of arguments is:

```
[x,fval] = quadprog(H,f,A,b,Aeq,Beq,l,u,x0,options);
```

but since we don't have equality constraints, upper bounds, and initial values in our problem, we gave the empty matrix `[]`. For other ways to call `quadprog`, get more output, and set other options, see the MATLAB documentation in the references on the final page.

The optimal solution and value are now in `x` and `fval` respectively:

```
disp(x);       % 0
disp(fval);    % 5
```

We can verify[3] that $x^* = 0, y^* = 5$, with an optimal value 20.

[3]It is crucial to verify the solution! Don't just trust what the solver gives back to you!

The code is reproduced below for your convenience:

```
% Define QP parameters
H = diag([1; 0]);
f = [3; 4];
A = [-1 -3; 2 5; 3 4];
b = [-15; 100; 80];
l = zeros(2,1);

% Set quadprog options
% R2011a or later
options = optimset('Algorithm','interior-point-convex');
% Before R2011a
options = optimset('Algorithm','interior-point','LargeScale','off','MaxIter',1000);

% Construct the QP, invoke solver
[x,fval] = quadprog(H,f,A,b,[],[],l,[],[],options);

% Extract optimal value and solution
disp(x);       % 0
disp(fval);    % 5
```

References:

`http://www.mathworks.com/help/toolbox/optim/ug/quadprog.html`
MATLAB documentation on the R2011b version of `quadprog`. The MATLAB documentation is generally very complete, almost to the point that it is overwhelming, hence the current summarized document. Other input/output forms and options for `quadprog` are explained.

`http://www.mathworks.com/help/toolbox/optim/ug/brhkghv-1.html`
The Optimization Toolbox user's guide, in particular the section on 'Setting Up an Optimization' linked above, is helpful for choosing which functions and algorithms to use.