# GSIPLib&Gen: A Library and Generator of General Semi-infinite Programming Test Problems

## Version 1.0

Tobias Seidel[a], Jan Schwientek[b]

Fraunhofer Institute for Industrial Mathematics (ITWM),

Fraunhofer-Platz 1, 67663 Kaiserslautern (Germany)

[a] tobias.seidel@itwm.fraunhofer.de (Corresponding author)

[b] jan.schwientek@itwm.fraunhofer.de

General semi-infinite programming (GSIP) is a 20 years-studied branch of continuous optimization and constitutes a very powerful modeling tool with many applications. There is a whole bunch of solution methods. However, with the exception of some these were mainly developed from a theoretical / conceptual point of view. The aim of this paper is to present a collection of test problems from academia as well as from diverse application areas which gives a common basis on which algorithms for the solution of GSIP problems can be tested and compared.For a popular GSIP application, namely design centering optimization it is possible to generate own problems in addition to pre-defined problems. All problems and the ingredients for stating a design centering problem are implemented in Matlab in an object-oriented manner.

# Contents

# 1. Introduction to General Semi-Infinite Optimization

In this section we introduce the terms of semi-infinite optimization, which are necessary to understand the subsequent sections of this manual and to work with the GSIP problem library and generator.

## 1.1. Basic terms

A *semi-infinite optimization problem* in its simplest shape, i.e. with one (semi-infinite) constraint, has the following form:

$$\begin{aligned} \mathsf{(G)SIP}: \quad & \min_{\mathbf{x}\in\mathbb{R}^m} \ f(\mathbf{x}) \\ & \text{s.t.} \quad g(\mathbf{x},\mathbf{y}) \le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}), \end{aligned} \tag{1.1}$$

where $f, g$ are real-valued, at least continuous functions on their respective domains and $Y : \mathbb{R}^m \rightrightarrows \mathbb{R}^n$ is a point-to-set mapping with $|Y(\mathbf{x})| = \infty$ for at least some $\mathbf{x}$. Because of the last, Problem (1.1) possesses an infinite number of constraints, but a finite number of decision variables, leading to the name semi-infinite optimization problem.

If the mapping $Y(\mathbf{x})$ does not depend on $\mathbf{x}$, i.e. $Y(\mathbf{x}) = \overline{Y} \subseteq \mathbb{R}^n$ for all $\mathbf{x} \in \mathbb{R}^m$, Problem (1.1) is called a *standard semi-infinite optimization problem* and abbreviated by SIP, otherwise Problem (1.1) is called a *general(ized) semi-infinite optimization problem* and abbreviated by GSIP. For a detailed introduction to semi-infinite optimization we refer in the case of SIP problems to the survey article [3] and the book [12] as well as for GSIP problems to the survey articles [2, 18] and the monographs [17, 30]. We focus on general semi-infinite optimization problems in this collection.

The set $Y(\mathbf{x})$, also called *infinite index set*, is usually given as solution set of a system of inequalities

$$Y(\mathbf{x}) := \{\mathbf{y} \in \mathbb{R}^n \mid v_j(\mathbf{x},\mathbf{y}) \le 0, \ j = 1,\dots,q\}.$$

The consideration of several semi-infinite constraints in combination with possibly completely different infinite index sets

$$g_i(\mathbf{x},\mathbf{y}) \le 0 \text{ for all } \mathbf{y} \in Y_i(\mathbf{x}), \ i = 1,\dots,p$$

is straight forward and will be utilized in this collection.

The key for the theoretical as well as the numerical treatment of semi-finite optimization problems lies in its *bi-level* structure. The parametric *lower level problem* is given by:

$$\begin{aligned} \mathsf{Q}(\mathbf{x}): \quad & \max_{\mathbf{y}\in\mathbb{R}^n} g(\mathbf{x},\mathbf{y}) \\ & \text{s.t. } v_j(\mathbf{x},\mathbf{y}) \le 0, \ j = 1,\dots,q. \end{aligned}$$

The decision variables $\mathbf{x}$ of the original problem become the parameters of the lower level problem and the index variables $\mathbf{y}$ its decision variables.

The function $\varphi(\mathbf{x}) := \max_{\mathbf{y}\in Y(\mathbf{x})} g(\mathbf{x},\mathbf{y})$ is called *optimal value function* of problem $\mathsf{Q}(\mathbf{x})$. Consequently, the feasible set $M$ of GSIP can be rewritten as

$$M = \{\mathbf{x} \in X \mid \varphi(\mathbf{x}) \le 0\}.$$

This is a description of $M$ with just finitely many, namely one constraint. But the constraint function $\varphi$ is only given implicitly and in general not differentiable. Furthermore, one has to solve an optimization problem to global optimality to compute a value of $\varphi$, which is a hard task generally.

## 1.2. Numerical methods

To date, solution methods for general semi-infinite optimization problems have been developed primarily from a conceptual point of view. To the best of our knowledge, comprehensive numerical evaluations exist only for the explicit smoothing approach [19, 17] in [17, 31, 32, 14] and its feasible variant [31, 22] in [31] as well as the transformation-based discretization method [14, 15] in the publications themselves. All in all, the methods developed so far rely on two concepts:

(1) generalization of methods for standard semi-infinite optimization problems and

(2) transformation of a general semi-infinite optimization problem into a standard one

The methods stemming from concept (1) can be further subdivided:

(A) discretization and exchange methods (e.g. [14, 15, 25, 24]),

(B) methods based on local reduction of the general semi-infinite problem (e.g. [20, 21, 23]),

(C) methods based on the reformulation of GSIP into a related problem class, so-called *lift-&-project* approaches (e.g. [1, 14, 17, 19, 22]).

**Discretization:** We briefly review the solution of standard semi-infinite optimization problems by means of discretization. For a detailed introduction to discretization methods we refer to [11, 12].

Let us consider a standard semi-infinite optimization problem with one semi-infinite constraints for sake of simplicity:

$$\begin{aligned} \mathsf{SIP}: \quad & \min f(\mathbf{x}) \\ & \text{s.t. } \mathbf{x} \in M := \{\mathbf{x} \in \mathbb{R}^m \mid g(\mathbf{x},\mathbf{y}) \leq 0 \text{ for all } \mathbf{y} \in Y\}, \end{aligned}$$

with $Y$ being a non-empty, compact, infinite (index) set, and real-valued, at least continuous functions $f, g$.

For a subset $\hat{Y}$ of $Y$ we introduce the optimization problem

$$\mathsf{SIP}(\hat{Y}): \quad \min\ f(\mathbf{x}) \ \text{ s.t. } \ \mathbf{x} \in M(\hat{Y})$$

with

$$M(\hat{Y}) := \left\{ \mathbf{x} \in \mathbb{R}^m \mid g(\mathbf{x},\mathbf{y}) \leq 0 \text{ for all } \mathbf{y} \in \hat{Y} \right\}.$$

If $\hat{Y} \subset Y$ is a finite set, $\mathsf{SIP}(\hat{Y})$ is called *discretized (SIP) problem* and also denoted by $P(\hat{Y})$.

The basic idea of discretization methods is to successively calculate a solution of discretized SIP problems $P(\dot{Y}_l), l \in \mathbb{N}_0$, by an algorithm for finite optimization problems, where $\{\dot{Y}_l\}_{l \in \mathbb{N}_0}$ is a sequence of finite subsets of $Y$. The grid sequence $\{\dot{Y}_l\}_{l \in \mathbb{N}_0}$ is either defined a priori or determined adaptively. In the latter case, informations of the $l$-th discretization stage are used to define the grid $\dot{Y}_{l+1}$.

The generalization of this idea to general semi-infinite optimization is difficult, because of the $\mathbf{x}$-dependence of the index set $Y(\mathbf{x})$ and, thus, its discretizations. In order to ensure the closedness of the feasible sets of the discretized problems, the discretization points must be constructed such that they depend at least continuously on $\mathbf{x}$. However, because of additional requirements that must be met for convergence of such methods in the GSIP-case, they are difficult to implement. Nevertheless, in [14, 15] an approach using a transformation is presented, avoiding these difficulties and performing well at the test problems of this library and problems stemming from gemstone cutting.

**Transformation:** In principle, under suitable assumptions, each GSIP problem can be transformed at least locally into an equivalent SIP problem (see [23, 29] for details). However, such a transformation is only useful in practise, if it's given globally.

The ideal situation is the following: Given a non-empty compact set $Z \subseteq \mathbb{R}^{\tilde{n}}$ and a map $t: \mathbb{R}^m \times Z \to \mathbb{R}^n$, such that $t(\mathbf{x}, Z) = Y(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^m$. Then, we can rewrite the general semi-infinite constraint

$$g(\mathbf{x},\mathbf{y}) \leq 0 \text{ for all } y \in Y(\mathbf{x})$$

to a standard semi-infinite constraint

$$\widetilde{g}(\mathbf{x},\mathbf{z}) := g(\mathbf{x}, t(\mathbf{x},\mathbf{z})) \leq 0 \text{ for all } \mathbf{z} \in Z. \tag{1.2}$$

For one-dimensional index sets $Y(\mathbf{x}) = [l(\mathbf{x}), u(\mathbf{x})]$ with $l(\cdot) < u(\cdot)$ such a transformation can easily be computed via convex combination of the interval limit. In higher dimensions a similar construction can be done for star-shaped index sets (see, e.g., [23]).

An advantage of this approach is that for SIP problems more sophisticated solution methods are available. However, a problematic disadvantage is that the transformation may destroy convexity in the lower level which is essential for solving a semi-infinite optimization problem performantly. In [14, 15] a method is presented which makes use of transforming the GSIP into a SIP, but avoids the destruction of convexity in the lower level by operating on the original convex GSIP lower level problems.

For enabling this library to be used with transformation-based methods, like the above mentioned one, we state a transformation for each problem in the collection, whenever it is possible.

Finally, with regard to the implementation of the problem library and generator, we rewrite problem (1.1) in more detail and repeat the naming.

In the implementation we allow and make use also of finite constraints on the upper as well as on the lower level variables and distinguish between inequality and equality as well as between linear and nonlinear constraints. In the lower level only inequality constraints are permitted. Furthermore, it is possible to handle multiple semi-infinite constraints with different index sets. Semi-infinite equality constraints are not admitted, because of technical difficulties. By that Problem (1.1) has the following form:

$$\begin{aligned} \mathsf{GSIP}: \quad & \min_{\mathbf{x}\in\mathbb{R}^m} \ f(\mathbf{x}) \\ & \text{s.t.} \quad g_i^{\mathrm{SI}}(\mathbf{x},\mathbf{y}) \leq 0 \text{ for all } \mathbf{y} \in Y_{\phi(i)}(\mathbf{x}),\ i \in I^{\mathrm{SI}}, \\ & \qquad g_i^{\mathrm{F}}(\mathbf{x}) \leq 0,\ i \in I^{\mathrm{F}}, \\ & \qquad h_j(\mathbf{x}) = 0,\ j \in J^{\mathrm{UL}}, \\ & \qquad \mathbf{A}^{\mathrm{UL}}\mathbf{x} \leq \mathbf{b}^{\mathrm{UL}}, \\ & \qquad \mathbf{C}^{\mathrm{UL}}\mathbf{x} = \mathbf{d}^{\mathrm{UL}}, \\ & \qquad \mathbf{l}^{\mathrm{UL}} \leq \mathbf{x} \leq \mathbf{u}^{\mathrm{UL}}, \end{aligned} \tag{1.3}$$

with

$$\begin{aligned} Y_k(\mathbf{x}) := \{\mathbf{y} \in \mathbb{R}^{n_k} \mid\ & v_{k,j}(\mathbf{x},\mathbf{y}) \leq 0,\ j \in J_k^{\mathrm{LL}}, \\ & \mathbf{A}_k^{\mathrm{LL}}(\mathbf{x}) \cdot \mathbf{y} \leq \mathbf{b}_k^{\mathrm{LL}}(\mathbf{x}), \\ & \mathbf{l}_k^{\mathrm{LL}}(\mathbf{x}) \leq \mathbf{y} \leq \mathbf{u}_k^{\mathrm{LL}}(\mathbf{x})\},\ k \in K, \end{aligned}$$

where $K$, $J_k^{\mathrm{LL}}$, $J^{\mathrm{UL}}$, $I^{\mathrm{F}}$, $I^{\mathrm{SI}}$ are finite index sets, $v_{k,j}$, $h_j$, $g_i^{\mathrm{F}}$, $g_i^{SI}$, $f$ are real valued, at least continuous functions, $\mathbf{A}_k^{\mathrm{LL}}(\cdot)$, $\mathbf{b}_k^{\mathrm{LL}}(\cdot)$, $\mathbf{l}_k^{\mathrm{LL}}(\cdot)$, $\mathbf{u}_k^{\mathrm{LL}}(\cdot)$ are matrix- resp. vector-valued, at least continuous functions and $\mathbf{A}^{\mathrm{UL}}$, $\mathbf{b}^{\mathrm{UL}}$, $\mathbf{C}^{\mathrm{UL}}$, $\mathbf{d}^{\mathrm{UL}}$, $\mathbf{l}^{\mathrm{UL}}$, $\mathbf{u}^{\mathrm{UL}}$ are matrices resp. vectors of matching dimension. The mapping $\phi$ assigns every semi-infinite constraint function to an infinite index set. Especially for the later introduced multiple body design-centering examples it is useful to have this information. For sake of simplicity we drop the indices, if they are obsolete.

Although already introduced, we want to repeat and summarize some terms with regard to the implementation of the collection:

- The function $f$ is called *objective (function).*
- The vector $\mathbf{x}$ constitute the *upper level* and the vector $\mathbf{y}$ the *lower level* (decision) variables.
- The functions $g_i^{\mathrm{SI}}, i \in I^{\mathrm{SI}}$, are called *semi-infinite constraint (functions).* They are constraint functions in the upper level as well as objective functions in the lower level.

- The sets $Y_k(\mathbf{x}), k \in K$ are called *infinite index sets* and constitute the feasible sets of the lower level problems.
- The functions $g_i^{\mathrm{F}}, i \in I^{\mathrm{F}}$, are the *finite* resp. *upper level nonlinear inequality constraint functions* (on $\mathbf{x}$).
- The functions $h_j, j \in J^{\mathrm{UL}}$, are the *finite* resp. *upper level nonlinear equality constraint functions* (on $\mathbf{x}$).
- The matrix $\mathbf{A}^{\mathrm{UL}}$ and the vector $\mathbf{b}^{\mathrm{UL}}$ constitute the *upper level linear inequalities* (on $\mathbf{x}$).
- The matrix $\mathbf{C}^{\mathrm{UL}}$ and the vector $\mathbf{d}^{\mathrm{UL}}$ constitute the *upper level linear equations* (on $\mathbf{x}$).
- The vectors $\mathbf{l}^{\mathrm{UL}}$ and $\mathbf{u}^{\mathrm{UL}}$ are the *upper level lower* and *upper bounds* (on $\mathbf{x}$).
- The functions $v_{k,j}, k \in K, j \in J_k$, are the *lower level nonlinear inequality constraint functions* (on $\mathbf{y}$, parametrized by $\mathbf{x}$).
- The matrix $\mathbf{A}_k^{\mathrm{LL}}$ and the vector $\mathbf{b}_k^{\mathrm{LL}}$, $k \in K$, constitute the *lower level linear inequalities* (on $\mathbf{y}$, parametrized by $\mathbf{x}$).
- The vectors $\mathbf{l}_k^{\mathrm{LL}}$ and $\mathbf{u}_k^{\mathrm{LL}}$, $k \in K$, are the *lower level lower* and *upper bounds* (on $\mathbf{y}$, parametrized by $\mathbf{x}$).
- The function $\mathbf{t}_k, k \in K$, is the *transformation function* transforming $Y_k(\mathbf{x})$ to $Z_k$ and back.

# 2. Basics about the collection

The examples are implemented in MATLAB [27]. This is done with an object oriented approach. Every implemented problem has its own class. The super-class of every problem is called `GSIPProblem` and inherits itself from the class `hgsetget`. From this super class functionality is inherited to the problems and the same structure of every problem is ensured. The description of this functionality is the main topic of this chapter.

We describe in the first section how the collection is started and simple problems can be initialized. We also describe the most important function evaluations which are needed to describe the problem. A complete list is given in the appendix. We focus here on problems found in the literature. In Chapter 3 we will describe more ways of generating examples using a modular principle. In the second section we describe how a new problem can be implemented which inherits again from the super class `GSIPProblem` the described functionality. In the final section we describe more methods which are implemented for the usage with this collection.

## 2.1. Getting Started: Initialize a problem

The first step, which has to be done every time before the collection can be used, is to load all needed paths. Therefore one first navigates in the folder `GSIPProblems` and then the script: `addpath` has to be executed.

We first only consider the initialization of problems directly given in the form of (1.3). All of these problems are listed in Appendix A.1. Later in chapter 3 we will describe more ways of generating examples. This will be done by reformulating design centering problems to general semi-infinite ones.

As mentioned before the problems are implemented in MATLAB as classes. Thus one problem can be initialized by using the constructor of this class. For example if we want to load Problem 1. We can easily do this by:

```
p=Problem1()
```

The Variable `p` now contains one instance of Problem1. The methods of this problem give us all information about the describing functions. For example we can compute the value of the objective function for a given pont by:

```
p.evalObjFun([0,0])
```

Some problems need an additional parameter for the description. This parameter has to be added as an input to the constructor. For example for Problem8:

```
p=Problem8(5)
```

If there is needed such an extra input, you can find it in the description of the problems in A.1.With:

```
help Problem1
```

A short description of Problem1 can be found.

The properties of the class `GSIPProblem` are protected. They can be read and modified by get and set functions. Functions to evaluate bounds, linear constraints and non-linear constraints are implemented separately. We give a full list of properties and methods for the class `GSIPProblem` in Appendix B.1. The most important methods can be found in the following table. We use the notations introduced in (1.3).

| **Function call** | **Inputs** | **Outputs** |
|---|---|---|
| `f=<problem>.evalObjFun(x)` | Upper level variables $(1 \times m)$. | Objective function value $(1 \times 1)$. |
| `g=<problem>.`<br>`evalSemiInfConstrs(i,x,y)` | Index of semi-infinite constraint $(1 \times 1)$, upper level variables $(1 \times m)$, $s$ points of infinite index set $(s \times n_{\phi(i)})$. | Value of $i$-th semi-infinite constraint $g_i^{SI}(\mathbf{x}, \mathbf{y})$ for $s$ points $(s \times 1)$. |
| `g=<problem>.`<br>`evalAllUppLevIneqConstrs(x)` | Upper level variables $(1 \times m)$. | Values of nonlinear constraints: $g_i^{NL}(\mathbf{x}), i \in I^{NL}$, linear inequalities: $\mathbf{A}^{UL}\mathbf{x} - \mathbf{b}^{UL}$ and bounds: $\mathbf{x} - \mathbf{u}^{UL}$, $\mathbf{l}^{UL} - \mathbf{x}$ (1×number of ordinary inequality constraints). The bounds $\pm$inf are dropped. |
| `g=<problem>.`<br>`evalAllUppLevEqConstrs(x)` | Upper level variables $(1 \times m)$. | Values of nonlinear equality constraints: $h_j^{NL}(\mathbf{x}), j \in J^{NL}$ and linear equalities: $\mathbf{C}^{UL}\mathbf{x} - \mathbf{d}^{UL}$ (1× number of equality constraints). |
| `v=<problem>.`<br>`evalAllLowLevConstrs(k,x,y)` | Index of infinite Index-set $(1 \times 1)$, Upper level variables $(1 \times m)$, point of infinite index set $(1 \times n_k)$. | Values of nonlinear describing functions $v_{k,j}(\mathbf{x}, \mathbf{y})$ for all j, linear describing functions: $\mathbf{A_k}^{LL}\mathbf{y} - \mathbf{b_k}^{LL}$ and bounds: $\mathbf{x} - \mathbf{u_k}^{LL}$, $\mathbf{l_k}^{LL} - \mathbf{x}$ (The bounds $\pm$inf are dropped) of the $k-th$ index set (1×number of describing functions). |
| `y=<problem>.`<br>`evalTrafoFun(k,x,z)` | Index of infinite Index-set $(1 \times 1)$, Upper level variables $(1 \times m)$, $s$ points of infinite index set $Z_k$ $(s \times \tilde{n}_k)$. | Transformed points $y = t_k(x, z)$ $(s \times n_k)$. |
| `z=<problem>.`<br>`evalInvTrafoFun(k,x,y)` | Index of infinite Index-set $(1 \times 1)$, Upper level variables $(1 \times m)$, $s$ points of infinite index set $Y_k(x)$ $(s \times n_k)$. | Points $z$ such that $t_k(x, z) = y$ $(s \times \tilde{n}_k)$. |

## 2.2. Implementing a new example

In the collection there is not only the possibility to use an already implemented problem, one can also implement a new one. This new problem can again be implemented as a subclass of the class `GSIPProblem`. If done so, all functionality described in this chapter is available for the new problem. For implementing a new problem one can proceed in the following way:

- In the first step a template has to be filled. In the folder `...GSIPProblems/ImplicitProblems` is a file `implicitProblemTemplate`. In there all functions are found. Some functions are not necessary to implement. If they are not needed, they can be deleted. These functions are marked with a small comment. To keep compatibility all functions have to take inputs and give outputs with the size specified in table B.2 from the Appendix B.1. Note, for the implementation of a class method, a further input `obj` for the object is needed. This input is not needed in the call of a function. Thats why this input is not listed. Of course more functions which are needed can be implemented. A list together with a description of all properties, which need to be specified for the new example, is given in the Appendix B.1 in Table B.1.

- After the implementation of a new problem one can check whether all functions work, take the right inputs and give the right outputs. This consistency check can be done by the function `checkProblem`. The function needs an instance of the problem as an input. If your Problem has the name `MyProblem` this can look as follows:
  ```
  p=MyProblem
  checkProblem(p,0,0)
  ```
  With this consistency check, one can check if a problem works correctly and it is easier to detect whether there is a mistake in the algorithm or in the problem. Moreover, one doesn't necessarily have to check if the functions have the right input and output sizes during runtime of an algorithm. If the second input is 1, the derivatives are compared to numerical derivatives. If the third input is 1 second derivatives are checked. Whenever a function doesn't work correctly or a derivative is not correct a warning with a small description is thrown.

## 2.3. Further Features

### 2.3.1. Plot problem (`plotSols`)

It is sometimes quite hard to see whether a algorithm gives a useful result or not. It's often easier to see this in a picture. Therefore we have implemented two plot routines. One is implemented only for design-centering problems (see chapter 3). We focus here on the plot routine implemented for all GSIP problems. We therefore need the following reformulation of the problem GSIP.

$$\begin{aligned} \mathsf{GSIP}: \quad & \min_{\mathbf{x} \in X \subseteq \mathbb{R}^m} \quad f(\mathbf{x}) \\ & \text{s.t.} \qquad Y_k(\mathbf{x}) \subseteq G_k(\mathbf{x}) \text{ for all } k \in K \end{aligned} \tag{2.1}$$

where

$$G_k(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^{n_k} \mid g_i^{SI}(\mathbf{x}, \mathbf{y}) \leq 0 \text{ for all } i \text{ with } \phi(i) = k\}$$

For a fixed $\mathbf{x}$ we can draw the sets $Y_k(\mathbf{x})$ and $G_k(\mathbf{x})$. And display the semi-infinite constraints in this way. We also allow it in the plot routines to mark sepecific points in different colours. This can be helpful especially when working with discretization methods or when the solution of the lower level problem should be marked. There are two functions which implement this routine in one and two dimensions. We describe them in the following:

- name:`<problemname>.plotContainmentCondition(indexOfLowLev,x,lb,ub,...`
  `points1,color1,points2,color2,...)`
- description: The first input `indexOfLowLev` is a vector. The containment conditions (2.1) is drawn for $k \in$`indexOfLowLev` and fixed `x`. By `lb` and `ub` the bounds of the plot have to be specified. If `lb` is a single vector, these bounds are used for every $k \in$`indexOfLowLev`. If it is a matrix the $k$-th row contains the bounds for the $k$-th index. If `lb` is a cell, the $k$-th entry contains the bounds for the $k$-th index in `indexOfLowLev`. The same holds for the upper bound `ub`. Special points can be marked by `points1,points2,...`. If `pointsi` is a matrix every row represents one point which is drawn for every indexset. If `pointsi` is a cell of matrices. Every entry represents points for one $k \in$`indexOfLowLev`. The corresponding color of every point set is given in `color1,color2,...`. Any matlab color specifier is possible.

The second possibility is to call the function `plotSols`. If no other routine is implemented the function looks as follows:

- name:`<problemname>.plotSols(x,points1,color1,points2,color2,...)`
- description: Calls the function `plotContainmentCondition`. The bounds are taken from the description of the infinite index-set.

For design-centering problems this function calls another plot routine. See for a detailed description Section 3.4.

# 3. Design Centering and Multiple-Body Design Centering Problems

A (multiple body) design-centering problem consists of maximizing the sum of volumes of parametrized bodies $Y_k(\mathbf{x_k})$, $k = 1, \dots q$. These bodies are called designs. The designs should be contained in a set $C$. This set is called container. For every design we get the following condition.

$$Y_k(\mathbf{x}) \subseteq C, k \in K \tag{3.1}$$

To reformulate them to semi-infinite constraints we assume that $C$ is given by functional constraints:

$$C = \{\mathbf{y} \in \mathbb{R}^n \mid g_i(\mathbf{y}) \leq 0 \text{ for all } i \in I\}.$$

We obtain the following constrains:

$$g_i(\mathbf{y}) \leq 0 \text{ for all } \mathbf{y} \in Y_k(\mathbf{x_k}), i \in I \text{ and } k = 1, \dots q$$

Moreover, we want two designs $Y_{k_1}(\mathbf{x_{k_1}})$ and $Y_{k_2}(\mathbf{x_{k_2}})$ with $k_1 \neq k_2$ not to overlap or to have some minimal distance. There are different ways to ensure this condition. We assume for the separation condition that the designs are convex. Under this assumption we can describe the separation with a hyperplane:

$$\begin{aligned} \boldsymbol{\eta_{k_1,k_2}}^T \mathbf{y} &\leq \beta_{k_1,k_2} - \frac{\delta}{2} \text{ for all } \mathbf{y} \in Y_{k_1}(\mathbf{x}), \\ \boldsymbol{\eta_{k_1,k_2}}^T \mathbf{y} &\geq \beta_{k_1,k_2} + \frac{\delta}{2} \text{ for all } \mathbf{y} \in Y_{k_2}(\mathbf{x}), \end{aligned} \tag{3.2}$$

where $\boldsymbol{\eta_{k_1,k_2}}$ and $\beta_{k_1,k_2}$ describe the hyperplane and $\delta$ is the asked distance between two designs. Note, if the designs are not convex, this condition is too strong. We assume that the normal vector is normed, i.e. :

$$< \boldsymbol{\eta_{k_1,k_2}}, \boldsymbol{\eta_{k_1,k_2}} >= 1,$$

where $< \cdot, \cdot >$ denotes the scalar-product.

We thus have two different types of semi-infinite constraints. For every design we have the inclusion constraints (3.1) and for each pair of designs the separation constraints (3.2). The upper level variables $\mathbf{x}$ consist of the parameters $\mathbf{x_k}$ of the designs and the separating hyperplanes $(\boldsymbol{\eta_{k_1,k_2}}, \beta_{k_1,k_2})$

We assume that the $k$-th design can be described as follows:

$$\begin{aligned} Y_k(\mathbf{x_k}) := \{\mathbf{y} \in \mathbb{R}^n \mid & v_{j,k}(\mathbf{x_k}, \mathbf{y}) \leq 0, \ j \in J_k, \\ & \mathbf{A_k}^{LL}(\mathbf{x_k}) \cdot \mathbf{y} \leq \mathbf{b_k}^{LL}(\mathbf{x_k}), \\ & \mathbf{l_k}^{LL}(\mathbf{x_k}) \leq \mathbf{y} \leq \mathbf{u_k}^{LL}(\mathbf{x_k})\} \end{aligned}$$

Moreover, we allow nonlinear and linear constraints on the parameters:

$$\begin{aligned} & g_{i,k}^{NL}(\mathbf{x_k}) \leq 0, i \in I_k^{NL}, \\ & h_{j,k}(\mathbf{x_K}) \leq 0, j \in J_k^{NL}, \\ & \mathbf{A_k}^{UL} \mathbf{x_k} \leq \mathbf{b_k}^{UL}, \\ & \mathbf{C_k}^{UL} \mathbf{x_k} = \mathbf{d_k}^{UL}, \\ & \mathbf{l_k}^{UL} \leq \mathbf{x_k} \leq \mathbf{u_k}^{UL}. \end{aligned}$$

By some abuse of notation we can denote by $Y_k(\mathbf{x}) := Y_k(\mathbf{x_k})$ the $k-th$ indexset of the general semi-infinite problem. Every nonlinear constraint on $\mathbf{x_k}$ gives a nonlinear constraint on $\mathbf{x}$ and a linear one gives a linear one.

## 3.1. Details of the implementation

For the implementation it is important to set the indices of the constraints and the parameters $\mathbf{x}$. We decided to do this in the following way:

- Parameters $\mathbf{x}$
  We begin with the parameters of the designs. The first parameters are the parameters of the first design $\mathbf{x_1}$, the next parameters are the parameters of the second design $\mathbf{x_2}$ and so on. The next parameters are the normal vectors and the translation of the separating hyperplanes. We begin with the normal vector and the translation value of the separation of the first and the second design $(\boldsymbol{\eta_{1,2}}, \beta_{1,2})$, then follows the separation of the first to the third and so on. We continue with the separation of the second design to the third and so forth.
- Semi-infinite constraints:
  The first $|J^{UL}|$ constraints are the containment constraints for the first design, the second $|J^{UL}|$ are the containment constraints for the second design and so on. All together there are the number of designs times $|J^{UL}|$ such containment constraints. After the containment constraints the separation constraints follow. First the separation of the first design to the second design, to the third design and so on. Then the separation of the second design to the third, to the fourth design and so on. All together there are $\frac{(k-1)k}{2} \cdot 2$ such separation constraints where $k$ is the number of designs
- Further constraints on $\mathbf{x}$:
  We begin with the constraints on the parameters for the first Design, then the second Design and so on. The structure is carried over. Nonlinear constraints on the parameters for the $k$-th design give nonlinear constraints on the upper level variables $\mathbf{x}$. Linear ones give linear constraints. After the nonlinear equality constraints on the design parameters the nonlinear equality constraints on the separation parameters. We add the constraints

  $$< \boldsymbol{\eta_{k_1,k_2}}, \boldsymbol{\eta_{k_1,k_2}} >= 1$$

  in the same order as the parameters.

## 3.2. Initialize an implemented design-centering or multiple body design-centering problem

There are different ways to initialize a design-centering problems. First there is in Appendix A.2 a list of examples, that can be initialized as described in chapter 2. These are examples which were found in the literature. For example:
`p=DCProblem1`
The variable `p` contains a GSIPProblem which corresponds to the first design centering problem given in A.2.
But given the designs and the container the reformulation is generic. Thats why one can get a problem also by defining the designs and the container. All implemented designs and containers are given in Appendix A.2.1 and A.2.2. An arbitrary (multiple body) design-centering problem can be initialized in three steps

- choose one or several designs:
  Also designs are implemented as an individual class. They can be initialized by it's constructor. For example if we want to get the design `DesignCircle` we can do this by:
  `d=DesignCircle`

Some of the designs need more information for example the dimension. Again we will remark this in the description of the according design in A.2.1. For example we could get the same circle by:
`d=DesignHyperBall(2)`

- choose a container:
  After we have chosen at least one design we have to choose a container as well. Again we get a container by it's constructor. For example if one wants to consider a Triangle:
  `c=ContainerTriangle`
  Again some containers are given more genarally. The information needed to construct the container can be found in the description of the container.
- plug both together:
  Now we have chosen the designs and the container. To get a design-centering problem with one design and one containter one can use the constructor `DCProblemAsGSIP` The constructor needs a design and a container and returns a GSIP, which has all the properties and methods described in chapter 2. So let `d` be a design and `c` be a container. We get the corresponding GSIP by:
  `p=DCProblemAsGSIP(d,c)`
  To get a multiple body design-centering problem we use the constructor `MBProblemsAsGSIP`. This method needs a container `c`, at least 2 designs `d1,d2,...,dk` and a distance `h` that should be between the designs. For 3 Designs this looks as follows:
  `p=MBProblemAsGSIP(c,h,d1,d2,d3)`
  This instance again has the same properties and methods described in section 2. Some examples for the initialization of design-centering problems can be found in the implementation.

## 3.3. Implementing a new design or a new container

The implementation of a design and a container works nearly the same way as the implementation of a general semi-infinite optimization problem. At first one can find templates in the folders `...ImplicitProblems/Designs` or `...ImplicitProblems/Containers`. They have the names `DesignTemplate` and `ContainerTemplate`. In these templates one can find all the functions that are necessarily needed and which can be deleted. All the functions one is going to implement must have the same size of inputs and outputs described in Table B.4 or B.6 respectively (given in the Appendices B.2 and B.3). Note that for every function (except for the constructor) one more input for the object is needed.
The properties that are needed for the implementation of a own example are listed in the tables B.3 and B.5 (given in the Appendices B.2 and B.3).
After the implementation of a new design or container you can again check whether all the functions work and have the correct inputs and outputs. This consistency check can be done by the function `checkDesign` or the function `checkContainer`. The function needs an instance of the object as an input. If the design has the name `myDesign` this can look as follows:

```
d=myDesign
checkDesign(d,1,1)
```

Second input: derivatives are checked. Third input: second derivatives are checked. Analogous for the container.

## 3.4. Plot a design-centering problem

For all designs and containers given in this collection we have implemented routines that can draw them in two and three dimensions. The routines are given as follows:

- Name: `<designname>.plotDesign(x)`
  Description: Plots the design in green for current parameter `x`.
- Name: `<containername>.plotContainer`
  Description: Plots the container in blue.

With these auxiliary functions we can illustrate the design-centering problem for given parameters `x`. We first plot the container in blue, then the designs in green and finally the separating hyperplanes in red. This is done when the function `plotSols(x)` is called for a two dimensional design-centering problem. In a three dimensional problem the same is done. Except for the hyperplanes They are not drawn in three dimensions.

# A. List of Examples

In the first Appendix we list all examples which are currently implemented in the collection. We begin with Problems which are formulated as GSIP directly (native Examples). We then proceed with Design-Centering problems which can be reformulated as GSIP. We list designs and containers separately. Furthermore all formulations as GSIPs are given.

## A.1. GSIP problems from literature (native ones)

In this chapter we list all problems we have found in the literature and which are not a design centering or robust portfolio problem. The name of each problem is also the name in the implementation. Thus also the name of the constructor. The first problem can be initialized by: `Prolem1`
The usage of these problems was described in detail in chapter 2. We will now describe every problem in detail. For some problems the index set can become empty. We have implemented constraints on the parameters $\mathbf{x}$ that exclude this effect.

**Problem 1**

Taken from [10].

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^2} \quad & x_1 + x_2 \\ \text{s.t.} \quad & -y \leq 0 \text{ for all } y \in Y(\mathbf{x}), \\ & -1 \leq x_1 \leq 1, \\ & -1 \leq x_2 \leq 1, \end{aligned}$$

where

$$\begin{aligned} Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } \quad & -y \leq -x_1, \\ & -y \leq -x_2\ \} \end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned} t: \quad \mathbb{R}^2 \times [0,\infty) \quad & \to \quad \mathbb{R} \\ (\mathbf{x}, z) \quad & \mapsto \quad t(\mathbf{x}, z) = \begin{cases} x_1 + z & \text{, if } x_1 \geq x_2, \\ x_2 + z & \text{, if } x_1 \leq x_2 \end{cases} \end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is linear. The optimal value function $\varphi(\mathbf{x})$ is non-convex. The index set can be given explicitly by: $Y(\mathbf{x}) = [\max\{x_1, x_2\}, \infty]$. Thus the feasible set is explicitly given by $M = \{x \in [-1,1]^2 \mid \max\{x_1, x_2\} \geq 0\}$. The two optimal solutions are $(0,-1)$ and $(-1,0)$ and the optimal value is $-1$.

---

**Problem 2**

Taken from [16].

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^2} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & y - x_1 - x_2 \leq 0 \text{ for all } y \in Y(\mathbf{x}), \end{aligned}$$

where

$$\begin{aligned} Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } \quad & -2y \leq x_2 + 3, \\ & y \leq x_1 - 2\ \} \end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned} t: \quad \mathbb{R}^2 \times [0,1] \quad & \rightarrow \quad \mathbb{R} \\ (\mathbf{x}, z) \quad & \mapsto \quad t(\mathbf{x}, z) = \begin{cases} z(-\frac{3}{2} - \frac{x_2}{2}) + (1-z)(x_1 - 2) & \text{, if } 2x_1 + x_2 - 1 \geq 0, \\ \text{n. def.} & \text{, otherwise} \end{cases} \end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is linear. The optimal value function $\varphi(\mathbf{x})$ is non-convex. The index set $Y(\mathbf{x})$ is empty for $2x_1 + x_2 - 1 < 0$, otherwise it is explicitly given by $Y(\mathbf{x}) = \left[\frac{-3-x_2}{2}, x_1 - 2\right]$. The feasible set is explicitly given by: $M = \{x \in \mathbb{R}^2 \mid x_2 - 2 \leq 0,\ 2x_1 + x_2 - 1 \geq 0\} \cup \{x \in \mathbb{R}^2 \mid 2x_1 + x_2 - 1 < 0\}$.
The linear constraint $2x_1 + x_2 - 1 \geq 10^{-6}$ can be added in the implementation.

---

**Problem 3**

Taken from [16].

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^2} \quad & \left(x_1 + \tfrac{1}{2} - \tfrac{1}{1+\sqrt{5}}\right)^2 + (x_2 - 2.5)^2 \\ \text{s.t.} \quad & y - x_1 + x_2 \leq 0 \text{ for all } y \in Y(\mathbf{x}), \\ & -5 \leq x_1 \leq 5, \\ & -5 \leq x_2 \leq 5, \end{aligned}$$

where

$$\begin{aligned} Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } \quad & -2y \leq x_2 + 3, \\ & y \leq x_1 - 2, \\ & -5 \leq y \leq 5 \ \} \end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned} t: \quad \mathbb{R}^2 \times [0,1] \quad & \rightarrow \quad \mathbb{R} \\ (\mathbf{x}, z) \quad & \mapsto \quad t(\mathbf{x}, z) = \begin{cases} (1-z)\frac{-x_2-3}{2} + z(x_1 - 2) & \text{, if } 2x_1 + x_2 - 1 \geq 0, \\ \text{n. def.} & \text{, otherwise} \end{cases} \end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is linear. The optimal value function $\varphi(\mathbf{x})$ is convex. The index set $Y(\mathbf{x})$ is empty for $2x_1 + x_2 - 1 < 0$, otherwise it is explicitly given by $Y(\mathbf{x}) = \left[\frac{-3-x_2}{2}, x_1 - 2\right]$. Thus the feasible set is explicitly given by $M = \{x \in [-5,5]^2 \mid x_2 - 2 \leq 0, 2x_1 + x_2 - 1 \geq 0\} \cup \{x \in \mathbb{R}^2 \mid 2x_1 + x_2 - 1 < 0\}$.
The linear constraint $2x_1 + x_2 \geq 1 + 10^{-6}$ can be added in the implementation.

---

**Problem 4**

Taken from [13].

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^4} \quad & x_4 \\ \text{s.t.} \quad & 3(x_1 - y)^2 + (2-y)x_2^2 + 5(x_3^2 + y) + 2x_1 + 3x_2 - x_3 + e^{4y^2} - x_4 \leq 0 \text{ for all } y \in Y(\mathbf{x}), \end{aligned}$$

where

$$\begin{aligned} Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } \quad & y \leq -\tfrac{1}{4}\sin(x_1 x_2) + \tfrac{1}{2}, \\ & 0 \leq y \leq 1 \qquad \} \end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned} t: \quad \mathbb{R}^4 \times [0,1] \quad & \rightarrow \quad \mathbb{R} \\ (\mathbf{x}, z) \quad & \mapsto \quad t(\mathbf{x}, z) = (1-z) \cdot \left(\tfrac{1}{2} - \tfrac{1}{4}\sin(x_1 x_2)\right) \end{aligned}$$

**Remarks:** The problem is unbounded. The lower level problem $Q(\mathbf{x})$ is non-convex. The index set can be given explicitly by $Y(\mathbf{x}) = \left[0, \frac{1}{2} - \frac{1}{4}\sin(x_1 x_2)\right]$.

---

**Problem 5**

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^2} \quad & x_1^2 + x_2^2 \\ \text{s.t.} \quad & -(y_1 - x_1)^2 - (y_2 - x_2)^2 + 1 \le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}), \end{aligned}$$

where

$$\begin{aligned} Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } & -y_1 \le -x_1, \\ & -y_2 \le 0 \quad \} \end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^2 \times [0,\infty)^2 & \rightarrow \mathbb{R}^2 \\ (\mathbf{x}, \mathbf{z}) & \mapsto \mathbf{t}(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} x_1 + z_1 \\ z_2 \end{pmatrix} \end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is convex. The optimal value function $\varphi(\mathbf{x})$ is non-convex. The index set can be given explicitly by $Y(\mathbf{x}) = [x_1, \infty) \times [0, \infty)$. It is unbounded. The solution of the lower level problem is $(x_1, \max(0, x_2))$. Thus the feasible set is explicitly given by $M = \{x \in \mathbb{R}^2 \mid 1 - x_2^2 \le 0, -1 \ge x_2\}$. The optimal solution is $(0, -1)$ and the optimal value is 1.

---

**Problem 6**

Taken from [26].

$$\begin{aligned} \min_{x\in\mathbb{R}} \quad & x \\ \text{s.t.} \quad & -\tfrac{7}{4} - x - y \le 0 \text{ for all } y \in Y(x), \\ & -1 \le x \le 1, \end{aligned}$$

where

$$Y(x) = \{y \in \mathbb{R} \text{ with } -1 - x^2 \le y \le 1 + x^2\}$$

Transformation to a fixed index set:

$$\begin{aligned} t: \quad \mathbb{R} \times [0,1] & \rightarrow \mathbb{R} \\ (x, z) & \mapsto t(x, z) = z \cdot (-1 - x^2) + (1 - z) \cdot (1 + x^2) \end{aligned}$$

**Remarks:** The lower level problem $Q(x)$ is linear. The optimal value function $\varphi(x) = -\frac{7}{4} - x + 1 + x^2$ is convex. The index set is explicitly given by: $Y(x) = [-1 - x^2, 1 + x^2]$. The solution of the lower-level-problem is $-1 - x^2$. The feasible set is explicitly given by: $M = \{x \in [-1, 1] \mid -\frac{3}{4} - x + x^2 \le 0\} = [-0.5, 1]$. The optimal solution is $-0.5$.

---

### Problem 7

Taken from [7].

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\mathbb{R}^2} & x_2 \\ \text{s.t.} & -x_2 - y^3 \leq 0 \text{ for all } y \in Y(\mathbf{x}), \end{array}$$

where

$$\begin{array}{rl} Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } & 2x_2 - y^3 - x_1^2 \leq 0, \\ & \qquad\qquad\quad y \leq 0\,\} \end{array}$$

Transformation to a fixed index set:

$$\begin{array}{rcl} t: \ \mathbb{R}^2 \times [0,1] & \to & \mathbb{R} \\ (\mathbf{x}, z) & \mapsto & t(\mathbf{x}, z) = \begin{cases} z \cdot \sqrt[3]{2x_2 - x_1^2} & , \text{if } 2x_2 - x_1^2 \leq 0, \\ \qquad \text{n. def.} & , \text{otherwise} \end{cases} \end{array}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The optimal value function $\varphi(\mathbf{x}) = -3x_2 + x_1^2$ is convex. The index set is empty if $2x_2 - x_1^2 > 0$ is explicitly given by $Y(\mathbf{x}) = [\sqrt[3]{2x_2 - x_1^2}, 0]$. Thus the feasible set is explicitly given by: $M = \{\mathbf{x} \in \mathbb{R}^2 \mid 2x_2 - x_1^2 \leq 0, -3x_2 + x_1^2 \leq 0\} \cup \{\mathbf{x} \in \mathbb{R}^2 \mid 2x_2 - x_1^2 > 0\}$. The optimal solution is $(0, 0)$ and the optimal value is 0.
The non-linear constraint $2x_2 - x_1^2 \leq -10^{-6}$ can be added in the implementation.

---

### Problem 8

Taken from [6].

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\mathbb{R}^m} & \sum_{i=1}^m \left(\frac{3(m-i+1)}{m} x_i^2 - 2x_i\right) \\ \text{s.t.} & y + \sum_{i=1}^m x_i^2 - 7 \leq 0 \text{ for all } y \in Y(\mathbf{x}), \end{array}$$

where

$$Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } \ -100 \leq y \leq \textstyle\sum_{i=1}^m \frac{3i}{m} x_i^2 - 6\}$$

Transformation to a fixed index set:

$$\begin{array}{rcl} t: \ \mathbb{R}^m \times [0,1] & \to & \mathbb{R} \\ (\mathbf{x}, z) & \mapsto & t(\mathbf{x}, z) = -100z + (1 - z) \cdot \left(\sum_{i=1}^m \frac{3i}{m} x_i^2 - 6\right) \end{array}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The optimal value function $\varphi(\mathbf{x})$ is convex. The index set is explicitly given by $Y(\mathbf{x}) = [-100, \sum_{i=1}^m \frac{3i}{m} x_i^2 - 6]$. Thus the feasible set is explicitly given by: $M = \{x \in \mathbb{R}^m \mid \sum_{i=1}^m \frac{m+3i}{m} x_i^2 - 13 \leq 0\}$.
This problem is given for a variable number of parameters $\mathbf{x}$. To initialize the problem the number of parameters has to be set, e.g. `Problem8(10)`.

---

**Problem 9**

Taken from [6].

$$\begin{array}{ll}\min\limits_{\mathbf{x}\in\mathbb{R}^m} & \sum_{i=1}^m \left(\frac{3(m-i+1)}{m}x_i^2 - 2x_i\right) \\ \text{s.t.} & \sum_{i=1}^m (x_i - y)^2 - 10m \le 0 \text{ for all } y \in Y(\mathbf{x}),\end{array}$$

where

$$\begin{array}{rl}Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } & -y \le -1, \\ & y \le \sum_{i=1}^m x_i^2 + 1\}\end{array}$$

Transformation to a fixed index set:

$$\begin{array}{rlll}t: & \mathbb{R}^m \times [0,1] & \to & \mathbb{R} \\ & (\mathbf{x}, z) & \mapsto & t(\mathbf{x}, z) = -z + (1-z)\cdot\left(\sum_{i=1}^m x_i^2 + 1\right)\end{array}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The optimal value function $\varphi(\mathbf{x})$ is convex. The index set is explicitly given by: $Y(\mathbf{x}) = [-1, \sum_{i=1}^m x_i^2 + 1]$.
This problem is given for a variable number of parameters $\mathbf{x}$. To initialize the problem the number of parameters has to be set, e.g. `Problem9(10)`.

---

**Problem 10**

Taken from [4].

$$\begin{array}{ll}\min\limits_{\mathbf{x}\in\mathbb{R}^2} & (x_1 - \frac{1}{4})^2 + x_2^2 \\ \text{s.t.} & y + x_2 \le 0 \text{ for all } y \in Y(\mathbf{x}), \\ & -1 \le x_1 \le 1, \\ & -1 \le x_2 \le 1,\end{array}$$

where

$$\begin{array}{rl}Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } & y^2 - x_1 \le 0, \\ & -1 \le y \le 1\}\end{array}$$

Transformation to a fixed index set:

$$\begin{array}{rlll}t: & \mathbb{R}^2 \times [0,1] & \to & \mathbb{R} \\ & (\mathbf{x}, z) & \mapsto & t(\mathbf{x}, z) = \begin{cases} -z\cdot\sqrt{x_1} + (1-z)\cdot\sqrt{x_1} & \text{, if } x_1 \ge 0, \\ \text{n. def.} & \text{, otherwise}\end{cases}\end{array}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is convex. The optimal value function $\varphi(\mathbf{x})$ is non-convex. The index set $Y(\mathbf{x})$ is empty for $x_1 < 0$, otherwise it is explicitly given by $Y(\mathbf{x}) = [-\sqrt{x_1}, \sqrt{x_1}]$. Thus, the feasible set is explicitly given by: $M = \{x \in [-1,1]^2 \mid x_1 < 0\} \cup \{x \in [-1,1]^2 \mid x_1 \ge 0, \sqrt{x_1} \le x_2\}$. The optimal solution is $(0,0)$ and the optimal value is $\frac{1}{16}$.
In the implementation the linear constraint $x_1 \ge 10^{-6}$ can be added.

---

**Problem 11**

Taken from [4].

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^2}\quad & x_2 \\
\text{s.t.}\quad & -y^3 + x_2 \le 0 \text{ for all } y \in Y(\mathbf{x}), \\
& -1 \le x_1 \le 1, \\
& -1 \le x_2 \le 1,
\end{aligned}$$

where

$$\begin{aligned}
Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } \; & 2x_2 - y^3 - x_1^2 \le 0, \\
& -1 \le y \le 0\}
\end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned}
t: \quad \mathbb{R}^2 \times [0,1] \quad & \to \quad \mathbb{R} \\
(\mathbf{x}, z) \quad & \mapsto \quad t(\mathbf{x}, z) = \begin{cases} z \cdot \max(\sqrt[3]{2x_2 - x_1^2}, -1) & \text{, if } 2x_2 - x_1^2 \le 0, \\ \text{n. def.} & \text{, otherwise} \end{cases}
\end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The optimal value function $\varphi(\mathbf{x})$ is non-convex. The index set $Y(\mathbf{x})$ is empty for $2x_2 - x_1^2 > 0$, otherwise it is explicitly given by: $Y(\mathbf{x}) = [\max\{\sqrt[3]{2x_2 - x_1^2}, -1\}, 0]$.Thus, the feasible set is explicitly given by: $M = \{x \in [-1,1]^2 \mid x_1^2 < 2x_2\} \cup \{x \in [-1,1]^2 \mid x_2 = -1\} \cup$ 
$\{x \in [-1,1]^2 \mid -1 \le 2x_2 - x_1^2 \le 0 \wedge x_2 - x_1^2 \le 0\}$.
The optimal solution is $(0,0)$ and the optimal value is 0.
In the implementation the non linear constraint $2 \cdot x_2 - x_1^2 \le -10^{-6}$ can be added.

---

**Problem 12**

Taken from [9].

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^2} \quad & \tfrac{1}{2}x_1^4 + 2x_1x_2 - 2x_1^2 \\
\text{s.t.} \quad & y_1^2 + y_2^2 - x_1 + x_1^2 - x_2 \le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}), \\
& 0 \le x_1 \le 1, \\
& 0 \le x_2 \le 1,
\end{aligned}$$

where

$$\begin{aligned}
Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^3 \text{ with } & y_1^2 + y_2^2 + y_3^2 - x_1 \le 0, \\
& 0 \le y_1 \le 1, \\
& 0 \le y_2 \le 1, \\
& 0 \le y_3 \le 1 \ \}
\end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned}
\mathbf{t}: \quad \mathbb{R}^2 \times \{\mathbf{z} \in [0,1]^3 \mid z_1^2 + z_2^2 + z_3^2 \le 1\} &\to \mathbb{R}^3 \\
(\mathbf{x}, \mathbf{z}) &\mapsto \mathbf{t}(\mathbf{x}, \mathbf{z}) = x_1 \cdot \mathbf{z}
\end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The optimal value function $\varphi$ is convex. The index set is explicitly given by $Y(x) = \{y \in [0,1]^3 \mid y_1^2 + y_2^2 + y_3^2 \le x_1\}$. Thus, the feasible set is explicitly given by $M = \{x \in [0,1]^2 \mid x_1^2 \le x_2\}$. The optimal solution is $(1,1)$ and the optimal value is $\frac{1}{2}$.

---

**Problem 13**

Taken from [23].

$$\begin{aligned}
\min_{x\in\mathbb{R}} \quad & x^2 \\
\text{s.t.} \quad & x - y \le 0 \text{ for all } y \in Y(x), \\
& -1 \le x \le 1,
\end{aligned}$$

where

$$\begin{aligned}
Y(x) = \{y \in \mathbb{R} \text{ with } & (y+1)^2 + x^2 \le 0, \\
& -2 \le y \le 2\}
\end{aligned}$$

**Remarks:** The lower level problem $Q(x)$ is convex. The optimal value function $\varphi(x)$ is non-convex. The index set $Y(\mathbf{x})$ is empty for $x \neq 0$, otherwise $Y(x) = \{1\}$. Thus, the feasible set is given by: $M = [-1,1] \backslash \{0\}$. The minimum is not reached, but the infimum is attained at $x = 0$.

---

**Problem 14**

Taken from [8].

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{R}^2} \quad & -x_1 \\ \text{s.t.} \quad & y_2 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}), \\ & -5 \leq x_1 \leq 5, \\ & -5 \leq x_2 \leq 5,\end{aligned}$$

where

$$\begin{aligned}Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \; & y_2 - y_1^2 - x_2 \leq 0, \\ & -x_2 y_1 + y_2 \leq x_1, \\ & -2 \leq y_1 \leq 2, \\ & -4 \leq y_2 \leq 4 \;\}\end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex.

**Problem 15**

Taken from [9],Ex.5.1.

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^2} \quad & 4x_1^2 - x_2 - x_2^2 \\
\text{s.t.} \quad & x_2 - y_2 \le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}), \\
& -3 \le x_1 \le 2, \\
& -3 \le x_2 \le 2,
\end{aligned}$$

where

$$\begin{aligned}
Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^3 \text{ with } \ (y_1+y_2)^2 - y_3 &\le 0, \\
y_1 &\le x_1, \\
y_2 &\le x_1, \\
-4 \le y_1 &\le 4, \\
-4 \le y_2 &\le 4, \\
0 \le y_3 &\le 16\,\}
\end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned}
\mathbf{t}: \quad \mathbb{R}^2 \times [0,1]^3 \quad &\to \quad \mathbb{R}^3 \\
(\mathbf{x},\mathbf{z}) \quad &\mapsto \quad \mathbf{t}(\mathbf{x},\mathbf{z}) = \begin{cases} \begin{pmatrix} -4z_1 + (1-z_1)x_1 \\ \max(-4,-4-y_1)z_2 + (1-z_2)x_1 \\ z_3(y_1+y_2)^2 + (1-z_3)16 \end{pmatrix} & \text{, if } x_1 \ge -2, \\ \\ \text{n. def.} & \text{, otherwise} \end{cases}
\end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is convex. The optimal value function $\varphi(\mathbf{x})$ is non-convex. The index set $Y(\mathbf{x})$ is empty for $x_1 < -2$, otherwise it is explicitly given by $Y(\mathbf{x}) = [-4, x_1] \times [\max\{-4, -4-y_1\}, x_1] \times [(y_1+y_2)^2, 16]$.
In the implementation the linear constraint $x_1 \ge -2 + 10^{-6}$ can be added.

---

**Problem 16**

Taken from [28].

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^2} \quad & -x_1 \\
\text{s.t.} \quad & 3x_2^2 - y^5 \leq 0 \text{ for all } y \in Y(\mathbf{x}), \\
& 0 \leq x_1 \leq 1, \\
& 0 \leq x_2 \leq 1,
\end{aligned}$$

where

$$\begin{aligned}
Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } & -y^5 - 4x_1^2 - x_2^2 + 1 \leq 0, \\
& -2 \leq y \leq 0\}
\end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned}
t: \quad \mathbb{R}^2 \times [0,1] \quad & \rightarrow \quad \mathbb{R} \\
(\mathbf{x}, z) \quad & \mapsto \quad t(\mathbf{x}, z) = \begin{cases} z \cdot (-4x_1^2 - x_2^2 + 1)^{\frac{1}{5}} & \text{, if } -4x_1^2 - x_2^2 + 1 \leq 0, \\ \text{n. def.} & \text{, otherwise} \end{cases}
\end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The optimal value function $\varphi(\mathbf{x})$ is convex. The index set $Y(\mathbf{x})$ is empty for $-4x_1^2 - x_2^2 + 1 > 0$, otherwise it is explicitly given by $Y(\mathbf{x}) = [(-4x_1^2 - x_2^2 + 1)^{\frac{1}{5}}, 0]$. Thus, the feasible set is explicitly given by $M = \{\mathbf{x} \in [0,1]^2 \mid 4x_1^2 + x_2^2 < 1\} \cup \{(\frac{1}{2}, 0)\}$. The optimal solution is $(\frac{1}{2}, 0)$ and the optimal value is $-\frac{1}{2}$.
In the implementation the non linear constraint $-4x_1^2 - x_2^2 + 1 \leq 10^{-6}$ can be added.

---

**Problem 17**

Taken from [28].

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^2} \quad & -x_1 \\
\text{s.t.} \quad & -y \cdot x_2 \le 0 \text{ for all } y \in Y(\mathbf{x}), \\
& -1 \le x_1 \le 1, \\
& -1 \le x_2 \le 1,
\end{aligned}$$

where

$$\begin{aligned}
Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } \quad & x_1 - y^2 \le 0, \\
& -1 \le y \le 1\}
\end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned}
t: \quad \mathbb{R}^2 \times [0,1] \quad & \rightarrow \quad \mathbb{R} \\
(\mathbf{x}, z) \quad & \mapsto \quad t(\mathbf{x}, z) = \begin{cases} -z \cdot \sqrt{x_1} + (1-z) \cdot \sqrt{x_1} & \text{, if } x_1 \ge 0, \\ \text{n. def.} & \text{, otherwise} \end{cases}
\end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The optimal value function $\varphi(\mathbf{x})$ is convex. The index set $Y(\mathbf{x})$ is empty for $x_1 < 0$, otherwise it is explicitly given by $Y(\mathbf{x}) = [-\sqrt{x_1}, \sqrt{x_1}]$. In the implementation the linear constraint $x_1 \ge 10^{-6}$ can be added.

**Problem 18**

Taken from [5].

$$\begin{aligned} \min_{x\in\mathbb{R}} \quad & x^2 \\ \text{s.t.} \quad & \exp(x)\cdot y^2 - x^2\cdot y \le 0 \text{ for all } y\in Y(x), \\ & -1 \le x \le 1, \end{aligned}$$

where

$$\begin{aligned} Y(x) = \{y\in\mathbb{R} \text{ with } & y^2x^3 - x - \tfrac{1}{5} \le 0, \\ & 0 \le y \le 1\} \end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned} t: \quad \mathbb{R}\times[0,1] \quad & \to \quad \mathbb{R} \\ (x,z) \quad & \mapsto \quad t(x,z) = \begin{cases} z\sqrt{\frac{5x+1}{5x^3}} & \text{, if } 0 \le \frac{5x+1}{5x^3} \le 1, \\ z & \text{, if } 1 < \frac{5x+1}{5x^3}, \\ \text{n. def.} & \text{, otherwise} \end{cases} \end{aligned}$$

**Remarks:** The lower level problem $Q(x)$ is non-convex. The optimal value function $\varphi(x)$ is non-convex. The index set is explicitly given by:

$$Y(\mathbf{x}) = \begin{cases} [0, \frac{5x+1}{5x^3}] & \text{,if } 0 \le \frac{5x+1}{5x^3} \le 1 \\ [0.1] & \text{,if } 1 \le \frac{5x+1}{5x^3} \\ \emptyset & \text{,otherwise} \end{cases}$$

Thus, the feasible set is explicitly given by $M = \{x\in[-1,1] \mid \frac{5x+1}{5x^3} < 0\}$.
In the implementation the linear constraint can be added.

---

**Problem 19**

Taken from [5].

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^3} \quad & x_1^2 + x_2^2 + x_3^2 \\
\text{s.t.} \quad & x_1 + x_2\exp(x_3 y) + \exp(2y) - 2\sin(4y) \leq 0 \text{ for all } y \in Y(\mathbf{x}), \\
& -5 \leq x_1 \leq 5, \\
& -5 \leq x_2 \leq 5, \\
& -5 \leq x_3 \leq 5,
\end{aligned}$$

where

$$\begin{aligned}
Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } & 2y \leq x_2 + 1, \\
& 0 \leq y \leq 1 \ \}
\end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned}
t: \quad \mathbb{R}^3 \times [0,1] \quad & \rightarrow \quad \mathbb{R} \\
(\mathbf{x}, z) \quad & \mapsto \quad t(\mathbf{x}, z) = \begin{cases} (1-z)\cdot\frac{x_2+1}{2} & \text{, if } x_2 \geq -1, \\ \text{n. def.} & \text{, otherwise} \end{cases}
\end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The index set $Y(\mathbf{x})$ is empty for $x_2 < -1$, otherwise it is explicitly given by $Y(\mathbf{x}) = [0, \frac{x_2+1}{2}]$.
In the Implementation the linear constraint $x_2 \geq -1 + 10^{-6}$ can be added.

---

**Problem 20**

Taken from [5].

$$\begin{aligned}
\min_{x\in\mathbb{R}} \quad & x^2 \\
\text{s.t.} \quad & \tfrac{1}{2}y^3 - x^2 \leq 0 \text{ for all } y \in Y(x), \\
& -1 \leq x \leq 1,
\end{aligned}$$

where

$$\begin{aligned}
Y(x) = \{y \in \mathbb{R} \text{ with } & x^2 - y^2 \leq 0, \\
& 0 \leq y \leq 1\}
\end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned}
t: \quad \mathbb{R} \times [0,1] \quad & \rightarrow \quad \mathbb{R} \\
(x, z) \quad & \mapsto \quad t(x, z) = (1-z)\sqrt{x^2} + z
\end{aligned}$$

**Remarks:** The lower level problem $Q(x)$ is non-convex. The index set is explicitly given by $Y(x) = [\sqrt{x^2}, 1]$. Thus the feasible set is explicitly given by: $M = [-1, -\frac{\sqrt{2}}{2}] \cup [\frac{\sqrt{2}}{2}, 1]$. The optimal solutions are $\frac{\sqrt{2}}{2}$ and $-\frac{\sqrt{2}}{2}$ and the optimal value is $\frac{1}{2}$.

---

**Problem 21**

Taken from [5].

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^3} \quad & \exp(x_1)+\exp(x_2)+\exp(x_3) \\ \text{s.t.} \quad & \tfrac{1}{1+y^2} - x_1 - x_2\cdot y - x_3\cdot y^2 \le 0 \text{ for all } y \in Y(\mathbf{x}), \\ & -1 \le x_1 \le 1, \\ & -1 \le x_2 \le 1, \\ & -1 \le x_3 \le 1, \end{aligned}$$

where

$$Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } \begin{array}{l} -\tfrac{1}{2}y \le -x_2 - x_3, \\ 0 \le y \le 1 \end{array} \}$$

Transformation to a fixed index set:

$$\begin{aligned} t: \quad \mathbb{R}^3 \times [0,1] & \to \mathbb{R} \\ (\mathbf{x}, z) & \mapsto t(\mathbf{x}, z) = \begin{cases} z\cdot\max\{0, 2(x_2+x_3)\} + (1-z) & \text{, if } x_2+x_3 \le \tfrac{1}{2}, \\ \text{n. def.} & \text{, otherwise} \end{cases} \end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The optimal value function $\varphi(\mathbf{x})$ is non-convex. The index set $Y(\mathbf{x})$ is empty for $x_2 + x_3 > \frac{1}{2}$, otherwise it is explicitly given by: $Y(\mathbf{x}) = [\max\{0, 2\cdot(x_2+x_3)\}, 1]$. Thus, the feasible set is explicitly given by $M = \{\mathbf{x} \in [0,1]^3 \mid x_2 + x_3 > \frac{1}{2}\} \cup \{\mathbf{x} \in [0,1]^3 \mid x_2 + x_3 < 0, x_1 = 1\}$
$\cup\{\mathbf{x} \in [0,1]^3 \mid x_2 + x_3 \ge 0, \frac{1}{1+x_1+x_2} - x_1 - x_2\cdot(x_2+x_3) - x_3\cdot(x_1+x_2) \le 0\}$.
In the implementation the linear constraint $x_2 + x_3 \le \frac{1}{2} - 10^{-6}$ is added.

---

**Problem 22**

Taken from [5].

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^3} \quad & x_1^2 + x_2^2 + x_3^2 \\
\text{s.t.} \quad & x_1(y_1 + y_2^2 + 1) + x_2(y_1 y_2 - y_2^2) + x_3(y_1 y_2 + y_2^2 + y_2) + 12 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}), \\
& -1 \leq x_1 \leq 0, \\
& -1 \leq x_2 \leq 0, \\
& -1 \leq x_3 \leq 0,
\end{aligned}$$

where

$$\begin{aligned}
Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } & x_1^2 - y_1^2 \leq 0, \\
& 0 \leq y_1 \leq 1, \\
& 0 \leq y_2 \leq 1 \,\}
\end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned}
\mathbf{t}: \quad \mathbb{R}^3 \times [0,1]^2 & \rightarrow \quad \mathbb{R}^2 \\
(\mathbf{x}, \mathbf{z}) & \mapsto \quad \mathbf{t}(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} z_1 + (1 - z_1)\sqrt{x_2^2} \\ z_2 \end{pmatrix}
\end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The optimal value function $\varphi(\mathbf{x})$ is non-convex. The index set is explicitly given by $Y(\mathbf{x}) = [0, \sqrt{x_2^2}] \times [0, 1]$.

---

**Problem 23**

Taken from [5].

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^2} \quad & x_2^2 - 4x_2 \\
\text{s.t.} \quad & x_1 \cos(y) + x_2 \sin(y) - 1 \leq 0 \text{ for all } y \in Y(\mathbf{x}), \\
& 0 \leq x_1 \leq 2, \\
& 0 \leq x_2 \leq 2,
\end{aligned}$$

where

$$\begin{aligned}
Y(\mathbf{x}) = \{y \in \mathbb{R} \text{ with } & -y^2 - \tfrac{7}{4}x_2 + \tfrac{23}{4} \leq 0, \\
& 0 \leq y \leq \pi\}
\end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned}
t: \quad \mathbb{R}^2 \times [0,1] & \rightarrow \quad \mathbb{R} \\
(\mathbf{x}, z) & \mapsto \quad t(\mathbf{x}, z) = z \cdot \sqrt{\left(\tfrac{23}{4} - \tfrac{7}{4}x_2\right)} + (1 - z)\pi
\end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex.

---

**Problem 24**

Taken from [5].

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^6} \quad & -4x_1 - \tfrac{2}{3}(x_4 + x_6) \\
\text{s.t.} \quad & x_1 + x_2 \cdot y_1 + x_3 \cdot y_2 + x_4 \cdot y_1^2 + x_5 \cdot y_1 y_2 + x_6 \cdot y_2 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}), \\
& 0 \leq x_1 \leq 2, \\
& 0 \leq x_2 \leq 2, \\
& 0 \leq x_3 \leq 2, \\
& 0 \leq x_4 \leq 2, \\
& 0 \leq x_5 \leq 2, \\
& 0 \leq x_6 \leq 2,
\end{aligned}$$

where

$$\begin{aligned}
Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \; & x_1\cos(y_1) - x_2\sin(y_1) \leq 0, \\
& -1 \leq y_1 \leq 1, \\
& -1 \leq y_2 \leq 1 \; \}
\end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned}
\mathbf{t}: \quad \mathbb{R}^6 \times [0,1]^2 \quad & \rightarrow \quad \mathbb{R}^2 \\
(\mathbf{x}, \mathbf{z}) \quad & \mapsto \quad \mathbf{t}(\mathbf{x}, \mathbf{z}) = \begin{cases} \begin{pmatrix} (1 - z_1)\arctan(\frac{x_1}{x_2}) + z_1 \\ -1 + 2z_2 \end{pmatrix} & \text{, if} \arctan\left(\frac{x_1}{x_2}\right) \leq 1, x_2 \neq 0, \\ \begin{pmatrix} -1 + 2z_1 \\ -1 + 2z_2 \end{pmatrix} & \text{, if } x_1 = x_2 = 0, \\ \text{n. def.} & \text{, otherwise} \end{cases}
\end{aligned}$$

**Remarks:** The lower level problem $Q(\mathbf{x})$ is non-convex. The optimal value function $\varphi(\mathbf{x})$ is non-convex. The index set $Y(\mathbf{x})$ is empty for $\arctan\left(\frac{x_1}{x_2}\right) > 1, x_2 \neq 0$ or $x_2 = 0, x_1 \neq 0$.

# A.2. Design-Centering Problems

One example of generalized semi-infinite problems are design-centering problems. The description of these problems needs designs and a container. They give all the information needed to generate the problem. How a design-centering can be formulated as a GSIP is shown in chapter 3. It is also described how these problems can be initialized, given several designs `d1`, `d2`, …, `ds`, a container `c` and a desired distance `h` the problem is initialized by:
`p=MBProblemAsGSIP(c,h,d1,d2 ...,ds)`
If only the first design should be indescribed the problem can be initialized by:
`p=DCProblemAsGSIP(d1,c)`
In this chapter of the appendix we list all implemented designs and containers. At the end we list the design-centering problems which were considered in the literature.

## A.2.1. Designs

For every design we give the describing function in dependence of the parameters **x**. We also give a transformation in the sense of the one described in chapter 1. Finally we list possible constraints on the parameters. If they are not fulfilled, the design degenerates. For every design the name is written in Brackets. For every design the constructor is given as `Design`<name>. For example the circle is initialized by:
`d=DesignCircle`

**Design 1** (`Circle`)

The first two parameters describe the position and the third describes the radius.

$$D(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \ (y_1 - x_1)^2 + (y_2 - x_2)^2 - x_3^2 \leq 0 \ \}$$

$$Vol(\mathbf{x}) = \pi x_3^2$$

Transformation to a fixed set:

$$\begin{array}{rrcl} \mathbf{t}: & \mathbb{R}^3 \times \Big([-\pi,\pi] \times [0,1]\Big) & \rightarrow & \mathbb{R}^2 \\ & (\mathbf{x},\mathbf{z}) & \mapsto & t(\mathbf{x},\mathbf{z}) = \begin{pmatrix} x_1 + z_2 x_3 \cos(z_1) \\ x_2 + z_2 x_3 \sin(z_1) \end{pmatrix} \end{array}$$

Constraints on the parameters:

$$10^{-6} \leq x_3$$

---

**Design 2** (`SemiCircle`)

The first two paramaters descirbe the position, the third the radius and the last two parameters describe the normal vector of the halfspace.

$$
\begin{aligned}
D(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } & (y_1 - x_1)\tfrac{x_4}{\sqrt{x_4^2+x_5^2}} + (y_2 - x_2)\tfrac{x_5}{\sqrt{x_4^2+x_5^2}} \le 0 \\
& (y_1 - x_1)^2 + (y_2 - x_2)^2 - x_3^2 \le 0 \ \}
\end{aligned}
$$

$$
Vol(\mathbf{x}) = \tfrac{\pi}{2} x_3^2
$$

Transformation to a fixed set:

$$
\begin{aligned}
\mathbf{t}: \quad \mathbb{R}^5 \times ([0,1] \times [-\tfrac{1}{2}, \tfrac{1}{2}]) \quad & \to \quad \mathbb{R}^2 \\
(\mathbf{x}, \mathbf{z}) \quad & \mapsto \quad t(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} x_1 + z_1 x_3 \cos(\vartheta^*(\mathbf{x}) + \pi z_2) \\ x_2 + z_1 x_3 \sin(\vartheta^*(\mathbf{x}) + \pi z_2) \end{pmatrix}
\end{aligned}
$$

where $\vartheta^*(\mathbf{x}) = 2 \arctan\left( \frac{-x_4}{\sqrt{x_4^2+x_5^2} - x_5} \right)$

Constraints on the parameters:

$$
\begin{aligned}
-x_4^2 - x_5^2 + 1 &\le 0 \\
10^{-6} &\le x_3
\end{aligned}
$$

**Design 3** (`Boat`)

The first two parameters describe the postion, the third the radius of the two circles and the parameters $x_4, x_5$ describe the rotation.

$$\begin{aligned} D(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } & \left(y_1 - \left[x_1 + \tfrac{x_3 x_4}{\sqrt{x_4^2+x_5^2}}\right]\right)^2 + \left(y_2 - \left[x_2 + \tfrac{x_3 x_5}{\sqrt{x_4^2+x_5^2}}\right]\right)^2 - x_3^2 \leq 0 \\ & (y_1 - x_1)^2 + (y_2 - x_2)^2 - x_3^2 \leq 0 \,\} \end{aligned}$$

$$Vol(\mathbf{x}) = \left(\tfrac{\pi}{3} - \tfrac{\sqrt{3}}{2}\right) x_3^2$$

Transformation to a fixed set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^5 \times \Big([0,1] \times [-1,1]\Big) \quad & \to \quad \mathbb{R}^2 \\ (\mathbf{x}, \mathbf{z}) \quad & \mapsto \quad t(\mathbf{x}, \mathbf{z}) = \begin{cases} \begin{pmatrix} x_1 + z_1 x_3 \cos(\vartheta^*(x) + \frac{\pi}{3} + \frac{2\pi}{3} z_2) \\ x_2 + z_1 x_3 \sin(\vartheta^*(x) + \frac{\pi}{3} + \frac{2\pi}{3} z_2) \end{pmatrix}, & \text{if } z_2 \leq 0 \\ \begin{pmatrix} x_1 + x_3 \frac{x_4}{\sqrt{x_4^2+x_5^2}} + z_1 x_3 \cos(\vartheta^*(\mathbf{x}) - \frac{4\pi}{3} + \frac{2\pi}{3} z_2) \\ x_2 + x_3 \frac{x_5}{\sqrt{x_4^2+x_5^2}} + z_1 x_3 \cos(\vartheta^*(\mathbf{x}) - \frac{4\pi}{3} + \frac{2\pi}{3} z_2) \end{pmatrix}, & \text{if } z_2 > 0 \end{cases} \end{aligned}$$

where $\vartheta^*(\mathbf{x}) = 2 \arctan\left(\frac{-x_4}{\sqrt{x_4^2+x_5^2} - x_5}\right)$

Constraints on the parameters:

$$\begin{aligned} -x_4^2 - x_5^2 + 1 & \leq 0 \\ 10^{-6} & \leq x_3 \end{aligned}$$

---

**Design 4** (`EllipseWithRotMat`)

The design is implemented for two and three dimensions. Thats why the constructor needs the dimension as an input.The first two or three parameters are the position, the next parameters describe the length of the semi axes and the last parameters are the rotation. This is the description of the two dimensional design.

$$D(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } (y - \mathbf{x}_{pos})^T\, Rot(\mathbf{x})^T A(\mathbf{x}) Rot(\mathbf{x})\, (y - \mathbf{x}_{pos}) - 1 \leq 0\}$$

where:

$$\mathbf{x}_{pos} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad \text{or} \quad \mathbf{x}_{pos} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$A(\mathbf{x}) = \begin{pmatrix} \frac{1}{x_3^2} & 0 \\ 0 & \frac{1}{x_4^2} \end{pmatrix} \qquad \text{or} \quad A(\mathbf{x}) = \begin{pmatrix} \frac{1}{x_4^2} & 0 & 0 \\ 0 & \frac{1}{x_5^2} & 0 \\ 0 & 0 & \frac{1}{x_6^2} \end{pmatrix}$$

$$Rot(\mathbf{x}) = \begin{pmatrix} \cos(x_5) & -\sin(x_5) \\ \sin(x_5) & \cos(x_5) \end{pmatrix} \quad \text{or}$$

$$Rot(\mathbf{x}) = \begin{pmatrix} \cos(x_7) & -\sin(x_7) & 0 \\ \sin(x_7) & \cos(x_7) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(x_8) & 0 & -\sin(x_8) \\ 0 & 1 & 0 \\ \sin(x_8) & 0 & \cos(x_8) \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(x_9) & -\sin(x_9) \\ 0 & \sin(x_9) & \cos(x_9)] \end{pmatrix}$$

$$Vol(\mathbf{x}) = \pi \cdot x_3 x_4 \qquad \text{or} \qquad Vol(\mathbf{x}) = \frac{4}{3}\pi \cdot x_4 x_5 x_6$$

Transformation to a fixed set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^5 \times B^2 &\rightarrow \mathbb{R}^2 \\ (\mathbf{x}, \mathbf{z}) &\mapsto t(\mathbf{x}, \mathbf{z}) = \mathbf{x}_{pos} + Rot(\mathbf{x}) \cdot \begin{pmatrix} z_1 \cdot x_3 \\ z_2 \cdot x_4 \end{pmatrix} \end{aligned}$$

or

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^9 \times B^3 &\rightarrow \mathbb{R}^3 \\ (\mathbf{x}, \mathbf{z}) &\mapsto t(\mathbf{x}, \mathbf{z}) = \mathbf{x}_{pos} + Rot(\mathbf{x}) \cdot \begin{pmatrix} z_1 \cdot x_4 \\ z_2 \cdot x_5 \\ z_3 \cdot x_6 \end{pmatrix} \end{aligned}$$

where $B^2$ and $B^3$ denote the two and three dimensional unit hyperballs.

Constraints on the parameters:

Every semi-axes has to be greater than $10^{-6}$.

---

**Design 5** (`NonRotEllipse`)

The parameters $x_1, x_2$ give the position of the center of the ellipse. The other two parameters are the lengths of the semi-axes.

$$D(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \tfrac{(y_1-x_1)^2}{x_3^2} + \tfrac{(y_2-x_2)^2}{x_4^2} - 1 \leq 0 \;\}$$

$$Vol(\mathbf{x}) = \pi x_3 x_4$$

Transformation to a fixed set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^4 \times \Big([0,1] \times [-\pi,\pi]\Big) \quad &\rightarrow \quad \mathbb{R}^2 \\ (\mathbf{x}, \mathbf{z}) \quad &\mapsto \quad t(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} x_1 + z_2 x_3 \cos(z_1) \\ x_2 + z_2 x_4 \sin(z_1) \end{pmatrix} \end{aligned}$$

Constraints on the parameters:

$$\begin{aligned} 10^{-6} &\leq x_3 \\ 10^{-6} &\leq x_4 \end{aligned}$$

---

## n-dimensional designs

All designs listed above are implemented for one dimension. The next four example are implmented for an arbitrary dimension.

**Design 6** (`HyperBall`)

Ball in $n$ dimensions. The constructor needs the dimension as input. The first $n$ parameters are the position and the last parameter is the radius.

$$D(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n \text{ with } \textstyle\sum_{i=1}^n (y_i - x_i)^2 - x_{n+1}^2 \leq 0\ \}$$

$$Vol(\mathbf{x}) = V \cdot (x_{n+1})^n$$

where $V$ ist the volume of the unit ball.
Transformation to a fixed set:

$$\begin{aligned} \mathbf{t}: \quad & \mathbb{R}^{n+1} \times \{\mathbf{z} \in \mathbb{R}^n \mid \textstyle\sum_{i=1}^n z_i^2 \leq 1\} && \to \quad \mathbb{R}^n \\ & (\mathbf{x}, \mathbf{z}) && \mapsto \quad t(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + x_{n+1} \cdot \mathbf{z} \end{aligned}$$

Constraints on the parameters:

$$10^{-6} \leq x_{n+1}$$

---

**Design 7** (`HyperCuboid`)

Cuboid in $n$ dimensions. The constructor needs the dimension as input. The first $n$ parameters are the position of the lower edge. The next $n$ parameters are the position of the upper edge.

$$D(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n \text{ with } \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \leq \mathbf{y} \leq \begin{pmatrix} x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{2n} \end{pmatrix} \}$$

$$Vol(\mathbf{x}) = (x_{n+1} - x_1) \cdot (x_{n+2} - x_2) \cdot \dots \cdot (x_{2n} - x_n)$$

Transformation to a fixed set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^{2n} \times [0,1]^n \quad &\rightarrow \quad \mathbb{R}^n \\ (\mathbf{x}, \mathbf{z}) \quad &\mapsto \quad t(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} x_1 + z_1 \cdot (x_{n+1} - x_1) \\ x_2 + z_2 \cdot (x_{n+2} - x_2) \\ \vdots \\ x_n + z_n \cdot (x_{2n} - x_n) \end{pmatrix} \end{aligned}$$

Constraints on the parameters:

$$\begin{pmatrix} 1 & 0 & \dots & 0 & -1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & -1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & -1 \end{pmatrix} \cdot \mathbf{x} \leq \begin{pmatrix} -10^{-6} \\ -10^{-6} \\ \vdots \\ -10^{-6} \end{pmatrix}$$

---

**Design 8** (`HyperEllipse`)

The constructor needs the dimension $n$ as an input. The Parameters $x_1$ up to $x_{\frac{(n+1)n}{2}}$ describe the matrix $A(\mathbf{x})$ (see below). The last $n$ parameters describe the position in the space.

$$D(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n \text{ with } \left(\mathbf{y} - \begin{pmatrix} x_{\frac{(n+1)n}{2}+1} \\ \dots \\ x_{\frac{(n+1)n}{2}+n} \end{pmatrix}\right)^T (A(\mathbf{x})A(\mathbf{x})^T)^{-1} \left(\mathbf{y} - \begin{pmatrix} x_{\frac{(n+1)n}{2}+1} \\ \dots \\ x_{\frac{(n+1)n}{2}+n} \end{pmatrix}\right) - 1 \leq 0\}$$

where

$$A(\mathbf{x}) = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ 0 & x_{n+1} & \dots & x_{2n-1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & x_{\frac{(n+1)n}{2}} \end{pmatrix}$$

$$Vol(\mathbf{x}) = V \cdot (x_1 \cdot x_{n+1} \cdot x_{2n} \cdot \dots \cdot x_{\frac{(n+1)n}{2}})$$

where V ist the volume of the unit ball.

Transformation to a fixed set:

$$\begin{array}{rccl} \mathbf{t}: & \mathbb{R}^{(n+2)n} \times B^n & \to & \mathbb{R}^n \\ & (\mathbf{x}, \mathbf{z}) & \mapsto & t(\mathbf{x}, \mathbf{z}) = \cdot A(\mathbf{x}) \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix} + \begin{pmatrix} x_{\frac{(n+1)n}{2}+1} \\ \dots \\ x_{\frac{(n+1)n}{2}+n} \end{pmatrix} \end{array}$$

where $B^n$ denotes the $n$-dimensional unit hyper ball.

Constraints on parameters:

For a Parmeter $x_i$ on the diagonal of $A(\mathbf{x})$ we have:

$10^{-3} \leq x_i$

For a Parameter $x_i$ not on the diagonal of $A(\mathbf{x})$ but in the matrix:

$-1 \leq x_i \leq 1$

For two different elements $x_i$ and $x_j$ on the diagonal of $A(\mathbf{x})$ we have:

$x_i \leq k \cdot x_j$

The Design has two function `getK()` and `setK(k)`. The default value for k is 5.

**Design 9** (`NonRotHyperEllipse`)

The design is a $n$-dimeinsional non rotatable ellipse. The constructor needs the dimension as an input. The first $n$ parameters are the postion of the centre. The next $n$-parameters are the lengths of the semi-axes.

$$D(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n \text{ with } \textstyle\sum_{i=1}^{n} \frac{(y_i - x_i)^2}{x_{n+i}^2} - 1 \leq 0\ \}$$

$$Vol(\mathbf{x}) = V \cdot \textstyle\prod_{i=1}^{n} x_{n+i}$$

where $V$ is the volume of the n-dimensional unit Ball
Transformation to a fixed set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^{2n} \times \times B^n \quad &\rightarrow \quad \mathbb{R}^n \\ (\mathbf{x}, \mathbf{z}) \quad &\mapsto \quad t(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} z_1 \cdot x_{n+1} \\ z_2 \cdot x_{n+2} \\ \vdots \\ z_n \cdot x_{2n} \end{pmatrix} + \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \end{aligned}$$

where $B^n$ denotes the $n$-dimensional unit hyper ball.

Constraints on the parameters:

$$\begin{pmatrix} 10^{-6} \\ 10^{-6} \\ \vdots \\ 10^{-6} \end{pmatrix} \leq \begin{pmatrix} x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{2n} \end{pmatrix}$$

### A.2.2. Containers

The second ingredient for a design centering problem is the container. We list containers we have found in the literature. For many purposes it is good to know some bounds on the container. We give for every container such a bounding box.

**Container 1** (`C1`)

$$\begin{aligned} C = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } & -(y_1+0.5)^2-(y_2+0.5)^2+0.04 \leq 0 \\ & -(y_1-0.5)^2-(y_2+0.5)^2+0.04 \leq 0 \\ & -y_1^2+y_2 \leq 0 \\ & y_1-y_2^2-1 \leq 0 \\ & \tfrac{y_1^2}{4}+y_2^2-1 \leq 0 \\ & -2y_1-y_2^2-1 \leq 0 \\ & -|y_1|-y_2-0.5 \leq 0\} \end{aligned}$$

$B_C = [-1.5, 1.5] \times [-1, 1]$

---

**Container 2** (`Triangle`)

$$\begin{aligned} C = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } & \tfrac{1}{4}y_1+y_2-\tfrac{3}{4} \leq 0 \\ & -y_1-1 \leq 0 \\ & -y_2-1 \leq 0\} \end{aligned}$$

$B_C = [-1, 7] \times [-1, 1]$

---

**Container 3** (`C3`)

$$\begin{aligned} C = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } & \left(y_1-\tfrac{1}{2}\right)^2+y_2^2-\tfrac{9}{4} \leq 0 \\ & 1-y_1^2-y_2^2 \leq 0 \\ & \tfrac{1}{4}-(y_1-\tfrac{3}{2})^2-y_2^2 \leq 0 \\ & -y_2 \leq 0\} \end{aligned}$$

$B_C = [-1, 2] \times [0, 1.5]$

---

**Container 4** (`ConcTriangle`)

$$\begin{aligned} C = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } & \tfrac{1}{4}y_1+y_2-\tfrac{3}{4} \leq 0 \\ & -y_1-y_2^2 \leq 0 \\ & -y_2-1 \leq 0\} \end{aligned}$$

$B_C = [-1, 7] \times [-1, 1]$

---

**Container 5** (`ConcTriangleWithGap`)

$$
\begin{aligned}
C = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \; -(y_1-2)^2 - (y_2+\tfrac{1}{2})^2 + \tfrac{1}{16} &\leq 0 \\
\tfrac{1}{4}y_1 + y_2 - \tfrac{3}{4} &\leq 0 \\
-y_1 - y_2^2 &\leq 0 \\
-y_2 - 1 &\leq 0\}
\end{aligned}
$$

$B_C = [-1, 7] \times [-1, 1]$

---

**Container 6** (`C6` )

$$
\begin{aligned}
C = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \; (y_1-\tfrac{1}{2})^2 - y_2 - 1 &\leq 0 \\
y_1^2 + y_2 &\leq 0\}
\end{aligned}
$$

$B_C = [-0.5, 1] \times [-1, 0]$

---

**Container 7** (`C7`)

$$
\begin{aligned}
C = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \; \tfrac{3}{10}\sin(\pi y_1) - y_2 &\leq 0 \\
y_1^2 + \tfrac{3}{10}y_2^2 - 1 &\leq 0\}
\end{aligned}
$$

$B_C = [-1, 1] \times [-0.5, 2]$

---

## n-dimensional container

**Container 8** (`Pyramid`)
The constructor needs the dimension as input. The Pyramid is growing with the dimension to avoid to rapid decrease of the volume.

$$\begin{aligned} C = \{\mathbf{y} \in \mathbb{R}^n \text{ with } & \textstyle\sum_1^n y_i - n \leq 0 \\ & -y_i \leq 0 \text{ for } i = 1, \ldots, n\} \end{aligned}$$

$B_C = [0, n]^n$

---

**Container 9** (`containerHyperCube`)
In the implementation one can deside whether the first constraint should be considered or not.
The first input of the constructor is the dimension the second a boolean.

$$\begin{aligned} C = \{\mathbf{y} \in \mathbb{R}^n \text{ with } & y_n - \tfrac{1}{2} - \tfrac{1}{2} \textstyle\sum_{i=1}^{n-1} (y_i - \tfrac{1}{2})^2 \leq 0 \\ & -y_i \leq 0 \text{ for } i = 1, \ldots, n \\ & -y_i - 1 \leq 0 \text{ for } i = 1, \ldots, n\} \end{aligned}$$

$B_C = [0, 1]^n$

---

**Container 10** (`ParamHyperCube`)
The first differences to the container Hyper cube is the choice of the edge length $l$ and the parameter $p$. The parameter $p$ gives the length of the distance of the lowest point of the parabola and the the lower side. Also it is not possible to turn off the first constraint in the implementation.
The first input of the constructor is the dimension, the second $l$ and the third $p$.

$$\begin{aligned} C = \{\mathbf{y} \in \mathbb{R}^n \text{ with } & y_n - 4 \cdot (l - p)/l^2 \cdot (\textstyle\sum_{i=1}^{n-1} (y_i - \tfrac{l}{2})^2) - p \leq 0 \\ & -y_i \leq 0 \text{ for } i = 1, \ldots, n \\ & -y_i - l \leq 0 \text{ for } i = n+1, \ldots, 2n\} \end{aligned}$$

$B_C = [0, l]^n$

---

### A.2.3. Problems from Literature

**DCProblem 1**

Taken from [16].

Inscribing of Circle (Design 1) into C1 (Container 1).

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^3} \quad & \pi x_3^2 \\
\text{s.t.} \quad -(y_1+0.5)^2-(y_2+0.5)^2+0.04 &\le 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
-(y_1-0.5)^2-(y_2+0.5)^2+0.04 &\le 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
-y_1^2+y_2 &\le 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
y_1-y_2^2-1 &\le 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
\tfrac{y_1^2}{4}+y_2^2-1 &\le 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
-2y_1-y_2^2-1 &\le 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
-|y_1|-y_2-0.5 &\le 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
10^{-6} &\le x_3
\end{aligned}$$

where:

$$Y(\mathbf{x}) = \{\mathbf{y}\in\mathbb{R}^2 \text{ with } (y_1-x_1)^2+(y_2-x_2)^2-x_3^2\le 0\ \}$$

Transformation to a fixed index set:

$$\begin{aligned}
\mathbf{t}:\quad \mathbb{R}^3\times\Big([-\pi,\pi]\times[0,1]\Big) &\to \mathbb{R}^2 \\
(\mathbf{x},\mathbf{z}) &\mapsto \mathbf{t}(\mathbf{x},\mathbf{z}) = \begin{pmatrix} x_1+z_2x_3\cos(z_1) \\ x_2+z_2x_3\sin(z_1)\end{pmatrix}
\end{aligned}$$

**Remarks:**

- $Q_1(\mathbf{x})$, $Q_2(\mathbf{x})$, $Q_4(\mathbf{x})$, $Q_5(\mathbf{x})$, $Q_6(\mathbf{x})$, $Q_7(\mathbf{x})$ : convex
- $Q_3(\mathbf{x})$ : non convex
- $\varphi_3$ : convex
- $\varphi_1, \varphi_2, \varphi_4, ..., \varphi_7$ : non convex

**DCProblem 2**

Taken from [1].

Inscribing of Circle (Design 1) into C3 (Container 3).

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^3} \quad & \pi x_3^2 \\
\text{s.t.} \quad \left(y_1 - \tfrac{1}{2}\right)^2 + y_2^2 - \tfrac{9}{4} &\le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
1 - y_1^2 - y_2^2 &\le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
\tfrac{1}{4} - (y_1 - \tfrac{3}{2})^2 - y_2^2 &\le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
-y_2 &\le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
10^{-6} &\le x_3
\end{aligned}$$

where:

$$Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \ (y_1 - x_1)^2 + (y_2 - x_2)^2 - x_3^2 \le 0 \ \}$$

Transformation to a fixed index set:

$$\begin{aligned}
\mathbf{t}: \quad \mathbb{R}^3 \times \Big([-\pi, \pi] \times [0,1]\Big) \quad &\to \quad \mathbb{R}^2 \\
(\mathbf{x}, \mathbf{z}) \quad &\mapsto \quad \mathbf{t}(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} x_1 + z_2 x_3 \cos(z_1) \\ x_2 + z_2 x_3 \sin(z_1) \end{pmatrix}
\end{aligned}$$

**Remarks:**

- $Q_1(\mathbf{x})$, $Q_2(\mathbf{x})$, $Q_4(\mathbf{x})$: convex
- $Q_3$: non convex
- $\varphi_3, \varphi_4$: convex
- $\varphi_1, \varphi_2$ : non convex

---

**DCProblem 3**

Taken from [22].

Inscribing of HyperEllipse (2 dimensionalDesign 8) into ConcTriangle (Container 4).

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^0} \quad & V \cdot x_1 x_3 \\
\text{s.t.} \quad & \tfrac{1}{4}y_1 + y_2 - \tfrac{3}{4} \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
& -y_1 - y_2^2 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
& -y_2 - 1 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
& x_1 - 5x_2 \leq 0 \\
& -5x_1 + x_2 \leq 0 \\
& 10^{-3} \leq x_1 \\
& -1 \leq x_2 \leq 1 \\
& 10^{-3} \leq x_3
\end{aligned}$$

where: V ist the volume of the unit ball,

$$Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \left(\mathbf{y} - \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}\right)^T (A(\mathbf{x})A(\mathbf{x})^T)^{-1} \left(\mathbf{y} - \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}\right) - 1 \leq 0\}$$

and

$$A(\mathbf{x}) = \begin{pmatrix} x_1 & x_2 \\ 0 & x_3 \end{pmatrix}$$

Transformation to a fixed index set:

$$\begin{aligned}
\mathbf{t}: \quad \mathbb{R}^5 \times B^2 & \rightarrow \mathbb{R}^2 \\
(\mathbf{x}, \mathbf{z}) & \mapsto \mathbf{t}(\mathbf{x}, \mathbf{z}) = A(\mathbf{x}) \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}
\end{aligned}$$

where $B^2$ denotes the unit circle.

**Remarks:**

- $Q_1$, $Q_2$, $Q_3$ : convex
- $\varphi_1$: non convex
- $\varphi_2, \varphi_3$ :convex

---

**DCProblem 4**

Taken from [22].

Inscribing of Circle (Design 1) into ConcTriangle (Container 4).

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^3} \quad & \pi x_3^2 \\ \text{s.t.} \quad & \tfrac{1}{4}y_1 + y_2 - \tfrac{3}{4} \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & -y_1 - y_2^2 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & -y_2 - 1 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & 10^{-6} \leq x_3 \end{aligned}$$

where:

$$Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } (y_1 - x_1)^2 + (y_2 - x_2)^2 - x_3^2 \leq 0 \}$$

Transformation to a fixed index set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^3 \times \Big([-\pi,\pi] \times [0,1]\Big) \quad & \rightarrow \quad \mathbb{R}^2 \\ (\mathbf{x},\mathbf{z}) \quad & \mapsto \quad \mathbf{t}(\mathbf{x},\mathbf{z}) = \begin{pmatrix} x_1 + z_2 x_3 \cos(z_1) \\ x_2 + z_2 x_3 \sin(z_1) \end{pmatrix} \end{aligned}$$

**Remarks:**

- $Q_1$, $Q_2$, $Q_3$ : convex
- $\varphi_1$, : non convex
- $\varphi_2, \varphi_3$ : convex

**DCProblem 5**

Taken from [22].

Inscribing of NonRotEllipse (Design 5) into ConcTriangle (Container 4).

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^4} \quad & \pi x_3 x_4 \\ \text{s.t.} \quad & \tfrac{1}{4}y_1 + y_2 - \tfrac{3}{4} \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & -y_1 - y_2^2 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & -y_2 - 1 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & 10^{-6} \leq x_3 \\ & 10^{-6} \leq x_4 \end{aligned}$$

where:

$$Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \tfrac{(y_1-x_1)^2}{x_3^2} + \tfrac{(y_2-x_2)^2}{x_4^2} - 1 \leq 0\ \}$$

Transformation to a fixed index set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^4 \times \Big([0,1] \times [-\pi,\pi]\Big) \quad & \to \quad \mathbb{R}^2 \\ (\mathbf{x},\mathbf{z}) \quad & \mapsto \quad \mathbf{t}(\mathbf{x},\mathbf{z}) = \begin{pmatrix} x_1 + z_2 x_3 \cos(z_1) \\ x_2 + z_2 x_4 \sin(z_1) \end{pmatrix} \end{aligned}$$

**Remarks:**

- $Q_1$, $Q_2$, $Q_3$ : convex

**DCProblem 6**

Taken from [22].

Inscribing of HyperCuboid (2 dimensional, Design 7) into ConcTriangle (Container 4).

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^0} \quad & (x_3 - x_1)(x_4 - x_2) \\
\text{s.t.} \quad & \tfrac{1}{4}y_1 + y_2 - \tfrac{3}{4} \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
& -y_1 - y_2^2 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
& -y_2 - 1 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
& x_1 - x_3 \leq \, -10^{-6} \\
& x_2 - x_4 \leq \, -10^{-6}
\end{aligned}$$

where:

$$Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \mathbf{y} \leq \begin{pmatrix} x_3 \\ x_4 \end{pmatrix} \}$$

Transformation to a fixed index set:

$$\begin{aligned}
\mathbf{t}: \quad \mathbb{R}^4 \times [0,1]^2 \quad & \rightarrow \quad \mathbb{R}^2 \\
(\mathbf{x}, \mathbf{z}) \quad & \mapsto \quad \mathbf{t}(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} x_1 + z_1 \cdot (x_3 - x_1) \\ x_2 + z_2 \cdot (x_4 - x_2) \end{pmatrix}
\end{aligned}$$

**Remarks:**

- $Q_1$, $Q_2$, $Q_3$ : convex
- $\varphi_1$: non convex
- $\varphi_2, \varphi_3$ : convex

---

**DCProblem 7**

Taken from [14].

Inscribing of HyperCuboid (2 dimensional, Design 7) into ConcTriangleWithGap (Container 5).

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^0} \quad & (x_3-x_1)(x_4-x_2) \\ \text{s.t.} \quad -(y_1-2)^2-(y_2+\tfrac{1}{2})^2+\tfrac{1}{16} &\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\ \tfrac{1}{4}y_1+y_2-\tfrac{3}{4} &\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\ -y_1-y_2^2 &\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\ -y_2-1 &\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\ x_1-x_3 &\leq -10^{-6} \\ x_2-x_4 &\leq -10^{-6} \end{aligned}$$

where:

$$Y(\mathbf{x}) = \{\mathbf{y}\in\mathbb{R}^2 \text{ with } \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \mathbf{y} \leq \begin{pmatrix} x_3 \\ x_4 \end{pmatrix}\}$$

Transformation to a fixed index set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^4\times[0,1]^2 &\rightarrow \mathbb{R}^2 \\ (\mathbf{x},\mathbf{z}) &\mapsto \mathbf{t}(\mathbf{x},\mathbf{z}) = \begin{pmatrix} x_1+z_1\cdot(x_3-x_1) \\ x_2+z_2\cdot(x_4-x_2) \end{pmatrix} \end{aligned}$$

**Remarks:**

- $Q_1$, $Q_2$, $Q_3$ : convex
- $\varphi_2, \varphi_2$ : convex
- $\varphi_1, \varphi_4$ : non convex

**DCProblem 8**

Taken from [14].

Inscribing of SemiCircle (Design 2) into ConcTriangle (Container 4).

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^5} \quad & \tfrac{\pi}{2}x_3^2 \\
\text{s.t.} \quad & \tfrac{1}{4}y_1 + y_2 - \tfrac{3}{4} \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
& -y_1 - y_2^2 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
& -y_2 - 1 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\
& -x_4^2 - x_5^2 + 1 \leq 0 \\
& 10^{-6} \leq x_3
\end{aligned}$$

where:

$$Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } (y_1 - x_1)\tfrac{x_4}{\sqrt{x_4^2+x_5^2}} + (y_2 - x_2)\tfrac{x_5}{\sqrt{x_4^2+x_5^2}} \leq 0 \quad (y_1 - x_1)^2 + (y_2 - x_2)^2 - x_3^2 \leq 0 \ \}$$

Transformation to a fixed index set:

$$\begin{aligned}
\mathbf{t}: \quad \mathbb{R}^5 \times ([0,1] \times [-\tfrac{1}{2}, \tfrac{1}{2}]) \quad & \to \quad \mathbb{R}^2 \\
(\mathbf{x}, \mathbf{z}) \quad & \mapsto \quad \mathbf{t}(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} x_1 + z_1 x_3 \cos(\vartheta^*(\mathbf{x}) + \pi z_2) \\ x_2 + z_1 x_3 \sin(\vartheta^*(\mathbf{x}) + \pi z_2) \end{pmatrix}
\end{aligned}$$

where $\vartheta^*(\mathbf{x}) = 2\arctan\left(\frac{-x_4}{\sqrt{x_4^2+x_5^2}-x_5}\right)$

**Remarks:**

- $Q_1$, $Q_2$, $Q_3$ : convex

---

**DCProblem 9**

Taken from [14].

Inscribing of Boat (Design 3) into ConcTriangle (Container 4).

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^5} \quad & \left(\tfrac{\pi}{3}-\tfrac{\sqrt{3}}{2}\right)x_3^2 \\ \text{s.t.} \quad & \tfrac{1}{4}y_1+y_2-\tfrac{3}{4}\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\ & -y_1-y_2^2\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\ & -y_2-1\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\ & -x_4^2-x_5^2+1\leq 0 \\ & 10^{-6}\leq x_3 \end{aligned}$$

where:

$$\begin{aligned} Y(\mathbf{x}) = \{\mathbf{y}\in\mathbb{R}^2 \text{ with } & \left(y_1-\left[x_1+\tfrac{x_3x_4}{\sqrt{x_4^2+x_5^2}}\right]\right)^2+\left(y_2-\left[x_2+\tfrac{x_3x_5}{\sqrt{x_4^2+x_5^2}}\right]\right)^2-x_3^2\leq 0 \\ & (y_1-x_1)^2+(y_2-x_2)^2-x_3^2\leq 0\ \} \end{aligned}$$

Transformation to a fixed index set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^5\times\Big([0,1]\times[-1,1]\Big) \quad & \to \quad \mathbb{R}^2 \\ (\mathbf{x},\mathbf{z}) \quad & \mapsto \quad \mathbf{t}(\mathbf{x},\mathbf{z}) = \begin{cases} \begin{pmatrix} x_1+z_1x_3\cos(\vartheta^*(x)+\frac{\pi}{3}+\frac{2\pi}{3}z_2) \\ x_2+z_1x_3\sin(\vartheta^*(x)+\frac{\pi}{3}+\frac{2\pi}{3}z_2) \end{pmatrix}, & \text{if } z_2\leq 0 \\ \begin{pmatrix} x_1+x_3\frac{x_4}{\sqrt{x_4^2+x_5^2}}+z_1x_3\cos(\vartheta^*(\mathbf{x})-\frac{4\pi}{3}+\frac{2\pi}{3}z_2) \\ x_2+x_3\frac{x_5}{\sqrt{x_4^2+x_5^2}}+z_1x_3\cos(\vartheta^*(\mathbf{x})-\frac{4\pi}{3}+\frac{2\pi}{3}z_2) \end{pmatrix}, & \text{if } z_2>0 \end{cases} \end{aligned}$$

where $\vartheta^*(\mathbf{x}) = 2\arctan\left(\frac{-x_4}{\sqrt{x_4^2+x_5^2}-x_5}\right)$

**Remarks:**

- $Q_1$, $Q_2$, $Q_3$ : convex

**DCProblem 10**

Taken from [14].

Inscribing of NonRotEllipse (Design 5) into Triangle (Container 2).

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^4} \quad & \pi x_3 x_4 \\ \text{s.t.} \quad & \tfrac{1}{4}y_1 + y_2 - \tfrac{3}{4} \le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & -y_1 - 1 \le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & -y_2 - 1 \le 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & 10^{-6} \le x_3 \\ & 10^{-6} \le x_4 \end{aligned}$$

where:

$$Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \tfrac{(y_1-x_1)^2}{x_3^2} + \tfrac{(y_2-x_2)^2}{x_4^2} - 1 \le 0 \}$$

Transformation to a fixed index set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^4 \times \Big([0,1] \times [-\pi,\pi]\Big) \quad & \to \quad \mathbb{R}^2 \\ (\mathbf{x},\mathbf{z}) \quad & \mapsto \quad \mathbf{t}(\mathbf{x},\mathbf{z}) = \begin{pmatrix} x_1 + z_2 x_3 \cos(z_1) \\ x_2 + z_2 x_4 \sin(z_1) \end{pmatrix} \end{aligned}$$

**Remarks:**

- $Q_1$, $Q_2$, $Q_3$ : konvex

### DCProblem 11

Taken from [14].

Inscribing of NonRotEllipse (Design 5) into ConcTriangleWithGap (Container 5).

$$
\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^4} \quad & \pi x_3 x_4 \\
\text{s.t.} \quad -(y_1-2)^2-(y_2+\tfrac{1}{2})^2+\tfrac{1}{16} &\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
\tfrac{1}{4}y_1+y_2-\tfrac{3}{4} &\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
-y_1-y_2^2 &\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
-y_2-1 &\leq 0 \text{ for all } \mathbf{y}\in Y(\mathbf{x}) \\
10^{-6} &\leq x_3 \\
10^{-6} &\leq x_4
\end{aligned}
$$

where:

$$Y(\mathbf{x}) = \{\mathbf{y}\in\mathbb{R}^2 \text{ with } \tfrac{(y_1-x_1)^2}{x_3^2}+\tfrac{(y_2-x_2)^2}{x_4^2}-1\leq 0\ \}$$

Transformation to a fixed index set:

$$
\begin{aligned}
\mathbf{t}:\quad \mathbb{R}^4\times\Big([0,1]\times[-\pi,\pi]\Big) \quad &\to \quad \mathbb{R}^2 \\
(\mathbf{x},\mathbf{z}) \quad &\mapsto \quad \mathbf{t}(\mathbf{x},\mathbf{z}) = \begin{pmatrix} x_1+z_2x_3\cos(z_1) \\ x_2+z_2x_4\sin(z_1) \end{pmatrix}
\end{aligned}
$$

**Remarks:**

- $Q_1$, $Q_2$, $Q_3$ : convex

---

**DCProblem 12**

Taken from [14].

Inscribing of HyperCuboid (2 dimensional, Design 7) into Triangle (Container 2).

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^0} \quad & (x_3 - x_1)(x_4 - x_2) \\ \text{s.t.} \quad & \tfrac{1}{4}y_1 + y_2 - \tfrac{3}{4} \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & -y_1 - 1 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & -y_2 - 1 \leq 0 \text{ for all } \mathbf{y} \in Y(\mathbf{x}) \\ & x_1 - x_3 \leq \; - 10^{-6} \\ & x_2 - x_4 \leq \; - 10^{-6} \end{aligned}$$

where:

$$Y(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^2 \text{ with } \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \mathbf{y} \leq \begin{pmatrix} x_3 \\ x_4 \end{pmatrix} \}$$

Transformation to a fixed index set:

$$\begin{aligned} \mathbf{t}: \quad \mathbb{R}^4 \times [0,1]^2 \quad & \rightarrow \quad \mathbb{R}^2 \\ (\mathbf{x},\mathbf{z}) \quad & \mapsto \quad \mathbf{t}(\mathbf{x},\mathbf{z}) = \begin{pmatrix} x_1 + z_1 \cdot (x_3 - x_1) \\ x_2 + z_2 \cdot (x_4 - x_2) \end{pmatrix} \end{aligned}$$

**Remarks:**

- $Q_1$, $Q_2$, $Q_3$ : convex
- $\varphi_1, ..., \varphi_3$ : convex

# B. List of Properties and Methods

In the Chapters 2-4 we only list the most important functions. They are enough to solve the problem. For some algorithms it may be more interesting to distinguish between different types of constraints. Also functions that return basic properties of the problems are helpful. We will list in this Appendix all methods which are implemented for the classes: `Problem`, `Design`, `Container` and UcSet. Also for the implementation it important to know the methodology of the functions. Moreover the properties and their meaning is needed. Thats why we begin every section with the properties of the class and then list the methods.

## B.1. GSIP-Problem

Super class of all implemented problems. We give for every property a short description and then the size.

| name of property | description | size |
|---|---|---|
| `nrOfUppLevVars` | Saves the number of the upper level vars $m$. | $1 \times 1$ |
| `boundsOnUppLevVars` | Matrix where the first row contains the lower bounds and the second row contains the upper bounds of the parameters (also `inf`,`-inf` possible). | $2 \times m.$ |
| `initValsOfUppLevVars` | Saves initial values for a starting point in a optimization algorithm. | $1 \times m$ |
| `nrOfNonlinIneqsUppLev` | The number of non linear inequality constraints $\lvert I^{UL} \rvert$ that are given in the function `evalUppLevNonlinIneqConstrs`. | $1 \times 1$ |
| `nrOfNonlinEqsUppLev` | The number of non linear equality constraints $\lvert J^{UL} \rvert$ that are given in the function `evalUppLevNonlinEqConstrs`. | $1 \times 1$ |
| `linIneqsOnUppLevVars` | `[A,b]` where `A`$x \leq$ `b` corresponds to $\mathbf{A}^{UL}\mathbf{x} \leq \mathbf{b}^{UL}$. If no such constraint is needed you can just give an empty matrix `[]`. | `size(`$\mathbf{A}^{UL}$`,1)` $\times (m+1)$ |

| name of property | description | size |
|---|---|---|
| `linEqsOnUppLevVars` | `[C,d]` where $\texttt{C}x = \texttt{d}$ corresponds to $\mathbf{C}^{UL}x = \mathbf{d}^{UL}$. If no such constraint is needed you can just give an empty matrix `[]`. | $\texttt{size}(\mathbf{C}^{UL}\texttt{,1}) \times (m+1)$ |
| `nrOfSemiInfConstrs` | Saves the number of semi-infinite constraints $I^{SI}$. | $1 \times 1$ |
| `nrOfIndexset` | Contains the number of different index sets. | $1 \times 1$ |
| `con2LowLev` | A vector that containes in the i-th entry the index of the Indexset that corresponds to the i-th semi-infinite constraint. | $1 \times \|I^{SI}\|$ |
| `nrOfLowLevVars` | A vector that contains in the j-th entry the number of lower level variables for the j-th indexset $n_k$. | $1 \times$ `nrOfIndexset` |
| `nrOfLowLevNonlinConstrs` | A vector that containes in the j-th entry the number of constraints $\|J_k\|$ of the j-th indexset(only the number of non linear constraints). | $1 \times$ `nrOfIndexset` |

Table B.1.: properties of a native GSIP.

In the following we give a list of the functions implemented. If one of these functions is not implemented for a new example, it is replaced by a standard function. In brackets the size of the inputs and outputs is given. We begin with functions that belong to the upper level, then the lower level and finally a transformation (see Chapter 1).

| **Function call** | **Inputs** | **Outputs** |
|---|---|---|
| upper level | | |
| `m=<problem>.getNrUppLevVars` | - | number of upper level variables $\mathbf{x}$ $(1 \times 1)$ |
| `x_0=<problem>.getInitValsUppLevVars` | - | possible initial values of the upper level variables $\mathbf{x}$ $(1 \times m)$ |
| `<problem>.setInitValsUppLevVars(x_0)` | sets the initial values of the upper level variables $\mathbf{x}$ $(1 \times m)$ | - |
| `f=<problem>.evalObjFun(x)` | upper level variables $(1 \times m)$ | objective function value $(1 \times 1)$ |
| `grad_f=<problem>.gradObjFun(x)` | upper level variables $(1 \times m)$ | gradient of the objective function$(m \times 1)$ |
| `n=<problem>.getNrSemiInfConstrs` | - | number of semi-infinite constraints $\lvert I^{SI} \rvert$ $(1 \times 1)$ |
| `g=<problem>.evalSemiInfConstrs(i,x,y)` | index of semi-infinite constraint $(1 \times 1)$, upper level variables $(1 \times m)$, $s$ points of index set corresponding to constraint $(s \times n_{\Phi(i)})$ | value of i-th semi-infinite constraint $g_i^{SI}(\mathbf{x}, \mathbf{y})$ for a variable number of points in the lower level. Every row in the output corresponds to one point $(s \times 1)$ |
| `n=<problem>.getNrUppLevNonlinIneqs` | - | number of non linear and non semi-infinte inequality constraints in the upper level $\lvert I^{NL} \rvert$ $(1 \times 1)$ |
| `g=<problem>.evalUppLevNonlinIneqConstrs(x)` | upper level variables $(1 \times m)$ | value of non linear inequality constraints on upper level variables$(\lvert I^{NL} \rvert \times 1)$ |
| `n=<problem>.getNrUppLevNonlinEqs` | - | number of non linear and non semi-infinite inequality constraints in the upper level $\lvert J^{NL} \rvert$ $(1 \times 1)$ |
| `h=<problem>.evalUppLevNonlinEqConstrs(x)` | upper level variables $(1 \times m)$ | value of nonlinear equality constraints on upper level variables $h^{NL}(\mathbf{x})(\lvert J^{NL} \rvert \times 1)$ |
| `[A,b]=<problem>.getLinIneqsOnUppLevVars` | - | linear inequalities on upper level variables $\mathtt{A}\mathbf{x} \leq \mathtt{b}$ (size($\mathbf{A}^{UL}$), size($\mathbf{b}^{UL}$) |

| **Function call** | **Inputs** | **Outputs** |
|---|---|---|
| `[C,d]=<problem>.`<br>`getLinEqsOnUppLevVars` | - | linear equalities on upper level variables `Cx` $=$ `d` (size($\mathbf{C}^{UL}$), size($\mathbf{d}^{UL}$)) |
| `[lb,ub]=<problem>.`<br>`getBoundsOnUppLevVars` | - | lower bounds on upper level variables ($1 \times m$), upper bounds ($1 \times m$). If there is no bound on a upper level variable the value `-inf` or `inf` is returned |
| infinite index sets | | |
| `n=<problem>.getNrIndexSets` | - | number of index sets ($1 \times 1$) |
| `k=<problem>.`<br>`getIndexSetOfSemiInfConstr(i)` | index of semi-infinite constraint ($1 \times 1$) | index of infinite index set($1 \times 1$) |
| `n=<problem>.getNrLowLevVars(k)` | index of index set($1 \times 1$) | number of lower level variables $n_k$ ($1 \times 1$) |
| `y=<problem>.`<br>`getInitValsLowLevVars(k,x)` | index of index set ($1 \times 1$), upper level variables ($1 \times m$) | Initial value for the optimization of the lower level problem ($1 \times n_k$) |
| `n=<problem>`<br>`.getNrNonlinLowLevConstrs(k)` | index of infinite index set ($1 \times 1$) | number of nonlinear inequality constraints on lower level variables $\|J_k\|$ ($1 \times 1$) |
| `v=<problem>.`<br>`evalLowLevNonlinConstrs(k,x,y)` | index of infinite index set ($1 \times 1$), upper level variables ($1 \times m$), $s$ points of index set ($s \times n_k$) | value of the nonlinear describing function of $k$-th index set for $s$ points ($s \times \|J_k\|$) |
| `[A,b]=<problem>.`<br>`getLinIneqsOnLowLevVars(k,x)` | index of infinite index set ($1 \times 1$), upper level variables ($1 \times m$) | linear inequalities for k-th index set `Ay` $\leq$ `b` (size(${\mathbf{A_k}}^{LL}(\mathbf{x})$), size(${\mathbf{b_k}}^{LL}(\mathbf{x})$)) |
| `[lb,ub]=<problem>.`<br>`getBoundsOnLowLevVars(k,x)` | index of infinite index set ($1 \times 1$), upper level variables ($1 \times m$) | lower bounds ($1 \times n_k$ and upper bounds ($1 \times m$) on the lower level variables |

| **Function call** | **Inputs** | **Outputs** |
|---|---|---|
| `n=getNrAllLowLevConstrs(k)` | index of infinite index set $(1 \times 1)$ | number of all constraints for $k$-th index set $(1 \times 1)$.The bounds $\pm\inf$ are dropped. |
| `v=<problem>.`<br>`evalAllLowLevConstrs(k,x,y)` | index of infinite index set $(1 \times 1)$, upper level variables $(1 \times m)$, $s$ points of index set $(s \times n_k)$ | evaluates all describing functions of the $k$-th index set (number constraints$\times 1$). The bounds $\pm\inf$ are dropped. |
| transformation | | |
| `y=<problem>.evalTrafoFun(k,x,z)` | index of infinite index set $(1 \times 1)$, upper level variables $(1 \times m)$, $s$ points of fixed index set $(s \times n_k)$ | transforms $s$ points of the fixed index set to the original infinite index set. |
| `z=<problem>.`<br>`evalInvTrafoFun(k,x,y)` | index of infinite index set $(1 \times 1)$, upper level variables $(1 \times m)$, lower level variables $(s \times \widetilde{n}_k)$ | returns an inverse image of the $k-$th transformation $(s \times \widetilde{n}_k)$ |

Table B.2.: List of the methods for class GSIPProblem

## B.2. Design

| name of property | description | size |
|---|---|---|
| `dim` | The dimension of the design $n$ | $1 \times 1$ |
| `nrOfDesignConstrs` | The number of the function that describe the design $\|J\|$ (only the non-linear ones) | $1 \times 1$ |
| `nrOfParams` | The number of parameters that describe the design $m$ | $1 \times 1$ |
| `initValsParams` | Initial values for the parameters which may used for an optimization algorithm | $1 \times 1$ |
| `boundsOnParams` | Matrix where the first row are the lower bounds $l^{UL}$ and the second row are the upper bounds $u^{UL}$ of the parameters (also `inf`,`-inf` possible) | $2 \times m$ |
| `linIneqParamConstrs` | `[A,b]` where `Ax` $\leq$ `b` corresponds to $\mathbf{A}^{UL}\mathbf{x} \leq \mathbf{b}^{UL}$. If no such constraint is needed you can just give an empty matrix `[]` | `size(`$\mathbf{A}^{UL}$`,1)` $\times (m+1)$ |
| `linEqParamConstrs` | `[C,d]` where `Cx=d` corresponds to $\mathbf{C}^{UL}\mathbf{x} = \mathbf{d}^{UL}$. If no such constraint is needed you can just give an empty matrix `[]` | `size(`$\mathbf{C}^{UL}$`,1)` $\times (m+1)$ |
| `nrOfNonlinParamIneqConstrs` | Contains the number of non-linear and non semi-infinite inequality constraints $\|I^{UL}\|$ that ar given in the function `evalNonLinIneqConstrsOnParams` | $1 \times 1$ |
| `nrOfNonlinParamEqConstrs` | Contains the number of non-linear and non semi-infinite equality constraints $\|J^{UL}\|$ that ar given in the function `evalNonLinEqConstrsOnParams` | $1 \times 1$ |

Table B.3.: Properties of a design

We start with methods which are connected to the parameters. Then we come to the description of the design and finally we give a transformation.

| **Function call** | **Inputs** | **Outputs** |
|---|---|---|
| design parameters | | |
| `n=<design>.getNrParams` | - | number of of parameters $(1 \times 1)$ |
| `x0=<design>.getInitValsParams` | - | initial values of parameters $(1 \times m)$ |
| `<design>.setInitValsParams(x)` | initial values for parameters $(1 \times m)$ | sets the initial values of the parameters to `x` |
| `n=<design>.getNrNonlinIneqConstrsOnParams` | - | number of non linear on parameters $\|J\|$ $(1 \times 1)$ |
| `g=<design>.evalNonlinIneqConstrsOnParams(x)` | parameters$(1 \times m)$ | value of nonlinear constraints $(\|I\| \times 1)$ |
| `n=<design>.getNrNonlinEqConstrsOnParams` | - | number of non linear equality constraints $\|J\|$ on parameters $(1 \times 1)$ |
| `h=<design>.evalNonlinEqConstrsOnParams(x)` | parameters $(1 \times m)$ | value of non linear inequality constraints on parameter $(\|J\| \times 1)$ |
| `[A,b]=<design>.getLinIneqsOnParams` | - | linear inequalities$\mathtt{A}\mathbf{x} \leq \mathtt{b}$ on parameters (size($\mathbf{A}^{UL}$), size($\mathtt{b}^{UL}$)) |
| `[C,d]=<design>.getLinEqsOnParams` | - | linear equalities $\mathbf{C}^{UL}\mathbf{x} = \mathtt{d}$ (size($\mathbf{C}^{UL}$), size($\mathbf{d}^{UL}$)) |
| `[lb,ub]=<design>.getBoundsOnParams` | - | lower bounds $(1 \times m)$ and upper bounds $(1 \times m)$ on parameters. If there is no bound on a variable `-inf` or `inf` is returned |

| **Function call** | **Inputs** | **Outputs** |
|---|---|---|
| Design describing functions: | | |
| `v=<design>.vol(x)` | Parameters $(1 \times m)$ | volume of design $(1 \times 1)$ |
| `grad_v=<design>.gradVol(x)` | Parameters( $1 \times m)$ | gradient of volume function $(m \times 1)$ |
| `n=<design>.getDim` | - | dimension of design $(1 \times 1)$ |
| `y=<design>.getInteriorPoint(x)` | Parameters $(1 \times m)$ | point in the interior of the design$(1 \times n)$ |
| `n=<design>.getNrNonLinDescrFuns` | - | number of non linear describing functions $\lvert J^{LL}\rvert$ $(1 \times 1)$ |
| `u=<design>.evalNonlinDescrFuns(x,y)` | Parameters $(1 \times m)$, point$(1 \times n)$ | value of nonlinear describing functions $(1 \times \lvert J\rvert)$ |
| `[A,b]=<design>.getLinDescrFuns(x)` | Parameters $(1 \times m)$ | linear inequalities on the design $\mathtt{Ay} \leq \mathtt{b}$<br>(size($\mathbf{A}^{LL}(\mathbf{x})$),size($\mathbf{b}^{LL}(\mathbf{x})$)) |
| `[lb,ub]=<design>.getBoundingBox(x)` | Parameters $(1 \times m)$ | lower and upper bounds on the design $(1 \times n, 1 \times n)$ |
| transformation | | |
| `y=<design>.evalTrafoFun(x,z)` | Parameters $(1 \times m)$, $s$ points of fixed design $(s \times \widetilde{n})$ | transforms the fixed points into points of the design$(s \times n)$ |
| `z=<design>.evalInvTrafoFun(x,y)` | Parameters $(1 \times m)$, $s$ points of the design $(s \times n)$ | one inverse image of the transformation $(s \times \widetilde{n})$ |

Table B.4.: List of the methods of a design

## B.3. Container

There are not very many properties and methods for the class container. This is due to the fact that the container of a design-centering problem is not variable. Moreover we do not distinguish between different type of constraints because they all become semi-infinite constraints.

| name of property | description | size |
|---|---|---|
| `dim` | The dimension of the container $n$ | $1 \times 1$ |
| `nrOfConstFuns` | The number of the functions that describe the container $I^{SI}$ (yielding semi-infinite constraints) | |
| `BoundingBox;` | A Box in which the container is in. A Matrix where the first row give the lower bounds and the seccond row gives the upper bounds of the container. | $2 \times n$ |

Table B.5.: Properties of a container

| **Function call** | **Inputs** | **Outputs** |
|---|---|---|
| `n=<container>. getDim` | - | dimension of container $(1 \times 1)$ |
| `n=<container>. getNrDescrFuns` | - | number of describing functions $(1 \times 1)$ |
| `g=<container>. evalDescrFuns(i,y)` | index of describing function $(1 \times 1)$, $s$ points $(s \times n)$ | value of $i$-th describing function for $s$ points $(s \times 1)$ |
| `[lb,ub]=<container>. getBoundingBox()` | - | bounding box $B_C$ given by $[lb, ub]$, in which the container is contained $(1 \times n$,$1 \times n)$ |

Table B.6.: List of methods of container

# Bibliography

[1] M. Diehl, B. Houska, O. Stein, and P. Steuermann. A lifting method for generalized semi-infinite programs based on lower level wolfe duality. *Comput. Optim. Appl.*, 54(1):189–210, January 2013.

[2] F. Guerra Vázquez, J.-J. Rückmann, O. Stein, and G. Still. Gerenalized semi-infinite programming: A tutorial. *Journal of Computational and Applied Mathematics*, 217:394–419, 2008.

[3] Rainer Hettich and Kenneth O Kortanek. Semi-infinite programming: theory, methods, and applications. *SIAM review*, 35(3):380–429, 1993.

[4] H. T. Jongen, J.-J. Rückmann, and O. Stein. Generalized semi-infinite optimization: A first order optimality condition and examples. *Mathematical Programming*, 83(1):145–158, 1998.

[5] P. Lemonidis. *Global optimization algorithms for semi-infinite and generalized semi-infinite programs.* PhD thesis, Massachusetts Institute of Technology, 2008.

[6] S.J. Li. *Semi-infinite programming and semi-definite optimization problems.* PhD thesis, Hong Kong Polytechnic University, 2003. `http://repository.lib.polyu.edu.hk/jspui/handle/10397/1033`.

[7] J.-J. Rückmann and J. A. Gómez. On generalized semi-infinite programming. *Sociedad de Estadística e Investigación Operativa*, 14(1):1–59, 2006.

[8] J.-J. Rückmann and A. Shapiro. First-order optimality conditions in generalized semi-infinite optimization. *Journal of Optimization Theory and Applications*, 101(3):677–691, 1999.

[9] J.-J. Rückmann and A. Shapiro. Second-order optimality conditions in generalized semi-infinite optimization. *Annals of Operations Research*, 9:169–186, 2001.

[10] J.-J. Rückmann and O. Stein. On linear an linearized generalized semi-infinite optimization problems. *Annals of Operations Research*, 101:191–208, 2001.

[11] R. Reemtsen. Discretization methods for the solution of semi-infinite programming problems. *Journal of Optimization Theory and Applications*, 71(1):85–103, 1991.

[12] R. Reemtsen and J.-J. Rückmann. *Semi-infinite Programming.* Kluwer Academic Publishers, Boston/Dordrecht/London, 1998.

[13] J.O. Royset, E. Polak, and A. Kiureghian. Adaptive approximations and exact penalization for dhe solution of generalized semi-infinite min-max problems. *Siam J Optim.*, 14(1):1–34, 2003.

[14] J. Schwientek. *Modellierung und Lösung parametrischer Packungsprobleme mittels semi-infiniter Optimierung - Angewandt auf die Verwertung von Edelsteinen.* PhD thesis, TU Kaiserslautern, 2013. erschienen im Fraunhofer-Verlag, Stuttgart, ISBN 978-3-8396-0566-0.

[15] J. Schwientek, T. Seidel, and Küfer; K.-H. A transformation-based discretization method for solving general semi-infinite optimization problems. Working paper.

[16] A. G. W. Selassie. *A coarse solution of generalized semi-infinite optimization problems via robust analysis of marginal functions and global optimization.* PhD thesis, Technischen Universität Ilmenau, 2004. zu finden unter: http://www.db-thueringen.de/servlets/DocumentServlet?id=2883.

[17] O. Stein. *Bi-level Strategies in Semi-infinite Programming.* Kluwer Academic Publishers, Boston/Dordrecht/London, 2003.

[18] O. Stein. How to solve a semi-infinite optimization problem. *European Journal of Operational Research*, 223(2):312–320, 2012.

[19] O. Stein and G. Still. Solving semi-infinite optimization problems with interior point techniques. *SIAM Journal of Contrl and Optimization*, 42(3):769–788, 2003.

[20] O. Stein and A. Tezel. The semismooth approach for semi-infinite programming under the reduction ansatz. *Journal of Global Optimization*, 41(2):245–266, 2008.

[21] O. Stein and A. Tezel. The semismooth approach for semi-infinite programming without strict complementarity. *SIAM Journal on Optimization*, 20(2):1052–1072, 2009.

[22] O. Stein and A. Winterfeld. Feasible method for generalized semi-infinite programming. *Journal of Optimization Theory and Applications*, 146(2):419–443, 2010.

[23] G. Still. Generalized semi-infinite programming: Theory and methods. *European Journal of Operational Reseachrch*, 119:301–313, 1999.

[24] G. Still. Discretization in semi-infinite programming: the rate of convergence. *Mathematical Programming*, 91:53–69, 2001.

[25] G. Still. Generalized semi-infinite programming: Numerical aspects. *Optimization*, 49:223–242, 2001.

[26] G. Still. Generalized semi-infinite programming: Numerical aspects. *Optimization*, 49(3):223–242, 2001.

[27] Natick Massachusetts United States The MathWorks, Inc. *MATLAB R2016b*. 2016.

[28] F. G. Vazquez and J.-J. Rückmann. Extensions of the kuhn-tucker constraint qualification to generalized semi-infinite programming. *SIAM Journal of Optimization*, 15(3):926–937, 2005.

[29] G.-W. Weber. Generalized semi-infinite optimization: On some foundations. *Journal of Computer Science and Technology*, 4:41–61, 1999.

[30] G.-W. Weber. *Generalized Semi-Infinite Optimization and Related Topics*, volume 29 of *Research and Exposition in Mathematics*. Heldermann-Verlag, Lemgo, 2003.

[31] A. Winterfeld. *Large-scale semi-infinite optimization applied to lapidary cutting*. PhD thesis, TU Kaiserslautern, 2007.

[32] A. Winterfeld. Application of general semi-infinite programming to lapidary cutting problems. *European Journal of Operational Research*, 191:838–854, 2008.