DuQuad: User Manual

Quadratic Programming Optimization

Contents

1 DuQuad: User Manual **1**
1.1 Introduction 1
1.1.1 Algorithms 1
1.1.2 Download and Installation 2
1.2 Short Tutorial 2
1.3 Specifications 4
1.3.1 Inputs 4
1.3.2 Outputs 5

1. DuQuad: User Manual

1.1. Introduction

The *DuQuad* optimization toolbox solves convex quadratic programs using dual first order optimization algorithms. The algorithms has predictable and fast convergence, low memory footprint, and uses only basic arithmetic and logical operations. DuQuad and is therefore suited to be utilized by real-time applications running on low-cost HW such as simple microcontrollers. Furthermore, DuQuad has an user friendly Matlab interface for maximum productivity, and the algorithms are implemented in efficient C-code.

The algorithms attempts to solve the quadratic programming problem:

$$\boxed{\begin{aligned} \min_{z} \quad & f(z) = \tfrac{1}{2} z^T H z + c^T z \\ \text{s.t.} \quad & \hat{lb} \leq Gz - g \leq \hat{ub} \\ & lb \leq z \leq ub, \end{aligned} \qquad (1.1)}$$

where $H \in \mathbb{R}^{n \times n}$ is the Hessian, $G \in \mathbb{R}^{m \times n}$ is a matrix for the linear constraints, and $c, lb, ub \in \mathbb{R}^n$ and $g, \hat{lb}, \hat{ub} \in \mathbb{R}^m$ are column vectors.

1.1.1. Algorithms

DuQuad contains the four different algorithms:

- Dual Gradient Method (DGM)
- Dual Fast Gradient Method (DFGM)
- Dual Augmented Lagrangian Method (ALM)
- Dual Fast Augmented Lagrangian Method (FALM)

Note that ALM and FALM can only solve problems with equality constraints, i.e. the case where $\hat{lb} = \hat{ub}$.

1.1.2. Download and Installation

Information, documentation, and code downloads can be found by following the link below.

- DuQuad webpage: `http://sverrkva.github.io/duquad/`
- Direct download: `https://github.com/sverrkva/duquad`
- Documentation of c-code: `http://sverrkva.github.io/duquad_doc_ccode/`

The c-code needs to be compiled into a mex-file. A makefile (make.m) is included in the code download. If running a linux distribution it should be adequate to run the make.m to compile the program. Furthermore, an example-file is included to get a quick start.

The download also includes a Matlab version of DuQuad where all the algorithms are implemented in Matlab. The Matlab version has the same behaviour and almost identically inputs and outputs as the main version.

1.2. Short Tutorial

Formulate the Problem

Formulate an optimization problem on the form of equation (1.1). For example:

```
H = [11 4 ; 4 22];  % Hessian matrix
c = [3 ; 4];        % gradient vector
G = [1 1;2 1];      % linear constraints matrix
g = [2 ; 3];        % linear constraints vector
lb_hat = [-2 ; -2]; % lower bound for the linear constraints
ub_hat = [2 ; 2];   % upper bound for the linear constraints
lb = [-1 ; -2];     % lower bound for optimization variable z
ub = [0.5 ; 2];     % upper bound for optimization variable z
z0 = [0.5 ; -0.5];  % initial point
```

Run the Program

To solve the problem, call the duquad function with the problem as input:

```
[zopt,fopt] = duquad(H,c,G,g,lb_hat,ub_hat,lb,ub,z0);
```

If the linear constraints is have lower bound, and the optimization variable has no upper upper bound then the the function will be called as follows:

```
[zopt,fopt] = duquad(H,c,G,g,[],ub_hat,lb,[],z0);
```

If the optimization variable is unbounded, the linear constraints are $G \leq g$, and there is no initial point, the function can be called as:

```
[zopt,fopt] = duquad(H,c,G,g);
```

In conclusion, DuQuad is flexible towards the inputs. Furthermore, grab all possible outputs by calling the function as:

```
[zopt,fopt,exitflag,output,lambda1,lambda2] = duquad(H,c,G,g);
```

An overview of all inputs and outputs is viewed in Matlab console if the user is running the Matlab command:

```
help duquad
```

Include Options

DuQuad can also take different options as input, e.g. maximum number of iterations, tolerance for stopping criteria etc. All these options are collected in a struct (Table 1.2) as follows:

```
% Maximum number of iterations in the outer loop
options.maxiter_outer = 1000;
% Maximum number of iterations in the inner loop
options.maxiter_inner = 100;
% Tolerance for dual suboptimality
options.eps_ds = 0.0001;
% Tolerance for primal feasibility
options.eps_pf = 0.001;
% Tolerance for primal feasibility in the inner problem
options.eps_inner = 0.0001;
% Penalty parameter used in ALM and FALM
options.rho = 1;
% Specifies the algorithm used to solve the problem.
options.algorithm = 1;
```

The option struct is included as input number 10 in the function:

```
[zopt,fopt] = duquad(H,c,G,g,[],[],[],[],[],options);
```

Note that the user must either specify all options or no options (when default values are utilized). This should be improved in a newer version of DuQuad.

1.3. Specifications

In this section, the different inputs and outputs are listed. DuQuad's full potential, regarding inputs and outputs, is utilized by running the following example Matlab command:

```
[zopt,fopt,exitflag,output,lambda1,lambda2]...
    = duquad(H,c,G,g,lb_hat,ub_hat,lb,ub,z0,options);
```

1.3.1. Inputs

Table 1.1 gives an overview of the different inputs to the function. In addition the dimensions for each input is listed. The last input to the function is a struct called *options*, which set some criteria for the solving process. The different options are summarized in Table 1.2. One of the options is to choose which algorithm that is solving the problem. This is specified by a number ranging from 1-8, and is listed in Table 1.3.

Table 1.1: The inputs for the duquad function

Input	**Name**	**Desciption**	**Dimension**
1	H	Hessian matrix	$n \times n$
2	c	Gradient vector	n
3	G	Linear constraints matrix	$m \times n$
4	g	Linear constraints vector	m
5	$\hat{lb}$	Lower bound for the linear constraints	m
6	$\hat{ub}$	Upper bound for the linear constraints	m
7	lb	Lower bound for optimization variable z	n
8	ub	Upper bound for optimization variable z	n
9	z_0	Initial point	n
10	options	Struct containing options for solver, see Table 1.2	

Table 1.2: Overview of the parameters in the *options* struct

Name	**Description**	**Default**
maxiter_outer	Maximum number of iterations in the outer loop	1000
maxiter_inner	Maximum number of iterations in the inner loop	100
eps_ds	Tolerance for dual suboptimality	0.0001
eps_pf	Tolerance for primal feasibility	0.001
eps_inner	Tolerance for primal feasibility in the inner problem	0.00001
rho	Penalty parameter used in ALM and FALM	1
algorithm	Specifies the algorithm used to solve the problem.	3

Table 1.3: Values of the *algorithm* parameter from the option struct in Table 1.2

Algorithm	**Value**
DGM last	1
DGM average	2
FDGM last	3
FDGM average	4
ALM last	5
ALM average	6
FALM last	7
FALM average	8

1.3.2. Outputs

The outputs of DuQuad is summarized in Table 1.4. Among the outputs is a struct called *output*. This struct contains some results from the solving process.

Table 1.4: The outputs of the duquad function

Output	**Name**	**Description**
1	z^*	Optimal point
2	f^*	Optimal value
3	exitflag	1 = solution found, 2 = max num of iteration reached, -1 = error
4	output	Struct containing various result, see Table 1.5
5	λ_1	Set of Lagrangian multipliers
6	λ_2	Set of Lagrangian multipliers

Table 1.5: Content of the output struct **output**

Name	**Description**
iterations	Number of outer iterations
iterations_inner_tot	Total number of iterations for the inner problem
time	Runtime of the algorithm after all initialization is done
time_tot_inner	Total time spent on solving the inner problem
flag_last_satisfied	Flag specifies which stopping criteria was resolved last. Value: 0 = dual suboptimality, 1 = primal feasibility
niter_feasible_ds	Number of iterations the criterion for dual suboptimality was satisfied
niter_feasible_pf	Number of iterations that the criterion for primal feasibility was satisfied
exitflag_inner	Exitflag for the inner problem. Values: 1 = feasible point found, 2 = Maximum number of iterations exceeded
num_exceeded_max_niter_inner	Total number of times the inner problem exceeded the number of iterations
ds_vector	Vector storing all the value of the dual suboptimality every iteration
pf_vector	Vector storing all the value of the primal feasibility every iteration
algorithm	Name of the algorithm used to solve the problem