On the feasibility for the system of quadratic equations MATLAB Library

Anatoly Dymarsky, Elena Gryazina, Boris Polyak, Sergei Volodin

1 Notations

The goal of the project is to solve a number of tasks for quadratic maps, which are

1. (Real case) The map $f\colon \mathbb{R}^n \to \mathbb{R}^m$ s.t.

$$f_i(x) = x^T A_i x + 2b_i^T x,\ A_i = A_i^T$$

2. (Complex case) The map $f\colon \mathbb{C}^n \to \mathbb{R}^m$ s.t.

$$f_i(x) = x^* A_i x + b_i^* x + x^* b_i,\ A_i = A_i^*$$

   Where $\cdot^*$ is Hermitian conjugate.

From this point on, X denotes $\mathbb{R}^n$ for real case or $\mathbb{C}^n$ for complex case.
We use the following notations:

Definition 1.1. For scalars, vectors, tuples of vectors or tuples of matrices $A = (A_1, ..., A_n) \in X^n$ and $B = (B_1, ..., B_n)$ the dot product is defined as following:

$$A \cdot B = \sum_{i=1}^{n} A_i \cdot B_i$$

For example, for a vector $c \in \mathbb{R}^n$ and a tuple of matrices $A = (A_1, ..., A_n)$, $A_i\colon m \times m$ the expression $c \cdot A = \sum\limits_{i=1}^{n} c_i A_i$ is a matrix $c \cdot A\colon m \times m$.

Definition 1.2. The image of f is denoted as F:

$$F = f(X)$$

Definition 1.3. The convex hull of F is denoted as G:

$$G = \operatorname{conv} F$$

Definition 1.4. The boundary points of F (or G) touched by a supporting hyperplane with the normal vector $c \in \mathbb{R}^m$:

$$\partial F_c = \partial G_c = \arg\min_{y \in F}(c \cdot y)$$

2 Functions

The library consists of a number of functions defined in separate .m files. Input format for the map is the following:

- The number $A(i, j, k)$ denotes i'th row and j'th column of the matrix A_k
- The number $b(i, j)$ denotes i'th element of the vector $b_j \in \mathbb{R}^m$

1. **Feasibility membership oracle.** Given:

- The map f as matrices A and vectors b
- A point $y \in \mathbb{R}^m$.

Determine: if $y \in F$

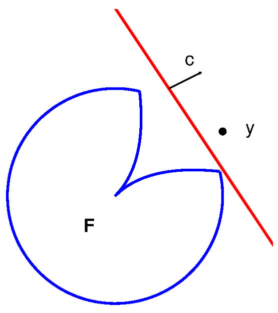

Figure 1: Infeasibility oracle: hyperplane c separates the point y^0 from the image F

```
is_infeasible = infeasibility_oracle(A, b, y)
```

This function tries to separate the point y from the convex hull G with a hyperplane. See Theorem 3.2 from the article.

Return value: 1 means that the separation was successful and the point $y \notin G$. This implies $y \notin F$. On the contrary, 0 means that the feasibility is uncertain.

2. **Boundary oracle.** Given:

   - The map f as matrices A and vectors b
   - A point $y \in G$
   - A direction $d \in \mathbb{R}^m$

   The following two tasks are considered:

   (a) **Find:** distance to the boundary from a given point inside G.

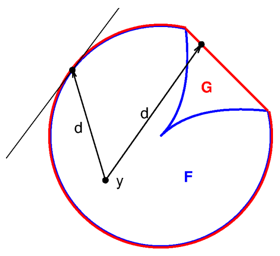

Figure 2: Boundary oracle: distance from y to the boundary ∂G in direction d

```
[t, is_in_F] = boundary_oracle(A, b, y, d)
```

This function finds the point $y + td$ on the boundary ∂G with the largest t:

$$t = \sup\{\tau \big| y + \tau d \in G\}$$

Return value:

- t is the largest step in direction d such that $y + td$ is still in G.
- is_in_F is a binary variable indicating if the resulting point $y + td$ belongs to F: it is 1 if it is true or 0 if the result is uncertain

Exception: if optimization task failed, in particular, if $y \notin G$ or the normal vector does not exist at that point.

(b) **Find:** the normal vector c at the boundary point $y + td$.

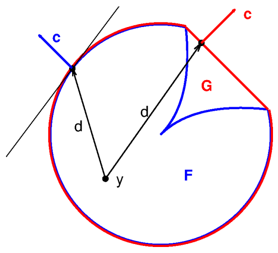

Figure 3: Boundary oracle: normal vector c at boundary point $y + td$

```
c = get_c_from_d(A, b, y, d)
```

This function obtains the normal vector c at the boundary point $y + td$ using dual problem (5) from the article.

Return value: the normal vector c s.t. $y + td \in \partial G_c$

Exception: if optimization task failed, in particular, if $y \notin G$ or the normal vector does not exist at this point.

3. **Nonconvexity certificate.** Given:

- The map f as matrices A and vectors b
- A point $y \in F$
- Number of iterations k

The following two tasks are considered:

(a) **Find:** a vector c such that ∂F_c is nonconvex.

```
c = get_c_minus(A, b, y, k)
```

This function is generating at most k random directions d and checks if the intersection of the boundary ∂G_c with a hyperplane with normal vector c at the boundary point $y + td$ is nonconvex.

Return value: c s.t. ∂F_c is nonconvex

Exception: if c was not found in k iterations

(b) **Determine:** if F is nonconvex.

```
is_nonconvex = nonconvexity_certificate(A, b, y, k)
```

This function checks if the image is nonconvex via obtaining $c \in C_-$. If c was found in k iterations, the image F is guaranteed to be nonconvex, Otherwise the result is uncertain

Return value: 1 if F is nonconvex, 0 if result is uncertain

4. **Positive-definite** $c \cdot A$**.** Given:

- The map f as matrices A and vectors b
- The initial normal vector p

The following three tasks are considered:

(a) **Find:** vector c_+, s.t. $c_+ \cdot A \succ 0$

```
c_plus = get_c_plus(A, k)
```

This function generates a random vector p and then finds c_+ nearest to it. Function generates at most k vectors p.

Return value: c_+ s.t. $c_+ \cdot A \succeq 0$

Exception: if c_+ was not found

(b) **Find:** c_+, s.t. the convex cut is maximal.

```
c_plus = get_max_c_plus(A)
```

This function returns the "best" vector c s.t. $c \cdot A \succeq 0$ and $\lambda_{\min}(c \cdot A) \to \max$. The spectrum of the resulting matrix $c_+ \cdot A$ is separated from 0 the most. This is a heuristic trying to achieve maximal value of $z_{\max}$ in this direction.

Return value: c_+ s.t. $c_+ \cdot A \succeq 0$

Exception: if c_+ was not found

(c) **Find:** c_+ close to given arbitrary p.

```
c_plus = get_near_c_plus(A, p, gamma);
```

This function finds the nearest to p vector c_+ such that $c_+ \cdot A \succeq 0$ heuristically from the neighbourhood of p.

Return value: c_+ s.t. $c_+ \cdot A \succeq 0$

Exception: if c_+ was not found

5. **Convex subpart.** Given:

- The map f as matrices A and vectors b
- The number $z_{\max}^{\text{guess}}$
- Number of iterations k
- Vector c_+ s.t. $c_+ \cdot A \succeq 0$

Find: maximal convex cut of F.

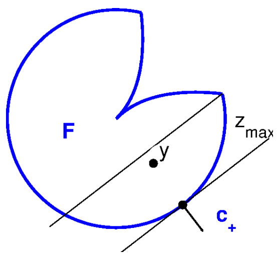

Figure 4: Convex subpart: maximal value of $z_{\max}$ such that a cut with a hyperplane c_+ is still convex

```
z_max = get_z_max(A, b, c_plus, z_max_guess, k)
```

This function returns the maximal value $z_{\max}$ such that the cut in the direction of c_+ of size $z_{\max}$ is still convex. This procedure is a heuristic (convexity of maximality is not guaranteed).

The value $z_{\max}^{\text{guess}}$ generates a point $y = y_0 - z_{\max}^{\text{guess}} c_+$, where y_0 is the touching point of hyperplane c_+ which is used for the nonconvexity certificate.

Return value: Maximal value $z_{\max}$ or Inf if no nonconvexities were found

Exception: None