CAQM: Convexity Analysis of Quadratic Maps

Anatoly Dymarsky, Elena Gryazina, Sergei Volodin, Boris Polyak

1 Additional functions in library/

Name	Input	Call	Description	Return value	Exception
Random map	Dimensions n, m	`get_random_f(n, m, is_complex)`	Generates random map f	`[A, b]`	None
Value at x	The point $x \in X$	`quadratic_map(A, b, x)`	Calculates $f(x)$	$y = f(x)$	None
Product $c \cdot A$	Normal vector c	`get_Ac(A, c)`	Calculates $c \cdot A$	$A_c = c \cdot A$	None
Get H_c	$c, y \in \mathbb{R}^m$	`get_H_c(A, b, c, y)`	Calculates $H_c = \begin{pmatrix} A_c & b_c \\ b'_c & -(c,y) \end{pmatrix}$	H_c	None
Minimize $z(c)$	c, c_+, step β	`minimize_z_c(A, b, c, c_plus, beta_initial, max_step)`	Calculates $\inf_{c \in C_-} z(c)$	`[z, c_array, z_array]`	If failed
$\mathbb{R}^n$ **projection**		`project(A, b, c, x_0, delta_c, normal, search_area_size)`	Projects $c + \Delta c$ to C_-	`[c_new, lambda]`	If failed
$\mathbb{C}^n$ **projection**		`project_descent(A, b, c, normal_1, normal_2)`	Projects c to C_-	`[c_new, distance]`	If failed
Gradient $\frac{\partial z}{\partial c}$	Normal c	`get_dz_dc(A, b, c)`	Calculates Q, $\nabla z(c)$, normal vectors n_1, n_2	`[Q, Q_inv, k, v, lambda_min, z, dz_dc, normal_re, normal_im, drho_dc]`	None
Change of basis	c_+	`change_basis(A, b, c_plus)`	Finds $\begin{cases} x = S(x' + x_0) \\ y = y' + y_0 \end{cases}$ s.t. $\begin{cases} c_+ \cdot A_0 = I \\ c_+ \cdot b_0 = 0 \end{cases}$	`[A_new, b_new, x0, y0]`	None
Boundary point ∂F_c	Normal c	`boundary_point(A, b, c)`	Calculates pre-image x: $f(x) \in \partial F_c$	`[x]`	If $c \cdot A < 0$

2 Numerical parameters in library/get_config.m

Parameter name	Target function	Purpose	Description	Default value
`Q_inv_eps`	`get_dz_dc`	Pseudoinverse	Tolerance for `pinv` function for Q matrix	10^{-5}
`c_plus_min_lambda`	`is_c_plus`	$\lambda > 0$ check	Minimal eigenvalue which is still considered > 0	10^{-4}
`c_minus_lambda_min`	`is_nonconvex`	$\lambda > 0$ check	Maximal eigenvalue which is still considered to be 0	10^{-7}
`c_minus_h_rank`	`is_nonconvex`	Rank check	Tolerance for `rank` function in homogeneous case	10^{-7}
`c_minus_ortho`	`is_nonconvex`	Orthogonality criteria	Maximal value of scalar product of $|x_0^T(c \cdot b)|$ s.t. they are still considered orthogonal	10^{-7}
`c_minus_f1f2rank`	`is_nonconvex`	Rank check	Tolerance for `rank` function for $f_1 \nparallel f_2$ check	10^{-7}
`descent_min_norm`	`minimize_z_c`	Stopping criteria	Minimal norm of a vector (gradient, norm)	10^{-3}
`descent_max_z`	`minimize_z_c`	Stopping criteria	Maximal value of z	10^{9}
`descent_rank_Q`	`minimize_z_c`	Stopping criteria	Tolerance for `rank(Q)` check	10^{-5}
`descent_min_beta`	`minimize_z_c`	Stopping criteria	Minimal step in direction of the gradient	10^{-15}
`descent_theta`	`minimize_z_c`	Step reduction	Multiplicative factor to reduce beta if projection failed	0.5
`descent_max_cos`	`minimize_z_c`	Stopping criteria	Maximal cos between grad and n s.t. they are still considered to be not parallel	$1 - 10^{-4}$
`project_descent_max_dist`	`project_descent`	Stopping criteria	Maximal tolerable distance to C_- as value of constraint	10^{-10}
`project_descent_min_alpha`	`project_descent`	Stopping criteria	Minimal step in the direction of the gradient	10^{-20}
`project_descent_theta`	`project_descent`	Step reduction	Multiplicative factor to reduce alpha	0.4
`project_bisection_min_rl`	`project`	Stopping criteria	Minimal value of $|r - l|$	10^{-8}
`project_bisection_min_value`	`project`	Stopping criteria	Minimal value of distance to C_- s.t. considered inside this set	10^{-9}
`project_bisection_max_value`	`project`	Precondition	Maximal value of the initial distance to C_- s.t. projection is not attempted	10^{-3}